A Predictive Maintenance Framework for Data Analysis and Resource Optimization in Industrial IoT

Ong Shen Hoong Kevin

School of Computer Science and Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

2022

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

13/07/2022

.....

Date

.....

Ong Shen Hoong Kevin

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

13/07/2022

.....

Date

.....

Prof. Dusit Niyato

Authorship Attribution Statement

This thesis contains materials from: (1) *two* papers published in peer-reviewed journals (2) *three* papers accepted at conferences and (3) *one* technical report, which I am named as the main author.

Chapter 4 is published as **KSH. Ong**, W. Wang, D. Niyato and T. Friedrichs, "Deep Reinforcement Learning Based Predictive Maintenance Model for Effective Resource Management in Industrial IoT", IEEE Internet of Things Journal. 2021 Sep 3. DOI: 10.1109/JIOT.2021.3109955. Part of the work in Chapter 4 is published as **KSH. Ong**, W. Wang, T. Friedrichs and D. Niyato, "Augmented Human Intelligence for Decision Making in Maintenance Risk Taking Tasks using Reinforcement Learning", in 2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC) 2021 Oct 17. DOI: 10.1109/SMC52423.2021.9658936., and **KSH. Ong**, D. Niyato, C. Yuen, "Predictive Maintenance for Edge-Based Sensor Networks: A Deep Reinforcement Learning Approach", in 2020 IEEE 6th World Forum on Internet of Things (WF-IoT) 2020 Jun 2. DOI: 10.1109/WF-IoT48130.2020.9221098.

The contributions of the co-authors are as follows:

- Prof. Dusit Niyato provided directions for designing the human risk system model, assisted in the review of the manuscript, and is the Principal Investigator for the NTU Institutional Review Board (IRB) experiment.
- I initiated the research direction, formulated the problem statement, prepared the manuscripts, performed all simulation experiments, designed and conducted all NTU-IRB experiments.[1] The manuscript was revised together with Dr. Wenbo Wang.
- Dr. Wenbo Wang provided research mentorship on deep reinforcement learning and critical comments to improve the manuscript quality.
- Dr. Thomas Friedrichs provided support and contacts for the factory technicians in the BOSCH China and Singapore offices.
- Prof. Chau Yuen provided critical comments to improve the manuscript's quality.

Chapter 5 is published as **KSH. Ong**, W. Wang, D. Niyato, NQ Hieu. and T. Friedrichs, "Predictive Maintenance Model for IIoT-based Manufacturing: A Transferable Deep Reinforcement Learning Approach", IEEE Internet of Things Journal. 2022. DOI: 10.1109/JIOT.2022.3151862.

The contributions of the co-authors are as follows:

[1]The experiments are conducted in accordance to IRB protocol with human participants from both Singapore and China. IRB reference number IRB-2021-019.

- Prof. Dusit Niyato proposed the initial paper idea, reviewed the manuscript, and provided guidance to improve upon the reviewer response letter.
- I conceptualized the problem statement, designed the system model, prepared the manuscript, and performed all the experiments.
- Dr. Wenbo Wang provided research mentorship, critical manuscript comments, and assisted in the manuscript revision.
- Nguyen Quang Hieu provided general guidance and experience for combining transfer learning with deep reinforcement learning.
- Dr. Thomas Friedrichs offered evaluative feedback on the manuscript and its applicability to BOSCH.

Chapter 6 details the work performed during the six months industrial research attachment at Robert Bosch (South-East Asia). Part of the work in Chapter 6 is *published* as **KSH. Ong**, D. Niyato, and T. Friedrichs, "Drift-Aware Edge Intelligence for Remaining Useful Life Prediction in IIoT", in 37th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC), May 2022.

The contributions of the co-authors are as follows:

- Dr. Thomas Friedrichs proposed relevant real-world BOSCH use-cases for consideration and provided guidance throughout the attachment period at BOSCH CI/SIX-AP department.
- Prof. Dusit Niyato proposed the initial idea of transfer learning, and comments to improve the manuscript's quality.
- I identified and formulated the real-world use-case as a multivariate regression problem. Performed exploratory data analysis and data pre-processing on the real-world dataset, and contributed fresh data insights. Additionally, I proposed a novel deep-learning model and applied it to the NASA and real-world industrial washing machine datasets. Lastly, I counter-proposed the idea of drift-aware edge models via incremental learning, performed extensive experiments, and prepared the manuscript for publication.

13/07/2022

Date

Ong Shen Hoong Kevin

Acknowledgements

Some breathtaking paths are only discoverable by becoming lost, and the last four years have been eventful. In retrospect, I am pleased to have tried and completed such a challenging intellectual milestone. Consequently, I can better comprehend and tackle challenging real-world problems independently. Clearly, this adventure would not have been possible without the following groups of individuals.

First, my sincere gratitude towards my supervisor, Prof. Dusit Niyato, for his excellent guidance and continuous support during my Ph.D. studies at Nanyang Technological University (NTU). His patience, passion for research, vast experience, and rigorous working style has positively influenced me.

Next, I would like to thank my co-supervisor, Dr. Thomas Friedrichs, for his continuous support and guidance throughout my Ph.D. studies at NTU. Besides my PhD supervisors, I am also deeply appreciative of Dr. Wenbo Wang from Bar Ilan University in Ramat Gan, Israel, for his mentorship, as well as the insightful discussions and suggestions that helped me acquire pertinent knowledge. Not forgetting Mr Toh and CNCL team member's support during my PhD studies at NTU.

On a related note, I thank the Singapore Economic Development Board (EDB) for the generous financial support of my work under the EDB Industrial Postgraduate Programme (IPP) with Robert Bosch (South-East Asia) Pte Ltd. Likewise, I thank my BOSCH colleagues' guidance and support during my industrial research attachment.

From the bottom of my heart, I am deeply grateful to my wife Gao Fangfang, my son Lemuel Ong, and my parents. Their unwavering love, support and patience constantly encourage me to confront the rigors and the seasonal emotional roller-coaster of the Ph.D. journey. This dissertation is thus dedicated to my ardent supporters.

Kevin Ong

Contents

List of Figures

List of Tables

Symbols and Acronyms

Acronyms

AI	Artificial Intelligence
C-MAPSS	Commercial Modular Aero-Propulsion System Simulation
IoT	Internet-of-Things
IIoT	Industrial Internet-of-Things
IRB	Institutional Review Board
IWM	Industrial Washing Machine
JPdMRM	Joint PdM and Resource Management
MDP	Markov Decision process
MG	Machine Group
MPV	Maintenance Priority Value
MRS	Maintenance Repair Simulator
MTTR	Mean-Time-To-Repair
PdM	Predictive Maintenance
PM	Parts Manufactured
PrM	Preventive Maintenance
RM	Reactive Maintenance
RUL	Remaining Useful Life
RxM	Prescriptive Maintenance
DAA	Data Aggregation and Analysis
EC	Edge Cloud
EG	Edge Gateway
ES	Edge Server / Edge Sensor
ESD	Edge Sensor Device

ESN	Edge Sensor Network
A2C	Advantage Actor Critic
AE	AutoEncoder
AMBFNet	Attentive Multi-branch Feature Network
ANN	Artificial Neural Network
BPTT	Backpropagation Through Time
CNN	Convolutional Neural Network
DAE	Deep Auto Encoder
DBN	Deep Belief Network
DDQN	Double Deep Q-Network
DNN	Deep Neural Networks
DQN	Deep Q-Network
DRL	Deep Reinforcement Learning
GA	Genetic Algorithm
GAE	Generalized Advantage Estimator
GAN	Generative Adversarial Network
GRU	Gated Recurrent Unit
KL	Kullback-Liebler
KD	Knowledge Distillation
LSTM	Long Short Term Memory
MLP	Multi Layer Perceptron
NN	Neural Network
NLP	Natural Language Processing
PER	Prioritized Experience Replay
PN	Parameter Noise
PPO	Proximal Policy optimization
QL	Q-Learning
RL	Reinforcement Learning
RBM	Restricted Boltzmann Machine
RNN	Recurrent Neural Network
SAE	Sparse Autoencoder
TD	Temporal Difference
TL	Transfer Learning
TLD	Transfer Learning from Demonstrations

TRPO Trusted Region Policy Optimization

Abstract

Ubiquitous sensors and networks are critical elements that seamlessly integrate into our daily lives and enable diverse commercial and engineering applications, including the Internet-of-Things (IoT). By extension, the Industrial IoT (IIoT) entails continuously monitoring revenue-generating assets, such as wafer dicing, robotic pick and place, and industrial washing machines, for anomalous patterns and minimizing unplanned breakdowns of critical machines via predictive maintenance (PdM). However, PdM must grow beyond equipment maintenance to achieve prescriptive maintenance, and we identified several research gaps, which we highlight and address in the following paragraphs.

In the first study, we consider that unplanned breakdown of critical equipment interrupts production throughput in IIoT, and data-driven PdM becomes increasingly important for companies seeking a competitive business advantage. Manufacturers must manually allocate competent manpower resources in the event of machine failure. Furthermore, human errors have a negative rippling impact on both overall equipment downtime and production schedules. To address these issues, we formulate the complex resource management of humans and machines as a resource optimization problem. We developed a maintenance repair simulator (MRS) game and conducted real-world experiments to support our findings. These results are contrasted against our proposed deep reinforcement learning (DRL) method to evaluate the efficacy of AI-based complex resource management for PdM applications. In addition, DRL improves its accuracy with more training data, self-learns an optimal maintenance strategy, and incorporates human feedback as part of its learning strategy. Notably, this study analyzes one machine for maintenance to limit the research scope of and validate our hypothesis.

In the second study, preliminary examination of existing literature reveals that existing IIoT-based PdM frameworks do not consider complex real-time production states, machine health, and maintenance manpower resources. For this reason, we propose a generic PdM optimization framework to help maintenance teams prioritize and resolve maintenance task conflicts. Specifically, the PdM framework

jointly optimizes edge-based machine network uptime and manpower allocation in a stochastic IIoT-enabled manufacturing environment utilizing model-free Deep Reinforcement Learning (DRL) methods. Since DRL requires a significant amount of training data, we propose and demonstrate the use of Transfer Learning (TL) method to help DRL learn more efficiently by incorporating expert demonstrations, termed TL with demonstrations (TLD). TLD reduces training wall-time by 58% compared to baseline methods, and we undertake numerous experiments to illustrate the performance, robustness, and scalability of TLD.

PdM research focuses on improving RUL prediction accuracy via end-to-end models (i.e., raw sensor to RUL prediction) to increase factory productivity. Since machine remaining useful life (RUL)(i.e., ground truth information) is often unavailable, an RUL prediction model's success is not readily transferable to similar applications. Besides, DL-based RUL models require considerable training data, including rare failure data, and critical monitoring of model performance drift is not actively researched. To address these concerns, we first present an attentive multi-branch feature network (AMBFNet) model to improve RUL prediction performance and benchmark AMBFNet against comparable work on real-world datasets. Secondly, we present an incremental learning-based approach for monitoring field-deployed edge-based RUL model performance drift. Notably, the AMBFNet model surpasses state-of-the-art RUL prediction models in terms of prediction error, and we are the first to report incremental learning results for a popular PdM dataset.

In summary, this thesis encloses several critical contributions to the corpus of knowledge in the PdM research topic. Specifically, we propose transforming PdM into a holistic maintenance strategy via AI-based methods, such as DRL. In this regard, DRL is used to learn data-driven decision-making strategies and provide actionable recommendations to augment human decision-makers, thereby termed "augmented intelligence". Furthermore, we consider hybrid deep learning methods to improve RUL prediction performance and batch-based incremental learning to mitigate model drift. Finally, unifying the proposed contributions creates a generic PdM framework for data analysis in IIoT to jointly manage resources (i.e., humans and machines) and achieve good RUL prediction performance via AI-based methods.

Chapter 1

Introduction

This chapter discusses the significance and motives behind the Predictive Maintenance (PdM) research presented in this thesis. Specifically, we discuss the importance of maintenance, the research problems, and challenges before delving into the specific research focus with corresponding research motivations. Finally, we summarize the corresponding research contributions in this thesis.

1.1 Background of Predictive Maintenance

1.1.1 Evolution of Maintenance

Manufacturing is a highly competitive industry, and businesses worldwide are embarking on a digitalisation journey to obtain a competitive edge and increase revenues. Maintenance remains an essential activity in the manufacturing sector that impacts a business's ability to compete on price, quality, and performance. In particular, unplanned downtime[1] of revenue-generating machines can cripple a business's operations, cause irreversible brand damage, and incur financial losses. For example, [7] reports that annual losses amounting to US$60 billion are due to improper machine maintenance, otherwise preventable. Likewise, recent industry research estimates that the worldwide predictive maintenance market will be worth US$21 billion by 2027, growing at a 25% compound annual growth rate (CAGR) [8]. Moreover, the ongoing COVID-19 pandemic has inadvertently accelerated

[1] In the manufacturing context, downtime denotes the duration for which a machine is not producing anything.

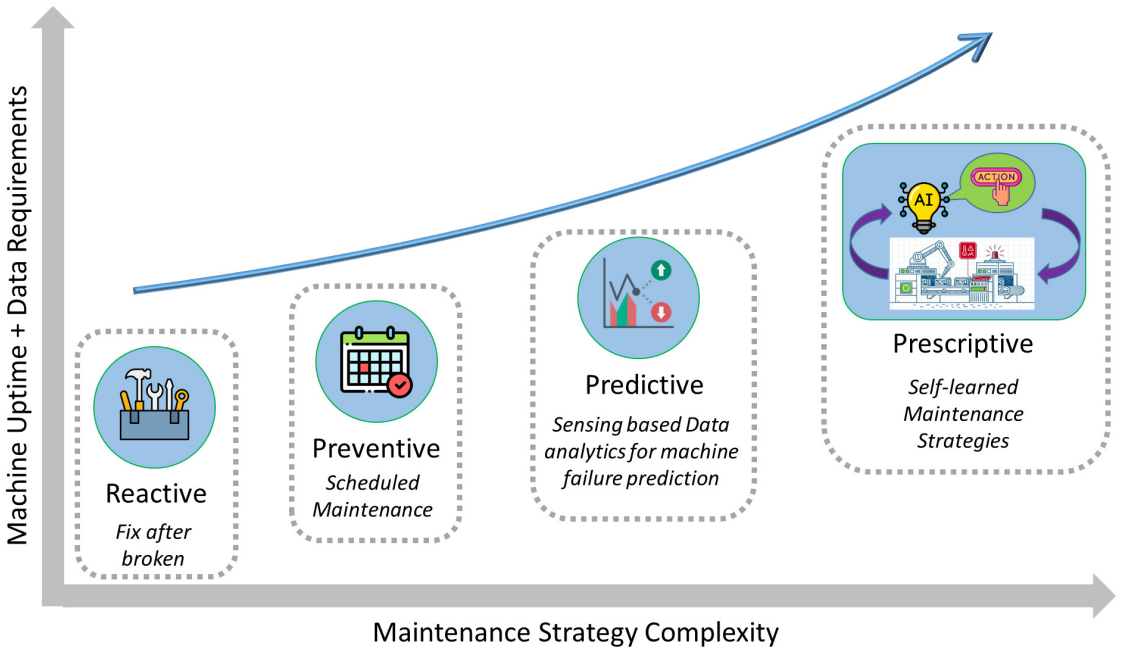


FIGURE 1.1: Evolution of maintenance strategy.

the adoption rate of digitalization with modern technologies such as the Industrial Internet-of-Things (IIoT), automation, and artificial intelligence (AI). Thus, leveraging these technological innovations to increase revenue-generating machines' uptime (i.e., assets) needs effective maintenance strategies to reduce the probability of unplanned machine failure, streamline operational costs, and improve manufacturing throughput.

In tandem with the fast-paced technological advancement, a similar shift in maintenance strategies is constantly taking place. Figure 1.1 depicts the evolution of maintenance strategies from *reactive maintenance* (RM) through *preventive maintenance* (PrM) and *predictive maintenance* (PdM), with *prescriptive maintenance* (RxM) as the most advanced. To elaborate, RM is a naive run-to-failure strategy to recover machines only after a malfunction occurs. Adopting such an approach can severely burden businesses with expensive repair bills and exasperated maintenance teams. On the other hand, PrM is the cheapest proactive maintenance strategy and performs maintenance either according to an original equipment manufacturer's (OEM) pre-determined schedule or usage iterations to manage unplanned machine breakdowns. However, PrM requires high inventory stock quantities and can result in excessive maintenance in some cases. Consider the PrM example of vehicle maintenance, in which manufacturers suggest changing the fluid oil every

Maintenance Strategy	**Reactive**	**Preventive**	**Predictive**	**Prescriptive**
Maintenance Focus	Component	Component, Operation	Component, Operation, System	Component, Operation, System
Historical Data Availability	None	Limited	Medium to High	High
Real-time Data Availability	None	None	Low to Medium	High
Maintenance Automation Level	None	None	Low to Medium	High
Machine Health Condition	None	Low to Medium	Low to Medium	High
Human Effort (Decision-making & Inspection)	High	Medium	Low to Medium	Low (ad-hoc)

TABLE 1.1: Comparison of maintenance strategies. [5]

six months or 10,000 kilometers. Conversely, end-users consider a fluid oil change unnecessary or excessive as their vehicle has been parked in a garage for eight months owing to bad weather. Table 1.1 depicts the evolution of maintenance and outlines the major differences between multiple maintenance strategies. The differences between PdM and RxM are given in the following paragraphs.

Predictive maintenance (PdM)'s goal is to minimize unplanned machine downtime via a data-driven approach [9], with the prediction model's accuracy requiring frequent updating and is highly reliant on both the quantity and quality of data available [10]. In particular, PdM attempts to optimize the trade-off between RM and PrM via continuous monitoring of the machine's real-time condition through sensors and with the help of advanced failure prediction models. Ultrasonic, temperature, and accelerometers are some examples of sensors.

Prescriptive maintenance (RxM) is the most advanced maintenance strategy and is not limited to predicting failures [11]. RxM recommends preventing or delaying potential machine breakdowns using AI-based prescriptive algorithms. RxM, in particular, needs a cognitive system capable of learning, adapting, and analyzing unseen data patterns. This solution combines AI, machine learning (ML), and PdM

data analytics with big data. For instance, thyssenkrup's predictive algorithm claims to be capable of forecasting an elevator's shutdown five days in advance owing to a door fault [12]. Being data-driven enables thyssenkrup's predictive technology to identify the four most probable sources of an issue. As a result, elevator technicians can resolve door faults on the first visit more than 90% of the time, which increases customer satisfaction. It is worth mentioning that additional data sources, such as quality control and engineering data, can further improve RxM's recommendation accuracy. Finally, RxM technology is relatively immature and requires high-speed IT infrastructure to securely and promptly deliver vast amounts of critical information [13].

1.1.2 Industrial Internet-of-Things (IIoT)

Modern sensors are the cornerstone of PdM. By acquiring vital machine data from sensors such as vibration, pressure, and temperature, manufacturers can make informed maintenance decisions through condition monitoring of machines, thereby mitigating the risk of disruption to production plans. Furthermore, with modern industrial equipment generating hundreds of gigabytes to terabytes of data daily, the IIoT enables a factory's network to be securely and reliably managed via cloud-based and edge-based solutions. When properly managed, manufacturers can have real-time visibility into individual machine health, obtain fresh insights into machine performance and utilization, and respond swiftly to avert potentially disruptive failures. However, AI-based PdM systems require vast amounts of high-quality data to extract patterns from complex data.

Nevertheless, IIoT-enabled PdM aims to reduce unscheduled machine downtime by providing timely and accurate maintenance insights. While accurate prediction of machine failure is necessary, PdM is much more than hardware and software [14]. For instance, PdM aggregates and analyzes machine sensor data in the background over the IIoT network and notifies the maintenance team of potential machine failures. However, arbitrarily halting the production machines impairs the execution and production delivery schedule [15], resulting in order-delivery delays and revenue loss. Furthermore, the complexity of decision-making increases greatly with machine heterogeneity, the availability of spare parts, and the dynamic availability of manpower resources and competence. Thus, these intermediary issues require

further research to effectively utilize data-driven AI technology to enhance PdM and overall factory productivity before achieving RxM.

Artificial intelligence (AI) models for PdM are developed and deployed in the cloud, where machine sensor data is stored and analyzed. However, the latency incurred as data traverses various networks can result in sluggish response time from these remote AI models. Edge technology promises to increase the speed and efficiency of predictive analytics by implementing machine learning (ML) models locally, rather than depending on costly yet reliable connection to transport data from the device to the cloud over the internet. Furthermore, edge-powered AI data is locally processed, reduces data storage costs (in the cloud), and provides real-time analytics, which is essential for operating the PdM-based AI model. Finally, edge-powered AI data processing enhances network resilience by preventing data loss during outages. These benefits are significant for manufacturing industry where unintentional delays in response time may result in revenue losses, client confidence, and unplanned machine downtime.

1.2 Research Challenges and Motivations

1.2.1 Challenges

Along with technological breakthroughs, the flexibility and improvements in PdM present new research challenges that require further investigation. These include the following but are not limited to:

- **Aging workforce:** A recent survey in [16] highlights that a wide variety of competent technicians are expected to retire within the next decade, causing a wider gap in labor's skill/knowledge. Notably, the field service industry anticipates a shortage of 2 million US workers [17]. Although technology can potentially close the knowledge-skill gap by digitally capturing domain expert knowledge, integrating expert knowledge to improve modern AI system performance is challenging and receives little attention in the existing literature for PdM applications.

- **Decision-support PdM framework:** A factory production system comprises a combination of human and machine resources that collaboratively transform raw materials into manufactured products (i.e., finished goods).

While increasing the prediction accuracy of machine failure is essential, maintenance decision-making remains a challenging task for human decision-makers. Specifically, the decision-making process is manual, laborious, error-prone, and experience-based, with negative consequences to the production throughput until machine failure rectification. Additionally, continually operating a machine nearing failure can squander valuable resources and disrupt downstream production capacity. Besides, traditional PdM approaches overwhelm decision-makers with threshold-based notifications and descriptive data rather than actionable recommendations (i.e., decision-support) to augment human intelligence.

- **AI-based maintenance scheduling automation:** Anomalies are swiftly identified and prevented by AI's capacity to process vast data, including audio and video. Compared to human experts, AI can continuously monitor revenue-generating machines with the model's learning capacity scale according to the available data quantity and computing resources. Despite the massive potential of AI-enabled PdM, automated scheduling of maintenance tasks based on numerous cost factors, stochastic resource availability, production deadline, and machine reliability requirements is challenging to optimize jointly with non-AI methods [18]. Recently, [10, 14] highlights the apparent scarcity of research into the convergence of AI-based fault detection and prediction algorithms with maintenance scheduling algorithms, which is vital to realizing an intelligent and autonomous PdM system.

- **Data and class imbalance issue:** Failure events become increasingly rare, with each generation of production machines claiming increased reliability. In addition, any attempt to continuously operate machines to failure is costly, and time-consuming within a production environment with potentially catastrophic consequences. Despite recent attempts to use transfer learning (TL)[19] and generative adversarial network (GAN)[20] to address data imbalance [21], achieving satisfactory results for diverse PdM applications and scenario remains an open research question [10].

1.2.2 Motivations

In this thesis, we only address the first three challenges and propose to exploit the differences between PdM and RxM as the overarching research motivation,

which we briefly mention below. Research challenge #4 was identified towards the end of my industrial research attachment at Bosch (see Chapter 6) and concluded in the last semester of my PhD candidacy. Owing to the PhD candidature time constraints, only the first three challenges are addressed.

PdM is a maintenance strategy that uses data analytic and condition monitoring methods to detect abnormalities in a machine's daily operations and minimize catastrophic machine failure. For instance, we can use vibration analysis to uncover structural health issues and highlight weaknesses in machines that can cause significant damage due to inaction [22, 23]. However, the existing PdM approach requires manual human intervention to analyze data, make a decision, and generate corresponding maintenance work orders. Besides, existing PdM research is myopic, from a production line perspective, with a laser focus on revenue-generating machines as the only resource type. Finally, the impending retirement of experienced technicians creates a vast experience gap, making the already challenging maintenance decision-making tasks more susceptible to human errors owing to inexperience.

In contrast, RxM continuously analyzes the machine data, automatically generates a work order, and transmits the relevant information to the maintenance team. Specifically, the RxM-based algorithm independently learns an appropriate reliability plan based on historical data (e.g., sensor, machine, and human actions taken), and recommends appropriate follow-up action. Consider the example where RxM successfully identifies the faulty machine component, but the part component is currently unavailable in the factory's inventory. The RxM algorithm identifies the component delivery lead time and recommends reducing the machine production throughput by 40% to stretch the residual time to machine failure. RxM may also seek consent from the human decision-maker to procure additional component quantities based on existing production plans as RxM detects comparable failure rates for the remaining machines in the factory. Consequently, factory productivity increases as maintenance teams can better focus on maintaining machines efficiently and spend less time troubleshooting the root causes of intermittent machine interruptions to the production line. Unlike the PdM research landscape, RxM is relatively new, with just 30 publications[2] in the last decade.

[2]Results are based on keyword search of "prescriptive maintenance" on the web of science platform.

Apart from the PdM and RxM differences, traditional optimization methods may be inadequate for solving complex, stochastic, and noisy real-world problems. Considering that traditional optimization methods perform well for one type of problem, but the final solution depends on the initially random solution chosen. In addition, the obtained solution is not guaranteed to be global optimum, and changes in problem complexity and dataset or data distribution may prevent traditional optimization methods from converging quickly to a solution. Conversely, machine learning (ML) is inspired by human learning behavior, in which we provide examples and let ML algorithms discover the solution. Similarly, the ML model can iteratively learn from various data types while solving the same optimization problem.

1.3 Research Scope

With reference to the aforementioned PdM challenges, the following sub-sections outline the research issues with proposed solutions in this thesis. Finally, we present the overarching theme that motivates this thesis's research topic.

1.3.1 Deep Reinforcement Learning Based Predictive Maintenance Model for Effective Resource Management in Industrial IoT[3]

According to [14], the majority of existing works use DL techniques in equipment failure prognostics. For example, DL-based solutions are used to improve an equipment's remaining useful life (RUL)[4] estimation and anomaly detection [27–30]. Some researchers have recently examined the applicability of DRL to PdM, with promising findings. To diagnose and classify faults in time-series-based equipment health sensor data, the authors [31] and [32] propose using deep Q network (DQN). Similarly, [33] investigates the use of temporal difference (TD) Learning for RUL forecasting. Finally, [26] proposes using Double DQN (DDQN) to learn the optimal replacement point for equipment based on its health index value, with good generalization performance across similar equipment. However, PdM encompasses more than equipment maintenance [10]. Instead, considerable emphasis should also

[3]Comprises of published works in [24], [25], and [26].

[4]The RUL metric is a second order derivative that provides failure prediction estimates to facilitate proactive maintenance scheduling.

be placed on enabling maintenance automation to optimize maintenance strategy, particularly manpower resource management.

Motivated by this research gap, the improved PdM framework must also consider manpower resource management, ensuring that maintenance tasks are allocated to the most competent technician. Although [25, 34, 35] make specific references to predictive maintenance and manpower resource management, their assumptions are mainly limited to more straightforward and ideal maintenance scenarios. With the improved PdM framework considerations, both businesses and decision-makers will significantly benefit from the data-driven maintenance recommendations since they will be able to take the best possible action for any physical equipment.

To address the stochastic resource optimization problem, we propose a DRL-based model framework that leverages the proximal policy optimization (PPO) long short term memory (LSTM) (i.e., PPO-LSTM) model:

- We introduce the edge-powered PdM framework architecture, enabling PdM for a network of equipment within a generic production facility. Secondly, we formulate pertinent PdM and resource management elements into a Markov Decision Process framework to discover the optimal decision policy.
- We coin the term "equipment severity rating," which quantifies equipment failure probability in relation to the widely used equipment health indicator value in PdM literature.
- We present a model-free PPO-LSTM model to address the reward sparsity issue in stochastic settings and discover the optimal decision policy given the stochastic resource optimization problem. Specifically, the LSTM module is used to capture pertinent spatial-temporal information before further processing by the PPO model.
- We conduct extensive simulation experiments using a maintenance repair simulator (MRS) to train the PPO-LSTM agent on the relationship between the equipment severity rating, the maintenance cost model, and the technician competence level. Additionally, we undertake IRB-approved real-world experiments with two groups of working professionals to provide a human-level baseline for comparison purposes. Consequently, we demonstrate the efficacy of our proposed approach as a decision-support tool.

- Our DRL approach is also extended to the NASA C-MAPSS dataset, a well-known time-series PdM dataset, and observed positive results.

1.3.2 Predictive Maintenance Model for IIoT-based Manufacturing: A Transferable Deep Reinforcement Learning Approach[5]

Given the abundance of sensor data, the majority of the existing literature is devoted to improving RUL prediction accuracy in PdM applications [14, 36]. However, focusing solely on machine maintenance with traditional PdM-based frameworks is a myopic resource management approach compared to the overall factory resource management strategy [10]. Recently, [24, 35, 37] attempted to address the optimization of manufacturing systems for energy consumption, manufacturing throughput, predictive maintenance, and manpower resource management. For instance, [37] proposes improving the production throughput by optimizing the preventive maintenance and energy scheduling processes while putting aside both PdM and manpower resource management. On the contrary, [35] and [24] exclude energy consumption and relevant production factors that can influence the maintenance priority of machines, such as residual days to delivery deadline and parts revenue. Thus, we are interested in exploring the unrealized potential of the simultaneous optimization of prioritization-based PdM and human resource management, which are frequently addressed independently.

In practice, multiple machines may produce near identical maintenance priority levels, and the time-varying manpower availability situation complicates an otherwise straightforward process of maintenance prioritization. In addition, the resource management problem in a non-stationary environment becomes intractable using traditional one-shot allocation methods such as stable-matching [38]. The rationale is that, since apriori knowledge about the dynamics of manpower resource availability cannot be known for a stochastic manufacturing setting, it is myopic to determine resources (i.e., human technicians) to allocate solely based on the needs of the current maintenance task. Hence, we extend from our recent work in [24] to include production-related data for analysis, automate maintenance task prioritization using model-free DRL methods, and dynamically manage manpower resources according to the runtime task state information.

[5]Work is published in [18].

To further accelerate learning with DRL, we introduce the concept of machine groups and propose a model-free DRL-based method for knowledge transfer from expert DRL models to another machine group, namely, TL from demonstrations (TLD). Considering that the maintenance schedule and manpower resource management are ad-hoc and highly subjective, the underlying decision-making process becomes challenging to model, and there lacks a ready-to-use mechanism for knowledge transfer to DRL models. Besides, investigations of TL in DRL-based PdM applications is also relatively under-explored [39].

1.3.3 Remaining Useful Life Prediction of IIoT-enabled Equipment using Attentive Multi-Branch Feature Network[6]

Model-based and data-driven are the main approaches for RUL prediction. The model-based approach relies on physics-based modeling to characterize the degradation process of a machine or component prior to estimating RUL [41, 42]. Nevertheless, developing such models often demands precise machine component modeling, complex operational condition information (i.e., internal or external), and highly specialized expert knowledge across heterogeneous machines. Besides, the model transferability between heterogeneous machines is severely restricted. The data-driven approach links RUL and the target machine features without requiring prior knowledge and extensive degradation modeling. With increasing machine sensorization, data availability facilitates the easier adoption of the data-driven approach for RUL prediction across various complex machines. Broadly, RUL prediction is solvable using standard ML methods such as support vector machines (SVM) [43, 44], random forest (RF) [45], and hilbert-huang transform (HHT) [45, 46]. However, beforehand, the feature selection step is human-intensive, and noisy raw sensor data affects RUL prediction accuracy [47]. Therefore, most researchers shift their focus towards deep-learning-based approaches instead [48].

The ability to automatically learn feature representations from raw sensor data drives the adoption of DL methods for RUL prediction. Due to the highly correlated spatial-temporal sensor data features, DL methods such as convolutional neural network (CNN) [49, 50], recurrent neural network (RNN) based autoencoder (AE) [51], and long short-term memory (LSTM) [52] architectures [53–56]

[6]This work is performed during a six-month research attachment at Robert Bosch (SEA) Pte Ltd. Part of this work is presented in [40].

have been proposed. For example, [50] models sensor data as an image and applies a one-dimensional deep CNN to filter out relevant sensor features. However, the convolution kernel size requires continuous increases to process long data sequences. Likewise, [51]'s RNN-based AE requires separate training of the AE and an ensemble of RNNs to achieve good RUL prediction results. Unfortunately, RNNs fail at learning long-term sequence modeling due to the vanishing gradient problem. To address this problem, [53, 54] adopts the LSTM approach to achieve excellent RUL prediction performance, easily outperforming CNN approaches.

A key component of effective resource management is accurate RUL prediction of time-series data [18]. In the context of time-series sequence modeling of multivariate data, the importance of individual sensors (i.e., features) can vary as degradation information becomes apparent. However, vanilla LSTM networks are prone to over-fitting and suffer from the vanishing gradient issue when modeling extended data sequences. Moreover, due to the sequential processing of historical sensor data, the training duration for LSTMs and RNNs is typically time-consuming and expensive [57]. Besides, even similar machines will exhibit different degradation features. Hence, automatic feature importance weighting within an end-to-end neural network model, from raw sensor data to prediction, becomes even more challenging without human intervention.

We introduce an attention-based deep learning model to overcome these constraints and automatically identify relevant features. Inspired by [58], we propose to study the effects of enhancing vanilla LSTM networks using the attention mechanism for RUL prediction. Specifically, we adopt a strategy similar to [55] in which the attention network extracts temporally correlated features from the data output of an existing LSTM network. However, unlike the global attention method suggested in [55], this work augments the LSTM network with the self-attention method prior to encapsulation in a module block dubbed the branch layer. Each branch is responsible for determining the feature importance and correlation between time-steps. Following that, the learned branch feature information is pooled for RUL prediction, resulting in the proposed model termed attentive multi-branch feature network (AMBFNet). We adopt two real-world datasets to evaluate the AMBFNet's efficacy and generalization in tackling similar RUL prediction problems. For comparison purposes, we benchmark AMBFNet against various existing

and state-of-the-art methods. Lastly, the AMBFNet model is compatible with our previously proposed improved PdM framework, as shown in Figure 1.2.

1.3.4 Connections among Research Issues

The underlying theme of these research issues is the need for PdM-based maintenance decision-support to holistically increase production throughput and productivity with the least amount of unexpected machine downtime possible. Notably, unplanned machine downtime can intermittently halt a machine when the anticipation is for the machine to continue manufacturing goods. Therefore, this thesis aims to accelerate the transition from PdM to RxM and propose a mindset shift for tackling maintenance problems from a component level to a more holistic systems perspective. A case in point, surveys [10, 14, 36] unanimously agree that the majority of existing literature is devoted to improving RUL prediction accuracy in PdM applications. Similarly, concentrating solely on machine maintenance using traditional PdM-based frameworks is a myopic resource management approach when contrasted to the overall factory resource management strategy [10]. Besides, a factory production line encompasses much more than product manufacturing. Thus, this thesis considers a generic PdM model to optimize factory resource management (i.e., manpower and machine).

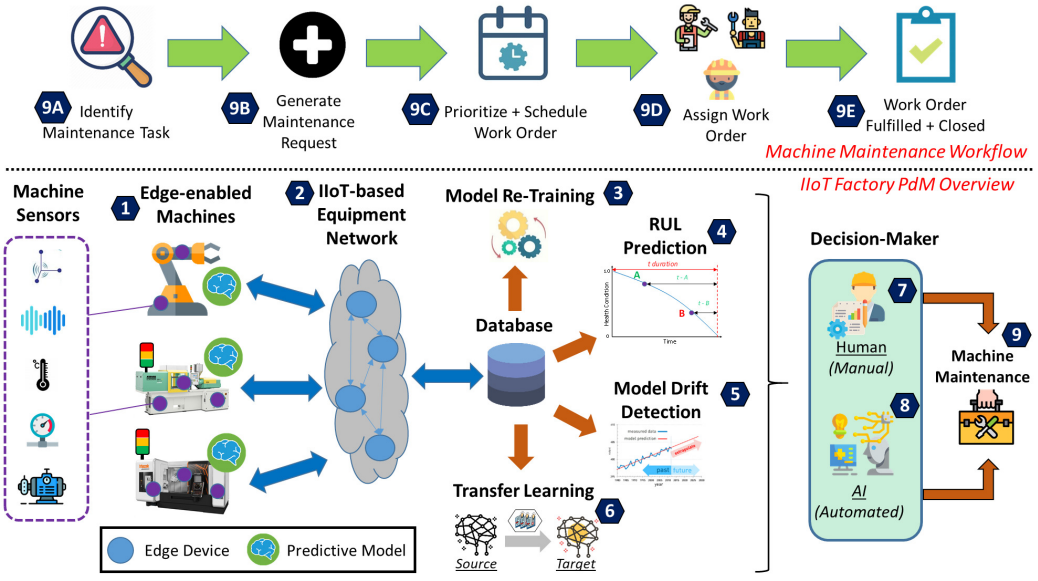


FIGURE 1.2: Schematic overview of the proposed IIoT-based PdM framework and machine maintenance workflow.

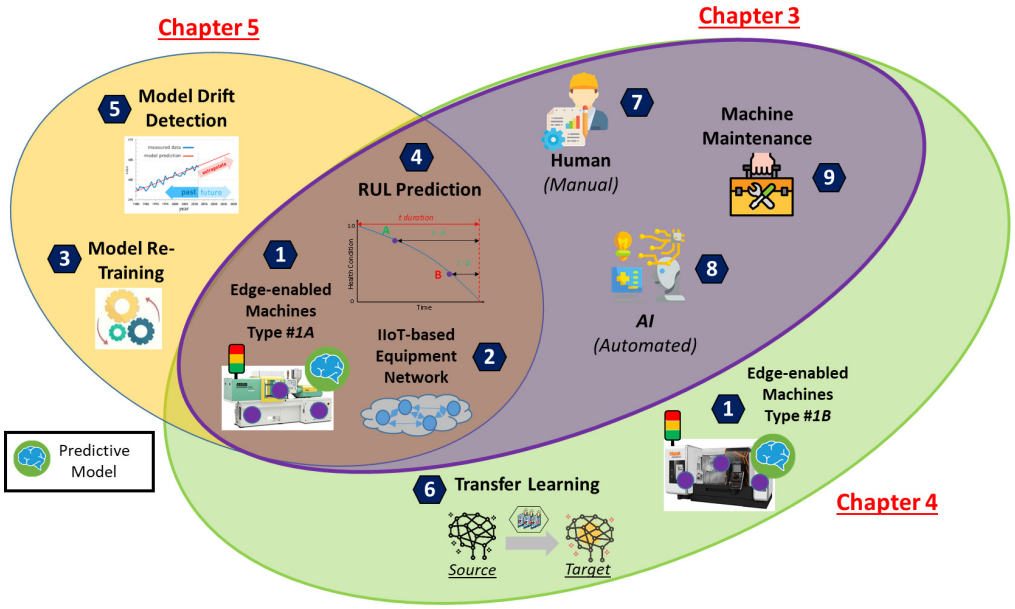


FIGURE 1.3: Overview of thesis contributions w.r.t. the proposed PdM framework in Figure 1.2.

Continuous data transmission of raw machine data over an IIoT factory network consumes precious communication bandwidth and induces additional network latency for time-critical factory operations. Cloud computing technologies, while extremely powerful, are not designed to process data in real-time. In reality, transmitting data to the cloud also results in noticeably slower transfer times, thus requiring expensive bandwidth communication to be transmitted and processed, where the internet link's quality-of-service is not always guaranteed [59]. On the other hand, data at the edge may pose security concerns, mainly when handled by disparate devices that are not as safe as centralized systems. Thus, this thesis considers an edge-based IIoT network with edge-enabled machines that operate a field-deployed PdM model in-situ.

Research Issues	Thesis-related Publications	Components	Resources
1	[24–26]	1, 2, 4, 7, 8, 9	Machine. Human
2	[18]	1, 2, 4, 6, 7, 8, 9	Machine, Human
3	[40]	1, 2, 3, 4, 5	Machine

TABLE 1.2: Overview of research contributions w.r.t. individual components in Figure 1.2.

For completeness, the respective research issues and proposed solutions are integrated into a PdM framework, see Figure 1.3. Detailed information is reported in Chapters 4, 5, and 6, with corresponding annotations in Table 1.2 for the reader's convenience. Lastly, equipment is synonymous with a machine unless otherwise stated.

1.4 Major Contributions

The main contributions of this thesis is summarized as follows:

- *Chapter 4*:
  - We formulate the PdM manpower allocation into a resource optimization problem and present a DRL-based PdM Model framework. The overall optimization objective is to increase production revenues while maximizing the cumulative equipment uptime in an IIoT network powered by edge computing. With the proposed data-driven framework in place, the routine but challenging task of manpower resource allocation is now automated, resulting in increased productivity for both production and maintenance teams.
  - We propose the PPO-LSTM model for automating the decision-making process associated with PdM-based resource management. LSTM is used to improve the performance of PPO in stochastic settings, while PPO is responsible for selecting the best state-action to perform.
  - We perform extensive simulation experiments using a Maintenance Repair Simulator (MRS) to evaluate the performance of PPO-LSTM, and we undertake IRB-approved real-world experiments, comprised of working professionals, to provide a human-level baseline for performance comparison. Empirically, the proposed PPO-LSTM outperforms both comparable DRL methods and human participants by 53% and 65%, respectively.
  - We also expand the PPO-LSTM model to the NASA C-MAPSS dataset, a well-known time-series PdM dataset. Interestingly, PPO alone reports a 73% improvement in learning efficiency relative to prior work. Overall, these empirical results demonstrate the effectiveness of our proposed

DRL-based PdM Maintenance Model framework as a decision-support tool.

- *Chapter 5*:
  - We propose the Maintenance Priority Value (MPV) metric for prioritizing machine maintenance. By including pertinent production parameters[7] within the MPV metric, prioritizing machines for maintenance repair becomes simpler and more compelling. Additionally, the MPV metric enables the distinction between the prediction-based and the actual machine stoppages.
  - We define the PdM resource management problem as a joint task scheduling and resource optimization problem, and present a two-stage DRL-based Joint Predictive Maintenance and Resource Management (JPdMRM) framework. In the first stage, a task scheduling problem is formulated and the randomly generated MPV-based maintenance tasks are sorted on the basis of priority in a time-slotted manner. In the second stage, the DRL-based TLD algorithm is then introduced to learn the optimal strategies to automate the PdM-based resource management process according to the task-priority information. The two-stage framework adopts a modular design that allows macro model fine-tuning with limited impact on the overall scheduling system and promotes model re-use.
  - We propose the TLD algorithm to accelerate model learning and extend the scalability and robustness of the JPdMRM framework to similar machines. We introduce the concept of machine groups (MGs) and production parts to facilitate knowledge transfer. Furthermore, by reusing small amounts of previously learned expert-demonstration data, which is stored in the prioritized experience replay buffer, the stored demonstrations can be utilized to improve the learning efficiency of TLD in a new environment (e.g., a new production line). Notably, with TLD, our study is among the first on transfer learning for PdM-based resource management in IIoT-enabled manufacturing. With numerical simulations, we empirically show the impact on performance of demonstration data size, which is also under-explored in the existing literature.

[7]Examples of production parameters include residual days to delivery deadline, potential revenue from manufactured parts and machine operating state.

- *Chapter 6*:
  - We propose an attention-based deep learning approach for RUL prediction, named attentive multi-branch feature network (AMBFNet). Time-sensitive multi-variate sensor information is first acquired for sequence modelling using the long-short term memory network (LSTM). Then, the attention network ingests the generated sequences and automatically learns critically relevant sensor features before predicting the RUL value.
  - We propose a multi-branch feature aggregation approach, defined as multi-branch pooling, to increase the RUL prediction performance. Specifically, our approach enables each branch to autonomously focus on locally relevant degradation features before aggregating all the data for regression modeling. Consequently, the regression model learns from higher-quality data, which improves the model's overall prediction performance.
  - We incorporate knowledge-based handcrafted features in both the training and testing datasets to improve the RUL prediction performance.
  - We evaluate the proposed AMBFNet model's RUL prediction performance using real-world dataset. Empirical results appear very promising, with AMBFNet surpassing recent attention-based works by up to 70%, even on the challenging FD004 scenario in the NASA C-MAPSS dataset. Similarly, we extend AMBFNet to a small real-world industrial washing machine (IWM) dataset with encouraging results.
  - We propose an incremental learning approach for mitigating model overfitting and drift detection. In this regard, we empirically determined the appropriate size of batch-sized data that produces the lowest error deviation compared to tabula rasa model training. Consequently, continuous learning on resource-constraint edge computing devices becomes achievable without constantly overloading the edge computing server with model training requests.

1.5 Organization of the Thesis

The remaining chapters of this thesis are organized as follows:

- *Chapter 3* reviews existing work related to PdM frameworks and presents a taxonomy of data analysis approaches. Accompanying critical analysis is also presented to highlight the relevant approach limitations. Specifically, the following PdM-related categories are investigated: (i) PdM applications and methods, (ii) PdM-based resource management frameworks, and (iii) RUL prediction.

- *Chapter 4* formulates the complex resource management problem as a resource optimization problem to determine if a model-free Deep DRL-based PdM framework can be used to learn an optimal decision policy from a stochastic environment automatically. Unlike the existing PdM frameworks, our approach considers PdM sensor information and the physical equipment and human resources as part of the optimization problem. The proposed DRL-based framework and PPO-LSTM model are evaluated alongside baseline results from human participants using a maintenance repair simulator. Finally, we compare the experimental results to demonstrate the efficacy of the PPO-LSTM model for complex maintenance decision-making under uncertainty.

- *Chapter 5* builds atop the work in Chapter 4, and considers the joint management of multiple machines and manpower resources to assist maintenance teams in prioritizing and resolving maintenance task conflicts under real-world manufacturing conditions. Specifically, the PdM framework aims to jointly optimize edge-based machine network uptime and allocate manpower resources in a stochastic IIoT-enabled manufacturing environment via model-free DRL methods. Since DRL requires a significant amount of training data, we propose and demonstrate using TL method to assist DRL in learning more efficiently by incorporating expert demonstrations, termed TL with demonstrations (TLD). TLD reduces training wall-time by 58% compared to baseline methods, and we conduct numerous experiments to illustrate the performance, robustness, and scalability of TLD. Finally, we discuss the general benefits and limitations of the proposed TL method, which are not well

addressed in the existing literature but could benefit to both researchers and industry practitioners.

