Physics-Integrated Data-Driven Approaches for Condition Monitoring of Manufacturing Machines and Human-centric Assembly Processes

by

Vignesh Selvaraj

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

(Mechanical Engineering)

at the

UNIVERSITY OF WISCONSIN-MADISON

2024

Date of final oral examination: 12/05/2024

The dissertation is approved by the following members of the Final Oral Committee:
Sangkee Min, Associate Professor, Mechanical Engineering
Frank E. Pfefferkorn, Professor, Mechanical Engineering
Lianyi Chen, Associate Professor, Mechanical Engineering
Sharon Yixuan Li, Assistant Professor, Computer Sciences
Shiyu Zhou, Professor, Industrial and Systems Engineering

Abstract

In the age of smart manufacturing, the ability to quickly identify faults in manufacturing machines and processes is paramount. This ability not only averts costly downtimes but also enhances product quality. Leveraging Industrial-AI alongside the accessibility of High-Performance Computing (HPC), deep learning tools have emerged as indispensable assets for autonomously and consistently monitoring manufacturing operations in real-time. However, while developing techniques for monitoring manufacturing operations is significant, the practical implementation also needs to be considered as they present distinct challenges. The challenges are twofold: (i) Data availability and quality—Insufficient labeled data for training deep learning models, and higher costs and machine constraints limiting retrofitting machines with sensors. (ii) Model generalization—Adapting models trained in laboratories to industries, and handling variations in machine operating parameters.

The primary aim of this research endeavor was to identify anomalies within manufacturing machines and processes. Our approach commenced with an investigation into the viability of employing deep learning techniques specifically tailored for fault detection and identification in industrial cold forging operations. A suite of condition monitoring hardware was developed to enable long-term studies in manufacturing industries by retrofitting a legacy machine(s) with a suite of sensors, enabling continuous data collection and real-time monitoring of its operational dynamics. Two learning frameworks, supervised and unsupervised, were explored for fault identification and detection. Subsequently, we devised preprocessing techniques for sensor data, and model architectures were meticulously crafted to efficiently process the sensor data and accurately classify occurrences of defects on the machine. The condition monitoring system performed effectively with an accuracy of 96.5% and 92.66%, to detect and identify defects, respectively. The techniques developed were then extrapolated to bearing condition monitoring scenarios to validate the universal applicability of the developed methodology.

As the next step of this research, challenges in industry deployment of the developed system were studied. Two key challenges were addressed through our work. Firstly, knowledge transferability has emerged as a critical requirement for the deployment of condition monitoring systems in industrial settings. The variability inherent in industrial operations—such as changes in machine

operating parameters, sensor calibration drifts, and equipment setups—can significantly degrade the performance of these systems. Secondly, the non-availability of reliable, easy-to-use, and customizable Data Acquisition (DAQ) systems posed a barrier to conducting long-term studies in industries and hindered the generation of data necessary for advancing smart manufacturing applications. To address this, we proposed modular DAQ and condition monitoring systems designed to be easily customizable based on specific industry requirements. Additionally, we developed open-sourced hardware tailored to condition monitoring use cases and supported it with software solutions to ensure ease of operation, aiming to promote widespread adoption and data proliferation for smart manufacturing.

Retrofitting presents a unique challenge, particularly concerning the limitations of sensor placement on a legacy machine. In the second phase of this research, we explored a methodology to model the energy consumption of ultra-precision CNC machines. The machine selected for this work was the FANUC ROBONANO α-0iB. The study began with retrofitting energy meters to the CNC machine, enabling continuous monitoring of the machine's energy input. This was followed by developing methodologies to identify equipment states in real-time by analyzing patterns and profiles in the power consumption data. From energy consumption data alone, we were able to achieve an accuracy of 98.76% in identifying the axis in operation and an R-squared value of 0.9960 in predicting the feedrate. Additionally, we developed a software tool, the G-code interpreter, capable of parsing G-codes to estimate the effective energy consumption associated with the machine's operations. This tool bridges the gap between CNC programming and energy consumption insights, helping to plan energy efficiency. The energy monitoring approach was further extended to develop an anomaly detection system, to address the challenge of retrofitting. This system successfully detected operational anomalies in real-time with an accuracy of 95%. To better understand the contributions of the extracted power consumption features, a feature importance study was conducted, offering valuable insights for advancing energy monitoring practices. To facilitate industrial adoption and promote accessibility, the anomaly detection pipeline was containerized and hosted on Amazon Web Services (AWS) servers. This setup provides industries with a real-time condition monitoring solution that integrates seamlessly with their existing infrastructure, demonstrating the scalability and practical application of our research outcomes.

Energy monitoring holds immense potential, offering opportunities to optimize and improve the efficiency of individual machines and entire manufacturing lines, as well as providing robust anomaly detection capabilities. However, there are significant challenges that hinder the realization of these benefits, particularly due to the limited or non-availability of specialized DAQ systems tailored for energy monitoring applications. Key challenges include the inability to acquire high-sampling-rate raw data that captures transients effectively and the need to interface with CNC controllers to achieve synchronization between machine operations and energy data. Addressing these gaps is critical for conducting comprehensive and accurate energy monitoring studies.

To overcome these limitations, a custom version-1 energy monitoring hardware was developed. This hardware is specifically designed to facilitate advanced energy monitoring studies, enabling researchers and industries to capture high-fidelity data and synchronize it seamlessly with CNC controllers. The hardware represents a foundational step toward implementing large-scale energy monitoring solutions, informed by our learnings and advancements in the domain of energy consumption monitoring.

In the final phase of this work, anomaly detection of manufacturing processes was studied, focusing on human-centered assembly operations. Real-time recognition and localization of assembly steps in an assembly operation are crucial to manufacturers. It helps improve productivity by automatically identifying bottlenecks and enhancing product quality by detecting errors. However, developing a robust and reliable assembly monitoring system is not trivial due to the varying lengths of fine-grained SOP steps and anthropometric variations associated with assembly workers. Our study led to the development of two approaches to detect and localize the actions performed in an assembly workstation. The first approach is called the State Machine Integrated Recognition and Localization (SMIRL), whereas, the second approach involves modeling the actions performed by a human operator in an assembly workstation as spatial-temporal graphs. SMIRL can measure the duration of each assembly step and also the assembly cycle, along with being able to detect assembly operation anomalies: Sequence breaks, and Missed steps. SMIRL was evaluated against two manual assembly operations in Foxconn. The results show that SMIRL can detect and localize actions with an Intersection over Union (IoU) score of 87.53%, and identify sequence breaks and missed steps with F1 scores of 86.64% and 87.45%, respectively.

SMIRL was further enhanced by integrating an advanced object detection model and developing a comprehensive real-time assembly guidance system with capabilities for real-time anomaly detection and indication. This was followed by the introduction of an approach to autonomously identify Non-Value-Added (NVA) activities as Out-Of-Distribution (OOD) instances. This strategy eliminated the need for explicitly training deep learning models on NVA activities, significantly reducing the dependency on extensive labeled datasets, which can be difficult to obtain. Building on this, efforts were directed toward modeling actions in an assembly workstation as spatio-temporal graphs. This approach aimed to represent assembly activities more effectively by capturing the underlying physics of the operations. By modeling interactions and transitions spatially and temporally, this method sought to enhance the accuracy and robustness of action detection and localization within assembly processes. These advancements underscore the potential for combining domain-specific physics with AI techniques for improvements in the monitoring of human-centric manufacturing operations.

The potential applications of deep learning tools for Industrial AI are colossal. However, several challenges prevent their widespread adoption by manufacturers. Through our work, we aim to identify and address some of the challenges to help drive future research work toward realizing AI in industries.

Acknowledgments

I would like to extend my deepest gratitude to everyone who has supported me throughout my PhD journey at the University of Wisconsin-Madison. This work would not have been possible without the guidance, encouragement, and assistance of numerous individuals and organizations.

First and foremost, I am immensely grateful to my advisor, Dr. Sangkee Min, for his invaluable expertise, guidance, and patience at every stage of my research. Throughout my PhD, Dr. Min provided me with the freedom to pursue new ideas and explore diverse research directions, all the while offering support and invaluable insights that helped steer me in the right direction. I am grateful for the welcoming atmosphere he fostered both inside and outside the laboratory, which has helped me feel encouraged, supported, and inspired to explore novel research directions.

I would like to extend my heartfelt gratitude to Dr. Frank Pfefferkorn for his invaluable mentorship and guidance throughout my PhD journey. From the very beginning, his thoughtful advice and unwavering support have had a profound impact on my growth as both a researcher and an educator. I am grateful for the time he devoted to fostering my professional development, helping me hone my skills, and build confidence in pursuing my career aspirations.

I am thankful to my committee members, Dr. Sangkee Min, Dr. Frank Pfefferkorn, Dr. Lianyi Chen, Dr. Sharon Yixuan Li, and Dr. Shiyu Zhou for their feedback and constructive suggestions that helped improve my work.

I am sincerely grateful for the support provided by FANUC Corporation, Japan, for the generous donation of an ultra-precision machine tool. I also extend my thanks to Semblex Corporation, Illinois, USA, and Foxconn-iAI, Wisconsin, USA, for supplying the valuable data used in this study. Their contributions have been essential to the advancement and success of my work.

I am deeply grateful for the collaborative efforts that significantly enriched my research. I extend my sincere thanks to Zhicheng Xu for his invaluable contributions and assistance in the equipment state identification and energy prediction studies. My gratitude also goes to Andrew Glaeser and the team from Postech, Korea—Sooyoung Lee, Yunseob Hwang, Kangsan Lee, and Namjeong Lee—for

their collaboration on the condition monitoring studies for industrial cold forging. I would also like to express my heartfelt appreciation to the Foxconn-iAI team, including Md Al-Amin, Xuyong Yu, and Wenjin Tao, for granting me access to industry data and providing valuable insights that were instrumental in the development of SMIRL. Lastly, I am thankful to Monami Bhuyan from MINLab for her assistance in the graph modeling of assembly operations operations research work.

A special thank you goes to my fellow MIN Lab members, whose insightful discussions and constructive feedback have greatly enriched my research. I am particularly grateful to Zhicheng Xu, Aditya Nagaraj, Andrew Glaeser, Suk Bum Kwon, Sangjin Maeng, Monami Bhuyan, Shodai Yamada, Dae Nyoung Kim, and Rui Liang. Our brainstorming sessions, within and beyond the scope of individual research areas, have helped broaden my perspectives. I deeply appreciate the camaraderie and warm atmosphere of the lab.

Lastly, my heartfelt thanks go to my family and loved ones, especially Pooja, for her unwavering support, encouragement, and understanding. My mom, for all the sacrifices she has made and all the support she has provided me over the course of my PhD, and my father for all the support.

Dedication

To Pooja Shamrao Shivale Patil, thank you for your unwavering support and for being my constant source of strength and inspiration every day.

Contents

List of Figures

List of Tables

Acronyms

ADDA Adversarial Discriminative Domain Adaptation. 42, 44, 45, 49, 50, 57

AI Artificial Intelligence. 19, 112, 151

AUC Area Under Curve. 80

AUROC Area Under Receiver Operating Characteristic. 148

AWS Amazon Web Services. 14, 17

BLE Bluetooth Low Energy. 10–13, 17, 105, 202, 203

CNC Computer Numerical Control. 212, 218

CPS Cyber-Physical Systems. 1

CV Cross Validation. 74

DAFD Deep neural network for domain Adaptation in Fault Diagnosis. 42

DAN Deep Adaptation Network. 42

DAQ Data Acquisition. 4, 5, 8, 9, 11–13, 102, 104, 200, 202, 203, 209–211

DCNN Deep Convolutional Neural Network. 27–29, 32, 35–37, 39, 47, 196

DL Deep Learning. 25, 38, 42, 58, 100, 102, 106, 177, 196, 203

FIFO First In First Out. 13, 129

FPN Feature Pyramid Network. 155

FPS frames-per-second. 113, 143

FS Full-Speed. 16, 105

GAN Generative Adversarial Network. 42

GCN Graph Convolutional Network. 166, 167, 169

GNN Graph Neural Network. 166

HMI Human Machine Interface. 103

HPC High Performance Computing. 112

I2C Inter-Integrated Circuit. 11

IIoT Industrial Internet of Things. 1

IMU Inertial Measurement Unit. 10, 11, 16, 181, 182, 194

IoU Intersection over Union. 125, 136, 141, 143, 157, 160, 162

IP Internet Protocol. 126

LSTM Long Short-Term Memory. 100

MAUM Manufacturing Advancement through Unprecedented Morphing. 17

MEMS Micro-Electro-Mechanical Systems. 16, 18

ML Machine Learning. 17, 18, 25, 38, 58, 102, 203

MMD Maximum Mean Discrepancy. 42, 46, 54, 56

NFC Near-Field Communication. 11

NIOSH National Institute for Occupational Safety and Health. 179, 181, 182, 184

NMAE Normalized Mean Absolute Error. 135, 141, 143, 147

NVA Non-Value Added. 5, 108, 112, 125, 130, 144–148, 177, 199, 222

OOD Out-Of-Distribution. 5, 58, 112, 144, 145, 147, 148, 177, 199, 222

PCB Printed Circuit Board. 11, 203, 205, 209, 210, 219

ROC Receiver Operating Characteristic. 80, 148

RoI Region of Interest. 156

RPN Region Proposal Network. 155

SBC Single Board Computer. 103, 106, 209, 219

SMIRL State Machine Integrated Recognition and Localization. 5, 6, 111, 113, 125, 127, 132, 135, 141, 142, 144, 147, 150–152, 173, 174, 176–178, 198–200, 221, 222

SoC System-on-Chip. 18

SOP Standard Operating Procedure. 116, 144–147, 166, 167

SPI Serial Peripheral Interface. 9, 11

UART Universal Asynchronous Receiver-Transmitter. 9

UPM Ultra-Precision Machining. 63, 65, 99, 102

UPMT Ultra-Precision Machine Tool. 62, 65, 99

USB Universal Serial Bus. 16, 17, 105

Chapter 1

Introduction

The development of AI tools like deep learning and machine learning, supported by the advancement in big data availability and high-performance computing, has propelled industries into the age of the fourth industrial revolution, Industry 4.0. In the new era of the industrial revolution, new technologies like the Industrial Internet of Things (IIoT), edge computing, Cyber-Physical Systems (CPS), Digital Twin, Visual Twin, etc., play a key role in enabling a better visualization of manufacturing machines and processes [1–5]. These tools and technologies enable the novel use of data for manufacturing industries and are fundamental to smart manufacturing [6]. Smart Manufacturing is an integrated approach to manufacturing that combines all process elements, from product demand to supply chain to the factory floor [7, 8]. Davis et al. [9] describe smart manufacturing as "the dramatically intensified and pervasive application of networked information-based technologies throughout the manufacturing and supply chain enterprise". Deep learning and machine learning tools empower us to process raw data efficiently and autonomously discern feature representations through multiple layers of abstraction. They serve as invaluable complements to the modeling of physical systems and phenomena, effectively addressing the stochasticity inherent in these systems.

1.1 Motivation

Using deep learning techniques to identify faults in manufacturing machines has been studied in the past [10, 11]. Within deep learning, Convolutional Neural Networks (CNNs) were generally recognized as a leading method for pattern recognition from images [12]. CNNs have been widely used in fault detection and fault classification of roller bearings [13, 14], fault recognition in rotating machinery and motors [15–18]. The motivation behind our study is threefold: Firstly, the literature so far explores the potential of deep learning models for fault classification, but none of them explore the case of implementation, the associated challenges, and the adaptability of the developed models in industries. The majority of the work conducted was performed in a laboratory setting,

either using data from simulation or from a mock-up of the actual manufacturing machine. The knowledge transferability from a laboratory setting to the industry is a challenge [19]. Additionally, generalization of the analysis techniques is required to enable the widespread adoption of AI in manufacturing industries. Deep learning techniques provide a variety of means to tackle the problem at hand, and the approach taken is highly dependent on the engineer designing the data processing and inference pipeline. Hence, generalizing the model development process such that it can be applied universally across machines and processes is also a challenge [6]. The hypothesis we aimed to test was the faulty operation condition of the manufacturing machines can be reliably identified in real-time using a sensor fusion approach in contrast to the slower traditional machine vision approach.

Retrofitting legacy machines has its challenges, it is not always possible to get the sensors installed closer to the event. Retrofitting a machine with sensors could potentially alter its parameters impacting its performance. The hazardous operating conditions of the machine could increase the cost of the retrofitting process or in some cases damage the sensors. Hence, an approach to monitor the machines with minimal augmentation needs to be studied. The second hypothesis we aimed to test was the energy consumption patterns of a machine can provide insights into its operation and can be potentially used to optimize the energy consumption of a manufacturing operation, in an effort towards Sustainable Green Manufacturing.

Secondly, in industries, up to 40% of the cost and 70% of the production time currently falls under assembly operations, either in intermediate assembly operation or final finished product assemblies [20]. The quality of the end product and the lead time of an assembly line is hugely impacted by the quality of the assembly operations performed at each assembly workstation. Hence, the concept of digitalizing humans, rather than displacing them has been a theme of interest in many manufacturing industries. It is imperative to continuously monitor assembly operations to improve product lead time and identify anomalies. The hypothesis we aimed to test was the human-centric assembly operations can be monitored continuously using vision cameras and the anomalies in the assembly operation can be identified in real-time.

As data-driven modeling approaches increasingly integrate into manufacturing machines and in-

dustries, the development of Data Acquisition Systems (DAQ) catered towards condition monitoring applications has become paramount. These systems must embody simplicity, cost-effectiveness, user-friendliness, and adaptability to industry demands. Presently, the challenge lies in the market's provision of rigid DAQ systems that lack flexibility for configuration changes. Moreover, the absence of open-source alternatives hampers repeatability and the adoption of AI by industries. The scarcity of data in the manufacturing sector for data-driven modeling studies is an artifact of this phenomenon. Our endeavors underscored this challenge, particularly as our studies unfolded in real manufacturing environments. This necessitated the design of custom DAQ systems to facilitate effective data collection and Edge AI implementation for long-term model reliability studies. Hence our final motivation is to architect a custom open-source DAQ system that serves two purposes: (i) Facilitate seamless deployment in manufacturing industries for data acquisition, cloud connectivity, and low to no-code implementation, and (ii) Provide researchers the ability to easily access raw data and hardware configurations, to support more studies in the domain of smart manufacturing and enable data proliferation.

1.2 Objectives

The overarching goal of this study is to bridge the gap between model development in a laboratory setting and actual implementation in an industrial scenario. The specific aim of this study can be divided into three segments: Firstly, we aim to study the application of deep learning in detecting and identifying faults in a cold forging industry. Through this work, we aim to understand the applicability of deep learning for fault identification of machines and the challenges faced in realizing an implementation, where stochastic perturbations from the industrial environment could affect the model's long-term reliability. Secondly, we aim to study the challenges associated with generalizing the model development and deployment process constrained by the type and location of sensors used in the industries. The first two scenarios do not involve a human in the loop, as seen earlier, assembly operations contribute to 70% of production time. Hence, in the third segment, we aim to study an approach to monitor assembly operations to evaluate them and identify faults in real time.

1.3 Thesis Composition

An overview of the research work done is detailed in this section. This thesis consists of six chapters from the introduction to the conclusion. A summary of each chapter is provided for the remainder of this section.

Chapter 2 delves into the methodologies developed to create a robust and effective condition monitoring system tailored for manufacturing industries. The chapter opens by describing a study conducted at a fastener manufacturing industry in Elmhurst, IL, highlighting the design and development of smart sensing systems—both hardware and software—that enable real-time Data Acquisition (DAQ) and seamless model deployment in industrial settings. It then discusses the methodology developed to detect and identify machine defects in real-time.

The latter section of the chapter addresses the challenges unique to condition monitoring through sensor data processing. A particular focus is given to universal applicability—the ability to replicate the developed techniques across different manufacturing processes or machines—and knowledge transferability—the ability to transfer insights and models derived from sensor data to similar machines or processes. To address these challenges, the chapter extends the developed methodologies to open-source datasets commonly utilized in condition monitoring research. Furthermore, it introduces a novel domain adaptation approach that builds on adversarial domain adaptation to enable knowledge transfer in scenarios involving varying operating parameters or between machines of similar type (same OEM) performing similar manufacturing processes. This work provides the foundation for a robust and scalable condition monitoring system for manufacturing machines focusing on widespread industry adoption.

Chapter 3 delves into energy monitoring of manufacturing machines to improve energy efficiency in industries and to detect anomalies in equipment's operation. The work starts by developing a methodology to characterize the energy consumption of an ultra-precision CNC machine tool. Through this work, we developed a G-code interpreter and an equipment state identification model that can process the patterns in energy consumption to identify the different machine states—Axis in operation and the feedrate. The G-code interpreter was further improved in the later sections

of this chapter to predict the energy consumption from G-code.

One of the key challenges with retrofitting is the inability to get closer to the event of interest to monitor the equipment's condition. Hence, this chapter extends to utilizing energy monitoring as an indirect means of monitoring equipment's condition. An approach to detect operational anomalies in CNC machines was developed and validated. To enable industries to adopt the techniques a complete pipeline for the energy consumption-based anomaly detection approach from DAQ to model development to model deployment and monitoring was designed and open-source with the respective publication. The final segment of the research focuses on one of the fundamental challenges faced by many other researchers in this field, the inability to acquire data synchronized with the CNC controller actions at a high sampling rate. To enable and support future studies in the field of energy monitoring custom hardware capable of measuring 3-phase current and voltage at sampling rates high enough to capture power transients was designed and manufactured. The hardware can interface with CNC controllers and synchronize the power data collection with the controller actions. Additionally, the hardware can communicate with each other over RS-485 to synchronize the power consumption across modules deployed on manufacturing lines.

Chapter 4 delves into approaches and methodologies developed for real-time monitoring of human-centric manual assembly operations. The work in this chapter was conducted in collaboration with Foxconn to develop a system to detect and localize the actions performed by a human operator in an assembly workstation. A novel algorithm called the State Machine Integrated Recognition and Localization (SMIRL) was developed to continuously monitor the assembly workstation using vision cameras and measure the step and cycle time, autonomously. Additionally, SMIRL can detect assembly operation anomalies—sequence breaks and missed steps–alerting the operator in real time. The chapter then focuses on an approach to detect Non-Value Added (NVA) activities in assembly workstations without explicit training of deep learning models on them, by considering NVA activities as Out-Of-Distribution (OOD) instances. The final leg of the chapter discusses an approach to model the actions and activities performed in an assembly workstation as spatiotemporal graphs. The graph modeling of assembly operations has enabled the capturing of physics process physics and reduced the computation requirement, as processing the graphs was less computation intensive compared to SMIRL. The work in Chapter 4 was developed into an app

with a GUI to interact with SMIRL to run inference on either a live stream from IP cameras or recorded data.

Throughout my research, various challenges fundamental to smart manufacturing—particularly those stemming from limited data availability—have been discussed extensively. Conducting studies in manufacturing industries required the development of custom condition monitoring hardware to facilitate long-term studies. Simultaneously, it was crucial to ensure that the data collection hardware remained inexpensive, reliable, and customizable to meet diverse needs. To address these challenges, I have advocated for the development of modular, easy-to-use, customizable, and open-source condition monitoring hardware that can be utilized by both industries and academic institutions. This effort represents an essential first step toward improving data availability for condition monitoring studies.

To demonstrate this concept, a prototype hardware was developed, briefly described in Section 2.2.2 and Appendix A.1. In Chapter 5 a case study was conducted to evaluate the capabilities of the prototype hardware. This study focused on risk assessment for repetitive lifting activities in manufacturing industries, where wireless sensing systems were attached to the lumbosacral region to prevent injuries and enhance workplace ergonomics. The potential impact of modular condition monitoring hardware on generating data for smart manufacturing use cases is discussed in Chapter 6, where it sets the stage for one of the key directions for future research. This approach emphasizes the importance of accessible and scalable hardware solutions to drive innovation and data-driven advancements in manufacturing.

Chapter 2

Robust Monitoring of Manufacturing Machines

This chapter presents a study on condition monitoring of machines through physics-incorporated data-driven modeling of multi-sensor data. The research was conducted in two distinct industries: a cold-forging industry in Chicago, USA, and a compact roller manufacturing industry in Asan-si, South Korea. The study has two primary objectives: (i) to detect and identify faults in an industrial cold forging machine, and (ii) to identify challenges that hinder the deployment of a robust fault monitoring system within the industry. This work extends our study conducted [21].

2.1 Literature Review

Cold forging is a manufacturing process with a high potential to benefit from advancements in deep learning applications. Cold forging is a process of shaping metal into desired geometry through impact forces. Due to the nature of cold forging, tools are subject to extreme loading, often resulting in fatigue fracture, plastic deformation, or wear [22–24]. Cold forging processes are capable of producing multiple parts per second. With high production rates, it becomes imperative to detect conditions like failure/wear, voids in stock material, operating conditions, etc., that lead to defective products early. Product scrap rates can be up to 20%, resulting in material wastage. Typically, in manufacturing industries, tool life prediction models are used to determine the life of the tool. But, in the case of cold forging, it is challenging to use tool life prediction models due to several factors impacting the life of the tool [25]. Hence, several efforts have been made to study and monitor the cold forging process. Behrens, Santangelo, and Buse [26] determined that the Acoustic Emission (AE) sensors could effectively indicate the formation of cracks during the cold forging of bevel gears. Li et al. [27] used force sensors to detect three types of defects in the cold forging of fasteners.

The general success of these studies indicates the potential of using deep learning in industries to detect and identify faults. It becomes important to continue the research efforts to improve

the existing technologies, find new applications, and identify challenges specific to manufacturing industries, i.e., the factory floor. The majority of the manufacturers still use legacy machines [6], these machines cannot collect and transfer data in real time unless retrofitted. AE and force sensors are commonly used in monitoring the cold forging process. However, in real applications, the in-die sensor fails to detect several common failures. The reason can be primarily attributed to the level of noise present on a factory floor. Vibration has been shown to detect fracture in other mechanical systems [28–30], so the expansion of process monitoring in cold forging and the effectiveness of monitoring vibration signal was explored in our previous study [31]. From our last study, we discovered that the processes producing geometrically different parts on the same machine could be differentiated using vibration profiles through applications of deep learning.

This work aims to study the prospect of implementing a reliable fault detection system in a cold forging industry. From our preliminary work [31], we have identified that 20% of the parts produced on a cold heading machine were scrapped due to defects, costing the company $80,000 per annum per machine. The traditional approach used a machine vision system to detect the cracks but was unable to meet the production demands. Through this study, we provide insights into the extent a legacy machine can be retrofitted and the importance of sensor location in realizing a reliable fault detection system. Finally, as this study was conducted in an actual manufacturing facility we also aim to identify the various challenges that could impact the model performance during a deployment. A part of the work presented in this chapter was published in Glaeser et al. [21].

2.2 Design of a Machine Monitoring System (MMS)

Enabling continuous real-time DAQ is challenging. The commercially available ones are more suitable for experimental studies in a controlled laboratory setting. To conduct this study we required DAQ systems that meet the following requirements: (i) Be able to provide access to the raw data for analysis and study, (ii) Enable synchronous data acquisition across multiple sensors, (iii) Configurable so that it can be integrated into the cloud services for remote data collection, (iv) Finally, should also be inexpensive as these systems were deployed in remote locations and are unmonitored for a long time for data collection.

2.2.1 Preliminary Design

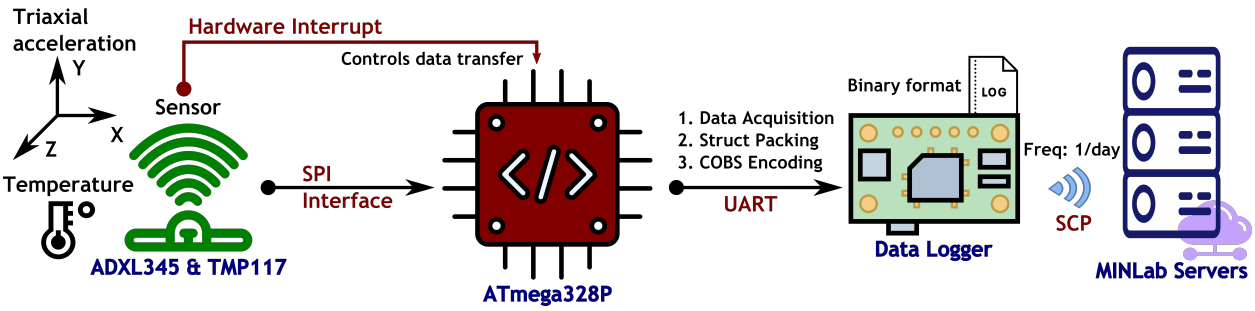


Figure 2.1: Machine monitoring system developed for continuous DAQ.

The very first prototype of a DAQ system was built to collect data and record it locally on a separate edge device before being transferred to a remote server. The DAQ system can be seen in Figure 2.1. The DAQ system consists of three main components, the sensor, a low-power microcontroller, and a data logger. The sensor can measure acceleration in the range of +16g to -16g with a maximum sampling rate of 1650Hz and consists of a digital anti-aliasing filter to prevent aliasing. The sensor-microcontroller interacted with each other using Serial Peripheral Interface (SPI), operating at a clock frequency of 16MHz, using hardware interrupts. The data communication between the microcontroller and the data logger was through the Universal Asynchronous Receiver-Transmitter (UART) protocol following a FIFO strategy. The data received by the microcontroller were packed into a struct containing three 16-bit unsigned integers and were then COBS (Consistent Overhead Byte Stuffing) packed before transferring over to the data logger through UART. The data logger handles data collection, storage, and transfer to the servers located at the University, once or twice a day. This process was completely automated, ensuring that the DAQ process was uninterrupted and timestamped. This device was deployed in the cold forging industry for continuous long-term DAQ from the cold-heading machines. The study conducted in Section 2.3 was enabled by this device.

2.2.2 Wireless Sensor Systems

Throughout my research, the need for continuous monitoring and data acquisition (DAQ) evolved significantly. It became crucial to monitor multiple machines or objects simultaneously to assess the long-term reliability and transferability of deep learning models. In certain cases, this approach

was also essential for studying the importance of sensor placement, helping to determine the optimal locations for sensors. A custom wireless sensor system capable of sampling acceleration and orientation across three axes, transmitting the data wirelessly via Bluetooth Low Energy (BLE) was developed. The system could achieve a sampling rate of 800 Hz when either acceleration or orientation data was enabled, and 400 Hz when both sensors were active. Battery-powered and capable of wireless data transfer, the system supported remote operation.

The following sections provide a detailed discussion of the sensor system's hardware and software architecture. Additionally, to streamline sensor interfacing, a custom application called **Supremus** was developed, with its features and capabilities described in Appendix A.1.

Hardware

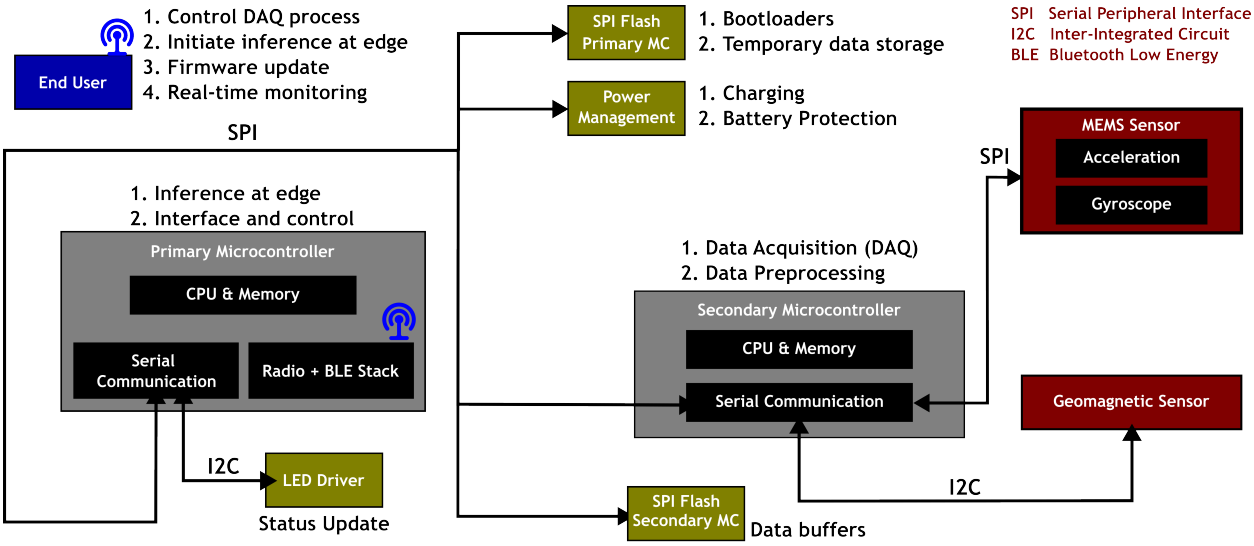


Figure 2.2: Wireless sensor system hardware architecture [32].

The sensor system is designed for continuous data logging and edge inference applications, comprised of two microcontrollers, one from Nordic Semiconductor and the other from Bosch Sensortec. The microcontroller from Bosch Sensortec contained an integrated Inertial Measurement Unit (IMU) capable of measuring acceleration and orientation. The microcontroller from Nordic Semiconductor, nRF52832, runs at 64 MHz with an Arm Cortex-M4 architecture and 512 kB of flash, referred to, as the primary microcontroller for the rest of the paper. The primary micro-

controller has BLE, and Near-Field Communication (NFC) controllers built-in and only needs an external antenna for both to function. Bluetooth is ubiquitous among cellphones and laptops, hence it was chosen as the means of interface for the device. Additionally, the BLE protocol, as the name states, requires low energy to operate, leading to a longer battery life. The chip from Bosch Sensortec, BHI260AP, came packaged with an IMU sensor and a microcontroller, called the secondary microcontroller in this paper. The secondary microcontroller runs at a lower clock frequency compared to the primary microcontroller, hence it was used for DAQ and simple feature extraction tasks.

The Printed Circuit Board (PCB) design was optimized to maximize the battery life. The primary microcontroller, while more powerful and versatile than the secondary microcontroller, consumes far more power. Hence, the PCB was designed to activate the primary microcontrollers only when they are required to operate. For instance, in the case of continuous monitoring, only the secondary microcontroller would be active unless an event was detected, which would then turn on the primary microcontroller. The event could be a press of a push button or the detection of an anomaly in the process being monitored. The other components in the PCB included the power controller and battery management system, LED indicators, and push buttons. The sensor interface with the secondary microcontroller could be over SPI or Inter-Integrated Circuit (I2C). The interface between the primary microcontroller and the secondary microcontroller is also SPI. The sensor system architecture can be seen in Figure 2.2. The sensor system was powered using a 230mAh LiPO battery at 3.3V and the microcontrollers were powered at 1.8V using a buck converter. The buck converter is part of a battery management system made by Texas Instruments (TI), BQ25120A. The battery management system monitors and controls the charging of LiPo batteries from a 5V supply. The battery's temperature is continuously monitored by the battery management system while charging to ensure the safe charging of the devices. The PCB designed for prototyping can be seen in Figure 2.3. The dimensions shown are only for the prototype and the final product would be shrunk further as JTAG pins would not be required and a smaller antenna can be used. For different variations of the sensor system, other sensors such as Geomagnetic, temperature, humidity, etc., could be potentially added to the SPI/I2C lines depending on requirements.

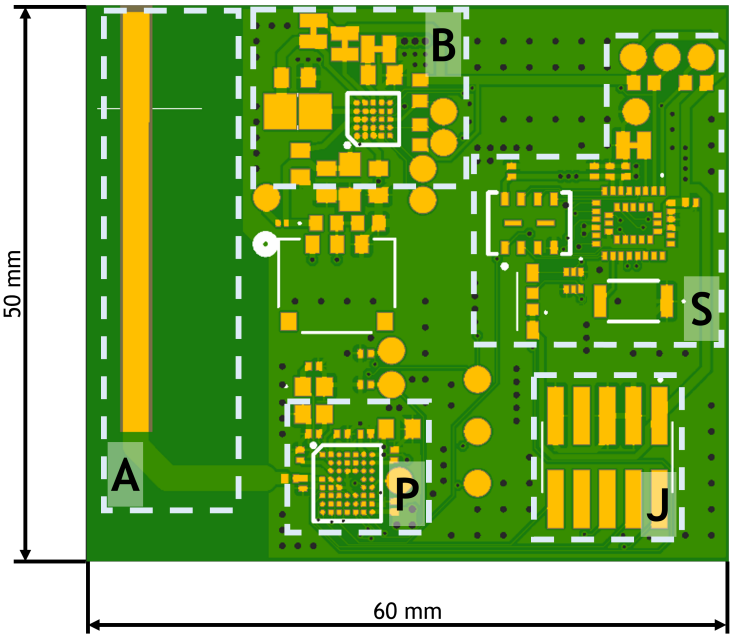


Figure 2.3: PCB for prototyping the wireless sensor system. A-2.4GHz Antenna; B–Power Management; P–Primary microcontroller; S–Secondary microcontroller and sensors; J–JTAG connector [32].

Software

The sensor system, particularly the primary microcontroller, was programmed using the RTOS ecosystem, specifically Zephyr RTOS. The Zephyr RTOS is small and easily scalable and can be used in resource-contained devices. Additionally, it can support multiple protocols from BLE to CANbus, etc. Zephyr RTOS has also been designed to use as little power as possible, enabling long operating hours for the DAQ and real-time inference. RTOSes enable handling multiple processes at a time and can respond to events within a predictable time limit. Hence, for the application in this work, the sensor system was programmed using RTOS over bare metal.

The software architecture designed for DAQ can be seen in Figure 2.4. A BLE device can take the role of a peripheral or a central. Any device can operate as a peripheral or a central. A peripheral is a device that accepts connections from a central and consumes less power. The sensor system acts as a peripheral, accepting a connection from a central that is running a custom-designed application. Once connected, the central device would set the sampling rate and sensor type for DAQ. The primary microcontroller of the sensor system was event-driven and could run multiple threads

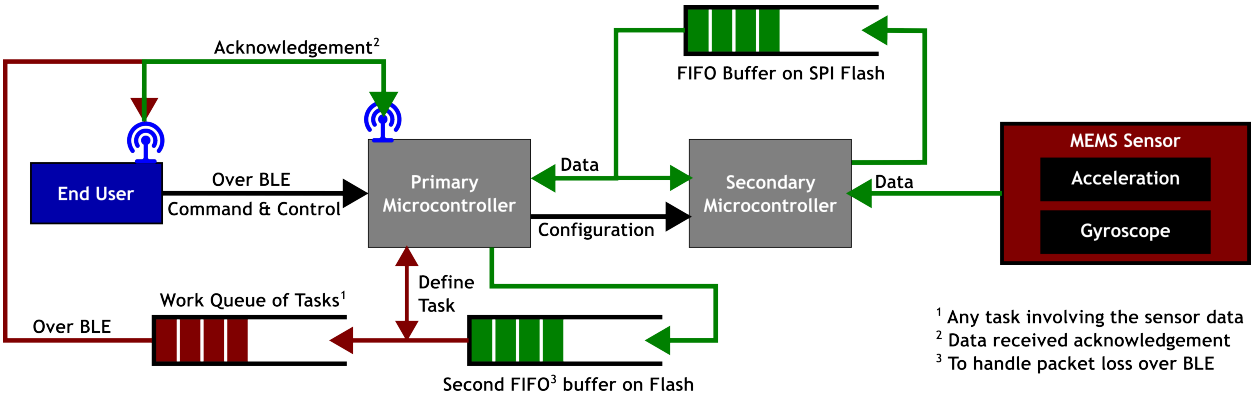


Figure 2.4: Sensor system software architecture [32].

concurrently based on their priority levels. Upon receiving the configuration, the sensor system would communicate with the secondary microcontroller to configure the sensors. The initiation of the DAQ process was performed by the central, once initiated, the secondary microcontroller would sample the data from the sensor and push it to the First In First Out (FIFO) buffer. The primary microcontroller would be responsible for querying the secondary microcontroller to receive the data. The secondary microcontroller can be programmed to push the raw data or preprocessed data to the FIFO buffers depending on user requirements. On receiving the data, the primary microcontroller can either make an inference or transfer the raw data over BLE for DAQ. In both cases, work queues were used to enable concurrency. Work queues enable processing the work items in a FIFO fashion. A work item corresponds to any task that the microcontroller needs to perform. A high-priority thread typically uses work queues to offload non-urgent processing to a lower-priority thread. In the case of DAQ for the sensor system, the work queues were designed to package the data into binary packets of 240 bytes, which were then transferred to the central with acknowledgment to ensure reliability. The FIFO process associated with the work queues, binary data transfer, and the BLE transfer acknowledgment was responsible for ensuring reliability in the BLE process by preventing packet loss and not exceeding the BLE bandwidth.

A custom application was written using NodeJS to interface with the sensor system, either with a single sensor system or multiple ones simultaneously. The application enabled remote firmware updates for both the primary and secondary microcontrollers, real-time data acquisition and logging, and finally, can perform machine learning model inference either at the edge or in the

cloud through Amazon Web Services (AWS). The sensor system was evaluated by conducting a case study on assessing the risk of lifting activities, seen in Chapter 5.

System Performance Evaluation And Challenges

The theoretical transmission rate for the BLE protocol varies depending on the version and specific device configurations. In our scenario, we capped the transmission rate at 1Mbps to accommodate the maximum sampling rate requirement. However, due to potential communication obstacles, achieving the theoretical throughput can be challenging in real-world conditions. Considering a sampling rate of 400 Hz across three axes for three different sensors (acceleration, gyroscope, and orientation), the maximum required throughput for our application was 115.2 Kbps, as it was 4 bytes per data instance per axis.

To assess our application's actual throughput, we conducted a rigorous three-day evaluation using software-based measurements, continuously transmitting data packets between two communication devices. The software can not only measure the throughput but also timestamp the data to measure the lag in data transfer. The observed data loss and the incurred lag during the study were negligible. Consequently, we are confident in our bandwidth allocation, affirming that each sensor can transmit data over BLE at a sampling rate of 400 Hz without encountering bandwidth limitations.

It is worth noting that the throughput measurement study was performed in a controlled environment, where the interference to the radio signals could be limited. In the real world, various factors could potentially impact the DAQ process, but we believe with the bandwidth availability and the capability of inference at the edge, we can overcome the challenges

Challenges

This section delves into several challenges encountered during the development process. These challenges can be categorized into three main areas:

1. **Synchronization of Multiple Sensors:** Integrating multiple sensors poses synchronization

challenges since these devices lack a shared clock. Currently, synchronization is achieved at the software level.

2. **Bandwidth Limitations with Increasing Sensor System Count**: As the number of sensors grows, the available bandwidth for data transfer diminishes. This necessitates exploring alternatives such as edge inference or reducing sampling rates to manage data throughput effectively.

3. **Role of Data Receiver Device**: The device responsible for receiving data from the sensor systems plays a crucial role in synchronization and minimizing data loss. Presently, an application running on the end user's laptop handles these tasks, which can be unreliable. Thus, a dedicated hub will be designed as part of future work. This hub will interface with multiple sensor systems, facilitating data collection, connection to cloud services, and, in some cases, edge inference or fine-tuning of model weights.

As the number of sensor systems collecting data grows, the abovementioned challenges will increasingly impact our operations. Therefore, as part of our future endeavors, we are committed to tackling these challenges. Our goal is to bolster the resilience and efficiency of our sensor systems, fostering seamless operation and reliable data analysis across a wide range of manufacturing scenarios.

2.2.3 In-situ condition monitoring

Smart sensing systems have made a pivotal shift in smart manufacturing. They enable real-time monitoring, data collection for logging and processing, and data analysis [33, 34]. A smart sensing system is an amalgamation of diverse sensor and embedded system technologies. Building blocks of a smart sensing system can be seen in Figure 2.5.

Vibration Monitoring Sensor System

In-situ monitoring has become essential for overseeing manufacturing machines and processes, as it augment the retrofitting process. As part of my research on condition monitoring of manufacturing

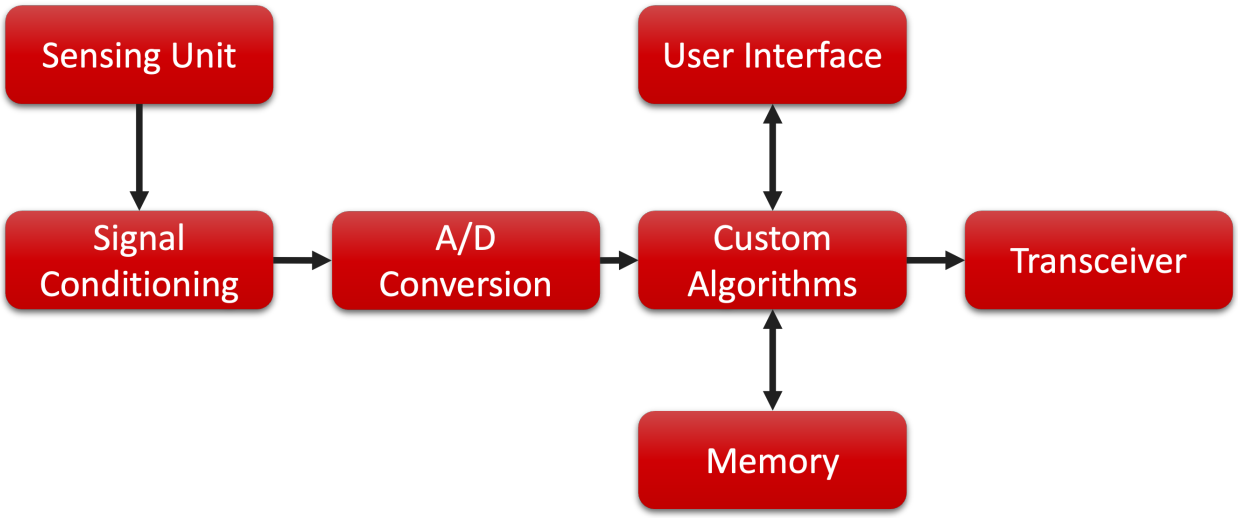


Figure 2.5: Smart sensor building blocks [33].

machines, custom sensor systems were developed to facilitate this monitoring. The overarching goal of this work was to create edge devices capable of data management, processing, logging, and ultimately, the development and deployment of machine learning models for condition monitoring at the edge.

The sensor systems were designed to be modular and seamlessly integrated with manufacturing machines and processes. One such system, shown in Figure 2.6, includes schematics, PCB layout, and firmware details, which are available in Appendix A.2. This system features a Micro-Electro-Mechanical Systems (MEMS) IMU sensor, whose specifications are provided in Table 2.1. The sensor is capable of streaming real-time data over Universal Serial Bus (USB) Full-Speed (FS) at 12 Mbit/s. The primary application of this sensor is to enable in-situ monitoring, where the sensing system is positioned as close to the event as possible for effective condition monitoring.

Table 2.1: LSM6DSLTR Sensor Specifications.

Specification	**Description**
Sensor Type	3D Accelerometer and 3D Gyroscope
Accelerometer Full-Scale Range	±2g, ±4g, ±8g, ±16g
Gyroscope Full-Scale Range	±125 dps, ±250 dps, ±500 dps, ±1000 dps, ±2000 dps
Output Data Rate (ODR)	Up to 6.66 kHz (Accelerometer & Gyroscope)
Operating Voltage	1.71 V to 3.6 V
Communication Interface	I2C (up to 1 MHz), SPI (up to 10 MHz)
FIFO Depth	9 KB FIFO for sensor data logging

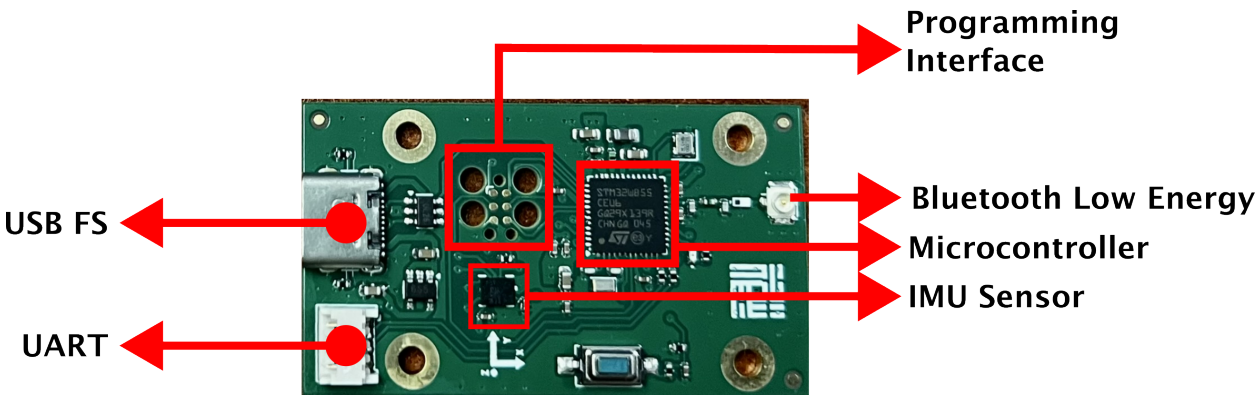


Figure 2.6: Sensor system developed for vibration monitoring.

With the help of BLE, the sensor system can be remotely operated, and the current firmware version, at the time of this writing, comes with the following features:

1. Remotely enable DAQ to start and stop operations.
2. Real-time Fast Fourier Transform (FFT) computation and USB logging.
3. Remotely start the edge Machine Learning (ML) model for running custom machine learning algorithms.
4. Capability to stream the data to a remote hub connected to AWS cloud services, thereby interfacing with the cloud services for cloud inference and logging purposes.
5. Remote Over-The-Air (OTA) firmware update.

Acoustic Monitoring Sensor System

To enable condition monitoring using acoustic sensor signals, a custom PCB was designed to acquire acoustic signals within the human audible range. This board was developed as part of the work for the Manufacturing Advancement through Unprecedented Morphing (MAUM) consortium in collaboration with Heungkuk Metaltech Co., South Korea. Its primary application was to monitor the welding process, preventing sputtering and indicating the need for a welding tooltip change in real-time.

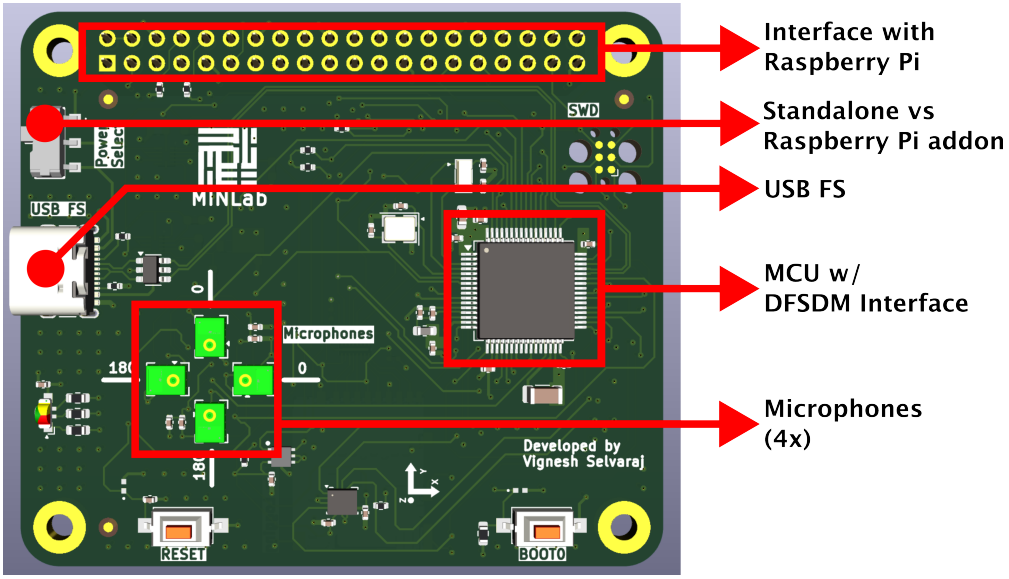


Figure 2.7: Front Side - Acoustic Monitoring Sensor System.

Although the MEMS-based acoustic monitoring sensor system was specifically developed for welding operations, it was designed with the flexibility to support future condition monitoring applications. The sensing system, shown in Figures 2.7 and 2.8, interfaces with commercially available microprocessors such as the Raspberry Pi 4 and 5 (at the time of writing). Its compatibility with other System-on-Chip (SoC) modules enhances its expandability, allowing it to adapt to various applications. Through interface headers, the sensing module communicates with the Raspberry Pi, enabling edge computation and cloud connectivity. Additionally, the system includes an SD card module for standalone data logging. Further details on the design are available in Appendix A.3.

The acoustic monitoring system has the following features that support real-time condition monitoring applications:

1. Capable of operating as a standalone system or interfacing with Raspberry Pis for real-time data logging and running ML models at the edge.
2. Integrated SD card module for standalone data logging.
3. Can be battery-powered with energy-efficient CPUs for portable applications.
4. Equipped with four microphones, strategically placed to perform beamforming and acoustic source localization in the human audible range (100 Hz to 20,000 Hz).

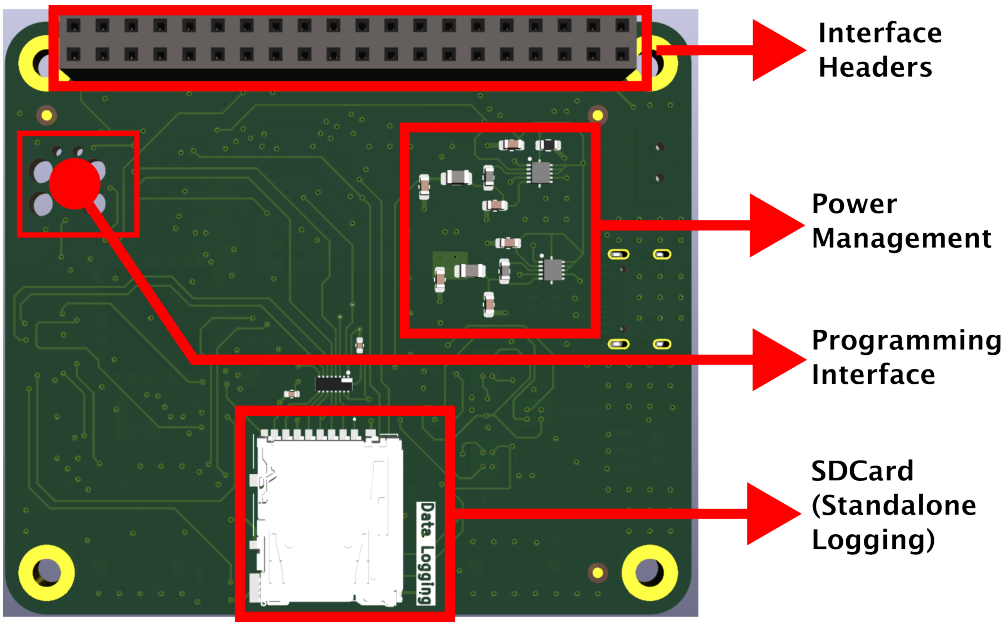


Figure 2.8: Back Side - Acoustic Monitoring Sensor System.

5. Includes additional vibration and temperature sensors to extend the system's usability.

Table 2.2: Specifications of IMP34DT05, IIS3DWB, and STTS22HTR Sensors.

Specification	**IMP34DT05**	**IIS3DWB**	**STTS22HTR**
Sensor Type	MEMS Microphone	3-Axis Accelerometer	Temperature
Frequency Response	20 Hz to 20 kHz	6 kHz bandwidth	N/A
Sensitivity	-38 dBFS ± 1 dB	±2g, ±4g, ±8g, ±16g	N/A
Communication Interface	PDM	SPI/I2C	I2C

2.2.4 Discussion - Why do we need Smart Sensing Systems?

To summarize, the features of the smart sensing system, as explored in several studies, are illustrated in Figure 2.9. A key factor enabling the reliable development of Artificial Intelligence (AI) models for condition monitoring of manufacturing machines is the availability of data. Currently, the amount of open-source data available for manufacturing industries is limited, and it does not fully explore the intricacies and nuances of manufacturing operations. For all the studies and analyses conducted throughout my research, data collection was necessary. While traditional DAQ systems from companies like NI are capable of collecting data, they are not well-suited for long-term studies, where sensing systems need to be deployed in industrial settings for continuous and real-time data streaming.

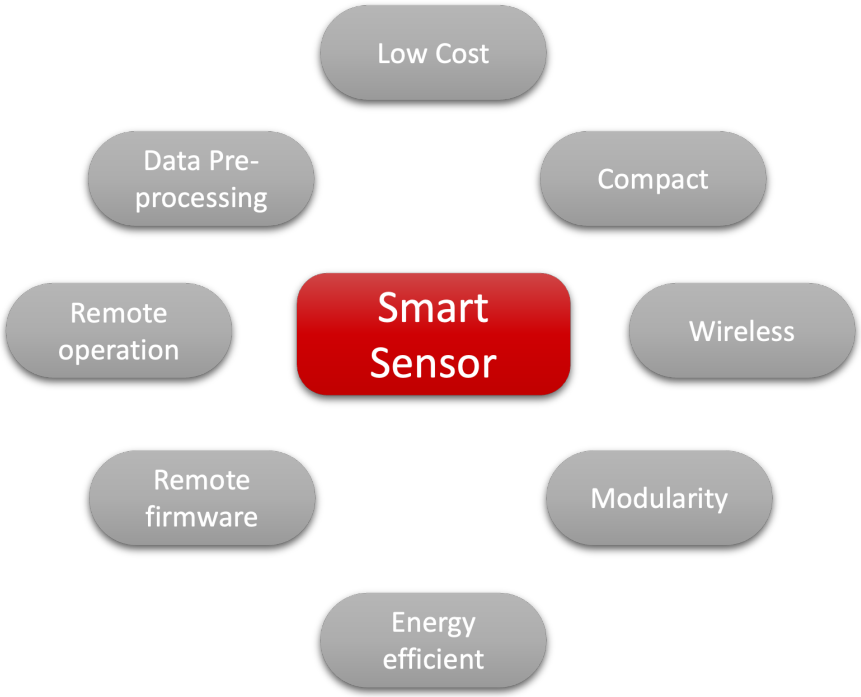


Figure 2.9: Smart sensing system features [33].

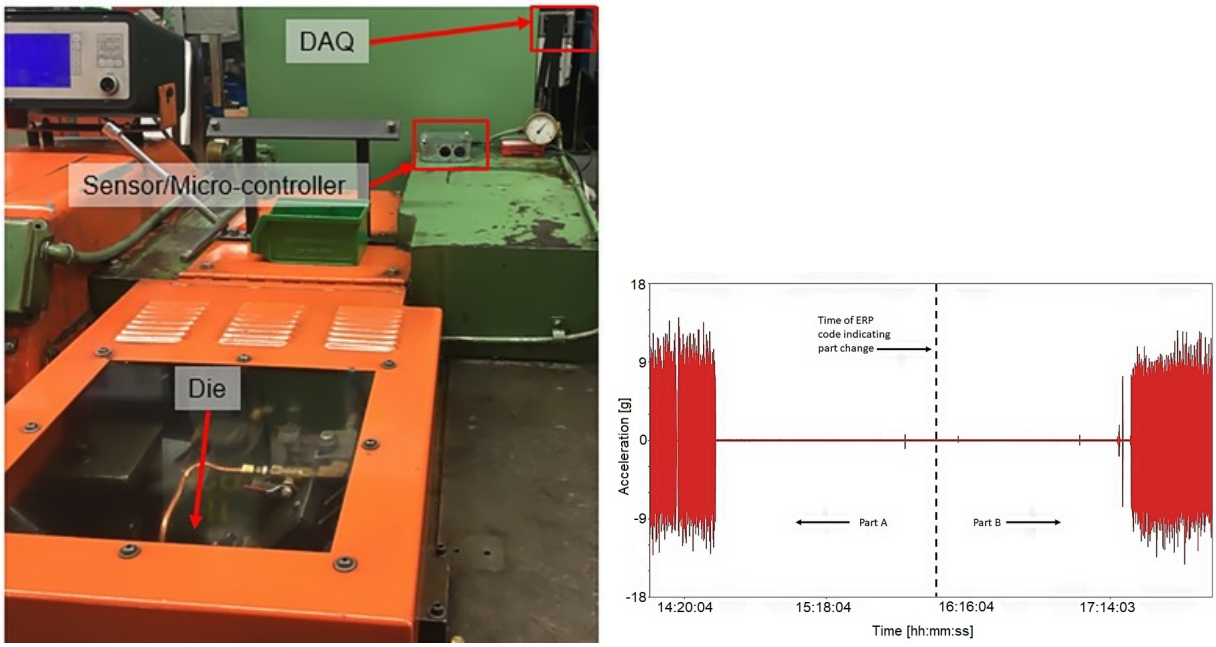
In addition to developing and studying approaches for condition monitoring of manufacturing machines and human-centric assembly operations, my research aims to create an open-source smart sensor system, including both hardware and software. The goal is to encourage more widespread data collection from real manufacturing industries for smart manufacturing applications. To this end, all the hardware described in Section 2.2 has been deployed in real manufacturing environments, providing us with valuable data for our studies. Efforts are also being made to modularize and open-source the hardware development so that academic researchers can build upon the existing designs, thereby increasing the volume of data available for smart manufacturing research.