- *Chapter 6* acknowledges the correlation between the accurate prediction of a machine's RUL information and the severity rating proposed in Chapter 5. Nonetheless, RUL prediction remains a challenging task and requires large amounts of multi-variate time-series data to achieve satisfactory performance. However, the vanilla LSTM mechanism alone is inadequate for processing long sequences of multi-variate sensor data and extracting spatial-temporal features. For this reason, different DL networks are suitable for different PdM applications, with hybrid approaches outperforming standalone DL network models. Thus, our work proposes an attention-based approach to augment existing LSTM models. The attention mechanism, in particular, automatically learns salient data features and time steps, while the LSTM network focuses on sequential time-series features. Based on empirical findings on real-world datasets, our proposed model outperforms baseline methods by a minimum margin of 24% and existing state-of-the-art approaches by up to 70% on challenging open-sourced real-world datasets. On the other hand, model drift in prediction performance necessitates continuous monitoring before choosing whether to execute incremental learning or request an updated model from the cloud for resource-constrained edge computing devices. Consequently, we perform extensive experiments to determine appropriate batch data size to minimize model over-fitting while maintaining low prediction errors, compared to tabular rasa model training, for deployment on edge computing devices.

- *Chapter 7* concludes this thesis and outlines future research directions.

Chapter 2

Preliminary

This chapter outlines the preparatory information necessary for the discussion of the literature review, with the focus towards our approaches in this thesis.

2.1 Deep Learning

2.1.1 Convolutional Neural Network

The convolutional neural network (CNN)[60] is a multi-layer feed-forward (FF) artificial neural network (ANN) which processes two-dimensional or three-dimensional inputs as images and learns abstract features by combining several deep neural networks (DNN) layer types. Layer examples of CNN architecture include *input*, *convolution*, *pooling*, and *fully-connected*. The *input* layer typically accepts two-dimensional data such as image, time-series, and time-frequency spectrum. Formally, we can represent the *input* data $x \in \mathbb{R}^{I \times J}$, where I and J denote the input data dimensions. As the name suggests, the *convolutional* layer performs the convolution operation (i.e., kernel filters) using a set of weights to denote the weighted average of data sampled and create a feature map that summarizes the presence of detected features in the *input*. The *convolutional* layer output expression is :

$$y_n = f(x * \boldsymbol{W}_n + b_n), \tag{2.1}$$

where $*$ denotes the convolution operator; n denotes the convolution filter layer index; $\boldsymbol{W}_n$ denotes the weight matrix of the n^{th} filter kernel; b_n denotes the filter's kernel bias and the activation function as f. The *pooling* layer is responsible

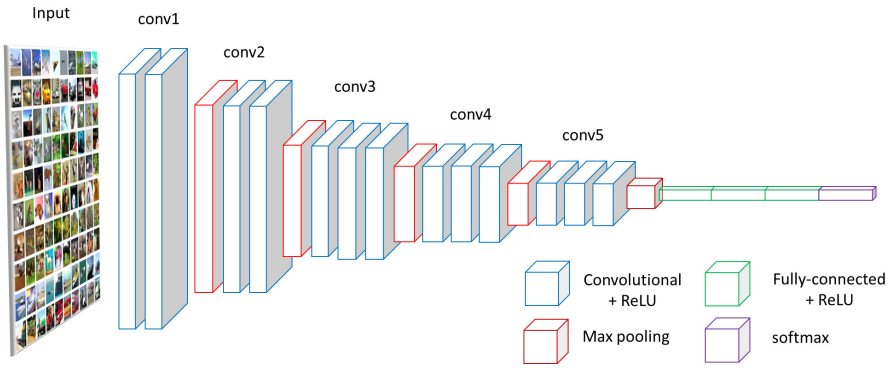


FIGURE 2.1: VGG16 - CNN-based architecture example.

for extracting key features from each *convolution* layer. Hence, extracting relevant features reduces the model training parameter, over-fitting, and training time. Following multiple combinatorial layers of *convolution* and *pooling* layers, a *fully-connected* layer performs classification or regression layer tasks. Specifically, the *fully-connected* layer is responsible for learning the non-linear combinations of the high-level features from the CNN and often requires several *fully-connected* layers to achieve the desired output for PdM application. For single- or multi-class image classification-based PdM applications, the last layer of the CNN model requires an activation function (e.g., softmax or sigmoid). For instance, the VGG16 [61] is a 16-layer neural-network-based model, see Figure 2.1, and requires a softmax activation function for object classification. Similarly, a sigmoid activation function is utilized for defect inspection but not for RUL prediction, a regression problem.

In deep learning (DL), a neural network (NN) without an activation function is considered a linear regression model; activation functions transforms linear input signals and models into non-linear output signals, thereby enabling deep networks to learn high-order polynomials and perform complex tasks. We direct interested readers to [62, 63] for current advancements and thorough explanations on activation functions and their usage.

2.1.2 Autoencoder

Autoencoder (AE) [64] is an unsupervised learning technique for extracting features automatically from the inputs and outputs an estimation of the H_t as in (2.15)[62]. A neural network (NN)-based AE design comprises the *encoder* and *decoder* stages with the following NN layer configuration: input(x), hidden(h), and output($\hat{x}$). An AE's objective is to learn a compressed representation of the inputs (x) via the

reconstruction error minimization between the input and the reconstructed output values. Formally, we let D_i and D_h as denoting i inputs and h hidden units of a NN. Given input $x \in \mathbb{R}^{D_i}$, the AE maps x to a compressed representation $h \in \mathbb{R}^{D_h}$ and performs a non-linear transformation using a non-linear activation function f following:

$$h = f(\boldsymbol{W}^{(i)}x + \boldsymbol{b}^{i}), \tag{2.2}$$

where $\boldsymbol{W}^{(i)} \in \mathbb{R}^{D_i \times D_h}$ denotes an encoding weight matrix and $\boldsymbol{b}^{i} \in \mathbb{R}^{D_h}$ as the vectored bias. The decoder then maps h back to vector $y \in \mathbb{R}^{D_i}$ to obtain a reconstruction approximate of the input vector x in the following:

$$\hat{x} = \boldsymbol{W}^{(o)}a + \boldsymbol{b}^{(o)} \approx x, \tag{2.3}$$

where $\boldsymbol{W}^{(o)} \in \mathbb{R}^{D_i \times D_h}$ denotes the decoding weight matrix and $\boldsymbol{b}^{o} \in \mathbb{R}^{D_i}$ as the vectored bias. The training process aims to minimize reconstruction error loss J_{AE} in (2.4) to improve feature extraction performance using gradient descent optimizers. The extraction of AE features is accomplished by dimensionality reduction (i.e., feature compression) through the setup of $h < i$, which forces the AE model to learn and prioritize information from the input data. Alternatively, a deep AE network comprises two symmetrical deep belief networks: an *encoder* and *decoder* stage. For example, [65] proposes an RUL prediction model based on a deep AE

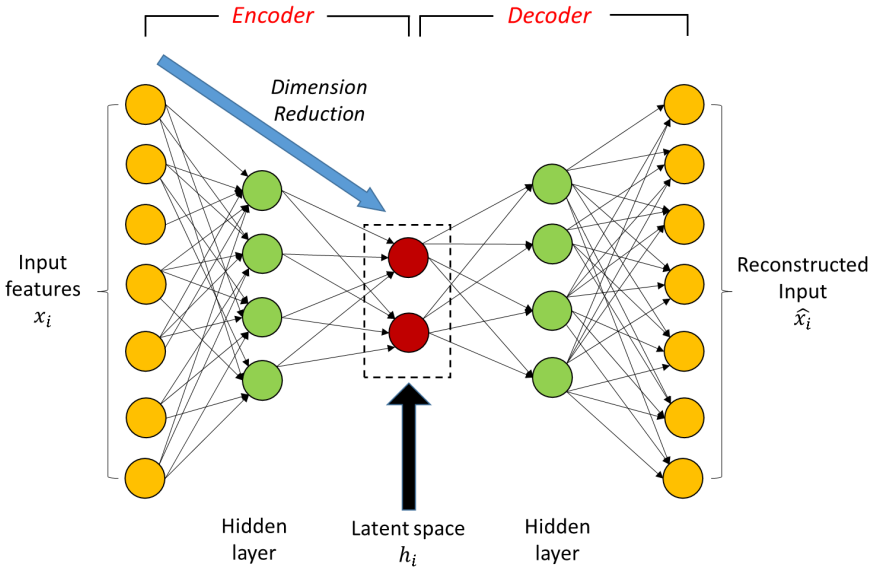


FIGURE 2.2: Autoencoder architecture example.

network for bearing failure. Given the time domain complexity of the feature signals used for bearing RUL prediction, the trained DAE *encoder* is responsible for the dimension reduction of features extracted from raw vibration signals, as shown in Figure 2.2.

On the other hand, a stacked sparse AE (SAE) variant is simply an AE with the addition of a sparsity penalty on the hidden layer, as in (2.5).

$$J_{AE} = \frac{1}{2}\sum_{m=1}^{M}(\hat{x}_m - x_m)^2, \tag{2.4}$$

$$J_{SAE} = J_{AE} + \alpha \sum_{i=1}^{s_l} KL(\rho|\rho_i). \tag{2.5}$$

where α denotes the tuning parameter; s_l the quantity of nodes in the l^{th} hidden layer; KL as the Kullback-Liebler (KL) divergence to nudge the respective node activations to approach the sparse parameter ρ; ρ as the sparse parameter constraint of the hidden layer; and ρ_i as the mean activation of i^{th} node over the dataset in a hidden layer.

2.1.3 Deep Belief Networks

A stacked configuration of multiple restricted boltzmann machines (RBMs) is known as a deep belief network (DBN) [66], where the hidden layer unit's outputs

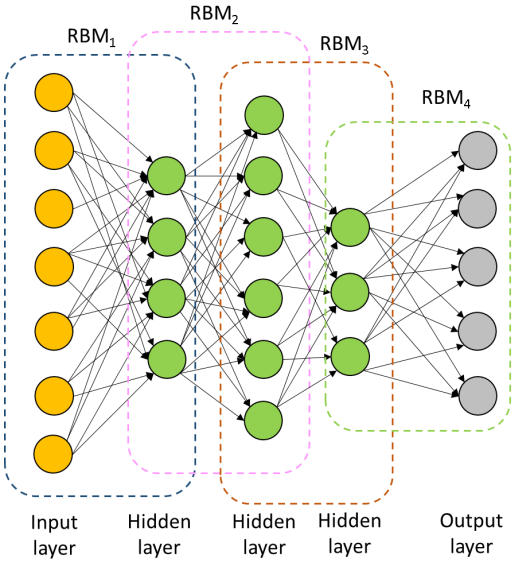


FIGURE 2.3: Deep belief network architecture example.

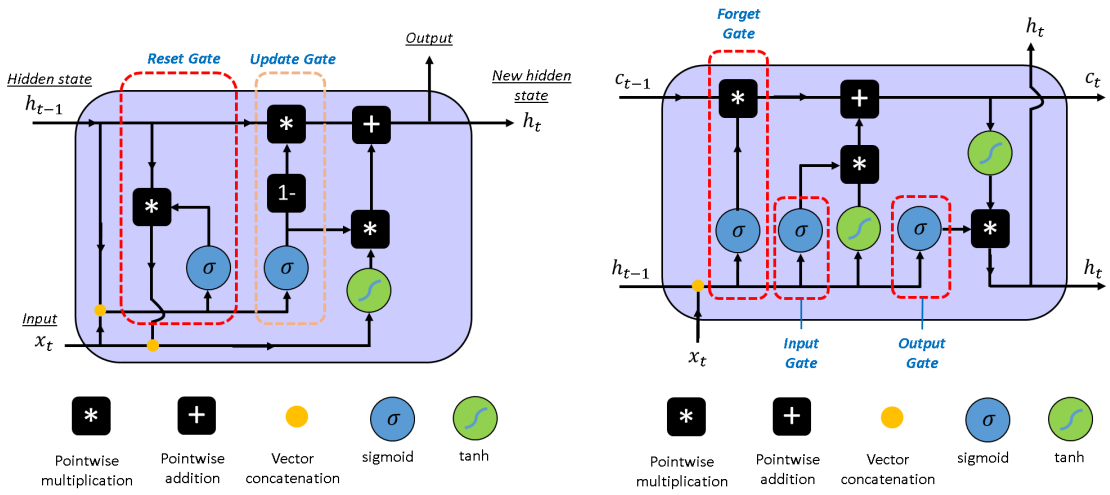
are input to the visible layer units. DBN, like CNN and AE, extracts high-level characteristics from raw sensor data, and DBN's layer-wise greedy learning approach enables the unsupervised model pre-training with flexible requirements on training data quantity [67]. While a DBN is structurally identical to a multilayer perceptron (MLP), their similarities stop there; a DBN employs an entirely different training process that enables it to deal with vanishing gradients [66]. Specifically, the DBN trains two layers simultaneously, with each layer considered as an RBM in Figure 2.3. In addition, the RBM's hidden layer serves as the input for successive layers for all network layers such that after the first RBM is trained, its outputs are used as inputs to the subsequent RBM. The model is trained across all RBMs until it reaches the output layer [68].

2.1.4 Recurrent Neural Networks

Recurrent neural networks (RNN) refers to a class of NNs designed to work with time-series or sequence data. Unlike conventional NNs, RNNs consider the temporal dependencies between the data points and store prior input information within the network's internal state before outputting the next data sequence. Formally, a transition function $\mathbb{H}$ in (2.6) considers the current data x_t, for every t time-step, and the previous hidden output h_{t+1} with current data output y. At t time-step, the current input is a combination of inputs x_t and x_{t-1}, and the output y_t, at any given time, is feedback into the network to improve the prediction accuracy of RNNs using the backpropagation through time (BPTT) algorithm [69].

$$h_t = \mathbb{H}(h_{t-1}, x_t). \tag{2.6}$$

When modeling extended data sequences, RNNs are known to suffer from vanishing gradients. Hence, the gated recurrent unit (GRU) [70] and long short term network (LSTM) [52] models are proposed instead, see Figure 2.4. Furthermore, with the forget gates in Figure 2.4b, LSTM networks replace the older state values with new ones, making LSTMs well-suited for sequence modeling and time-series-based prediction tasks. Thus, the widespread use of LSTMs for time-series data modeling, notably RUL prediction for PdM.



(a) Gated Recurrent Unit.

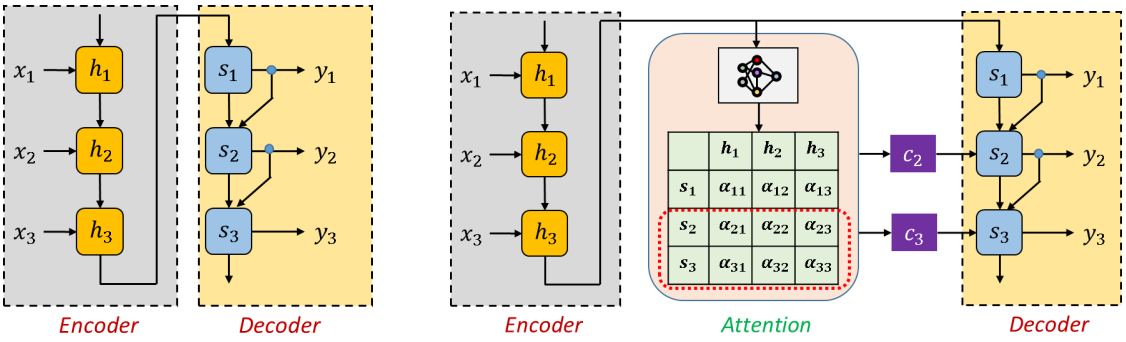
(b) Long Short Term Memory.

FIGURE 2.4: Overview of RNN architectures.

2.1.5 Attention Mechanism

All human perceptual and cognitive functions stem from the attention concept [71]. Given our limited capacity to digest information from disparate sources, the innate attention mechanism processes prioritize, modify, and fixate on the most relevant information. For decades, the human attention mechanism has been researched in philosophy, psychology, neuroscience, and computer science subjects. However, it was only until recently that the attention mechanism was formalized and proven using deep neural networks, heralding a breakthrough in the AI field. Consequently, neural attention models are representative of many state-of-the-art in DL for a variety of application areas, including voice [72], computer vision [73, 74], and natural language processing (NLP) [57, 75].

Consider the example of sequence-to-sequence modeling for a generic translation task. Two recurrent neural networks, named *encoder* and *decoder*, compose the attention network. The encoder of the attention network starts by encoding an entire input sentence $\{x_1, x_2, \ldots, x_T\}$ into a context vector $\{\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_T\}$, where T denotes the *input* sequence length. The attention block automatically learns and assigns relative weights α_{ij} of each input corresponding to the target language translation output. A context vector $\boldsymbol{c}$ is formulated and transmitted to the decoder based on the attention weights. Unlike traditional encoder-decoder, see Figure 2.5a, which considers c as the last hidden state of the encoder (i.e., $\boldsymbol{h}_T$), the attention mechanism's decoded context vector $\boldsymbol{c}_j$ for output y_t is deemed as the weighted



(a) (b)

FIGURE 2.5: Generic encoder-decoder architecture: (a) RNN-based; (b) Attention-based;

sum of the encoder's outputs as in (2.7), see Figure 2.5b. Thanks to the context vector, the decoder can determine the output word sequence $\{y_1, y_2, \ldots, y_{T'}\}$ by combining the context vector with the decoder's hidden state, where T' denotes the *output* sequence length. Thus, the decoder learns to focus on relevant positions of the input sequence. Finally, the sentence is translated to the target language. Note that the weighting strategy used in the attention mechanism is automatically calculated and varies with problem types. In this regard, the attention mechanism will be beneficial in overcoming the inefficiencies of RNN networks by automatically determining the importance of input features when dealing with large sequences of sequential data.

$$\boldsymbol{c}_j = \sum_{i=1}^{T} \alpha_{ij} h_i. \tag{2.7}$$

where $\boldsymbol{c}_j$ denotes the decoding position j, h_i denotes the encoder's hidden state, and the decoder's hidden state as $s_t = f(s_{t-1}, y_{t-1}, \boldsymbol{c}_t)$.

2.2 Deep Reinforcement Learning

Deep reinforcement learning (DRL) combines DL and reinforcement learning (RL) and leverages DNN to learn an optimal decision-making strategy over a diverse range of real-world tasks with vast state-action spaces. In contrast to supervised and unsupervised learning methods, DRL introduces the concept of an agent in an environment to perform the optimal action in the present state in order to maximize long-term rewards. The agent's action and state interaction is modeled using the markov decision process (MDP) mathematical framework [76]. MDP comprises a

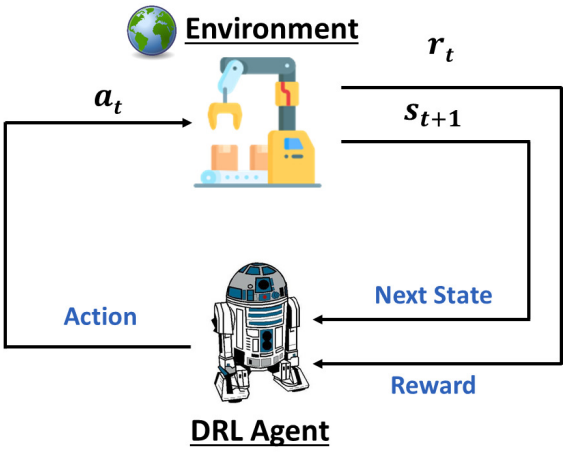


FIGURE 2.6: Deep reinforcement learning agent interaction with environment.

tuple expression of $< \mathcal{S}, \mathcal{A}, T, R >$, where $\mathcal{S}$ denotes the state space; $\mathcal{A}$ denotes the list of actions; T denotes the state transition probability function between any state-action pair combinations; R denotes the immediate reward function. For the reader's understanding, we briefly introduce the DRL fundamentals and classifications in the following paragraphs and refer readers to [76] for details. Subsequently, we highlight the relevant DRL works.

2.2.1 Markov Decision Process (MDP)

The MDP is a mathematical framework to capture the problem space parameters and is used by RL methods to find the optimal decision policy. Specifically, the RL agent interacts with the environment, observes state transitions, receives a reward, and aims to find an optimal action for each state. Therefore, the MDP framework is used to account for changes in environment states that might lead to finding the corresponding optimal action, which is not considered in the k-armed bandit framework. The optimal decision policy involves learning the state-to-action mapping recursively, and Q-learning (QL) can be used to learn the optimal policy by maximizing the action-value function over all Q-value policies. For small and discrete environments, QL maintains a look-up table to store the encoded state and actions of the environment interaction. Considering real-world environments, such as PdM, are continuous and have numerous variables, the Q-table becomes exponentially large and impractical to maintain. Similarly, the time-taken for QL to find the optimal state-action combination becomes ill-suited for real-time PdM-based applications. Therefore, deep neural network (DNN)-based DRL is preferred

to approximate the Q-table and learn the optimal policy. Additional details on the pros and cons of different DRL-based methods are extensively discussed in [77].

Let us consider the agent receives an observation signal of the current environment state s_t, and selects action a_t based on a decision policy. The decision policy is simply a mapping of any state $s \in \mathcal{S}$ to action $a \in \mathcal{A}$. Following the selection of a_t, the agent receives a corresponding reward $r_t = R(s_t, a_t)$ and the next environment state s_{t+1} is presented in Figure 2.6, which follows an unknown environment transition probability $P(s_{t+1}|s_t, a_t)$. To ensure model training convergence stability, the acquired knowledge is stored as $(s_t, a_t, s_{t+1}, a_{t+1})$ within a buffer. Given a trajectory τ of state-action pairs, the agent calculates the long-term cumulative reward as in (2.8) to evaluate the efficacy of the selected markovian policy π.

$$G(\tau) = \sum_{i=0}^{T-t} \gamma^i R(s_{t+i}, a_{t+1}), \tag{2.8}$$

where T denotes the training steps and discounting factor $\gamma \in [0, 1]$ denoting the future reward importance. Note that $T \to \infty$ for continuous MDP. There are two classes of DRL algorithms to learn the optimal policy that maximizes the long-term cumulative reward: value-based DRL and policy-based DRL.

In *value-based* DRL, the DRL algorithm attempts to approximate the value function in (2.9), which is the estimation of the expected long-term reward of a state (or a state-action pair). Accordingly, the value function can be defined as a mapping from each state to its corresponding long-term reward, namely $V_\pi : \mathcal{S} \to R$, where π denotes the learned policy from the value function and state $s_t \in \mathcal{S}$.

$$V_\pi(s_t) = \mathbb{E}_\pi[G(\tau|s_t)] = \sum_\tau P(\tau|\pi, s_t)G(\tau), \tag{2.9}$$

where $V_\pi(s)$ denotes the state-value policy; the cumulative reward $G(\tau)$ in (2.8); and $P(\tau|\pi, s_t)$ denoting the trajectory probability following π policy. For the agent to maximize the long-term cumulative reward, there exists an optimal state-value policy which maximizes the value function, in (2.10), over all existing decision policies.

$$V^*(s_t) = \max_\pi V^\pi(s_t), \quad \forall s_t \in \mathcal{S}. \tag{2.10}$$

The Q-learning (QL) algorithm can learn the optimal policy by maximizing the

action-value function ($Q_\pi(\mathcal{S}, \mathcal{A})$) over all Q-value policies in (2.11). The Q-value is recursively updated using Temporal Difference (TD) Learning [76], and the off-policy transitions (s_t, a_t, r_t, s_{t+1}) are learned.

$$Q_\pi(s, a) = \mathbb{E}_\pi[r_{t+1} + \gamma Q_\pi(s_{t+1}, a_{t+1})|s_t = s, a_t = a], \tag{2.11}$$

$$Q^*(s, a) = (1 - \alpha)Q(s, a) + \alpha Q_{\text{obs}}(s, a), \tag{2.12}$$

The optimal Q-function is obtainable as $Q^*(s, a) = \max_\pi V_\pi(s, a)$. The Q-function is updated following (2.12), where α is the learning rate and $Q_{\text{obs}}(s, a) = r(s, a) + \gamma \max_{a' \in \mathcal{A}} Q'(s', a')$, where $s' = s_{t+1}$ and $a' = a_{t+1}$. With $Q^*(s, a)$, the optimal policy is

$$\pi^*(s) = \arg\max_{a \in \mathcal{A}} Q^*(s, a). \tag{2.13}$$

Take note that this thesis focuses mostly on model-free MDP models within a stochastic environment with finite time horizon. Depending on the problem formulation, we use either value iteration [78] or policy iteration [79] to discover the best decision policy π^*. For environments with a continuous state space, we discretize the state space and use DNNs for value function approximation.

2.3 Transfer Learning

Transfer learning (TL) mimics human learning efficiency by transferring information from the source domain to the target domain to enhance the target domain's learning performance [80]. Let us consider two separate datasets, D_{source} and D_{target}, and learning tasks ψ_{source} and ψ_{target}. Transfer learning aims to transfer the acquired knowledge from D_{source} and ψ_{source} to improve the learning performance of task ψ_{target} via prediction function $f_\psi(\cdot)$. Note that $\psi_{\text{source}} \neq \psi_{\text{target}}$, and the size of D_{source} (i.e., N_{source} is often assumed to be larger than D_{target} (i.e., N_{target}. We can mathematically represent transfer learning as a tuple $< f_\psi(\cdot), D_{\text{source}}, D_{\text{target}}, \psi_{\text{source}}, \psi_{\text{target}} >$, and we illustrate the TL process in Figure 2.7a.

[1]Note that the learning task across two environment contexts must be comparable for transfer learning to be beneficial.

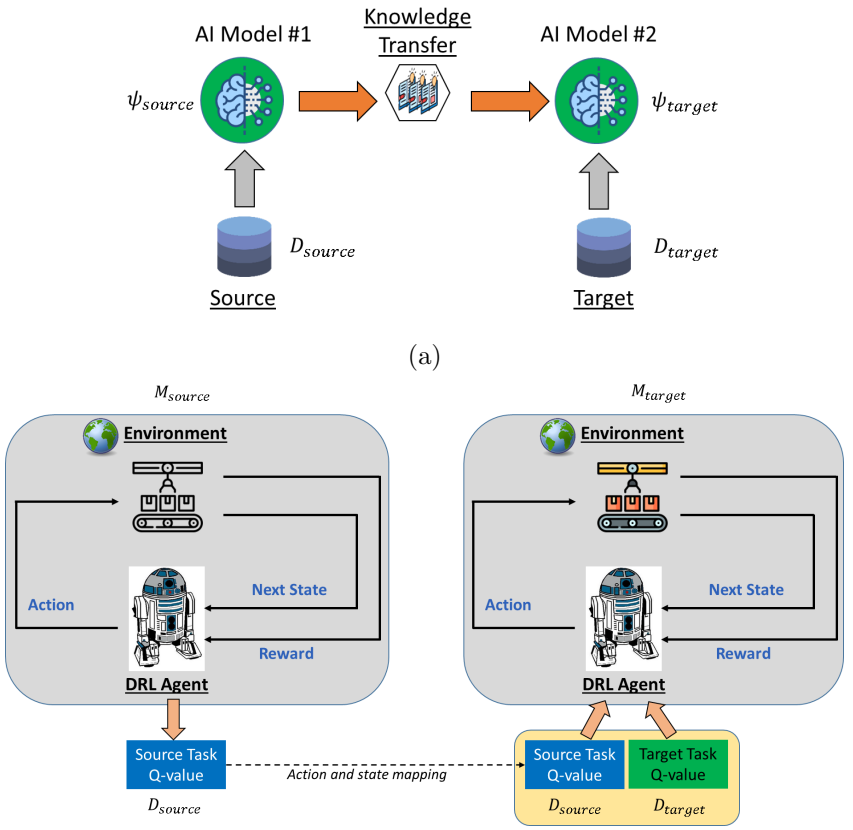


FIGURE 2.7: Transfer learning[1]: (a) Deep Learning; (b) Deep Reinforcement learning;

Recently, [81] presents a TL variant called Knowledge Distillation (KD). KD can extract implicit information from a well-trained model (i.e., teacher), whose inference performs well but requires significant computing resources for model training. Upon creating the structure and objective function of the target DL model, the acquired knowledge is then *transferred* to a smaller DL model (i.e., student) to achieve the highest performance achievable with the drastically reduced (pruned or quantized) target DL model. In other words, existing data acquired through disparate computing resources may be transferred to a new scenario, lowering model development costs and reducing the model training process. The popularity of DL models in the research literature is unquestionable, with much emphasis on methods to efficiently transfer knowledge via DNN [82]. In this regard, the prediction function $f_{\psi}(\cdot)$ represents the non-linear function using DNN with no changes to the aforementioned tuple.

On the contrary, therein lies a subtle difference for transfer learning in DRL. For

instance, the RL-based TL aims to learn the optimal policy π^* for the target domain $\mathcal{M}_{\text{target}}$ and exploits external knowledge $\mathcal{D}_{\text{source}}$ (e.g., experience buffer), gained from the source domain $\mathcal{M}_{\text{source}}$, in addition to the knowledge $\mathcal{D}_{\text{target}}$ learned from $\mathcal{M}_{\text{target}}$. Besides, policy π is learned via a DNN, with Figure 2.7b illustrating the DRL knowledge transfer process. According to [39], TL in DRL is under-explored with many open research challenges.

2.4 Predictive Maintenance

2.4.1 Health Assessment

In-situ sensors in modern manufacturing equipment monitor numerous process parameters for signs of process deviations that can affect product quality and provide early warning of equipment failure, see Figure 2.8 (Steps 1 and 2). However, numerous factors can corrupt sensor data, resulting in anomalous data, in the form of false system alarms and alert notifications, see Figure 2.8 (Step 3). An anomaly is a statistical concept that refers to data points which vary greatly from other measurements, and previous studies [55, 83–85] suggest that the fault evolution of various system degradation models may be modeled as an exponential decay function following:

$$F(t) = e^{-\lambda t}. \tag{2.14}$$

In addition, it is proposed in [6] that the initial degradation conditions, stress, and wear-rate of individual components can also be generalized into a normalized

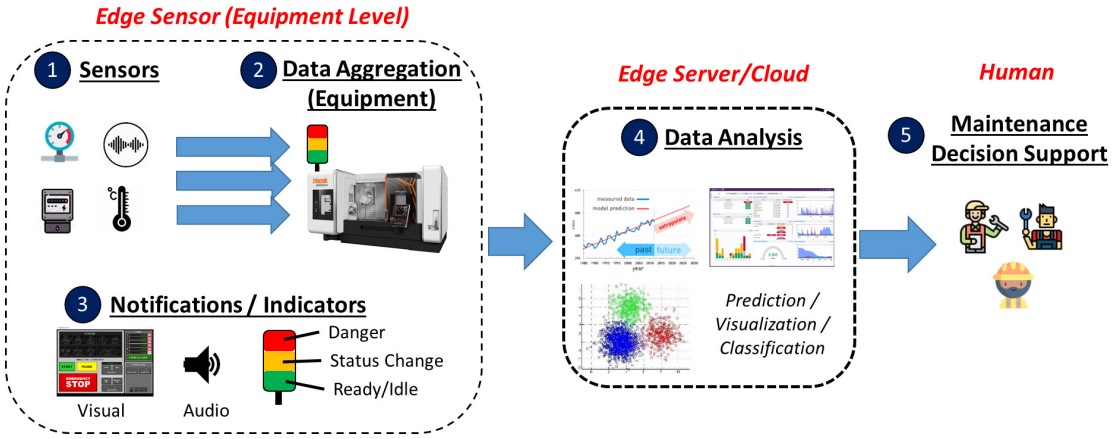


FIGURE 2.8: Predictive maintenance model overview: equipment data generation (Steps 1, 2), equipment data notification and indicators (Step 3), equipment data analysis (Step 4), maintenance decision-support (Step 5).

time-varying health index (H_t) with the following mathematical expression:

$$H_t = 1 - g - e^{at^b}, \tag{2.15}$$

where the initial degradation condition of g is non-zero; a and b are generalized wear-rates that correspond to the effects of temperature and relevant sub-system stress terms [6].

Overall, the health index models the various component sub-systems, and the trained machine-learning model can be deployed on the equipment either in situ or retrofitting of edge sensors [26]. At the same time, decoupling the model training and re-training processes needed for each equipment type opens up a conceptual paradigm shift that significantly reduces the risk of re-training a monolithic machine-learning model. In this thesis, we re-formulate the time-varying health index in [26] to better reflect the maintenance context, with formal definitions described in Chapter 4.

2.4.2 Resource Management

One of PdM's primary functions is to predict a equipment's RUL, which may actually lower the economic cost of maintenance. When the cost model-derived maintenance interval T_i is less than the equipment's remaining useful life T_{RUL}, the cost model result may be used for maintenance decision-making. On the contrary, $T_i > T_{RUL}$ implies that equipment failure is impending prior to the next monitoring point. Therefore, maintenance decisions are made using RUL data received from the PdM model, as shown in Figure 2.8 (Step 4).

The proposed secondary function of PdM is the allocation and management of the maintenance resources, see Figure 2.8 (Steps 5). Maintenance resources, in this sense, refer to a group of maintenance staff comprising engineers and technicians. Effective management of maintenance resources is considered an NP-hard combinatorial problem [86], and conventional mathematical programming techniques are incapable of providing exact solutions within a reasonable time frame. Consider the conventional maintenance routine, where varying complexity of maintenance tasks are allocated to a fixed-shift of technicians with different responsibilities. Simultaneously, the maintenance engineer is accountable for manually allocating

maintenance tasks to suitably competent technicians. In this regard, the decision-maker (e.g., maintenance engineer) may easily have neglected other maintenance factors, such as the cost of technician assigned, complexity of equipment maintenance, and expected mean time to repair. As a result, more equipment downtime is unintentionally incurred relative to the first effort at assigning competent maintenance technicians. Due to the fast-paced nature of the work environment, technicians may perform internal task re-scheduling improvisation, which increases the likelihood of longer-than-planned equipment downtime.

2.5 Summary

Following this chapter's introduction, Chapter 3 discusses PDM-based relevant works, and Chapter 4 introduces DRL as a decision-support tool for PdM-based maintenance teams, laying the foundation for the initial PdM framework. In Chapter 5, we significantly enhance the existing PdM framework and propose a TL-based approach for DRL decision-support considering near real-world production factors. Finally, Chapter 6 brings PdM research journey full circle by proposing a DL model to estimate real-world machine failure. Specifically, we augment an LSTM model with an attention-based mechanism to prioritize the multi-variate input features for improved RUL prediction accuracy on an edge-based IIoT equipment network.

Chapter 3

Literature Review

PdM is a data-driven technique that leverages prediction-based technologies to determine when maintenance is needed [36]. Besides, RUL prediction can be considered a sub-category of the PdM-based resource management schemes [10]. In this thesis, PdM serves two functional purposes, *equipment health assessment* and *maintenance resource management*. This chapter introduces each function with a simplified overview of related works in Figure 1.3). Herein, Section 2.4 brings readers up to speed on the essential concept of PdM related to the *equipment health assessment* (e.g., RUL) of equipment and *maintenance resource management framework*. Notably, we only consider *regression-based RUL prediction* in this thesis. To provide insight into the background of our research undertaking, we summarize the PdM-based resource management approaches in the literature concerning existing approaches in the literature (e.g., reinforcement learning, transfer learning, and resource management frameworks). Finally, we conduct a critical analysis of prior work to justify the research issues addressed in this thesis. Interested readers may refer to [87–89] for further information and in-depth discussions on the reasons why certain methods are superior to others.

3.1 Equipment Health Assessment - Artificial Intelligence for PdM Data Analysis

This section groups the different machine PdM data analysis approaches into two broad PdM categories: *equipment health assessment* and *maintenance resource*

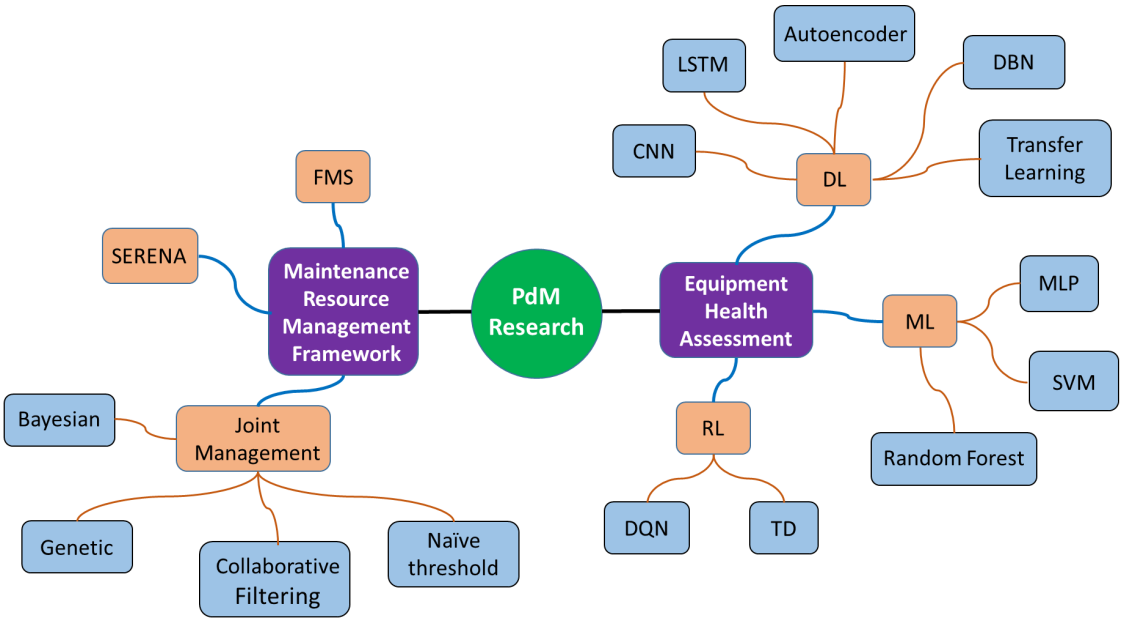


FIGURE 3.1: Mind map overview of approaches to PdM.

management. We outline the various methods within each category with an accompanying short introduction to aid the reader's understanding. Likewise, an overview of existing approaches is illustrated as a mind map in Figure 3.1 and summarized in Table 3.1 for the reader's convenience. Lastly, we refer interested readers to [10, 14] for comprehensive reviews on PdM.

3.1.1 Machine Learning

Data-driven PdM research has been ongoing for numerous years, with various approaches proposing machine learning (ML) based techniques as a baseline. For instance, [49] proposed multi layer perceptron (MLP) for estimating the RUL of aircraft turbofan engines. However, the *shallow* learning capability of MLP is inefficient in extracting hidden features and offers poor RUL prediction accuracy in the presence of noisy raw sensor data [47]. [43, 90] propose support vector regression (SVR), a common supervised learning approach, to map the relationship between input features and RUL. Random forest (RF), a decision tree ensemble, is better suited for time-series regression problems than SVR, as reported in [91]'s findings for tool wear prediction in milling applications. RF tuning is conversely time-consuming and prone to model over-fitting. Nevertheless, these machine learning techniques need extensive human feature engineering to obtain satisfactory results,

prompting mainstream researchers to shift their attention to DL-based approaches instead [48].

3.1.2 Deep Learning

DL-based methods attract increasing interest for RUL prediction applications due to DL's capacity to model complex machine degradation processes [22] while outperforming traditional ML methods. The broad applications and impacts of DL can be found in industrial applications such as RUL prediction of aircraft engines [55], RUL of ball-bearings [22], RUL of battery life [83], signal feature extraction [30], and anomaly detection [92]. Categories of DL include supervised and unsupervised learning methods, with the former accounting for the majority of research owing to the need for ground truth failure information for model validation purposes.

CNN: Based on the seminal work in [93], coupled with the outstanding performance of CNN in large-scale image classification tasks, researchers extended the use of CNN-based techniques for diagnosing machinery faults and image-based tasks, such as performing visual quality inspections [94]. On the contrary, an imbalanced number of research works focus on RUL prediction. For instance, [49] first proposed using vanilla CNN for RUL prediction of aircraft engines by processing raw sensor data as an image, with reports of the CNN approach achieving significantly lesser prediction errors than a range of ML-based approaches. Similarly, [50] presents a temporal CNN (TCNN) (i.e., one-dimensional CNN) that utilizes a time window approach for pre-processing raw sensor data to achieve state-of-the-art results, on the popular C-MAPSS dataset, without domain knowledge expertise. Specifically, the input data preparation remains in a two-dimensional format for compatibility with CNN design, but with one dimension representing sensor data (i.e., feature) and the other representing the time sequence of each feature.

Alternatively, CNN models have been applied to the RUL prediction of bearings with vibration data as the primary source. For example, the authors in [22] present a multi-scale CNN (MSCNN) model that synchronizes the global and local fault information. Notably, the wavelet transform is applied to pre-process data inputs before inputting to MSCNN for feature extraction. Thereafter, a fully-connected layer performs RUL prediction. Similarly, CNN variants of residual CNN[23] and double CNN[22] have also been proposed for RUL prediction. Furthermore, [95] presents a unique joint loss CNN (JL-CNN) model for bearing fault diagnostics and

RUL prediction, often independently addressed. As a result, the hybrid strategy forces the JL-CNN model to map shared feature representations across the two distinct task-based networks before outputting to separate fully-connected layers for RUL prediction and fault diagnostic. The main limitation of JL-CNN is the necessity of more than one training dataset and fault labels to train the two networks.

Autoencoder: Considering AE's capability, only [65, 96, 97] report hybrid AE-based solutions for RUL prediction, with the majority focused on fault classification and diagnostics. For instance, [65] uses deep autoencoder (DAE) for feature extraction and DNN for RUL prediction. Likewise, [97] presents a stacked sparse AE (SAE) for feature extraction and logistic regression (LR) for RUL prediction. Lastly, [96] uses the AE for feature extraction and data analysis techniques, such as data normalization and moving average filter, for RUL prediction. In summary, AE approaches are *unsuitable for direct RUL prediction* purposes.

Deep Belief Networks: In [98], the authors propose feed-forward (FF) based deep belief networks (DBN-FF) for RUL prediction on rotating machine components, such as spiral gear and ceramic bearings. Experimental results illustrate the feasibility of DBN-FF's ability for automatic feature extraction from vibration signals. Although DBN-FF's RUL prediction performance is not comparable to particle filter, DBN-FF can still provide 5 and 50 minutes advanced notice accurately. Considering the disadvantage of the respective ML approaches, [45] proposes the multi-objective deep belief networks ensemble (MODBNE) method, where Deep Belief Networks (DBN) are trained for accuracy and diversity via a multi-objective evolutionary method. The evolving DBNs are then pooled together as an ensemble model for RUL prediction. However, evolutionary-based algorithms require significant parameter tuning, and proposed solutions may not provide the best performance. Besides, DBNs do not account for the two-dimensional structure of the input data or images, which is solvable using the CNN approach.

Recurrent Neural Networks: [53] first proposed the LSTM network for RUL prediction and reported superior performance across several datasets, such as aircraft engines, PHM08 challenge, and data milling. Apart from the proposed LSTM network architecture, the manual data pre-processing step is essential to achieve the aforementioned performance by transforming raw sensor data into a pre-defined

sequence length to predict the corresponding RUL value. Along with improvements to the LSTM model, [54] presents an ensemble of different Bidirectional LSTM (BiLSTM) to predict RUL under various conditions for aircraft engines in a stacked network configuration. However, the proposed BiLSTM network only monitors the fluctuation in the health index, while the RUL was predicted using the recursive one-step ahead technique. On the other hand, [99] presents a bidirectional handshaking LSTM (BHLSTM) architecture for RUL prediction using short observation data sequences, assuming random initial wear of aircraft engines. The handshaking mechanism, in particular, configures the second set of LSTM cells to examine the supplied sequence in reverse order, starting with the most recent observation and ending with another summary vector at the initial observation. As a result, the LSTM network may grasp better features by analyzing the sequence's trend in both directions. In addition, the proposed asymmetric squared error objective function increases RUL prediction accuracy by punishing predictions made later rather than earlier, which is advantageous in reality when the initial machine state is unknown. To mitigate the potentially strong noise signal in RUL prediction, [100] performs RUL prediction and generates the failure probability using an LSTM network.

When modeling very long data sequences, the RNN and LSTM-based models perform poorly, with complex model configurations necessary to detect temporally significant features at the expense of the laborious hyper-parameter search process during model training. Furthermore, LSTM models cannot prioritize relevant features from the given input sequence, which leads to less desirable prediction results [57]. Given LSTM's limitations, alternative models and hybrid LSTM approaches may be suitable solutions for improving RUL prediction performance.

Attention: Recently, there has been growing interest in using the attention-based mechanism to augment LSTM and CNN networks with excellent RUL prediction performance. For instance, [101] proposed the LSTM-AON method and effectively improved the RUL prediction for gear-based problems. Likewise, [102] also achieves excellent RUL prediction performance for bearings application via a gated recurrent unit (GRU) with attention network configuration. For RUL prediction of aircraft engines, [55, 56] achieves state-of-the-art results using attention augmented CNN and LSTM configurations. Compared to [55], [103]'s LSTM-based

multi-layer self-attention (MLSA) performs much better and overtakes [56]'s CNN-based attention model in some performance metrics (e.g., root-mean-square-error and score function).

In general, DL-based solutions are application-specific, with certain approaches being more suited to analyzing specific equipment components than others. Notably, little effort is made to expand the current works' recommended solution to multi-component systems, where the intricate interplay of internal system components may result in cascading machine failures, atypical of real-world equipment [10]. Nevertheless, RUL prediction remains an essential research problem for data-driven PdM, with significant emphasis on the *machine management* aspect.

3.1.3 Deep Reinforcement Learning

Deep reinforcement learning (DRL)'s self-learning capabilities make it attractive to businesses since it eliminates the need for laborious data labeling. As a result, DRL is gaining research traction in PdM applications. For example, [32] proposes formulating a fault diagnosis problem as a diagnostic game, which is then applied to roller bearings and hydraulic pumps with very high classification accuracy results. Similarly, the work in [31] utilizes Deep Q-Network (DQN) for data classification with the UCR dataset [104]. DRL has also been proposed for the management of IIoT wireless networks [105, 106], machine anomaly detection [31, 32], and RUL prediction [33]. Recently, the authors in [33, 107–109] investigated the application of DRL for data-driven maintenance-related problems using the popular NASA C-MAPSS dataset [6]. For instance, [33] proposes a hybrid approach, combining temporal difference (TD) learning with a sequence-to-sequence network to derive the equipment health state degradation and perform RUL prediction. Similarly, [107] uses the Bayesian filtering method to obtain the belief state values from noisy machine sensor states. The belief states are then input to a vanilla DQN for maintenance warning generation and RUL prediction.

Notably, DRL-based PdM research appears to have been mostly limited to fault classifications and RUL prediction until recently. [26] formulates the equipment health indicator as a function of equipment run-time to recommend an appropriate replacement point based on the equipment's health. Their proposed Double-DQN-based solution self-learns a maintenance strategy from the NASA C-MAPSS

dataset and reportedly achieves good learning performance and data sampling efficiency. Similarly, [109] demonstrates that a multi-layer MLP-based DRL method can recommend maintenance repair actions without RUL machine information. Notably, [109] explores hypothetical repair actions very close to complete machine failure, but ignores the impact of the action on the machine's health state. On the other hand, [26] considers replacement actions prior to complete machine failure and the real-world consequences on the machine's health state. Nonetheless, DRL-based research for PdM applications remains under-explored despite the encouraging results.

3.1.4 Transfer Learning

DL-based methods require vast amounts of raw and labeled data to achieve satisfactory results. However, obtaining machine failure data is challenging as modern production machine failure is costly, and running machines to failure is impractical. Hence, transfer learning (TL) is an attractive approach to facilitate knowledge transfer between two highly similar problems (i.e., source and target) to improve the target algorithm's performance in fault diagnosis (i.e., classification, clustering, and anomaly detection). Recently, researchers [110–113] experimented with using transfer learning for RUL prediction. In [110], three techniques are proposed to enable knowledge transfer for an SAE network: weight transfer, weight update, and hidden feature transfer. Based on the acquired cutting tool data, [110] initially trains the SAE model using run-to-failure data and RUL labels before transferring knowledge to a similar cutting tool. Likewise, [111] presents transfer learning for Bidirectional Long Short-Term Memory (BLSTM) recurrent neural networks. Unlike [110], the source trained model is fine-tuned using data from the target task. However, [111] discovered that negative transfer learning instances would occur when knowledge is passed from a more complex source dataset to a simple target dataset. For the RUL prediction of bearings, [112] presents a contrastive denoising autoencoder (CDAE) for extracting features from the marginal spectrum of raw auxiliary bearing vibration data. The Pearson correlation coefficient is used to categorize each bearing's life into normal and rapid deterioration for labeling purposes. A least-square SVM is also used to train an RUL prediction model for quick deterioration. Thereafter, [112] uses an online transfer component analysis to successfully adjust the target bearing characteristics to those of the source auxiliary bearings before estimating the target bearing's RUL. Lastly, [113] propose

a knowledge distillation framework, named KDnet-RUL, for the RUL prediction of aero turbofan engines. Specifically, the generative adversarial network-based knowledge Distillation (GAN-KD) method enables the knowledge distillation from a complicated LSTM model to a simple CNN model for efficient RUL prediction on edge-computing devices. In addition, the learning-during-teaching knowledge distillation (LDT-KD) method for identical architecture knowledge transfer called is presented to improve the RUL performance of the learned CNN model.

The theoretical understanding of TL in unsupervised and reinforcement learning remains an open issue, with [114, 115] reporting satisfying prediction accuracy based on TL for diagnosing similar machine faults. However, the inter-dependency of machines may introduce new faults beyond the expressiveness of TL into the production process [116]. [15] seeks to enhance the production planning forecast accuracy by combining a DAE with DBN. Specifically, ten ensemble DAEs for feature compression coupled with TL yield a 50% increase in *machine management* (i.e., RUL prediction accuracy) for up to 13 machines. In reality, acquiring and incorporating expert demonstration data is often time-consuming, expensive, and technically challenging. Furthermore, estimating the minimal amount of expert demonstration data needed to generate a reasonable learning speed-up is under-explored in the current literature, particularly in the PdM and DRL context.

3.2 Maintenance Resource Management Frameworks

SERENA [117] is a platform for predictive analytics that runs on a lightweight hybrid architecture that combines cloud and edge computing. However, the SERENA-enabled predictive analytic service provides only equipment condition monitoring using a variety of established machine learning algorithms: Random Forest, Decision Tree, and Gradient Boosted Tree. On the contrary, [118, 119] focus on actionable maintenance recommendations. Both works, which are based on the PHM 2013 Data Challenge, utilize collaborative filtering and the Bayesian inference methodology to achieve accurate RUL estimate results for maintenance recommendations. [28] focuses on RUL prediction and proposes the use of Genetic Programming to fuse data from various sensor sources into a composite Health Index, as stated in (2.15). Performance degradation modeling and threshold-based

monitoring approaches have also been recently investigated in [120] to alert users proactively. However, [120]'s approach suffers from frequent false alarms with decreasing trust by the human decision-maker.

From Table 3.1, only a few studies examine PdM in conjunction with resource management. For instance, [34] presents a generic Internet-of-Things (IoT) based framework for enterprise adoption in Fleet Management Systems (FMS) applications. Recently, [121] proposed an Autoencoder-based PdM framework for a data-center-based high-performance computing cluster. However, these approaches do not account for the cost and competency of human operators. [35] introduces Genetic Algorithms (GA) to minimize equipment failure through the joint management of machines and human resources. Conversely, as the problem size and complexity increase, GA-based solutions may be unsuitable for decision-making scenarios that require near-real-time recommendations. In this context, the use of DRL for decision support in PdM systems is non-existent, prompting us to conduct our research investigations.

Ref.	Resource		Fog/Cloud/ Edge/Local	Resource Performance Metrics			Transfer Learning	Method
	Physical	Manpower		Cost	Competency Distribution	Manpower to Machine Ratio		
[105]	✓	-	N/A	-	-	-	-	Deep Deterministic Policy Gradient
[106]	✓	-	Cloud	-	-	-	-	Twin Delayed Deep Deterministic Policy Gradient
[31, 32]	✓	-	N/A	-	-	-	-	DQN
[33]	✓	-	N/A	-	-	-	-	Temporal Difference Learning
[107]	✓	-	N/A	-	-	-	-	Bayesian Filtering, DQN
[108]	✓	-	Edge	-	-	-	-	DDQN-PER-PN
[109]	✓	-	N/A	-	-	-	-	LSTM, Multilayer Perceptron
[15]	✓	-	N/A	-	-	-	✓	DAE, DBN, Transfer Learning
[110]	✓	-	N/A	-	-	-	✓	Sparse Autoencoder, Transfer Learning
[114]	✓	-	Local	-	-	-	✓	Convolutional Neural Network, Transfer Learning
[115]	✓	-	Local	-	-	-	✓	LSTM, Transfer Learning
[34]	✓	✓	N/A	-	-	-	-	IoT Asset & Fleet Management Framework
[121]	✓	-	Cloud	✓	-	-	-	High Performance Computing, Autoencoder, PdM Framework
[35]	✓	✓	Fog	✓	-	-	-	Asset Management Framework, Genetic Algorithm
[24]	✓	✓	**Edge**	✓	**Fixed**	**3:1**	-	**PPO-LSTM, DRL-based PdM Framework**
[18]	✓	✓	**Edge**	✓	**Fixed, Multinomial**	**Many:1, Many:Many**	✓	**TLD, DRL-based PdM Framework**

TABLE 3.1: Overview of resource management methods and AI-based PdM frameworks for IIoT-enabled factory.

3.3 Summary of Research Gaps

To the best of our knowledge, the aforementioned works in the literature do not consider the following aspects:

- **Research Problem #1** - PdM enable maintenance teams to address issues prior to equipment failure, and Table 3.1 summarizes the existing PdM and resource management techniques. As IIoT-enabled PdM is still in its infancy [14], the mainstream research in data-driven prognostics remains laser-focused on improving the RUL prediction accuracy and ignores manpower resource management [35, 122, 123]. Except for [25], none of the current work addresses all four performance metrics: time, cost, competency levels, and maintenance complexity when proposing a resource management method that encompasses physical equipment and human resources. For example, operational and maintenance costs, such as skilled technicians' man-hours, mean time to repair, parts replacement and maintenance budget, are often ignored in existing PdM frameworks. Furthermore, using AI-based decision-support methods to augment human decision-making in complex maintenance scenarios via maintenance strategy remains an open research question. Besides, none of the mentioned works include real-world trials to demonstrate that AI-based solutions are more effective at decision-making than human experts while performing routine maintenance tasks. Therefore, the major focus of the first research problem is defined and is addressed in Chapter 4. The corresponding research outcome is included in Table 3.1 as [24] for the reader's reference.

- **Research Problem #2** - Building upon research problem **#1**, the joint optimization of capital resources, physical machines, and manpower resources in IIoT-enabled manufacturing considers ideal and simplified environment parameters to validate the initial hypothesis. In reality, several challenging realistic factors in IIoT-enabled manufacturing are not considered. For instance, several machines may fail concurrently, and the optimization model must evaluate the entire sequence of maintenance requests before allocating proper manpower to the corresponding machines. Similarly, identical machines may manufacture similar components/parts with varying machine degradation rates. Maintaining part-specific AI models is also inefficient. Furthermore, existing PdM frameworks have not yet explored methods for

knowledge transfer from experts to AI models (see Table 3.1), and TL in DRL models receives little attention for application in PdM-based IIoT-enabled manufacturing. The disparity might be explained by various factors, including the safety and cost of performing online experiments (e.g., running equipment to failure). An alternative approach is to use DRL algorithms that effectively harness historical knowledge (i.e., offline DRL data-driven DRL). By incorporating TL and DRL model into a generic PdM-based resource management framework, human decision-makers may more effectively leverage on DRL model's learning performance and readily generalize across heterogeneously similar machines, using lesser model training resources. Thus, the second research problem is defined and addressed in Chapter 5.1. Again, the corresponding key research outcome is included in Table 3.1 as [18] for the reader's reference.

- **Research Problem #3** - Recall from Section 3.1.4, automating the feature extraction process from raw data is highly desirable, and the proposed models for RUL prediction are application-specific with different degradation features for similar machines. Furthermore, the RUL metric is a derivative virtual feature, with the relevancy of RUL calculation to real-world datasets seldom discussed in the literature. On the other hand, different DL networks are suitable for different PdM applications, with hybrid approaches outperforming standalone DL network designs. Hence, improvements to the state-of-the-art RUL prediction results on similar open-source and real-world datasets necessitate additional exploration in hybrid network designs. Besides, only a handful of PdM works investigate the use of TL methods to overcome data imbalance and reduce model training time. For example, the surveyed works on DL-based TL predominantly adopt AE-based solutions for knowledge transfer, with minimal discussions into TL's generalization to similar machines or datasets. Furthermore, existing approaches overlook the implications and solutions to address model drift for deployed models on edge-based heterogeneous devices - compute and memory bounded. Thus, the third research problem is defined and addressed in Chapter 6. Again, this research problem is tackled during the research attachment at Robert Bosch (South-East Asia) Pte Ltd using the NASA C-MAPSS and real-world IWM dataset. Part of this research outcome is presented in [40].

Finally, we conclude this chapter and examine the research problems in subsequent chapters of this thesis.

Chapter 4

Deep Reinforcement Learning Based Predictive Maintenance Model for Effective Resource Management in Industrial IoT

4.1 Introduction

In this chapter, we formulate the complex resource management problem as a resource optimization problem to determine if a model-free deep reinforcement learning (DRL) based PdM framework[1] can be used to automatically learn an optimal decision policy from a stochastic environment. Unlike the existing PdM frameworks, our approach considers PdM sensor information and the resources of both physical equipment and human as part of the optimization problem. The proposed DRL-based framework and proximal policy optimization long short term memory (PPO-LSTM) model are evaluated alongside baselines results from human participants using a maintenance repair simulator. Empirical results indicate that our PPO-LSTM efficiently learns the optimal decision-policy for the resource management problem, outperforming comparable DRL methods and human participants

[1]The work in this chapter has been presented in [24]. Parts of this work's findings are presented in [25] and [26].

by 53% and 65% respectively. Overall, the simulation results corroborate the proposed DRL-based PdM framework's superiority in terms of convergence efficiency, simulation performance and flexibility.

4.2 System Model and PdM Framework

We consider a generic production facility for predictive equipment maintenance in IIoT (Figure 4.2), which comprises an edge cloud (EC), edge sensor (ES), and manpower resources. At the equipment level, the system model consists of a network of ESs with direct connection to the EC. Due to computational resource constraints, each ES aggregates in situ time-series sensor data and transmits it to EC for storage and analysis, as shown in Figure 4.1 (Steps 1 and 2). Meanwhile, a predictive model (i.e., agent) monitors the incoming data for anomalous behavior and triggers notifications or alarms based on preset parameter threshold.

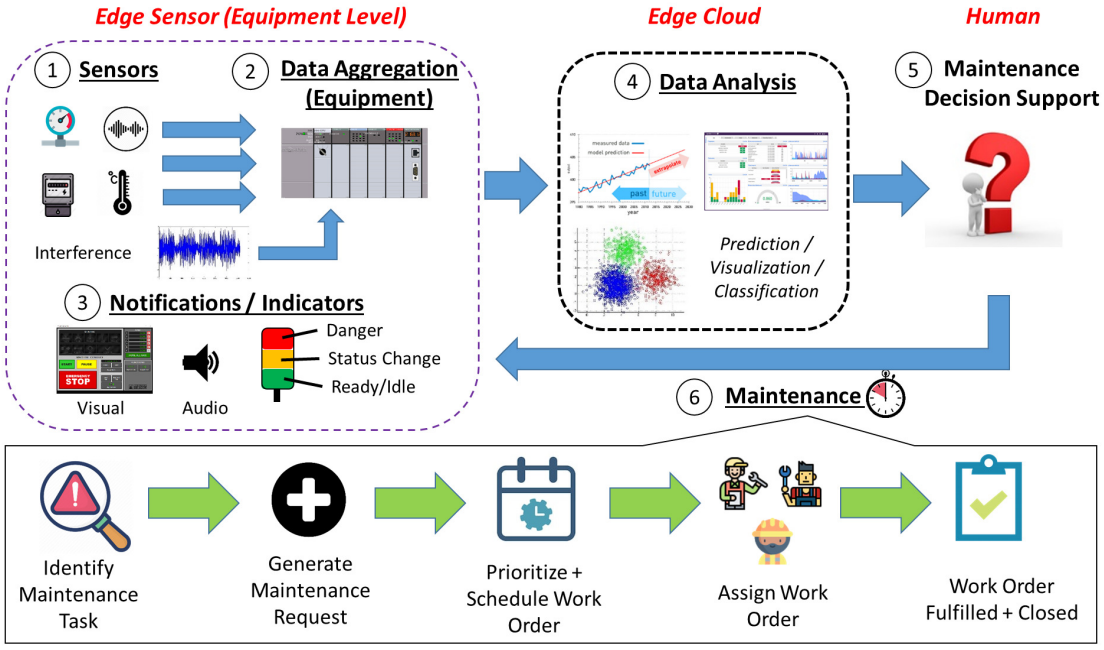


FIGURE 4.1: Predictive maintenance model overview: equipment data generation (Steps 1, 2), equipment data notification and indicators (Step 3), equipment data analysis (Step 4), maintenance decision-support (Step 5), and maintenance task workflow (Step 6).

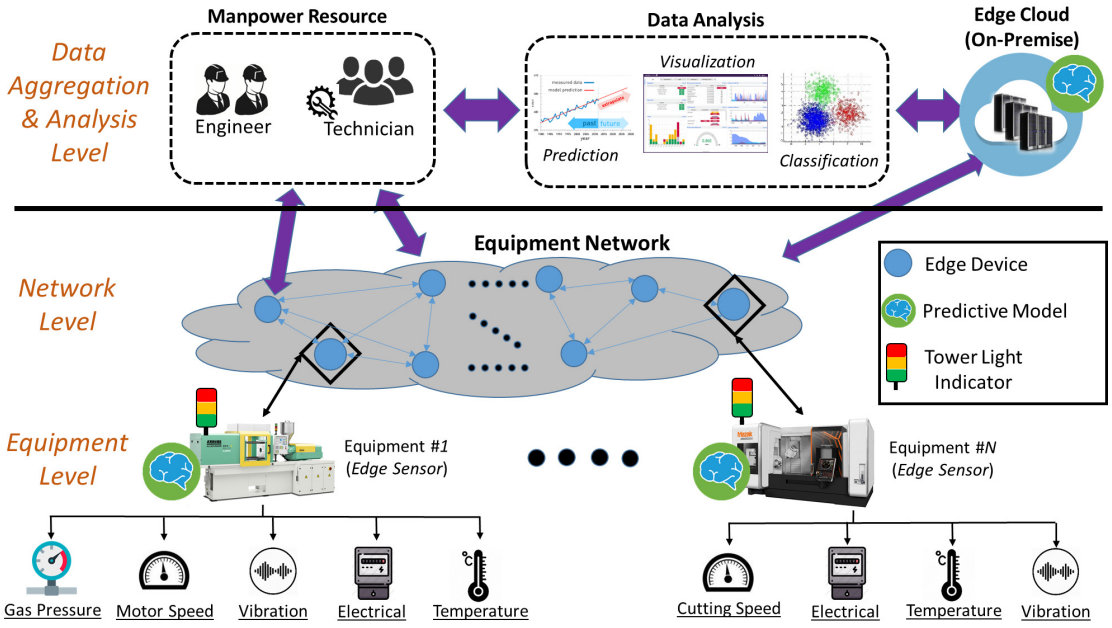


FIGURE 4.2: System overview of the proposed predictive maintenance framework for IIoT networks with edge computing capability per equipment, with on-device predictive models.

4.2.1 Proposed Predictive Maintenance Framework Architecture

Design Objective: The pandemic's widespread effects continues to be felt as businesses struggle to remain financially viable, and low-wage employees like technicians are perceived as replaceable by automation or rehiring. Given the commercial confidentiality on staffing capacity information, we will assume in this chapter that the maintenance personnel to critical equipment ratio increases from 1:1 to 1:10. In particular, our proposed framework considers the real-world constraints of the maintenance budget and technician actions within the decision-making framework, which are both challenging to model and under-explored in similar literature. The proposed framework is briefly illustrated in Figure 4.2 for reference purposes.

Edge Sensor (Equipment): A static ES symbolizes an intelligent sensor predictive model that generates the metadata of the monitored equipment. Metadata is considered to be an abstract representation of the complex relationship between multiple components, where sensors constantly monitor essential component parameters. Examples of sensor data include pump pressure, oil temperature, and milling speed. Generally, a condition-based approach establishes the minimum and maximum limits for each sensor in order to estimate the current health state

of the equipment on the basis of domain knowledge. Such an approach is likely to result in a significant number of false alarms, reduces the overall equipment runtime, and wastes precious network bandwidth. Alternatively, an in situ PdM model may be deployed on an ES device with adequate computational resource, and generates metadata upon detection of abnormal sensor signals, such as probabilistic based alarm information and maintenance action recommendations, whilst preserving precious network bandwidth.

Here, we build on [26]'s work, and the proposed list of available actions is based on the data from multiple sensors observed: [*Severity*, *Repair*, *Replace*, *Hold*]. *Severity* denotes the suspected abnormality pertaining to the operational state of the equipment, raw sensor data inputs, and the severity ratings can either be software or hardware based. Depending on the severity level, the PdM model may recommend either *Repair* or *Replace* to the maintenance team. With adequate data and model training, the PdM model is able to assess the optimal action to be taken via a threshold-free approach, and can be remotely deployed on the ES device in order to achieve the outset objective. To be clear, both *Severity* Level 1 and *Hold* are the default states and actions. Besides, the generated metadata is transmitted to the data aggregation and analysis (DAA) layer via a network of edge sensor network (ESN), which can either be a wired or wireless network. The ESDs need only communicate with the edge cloud with minimal network latency following the use of high-speed communication technologies such as Gigabit Ethernet, Wi-Fi 6 (IEEE 802.11ax), and 5G.

Edge Cloud (On-Premise): In general, the on-premise EC has orders of magnitude more processing power and memory resources than ES. EC offers several functionalities: (1) Storage for industrial big data; (2) Machine learning pipeline analytic such as data pre-processing, model training and testing, prediction and model deployment; (3) Real-time monitoring and analysis of production planning; (4) Resource management for application-specific decision-making. These resources include but are not limited to human, autonomous robots, and departments. In this chapter, we are only concerned with the human manpower resource.

Manpower Resource: Recent survey [16] highlights that a wide variety of competent technicians are expected to retire within the next decade, causing a wider gap in labor's skill/knowledge. To close the gap and achieve effective resource management, we propose that the manpower resource layer in Figure 4.2 offers multiple

functions: (1) Database of relevant employee information, such as experience levels; (2) Estimated manpower cost; (3) Behavioral Risk Profile. We focus on the issues of maintenance management, which are essential to optimizing production time of the machinery. One of the decision-making priorities is to assign the correct maintenance technician, for any given equipment severity rating, in order to maximize the probability of fixing the equipment at the first time round, thereby minimizing the equipment downtime. Furthermore, each technician wears a mobile wireless device that functions as a PdM model feedback mechanism, and the user inputs are captured by means of feedback ratings through the mobile device's touch-screen interface.

4.3 Problem Formulation

Our main objective is to optimize the total production throughput (i.e., equipment uptime) with minimum cost by leveraging the current manpower resource. Given the cost constraints, achieving such trade-off is challenging. We first consider the ESN[2] and DAA layers, and formalize their objectives separately. Then, we merge both layers' definitions and formulate the two-layer interactions. Finally, we conduct problem transformation to contextualize the system parameters and obtain the optimal task assignment strategy. For readers' convenience, we provide a summary of notations used across this chapter in Table 4.1.

4.3.1 Edge Sensor Network - Sensor MetaData

The objective of the ESN is defined in (4.1), with the aim to maximize the cumulative uptime (ρ_{E}) of a network of equipment, where g_q denotes the individual uptime of equipment q ($q \in \{1, 2, \ldots, N\}$). Without loss of generality, a single ES is assumed and communicates directly with the on-premise EC.

$$\max \rho_{\mathrm{E}} = \sum_{q=1}^{N} g_q. \tag{4.1}$$

Random ambient noise can affect sensor measurements, and a higher incident rate of false alarms is to be anticipated if the noise characteristics are not quantifiable or

[2]This model was introduced in [26] with simple assumptions, which does not consider the significant implications of severity rating in a resource management problem within the broader maintenance framework. Furthermore, the use of raw sensor data with an unknown noise function necessitates the use of a different problem formulation, which is not considered in [26], either.

TABLE 4.1: List of Important Notations

Symbol	Definition
ρ_{E}	(Objective) Maximize equipment network uptime
ρ_{D}	(Objective) Optimize resource management
ρ_{H}	(Objective) Optimize user-rating
g_q	Equipment q's uptime
ι	Overall Factory Revenue
N	Number of equipment
M	Number of manpower resource (e.g. technician) set
Γ	Manpower cost w.r.t. experience-level set
τ	Maintenance Budget
$\mathcal{C}$	Cost constraint w.r.t. Maintenance Action
U	User-ratings received w.r.t. contextual situation set
x_t^i	ith sensor's data value at t time-step
H_t	Normalized equipment health value at t time-step
ψ, ψ_t^i	Normalized Equipment Severity Rating set, scalar value with i ratings at time-step t
$\mathcal{S}_{\mathrm{E}}^{\psi}$, $\mathcal{S}_D^{\psi}$	(State) Discrete severity level at ESN and DAA layers respectively
S^Z	(State) Human emotion
n	Number of human emotion categories/levels
Υ	Equipment priority
χ	Array or group of repair actions
κ	(Action) Idle/Hold
η	(Action) Repair
ϵ	(Action) Replace
y_η	Number of repair actions taken until successful fix
y_ϵ	Number of replace actions taken until successful fix
ω_{k}	Positive constants
π^*	Optimal policy
α	Learning Rate
γ	Discounting factor
$\hat{A}_t$	Estimation of Advantage Function at t time-step
G	Number of actors in actor-critic network
d_{target}	Target value of KL Divergence
β	Weighted factor for KL-Divergence
f_t, i_t, o_t	Forget, Input and Output Gates at t time-step
h_t	State of hidden layer at t time-step
b_f, b_i, b_C, b_o	Bias vectors
$c_t, \tilde{c}_t$	Cell state memory, Cell state candidate at t time-step
σ	Sigmoid activation function

observable. Considering external noise as a hidden feature and ES data acquisition is Markovian [33], we propose modeling the complex environment as a sequential decision-making problem using the partially observable markov decision process (POMDP) framework. Within the ESN layer, the predictive maintenance module is denoted as an agent for modeling purposes.

Sensor ($\mathcal{S}_{\mathbf{E}}^{x}$): At each equipment, an agent observes at every time-step t the sensor state information, denoted by x_t^i, where $i \in \{1, 2, \ldots, \varrho\}$; ϱ denotes the upper-bound constraint on the number of sensors per equipment for monitoring and practicality purposes. Considering that the sensor data (i.e., state) is sampled at every time-step, the cumulative states theoretically become a continuous state space, and providing exact solutions within a reasonable time period becomes impractical [76]. We handle this limitation by grouping the data samples into a series of time slices (φ) of constant length μ. Namely, each φ value is a discrete representation of the arithmetic mean sensor data ($\bar{x}_t^i$), such that $\bar{x}_t^i \in \left\{x_1^i, x_2^i, \ldots, x_\mu^i\right\}$. Without loss of generality, $\bar{x}_t^i$ is formalized within the environmental state ($\mathcal{S}_{\mathrm{E}}^{x}$) as $\mathcal{S}_{\mathrm{E}}^{x} \leftarrow \bar{x}_t^i \; ; \; \forall\varphi$.

Operating Condition ($\mathcal{S}_{\mathbf{E}}^{L}$): Owing to the repeated use of equipment and complex environmental conditions, the overall performance of the equipment steadily deteriorates over time and a concave-like exponential decay trend [6] is observed before eventual equipment failure, see Figure 4.3a. Likewise, the sensor's life expectancy decreases over time where some sensors fail faster than others due to ageing and environmental exposure, such as operating temperature. For modeling purposes, the environmental state ($\mathcal{S}_{\mathrm{E}}^{L}$) can be influenced by the operating temperature condition (L) and binary operating status is utilized. For instance, normal operating status ($\mathcal{S}_{\mathrm{E}}^{L} = 0$) is user-defined and conditional on $L \in [25, 70]$. Otherwise, abnormal operating status ($\mathcal{S}_{\mathrm{E}}^{L} = 1$) is assumed.

Severity Rating ($\mathcal{S}_{\mathbf{E}}^{\psi}$): Given N equipment, each ESN-level equipment outputs a severity rating (ψ_t^i), where $i \in [0, \delta]$ at every time-step (t), and the range of severity rating is arbitrarily defined. We normalize i, where $\delta = 1$, and ψ_t^i is formalized within environment state ($\mathcal{S}_{\mathrm{E}}^{\psi}$) as $\mathcal{S}_{\mathrm{E}}^{\psi} \leftarrow \psi_t^i$. The observed severity rating sequentially increases in accordance to the equipment's operational status. For example, we can assume that $\psi_t^{0.2}$ indicates normal operational state while ψ_t^1 indicates critical operational state.

Thus, the overall ESN-based state space $s \in \mathcal{S}_{\mathrm{E}}$ is summarized as an N-set Cartesian product using our system model and POMDP framework as follows:

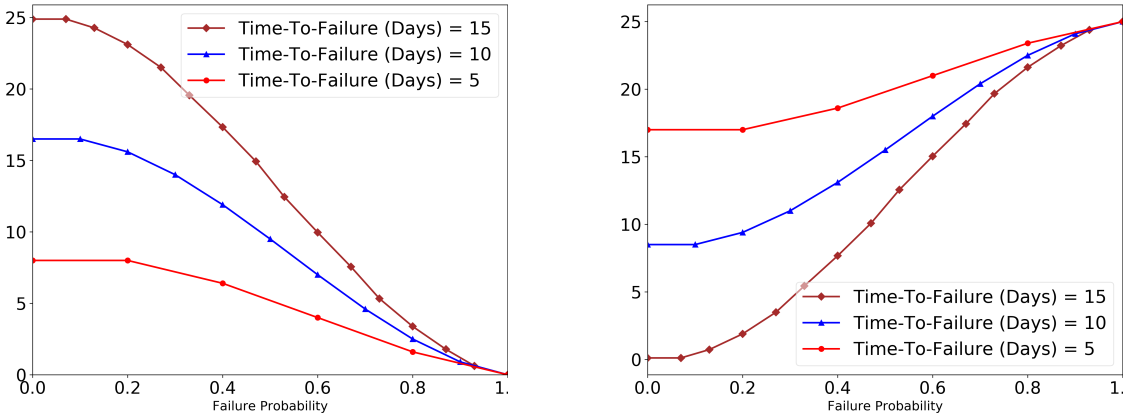
$$\mathcal{S}_{\mathrm{E}} = \mathcal{S}_{\mathrm{E}}^{x} \times \mathcal{S}_{\mathrm{E}}^{L} \times \mathcal{S}_{\mathrm{E}}^{\psi}. \tag{4.2}$$

Action ($\mathcal{A}_{\mathbf{E}}$): The action space comprises of *Hold*(κ), *Repair*(η) and *Replace*(ϵ), and a maintenance budget (τ) is managed by the agent. In order to simulate the maintenance decision-making process, an action-based cost constraint ($\mathcal{C}$) helps to ensure the agent establishes a reasonable maintenance strategy, where $\mathcal{C} \in \{\mathcal{C}_{\kappa}, \mathcal{C}_{\eta}, \mathcal{C}_{\epsilon}\}$ actions are available. By assuming the equipment's health degradation follows (2.14), the agent is then tasked with selecting the sequence of actions to take at each state without violating τ. By default, κ action incurs zero maintenance cost and the list of action cost relationship is defined as follows:

$$\mathcal{A}_{\mathrm{E}} = \begin{cases} (\epsilon, \eta, \kappa) \mid \\ \sum\limits_{q=1}^{N} \mathcal{C}_{\epsilon}^{q} \geq 2 \sum\limits_{q=1}^{N} \mathcal{C}_{\eta}^{q} \;\text{ and }\; \beta - \sum\limits_{q=1}^{N} \mathcal{C}_{\epsilon}^{q} \geq 0, \\ \sum\limits_{q=1}^{N} \mathcal{C}_{\epsilon}^{q} \leq \sum\limits_{q=1}^{N} (\mathcal{C}_{\eta}^{q}/2) \;\text{ and }\; \beta - \sum\limits_{q=1}^{N} \mathcal{C}_{\eta}^{q} \geq 0. \end{cases} \tag{4.3}$$

Belief State Transition ($\mathcal{O}$): Compared to a new sensor, the damaged sensor can record values of the sensor state values that are likely different, and such behavior is detected externally due to multiple state skipping. Assuming that the multiple state skipping phenomenon can be used to infer the existence of an indirectly observable interference, the environmental state (s) is therefore deemed partially observable (o). For convenience, we set o to be s, and the observation-transition probability ($\mathcal{O}$) is instead used to identify the true value of s. In a sense, the environmental state in POMDP is encoded, and the agent relies on a set of *beliefs* (b) for a probability distribution over s, where $b_t(s) = P(s_t = s | o_t, a_{t-1}, \ldots, b_o)$ and b_o is the initial belief vector. The Bayes rule is then used to calculate the belief state transitions, defined as follows:

$$\begin{aligned} b'(s) &= P(s|o,a,b) \\ &= \frac{\Omega(o|s',a) \sum\limits_{s \in S} T(s'|s,a)\, b(s)}{\sum_{s' \in S} \Omega(o|s',a) \sum_{s \in S} T(s'|s,a)\, b(s)}. \end{aligned} \tag{4.4}$$



(a) Relative Equipment Health.

(b) Relative Severity.

FIGURE 4.3: H_t in (2.15) is obtained by dimension reduction of the multiple sensor data acquired using Principal Component Analysis, and follows the decay pattern in (2.14). We then slice H_t over an arbitrary time-interval (in days) and apply the Markov chain rule to obtain the state-transition values as H_t approaches 0, indicative of equipment failure. The corresponding range of severity ratings is based on (4.5).

For an equipment under monitoring, the belief state transitions for each equipment sensor are iteratively calculated and analyzed by the agent. A single set of complementary metadata information is comprised of *Equipment Health* (H) and *Severity* (ψ), defined in (2.15) and (4.5) respectively. Recall, H presumably follows the exponential decay trend in (2.15). Intuitively, we expect ψ to increase with decreasing values of H_t. Formally, $\psi \in \psi_t^i$ is a relative complement of $H \in H_t$, where $H_t \in [0, 1]$ and $\psi \in [0, 1]$. For reference, we visually describe the complementary relationship in Figure 4.3 and is mathematically expressed as follows:

$$\psi = 1 - H. \tag{4.5}$$

Given the stochastic environment, the agent will randomly choose actions from (4.3) to perform based on the observed belief state changes and transitions. Consequently, an equipment's H_t can be restored with ϵ to an almost new condition, while η regresses H_t to a previously observed state by y_η states. Furthermore, the quality of repairs varies and the equipment health state change $\phi(\mathcal{S}_\mathrm{E})$ will differ depending on the $\mathcal{A}_\mathrm{E}$. For example, performing η at $\mathcal{S}_\mathrm{E} = y - 1$ induces a belief state transition from $\mathcal{S}_\mathrm{E} = y$ to $\mathcal{S'}_\mathrm{E} = y - |\phi(\mathcal{S}_\mathrm{E})|$. Such behavior is concisely represented as $y_\eta \in \{\phi(\mathcal{S}_\mathrm{E}) | \mathcal{S}_\mathrm{E} \in \mathcal{A}_\mathrm{E}\}$.

Reward ($\mathcal{R}_{\mathbf{E}}$): To contextualize (4.1) within the POMDP framework, we propose for ρ_{E} be re-expressed as $\mathcal{R}_{\mathrm{E}}$, where $\rho_{\mathrm{E}} \rightarrow \mathcal{R}_{\mathrm{E}}$. Based on the observed values of $\mathcal{A}_{\mathrm{E}}$ and $\mathcal{S}_{\mathrm{E}}$, the agent learns to make better decisions, and this behavior is motivated by the following reward function:

$$r_t(s_t, a_t) = \begin{cases} \mathcal{R}_{\epsilon}, & \text{if } \mathcal{S}_{\mathrm{E}}^{x} > 0, \quad \beta > 0, \\ \mathcal{R}_{\eta}, & \text{if } \mathcal{S}_{\mathrm{E}}^{x} > 0, \quad \beta > 0, \\ \mathcal{R}_{\mathrm{Exp}}, & \text{if } \mathcal{S}_{\mathrm{E}}^{x} > 0, \\ \mathcal{R}_{\mathrm{Frug}}, & \text{if } \mathcal{S}_{\mathrm{E}}^{x} > 0, \quad \beta > 0, \\ -1, & \mathit{Otherwise}, \end{cases} \tag{4.6}$$

where $r_t \in \mathcal{R}_{\mathrm{E}}$, $s_t \in \mathcal{S}_{\mathrm{E}}$ and $a_t \in \mathcal{A}_{\mathrm{E}}$ at every time-step (t). Given current state s_t and $(\tau - \mathcal{C}_{\epsilon} > 0) \wedge (\tau - \mathcal{C}_{\eta} > 0)$, the agent selects either Replace (r_{ϵ}) or Repair (r_{η}) actions and values of s_{t+1} and r_t are received. Otherwise, the agent is penalized where $(s_t = H_t = 0) \vee (a_t = \kappa)$. In order to mitigate the reward sparsity problem in real-world applications, the agent is encouraged to explore the problem state space using $\mathcal{R}_{\mathrm{Exp}}$. Similarly, $\mathcal{R}_{\mathrm{Frug}}$ is defined to positively reinforce the agent's frugal behavior when learning the optimal action to take, emulating real-world human decision-making parameter.

4.3.2 DAA - Resource Management

Recall, the objective of this work is to jointly optimize the factory resources (i.e., machine and manpower) in order to maximize the total production throughput via equipment uptime. For joint PdM management to be effective, the maintenance engineer requires access to pertinent machine information, such as the equipment severity rating (ψ) in (4.5). Otherwise, machine repair prioritization without ψ is subjective and may not be optimal. A maintenance budget (tau) is also necessary to emulate the real-world limitations on manpower hiring. Similarly, the same hiring constraints may be exploited to the optimization model's advantage in order to maximize the total equipment runtime by allocating an adequately competent technician based on values of ψ and tau. Therefore, the optimization objective for DAA may be considered as a logical extension of ESN with pertinent constraints to improve the decision-making process of the proposed optimization algorithm.

Similar to ESN in (4.1), the DAA layer (ρ_{D}) aims to optimize the total equipment run-time (g_q), based on the equipment severity ratings (ψ) and maintenance budget constraints (τ), and is denoted as follows:

$$\max \rho_{\mathrm{D}} = \max \rho_{\mathrm{E}} \mid (\psi, \tau) = \max \sum_{q=1}^{N} g_q \mid (\psi, \tau)\,. \tag{4.7}$$

Given the similarity to ESN's objective, and that DAA's environmental constraints are fully-observable, we exploit this information and re-express the objective function as a fully-observable markov decision process (MDP). Within the DAA layer, the resource management model or human participant is abstracted as an agent for modeling purposes. Every equipment is uniquely represented as $\mathcal{S}_{\mathrm{E}}^{i}$, where $1 \leq i \leq N$ denotes the i^{th} equipment in the equipment network.

Equipment Severity Rating ($\mathcal{S}_{\mathrm{D}}^{\psi}$): Previously in Section 4.3.1, the initial values of $\mathcal{S}_{\mathrm{E}}^{\psi}$ are considered to be partial observations o in the POMDP context. Subsequently, the o values are believed to be accurate because data processing using traditional machine learning technique is utilized to calculate and acquire accurate equipment health H metadata as defined in (4.5). As a result, the o values of ψ in (4.5) can be regarded as the true equipment severity rating state within the fully-observable MDP context. In other words, based on the received severity rating information from $\mathcal{S}_{\mathrm{E}}^{\psi}$ in (4.2), the observed data is assumed accurate and requires no further data processing. Thus, we can mathematically represent $\mathcal{S}_{\mathrm{E}}^{\psi}$ within DAA's environment state ($\mathcal{S}_{\mathrm{D}}^{\psi}$) as $\mathcal{S}_{\mathrm{D}}^{\psi} \leftarrow \mathcal{S}_{\mathrm{E}}^{\psi}$.

User Emotion ($\mathcal{S}_{\mathrm{D}}^{Z}$): Unlike existing works that assume a completely rational agent, the human action is a function of multiple parameters, such as emotional states, age, and risk-aversion attitude [124]. While the human emotional states are stochastic, we employ the Markov chain [125, 126] to model human emotions (S_{D}^{Z}) into n emotional states, such as *Calm*, *Cautious*, and *High Alert* with conditional constraints in (4.8). In addition, other factors may affect the equipment's severity level to behave stochastically with the similar behavior as observed over N equipment.

$$\mathcal{S}_{\mathrm{D}}^{Z} = \begin{cases} \text{Calm}, & \text{if } Z \in [0, 1/(n)] \mid \{n \in \mathbb{R}\}, \\ \text{Cautious}, & \text{if } Z \in [0.01 + 1/n, 2/n] \mid \{n \in \mathbb{R}\}, \\ \text{High Alert}, & \text{if } Z \in [0.01 + 2/n, 1.0] \mid \{n \in \mathbb{R}\}. \end{cases} \tag{4.8}$$

With reference to our system model and MDP framework, we integrate the S_{D}^{Z} into the DAA-based state space $s \in \mathcal{S}_{\mathrm{D}}$ as an N-set Cartesian product as follows:

$$\mathcal{S}_{\mathrm{D}} = \mathcal{S}_{\mathrm{D}}^{(1,\,\psi,\,Z)} \times \mathcal{S}_{\mathrm{D}}^{(2,\,\psi,\,Z)} \times \cdots \times \mathcal{S}_{\mathrm{D}}^{(N,\,\psi,\,Z)}, \tag{4.9}$$

where $s \in \mathcal{S}_{\mathrm{D}}$; $N \in [1, \infty]$ represents the equipment index; $Z \in [0, 1]$ represents the normalized emotional state; n denotes the user-defined levels of human emotional states; ψ denotes the equipment's fully observable severity level state.

Although the degradation behavior of no two identical equipment is alike, continuous equipment usage, after some time, inherently leads to an exponential increase in occurrences of non-operational states, where $\psi_t^i > 1$. In the event where $N > 1$ equipment reports $\psi_t^i > 1$ values, a human operator psychologically associates and combines present state information [127], prioritizes the equipment information received, before focusing on the subsequent course of action.

Action ($\mathcal{A}_{\mathrm{D}}$): The similarities between $\mathcal{A}_{\mathrm{E}}$ and the proposed action space ($\mathcal{A}_{\mathrm{D}}$) is $\mathcal{A}_{\mathrm{E}} \subset \mathcal{A}_{\mathrm{D}}$, where $\mathcal{A}_{\mathrm{D}}$ comprises two additional independent action groups: *Equipment Priority* (Υ) and *Repair Type* (χ). An array of $\mathcal{S}_{\mathrm{D}}^{(N,\,i,\,Z)}$ values, from (4.9) are stored within Υ, and *heuristic* is used to recommend the appropriate equipment sequence to action upon. The scalar actions of *Hold*(κ), *Repair*(η) and *Replace*(ϵ) can be compacted as $\chi \in \{\kappa, \eta, \epsilon\}$. The frequencies of actions ϵ and η are finite, constraint by τ, so as to imitate real-world decision-making. We propose to characterise the maintenance action constraint as $\mathcal{C}$, where $\mathcal{C} \in \{\mathcal{C}_\kappa, \mathcal{C}_\eta, \mathcal{C}_\epsilon\}$ actions are available. Otherwise, κ remains the default action and zero cost is incurred. Labor costs (Γ) in particular take into account the aforementioned maintenance action as well as the skill levels of the dispatched technician. For example, the skill levels can be classified as: *0 to 5 years* ($\Gamma_{l=1}$), *6 to 15 years* ($\Gamma_{l=2}$) and *≥15 years* ($\Gamma_{l=3}$), where $\Gamma \in \Gamma_l \mid \Gamma_l \in \{\Gamma_1, \Gamma_2, \Gamma_3\}$. The action space, which includes the corresponding maintenance action and manpower costs, can then be defined as follows:

$$\mathcal{A}_{\mathrm{D}} = \begin{cases} (\overbrace{\epsilon, \eta, \kappa,}^{\chi} \Gamma, \Upsilon) \mid \\ \sum\limits_{q=1}^{N} \mathcal{C}_\epsilon^q \geq 2 \sum\limits_{q=1}^{N} \mathcal{C}_\eta^q \;\text{ and }\; \tau - \Gamma - \sum\limits_{q=1}^{N} \mathcal{C}_\epsilon^q \geq 0, \\ \sum\limits_{q=1}^{N} \mathcal{C}_\epsilon^q \leq \sum\limits_{q=1}^{N} (\mathcal{C}_\eta^q / 2) \;\text{ and }\; \tau - \Gamma - \sum\limits_{q=1}^{N} \mathcal{C}_\eta^q \geq 0. \end{cases} \tag{4.10}$$

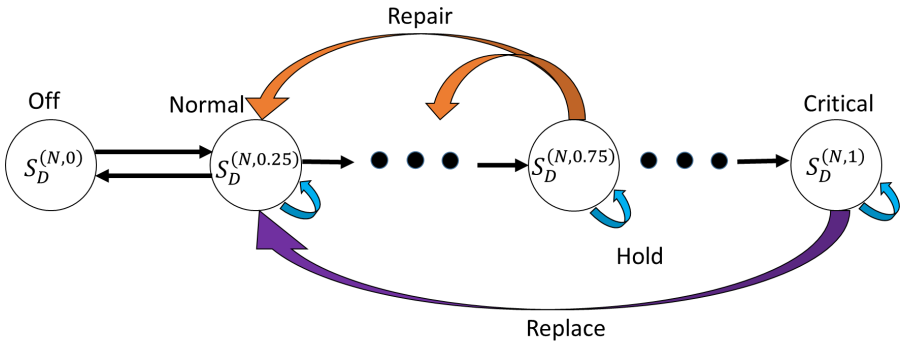


FIGURE 4.4: An example of severity rating state transition for N^{th} equipment w.r.t. types of maintenance action repair performed by the maintenance agent (e.g. human technician or optimization) algorithm.

State-Action Transition ($\mathcal{T}$): Consider that the following $N = 3$ equipment states are observed: $\mathcal{S}_{\text{D}}^{(1,\,0.75)}$, $\mathcal{S}_{\text{D}}^{(2,\,1)}$, $\mathcal{S}_{\text{D}}^{(3,\,0.25)}$. The optimization algorithm first chooses action Υ and heuristically determines the equipment order priority as: $\mathcal{S}_{\text{D}}^{(2,\,1)}$, $\mathcal{S}_{\text{D}}^{(1,\,0.75)}$. Thereafter, the agent will determine an appropriate action to perform, based on the current value i, and notify the appropriate maintenance technician accordingly. However, in the real world, the maintenance technician is unable to perform maintenance on multiple equipment at the same time, and the severity rating of each unattended equipment will remain at current levels, with κ continuously invoked, until the maintenance personnel invokes either ϵ or η. For example, ϵ is performed on $N = 2$ equipment index, and the observed changes in severity rating state ($\phi(\mathcal{S}_{\text{D}})$) transitions from $\mathcal{S}_{\text{D}}^{(2,\,1)}$ to $\mathcal{S}_{\text{D}}^{(2,\,0.25)}$, which is the default equipment severity rating state, see Figure 4.4.