2.3 Part Classification

In this section, the preliminary work that was conducted to showcase the prospects of using vibration data to monitor a cold forging process is presented. The purpose of this study was also to evaluate the machine monitoring system for deployment in manufacturing industries.

2.3.1 Experiments

After deploying the machine monitoring system that we developed on a cold forging machine, the data were collected for a month. The DAQ system was developed to operate and transfer data autonomously from the factory floor. At the end of each day, the collected data were transferred to a server located at MINLab. The data transfer was timed to overlap with the non-operating hours of the machine.



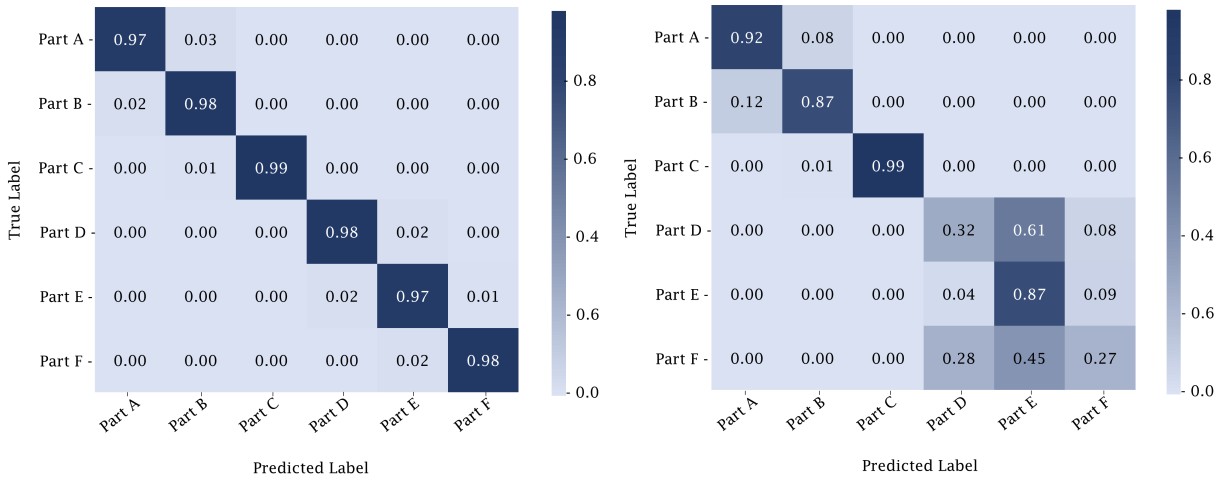
(a) One die two blow cold forging machine. (b) Labeling of acceleration sensor signals.

Figure 2.10: Experiment setup and labeling process for parts classification [31].

The data were correlated with the machine operating states using the factory's Engineering Resource Planning (ERP) system. The information from the ERP system was timestamped and contained a series of codes indicating different operating states of the machine. The codes referring to "machine setup" were isolated and used to partition and label the sensor data. The sensor placement and the instance of the partitioning process are shown in Figure 2.10. Entries in the ERP system were done manually, leading to human errors by some margin. Hence, to reliably label the acquired data, a custom pattern-matching algorithm was run to identify the running state of the machine after the detection of a "machine setup" code by identifying the signal's Root Mean Square (RMS), Signal Energy, and common frequencies in the range of 0 - 500Hz.

2.3.2 Results and Discussion

The raw signal after partitioning and the labeling process was split into multiple 1-second segments. The time series data were then transformed into a scalogram using a wavelet transformation. The scalogram was obtained for each axis of the triaxial acceleration sensor, leading to a final input data shape of $40 \times 40 \times 3$. A fully connected Convolution Neural Network (CNN) was then used to classify the parts produced by the machine. The results of this classification study are presented in Figure 2.11. Two test datasets separated by the date of machine operation, i.e., test dataset II were collected 2 weeks after collecting test dataset I. Through this approach, we can evaluate the model's long-term reliability.



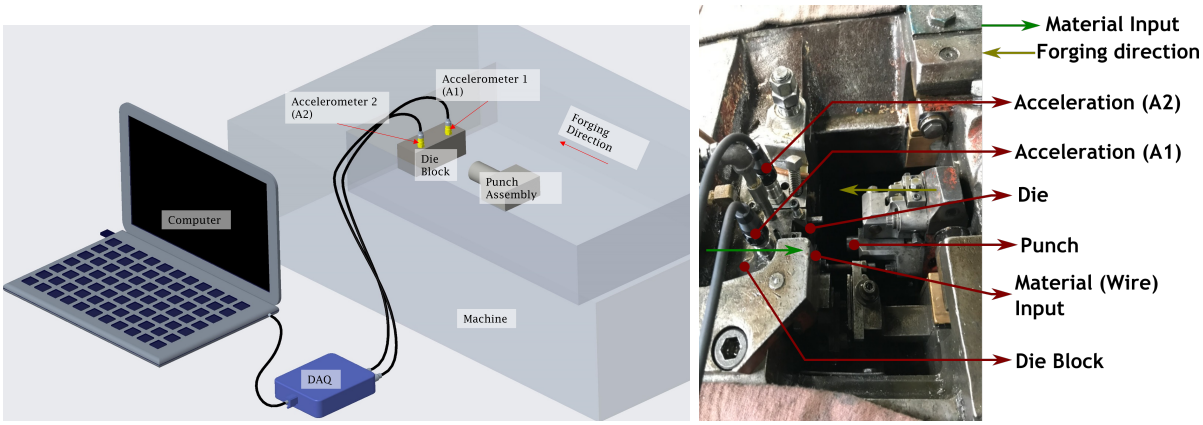
(a) Confusion Matrix for the test dataset - I.

(b) Confusion Matrix for the test dataset - II.

Figure 2.11: Confusion matrices for two test datasets separated by date of equipment operation.

From Figure 2.11, it can be seen that the 3-axis acceleration signals can be used as a distinguishing feature to separate the part geometries. In the case of test dataset II, the model has trouble classifying parts D, E, and F, this can be attributed to stochasticity introduced into machines by human operators. In cold-forging, machine setup can change from operator to operator, and the changes in the machine setup could affect the properties of the signals. Throughout this study, the same tooling was used until failure, hence deterioration associated with the tools and the machine could impact the performance of the models. In the next section, stochasticity associated with the machine setup, sensor calibration, and operation temperature was further explored by testing the

ability to transfer knowledge between machines of similar types and processes.



(a) Experiment setup for DAQ using NI-9234. (b) Retrofitted cold heading machine.

Figure 2.12: Retrofitting a cold heading machine for DAQ.

2.4 Fault Detection and Identification

From the work in Section 2.3, it was determined that the acceleration sensor signals contain distinguishing properties that can be used to classify parts with varying part geometries. Additionally, we were also able to get a hint on the challenges associated with time for long-term model reliability. In this section, we aim to detect and identify faults in a cold-forging machine using both supervised and unsupervised learning techniques.

Label	**Run time (mins)**	**Description**
Good	45	Machine operating in an acceptable state
Wire (Raw Material)	30	Faulty wire drawing process or damage during transport. All other conditions were acceptable
Chipped Die	30	A die chipped on its surface. All other conditions were acceptable
Die Internal Crack	30	A die with a crack at the transition radius. All other conditions were acceptable
Punch Defect	30	Punch assembly with a cracked insert. All other conditions were acceptable
Pin Defect	30	Punch assembly with a damaged pin. All other conditions were acceptable

Table 2.3: Defects considered in this study and their associated run time.

2.4.1 Experiments

The cold heading machine in this study was retrofitted with two piezoelectric acceleration sensors having a measurement range of +100g to -100g. The two acceleration sensors were mounted on the die block, one being directly on top of the die (Accelerometer 1, A2) and the other on top of the material input column (Accelerometer 1, A1). The sensor used in this study was 603C05 from PCB Piezotronics. The sensor output connector was a 2-pin MIL-C-5015. The DAQ was done using a NI-9234 unit at a sampling rate of 8.3 KHz. The schematic diagram of the experimental setup and the retrofitted machine can be seen in Figure 2.12. The LabVIEW program can be seen in Appendix A.4.

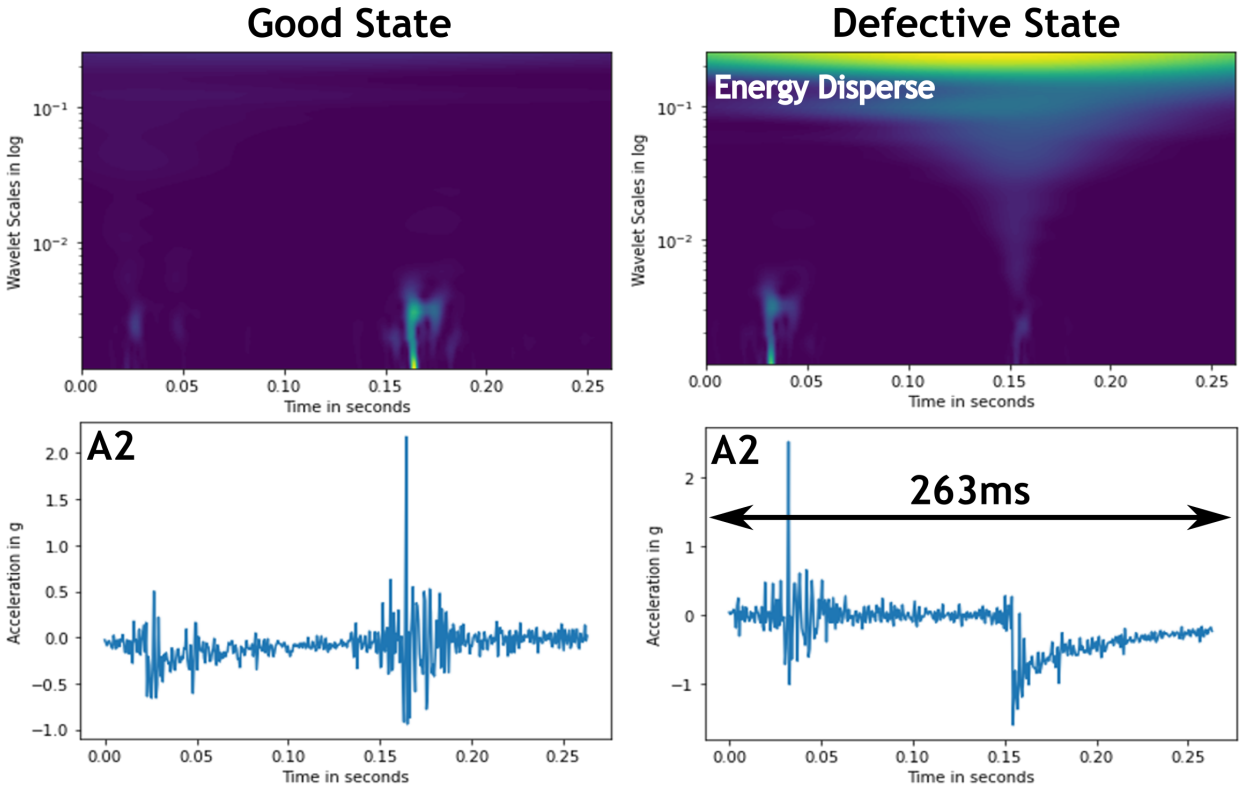


Figure 2.13: Wavelet Power Spectrum obtained from raw time series acceleration data.

Data Collection

The manufacturing process involved in the cold heading machine was a one-die, two-blow process. The first blow extrudes the shaft, and the second one forms the final geometry of the part. The

defects considered in this study were determined based on the commonly occurring faults in the industry and can be seen in Table 2.3. Additionally, these defects were hard to detect during the operation of the machine and could lead to material wastage.

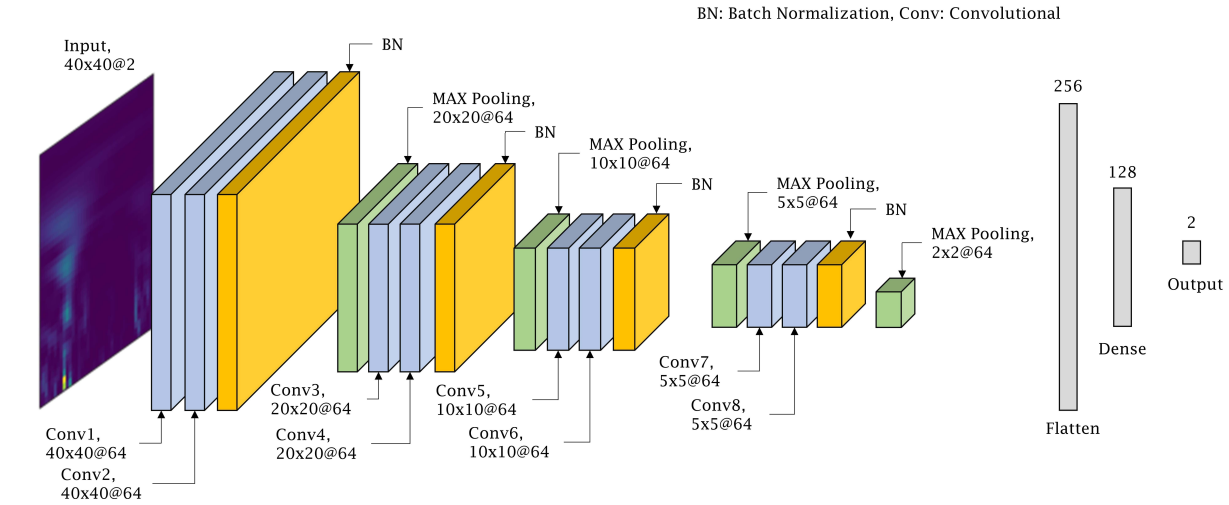
The cold heading machine under study was capable of producing 200 parts/min. The data were collected by operating the machine in a defective state, i.e., using defective tools and/or raw materials. During its operation, the acceleration data were sampled from both the sensors (A1 & A2) simultaneously and stored in a Technical Data Management Streaming (TDMS) file. The "Run time" column in Table 2.3 specifies the machine operating time in minutes in the respective defect state. In addition to the defective operating state of the machine, data was also collected when all machine parameters were acceptable and the machine was producing conforming products.

2.4.2 Machine Defects Classification

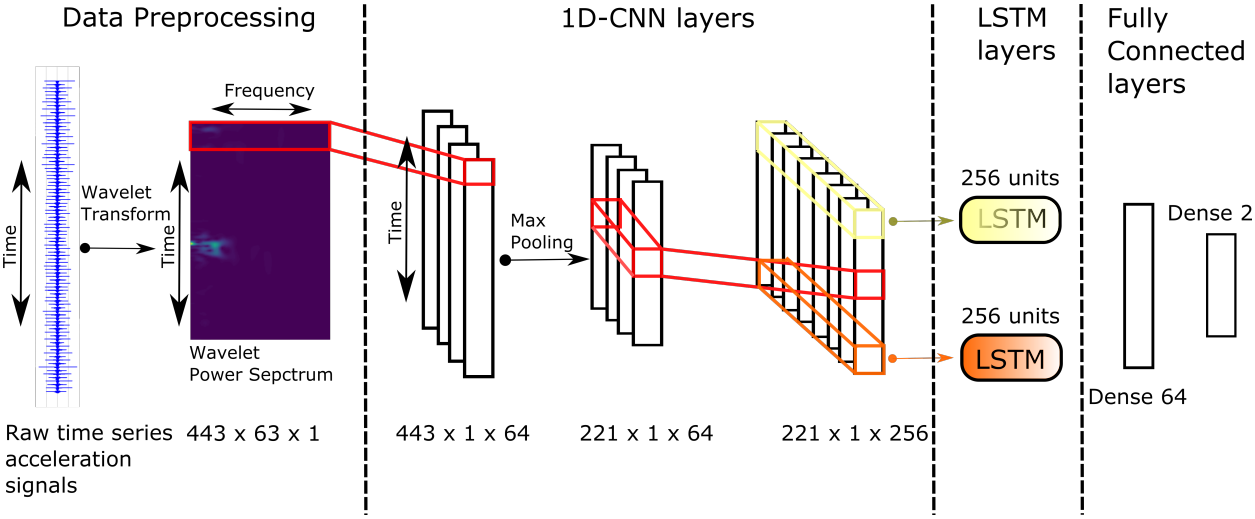
The first step towards being able to use ML or Deep Learning (DL) for defect classification was to preprocess the data to augment the features present in the time series signal. Several approaches to enable the data augmentation was studied and few of them were presented in this section. Following the data preprocessing stage, DL model architectures were designed to classify the different machine defects.

Conditions	**DCNN**	**1D CNN-LSTM**
Information	Spatial	Temporal
Sensor Channels	2 (A1 & A2)	1 (A2)
Wavelet Shape	40 x 40 x 2	443 x 63 x 1
Training time	20.82 min	12.34 min
Inference time	0.051 ms	0.076 ms
Preprocessing	Resizing of wavelet power spectrum required to reduce computation	Can use original wavelet size with minimal impact on computation
Modeling	Not the best approach for time series data	LSTM layer enables modeling of the past information
Sensor Fusion	Multiple sensor power spectrums can be concatenated	Each power spectrum must be sent to the model individually
Convergence	Slow	Fast
Gradient Space	Coarse	Smooth

Table 2.4: Comparison between the two model architectures developed for this study.



(a) DCNN Architecture [21].



(b) 1D CNN-LSTM Architecture.

Figure 2.14: Model architectures developed for the industrial cold forging study.

Data Preprocessing

The raw data collected were labeled according to the defects from Table 2.3. The raw time series data was then segmented such that each segment includes a single cold heading operation, i.e., the cold forging of a single fastener. Based on the machine operating parameters, the segmentation window was set at 263ms. Studies have shown that overlapping the segmentation windows was beneficial [35]. In our study, both the cases of segmentation, overlapping, and non-overlapping were explored. After segmentation, the power spectrum was obtained by Wavelet Transformation using

Morlet Wavelets. The Wavelet Transformation helped identify the frequencies, their respective amplitude, and time of occurrence, within the 263ms segment. The power spectrum for an instance of segmentation for both good and bad signals can be seen in Figure 2.13.

Model Architectures

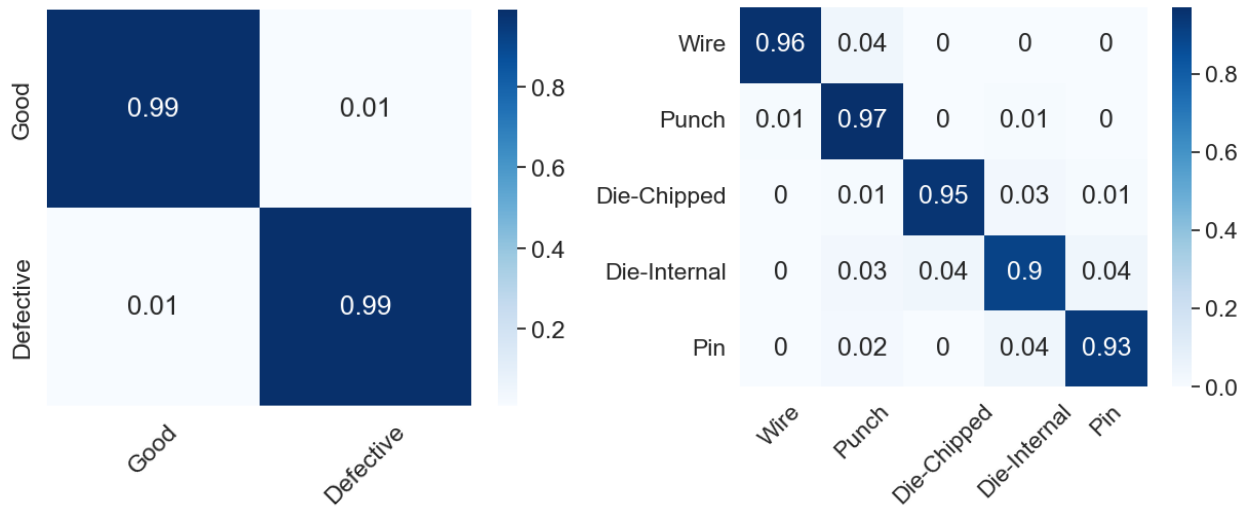
Several studies have explored the benefits of using CNN in image classification [10, 36–38]. Additionally, several architectural design patterns were also studied [39, 40]. In our study, Deep Convolutional Neural Network (DCNN), Figure 2.14a, and a 1-dimensional Convolutional Neural Network - Long Short Term Memory Networks (1D CNN-LSTM), Figure 2.14b, were developed to process the wavelet power spectrum to classify between different fault states. The pros and cons of the two models were experimentally evaluated and presented in Table 2.4.

Supervised Classification

Classification Type	Sensor	Accuracy
Binary Classification	A1&A2	$99.02 \pm 0.39\%$
	A1	$97.71 \pm 0.44\%$
	A2	$98.79 \pm 0.33\%$
Multiclass Classification	A1&A2	$92.66 \pm 1.19\%$
	A1	$85.31 \pm 1.39\%$
	A2	$91.35 \pm 1.94\%$

Table 2.5: Evaluation of defect detection and identification ability of DCNN model [21].

The classification study was performed in two stages: Binary Classification and Multiclass Classification. In the binary classification, we determined if the parameters producing conforming parts (Good) could be classified from all fault conditions (Defect). All the fault states of the machine were combined into a single representative class and were used in the training of the deep learning models, see Figure 2.15a. For the case of multiclass classification, we determined if the type of fault could be identified from each other, see Figure 2.15b. For both types of classification, the training process involved 5-fold cross-validation (CV). We also studied the impact of sensor location on the model's performance by selectively training the deep learning model on data corresponding to specific sensors. The average accuracy and its standard deviation, for both binary and multiclass classification, and different sensor location combinations, can be seen in Table 2.5.



(a) Confusion Matrix - Binary Classification. (b) Confusion Matrix - Multiclass Classification.

Figure 2.15: Confusion matrices for DCNN model.

The distribution corresponding to each defect was visualized to understand the classification process better and determine the relationship between different defect classes. To plot the distribution of the defect classes in 2D, a pre-trained DCNN was modified by removing the softmax layer. For every input data instance from the test dataset, the output of the DCNN model was recorded and labeled depending on the test label. Through the feature extraction process, the information present in the wavelet power spectrum ($40 \times 40 \times 2$) was compressed to a 128×1 vector. t-distributed Stochastic Neighbor Embedding (t-SNE) was then applied to plot each data instance in the test dataset on a 2D plane, see Figure 2.16. From the plot, it can be seen that the distributions of similar defects are closer to each other, i.e., the two types of die defect instances. Hence it can be reliably stated that the deep learning models used in this study were able to extract the essential patterns from the wavelet power spectrum to classify the defects.

Impact of sampling rate

In this section, we wanted to identify the frequencies that contribute to the defect classification process. Hence, during the controlled experiments, the acceleration data was sampled at 8.5kHz instead of the 1.6kHz used during the preliminary experiments. The higher sampling rate enabled us to conduct a feature-importance study by selectively cutting off the frequencies using a zero-

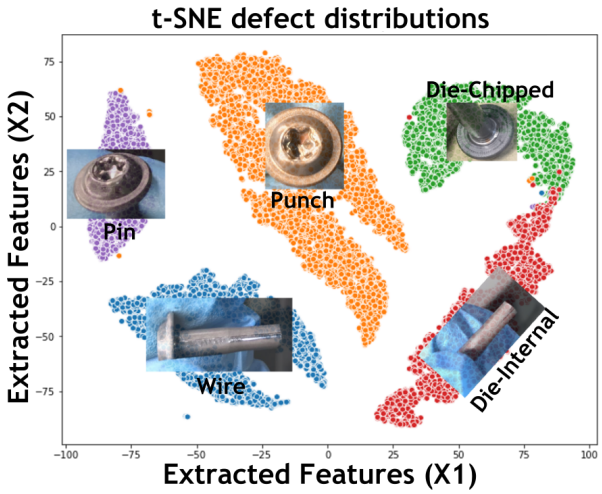


Figure 2.16: t-SNE plot of the defect classes.

		Actual	
Threshold: 0.0172		**Good**	**Defective**
Predicted	**Good**	96.75%	3.25%
	Defective	3.33%	96.67%

Table 2.6: Evaluation of anomaly detection.

phase IIR filter. For every reduced rate, the DCNN model was trained and evaluated for both binary and multiclass classification. From Figure 2.17, it can be seen that the model performance degrades with a reduction in the sampling rate, which is expected, and the rate of performance degradation was steeper for multiclass classification. From this study, the ideal sampling rate was identified to be 2.1kHz.

2.4.3 Anomaly Detection

The key challenge in using the above approach was the need to generate defective data to train the models. In controlled experiments carried out in the cold forging industry, it cost upwards of $2000. Hence in this section, an unsupervised approach to detect defects was studied. The objective of this study was to use only the data corresponding to the good instances and be able to identify the anomalous instances from them.

The data preprocessing steps involved in this study were similar to the ones for the supervised

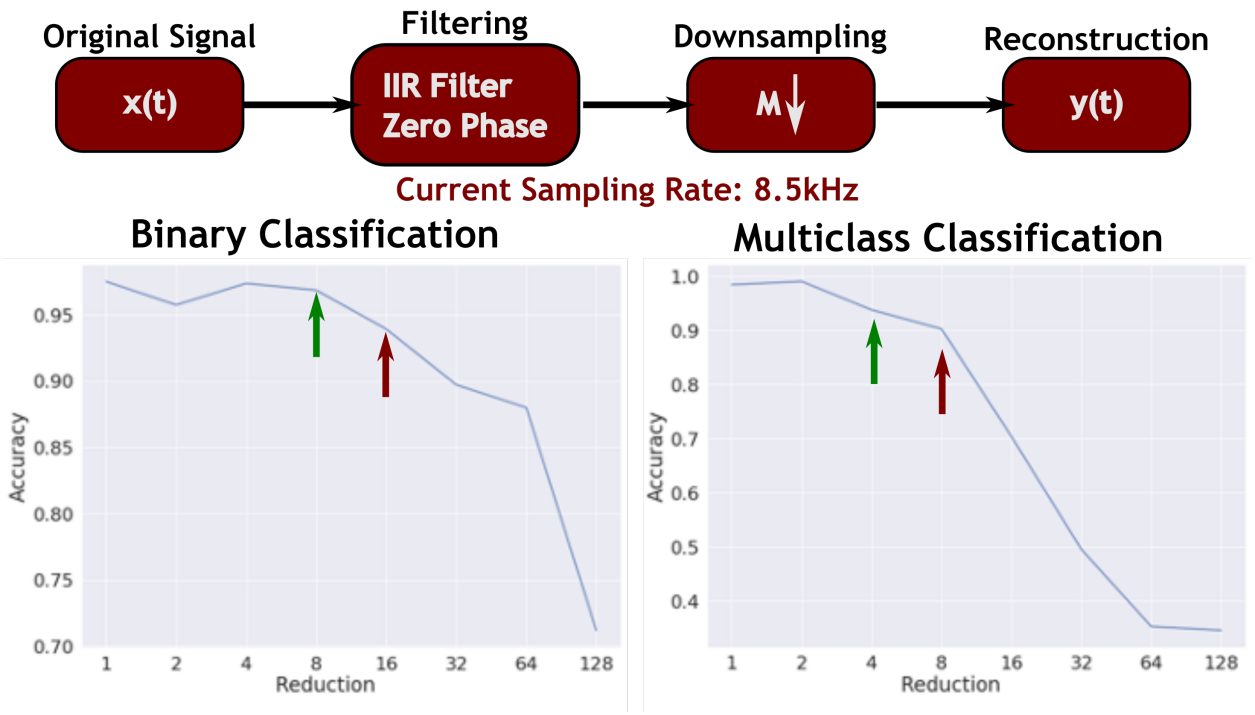


Figure 2.17: Study to determine the impact of sampling rate on model performance.

model development, see Section 2.4.2, with the only difference being reshaping the wavelet power spectrum to $200 \times 200 \times 2$ instead of $40 \times 40 \times 2$.

Model Development

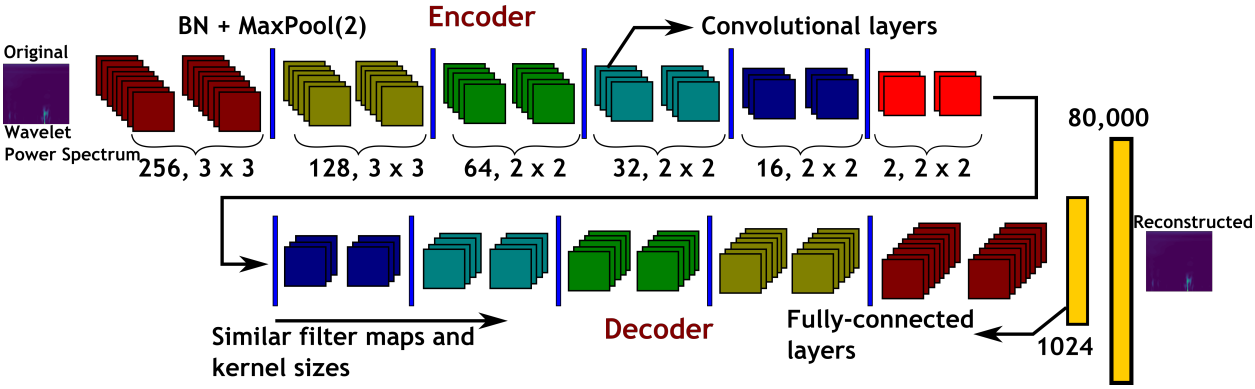


Figure 2.18: Architecture of the Convolutional Autoencoder.

The model used for this study was Convolutional Autoencoders. The autoencoder extracts the information from the input image by compressing the input to a lower dimension, followed by reconstructing the compressed information back to the original image. The architecture of the

autoencoder used in this study is shown in Figure 2.18.

$$lr_{base} + (lr_{max} - lr_{base}) * max(0, (1 - x)) * \lambda^{iterations} \tag{2.1}$$
$$x = |\frac{iterations}{stepsize} - 2 \times cycle + 1|$$

The training process for the autoencoder was challenging. This is because the encoder-decoder halves of the network work against each other leading to the model just copying the information from the input to the output rather than compressing and decompressing the information. Through our study, we found that the cyclical learning rate [41] with exponential envelops, see Eqn 2.1, performed effectively on all instances of the model training process.

Unsupervised Classification

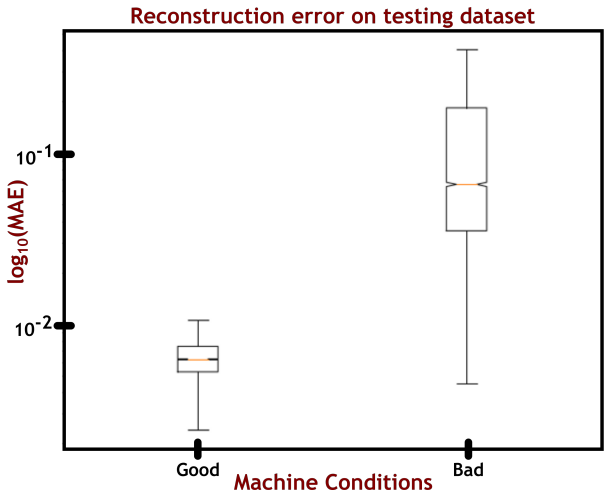


Figure 2.19: Reconstruction error for good and defect data classes.

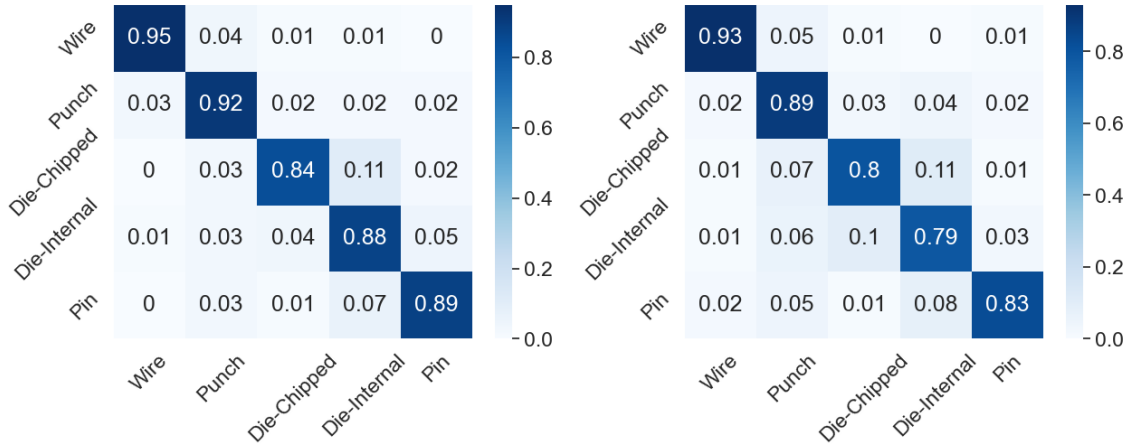
The trained model was then evaluated against an unseen testing dataset. For every new wavelet power spectrum generated from the acceleration data, the ability of the autoencoders to reconstruct the input after encoding and decoding was tested. The reconstruction error was determined as the Mean Absolute Error (MAE) at the pixel level for the input and output images. The threshold for the error was determined from the validation dataset and evaluated against the test dataset. The box plot showing the distribution of the MAEs for both the good and defective instances across the testing dataset can be seen in Figure 2.19. The performance of the convolutional autoencoders

to detect anomalies can be seen in Table 2.6.

2.4.4 Impact of sensor location

In this section, the impact of sensor location variations on model performance was discussed. As previously described, the sensors were installed by augmenting the die block inside the cold forging machine. Two distinct sensor locations were evaluated in this study: A1 and A2, seen in Figure 2.12b. The A2 sensor was directly positioned above the event, while the A1 sensor was displaced to the right. The impact of these different sensor placements on model performance was carefully examined to understand how sensor proximity to the event influences the accuracy and reliability of the model's performance.

To perform this study, DCNN model was trained on the data from different sensor locations and the results can be seen in Figure 2.20. As seen from Figure 2.20a, the sensor located above the event had the highest impact on the model performance. Hence, in cases where the number of sensors needs to be reduced, the A1 sensor can be omitted with a relatively smaller impact on the model performance.



(a) Confusion Matrix - Defect classification only using data from A2 location.

(b) Confusion Matrix - Defect classification only using data from A1 location.

Figure 2.20: Confusion matrix, sensor location importance study.

2.5 Universal Applicability

In this section, the applicability of the developed approach to other machines and processes that were monitored using acceleration sensors was studied. The data were collected from an experimental setup developed by the University of Paderborn, Germany [42]. Several other researchers have used this dataset to evaluate their models and approaches [43].

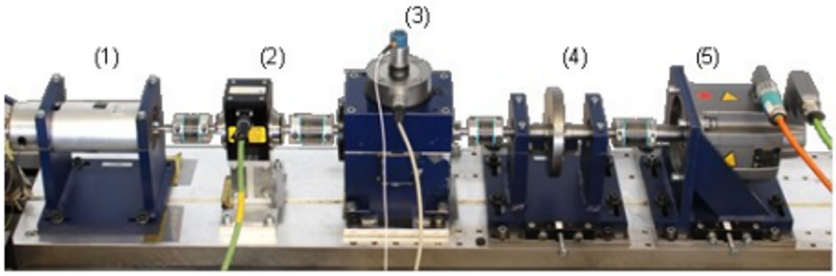


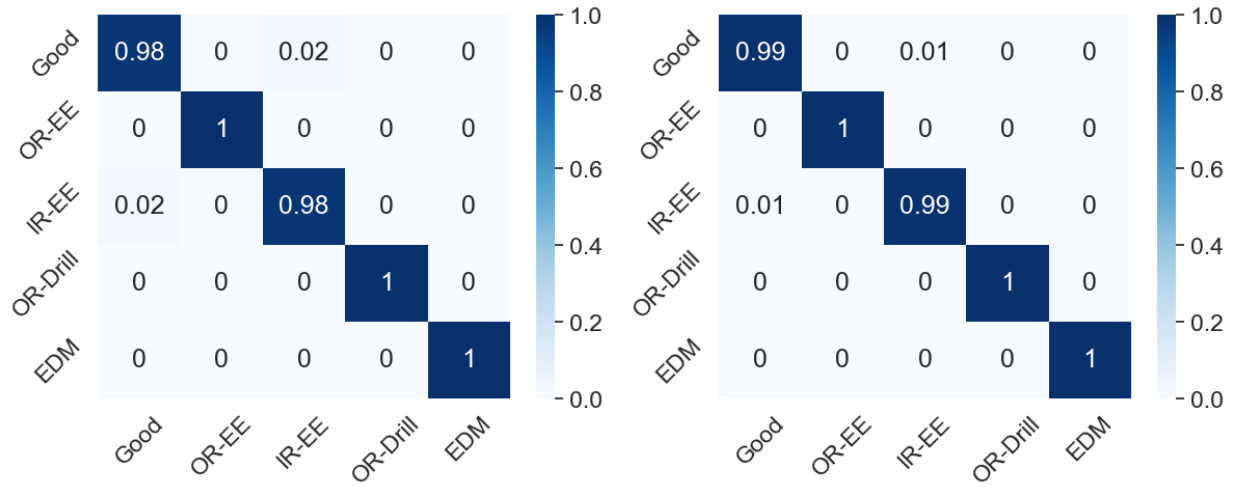
Figure 2.21: Experimental setup [42].

The experimental setup can be seen in Figure 2.21. The test rig consists of several modules: an electric motor (1), a torque measurement shaft (2), a rolling bearing test module (3), a flywheel (4) and a load motor (5). The acceleration sensor was attached to (3) and collected data at a sampling rate of 64kHz. The defects considered in this data set were synthetically developed by electric discharge machining (EDM), drilling, and artificial pitting, to closely reflect the actual ones, see Table 2.7.

Label	**Location**	**Description**
Normal	-	No defects on the bearing
OR-EDM	Outer race	Outer race defect made by EDM
OR-EE	Outer race	Outer race defect made by EE
OR-Drill	Outer race	Outer race defect made by drilling
IR-EE	Inner race	Inner race defect made by EE

Table 2.7: Bearing defects considered in the Paderborn University's dataset and their location.

Our approach was applied to identify and classify the defects. The two model architectures, Figure 2.14, were applied to this dataset and the results can be seen in Figure 2.22. It can be seen that our approach can be extended to other applications that have similar data properties and require feature processing in the time-frequency domain.



(a) DCNN for bearing fault identification.

(b) 1DCNN for bearing fault identification.

Figure 2.22: Fault identification on the Paderborn University's bearing dataset.

2.5.1 Challenges

The approach developed in this chapter effectively detects and classifies faults in manufacturing machines, but it has challenges in **Knowledge Transferability** and **Sensor location sensitivity**.

Knowledge Transferability - Machine Condition Monitoring

Knowledge transferability is the ability to transfer knowledge (trained models) from one machine to another of similar types and processes. It enables the widespread deployment of AI models in manufacturing industries. In this section, we aim to study the challenges associated with knowledge transfer between machines and suggest alternatives/options to overcome them.

		Testing	
		Machine-1	Machine-2
Training	Machine-1	97.5%	43.2%
	Machine-2	52.0%	96.0%

Table 2.8: Knowledge transferability between machines.

To study knowledge transferability, a second set of experiments was conducted on another machine of a similar type performing similar process. As seen in Table 2.8, the ability to transfer

knowledge across these machines was challenging as the model developed on one did not work reliably on the other. One approach to overcome the current challenge was through transfer learning, where a select number of layers were fine-tuned to update the model parameters to a new machine of a similar type. The selected layers were fine-tuned in this work, following the strategy described in Yosinski et al. [44]. Table 2.9 shows the number of trainable layers and the corresponding performance of the model developed on machine-1 and tested on machine-2.

	Accuracy[1]		
	Trained	**Transfer Learned**	
Trainable Layers[2]	**Machine-1**	**Machine-1**	**Machine-2**
2-layer	99.0%	80.5%	90.0%
4-layer	99.0%	91.0%	82.5%
8-layer	99.0%	**96.0**%	**88.5**%

[1] The model trained on Machine-1 and transferred to Machine-2.
[2] Number of layers that were fine-tuned. Model - DCNN.

Table 2.9: Knowledge transfer using transfer learning between machines of similar type.

Knowledge Transferability - Paderborn Bearing Dataset

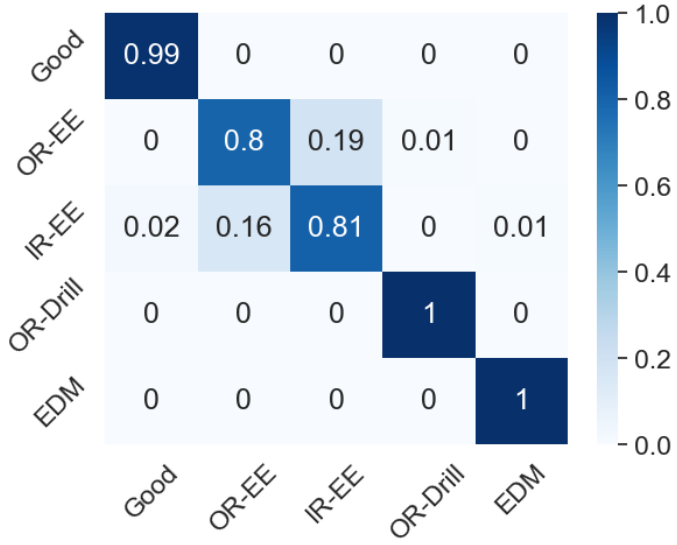


Figure 2.23: Impact of change in the extent of the damage for the defect class OR-EE.

The operating parameters of bearings and fault intensity levels play a crucial role in the performance of condition-monitoring models. The DCNN model developed in this work was successfully applied to the condition monitoring of bearings in Section 2.5. However, in real-world applications,

varying operating conditions of the bearings significantly affected the performance of deep learning models used for condition monitoring. In this section, we explore how different bearing operating parameters impact model performance and discuss their implications for practical deployment.

The fault intensity levels in the Paderborn bearing dataset represent the extent of damage to the bearings. The dataset includes three levels of fault severity, with intensity increasing from Level 1 to Level 3. However, the fault intensities are not progressive over time; rather, they occur instantaneously based on the bearing's condition. In the initial analysis, the impact of varying fault intensities on the performance of the deep learning model was evaluated. The DCNN model was trained using fault intensity Levels 1 and 2 and then tested on Level 3 to assess its ability to generalize to more severe fault conditions. In Figure 2.23, the impact of variation in fault intensity for the defect class EE-OR is shown. The DCNN model was developed for intensity levels 1 and 2 and was tested on level 3. It can be seen that there was a performance drop for the OR-EE defect class compared to Figure 2.22a, otherwise, the "Good" class was predicted without any performance drop.

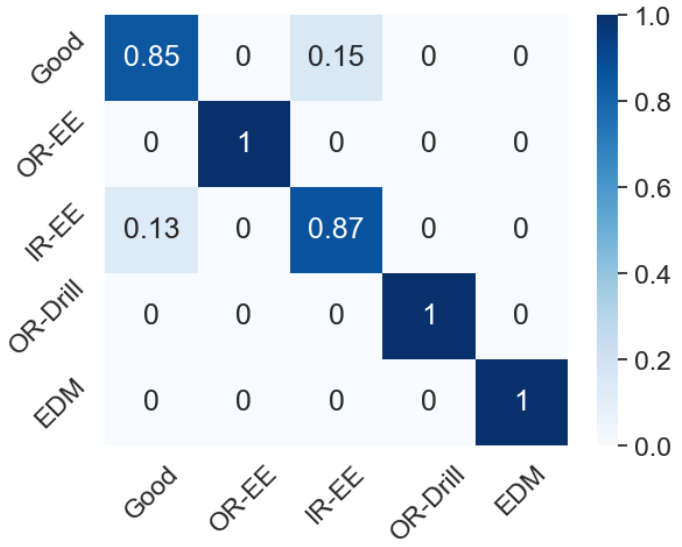
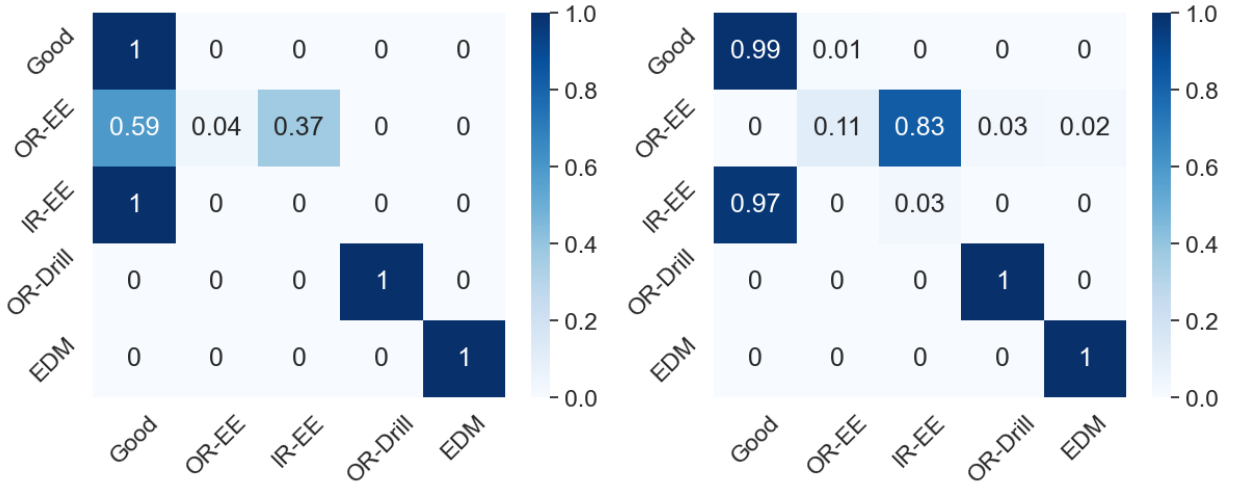


Figure 2.24: Impact of Load Torque variation from 0.7Nm to 0.1Nm on model performance.

The operating parameters considered in this study were load torque and radial force, as bearings frequently experience variations in these parameters during routine operations. To assess the robustness of the model, the impact of these parameter changes on its performance was evaluated.

Figure 2.24 illustrates the model's performance when the load torque was varied from 0.7Nm to 0.1Nm after model training. Although the performance impact was minimal, and the DCNN model architecture effectively accommodated these variations, it is important to note that load torque fluctuations should be continuously monitored, and if necessary, the model may need retraining to maintain performance.



(a) Radial force variation - DCNN model.

(b) Radial force variation - 1DCNN-LSTM model.

Figure 2.25: Impact of radial force variation from 1000N to 400N on model performance.

Similarly, Figure 2.25 shows the impact of radial force variations on model performance. Unlike load torque, changes in radial force had a more pronounced effect. While the "Good" class remained unaffected, the model exhibited a significant drop in performance for the "OR-EE" and "IR-EE" fault categories, indicating the model's sensitivity to radial force variations in these fault types.

2.6 Domain Adaptation

Deep learning and machine learning algorithms have made a significant impact in fault detection and classification, as discussed in Section 2.4. A fundamental requirement for these models is that the input data samples provided during inference align with the distribution of the training dataset used in model development. When the input data distribution diverges from the training data, model performance can degrade substantially, as seen in Section 2.5.1. This distribution mismatch is a common issue when transferring a model from a laboratory setting to an industrial

environment or between machines of similar OEMs executing the same processes. Regardless of the specific features extracted from the time series sensor data, this mismatch often leads to poor model performance.

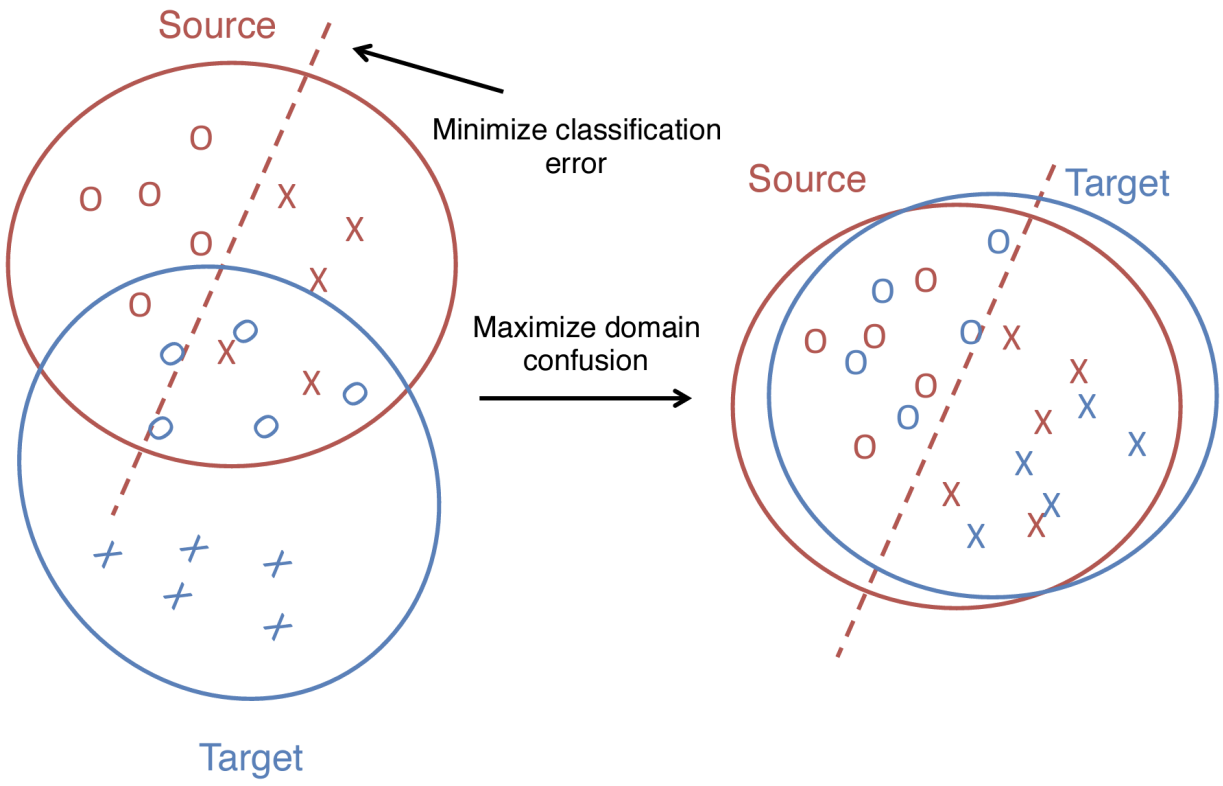


Figure 2.26: Optimizing an objective that simultaneously minimizes classification error and maximizes domain confusion [45].

Domain adaptation is a technique where the ML/DL model is trained on one data distribution (source domain) to perform effectively on a different, yet related, data distribution (target distribution). It enables us to address the distribution mismatch, that often occurs when the models are applied across various environments, equipment, or processes. This process is called cross-domain learning, where the features learned represent both the source domain and target domain, as seen in Figure 2.26.

In equipment condition monitoring, acquiring data for the machine's normal operating states is generally more feasible than gathering data on defect states. As discussed in previous sections, generating data for defect conditions can be expensive. Given the challenges in knowledge transferability, the need to create defect data across machines of similar types and processes quickly

becomes impractical. Therefore, in this section, we explore a domain adaptation approach to enhance knowledge transferability between distinct but related data distributions. In our approach, we consider a scenario where the source domain contains data on all the states of a machine, normal and different defects, whereas the target domain contains data on at least one machine state. This scenario is more common in machine condition monitoring monitoring, as at the very minimum it would be possible to collect sensor data on the normal operating states of the machine. The work done in this section is three folds:

1. An approach that enables domain adaptation using DCNN was developed. The DCNN model was split into an encoder head and a classifier head to to facilitate cross-domain feature extraction and learning.

2. The training process and the associated loss functions were discussed in detail. During training, the model utilizes both source domain data and target domain data to learn features that are common and relevant across domains. To prevent overfitting on the source domain, where cross-entropy loss was calculated, both L1 and L2 regularization techniques were applied. These regularization methods were essential for ensuring the model effectively learned features applicable to both domains. Finally, an approach to facilitate both **Domain Invariance** and **Task Discriminability** is discussed.

3. The developed model was evaluated to assess its domain adaptation capabilities in an unseen target scenario. These scenarios included variations in operational parameters for bearing condition monitoring.

Domain Invariance vs. Task Discriminability: In domain adaptation, achieving domain alignment can sometimes result in insufficient preservation of task-relevant features. This challenge often arises when the focus is primarily on making features appear similar across domains, potentially at the expense of the model's discriminatory power. Discriminative ability is essential for high-accuracy classification; without it, the model may perform poorly on the target domain, even if feature spaces are aligned. Figure 2.27 illustrates the contrast between effective and ineffective domain adaptation approaches.

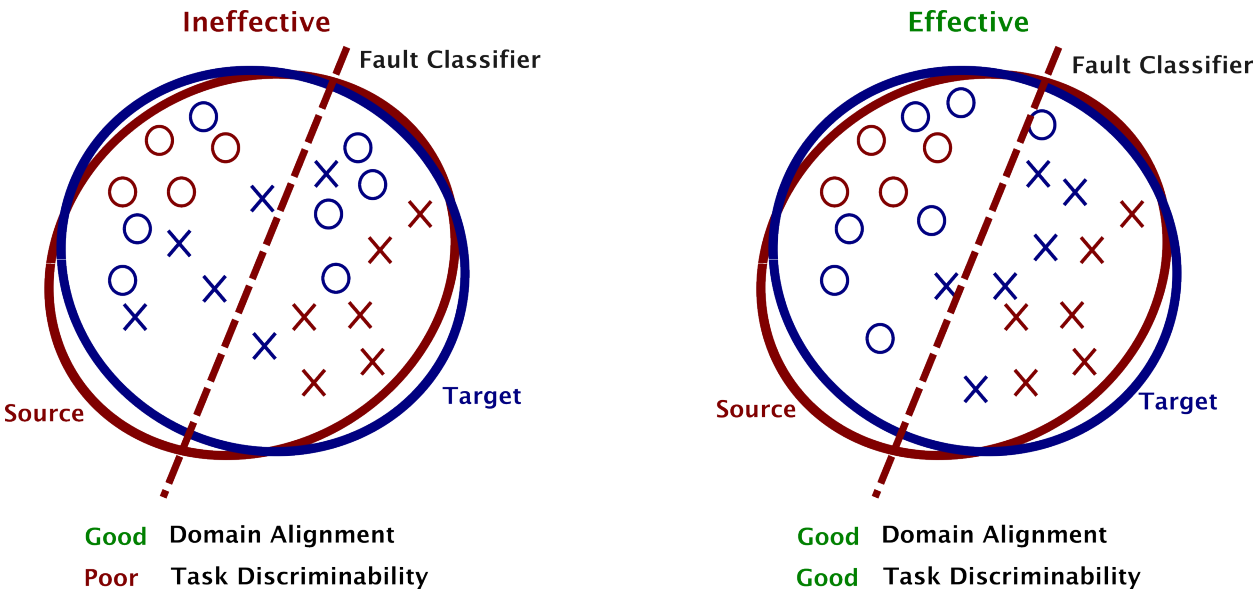


Figure 2.27: The importance of Domain Alignment vs Task Discriminability.

2.6.1 Problem Formulation

Domain adaptation is particularly useful in scenarios where the training data distribution (source domain) differs from the testing data distribution (target domain). By facilitating generalization, domain adaptation enables deep learning models to perform effectively in new environments. When the source and target distributions are identical, domain adaptation becomes unnecessary. In fault detection, however, differences in data distributions between source and target domains are frequent [19]. These distribution variations may occur immediately, such as when transferring knowledge from laboratory settings to industrial applications, or they may evolve over time, as in the case of knowledge transfer between similar machines.

1. A typical domain $\mathcal{D}$ is represented by $\mathcal{D} = \{\mathcal{X}, P(X)\}$, where X is the feature space and $P(X)$ is the marginal probability distribution. The feature space X is represented as $X = \{x_i\}_{i=1}^{n} \in \mathcal{X}$

2. The task (fault identification) $\mathcal{T} = \{\mathcal{Y}, f(X)\}$, where $f(X)$ is the classifier represented by conditional probability distribution $f(X) = P(Y|X)$, and $\mathcal{Y}$ is the label space with $Y = \{y_i\}_{i=1}^{n} \in \mathcal{Y}$.

3. In **domain adaptation**, the source domain and target domain are represented as $\mathcal{D}_s$ and $\mathcal{D}_t$, respectively. They tend to have different feature space, $\mathcal{X}_s \neq \mathcal{X}_t$, and different marginal probability distribution, $P(X_s) \neq P(X_t)$. The key factor for domain adaptation considered for this study is both source and target domain share the same tasks, i.e., $\mathcal{T}_s = \mathcal{T}_t$ when $\mathcal{D}_s \neq \mathcal{D}_t$.

4. Domain adaptation is very similar to transfer learning. In transfer learning, the source and target domains along with the task space are different, i.e., $\mathcal{D}_s \neq \mathcal{D}_t \quad \& \quad \mathcal{T}_s \neq \mathcal{T}_t$, whereas in domain adaptation, $\mathcal{D}_s \neq \mathcal{D}_t \quad \& \quad \mathcal{T}_s = \mathcal{T}_t$.

Liu et al. [19] classified the domain adaptation process into four categories: homogeneous data tasks, heterogeneous feature tasks, heterogeneous label tasks, and heterogeneous data tasks. Homogeneous data tasks involve both the source and target domain share the same feature and labels spaces while feature distribution divergence exists, i.e., $\mathcal{D}_s = \mathcal{D}_t$ and $\mathcal{Y}_s = \mathcal{Y}_t$, while $P(X_s) \neq P(X_t)$. This scenario occurs when there is a change in working conditions, sensor locations, and deteriorating fault severities. In heterogeneous data tasks, the source and target domain differ in feature spaces but they share the same label spaces, i.e., $\mathcal{D}_s \neq \mathcal{D}_t$ while $\mathcal{Y}_s = \mathcal{Y}_t$. This scenario occurs when transferring between different sensor data types or between different sensor sampling rates. In our study, we will consider scenarios for homogeneous data tasks and heterogeneous feature tasks.

In our research, the source domain $\mathcal{D}_s$ comprises controlled experiments conducted to collect data across various machine states, including normal operating conditions and the different defect states commonly encountered during machine operation. The target domain $\mathcal{D}_t$ represents the varying working conditions or operating parameters of the same machine, or a completely different machine of similar type conducting the same process. For instance, in bearing condition monitoring, different working conditions might include changes in axial load, radial load, RPM, and defect intensity. The objective, therefore, is to transfer knowledge between these two domains to enable reliable and robust condition monitoring for manufacturing machines and processes.

2.6.2 Related Work

Extensive work has been conducted on domain transfer learning, with recent efforts focusing on transferring knowledge from a source domain to a target domain in both supervised and unsupervised fashion. Recent research has applied Maximum Mean Discrepancy (MMD) loss to measure and minimize differences between the source and target domains. For example, [45] used MMD to regularize the model training process alongside classification loss, enabling the DL model to learn both discriminative and domain-invariant features. In Deep Adaptation Network (DAN) [46], MMD was applied to the deep features of a CNN within a reproducing kernel Hilbert space, allowing for effective domain adaptation. Other approaches for domain adaptation incorporate adversarial loss, similar to Generative Adversarial Network (GAN)s. Tzeng et al. [47] proposed an approach called Adversarial Discriminative Domain Adaptation (ADDA), combining discriminative modeling, untied weight sharing, and a GAN loss to facilitate domain adaptation. Domain adaptation for condition monitoring has also been studied. Liu et al. [19] discussed the taxonomy of knowledge transfer and a review of different knowledge transfer techniques for fault detection of rotary machines. Lu et al. [48] developed an approach called Deep neural network for domain Adaptation in Fault Diagnosis (DAFD). DAFD uses MMD and model weights regularization to enable knowledge transfer between source and target domains.

For machine condition monitoring, particularly for identifying and classifying faults, the ADDA approach offers significant benefits:

1. Untied Weight Sharing: In ADDA, weights are not shared between the source and target domain models, allowing for the training of a target model that is distinct from the source model. This approach is advantageous in fault detection, as the target domain dataset is often semi- or partially labeled. For instance, in the industrial cold-forging study discussed earlier, the common class was the “Normal” or “Good” scenario. Thus, even in cases with limited labels, obtaining data for a machine’s normal operating state is likely possible. Additionally, it was shown in Rozantsev, Salzmann, and Fua [49] that partially shared weights can lead to effective source to target domain adaptation, in any learning setting.

2. Adversarial Loss Benefits: Although adversarial loss introduces the challenge of balancing two competing objectives, it offers insights into the complexities of domain adaptation for fault classification. These insights are especially valuable for smaller datasets typical in fault detection, where issues such as domain alignment versus task discriminability need to be carefully managed.