Likewise, η generally reverts the current severity rating state towards $\mathcal{S}_{\text{D}}^{(1,\,0.25)}$. Besides, repair quality is likely to vary, and additional χ actions may be required to adjust an equipment's $\phi(\mathcal{S}_{\text{D}})$ by y_η times. For reader's convenience, we summarize the aforementioned state-transitions in (4.11) and (4.12) respectively.

$$y_\epsilon \in \phi(\mathcal{S}_{\text{D}}), \quad \text{if } \mathcal{S}_{\text{D}}^{(N,\,i)} \in \chi, \quad 0.75 \leq i \leq 1, \tag{4.11}$$

$$y_\eta \in \phi(\mathcal{S}_{\text{D}}), \quad \text{if } \mathcal{S}_{\text{D}}^{(N,\,i)} \in \chi, \quad 0.25 < i \leq 1. \tag{4.12}$$

According to [124], the Wundt curve model [128] is widely used to model the underlying human behavioral trend with respect to increasing rewards and increasing stimulus intensity. Notably, it is possible to re-interpret the Wundt curve model

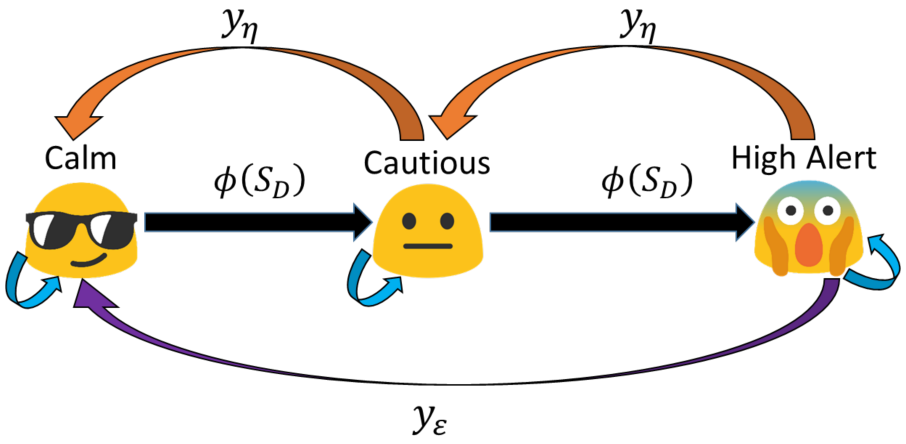


FIGURE 4.5: Proposed human emotional state-transition for maintenance resources based on emotional and mental state transition network models [1, 2] in response to external stimuli [3], such as equipment severity rating.

by splitting it into two partial systems, i.e., a primary reward system and a risk-aversion system with respect to increasing external stimuli [128]. Intuitively, we can correlate external stimuli with equipment severity rating and behavioral based human actions with emotional states. Thus, we can use the state-action transition diagram in Figure 4.5 to visually describe these correlations, and the actions under consideration includes both action groups Υ and χ. [3] empirically validated the plausibility that human emotions undergo temporal state transitions that typically follow exponential decay, with the decay rate also being situational dependent. In the absence of relevant literature for our maintenance-based research problem, we propose following [3]'s state-transition assumptions and evaluate it against the state-transitions from real-world human participant experimental data.

Considering the maintenance context problem, we assume a non-linear correlation exists between $\mathcal{S}_{\mathrm{D}}^{\psi}$ and $\mathcal{S}_{\mathrm{D}}^{Z}$ for N equipment. When, for example, the severity rating state of equipment index $N=1$ rises from $\mathcal{S}_{\mathrm{D}}^{(1,\,\psi=0.2)}$ to $\mathcal{S}_{\mathrm{D}}^{(1,\,\psi=0.3)}$, a similar user emotional transition is predicted following (4.8). Given $Z \in \mathcal{S}_{\mathrm{D}}^{Z} \mid \{n = 3\}$, no emotional state transition occurs until $\psi > 1/n$, which consequently triggers $\mathcal{S}_{\mathrm{D}}^{Z}$ state transition from *Calm* to *Cautious*, as shown in Figures 4.4 and 4.5 respectively. Formally, we can express this state-transition correlation as:

$$\phi(\mathcal{S}_{\mathrm{D}}) \Rightarrow \left\{\phi(\mathcal{S}^{\mathrm{Z}}) | Z \in \chi,\ Z \in [0,1]\right\}. \tag{4.13}$$

Reward ($\mathcal{R}_{\mathbf{D}}$): When invoking an action from $\mathcal{A}$ given severity rating of state $\mathcal{S}_{\mathrm{D}}$,

the quality of the decision-making policy can be improve via the reward function $\mathcal{R}_{\mathrm{D}}(s_t, a_t, s_{t+1})$. Recall (4.7), the cumulative sum of equipment runtime can be re-defined as the cumulative rewards received from N equipment, based on the values of $\mathcal{S}$ and $\mathcal{A}$ taken at each time-step. In consideration of the time-variant repair action, we let the success probability of the η action $P(\varsigma_j = 1 \mid \varsigma_{j-1} = 0)$ be uniformly distributed within Ω time-steps. Hence, a successful repair is denoted as $\varsigma_j = 1$ and the reward function is defined as follows:

$$r_t(s_t, a_t) = \begin{cases} r_\epsilon, & \text{if } \xi \wedge a_t = \epsilon, \\ r_\eta = \Omega - j + 5, & \text{if } \xi \wedge a_t = \eta \wedge \varsigma_j = 1, \\ r_{\eta-1} = +5, & \text{if } \xi \wedge a_t = \eta \wedge , \\ & (0 < \varsigma_j < 1), \\ -0.05, & \mathit{Otherwise}, \end{cases} \tag{4.14}$$

where $r_t \in \mathcal{R}_{\mathrm{D}}$ at every time-step (t) and $\xi \Rightarrow (\mathcal{S}_D^{\psi} > 0, \tau > 0)$; $j \in [1, \ldots, \Omega]$ represents time-to-repair, defining at which time-step the equipment repair is successful. Given the current value of $\mathcal{S}_{\mathrm{D}}$, the reward signal from r_ϵ and r_η corresponds to the Replace and Repair actions respectively. Furthermore, we enforce a negative reward to encourage state-space exploration at every time-step regardless of the state-action pair selection. For the purpose of contextualizing (4.7) within the MDP framework, we re-express ρ_{D} as the maintenance resource reward function $\mathcal{R}_{\mathrm{D}}$, where $\rho_{\mathrm{D}} \rightarrow \mathcal{R}_{\mathrm{D}}$.

4.3.3 DAA - User-Rating

Acquiring user-ratings is often challenging, and the analytical process is complicated by the lack of dependent variables. As in the maintenance scenario, we define the user-ratings $(U \in U_p)$ as an interplay of multiple situational factors, and the user-ratings function (ρ_{U}) is defined as follows:

$$\max \rho_{\mathrm{U}} = U_p \times \psi_t^i \times \Gamma_l, \tag{4.15}$$

where Γ_l and ψ_t^i refers to the skill level of the technician and the equipment severity rating respectively. In other words, both ψ_t^i and Γ_l can be loosely represented as state $s \in \mathcal{S}_{\mathrm{U}} \mid \{\mathcal{S}_{\mathrm{U}} \leftarrow \psi_t^i\}$ and action $a \in \mathcal{A}_{\mathrm{U}} \mid \{\mathcal{A}_{\mathrm{U}} \leftarrow \Gamma_l\}$. M represents the total

number of equipment technicians and the user-defined user-ratings are normalize to $U_p \in [0, 1]$, where $p \in [1, 2, \ldots, M]$. For example, let us consider a user-rating range between 0 (worst) to 10 (best), and the user 2 inputs a feedback rating of 8. Therefore, $U_{p=2} = 0.8$ (i.e., $\frac{8}{10}$).

Next, we cast ρ_{U} into the MDP framework by re-expressing $\rho_{\mathrm{U}} \rightarrow \mathcal{R}_{\mathrm{U}}$, where $\mathcal{R}_{\mathrm{U}}$ denotes the reward function for user-ratings. Consequently, an optimization algorithm (i.e., user-rating agent) learns to select the optimal action ($\mathcal{A}_{\mathrm{U}}$) with respect to the $\mathcal{S}_{\mathrm{U}}$ in order to maximize $\mathcal{R}_{\mathrm{U}}$ received, which generally leads to improved values of U_p.

4.3.4 Overall Problem Formulation

Recall, the aim of this chapter is to collectively maximize overall factory revenue (ι) while optimizing the factory resources, as described in Sections 4.3.1, 4.3.2 and 4.3.3. Taking these multiple objectives into account, the sub-objectives be integrated into the MDP Framework as: $\mathcal{S} \in \{\mathcal{S}_{\mathrm{E}},\ \mathcal{S}_{\mathrm{D}},\ \mathcal{S}_{\mathrm{U}}\}$, $\mathcal{A} \in \{\mathcal{A}_{\mathrm{E}},\ \mathcal{A}_{\mathrm{D}},\ \mathcal{A}_{\mathrm{U}}\}$ and $\mathcal{R} \in \{\mathcal{R}_{\mathrm{E}},\ \mathcal{R}_{\mathrm{D}},\ \mathcal{R}_{\mathrm{U}}\}$.

Considering the optimization problem for resource management, the resource management algorithm (i.e., agent) interacts with the environment state $\mathcal{S}$ at t time-step, selects an action $\mathcal{A}$ to perform, and receives new values of $\mathcal{S}$ and $\mathcal{R}$ from the environment respectively. As a result, the agent will continue to interact with the environment iteratively, optimizing the actions taken based on each state value in order to maximize the cumulative rewards received $\mathcal{R}$. In retrospect, we let $\mathcal{R} \in \mathcal{R}_{\iota}$ by re-expressing $\iota \rightarrow \mathcal{R}_{\iota}$, and the new factory revenue reward function is designed as follows:

$$\begin{aligned} \max\ & \mathcal{R}_{\iota} = (\omega_1 \mathcal{R}_{\mathrm{E}} + \omega_2 \mathcal{R}_{\mathrm{D}} + \omega_3 \mathcal{R}_{\mathrm{U}}) \mid \mathcal{C} \\ \text{s.t. } & (a):\ \omega \in \{\omega_1, \omega_2, \omega_3\}, \\ & (b):\ \omega_1 + \omega_2 + \omega_3 = 1, \\ & (c):\ \mathcal{C} \in \{\mathcal{C}_{\kappa}, \mathcal{C}_{\eta}, \mathcal{C}_{\epsilon}\}, \end{aligned} \tag{4.16}$$

where $\mathcal{C}$ refers to the maintenance action cost constraints from (4.10), and the remaining parameter constraints are defined in (4.16a, 4.16b, 4.16c). Positive weight constants $\omega_1, \omega_2,$ and ω_3 are necessary for balancing the three sub-rewards. For page economy and research scope reasons, we set $\omega_1 = 0.1$, $\omega_2 = 0.9$, $\omega_3 = 0$, where ω_3 is considered for future work.

4.4 Problem Transformation based on RL

The model optimization problem in (4.16) is challenging to solve as the optimization objective requires to manage maintenance resource effectively in the absence of limited or lack of information for modeling purposes. Moreover, the selection of an insufficiently skilled technician to conduct maintenance repair on an equipment with critical severity status potentially leads to a sub-optimal solution (i.e., revenue reduction due to longer equipment downtime). Likewise, learning a hidden Markov model from noisy or stochastic environments is challenging for the reinforcement learning (RL) agent, particularly when incomplete and temporal-dependent environment information are critical to obtaining the optimal solution of the model optimization objective. In this context, traditional model-based dynamic programming tools are also unsuitable due to the significance of time-critical maintenance and an agent's inability to anticipate what the next state will be before the chosen action is taken.

Therefore, in this section, we apply the MDP framework to address our maintenance resource management problem and adopt model-free RL as a solution tool. In what follows, we describe how the MDP framework can be used to achieve optimal decision-making policy. For clarity and brevity, standard RL notations are used for the parameters of *state*, *action*, and *rewards*. The limitations of related RL approaches are briefly highlighted to motivate our proposed DRL solution.

The RL agent's objective is to learn an optimal policy from (4.16) through *trial-and-error* interactions within the stochastic environment. For every interaction with the environment, the agent receives information about the next state s_{t+1} in addition to the current state reward r_t received. Then, the agent attempts to maximize the long-term cumulative expected reward values of being in state s_t recursively, and the optimal state-value policy ($V_*(s)$) is achieved by maximizing the value function, in (4.17), over all existing decision policies, where $\gamma \in [0, 1]$ is the discounting factor.

$$V_\pi(s) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma r_{t+1} | s_t = s]. \tag{4.17}$$

The state transition probability $P(s^{'}, r|(s, a))$ of any stochastic environment is both dynamic and unknown. Hence, the RL agent's strategy is to search recursively for an optimal decision policy $\pi^* : \mathcal{S} \rightarrow \mathcal{A}$ that maps the state $s_t \in \mathcal{S}$ to action $a_t \in \mathcal{A}$.

The Q-learning (QL) algorithm can learn the optimal policy by maximizing the action-value function ($Q_\pi(\mathcal{S}, \mathcal{A})$) over all Q-value policies in (4.18). The Q-value is recursively updated using temporal difference (TD) Learning[76], and the off-policy transitions (s_t, a_t, r_t, s_{t+1}) is learnt.

$$Q_\pi(s, a) = \mathbb{E}_\pi[r_{t+1} + \gamma Q_\pi(s_{t+1}, a_{t+1})|s_t = s, a_t = a], \tag{4.18}$$

$$Q^*(s, a) = (1 - \alpha)Q(s, a) + \alpha Q_{\text{obs}}(s, a). \tag{4.19}$$

The optimal Q-function is obtainable as $Q^*(s, a) = \max_\pi V_\pi(s, a)$. The Q-function is updated following (4.19), where α is the learning rate and $Q_{\text{obs}}(s, a) = r(s, a) + \gamma \max_{a' \in \mathcal{A}} Q'(s', a')$. With $Q^*(s, a)$, the optimal policy is

$$\pi^*(s) = \arg\max_{a \in \mathcal{A}} .Q^*(s, a). \tag{4.20}$$

Reward sparsity is a well-known RL problem and with policy gradient based RL methods, the effect of multiple actions makes it challenging to identify the series of optimal actions to take given a sequence of steps/states, especially in an online setting. In the context of our maintenance problem, the state-of-the-art actor-critic RL approach is more adept and is able to converge to an optimal decision policy in both online and offline policy settings. For instance, the generalized advantage estimator (GAE)[129] approach calculates the advantage of taking an action by using a weighted average of individual advantages over n-steps so as to reduce the variance of the estimator whilst minimizing bias. However, the use of multiple actors with GAE trade-off learning efficiency with high variance in subsequent policy network update intervals [130], which can result in sub-optimal decision policy convergence too. A potential solution would be to integrate a shared long short term memory module into the actor-critic network in order to reduce policy network variance in estimating actions based on current environment state, at the expense of increased GPU memory resource requirements and longer training time.

In this work, we propose using the proximal policy optimization method to improve the policy network variances of actor-critic solutions, which is responsible for action estimation and is further discussed in the next section. Briefly, the proximal policy optimization imposes penalty-like restrictions to further reduce the variance between policy network updates, at the expense of adding some bias while reducing the occurrence of the agent taking sub-optimal actions. Likely benefits

include quicker learning convergence and attaining optimal performance for our maintenance resource management problem. Furthermore, we would also like to investigate whether it is possible for a non-memory-based actor-critic solution to outperform the LSTM-based actor-critic solutions before extending the comparison to even more challenging equipment maintenance problems.

4.5 Proximal Policy Optimization for Effective Maintenance Resource Management

Policy gradient methods operate by calculating an estimation of a policy gradient and optimizing it by using the stochastic gradient ascent algorithm. The estimator $\hat{g}$ can be obtained by differentiating the objective

$$\mathcal{L}^{PG}(\theta) = \hat{\mathbb{E}}_t \left[\log \pi_\theta (a_t|s_t) \hat{A}_t \right], \tag{4.21}$$

where $\hat{\mathbb{E}}_t [\cdots]$ denotes an empirical average expectation of a batch of finite samples within a sampling and optimization algorithm; π_θ denotes a stochastic policy and $\hat{A}_t$ is an estimation of the advantage function at time-step t.

Policy gradient methods suffer from two main problems: *Unstable Policy Updates* and *Data Inefficiency*[76]. Policy changes are unpredictable for policy gradient methods because of their large step updates leading to poor policy updates, which consequently lead to learning bad policies. On the contrary, smaller step updates lead to slower learning. It is also preferable for these learning methods to learn from recent experience and exploit. However, current policy gradient methods discard this experience following gradient changes and this exacerbates the learning process, as a neural network requires a large amount of data to learn effectively. In this section, we propose that these issues be mitigated through the proximal policy optimization (PPO) algorithm[130]. Furthermore, to cope with stochastic environments with long-term dependencies, we suggest supplementing PPO with LSTM, which we explain in this section.

4.5.1 Objective Clipping

Based on the idea of importance sampling and a neural network's preference for normalized data, PPO requires to maintain two policy networks. The first policy

network $\pi_\theta(a_t|st)$ is used to refine the policy updates based on the previous policy $\pi_{\theta_{\text{old}}}(a_t|st)$, in which this ratio is clipped and the minimum of the both policy actions will instead be considered. In doing so, large policy network updates will be restricted by the clipping threshold (ϵ), and the clipped objective function is described as follows:

$$\mathcal{L}^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\right)\hat{A}_t \right], \tag{4.22}$$

where the probability ratio $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}\hat{A}_t$ and $r_t(\theta_{\text{old}}) = 1$. Depending on the value of $\hat{A}_t$, the choice of clipping ratio, $\text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)$, can be either $1-\epsilon$ or $1+\epsilon$ interval range. The pseudocode is shown in Algorithm 1.

Algorithm 1: PPO with Clipped Objective

Input: Initial policy parameters θ_0, clipping threshold ϵ
for *k=0,1,2,...* **do**
  Collect set of partial trajectories D_k on policy $\pi_k = \pi(\theta_k)$
  Estimate advantages $\hat{A}_t^{\pi_k}$ using GAE algorithm [129]
  Compute policy update:
    $$\theta_{k+1} = \underset{\theta}{argmax}\, L_{\theta_k}^{CLIP}(\theta)$$
  by taking K steps of minibatch SGD (via Adam), where
    $$\mathcal{L}_{\theta_k}^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\right)\hat{A}_t \right]$$
end

4.5.2 Adaptive Kullback-Liebler Penalty Coefficient

With reference to trusted region policy optimization (TRPO)[131], we can assume that the optimal policies calculated in the trust region are always better, with some upper bound guarantee, over the old policy. Thus, the objective function can be calculated as:

$$\max_\theta \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}\hat{A}_t \right] - \beta KL\left[\pi_{\theta_{old}}(\cdot|s_t), \pi_\theta(\cdot|s_t)\right], \tag{4.23}$$

where β induces a weighted factor to the kullback-liebler (KL) KL-divergence penalty, to penalize or incentivize some target value of KL-divergence (d_{target}) during policy updating. In other words, the KL-penalized objective, after several policy updates by stochastic gradient descent, can be written as:

$$\mathcal{L}^{KLPEN}(\theta) = \hat{\mathbb{E}}_t \left[r_t(\theta) - \beta KL\left[\pi_{\theta_{old}}(\cdot|s_t), \pi_\theta(\cdot|s_t)\right]\right]. \tag{4.24}$$

Likewise, the KL-divergence, denoted as d in (4.25), is also computed after every policy updates such that if $d < d_{\text{targ}}/1.5, \beta \leftarrow \beta/2$; $d > d_{\text{targ}} \times 1.5, \beta \leftarrow \beta \times 2$. Then, the updated β value is used in the next policy update interval.

$$\hat{\mathbb{E}}_t \left[r_t(\theta) - \beta KL \left[\pi_{\theta_{old}}(\cdot|s_t), \pi_\theta(\cdot|s_t) \right] \right]. \tag{4.25}$$

As a result, PPO is able to inherit TRPO performance, and is optimized by gradient descent methods. For reference, the pseudocode algorithm is described in Algorithm 2.

Algorithm 2: PPO with Adaptive KL Penalty Coefficient

Input: Initial policy parameters θ_0, initial KL penalty β_0, target KL-divergence δ
for *k=0,1,2,...* **do**
  Collect set of partial trajectories D_k on policy $\pi_k = \pi(\theta_k)$
  Estimate advantages $\hat{A}_t^{\pi_k}$ using GAE algorithm [129]
  Compute policy update:
    $\theta_{k+1} = argmax L_{\theta_k}(\theta) - \beta_k D_{KL}(\theta||\theta_k)$
  by taking K steps of minibatch SGD (via Adam)
  if $D_{KL}(\theta_{k+1}||\theta_k \geq 1.5\delta)$ **then** $\beta_{k+1} = 2\beta_k$
  else if $D_{KL}(\theta_{k+1}||\theta_k \leq \delta/1.5)$ **then** $\beta_{k+1} = \beta_k/2$
  end if
end

4.5.3 Recurrent Neural Network

Long short term memory (LSTM)[52] is a variant of recurrent neural network, and is often used in DRL literature for spatial-temporal feature learning. Individual cells can extract feature states across a recurrent network while preserving temporal information within each cell state, and LSTM uses a gated-like structure for selective transmission of sequential information. Each LSTM cell consists of input gates i_t, output gates o_t and forget gates f_t. We formally describe the gate structures as:

$$\begin{aligned}
f_t &= \sigma_l(W_f \cdot [h_{t-1}, x_t] + b_f), \\
i_t &= \sigma_l(W_i \cdot [h_{t-1}, x_t] + b_i), \\
\tilde{c}_t &= \tanh(W_c \cdot [h_{t-1}, x_t] + b_c), \\
c_t &= f_t * c_{t-1} + i_t * \tilde{c}_t, \\
o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o), \\
h_t &= o_t * \tanh(c_t).
\end{aligned} \tag{4.26}$$

From (4.26), we denote $W_f, W_i, W_c,$ and W_o as weight matrices and $b_f, b_i, b_c,$ and b_o as bias vectors for input vector of sensor data x_t at time-step t; h_t represents the hidden layer's state at time-step t whereas h_{t-1} represents the hidden layer's state at time-step $t-1$, and c_t denotes the cell state memory at t time-step; $\tilde{c}_t$ denotes a vector of candidate cell states at time-step t; $*$ denotes the element-wise multiplication of the vectors and the gated structure behavior follows a sigmoid activation function σ.

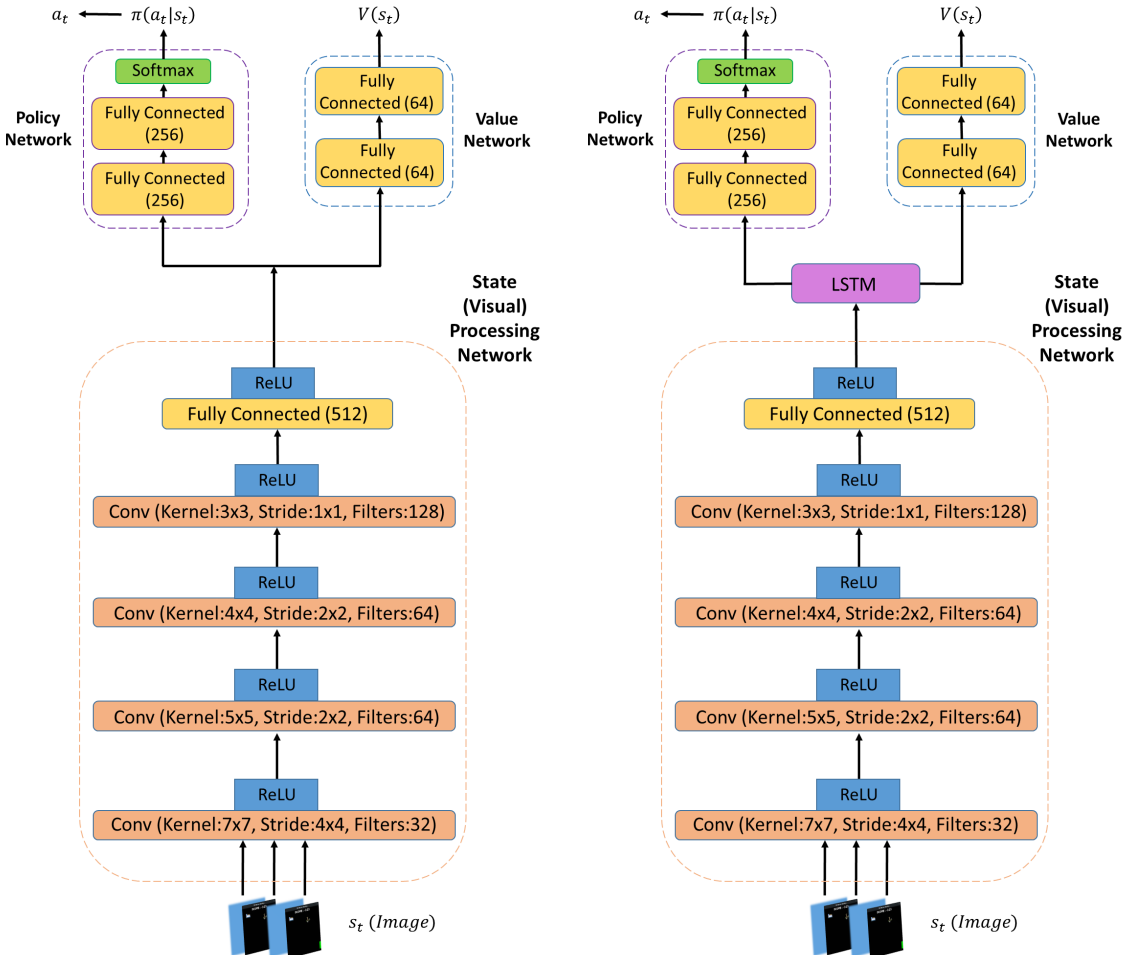
4.5.4 PPO Algorithm

For our maintenance simulation problem, we consider an image-based state space (i.e., 500 x 500 pixels) with state representation at the pixel level. Therefore, a convolutional neural network(CNN) is warranted for PPO's policy and value function to learn via shared parameters, hereby termed PPO-CNN. For optimization purposes, a loss function is necessary to backpropagate the policy target and value function gradients. In PPO, we also consider an entropy bonus S to manage the state-space exploration and exploitation trade-off in a similar way to the epsilon-greedy strategy of Deep Q Network. Following [130], S can be combined with (4.24), (4.22) and (4.21) at each iteration to optimize an overall objective function, defined as follow:

$$\mathcal{L}^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}_t \left[\mathcal{L}^{CLIP}(\theta) - c_1 \mathcal{L}^{VF} + c_2 S[\pi_\theta](s_t) \right]. \tag{4.27}$$

From (4.27) we denote $c_1,$ and c_2 as regularisation coefficients, S denotes an entropy bonus; $\mathcal{L}^{VF}$ is a compact representation of the squared error loss between the learned state-value function $V(s)$ and the target state-value function as $\left(V_\theta(s_t) - V_t^{\text{target}}\right)^2$.

One major drawback of parameter sharing, of both value and policy networks, is performance instability due to the simultaneous backpropagation of gradients across the network during the model learning phase. As such, we plan to empirically identify suitable hyperparameters to manage this issue. In this work, the PPO algorithm to be implemented utilizes a fixed-length trajectory where G actors will each collect T time-steps of training data in parallel. Then, the losses for each corresponding objectives on GT time-steps of data will be optimized using a gradient descent-based methods for K epochs, such as the adaptive moment estimation optimizer (Adam).



(a) CNN based PPO (PPO-CNN).

(b) CNN+LSTM based PPO (PPO-CNNLSTM).

FIGURE 4.6: PPO-based actor-critic architecture variants.

The above-mentioned solution, however, is suitable for stochastic environments with short spatial-temporal dependencies (i.e., state-action value pair) and may not achieve optimal results, as it requires the PPO agent to retrieve older state-action sequence information. For example, given the same equipment and technician, the mean-time-to-repair of the equipment can vary greatly due to the different rate of equipment degradation and environmental factors. To address this issue, we propose to modify existing PPO-CNN (Figure 4.6a) architecture by inserting an LSTM layer between the Convolutional and Feedforward layers. This model variant is termed PPO-LSTM (Figure 4.6b), and the cells in the LSTM layer with $h_t = \text{LSTM}\,(o_t,\ h_{t-1})$ are thus used to estimate the $Q\,(h_t, a_t)$ instead of $Q\,(s_t, a_t)$. To be clear, o_t refers to the current observation o_t and may not necessarily correspond to the current environment state s_t while h_{t-1} represents the state of the hidden layer at time-step $t-1$.

In summary, the proposed policy network constraints of *objective clipping*, *adaptive KL divergence penalty* and *entropy bonus* will be validated with an actor-critic based neural network solution. For reader's understanding, the proposed actor-critic based PPO architecture variants are shown in Figure 4.6 and the pseudocode is given in Algorithm 3.

Algorithm 3: PPO, Actor-Critic Style

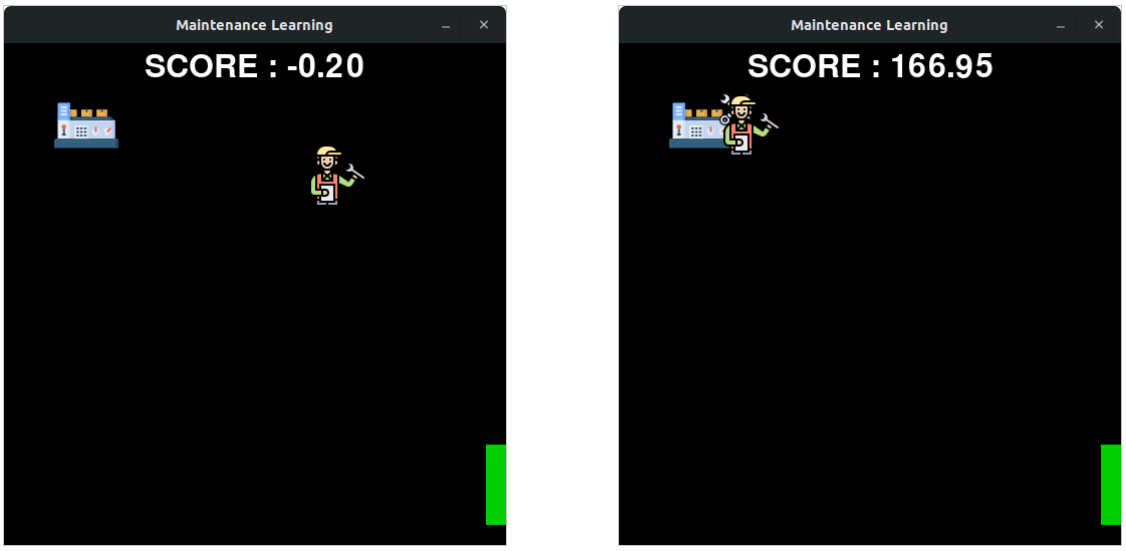
for *iteration=1,2,...* **do**
  for *actor=1,2,...,N* **do**
    Run policy $\pi_{\theta_{\text{old}}}$ in environment for T time-steps Compute advantage estimates $\hat{A}_1, \ldots, A\hat{} + t$
  end
  Optimize surrogate L w.r.t. θ, with K epochs and minibatch size $M \leq GT$
  $\theta_{\text{old}} \leftarrow \theta$
end

4.6 Experiment Setup

Due to the scope of the proposed DRL framework, we propose and describe three experiments in this section.

4.6.1 Maintenance Repair Simulator

We create a maintenance repair simulator (MRS) that is adequately versatile for a range of purposes, such as model training and validation for the DRL agent and data collection from human participants for benchmark purposes. As a baseline, we set $\Omega = 120$ in MRS and at each simulation step, the machine repair success probability decreases linearly as $\frac{1}{120}$, $\frac{1}{119}$ at $i = 2$, and so on to emulate real-world scenario in which repair times differ with varying equipment severity rating. Similarly, we set 5 severity levels and generate 3 technician skill levels (eg. Junior, Senior, Expert). Through iterative interaction with MRS, the rewards received by the DRL model also takes into account the different cost constraints and will, based on the skill level of the selected technician, learn an optimal decision-making policy to satisfy (4.16). We then report the corresponding results for all severity levels, technician skill levels, and PPO respectively.



(a) Random Initial State.

(b) Repair Action State.

FIGURE 4.7: Maintenance Repair Simulator experimental interface.

An example of the MRS environment is shown in Figure 4.7a, which comprises of 3 interactive objects: the potentially malfunctioning Equipment, Exit Door (i.e., the green rectangle) and the Avatar representing a human attendee to Equipment. For each new trial, the avatar's initial starting position is random, and only the Avatar icon is controllable by the test subject. As the Avatar navigates and interacts within MRS, several behavioral tendencies may be observed. For instance, the Avatar may rush straight towards the exit and obtain an exit reward, randomly wander around or idle. When the Avatar is right beside the equipment, the "*Repair*" action can be performed to receive a positive reward, see Figure 4.7b, and the current trial session ends. Similarly, the current trial session reaches the terminal state if the agent selects the "*Exit Door*" option. Otherwise, the current trial session terminates at the end of the trial time-limit, with a negative reward. In order to enable the inference of human risk-attitudes in MRS, we conveniently categorize frequent *"Exit Door"* activities and random wandering behavior tendencies as *risk-seeking*, and activities that lead to the *"Repair"* action as *risk-averse*, see Figure 4.7b. Interested readers may refer to [25] for additional details pertaining to the MRS environment.

4.6.2 Human Participants

A total of 26 working professionals participated in our IRB-approved experiment which consists of both white-collar and blue-collar workers. The mix-gender participants, aged between 20 to 50 are from Singapore and China, and the overall average participant age falls within the range of 30 to 40 years old. Each participant is presented with a set of instruction and the game objective, which is to maximize the total game rewards received. Relevant game data is automatically captured and every human participant utilizes the keyboard to navigate the MRS game environment. Additionally, a warm-up game followed by two games is permitted for each participant, where each game comprises of 30 rounds of gameplay, for example. Yet unbeknownst to all participants, they will play the same game environment under four different game difficulty, where game difficulty is synonymous with equipment severity rating.

The experimental data for all participants is aggregated, pre-processed, and we perform the statistical analysis. Example of data collected for each human participant are user's score, time taken to game completion, and a snapshot of actions taken to complete each game. Thereafter, we demonstrate the efficacy of the AI-based solution for maintenance decision-making by comparing the human participant results to the proposed DRL variants. For disclaimer purposes, the anonymity of all human participant is strictly enforced for data security and privacy purposes throughout the course of experiments.

4.6.3 Turbofan Engine Dataset and Data Preparation

The NASA commercial modular aero-propulsion system simulation (C-MAPSS) dataset[6] is generated from a commercial degradation simulator for turbofan engines. It includes measurements that simulates failure under various operating conditions for several turbofan engines. A quick overview of the engine datasets FD001 and FD003 are shown in Table 4.2 as well as the respective fault conditions across multiple sensor measurements.

Each dataset consists of 26 data columns. Columns 1 and 2 refer to engine cycle for specific engines; Columns 3, 4, and 5 denote the sensor measurements, such

[3]https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/ #turbofan

Dataset	FD001	FD003
Training Set	100	100
Test Set	100	100
Operating Conditions	1	1
Fault Conditions	1	2

TABLE 4.2: C-MAPSS Dataset[3]under test.

as temperature and pressure; the remaining columns reflect the simultaneous condition monitoring of 21 sensors. We apply standard data normalization [26] on the sensor data and assume equipment degradation behavior following (2.15). The objective of this experiment is to learn and consistently recommend an effective replacement action ϵ before imminent failure of turbofan engines, based on varying states of equipment health degradation.

4.7 Results and Discussion

We shall briefly highlight the organisation of results for reader's convenience, with details in the respective sub-sections. Firstly, we present the results from multiple experiments in order to highlight the benefits of PPO, based on MRS Game-1, and articulate the performance contributing factors for the policy network components in comparison to the baseline models and existing work. Secondly, we present results from the human participants and highlight important statistical findings. Table 4.3 is then compiled to aggregate both the human participant and DRL results to illustrate the potential significance of augmenting human technicians in complex decision-making situations. In doing so, we demonstrate the applicability of PPO at the DAA level. Finally, we discuss the effectiveness of our proposed DRL system by presenting key results from C-MAPSS, which is utilized to mimic in situ decision-making at the equipment or ES layer. Furthermore, all model results reported in this section are the median average over 5 independent runs.

4.7.1 Performance Evaluation of Proposed Algorithm

Empirically, a deeper CNN design (i.e., PPO-CustomCNN) benefits from an increase of 7% in learning efficiency and 2% improvement in mean score deviations, when compared to the 3-layer CNN PPO algorithm (i.e., PPO-CNN). For baseline purposes, we hereby denote the non-clipped form of PPO to be implicitly convergent (I.C.), and benchmark PPO's performance against the advantage actor-critic

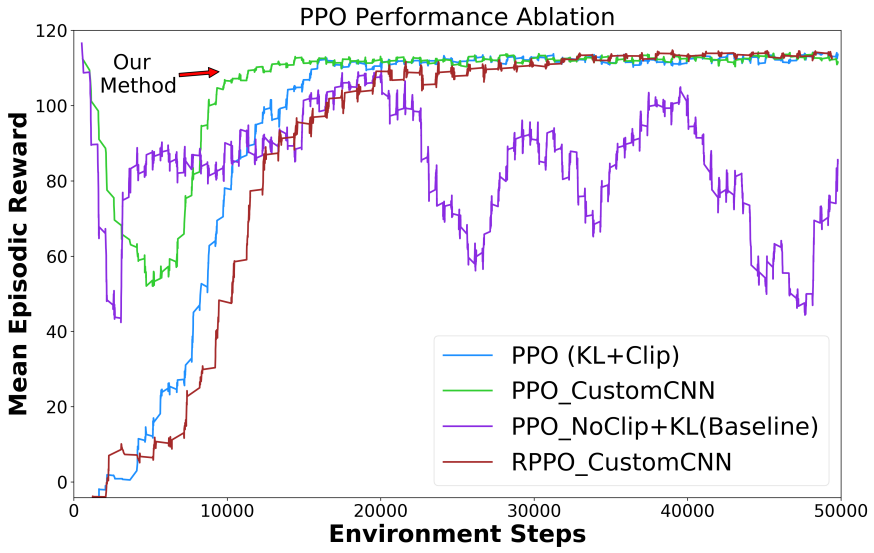


FIGURE 4.8: Performance of our proposed PPO solutions.

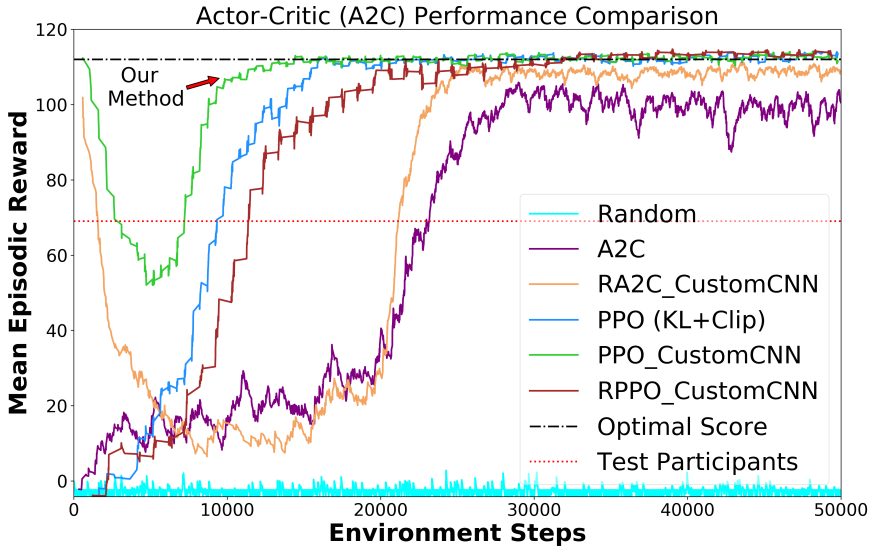


FIGURE 4.9: Performance comparison between A2C variants, PPO variants, and human participants.

(A2C) variants, in Figure 4.9. Notably, the proposed PPO variants are able to achieve up to 42% increase in learning efficiency. Besides, PPO reliably achieves an optimal score for Game-1, unlike the A2C variants which are merely near optimal. Besides, the performance of PPO with clipping is almost empirically identical to previous state-of-the-art results (i.e., A2C-LSTM). For completeness, we also include the PPO-CNNLSTM variant as part of our performance comparison, and the results are shown in Figures 4.8 and 4.9.

Compared to atypical A2C networks, PPO is more robust to hyperparameter tuning. On the contrary, the PPO-CNNLSTM variant (i.e., RPPO_CustomCNN) requires a reasonably comprehensive hyperparameter tuning in order to achieve learning convergence. In other words, we discover that increasing the number of recurrent cells in the LSTM network to twice the step size of PPO yields best results and convergence. One interesting insight from PPO-CNNLSTM variant is that it appears to trade-off learning efficiency with higher overall scores as opposed to non memory-augmented PPO policies. A beneficial side-effect of the LSTM network inclusion is the overall reduction of policy variance losses, and learning convergence variability in reward scores, as shown in Figure 4.8. Across all 4 games, the PPO-CNNLSTM design on average outperforms both PPO-CNN and A2C-CNN-LSTM by 4% and 3% respectively. For reader's convenience, we highlight the top performers per game, in bold, and the experimental results are listed in Table 4.3.

Game Modes	**Time-steps to Learning Convergence (10^3)**					**Mean Reward Received**					**Human Participants Score**
	A2C	**PPO**				**A2C**	**PPO**				
	CNN + LSTM	CNN	(w/CustomCNN)			CNN + LSTM	CNN	(w/CustomCNN)			
		w/ Clipping	PG No Clipping	Clipping	Clipping + LSTM		w/ Clipping	PG No Clipping	Clipping	Clipping + LSTM	
Game 1 (P(Fix)=1.0)	26	16	I.C.	**15**	32	108 ± 9.7	112 ± 6.2	88 ± 38.5	112 ± 6.1	**113** ± **5.5**	66 ± 47
Game 2 (P(Fix)=0.9)	27.5	**19**	I.C.	24	32	95 ± 9.5	98 ± 6.5	81 ± 28.8	**98** ± **6.2**	97 ± 7.1	72 ± 42
Game 3 (P(Fix)=0.6)	28	**12.5**	I.C.	15.5	21	143 ± 84.7	134 ± 55.1	92 ± 68.8	138 ± 46.8	**154** ± **48.3**	70 ± 52
Game 4 (P(Fix)=0.5)	31.5	**18.5**	I.C.	**18.5**	26	121 ± 63.4	115 ± 39	61 ± 57.9	114 ± 36.1	**118** ± **32.4**	88 ± 51

TABLE 4.3: Sample efficiency (lower is better) for A2C and PPO on 4 game environments are shown with and without the proposed optimizations. Performance results (higher is better) of each test with the mean rewards obtained for each benchmark. For comparison, the human participants score are also shown.

4.7.2 Human Participant Analysis

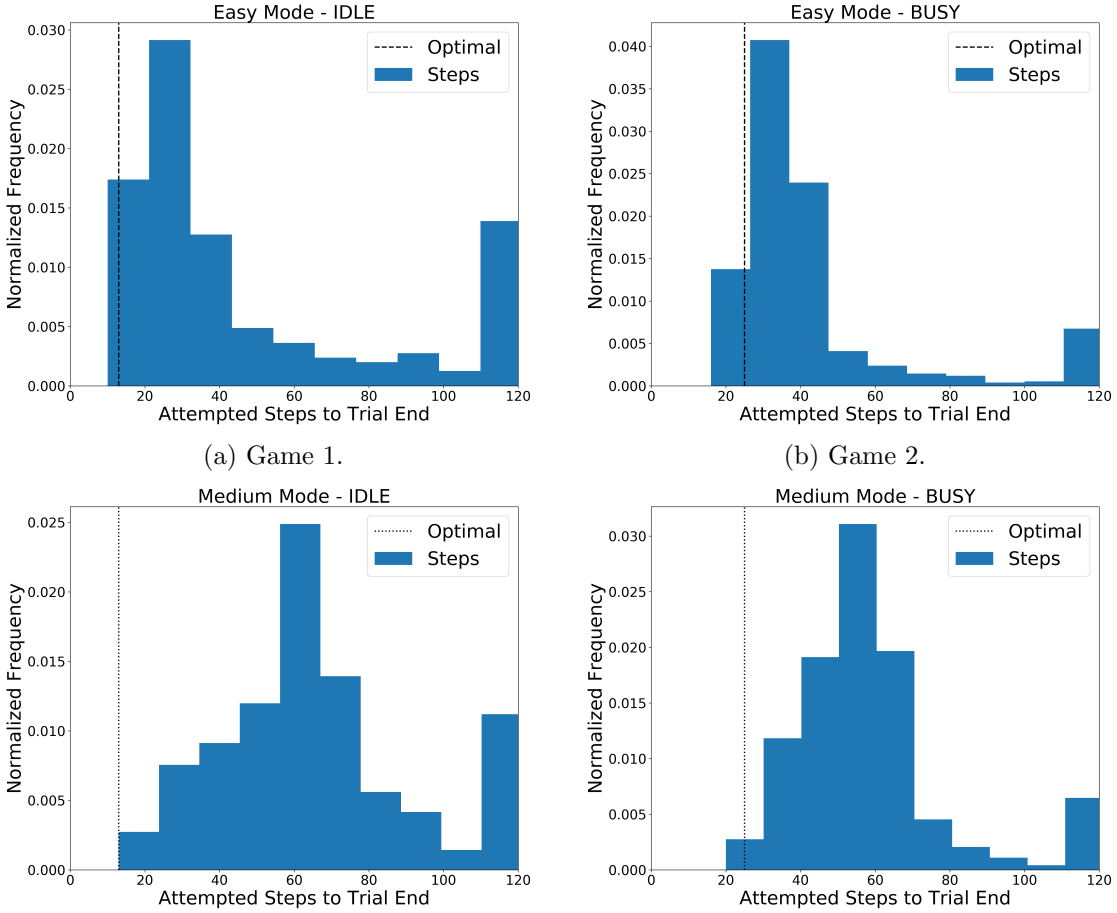
In relation to Figure 4.9 and Table 4.3, the human participant scores are clearly sub-optimal across all 4 games. By performing statistical analysis, we obtain insights into the human participant group's mean performance distribution as well as relative individual performance metrics in every game. For reference, the histogram analysis for all human participant performance, in all 4 games, is described in Figure 4.10.

4.7.3 Equipment Severity Rating w.r.t. Technician Skill Level

With the encouraging results from Table 4.3, the PPO-CNNLSTM model evaluations are also conducted on the MRS simulator with all technician skill levels being effected across the range of equipment severity ratings. mean-time-to-repair (MTTR) information can be derived from the rewards earned by each technician and, by normalizing the MTTR information, the probability of successful repairs can be obtained, see Figure 4.11a. Notably, these findings are reasonably consistent with our hypothesized risk-based state-transition relationship in Section 4.3.2.

The overall performance of PPO-CNNLSTM is empirically consistent between the senior and expert technicians, and its performance is characterised by considerations of the cost and availability of the manpower resources at any given time. For instance, once PPO-CNNLSTM identifies and dispatches a technician to repair the equipment, further resource substitution is disallowed, and the repair job must be completed by the dispatched technician, i.e., the DRL agent in this simulation case. Through iterative interaction with MRS, PPO-LSTM learns to dispatch the optimal skilled technician so as to maximize its overall reward received at all severity ratings. Accordingly, it is sub-optimal to select the expert level technician for the majority of severity ratings and its performance is justified by the severity level based technician selection probabilities in Figure 4.11b.