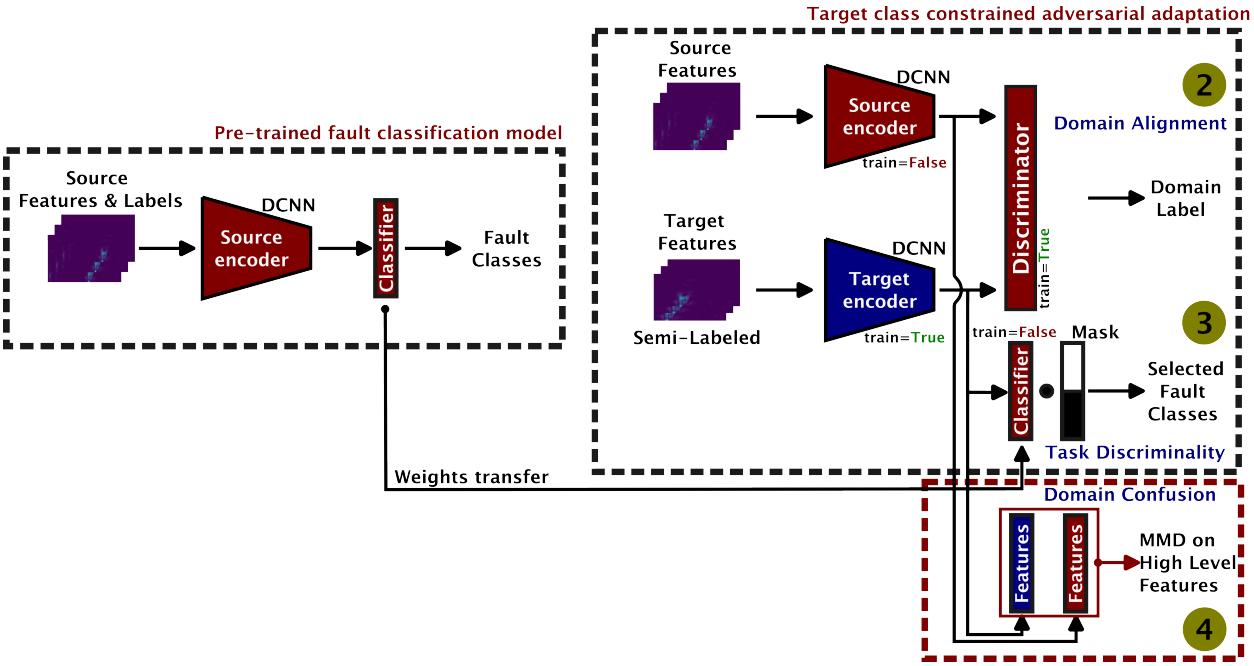


Figure 2.28: Approach to enable domain alignment and task discriminability to transfer knowledge from the source domain to target domain.

2.6.3 Methodology

Most domain adaptation methods can be classified into two main categories [48]: instance transfer and subspace/feature learning. In instance transfer, the source data is reweighted based on the shared information contained within the target data. In subspace learning, a feature subspace is identified where a classifier trained on the source data can generalize effectively to the target domain. The primary objective of our work was to transfer knowledge from the source domain to the target domain, enabling good domain alignment and task discriminability for the cases of machine condition monitoring.

Adversarial Domain Adaptation - Class Constrained with Maximum Mean Discrepancy (MMD)

Our approach extends ADDA [47] by introducing a new loss function, referred to as the **target-task loss**, on the labeled task domain instances. This additional loss complements the **domain loss**, which promotes domain invariance between the source and target domains, by enhancing task discriminability. Figure 2.28 provides an overview of the training process that incorporates both domain loss and target-task loss. While traditional ADDA, as described in Tzeng et al. [47], is effective in aligning domains, it often falls short in task discrimination. This limitation is particularly relevant in condition monitoring and fault detection scenarios, where obtaining data on defective states is challenging and dataset size is limited.

By constraining adversarial domain adaptation using the available fault classes within the target domain, our approach aims to achieve both domain alignment and improved task discriminability. This strategy not only addresses the data limitations inherent to condition monitoring but also strengthens the model's ability to distinguish between different operational states in the target domain.

For unsupervised/semi-supervised domain adaptation, in the source domain, D_s, the features X_s and labels Y_s were drawn from the distribution $p_s(x, y)$. In the target domain D_t, the features X_t were drawn from the distribution $p_t(x, y)$. In the target domain, the labels Y_t are either not available (unsupervised) or partially available (semi-supervised),

$$Z_t = \{y_i \in Y_t \mid \text{label } y_i \text{ is available in the semi-supervised setting}\} \quad \& \quad Z_t \subset Y_t$$

The semi-supervised domain adaptation case is particularly useful in machine condition monitoring, as at the very minimum, data is available on "Normal" operating state of the machine. In adversarial domain adaptation, the objective is to learn the target representations M_t and classifier f_t that can correctly classify the target domain defects into one of the K defect categories. In ADDA [47], the training process is guided by three objectives:

1. **Source Classification Loss:** This objective ensures that the source features X_s are accurately classified by the source classifier. This is formulated as a cross-entropy loss over the labeled source data:

$$\min_{M_s, f} \quad \mathcal{L}_s^{cls}(X_s, Y_s) = -\mathbb{E}_{(\mathrm{x}_s, y_s)\sim(X_s, Y_s)} \sum_{k=1}^{K} \mathbb{1}_{[k=y_s]} \log f(M_s(\mathrm{x}_s)) \tag{2.2}$$

2. **Domain Alignment Loss:** This objective has two components: $\mathcal{L}_{adv_D}$ is the standard classification loss that helps in training the discriminator D, enabling it to separate the source domain features from target domain features. $\mathcal{L}_{adv_M}$ is another standard classification loss used to learn the mapping M_t for the target encoder.

$$\begin{aligned} \min_{D} \quad \mathcal{L}_{adv_D}(X_s, X_t, M_s, M_t) = &-\mathbb{E}_{\mathrm{x}_s\sim X_s}\left[\log D(M_s(\mathrm{x}_s))\right] \\ &-\mathbb{E}_{\mathrm{x}_t\sim X_t}\left[\log(1 - D(M_t(\mathrm{x}_t)))\right] \end{aligned} \tag{2.3}$$

$$\min_{M_s, M_t} \quad \mathcal{L}_{adv_M}(X_s, X_t, D) = -\mathbb{E}_{\mathrm{x}_t\sim X_t}\left[\log D(M_t(\mathrm{x}_t))\right] \tag{2.4}$$

3. **Target Task Loss**: This objective has been added to ADDA to improve the task discriminability for condition monitoring of machines. The objective ensures that target domain features corresponding to partially available labels (filtered labels 0, 1, 2, etc.,) are classified correctly. By detaching the target encoder output (after weight sharing) and passing it through the source classifier, we ensure that only the target encoder parameters are updated, without backpropagating gradients to the source classifier.

$$\min_{M_t} \quad \mathcal{L}_t^{cls}(X_t, Z_t) = -\mathbb{E}_{(\mathrm{x}_t, z_t)\sim(X_t, Z_t)} \sum_{k=1}^{n} \mathbb{1}_{[k=z_t]} \log f(M_t(\mathrm{x}_t)) \tag{2.5}$$

where:

(a) $Z_t \subset Y_t$ are the particular labels in the target domain.

(b) n is the number of available labels for the target domain.

4. **MMD Loss:** To further improve domain alignment enabling domain confusion, we have added another loss called the MMD loss. The MMD loss was applied to the features at the output of the source and target encoders, as seen in Figure 2.28. The MMD loss is a statistical measure used to compare two probability distributions by mapping them into a Reproducing Kernel Hilbert Space (RKHS) and measuring the distance between their means in that space. The source and target encoders extract the features from the input wavelet power spectrums. By applying the MMD loss on the extracted features from the source and target encoders we aim to minimize the distance between the two distributions. This will further enhance the domain alignment between the source and target domains. Theoretically, the MMD loss and the domain alignment loss in Equations 2.3 and 2.4 aims to achieve the same objective, the reasons on why both of them were used was discussed later in this section.

$$\mathcal{L}_{\text{mmd}}(g_s, g_t) = ||\mathbb{E}_{x\sim g_s}[\phi(x)] - \mathbb{E}_{y\sim g_t}[\phi(y)]||^2 \tag{2.6}$$

where:

(a) g_s, g_t, correspond to the feature output at the source and target encoders; The source and target distributions.

(b) $\phi(\cdot)$ is a feature mapping function to the RKHS.

(c) $\mathbb{E}$ represents the expected value of the feature representation.

5. **Consistency Regularization**: As a result of incorporating the target task loss objective, there is a risk of the target encoder overfitting the limited labeled data in the target domain during training. Such overfitting can degrade the model's performance and hinder its generalization ability. To mitigate this, consistency regularization is employed to ensure the model performs robustly in the vicinity of the available training samples [50]. In the context of

condition monitoring for machines and processes, training samples often exhibit relationships based on shared defect categories. The model's training process is stabilized by introducing a consistency loss into the training objective, enhancing its reliability across similar samples/defect features. This approach helps improve the model's generalization and robustness in practical deployment scenarios. The consistency regularization applied during training involves perturbing the target domain features with noise from standard normal distribution and computing Mean Squared Error (MSE).

$$\mathcal{L}_{\mathrm{c}} = \mathrm{MSE}\left(f_t(M_t(X_t))\,,\; f_t(M_t(X_t + 0.01 \cdot \mathcal{N}(0, I)))\right) \tag{2.7}$$

where f_t is the classifier for the target domain, M_t is the target domain encoder, $0.01 \cdot \mathcal{N}(0, I)$ is the noise from standard normal distribution scaled by 0.01, and MSE is the Mean Squared Error.

The effective loss for training the target encoder is given in Equation 2.8.

$$\min \quad \mathcal{L} = \mathcal{L}_{adv_M}\left(X_s, X_t, D\right) + \lambda_1 \cdot \mathcal{L}_t^{cls}\left(X_t, Z_t\right) + \lambda_2 \cdot \mathcal{L}_c(X_t, M_t, f_t) + \lambda_3 \cdot \mathcal{L}_{\mathrm{mmd}}(g_s, g_t) \tag{2.8}$$

where λ_1 is a regularization parameter that controls the balance between domain alignment and task discriminability. A higher value of λ_1 can increase the influence of the target task loss, enhancing task-specific feature learning in the target domain. However, setting λ_1 too high risks overfitting the target classes, which can lead to diminished domain alignment and an overall drop in generalization. The implications of a high λ_1 include reduced model performance on source domain features and limited ability to generalize to other classes within the target domain.

An overview of the training process is shown in Figure 2.28 for practical and implementation purposes. The training process is split into two stages. In the first stage, a DCNN model, initially developed for condition monitoring, was trained on the source domain data. This model was

then modified by dividing it into two parts: an encoder, referred to as the source encoder, and a classification layer (comprising a linear layer with softmax activation), termed the source classifier. For the target domain, a new model—called the target encoder—was initialized with an identical architecture to the source encoder, and its weights were set to those from the source domain training. The source classifier was used as the classification layer for the target encoder, allowing it to classify the various target domain classes.

The second stage involved adversarial adaptation, where a discriminator was trained to align the source and target domain features by making them indistinguishable. This alignment was achieved by ensuring the discriminator could not differentiate between source and target domain features. Additionally, task discriminability was preserved by applying a masked target task loss during adaptation, thereby enhancing classification accuracy on the target domain while maintaining domain alignment.

2.6.4 Evaluation

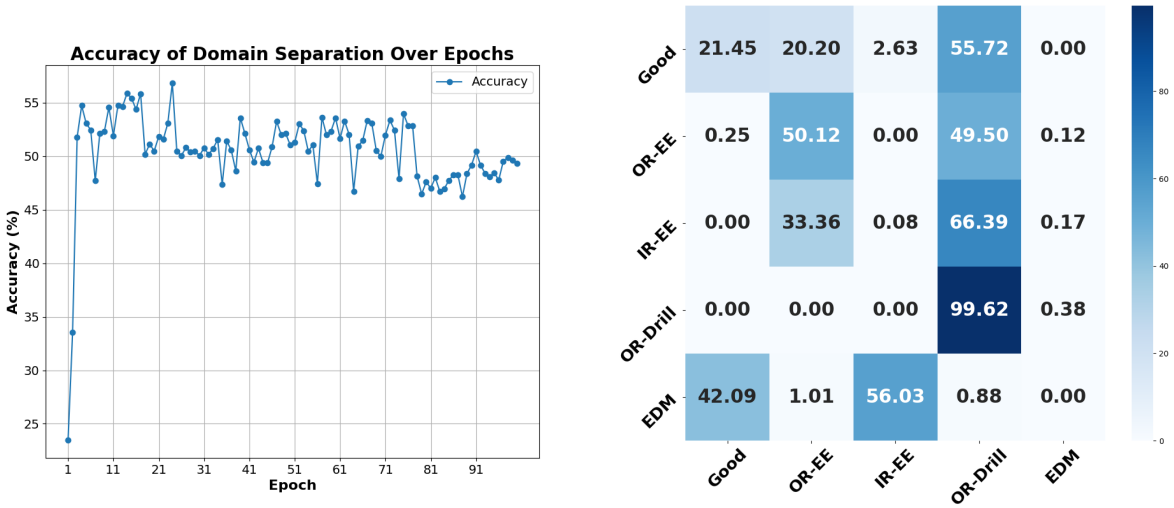
To effectively evaluate our approach, we used the open-source dataset provided by the University of Paderborn, Germany, as detailed in Section 2.5. This dataset offers a comprehensive range of scenarios, with data collected under various operating conditions for both "Normal" and "Defective" bearings. The specific operating parameters considered during data collection are summarized in Table 2.10. These parameters serve as the source and target domains for domain adaptation in bearing condition monitoring. When an operating parameter changes, model performance can decline, depending on the type and extent of variation. For instance, variations in bearing RPM are likely to have the most pronounced effect on model performance, as they directly influence frequency and time-frequency domain features critical to condition monitoring. Therefore, any change in operating parameters is treated as a shift in domain, necessitating knowledge transfer.

It is essential to understand the model training process for this work. In Phase I, the source encoder and classifier were trained separately on the source domain data. During the training process, early stopping was implemented to terminate training if there were no improvements in the validation loss. The best-performing model—defined as the one achieving the lowest validation

loss—was saved for use in Phase II domain adaptation. In Phase II, the pre-trained source encoder and classifier from Phase I were loaded, and the domain adaptation process began. Both the source domain and target domain data were used for the training, in the target domain the normal/good class along with the two other defects were assumed to be labeled whereas the remaining data was assumed to be unavailable. Training continued until the discriminator, responsible for distinguishing between source and target domains, achieved an accuracy of approximately 50%. The best-performing target encoder, determined by the lowest cross-entropy loss on the target domain, was tracked for the subsequent training epochs. The best-performing target encoder was then used to generate the confusion matrices discussed in the subsequent sections.

Table 2.10: Experimental Settings provided by University of Paderborn, Germany.

No.	Rotational Speed (rpm)	Load Torque (Nm)	Radial Force (N)
0	1500	0.7	1000
1	900	0.7	1000
2	1500	0.1	1000
3	1500	0.7	400



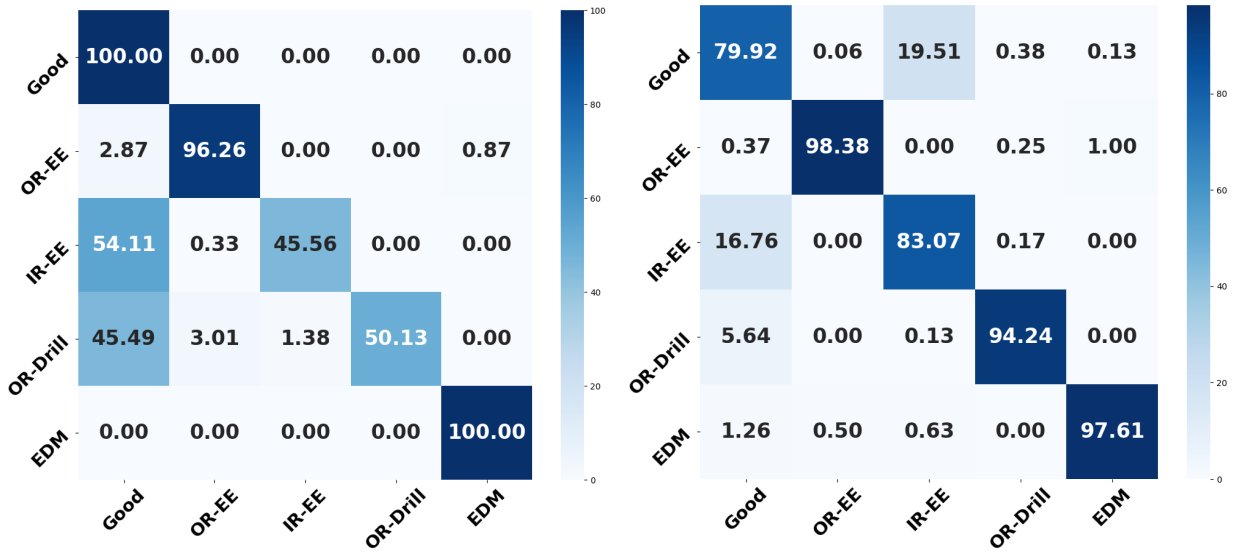
(a) Performance of the discriminator.

(b) Target classifier on target domain.

Figure 2.29: Evaluating the performance of ADDA in enabling knowledge transfer from an operating parameter with a radial force 1000N to 400N.

The evaluation in this section is structured in two parts. First, we discuss the impact of integrating the target task loss into ADDA. Following this, we evaluate our approach's ability to facilitate knowledge transfer across different operating conditions, focusing on the following parameters: (i) changes in bearing RPM, (ii) fluctuations in radial force, and (iii) using artificially

generated defects to learn about real ones. The variations in load torque were not considered for this study because there is no domain shift when the load torque changes, as can be seen from Figure 2.23 in Section 2.5.1. The approach described in ADDA was tested on the condition monitoring problem domain to assess it capabilities, seen in Figure 2.29, ADDA was tested only on the operating parameter variation—change in the radial force from 1000N to 400N. From Figure 2.29a, it can be seen that the discriminator has converged to an accuracy of 50%, meaning that there is domain alignment, but the task discriminability was poor, as can be seen in Figure 2.29b. Although, it is worth noting that domain alignment by itself had enabled an improvement in classification performance for the defect class "OR-Drill", where the classification performance increased from 50.13% (no domain alignment) to 99.50% after domain alignment. For an overview of the source domain model performance on the variation in radial force, please refer to Figure 2.30a.



(a) Source classifier on target domain.

(b) Target classifier on target domain.

Figure 2.30: Domain adaptation enabling knowledge transfer from the source domain (Radial force 1000N) to target domain (Radial force 400N).

In Figure 2.30, we illustrate the knowledge transfer from the source domain using our approach, where a radial force of 1000N was applied to the bearing, to the target domain, where the radial force was reduced to 400N. When using the source domain model on the signals from the 400N radial force scenario, a noticeable drop in performance is observed, as depicted in Figure 2.30a. However, the target domain model developed through our approach significantly improves performance for the target domain with a 400N radial force, depicted in Figure 2.30b. The performance improvements

seem consistent across all defect classes considered in this study. Similarly, Figure 2.31 examines the knowledge transferability from a source domain where the bearing rpm was 1500 to a target domain where the rpm was reduced to 900. In both cases, the proposed approach significantly improved the performance of the condition monitoring models on the target domain with limited labeled data. It could also be noticed that there is a performance drop for the defect class "EDM", and this defect class tends to be classified as "IR-EE" by the target domain model, circled in red. The reason could be attributed to the fact that the EDM (Electrical Discharge Machining) defect is on the inner race of the bearing and has defect features that resemble the Electric Engraver (EE) on the inner race. Hence, the vibration signal features for these two cases, EDM and IR-EE, closely resemble each other leading to the model misclassifying them without supervised learning. The F1-Scores for each category of knowledge transfer considered for this evaluation can be seen in Table 2.11. These results underscore the robustness and effectiveness of our domain adaptation methodology in addressing variations in operating parameters, thereby enhancing the generalizability of condition monitoring models.

(a) Source classifier on target domain.

(b) Target classifier on target domain.

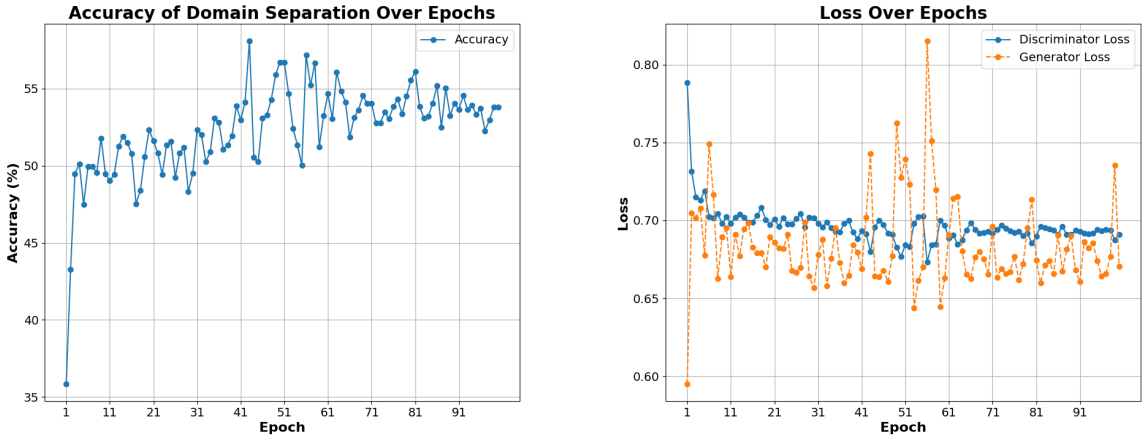
Figure 2.31: Domain adaptation enabling knowledge transfer from source domain (1500 RPM) to target domain (900 RPM).

In addition to comparing the target encoder performance between the source domain and target domain, it becomes equally important to evaluate the performance of the discriminator. In an ideal case, the domain alignment happens when the discriminator is unable to separate the features from

Table 2.11: F1-Scores Comparison: Knowledge Transfer from Source to Target Domain.

Knowledge Transfer Scenario	Source Model F1-Score	Target Model F1-Score
RPM: 1500 to 900	0.4979	**0.8778**
Radial Force: 1000N to 400N	0.7983	**0.8985**
Defect Type: Artificial to Real	0.5996	**0.7674**

the source domain and target domain. In Figure 2.32 and Figure 2.33, the performance of the discriminator in separating the source domain from the target domain for the case of knowledge transfer between the bearing operating conditions is shown. It can be seen from both figures that the discriminator is unable to separate the target domain from the source domain, confirming the fact that there is indeed domain alignment happening.

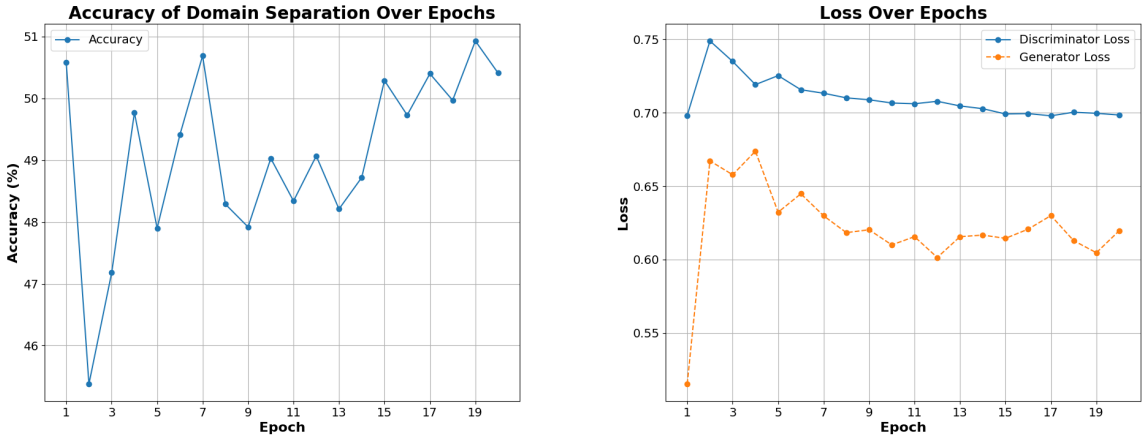


(a) Separating source domain from target domain.

(b) Generator and discriminator loss.

Figure 2.32: Evaluating the performance of the discriminator during training for the case of source domain—Radial Force 1000N and target domain—Radial Force 400N. The accuracy of 50% shows that the discriminator is unable to separate between the two domains, which is good, as it validates the effectiveness of the training process.

The final evaluation in this performed in this section involves enabling knowledge transfer between artificially generated defects on bearings to real defects on bearings, which occurs naturally after certain operation hours. Developing an open-source dataset containing both artificial and real defects is challenging, as it can be expensive and time-consuming. Hence, to conduct this evaluation, vibration sensor signals were selectively chosen from the University of Paderborn's dataset corresponding to real and artificial defects. Hence, for this analysis, the number of classes was reduced to three from five. The three classes corresponding to different operating conditions



(a) Separating source domain from target domain.

(b) Generator and discriminator loss.

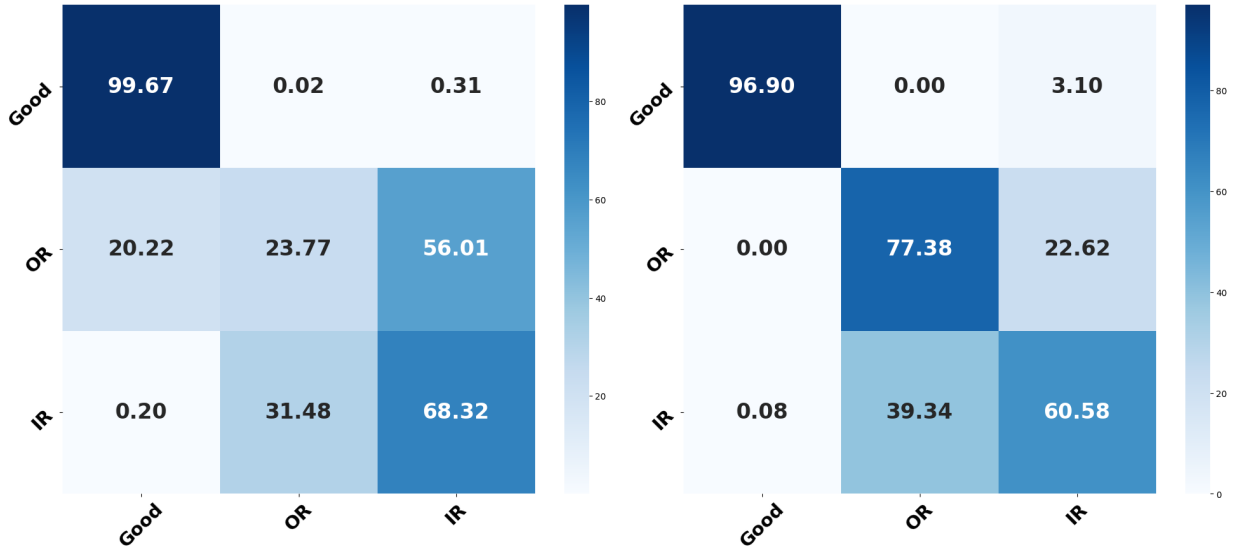
Figure 2.33: Evaluating the performance of the discriminator during training for the case of source domain—1500 RPM and target domain—900 RPM. The accuracy of 50% shows that the discriminator is unable to separate between the two domains, which is good, as it validates the effectiveness of the training process.

of the bearing are: "Good", "OR", and "IR". A description of the defects is given in Table 2.12. Figure 2.34 shows our capability of transferring knowledge from the source domain (Artificial defects generated on real bearings) to the target domain (Real defects on real bearings occurring naturally).

Label	**Description**
Good	Bearing operating in an acceptable state, anywhere from 1h to $>$ 50h of run-time. Includes scenarios with varying operating parameters and defect intensities.
OR (Outer Race Defect)	Bearing with outer race defect(s). Includes scenarios with varying operating parameters and defect intensities.
IR (Inner Race Defect)	Bearing with inner race defect(s). Includes scenarios with varying operating parameters and defect intensities.

Table 2.12: Defects considered in this study and their descriptions.

The transfer of knowledge from the source domain to the target domain has demonstrated effectiveness across both variations in operating parameters (RPM and Radial Force) and defect type variations (Artificial to Real). However, a critical aspect to consider is the impact of the target task loss, as it has the potential to fine-tune or overfit the target encoder to the semi-labeled



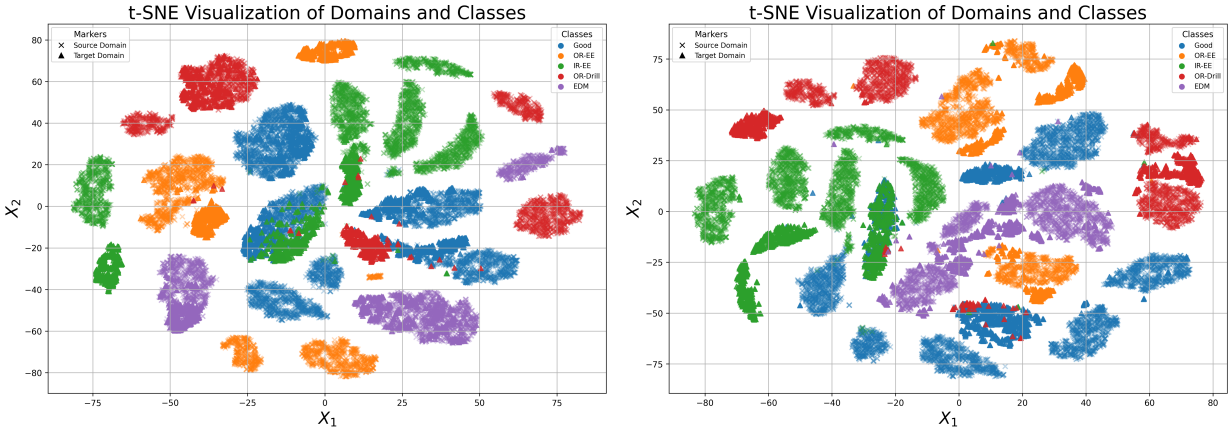
(a) Source classifier on target domain.

(b) Target classifier on target domain.

Figure 2.34: Domain adaptation enabling knowledge transfer from artificially generated defect to real defect on bearing.

instances of the target domain data. This phenomenon is evident in the confusion matrix shown in Figure 2.34b, where the performance for the classes "Good" and "OR" improved significantly, while the performance for "IR" dropped by approximately 7%. This performance disparity likely stems from the fact that the classes "Good" and "OR" were actively involved in the knowledge transfer process during the training of the target encoder through the target task loss, whereas "IR" was not as well-represented. To mitigate such issues, careful consideration is required when selecting the weights λ_1 and λ_2 during training. Proper tuning of these weights is essential to balance domain alignment and task discriminability, ensuring robust performance across all defect classes in the target domain.

In addition to the confusion matrices showcasing the improvements in performance in the target domain, it is equally beneficial to visualize the improvements made by our domain adaptation approach. To effectively visualize the performance of the source and target encoders, a t-SNE plot was created by applying dimension reduction on the source and target encoder outputs before passing them to the classification layer (softmax layer). In Figure 2.28, this would be the location where the MMD loss was also computed. The t-SNE plots created for the operating parameter variations in radial force and RPM can be seen in Figure 2.35 and Figure 2.36, respectively.

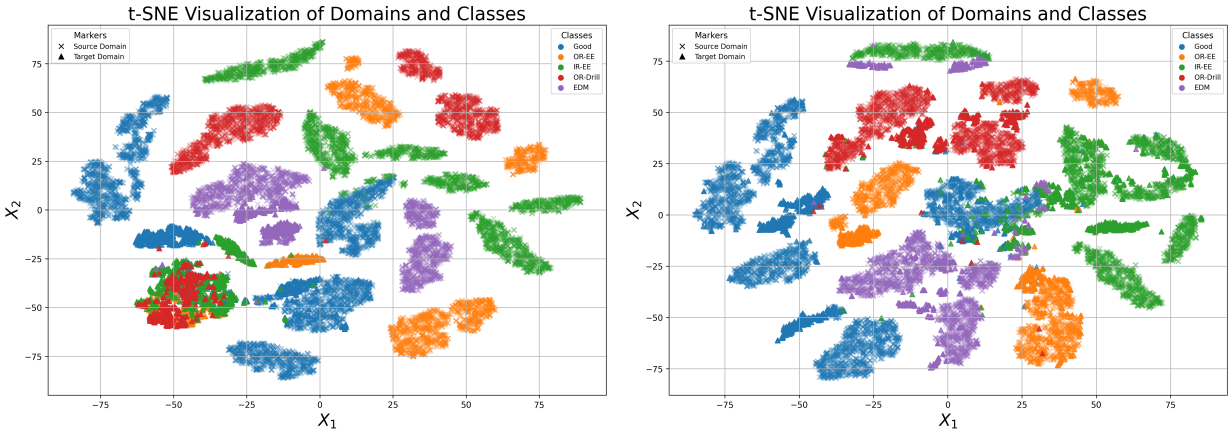


(a) Source encoder applied target domain.

(b) Target encoder applied on target domain.

Figure 2.35: t-SNE plot on radial force variation from 1000N to 400N.

In the t-SNE plots, the extracted source and target domain indicate the effective domain alignment by the extracted features being inseparable, i.e., the extracted source and target domain features are grouped. From Figure 2.35a and Figure 2.36a, for the case of both operating parameter variations, the source encoder alone was not able to achieve domain alignment, contrary to target encoder which was successful in achieving domain alignment, seen in Figure 2.35b and Figure 2.36b. As previously discussed, in addition to domain alignment it is equally important to have task discriminability. In the t-SNE plots, the task discriminability can be seen by the encoders grouping the data instances corresponding to the respective classes together. As can be seen in the figure, our approach was effective in enabling domain alignment and task discriminability.



(a) Source encoder applied target domain.

(b) Target encoder applied on target domain.

Figure 2.36: t-SNE plot on RPM variation from 1500rpm to 900rpm.

The incorporation of the MMD loss, seen in Equation 2.6, to our adversarial domain adaption was beneficial in two ways: (1) It improved domain alignment, seen from the t-SNE plot where the source and target domain features were grouped, (2) It reduced the labeling requirement in the target domain, required by the target task loss seen in Equation 2.5. Without the MMD loss, at least 3 defect classes were required to be labeled in the target domain to get a reasonable performance closer to the ones presented above. With the MMD loss, the target encoder outperformed the case without MMD loss with only 2 defect classes labeled. Currently, the hypothesis is that in condition monitoring use cases the data availability is limited, and adversarial domain adaptation is not effective in the cases of limited data. By adding the MMD loss we were able to overcome the issue of limited data availability. This hypothesis will be further validated in future studies to identify the effective and computationally less intensive way to enable domain alignment with task discriminability.

2.6.5 Discussion

The following inferences can be made from the studies conducted in this section:

1. Through the developed approach it was possible to enable domain alignment between the source domain and target domain, all the while maintaining task discriminability in the target domain.

2. Changes in operating parameters, such as fluctuations in motor RPM or variations in bearing loads, are common in real-world scenarios. Consequently, it is impractical to rely on a condition monitoring model developed for a fixed set of operating parameters and expect it to effectively monitor bearing conditions under different circumstances. Through our approach, we demonstrated how knowledge can be transferred between different operating parameters to enhance the performance of condition-monitoring models. This enables the development of models that perform robustly across both the source and target domains, ensuring adaptability to varying operational conditions. For instance, consider the case of Radial Force change from 1000N to 400N, the knowledge transferred target domain model performed with a macro-average F1-Score of 0.9212, seen in Table 2.11, the same model performed with a

macro-average F1-Score of 0.8508. This goes to show that our approach has not lost the source domain knowledge and carries it even when performing effectively on the target domain. **Note:** There was performance degradation in the source domain when using the target encoder, contrary to using the source encoder on the source domain data. In other words, there was a shift in feature representation.

Key considerations for employing this approach in developing condition monitoring systems, aimed at addressing challenges and guiding future research, include:

1. It is important to emphasize that enabling knowledge transfer from the source domain to the target domain necessitates the availability of data from the target domain. For instance, when transferring knowledge between different operating parameters in the above evaluation, it was crucial to have data from the target domain's operating parameters. Additionally, the target domain data must be semi-labeled, as the degree of labeling significantly influences the performance of the target domain model. A higher proportion of labeled data typically results in better model performance in the target domain.

2. The target task loss in the domain adaptation process, while beneficial for improving task-specific performance, carries the risk of causing the target domain model to overfit the limited labeled instances. To mitigate this, consistency loss is incorporated alongside the target task loss to regularize the model training and reduce overfitting. However, it is equally important to address any class imbalance in the target domain data by appropriately weighting the classes.

3. As the target task loss can lead to overfitting, other losses such as KL Divergence loss can be explored to improve task discriminability. The KL Divergence loss encourages the predicted distributions to match the pseudo-label distribution generated by the model.

4. One hypothesis for why ADDA struggled with achieving task discriminability in the target domain lies in the preprocessing methods used during the development of condition monitoring models. In all our studies, the data were processed to extract time-frequency domain power spectrums, effectively reducing noise and emphasizing the most relevant features. While this

preprocessing enhances the clarity of the input data, it may inadvertently limit the diversity of pathways available for the model training process to achieve both domain alignment and task discriminability. By narrowing the feature space, the model may lack the flexibility to simultaneously adapt to domain differences and preserve task-specific nuances. In the future, other feature extraction pathways that are more conducive to domain adaptation can be explored for condition monitoring studies.

5. **Simulated Defects:** We saw that it was possible to transfer knowledge from artificial data domain to real data domain. In future work, the knowledge transfer can be explored for cases where the defect data is not available in the target domain but can filled in using simulation. The question we would aim to answer here is, does the knowledge contained in the time-frequency features and the simulation sufficient enough to learn about the target domain in the real world?

2.7 Future Work

Our future work in this domain is in two folds: Firstly, at the time of this writing, the most widely adopted approach for transferring knowledge between machines of similar types and processes, across varying operating parameters, and from laboratory to industry, is through transfer learning [19]. While this approach has proven effective, as demonstrated in our work in Section 2.5, it lacks scalability. Therefore, in the future, we aim to address the challenges associated with deploying DL and ML models in industrial settings. Through preliminary work and industry collaborations, we will investigate the challenges introduced in the machine monitoring process that relies on sensor data. These challenges include equipment setup variations, equipment types, sensor calibration drifts, and the impact of time on monitoring performance.

OOD detection for deep learning models has shown effectiveness in identifying when the input data distribution diverges from the training distribution. The energy-based OOD detection approach is of particular interest for future research, as we aim to explore the question: can the low-energy regions of an arbitrary energy curve be linked to the underlying physical functions of the process being monitored? If this approach is successful, I believe we can potentially extrapolate

between the approximate physics-based model and the data-driven model, effectively.

Secondly, at the time of writing, a common concern among researchers was the limited availability of data in the manufacturing domain. Having a reliable and extensive data source is crucial for model development and evaluation in long-term reliability studies on condition monitoring models for manufacturing industries. Industries can also use this data to train foundation models for condition monitoring applications specific to a manufacturing process, which can then be fine-tuned for a particular machine performing the process.

To address the need for increased data availability in the manufacturing domain, we aim to develop an online platform where manufacturing industries and academic institutions can both upload and download data to support research and industry needs in smart manufacturing. The platform will be structured to categorize the data according to specific manufacturing processes, ensuring that the data collection aligns with the underlying physics of the process. This includes carefully selecting sensor placement and sensor types to capture the relevant physical phenomena during data collection effectively.

Through this work, we also aim to address the challenge of hardware non-availability for ease of setup, deployment, data collection, and inference by open-sourcing the condition monitoring hardware designed specifically for long-term studies and real-time monitoring. By making this hardware open-source, we expect to significantly increase the amount of data collected, as it lowers the barriers to entry for both industry and academia. Institutions and companies will be able to quickly deploy these open-source systems, leading to broader adoption and more widespread data collection efforts. This will allow for a greater variety of datasets across different machines, processes, and environments, ultimately fostering the development of more robust and generalized condition-monitoring models.

Our goal is to foster open-source hardware development for deploying condition monitoring and data collection systems for smart manufacturing applications. We believe that this approach will empower academic institutions and industries to contribute data, resulting in a richer data ecosystem that supports further innovation in the field.

Chapter 3

Monitoring Equipment Energy Consumption to Detect Anomalies

Retrofitting legacy machines has its challenges. It is not always possible to attach sensors close to the target due to constraints in the manufacturing process. In the cold forging scenario presented in Chapter 2, the sensors were carefully chosen to withstand the high temperature, force, and pressure involved in a typical cold forging operation. Hence, in this chapter, an alternative approach to detect anomalies in equipment energy consumption patterns was studied. We start by identifying the equipment states from energy followed by identifying anomalies in its operation. This study was conducted on an ultra-precision Computer Numerical Control (CNC) machine, FANUC ROBONANO α-0iB. The reason behind the choice of machine was accessibility and nuanced control over the DAQ process for this study. Additionally, ultra-precision CNC machining operations take long operating hours, and any anomalies can go undetected.

3.1 Literature Review

Ultra-precision manufacturing (UPM) is a manufacturing technology for machining micro components with a micrometer or nanometer precision [51]. As an indispensable part of UPM, Ultra-precision CNC machine tools undertake the fabrication task for the designed parts. Because of the precision requirement, the machines are quite sensitive to any subtle change in the environment. These changes potentially impact the quality of the part manufactured. Additionally, inappropriate process parameters such as exorbitant feedrate, insufficient coolant, or component failures in the machine tool will lead to inefficient cutting and potential safety hazards [52]. Hence, continuous monitoring of the working state of the ultra-precision machine tool is imperative.

Energy consumption monitoring of the Computer Numerical Control (CNC) machine tools has been a topic of interest and several studies were conducted to understand the patterns embedded in the energy/power consumption data to either develop energy-saving strategies or to understand the manufacturing process in granular details [53, 54]. Real-time monitoring of energy consumption

has been of importance to the research community. Vijayaraghavan and Dornfeld [55] introduced an event stream processing technique to automate the energy consumption monitoring of the vast number of manufacturing machines. The work also describes the importance of energy consumption monitoring, particularly focusing on reducing energy usage, reducing disruptions in operations, ability to track the maintenance state of the machines, and finally the ability to detect the impact on the environment. In a typical manufacturing facility, a major proportion of the machines used cannot be connected to the cloud, nor do they have the means to collect data about their operation pre-built. Several studies explored the possibility of resolving this issue [56, 57].

To extract information from the sensor data and to provide meaningful inference with limited domain knowledge, machine learning, and deep learning have become important in the era of Industry 4.0. The reasoning behind why this research work was conducted is as follows: First, as we are currently at the edge of this transition, it is highly unlikely that the sensors required are already present in the OEMs. Second, retrofitting of the manufacturing machines can quickly become expensive and invasive, considering the plethora of sensors that needs to be installed on the machine to make reliable inferences. Third, majority of the work presented above showcase a model development process that works very well in a controlled scenario. The deployment scenario with various challenges associated with the input distribution drift, sensor perturbations, etc., was not discussed. Fourth, no baseline was previously developed that discusses the complete model development cycle. In addition, none of the works presented above and to the extent of our knowledge have open-sourced the code in a plug-and-play fashion to replicate or improve our work. Fifth, our work was conducted on ultraprecision machines with very low material removal rate. Hence, visually identifying and trimming the patterns from the power consumption data is challenging. Sixth, it is not straightforward to detect and classify the anomalies in real-time from the energy data and publish the interface online for the public to interact. Finally, we aim to develop a methodology for energy monitoring that can be translated to industries for energy prediction and optimization for sustainable manufacturing.

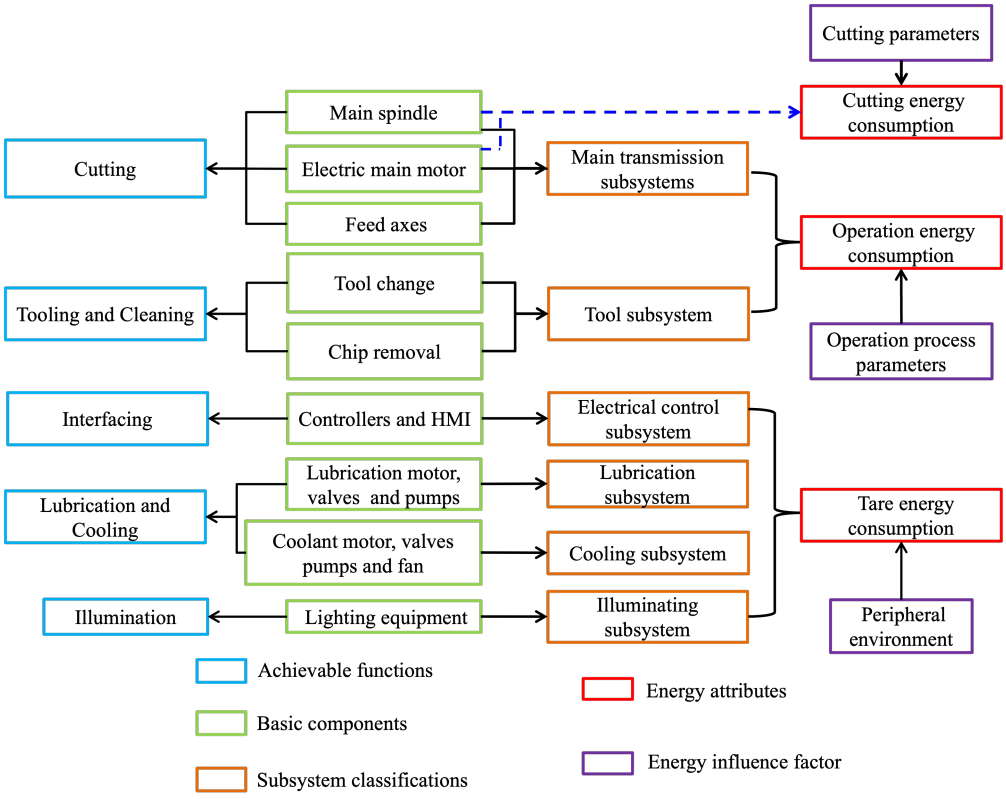


Figure 3.1: Categorization of energy consumed by a UPMT [58].

3.2 Equipment State Identification

The first step toward detecting anomalies in the equipment's operation was to identify and extract the associated patterns in its energy consumption data. Hence, in this section, a study was conducted to determine if the state of the machine can be identified from its energy consumption data. A Multi-output Densely Connected Convolutional Network (MDCCN) was developed as a part of this study. The model identifies the axis in motion and predicts the feedrate simultaneously by mining the underlying pattern from the power consumption data. The details of this work can be found in our publications Selvaraj, Xu, and Min [58] and Xu, Selvaraj, and Min [59, 60].

3.2.1 Energy Characteristic of machine tool

In an Ultra-Precision Machine Tool (UPMT), the primary energy consumption element, known as the basic energy conversion unit, includes the motors that control the axes and main spindle,

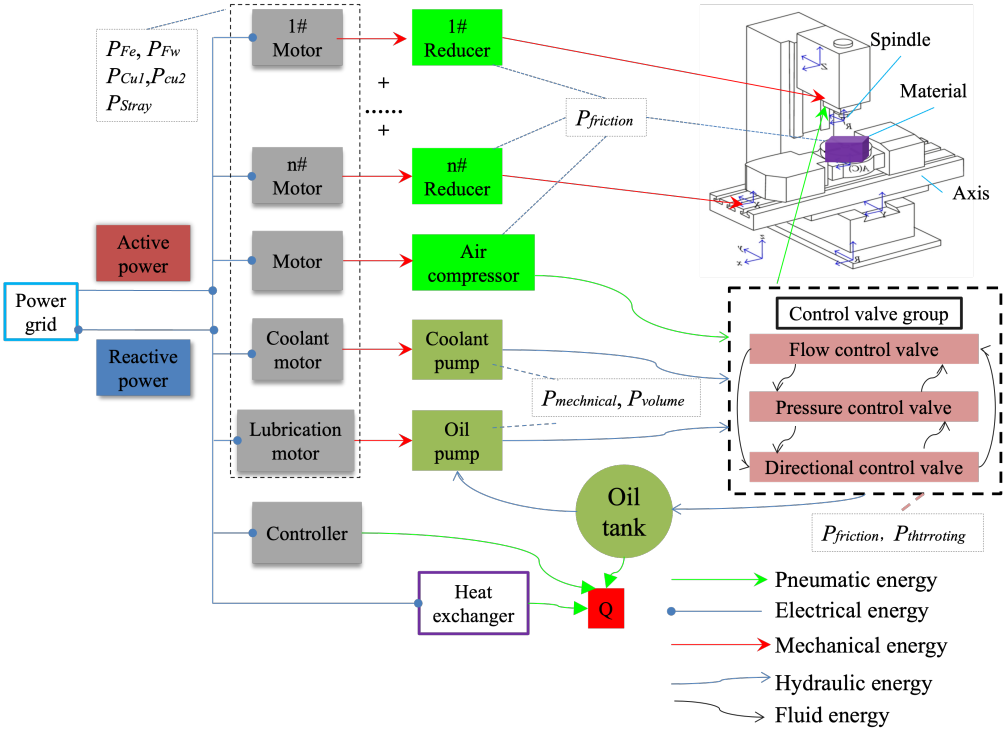


Figure 3.2: Energy flow within a typical CNC machine tool [58].

the controller, and accessories such as the cutting fluid pump and mist collector. Most CNC machines are equipped with certain essential functions that are integral to their operation: (1) a controller that interprets G-code to manage cutting speed, travel, and feed rate; (2) automatic tool changers (ATCs); and (3) cutting fluid systems for lubrication and cooling. These components collectively form the major energy-consuming units of a CNC machine. The power consumed by the machine tool can be broadly categorized into three segments. First, a portion of power is consumed by accessories and maintaining machine operations. Second, power is drawn by the motors controlling the axes and spindle, primarily influenced by operational process parameters rather than material removal, meaning this power largely depends on machine design. Finally, cutting energy consumption is driven by cutting parameters, the type of processing, and material properties. The distribution of energy consumption across these three segments in a typical CNC machine is illustrated in Figure 3.1.

The different components within a CNC machine, in terms of their power consumption, can be broadly classified as shown in Figure 3.2. In the case of Ultra-Precision Machining (UPM), the

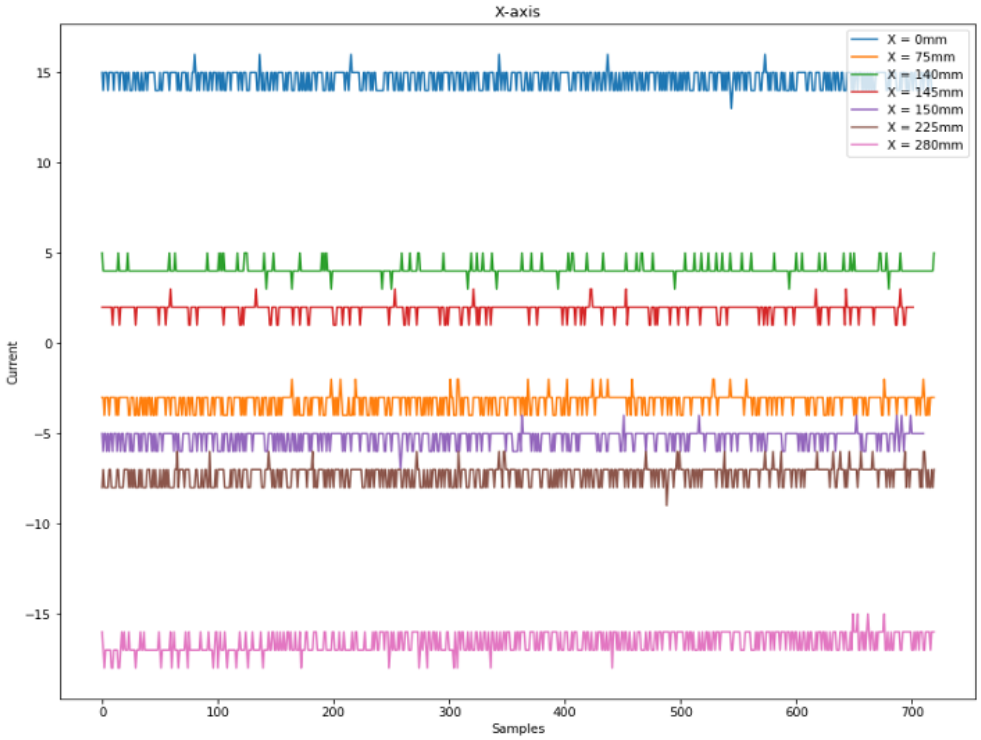
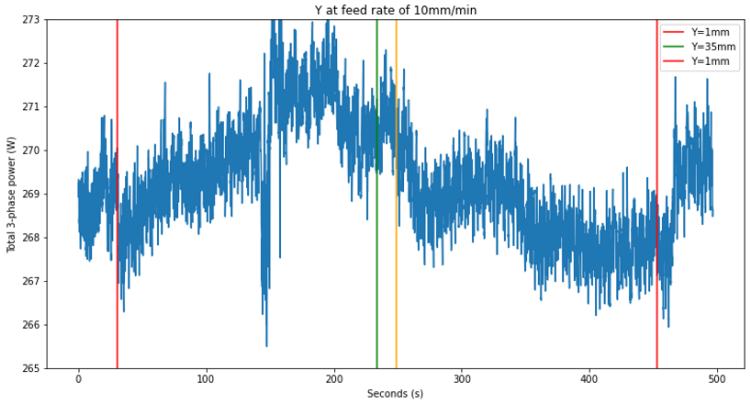


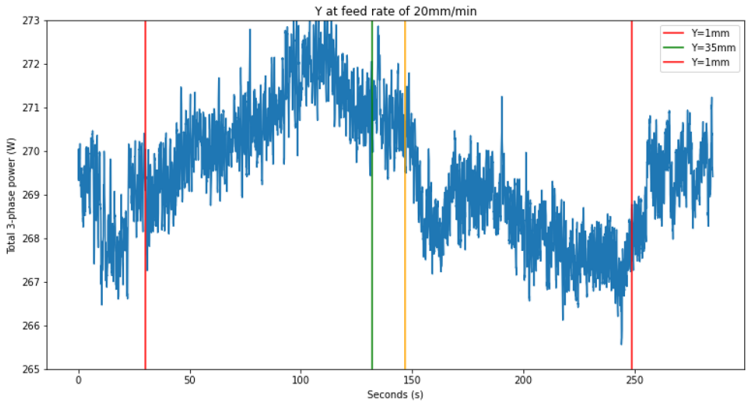
Figure 3.3: X-axis servo current consumption indicating the position sensitivity.

power consumption predominantly depends on the position of each axis with respect to a datum, something we validated experimentally by interfacing with the FANUC controller using FOCAS2 and directly measuring the energy consumed by the servos for different axis positions, can be seen in Figure 3.3. In Figure 3.3, the current consumption was significantly impacted by the axis location, which in turn depends on the machine design. In FANUC ROBONANO $\alpha 0i$B, the B-axis sits directly on top of the X-axis and the B-axis load will impact the effective energy consumption of the machine tool. The same can be shown for the Y-axis seen in Figure 3.4, where the power consumption for different positions across two different feedrates was shown. It can be seen that irrespective of the feedrate the biggest impact on the power consumption was from the axis' position relative to the datum.

Both active and reactive power are considered in our analysis. Active power, also called real power, is the net power transferred in one direction. The power consumed while cutting the material is a part of the active power. The energy input to the machine can take different forms such as pneumatic, electrical, mechanical, hydraulic, etc. Each component that does the conversion can



(a) Y-axis travel 10mm/min feedrate.



(b) Y-axis travel at 20mm/min feedrate.

Figure 3.4: Impact of Y-axis position on the power consumption for different feedrates.

be thought as a resistance load in series with the energy input. By determining the instantaneous energy input and the instantaneous energy output, the energy consumed by the equipment can be computed.

In UPM, the cutting energy—i.e., the energy consumed for material removal—represents only a small fraction of the machine's overall energy consumption, much lower than a typical CNC machine. Therefore, to effectively model the energy usage of an UPMT, it is essential to continuously monitor the machine's operational state by accurately identifying which axes are in operation and their respective states at all times.

3.2.2 Methodology

The framework of the approach can be seen in Figure 3.5. It consists of an offline model training phase and an online model inference phase. In the offline model training phase, the power consumption data were collected by conducting controlled experiments using a high-precision power analyzer. Parallelly, we developed an application called the G-code interpreter that can parse the G-code information and construct a working status matrix that can correlate the sensor data with the operating status of the machine. The working status matrix was then used to train the MDCCN model. In summary, the G-code interpreter automates the data labeling process.

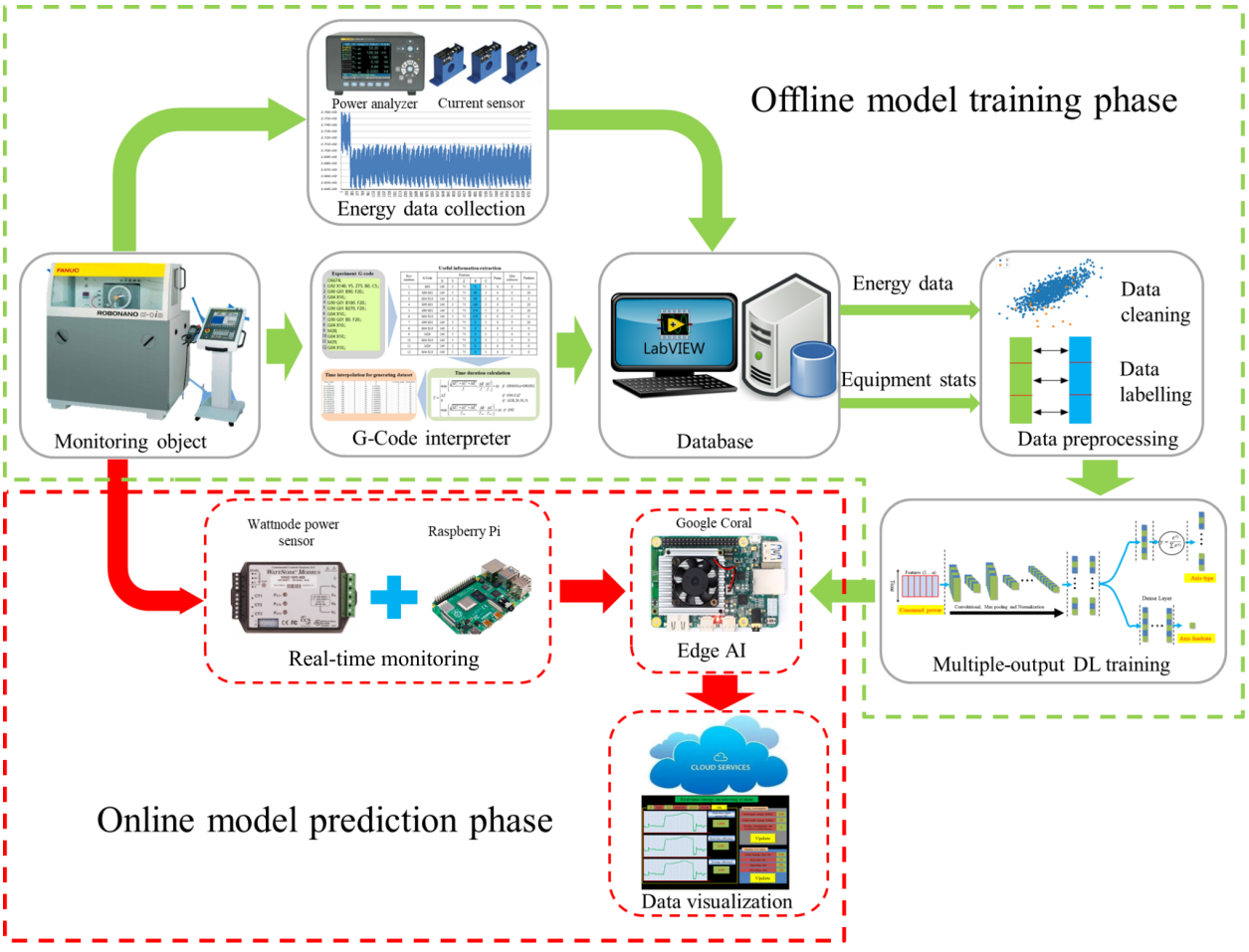


Figure 3.5: Framework of the equipment state identification system [59].

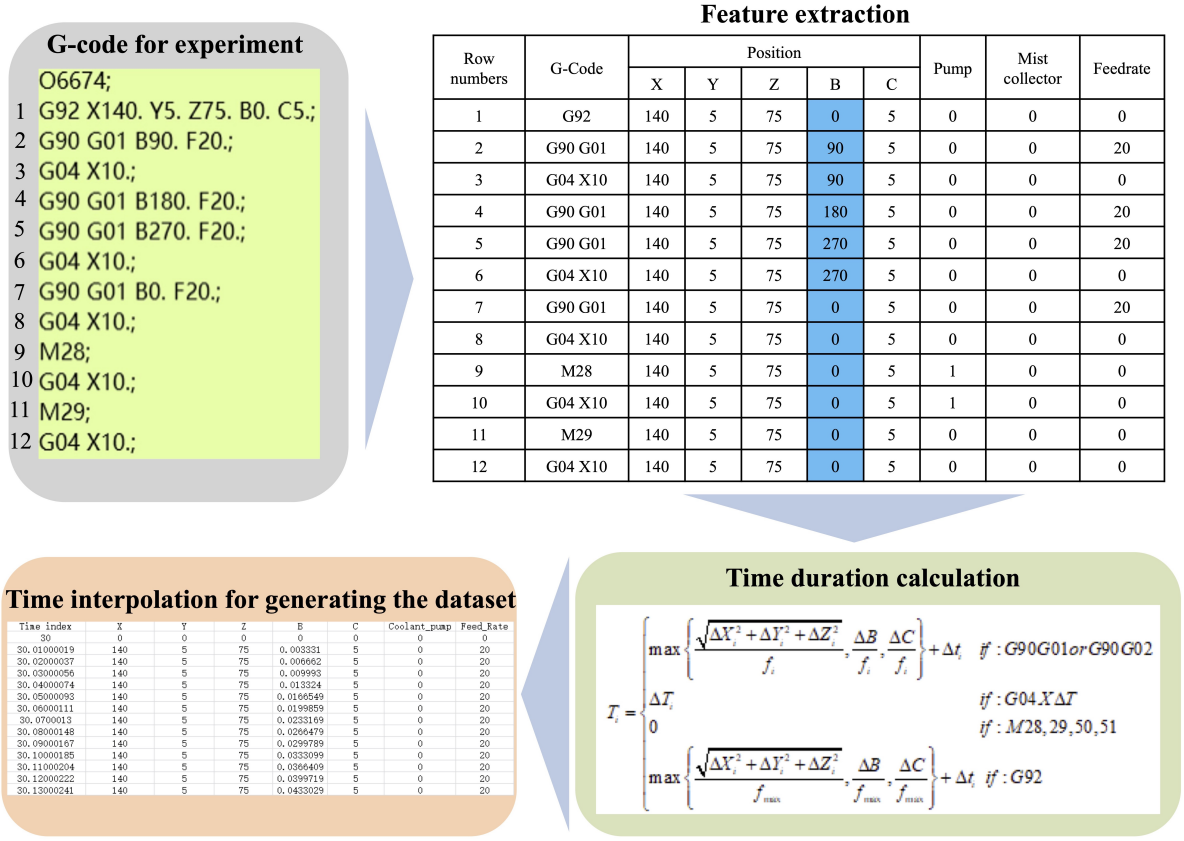
$$f(x) = \begin{cases} \max\left\{ \dfrac{\sqrt{\Delta X_i^2 + \Delta Y_i^2 + \Delta Z_i^2}}{f_i}, \dfrac{\Delta B}{f_i}, \dfrac{\Delta C}{f_i} \right\} + \Delta t_i & \text{if: } \{\text{G90 G01, G90 G02}\} \\ \Delta T_i & \text{if: } \{\text{G04 X}\Delta T\} \\ 0 & \text{if: } \{\text{M28, M29, M50, M51}\} \\ \max\left\{ \dfrac{\sqrt{\Delta X_i^2 + \Delta Y_i^2 + \Delta Z_i^2}}{f_{\max}}, \dfrac{\Delta B}{f_{\max}}, \dfrac{\Delta C}{f_{\max}} \right\} + \Delta t_i & \text{if: } \{\text{G92}\} \end{cases} \tag{3.1}$$

where ΔX_i, ΔY_i, ΔZ_i, ΔB_i, and ΔC_i are the increments in displacement for each axis, f_i is the feedrate, Δt_i is the compensation factor determined experimentally for axis acceleration and deceleration process, ΔT_i is the delay corresponding to the G-code G04, and f_{max} is the maximum feedrate.

G-code Interpreter

The purpose of this application was to automate the training process for the MDCCN model. The G-code interpreter application helps in the generation of the working status matrix. The application facilitates the following items: 1) Interpolate the axis position at all times while the G-code is running on the machine, 2) Determine the state of all auxiliary components on the machine, 3) Associate the power consumption data from the sensors with the axis in motion and its feedrate, 4) Generate the training data with the required pre-processing to be used for the MDCCN model.

The application takes three inputs, the configuration file describing the machine running the G-code, the G-code that was run on the machine, and finally the sensor data. The lines of the G-code were read sequentially, and the information on the processing parameters was extracted. Buffers were used to keep track of the operation status of different components in a machine, such as the five axes, coolant pump, mist collector, etc. The axis position paired up with its operation status information on the buffer was tracked over time as the G-code was executed. The execution time T_i was calculated between i and $i+1$ rows using Equation 3.1.



Row numbers	G-Code	Position					Pump	Mist collector	Feedrate
		X	Y	Z	B	C			
1	G92	140	5	75	0	5	0	0	0
2	G90 G01	140	5	75	90	5	0	0	20
3	G04 X10	140	5	75	90	5	0	0	0
4	G90 G01	140	5	75	180	5	0	0	20
5	G90 G01	140	5	75	270	5	0	0	20
6	G04 X10	140	5	75	270	5	0	0	0
7	G90 G01	140	5	75	0	5	0	0	20
8	G04 X10	140	5	75	0	5	0	0	0
9	M28	140	5	75	0	5	1	0	0
10	G04 X10	140	5	75	0	5	1	0	0
11	M29	140	5	75	0	5	0	0	0
12	G04 X10	140	5	75	0	5	0	0	0

Time index	X	Y	Z	B	C	Coolant_pump	Feed_Rate
30	0	0	0	0	0	0	0
30.01000019	140	5	75	0.003331	5	0	20
30.02000037	140	5	75	0.006662	5	0	20
30.03000056	140	5	75	0.009993	5	0	20
30.04000074	140	5	75	0.013324	5	0	20
30.05000093	140	5	75	0.0166549	5	0	20
30.06000111	140	5	75	0.0199859	5	0	20
30.0700013	140	5	75	0.0233169	5	0	20
30.08000148	140	5	75	0.0266479	5	0	20
30.09000167	140	5	75	0.0299789	5	0	20
30.10000185	140	5	75	0.0333099	5	0	20
30.11000204	140	5	75	0.0366409	5	0	20
30.12000222	140	5	75	0.0399719	5	0	20
30.13000241	140	5	75	0.0433029	5	0	20

$$T_i = \begin{cases} \max\left\{\dfrac{\sqrt{\Delta X_i^2 + \Delta Y_i^2 + \Delta Z_i^2}}{f_i}, \dfrac{\Delta B}{f_i}, \dfrac{\Delta C}{f_i}\right\} + \Delta t_i & if: G90G01 or G90G02 \\ \Delta T_i & if: G04\,X\Delta T \\ 0 & if: M28, 29, 50, 51 \\ \max\left\{\dfrac{\sqrt{\Delta X_i^2 + \Delta Y_i^2 + \Delta Z_i^2}}{f_{max}}, \dfrac{\Delta B}{f_{max}}, \dfrac{\Delta C}{f_{max}}\right\} + \Delta t_i & if: G92 \end{cases}$$

Figure 3.6: The working principle of G-code interpreter [59].

To summarize, the G-code interpreter application enables the generation of the working status matrix that automates the labeling process for the sensor data, which is a challenge for manufacturers to realize reliable models. The working principle of the G-code interpreter can be seen in Figure 3.6.

Table 3.1: 1D-CNN hyperparameters [59].

Model structure	Hyperparameters	Hyperparameter space
1D-CNN layers	Layers, Kernel size, Filter size, Pooling size	[1:6], {3,5}, {32,64,128}, {3,5}
Dense layers	Neurons, layer counts	[16:512], [1:6]
Dropout layer	Dropout ratio	{0.2, 0.5, 0.7}
Model learning rate	Learning rate	{1e-2, 1e-3, 1e-4}

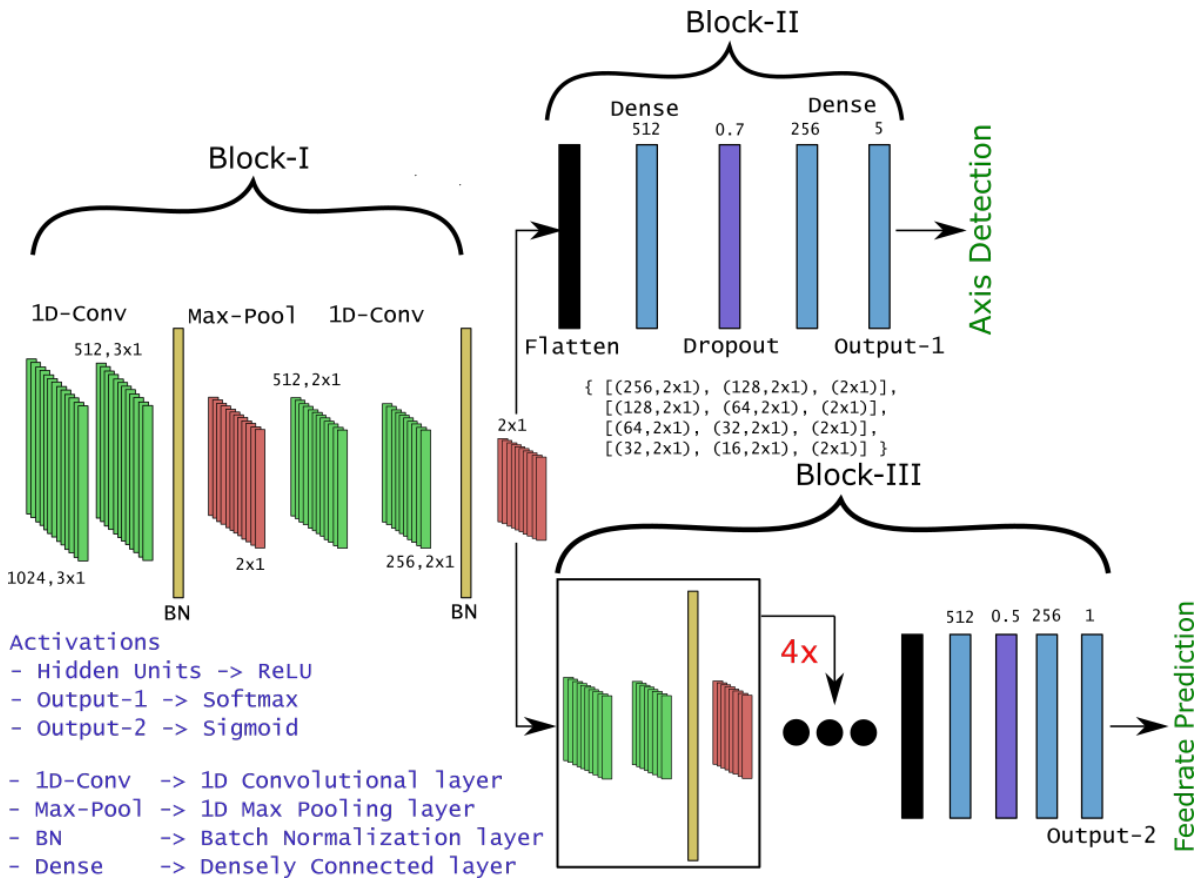


Figure 3.7: The architecture of the MDCCN [59].

MDCNN Architecture

As the name states, the multi-output densely connected convolution network has two outputs, one for axis detection and the other for feed rate prediction. The model architecture was constructed with the notion that the axis and feedrate will have similar/related energy consumption patterns. The model architecture consists of three blocks. The first block forms the base for the remaining

Table 3.2: Experiment plan [59].

Item	X-axis mm/min	Y-axis mm/min	Z-axis mm/min	B-axis degree/min	C-axis degree/min
Min feedrate	0	0	0	0	0
Max feedrate	1000	50	500	1800	1800
Increment	20	5	20	20	20
Total times	50	10	25	90	90

two blocks and consists of two composite functions. A composite function in our case corresponds to two 1D-Convolution blocks followed by a Batch Normalization (BN) layer and a Max Pooling layer. The choice of layers in the composite function is due to the following reasons, 1D Convolution layers were due to the type of data input to the model. The Batch Normalization layers were used to prevent gradient vanishing problems associated with the deeper networks, [61]. The gradient vanishing problem refers to a situation that can arise during the training of deep neural networks, where the gradients of the loss function with respect to the weights in the earlier layers become extremely small. This can prevent the weights from being updated during the training process leading to slowing down of the training process. The 1D Max Pooling layers carry out sub-sampling of the inputs by picking the most active neuron. The key advantage of the Max Pooling layer is also to achieve translational invariance. The second block corresponds to the output that can detect the axis controlled using softmax activation in the last layer. The third and final block of the model includes four composite functions with a sigmoid activation function. The architecture is shown in Figure 3.7.

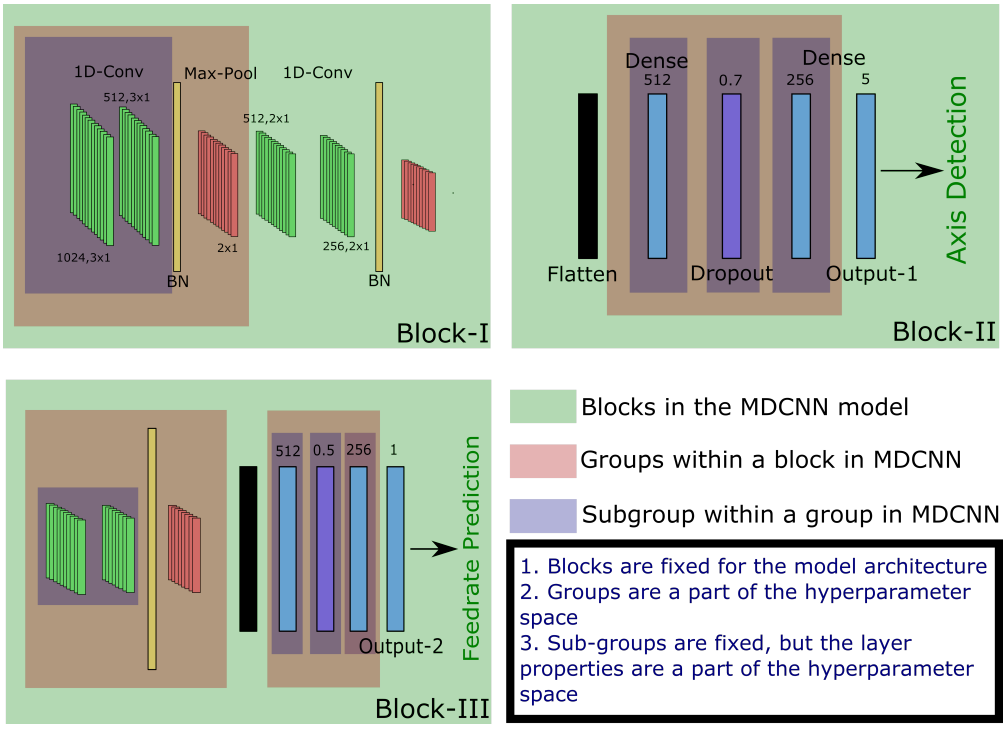


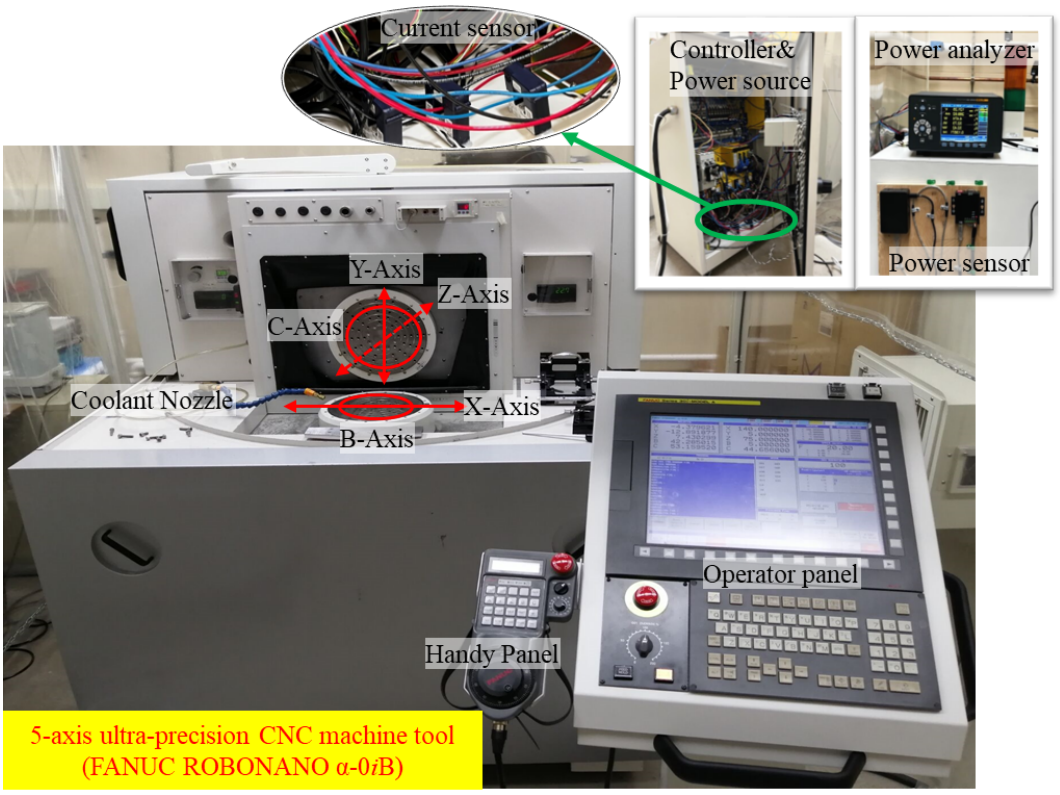
Figure 3.8: Design for hyperparameter optimization [59].

Optimizing the hyperparameters was crucial to evaluate the capability of the developed model

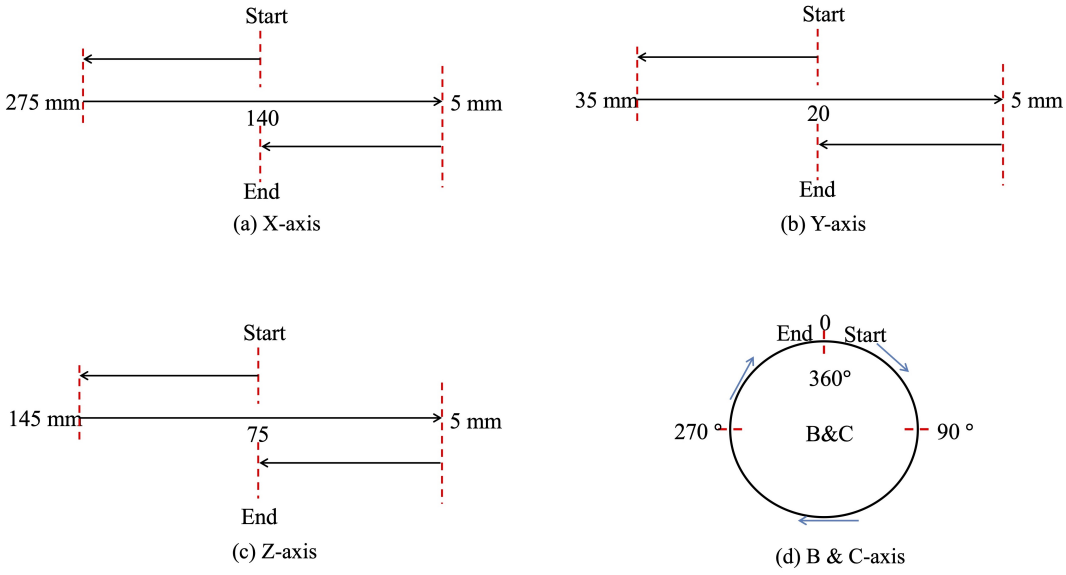
better. Hence the hyperparameters of the model were optimized using a Bayesian Optimization technique implemented using keras-tuner [62]. The list of hyperparameters that were optimized is given in Table 3.1. The parameters of the model after the hyperparameter optimization are shown in Figure 3.7. Since the hyperparameter space for the architecture in consideration is quite large, the hyperparameter tuning process was constrained on what it can explore to identify the best-performing architecture. The constraints placed can be seen in Figure 3.8, and it involves the following: 1. The blocks corresponding to different segments of the model remain fixed and they can neither be increased nor decreased. 2) The groups within the block, corresponding to the composite function, can be increased as whole to develop deeper architectures. 3) The subgroups within a group cannot be increased, but the parameters of each layer can be modified as required. The fundamental idea behind the design was that the deeper models improve generalization [37, 63]. Hence, the hyperparameters were designed such that the depth of the model was increased rather than just the number of learning parameters i.e., the width of the models.

3.2.3 Experiments

The FANUC ROBONANO α-0iB consists of 5 axes: X, Y, Z, B, and C. The voltage and current input to the equipment was at 220V, 30A, and is supplied to 5 servos, controlling each axis, and two accessories, a cutting fluid pump, and a mist collector. The CNC equipment along with the experimental setup can be seen in Figure 3.9a. The first step in energy consumption monitoring was to establish the equipment baseline. This serves two purposes, firstly, it was to determine the fundamental energy consumption unit of the machine, and secondly, it will serve as a benchmark to detect anomalies and equipment deterioration over time. Experiments were then conducted by operating each of the axes of the machine over its range of motion, seen in Figure 3.9b and Table 3.2. The choice of feedrates for the axis operation was determined by dividing the range of feedrates offered by the machine for a particular axis into multiple equally spaced segments. This study aims to determine if the equipment's power consumption data could be used to determine the state of the machine, hence the multi-axis motion will be considered as future work. The power consumption was measured at the input to the machine using a Fluke Norma 4000 power analyzer. The power analyzer was set in a 3P3W (Three Phase Three Wire) configuration to enable individual



(a) Experiment setup FANUC ROBONANO α-0iB.



(b) Experiment plan.

Figure 3.9: Experiments conducted on FANUC ROBONANO α-0iB [59].

measurements at each input phase of the machine.

The data were collected on three types of power consumed by the machine, Active Power, Reactive Power, and Apparent Power summed across all 3-phases of the machine. Each of these power attributes has an impact on different components of the system and will enable efficient extraction of the features of interest. Since the power analyzer was connected at the input of the machine without any filter, noise from the AC supply was usually present in the data. Filtering was applied at the power analyzer's end to ensure that the noise was removed before processing the data. The three power components were used as the features of the model.

$$J_{ce}(w; X, y) = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)] \tag{3.2}$$

$$J_{mse}(w; X, y) = L(y, \hat{y}) = \frac{1}{N}\sum_{i=0}^{N}(y - \hat{y}_i)^2$$

$$\tilde{J}(w_1, w_2; X_1, X_2, y_1, y_2) = \beta_1 \tilde{J}_{ce}(w_1; X_1, y_1) + \beta_2 \tilde{J}_{mse}(w_2; X_1, y_1)$$

The MDCCN model has two outputs, hence, the optimization objective of the model involves a weighted sum of cross-entropy (CE) loss and mean squared error (MSE) loss as can be seen in Eqn 3.2. Within each loss, the X represents the input data features, y represents the true values or target values, $\hat{y}$ represents the predict value, and w represents the model parameters. The weightage, β_1 and β_2, associated with each loss term controls how the model prioritizes the inference. In both cases, L1 and L2 regularizations were applied to prevent overfitting, which was likely in our case. The L1 regularization adds a penalty equal to the absolute value of the magnitude of the coefficients to the loss function. The L2 regularization adds a penalty equal to the square of the magnitude of the coefficients to the loss function. L1 regularization encourages sparsity in the model whereas the L2 regularization discourages large coefficients. The impact of L1 and L2 regularization on the loss function can be seen in Equation 3.3.

Table 3.3: Cross Validation metrics for state identification [59].

Metric	Classification loss	Prediction loss	Classification accuracy
Mean	0.0602	170.1260	98.76%
SD σ	0.0081	14.9659	0.0027%
Max	0.0826	190.3118	98.95%

Table 3.4: Classification performance for axis identification [59].

Axis type	**Precision**	**Recall**	**F1-Score**
X-axis	0.97	0.99	0.98
Y-axis	0.97	0.85	0.90
Z-axis	0.92	0.95	0.94
B-axis	0.91	0.96	0.93
C-axis	0.96	0.90	0.93

$$\texttt{Loss} = \texttt{Original Loss} + \lambda \sum_i |\theta_i| \quad \texttt{(L1 Regularization)} \tag{3.3}$$

$$\texttt{Loss} = \texttt{Original Loss} + \lambda \sum_i \theta_i^2 \quad \texttt{(L2 Regularization)}$$

3.2.4 Evaluation

The primary objective of this study was to determine if energy consumption data has sufficient information that can be used to identify the state of the machine at all times. To ensure that the inference process was reliable, a 10-fold Cross Validation (CV) was conducted. The results were summarized across each fold and can be seen in Table 3.3.

Axis Detection

In this section, the MDCCN model's ability to identify the axis in operation is discussed. The confusion matrix for the axis detection output of the model can be seen in Figure 3.10. The major portion of the classification error for the Y-axis and C-axis falls under the Z-axis and B-axis,

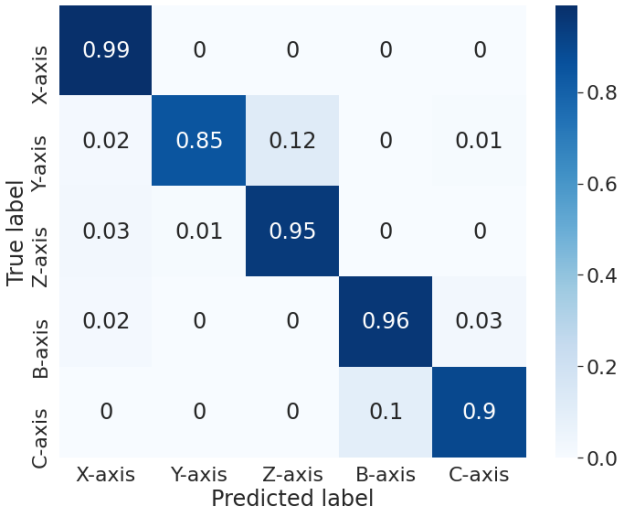


Figure 3.10: Confusion matrix, axis detection [59].

respectively. The reason behind this could be explained by the machine design. In the case of the Y & Z-axis, the entirety of the Y-axis rests on the Z-axis, hence the power consumption patterns are correlated between the two axes. Similarly, in the case of the B & C-axis, they are rotary axes of the machine, having similar power consumption patterns. The F1-Scores, which can be seen in Equation 3.4, were computed for each axis using the "one-vs-all" approach and are shown in Table 3.4.

$$\begin{aligned} \text{Precision} &= \frac{TP}{TP + FP} \\ \text{Recall} &= \frac{TP}{TP + FN} \\ \text{F1-Score} &= 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \end{aligned} \tag{3.4}$$

TP - True Positive, FP - False Positive, FN - False Negative.

Feedrate Prediction

In this section, the ability of the model to predict the feedrate is evaluated. In the current study, the compound movement of the five axes was not considered and is left as future work. The metrics

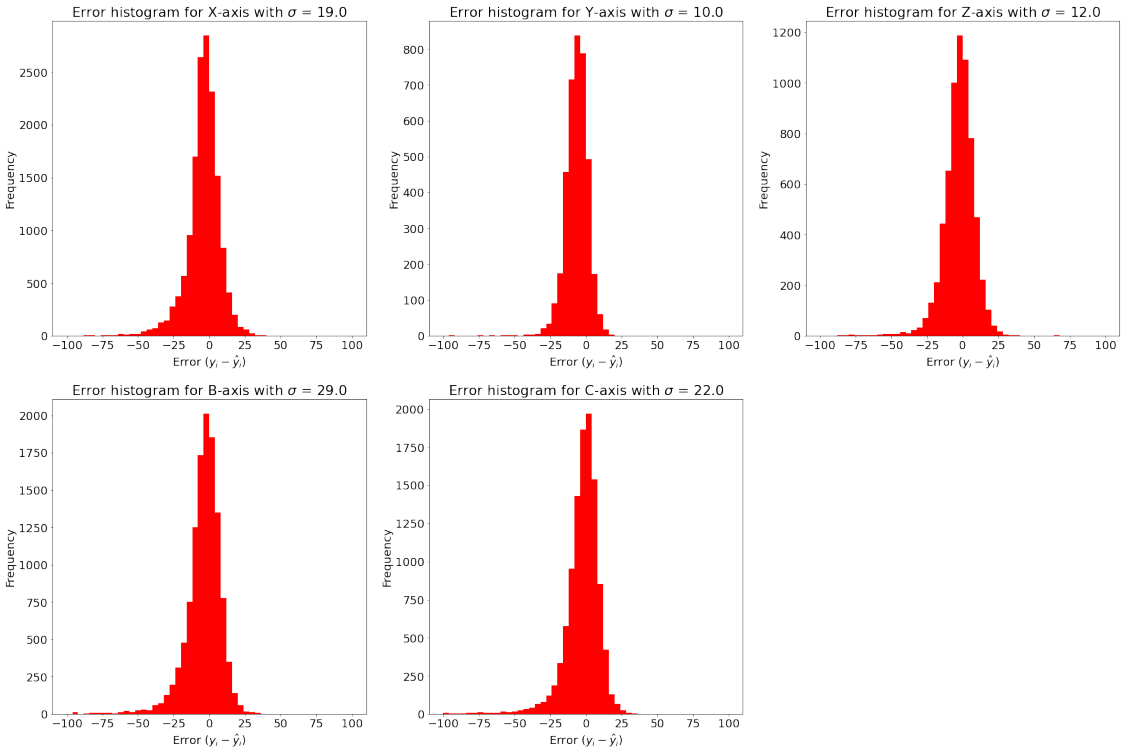


Figure 3.11: Error histogram for the feedrate prediction error by axis [59].

used to assess the model's performance were Root Mean Squared Error (RMSE) and R^2 and were determined to be 12.53 and 0.9960, respectively. R^2 values represent the number of variations captured by the model from the provided input data, the higher the value better is the model's ability to generalize against unseen data. The error histogram showing the distribution of the error $(y_i - \hat{y}_i)$ can be seen in Figure 3.11. From the histogram plot, the error is centered around 0.0 with a standard deviation of 21.900. Hence, it can be said that 99.7% of the error in feedrate lies within $\pm 3\sigma$, which is ± 65.7.

Feature Importance Study

During an implementation scenario, reducing the amount of data required to make a prediction will be beneficial, as the latency involved with the data transfer might be higher than computational latency. To reduce the number of features used for the model, a feature importance study was conducted. The study uses a method called Permutation Importance. The key idea behind the method was to randomly permute the data corresponding to a feature of the data set, one after

the other. Every time a feature was permuted, the resulting dataset was then used to make a prediction, and the loss metrics were cataloged. This process was repeated until all the features in the dataset were randomly permuted one after the other. The permuted feature that causes the highest prediction error was the most important feature in the dataset and for the model. In our case, these features correspond to the three different power components in the 3-phase input to the machine. After the process of random permutation and prediction, it was found that the most important feature was the Active power component, with the Reactive power component being the least important. The model loss after each permutation can be seen in Figure 3.12. From this study, it was clear that in the scenarios where the data dimension needs to be reduced to improve computational performance, the reactive power component can be dropped first followed by the apparent power component.

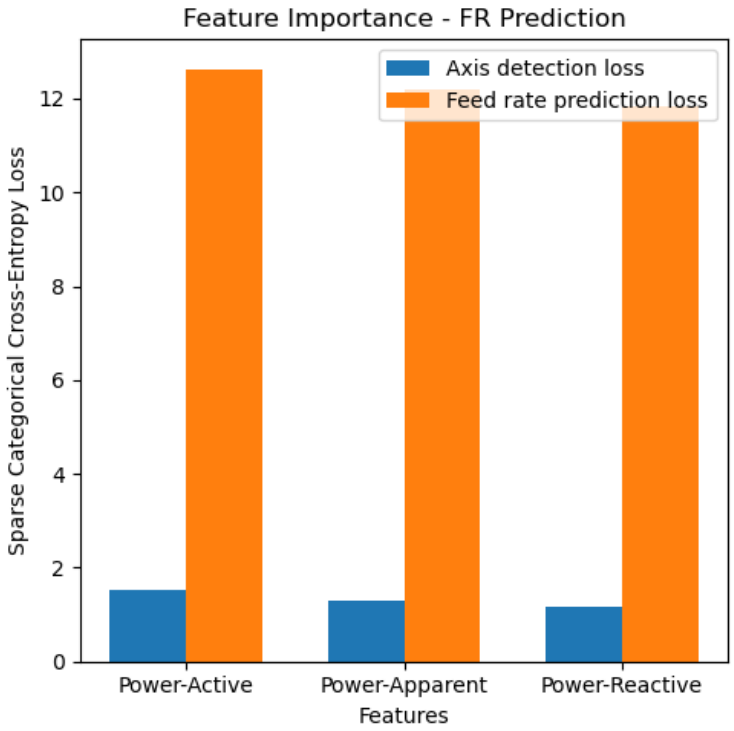


Figure 3.12: Feature importance study [59].

Discussion

The work presented so far mainly focuses on identifying the working state of the machine by detecting the axis in motion, and the associated feedrate from the power consumption data. This was a precursor to being able to detect anomalies from equipment's power consumption, see Section 3.3. Even though the servos across the 5 axes of the CNC machine were similar, the difference in the configuration of the axes led to unique energy consumption patterns. In this study, we were able to isolate and learn the pattern to identify the state of the machine. The code database used for this work can be found at https://github.com/vigneshuw/energy_state_identification.git.

3.3 Operation Anomalies from Power Consumption

As discussed in Section 2.5.1, retrofitting legacy machines has limitations. Hence, in this section, a study was conducted to understand the ability to detect anomalies in equipment operation from its power consumption data. This study was for the following reasons: First, as we are currently at the edge of this transition, it is unlikely that the sensors required are already present in the OEMs. Second, retrofitting the manufacturing machines can quickly become expensive and invasive, considering the plethora of sensors that need to be installed on the machine to make a reliable inference. Third, the majority of the work present in the current literature showcases a model development process that works very well in a controlled scenario. Fourth, no baseline exists that discusses the complete model development cycle. Finally, our work [64] was conducted on ultraprecision machines where the material removal rate is very low.

3.3.1 Methodology

The framework for the anomaly detection and real-time monitoring system can be seen in Figure 3.13. The framework is provided to guide other practitioners who are interested in using energy as one of the means to detect anomalies in the equipment's operation. It consists of three major segments: model training, model deployment, and model monitoring. An ultra-precision CNC machine was retrofitted with power meters for this study. Additionally, in this section, we discuss

the reasoning behind using the input energy consumption as an indicator of anomaly detection in manufacturing machines.

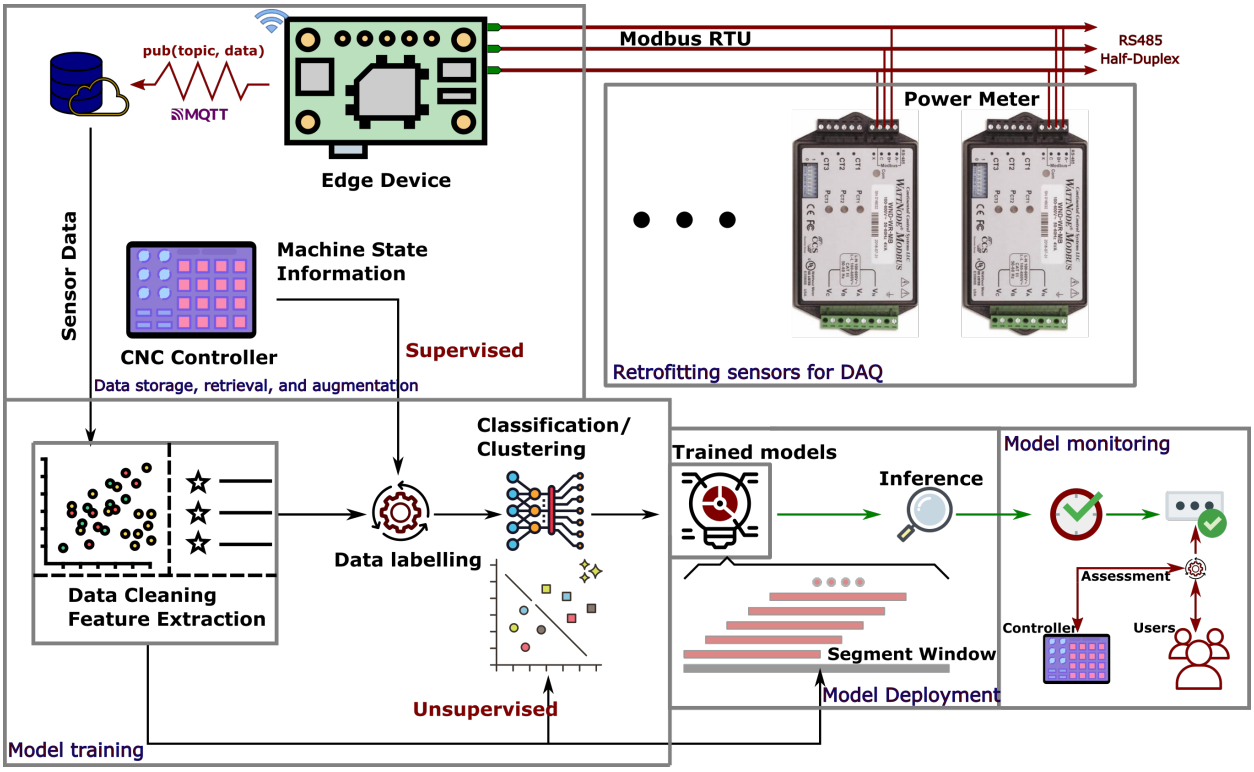


Figure 3.13: Framework for fault detection system based on equipment's power consumption [64].

Retrofitting Legacy Machines

The first step towards implementing a fault detection system on a legacy machine is to retrofit it with sensors that are conducive to identifying the operation patterns of the machine. In our case, the sensors used were power and energy meters – WattNode Wide-Range Modbus from Continental Control Systems, LLC. These sensors were compact and inexpensive and can potentially enable widespread deployment of the system in a manufacturing facility. The retrofitted sensors communicate over the RS-485 half-duplex using the Modbus RTU protocol. The sensor data was processed by an edge device that which correlates the sensor data with the machine ID based on the slave addresses on the Modbus line. The processed data was then packaged and sent to a NoSQL (Not Only Structured Query Language) low-latency database located in the cloud. The low latency is particularly important considering the real-time requirement for inference and monitoring. The data was transmitted wirelessly using the MQTT protocol on multiple topics for visualization,

storage, and processing.

Model Training

Once retrofitted and deployed, the data collection process occurred continuously and was timestamped for future use. In addition to the real-time data collection, the machine states at any point in time were also continuously monitored. During the model training process, the required sensor data were queried from the database and correlated with the data from the machine's controller. This process helped in identifying the state of the machine. The sensor data further undergoes processing to augment the features of interest corresponding to the classification task in hand. After data labeling and pre-processing, machine learning models were trained to classify and cluster the different error states of the machine. The model training in this study typically involves hyper-parameter optimization followed by 10-fold Cross-Validation (CV) on the best-performing hyper-parameters. The best-performing models were chosen based on the F1-Score, Receiver Operating Characteristic (ROC), and Area Under Curve (AUC). ROC is a graphical plot that illustrates the performance of the binary classification model by plotting the True Positive Rate (TPR) (also called Recall) against the False Positive Rate (FPR) at various threshold levels. The TPR is the proportion of actual positives that are correctly identified, whereas, the FPR measures the proportion of actual negatives that are incorrectly identified as positives. The AUC represents the area under ROC.

Model Deployment

The models developed in this study were not computationally intensive, hence, they could be potentially deployed at the edge or the cloud. During the inference phase, the data were periodically queried from the database and sent to the model for prediction. To reduce the False Positive Rate (FPR), for a single segment of data for prediction, multiple overlapping predictions were made, and a majority vote was taken for the whole segment to decide on a prediction. Several studies have explored the benefits of different model architectures in different scenarios/applications [43]; hence, this framework was developed to allow an ensemble of separately trained models to make inferences without any dependence on each other. A majority voting system can then be followed

to arrive at a single inference.

Machine learning and Deep learning are almost always a lifelong learning process [65]. Monitoring the performance of the deployed models is an active field of research and several studies are being conducted to determine when the model's inference was not reliable. Luckily, in our case, we can interact with the machine's controller, and the model's performance evaluation is complemented by the information extracted from the controller. Furthermore, the equipment's operator can continuously perform corrective actions during erroneous inferences creating a loss-gain scenario. This process will be integrated as a reinforcement learning scenario where there are rewards for correct inferences and penalties for incorrect inferences. Through this process, the weights of the trained models will be updated over time leading to a reduction in FPR and an improvement in the model's robustness over time.

Relation between power consumption and machine error states

The power meters were attached to the 3-phase input of the equipment to ensure reliable monitoring of the power consumed by different components of the machine. The cutting process accounts for only about 15% of the total energy consumed by the machine, which varies depending on machining scenarios. The components that are external to cutting consume the most energy [55]. Hence, it is vital to go beyond the tool-chip interface to monitor the condition of the machine using the power consumption data. The power consumed by the manufacturing machine can be classified into three categories: 1) Power consumed by the accessories, 2) Power consumed by the motors controlling the axes and spindle, and 3) Power consumed when cutting, which is highly dependent on the cutting parameters, and material types. Several works have tried to characterize the equipment solely based on the energy consumption data [53]. Real-time data collection and visualization from our setup provide an insight into the equipment's operation, seen in Figure 3.14. The patterns in the power consumption of a machine tool have valuable information that is readily available and provides a slew of information on the equipment's condition when used appropriately. This task can quickly become insurmountable when considering the different operating states of the machine, cutting operations, tool-workpiece interactions feedrates, etc.

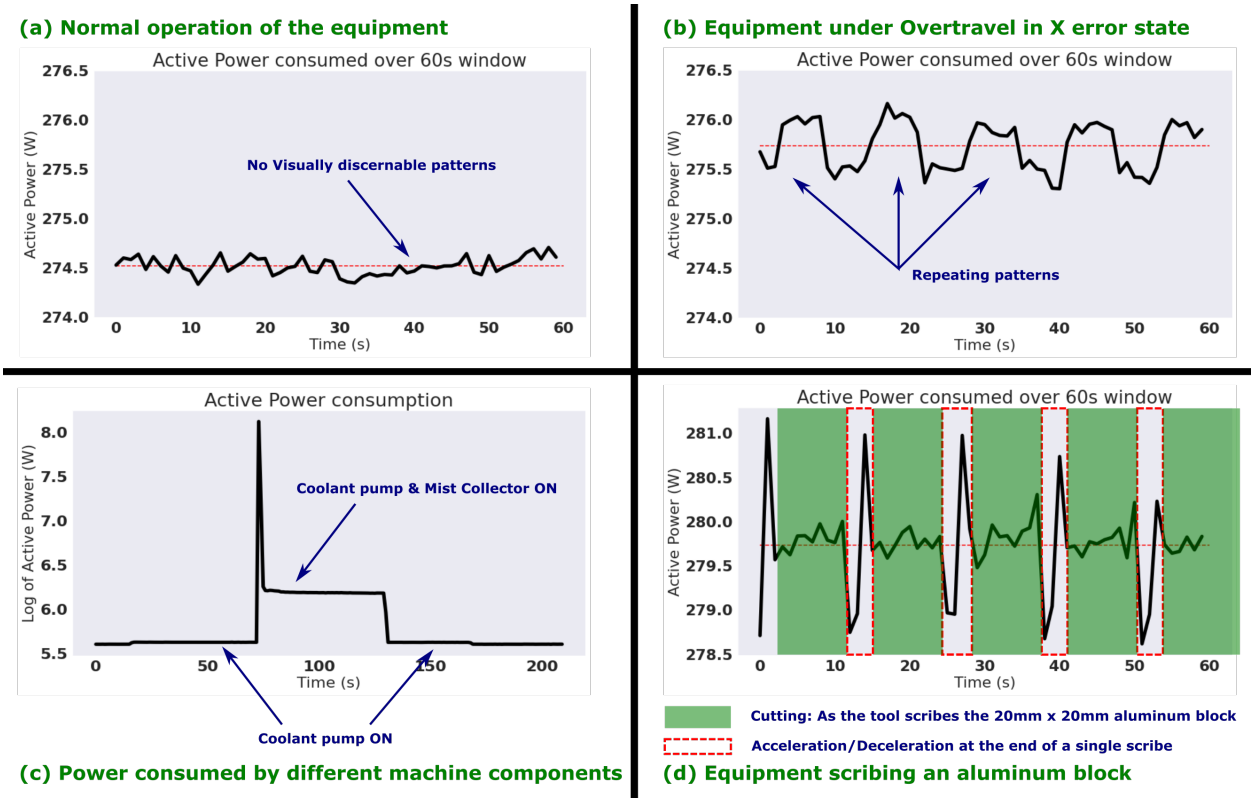
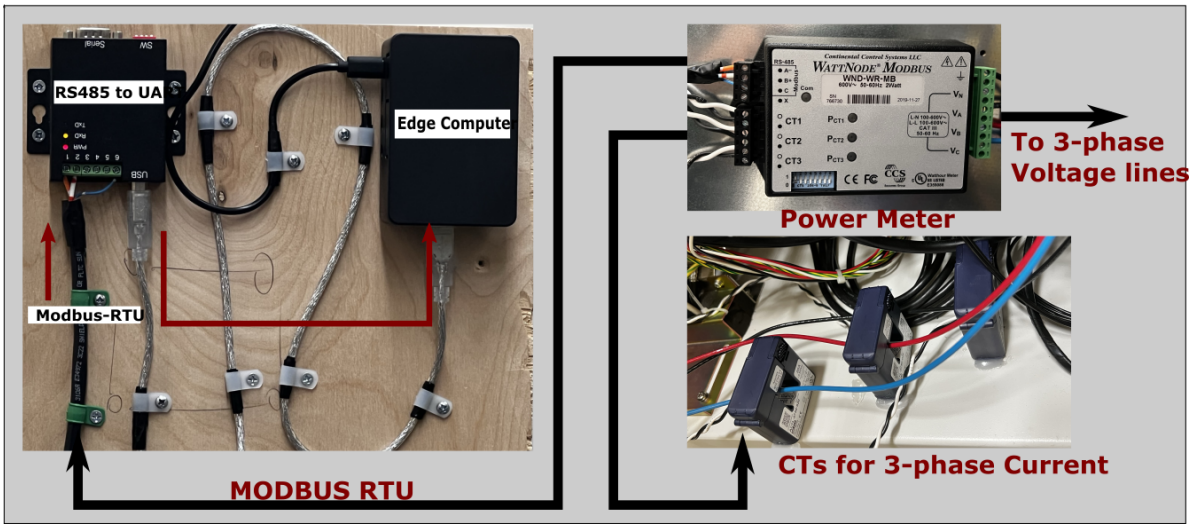


Figure 3.14: Visualizing the active power consumption pattern for different equipment states [64].

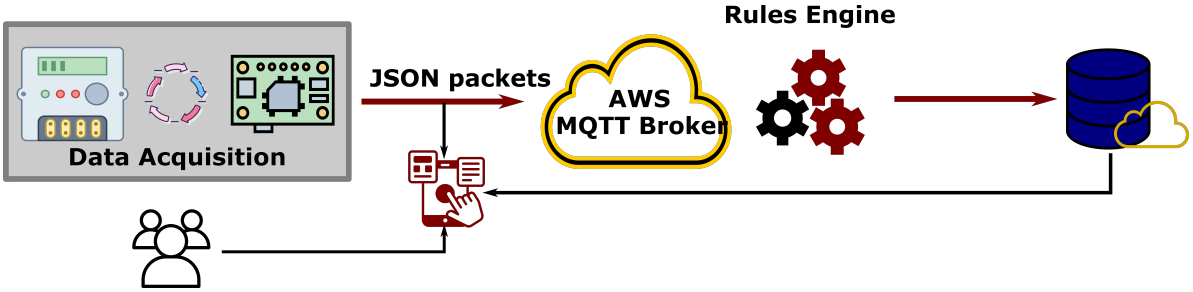
3.3.2 Experiments / Engineering Application

In this section, the following are discussed: 1) Retrofitting a legacy machine to obtain power consumption data in real-time, 2) Applying machine learning to extract patterns of interest from energy consumption data, and 3) Implementing an MLOps pipeline for real-time inference.

For continuous and real-time monitoring of the equipment anomalies from energy consumption, high-precision power analyzers like Fluke Norma 4000 are not practical. Hence, the FANUC ROBONANO-$\alpha 0i$B was retrofitted with a WattNode Wide-Range Modbus from Continental Control Systems (CCS) that can measure power consumption continuously in real-time without extensive modifications to the machine. The power meter can update data at a sampling rate of 10 Hz and communicate over RS485 using Modbus protocol. The power meter was attached to the input of the machine, using 3P3W (3 Phase 3 Wire) configuration. The firmware for the device was custom-written and can be found in the GitHub link associated with this work. 32 input power and energy parameters were obtained from the 3-phase input of the machine and were packaged into



(a) Retrofitting FANUC ROBONANO α-0iB.



(b) Experiment plan.

Figure 3.15: DAQ, transmission, and storage [64].

JSON (JavaScript Object Notation) for transmission. The retrofitted setup along with the process involved in data communication can be seen in Figure 3.15.

The edge device validates and timestamps the sampled data and then sends it to a remote NoSQL database through the MQTT protocol. The rules engine was implemented at the AWS servers to further process the data. The database used was DynamoDB and the MQTT broker used in our case was the AWS IoT Core. The process involved in the data acquisition is shown in Fig. 6. The dataset for this study was created by synthetically generating anomalies on the machine tool and from real production processes. This was feasible because the power consumption data was continuously collected from the machine irrespective of its state, which was then cleaned and categorized appropriately.

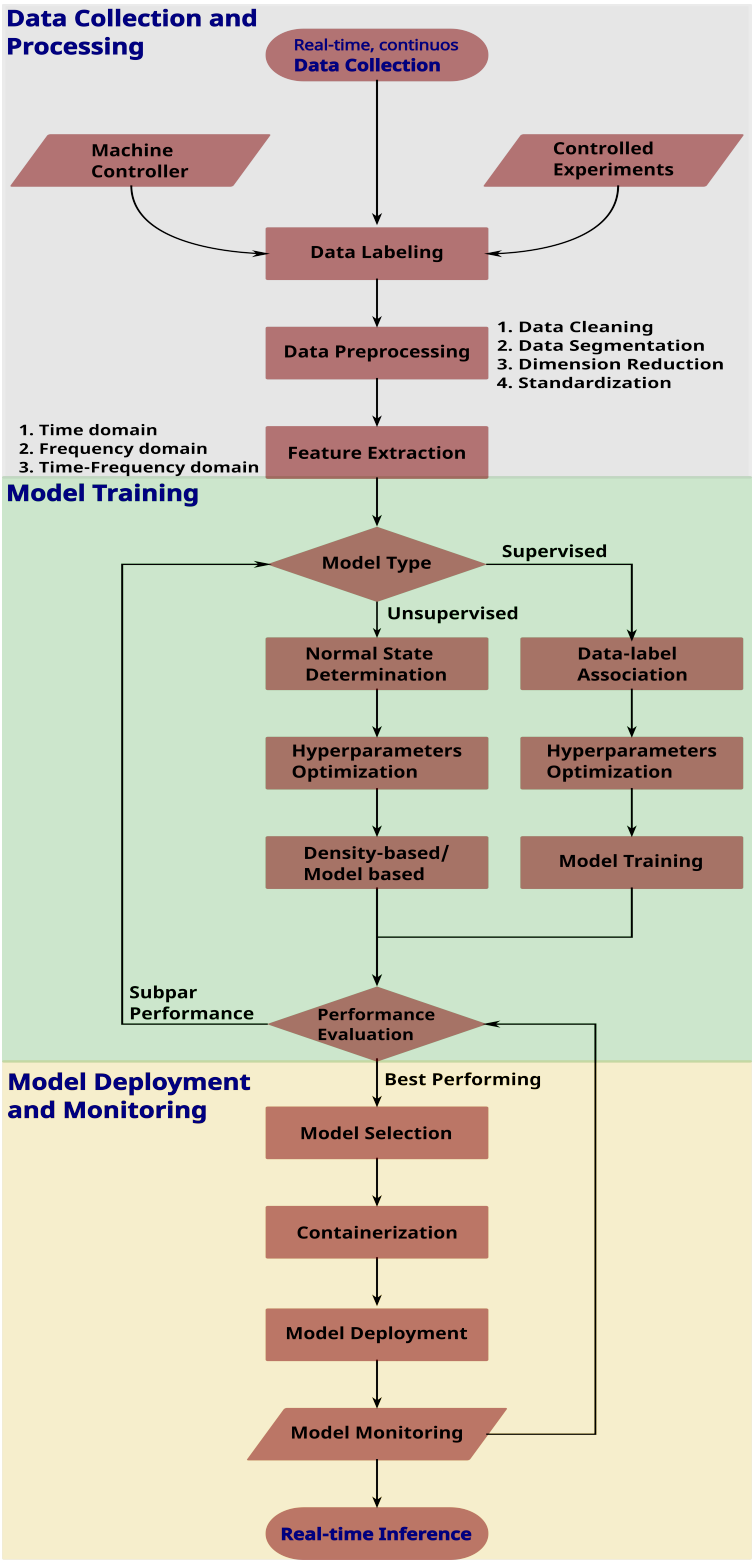


Figure 3.16: Supervised and Unsupervised Model Development Flowchart [64].

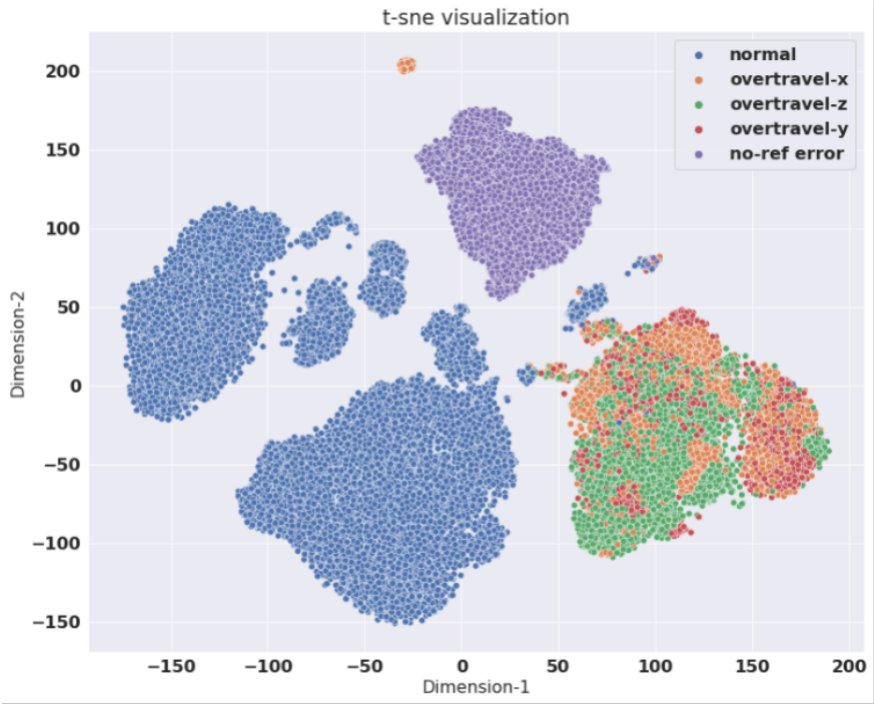


Figure 3.17: t-SNE visualization of the model's input data [64].

Model Development

Once the machine has been retrofitted with appropriate sensors, and the process flow has been defined for data storage and retrieval, the data collection process happens continuously and in real-time. The next challenge was to correlate the data collected from the sensors to the events and anomalies associated with the sensor. In this work, the correlation was done manually by controlled experiments. The different states of the machines were observed and tagged on the database, the equipment was manually pushed to anomalous/error states and the appropriate energy consumption patterns were also tagged in the database. The process flow involved in the model development can be seen in Figure 3.16. The model development process can be categorized into three major categories 1) Data preprocessing, 2) Model training, and 3) Model development and monitoring.

The input data to the model was processed to augment the training process. The first step involved labeling the data that was collected continuously and stored at a NoSQL database. Regarding this study, the labeling process was done manually by associating the machine controller information and controlled experiments. The labeling process involves assigning classes to different

time intervals. The power meter catalogs 32 different input energy parameters, based on our preliminary study, 9 of them were selected for this study. The input energy parameters chosen were Active Power, Reactive Power, and Apparent Power for the 3-phase input of the machine. The time series data was then segmented using windows of size 15s, 30s, 60s, and 120s, with an overlap rate of 60%. The choice of window size was determined later. After segmentation, the features were extracted from the chosen 9 input parameters. The extracted features can be seen in Table 3.5. 153 features, encompassing the time, frequency, and time-frequency domains, were extracted from each segment. If required, the dimension reduction techniques were applied to improve computational efficiency. The data were then standardized to a standard normal distribution to complement the model training process, as the variations in magnitude between the features were high.

Before model development, the data was visualized to better understand the relationship between the classes in an unsupervised manner. To visualize the data in two dimensions, the 153-dimensional features were reduced to 50 using PCA (Principal Component Analysis), followed by t-SNE (t-distributed Stochastic Neighbor Embedding) to 2-dimensions. The machine states considered in the visualization process: normal, machine not referenced, overtravel in X, Y, and Z. From the t-SNE plot, Figure 3.17, the "normal" state, the "not-referenced" state, and combined "overtravel" state clusters were separated from each other. On the other hand, the clusters corresponding to overtravel in X, Y, and Z were overlapping, indicating the prospective issues the model might face when generalizing with regards to these classes. The "machine not referenced" state in this work corresponds to the machine's state after a power cycle, where a referencing operation is required before beginning operations on the machine. In the ultra-precision machine used in this study, it was challenging to identify and classify the state of the machine. Through our preliminary work [59], it was ensured that the change in power consumption during the "overtravel" and "machine not referenced" states were correlated with the machine's state and not the sensor's state. The overtravel of the axes in general leads to an increase in power consumption because of them being at the extreme ends of the axis travel. The difference in overtravel energy consumption between the five axes of the machine was dependent on the machine design and was indiscernible visually from the energy profiles. Since the "overtravel" and "machine not referenced" states were determined not to be a part of the sensor artifact, the impact of different sensors in different machines on the

model development process can be safely ignored.

The model training process was categorized into supervised and unsupervised depending on the application domain. The unsupervised approach does not require the generation of defect instances, thereby simplifying the implementation process.

Time-domain ↓	Impulse Factor	Energy of FFT
Root Mean Squared	Shape Factor	FFT Power Spectral Density
Peak Value	Peak to Peak	**Time-frequency domain ↓**
Variance	Shannon Entropy	Energy of WPD Coefficient one
Crest Factor	Skewness	Energy of WPD Coefficient two
Kurtosis	**Frequency-domain ↓**	Energy of WPD Coefficient three
Clearance Factor	Peak Value FFT	

Table 3.5: 17 features extracted from the 9 chosen parameters for anomaly detection [64].

Supervised Model Development: Before the start of the training process the hyperparameters of the models were optimized using an exhaustive grid search. Once the best-performing hyperparameters were identified, the models were trained using a 10-fold cross-validation (CV). The six models that were developed from this process were then evaluated based on their performance on the testing data, and robustness in the field.

Unsupervised Model Development: Unsupervised anomaly detection was conducted using two approaches: density estimation by profiling the normal instances and model-based.

In the density-estimation-based approach, the normal instances corresponding to the good operation condition of the machine were used to estimate the distribution of the data corresponding to the good state of the machine. Two methods were used to estimate the distribution of the normal instances: (a) Mahalanobis distance-based approach, and (b) Kernel density estimation. The input power consumption corresponding to the normal operating condition of the machine, particularly when the machine was idle, i.e., not performing any manufacturing operation, was assumed to resemble a normal distribution.