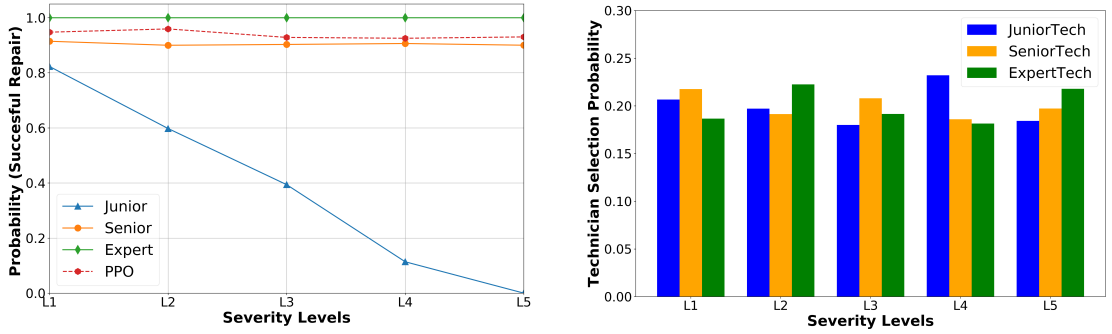
To summarize the first two experiments, the potential benefits of DRL augmented human decision-making for predictive maintenance action recommendation are clearly shown. Besides, PPO-CNNLSTM's improved learning efficiency results



(a) Game 1. (b) Game 2.

(c) Game 3. (d) Game 4.

FIGURE 4.10: Histogram analysis of human participant results (bins=10).



(a) Repair probability per severity level. (b) Normalized PPO action probability.

FIGURE 4.11: Evaluating the decision-making effects of technician selection w.r.t. severity levels.

are due to the use of multiple actors, which utilize modern edge computing resources, such as multi-core CPUs and GPUs, to realize a practical reduction in model training wall time when compared to similar DRL approaches.

4.7.4 C-MAPSS

When the original $N = 256$ learner hyperparameter is applied, poor performance convergence is observed because the multiple actors execute conflicting updates with our 256-step trace updates hyperparameter, causing PPO to behave like a monte carlo process [76]. Furthermore, by replacing the CNN module with two fully-connected layer with 64 neurons each, the standalone PPO model achieves comparable performance to [26] with the added benefit of learning efficiency improvement of 73% (i.e., reduction from 12×10^3 to 3.2×10^3 time-steps). Notably, the main hyperparameter for attaining learning convergence is to reduce the number of PPO actors to a single learner and environment.

4.8 Summary

In this chapter, we presented a deep reinforcement learning framework for an edge computing-based predictive maintenance model, to effectively manage the dynamic decision-making process involving equipment maintenance, maintenance cost model, and manpower resource. We formulated the complex resource management as a deep reinforcement learning problem for learning an optimal decision policy given a stochastic environment and time-series data. We evaluated the performance of the proposed PPO-LSTM using a maintenance repair simulator, and the findings are compared to those of human participants. The simulation results verify the efficacy of our framework and PPO-LSTM approach in addressing the challenging maintenance resource management problem, outperforming both human participants and the baselines in terms of convergence rate and performance. Chapter 5 extends this chapter's work, and focuses on enhancing the efficiency of reinforcement learning via transfer learning approach.

Chapter 5

Predictive Maintenance Model for IIoT-based Manufacturing: A Transferable Deep Reinforcement Learning Approach

5.1 Introduction

The Industrial Internet-of-Things (IIoT) is crucial for accurately assessing the state of complex equipment in order to perform predictive maintenance (PdM) successfully. However, existing IIoT-based PdM frameworks do not consider the influence of various practical yet complex system factors, such as the real-time production states, equipment health, and maintenance manpower resources.

In this chapter, we propose a generic PdM optimization framework to assist maintenance teams in prioritizing and resolving maintenance task conflicts under real-world manufacturing conditions. Specifically, we introduce the integration of TL and DRL model into the generic PdM-based resource management framework[1] to augment human intelligence.

[1]The work in this chapter has been presented in [18].

5.2 System Model and Joint Optimization Framework

5.2.1 System Model

Without loss of generality, we consider a generic IIoT-enabled manufacturing production line for edge-based predictive maintenance of critical machines. The following system components are to be part of the system model:

- **Edge Sensor Device (ESD):** An ESD denotes a sensing-computing device that continuously monitors and pre-processing time-series sensor data using an in-situ predictive model before outputting machine-specific metadata. Metadata, such as equipment health value, is a compressed numerical representation of multiple in-situ sensor data. Air pressure, water pump temperature, and motor rotational speed are some examples of in-situ machine sensors, see Figure 5.1.

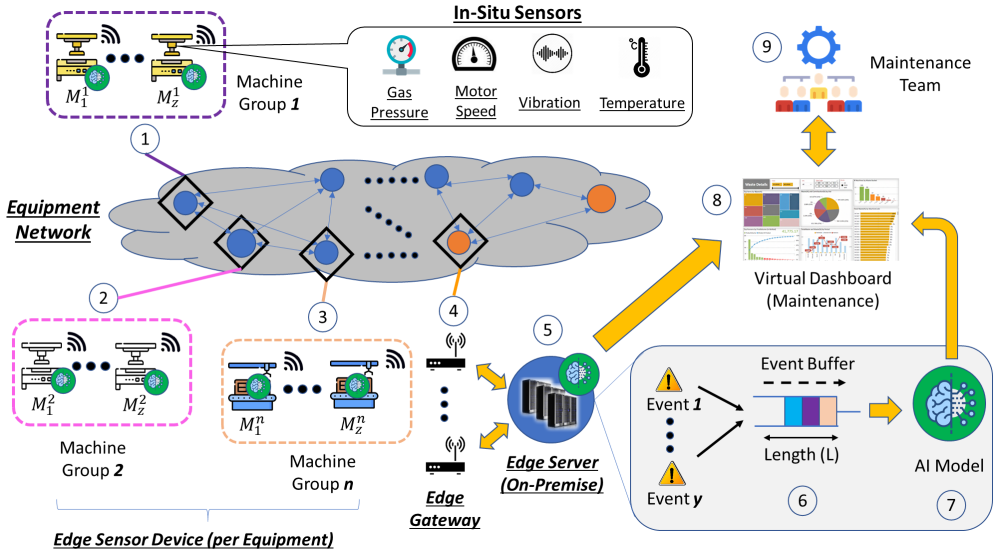


FIGURE 5.1: IIoT-enabled factory overview: (1) Generic machine with in-situ sensors; (2) Identical machine for similar manufacturing process; (3) n machine groups equipped with individual edge sensor device; (4) Edge gateway for data aggregation; (5) On-premise edge server for data analysis, data storage, model training, decision-making; (6) Temporary storage of maintenance request tasks; (7) AI model for maintenance decision-support; (8) Visual analytics and maintenance recommendation; (9) Maintenance team manpower resource information.

- **Edger Server (ES):** ES denotes an on-premise cluster of high-performance computing hardware capable of concurrently hosting multiple services such as web, data storage, analytics, and AI model training.
- **Edge Gateway (EG):** A computing device that resides between ESDs and the ES is referred to as an EG. The EG manages the ESDs linked to it and interacts with other EGs for machine-to-machine communication while reducing network congestion. In addition, EGs aggregate and encrypt data before transmitting it to the cloud or ES for further complex analysis.
- **Manpower Resource:** A team of technicians and engineers are responsible for the overall maintenance of the machines in the factory. An ES hosted database of maintenance manpower resources will capture the following information: technician skill-sets, experience levels, repair time and hourly capital expenditure of manpower.

Every machine is equipped with an ESD connected to the factory machine network and directly communicates with the on-premise ES via the EG using IIoT network protocols (see [132] for a survey of them). By design, ESD is resource-constraint, and its functionality is limited to in-situ-based sensor data aggregation and predictive model-based meta-data generation. Via the EGs, meta-data is transmitted to ES for storage, analysis, and AI model training as illustrated in Figure 5.1 (Step 1 to Step 5). If the ESD-based predictive maintenance model deems essential maintenance necessary, a maintenance task request is generated and temporarily aggregated at the ES-based maintenance buffer (Step 6). The AI model then analyzes the task state and schedules a qualified technician to fulfill each maintenance request (Step 7). Finally, the maintenance team will be notified via a virtual dashboard application (Step 8 and Step 9 in Figure 5.1).

5.2.2 Overview of the Joint Predictive Maintenance and Resource Management Framework

In the following, we propose a two-stage framework that pre-processes the maintenance tasks (Stage-1) before allocating qualified technicians to each maintenance task (Stage-2) in a time-slotted manner, see Figure 5.2. Specifically, the proposed JPdMRM Framework aims to address Step 6 to Step 9 in Figure 5.1 with details in

the next sections. Our two-stage framework enables modular fine-tuning of stage-based DRL models without impairing the overall operations and performance of the manufacturing system substantially. Besides, the module-based DRL architecture benefits from accelerated training performance (i.e., reduced model training wall-time), which complements the secondary objective of leveraging on TL method for knowledge transfer.

Machine Group: In the context of generic manufacturing systems, the machines have identical production capabilities but are segregated according to their respective manufacturing operations, with minor functional overlaps. Thus, these machines can be organized into Machine Groups (MGs) to increase production efficiency [133], where each MG is responsible for a sub-set of Parts Manufactured (PM). Table 5.1 provides an example of 3 MGs, each containing z machines, producing 5 different PMs. Different MG is dedicated to the production of a sub-set of the PMs (indicated by ✓). In what follows, we consider that one MG corresponds to an independent environment for strategy-learning agent deployment, since the degradation rate for the z machines across the MGs evolves following some independent stochastic process, leading to variation in maintenance tasks generation during PM production. The similarity between different MGs lays the foundation for our proposed TL model (cf. [39]). Meanwhile, the MG-based learning agent deployment helps to relieve the computation load compared to machine-specific AI models.

Maintenance Request Task Scheduling (Stage-1): Maintenance scheduling is critical when there are more maintenance tasks than personnel or resources available to accomplish them. For modeling purposes, we consider that y maintenance request tasks are randomly generated with an occurrence rate λ. Assuming the limited capacity of the maintenance team, we consider a task buffer I of L task

Part No.	$\boldsymbol{MG_z^1}$	$\boldsymbol{MG_z^2}$	$\boldsymbol{MG_z^3}$
PM-1	✓	✓	✓
PM-2	✓		✓
PM-3		✓	✓
PM-4	✓	✓	
PM-5			✓

TABLE 5.1: Illustration of the heterogeneous manufacturing capability across MGs.

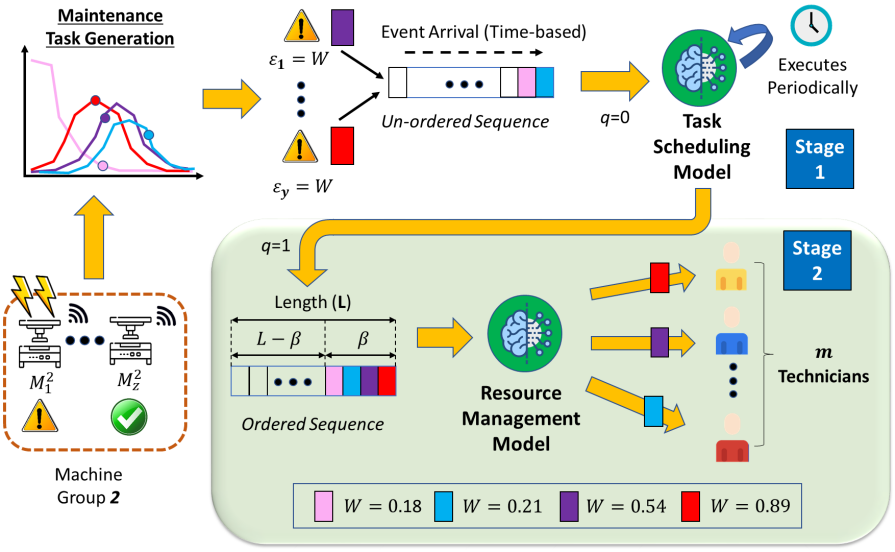


FIGURE 5.2: Proposed maintenance-based joint resource model decision pipeline. The task scheduling model turns random maintenance jobs into an ordered collection of work sequences. The resource management model then assigns the ordered job sequences to qualified technicians.

slots for temporary storage of maintenance tasks. It is worth mentioning that task arrivals in I are initially in accordance with the task generation time.

Traditional heuristics like first-in-first-out (FIFO) and last-in-first-out (LIFO) struggle to prioritize sporadic maintenance tasks while ensuring that only production-critical machine failures receive timely maintenance attention. Thus, a data-driven approach for production machines is needed to prioritize the maintenance tasks qualitatively and quantitatively. We propose to re-express a series of maintenance tasks as an unsorted sequence of tasks or machine tasks with stochastic priority distribution to overcome these challenges. We also introduce the Maintenance Priority Value (MPV) W to encompass both the production and machine-relevant data. In brief, the scheduling algorithm is expected to derive the optimal schedule to maximize the production uptime for z machines, B_t^z, via W values in the buffer I.

As illustrated in Figure 5.2, an unordered set of y tasks within the task buffer I will occupy $\beta = y$ task slots with $L - \beta$ vacant slots. Prior to allocating team members and negotiating with the production team, the maintenance team must prioritize machines based on various maintenance criteria. Otherwise, no action is performed for the remaining $L-\beta$ task slots. In this chapter, we consider the following maintenance indices of a machine: (a) severity rating ψ, (b) health-index H, (c) stoppage

time Λ and (d) operation status b. Due to the need of real-time scheduling (e.g., in several seconds) of time-critical maintenance tasks, which is beyond the capability of human decision-makers, we offer a DRL-based resource scheduling model for analyzing multi-dimensional data and provide maintenance-based recommendations to support decision-makers. With the proposed framework, we expect to help the manufacturing facilities significantly reduce the overall machine stoppage duration.

Manpower Management (Stage-2): Compared with [134], our proposed manpower resource management model is among the first to address multiple issues simultaneously by jointly considering the following system states on the technician team side: (a) the size m of a team, (b) the technician availability status G, and (c) the manpower cost c_η determined by the technician's competency level η. The proposed management model aim to appropriately assign a technician for machine attendance based on the ordered values of W from Stage-1 in order to maximize the repair success probability of the faulty machine, see Figure 5.2. The remaining $m-1$ technicians will be assigned to $0 < \beta \leq L$ tasks. Formally, we model the maintenance resource parameters as an optimization problem to maximize the overall production uptime (i.e., minimize machine stoppage duration for all machines in production line), ρ_t, through optimal resource allocation and explain the details in Section 5.3.

5.3 Problem Formulation

We formulate the scheduling problem into a sequential decision-making process by introducing the Markov Decision Process (MDP)-based model [76], with the aim of optimizing the overall uptime of the machines in an MG. We note that a generic MDP can be expressed as a 4-tuple [76], i.e., $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, T(s'|s,a), \{R(s,a)\} \rangle$, where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ is the action space, $T(\cdot)$ is the state transition probability map, and $R(\cdot)$ is the immediate reward function with $s \in \mathcal{S}$ and $a \in \mathcal{A}$.

Before formulating the MDP that reflects the task-resource dynamics w.r.t. predictive maintenance-based parameters, we first describe a snapshot of the concerned system as an optimization problem of two stages: 1) maintenance request task scheduling, and 2) manpower resource management. By doing so, we are able to offer in the proposed JPdMRM framework (cf. Figure 5.2) flexible representation of the environment information as well as tractability of the solution for

PdM-based resource management. In what follows, the inter-model interactions are briefly described, and the MDP formulation is presented in the framework of Bellman equations before motivating our proposed DRL approach and TL in the later sections. For the reader's convenience, we summarize the necessary notations of this chapter in Table 5.2.

Consider a machine network of N MGs, $\{M_z^n\}_{n=1,\ldots,N}$, with each MG containing Z machines. Let $B_t^{z,n}$ denote the production uptime for machine z ($z \in \{1,\ldots,Z\}$) in MG n. For ease of exposition, we first consider the scenario of one MG and ignore the index n. For a decision time step t, the goal of scheduling is to maximize the overall production uptime (denoted by ρ_t):

$$\max \rho_t = \sum_{z=1}^{Z} B_t^z. \tag{5.1}$$

In the following sub-sections, we define the stage-based formulations that influence the uptime of machine z, B_t^z.

5.3.1 Maintenance Request Task Scheduling

This sub-section describes the stage-1 scheduling algorithm that aims to maximize the overall run-time of z machines. In particular, the scheduler learns a decision strategy to re-arrange a small array of maintenance request tasks according to their priority level given any dynamic sequence of maintenance request tasks. Notably, the learned knowledge is expandable to sort an array of tasks of arbitrary length. We shall now define the MDP model details.

Machine State ($s_t^{\mathbf{E}} \in S^{\mathbf{E}}$): When a machine stops abruptly, a *sporadic* maintenance request task, denoted as ϵ_w, is generated and is characterized as (see also Table 5.2):

$$\epsilon_{w,t} = [z, \psi_t, \nu_t, \chi_t, \Lambda_t, b_t], \tag{5.2}$$

where w denotes the task ID, $z \in \{1,\ldots,Z\}$ is the unique machine ID which is a time-invariant variable, and $\psi_t \in [0,1]$ is given as:

$$\psi_t = g + e^{at^f}, \tag{5.3}$$

[2]Note that the model parameter can be straightforwardly extended for any number.

Table 5.2: List of Important Notations.

Symbol	Description
$\rho_t; B_t^z$	Overall Production uptime at decision t time-step; Production uptime for machine z
$z; n$	Number of machines, $z \in \{1, \ldots, Z\}$; Number of machine group (MG), $n \in \{1, \ldots, N\}$
$m; G; g_t^m$	Number of technicians, $m \in \{1, \ldots, M\}$; Total availability of m technicians; Availability status of m^{th} technician at t time-step
$H_t; b$	(**Machine Operation Variables**) Normalized machine health value at t time-step; Machine fault & operation state, $b \in \{1, 2, 3, 4\}$ [2]
$I; \epsilon_w$	Task buffer of maintenance request tasks ϵ_w; Sporadic maintenance request task with task-id w, $w \in \{1, \ldots, L\}$
$y; L; \beta$	Number of maintenance request tasks generated; Maintenance request task buffer length; Number of maintenance request tasks in Γ
ψ; ψ_t^δ	Normalized machine severity rating; Scalar value with δ rating at time-step t
$W; \nu;$ $\chi; \Lambda$	(**MPV and related variables**) Maintenance priority value (MPV); Potential component revenue; Residual days to delivery deadline; Machine stoppage duration (i.e., since $b = 4$)
q	Task scheduling status, $q \in \{0, 1, 2\}$ [2]
x	Maintenance urgency category, $x \in \{1, 2, 3\}$ [2]
ξ	Mean-time-to-repair lookup table
c_η^d	Manpower cost associated with η competency options for task index d
τ	Maintenance budget
μ	(**Time Constraint**) Task scheduling decision, $\mu \in \{1, \ldots, \Omega^E\}$ Resource management, $\mu \in \{1, \ldots, \Omega^M\}$
j; η; h_w; κ	(**Actions**) Evaluate Task-Set, $j \in \{0, 1\}$; Technician Competency Level, $\eta \in \{1, 2, 3\}$; Insert current ϵ_w task to index w for insertion, where $h_w \in h$; Idle

where g is the parameter to determine the initial non-zero degradation condition, a and f are generalized wear-rates that correspond to the effects of temperature and relevant sub-system stress terms with respect to time t[6]. Correspondingly, let $H_t = 1 - g - e^{atf}$ represent the health-based meta-data generated from in-situ predictive maintenance model (see also Section 5.2.2). In return, $\psi_t = g + e^{atf}$ in (5.3) is the complementary of H_t [24]. In practice, H_t is a second-order derivative and can neither be obtained nor validated in the absence of the ground truth. For modeling purposes, we consider the equipment's performance (i.e., H_t) degrades exponentially with recurrent equipment usage and complex environmental conditions [6]. In addition, we assume the equipment PdM model is noise-invariant and outputs highly accurate health-based meta-data values of H_t. Consequently, the accuracy of ψ_t is likewise preserved.

Based on the in-situ controller fault codes within modern production machinery, we can identify the operational condition of the machine (denoted as b) as one of the following states: (a) idle ($b = 1$), (b) in production ($b = 2$), (c) under repair ($b = 3$) and request maintenance attention ($b = 4$). Note that the model can be conveniently extended for any number of codes. When an unexpected machine stoppage occurs, the state is set as $b=4$. We use parameter Λ to indicate the minute-based machine stoppage duration and fractionally express it in unit of hours. Practically, the maintenance request at $b=4$ is set using the in-situ ESD-based PdM model[24], which detects intermittent sensor data spikes and predicts imminent machine stoppage. To distinguish a predicted stoppage from the actual stoppage, we fix $\Lambda=0.01$ for PdM-predicted (virtual) stoppage and assume $\Lambda>0.01$ for actual machine stoppage time duration, respectively. Thus, we alleviate the cognitive burden of the maintenance-based decision-maker by using Λ as a weighted factor to prioritize actual machine stoppage over prediction-based outputs. Then, the maintenance team or resource management scheduler are able to utilize this distinction to prioritize maintenance requests dynamically. For operational states such that $b \neq 4$, $\Lambda=0$.

We define the production urgency index as the ratio of the potential component revenue manufactured, denoted by v, to the residual days before the delivery deadline, denoted by χ. Notably, χ is a hard constraint that can be calculated by subtracting the machine stop time from the component delivery date, expressed in days. The delivery date information is accessible from a production planning

database system. Then, we can incorporate this ratio with predictive maintenance state parameters Λ and ψ into the Maintenance Priority Value (MPV) function in (5.4):

$$W = \begin{cases} \frac{\nu}{\chi}\psi\Lambda, & \text{if } b = 4, \\ 0, & \text{Otherwise}, \end{cases} \tag{5.4}$$

with which the maintenance task ϵ_w in (5.2) can be re-expressed as $\epsilon_w = [z, W]$. We restrict $W \in [0, 1]$ for formulation into our task request scheduling problem.

Recall that a task buffer I is a group of y maintenance request tasks that require maintenance attention, where the y tasks are generated by Z machines. The task buffer I of L task slots is formally defined as $I = \{\epsilon_{w=1}, \ldots, \epsilon_{w=L}\}$. We denote β ($\beta \leq y$) as the number of loaded task slots requiring maintenance attention, and the remaining $L - \beta$ slots are free. The task buffer can be interpreted as a task array of length L, with each array element containing the value for a maintenance task ϵ_w. The content of ϵ_w in each of the β task slots is based on (5.4) with $W_w > 0$ in β task slots, and the task buffer ID is denoted as $w \in \{1, \ldots, L\}$. Unlike [24], deadlocks induced by identical values of ψ are now mitigated via the merging of unique machine-ID information with production-relevant information, contained in each ϵ_w value.

Within the task buffer I, the initial order of each task element in β is aligned with task generation time rather than W values. To guarantee task synchronization within our dual-stage JPdMRM framework (see Figure 5.2), we define the task scheduling state of buffer I using $q \in \{0, 1, 2\}$. $q = 0$ indicates a standby (i.e., default) state, $q = 1$ as scheduling in-progress and $q = 2$ as scheduling completion. Formally, we can represent the overall task scheduling model state space as: $s_t^{\mathrm{E}} = \{q, I\} \in \mathcal{S}^{\mathrm{E}}$, with $\mathcal{S}^{\mathrm{E}} \triangleq \mathcal{Q} \times \mathcal{I}$, $q \in \mathcal{Q}$ and $I \in \mathcal{I}$.

Action Sub-Space ($a_t^{\mathbf{E}} \in \mathcal{A}^{\mathbf{E}}$): The task scheduler takes proper actions to maximize the received rewards by periodically sorting W values within μ time-period, as shown in Figure 5.2. Let $\mu \in \{0, 1, \ldots, \Omega^{\mathrm{E}}\}$ denote the real-world decision-making time constraints, where the maximum allowable time, Ω^{E}, is user-defined. Given some random initial state (i.e., $S_{t=0}^{I} = I_0$), the agent interacts with the environment by choosing the following actions: idle (κ), task index manipulation (h) and evaluate task-set ($j \in \{0, 1\}$). The set of actions h is similar to array-based manipulation operations used in generic sorting applications, such as insertion sort.

Consider the example of $S^I = [0.1, 0.5, 0.3, 0.9, 0.0]$ and $\beta = 4$. S^I elements (i.e, ϵ_w) should be arranged in descending order of priority, with $S^{I_{w=1}}$ denoting the highest priority task. At $t = 1$, $S^I_{t=1} = 0.5$ and selecting $h_{w=1}$ inserts 0.5 into task buffer index $w = 1$, where $1 \leq w \leq \beta$. Upon insertion into $w = 1$, the existing value is automatically shifted to $w = 2$, and $S^I = [0.5, 0.1, 0.3, 0.9, 0.0]$ is observed.

Note that the constraints are necessary to ensure that the selected action at t (i.e., a^{E}_t) do not violate the task buffer size L while selecting appropriate actions from h for β task slots. Thus, the action space of the task scheduler is formalized in (5.5), where $a^{\mathrm{E}}_t \in \mathcal{A}^{\mathrm{E}}$.

$$a^{\mathrm{E}}_t = \begin{cases} h_w \in h, & \text{if } (1 \leq w \leq \beta) \text{ and } (t < \Omega^{\mathrm{E}}), \\ \kappa, & \text{if } t < \Omega^{\mathrm{E}}, \\ j, & \text{Otherwise.} \end{cases} \tag{5.5}$$

Machine State Transition Map ($T^{\mathbf{E}}$): The state transition probabilities depict the environment dynamics, which are unknown to the scheduler. Assume that y tasks are to be generated within L task slots in such a way that the inter-arrival of tasks are generated independently following some unknown stationary stochastic process. Considering exponential machine health degradation behavior, maintenance-based tasks are assumed to follow a Poisson distribution behavior $P(y) = \frac{e^{-\lambda}\lambda^y}{y!}$ [135], where λ denotes the occurrence rate of y tasks at each task slot or index. Note that task buffer I is infinitely large and does not have to equal to L. In other words, anytime $y > L$ events occur, the scheduling agent considers task scheduling only up to L task slots in I, with $y - L$ tasks queued for future scheduling operations.

Let us consider $s^I_{t=0}$ to be initially set to zero, with κ (idle) chosen as the default action and no state-transition occurs. The number of tasks to be attended, β, increases as one task is generated for every observed machine state instance $b=4$. At this point, the task scheduler will decide a batch of actions to perform for each element task ϵ_w of $\mathcal{I}$ (i.e., s^{E}_t), such that $a^{\mathrm{E}}_t \neq j$ and the task re-ordering state transitions are observed as $(s^{\mathrm{E}})' = \phi(s^{\mathrm{E}})$. So, we let $a^{\mathrm{E}}_t|_h : I \rightarrow I'$, and the scheduler state is updated as $q=1$ in (5.6). To conclude the completion of task re-ordering, the scheduling agent will invoke $a^{\mathrm{E}}_t|_{j=1}$, and thereby indicate that the random task buffer I is transformed to an ordered task buffer I', which leads to the following

change in s_t^{E} by y_{j} times, as $y_{\mathrm{h}} \in \phi_h(s^{\mathrm{E}})$. Likewise, scheduling state is changed to completed (i.e., $q = 2$) as in (5.6). Formally, the described transition behavior is modeled as $T_{\mathrm{E}} = \Pr(s_t'^{\epsilon} \mid s_t^{\mathrm{E}}, a_t^{\mathrm{E}})$.

$$y_{\mathrm{q}} \in \phi_q(s^{\mathrm{E}}) = \begin{cases} q = 1, & \text{if } \; y_{\mathrm{h}} \in \phi_h(s^{\mathrm{E}}), a_t^{\mathrm{E}} \neq j, \; 1 \leq w \leq \beta, \\ q = 2, & \text{if } \; y_{\mathrm{j}} \in \phi_j(s^{\mathrm{E}}), \; a_t^{\mathrm{E}}|_{j=1}, \\ q = 0, & \text{Otherwise.} \end{cases} \tag{5.6}$$

Reward Function ($r^{\mathbf{E}}$): At each time-step t, the agent selects a_t^{E} according to (5.5), and the scheduling agent receives a reward signal r_t^{E}. Prior to choosing $a_t^{\mathrm{E}} = j$, the agent will execute a series of actions without knowing if the suggested task sequences are properly ordered until action $a_t^{\mathrm{E}}|_{j=1}$ is invoked. Subsequently, the agent learns the reward function iteratively. We arbitrarily set a modest penalty score of -0.001 to intrinsically motivate the agent without impairing the overall state-space exploration progress for reward shaping purposes. Furthermore, we incorporate a time-varying incentive r^q as a supplemental reward for r^I to emulate the urgency associated with real-world decision-making. For example, suppose we let decision time-constraint $\Omega^{\mathrm{E}} = 20$ time-steps, and the scheduler invokes $a_t^{\mathrm{E}}|_{j=1}$ at $t = 18$ time-step to signal sorting completion. Simultaneously, a sorting score of $r^I = 10$ is computed. The sum of r^q and r^I yields a scheduling reward score of 12, as defined in (5.7). To maximize reward, the scheduler must master performing the NP-hard combinatorial task sorting operations correctly and efficiently. For instance, consider a set of three random tasks arriving at task buffer I with the following W values and buffer index sequence: $[0.18|_{w=1}, 0.89|_{w=2}, 0.54|_{w=3}]$. Given the FIFO-based task buffer and unsorted W values, a trained scheduler will automatically sort the W values in descending order using the available actions in a_t^E in order to maximize the reward for the following combination: $[0.89|_{w=1}, 0.54|_{w=2}, 0.18|_{w=3}]$. Note that r_t^{E} at time-step t and the reward function is described in (5.7) alongside reward variables in (5.8) and (5.9).

$$r_t^{\mathrm{E}}(s_t^{\mathrm{E}}, a_t^{\mathrm{E}}) = \begin{cases} r^q + r^I, & \text{if } \; s_t^{\mathrm{E}} \in \beta, \; a_t^{\mathrm{E}}|_{j=1}, \\ -0.001, & \text{Otherwise,} \end{cases} \tag{5.7}$$

where

$$r^I = \sum_{w=1}^{\beta} \frac{w W_w}{\beta}, \quad W_w \in \mathcal{I}, w \in \{1, \ldots, \beta - 1, \beta\}\,; \tag{5.8}$$

$$r^q = \Omega^{\mathrm{E}} - t; \tag{5.9}$$

5.3.2 Manpower Resource Management

According to Figure 5.2, the results of ordering, I, from $S^{\mathrm{E}}|_{q=2}$ serves as an input to Stage-2 of the JPdMRM framework. The objective of Stage-2 is to maximize the production uptime ρ_t in (5.1) for z machine in n machine group by optimally assigning the manpower to ordered task buffer, as in (5.10). The specifics are elaborated in this sub-section for the reader's benefit.

$$\begin{aligned} \max_{a_t^{\mathrm{M}} \in \mathcal{A}^{\mathrm{M}}} \rho_t &= \sum_{z=1}^{Z} B_t^z(a_t^{\mathrm{M}} | \epsilon_w, c_\eta^d, a_t^{\mathrm{E}}) \\ \text{s.t.} \quad &(a)\ \epsilon_w \in S^{\mathrm{E}}, W > 0, \\ &(b)\ \tau - \sum_{d=1}^{\beta} c_\eta^d > 0. \end{aligned} \tag{5.10}$$

In this sub-section, we let $m \in \{1, \ldots, M\}$ technicians be grouped according to competency levels as $\eta \in \{1, 2, 3\}$, where $\eta = 1$ denotes junior (e.g., *0 to 5 years*), $\eta = 2$ denotes senior (e.g., *6 to 15 years*), and $\eta = 3$ for expert (e.g., $\geq$*15 years*) respectively. Note that the model is general and applicable to arbitrary number of competency levels.

State Space ($S^{\mathbf{M}}$): The resource management state space S^{M} is obtained by extending S^{E} the the following binary variable indicating the *Technician Availability*: g^m. More specifically, let the technician availability state be defined as Available ($g^m = 1$) and Busy ($g^m = 0$), where $m \in \{1, \ldots, M\}$ is used to uniquely represent the technician's identity. Therefore, the total availability status (G) of M technicians[3] at time-step t is denoted as $G = \sum_{m=1}^{M} g_t^m$. Intuitively, when M technicians are identified with their competency levels, we obtain a static array of technicians $\{\eta_1, \ldots, \eta_i\}$ (assume that η_i is used to indicate the level of technician i). Then, we are constrained by the availability of technicians, which is related to (5.11).

$$G_t = \sum_{\eta=1}^{3} \sum_{m=1}^{M} g_t^{(m,\eta)}. \tag{5.11}$$

[3]When the competence level of technicians is also considered, information about a technician's availability becomes necessary to guide the resource scheduler's decision-making process.

Initially, $G = M$ when $\beta = 0$, the machines are deemed critical to the production team, necessitating dedicated manpower resources to assure maximum production uptime. Therefore, an allocated technician cannot be re-assigned to another task irrespective of their competency level η at the same time. We also assume that G is fully observable. Thus, only when all β tasks have been allocated, with G technicians, the resource allocation problem will be considered solved, and previously assigned technicians become available immediately after the task is completed. Recall from Stage-1 of the JPdMRM framework, task buffer $I \in S^{\mathrm{E}}$ values are sorted in descending orders of W (i.e., ordered sequence of maintenance request tasks) whenever $q = 2$. Given the fully-observable output of $I \in S^{\mathrm{E}}$ is an input into S^{M}, the association can be mathematically expressed as $\mathcal{Q} \in S^{\mathrm{E}}; (q = 2) \implies S^{\mathrm{E}} \subset S^{\mathrm{M}}$ without loss of generality. Therefore, we can compactly define the resource management model state space as $S^{\mathrm{M}} \triangleq \{S^{\mathrm{E}}, G\}$, and system state at t time step as $s_t^{\mathrm{M}} \in S^{\mathrm{M}}$.

Action Space ($a_t^{\mathbf{M}} \in \mathcal{A}^{\mathbf{M}}$): At t^{th} time-step, the resource scheduler requires to allocate a sufficiently competent technician, from η, based on the current value of s_t^{M}. In addition, there is a manpower cost associated with corresponding choices of η (i.e., denoted by c_η), and τ regulates the selection frequency of η to simulate real-world decision-making for resource allocation. When no technician is assigned, *Idle* remains the default action and incurs zero cost. Here, we let $\mu \in \{0, 1, \ldots, \Omega^{\mathrm{M}}\}$ denote the real-world decision-making time constraints, where the maximum allowable time, Ω^{M}, is user-defined. Note that $\Omega^{\mathrm{E}} \neq \Omega^{\mathrm{M}}$. The instantaneous action can be compactly expressed with technician selection action η in (5.12).

$$a_t^{\mathrm{M}} \triangleq \left[(\kappa, \eta) |\ t \leq \Omega^{\mathrm{M}} \text{ and } \tau - \sum_{d=1}^{\beta} c_\eta^d > 0 \right]. \tag{5.12}$$

State Transition Probabilities ($T^{\mathbf{M}}$): In this work, we only consider one technician from M technicians be allocated to each task subject to maintenance budget constraint $\tau > 0$ in (5.12). In addition, no state-transition will occur when either $q \neq 2$ and $\beta = 0$ is detected. Considering $q = 2$ and $\beta > 0$, the task buffer's index state increments by d_η from S_I^{M} to $S_{I'}^{\mathrm{M}}$ with a certain probability (i.e., $T^{\mathrm{M}} = P(s_t'^{\mathrm{M}} | s_t^{\mathrm{M}}, a_t^{\mathrm{M}})$) as a_t^{M} is invoked at every time-step. We denote the overall state-transitional change as $\phi(S^{\mathrm{M}})$ in (5.13), and assume the combinations of the two cases ($M > \beta$) and ($\beta > M$) to mimic real-world combinatorial conditions

for both tasks and technicians.

$$d_\eta \in \phi(S_{\mathrm{M}}),\ \text{if}\ G>0,\ a_t^{\mathrm{M}} \neq \kappa. \tag{5.13}$$

The resource allocation is done when either $(G=0)\in S_{\mathrm{M}}$ or β slots are completely allocated. Yet, the resource management agent is unaware of the environment's state transition probabilities and the stochastic distribution of $\epsilon_w \in S_{\mathrm{M}}$ tasks (i.e., black box). Thus, based on the MDP framework, the agent learns the state transition probabilities and task distribution information via continuous reward signal feedback.

Reward Function ($r^{\mathbf{M}}$): We assign $W \in S_{\mathrm{M}}$ values into $x \in \{1,2,3\}$ priority categories for ease of analysis, with conditional constraints defined in (5.15). For example, $x=1$ denotes *Urgent*, $x=2$ denotes *High*, and $x=3$ denotes *Medium* priority respectively. Both x and η are transformed into a mean-time-to-repair lookup table $\xi \in \xi_{(s_t^{\mathrm{M}},a_t^{\mathrm{M}})}$, and are multiplied against Ω^m time-steps to obtain a time-based reward, refer to (5.14) for details. For instance, the reward received from selecting $\eta=3$ is always greater than the reward received from selecting $\eta=1$ when $x=1$, and the action selection frequency is cost constraint. Since the ξ values are initially unknown, the agent must therefore learn from the reward feedback signal (r_t^{M}) and cumulatively construct the reward function $r^{\mathrm{M}}(s_t^{\mathrm{M}}, a_t^{\mathrm{M}}, s_{t+1}^{\mathrm{M}})$, before determining an optimal resource management decision-policy. Note that a negative reward is defined to stimulate the agent for the goal of state-space exploration, and the reward function is described in (5.14),

$$r_t^{\mathrm{M}}(s_t^{\mathrm{M}}, a_t^{\mathrm{M}}) = \begin{cases} r_\eta, & (1-\xi_{(s_t^{\mathrm{M}},a_t^{\mathrm{M}})})\Omega^{\mathrm{M}}; \\ & \left\{s_t^{\mathrm{M}}=x;\ a_t^{\mathrm{M}}=\eta\right\}, \\ -0.05, & \text{Otherwise}, \end{cases} \tag{5.14}$$

where ξ is obtained based on the following state abstraction

$$x = \begin{cases} \text{Urgent}, & \text{if}\ W \in [0.01+2/3, 1.0], \\ \text{High}, & \text{if}\ W \in [0.01+1/3, 2/3], \\ \text{Medium}, & \text{if}\ W \in [0, 1/3]. \end{cases} \tag{5.15}$$

5.3.3 Overall optimization Formulation

The overall optimization objective is to maximize the accumulated production uptime ρ_t over a time horizon by maximizing the overall machine network production uptime via an automated decision-making process of predictive maintenance (PdM) request task scheduling and manpower resource allocation. With this in mind, a dynamic optimization approach outperforms myopic one-shot optimization methods when dealing with a black box environment model. Namely, the current decision will influence future decisions for the remaining task sequences in the task buffer or the number of available technicians in the maintenance team. For ease of explanation, we propose the following integration of the sub-objectives in Section 5.3.1 and 5.3.2 into the generic MDP Framework tuple as follows: $\mathcal{M} = \langle \mathcal{S} = S^{\mathrm{E}} \times S^{\mathrm{M}}, \mathcal{A} = A^{\mathrm{E}} \times A^{\mathrm{M}}, T = (T^{\mathrm{E}}, T^{\mathrm{M}}), R = (r^{\mathrm{E}}, r^{\mathrm{M}}) \rangle$.

At the end of each iteration, the stage-based schedulers select action a_t based on the current environment state s_t and receives new values of r_t and s' from the environment. Policy π guides the scheduler's action to maximize the cumulative average rewards received over all states. Thus, after several iterations, the scheduler eventually finds an optimal policy (π^*) to maximize the expected long-term discounted reward r of the PdM environment problem (i.e., cumulative production uptime).

Recall, the disparate stage-based reward functions of r^{E} and r^{M} must work in tandem to cumulatively maximize ρ_t in relation to the given constraints. Given that our research problem is essentially one of scheduling machine and manpower resources in order to maximize the equipment runtime (i.e., $\sum_{z=1}^{Z} B_t^z$), we define $\rho_t = r^{\mathrm{E}} + r^{\mathrm{M}}$. However, the problem may be too complicated to maximize both sub-objectives concurrently while achieving solution tractability within reasonable training time. By exploiting the flexibility of our two-stage JPdMRM framework, the complex optimization problem can also be deconstructed such that stage-based models may be trained independently to maximize the sub-objective reward with respect to r^{E} and r^{M}.

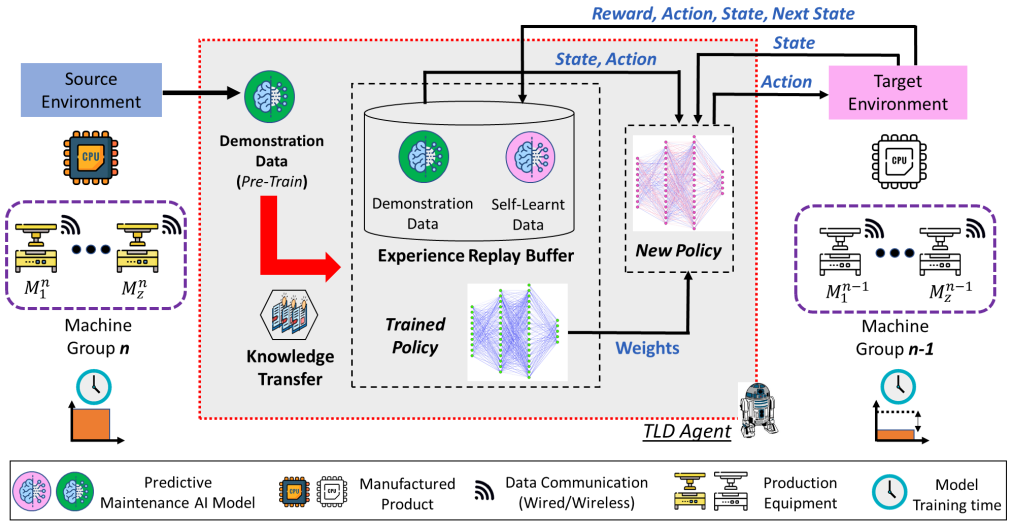


FIGURE 5.3: Transfer learning with demonstrations approach which facilitates knowledge transfer efficiently between environments with machines producing similar parts with either identical or similar machinery.

Recall, policy π guides the scheduler's action to maximize the cumulative production uptime. Therefore, we denote the optimization problem as follows:

$$\begin{aligned} &\max_{\pi} \sum_{t=0}^{\infty} \rho_t = \max_{\pi} R(\pi), \\ &\text{s.t.} \quad R(\pi) = \mathbb{E}\left\{ \sum_{t=0}^{\infty} \gamma r_t(\pi) \,|s_t = S \right\}, \end{aligned} \tag{5.16}$$

where $R(\pi)$ denotes the average expected return under the policy π; $r_t = \rho_t$ refers to the immediate reward under policy π; $t \to \infty$ denotes the time horizon, and discount factor $\gamma \in [0, 1]$. With the optimal policy π^*, the agent will select the optimal actions, at any state S. In other words, the agent will have learnt π^*, and is able to maximize the cumulative production uptime for N machines (i.e., reward) via appropriate manpower resource allocation (i.e., action) to attend to priority scheduled maintenance tasks (i.e., state). Practically, environment uncertainty often leads to unknown T, and a model-free reinforcement learning approach can be applied to learn π^* in (5.16).

5.3.4 Problem Transformation

The Q-learning (QL) algorithm [136] obtains the optimal decision policy $\pi^* : S \to A$ by using a Q-table to recursively map state $s_t \in S$ to action $a_t \in A$. The optimal policy is learned by maximizing the state-action pair value function $Q^{\pi}(S, A)$ over

all Q-value policies in (5.17). Consequently, the state-transitions (s_t, a_t, r_t, s_{t+1}) is obtained as the Q-values are recursively updated using Temporal Difference (TD) learning [76].

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}[r_{t+1} + \gamma Q^{\pi}(s_{t+1}, a_{t+1})|s_t = s, a_t = a]. \tag{5.17}$$

Once the Q-table has been updated, the optimal policy for the agent is determined by selecting the largest state-action value within Q-table: $\pi^*(s) = \arg\max_{a \in A} Q^*(s,a)$. However, the dimensionality curse of Q-learning [76] restricts its applicability to real-world problems, which often involve large state space and requires rapid solution convergence. In other words, as the state and action spaces expand, the size of the Q-table grows quadratically, increasing the time needed to update the whole Q-table and, therefore, the time required to achieve an optimum policy. In the case of our dual-stage JPdMRM framework, the overall size of the Q-table will be $2LM$ and will grow quadratically with increasing numbers of L to accommodate more machine maintenance requests. Therefore, [137] utilizes Neural Network (NN) to approximate Q-values, also known as Deep Q-Network (DQN), and effectively overcome the dimensionality limitation. The input and outputs of the NN are designed to be compatible with the JPdMRM framework and predictive maintenance-based resource scheduling problem.

In reality, dynamic operating conditions will alter the state-transition probabilities and data distribution for identical machines (i.e., environment). Moreover, dynamic conditions such as machine utilization, ambient temperature, machine health degradation rates, and the maintenance intervals for individual machines necessitate different environments. In this context, applying the conventional DQN algorithm to a comparable MDP and maintenance resource scheduling problem can be inefficient.

A more efficient approach is to transfer the knowledge-based information through external demonstrations from different sources and exploit the acquired knowledge to achieve accelerated DQN learning. Besides, the concept of transfer learning is largely under-explored for predictive maintenance-based resource scheduling applications. Therefore, we evaluate the advantages and limitations of transfer learning using our JPdMRM framework and maintenance scenarios.

5.4 Transferable DRL for Joint Predictive Maintenance and Resource Management

5.4.1 Motivation

In Section 5.3, we briefly described how a deep reinforcement learning (DRL) algorithm, such as DQN, derives an optimal policy from any environment. Based on the aforementioned practical considerations, we are motivated to propose the transfer learning approach for DRL, an under-explored method for accelerating solution convergence in the similar environments (i.e., *target* environment). Figure 5.3 illustrates how the trained NN policy (i.e., knowledge) from one group of machines (i.e., *source* environment) is transferable to a similar group of machines (i.e., *target* environment) with the benefit of reduced model training time. In comparison to random initialization, model pre-training provides a more optimum starting point for training weights to the NN while adding a step to the overall training process. Furthermore, since the starting point is close to optimal, we decrease the likelihood of divergent learning, resulting in a more stable model. In this section, we present details for the proposed transfer learning with demonstrations algorithm.

5.4.2 Transfer Learning Overview and Approach

In the context of transfer learning, the state-transition probability is the only difference between the *source* $\boldsymbol{\mathcal{M}_s}$ and *target* $\boldsymbol{\mathcal{M}_t}$ environments. Thus, their MDPs can be expressed as $\boldsymbol{\mathcal{M}_s} =< S, A, T_s, R >$ and $\boldsymbol{\mathcal{M}_t} =< S, A, T_t, R >$ respectively. With reference to our JPdMRM framework in Figure 5.3, we formalize Transfer Learning (TL) in the following:

Definition 5.1 (**Transfer Learning in the Context of Predictive Machine Maintenance and Resource Scheduling**). Consider machine groups n and $n-1$ as **source** environment $\boldsymbol{\mathcal{M}_s}$ and **target** environment $\boldsymbol{\mathcal{M}_t}$ respectively. TL aims to learn an optimal policy π_t^* within the $\boldsymbol{\mathcal{M}_t}$ environment, via prior knowledge exploitation of $\mathcal{D}_s$ from $\boldsymbol{\mathcal{M}_s}$ in order to augment its current knowledge $\mathcal{D}_t$ in $\boldsymbol{\mathcal{M}_t}$, such that:

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{(s\sim s_t, a\sim\pi)}[Q^{\pi}_{\mathcal{M}}(s,a)] \tag{5.18}$$

where $\pi = \phi(\mathcal{D}_s \sim \boldsymbol{\mathcal{M}_s}, \mathcal{D}_t \sim \boldsymbol{\mathcal{M}_t})$ such that π denotes the function mapping of the *target* environment's states to actions $S \rightarrow A$. Then, the neural network

learns the approximated representation of policy π based on values of $\mathcal{D}_s$ and $\mathcal{D}_t$ [39]. Otherwise, policy $\pi = \phi(\mathcal{D}_t)$ is equivalent to retraining the DRL algorithm from an initial random state within the *target* environment. Note that the slight differences between both environments imply that distinct optimal policies require to be learned.

While various TL techniques have been presented to tackle (5.18), e.g., [39], their efficacy and relevance to the joint maintenance scheduling problem merits need further investigation. We specifically focus on the transfer learning with demonstrations (TLD) technique [138] in order to reduce the overall training wall-time required, which has not been done before and it is the contribution of this work. TLD's underlying idea is to pre-train the neural network model with an initial demonstration data $\mathcal{D}_s$ until an acceptable level of performance in $\mathcal{M}_s$ is attained. The trained neural network and the accumulated experiences are then transferred to $\mathcal{M}_t$ environment, with the corresponding experiences saved as $\mathcal{D}_t$. Notably, both $\mathcal{D}_s$ and $\mathcal{D}_t$ are namely tuples of S, A, R, and S' information, respectively.

5.4.3 Solution of Transfer Learning with Demonstrations (TLD)

During initialization, the experience replay (ER) buffer $\mathcal{D}_{rep}$ is filled with random demonstration data $\mathcal{D}_s$ along with random weights initialization for both the separate (i.e., online and target networks) neural networks of DQN. The inputs of the network will be the previously defined state-space $\mathcal{S}$ and Q-values as the output. During model training, the load demonstration data flag (l) is deactivated and the separate NN weights are updated (Lines 10-30 in Algorithm 4). At each training step, a mini-batch of state-action transitions are sampled with probability $P(i)$, from $\mathcal{D}_{rep}$ in (5.19).

$$P(i) = \frac{p_i^\alpha}{\sum_k^{k \in |\mathcal{D}_{rep}|} p_k^\alpha}, \tag{5.19}$$

where p_i denotes the proportional priority of i^{th} transition; the exponent $\alpha \in [0, 1]$ is used to introduce randomization into the experience selection in $\mathcal{D}_{rep}$, with $\alpha = 1$ highlighting experiences that require the highest priority; $p_i = |\delta_i| + e$, where δ_i denotes the previously calculated TD error for transition i and the positive constant e ensures all transitions are visited following some probability, even the least visited transitions where the error is zero. ER value normalization is enforced with

$\sum_{k}^{k\in|\mathcal{D}_{rep}|} p_k^{\alpha}$ over the size of $\mathcal{D}_{rep}$. Network updates are adjusted with the Importance Sampling Weights (IS) in (5.20) to correspond to the underlying distribution of experiences and minimize bias.

$$w_i = \left(\frac{1}{|\mathcal{D}_{rep}|} \cdot \frac{1}{P(i)}\right)^{\iota}, \tag{5.20}$$

where $|\mathcal{D}_{rep}|$ denotes the size of ER; $\iota \in [0,1]$ controls the effect of IS on the experience sampling process, and is annealed to 1 over the training duration T [139]. Consequently, the size of the demonstration data will be significantly reduced as the prioritized ER (PER) mechanism in TLD algorithm only samples important transitions[138].

The pre-training phase's aim is to emulate the demonstrator using a value function that fulfills the Bellman equation and perform TD updates on itself as soon as the agent starts interacting with the environment. During the pre-training phase, the agent performs mini-batch sampling with prioritization from the demonstration data, and updates the NN weights by minimizing the overall loss function $J(Q)$, which comprises of four loss components: DQN Loss $J_{DQN}(Q)$, multi-step TD Loss $J_n(Q)$, margin classification loss $J_{MC}(Q)$ and L2 regularization $J_{L2}(Q)$. Firstly, the DQN loss for determining the optimal policy is calculated as follows:

$$J_{DQN}(Q) = \left(r_t + \gamma Q(s_{t+1}, a_{t+1}^{\max}; \theta') - Q(s, a; \theta)\right)^2, \tag{5.21}$$

where $a_t^{\max} = \arg\max_a Q(s_{t+1}, a; \theta)$ with θ are the parameters of the online network, and θ' the parameters of the target network. Secondly, the multi-step TD loss $J_n(Q)$ is defined as:

$$J_n(Q) = \left(R_t^{(n)} + \gamma^n \max_a Q(s_{t+n}, a; \theta') - Q_{on}\right)^2, \tag{5.22}$$

where $Q_{on} = Q(s_t, a_t; \theta)$, and $R_t^{(n)}$ denotes the n-step return from s_t, which is formulated as $R_t^{(n)} = \sum_{k=0}^{n-1} \gamma^k r_{t+k+1}$. Thirdly, the margin classification loss between $\mathcal{D}_s$ sample data and sampled data from $\mathcal{D}_t$ is defined as:

$$J_{MC}(Q) = \max_a [Q(s, a) + l(a_e, a)] - Q(s, a_{MC}), \tag{5.23}$$

where a_{MC} denotes the expert demonstrator action, i.e., DQN with $\mathcal{D}_t$, takes in state s and $l(a_{MC})$ is a margin function that equals 0 when $a = a_{MC}$ and positive

otherwise. We avoid over-fitting on the small demonstration dataset by incorporating the L2 regularization loss $J_{L2}(Q) = ||\theta||_2^2$, expressed as the sum of the squares of the NN weights [140]. Finally, the overall loss function $J(Q)$ is defined in (5.24), and parameters λ_1, λ_2, and λ_3 are weighting factors corresponding to multi-step TD loss, classification loss, and regulation loss, respectively.

$$J(Q) = J_{DQN}(Q) + \lambda_1 J_n(Q) + \lambda_2 J_{MC}(Q) + \lambda_3 J_{L2}(Q). \tag{5.24}$$

Upon completion of the pre-training phase, the model performs training in the *target* environment, and the pseudocode is described in Algorithm 4. There are four differences between TLD and DQN: (i) the ER buffer design $\mathcal{D}_{rep}$, (ii) the PER mechanism, (iii) the overall loss function $J(Q)$, and (iv) model pre-training step. In the next section, we shall empirically demonstrate the learning efficacy of TLD for the AI models within the dual-stage JPdMRM Framework.

5.4.4 Discussion of the Algorithm

5.4.4.1 DQN Convergence

Recall, the scheduling algorithm's objective is to maximize the overall production uptime of z machines, within the machine network of n MGs, through a maintenance resource scheduling method. Simultaneously, the objective of the Q-network training is to minimize the mean square error between the actual and predicted values for a given state-action pair. As long as the value function is well-fit to the problem, TLD is able to learn the optimal policy in accordance to the Bellman equation [76].

Unfortunately, a meaningful comparison with related works in Chapter 3 is not possible since the related works do not examine the same context, issue formulation, or set of assumptions. At the same time, the NN-based Q value function is not guaranteed to achieve its optimal value given dynamic operating conditions and limited training iterations. Recently, [141] showed the DQN convergence to an optimal value function using a statistical analysis approach and the neural fitted Q-Iteration algorithm as an analytical example. However, the coherent understanding of the DQN's statistical and computational features remains an open question. Therefore, we adhere to established DRL analytical practices, including providing as much relevant empirical data as feasible to demonstrate solution convergence

Algorithm 4: Transfer Learning with Demonstrations

Inputs:
$\mathcal{D}_{rep}$: Replay Buffer with Demonstration data,
θ: Network Weights - Initial Behavior (randomized),
θ': Network Weights - Target Network (randomized),
f: Updating frequency for target network,
T_{pre}: Number of pre-training gradient updates.
l: Load Pre-Train or Demonstration Data (Binary)
Output: $\pi_t^*(s) = \arg\max_{a \in \mathcal{A}} Q^*(s, a; \theta)$
// Determine Operational Mode
if $l = 1$ **then**
    $T = T_{pre}$
end
for $t \in \{1, \ldots, T\}$ **do**
    if $\rho = 0$ **then**
        // Training mode (Source)
        Sample action from behavior policy $a_t \sim \pi^{\epsilon Q_\theta}$
        Execute a_t and observe (s_{t+1}, r_t)
        Store (s_t, a_t, s_{t+1}, r) into $\mathcal{D}_{rep}$
        Overwrite earliest self-generated transition if $|\mathcal{D}_{rep}|$ is full
    end
    Mini-batch sample N_q transitions from $\mathcal{D}_{rep}$ with prioritization
    Calculate TD loss of $J(Q)$ using target network
    Perform step-wise gradient descent to update θ
    if $t \ mod \ f = 0$ **then**
        $\theta' \leftarrow \theta$
    end
    if $\rho = 0$ **then**
        $s_t \leftarrow s_{t+1}$
    end
end

in terms of mean episodic reward and learning efficiency relative to similar DRL methodologies.

5.4.4.2 Pre-Training Convergence

According to [142], [143] carefully examines and establishes the conditions under which task transfer Q-learning is effective. [143, Proposition 1] indicates that the learned Q-function from the pre-training task is closer to the optimal policy compared to random initialization, when the distance between $\mathcal{M}_s$ and $\mathcal{M}_t$

is minimal. The speed of convergence is increased as a result of this. Furthermore, [143, Theorem 4] supposes that convergence is guaranteed if the error ratio condition ($\hat{\beta}_n$) in (5.25) is met.

$$\hat{\beta}_n \triangleq \frac{\Delta\left(\mathcal{M}_s, \mathcal{M}_t\right)}{\delta_t} \leq 1. \tag{5.25}$$

$$\Delta\left(\mathcal{M}_s, \mathcal{M}_t\right) = \max_{s,a} \left|Q_s^{\pi}(s,a) - Q_t^{\pi}(s,a)\right| \tag{5.26}$$

where $\delta_t = r_{t+1} + \gamma \hat{Q}(s_{t+1}, a_{t+1}) - \hat{Q}(s_t, a_t)$ denotes the TD-error, which is the difference between the estimates of $Q^{\pi}(s_t, a_t)$ in (5.17), the expected reward and next state-action value [76]; $\Delta\left(\mathcal{M}_s, \mathcal{M}_t\right)$ in (5.26) denotes the optimal Q-function difference between $\mathcal{M}_s$ and $\mathcal{M}_t$, where the optimal Q-functions for *source* and *target* environments are Q_s^{π} and Q_t^{π} respectively. According to [143, Theorem 4], Q-learning convergence will be guaranteed pending the pre-training environment converges, where $\Delta\left(\mathcal{M}_s, \mathcal{M}_t\right)$ is small, and [143, Proposition 1] provides detailed proofs for $\Delta\left(\mathcal{M}_s, \mathcal{M}_t\right)$.

Intuitively, the learned Q-function in the *source* environment is likely near to the optimal Q-function when two RL environments are comparable. Thus, $\hat{\beta}_n$ will be small during the early training phase as we transfer the learned Q-function from the *source* environment to the *target* environment. By contrast, if $\Delta\left(\mathcal{M}_s, \mathcal{M}_t\right)$ is sufficiently large, its dissimilarity correlates to a shift in the learning syllabus, thus diminishing the benefit and performance of transfer learning techniques. In other words, we may either observe sub-optimal performance or delayed solution convergence as the learned Q-function requires additional training iterations for adaptation to the *target* environment. For a comprehensive proof of transfer learning-based solution convergence, we refer interested readers to [143]. Notwithstanding that we use an NN-based approach for the Q value function, we present empirical evidence for solution convergence following established DRL analytical methodology.

5.5 Experiment Setup

To assess the proposed TLD's efficacy, we run numerous simulations that imitate real-world machine degradation circumstances. In addition, we investigate two machine group configurations to demonstrate our solution's versatility in stochastic

environments, apart from the learning convergence results. Specifically, our empirical findings will illustrate the TLD approach's scalability and robustness to similar environments, inadequately discussed in prior works. Finally, this section describes the experimental setup.

5.5.1 Simulation Environment Settings

Our simulation model considers a generic manufacturing system comprising $z=40$ machines, distributed across $n=3$ MGs. MG_1 and MG_2 comprises $z=10$ machines respectively with the remaining machines assigned to MG_3. Overall, the MGs are responsible for producing products based on three parts, PM-1, PM-2 and PM-3. Given that PM-1 generates more revenue per part than those of PM-2 and PM-3, we set that at least 40% of each MG's manufacturing capacity is always committed to PM-1 production. Consequently, we estimate that the maintenance task arrival rate follows a Poisson distribution mean of 4 (i.e., $0.4\times L$), where the maintenance request tasks may include machines responsible for PM-1 partially or entirely.

Within each MG, we consider a pyramidal distribution of manpower to mirror real-world competency level distribution of technicians and that there is always an available technician to attend to machines that require attention. For the example of MG_1 where $Z=M=L=10$, the technician competency level η distribution mapping follows: $\eta \in \{1,2,3\} \rightarrow m \in \{6,3,1\}$. Furthermore, we consider unit service time applies independent of the technician's competency in this work and that an allocated technician cannot be dynamically re-allocated within L time-steps. We set the maintenance budget τ to accommodate the proposed η technician distribution, following the discrete action set in (5.12) for the DRL agent selection. Likewise, we let $\Omega^m = L$ time-steps, per training epoch, and we describe the technician cost constraint details in Table 5.3. Compared to [37], our JPdMRM framework utilizes a threshold-free reward signal approach to determine the optimal maintenance action to take given the current time-step ϵ_w value in I task-set to satisfy the optimization function in (5.16).

In this work, we contextualize the MG configurations by presenting four scenarios in which a pre-trained Resource Scheduling agent for MG_1 (i.e., *source* environment) is re-deployed to MG_2 or MG_3 (i.e., *target* environment). Scenario-1 considers the Poisson mean of 4 and 7 hourly maintenance request tasks in the *source* and *target* environments, respectively, with random task priority distribution. Scenario-1

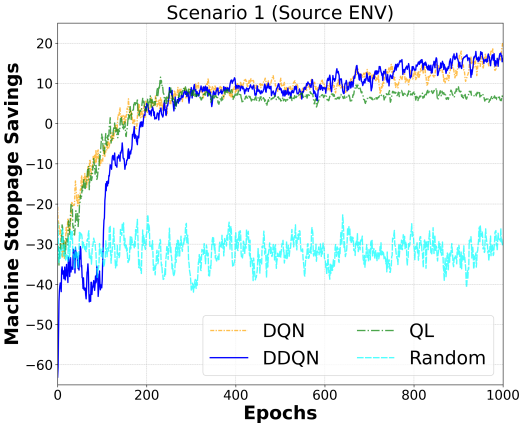
Parameter	*Source* Environment	*Target* Environment
γ	0.99	
ϵ-greedy	$1.0 \rightarrow 0.01$	
Batch size	64	
T_{pretrain}	10000	
α, β_0	0.4, 0.6 [139]	
λ_1, λ_2, λ_3	1.0, 1.0, 10^{-5}	
$\lvert\mathcal{D}_s\rvert$, $\lvert\mathcal{D}_{\text{rep}}\rvert$	20000, 50000	
$c_{\eta=1}$, $c_{\eta=2}$, $c_{\eta=3}$	1, 3, 6	
Scenario-1 λ	$\lambda=4$	$\lambda=7$
Scenario-2	Random, (where $\lambda=4$)	$P(\beta \geq 0.3L\lvert x)=[0,1]$, (where $\lambda=7$ and $x=1$)
Scenario-3 $L=M=10$	Fixed Distribution, where $\lambda=4$	Multinomial Distribution, where $\lambda=7$
Scenario-4 $Z=L=M$ $\lambda=0.4L$	$L=10$	$L\in\{20,30,40,50\}$

TABLE 5.3: Transfer learning parameters

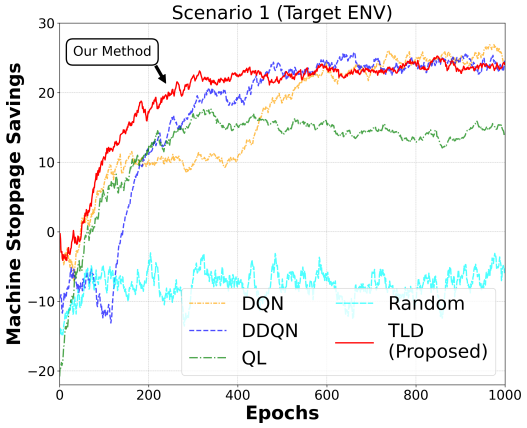
intends to illustrate the efficacy of Transfer Learning for $m=10$ machines. On the other hand, Scenario-2 demonstrates Transfer Learning's robustness in the presence of varying maintenance request probability, where at least three $x=1$ priority maintenance request tasks, in the *target* environment. Similarly, Scenario-3 takes a multinomial distribution of manpower expertise into account in the *target* environment, where $L=M=10$. Given the substantial shift in manpower distribution, we are also interested in quantifying potential performance gaps, followed by a brief discussion and pertinent recommendations for addressing them. Finally, Scenario-4 demonstrates the scalability of Transfer Learning by initially training the TLD model on the *source* environment for $m=10$ machines before re-deploying it to the *target* environment, comprising machine sets $m\in\{20,30,40,50\}$ respectively. Table 5.1 in Section 5.2 contrasts the environment parameter differences.

5.5.2 Transfer Learning Configuration and Implementation

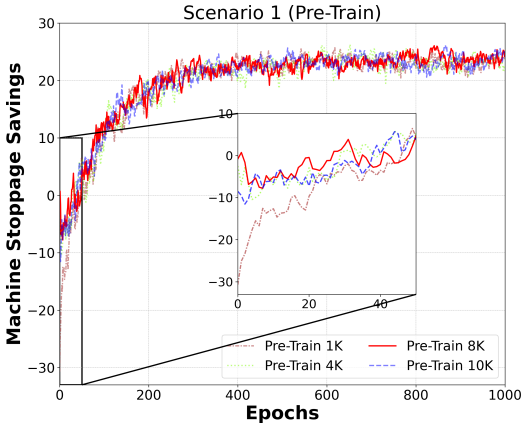
Consider the plastic injection molding (PIM) machines that manufacture the plastic casings for generic power tools. [144] states that manufacturers usually have >80 machines per manufacturing site. Identical machines can be located at the same site, yet they produce various goods but with different operational settings and materials. Assume MG_1 and MG_2 comprises of PIM machines producing the



(a) Source ($\lambda = 4$ requests per hour).



(b) Target ($\lambda = 7$ requests per hour).



(c) Pre-Training epochs performance ablation.

FIGURE 5.4: Performance analysis for Scenario-1.

front and rear housings of a generic power tool separately for eventual assembly during the production process. Because of our two-stage framework, the resource allocation model alone can acquire both human and AI-based demonstrations to improve TLD's learning efficiency. Ideally, TLD will learn from offline demonstrations by human experts. Due to the unavailability of human experts, this chapter focuses only on AI-based demonstrations.

The transfer learning process commences with acquiring expert demonstration data from the *source* environment, followed by pre-training the TLD agent to initialize of NN weights in the *target* environment. Upon pre-training completion (i.e., knowledge transfer), the TLD agent interacts within the new *target* environment and learns the optimal policy via continuous updates to its NN weight while also leveraging prior demonstration data. TLD comprises two fully connected hidden layers of size 24 neurons each, with the number of inputs and outputs corresponding

to $\mathcal{S}$ and $\mathcal{A}$ respectively.

The baseline methods of DQN, double DQN (DDQN), Q-learning (QL), and random policy are then investigated for the purpose of evaluating TLD performance and to emphasize TLD's efficacy. TLD's network is updated every 100 episodes during training, and the training parameters are summarized in Table 5.3.

We appoint DDQN as the expert demonstrator for the pre-training of TLD before deploying to the *target* environment setting. Unlike TLD, the suggested baseline methods learn directly from each environment using randomly initialized NN weights. DDQN and DQN's NN parameters are adapted from [145], which implements two fully connected hidden layers with a hidden layer size of 32 and batch size of 64. For Q-Learning, we set the learning rate and discount factor as $\alpha = 0.9$ and $\gamma = 0.9$, respectively. In the experiments, we use the heuristics-based sorting algorithm to obtain the order of maintenance request tasks in I instead of sampling over the entire action space, mainly for space complexity reduction and solution tractability. In the next section, we will highlight and discuss the more intriguing transfer learning results.

5.6 Result Analysis and Discussion

In this section, we evaluate the TLD's performance in terms of mean reward, and the number of training steps needed to reach solution convergence using the four scenarios from Table 5.3. The mean reward of an epoch is determined by averaging all the immediate rewards, as in (5.14). An epoch's duration is equivalent to the buffer length L. Following that, the number of epochs required to reach solution convergence are established, and empirically evaluated to illustrate the efficacy and limitations of the TLD method. For each scenario, we utilize the mean reward averaged over five different runs to account for the stochastic nature of NNs. Finally, we rename the *mean reward* (i.e., overall production uptime) as *machine stoppage savings* to help readers better understand the problem context and interpret the results.

5.6.1 Efficiency of Transfer Learning - Scenario-1

Figure 5.4a illustrates the performance of the various baseline techniques in the *source* environment, reiterating our choice of DDQN as the expert demonstrator

for generating expert data. Interestingly, we discover that the NN size utilized to produce the demonstration data, DDQN, has a minimal impact on TLD's learning performance. Based on the empirical results, we are confident that the proposed DDQN network's learning capacity is adequate for efficiently learning the mapping function for the complex optimization problem. Thus, each increase in computing resource yields negligible gains in learning performance at the cost of increasing computational resource consumption and model training wall-time.

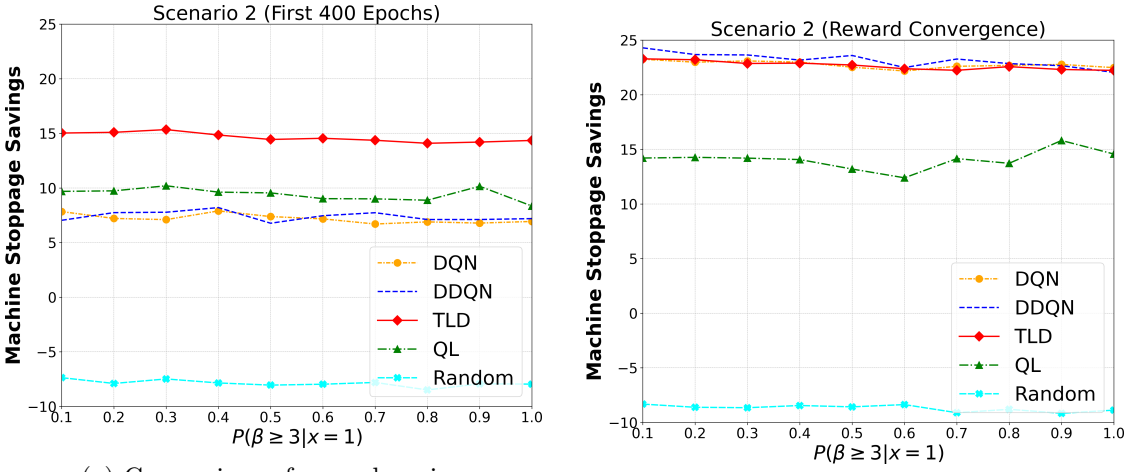
Given the high cost, scarcity, and difficulty of collecting real-world expert demonstration data, we undertake an additional performance experiment to ascertain the impact of pre-training dataset size on TLD's learning performance. As shown in Figure 5.4c, TLD converges within 400 epochs, independent of the number of demonstration data points. The only discernible difference arises during the first 50 epochs, where the initial high scores are directly attributable to the increased availability of available demonstration data. In other words, pre-training TLD for Scenario-1 beyond 8000 epochs (i.e., 80,000 demonstration data points) does not significantly improve performance over TLD pre-training with 10,000 epochs. For reference, the mean reward computed over the last 200 epochs is 24 ± 3.0 and 23 ± 3.4 for the 8000 and 10,000 pre-training epochs, respectively. Unless otherwise specified, we will now use the 8000 epochs pre-training option for all TLD benchmarks.

	DQN (Baseline)	DDQN	**TLD**	QL
Epochs to Converge	600	470	**380**	300[4]
Improvement (%)	-	28	**58**	NA

TABLE 5.4: Comparison of learning efficiency for Scenario-1.

In contrast to the baseline, which learn tabula rasa, TLD is more efficient in learning the optimal decision-policy in the *target* environment. Figure 5.4b illustrates the benefit of TLD, and Table 5.4 summarizes the results for the reader's reference. Conversely, QL converges at non-optimal local maxima, while the PER mechanism in TLD avoids this limitation by sampling the more critical transitions within the ER buffer. As shown in Figure 5.3, the ER buffer contains two types of data: expert demonstrations data (i.e., source) and the self-learned data (i.e., target). When PER is enabled, the TLD agent samples the more important state-action

[4]The corresponding algorithm converges to a sub-optimal score relative to DDQN (i.e., the Expert)



(a) Comparison of mean learning performance.

(b) Comparison of reward convergence scores.

FIGURE 5.5: Performance analysis for Scenario-2.

transitions earlier in the training phase, thus accelerating the process of learning exploration.

5.6.2 Robustness of Transfer Learning

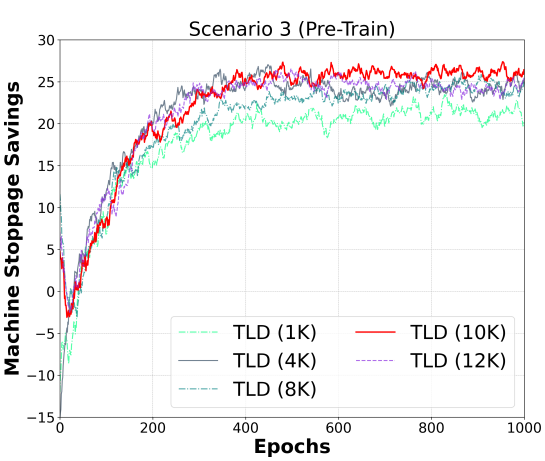
5.6.2.1 Scenario-2

Scenario-2 is similar to Section 5.6.1 in that we conduct transfer learning to the proposed *target* environment with varying probabilities of urgent priority maintenance requests. In comparison to the baseline, Figure 5.5a demonstrates TLD's superior learning efficiency, with comparable learned reward performance to DQN and DDQN methods, as shown in Figure 5.5b.

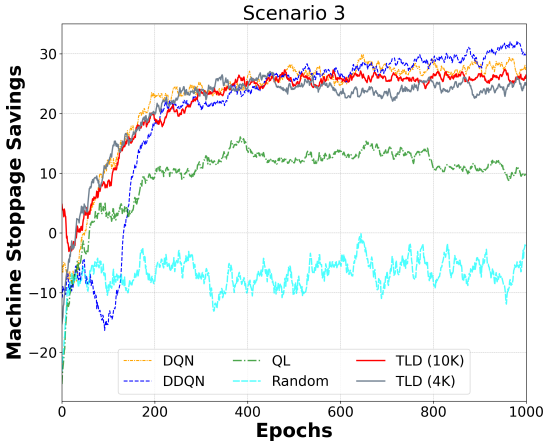
5.6.2.2 Scenario-3

Figure 5.6b depicts the anticipated effects of the significant changes in the distribution of manpower resources toward TLD learning performance. Although TLD's performance converges, it remains slightly lower than that of DQN and DDQN. Accordingly, we propose the following points to elucidate the factors behind TLD's current performance, namely *multi-environment learning generalization*, *pre-Training dataset size*, and *TLD learning capacity*.

Considering how well the pre-trained model performs in the *source* environment, we hypothesize that the corresponding model is likely trapped in a local minima



(a) Pre-Training epochs performance ablation.



(b) Multinomial distribution for $M = 10$ technicians.

FIGURE 5.6: Performance analysis for Scenario-3.

optimization point as it fine-tunes the learned decision policy, from the *source* environment, for the *target* environment. As a result, the performance impact is less apparent in scenarios with similar data distribution, such as Scenario-1. However, with the significant shifts in manpower distribution in Scenario-3, the pre-trained TLD agent may interpret its new observations as noise, rendering it incapable of adapting effectively to the *target* environment. Let us consider the *target* environment with the following manpower distribution: one expert, six senior, and three junior technicians. In the *source* environment, however, the manpower distribution is fixed with one expert, three senior and six junior technicians. Assume there is now just one high-priority and nine low-priority tasks. Without regard for the expert technician assignment, the pre-trained TLD agent will automatically allocate all junior technicians and may, with some unknown probability, assign the remaining maintenance tasks to the senior technicians. Hence, TLD will perform no better than random initialization-based tabula rasa DRL training, and optimizing the learning efficiency of Transfer Learning methods remains an open research question [39, 146, 147]. Similarly, given the *MG* configurations in Table 5.1, it is both inefficient and impractical to maintain part-specific DRL models. Therefore, we believe that the minor performance penalty associated with TLD is justified in exchange for the flexibility to support multiple parts.

The quantity of demonstration data required to achieve reasonable performance using Transfer Learning techniques is seldom published in the literature and is often application-specific. Therefore, we undertake a performance ablation study for the

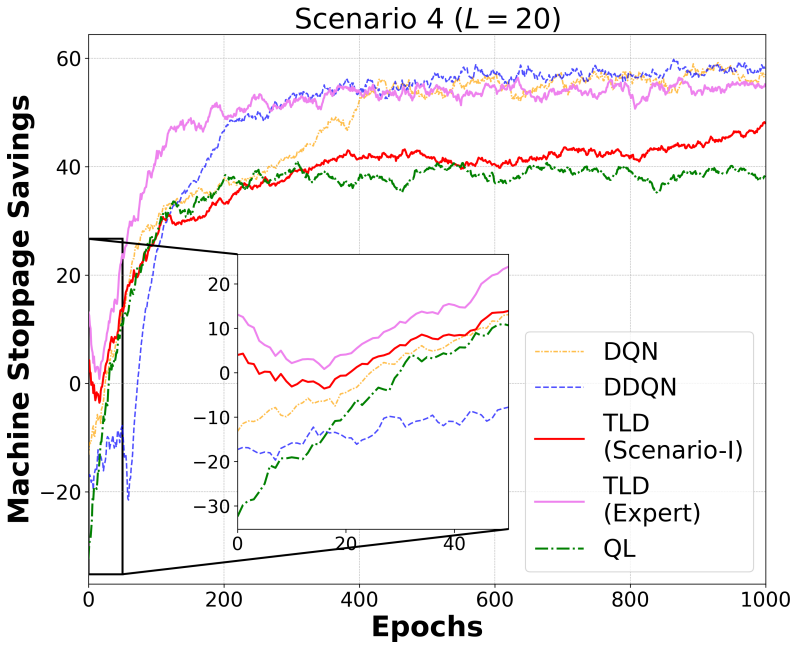


FIGURE 5.7: Performance comparisons of baseline and TLD variants for $L = 20$. The performance difference is due to different knowledge transfer source used in TLD variants.

pre-training phase and present our findings in Figure 5.6a. We initially assume that expert demonstration data is readily available and beneficial towards improving TLD performance. From Figure 5.6a, the amount of expert demonstration data for pre-training does have a noticeable effect on TLD learning performance. For instance, pre-training TLD for 4000 epochs (i.e., TLD (4K)) maximizes the learning efficiency rate while producing a slightly lower mean reward score than that of pre-training with TLD for 10,000 epochs option (i.e., TLD (10K)). Alternatively, augmenting the limited demonstration dataset with additional real-world expert data may be explored in future works.

As with Scenario-1, there is a diminishing performance increase for TLD model pre-training beyond 10,000 time-steps, which is tightly related to the learning capacity of our proposed TLD architecture. Additionally, the TLD's NN neuron configuration implicitly imposes a learning performance constraint on individual TLD layers, which may be overcome by increasing the number of NN neurons per layer at the expense of additional NN hyperparameter fine-tuning, which can be the future work.

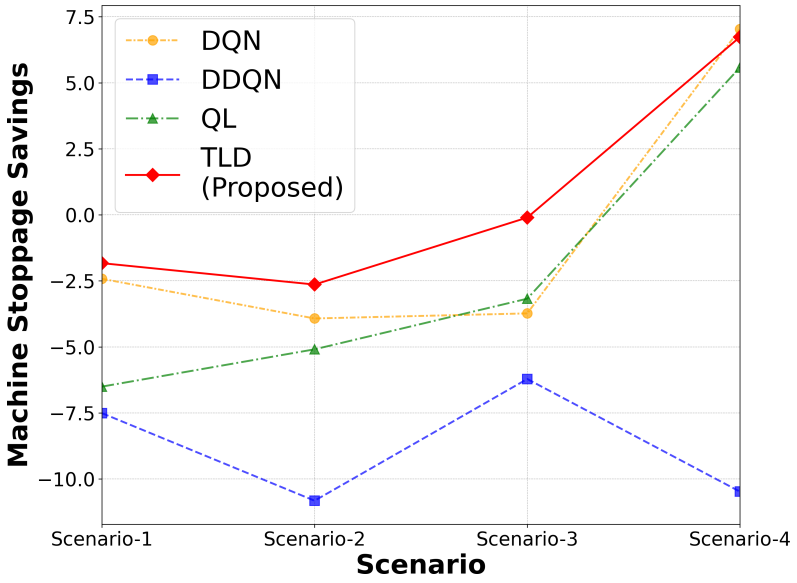


FIGURE 5.8: TLD validates the advantages of learning from expert demonstrations and is superior to baseline techniques throughout Scenarios 1 to 4. Furthermore, it is reasonable to anticipate that the advantages of increased learning efficiency diminish with progressively divergent scenario objectives (e.g., Scenario-4) relative to the expert demonstration dataset.

5.6.3 Scalability of Transfer Learning - Scenario-4

Only Scenario-4 considers two TLD variants, "TLD (Scenario-1)" and "TLD (Expert Demo)": "TLD (Expert Demo)" is pre-trained with demonstration data from Scenario-4; "TLD (Scenario-1)" is pre-trained with demonstration data from Scenario-1. With the TLD variants, we can now gain insights into the performance scalability impacts associated with the knowledge transfer source types and is under-explored in prior research works. In retrospect, we anticipate that the efficiency and learning performance advantages of 'TLD (Scenario-1)" will deteriorate when the *target* environment's data distribution changes in response to increasing problem complexity. When $L = 20$, "TLD (Expert)" achieves the highest overall performance, but "TLD (Scenario-1)" only manages to outperform QL slightly, as shown in Figure 5.7. When $L = 50$, however, the reverse is true, as "TLD (Scenario-1)" achieves higher performance after the 100^{th} epoch has elapsed with both methods converging at the 1000^{th} epoch. Interestingly, we note a 5% difference in learning efficiency with a 19% difference in mean reward received, and Table 5.5 summarizes the results for Scenario-4 experiments. Except for $L = 20$, TLD's learning convergence performance is less than the DQN baseline but better than QL. We attribute the performance difference between TLD and DQN to the

substantial distribution shift of state-action transitions between the *source* and *target* tasks. Consequently, the intrinsic knowledge bias derived from TLD's prior knowledge influences decision-making towards the target task scenario, resulting in TLD's learning performance being close to DQN. On the other hand, it is well-established that DQN outperforms QL. The results in Table 5.5 corroborate our previous explanation in Section 5.6.2.2 and elucidate the limitations of Transfer Learning technique as an open research problem [39, 147].

Event Buffer Size (L)	Epochs to Learning Convergence[5]					Mean Convergence Reward (Last 400 epochs)				
	QL	DQN	DDQN	TLD (Scenario -1)	TLD (Expert Demo)	QL	DQN	DDQN	TLD (Scenario -1)	TLD (Expert Demo)
20	300	430	380	400	380	38 ± 3.6	55 ± 6.8	57 ± 5.4	43.5 ± 4.1	54 ± 5.5
30	210[4]	500	460	850[4]	610[4]	62 ± 4.7	85 ± 7.5	89 ± 7.3	75 ± 6.3	77 ± 9.0
40	540[4]	620	570	660[4]	800[4]	86 ± 5.1	123 ± 8.6	123 ± 7.6	108 ± 8.2	105 ± 12
50	290[4]	530	470	800	820	109 ± 6.9	156 ± 9.7	156 ± 9.7	146 ± 10.7	145 ± 11.9

TABLE 5.5: Learning efficiency (lower is better) between proposed baseline methods and TLD for different values of L. Performance results (higher is better) for values of L with the mean rewards are presented. Two TLD variants are introduced for comparison: TLD (Scenario-1) and TLD (Expert Demo).

[5]Convergence can be verified by re-running the *target* environment trained DRL models in test mode for 100 epochs with the option of human validation.

5.6.4 Brief Discussion of Overall Results

Overall, we have demonstrated the efficacy and benefits of incorporating TL into our JPdMRM DRL-based framework. To accelerate the learning performance of DRL algorithms, we investigated the use of TL with expert demonstrations. Furthermore, the scalability and limits of TLD have been discussed, arising from the distributional shift of the resource optimization problem. Therefore, we summarize the results from all four scenarios in Figure 5.8 for the first 50 epochs. The slight overhead from pre-training is considerably beneficial for highly similar scenarios (e.g., scenarios 1 and 2) and offsets the need to train DRL models from scratch. TLD can also be used to capture human expert demonstrations, but doing so exceeds the scope of this work.

5.7 Summary

Data-driven resource management is critical in the Industry 4.0 context for decision support to maintenance teams to minimize interruptions to production schedules. This chapter proposes a two-stage TL-based DRL approach for an edge-based machine group of predictive maintenance models. We begin by framing the machine maintenance request events as a task optimization problem based on the predictive model data. Following that, we formulate the sequence of maintenance request tasks as a sequential decision-making problem using the Markov Decision Process to learn an optimal decision policy and enable effective manpower resource management. We can inherently accelerate the DRL model's learning performance by enabling the proposed DRL model to harness relevant experiences from similar machine configurations. Empirically, our experiments demonstrate that coupling TL with the two-stage PdM framework significantly improves learning efficiency and performance over baseline methods. However, our proposed TLD model is feasible for discrete action space problems and requires alternative models for continuous action space problems. Moreover, this work did not examine how knowledge distillation can boost TLD's learning efficiency. Chapter 7 details some suggestions on future research directions.

Chapter 6

Remaining Useful Life Prediction of IIoT-Enabled Equipment using Attentive Multi-Branch Feature Network

6.1 Introduction

Effective decision-support-based PdM framework requires high equipment severity rating prediction accuracy to instill trust with the human decision-maker and minimize false maintenance alerts. Considering the link between equipment severity rating and RUL, improving RUL prediction accuracy increases the equipment severity rating accuracy intrinsically. In order to predict RUL for heterogeneous industrial machines, this chapter formulates PdM as a multivariate regression-based problem.

This chapter formulates PdM as a multi-variate regression-based problem to predict RUL for heterogeneous industrial machines. Unlike typical LSTM-based networks, our approach considers LSTM networks integrated with a multi-branch attention-based feature network (AMBFNet) to prioritize time-series sensor data for data analysis automatically. When benchmarked on the popular C-MAPSS dataset, the empirical results illustrate the efficacy of AMBFNet, even outperforming existing

works by 11% on average. Furthermore, we observe comparable performance trend results for a real-world industrial washing machine dataset.

Limited real-world data remains an issue in practice and often leads to sub-optimal model prediction accuracy, and the un-monitored field-deployed model's performance on edge-based devices tends to drift over time.[1] Besides, edge devices are compute and memory bounded with diverse configurations. This work considers RUL prediction model re-training on an edge server before conducting extensive experiments to identify the appropriate amount of batch-sized data with the lowest error deviation compared to tabula rasa model training. Once the batch size is confirmed, the model is ready for deployment to field edge devices. Our results indicate that batch-based incremental training saves an average of 60% in training time on the C-MAPSS dataset, resulting in a mean performance difference of -1.1% and -4.9% for the prognostics performance measures root-mean-square-error and score, respectively. However, the proposed incremental learning method is not implemented on the IWM dataset owing to a lack of data. Conversely, the current literature has not investigated incremental learning for the C-MAPSS dataset.

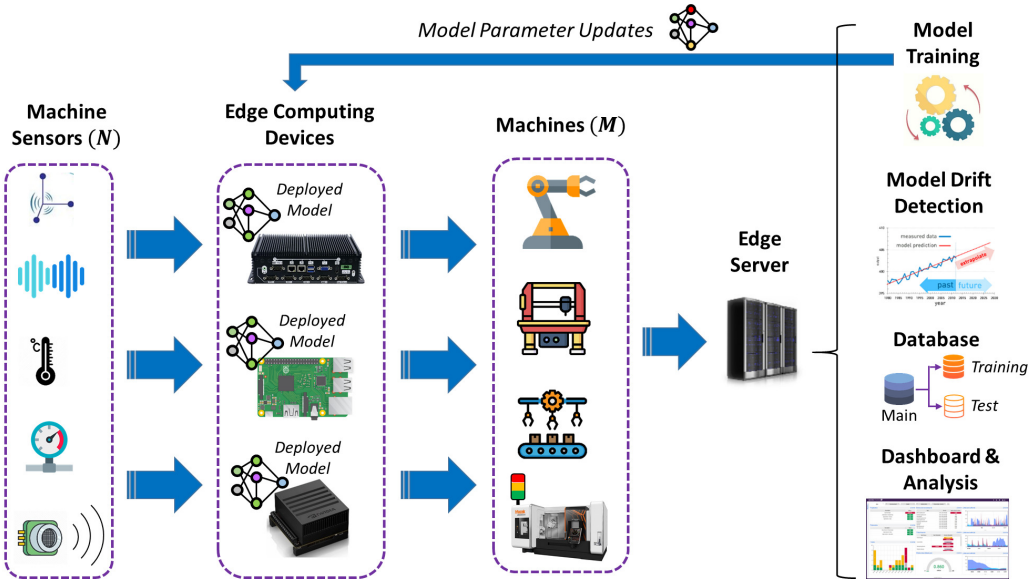


FIGURE 6.1: IIoT-enabled system model overview for RUL prediction and model drift monitoring.

6.2 System Model and Problem Formulation

In this work, we consider a generic production facility for predictive machine maintenance in IIoT (Figure 6.1), comprising of an Edge Server (ES) and Edge Computing Device (ECD).The system model in Figure 6.1 considers a network of ECDs with a direct connection to the ES at the production facility level. These machines are sensor-equipped for data collecting and ECDs with PdM models for online condition monitoring. We denote sensor data from m^{th} machine as $x_t^{(m,n)}$, where $m \in \{1, \ldots, M\}$ denotes the number of machines monitored, and $n \in \{1, \ldots, N\}$ specifies the index of the heterogeneous sensor. For practical reasons, we assume N sensors per machine. Temperature, pump pressure, and vibration are examples of n sensors. Individual ECDs aggregate in situ time-series sensor data before transmission to ES for PdM analysis, storage, and model training. Meanwhile, a predictive model continuously monitors the incoming data for anomalous behavior and triggers notifications or alarms based on parameter threshold presets. From the dataset perspective, we consider each sensor an input feature, and machine operational cycle time (i.e., lifecycle) is automatically calculated and verified with existing production processes. After that, we can derive an artificial second-order signal from the machine's information which is termed RUL (i.e., Y_{train}).

Time-series sensor data is periodically transmitted from each machine and stored in a database for logging purposes. At the same time, a PdM model performs background monitoring of incoming data for irregular patterns and provides maintenance recommendations (e.g., RUL predictions) to the maintenance team. To ensure the PdM model remains updated, selected ranges of historical sensor and RUL data, termed training dataset (X_{train}), is automatically retrieved for training PdM models, where X_{train} is a tuple of $<\chi, Y_{\text{train}}>$, where $\chi \in x_{t\to\infty}^{(M,N)}$. In reality, only machine sensor data (i.e., $X_{\text{test}} = x_{t+1}^{(M,N)}$) is available, whereas the matching RULs are unknown ahead of time and must be predicted (i.e., $\hat{Y}_{\text{test}}$).

$$\hat{Y}_{\text{test}} = f(X_{\text{test}}). \tag{6.1}$$

The main objective of regression techniques is to map the relationship between χ and Y_{train}. From the learned mapping function f, the regression algorithm will predict the corresponding $\hat{Y}_{\text{test}}$ values based on real-time sensor data X_{test}, as defined in (6.1). However, the real-world machine operating conditions are stochastic, have

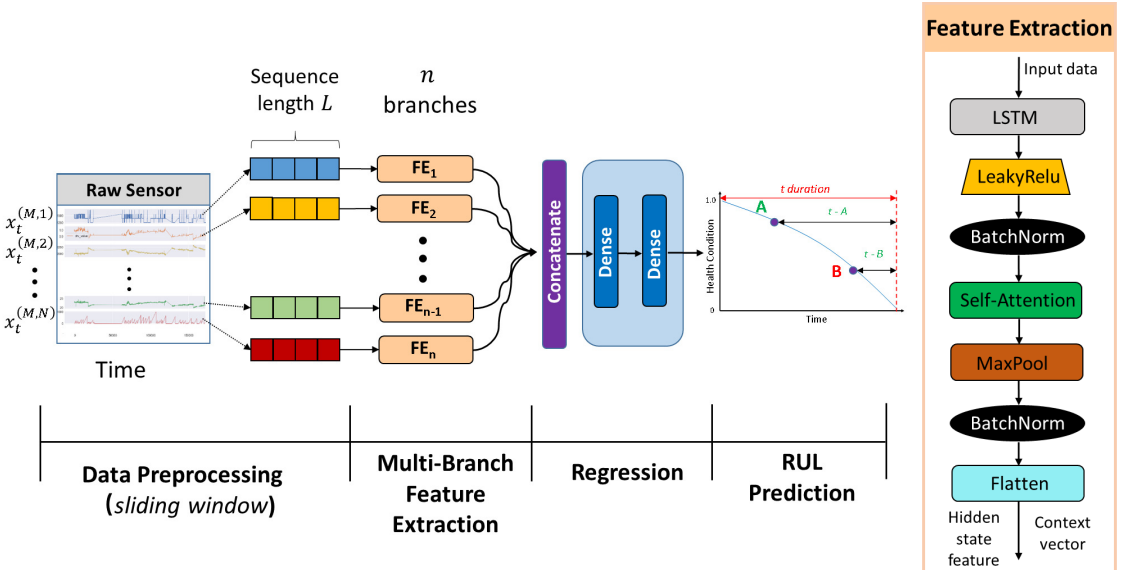
[1]Part of the work in this chapter is **published** in [40].

unknown noise distributions, and have multiple sensors as inputs (i.e., features). Hence, f is non-linear and becomes challenging to provide accurate real-time predictions of $\hat{Y}_{\text{test}}$ for similar and identical machines. Therefore, this work proposes a deep learning approach for establishing f via the attention-based mechanism to improve RUL prediction accuracy.