Hence in the case of the Mahalanobis distance-based anomaly detection, Eq. 3.5, the distribution corresponding to the "normal" instances was determined by computing the mean and the covariance of the respective data. The covariance helps in understanding the dependence between

Models and Hyperparameters	
Supervised Learning	
Logistic Regression	Class weight: *Balanced*; Penalty: *L2*
Decision Tree Classifier	Class weight: *Balanced*; *Gini Impurity*; MinSamplesSplit: *100*
k-NN Classifier	Neighbors k: *5*
SVC	Kernel: *Linear kernel*
Bagging Classifier	Estimators: *100*
Random Forest Classifier	Estimators: *100*; Max Tree Depth: *15*
Unsupervised Learning	
Mahalanobis Distance Classifier	PCA components: *140*; Threshold: 3σ
Kernel Density Estimation	PCA components: *140*; Quantile Threshold: *0.02*
Isolation Forest	PCA components: *140*; Estimators: *1000*

Table 3.6: Supervised and Unsupervised Models and their Hyperparameters [64].

the 153-dimensional features. As the dimensions increase, due to the "curse of dimensionality" it will get challenging to train, and/or make robust inferences, hence, the dimensions of the data were reduced using dimension reduction techniques. For a new data point, the Mahalanobis distance determines the distance of the data point from the center of the distribution. The Mahalanobis distance threshold to identify an anomaly was set at the 3σ level of the distribution, ensuring a theoretical TPR of 99.7%.

$$\mathcal{D}_M(x) = \sqrt{(\bar{x} - \bar{\mu})^T \Sigma^{-1} (\bar{x} - \bar{\mu})} \tag{3.5}$$

where $x = (x_1, x_2, x_3, ...)$, μ is mean, and Σ is covariance

For Kernel Density Estimation (KDE), the density of the "normal" instances was estimated directly using a Gaussian kernel. The density estimation process using a kernel function can be seen in Eq. 3.6. Once the probability density function (pdf) has been estimated, the data instances corresponding to the normal state of the machine were scored using the determined pdf, and the threshold for anomaly detection was determined using trial and error.

$$f_h(x) = \frac{1}{nh} \sum_{i=1}^{n} K_h(\frac{x - x_i}{h}) \tag{3.6}$$

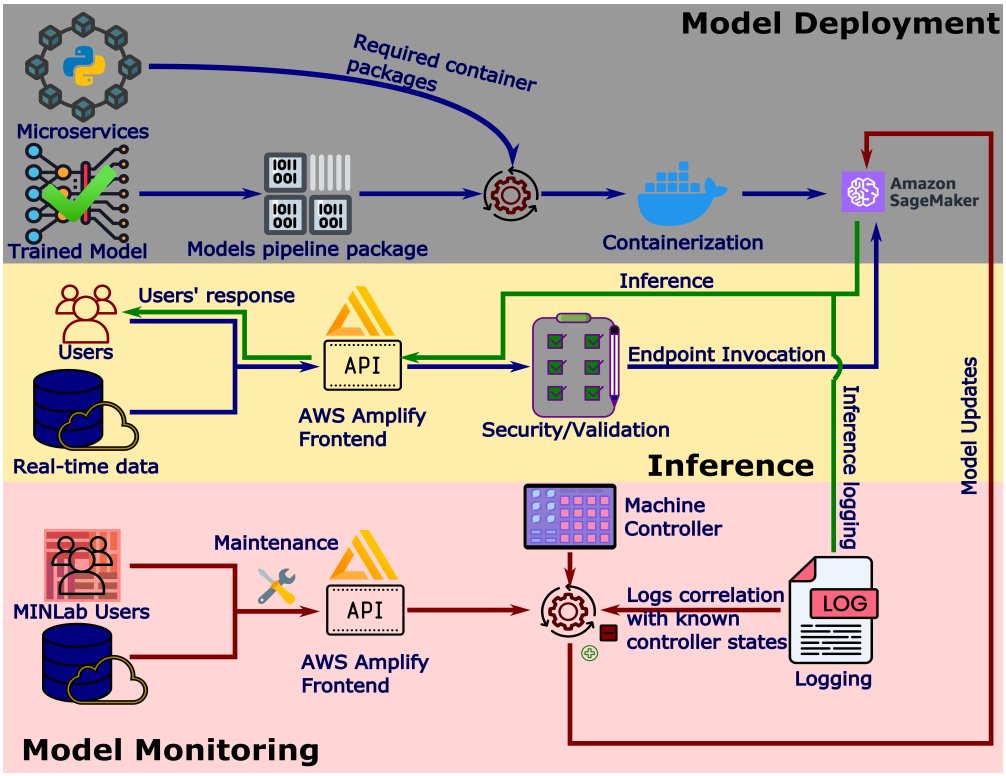


Figure 3.18: Model deployment for energy based anomaly detection [64].

where $K_h(x - x_i)$ is the kernel function.

The methods discussed so far were density estimation techniques, where different approaches were used to estimate the probability density function (pdf) of the normal instances – corresponding to the normal operating conditions of the machine. For the case of model-based anomaly estimation, a technique called the Isolation Forest [66] was used to identify/detect anomalies and outliers in the data. The Isolation Forest was trained on the normal operating conditions of the machine and was tested on the anomalous instances. All the developed models and their hyperparameters are summarized in Table 3.6.

Model Deployment and Monitoring

Once the model training was completed, they were containerized with the necessary packages that complement the inference process. The containers were then hosted using Python-based microservices, such as CaaS (Container as a Service) using AWS Sagmaker Serverless endpoints. The

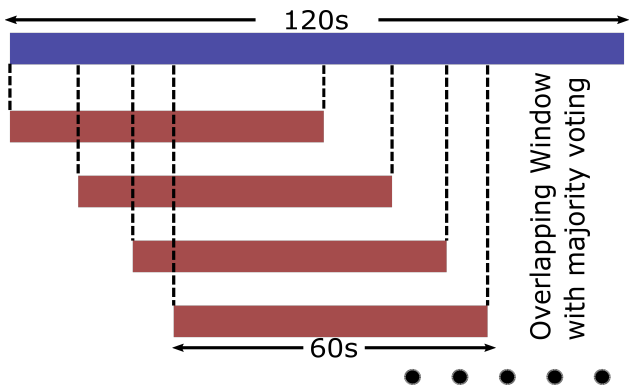


Figure 3.19: Segmented inference process [64].

process flow involved in the model deployment can be seen in Figure 3.18. During the inference process, any user querying the API, located at the AWS Amplify frontend, with the appropriate power consumption data was first validated to ensure the data integrity, followed by directing the request to the AWS Sagemaker with the input data. The inferences from all the models deployed were made using a serverless interface and was then returned to the user as well as logged with appropriate timestamps for further analytics down the line. The inference was categorized into anomaly detection followed by defect identification. The final inference provided to the user was the majority vote of the inferences from all the models trained. The model's performance was monitored periodically by accessing the inference logs and comparing them with the known states of the machine. Currently, the process of model updates is done manually by assigning weighted positive and negative scores to correct and incorrect inferences. The weight determination for the scores depends on how reliably a particular state can be identified by using the controller data. Repeated application of this process will enable us to obtain robust training data over a period, which will then be used to update the model weights. The principle behind this approach lays down the foundation for reinforcement learning, which will be further augmented in future work.

Inference Process

The window size for the segment was set at 60s. In our application scenario, getting an inference every 120 seconds was satisfactory. Hence, to reduce the number of false positives, the 60s overlapping inference over the 120s of data was conducted by shifting the inference window by 10s

for every inference made. This process leads to 6 inferences per 120s segment, and then majority voting was conducted to determine the error state of the machine. In case of a tie, it was always broken by choosing the normal operating state of the machine. The segmentation process can be seen in Figure 3.19.

3.3.3 Results and Discussion

The results from the model training process for supervised and unsupervised learning techniques are discussed in this section. The analysis of the impact of energy consumption monitoring on the anomaly detection of manufacturing machines follows this.

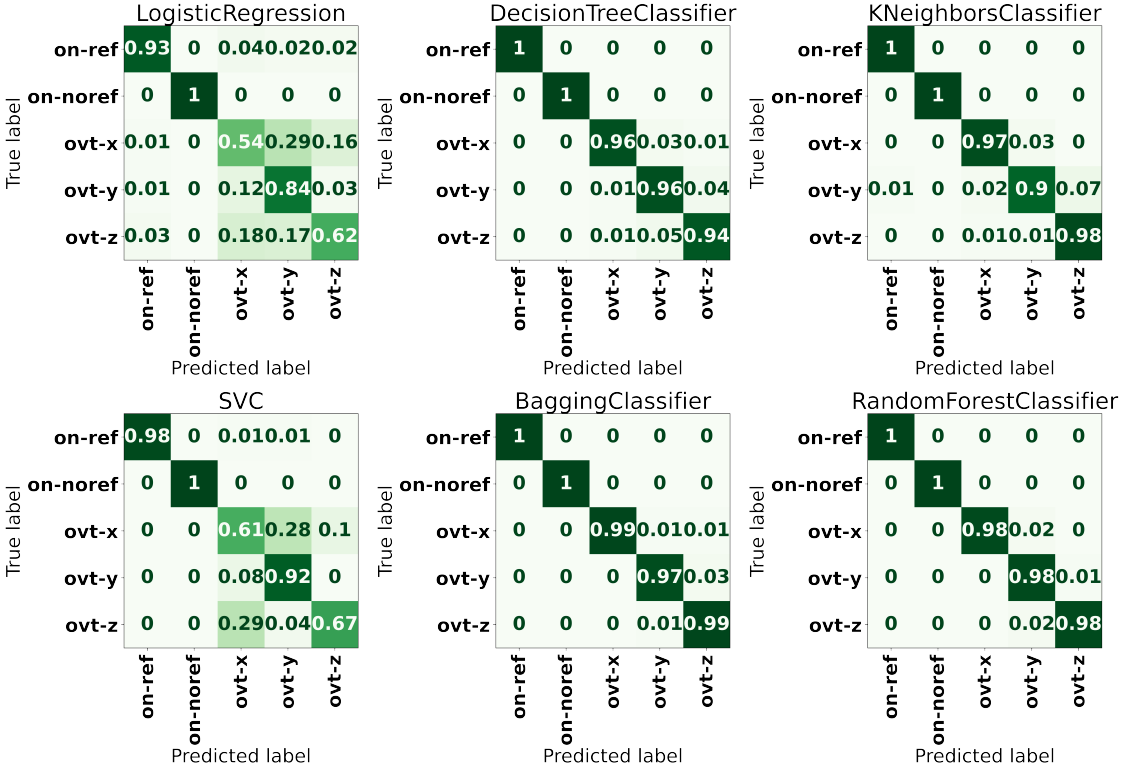


Figure 3.20: Confusion matrix for supervised learning models [64].

Model Evaluation

In the case of supervised learning model development, the models were trained using a 10-fold Cross-Validation. The performance metrics that were used to assess the models' performance were F1-Score, Precision Score, and Recall Score for the minority labels with micro-averaging to eliminate

Defect Type	**Description**
Normal	Machine operational; Laser scale operating and referenced
ON-NotReferenced	Machine ON and operating; Laser scale operating but NOT referenced
Overtravel-X	Laser scale defective or Overtravel in X
Overtravel-Y	Laser scale defective or Overtravel in Y
Overtravel-Z	Laser scale defective or Overtravel in Z

Table 3.7: Defects considered in this study and their description [64].

Metric	**Logistic Regression**	**Decision Tree**	**k-NN**	**SVM**	**Bagging Classifier**	**Random Forest**
Precision	0.8383 ±0.00615	0.9707 ±0.00494	0.9002 ±0.00637	0.8497 ±0.0067	**0.9898 ±0.0025**	**0.9886 ±0.0024**
Recall	0.8388 ±0.00616	0.9719 ±0.00456	0.8982 ±0.00693	0.8499 ±0.0067	**0.9900 ±0.0026**	**0.9891 ±0.0027**
F1-Score	0.8385 ±0.00615	0.9713 ±0.00473	0.8992 ±0.00662	0.8498 ±0.0068	**0.9899 ±0.0025**	**0.9888 ±0.0025**

Table 3.8: Evaluation of defect identification models using 10-fold CV [64].

the impact of class imbalance [67]. Precision quantifies the number of correct positive predictions out of all the positive predictions. Recall quantifies the number of correct positive predictions made from all the correct positive predictions that could have been made. Recall can indicate the missed positive predictions. Maximizing the precision will minimize the number of false positive errors, whereas maximizing the recall will minimize the number of false negative errors [68]. Based on the above metrics the models that performed well were Random Forests and Bagging Classifier. The defects considered in this study were specific to FANUC ROBONANO-$\alpha 0i$B and can be seen in Table 3.7. Six different models were used to detect and classify the defects, and the results from the evaluation can be seen in Table 3.8.

The confusion matrices corresponding to each of the models can be seen in Figure 3.20. From the confusion matrices, across all the models, the worst performing classes were the overtravel in X, Y, and Z. The effect is particularly amplified in Logistic Regression and Support Vector Machines models. This reflects on the fact that the data clusters were not easily separable when using t-SNE visualization, as seen in Figure 3.17.

In case of unsupervised learning models, the models were trained either by estimating the density of the normal instances. Once the training was completed the models were tested against

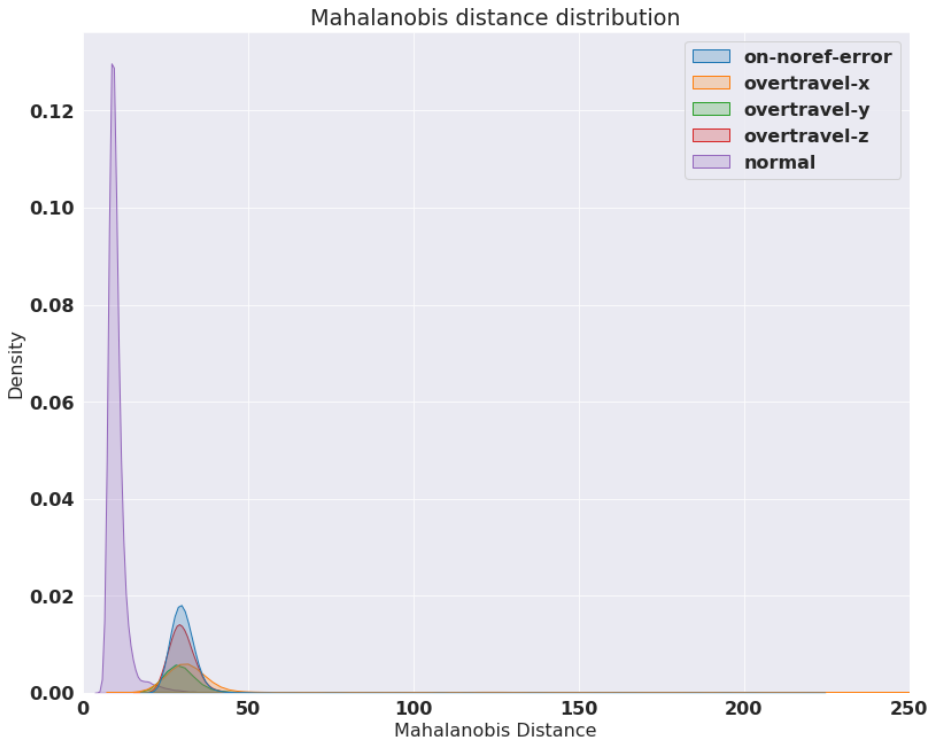


Figure 3.21: Distribution of Mahalanobis distances for all classes [64].

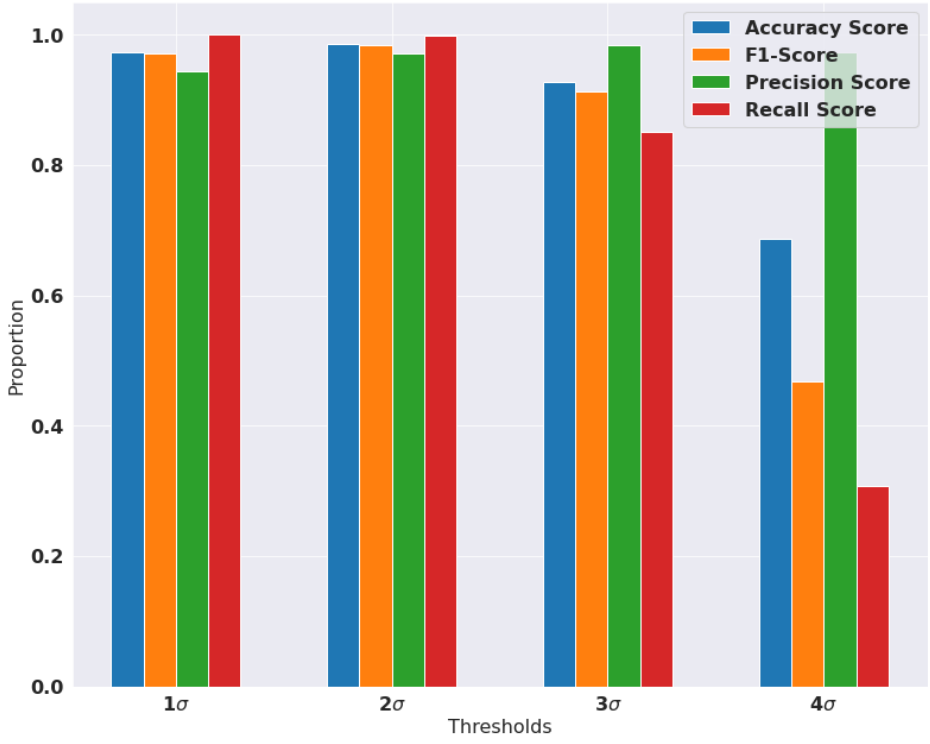


Figure 3.22: Impact of threshold-levels on the performance metrics [64].

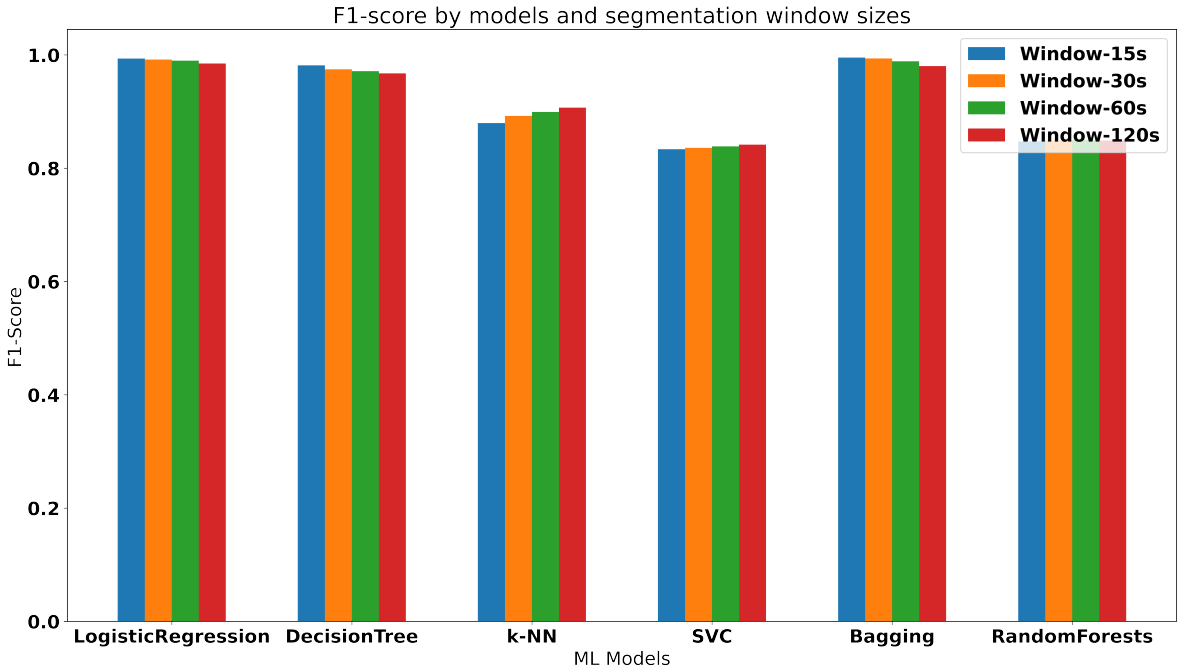


Figure 3.23: Impact of segmentation window sizes [64].

the anomalous instances, which correspond to the different error states of the machine, as well as other unseen "normal" instances. The metrics used to evaluate the model's performance were Precision, Recall, and F1-Score. For the unsupervised learning models: Mahalanobis-distance based, and isolation forest, the F1-Score was determined to be 0.9848, and 0.9701, respectively. To better understand how Mahalanobis distance was used to determine the threshold for anomaly detection, the distribution corresponding to the Mahalanobis distances was plotted for each of the classes considered in this study, see Figure 3.21. From the plot, the data instances corresponding to the normal operation of the machine are closer to zero, and the distributions of the overtravel in X, Y, and Z tend to overlap leading to difficulty in categorizing without a supervised model.

The threshold level in this work was set to be at the 3σ level of the Mahalanobis distance distribution. To better understand the impact of threshold levels, the precision and recall values were computed for different pre-determined thresholds: 1σ, 2σ, 3σ¸ and 4σ, and were plotted in Figure 3.22. Depending on the requirement the σ levels can be updated accordingly.

Impact of data-segmentation window sizes

The segmentation window sizes determine how quickly a prediction can be made. The smaller the size of the segmentation window quicker an inference can be made, considering no overlapping windows/inferences. Reducing the segmentation window size below a threshold can lead to unreliable model performance because the input data packet to the model might not contain sufficient information to make a reliable inference. Hence identifying an optimal window was required to enable real-time inference. Based on the study that was conducted there was a noticeable trend as the window size was reduced, as can be seen in Figure 3.23. In all the models, except k-NN and SVC, the window of size 15s has slightly better performance than other window sizes. In our case, the window size of the 60s was optimal as that provided a balance between the inference speed and information packet size.

Feature Importance Study

In this section, we focus on measuring the impact the input power consumption of the machine has on energy consumption monitoring and anomaly detection. In ultra-precision machining operations, the Material Removal Rate (MRR) is very low to understand the power consumption patterns present in the data visually. Hence sophisticated tools are required to better extract and analyze the patterns. In the first scenario, we aim to understand the top 5 important features among the 153 input features to the models. The method used in identifying the important features is called Permutation Importance. The feature importance study was repeated 100 times with different random seeds for every model, and the average decrease in the mean accuracy was computed on the testing dataset. To better understand the impact of the input features, dimension reduction techniques were not applied during this study. The most impactful features for the anomaly detection process were identified by selecting 10 features for each model that have the highest decrease in the mean accuracy. This amounts to a total of 60 items, 10 for each model used in this study, then, the frequency of occurrence of each feature was computed, see Figure 3.24. The top 5 important features i.e., the most frequently occurring ones, were Wavelet Packet Decomposition (WPD)-3 energy, peak value, WPD-2 energy, WPD-3 energy, and peak FFT, in the

order of their importance. Out of the top 5 features, the majority fall under the time-frequency domain features category.

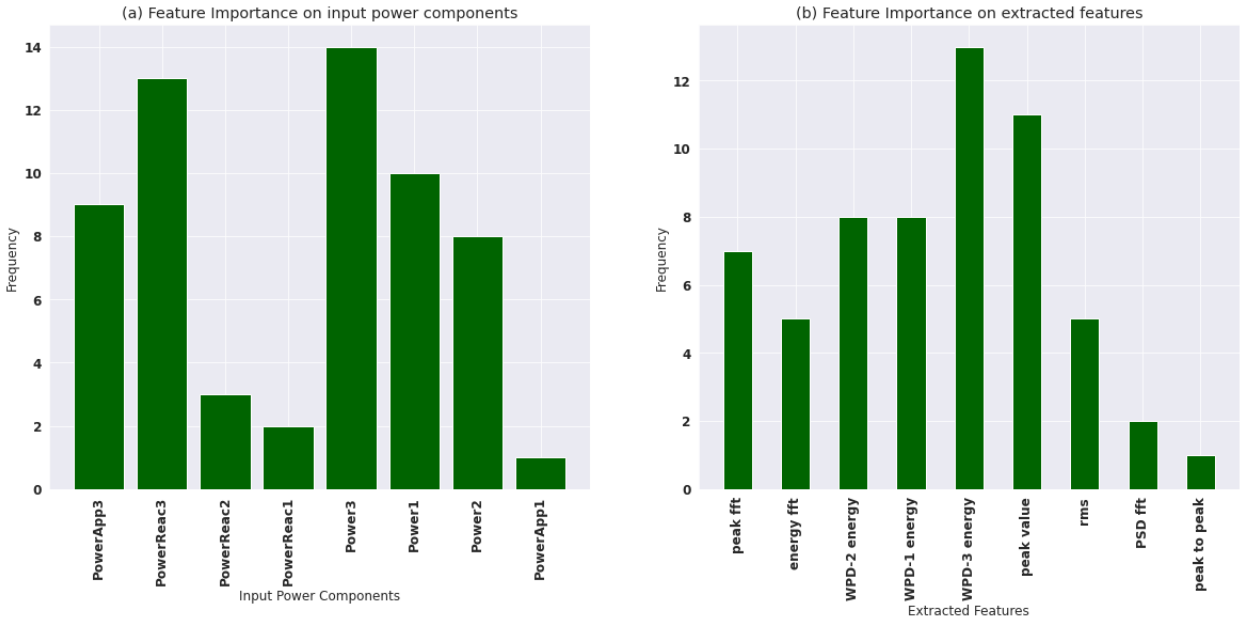


Figure 3.24: Two-folded feature importance study [64].

The feature extraction process in this study was done on two levels, first, the features corresponding to the input power components, second, the features extracted in time, frequency, and time-frequency domain from each of the input power components. From the Figure 3.24, the top-5 power components were Active Power of Phase-3 (Power3), Reactive Power of Phase-3 (PowerReac3), Active Power of Phase-1 and Phase-2, and Apparent Power of Phase-3 (PowerApp3). From the above analysis the following inferences can be made: (a) Time-frequency domain features have the most impact in enabling the defect identification process. This aligns with our hypothesis that within a segment of data that was used to identify any anomalies, it is important to understand the frequency components present and the time at which they occur. (b) The Active Power component is the most important for anomaly detection, the Reactive Power follows this. This again aligns with our hypothesis and has been validated by our preliminary work on equipment state identification using Deep Learning [34]. When anomaly detection/defect identification was carried out in real-time on a machine under operation, effectively we are looking into the motors controlling the machine's axes. Hence, the Reactive Power components become important. (c) The energy meter was attached to the input of the machine so that the power consumed by all com-

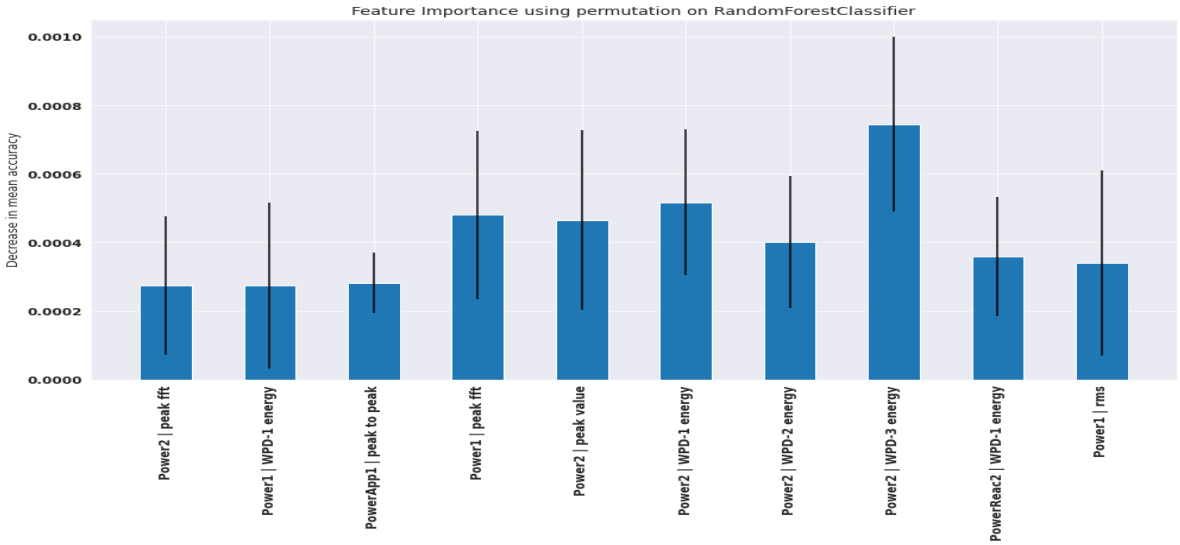


Figure 3.25: Top-10 important features for Random Forest classification algorithm [64].

ponents of the machine could be tracked and measured. The reasoning behind why the 3rd Phase power components were the most important was because of the fact all the motors controlling the five axes of the machine were attached to the input Phase-3. Hence, they had the most impact when detecting anomalies corresponding to the machine's operation.

Finally, using the feature importance study, the robustness of the developed models was tested. It was found that the ensemble models, bagging trees and random forest, were highly robust to the permutation of any single input feature. The mean decrease in accuracy for these models was lowest when compared to all the other models used in this study. Figure 3.25 shows the mean decrease in accuracy for the random forest classifier with the error bars for standard deviation after 100 repetitions of the feature importance study. As can be seen from the plot, the average decrease in the mean accuracy was 0.05%.

Comparison with other DAQ devices

In an industrial scenario, using identical DAQ devices across all manufacturing machines might not always be possible. Since feature extraction was adopted in this study instead of the feature learning process, the models developed should be transferrable to other DAQ devices without the need for retraining. Hence, in this section, the robustness of the developed models was tested

by evaluating their performance on the data collected by a different power analyzer. The models were trained using the data from a WattNode and were tested on a high-precision one by Fluke, Norma-4000. All the models used in this study performed well. The F1-Score, Precision, and Recall were respectively, 0.9829, 0.9871, and 0.9789 for the Random Forest classifier. It can be seen from this study that the developed models can be easily ported across different power meters of different resolution and accuracy without the need for any re-training if the input features to the models were kept consistent. We believe this would enable or at the very least complement easier deployment of machine learning models across the manufacturing facility swiftly, enabling widespread deployment. However, further validation with more DAQ devices is necessary.

3.3.4 Discussion

This chapter studied the prospect of using equipment's power consumption data to identify anomalies in its operation. A feature importance study was then conducted to identify the key features contributing to ultra-precision CNC machine anomaly detection. Dashboards have been built using AWS (Amazon Web Services) to showcase our work and validate the system's performance. The dashboard showing the live energy reading for the machine can be seen here, and the fault detection system for the same machine can be seen here, at the time of this writing. This study proves that monitoring the energy consumption of the machine can complement the fault detection of manufacturing machines. The repository containing the code for model design, training, evaluation, and hosting using a serverless architecture can be found at https://github.com/vigneshuw/machine_fault_identification.git.

3.4 G-code based power prediction

Predicting the energy consumption of CNC machining operations offers significant benefits, including insights into energy planning, optimization of machining strategies, and energy conservation. Recently, data-driven models for energy prediction have gained popularity due to their high accuracy, the ability to complement and, in some cases, even enhance physics-based models, and relatively lower modeling complexity [60, 69]. While energy prediction is a common area of study,

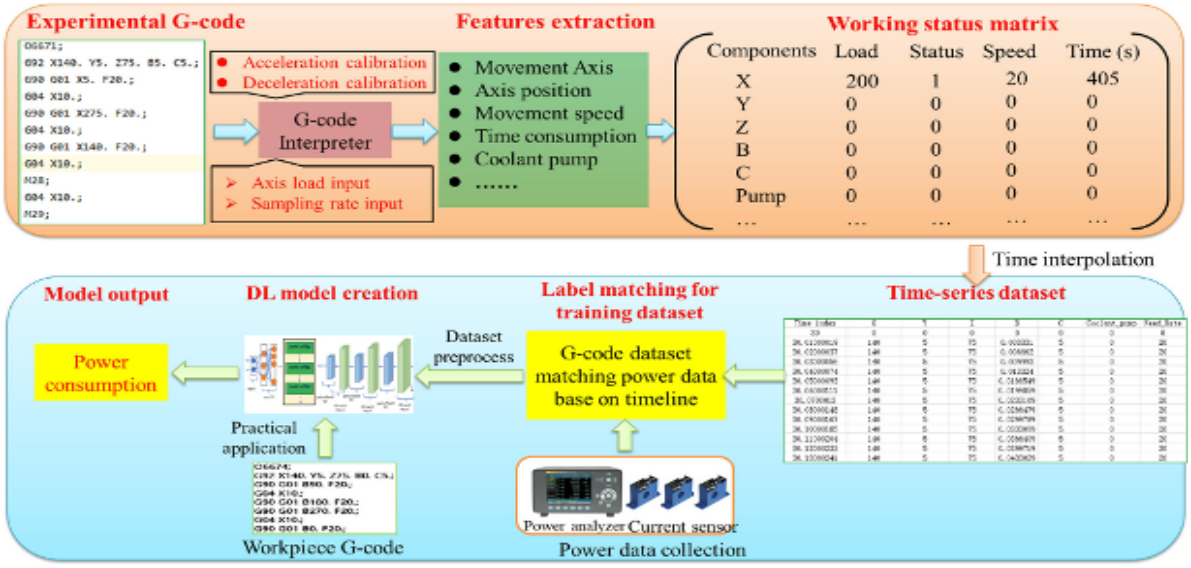


Figure 3.26: Framework for the energy prediction model development [60].

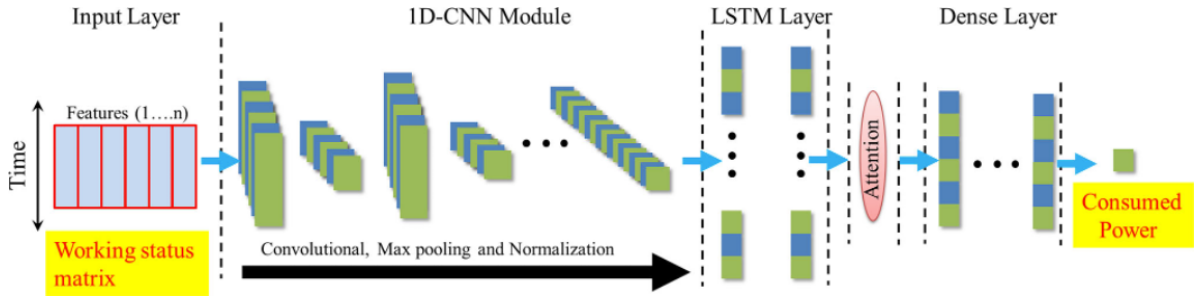
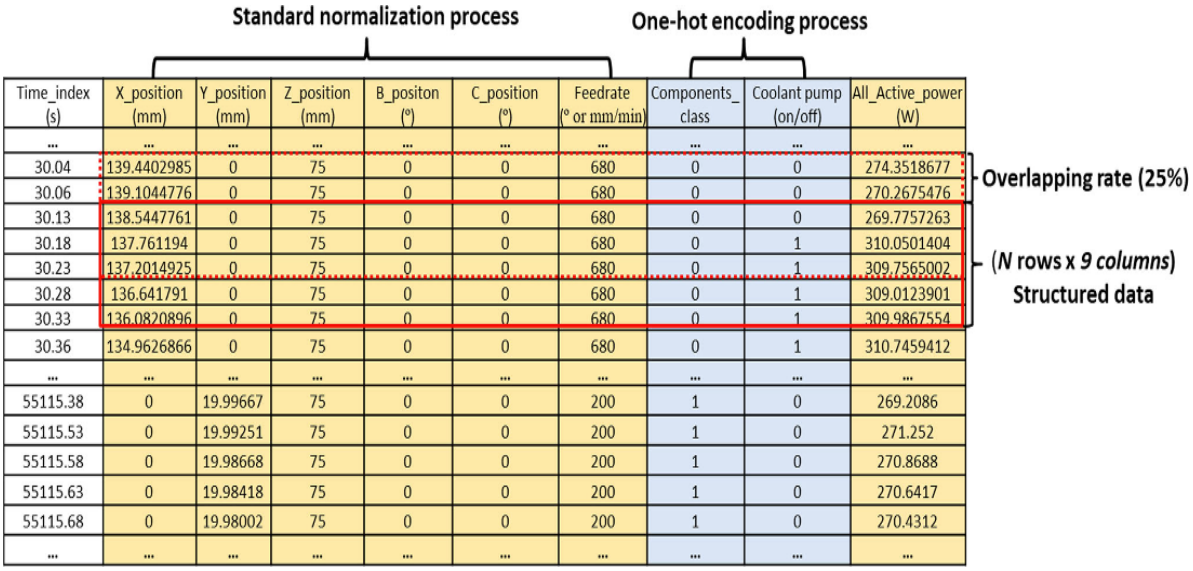


Figure 3.27: Architecture of 1DCNN-LSTM-Attention model used for energy prediction [60].

its applications in UPM remain largely unexplored. This section presents a data-driven model developed to predict the instantaneous power profile of a 5-axis UPMT. Additionally, we explore future research directions where data-driven models could be used to augment physics-based models, highlighting potential advancements in predictive accuracy and practical applications.

For an UPMT, the power consumption of the machine tool can be broadly classified into the following three categories: 1) Constant power—power consumed by the controllers and other accessories, 2) Variable power—power consumed by the servos controlling the axes, which depends on the position the axis relative to a datum, 3) Material removal power—power required to remove material, highly dependent on the process parameters and the material's property. Unlike conventional machining, the material removal power is low in UPM.



Time_index (s)	X_position (mm)	Y_position (mm)	Z_position (mm)	B_positon (°)	C_position (°)	Feedrate (° or mm/min)	Components_ class	Coolant pump (on/off)	All_Active_power (W)
...	...	...	...	...	...	...	...	...	...
30.04	139.4402985	0	75	0	0	680	0	0	274.3518677
30.06	139.1044776	0	75	0	0	680	0	0	270.2675476
30.13	138.5447761	0	75	0	0	680	0	0	269.7757263
30.18	137.761194	0	75	0	0	680	0	1	310.0501404
30.23	137.2014925	0	75	0	0	680	0	1	309.7565002
30.28	136.641791	0	75	0	0	680	0	1	309.0123901
30.33	136.0820896	0	75	0	0	680	0	1	309.9867554
30.36	134.9626866	0	75	0	0	680	0	1	310.7459412
...	...	...	...	...	...	...	...	...	...
55115.38	0	19.99667	75	0	0	200	1	0	269.2086
55115.53	0	19.99251	75	0	0	200	1	0	271.252
55115.58	0	19.98668	75	0	0	200	1	0	270.8688
55115.63	0	19.98418	75	0	0	200	1	0	270.6417
55115.68	0	19.98002	75	0	0	200	1	0	270.4312
...	...	...	...	...	...	...	...	...	...

Figure 3.28: Dataset for training the 1DCNN-LSTM-Attention model [60].

The framework behind the approach developed for energy prediction can be seen in Figure 3.26. The G-code interpreter discussed in Section 3.2.2 was used to obtain the working status matrix similar to our previous studies. To enable energy prediction, the G-code interpreter was further upgraded to include to the acceleration and deceleration effects. These effects were experimentally determined by interfacing with the CNC controller and determining the rate of change of feedrate as the axis comes to an halt. Incorporating these effects into the G-code interpreter helped improve the accuracy of the app to perform interpolation between successive lines of G-code. The working status matrix along with the data logged by the power meter was used to train the DL model for energy prediction. Figure 3.27 shows the 1DCNN-LSTM-Attention model used for this study to predict the energy consumption from the working status matrix.

3.4.1 Model Development

The develop the data-driven model for energy prediction, the same set of experiments described in Section 3.2.3 and illustrated in Figure 3.9 were conducted. Given the effectiveness of the Long Short-Term Memory (LSTM) architecture in capturing long-term dependencies within time series data, this model was selected for training. After parsing the G-code with the G-code interpreter, the dataset prepared for model training is shown in Figure 3.28. This dataset encapsulates key

information essential for predicting energy consumption across different equipment states:

1. The travel distances of each axis in the 5-axis machine within a predetermined time window.
2. The operational status of the machine's auxiliary components.
3. Synchronized power consumption recorded at each time instant.
4. The machine's feedrate.

3.4.2 Evaluation

Table 3.9: Error Metrics Descriptions and Formulas.

Metric	**Description**	**Formula**
Mean Absolute Error (MAE)	The average magnitude of the errors in predictions	$\frac{1}{N}\sum_{i=1}^{N}\lvert y_i - \hat{y}_i\rvert$
Mean Squared Error (MSE)	The average of the squared errors in predictions	$\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$
Mean Absolute Percentage Error (MAPE)	The percentage of the absolute error relative to the actual values	$\frac{100}{N}\sum_{i=1}^{N}\left\lvert\frac{y_i-\hat{y}_i}{y_i}\right\rvert$
R-squared (R^2)	The proportion of variance explained by the model	$1-\frac{\sum_{i=1}^{N}(y_i-\hat{y}_i)^2}{\sum_{i=1}^{N}(y_i-\bar{y})^2}$

The evaluation was conducted on two levels. First, after training, the model was tested on an unseen dataset to assess its generalization ability. Alongside the 1DCNN-LSTM-Attention model, other model architectures were evaluated to demonstrate its performance advantage. As shown in Figure 3.29, the 1DCNN-LSTM-Attention model exhibited the lowest prediction error among the models tested. The various error metrics used for comparison are detailed in Table 3.9.

To further validate the developed model, we designed a workpiece sample that required a face milling operation. The generated G-code for this operation was input into the 1DCNN-LSTM model to predict power consumption. This process is illustrated in Figure 3.30. The results demonstrate that the model accurately predicted power consumption throughout the CNC machining process, with a prediction error of approximately 0.3W. The mean predicted power was 270.3629W, closely aligning with the true power of 270.3609W. Using the G-code interpreter, the power consumption was mapped to specific machine components active at various time instances. This analysis revealed

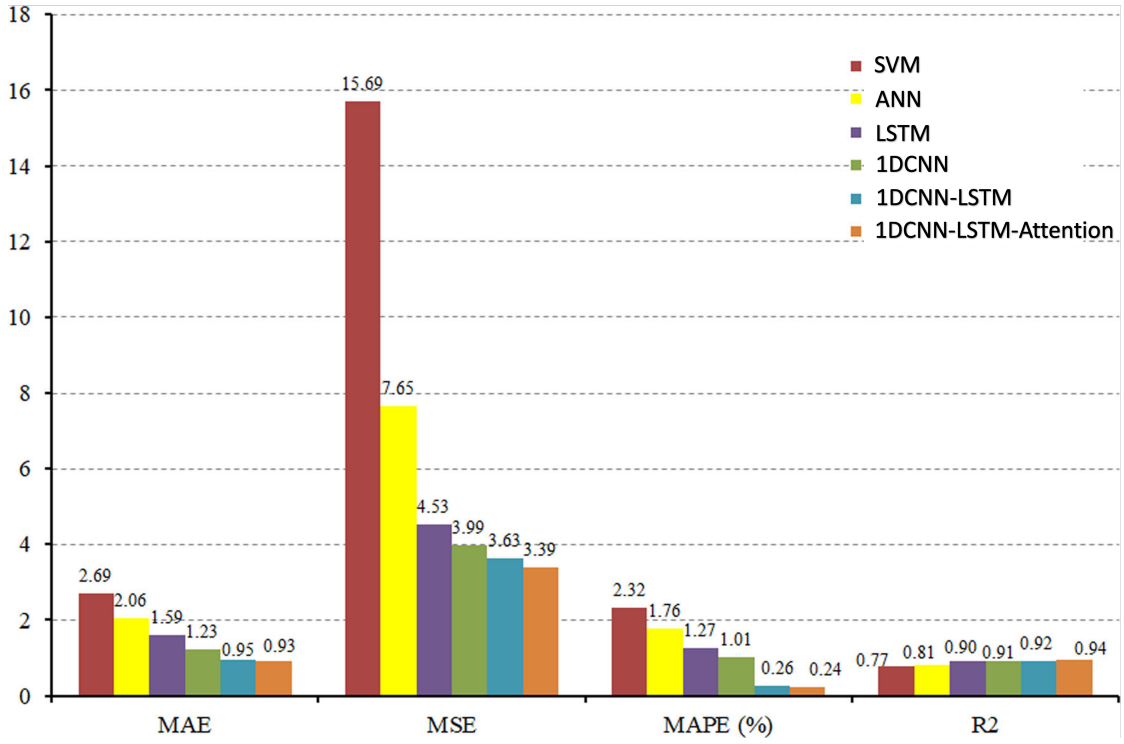


Figure 3.29: Comparison of prediction errors across multiple models [60].

that approximately 36% of total energy consumption was attributed to accessories, followed by 25.2%, 21.4%, and 14.3% consumed by the X, Z, and Y axes, respectively.

3.4.3 Discussion

The study on energy prediction, combined with equipment state identification, demonstrated the effectiveness of a physics-augmented, data-driven approach for energy monitoring. Leveraging the G-code interpreter, we effectively generated training data to enhance data-driven model training. The 1DCNN-LSTM-Attention model outperformed several other ML and DL models in predicting energy consumption directly from G-code, validating the feasibility of the proposed data-driven modeling approach for applications in UPM. Future work incorporating more integrated and higher sampling rate DAQ systems will likely further enhance both model performance and the G-code interpreter application. By utilizing higher-quality data, we can improve the underlying physics modeled within the G-code interpreter, advancing the accuracy and reliability of energy predictions.

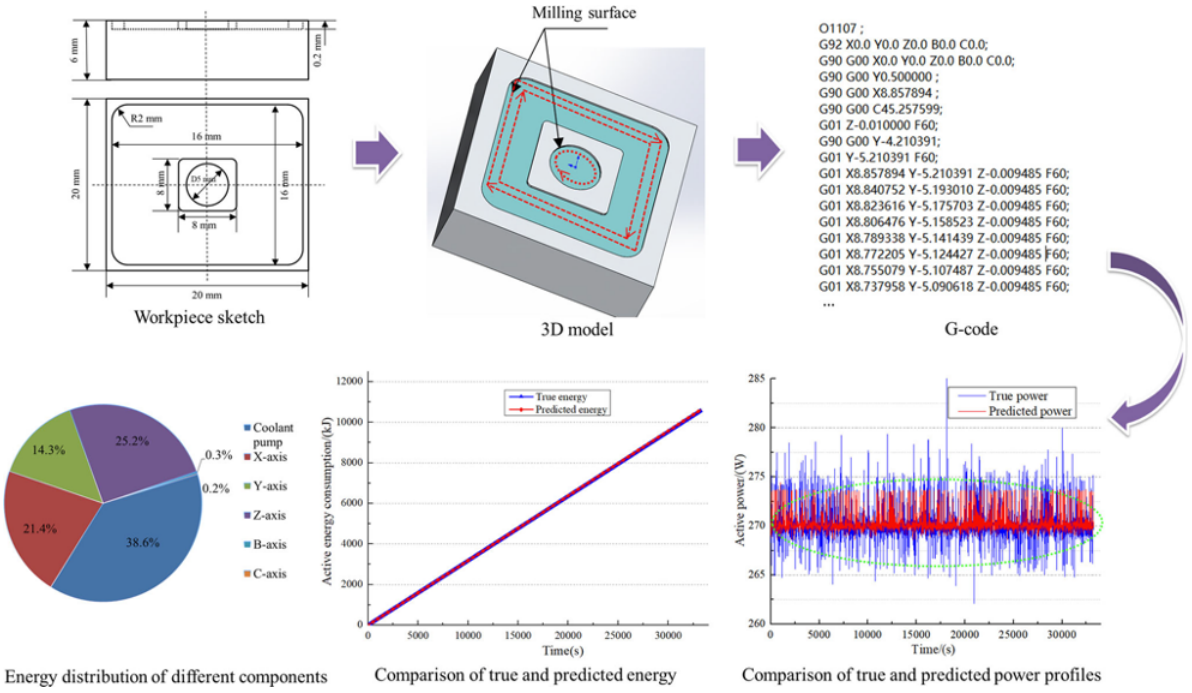


Figure 3.30: Evaluation of the energy prediction system on an engineering use case [60].

3.5 Enabling Effective DAQ and Logging

Developing data-driven models for energy consumption monitoring required us to conduct multiple controlled experiments where both G-code information and energy parameters had to be measured synchronously. Alternatively, we needed to bookmark the specific G-code executed on the CNC machine at any given point in time. It would be highly advantageous to develop a software application that could run on a Single Board Computer (SBC) located near the CNC machine's Human Machine Interface (HMI), interfacing with the machine controller via Ethernet, RS232, or other serial protocols. To achieve the above-mentioned use case, a custom application named "Machine Whisperer" was developed as shown in Figure 3.31, and detailed in Appendix B.2. The application would offer several key benefits:

1. Enabling CNC machine operators to log their activities, providing enriched data on cutting tools, workpieces, and operational details that enhance our understanding of the energy dynamics in various machining scenarios.

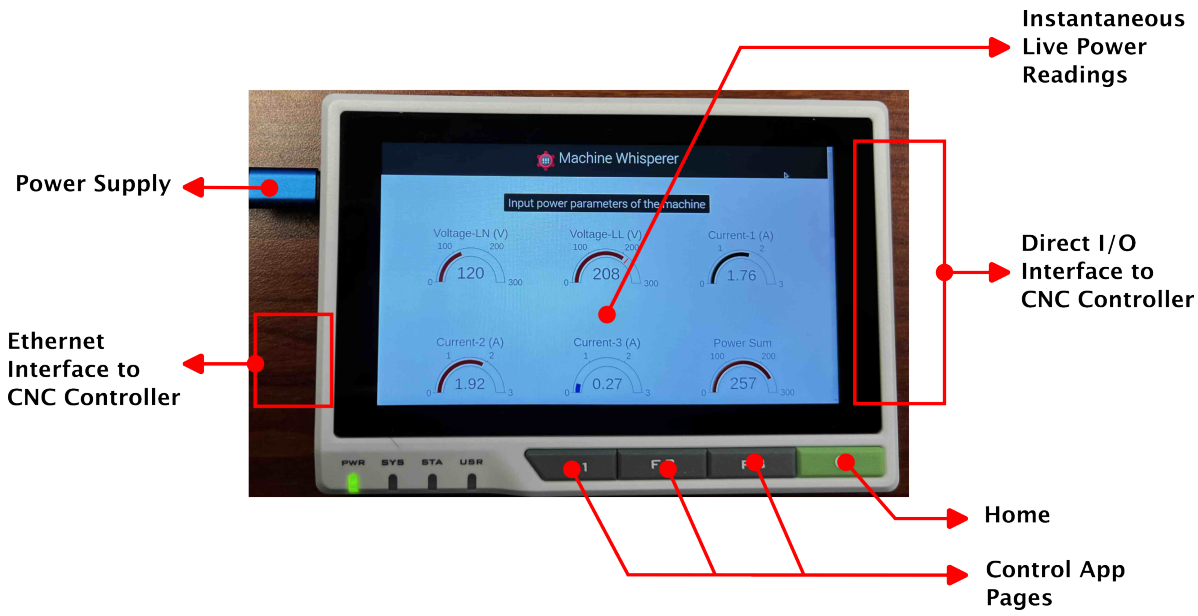


Figure 3.31: An app to enable autonomous data collection for energy monitoring.

2. Synchronizing machine operations with energy data acquisition by interfacing with the machine controller, facilitating more precise energy monitoring without the need for manual annotation.

3. Generating high-quality, synchronized data, reducing the reliance on controlled experiments, and enabling the collection of diverse, real-world data, thereby enhancing model robustness and adaptability.

The primary objective of this application is to automate the data collection process for energy monitoring studies. By leveraging Ethernet and Direct I/O interfaces, the application reliably captures all interactions with the CNC machine. Even if the user forgets to log an entry, the G-code information can still be extracted directly from the CNC machine controller via Ethernet, ensuring comprehensive and accurate data acquisition. This setup provides a dependable DAQ and labeling system for energy monitoring, significantly streamlining the study process and enhancing data reliability.

3.6 Challenges

In AC power systems, **transients** refer to short-duration, rapid changes in voltage, current, or power that occur due to sudden disturbances in the electrical system. These transients are typically characterized by their short duration, high amplitude, and non-periodic nature. In the context of energy monitoring for CNC systems, various types of transients can occur, and capturing these transients is crucial for performing effective energy monitoring studies. The primary focus of this study is on current transients, which may arise due to inrush currents or load changes. Inrush transients can occur when the motors in a CNC machine are powered on or during speed changes, while load transients result from sudden shifts in load, such as rapid acceleration or deceleration of the CNC table or changes in cutting forces acting on the tool.

Commercially available power analyzers are effective for assessing overall power quality but fall short when it comes to measuring transients for our use cases, at the time of this writing. To address this limitation and facilitate future studies, a custom energy monitoring hardware was designed. The insights from our research work have been integrated into this module to conduct energy monitoring studies at a large scale. This system can synchronously sample voltage and current at a maximum rate of 32KS/s, which exceeds the required sampling rate of 5–10KS/s necessary to capture current transients. The energy monitoring hardware developed for these studies is shown in Figure 3.32, and the system's capabilities are outlined in Table 3.10.

Specification	**Description**
Voltage Input	3-phase Voltage, 230V max
Current Input	3-phase Current (0 – 0.3333Vac CTs)
Sampling Rate	Synchronous 32 KS/s per channel
Data Logging	SD Card & USB-FS
Wireless Communication	BLE (Bluetooth Low Energy)
RS485 Communication	Isolated RS485 – High Voltage Isolation

Table 3.10: Energy Monitoring Hardware Specifications.

The hardware enables synchronous sampling of voltage and current for further processing or logging raw data to an SD card. Additionally, it is equipped with BLE for remote configuration of the monitoring process, along with a dedicated USB-FS interface for data streaming to edge devices

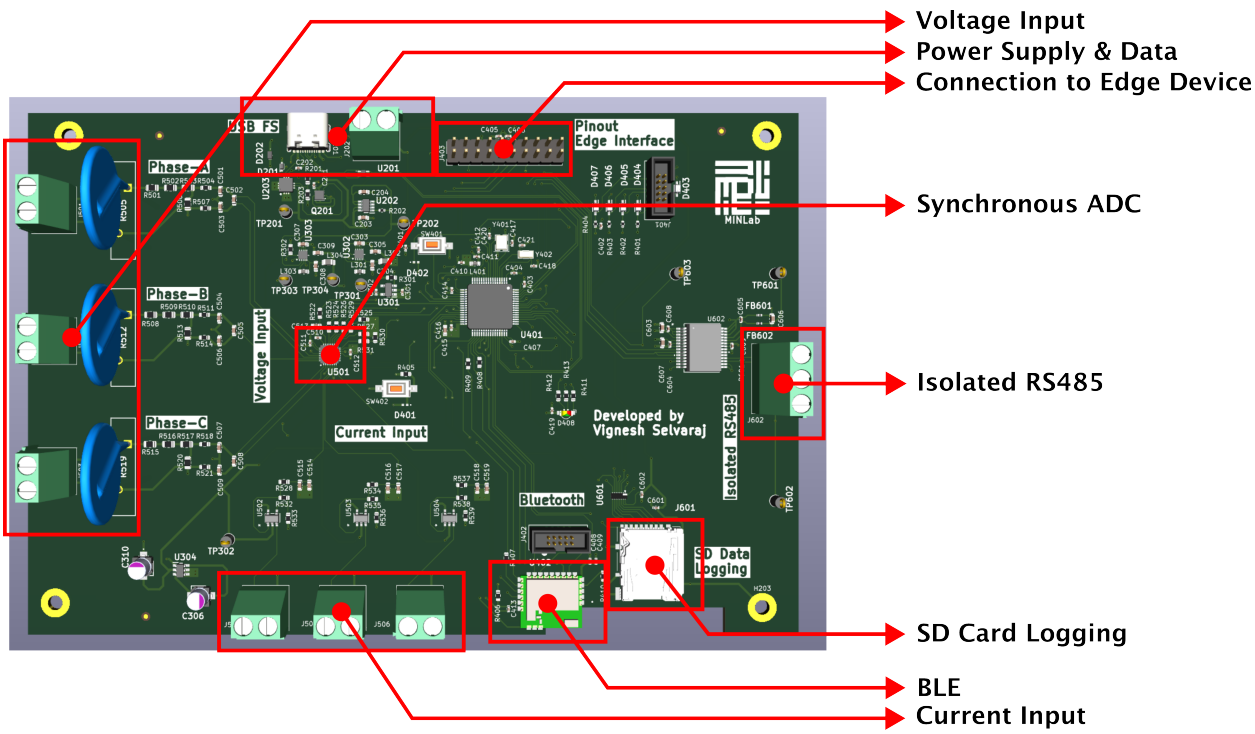


Figure 3.32: Custom-designed energy monitoring hardware for future studies.

if needed. Synchronizing the data collection process with the CNC controller running the G-code is essential, as it allows correlation between power data and CNC operations. This synchronization enables the energy consumption of the machine to be mapped to each line of G-code. Lastly, the energy monitoring hardware includes a dedicated port for interfacing with an SBC to execute advanced algorithms at the edge. The schematics and more detailed information on the hardware can be found in Appendix B.1.

3.7 Future Work

The future work is divided into two phases. In Phase 1, the energy monitoring study will be expanded to develop energy-saving strategies for CNC machines. A more detailed analysis will be conducted using the newly created energy monitoring hardware to measure current transients in real time and correlate them with the G-code. In our preliminary work [60], we developed a DL model capable of predicting energy consumption from G-code for a specific CNC machine. Building on this, our Phase 1 work will not only focus on predicting energy consumption based on G-code but also on suggesting strategies to reduce energy usage and minimize peak energy consumption,

contributing to sustainable manufacturing.

In Phase 2, we aim to address a fundamental challenge that is pervasive in manufacturing industries: the inability to capture "Institutional Knowledge" or "Tribal Knowledge effectively." Trial knowledge, also known as institutional knowledge, refers to the collective understanding, experiences, practices, and expertise that accumulate over time within an organization or industry. This knowledge is often informal and undocumented, yet it plays a critical role in day-to-day operations and decision-making processes. Retaining and transferring this knowledge is essential for maintaining continuity and efficiency, especially when key personnel leave the organization.

In simpler terms, our goal in Phase 2 is to capture the tribal knowledge associated with CNC operation—particularly in areas such as improving product quality and energy efficiency—by developing an autonomous agent. This agent will continuously monitor the user's interaction with the machine, with energy consumption being one of the key parameters it tracks. The insights gathered will help encode the tacit knowledge typically held by experienced operators, enabling more effective decision-making and operational improvements. Our Phase 2 work will begin by employing the "Imitation Learning" strategy, where the reward function models expert knowledge, constrained by the machine's effective energy consumption. This approach will allow the agent to learn and replicate the best practices of skilled operators, optimizing both energy use and operational efficiency.

Chapter 4

Monitoring Of Human-Centric Assembly Processes

This chapter extends the monitoring of manufacturing operations to human-centric assembly operations. A novel approach and algorithm were developed to monitor assembly operations autonomously and in real time. The objective of this study is threefold: 1) To determine the step time and cycle time of a human-centric assembly operation in real time. 2) To detect anomalies like sequence breaks and missed steps in an assembly operation. 3) Finally, to develop an approach to detect NVA activities in assembly operation without explicit training of deep learning algorithms on them. A part of our work presented in this chapter has been published in Selvaraj et al. [70, 71] and Selvaraj and Min [72].

4.1 Literature Review

In a typical manufacturing facility, each product passes through a series of assembly stations consisting of predefined steps called Standard Operating Procedures (SOPs). Improvement in the assembly work is an important task for increasing the productivity of the assembly line [73]. In assembly operations, an operator repetitively performs a series of predefined tasks. The time associated with each task is called the step time, and the total time required to complete all of them in an assembly workstation is called the cycle time. In industries, up to 40% of the cost and 70% of the production time falls under assembly operations, either in intermediate assembly operations or in final finished product assemblies [20]. Hence, the quality of the assembly operations hugely impacts the end-product quality and the lead time of a product. Human-centric smart manufacturing (HSM) has been growing in the recent decade. The HSM involves the integration of human-in-loop technologies [74, 75], and we believe monitoring of human-centric manual assembly operations in manufacturing facilities is a part of HSM.

Activity recognition using body-worn sensors has been studied extensively [76–79]. In addition to manufacturing, activity recognition is used in healthcare, elderly care, fitness tracking, activity

logging, etc. [80, 81]. Maekawa et al. [73] used wearable sensors to determine the start time and end time of an assembly step. Koskimaki et al. [82] studied an approach where a single wrist-worn inertial measurement sensor was used to identify some basic tasks such as hammering, screwing, spanner use, etc., using k-NN. This study provides a means for identifying the non-value-added (NVA) activities by using a specific distance boundary for the k-NN model. Stiefmeier et al. [83] presented an approach of continuous activity recognition using motion sensors and ultrasonic hand tracking on a bicycle maintenance operation. The work also features the recognition of a "NULL" class, corresponding to the NVA activities. Tao et al. [84] used data from IMU and surface electromyography (sEMG) sensors to identify some of the common assembly activities with Convolutional Neural Networks (CNNs). Tao et al. [85] studied the importance of sensor locations on the human body for different activities using an attention-based sensor fusion mechanism. Some approaches have also tried combining both IMU sensors and cameras to detect human actions in assembly lines [86]. Chen, Jafari, and Kehtarnavaz [87] surveys the use of depth sensors and inertial sensors simultaneously for human action recognition. The work states that using both sensors simultaneously helps in increasing the performance compared to them being used alone. Deep learning has pushed the boundaries of Human Activity Recognition (HAR) using wearable sensors. Zhang et al. [88] provides a comprehensive survey summarizing the existing work on HAR using wearable sensors and deep learning.

When using hand-worn or body-worn sensors, a few challenges need to be addressed to enable the complete realization of the developed technology in manufacturing industries. The two main concerns are operator discomfort and safety concerns that prevent operators from wearing the sensors. In certain cases, the sensors themselves could potentially damage the product being assembled, for instance, the assembly of electrically sensitive components. To ensure continuous and uninterrupted action detection and localization when using body-worn sensors, it is up to the operators to continuously charge, attach, and turn them on manually when required. The authority to ensure that the sensors are operating effectively is up to assembly operators, which is not ideal. When monitoring an assembly operation it is not only required to detect the action but also to determine how long a particular action is being performed. To the best of our knowledge, the majority of the studies conducted through body-worn sensor monitoring aim to recognize the

actions. In addition to action recognition, it is equally important to localize the detected actions. Identifying the time associated with each action is important in the case of assembly operation monitoring as it enables determining the step time and identifying assembly anomalies. Hence, it would be increasingly beneficial to have a non-contact system with the least impact on the assembly line operators.