6.3 Solution of Deep Learning

6.3.1 Model Overview

We present the attentive multi-branch feature network (AMBFNet) method in Figure 6.2a for equipment RUL prediction, such as the aircraft turbofan engine and industrial washing machines. Within each branch of AMBFNet, the LSTM network is responsible for modeling the input sequences of n sensors. The attention mechanism then dynamically assigns weights to the sequenced data to determine which time-series and multi-variate sensor data require more attention, relatively. Then, the sensor data and the attention weights from all branches are aggregated. Finally, a fully-connected network (FCN) performs regression based on the branch aggregated data and forces the FCN to learn and select more essential RUL prediction features. Note that each feature branch includes the batch normalization layer



(a) AMBFnet Model Architecture.

(b) Branch-based FE_n.

FIGURE 6.2: Overview of attentive multi-branch feature network.

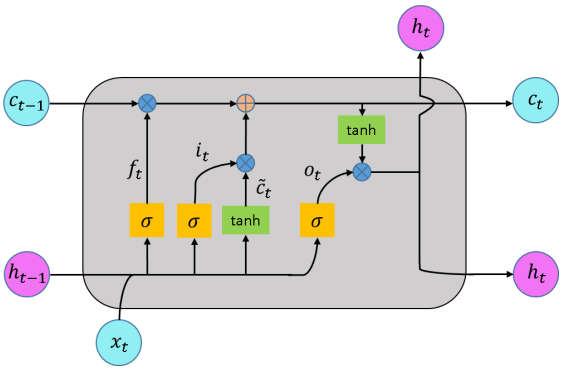


FIGURE 6.3: Overview of LSTM cell.

and appropriate activation functions to minimize model over-fitting between different models, see Figure 6.2b. Before discussing the remaining model components, we explore a uni-branch feature extractor, as shown in Figure 6.2b, to explain the AMBFnet model.

6.3.2 Long Short-Term Memory (Encoder)

The long-short term memory network (LSTM) [52] is a type of RNN for modeling long sequences of data by temporal learning of feature patterns. Without loss of generality, we consider $M = 1$ machine and N sensors, which results in an input sample of $X = \{x_1, x_2, \ldots, x_L\}$ at each time-step t with a data sequence length of L. LSTM comprises a collection of information-processing gates that are controlled by the current values of the input $x_t^{(M,N)}$ and cell c_t at time-step t, in addition to various gate-specific characteristics. Specifically, the forget gate f_t, input gate i_t, output gate o_t, the hidden state h_t at time-step t, and the memory state c_t are all components of a typical LSTM cell, see Figure 6.3. The gate structures are formally described as follows:

$$
\begin{aligned}
f_t &= \sigma_l\left(W_f \cdot \left[h_{t-1}, x_t^{(N)}\right] + b_f\right), \\
i_t &= \sigma_l\left(W_i \cdot \left[h_{t-1}, x_t^{(N)}\right] + b_i\right), \\
\tilde{c}_t &= \tanh\left(W_c \cdot \left[h_{t-1}, x_t^{(M,N)}\right] + b_c\right), \\
c_t &= f_t * c_{t-1} + i_t * \tilde{c}_t, \\
o_t &= \sigma\left(W_o \cdot \left[h_{t-1}, x_t^{(N)}\right] + b_o\right), \\
h_t &= o_t * \tanh\left(c_t\right),
\end{aligned}
\tag{6.2}
$$

where b_f, b_i, b_c, and b_o are bias vectors for input vector of sensor data $x_t^{(M,N)}$ at time-step t; W_f, W_i, W_c, and W_o denotes the weight matrices; $\tilde{c}_t$ represents a candidate for some cell state at time-step t; h_{t-1} represents the state of the hidden layer at time-step $t-1$; $*$ represents the element-wise vector multiplication, and the sigmoid activation function σ governs the gating behavior.

Overall, the LSTM ingests input sequence X and outputs L sequence of hidden states as in (6.3). In some sense, the LSTM assumes a pseudo feature encoder role f_{enc} with formal definition in (6.4).

$$\hat{y} = \{h_{j=1}, h_{j=2}, \ldots, h_{j=L}\}, \tag{6.3}$$

$$\hat{y} = f_{enc}(X). \tag{6.4}$$

Although [53, 55, 103] have previously examined the use of LSTM for RUL prediction, the performance of RUL prediction is not as competitive as CNN-based models. We hypothesize that [55] and [103]'s sole use of LSTM-based network configurations may obfuscate feature analysis at the regression model layer. Furthermore, current LSTM models have a propensity for catastrophic forgetting when modeling extensive data sequences. We, therefore, propose to mitigate these problems using an attention mechanism to supplement the LSTM network and provide additional feature-based information to enhance the RUL prediction performance.

6.3.3 Local Attention (Decoder)

[57] proposed an attention method based on how people learn to focus on some regions of a picture during image recognition. Specifically, the attention mechanism dynamically weighs the importance of various pixel areas for every image. The attention method and its variants are a staple component for modern neural network architecture and can be found in various applications, including speech, language translation, and time series prediction [148, 149]. The attention model's role in RUL prediction is to annotate critical features (i.e., sensor values) for multiple sensors across all time steps. In contrast to the LSTM network, we consider the attention model as a pseudo decoder.

Three parameters are input into decoder model f_{dec} to retrieve the decoder's current hidden state s_j at time step j: context vector (z_j), LSTM encoder output $(\hat{y}_j)$

from (6.3), and the decoder's previous hidden state s_{j-1}. We formally define these variable's relationship in (6.5).

$$s_j = f_{dec}(z_j, h_j, s_{j-1}), \tag{6.5}$$

Recall Chapter 2.1.5, the central concept of attention mechanism is attributed to the attention score c_j which weighs the importance of a feature input. For computation of the attention scores with respect to LSTM encoder output, we redefine z_j as a weighted sum of h_j in (6.6) and Figure 6.4.

$$z_j = \sum_{i=1}^{N} \alpha_{i,j} h_j, \tag{6.6}$$

where z_j denotes the decoding position j, h_j for the j^{th} encoder state in (6.3), L represents the data sequence length from the LSTM encoder, and α_{ij} represents the degree to which the i^{th} output should pay attention to the j^{th} input. Therefore, we can obtain $\alpha_{i,j}$ by computing the softmax over the attention scores $e_{i,j}$ of the inputs in relation to the i^{th} output as in (6.7).

$$\alpha_{i,j} = \text{softmax}(e_{i,j}) = \frac{exp(e_{i,j})}{\sum_{k=1}^{T} exp(e_{i,j'})}, \tag{6.7}$$

where

$$e_{i,j} = f(s_{i-1}, h_j). \tag{6.8}$$

Note that f is an alignment function (i.e., *alignment score*) which grades how

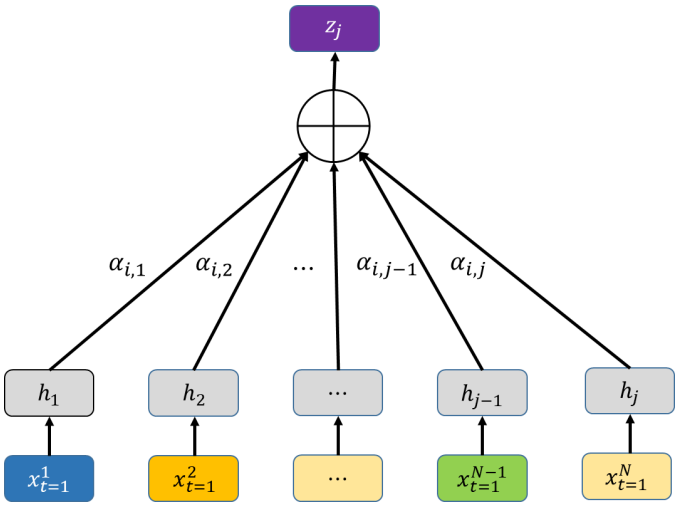


FIGURE 6.4: Attention-based context calculation for encoder hidden states.

Variants	Alignment Score Function	Reference
General	$h_i^\top W_a h_s$	[150]
Dot product	$h_i^\top h_s$	[150]
Scaled dot product	$\frac{h_i^\top h_s}{\sqrt{d_k}}$	[57]
Additive attention	$v_a^\top tanh(W_a [h_i : h_s])$	[75]
Generalized Kernel	$\phi(h_i)^\top \phi(h_s)$	[151]

TABLE 6.1: Examples of attention-based variants with corresponding alignment score functions.

well the inputs near position j and the output at position i match, s_{i-1} denotes the hidden state from the previous time step, and j' denotes the next position. The alignment function can be approximated using neural networks and optimized using gradient methods, such as gradient descent.

According to [149], there exists various attention-based variants: (i) additive attention, (ii) self-attention, (iii) scaled dot-product attention, and (iv) multiplicative attention. Table 6.1 depicts the attention variants categorized according to their alignment score (h_j, h_s) (i.e., f in (6.8)). Herein, the trainable weight matrices W_a and v_a, source hidden state h_s, length of input hidden state d_k, and target hidden state h_t.

Unlike the existing works (e.g., [55, 103, 152]), which adopt multi-head attention for RUL prediction, we adopt the *self-attention*[153] method instead of prioritizing the LSTM network's output data sequences. The critical distinction is that *self-attention* may use any of the alignment score functions in Table 6.1 and requires the locations of the input sequence to calculate and output a representation of the sampled sequence [153]. Notably, the *self-attention* method calculates the following variables from (6.9) to (6.12): attention weight α_t, the alignment scores $h_{j,j'}$, the alignment model $e_{j,j'}$, and context vector z_i.

$$h_{i,j} = tanh(x_i^\top W_i + x_j^\top W_x + b_i), \tag{6.9}$$

$$e_{i,j} = \sigma(W_a h_{i,j} + b_a), \tag{6.10}$$

$$\alpha_{i,j} = softmax(e_{i,j}), \tag{6.11}$$

$$z_j = \sum_{i=1}^{L} \alpha_{i,j} h_j, \tag{6.12}$$

where the weights $W \in \{W_i, W_x, W_a\}$ and biases $b \in \{b_i, b_a\}$ are trainable parameters to be learned, and σ denotes the sigmoid activation function. Furthermore, we adopt the *multiplicative attention* method in (6.13)[150] for calculating the attention score or *branch* score, in the context of AMBFNet model. We argue that multiplicative attention is a more efficient use of memory and computing resources than the above-mentioned alternatives by exploiting modern graphical processing units (GPUs) for matrix multiplication.

$$branch = score(h_i, h_j) = h_i^\top W h_j, \tag{6.13}$$

6.3.4 Multi-branch Attention

Unlike [57], we consider separate instances of the attention mechanism as a *branch*, where each *branch* is responsible for learning from the assigned sensor data sequences. When dealing with multiple sensors (i.e., *branches*), our proposed AMBFNet model concatenates N branches to generate output s and is termed *multi-branch attention* in (6.14). Although figure 6.5 depicts the AMBFNet with $N = 4$ sensor inputs, AMBFNet can accommodate multi-variate time series data for up to N inputs or features.

$$s = \text{concat}[\{branch_p\}_{p=1}^{N}], \tag{6.14}$$

6.3.5 RUL prediction

The multi-branch component in (6.14) is part of the network, with its output routed to the concatenation layer for aggregation purposes. The *branch*-based information enhances the model's understanding by aggregating the data to provide an overall spatial representation via feature ranking, increasing the feature-to-noise ratio to improve overall RUL prediction performance. The concatenated data then undergoes feature mapping to a single RUL value, see (6.15), via the stacked fully-connected network (FCN) layers. Finally, a single RUL prediction value is estimated, as shown in Figures 6.2a and 6.5.

$$\hat{RUL} = f_{pred}(s). \tag{6.15}$$

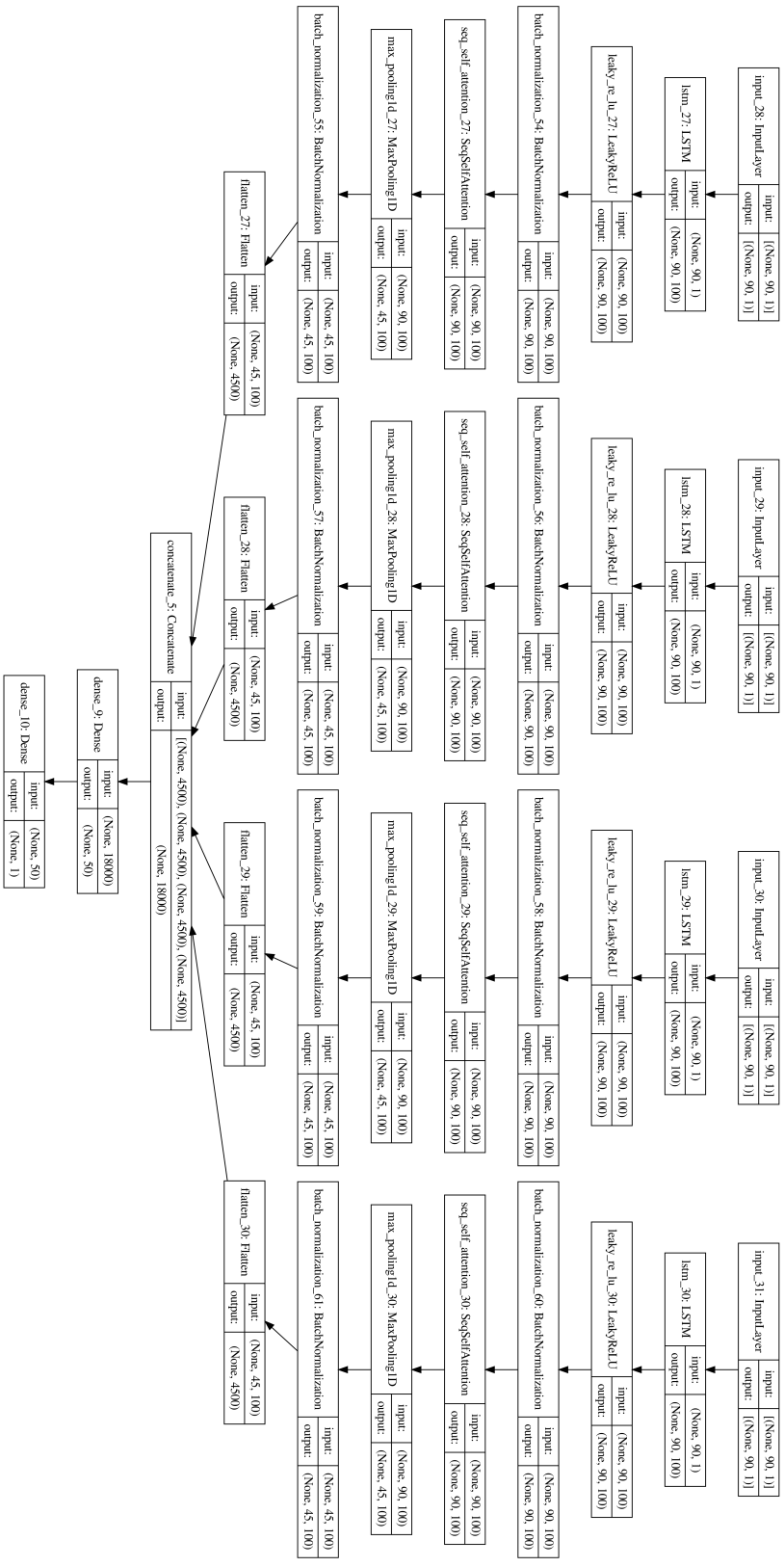


FIGURE 6.5: Example AMBFNet model architecture with $N = 4$ branches.

where f_{pred} represents the non-linear FCN model, $\hat{RUL}$ denotes the predicted RUL value, and s as the aggregated data from the multi-branch network.

6.3.6 Handcrafted Features

Feature engineering is widely known to increase the accuracy of a machine learning model, with some success on RUL prediction reported in [43]. Our study also discovered numerous sensors exhibiting shared properties via exploratory data analysis. For example, we observed properties such as increasing and decreasing trend lines from raw sensor data. Additionally, we adopted the exponential moving average approach to creating smoothing features for each trend. Notably, we normalize these new handcrafted features and the original sensor data for $N + 3$ sensor inputs.

6.3.7 Incremental Learning

DL models require reasonably large data to achieve good RUL prediction performance. During early model deployment, however, only limited high-quality data is available. Unfortunately, transfer learning's efficiency cannot be guaranteed when applied to real-world data with stochastic properties, resulting in lower prediction accuracy performance (i.e., negative transfer)[154, 155]. As more data becomes available, setting the criteria (e.g., frequency and minimum data points) to trigger model re-training is non-trivial and dataset-specific. In addition, frequent access to cloud computing resources for model re-training is costly and poses privacy and security risks. Conversely, continual training on limited data leads to model over-fitting.

We offer an incremental learning technique to address these concerns and assume online model re-training occurs on an edge computing device to guarantee the privacy of both data and application. Specifically, only partial model parameter updating is allowable for accuracy improvements in the short term without over-fitting. Consequently, the pre-trained model significantly reduces the time required to train a model on new data, as shown in Figure 6.6, and augments the model's knowledge with new data batches. Besides, establishing the ideal batch size configuration that delivers the lowest prediction accuracy deviation while reducing model training time is an under-explored topic in the existing literature but is valuable to industry practitioners.

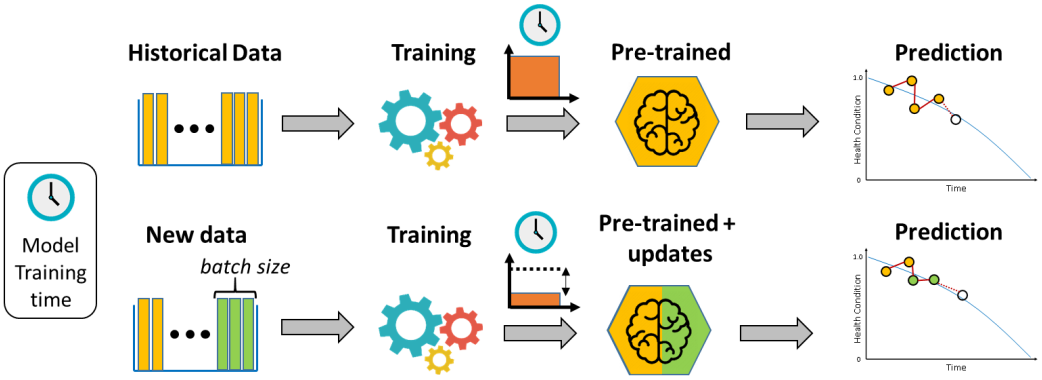


FIGURE 6.6: An incremental learning strategy which enriches a pre-trained model's knowledge with new data.

6.4 Experiments

Within this section, we evaluate the proposed AMBFNet model on two real-world datasets and conduct extensive experiments to demonstrate the efficacy and generalization of our approach. Each dataset is described briefly in its own sub-section and is subjected to the same pre-processing and evaluation metrics.

6.4.1 Dataset Description

6.4.1.1 C-MAPSS Dataset

NASA produced the Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) dataset [6] to simulate turbofan engine deterioration. Based on operating circumstances and failure modes, we summarize the four distinct data subsets in Table 6.2. For example, the FD001 subset contains 21 sensor measurements, fault modes, and operating conditions. Sensors such as speed, pressure, and temperature data are necessary to monitor each subset's turbofan engine condition. Then, various turbofan engine data are divided into training and test sequences to facilitate verification. Furthermore, the diverse wear rate and manufacturing variances make RUL prediction challenging. Besides, the dataset's objective is to perform RUL prediction for any turbofan engine in the test dataset using raw sensor data, and Figure 6.7 depicts an abstract representation of the turbofan engine with detailed C-MAPSS information in [4].

Not all sensor measurements offer useful information for estimating RUL, sensor values that stay constant over time can be disregarded. For example, we eliminated

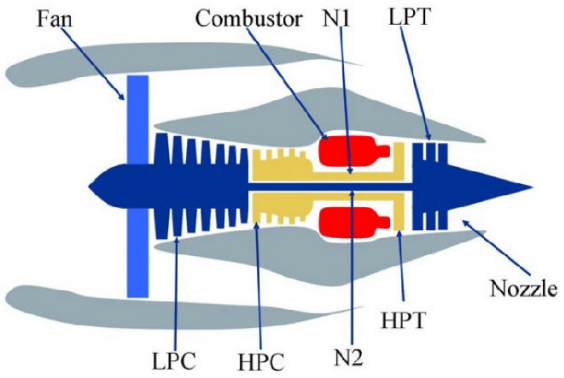


FIGURE 6.7: Abstract representation of turbofan engine[4].

Dataset	FD001	FD002	FD003	FD004
Training Sequences	100	260	100	249
Testing Sequences	100	259	248	100
Operating Conditions	1	6	1	6
Fault Conditions	1	1	2	2

TABLE 6.2: C-MAPSS Dataset [6] details.

sensors $[1, 5, 6, 10, 16, 18, 19]$ and *setting3* from FD001 and FD003, as initially proposed in [45]. However, we consider all FD002 and FD004 data for model training. Lastly, we identify and augment existing dataset values for RUL prediction using the proposed handcrafted features in all C-MAPSS subsets.

6.4.1.2 IWM Dataset

The dataset comprises machine data collected from an automotive manufacturing plant in China. The machine of interest is an industrial washing machine (IWM) responsible for cleaning manufactured automotive parts with various failure types and operating conditions summarized in Table 6.3. According to maintenance log data, we identify eleven different failure types across three different machine operating conditions: idle, busy, and failure. Sensor data from each machine, such as pump speed, nozzle pressure, and pump gear, are critical input parameters that the in-situ machine controller processes prior to actuating the pressure regulator valve (i.e., drv valve) seen in Figure 6.8. Note that the pressure regulator valve maintains the output nozzle pressure within the machine manufacturer's specifications. Through exploratory data analysis, feature engineering, and feature selection, we derive the lifecycle and RUL information for the IWM dataset. Overall, nine sensor features are available, with the dataset split into training and test, respectively.

Dataset	Set 1
Machines (Train)	13
Machines (Test)	13
Operating Conditions	3
Fault Conditions	11

TABLE 6.3: IWM dataset details.

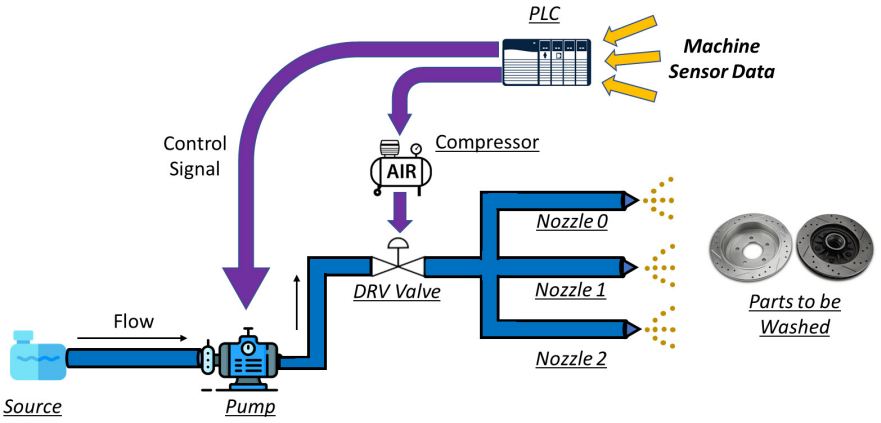


FIGURE 6.8: Abstract representation of the IWM system.

We made extra efforts to ensure an even distribution of failure events across both training (X_{train}) and test (X_{test}) datasets with limited failure data.

6.4.2 Data Pre-processing

Given that identical sensors provide different values under various operating conditions, we normalize sensor data using min-max normalization to $[0, 1]$. Consequently, model convergence is accelerated, and the effects of different operating conditions are minimized. For model training purposes, each data subset (e.g., FD001) is split into training and test sets as 80% and 20%, respectively.

A sliding window of size L is considered to segment the data according to step-size q, see Figure 6.9. Given the run-to-failure characteristics, the sliding window moves with q steps from the initial cycle to the final cycle, and T denotes the total operating life-cycle of each machine or engine. Following this, the window sample size over $n \in \{1, \ldots, N\}$ sensors is $L \times n$. Conversely, only the last data segment window size is used for each machine (i.e., washing machine or turbofan engine) in the X_{test} dataset.

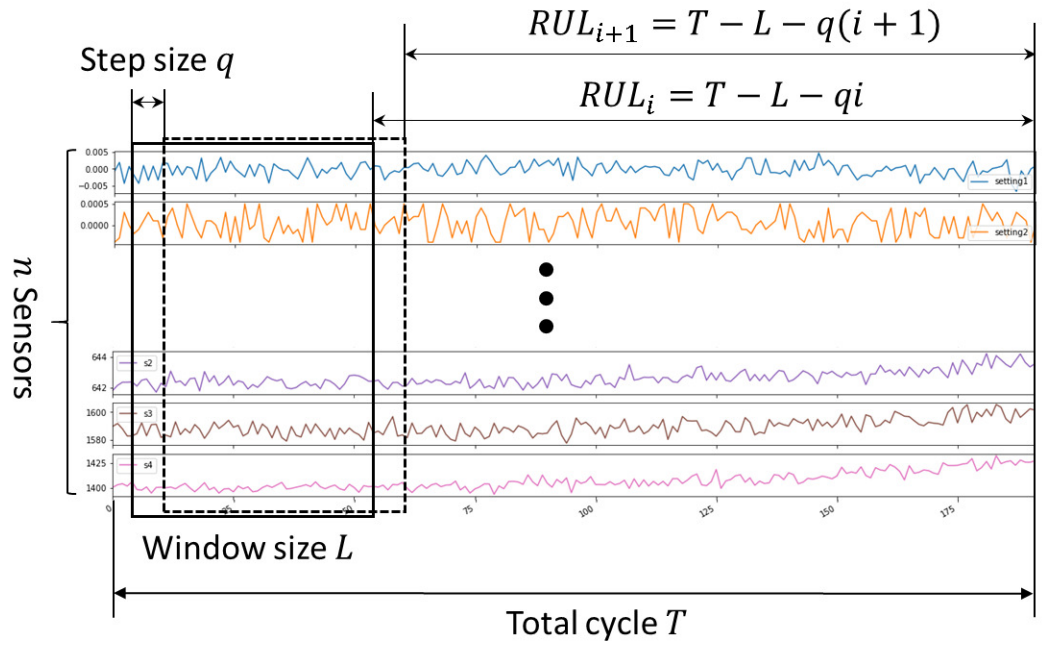


FIGURE 6.9: Sliding window across multiple sensors (i.e., features).

The RUL for the initial sample window i and $(i+1)$ is shown in Figure 6.9. Notably, the initial degradation of the machine is inconsequential as more consideration should be given towards the end of the machine's operation cycle. The assumption is reasonable in practice and investigated in previous works [156]. Moreover, the adoption of piece-wise linear RUL [157] improves the model's RUL prediction performance [53, 55, 158] and is formally described in (6.16). In this work, we configure $\text{RUL}_{\text{max}} = 125$, $L = 90$, and $q = 1$. In addition, we investigate the effects of $L \in [30, 50, 70, 90, 100]$ to illustrate the effects of a sliding window on different datasets.

$$RUL = \begin{cases} T - L - qi, & \text{if} \quad \text{RUL} < \text{RUL}_{\text{max}}, \\ \text{RUL}_{\text{max}}, & \text{Otherwise.} \end{cases} \tag{6.16}$$

Meanwhile, labeled data and the ground truth are not always available in practice. Likewise, determining a threshold for automatic triggering of incremental model training is challenging. In this regard, we recommend dedicating 50% of the respective datasets to emulate fresh data and the remaining 50% to initial model training. For instance, we randomly select fifty C-MAPSS engines for initial model training before subjecting the remaining turbofan engines to incremental learning. Additionally, we examine the effects of performance deviation and training time savings associated with new data batch sizes for C-MAPSS and IWM datasets, respectively.

Due to AMBFNet's unique model design, the data pre-processing step for both X_{test} and X_{train} requires re-shaping such that N branches correspond to N sensors. To evaluate the performance benefits of our approach, we perform a model ablation study. Another benefit of our proposed architecture is its design flexibility to diverse sensor input combinations.

6.4.3 Evaluation Metrics

We evaluate our prediction models using well-recognized prognostic metrics of root-mean-square-error (RMSE) and score function [6]. From (6.17), the RMSE quantifies the cumulative errors in model predictions, where N denotes the total number of samples; $\hat{y}_i$ and y_i denote the predicted RUL and true RUL values, respectively. On the contrary, the score function in (6.18) intends to penalize late model predictions (i.e., $\hat{y}_i > y_i$), which may result in severe failure consequences practically. This assumption is vital for timely maintenance actions based on accurate RUL predictions. Specifically, the lower the RMSE and score function values, the higher the prediction accuracy.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}, \tag{6.17}$$

$$Score = \begin{cases} \sum_{i=1}^{N}\left(e^{\frac{\hat{y}_i - y_i}{13}} - 1\right), & \text{if} \quad \hat{y}_i < y_i, \\ \sum_{i=1}^{N}\left(e^{\frac{\hat{y}_i - y_i}{10}} - 1\right), & \text{Otherwise.} \end{cases} \tag{6.18}$$

6.4.4 Experiment Setup

To evaluate our proposed AMBFNet approach, we conduct an initial test using the FD001 training dataset before validation on the test dataset. Then, we assess the model performance against comparable LSTM-based state-of-the-art (SOTA) models such as LSTM, LSTM-MLSA, and LSTM-Attn. Based on the results obtained, fairly extensive fine-tuning is necessary to extract the maximum performance for the respective datasets and subsets. Note that all benchmarks utilize identical training and test datasets.

AMBFNet features multiple branches of sequence modeling block. Each sequence modeling block (i.e., *branch*) comprises of single-layer LSTM model with 100 hidden units for sequence modeling, and data is piped to the attention block, comprising

64 hidden units, and prioritizes essential time-series features. Individual *branch* data are merged and processed by two fully-connected layers of node sizes 50 and 1, respectively. For more efficient learning, we included batch normalization and utilized *EarlyStopping* and *ModelCheckpoint* methods to prevent model over-fitting. The Adam optimizer learning rate is 0.00003 and is trained for up to 200 epochs. A batch size of 512 is set to help stabilize model performance while speeding up the model training process.

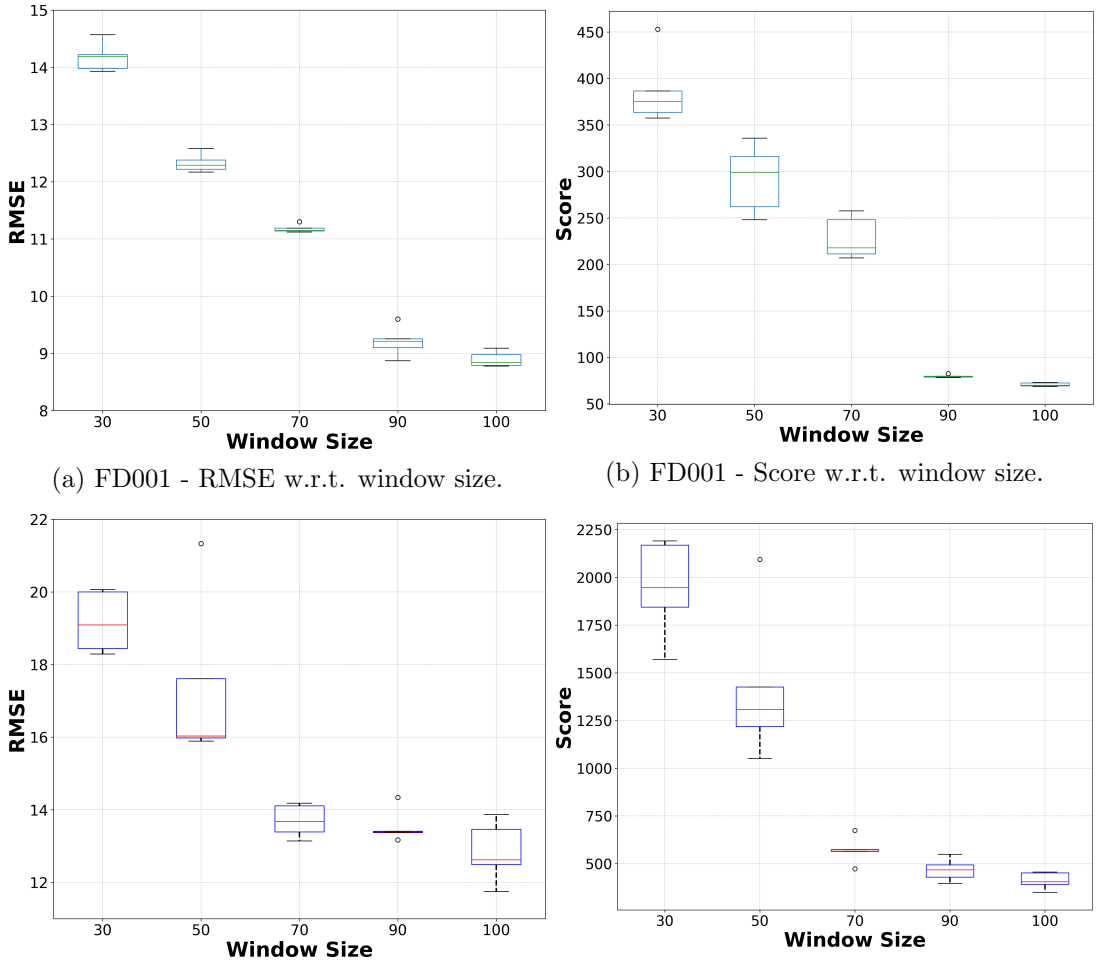
According to Table 6.2, the FD001 and FD003 subsets' operating conditions are simpler than FD002 and FD004. Surprisingly, our independent exploratory data analysis reveals that operating conditions may vary across time-steps rather than between different engines within the C-MAPSS dataset, as previously hypothesized. Therefore, we perform an additional benchmark for AMBFNet to compare the potential performance difference between normalizing data across all sensors versus individual operating conditions. Finally, we compare AMBFNet's performance on the IWM and C-MAPSS dataset against popular machine learning approaches such as convolutional neural networks (CNN), LSTM, and multi-layer perceptron (MLP). Note that the reported experiment results are averaged over five independent runs to account for the stochastic nature of neural networks.

6.5 Discussion and Results

6.5.1 C-MAPSS Dataset

6.5.1.1 Window Size Analysis

The prediction model's performance is often dataset-dependent and highly influenced by the choice of window size. We investigate AMBFNet over a wide range of window sizes (i.e., L), with intriguing results in Figure 6.10. Take FD001 for example, both RMSE and score performance improves significantly for $70 \leq L \leq 100$, hitting a performance plateau for $L > 100$. Unlike [55], the batch normalization layers minimize model over-fitting and enable AMBFNet to surpass SOTA models. Contrary to the suggestions of [50] and [56], which assert that performance degrades for $L > 30$, Figure 6.10 clearly illustrates the benefits of larger window sizes. Considering the prediction model's objective is to achieve low RMSE and score values, we effect $L = 90$ across C-MAPSS subsets.



(a) FD001 - RMSE w.r.t. window size.

(b) FD001 - Score w.r.t. window size.

(c) FD002 - RMSE w.r.t. window size.

(d) FD002 - Score w.r.t. window size.

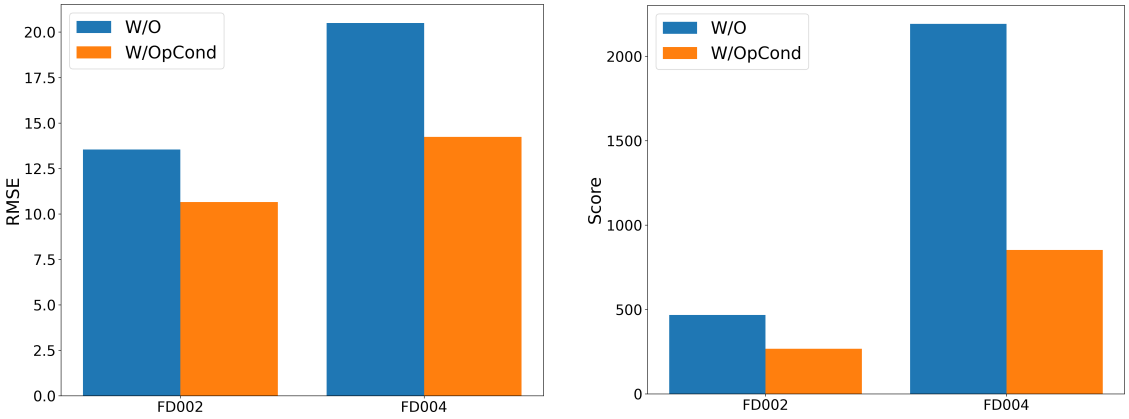
FIGURE 6.10: Window size performance evaluation (lower is better) on C-MAPSS subsets FD001 (a, b) and FD002 (c, d).

6.5.1.2 Model Ablation Study

We illustrate the efficacy of our proposed approach sequentially, from LSTM to AMBFNet, and summarize the findings in Table 6.4 for the reader's convenience. In addition, the hidden units of the LSTM network remain constant throughout the ablation study. During the model training process, the attention mechanism uses dynamic feature weighting to help the prediction model focus on vital degradation features (i.e., sensor data). Notably, the multi-branch and feature crafting mechanism is shown to improve prediction performance significantly. For example, combining attention with multi-branch mechanisms results in a 35% and 80% improvement in RMSE and score, respectively. Similar increases are observable for FD002, with up to a 95% increase in score results (i.e., higher prediction accuracy towards the end of the machine cycle). However, adding handcrafted features

Methods	FD001		FD002	
	RMSE	**Score**	**RMSE**	**Score**
LSTM	14.15	212.12	35.60	8947.50
LSTM + Single Attention Feature Net (AFNet)	13.98	244.78	14.29	644.16
Multi-branch AFNet (AMBFNet)	9.21	79.76	13.54	467.06
AMBFNet + Handcraft.	9.29	77.05	14.22	530.44

TABLE 6.4: Model Ablation Study.



(a) RMSE. (b) Score.

FIGURE 6.11: Comparison of operating condition-based data normalization for FD002 and FD004 with AMBFNet.

makes the prediction model less accurate, which we ascribe to the various operating conditions. Hence, the omission of handcrafted features for FD002 and FD004 subsets. Lastly, the attention mechanism significantly improves RUL prediction, especially when processing the relatively complex FD002 subset.

6.5.1.3 Sensor normalization for FD002 and FD004

We note that [113] and [159] achieve SOTA results with brief mentions of such normalization methods without further explanation. In addition, the performance benefits arising from such normalization is not discussed. To the best of our knowledge, we are the first to highlight the potential performance benefits and summarize our empirical results in Figure 6.11. For instance, we observe RMSE result improvements of between 21% and 31% in Figure 6.11a. Similarly, Figure 6.11b illustrates impressive score improvements of between 42% and 61% for the FD002 and FD004 datasets. Hence, we use the condition-based normalization results for comparison in Section 6.5.1.4.

6.5.1.4 Comparison with State-of-the-Art Methods

We compare our AMBFNet model against recent SOTA results and conventional deep-learning and machine learning approaches. According to Tables 6.5 and 6.6, the efficacy of our approach is very promising in all subsets except FD004. When evaluating the easier FD001 and FD003 subsets, the RMSE improvements over SOTA results are around 20%. Similarly, the score improvements are 65% and 50%, respectively. On the complex FD002 and FD004 subsets, the benefits of operating condition-based data normalization are evident, with reported improvements of 24% and 70% for RMSE and score, respectively. Comparing our AMBFNet model to recent LSTM-based works [103] and [55], our model readily outperforms their implementation without requiring hand-crafted features or rigorous hyper-parameter tuning. Additional comparisons to the transformer encoder-based approach in [160] show that our AMBFNet's RMSE scores are 32% more accurate.

TABLE 6.5: RMSE comparison with existing works on C-MAPSS

	Machine Learning		Deep Learning				Hybrid				
	MLP [49]	RF [45]	CNN [49]	LSTM [53]	DCNN [50]	AutoEncoder [51]	LSTM-MLSA [103]	KDnet-RUL [113]	AGCNN [56]	LSTM-Attn + Handcraft [55]	**Proposed**
FD001	37.56	17.91	18.45	16.14	12.61	13.58	11.57*	13.68	12.42	14.53	**9.29**
FD002	80.03	29.59	30.29	24.49	22.36	19.59	14.02*	14.47	19.43	-	**10.66**
FD003	42.00	20.27	19.82	16.18	12.64	19.16	12.13*	12.95	13.39	-	**9.68**
FD004	77.37	31.12	29.16	28.17	23.31	22.15	17.21	15.96*	21.50	27.08	**14.26**
Average	59.24	24.72	24.43	21.24	17.73	18.62	13.73*	13.73*	16.68	20.81	**11.96**

The **bold** results represent the best performance (lower is better); * denotes prior best performance; - denotes did not report.

TABLE 6.6: Score comparison with existing works on C-MAPSS (rounded to nearest number)

	Machine Learning		Deep Learning				Hybrid				
	MLP [49]	RF [45]	CNN [49]	LSTM [53]	DCNN [50]	AutoEncoder [51]	LSTM-MLSA [103]	KDnet-RUL [113]	AGCNN [56]	LSTM-Attn + Handcraft [55]	**Proposed**
FD001	17,970	480	1287	338	274	228	253	362	226*	322	**77**
FD002	780×10^4	7046	13,600	4450	10,400	2650	899*	929	1492	-	**271**
FD003	17,410	711	1596	852	284	1727	370	327	227*	-	**113**
FD004	562×10^4	4656	7886	5550	12,500	2901	1558	1303*	3392	5649	**847**
Average	336×10^4	3223	3032	1798	5864	1877	770	730*	1334	2986	**280**

The **bold** results represent the best performance (lower is better); * denotes prior best performance; - denotes did not report.

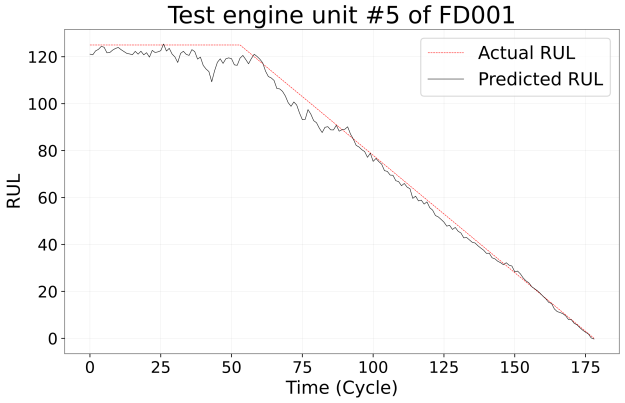
As stated in Section 6.4.3, we expect more prediction errors early in the machine degradation cycle, with increasing prediction accuracy as the degradation cycle progresses. For the reader's reference, we illustrate the described prognostics behavior, across all C-MAPSS subsets, in Figure 6.12, with degradation signals becoming more pronounced as time progresses. Finally, we anticipate that untapped performance gains are unlockable via comprehensive hyper-parameter tweaking using grid search or bayesian optimization strategies.

6.5.1.5 Incremental Learning

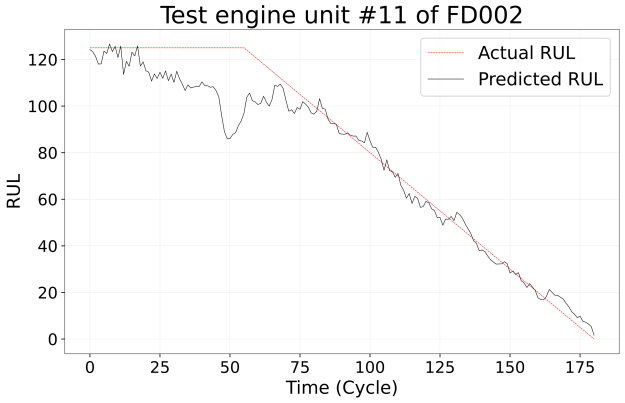
Model drift in prediction performance necessitates continuous monitoring before choosing whether to execute incremental learning or request an updated model from the cloud for resource-constrained edge computing devices. Our experiments empirically investigate the ideal data batch size configuration for incremental learning. According to Table 6.7, batch = 40 is the best configuration for FD001 and FD003. Likewise, we observe favorable RMSE and score outcomes in FD002 and FD004. In the case of FD004, where batches of new machine data significantly overlap the existing model's knowledge distribution, the model's performance (i.e., RMSE and score) improves over the tabula rasa method and incurs significant training time savings. Regarding incremental learning frequency, we recommend comparing incremental learning and tabula rasa models weekly before determining if the incremental learning model requires updating on the edge-computing device. In summary, incremental training yields an average training time savings of 60% across all NASA C-MAPSS subsets. In contrast to the baseline of tabula rasa

Dataset	New machine data batch size	RMSE Diff (%)	Score Diff (%)	Trng Time Savings (%)
FD001	40	-8.22	-10.13	61.0
	50	-2.54	-3.73	59.5
FD003	40	-4.43	-8.84	54.6
	50	-4.76	-4.80	49.5
FD002	40	-2.49	6.3	61.1
	50	-0.76	-6.36	56.6
FD004	40	5.36	7.11	68.0
	50	0.36	-28.91	62.6

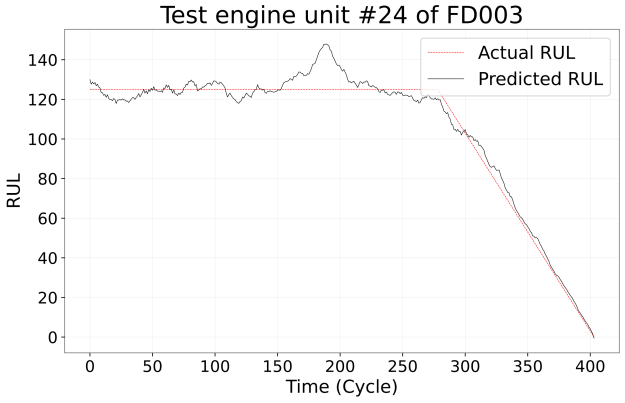
TABLE 6.7: Incremental learning - prediction accuracy vs training time savings compared to baseline. Data batch size refers to the quantity of newly added machines.