The temporal action localization of the assembly operations using cameras, with the ability to detect anomalies in real-time, was novel at the time of this writing. The field of action localization from videos has been studied using traditional approaches for many years, with the deep learning-based methods gaining traction recently [89]. The conventional methods were built based on hand-crafted features, where features like SIFT, HOG, etc., were extracted from static images. The traditional methods have strong interpretability but can be specific to the problem. In our case, the type of features extracted will depend on how the camera was positioned. If the assembly operator were to be in the view of the camera, then features specific to human behavior need to be considered. Hence the generalization ability of the traditional methods is limited. On the other hand, deep learning methods enable feature learning and do not necessitate designing the hand-crafted feature extraction process. Karpathy et al. [90] used CNNs to classify 1 million YouTube videos belonging to 487 classes by identifying the actions performed. Simonyan and Zisserman [91] proposed a two-stream deep convolution network architecture that incorporates spatial and temporal networks. It was also found that temporal convolutional networks significantly outperformed spatial convolutional networks. Tran et al. [92] proposed a spatiotemporal feature learning process using deep 3D-CNN for a large-scale video dataset. Using the C3D model, the authors could process the RGB frames and optical flow information simultaneously. In addition to CNN-based architectures, recurrent neural networks were also used to capture time series historical information for action recognition [93, 94]. Carreira and Zisserman [95] re-evaluated the state-of-the-art architectures for action classification on a new Kinetics Human Action video dataset in addition to proposing a novel Two-Stream Inflated 3D ConvNet called the I3D-ConvNet. The works presented so far aim to recognize the actions in a trimmed video, where the video segment contains only one action, and the deep learning models were used to recognize the actions performed. In the case of untrimmed videos, it is required to recognize the actions and localize them, as a video could

contain multiple actions. Several approaches were followed that aim to detect the temporal action proposal i.e., the temporal interval that specifies the start and end of an action segment [96–100]. In these studies, it was assumed that the complete trimmed or untrimmed videos were available during the time of inference. In our case, the inference was required in real-time, hence, the model has access only to past information. The real-time requirement also necessitates the inference to be made for each video frame.

The complexity and variety of assembly operations demand the use of digital technologies to support and ensure the safety of human operators. Xiong et al. [101] used the two-stream approach that was developed in Simonyan and Zisserman [91] for human activity detection in an assembly scenario to detect and recognize the actions. Urgo, Tarabini, and Tolio [102] used deep learning to identify the actions performed by assembly operators. A hidden Markov model was used to identify deviations from planned execution and alarms were raised when assembly operations were not completed. The challenge with this approach is that it cannot measure the step time and cycle time of an assembly operation, and it requires pose estimation and mapping of working regions to a specific operator pose. Chen et al. [103] used deep learning models to recognize repeated actions in an assembly operation. Two models were used in this approach, a YOLOv3 [104] was used in detecting the tools, and a Convolutional Pose Machine (CPM) [105] was used to identify the human joint coordinates. The use of object detection models, such as YOLOv3, along with other deep learning algorithms to detect human actions in assemblies has been studied extensively [106, 107]. Chen et al. [108] used an image segmentation technique to monitor assembly operations. The system was able to detect missing and wrong assemblies, and any errors in the human pose information leading to an incorrect assembly sequence. This approach only works after the product has been assembled, and the inferences on the assembly quality were made on the end product, hence, real-time detection of assembly defects was not possible.

Deep learning models for detecting human actions in assembly processes have been extensively studied. However, the works mentioned earlier either focus on measuring step time or detecting defects in assembled products, but not both. Moreover, none of these studies explore or evaluate real-time monitoring capabilities. This chapter addresses these gaps through three key contributions. First, a novel algorithm called SMIRL was developed to detect and localize actions, as well

as identify anomalies in an assembly workstation. Second, a method for detecting NVA activities in an assembly workstation, without explicit training of the deep learning model, was introduced using an OOD detection approach based on energy-based learning. Lastly, to enhance action detection and localization, a graph-based modeling approach was developed, enabling the decomposition of videos into spatial and temporal graphs.

4.2 Problem Statement

Continuous real-time monitoring of worker-centric assembly operations allows manufacturers to not only estimate cycle and step durations accurately but also detect anomalies, such as sequence breaks and missed steps. The ability to continuously measure step durations helps to identify bottlenecks in the assembly process effortlessly. Moreover, by detecting errors as they occur, real-time monitoring improves the quality of the final product, reducing both rework time and costs. As a result, this study aims to enable manufacturing industries to optimize productivity, identify the root causes of quality issues, and enhance first-pass yield by providing immediate feedback to rectify defects.

Traditional methods for monitoring manual assembly, such as using a stopwatch, have several limitations. Firstly, the process is tedious and exhausting, requiring an individual to observe operations for extended periods. Secondly, the likelihood of detecting sequence breaks and missed steps is low due to human error. Most importantly, operators may feel uncomfortable performing their tasks while being actively monitored by another person. Therefore, there is a clear need for a robust and reliable assembly monitoring system that minimizes impact on operators while enhancing productivity. Advances in High Performance Computing (HPC) and AI have made automatic assembly monitoring feasible. With operator convenience in mind, this study explores the use of overhead cameras to record assembly operations. However, analyzing recorded video data for assembly performance poses three key challenges: (1) reliably and robustly inferring the steps of assembly operations, as deep learning models may jump between assembly steps at a frame-level inference, (2) continuously tracking and identifying the system's global state at any given time to automatically calculate cycle time, and (3) ensuring real-time monitoring with inferences made at

a minimum of 30 frames-per-second (FPS).

Several studies have attempted to understand operational behavior by attaching wearable sensors to workers. While this method is effective in recognizing actions, it struggles with generalization, as sensor types and placement must be customized based on the tasks and operators involved. Additionally, this approach can burden assembly operators and negatively impact productivity, as they are required to wear the devices while performing their tasks. To address these limitations, this study proposes using vision-based technology to monitor worker-centric manual assembly operations. Two algorithms were developed to detect and localize the actions performed in an assembly workstation. The first algorithm called SMIRL was capable of not only detecting and localizing the actions but also was able to identify assembly operation anomalies in real-time. The second algorithm involves deconstructing the video into spatial and temporal graphs, thereby being able to model the actions performed in an assembly workstation effectively.

4.3 SMIRL

This section outlines the methodology used for action detection and localization of assembly SOP steps. It begins by introducing the framework of the action recognition and localization study, followed by an in-depth discussion of the model architectures. Finally, the real-time inference module is presented, which is comprised of two key components: the inference machine and the state machine, both of which will be covered in detail. The latter part of this section focuses on the evaluation of SMIRL.

4.3.1 Framework

The framework developed in this work aims to generalize the detection and localization of actions from video streams for assembly operation monitoring. The requirement for real-time monitoring mandates the inference to be made at the frame level, where the deep learning model can only use the video frames in the past to make an inference. Firstly, the deep learning models were required to be trained on the assembly steps before being linked to the state machine. The framework, from model training to model inference, can be seen in Figure 4.1.

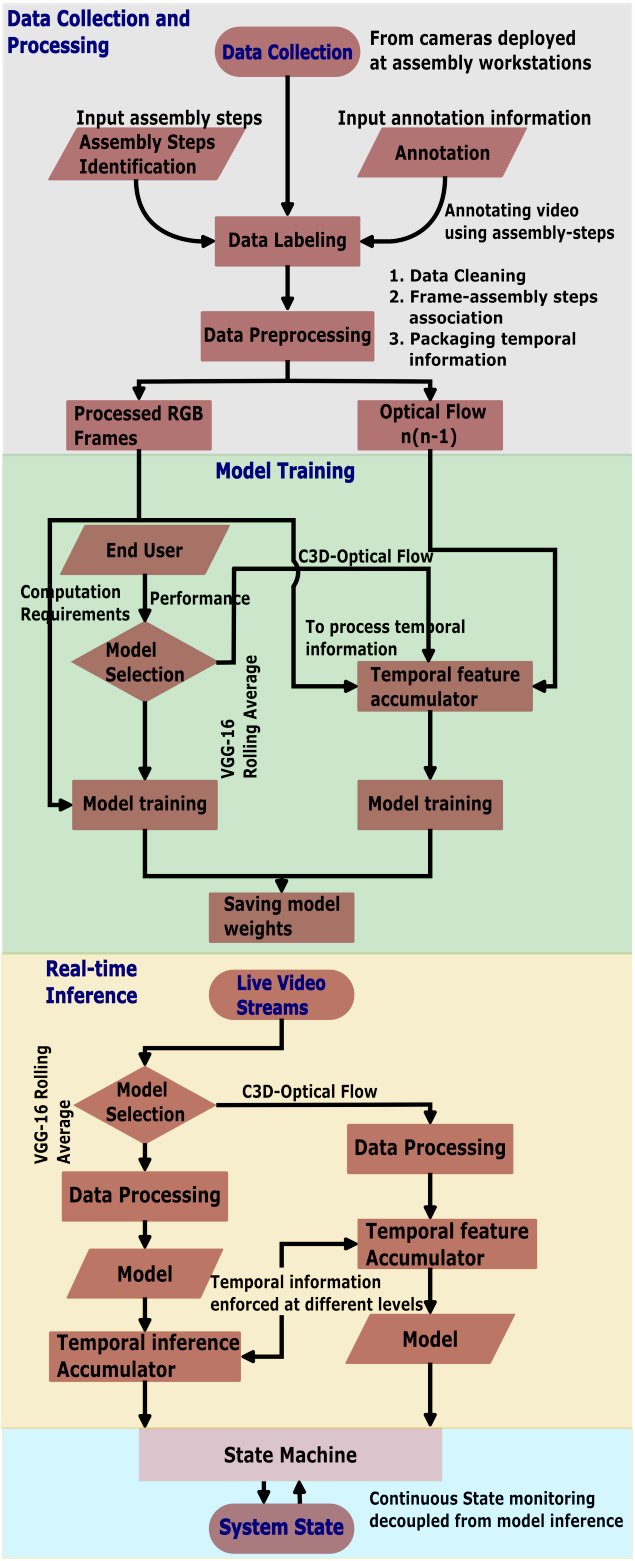


Figure 4.1: Schematic diagram of our proposed system architecture showcasing model development and real-time inference as a combined process flow diagram [70].

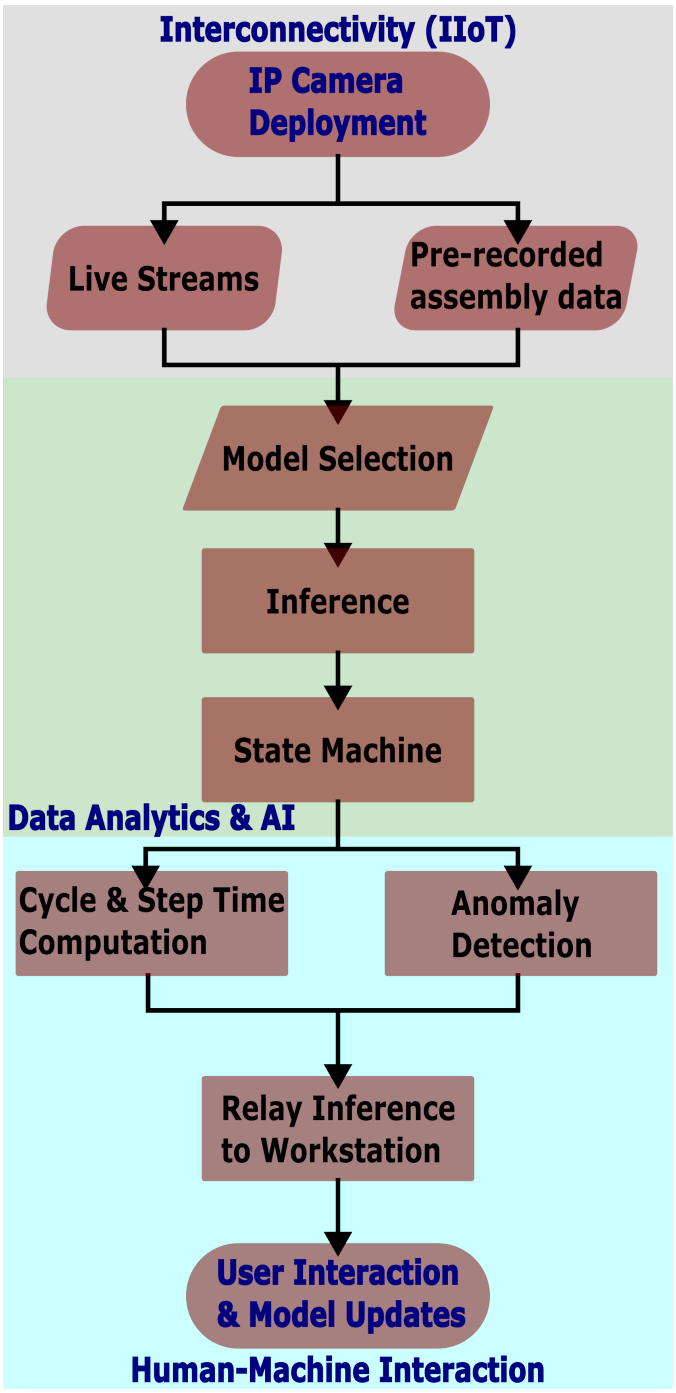


Figure 4.2: Simplified process flow diagram for real-time inference interconnected with the Industry 4.0 pillars [70].

The framework was divided into three segments: Data collection and processing, Model training, and Real-time inference. The data collection and processing segment involves setting up cameras and collecting data on assembly operations on the factory floor. The data was then annotated appropriately based on predetermined Standard Operating Procedure (SOP)s for the assembly operation. After annotation, it was processed by computing the optical flow information between successive frames depending on the type of model used in the inference machine. In the model training segment of the framework, the end-user decides on the type of deep learning model depending on the computation requirements and the performance. The data processing for model training depends on this choice. To enable reliable action detection, it was required to present the temporal information to the deep learning models [91]. The temporal accumulator shown in the flowchart decides when the temporal information was utilized. For the case of the C3D-OpticalFlow model, the temporal information was presented at the model's input by passing a sequence of video frames to the model, whereas, for the VGG16-RollingAverage model the temporal information was incorporated by aggregating the model's softmax probabilities at the time of model inference. The real-time inference segment of the framework contains two sub-segments: model inference, and state machine inference. The model inference was a part of the inference machine and enabled action detection, which was then piped to the state machine for action localization. The process flow diagram in Figure 4.1 involves algorithm development and real-time inference. Hence in Figure 4.2, a simplified process flow diagram for real-time inference and their relation to the industry 4.0 pillars was presented.

4.3.2 Model Architectures

Two models were developed to detect the actions present in a video stream, VGG16-RollingAverage and C3D-OpticalFlow, and their respective architectures are shown in Figure 4.3. The VGG16 model forms a part of the VGG16-RollingAverage model and uses the weights trained on the ImageNet dataset [40]. During the training process, the classification layer of the original VGG16 model was removed and the model was appended with dense layers. The number of layers appended to the pre-trained VGG16 model was a hyperparameter determined using a simple grid search approach. The layer count was set to three for the architecture used in this work. Finally, a

new classification layer was added to the model corresponding to the number of action classes. During the training process, the input to the VGG16-RollingAverage was the frames from the video, preprocessed and reshaped to $224 \times 224 \times 3$. When making inferences using the VGG16-RollingAverage model, the inference was obtained by averaging the softmax probabilities for the past n frames and then picking the class with the highest probability. The parameter n controls the temporal resolution, i.e., the number of frames in the past to be used when making an inference called the temporal length. The averaging process applied here was the rolling average, where the new softmax probabilities for a video frame push out the old ones in a buffer of size n, hence the name VGG16-RollingAverage.

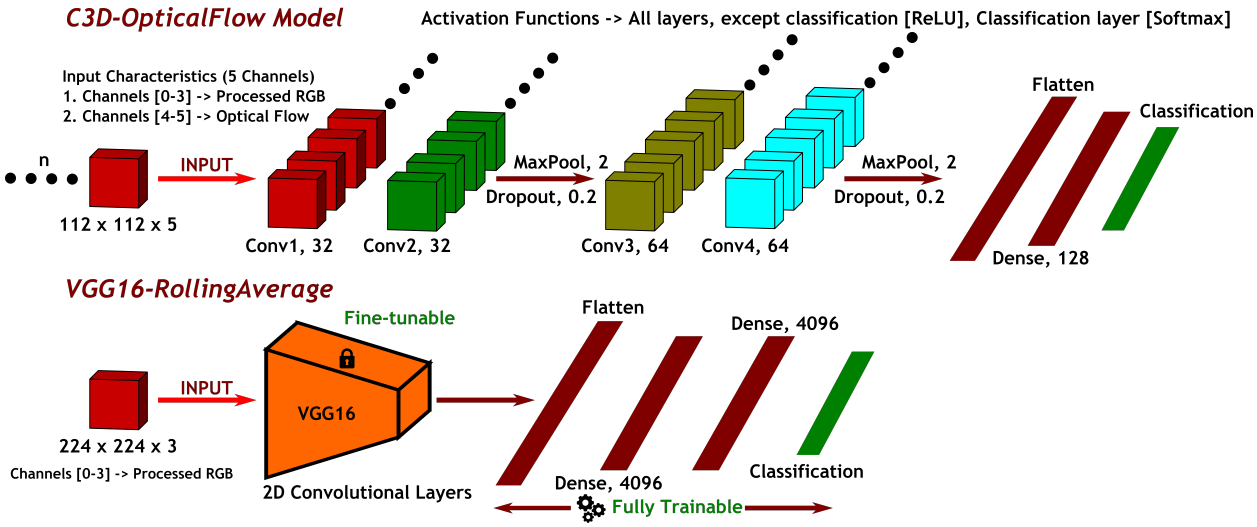


Figure 4.3: Architectures of C3D-OpticalFlow and VGG16-RollingAverage models [70].

The second model used in this study was C3D-OpticalFlow. Unlike the VGG16-RollingAverage model, which used pre-trained weights, this model was trained from scratch. In this model, the temporal information was incorporated right at the time of model training by applying 3D convolution to 4D input. The input to the model after preprocessing and computing optical flow had a shape of $n \times 112 \times 112 \times 5$. The camera capturing the assembly station was assumed to be fixed, and the assembly station was assumed to be illuminated with minimal variation. In these situations, optical flow helps better understand the motion between the frames, as the key assumption for optical flow computation was that the brightness of an image point remains constant over time. The method used in optical flow computation for this study was Gunner-Farneback optical flow, also called the Dense Optical Flow, see Equation (4.1). The optical flow was computed after con-

verting the RGB image frames to Grayscale and was then reshaped to 112×112. Before combining the optical flow information with the RGB frames information, the magnitudes and angles of the optical flow vectors (u, v) were computed.

$$\frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \frac{\partial I}{\partial t} = 0 \tag{4.1}$$

where $u = \frac{dx}{dt}$ and $v = \frac{dy}{dt}$, are the optical flow. $\frac{\partial I}{\partial x}$, $\frac{\partial I}{\partial y}$ and $\frac{\partial I}{\partial t}$ are a gradient of image intensities across the horizontal axis, vertical axis and time, respectively.

Several deep learning architectures can be potentially used for action recognition. VGG16 architecture has been shown to perform well in the image classification domain [40], and the 3D Convolutional Networks were shown to effectively learn from the temporal stream [95]. Additionally, it was also shown that the optical flow information can potentially improve the action detection/recognition ability of the deep learning models [91]. The two models developed in this study had their own set of pros and cons. The VGG16-RollingAverage model had the shortest training time, as transfer learning was applied, however, since the temporal information was aggregated post the model inference by accumulating the softmax probabilities, the layer weights for the VGG16 architecture would not be able to represent the temporal features in the video. Finally, the VGG16-RollingAverage model was pretrained, hence it brings in the knowledge (trained weights) from the ImageNet dataset. In the case of C3D-OpticalFlow, the model was trained from the ground up, hence, it requires a relatively longer training time than the VGG16-RollingAverage. But it tends to use temporal information at the time of training making the model relatively robust. The integration of optical flow with the temporal information from the past n frames proved beneficial in reliably detecting the actions from video data.

4.3.3 Algorithm Architecture

The process of action detection and localization was done in two stages: The inference machine and the state machine. The inference machine detects the actions at every video frame, and the state machine localizes the detected actions at the temporal level. Additionally, the state machine manages the global state of the inference module and identifies the anomalies in the assembly operations. The real-time inference process requires the inference module to make inferences at a rate equal to the camera's frames-per-second (FPS).

The real-time inference architecture enables the measurement of cycle time and step time and the detection of anomalies in an assembly operation. The cycle time of an assembly operation helps in production planning and determining the product's time to market. Several efforts have been made to reduce the cycle time in manufacturing industries, the challenges in the majority of the cases are twofold: (i) Obtaining reliable step-level information to understand the processes within a cycle better. (ii) Continuous monitoring of the assembly operation to conduct time and motion study. The real-time inference architecture enables determining the step time by continuously monitoring the assembly cycle as it is being performed. In addition to accurately determining the step time of the assembly operation, it was equally important to localize the detected assembly steps. The step time for a step in an assembly cycle is the total time required to complete the step, the step time value by itself does not provide any information on when the assembly step started or ended. The process of determining the start time and end time for an assembly step was called the localization of assembly steps within an assembly cycle.

The anomalies in an assembly operation considered for this study were sequence breaks and missed steps. The sequence break corresponds to the break in the sequence of an assembly operation, and missed step(s) corresponds to assembly step(s) that was(were) missed at the end of an assembly cycle. Through real-time inference, we aim to detect the sequence breaks and missed steps the instant they happen.

Inference Machine

The inference machine encompasses the deep learning model used to detect actions from video frames. For every RGB frame sent to the inference machine, the data were preprocessed and added to two different buffers following the First In First Out (FIFO) strategy. The size of the buffers depends on the temporal length n. The data preprocessing consists of the normalization of the RGB frame and optical flow computation. One buffer stacks the processed RGB frame information, and the other stacks the optical flow magnitudes and angles. At every video frame, the data from the buffers were sent to the deep learning model to get an inference. The inference from the model was then added to another FIFO buffer called the inference aggregation buffer. The size of the inference aggregation buffer depends on the inference length l. The temporal length n determines the amount of information in the past that the model looks at when making an inference, whereas, the inference aggregation buffer with inference length l enables the majority voting of a set number of inferences to improve robustness. The final inference made by the inference machine for a video frame was determined by taking a majority vote on the inference aggregation buffer. Hence, the final inference output by the inference machine may not necessarily match the model's inference at any instant. This approach was followed to improve the robustness of the inference machine and prevent jumping of inferences between subsequent frames. The inference machine can be seen in Figure 4.4. Increasing the length of the inference aggregation buffer tends to create more robust inference, but on the other hand, it can delay state changes i.e., the step change. This methodology adopted for the inference machine facilitates effortless modifications to the deep learning models, as the models can be modified using a plug-and-play approach.

State Machine

The state machine is responsible for localizing the actions detected by the inference machine. It provides a way to handle the model's inference by continuously tracking and recording the system's state in real time. In a typical assembly operation, it is crucial to identify the assembly steps that are completed and missed at the end of an assembly cycle. It helps in determining the quality of an assembly cycle and, thereby the end-product quality. The information on the assembly's sequence

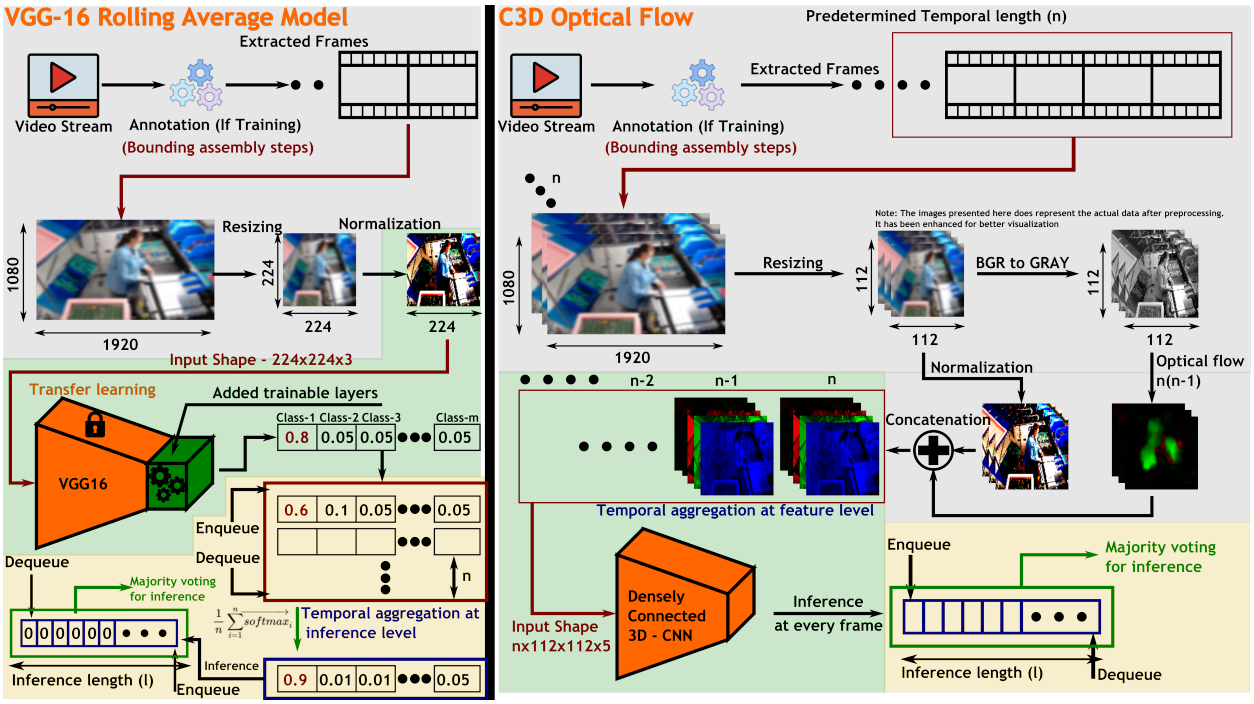


Figure 4.4: Inference Machine using either C3D-OpticalFlow or VGG16-RollingAverage model [70].

of operations and the mandatory steps are available at the time of inference. It was up to the state machine to determine if the requirements were met before an assembly cycle was completed.

State machines are abstract machines that can exist only in one state at any time during the system's operation. Each state in a state machine has a set of outputs that specify the tasks to be performed by that state. To enter into a state from other states, firstly, there should be a path leading into that state, secondly, the state transition criteria associated with that path should be met. The state transition criteria is a set of logic that govern the change in the system's state, without satisfying the transition logic, the system cannot change its current state. An analogy can be created between an assembly cycle and the state machine. Each step of an assembly cycle can be considered to be a state. A state transition is analogous to transitioning from one assembly step to the other.

The future assembly steps in an assembly cycle can be reached only when the past steps are completed, thereby defining the path leading into and out of an assembly step. If any of the assembly steps were to be reached without following the predefined state transition path then it contributes to sequence breaks in the assembly cycle. Similarly, if all states of the state machine were not reached

Algorithm 1 Condition for State Transition and Sequence Break detection [70].

$current\_state = s$, where $s \in \{0, 1, 2, ..., n\}$
▷ System's current state

$input\_states \subset \{0, 1, 2, ..., n\}$, when
$current\_state = s$
▷ Obtained from the state dependencies matrix

$past\_state \in \{0, 1, 2, ..., n\}$
▷ The system's state before current state

$unreached_states \subset \{0, 1, 2, ..., n\}$
▷ States not visited for this cycle

$d1$, time(s), ▷ hyperparameter

$time\_in\_state(s)$, where $s \in \{0, 1, 2, ..., n\}$
▷ Time elapsed when SM was in state s

if $time\_in\_state(s) >= d1$ **then**

$state\_change \leftarrow True$
$unreached\_state \setminus current\_state$

if $past\_state \notin input\_states$ **then**

$sequence\_break \leftarrow True$

end if

end if

at least once during an assembly cycle then there are missing steps in the assembly cycle. Finally, each step of an assembly cycle is governed by a set of actions unique to that step, satisfying the fundamental assumption behind the abstract state machine. Hence, state machines complement assembly operations, and using them will be beneficial in monitoring the assembly operations autonomously using deep learning models. The state machines also enable easier detection of assembly operation anomalies as they can better track the global state of the system at any point in time.

The high-level architecture of the state machine for real-time inference is seen in Figure 4.5.

Algorithm 2 Condition for Cycle Reset and Missed Steps [70].

$current\_state = s$, where $s \in \{0, 1, 2, ..., n\}$
▷ System's current state

$input\_states \subset \{0, 1, 2, ..., n\}$, when
$current\_state = s$
▷ Obtained from the state dependencies matrix

$past\_state \in \{0, 1, 2, ..., n\}$
▷ The system's state before current state

$unreached\_states \subset \{0, 1, 2, ..., n\}$
▷ States not visited for this cycle

$d2$, time(s), ▷ hyperparameter

$time\_in\_state(s)$, where $s \in \{0, 1, 2, ..., n\}$
▷ Time elapsed when SM was in state s

if $current\_state = 0$ &
$past\_state \in input\_states$ &
$time\_in\_state(s) >= d2$ **then**

    $cycle\_reset\_condition \leftarrow True$

    if $unreached\_states \neq \emptyset$ **then**

        $missed\_steps \leftarrow True$

    end if

end if

The architecture shows the interaction between the inference machine and the state machine. The state machine was decoupled from the inference machine. This modular approach enables quick swapping of the deep learning model within the inference machine without affecting the state machine's operation. During the initialization process, the default state or the first state of the state machine needs to be identified. The first state of the state machine can be one of the following states: (i) The first step of the assembly operation, (ii) The miscellaneous step, non-value added step, of the assembly operation, or (iii) Lastly, the very first inference from the inference machine. For the real-time inference process, it was challenging to determine the initial state of the

assembly cycle, hence, the first state was set to the first inference made by the inference machine. During the initialization process, the state dependency matrix needs to be determined. The state dependencies matrix governs the relationship between the steps for an assembly operation. It provides information on what steps need to be completed before starting a specific step in an assembly cycle. Additionally, the state dependencies matrix enables the identification of sequence breaks and missing steps in an assembly operation. Hence, the state dependencies matrix plays a key role in the assembly operation action-localization process and needs to be carefully designed to reflect the properties of the assembly cycle for the inference module to operate effectively.

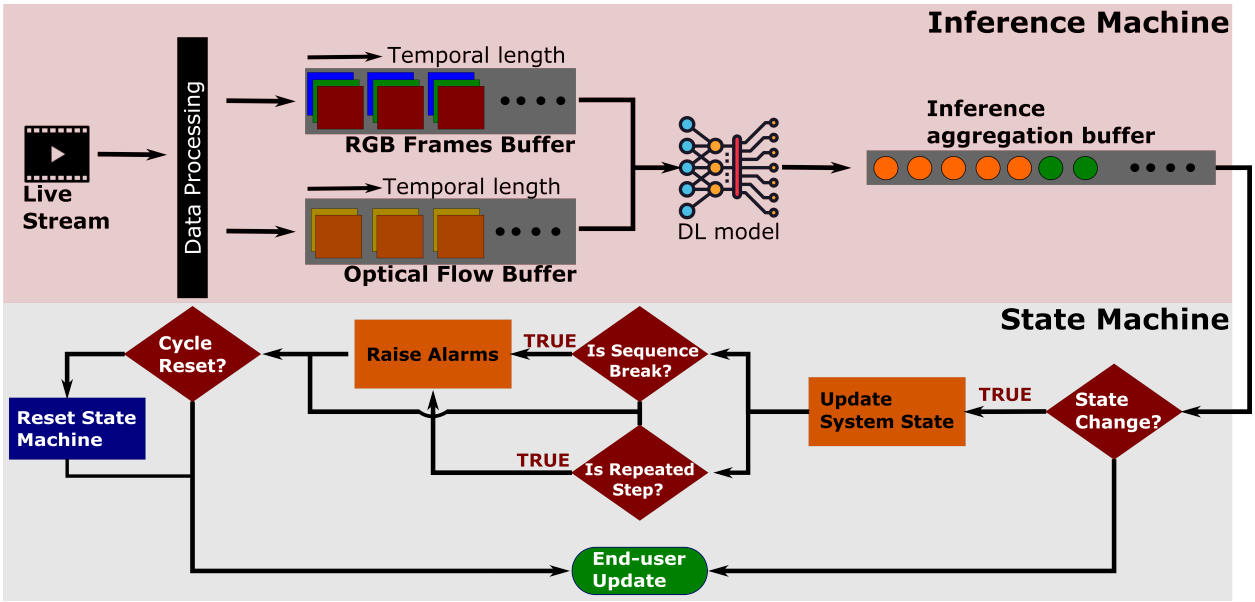


Figure 4.5: The Inference Module comprising of Inference Machine and State Machine [70].

The inference made by the inference machine, i.e., the inference obtained from the majority voting of the inference aggregation buffer, was sent to the state machine for every input video frame. Once the inference is received, the state machine compares the inference with the current state of the system. If the inferences match, the system is in the same state as the inference, and no updates are done to the state machine. If the inference and system state do not match, the system enters into the state transition phase where certain criteria are evaluated to validate the state change. The algorithm corresponding to the state transition can be seen in Algorithm-1. During the state transition process, the state machine also validates the possibility of sequence breaks and missed steps. The sequence break was identified using the state dependency matrix

by considering the current state of the system and the future state based on the inference. If the future state cannot be reached from the current state, the system raises a sequence break flag to alert the assembly operator.

The state machine has timer blocks that determine the time the state machine was in a particular state. The state machine also determines the NVA time, the time when the operator was not performing any one of the predetermined assembly steps for the assembly operation. In addition to the timer blocks, several other metrics such as the order in which the states were visited, repeated assembly steps, and assembly cycle time were determined. After an assembly cycle completion, the state machine would be reset to enable the tracking of the next assembly cycle autonomously. The algorithm for assembly cycle reset can be seen in Algorithm-2.

The inference module was designed to enable the generalization of action-localization in assembly operations. By updating the state dependencies matrix, it was possible to apply the inference module to any assembly operation, provided there was a deep learning model to make an inference on the video frames. Currently, the algorithms corresponding to state transition, cycle reset, sequence breaks, and missed steps identification are designed not to receive any physical input but can be customized to fit the needs of the assembly operations. For instance, the state transitions could integrate the bar code scanner to indicate the transition from one state to the other. This approach can make the state transitions robust, and the feedback loop created can be used to improve the model's performance by limiting the set of possible inferences to a subset. Finally, the inference module was designed by following a modular approach, where the inference machine and state machine were separated. This approach enables effortless updates to the deep learning models, either in terms of improvements to the model architecture or updates to the model weights over time to accommodate for changes in the workstation.

4.3.4 Engineering Application

This section evaluates SMIRL against two datasets acquired from an industry. The inference module was evaluated by comparing the actual and inferred step time of assembly operations, and the Intersection over Union (IoU) metric, followed by its ability to detect anomalies.

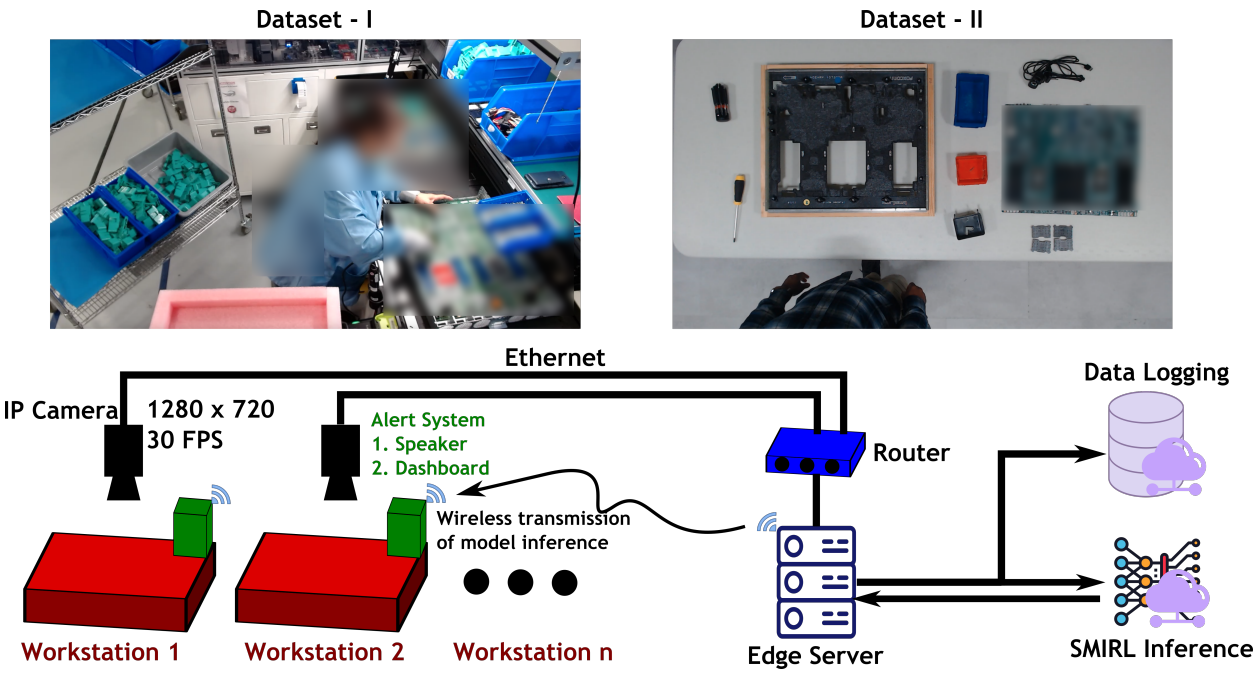


Figure 4.6: Experiment setup and data used for assembly monitoring [70].

4.3.5 Experiment Setup

The experimental setup involves monitoring the assemblies using overhead cameras. Two assemblies were monitored for this study. Out of the two assemblies monitored, one was a Demo assembly (Dataset-II), created to evaluate the developed approach, while the other was a real assembly (Dataset-I) used for assembling data servers. The monitoring process involves setting up an RGB Internet Protocol (IP) camera overlooking the assembly workstation. The real assembly was continuously monitored for action and the data were streamed to local storage over Ethernet. For model development, the data from the storage were used, and for real-time inference, the specific camera was queried using its IP address, and the inference was made at the edge or in the cloud. The experimental setup for data acquisition and real-time inference can be seen in Figure 4.6. This approach simplified our data collection and inference process. Throughout this work, the programming language used was Python, and the model development framework was PyTorch.

Table 4.1: Assembly steps for the operation involved in the datasets [70].

Dataset-I	Dataset-II
Labelling-I	Position Motherboard
Labelling-II	Attach Bracket
Position Motherboard	Secure Motherboard
Scan Motherboard and Label	Insert Card
Insert PCIe Fillers	Attach Device
Insert CSSD Card	Remove Battery
Push-Secure Motherboard	Miscellaneous
Route Cabling	-
Get next Chassis	-
Miscellaneous	-

Table 4.2: Dataset Characteristics [70].

Assembly	Training	Evaluation	Average Cycle Time [1]
Dataset-I	25	280	$111.9s \pm 24.7s$
Dataset-II	13	138	$107.5s \pm 7.9s$

[1] Average of the human-annotated cycle time.

Dataset Configuration

Two different datasets were used in this study to test the inference state machine. The first dataset, with the nomenclature Dataset-I, involves the assembly of a server and constitutes a total of 9 steps. The detailed information on the assembly steps was redacted to maintain confidentiality. The data were collected by continuously monitoring the assembly station as the assembly operations were performed using installed cameras. The camera used has a resolution of 1280 x 720 with a frames-per-second (FPS) of 30. The second dataset, with the nomenclature Dataset-II, involves a custom-designed assembly operation to test the generalization ability of SMIRL. Table 4.1 shows the fine-grained assembly steps for both assembly operations. The data were collected by recording the assembly workstation using overhead cameras as the assembly cycles were performed. The number of assembly cycles used for model training and evaluation can be seen in Table 4.2. In both cases, data were stored in a remote server for model development and were streamed at an endpoint for the implementation of real-time inference. The dataset used in this study was not published because the assembly workstations and the components assembled were confidential. Figure 4.6 shows a snapshot of the assembly workstation.

Data Preprocessing

The data preprocessing step depends on the model used in the inference machine. As discussed in Section 4.3.2, the models developed in this study have their own data preprocessing steps and it depends on how the temporal information was utilized. Since inference was to be made for every video frame, the video was annotated manually by assigning labels to the video segments corresponding to the assembly steps. The segment-level labeling for the videos was then passed down to the frame-level labeling. The annotated frames were then extracted and associated with the respective assembly steps. The assembly steps, Table 4.1, correspond to classes in the classification models.

In the case of the VGG16-RollingAverage, the input data to the model follows the same shape as the original VGG16 architecture, individual frames of the video were resized and normalized before the input layer of the model. The temporal information was incorporated into the model during inference, where the softmax probabilities were added to a buffer of length n. The inference for the VGG16-RollingAverage model was made by averaging the softmax probabilities in the buffer. For the C3D-OpticalFlow model, to enhance the feature learning process of the deep learning models, the video data were processed in two ways. Firstly, the video frame was resized to $112 \times 112 \times 3$, followed by a normalization. Secondly, the optical flow between the subsequent frames of the video was extracted from the grayscaled and resized video frames. Hence, for each frame of the video, the optical flow was computed for a single past frame, leading to an $n(n-1)$ optical flow. The optical flow computation provides the (u, v) vectors for each pixel, using which the magnitude and angle at each pixel were computed, leading to a shape of $112 \times 112 \times 2$. The resized RGB frames and the optical flow were concatenated to obtain an input data shape of $112 \times 112 \times 5$. The temporal information for the C3D-OpticalFlow model was integrated into the model training process by considering the past n frames as a part of the input. The final shape of the input data after data preprocessing was $n \times 112 \times 112 \times 5$, where the n corresponds to the temporal length and was set to 16 for this study. On a high level, the model makes an inference by considering the past 16 frames of RGB and optical flow information instead of just one frame as can be seen in Table 4.3.

Table 4.3: The data input shape for the models [70].

Model Name	Input Shape
VGG16-RollingAverage	$224 \times 224 \times 3$
C3D-OpticalFlow	$n \times 112 \times 112 \times 5$

Model Training

The models developed in this study were trained on both datasets. For Dataset-I, the models were trained on 17 cycles, whereas, for Dataset-II, they were trained on 13 cycles. The same assembly cycles were used across both models to ensure consistency. For C3D-OpticalFlow, the model was trained from the ground up, whereas transfer learning was used to train the VGG16-RollingAverage. In both cases, the models were trained/fine-tuned using the Backpropagation algorithm, [109].

The VGG16 model with the weights previously trained on the ImageNet dataset was used to initialize the training process of the VGG16-RollingAverage model. The modifications involved removing the classification layer of the original VGG16 model followed by adding a few dense layers and a new classification layer corresponding to the number of assembly steps. The number of dense layers that were added was a hyperparameter which was determined to be three using a grid-searching approach. Except for the newly added layers in the VGG16-RollingAverage model, all the other layers were set to be fine-tunable during the model training process. The inference was made for every video frame and the softmax probabilities were added to the buffer. The buffer follows a FIFO strategy, and the final softmax probabilities were computed by averaging the elements in the buffer. The size of the buffer was set to be n, corresponding to the temporal length of the inference module. In VGG16-RollingAverage, the temporal information was utilized during the time of inference, by computing the rolling average of the buffer of softmax probabilities. Hence, the temporal relevance between the frames was not learned during the model training process. In the case of the C3D-OpticalFlow model, the temporal information was presented to the model during the training process. The model input involves a circular buffer of n temporal frames that follows the FIFO strategy during training and inference. Using Gunner-Farneback's approach, the dense optical flow was computed between every subsequent frame over the predetermined temporal length n. RGB frames and optical flow were both concatenated and used in model training. The

temporal length n was another hyperparameter that controls the amount of frames in the past that the model needs to look at to make an inference. The higher the value of n, the better the performance of the models as they have access to a larger window of data, but this leads to an increase in computation and the resolution of step time and cycle time measurements. Hence, considering the computational requirements and step time resolution we set the temporal length n to be 16.

During the model training, the annotation at the video level was propagated to the frame level. The frames corresponding to the NVA activities of an assembly operation were categorized into a "Miscellaneous" class. In cases where the NVA activities can be identified, they were explicitly labeled to be used in the model training process. The models were trained for 500 epochs with early stopping. The loss function used in the model training process was categorical cross entropy, seen in Equation 4.2, and it contains both L1 and L2 regularization. The learning rate used in training was adaptive and updated if there was no improvement to the validation loss over a certain number of epochs. The number of epochs for adaptive learning rate update was a hyperparameter and it was set to 10. The validation loss was monitored, and the model weights were saved when the validation loss decreased between successive Epochs. Additionally, in a typical assembly operation, the assembly steps could have different step times. This led to an imbalance during the model training process. Hence, class weights were incorporated into the model training to reduce the overfitting of the majority class.

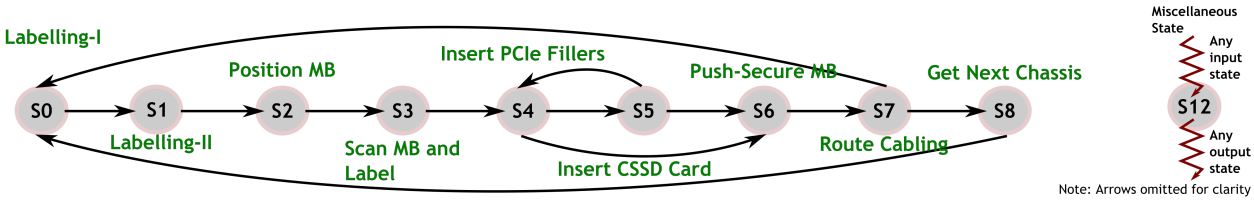
$$\tilde{J}(w; X, y) = J_{ce}(w; X, y) + \frac{\alpha_1}{2} w^T w + \alpha_2 ||w||_1 \tag{4.2a}$$

$$J_{ce}(w; X, y) = -\frac{1}{N} \sum_{i=1}^{N} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)] \tag{4.2b}$$

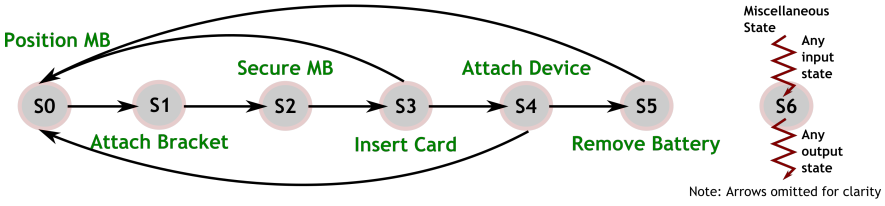
where α_1 and α_2 control the weights of each regularization term while minimizing the cost function. They were set to 0.001 and 0.001, respectively.

State Machine Integration

The detection of actions and their localization was done in two stages: the inference machine, responsible for action detection, and the state machine, responsible for localization, see Figure 4.5. In the inference machine, the inferences made by the deep learning model were added to the inference aggregation buffer of size 30. The buffer followed a FIFO strategy, and the final inference was determined using majority voting. This concluded the action detection process, and the detected action was sent to the state machine. The initialization process for the state machine required a state dependency matrix, which encoded the information on the assembly operation monitored. The state dependency matrix contained information on the sequence of the operations performed for the assembly. Additionally, the sequence-of-operation information was exploited to detect assembly operation anomalies. In our case study, a state dependency matrix was created for each assembly operation, seen in Figure 4.7.



(a) Abstract state machine for the assembly operation in Dataset-I



(b) Abstract state machine for the assembly operation in Dataset-II

Figure 4.7: State dependencies diagram for the two assembly operations considered in this study [70].

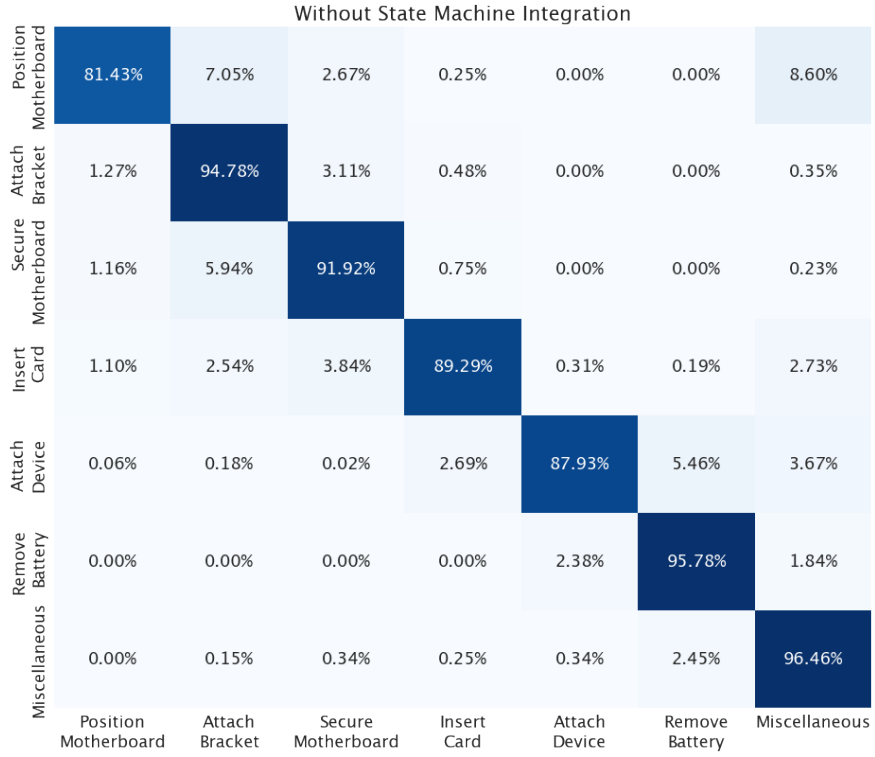
The state dependency matrix defines the sequence of operations performed in an assembly operation as can be seen in Figure 4.7. In the case of Dataset-I, step-4 (S4) and step-5 (S5) were interchangeable, meaning the assembly operator could swap between S4 and S5, and it was up to their own volition. The state dependency matrix enables the integration of such information into the inference module without any changes to the algorithm itself. From the state transition

diagrams for both datasets, the steps toward the end of the assembly cycle lead back into step 0 (S0). These transitions indicate the ability of the state machine to reset if the system were to be in one of those states. In Dataset-I, the first six steps of the assembly operation were mandatory as they involved the component being moved into the assembly workstation. The assembly cycle cannot be completed without the operator performing these steps. On the other hand, the remaining steps can be skipped and the operator can still complete the assembly cycle. Hence, the system can be reset from these steps and not from the first six steps of the assembly cycle. A similar case applies to Dataset-II as well. Sequence break identification happens during the state transition by checking if the past state was part of the input states of the current state. The past state here corresponds to the state the system was in before the transition. The missing states on the other hand were identified by determining the states the system has not visited at the time of the cycle reset. Finally, for every state machine created for an assembly operation, a miscellaneous state was added by default. The miscellaneous state corresponds to all possible actions that do not fall under any of the assembly steps. The miscellaneous state can be identified by explicitly training the inference machine on unwanted actions or by identifying the Out-of-distribution (OOD) instances. The time incurred when the system was in a miscellaneous state was considered to be non-value-added time.

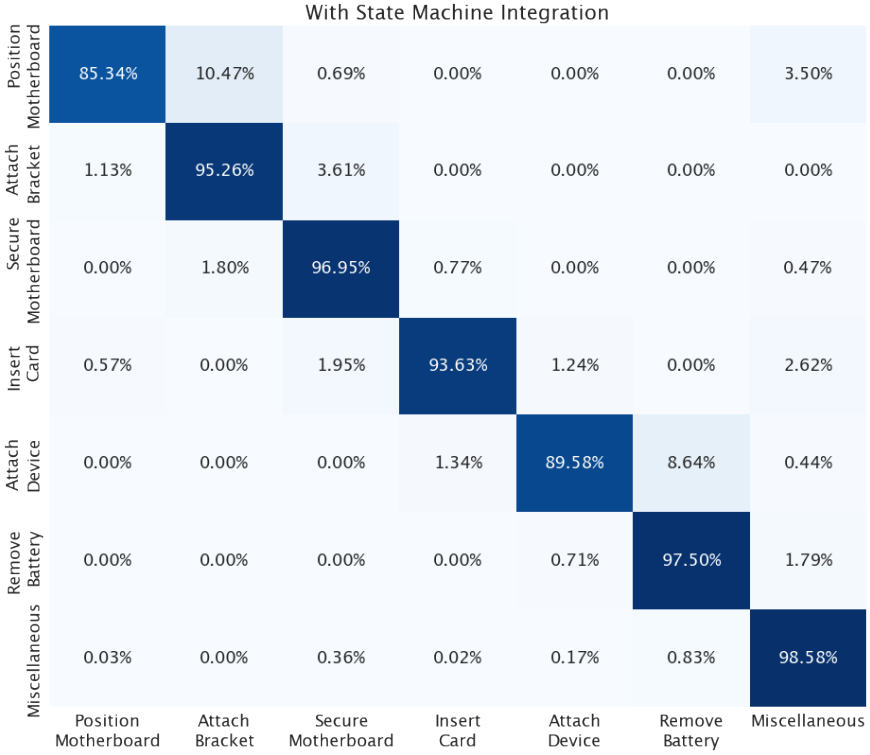
At the end of every assembly cycle, a series of summary statistics were computed and stored in a database for future reference. The sequence break alerts were provided to the assembly operator in real-time, and the missing steps were identified at the end of the assembly cycle. When corrective actions were performed, the time corresponding to those steps was adjusted accordingly. Finally, on cycle reset, the system counters were reset to begin the next cycle. This enables continuous monitoring of the assembly workstation with no human intervention.

4.3.6 SMIRL Evaluation

In this section, we discuss the evaluation of SMIRL on the two assembly operations. We start by determining the impact of integrating the inference machine and state machine thereby forming the inference module. We then demonstrate the ability of the state machines to detect sequence breaks and missed steps of the two assembly operations considered in the study.



(a) Model's direct inferences.



(b) State machine integrated inferences

Figure 4.8: Confusion matrices after converting time-level inferences to frame-level inferences [70].

Table 4.4: F1-Scores for the Dataset-II inferences without and with state machine integration [70].

Assembly Steps	F1-Score	F1-Score (SM)
Position Motherboard	0.87	**0.90**
Attach Bracket	0.92	**0.94**
Secure Motherboard	0.90	**0.95**
Insert Card	0.93	**0.96**
Attach Device	0.91	**0.93**
Remove Battery	0.93	**0.95**
Miscellaneous	0.95	**0.97**

The inference module comprising of the inference machine and state machine can detect sequence breaks, missed steps, and cycle resets, additionally, it was found to improve the performance of the whole system by accurately classifying the assembly steps, as seen in Figure 4.8. The inference module was designed to output the time incurred for the step detected, we call it the time-level inference. To enable the evaluation process using confusion matrices, we converted the time information at each assembly step into frames with labels, we call this frame-level inference. The frame-level inference was compared against the human-annotated frame-level information to create the confusion matrices. The confusion matrix in Figure 4.8b is the evaluation of the model without the integration of the state machine, i.e., the output from the inference machine was considered to be the final output. Whereas, in Figure 4.8a, the evaluation was performed with the integration of the state machine, i.e., the output from the inference machine is piped into the state machine before being considered as the final output. It can be seen that the integration of the state machine helps in improving the system's performance across the board, i.e., in all steps of the assembly cycle. The reason for the performance improvement can be attributed to the fact that the state machine integration enables reliable state transfers from one assembly step to the other. The F1-Score corresponding to both the methods of inference can be seen in Table 4.4. Similar to the confusion matrices, the F1 scores were improved across all the steps of the assembly cycle. The model used in the inference machine to create the confusion matrices in Figure 4.8 was C3D-OpticalFlow. The above evaluation aimed to determine the impact of integrating the state machine with the inference machine. Since C3D-OpticalFlow was the best-performing model when compared to VGG16-RollingAverage, the analysis using the model VGG16-RollingAverage in the inference machine was not presented.

Table 4.5: Normalized Mean Absolute Error for the model C3D-OpticalFlow [70].

Dataset-I	NMAE	Dataset-II	NMAE
Labelling-I	0.17	Position Motherboard	0.07
Labelling-II	0.25	Attach Bracket	0.12
Position Motherboard	0.31	Secure Motherboard	**0.17**
Scan Motherboard and Label	0.42	Insert Card	0.07
Insert PCIe Fillers	0.38	Attach Device	0.1
Insert CSSD Card	0.38	Remove Battery	0.1
Push-Secure Motherboard	0.22	Miscellaneous	0.04
Route Cabling	0.14	-	-
Get next Chassis	**0.47**	-	-
Miscellaneous	0.43	-	-

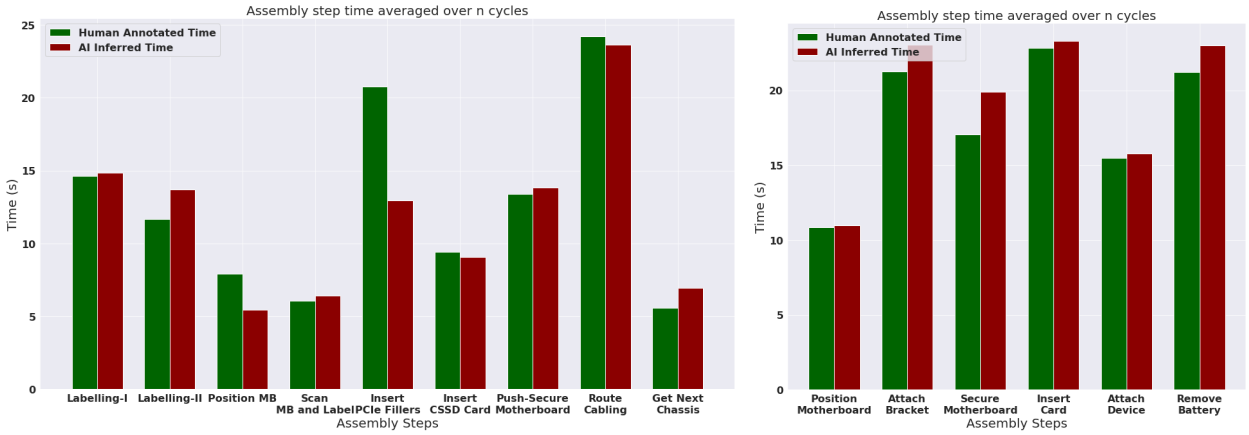
Table 4.6: IoU by assembly steps for Dataset-II [70].

Dataset-II	IoU
Position Motherboard	0.8427
Attach Bracket	0.8959
Secure Motherboard	0.9055
Insert Card	0.9146
Attach Device	0.8336
Remove Battery	0.8596

The inference module enables the determination of the step time and the cycle time of an assembly operation, hence, it is crucial to test the system's performance by comparing the SMIRL inferred time at the step level with the human-annotated step time. In Figure 4.9, the inferred and the actual step times were plotted for both datasets. For both datasets, the Normalized Mean Absolute Error (NMAE) is given by Equation 4.3 and can be seen in Table 4.5. The max NMAEs were 0.47 and 0.17 for the steps "Get Next Chassis" and "Secure Motherboard" for Dataset-I and Dataset-II, respectively.

$$NMAE = \frac{abs(Inferred_{st_i} - Actual_{st_i})}{\mu_{st_i}} \tag{4.3}$$

where st_i is the step time for step-i, and μ_{st_i} corresponds to the mean of human-annotated step time for step-i.



(a) Step Time for Dataset-I.

(b) Step Time for Dataset-II.

Figure 4.9: Inferred and Actual step time for Dataset-I and Dataset-II using C3D-OpticalFlow model [70].

By localizing the assembly steps we can determine their start and end points within an assembly cycle. Hence, to better understand how accurately the developed system was able to localize the detected assembly steps, the IoU metric was used. The definition of IoU can be seen in Equation 4.4 and was primarily used in object detection studies to compute the overlap between the inferred and actual bounding boxes. In our case, the IoU was performed by measuring the overlap between the inferred and actual step time in one dimension using the start time and end time of the assembly steps. The IoU determination process can be seen in Figure 4.10. IoU was computed for each step of the Dataset-II and can be seen in Table 4.6. The Miscellaneous category was ignored in the IoU computation as they can happen at any point during the inference process which was generally ignored during the human labeling of the videos. A total of 170 assembly cycles were used to determine the IoU for each assembly step and the IoU value averaged across all the cycles was 0.8753 ± 0.1282. The average overlap for the actual and inferred assembly step time was 87%, exhibiting an ability that SMIRL can reliably localize the detected assembly steps in real time.

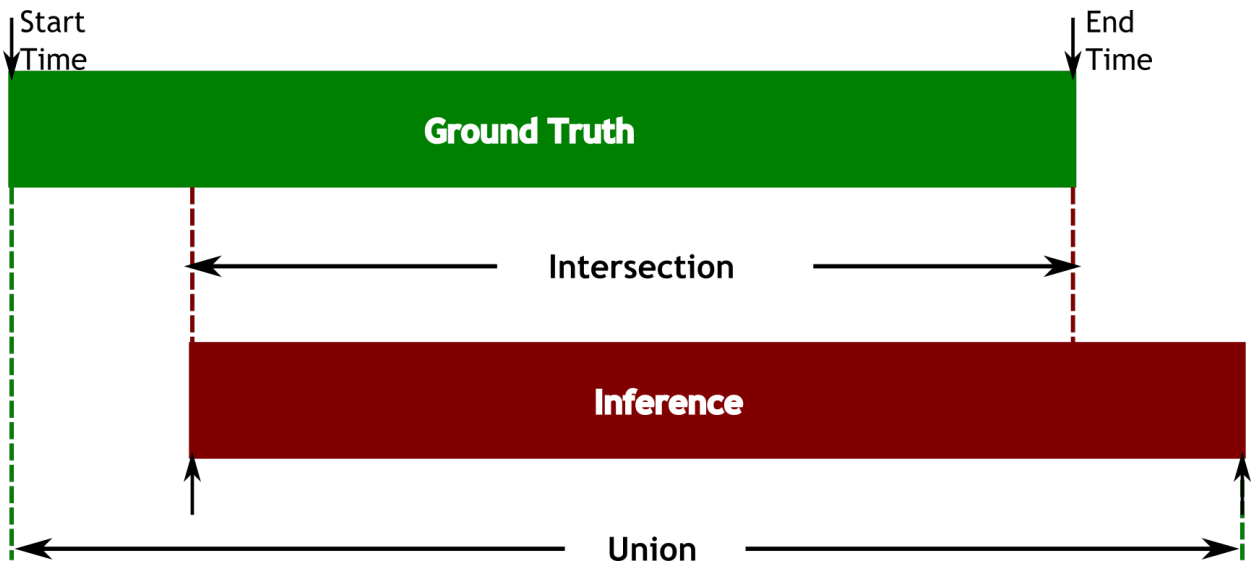


Figure 4.10: Intersection over Union in 1D [70].

$$IoU = \frac{Intersection}{Union} \tag{4.4}$$

where,

$$Intersection = min(actual_{end}, inferred_{end})$$
$$-max(actual_{start}, inference_{start})$$
$$Union = actual_{elapsed} + inferred_{elapsed}$$
$$-Intersection$$

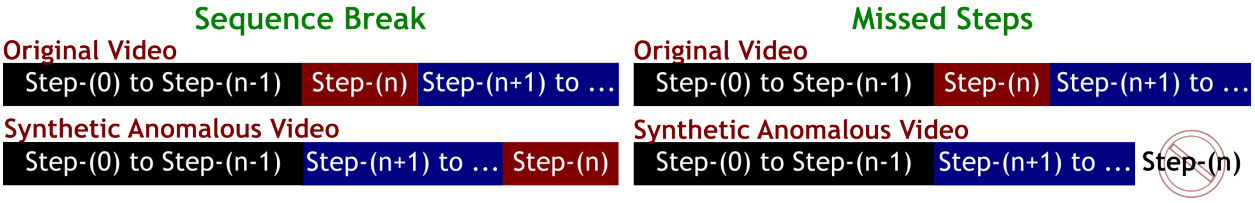


Figure 4.11: Anomalous Data Generation [70].

With being able to determine and localize the detection of assembly steps accurately, the ability of the system to detect the Sequence breaks and identify missed steps was then evaluated. The proportion of anomalies in both datasets was very low to evaluate the anomaly detection ability of SMIRL reasonably. Hence, anomalies were synthetically created by artificially snipping and

Table 4.7: Evaluation of anomaly detection ability for the two models [70].

		C3D-OpticalFlow		**VGG16-RollingAverage**	
Assembly	Anomaly	Cycles	F1-Score	Cycles	F1-Score
Dataset-I	Sequence Break	137	0.8188	137	0.7241
	Missed Step	151	0.8496	151	0.7432
Dataset-II	Sequence Break	173	0.8664	173	0.8835
	Missed Step	174	0.8745	173	0.8780

concatenating the video data associated with different assembly steps. The process of anomalous data generation can be seen in Figure 4.11. For the case of sequence breaks, the sequence of the steps involved was altered using the human-annotated time information. In the case of missed steps, steps were cropped out of the videos. Only one step was altered within an assembly cycle to better visualize and analyze the synthetic dataset. The approach for evaluating the ability of the system to detect anomalies can be seen in Algorithm 3. The number of generated cycles and the F1-Score for the anomaly detection process can be seen in Table 4.7. Through SMIRL the assembly operation's anomalies can be detected reliably in real-time, which was otherwise not possible. Additionally, by defining the state dependencies appropriately, we can easily define what constitutes an anomaly for any assembly operation. For instance, in the case of Dataset-II, the steps "Insert PCIe Fillers" and "Insert CSSD Card" can be performed interchangeably without the system detecting them as a sequence break.

Algorithm 3 Evaluation of Anomaly Detection [70].

$evaluation\_set \leftarrow \{\}$
$cycles = \{1, 2, 3, ..., n\}$

for cycle in cycles **do**
  if $inferred\_anomaly = actual\_anomaly$ **then**
    $value \leftarrow 1$
  else
    $value \leftarrow 0$
  end if
  $evaluation\_set \ll value$
end for

SMIRL was designed to enable a plug-and-play approach for deep learning models. The inference machine in the inference module can be swapped for model upgrades or replacements. In addition

Table 4.8: Normalized Mean Absolute Error for the model VGG16-RollingAverage [70].

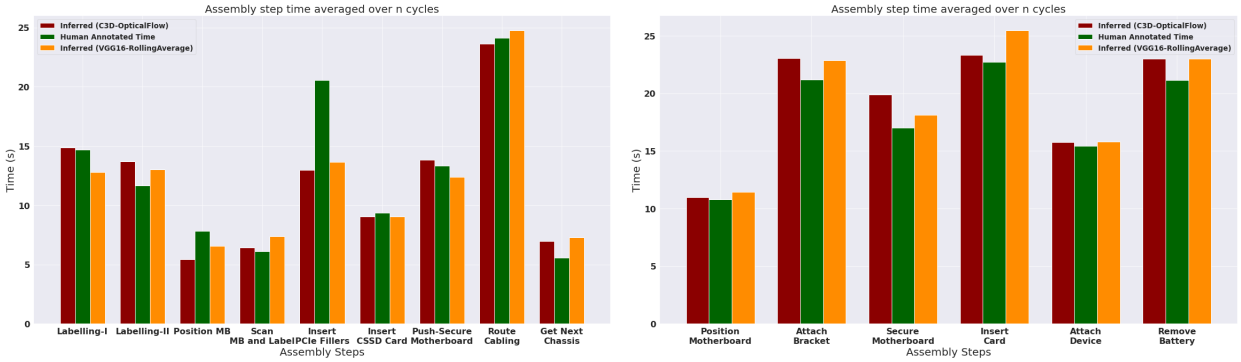
Dataset-I	NMAE	Dataset-II	NMAE
Labelling-I	0.20	Position Motherboard	0.12
Labelling-II	0.29	Attach Bracket	0.09
Position Motherboard	0.41	Secure Motherboard	0.09
Scan Motherboard and Label	**0.72**	Insert Card	**0.13**
Insert PCIe Fillers	0.41	Attach Device	0.05
Insert CSSD Card	0.38	Remove Battery	0.09
Push-Secure Motherboard	0.18	Miscellaneous	0.03
Route Cabling	0.13	-	-
Get next Chassis	**0.72**	-	-
Miscellaneous	0.51	-	-

to C3D-OpticalFlow, VGG16-RollingAverage was trained on the two datasets used in this study. As described in Section 4.3.2, the temporal information was added during the time of inference and not during training for the VGG16-RollingAverage model. Additionally, this model does not incorporate the OpticalFlow information, in contrast to the C3D-OpticalFlow model. The step time determined using the VGG16-RollingAverage for both the datasets and the associated NMAE can be seen in Figure 4.12 and Table 4.8, respectively. The VGG16-RollingAverage model was similar in performance to the C3D-OpticalFlow model for Dataset-I, but the C3D-OpticalFlow outperforms VGG16-RollingAverage for Dataset-II by a small margin.

Similarly, the anomaly detection ability of the VGG16-RollingAverage model was tested against the two datasets under similar conditions as the C3D-OpticalFlow model using Algorithm-3. The results for VGG16-RollingAverage can be seen in Table 4.7. The performance of the two models was similar on Dataset-II but there was a 8% drop in performance when using VGG16-RollingAverage for Dataset-I. Dataset-I was challenging because of the higher proportion of NVA activities present in an assembly cycle compared to Dataset-II, as Dataset-II was generated in a laboratory under a controlled environmental setting. Hence, the improvement in performance for the model C3D-OpticalFlow can be attributed to the integration of optical flow, and the use of temporal information at the time of model training.

The ability of the system to perform in real time was tested by determining the maximum number of frames the inference module could process every second. It was determined to be 87.90±

0.64 and 102.44 ± 1.27 for the model C3D-OpticalFlow and VGG16-RollingAverage, respectively, from 173 assembly cycles in Dataset-II. The inference module comprises of inference machine and a state machine. The major proportion of the time was incurred at the inference machine due to the higher computational requirements of the deep learning model. The inference process was made using a single GPU, NVIDIA RTX 6000, and the state machine was handled using a single CPU core. During the real-time inference process, for every frame of the video, the data were transferred to the GPU memory after preprocessing to make an inference using the deep learning model in the inference machine. After the GPU computation, the inference machine's output was transferred to the host for further processing of the data by the state machine.



(a) Step Time for Dataset-I.

(b) Step Time for Dataset-II.

Figure 4.12: Inferred and actual step time for two assembly datasets between the two models [70].

4.3.7 SMIRL vs Two-Stream Approach

The Two-Stream approach involved capturing the information on actions from still frames and motion between frames as separate spatial and temporal streams [91]. A high-level overview of the two-stream model architecture, adapted for the problem domain studied in our research can be seen in Figure 4.13. When incorporating Two-Stream approach the spatial and temporal information were separated, and separate models were used to process the information. This can be seen from the Figure 4.13. The model used for processing the spatial stream was VGG16 and was first pretrained on the ImageNet and then fine-tuned on the assembly dataset, as recommended in Simonyan and Zisserman [91]. The C3D-OpticalFlow model, discussed in Section 4.3.2, was used to process the temporal stream. The key difference between the two different types of C3D-OpticalFlow was how their input was configured. For the case of the Two-Stream approach, the

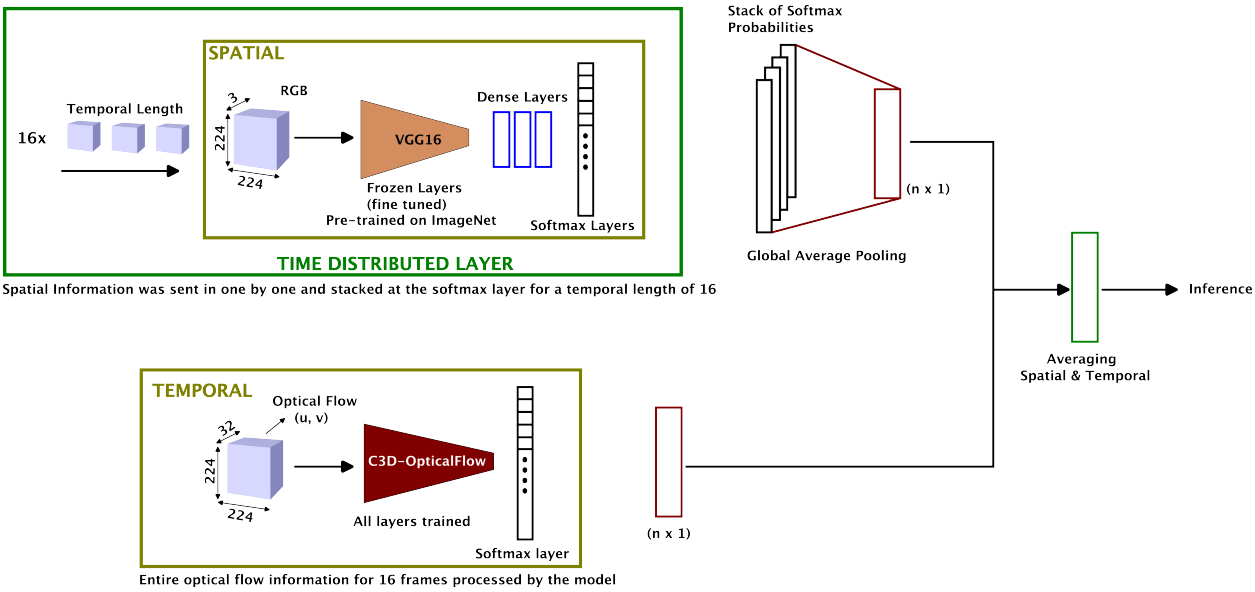


Figure 4.13: Two-stream model architecture adapted for assembly monitoring.

C3D-OpticalFlow processed only the optical flow information, whereas, for the case of SMIRL, the C3D-OpticalFlow processed both the optical flow information and the normalized RGB data from video frames simultaneously. The Two-Stream is not a model but rather a different approach to process the information. In our case, the models used in Two-Stream information processing were the ones developed for SMIRL. At the time of writing this thesis, the two-stream approach outperformed other action recognition approaches [95]. The modular nature of SMIRL, enabled us to augment the inference machine to incorporate the Two-Stream approach of presenting the data. For the case of simplicity, only Dataset-II was used for analysis. In Table 4.9, NMAE was computed between original SMIRL, discussed previously, and Two-Stream Fusion. In Table 4.10, 4.11, and 4.12, the comparison between the two approaches for IoU, sequence breaks, and missed steps were shown. In almost all cases, it can be seen that the Two-Stream fusion approach for inference machine outperformed the C3D-OpticalFlow used in our original SMIRL design by a small margin, but, this comes at a cost of increased computational requirement and longer training and inference times which impacted the real-time inference requirements for SMIRL. For further analysis, C3D-OpticalFlow which processes both optical flow and RGB information simultaneously was selected as the inference machine considering its performance being closer to the Two-Stream fusion approach and has low computational requirement.

Table 4.9: Comparison of NMAE for C3D-OpticalFlow and TwoStreamFusion on Dataset-II

Dataset-II	**NMAE C3D-OpticalFlow**	**NMAE TwoStreamFusion**
Position Motherboard	0.07	0.11
Attach Bracket	0.12	**0.08**
Secure Motherboard	0.17	**0.09**
Insert Card	0.07	0.10
Attach Device	0.10	**0.06**
Remove Battery	0.10	**0.08**
Miscellaneous	0.04	**0.05**

Table 4.10: Comparison of IoU for C3D-OpticalFlow and TwoStreamFusion on Dataset-II.

Dataset-II	**IoU C3D-OpticalFlow**	**IoU TwoStreamFusion**
Position Motherboard	0.8427	**0.8719**
Attach Bracket	0.8959	**0.9354**
Secure Motherboard	**0.9055**	0.9024
Insert Card	0.9146	**0.9296**
Attach Device	0.8336	**0.8980**
Remove Battery	0.8596	**0.9000**
Miscellaneous	-	-

4.3.8 Discussion

Action detection and localization of human operation for manufacturing process optimization help improve operators' performance and reduce costs and production time. In the majority of applications, industrial safety, and operator discomfort do not allow body-worn sensors to detect human activities. In this work, we developed a non-contact approach to detect the actions performed in an assembly workstation. A video camera was installed overlooking the assembly workstations as the assembly operations were performed. The video data were then used in real-time to infer the operations performed by the assembly operator. Through this work, an inference module was developed that integrates an inference machine and a state machine to detect and localize the actions, respectively. The approach of decoupling the action recognition and localization process was novel at the time of this writing, titled SMIRL. The modular approach enables easier updates to the deep learning models in the inference machine, in an effort toward life-long learning, without making any changes to the state machine. In addition to localization, the state machine was designed to identify anomalies like sequence breaks and missed steps in the assembly operation in real-time. The developed approach was tested on two assembly operations to evaluate its performance and

Table 4.11: Comparison of sequence break detection for C3D-OpticalFlow and TwoStreamFusion on Dataset-II.

Dataset-II	**C3D-OpticalFlow**	**TwoStreamFusion**
Precision	0.9821	**1.0**
Recall Score	0.7971	**0.8768**
F1-Score	0.8800	**0.9344**
Accuracy	79.71%	**87.68%**

Table 4.12: Comparison of missed step detection for C3D-OpticalFlow and TwoStreamFusion on Dataset-II.

Metric	**C3D-OpticalFlow**	**TwoStreamFusion**
Precision	0.9821	**1.0**
Recall Score	0.7971	**0.8768**
F1-Score	0.8800	**0.9344**
Accuracy	79.71%	**87.68%**

ability to perform in real-time. The summary of work done in this section is listed below,

1. The step time for each step in both assembly operations was inferred using the developed approach. The maximum NMAE was determined to be 0.47 and 0.17 for Dataset-I and Dataset-II, respectively.

2. The IoU score was computed to determine the overlap between the inferred step time and actual step time for Dataset-II. The average IoU was determined to be 0.8753.

3. The ability of the system to detect the assembly operation's anomalies was tested. The sequence breaks and missed steps were successfully identified with an F1-Score of 0.8188 and 0.8496 for Dataset-I, and 0.8664 and 0.8745 for Dataset-II, respectively.

4. To showcase the ability to swap the deep learning model in the inference machine, the VGG16-RollingAverage model was developed alongside C3D-OpticalFlow. A performance comparison was made between the two models and it was found that the use of optical flow information for action detection was beneficial.

5. Finally, the maximum FPS the system could handle was determined by testing the real-time performance of the inference module. The maximum FPS depends primarily on the complexity of the inference module, as it holds the deep learning model. The model C3D-

OpticalFlow was determined to be 87 frames per second.

Although SMIRL successfully achieved the objectives and was effectively deployed in industrial settings, several underlying challenges may have hindered its widespread adoption. Some of these challenges are listed below:

1. The current inference machine requires training on the NVA activities in assembly operations. This is impractical, as it is not always feasible to identify all possible NVA scenarios within an assembly process.
2. The system cannot integrate end-user feedback to update the inference machine in real-time. Incorporating real-time feedback would create a closed-loop system, enabling the model to stay updated with changes in the assembly workstation, which is common in manufacturing environments.
3. C3D models are computationally intensive, leading to longer training times and requiring high-performance computers for action detection and localization. Alternative approaches that better capture the physics of the assembly operations, while improving computational efficiency, are essential.

4.4 Autonomous Detection of NVA activities

The primary challenge with SMIRL was its inability to detect NVA activities in an assembly operation without explicitly training the deep learning models on them. NVA activities refer to any actions performed at an assembly workstation that are not part of the assembly SOP steps. Training deep learning models on all possible NVA activities is impractical, as these activities can vary widely, making it impossible to identify all potential scenarios. To address this issue, we developed an approach in this study where NVA activities are identified as OOD instances. This method was then evaluated for its accuracy in detecting NVA activities at each step of the assembly process.

The data used in this study were collected by recording several cycles of motherboard assembly.

During these operations, NVA activities were artificially generated by allowing the operator to remain idle or perform tasks unrelated to the assembly SOP steps. The data were recorded from two different operators performing the same assembly steps. In this section, the terms NVA activities and OOD are used interchangeably.

4.4.1 Energy-based Modeling for NVA activities detection

The basis of this approach was to use energy-based modeling to map each point of the input space to a single non-probabilistic scalar called the energy. An energy-bounded learning objective developed by Liu et al. [110] was used to explicitly create an energy gap between the in-distribution and out-of-distribution data. The objective function used in the model training process can be seen in Equation 4.5. The overall training objective is the combination of cross entropy loss and the regularization loss in energy, Equation 4.6.

$$\min_{\theta} E_{(x,y)\sim D_{in}^{train}} [-\log \mathcal{F}_y(x)] + \lambda \cdot \mathcal{L}_{energy} \tag{4.5}$$

$$\begin{aligned} \mathcal{L}_{energy} = E_{(x_{in},y)\sim D_{in}^{train}}(max(0, E(x_{in}) - m_{in}))^2 \quad + \\ E_{x_{nva}\sim D_{nva}^{train}}(\max(0, m_{nva} - E(x_{nva})))^2 \end{aligned} \tag{4.6}$$

where D_{in}^{train} corresponds to the in-distribution training data, corresponding to the assembly SOP steps, and D_{nva}^{train} was the auxiliary data corresponding to the NVA activities. For the first term in L_{energy}, the model penalizes the samples corresponding to the assembly steps that produce energy higher than m_{in}. Similarly, for the second term, the model penalizes the samples corresponding to NVA activities that produce energy lower than m_{nva}. The m_{in} and m_{nva} are hyperparameters and need to be determined experimentally.

After pre-processing, the data for the model training process had the dimension $16\times112\times112\times5$,

where the first dimension was the temporal length, the second and third dimensions referred to the width and height of frames, and the fourth dimension was the concatenation of RGB frames and Optical Flow information. The terms, $E(x_{in})$ and $E(x_{nva})$ represent the expectation along the temporal length dimension. The dimension to computing the expectation was experimentally evaluated, and the temporal length dimension was found to be best performing in terms of creating the energy gap between assembly steps and NVA activities.

$$E(x; f) = -T \cdot \log \sum_{i}^{K} e^{f_i(x)/T} \tag{4.7}$$

During the evaluation, for every input, the energy was computed as seen in Equation 4.7. Then using the predetermined threshold δ, the inputs were classified as assembly SOP steps or NVA activities. In Equation 4.7, T is the temperature scaling factor, $f_i(x)$ are the logits from the last dense layer of the deep learning model, and K is the number of logits in the penultimate layer of the neural network classifier. After computing the energy, the input data were classified between VA (value-added), class-0, and NVA, class-1, using an experimentally determined threshold as shown in Equation 4.8.

$$g(x; \tau, f) = \begin{cases} 0 & \text{if } E(x; f) \leq \tau \\ 1 & \text{if } E(x; f) > \tau \end{cases} \tag{4.8}$$

The notion of using a regularizer, such as L_{energy}, was to push the optimization objective for deep learning models, Equation 4.5, to create a gap in the dimensionless energy, $E(x; f)$, between the in-distribution and NVA data instances.

Table 4.13: NMAE after NVA activities identification [71].

Assembly SOP steps	Without NVA detection	With NVA detection
Position Motherboard	3.05	0.15
Attach Bracket	0.12	0.12
Secure Motherboard	0.20	0.21
Insert Card	0.15	0.15
Attach Device	0.07	0.07
Remove Battery	0.36	0.14
Miscellaneous	-	0.18

4.4.2 Evaluation of NVA activities detection

The deep learning model in the inference machine of SMIRL was trained using 13 assembly cycles, with an average cycle time of $107.5 \pm 7.9s$. Each step of the assembly operation was manually labeled, including the NVA activities in each of the cycles in the training data. The training data constitutes 6 assembly steps, with one more added to represent all NVA activities. Based on the annotation information, the data were categorized into D_{in}^{train} and D_{nva}^{train}. The hyperparameters used in the energy score computation, T, m_{in}, and m_{nva}, were identified using the grid searching approach recommended in [110], and were determined to be 500, -5, and -27, respectively. The trained model was then evaluated on 206 unseen assembly cycles. The distribution of the energy computed across all these cycles for all assembly SOP steps was then plotted in Figure 4.14. From the distribution, the threshold for the energy between assembly SOP steps and NVA activities was determined. Using the determined threshold, the input to the model was classified between assembly SOP steps, all 6 of them, or NVA activities. The F1-score for this binary classification was determined to be 0.9744, with precision and recall being 0.9799 and 0.9689, respectively.

In this section, the NVA activities detection was evaluated by integrating OOD detection approach with SMIRL. The assembly step time was determined for the cases of with and without OOD detection and was compared with the human-inferred step time using NMAE, Equation 4.3. The results can be seen in Table 4.13. Without the energy-based approach for identifying NVA activities, all the NVA activities in each assembly cycle were detected to be step-1 (Position Motherboard), leading to its step time being 3-fold the human-annotated time.

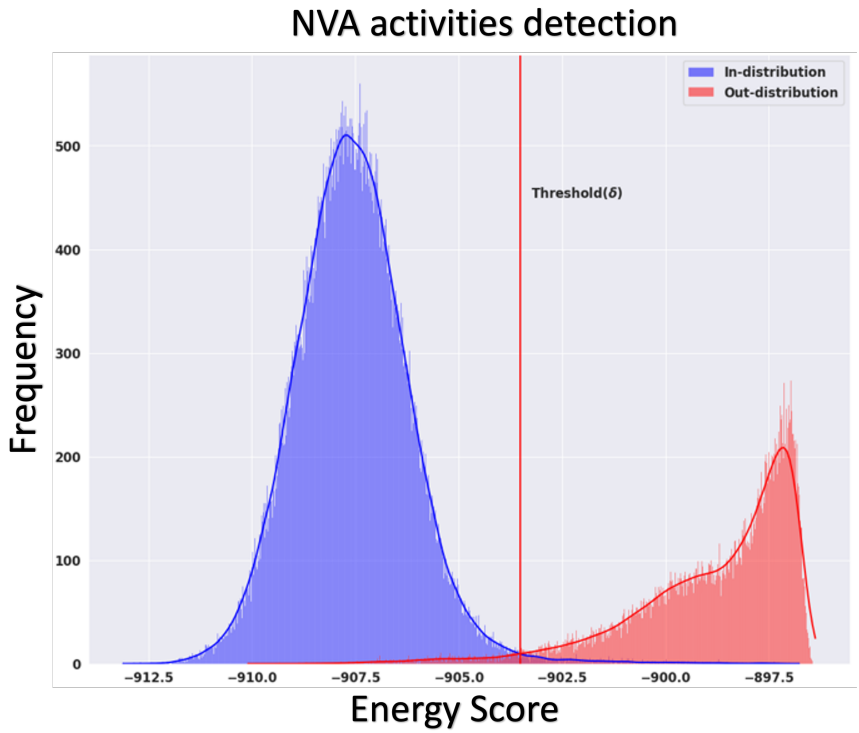


Figure 4.14: Distribution of energy scores for SOP steps and NVA activities [71].

In contrast to the approach in Table 4.5, where the model was exclusively trained on NVA activities, the current approach requires far less data with only a slight loss in performance. Additionally, the NVA activities across different assembly workstations in an assembly line can be collated to represent D_{nva}^{train} distribution of the assembly line.

The proportion of NVA activities used as D_{nva}^{train} can impact its detection ability. Hence, the OOD detection ability was evaluated, by computing the F1-Score and the Area Under Receiver Operating Characteristic (AUROC) curve, for different proportions of NVA activities, as seen in Table 4.14. The ROC curve is a plot between true positive rate and false positive rate, and AUROC summarizes the curve into a single number between 0 and 1. The proportion here corresponds to the ratio of total time spent in NVA activity to total time spent working on the assembly SOP steps, across all 13 cycles used in the model training.

Finally, the sensitivity of the developed approach to changes in lighting conditions, anthropometric variations of the assembly operators, and speed of the assembly operation were studied.

Table 4.14: Proporion of NVA activities and its impact on OOD detection [71].

Proportion of NVA activities	F1-Score	AUROC Curve
3.2%	0.8321	0.8752
14.3%	0.8666	0.8965
24.5%	0.9373	0.9495
51.8%	0.9744	0.9804

From our study, the following deductions were made: 1) The anthropometric variations associated with the operators and speed of motion had minimal impact. The reason can be attributed to the diversity in the training dataset, which enabled the model to form a resistance against anthropometric variations. Also, the higher camera FPS enabled capturing rapid human actions reliably. 2) The most impactful factor was the lighting of the workstation; this is because the optical flow relies on the lighting intensity variation between frames. Hence, in this work, we assumed the lighting conditions of the workstations to be unchanged during their operation and monitoring.

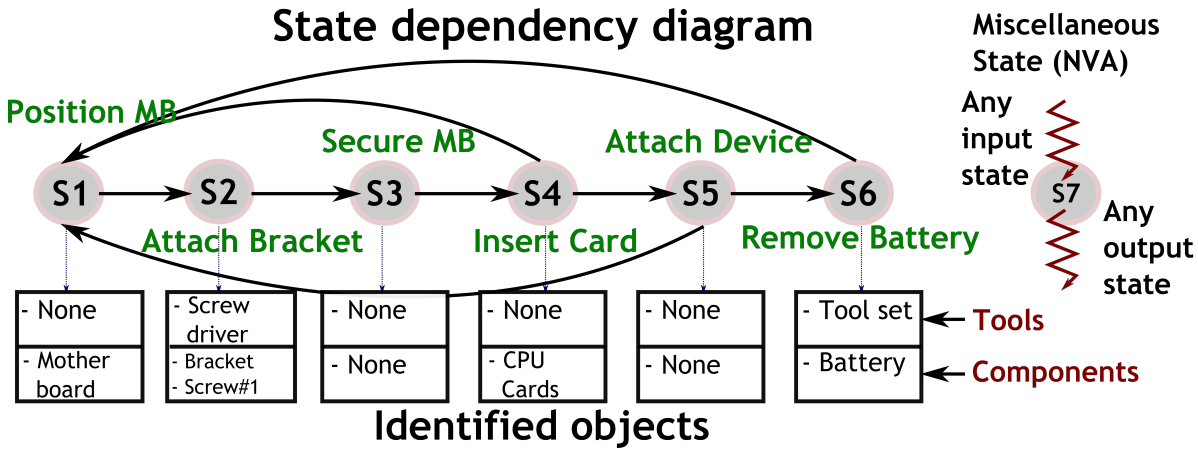


Figure 4.15: Relationship between assembly steps and its associated objects [71].

4.5 Development of Human-centric Assembly Guidance System

The assembly guidance system intends to guide the assembly operators in performing their tasks reliably, either during the training phase or in continuous operation. It also provides the means to ensure that the 5S practices (sort, set in order, shine, standardize, and sustain) are followed in the workstation without fail, to create an organized and productive workspace.

To guide the assembly operators in performing their tasks, the objects: tools, and parts, cor-

responding to each assembly step need to be identified in real time. During the initialization of SMIRL, a state dependency matrix was provided that encapsulates the relationship between the assembly steps in an assembly operation. To detect objects, in addition to the state relationships, information on the tools and parts corresponding to each step was also provided. The state dependency matrix along with the objects associated with each assembly step can be seen in Figure 4.15.

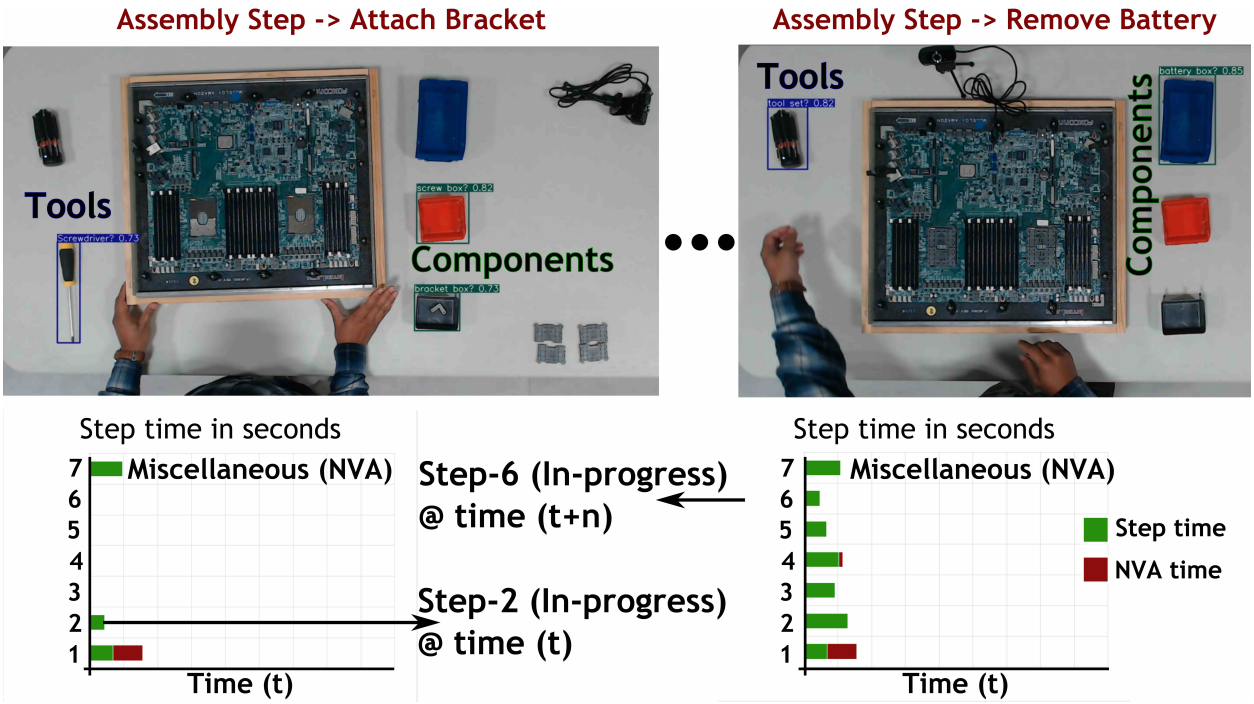


Figure 4.16: Integration of object detection and assembly action-localization [71].

An object detection module was integrated with SMIRL to realize a functioning guidance system. The model used to detect the objects was YOLOv7, which was transfer learned by fine-tuning the layers to identify the objects corresponding to the assembly operation under study. During inference, the frames from the video stream were processed by both SMIRL and the object detection module in parallel. The frames input to the models were preprocessed. For the case of SMIRL, the preprocessing involves dense optical flow computation, normalization, and resizing the frames to the dimension 112×112. The optical flow augments motion detection from video frames and normalization improves the deep learning model training process. The SMIRL identifies the assembly step s_t at time t. YOLOv7 identifies the objects present in the workstation. To guide the assembly operators, the objects corresponding to the next step, $(s+1)_t$, need to be identified.

The inference from SMIRL and the state dependency matrix was used to identify the prospective next assembly step(s). A binary mask that highlights the tools and components was generated depending on the assembly step, $(s+1)_t$. The binary mask selectively removes and/or highlights the objects corresponding to the state of the assembly cycle. The objects corresponding to assembly tools were highlighted using blue bounding boxes, and the components that go into the part were highlighted using green bounding boxes, see Figure 4.16. The described inference process can be seen in Figure 4.17. The developed assembly guidance system can infer at a rate of 55 FPS using a single RTX-6000 GPU. In addition to guiding the assembly operators in performing their tasks, it can also ensure that the 5S standards, particularly "Set in Order", are maintained at the workstation. Selectively identifying the tools required to perform an assembly step ensured that they were "Set in Order" once the tasks were complete. A snapshot of the inference process showing the assembly step inferred by SMIRL and the identified objects can be seen in Figure 4.16.

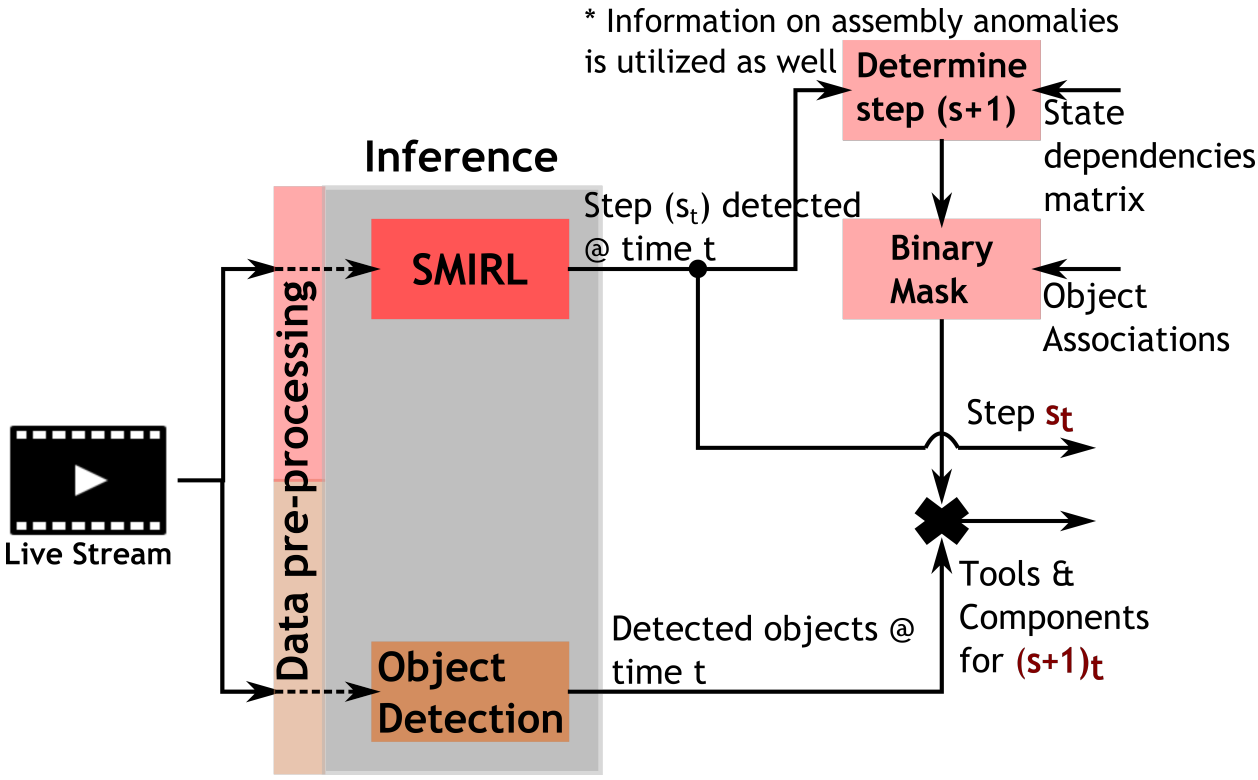


Figure 4.17: Inference process for the assembly guidance system [71].

The framework of the AI integrated assembly guidance system is shown in Figure 4.18. The system is comprised of four modules: 1) SMIRL, 2) Object detection system, 3) Projection system,

to overlay instructions directly on the workbench, and 4) Feedback module to rectify detected anomalies. The camera overlooking the assembly workstation streams the assembly operation. The captured frames are sent to the SMIRL and object detection model for inference. Using the results, the tools and parts on the workstation are highlighted using a projector. The camera and projector view are calibrated to ensure the location of the detected bounding box matches the projected one. A set of performance measures along with the feedback to rectify the assembly mistakes (sequence break, missed step) are shown using a monitor attached to the workbench.

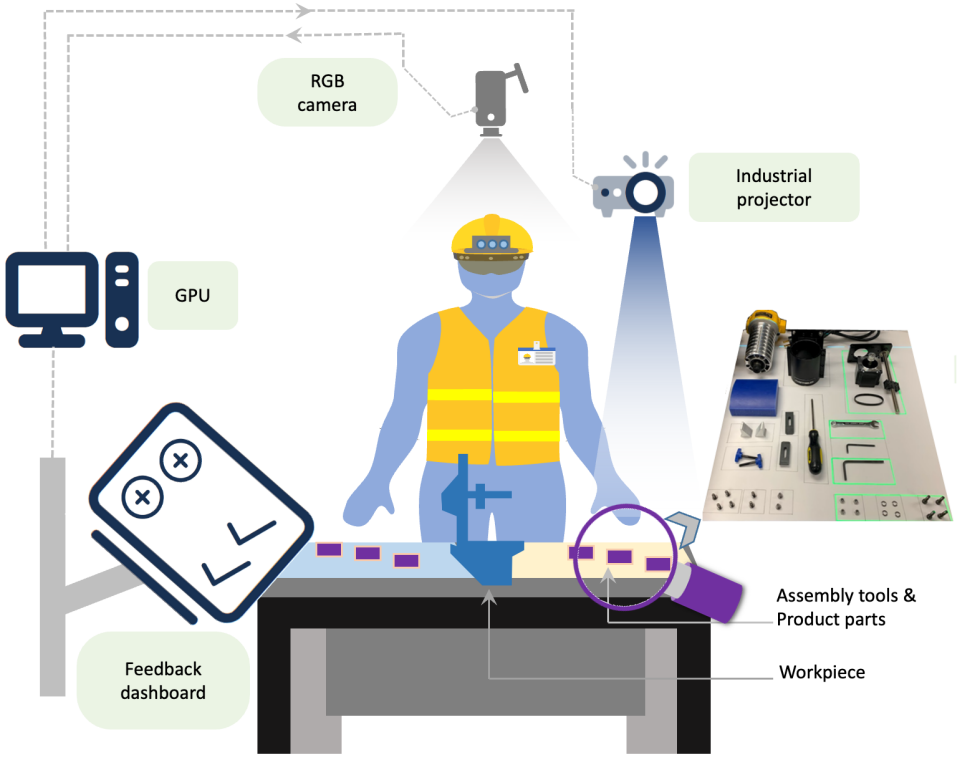


Figure 4.18: AI integrated assembly guidance system [71].

4.6 Graph Modeling of Assembly Operations

Some efforts have been made in computer vision action detection tasks toward capturing detailed contextual relationships between entities in a video in both spatial and temporal space by modeling images and videos as graphs [111–113]. A few studies explored the capabilities of GCN models in video action classification and localization tasks.Zeng et al. [100] used temporal proposals with GCN for temporal action localization. However, their approach assumed that the entire video

was available when making inferences. Wang and Gupta [114] used space-time graphs to represent videos and a GCN was used for action classification. Their model circumvents some of the limitations of other computationally expensive models but at the same time captures long-term dependencies between different object proposals by using similarity graphs and state changes of those objects using spatial-temporal graphs. Ji et al. [115] learned the complex object interactions by decomposing human actions in videos into spatiotemporal scene graphs.

These studies demonstrated that modeling videos as graphs can capture long-range spatial and temporal dynamics in video sequences. For human-centric assembly operations, it is crucial to develop monitoring systems that can account for the diversity and complexity of assembly lines, ensuring the safety of operators, and address the various constraints present in production environments. Building on these previous works, we deployed a novel method for assembly monitoring by modeling assembly videos as graphs. This approach allowed us to capture the intrinsic physics of assembly operations as well as the long-range spatiotemporal relationships between operators, tools, and other objects involved in a sequence of assembly tasks. By structuring the data as a graph, we were able to represent complex interactions and dependencies that occur during assembly processes, which are often missed by traditional video analysis methods. At the time of writing, there was little to no development in the deployment of graph convolutions for action detection and localization specifically in assembly tasks. Although, in Chen et al. [116], an attention mechanism-based Graph Convolutional Network (GCN) with multi-scale features to model assembly behavior recognition, their dataset was not representative of the complex settings and multi-step processes typical of a real industrial environment. This limitation highlights the need for further research and development in applying graph-based methods to reflect the intricacies of industrial assembly operations.

In this section, the problem we are trying to solve is twofold: Firstly, we aim to model the actions/tasks performed with each step in an assembly cycle as spatial and temporal graphs. The spatial graphs will model the instantaneous information, whereas the temporal graphs will model the time-dependent information. Secondly, we aim to develop a Heterogeneous Graph Neural Network (GNN) that processes the graphs to identify the steps performed within an assembly cycle. Finally, the graph modeling approach of monitoring assemblies will be evaluated against two

assembly datasets captured from industries and contrasted with our previous approach, SMIRL, to showcase the impact of increased temporal length to improve the action detection and localization performance.

4.6.1 Methodology

An approach for modeling human actions within an assembly workstation as spatial and temporal graphs was developed. This was followed by the construction of an adjacency matrix construction and the heterogeneous graph architecture design for action detection and localization.

Graphs in Assembly Monitoring

It has been extensively studied that action detection and localization require capturing spatial and temporal information from videos [95]. The use of 3D Convolutional Neural Networks (3D CNN) have been used extensively in the literature to model the spatial and temporal information from videos, but, studies have shown their limitations when it comes to modeling the temporal information effectively and the computational requirements for deep learning model training [91]. Modeling the information in video data as graphs helps capture the object-object and human-object interactions. This is especially pronounced in assembly monitoring, as the object-object and human-object interactions are integral to the assembly of a component in human-centric assembly operations.

Each step with an assembly cycle constitutes a series of actions that must be modeled temporally and spatially to enable reliable action detection and localization. In this work, the information in a video, both within a frame and across several frames, was modeled as graphs with nodes and edges. Through this process, we can capture the spatial and temporal relationships between the objects in an assembly workstation. The graphs enable the capture of the physics associated with the assembly process, such as operation sequences and interaction of objects. Additionally, in an assembly operation, where the sequence of operations was predetermined, the likelihood of future steps depending on the completion of past steps is high. Traditional approaches of action detection and localization on videos do not consider this constraint, hence it was challenging to model such

relationships using 3D CNN, RNN, LSTM, etc., but, in the case of graphs such relationships can be effectively modeled as nodes and edges.

Spatial and Temporal Information Modeling

Assemblies were distilled into the interactions between objects and the operator performing the assembly operations, categorized into object-object, object-person, and person-person interactions. To effectively capture these interactions, it was essential first to identify the various objects within a workstation. Next, a metric was developed to quantify the level or intensity of these interactions. Additionally, interactions were classified into spatial and temporal categories based on their relationship with time. The metric accommodated the time dependency of interactions, ensuring a comprehensive analysis of both their spatial and temporal dimensions.

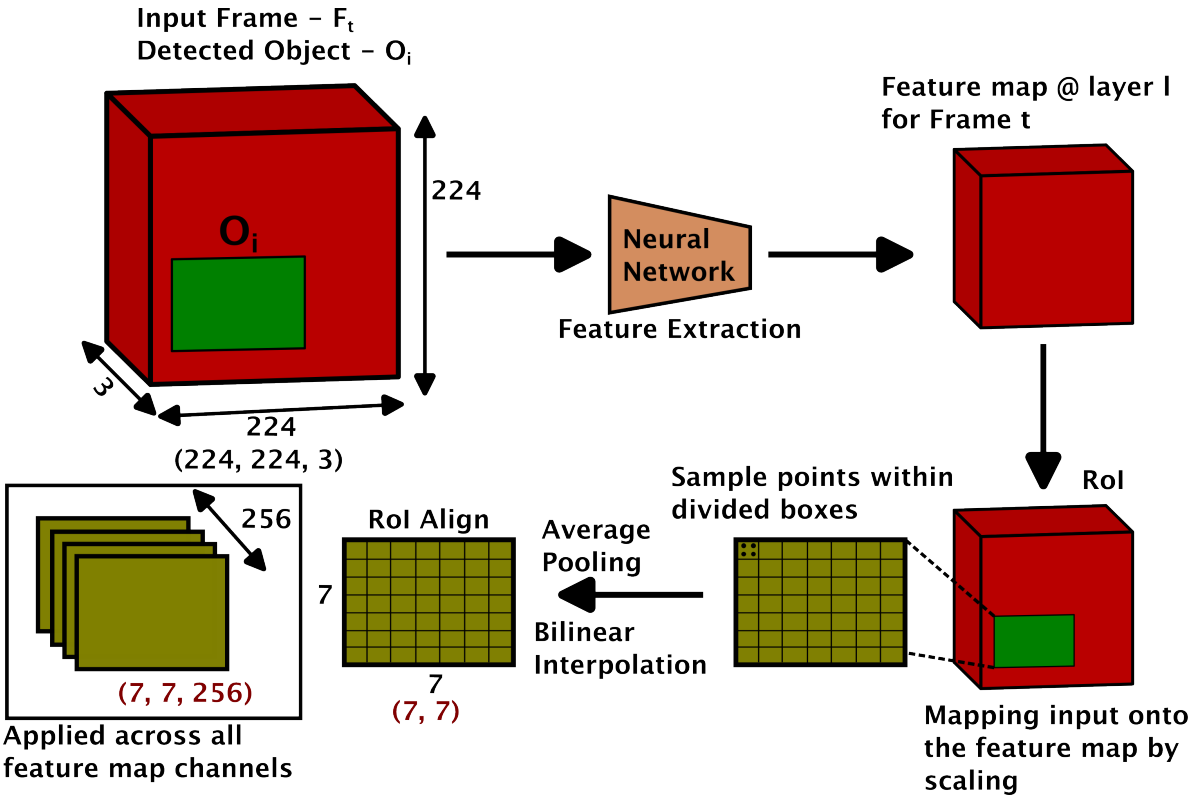


Figure 4.19: RoIAlign process for object feature extraction.

To initiate the information modeling process, objects within the assembly workstation were detected using a straightforward object detection model. Specifically, the Region Proposal Network (RPN) with an Feature Pyramid Network (FPN) backbone [117, 118], implemented via Facebook-

AI's Detectron2 library [119], was employed. The objective was to detect the various objects present without requiring object tracking. Each RGB video frame was resized to a dimension of 224×224. The detected objects, along with their bounding box information, were then processed for feature extraction. This feature extraction was either integrated within the object detection model or performed using a separate 3D-CNN model trained on video data from multiple assembly workstations. In both cases, the extracted feature map for each frame had a dimensionality of 256×1.

The Region of Interest (RoI)Align approach [117] was utilized to map and localize features corresponding to the detected objects within the model architecture, as illustrated in Figure 4.19. RoIAlign enhances object localization (localizing objects within a video frame) by rescaling the feature maps of the input frame. Each RoI is divided into a uniform grid of bins, with each bin containing adaptively calculated sampling points. These points are continuously mapped back to the input feature map, with bilinear interpolation applied at fractional locations to eliminate the drawbacks of quantizing RoI boundaries. The sampled values are subsequently aggregated through average pooling to yield a final feature map of the RoI, ensuring the output size after pooling is 7×7. By employing RoIAlign, the model architecture achieved improved object localization without information loss. Consequently, the feature representation for each detected object was standardized to a shape of $256 \times 7 \times 7$.

Formally, given a set of frames $\{I_t\}_{t=0}^{T}$ for a single assembly video, where t is the frame index and T is total number of frames, a set of feature maps $\{F_t\}_{t=0}^{T}$ is extracted using a CNN model θ_{CNN} given by:

$$F_t = \theta_{\text{CNN}}(I_t) \tag{4.9}$$

Then RoI feature X_t^i for the i^{th} object in frame t is given by:

$$X_t^i = RoIAlign(F_t) \tag{4.10}$$

As the next step, the relationship between the objects was established using IoU as the metric. Physically, it measures interaction between the objects; if the objects come in contact with each then the IoU metric is high, and it is low otherwise. IoU enabled us to model how close the objects were to each other and whether they were in contact. Spatial information was modeled by computing the IoU metric for each object across all other objects within a frame. In contrast, temporal information was modeled by capturing interactions across the frame. The spatial and temporal information modeling process can be seen in Fig. 4.20.

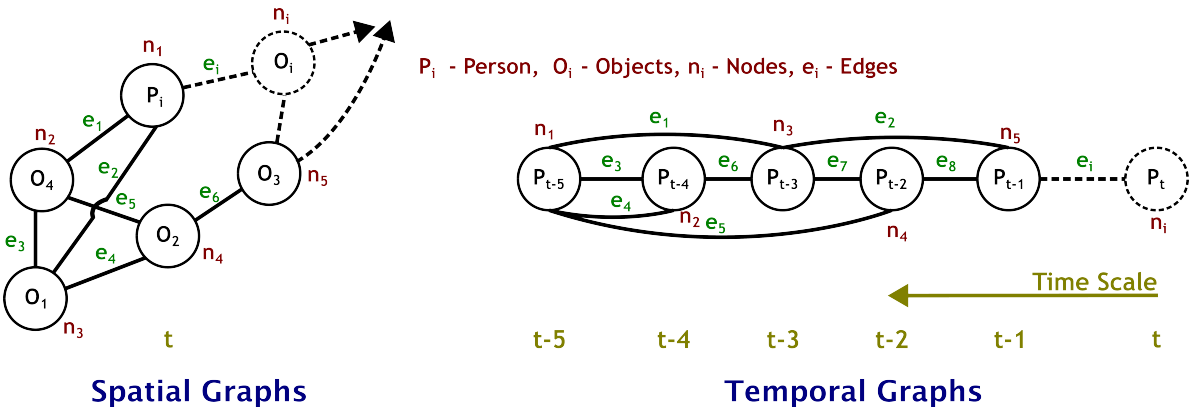


Figure 4.20: Assembly operations to spatial and temporal graphs.

4.6.2 Framework

This section discusses the framework behind our work, covering data collection, feature extraction, graph construction, model development, and the inference process for detecting and localizing assembly SOP steps. The framework is illustrated in Figure 4.21. It is divided into four segments:

1. **Data Collection:** This process involved setting up cameras around the assembly workstation to record the assembly operations. For practical applicability, only one camera was positioned to overlook the workstation.

2. **Interaction Modeling:** This process computed the level of interactions between objects, both instantaneous and over time.

3. **Graph Construction and Feature Extraction:** This step involved distilling the assembly operations into graphs by identifying the different objects, including humans, with the camera

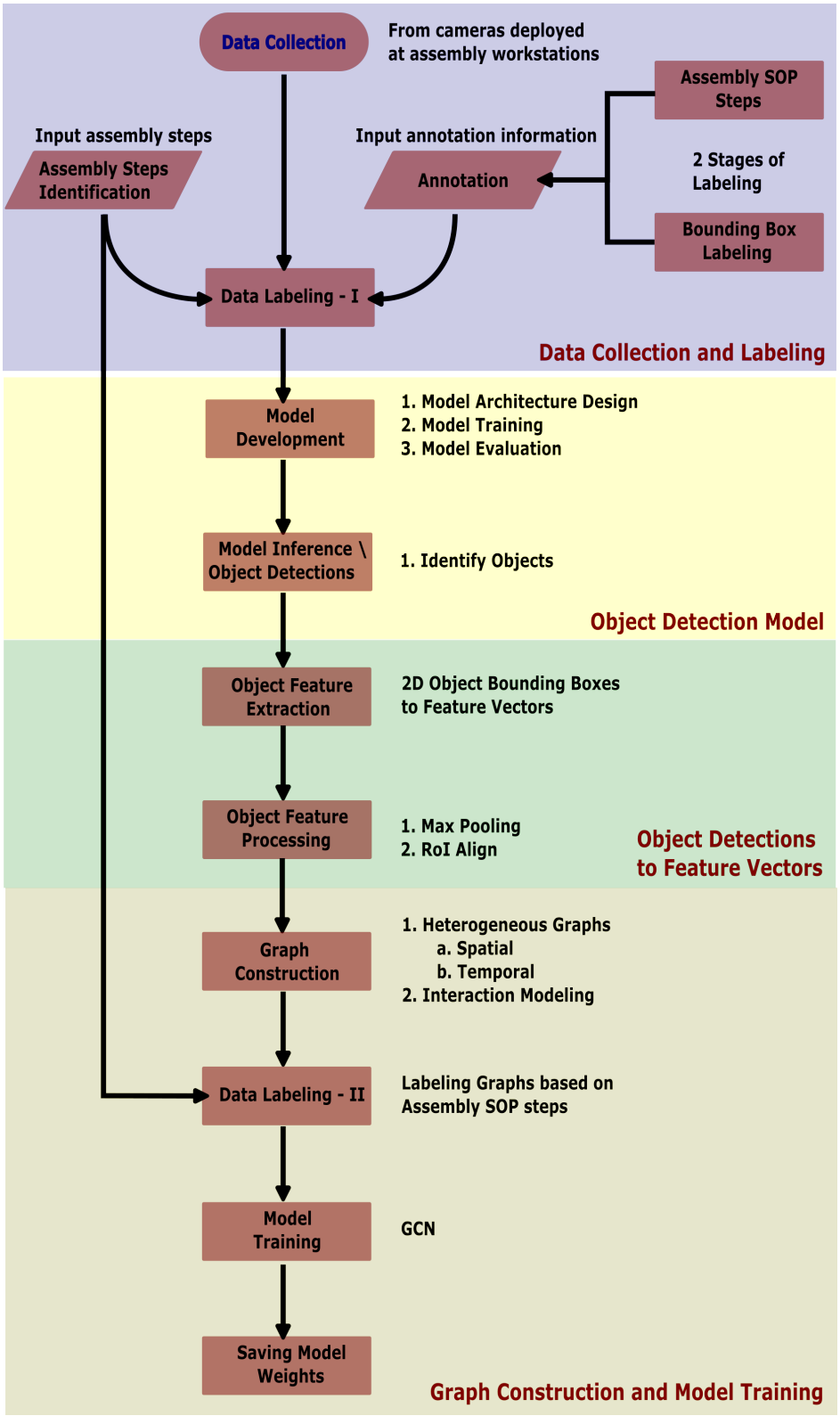


Figure 4.21: Framework for the process of deconstructing assembly operations as graphs.

view and extracting features for the feature learning process.

4. **Modeling Actions into Graphs:** This process involved the construction of spatial and temporal graphs from the operations performed in an assembly workstation. Additionally, the selection of hyperparameters and the object detection process were discussed.

5. **Inference:** The final step was detecting and localizing the assembly SOP steps based on the collected data and constructed graphs.

Each specific segment is further explained in detail in this section.

Data Collection

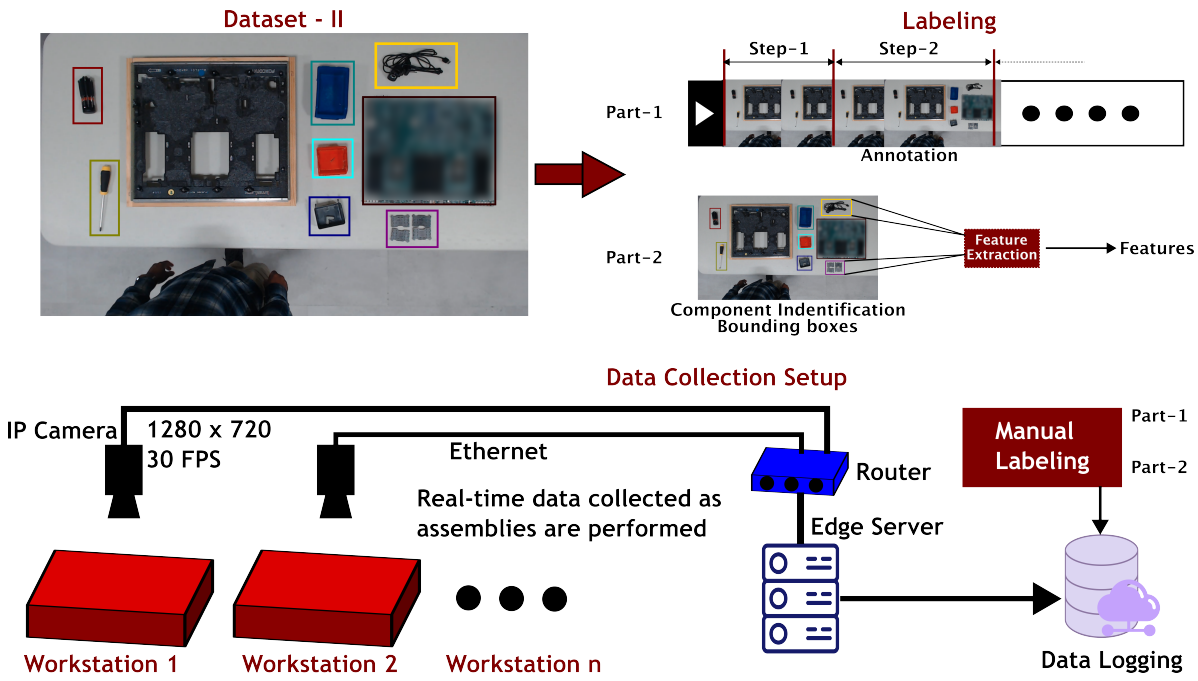


Figure 4.22: Experiment setup for data collection and labeling.

The data collection process can be seen in Fig. 4.22. The cameras installed overlooking the assembly operations were IP cameras with a resolution of 1280 x 720 pixels and 30 FPS. The video data were recorded and streamed in real-time to an edge server to enable real-time inference. For this study, the recorded data were used for two purposes: 1) For model development and training, and 2) To enable the evaluation of the spatial-temporal modeling of human actions, against the traditional approach of using 3D CNN for action-detection. Hence, the recorded data

were manually annotated by labeling the assembly step with each assembly cycle, additionally, the components present within an assembly workstation were labeled by creating bounding boxes around them. The bounding boxes created around the objects complement the feature extraction process discussed in Section 4.6.2. The DAQ setup enabled us to collect data from real assemblies located in manufacturing industries to evaluate the validity of our approach in real-world scenarios.

Interaction Modeling

Determining the interactions between objects and humans in an assembly workstation was a crucial aspect of representing assemblies as spatial-temporal graphs. Interactions between objects within a single frame were defined as spatial interactions, while interactions between objects across different time frames were considered temporal interactions. To compute these interactions, the IoU metric was used, as shown in Equation 4.11. The IoU metric measures the degree of overlap between two objects.