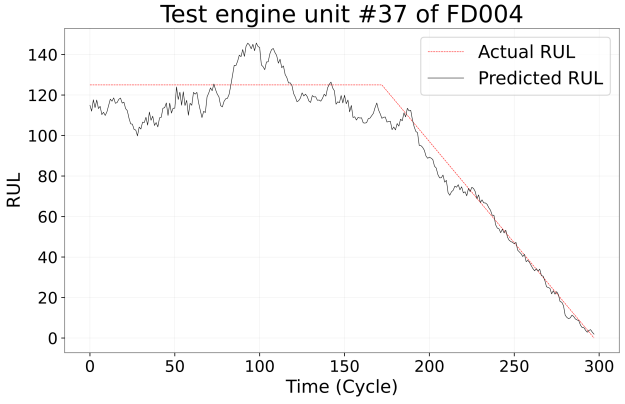
(a) FD001.



(b) FD002.



(c) FD003.



(d) FD004.

FIGURE 6.12: Actual and predicted RUL values for random engines on the FD001-FD004 data subsets using AMBFNet.

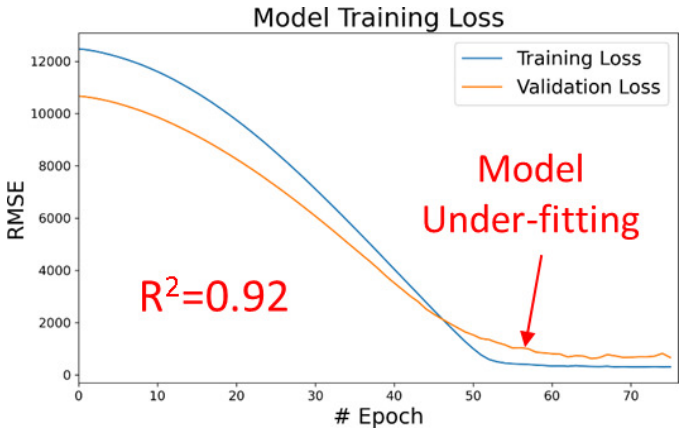


FIGURE 6.13: Model loss results for the IWM dataset.

training, we observe a mean difference of -1.1% and -4.9% for RMSE and score, respectively (i.e., batch $= 40$). Finally, owing to the limited data in FD001 and FD003, the maximum batch$=50$ for all NASA C-MAPSS subsets.

6.5.2 IWM Dataset

We empirically determined the sequence length as $L = 40$ and RUL clipping values at $18,000$ cycles. AMBFNet's prediction performance is 6.7% better than vanilla LSTM networks despite its large RMSE result. Further investigation indicates that despite achieving an R^2 model fitting score of 0.92,[2] the model loss results in Figure 6.13 clearly exhibit classic model under-fitting characteristics.

In retrospect, the IWM dataset has 10 failure instances, whereas C-MAPSS subsets have between 100 and 260 failure instances. Due to the limited failure instances, we resort to manual data augmentation to artificially increase the failure instances, with mediocre RMSE improvement. By interpreting these results with further analysis of the model training loss in Figure 6.13, it becomes more apparent that the training dataset may have too few examples relative to the validation dataset, since there is a significant loss gap after epoch 50. Hence, there are future plans to investigate methods to address this issue algorithmically and improve the dataset data quality. To summarize this sub-section chapter, we attribute the low AMBFNet prediction performance to a lack of failure events and the need for improved data quality.

[2]Note that a high R^2 value implies robust model fit on the selected dataset.

6.6 Summary

Data-driven PdM continues to be a critical area of research, aiming to minimize unscheduled machine disruptions to production schedules. This chapter outlines an end-to-end deep learning model, AMBFNet, for autonomous feature extraction and RUL estimation on two industrial datasets. Specifically, we consider PdM a multi-variate regression-based problem to predict RUL for heterogeneous industrial machines. We then perform feature engineering to improve the feature signal-to-noise ratio for AMBFNet, with prediction results exceeding state-of-the-art PdM models. Furthermore, we empirically demonstrate that incremental learning yields significant time savings than model training from scratch. Likewise, we conducted extensive experiments to establish the best possible batch-sized data configuration to achieve comparable prediction performance to model training from scratch. Besides, our solution is automated and scalable for regression-based PdM problems and has been unexplored in current literature until now. Finally, AMBFNet outperforms vanilla LSTM networks despite limited failure data from the IWM dataset, and further investigations support the notion of model under-fitting. Chapter 7 provides several suggestions to overcome these problems in future works.

Chapter 7

Conclusion and Future Work

In this chapter, we first review the thesis's contributions. We then identify prospective research directions that can be pursued using the PdM framework described.

7.1 Conclusion

The IIoT is essential for manufacturers to remain competitive and offers real-time insights into their business operations for resource optimization via increased sensorization of equipment. In particular, this thesis exploits the differences between PdM and RxM as the overall research motivation, presenting a PdM framework for automated data analysis and the joint optimization of heterogeneous resources, such as man and machine. We provide a summary of the significant contributions of this thesis in the following paragraphs:

- *Chapter 4:* We presented a DRL framework for an edge computing-based predictive maintenance model to effectively manage the dynamic decision-making process involving equipment maintenance, maintenance cost model, and manpower resource. We formulated the complex resource management as a DRL problem for learning an optimal decision policy given a stochastic environment and time-series data. We evaluated the performance of the proposed DRL algorithm using a maintenance repair simulator and compared the findings against those of human participants. The simulation results verify the efficacy of our framework and our DRL-based approach in addressing the challenging maintenance resource management problem, outperforming both

human participants and the comparable baselines in terms of convergence rate and performance.

- *Chapter 5:* Data-driven resource management is critical in the Industry 4.0 context for decision support to maintenance teams to minimize interruptions to production schedules. We propose a two-stage TL-based DRL approach for an edge-based machine group of predictive maintenance models. We begin by framing the machine maintenance request events as a task optimization problem based on the predictive model data. Following that, we formulate the sequence of maintenance request tasks as a sequential decision-making problem using the Markov Decision Process to learn an optimal decision policy and enable effective manpower resource management. We can inherently accelerate the DRL model's learning performance by enabling the proposed DRL model to harness relevant experiences from similar machine configurations. Empirically, our experiments demonstrate that coupling TL with the two-stage PdM framework significantly improves learning efficiency and performance over baseline methods.

- *Chapter 6:* The ability to accurately anticipate a machine's RUL is crucial for reducing unexpected maintenance expenses. We present an attention-based LSTM approach (i.e., AMBFNet) for RUL prediction to cope with the complexities of multi-variate time-series data automatically. Specifically, the attention mechanism automatically learns salient data characteristics while the LSTM network focuses on sequential time-series features. Furthermore, we propose simple handcrafted features to increase RUL model prediction performance. Based on empirical findings on real-world datasets, the AMBFNet model outperforms baseline methods by a minimum margin of 24% and existing state-of-the-art approaches by up to 70% on challenging open-sourced real-world datasets. Besides proposing AMBFNet, we propose an incremental learning-based approach to mitigate poor RUL model prediction performance drift on resource-constraint edge computing devices. We identified the optimal batch-sized data configuration to achieve comparable RUL prediction performance to tabula rasa model training via extensive experiments with significant time and cost savings. Besides, our solution is automated and scalable for regression-based predictive maintenance problems.

7.2 Future Work

7.2.1 Enabling Prescriptive Maintenance through advanced recommendation systems

Existing PdM approaches focus solely on improving machine reliability and assume disentanglement from real-world production planning and scheduling, delaying PdM adoption. Moreover, current PdM solutions offer limited to no recommendations for maintenance actions, which is an essential step for attaining RxM [10]. We hypothesize similar benefits for PdM applications via advanced recommendation systems, considering the successful implications of DL-based recommendation in real-world applications. Specifically, advanced recommendation systems will holistically consider factory-wide resources and production planning parameters. Examples of resource parameters include machine, human, and maintenance part inventory. Likewise, what-if analysis, production scheduling, and warehouse inventory management are examples of production planning parameters. Therefore, we anticipate that PdM-based advanced recommendation systems can offer online decision support to key manufacturing decision-makers and, when required, dig deeper into the specifics. Consequently, decision-makers can swiftly and confidently make better-informed decisions.

7.2.2 Effective Knowledge Transfer Methods for Reinforcement Learning

Despite the plethora of advances in TL and DRL research, the adoption of a hybrid framework of TL and DRL methodologies for PdM applications [39] was absent prior to our study [18]. Nonetheless, knowledge advancements in hybrid frameworks (e.g., TLD) can further accelerate PdM adoption in practice. Future works on the interpretability of hybrid models, especially explainable representations of DRL, are necessary to provide macro-level details to give decision-makers confidence. Besides, approaches for quantifying task similarity to optimize knowledge transfer methods in IIoT-enabled PdM problems are scarce. Likewise, the effects of knowledge distillation on improving DRL's learning efficiency is a potential research problem. Lastly, the impact of offline-to-online reinforcement learning on diverging task scenarios and learning convergence performance remains an open research challenge and is an avenue for future work.

7.2.3 Time-series Data Augmentation methods

Data-driven PdM approaches require significant learning samples to achieve great prediction results. Inadvertently, failure and machine deterioration events are infrequent, and assembling a representative dataset is time-consuming. Interestingly, our first-hand experience with the BOSCH industrial dataset concurs with the aforementioned phenomena. A data augmentation strategy for time-series data has recently been suggested to enhance prognostic outcomes but is not actively investigated in the prognostics literature [161]. Therefore, signaling the open research challenge in time-series data augmentation techniques for general PdM applications.

List of Author's Awards, Patents, and Publications

Journal Articles

Articles Related to Thesis

- **KSH. Ong**, W. Wang, D. Niyato, NQ. Hieu, and T. Friedrichs, "Predictive Maintenance Model for IIoT-based Manufacturing: A Transferable Deep Reinforcement Learning Approach", **Accepted by** IEEE Internet of Things Journal, pp. 1-1, Feb 16 2022.
- **KSH. Ong**, W. Wang, D. Niyato and T. Friedrichs, "Deep Reinforcement Learning Based Predictive Maintenance Model for Effective Resource Management in Industrial IoT", **Published in** IEEE Internet of Things Journal, vol. 9, no. 7, pp. 5173-5188, April 1 2022.

Articles not Related to Thesis

None.

Conference Proceedings

Articles Related to Thesis

- **KSH. Ong**, W. Wang, T. Friedrichs and D. Niyato, "Augmented Human Intelligence for Decision Making in Maintenance Risk Taking Tasks using Reinforcement Learning", **Published in** 2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC) 2021 Oct 17 (pp. 3114-3120).

- **KSH. Ong**, D. Niyato, and C. Yuen, "Predictive Maintenance for Edge-Based Sensor Networks: A Deep Reinforcement Learning Approach", **Published in** 2020 IEEE 6th World Forum on Internet of Things (WF-IoT) 2020 Jun 2 (pp. 1-6).
- **KSH. Ong**, D. Niyato, and T. Friedrichs, "Drift-Aware Edge Intelligence for Remaining Useful Life Prediction in IIoT", **Accepted by** 37th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC), May 2022.

Articles not Related to Thesis

- **KSH. Ong**, D. Niyato, C. So-In, and T. Friedrichs, "Accelerating Low-Cost Edge-Based Real-Time Video Analytics Using Task Scheduling," **Published in** 2021 IEEE 7th World Forum on Internet of Things (WF-IoT) 2021 Jun 14 (pp. 598-603).
- **KSH. Ong**, Y. Zhang, and D. Niyato, "Cognitive Radio Network Throughput Maximization with Deep Reinforcement Learning," **Published in** 2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall) 2019 Sep 22 (pp. 1-5).

Technical Report

Articles Related to Thesis

- **KSH. Ong**, D. Niyato, and T. Friedrichs, "Remaining Useful Life Prediction of IIoT-Enabled Equipment using Attentive Multi-Branch Feature Network", **Published in BOSCH internal**.

Articles not Related to Thesis

None.

Bibliography

[1] Hua Xiang, Peilin Jiang, Shuang Xiao, Fuji Ren, and Shingo Kuroiwa. A model of mental state transition network. *IEEJ Trans. on Electronics, Information and Systems*, 127(3):434–442, 2007. xvi, 62

[2] Prasetio Barlian Henryranu, Tamura Hiroki, and Tanno Koichi. Deep time-delay markov network for prediction and modeling the stress and emotions state transition. *Scientific Reports (Nature Publisher Group)*, 10(1), 2020. xvi, 62

[3] Moritz Sudhof, Andrés Goméz Emilsson, Andrew L Maas, and Christopher Potts. Sentiment expression conditioned by affective transitions and social forces. In *Proceedings of the 20th ACM SIGKDD*, pages 1136–1145, 2014. xvi, 62

[4] A Saxena and K Goebel. Phm08 challenge data set. *NASA Ames Prognostics Data Repository*, 2008. xvii, 130, 131

[5] Marián Kučera, Silvia Kopčanová, and Marie Sejkorová. Lubricant analysis as the most useful tool in the proactive maintenance philosophies of machinery and its components. *Management Systems in Production Engineering*, 2020. xix, 3

[6] Abhinav Saxena, Kai Goebel, Don Simon, and Neil Eklund. Damage propagation modeling for aircraft engine run-to-failure simulation. In *2008 ICPHM*, pages 1–9. IEEE, 2008. xix, 32, 33, 40, 55, 74, 91, 130, 131, 134

[7] R Keith Mobley. *An introduction to predictive maintenance*. Elsevier, 2002. 1

[8] Ravikiran Gaikwad. Predictive maintenance market research analysis and forecasts to 2026. `https://www.linkedin.com/pulse/predictive-maintenance-market-research-report-analysis-gaikwad`, 2022. Accessed: 2022-04-01. 1

[9] Hashem M Hashemian. State-of-the-art predictive maintenance techniques. *IEEE Transactions on Instrumentation and measurement*, 60(1):226–236, 2010. 3

[10] Yongyi Ran, Xin Zhou, Pengfeng Lin, Yonggang Wen, and Ruilong Deng. A survey of predictive maintenance: Systems, purposes and approaches. *arXiv preprint arXiv:1912.07383*, 2019. 3, 6, 8, 10, 13, 35, 36, 40, 147

[11] Tangbin Xia, Yifan Dong, Lei Xiao, Shichang Du, Ershun Pan, and Lifeng Xi. Recent advances in prognostics and health management for advanced manufacturing paradigms. *Reliability Engineering & System Safety*, 178:255–268, 2018. 3

[12] Sheila Kennedy. Rxm: What is prescriptive maintenance, and how soon will you need it? `https://www.plantservices.com/articles/2017/rxm-what-is-prescriptive-maintenance-and-how-soon-will-you-need-it/`, 2016. Accessed: 2022-04-01. 4

[13] Tanja Nemeth, Fazel Ansari, Wilfried Sihn, Bernhard Haslhofer, and Alexander Schindler. Prima-x: A reference model for realizing prescriptive maintenance and assessing its maturity enhanced by machine learning. *Procedia CIRP*, 72:1039–1044, 2018. 4

[14] Michele Compare, Piero Baraldi, and Enrico Zio. Challenges to iot-enabled predictive maintenance for industry 4.0. *IEEE Internet Things J.*, 7(5):4585–4597, 2019. 4, 6, 8, 10, 13, 36, 45

[15] Shaohua Huang, Yu Guo, Daoyuan Liu, Shanshan Zha, and Weiguang Fang. A two-stage transfer learning-based deep learning approach for production progress prediction in iot-enabled manufacturing. *IEEE Internet Things J.*, 6(6):10627–10638, 2019. 4, 42, 44

[16] Bas Weber. Predict the unpredictable. `https://www.pwc.nl/nl/assets/documents/pwc-predictive-maintenance-4-0.pdf`, 2017. Accessed: 2020-08-04. 5, 52

[17] ServicePower. Preventive vs predictive maintenance. `https://https://www.servicepower.com/hubfs/Addressing%20the%20aging%20workforce_infographic.pdf`, 2020. Accessed: 2020-08-04. 5

[18] Kevin Shen Hoong Ong, Wenbo Wang, Nguyen Quang Hieu, Dusit Niyato, and Thomas Friedrichs. Predictive maintenance model for iiot-based manufacturing: A transferable deep reinforcement learning approach. *IEEE Internet of Things Journal*, pages 1–1, 2022. doi: 10.1109/JIOT.2022.3151862. 6, 10, 12, 14, 44, 46, 83, 147

[19] Weiting Zhang, Dong Yang, and Hongchao Wang. Data-driven methods for predictive maintenance of industrial equipment: a survey. *IEEE Systems Journal*, 13(3):2213–2227, 2019. 6

[20] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014. 6

[21] Beomsoo Kim, Jang-Ho Choi, and Jaegul Choo. Augmenting imbalanced time-series data via adversarial perturbation in latent space. In *Asian Conference on Machine Learning*, pages 1633–1644. PMLR, 2021. 6

[22] Jun Zhu, Nan Chen, and Weiwen Peng. Estimation of bearing remaining useful life based on multiscale convolutional neural network. *IEEE Transactions on Industrial Electronics*, 66(4):3208–3216, 2018. 7, 37

[23] Long Wen, Yan Dong, and Liang Gao. A new ensemble residual convolutional neural network for remaining useful life estimation. *Math. Biosci. Eng*, 16 (2):862–880, 2019. 7, 37

[24] Kevin Shen Hoong Ong, Wang Wenbo, Dusit Niyato, and Thomas Friedrichs. Deep reinforcement learning based predictive maintenance model for effective resource management in industrial iot. *IEEE Internet Things J.*, 9(7):5173–5188, 2022. doi: 10.1109/JIOT.2021.3109955. 8, 10, 14, 44, 45, 49, 91, 92

[25] Kevin Shen Hoong Ong, Wang Wenbo, Thomas Friedrichs, and Dusit Niyato. Augmented human intelligence for decision making in maintenance risk taking tasks using reinforcement learning. In *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 3114–3120. IEEE, 2021. 8, 9, 45, 49, 73

[26] Kevin Shen Hoong Ong, Dusit Niyato, and Chau Yuen. Predictive maintenance for edge-based sensor networks: A deep reinforcement learning approach. In *2020 IEEE 6th WF-IoT*, pages 1–6. IEEE, 2020. 8, 14, 33, 40, 41, 49, 52, 53, 75, 81

[27] Guansong Pang, Chunhua Shen, Longbing Cao, and Anton Van Den Hengel. Deep learning for anomaly detection: A review. *ACM Computing Surveys (CSUR)*, 54(2):1–38, 2021. 8

[28] Pengfei Wen, Yong Li, Shaowei Chen, and Shuai Zhao. Remaining useful life prediction of iiot-enabled complex industrial systems with hybrid fusion of multiple information sources. *IEEE Internet Things J.*, 8(11):9045–9058, 2021. 42

[29] Liang Guo, Naipeng Li, Feng Jia, Yaguo Lei, and Jing Lin. A recurrent neural network based health indicator for remaining useful life prediction of bearings. *Neurocomputing*, 240:98–109, 2017.

[30] Pengfei Lin and Jizhong Tao. A novel bearing health indicator construction method based on ensemble stacked autoencoder. In *2019 ICPHM*, pages 1–9. IEEE, 2019. 8, 37

[31] Coralie Martinez, Guillaume Perrin, Emmanuel Ramasso, and Michèle Rombaut. A deep reinforcement learning approach for early classification of time series. In *2018 26th EUSIPCO*, pages 2030–2034. IEEE, 2018. 8, 40, 44

[32] Yu Ding, Liang Ma, Jian Ma, Mingliang Suo, Laifa Tao, Yujie Cheng, and Chen Lu. Intelligent fault diagnosis for rotating machinery using deep q-network based health state classification: A deep reinforcement learning approach. *Advanced Engineering Informatics*, 42:100977, 2019. 8, 40, 44

[33] Chi Zhang, Chetan Gupta, Ahmed Farahat, Kosta Ristovski, and Dipanjan Ghosh. Equipment health indicator learning using deep reinforcement learning. In *Joint ECML PKDD*, pages 488–504. Springer, 2018. 8, 40, 44, 55

[34] Jere Backman, Janne Väre, Kary Främling, Manik Madhikermi, and Ossi Nykänen. Iot-based interoperability framework for asset and fleet management. In *2016 21st Int. Conf. on ETFA*, pages 1–4. IEEE, 2016. 9, 43, 44

[35] Yyi Kai Teoh, Sukhpal Singh Gill, and Ajith Kumar Parlikad. Iot and fog computing based predictive maintenance model for effective asset management in industry 4.0 using machine learning. *IEEE Internet Things J.*, 2021. 9, 10, 43, 44, 45

[36] Thyago P Carvalho, Fabrízzio AAMN Soares, Roberto Vita, Roberto da P Francisco, João P Basto, and Symone GS Alcalá. A systematic literature review of machine learning methods applied to predictive maintenance. *Comput Ind Eng.*, 137:106024, 2019. 10, 13, 35

[37] Zeyi Sun, Fadwa Dababneh, and Lin Li. Joint energy, maintenance, and throughput modeling for sustainable manufacturing systems. *IEEE Trans. Syst. Man Cybern.: Syst.*, 50(6):2101–2112, 2018. 10, 107

[38] David Gale and Lloyd S Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 120(5):386–391, 2013. 10

[39] Zhuangdi Zhu, Kaixiang Lin, and Jiayu Zhou. Transfer learning in deep reinforcement learning: A survey, 2021. 11, 32, 86, 102, 113, 116, 147

[40] Kevin Shen Hoong Ong, Dusit Niyato, and Thomas Friedrichs. Drift-aware edge intelligence for remaining useful life prediction in industrial internet of things. In *2022 37th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC)*, pages 1–1. IEEE, 2022. 11, 14, 46, 121

[41] Rodney K Singleton, Elias G Strangas, and Selin Aviyente. Extended kalman filtering for remaining-useful-life estimation of bearings. *IEEE Transactions on Industrial Electronics*, 62(3):1781–1790, 2014. 11

[42] Naipeng Li, Yaguo Lei, Liang Guo, Tao Yan, and Jing Lin. Remaining useful life prediction based on a general expression of stochastic process models. *IEEE Transactions on Industrial Electronics*, 64(7):5709–5718, 2017. 11

[43] Racha Khelif, Brigitte Chebel-Morello, Simon Malinowski, Emna Laajili, Farhat Fnaiech, and Noureddine Zerhouni. Direct remaining useful life estimation based on support vector regression. *IEEE Transactions on industrial electronics*, 64(3):2276–2285, 2016. 11, 36, 129

[44] PJ García Nieto, E García-Gonzalo, F Sánchez Lasheras, and Francisco Javier de Cos Juez. Hybrid pso–svm-based method for forecasting of the remaining useful life for aircraft engines and evaluation of its reliability. *Reliability Engineering & System Safety*, 138:219–231, 2015. 11

[45] Chong Zhang, Pin Lim, A Kai Qin, and Kay Chen Tan. Multiobjective deep belief networks ensemble for remaining useful life estimation in prognostics. *IEEE Trans. on Neural Networks and Learning Systems*, 28(10):2306–2318, 2016. 11, 38, 131, 139

[46] Mrutyunjaya Sahani and Pradipta Kishore Dash. Automatic power quality events recognition based on hilbert huang transform and weighted bidirectional extreme learning machine. *IEEE Transactions on Industrial Informatics*, 14(9):3849–3858, 2018. 11

[47] Theodoros H Loutas, Dimitrios Roulias, and George Georgoulas. Remaining useful life estimation in rolling bearings utilizing data-driven probabilistic e-support vectors regression. *IEEE Transactions on Reliability*, 62(4):821–832, 2013. 11, 36

[48] Rui Zhao, Ruqiang Yan, Zhenghua Chen, Kezhi Mao, Peng Wang, and Robert X Gao. Deep learning and its applications to machine health monitoring. *Mechanical Systems and Signal Processing*, 115:213–237, 2019. 11, 37

[49] Giduthuri Sateesh Babu, Peilin Zhao, and Xiao-Li Li. Deep convolutional neural network based regression approach for estimation of remaining useful life. In *Int. Conf. on database systems for advanced applications*, pages 214–228. Springer, 2016. 11, 36, 37, 139

[50] Xiang Li, Qian Ding, and Jian-Qiao Sun. Remaining useful life estimation in prognostics using deep convolution neural networks. *Reliability Engineering & System Safety*, 172:1–11, 2018. 11, 12, 37, 135, 139

[51] Wennian Yu, II Yong Kim, and Chris Mechefske. An improved similarity-based prognostic algorithm for rul estimation using an rnn autoencoder scheme. *Reliability Engineering & System Safety*, 199:106926, 2020. 11, 12, 139

[52] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 11, 25, 69, 123

[53] Shuai Zheng, Kosta Ristovski, Ahmed Farahat, and Chetan Gupta. Long short-term memory network for remaining useful life estimation. In *2017 ICPHM*, pages 88–95. IEEE, 2017. 11, 12, 38, 124, 133, 139

[54] Cheng-Geng Huang, Hong-Zhong Huang, and Yan-Feng Li. A bidirectional lstm prognostics method under multiple operational conditions. *IEEE Transactions on Industrial Electronics*, 66(11):8792–8802, 2019. 12, 39

[55] Zhenghua Chen, Min Wu, Rui Zhao, Feri Guretno, Ruqiang Yan, and Xiaoli Li. Machine remaining useful life prediction via an attention based deep learning approach. *IEEE Trans. on Industrial Electronics*, 2020. 12, 32, 37, 39, 124, 126, 133, 135, 138, 139

[56] Hui Liu, Zhenyu Liu, Weiqiang Jia, and Xianke Lin. Remaining useful life prediction using a novel feature-attention-based end-to-end approach. *IEEE Transactions on Industrial Informatics*, 17(2):1197–1207, 2020. 11, 39, 40, 135, 139

[57] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 12, 26, 39, 124, 126, 127

[58] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. *arXiv preprint arXiv:2201.03545*, 2022. 12

[59] Kevin Shen Hoong Ong, Dusit Niyato, Chakchai So-In, and Thomas Friedrichs. Accelerating low-cost edge-based real-time video analytics using task scheduling. In *2021 IEEE 7th World Forum on Internet of Things (WF-IoT)*, pages 598–603. IEEE, 2021. 14

[60] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86 (11):2278–2324, 1998. 21

[61] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 22

[62] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. Adaptive computation and machine learning. MIT Press, 2016. ISBN 9780262035613. URL `https://books.google.co.in/books?id=Np9SDQAAQBAJ`. 22

[63] Shiv Ram Dubey, Satish Kumar Singh, and Bidyut Baran Chaudhuri. Activation functions in deep learning: a comprehensive survey and benchmark. *Neurocomputing*, 2022. 22

[64] Marc'Aurelio Ranzato, Y-Lan Boureau, Yann Cun, et al. Sparse feature learning for deep belief networks. *Advances in neural information processing systems*, 20, 2007. 22

[65] Lei Ren, Yaqiang Sun, Jin Cui, and Lin Zhang. Bearing remaining useful life prediction based on deep autoencoder and deep neural networks. *Journal of Manufacturing Systems*, 48:71–77, 2018. 23, 38

[66] Roland Memisevic and Geoffrey E Hinton. Learning to represent spatial transformations with factored higher-order boltzmann machines. *Neural computation*, 22(6):1473–1492, 2010. 24, 25

[67] Zhuyun Chen and Weihua Li. Multisensor feature fusion for bearing fault diagnosis using sparse autoencoder and deep belief network. *IEEE Trans. on Instrumentation and Measurement*, 66(7):1693–1702, 2017. 25

[68] Meng Ma, Chuang Sun, and Xuefeng Chen. Discriminative deep belief networks with ant colony optimization for health status assessment of machine. *IEEE Transactions on Instrumentation and Measurement*, 66(12):3115–3125, 2017. 25

[69] Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990. 25

[70] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014. 25

[71] Alana de Santana Correia and Esther Luna Colombini. Attention, please! a survey of neural attention models in deep learning. *arXiv preprint arXiv:2103.16775*, 2021. 26

[72] Kyunghyun Cho, Aaron Courville, and Yoshua Bengio. Describing multimedia content using attention-based encoder-decoder networks. *IEEE Transactions on Multimedia*, 17(11):1875–1886, 2015. 26

[73] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. Recurrent models of visual attention. *Advances in neural information processing systems*, 27, 2014. 26

[74] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057. PMLR, 2015. 26

[75] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014. 26, 126

[76] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018. 27, 28, 30, 55, 66, 67, 81, 88, 100, 104, 106

[77] Lei Lei, Yue Tan, Kan Zheng, Shiwen Liu, Kuan Zhang, and Xuemin Shen. Deep reinforcement learning for autonomous internet of things: Model, applications and challenges. *IEEE Commun Surv Tutor*, 2020. 29

[78] Richard Bellman. A markovian decision process. *Journal of mathematics and mechanics*, pages 679–684, 1957. 30

[79] Ronald A Howard. Dynamic programming and markov processes. *APA PsycNet*, pages 1–1, 1960. 30

[80] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009. 30

[81] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network (2015). *arXiv preprint arXiv:1503.02531*, 2, 2015. 31

[82] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2020. 31

[83] Lyu Li, Yu Peng, Yuchen Song, and Datong Liu. Lithium-ion battery remaining useful life prognostics using data-driven deep learning algorithm. In *2018 PHM-Chongqing*, pages 1094–1100. IEEE, 2018. 32, 37

[84] Jay Lee, Fangji Wu, Wenyu Zhao, Masoud Ghaffari, Linxia Liao, and David Siegel. Prognostics and health management design for rotary machinery systems—reviews, methodology and applications. *Mechanical systems and signal processing*, 42(1-2):314–334, 2014.

[85] Guangquan Zhao, Guohui Zhang, Yuefeng Liu, Bin Zhang, and Cong Hu. Lithium-ion battery remaining useful life prediction with deep belief network and relevance vector machine. In *2017 ICPHM*, pages 7–13. IEEE, 2017. 32

[86] Sana Bouajaja and Najoua Dridi. A survey on human resource allocation problem and its applications. *Operational Research*, 17(2):339–369, 2017. 33

[87] Oscar Serradilla, Ekhi Zugasti, Jon Rodriguez, and Urko Zurutuza. Deep learning models for predictive maintenance: a survey, comparison, challenges and prospects. *Applied Intelligence*, pages 1–31, 2022. 35

[88] Ninad Arabekar, Wassim Derguech, Eanna Burke, and Edward Curry. Autonomic source selection for real-time predictive analytics using the internet of things and open data. In *Real-time Linked Dataspaces*, pages 237–253. Springer, 2020.

[89] Pamela K Douglas, Sam Harris, Alan Yuille, and Mark S Cohen. Performance comparison of machine learning algorithms and number of independent components used in fmri decoding of belief vs. disbelief. *Neuroimage*, 56(2): 544–553, 2011. 35

[90] Jingwen Wei, Guangzhong Dong, and Zonghai Chen. Remaining useful life prediction and state of health diagnosis for lithium-ion batteries using particle filter and support vector regression. *IEEE Transactions on Industrial Electronics*, 65(7):5634–5643, 2018. doi: 10.1109/TIE.2017.2782224. 36

[91] Dazhong Wu, Connor Jennings, Janis Terpenny, Robert X Gao, and Soundar Kumara. A comparative study on machine learning algorithms for smart manufacturing: tool wear prediction using random forests. *Journal of Manufacturing Science and Engineering*, 139(7), 2017. 36

[92] Gurkan Aydemir and Burak Acar. Anomaly monitoring improves remaining useful life estimation of industrial machinery. *Journal of Manufacturing Systems*, 56:463–469, 2020. 37

[93] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012. 37

[94] Jinjiang Wang, Yulin Ma, Laibin Zhang, Robert X Gao, and Dazhong Wu. Deep learning for smart manufacturing: Methods and applications. *Journal of manufacturing systems*, 48:144–156, 2018. 37

[95] Ruonan Liu, Boyuan Yang, and Alexander G Hauptmann. Simultaneous bearing fault recognition and remaining useful life prediction using joint-loss convolutional neural network. *IEEE Transactions on industrial informatics*, 16(1):87–96, 2019. 37

[96] Ramin M Hasani, Guodong Wang, and Radu Grosu. An automated auto-encoder correlation-based health-monitoring and prognostic method for machine bearings. *arXiv preprint arXiv:1703.06272*, 2017. 38

[97] Yinghua Yang, Dandan Yao, and Xiaozhi Liu. Remaining useful life prediction based on stacked sparse autoencoder and echo state network. In *2020 39th Chinese Control Conference (CCC)*, pages 5922–5926. IEEE, 2020. 38

[98] Jason Deutsch and David He. Using deep learning-based approach to predict remaining useful life of rotating components. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 48(1):11–20, 2017. 38

[99] Ahmed Elsheikh, Soumaya Yacout, and Mohamed-Salah Ouali. Bidirectional handshaking lstm for remaining useful life prediction. *Neurocomputing*, 323: 148–156, 2019. 39

[100] Mei Yuan, Yuting Wu, and Li Lin. Fault diagnosis and remaining useful life estimation of aero engine using lstm neural network. In *2016 IEEE international conference on aircraft utility systems (AUS)*, pages 135–140. IEEE, 2016. 39

[101] Sheng Xiang, Yi Qin, Caichao Zhu, Yangyang Wang, and Haizhou Chen. Lstm networks based on attention ordered neurons for gear remaining life prediction. *ISA transactions*, 106:343–354, 2020. 39

[102] Yi Qin, Dingliang Chen, Sheng Xiang, and Caichao Zhu. Gated dual attention unit neural networks for remaining useful life prediction of rolling bearings. *IEEE Transactions on Industrial Informatics*, 17(9):6438–6447, 2020. 39

[103] Jun Xia, Yunwen Feng, Cheng Lu, Chengwei Fei, and Xiaofeng Xue. Lstm-based multi-layer self-attention method for remaining useful life estimation of mechanical systems. *Engineering Failure Analysis*, 125:105385, 2021. 39, 124, 126, 138, 139

[104] Hoang Anh Dau, Eamonn Keogh, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Gharghabi Shaghayegh, Chotirat Ann Ratanamahatana, Yanping, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, and Hexagon-ML Batista, Gustavo. The ucr time series classification archive, October 2018. `https://bit.ly/3CjHcDj`. 40

[105] Ying Chen, Zhiyong Liu, Yongchao Zhang, Yuan Wu, Xin Chen, and Lian Zhao. Deep reinforcement learning-based dynamic resource management for mobile edge computing in industrial internet of things. *IEEE Trans. Industr Inform*, 17(7):4925–4934, 2020. 40, 44

[106] Weiting Zhang, Dong Yang, Peng Haixia, Wen Wu, Wei Quan, Hongke Zhang, and Xuemin Shen. Deep reinforcement learning based resource management for dnn inference in industrial iot. *IEEE Trans. Veh. Technol.*, 2021. 40, 44

[107] Erotokritos Skordilis and Ramin Moghaddass. A deep reinforcement learning approach for real-time sensor-driven decision making and predictive analytics. *Comput Ind Eng.*, 147:106600, 2020. 40, 44

[108] Kevin Shen Hoong Ong, Dusit Niyato, and Chau Yuen. Predictive maintenance for edge-based sensor networks: A deep reinforcement learning approach. In *2020 IEEE 6th WF-IoT*, pages 1–6, 2020. doi: 10.1109/WF-IoT48130.2020.9221098. 44

[109] Hamed Khorasgani, Haiyan Wang, Chetan Gupta, and Ahmed Farahat. An offline deep reinforcement learning for maintenance decision-making. *arXiv preprint arXiv:2109.15050*, 2021. 40, 41, 44

[110] Long Wen, Liang Gao, and Xinyu Li. A new deep transfer learning based on sparse auto-encoder for fault diagnosis. *IEEE Trans. Syst. Man Cybern.: Syst.*, 49(1):136–144, 2017. 41, 44

[111] Ansi Zhang, Honglei Wang, Shaobo Li, Yuxin Cui, Zhonghao Liu, Guanci Yang, and Jianjun Hu. Transfer learning with deep recurrent neural networks for remaining useful life estimation. *Applied Sciences*, 8(12):2416, 2018. 41

[112] Wentao Mao, Jianliang He, and Ming J Zuo. Predicting remaining useful life of rolling bearings based on deep feature representation and transfer learning. *IEEE Transactions on Instrumentation and Measurement*, 69(4):1594–1608, 2019. 41

[113] Qing Xu, Zhenghua Chen, Keyu Wu, Chao Wang, Min Wu, and Xiaoli Li. Kdnet-rul: A knowledge distillation framework to compress deep neural networks for machine remaining useful life prediction. *IEEE Transactions on Industrial Electronics*, 69(2):2022–2032, 2021. 41, 137, 139

[114] Siyu Shao, Stephen McAleer, Ruqiang Yan, and Pierre Baldi. Highly accurate machine fault diagnosis using deep transfer learning. *IEEE Trans. on Industr. Inform.*, 15(4):2446–2455, 2018. 42, 44

[115] Yunhong Che, Zhongwei Deng, Xianke Lin, Lin Hu, and Xiaosong Hu. Predictive battery health management with transfer learning and online model correction. *IEEE Trans. Veh. Tech.*, 70(2):1269–1277, 2021. 42, 44

[116] Xiaogang Wang, Xiyu Liu, and Yu Li. An incremental model transfer method for complex process fault diagnosis. *IEEE/CAA J. Automatica Sin*, 6(5): 1268–1280, 2019. 42

[117] Simone Panicucci, Nikolaos Nikolakis, Tania Cerquitelli, Francesco Ventura, Stefano Proto, Enrico Macii, Sotiris Makris, David Bowden, Paul Becker, Niamh O'Mahony, et al. A cloud-to-edge approach to support predictive analytics in robotics industry. *Electronics*, 9(3):492, 2020. 42

[118] Santanu Das. Maintenance action recommendation using collaborative filtering. *International Journal of Health Policy and Management*, 4(2):7–12, 2013. 42

[119] Vassilis Katsouros, Vassilis Papavassiliou, and Christos Emmanouilidis. A bayesian approach for maintenance action recommendation. *International Journal of Prognostics and Health Management*, 2013. 42

[120] Ahmed Khairy Farahat, Chetan Gupta, and Hsiu-khuern Tang. System for maintenance recommendation based on maintenance effectiveness estimation, October 23 2018. US Patent 10,109,122. 43

[121] Andrea Borghesi, Alessio Burrello, and Andrea Bartolini. Examon-x: a predictive maintenance framework for automatic monitoring in industrial iot systems. *IEEE Internet of Things Journal*, 2021. 43, 44

[122] Jinjiang Wang, Lunkuan Ye, Robert X Gao, Chen Li, and Laibin Zhang. Digital twin for rotating machinery fault diagnosis in smart manufacturing. *International Journal of Production Research*, 57(12):3920–3934, 2019. 45

[123] Leticia Decker, Daniel Leite, Luca Giommi, and Daniele Bonacorsi. Real-time anomaly detection in data centers for log-based predictive maintenance

using an evolving fuzzy-rule-based approach. In *2020 IEEE Int. Conf. on Fuzzy Systems (FUZZ-IEEE)*, pages 1–8. IEEE, 2020. 45

[124] Jürgen Beckmann and Heinz Heckhausen. *Situational Determinants of Behavior*, pages 113–162. Springer International Publishing, Cham, 2018. ISBN 978-3-319-65094-4. doi: 10.1007/978-3-319-65094-4_4. 59, 61

[125] Peng Xiaolan, Xie Lun, Liu Xin, and Wang Zhiliang. Emotional state transition model based on stimulus and personality characteristics. *China Communications*, 10(6):146–155, 2013. 59

[126] Mark A Thornton and Diana I Tamir. Mental models accurately predict emotion transitions. *Proceedings of the National Academy of Sciences*, 114 (23):5982–5987, 2017. 59

[127] Johan E. Korteling, Anne-Marie Brouwer, and Alexander Toet. A neural network framework for cognitive bias. *Frontiers in Psychology*, 9:1561, 2018. ISSN 1664-1078. doi: 10.3389/fpsyg.2018.01561. 60

[128] DE Berlyne. The vicissitudes of aplopathematic and thelematoscopic pneumatology (or the hydrography of hedonism). *Pleasure, reward, preference: Their nature, determinants, and role in behavior*, pages 1–33, 1973. 61, 62

[129] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015. 66, 68, 69

[130] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 66, 67, 70

[131] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *ICML*, pages 1889–1897. PMLR, 2015. 68

[132] Stefan Profanter, Ayhun Tekat, Kirill Dorofeev, Markus Rickert, and Alois Knoll. Opc ua versus ros, dds, and mqtt: performance evaluation of industry 4.0 protocols. In *2019 ICIT*, pages 955–962. IEEE, 2019. 85

[133] Ch Cheng. Algorithms for grouping machine groups in group technology. *Omega*, 20(4):493–501, 1992. ISSN 0305-0483. doi: https://doi.org/10.1016/0305-0483(92)90023-Z. 86

[134] Jun Ni and Xiaoning Jin. Decision support systems for effective maintenance operations. *CIRP annals*, 61(1):411–414, 2012. 88

[135] Marvin Rausand and Arnljot Hoyland. *System reliability theory: models, statistical methods, and applications*, volume 396. John Wiley & Sons, 2003. 93

[136] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992. 99

[137] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015. 100

[138] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. Deep q-learning from demonstrations. In *Proc AAAI Conf on AI*, volume 32, 2018. 102, 103

[139] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015. 103, 108

[140] Jesse Farebrother, Marlos C Machado, and Michael Bowling. Generalization and regularization in dqn. *arXiv preprint arXiv:1810.00123*, 2018. 104

[141] Jianqing Fan, Zhaoran Wang, Yuchen Xie, and Zhuoran Yang. A theoretical analysis of deep q-learning. In *Learning for Dynamics and Control*, pages 486–489. PMLR, 2020. 104

[142] Jivitesh Sharma, Per-Arne Andersen, Ole-Christoffer Granmo, and Morten Goodwin. Deep q-learning with q-matrix transfer learning for novel fire evacuation environment. *IEEE Trans. Syst. Man Cybern.: Syst.*, 2020. 105

[143] Yue Wang, Yuting Liu, Wei Chen, Zhi-Ming Ma, and Tie-Yan Liu. Target transfer q-learning and its convergence analysis. *Neurocomputing*, 392:11–22, 2020. 105, 106

[144] Robert Bosch Manufacturing Solutions GmbH. The osram ticket manager. `https://bit.ly/3uAlOkI`, 2019. Accessed: 2021-09-20. 108

[145] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Thirtieth AAAI conference on artificial intelligence*, 2016. 110

[146] Haitham Bou Ammar, Eric Eaton, Matthew E Taylor, Decebal Constantin Mocanu, Kurt Driessens, Gerhard Weiss, and Karl Tuyls. An automated measure of mdp similarity for transfer in reinforcement learning. In *Workshops at the Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014. 113

[147] Lucas Lehnert, Stefanie Tellex, and Michael L Littman. Advantages and limitations of using successor features for transfer in reinforcement learning. *arXiv preprint arXiv:1708.00102*, 2017. 113, 116

[148] Yuankai Wu, Huachun Tan, Lingqiao Qin, Bin Ran, and Zhuxi Jiang. A hybrid deep learning based traffic flow prediction method and its understanding.

Transportation Research Part C: Emerging Technologies, 90:166–180, 2018. 124

[149] Sneha Chaudhari, Varun Mithal, Gungor Polatkan, and Rohan Ramanath. An attentive survey of attention models. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 12(5):1–32, 2021. 124, 126

[150] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015. 126, 127

[151] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020. 126

[152] Zhizheng Zhang, Wen Song, and Qiqiang Li. Dual-aspect self-attention based on transformer for remaining useful life prediction. *IEEE Transactions on Instrumentation and Measurement*, 71:1–11, 2022. 126

[153] Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*, 2016. 126

[154] Zirui Wang, Zihang Dai, Barnabás Póczos, and Jaime Carbonell. Characterizing and avoiding negative transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11293–11302, 2019. 129

[155] Wen Zhang, Lingfei Deng, Lei Zhang, and Dongrui Wu. A survey on negative transfer. *arXiv preprint arXiv:2009.00909*, 2020. 129

[156] Abiodun Ayodeji, Zhiyu Wang, Wenhai Wang, Weizhong Qin, Chunhua Yang, Shenghu Xu, and Xinggao Liu. Causal augmented convnet: A temporal memory dilated convolution model for long-sequence time series prediction. *ISA transactions*, 2021. 133

[157] Felix O Heimes. Recurrent neural networks for remaining useful life estimation. In *2008 international conference on prognostics and health management*, pages 1–6. IEEE, 2008. 133

[158] Huihui Miao, Bing Li, Chuang Sun, and Jie Liu. Joint learning of degradation assessment and rul prediction for aeroengines via dual-task deep lstm networks. *IEEE Transactions on Industrial Informatics*, 15(9):5023–5032, 2019. 133

[159] Mohamed Ragab, Zhenghua Chen, Min Wu, Chuan Sheng Foo, Chee Keong Kwoh, Ruqiang Yan, and Xiaoli Li. Contrastive adversarial domain adaptation for machine remaining useful life prediction. *IEEE Transactions on Industrial Informatics*, 17(8):5239–5249, 2020. 137

[160] Yu Mo, Qianhui Wu, Xiu Li, and Biqing Huang. Remaining useful life estimation via transformer encoder enhanced by a gated convolutional unit. *Journal of Intelligent Manufacturing*, 32(7):1997–2006, 2021. 138

[161] Olga Fink, Qin Wang, Markus Svensen, Pierre Dersin, Wan-Jui Lee, and Melanie Ducoffe. Potential, challenges and future directions for deep learning in prognostics and health management applications. *Engineering Applications of Artificial Intelligence*, 92:103678, 2020. 148