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}} \tag{4.11}$$

Interaction Modeling:

$$iou(i, j) \quad \text{where } i \neq j, \tag{4.12a}$$

$$iou(i_t, i_{t-n}) \quad \text{where } n = 1, 2, 3, ..., \tag{4.12b}$$

Spatial interactions were computed by calculating the IoU between each object i and every other object j within the same video frame. Temporal interactions, on the other hand, were computed by comparing object i's current position with its occurrences over the previous n video frames. The parameter n is a hyperparameter that was chosen based on how much historical information the model required to accurately detect an assembly step. The constraint for n was that it should be less than the duration of the shortest assembly step.

The process of modeling these interactions, both spatially and temporally, is illustrated in

Figure 4.23, while the equations governing these interactions are detailed in Equations 4.12a and 4.12b.

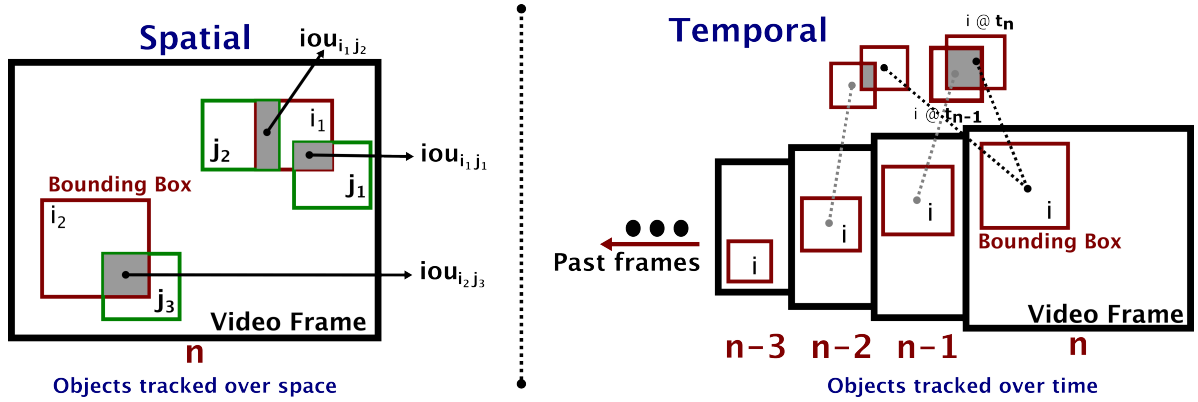


Figure 4.23: Modeling of spatial and temporal object interactions.

Graph Construction Process

The graph construction process essentially involves distilling the information present in the video into graphs. In this section, the approach taken in converting videos into graphs is discussed. Each video $\mathcal{V}$ defined as $\mathcal{V} = \{\mathcal{F}_t \in \mathbb{R}^{\mathcal{H} \times \mathcal{W} \times 3}\}_{t=0}^{T}$, where $\mathcal{F}$ is the frame with height $\mathcal{H}$ and width $\mathcal{W}$ and T presents the total number of frames in the video. Within each frame, $\mathcal{O}_{t,i}$ represents the i^{th} object in the t^{th} frame with $\mathcal{O}_{t,i} \in \{O_{t,1}, O_{t,2}, \ldots, O_{t,N}\}$ and N is the total number of objects in the t^{th} frame. For each object $\mathcal{O}_{t,i}$, a representation of the object was created using a d-dimensional feature vector $x_{t,i}$, where $x_{t,i} \in \mathbb{R}^d$ and d was an hyperparameter. The features were extracted using a separate feature extraction model that starts by identifying the objects present within a video frame followed by a feature extraction process.

From section 4.6.1, the shape of the $x_{t,i}$ determined by $X_t^i = RoIAlign(F_t)$ for the i^{th} object in frame t and was $d \times h \times h$, where h is the spatial output size of the detected object after pooling and bilinear interpolation. This process can be seen in Figure 4.19. The Average Pooling process was then applied to the extracted features of shape $d \times h \times h$ to get a shape of $d \times 1 \times 1$, similar to the approach followed by [114]. Thus, the d-dimensional feature vector $x_{t,i} \in \mathbb{R}^d$ of an object i in frame t has a shape of $d \times 1$. The hyperparameters d and h were chosen as 256 and 7, respectively.

Each detected object in a frame served as a node for our graph, and the feature representation

for a node can be given as $x_{t,N} \in \mathbb{R}^{N\times 256}$, where N is the number of objects in a frame. The interaction modeling between the objects within and between frames was achieved by computing the IoU between object-object, object-person, and person-person pairs as given in Equation 4.11, 4.12a and 4.12b. These interactions serve as the edges of the graph. The size of a single graph was determined by the hyperparameter l, which is defined as the temporal length of the graph. The temporal length specifies the number of frames from the past that a model could use to make predictions. Our graphs were heterogeneous, with two types of edges: spatial and temporal, where the spatial edges encapsulated the spatial interactions between detected objects in a single frame, and the temporal edges captured the temporal changes of certain objects between frames across the temporal length l as the assembly step progressed. The spatial graphs were thus constructed by computing the IoU between all object-object, object-person, and person-person pairs in a single frame, i.e., between $\mathcal{O}_{t,i}$ and $\mathcal{O}_{t,j}$, where $i \neq j$. The temporal graphs were constructed to capture the change in the position of a single entity over time. In other words, the temporal edges were only built between objects of the same class between frames, i.e., between $\mathcal{O}_{t,i}$ and $\mathcal{O}_{t-n,i}$.

In a typical manual assembly process, an operator performs a series of tasks, and the operator's actions provide crucial temporal cues, as they dictate the sequence of assembly steps. Additionally, in a typical assembly workstation, there is a central component being assembled, with other components being added to it. By capturing the temporal motion of both the operator and the main component being assembled, we hypothesized that it would be possible to track time-dependent changes in actions as the component progressed through the assembly cycle. Based on this hypothesis, the operator nodes and main component nodes were tracked temporally. In cases where the operator was not detected in the video frame, the hands were tracked and monitored temporally, owing to the camera's position.

A graph can be mathematically represented by an adjacency matrix $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$, where the rows and columns of the matrix are indexed by each node in the graph. $\mathbf{A}[u, v]$ represents the strength of connection or relationship between the nodes u and v, termed as edge weight. For a heterogeneous graph, the adjacency matrix $\mathbf{A}_r$ can be defined for each edge with relation type r where $(u, r, v) \in \mathcal{E}$, and $r \in (r_{spatial}, r_{temporal})$. The entire heterogeneous graph can be summarised as the tensor $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{R}|\times|\mathcal{V}|}$. The spatial graph $\mathbf{A}_{spatial} \in \mathbb{R}^{|N|\times|N|}$ for this work between object

nodes i and j for video frame t with a total number of N object nodes thus defined as

$$\mathbf{A}_{spatial}(\mathcal{O}_{t,i}, \mathcal{O}_{t,j}) = iou(i_t, j_t), \text{where } i \neq j \tag{4.13}$$

The temporal graph $\mathbf{A}_{temporal} \in \mathbb{R}^{|N| \times |N|}$ between object i in video frame t and all past occurrences of i over l frames with an effective N detected objects is defined as

$$\mathbf{A}_{temporal}(\mathcal{O}_{t,i}, \mathcal{O}_{(t-l),i}) = iou(i_t, i_{t-l}) \tag{4.14}$$

It is to be noted that the spatial graphs were undirected with no self-loops, while the temporal graphs were directed with no self-loops. The entire heterogeneous graph for temporal length l can then be represented as

$$\mathcal{G} = \{\mathcal{O}_{l,M}, \mathcal{E}, \mathcal{R}\} \tag{4.15}$$

where $\mathcal{O}_{l,M}$ is the set of M nodes or total number of detected objects in a frame window of temporal length l, i.e, $\mathcal{O}_{l,M} = \{O_{0,1}, O_{0,2}, \ldots, O_{0,p}, \ldots, O_{1,q}, O_{1,(q+1)}, \ldots, O_{1,s}, \ldots, O_{l,M}\}$, where $(O_{t,Q}, r, O_{t',Q'}) \in \mathcal{E}$ is an edge with relation type r from node $O_{t,Q} \in \mathcal{O}_{l,M}$ to node $O_{t',Q'} \in \mathcal{O}_{l,M}$, and $r \in \mathcal{R}$ depicts the relation type where $\mathcal{R} = \{r_{spatial}, r_{temporal}\}$. In the case of spatial graphs, $(O_{t,Q}, r_{spatial}, O_{t',Q'}) \in \mathcal{E} \rightarrow t = t', Q \neq Q'$, whereas, in the case of temporal graphs, $(O_{t,Q}, r_{temporal}, O_{t',Q'}) \in \mathcal{E} \rightarrow t \neq t', Q = Q'$. The two adjacency matrices, spatial given by,

$$\mathbf{A}_{spatial} \in \mathbb{R}^{|\mathcal{O}_{l,M}| \times |\mathcal{O}_{l,M}|} \tag{4.16}$$

and the temporal given by,

$$\mathbf{A}_{temporal} \in \mathbb{R}^{|\mathcal{O}_{l,M}| \times |\mathcal{O}_{l,M}|} \tag{4.17}$$

with the effective adjacency matrix obtained by the combination of spatial and temporal given by,

$$\mathcal{A} \in \mathbb{R}^{|\mathcal{O}_{l,M}| \times |\mathcal{O}_{l,M}|} \tag{4.18}$$

can be seen in Figure 4.24.

After assigning edge weights to the adjacency matrix $\mathcal{A}$, the weights were normalized. Spa-

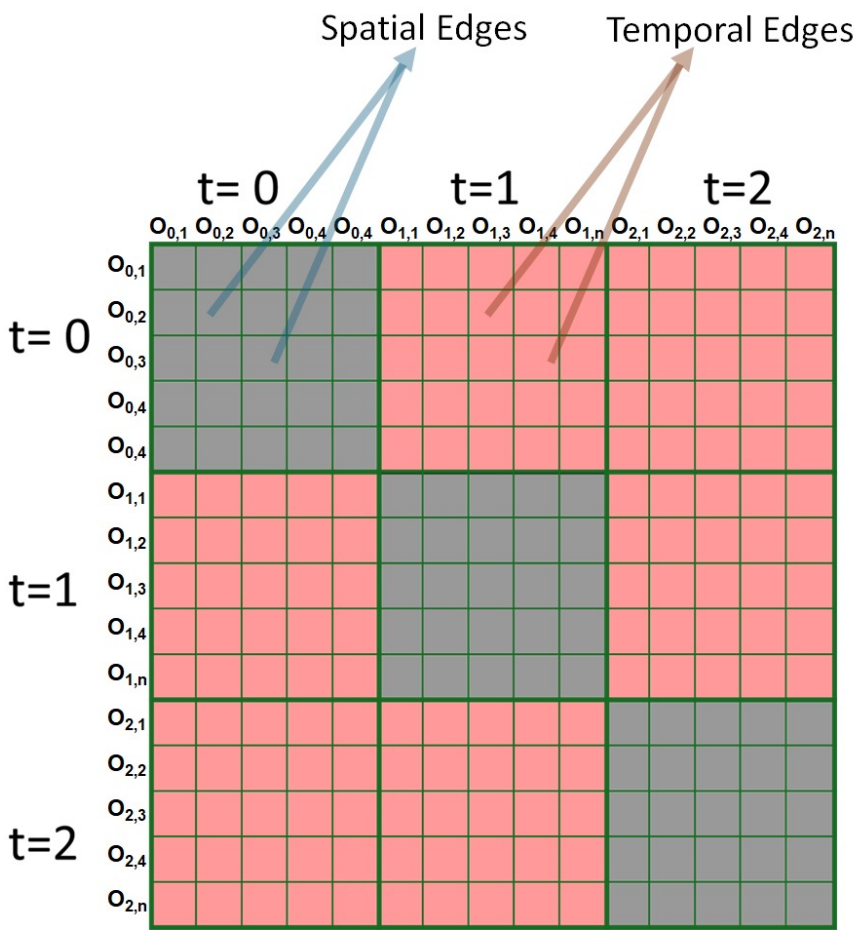


Figure 4.24: Structure of the adjacency matrix.

tiotemporal graphs were constructed for each frame window, ensuring a certain percentage of overlap between consecutive windows (i.e., graphs). This overlapping window approach was used to incorporate past information as a hyperparameter, allowing the model to capture long-range spatiotemporal interactions in the assembly operations effectively. By using this method, the system could better represent the sequence of actions and their temporal dependencies across multiple frames.

The graph construction methodology was not without its limitations. For example, assembly operations often involve multiple objects of the same type, but our modeling approach did not differentiate between objects belonging to the same class. Additionally, the object detection models used were prone to missed detections, occlusions, and the disappearance or reappearance of objects. To mitigate this, we constructed spatial graphs that connected all detected objects, regardless of

their class, making the approach class-agnostic. This underscores the need to develop methods for establishing similarity metrics for objects of the same type, such as multi-object tracking [120].

Moreover, the datasets in this study were recorded using a single overhead camera. This single-viewpoint setup inherently limits the ability to capture all granularities of interactions between entities during an assembly step, increasing the likelihood of missed objects and occlusions. A more effective approach would involve using multiple cameras from various vantage points to overcome this limitation. A multi-view setup would provide a comprehensive understanding of the assembly environment from different angles, leading to a more robust representation of the spatial relationships between objects and operators.

Modeling Actions into Graphs

The graph construction process required several hyperparameters to be predetermined to effectively model assemblies as spatiotemporal graphs. The primary hyperparameter was the temporal length l, which determined the number of past frames considered during the graph construction process. A higher temporal length improved the accuracy of action detection, but the duration of the shortest assembly step constrained it.

In real-time monitoring, inferences needed to be made for every video frame. As a result, spatial and temporal graphs were constructed for each frame. The spatial graphs captured the interactions between key objects identified by the object detection algorithm, while the temporal graphs modeled the interactions of selected objects across the past l frames. In the temporal graphs, these objects were tracked over time by analyzing changes in their features and interactions.

The first step in the graph construction process, as outlined in Section 4.6.2, was the detection and identification of objects. Objects in the assembly workstation were annotated using bounding boxes and labels. These annotated objects were then used to fine-tune an open-source object detection model for consistent and accurate identification. Training the object detection models involved selecting specific video frames from within an assembly cycle to ensure robustness.

Inference

The inference process involves the continuous, real-time monitoring of human-centric assembly operations. Once the graphs were constructed for several assembly cycles, they were used to train a Heterogeneous Graph Convolutional Network (GCN) classifier. The GCN was employed to classify each graph according to the actions associated with different SOP steps in the assembly process. The inference module played a crucial role by integrating the GCN's classifications. This module combined the classifications to determine step times and cycle times for each assembly cycle. By analyzing the spatial and temporal interactions, the inference module provided valuable insights into the efficiency and timing of the assembly process.

4.6.3 GCN Model Development

The model used to process the graphs was a GCN, which is a variant of Graph Neural Network (GNN) [121]. GNNs can be considered a neural message-passing scheme, where graph nodes exchange messages with their neighbors, aggregate the information, and then learn from it through training. The model architecture can be seen in Figure 4.25, and the model development process can be summarized as follows:

1. An off-the-shelf object detection model was employed to identify the various objects in an assembly workstation. Each identified object, along with its bounding box information, was embedded with its CNN features (extracted using either the object detection backbone model or a separate 3D-CNN model) and RoIAlign, resulting in a feature representation of $x_{t,N} \in \mathbb{R}^{256}$, where N is the number of objects in frame t.

2. A heterogeneous graph was then constructed with two types of edges/relations: spatial and temporal. This graph contained M nodes, corresponding to the M objects identified within a frame window of length l. The key hyperparameters for graph construction were the temporal length l, the overlapping window, and the type of temporal edges used in model development.

3. Each adjacency matrix, used to model spatial and temporal information, was associated with a feature matrix of shape $M \times 256$. The feature matrix was row-normalized to standardize

the inputs.

4. The model processes a heterogeneous graph $\mathcal{G} = \{\mathcal{O}_{l,M}, \mathcal{E}, \mathcal{R}\}$, where the node feature matrix has a shape of $M \times 256$, the edge index tensor shape of (2×Number of edges in relation type r), and the edge weight tensor shape of (number of edges in relation type r $\times 1$). The model performs heterogeneous convolutions on this graph, producing final graph embeddings. A linear layer is then applied to these embeddings to output the class logits for each graph.

Training Methodology

The GCN model was trained using various temporal lengths, overlapping windows, and types of temporal edges. The dataset was split into training, validation, and testing sets, with 70% of the assembly cycles used for training, 15% for validation, and the remaining 15% for testing. To address data imbalance, which arises due to the varying step times in assembly operations, class weights were applied during model training.

The model was trained with 64 hidden channels, a batch size of 128, and 250 epochs with early stopping. The Adam optimizer was used, with a learning rate of 0.00001. The loss function was categorical cross-entropy, which internally applies a softmax function when calculating the loss. Validation loss was monitored during training, and the model weights were saved when the best performance was achieved. The training process was carried out using a single NVIDIA RTX 6000 GPU.

4.6.4 Evaluation

The dataset used for evaluation in this section were the ones discussed in Section 4.3.6. To ensure consistency two different modeling approaches were evaluated against the same dataset. The evaluation of this approach was conducted in three phases:

1. **Frame-level Inference:** The first phase focused on assessing the model's ability to detect actions by classifying sequences of video frames recorded during an assembly cycle based on the assembly SOP steps. This process, called frame-level inference, involved analyzing each

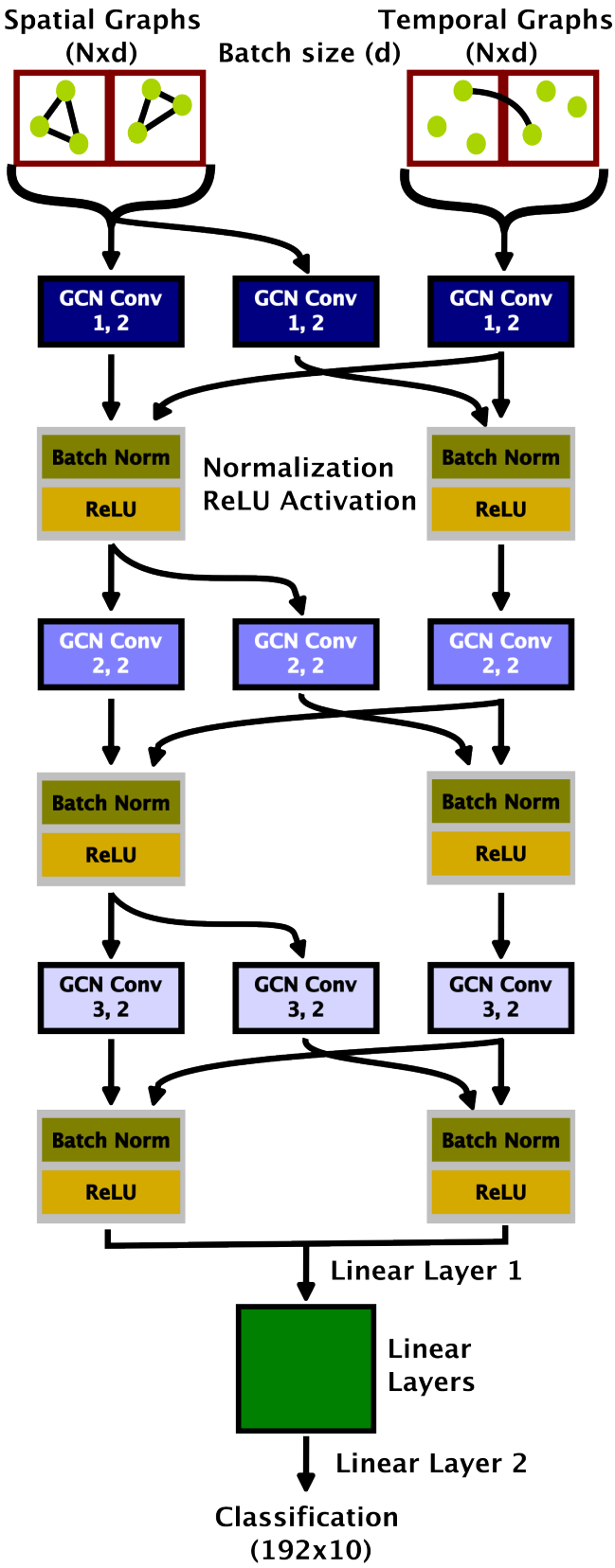


Figure 4.25: The GCN model architecture developed for this study.

Table 4.15: Macro-average F1-Scores.

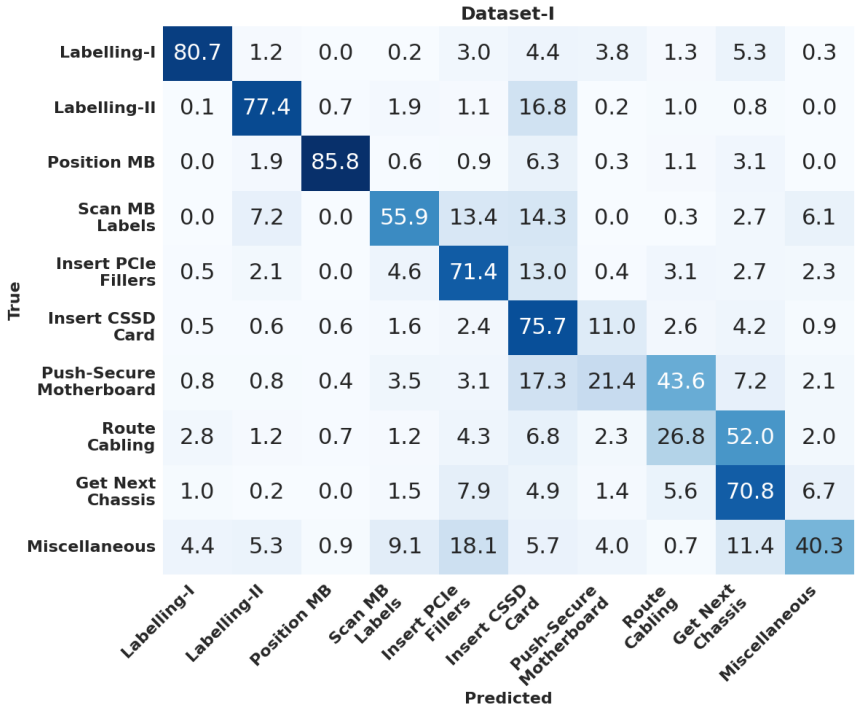
Dataset-I	F1-Score	Dataset-II	F1-Score
Labelling-I	0.8489	Position Motherboard	0.9816
Labelling-II	0.8109	Attach Bracket	0.8963
Position Motherboard	0.8821	Secure Motherboard	0.9150
Scan Motherboard and Label	0.6005	Insert Card	0.8814
Insert PCIe Fillers	0.7227	Attach Device	0.9004
Insert CSSD Card	0.6488	Remove Battery	0.9048
Push-Secure Motherboard	0.2776	Miscellaneous	0.8839
Route Cabling	0.3374	-	-
Get next Chassis	0.5067	-	-
Miscellaneous	0.3393	-	-

frame window to determine the assembly step being performed. The model's performance was measured by correctly identifying the assembly steps from the video frame sequence.

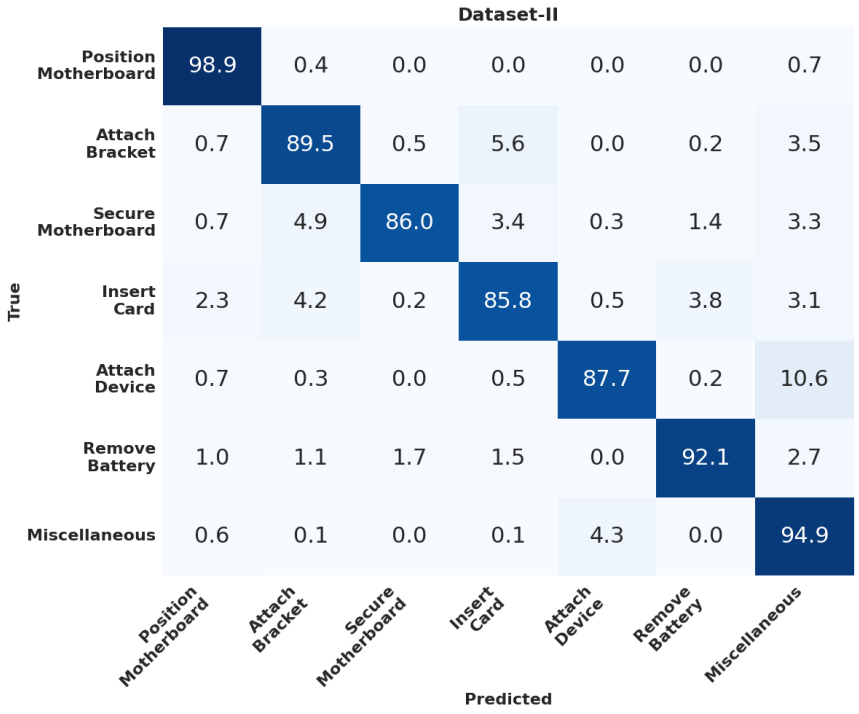
2. **Time-level Inference:** The second evaluation phase involved measuring step time and cycle time. Step time refers to the time taken to complete each assembly SOP step, whereas, cycle time refers to the time taken for each assembly cycle. The model's determination of these times was compared against human-annotated time information.

3. **Model Characteristics:** In the third phase of evaluation, the impact of the hyperparameters used in the model development process, the impact of the different approaches of feature extraction from objects, and finally the impact of different objects tracked temporally over time were studied. This study enabled us to understand the best approach to construct graphs from assembly operations.

For the cases of frame-level and time-level inference, the model with the best characteristics was used. Additionally, for both datasets' training and testing data instances were kept consistent to ensure repeatability. In the case of frame-level inference, the effectiveness of our approach to detect assembly actions was evaluated by testing the trained Heterogeneous GCN model on the testing dataset. This dataset included graphs corresponding to different assembly SOP steps, which were not seen by the model during the training phase. The performance of the GCN model in classifying these graphs was assessed by examining how accurately it identified each SOP step. The results of



(a)



(b)

Figure 4.26: Confusion matrix for Dataset-I (a) and Dataset-II (b) created using the best performing model for each case.

this classification are illustrated through confusion matrices, seen in Figure 4.26, and F1-Scores for each dataset used in the study, seen in Table 4.15.

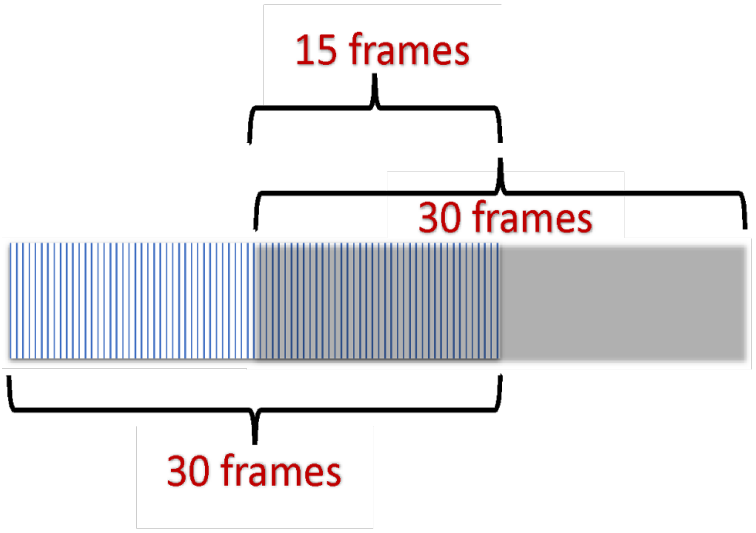


Figure 4.27: Overlap rate for the video frames.

When constructing graphs, the information from the video was represented as nodes and edges. In contrast to traditional computer vision models, there was a deliberate process of information reduction and assimilation. To better understand how graphs process information and the effectiveness of representing videos as spatiotemporal graphs, several evaluations were conducted. The following sections closely examine the various analyses performed to evaluate our approach to representing videos as spatiotemporal graphs.

The first analysis focused on studying the impact of the overlap rate on the graph development process. The computation process for the overlap rate is illustrated in Figure 4.27. Using Dataset-I, the process of graph construction, model development, training, and evaluation on the testing dataset was performed for various overlap rates. As discussed previously, our approach allows the selection of objects to be tracked temporally based on experience. For this overlap rate study, the objects tracked temporally were either the motherboard or the person. In Figure 4.28, the motherboard was tracked temporally across different overlap rates, while in Figure 4.29, the person was tracked temporally for the same range of overlap rates.

In the second analysis, the impact of different choices for temporal connections was studied. Various objects were tracked temporally in both Dataset-I and Dataset-II. For Dataset-I, the objects

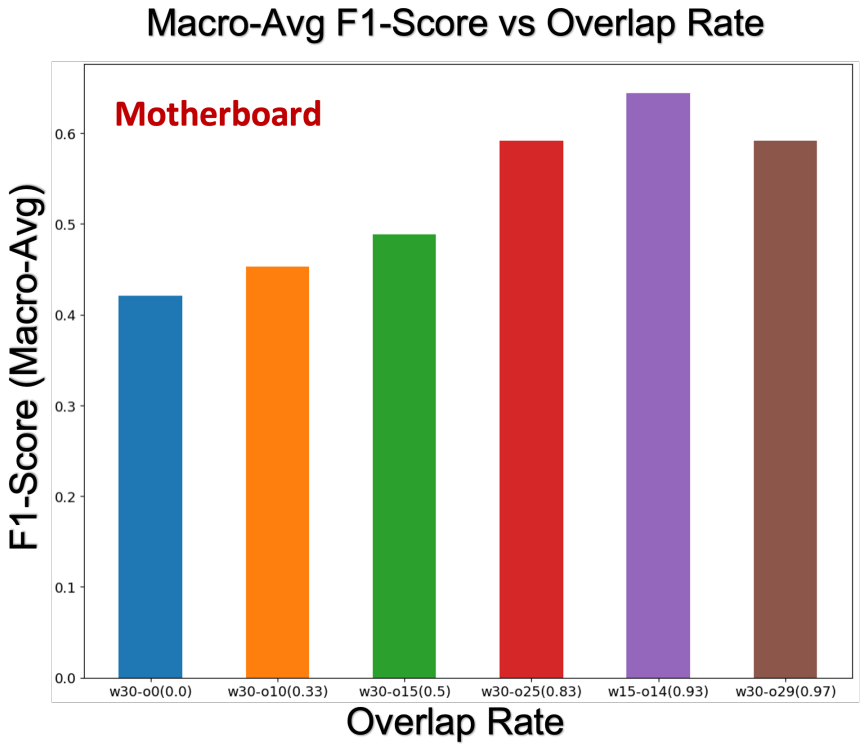


Figure 4.28: Impact of overlap rate for Dataset-I with Motherboard tracked temporally.

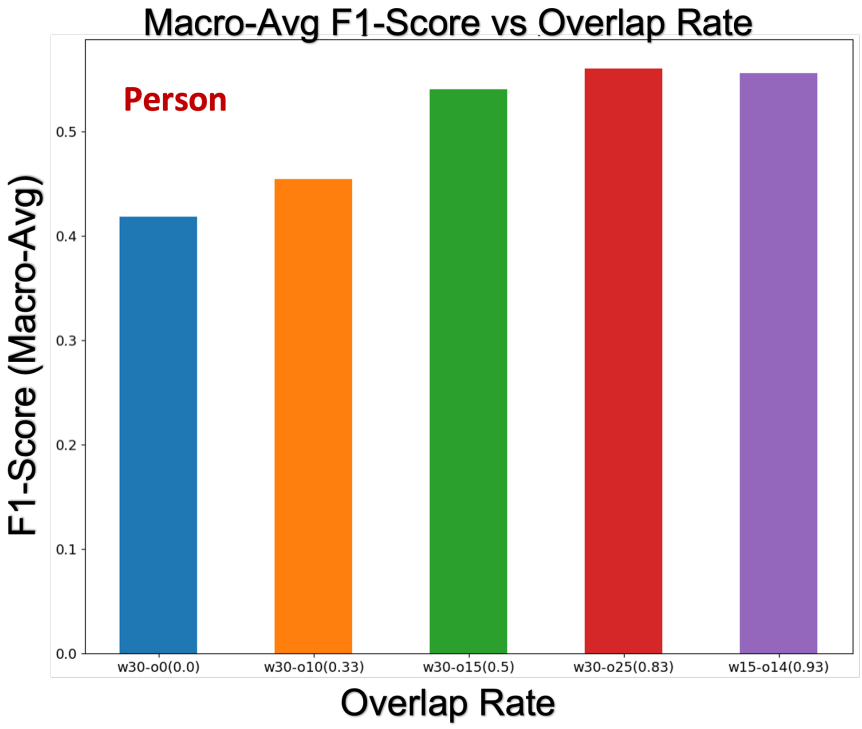


Figure 4.29: Impact of overlap rate for Dataset-I with Person tracked temporally.

tracked temporally included the motherboard, the person, and a combination of both. In the case of Dataset-II, the objects tracked temporally were the hands, the motherboard, and a combination of hands and the motherboard. The impact of temporal connections for Dataset-I and Dataset-II is shown in Figure 4.30 and Figure 4.31, respectively. In these datasets, the objects with significant temporal motion were the person in Dataset-I and the hands in Dataset-II. The plots indicate that tracking objects with significant temporal motion resulted in improved performance. Additionally, tracking multiple objects temporally also led to generally better performance.

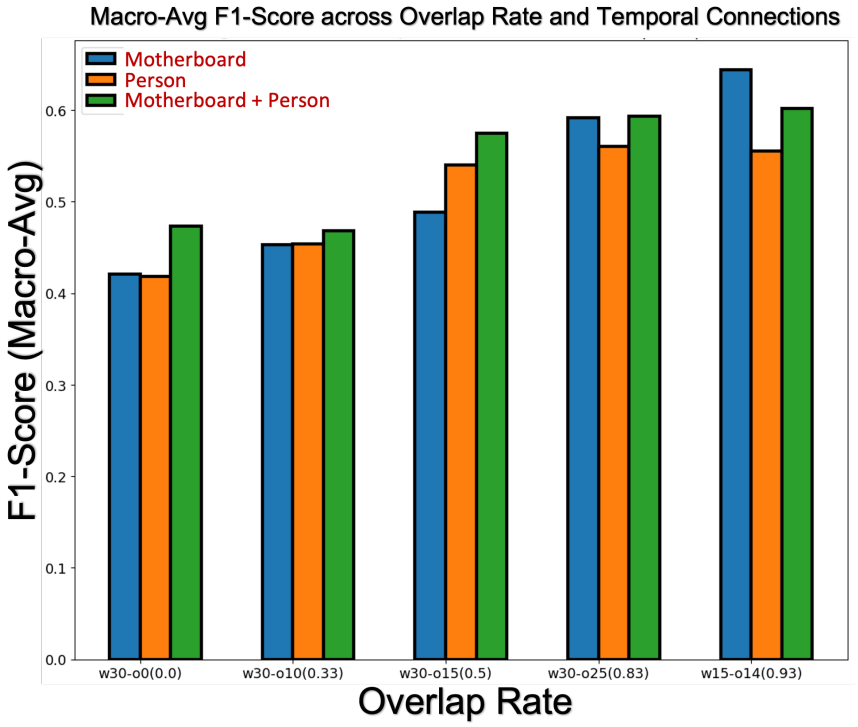


Figure 4.30: Impact of temporal connections for Dataset-I.

The final analysis involved comparing the performance of graph-based modeling of assemblies to our previous approach, SMIRL. Figure 4.32 illustrates the relative performance for each assembly step in Dataset-II. The figure demonstrates that the performance of the graph-based modeling approach is comparable to SMIRL, and in certain cases, it even outperformed SMIRL. These results suggest that graph-based modeling holds significant potential for efficiently representing assemblies as spatiotemporal graphs.

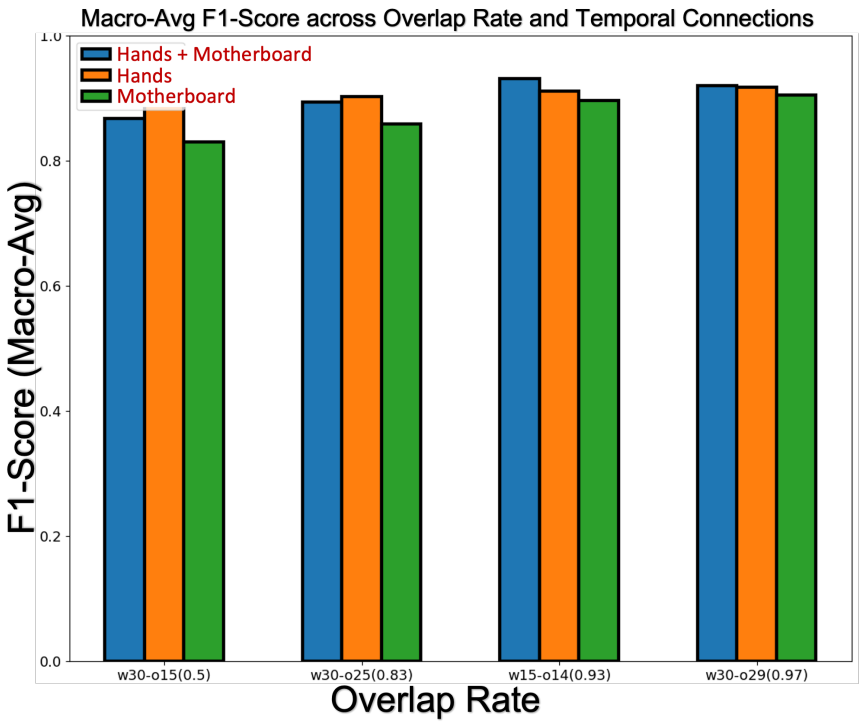
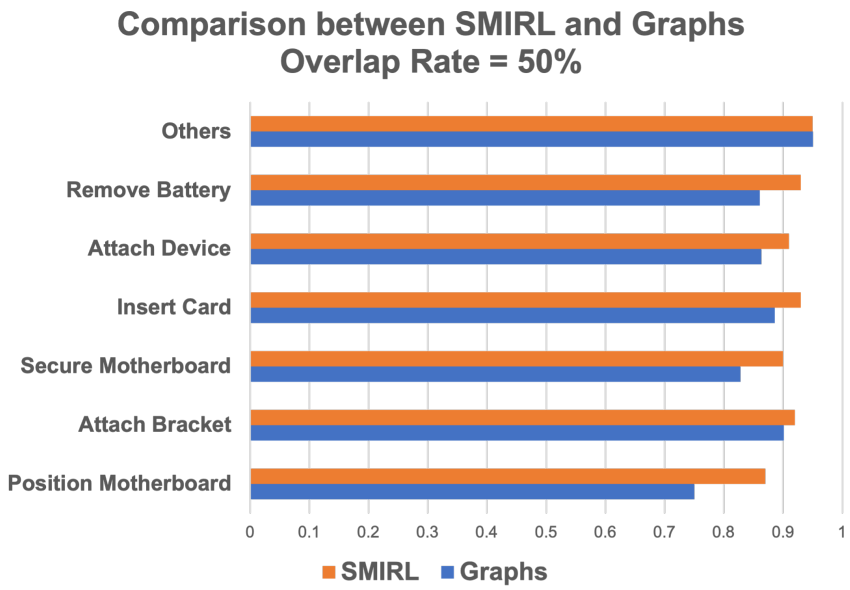


Figure 4.31: Impact of temporal connections for Dataset-II.

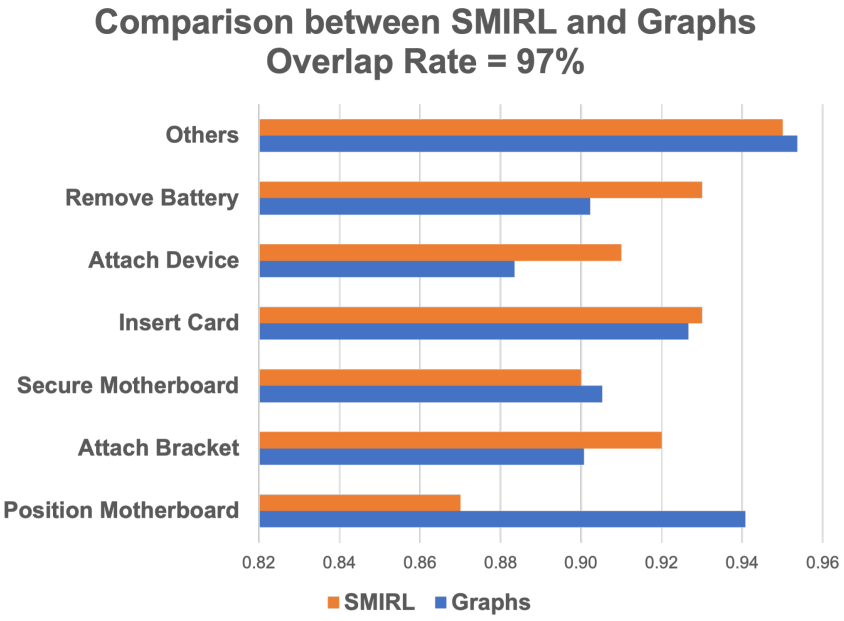
4.7 GUI Development for Deployment at Industries

To demonstrate the capability of our approach, a graphical user interface (GUI) was developed to run SMIRL. This tool allows users to select algorithms, assembly videos for inference, and adjust hyperparameters such as state transition time and cycle reset time. Once inference is initiated, the GUI can stream video from IP cameras for real-time analysis. A snapshot of the GUI is shown in Figure 4.33. As the inference runs, the system computes the proportion of value-added and non-value-added activities in real-time. Additionally, step and cycle times are displayed in a plot. When a cycle changes, information from the previous cycle is stored and tracked in an SQL database for future reference and analysis.

From the video stream shown in Figure 4.33, the object detection algorithm was also integrated with SMIRL. This can be seen as the objects—such as tools and components corresponding to the assembly step—were identified using an overhead projector. Finally, any anomalies detected during the assembly operation were instantly flagged, with alerts sent to the operator in real time. The



(a) SMIRL vs Graphs at an Overlap rate of 50%.



(b) SMIRL vs Graphs at an Overlap rate of 97%.

Figure 4.32: Confusion matrices after converting time-level inferences to frame-level inferences [70].

app was developed using Python and PyQt6. The associated code and the process to run the GUI can be seen in Appendix C.1.

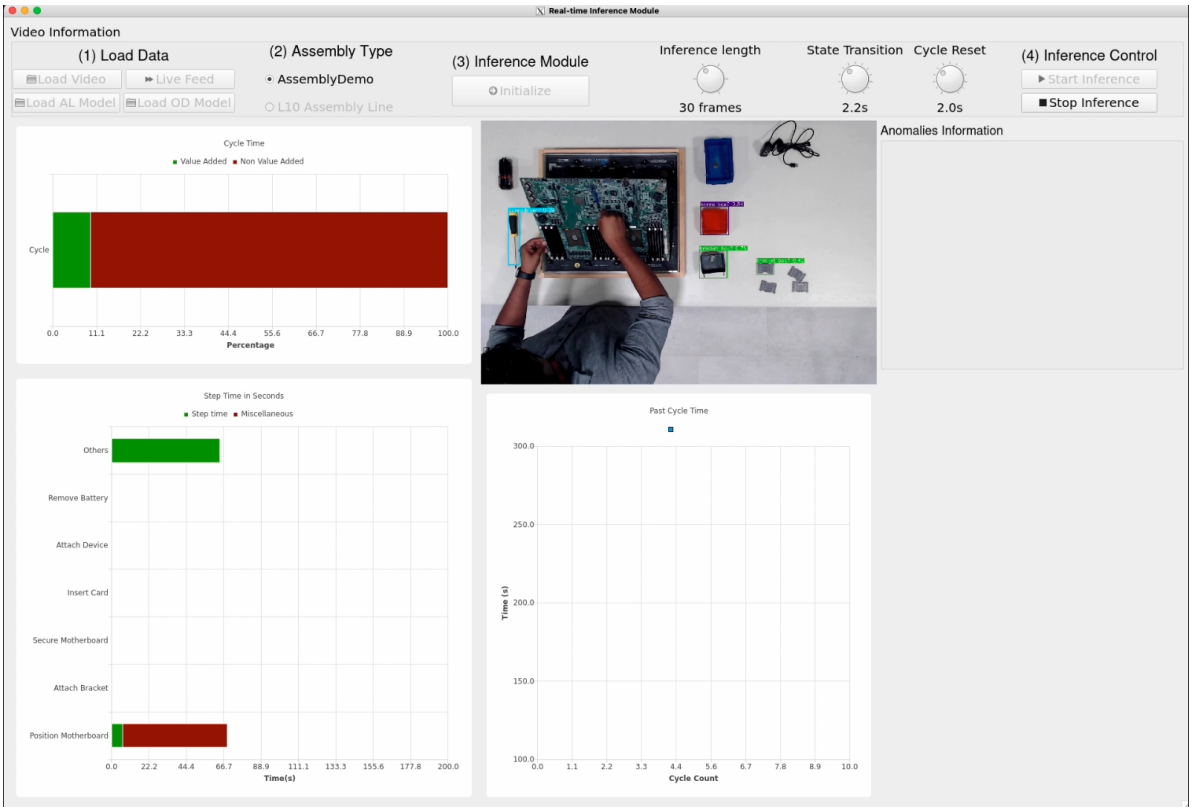


Figure 4.33: GUI developed for real-time assembly monitoring system.

4.8 Conclusion and Discussion

The work done in this chapter was three-fold. First, we developed an algorithm called SMIRL that can detect and localization actions performed by human operators in an assembly workstation, measuring the step time and cycle time and detecting anomalies in the assembly operations performed. The capabilities of SMIRL are listed below:

1. An algorithm called SMIRL to measure the step time and cycle time, along with identifying assembly operation anomalies was developed.

2. To showcase the ability to swap the deep learning model in the inference machine, the VGG16-RollingAverage model was developed alongside C3D-OpticalFlow. A performance comparison was made between the two models and it was found that the use of optical flow information for action detection was beneficial.

3. The algorithm SMIRL was compared against the Two-Stream approach, which was one of the state-of-the-art at the time of this writing. It was found that SMIRL performed comparable to the Two-Stream approach.

4. Finally, the maximum FPS the system could handle was determined by testing the real-time performance of the inference module. The maximum FPS depends primarily on the complexity of the inference module, as it holds the deep learning model. The model C3D-OpticalFlow was determined to be 87 frames per second.

One of the key requirements for SMIRL was to collect data on the NVA activities. This proved to be challenging as it is not possible to explore all possible scenarios for identifying the NVA. Hence in the second phase of this research work, we developed an approach to autonomously identify NVA activities by considering them as OOD instances. The insights obtained from this phase of research are listed below:

1. Using our approach, the NVA activities can be identified with an F1-Score of 0.9744 with no explicit training of the DL models on NVA activities.

2. Our approach was evaluated against the traditional approach of supervised learning of NVA activities. The performance of the system for different proportions of NVA activities was also evaluated.

3. As the system can now handle unseen scenarios, a full-fledged assembly guidance system was developed that can identify the assembly step being performed and guide the operators by identifying the required tools and components.

The final phase of this study was to improve the transparency of modeling the assembly operations by capturing the physics involved in the assembly process. Hence, to that effect, we developed an approach that models the human actions performed in an assembly workstation as spatio-temporal graphs. The insights from this work are listed below:

1. By modeling the actions performed in an assembly workstation as spatiotemporal graphs, the model training time and the inference time were significantly reduced.

2. The physics behind the assembly operation was modeled into nodes and edges for both spatial and temporal graphs. These graphs captured the interaction between the components and humans present in an assembly workstation.

3. The graph-based modeling approach performed on par with SMIRL, having C3D-OpticalFlow as the inference machine, with significantly less computational requirement.

4.9 Future Work

There are several avenues for advancing this research. While graphs have proven effective in capturing the underlying physics of assembly operations, they currently cannot directly measure step time and cycle time for an assembly cycle. Instead, they rely on the state machine developed in our previous work, SMIRL, to aggregate frame-level inferences into time-level inferences.

In future work, the constructed graphs could be modeled as a Markov chain that represents interactions between different components in an assembly workstation and the human operator. The generation of Markov Chains from assembly operations could be an intensive manual process where our graph construction approach could help speed up the process. By translating graphs into Markov chains, we could assign probabilities to each object-to-object and object-to-person interactions, enabling a predictive approach where the likelihood of the operator's next action is anticipated. This could offer two major benefits: first, it would facilitate the integration of the state machine within the graph architecture, allowing for seamless step time and cycle time measurement; second, it could allow for the early detection of potential mistakes, providing alerts to the operator before errors occur. This predictive capability would improve upon SMIRL, where mistakes are only detected during or after the step is performed.

Research in this direction could also support human-robot collaborative assemblies. By predicting human actions, the graph-based approach could enable task generation for collaborative robots (cobots), constrained by human capability and convenience in performing assembly tasks. This synergy between humans and machines could optimize workflow, reduce human error, and create a more adaptable assembly environment that dynamically responds to human actions.

Chapter 5

Assessing the Risk for Lifting Activities

The sensing system developed as a part of the machine monitoring system design in Section 2.2.2 is evaluated in this section. To evaluate the Smart Sensing System designed as a part of developing a condition monitoring system in Section 2.2, a case study was conducted to evaluate the risks involved in manual lifting tasks. As manufacturing transitions from Industry 4.0 to Industry 5.0, a renewed focus is placed on human-centric operations and the well-being of workers [75]. According to the data analysis from the Bureau of Labor and Statistics by the National Safety Council (NSC), about 24% of all non-fatal injuries that led to days away from work were due to back and lumbar injuries in 2019. Further, about 30% of these injuries were recorded among workers involved in manual lifting and material handling tasks [122]. Subsequently, lower back disorders contribute to over $91 billion in health care expenditure and $15 billion towards worker compensation in the United States each year [123, 124]. To expand the capability of our condition monitoring system, in this section, we aim to study a use-case where the wireless sensing system can be potentially used for real-time monitoring of worker ergonomics on a factory floor. The work in this chapter is published in Selvaraj et al. [32].

5.1 Case Study Introduction

The National Institute for Occupational Safety and Health (NIOSH) has published a detailed paper and manual to assess a given lifting operation and guide ergonomics professionals in designing workspaces to minimize injuries [125, 126]. The manual considers various parameters, such as how far a person must reach to grab the object, the weight of the object being lifted, and so on, to assign a score to the lifting task. Based on the score, the risk of injury can be assessed, and necessary changes can be implemented. For this case study, the procedure detailed in the NIOSH lifting applications manual was followed and is explained below.

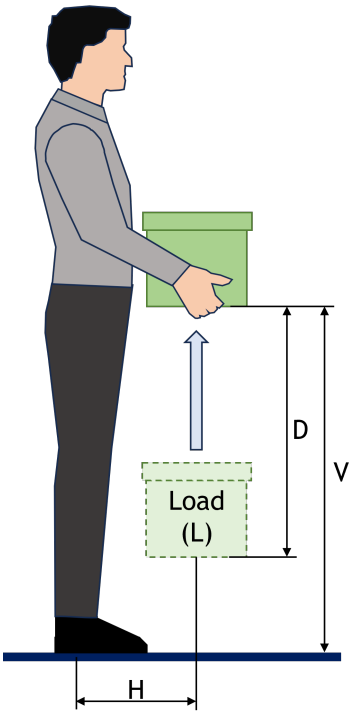


Figure 5.1: Geometric parameters used to determine coefficients in the lifting equation [32].

$$RWL = LC \times HM \times VM \times DM \times AM \times FM \times CM \tag{5.1}$$

where RWL is the recommended weight limit, HM is the horizontal multiplier, which is dependent on how far forward the person needs to reach to grip the object to be lifted (H), VM is the vertical multiplier which is related to how far up from the group the object needs to be lifted (V), DM is the distance multiplier which is based on the vertical distance between the start and end points of the lift (D), AM is the asymmetry multiplier which is dependent on how much the body needs to twist during the lift, FM is the frequency multiplier which is related to the duration of the lifting task and number of lifts to be performed in a minute, and CM is the coupling multiplier which is dependent on how easy or difficult it is to hold the object to be lifted. The RWL is the weight of the load nearly all healthy workers can lift over a substantial period without an increased risk of developing lifting-related lower back problems. The RWL is defined for a specific set of task conditions. The distances used to determine HM, VM, and DM are shown in Figure 5.1.

$$LI = \frac{L}{RWL} \tag{5.2}$$

where LI is the lifting index, a relative estimate of the physical stress associated with a given manual lifting task, and L is the load weight. Additionally, when multiple lifting tasks are involved, frequency-independent recommended weight limit (FIRWL) and frequency-independent lifting index (FILI) can be calculated using equation 5.1 and 5.2 without considering the effect of lifting frequency ($FM = 1$) to get an idea about each lifting task independently.

As the NIOSH revised lifting equation and multiplication coefficients are derived based on different human factors considerations, the equation best applies to an average human under controlled environmental conditions. However, as the real world is far from idealistic, many factors such as age, gender, fatigue, environmental conditions, and physiology can play a role in determining the risk of injury during a lifting task on a given day. To further emphasize this point, a weight one human might find easy to lift might be difficult for another person, depending on their physical condition. Hence, we believe that the NIOSH lifting equation would serve as a good guide for designing a workspace for lifting activities. However, a continuous monitoring system that could consider some of these risk factors would better predict injuries in advance. We hypothesize that these risk factors would influence the lifting task long before the occurrence of an injury and can be identified by continuous monitoring of the lifting action using a variety of sensors such as an IMU or sEMG sensor [127, 128].

In addition to implementing sensor systems, machine learning algorithms have been used to make lifting assessments. Donisi et al. performed tests while varying the lifting weight, displacement, and lifting frequency and captured accelerometer and gyroscope data using a single sensor placed at the lower back [129]. Accelerometer data was found to be more important in predicting the lifting index than gyroscope data. Decision trees, Boosted algorithms, and Multi-Layer Perceptron algorithms had similar accuracy and performed better than other algorithms tested. Similarly, Lu et al. used a 5 IMU sensor system mounted in different locations on the body to develop an algorithm to predict the lifting duration and some lifting parameters such as H, V, and twist (T) in the body [124]. The machine learning algorithms were found to be accurate in determining the

lifting duration and twist but were not as robust when predicting H and V values. In the research by Thomas et al., a methodology to detect lifting activities in IMU sensor data using machine learning algorithms was proposed, and a transfer learning approach was devised to make accurate predictions with a limited amount of data [130]. The models developed performed with over 84% accuracy in classifying a lifting task as low, medium, or high risk. Compared to previous literature, this paper proposes a real-time monitoring system that continuously tracks lifting activities over a fixed period and provides feedback to the user. Although many researchers have tried to develop a methodology to calculate the lifting index in a scenario with multiple possibilities for a given lift, such as scenarios encountered in a warehouse, the current methods have many limitations, including accurately determining the frequency of each type of lift and some of the factors mentioned previously [131, 132]. Investigating commercially available sensor systems (with accelerometer, IMU, and microcontroller), it was found that most of the products for human activity monitoring have one or more of the following shortcomings – low power efficiency, low sampling rate, lack of processing power to perform edge inference, lack of modularity to attach and configure different types of sensors, and challenges associated with customization of software architecture and data processing pipeline. In the first part of this study, we aimed to address these issues and we designed and built the sensor system described in the manuscript. To test the capabilities of the sensor system, we performed the lifting risk assessment study based on the revised NIOSH lifting equation.

Challenges with traditional lifting assessment methods, such as those utilizing NIOSH equations, include the statistical nature of assessments that may overlook individual variations and the limited scope that may not address issues related to the poor lifting form of an individual. However, our sensor system and proposed approach solve these challenges by providing real-time monitoring and personalized feedback. This enables a more dynamic assessment of lifting techniques, reducing the need for extensive experimentation and offering insights tailored to individual behaviors and contexts. Additionally, our system empowers workers and supervisors to proactively identify and address injury risks associated with poor lifting form, thereby enhancing workplace safety and efficiency. The following sections detail the experimental plan, data analysis framework, and results.

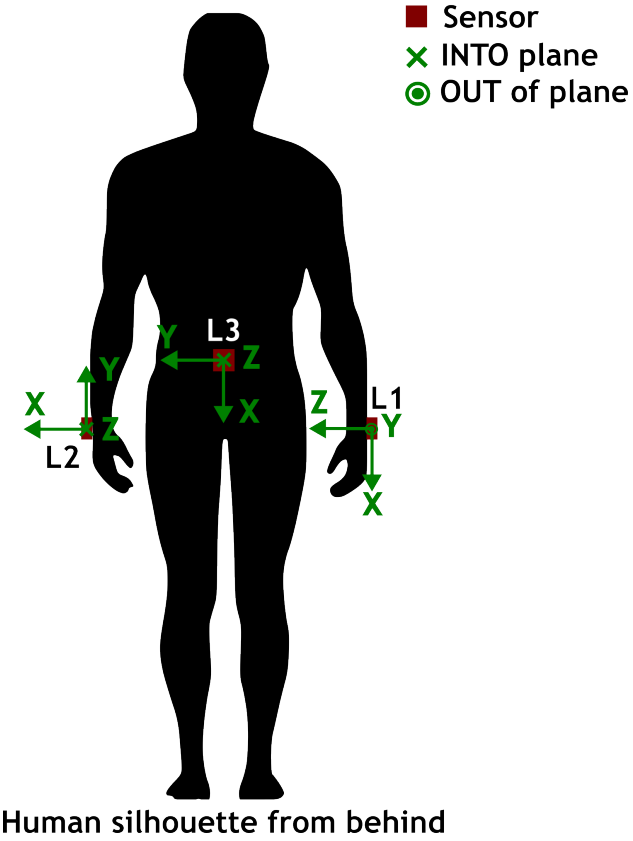


Figure 5.2: Sensor system locations on the human body for worker monitoring [32].

Obj Weight (lbs.)	Hand Location (in.)				Vertical Distance (in.)	Asymmetry Angle (°)		Frequency (Lifts/min.)	Duration (Hours)	Coupling
	Origin		Destination			Origin	Destination			
	H	V	H	V	D	A	A	F	-	C
2.0	16.5	1	16.5	33	32	0	0	3	1	0.9
5.0	16.5	1	16.5	33	32	0	0	3	1	0.9
10.0	16.5	1	16.5	33	32	0	0	3	1	0.9
15.4	16.5	1	16.5	33	32	0	0	3	1	0.9
20.0	16.5	1	16.5	33	32	0	0	3	1	0.9
30.0	16.5	1	16.5	33	32	0	0	3	1	0.9

Table 5.1: Experimental parameters while lifting the box [32].

Obj Weight (lbs.)	Hand Location (in.)				Vertical Distance (in.)	Asymmetry Angle (°)		Frequency (Lifts/min.)	Duration (Hours)	Coupling
	Origin		Destination			Origin	Destination			
	H	V	H	V	D	A	A	F	-	C
2.0	16.5	10	16.5	40	30	0	0	3	1	1
5.0	16.5	10	16.5	40	30	0	0	3	1	1
10.0	16.5	10	16.5	40	30	0	0	3	1	1
15.0	16.5	10	16.5	40	30	0	0	3	1	1
20.0	16.5	10	16.5	40	30	0	0	3	1	1
30.0	16.5	10	16.5	40	30	0	0	3	1	1

Table 5.2: Experimental parameters while lifting the crate [32].

5.1.1 Experiment details

For the lifting risk assessment study, three sensors were attached to the human subjects using velcro straps to the right wrist (sensor L1), lumbosacral region (sensor L2), and the left wrist (sensor L3) as shown in Figure 5.2. The human subjects performed lifting tasks involving lifting a box and crate of different weights. The details of the lifting tasks are listed in Tables 5.1 and 5.2 in the appendix, respectively. Acceleration and gyroscope data were collected simultaneously from all three sensors synchronized at a sampling rate of 400 Hz. To estimate the lifting index based on the NIOSH lifting equation, different multipliers were determined from the NIOSH revised lifting equation applications manual [126] as detailed in section 5.1. A load constant (LC) of 51 and HM, VM, and DM values were calculated using the formulae provided in the reference. As the lift did not involve any twisting, the AM value was considered to be 1. Since the crate had cutouts for handles, it was easier to lift, and the CM was selected to be 1 (good). On the other hand, the box was more difficult to grip due to its large size and lack of handles, and a CM of 0.9 (poor) was assigned. To calculate the single task lifting index, an assumption of a 1 - 2 hour working duration (before a break is taken) and three lifts per minute was made, and the appropriate FM value was selected. For each scenario, ten lifts were performed by each person at a frequency of 3 lifts per minute. Based on Tables 5.4 and 5.3, it can be seen that 30 lb weights for the box and crate were beyond the recommended lifting index of 1 considering the FILI value whereas, when the lifting frequency of 1 - 2 hour working duration with 3 lifts per minute is considered, the 20 and 30 lb weights for the box and crate exceed the recommended lifting index (STLI).

Object Weight (lbs.)	LC	HM	VM	DM	AM	CM	FIRWL	FM	STRWL	FILI	STLI
2.0	51	0.61	0.93	0.88	1	1	25.16	0.79	19.88	0.08	0.10
5.0	51	0.61	0.93	0.88	1	1	25.16	0.79	19.88	0.20	0.25
10.0	51	0.61	0.93	0.88	1	1	25.16	0.79	19.88	0.40	0.50
15.0	51	0.61	0.93	0.88	1	1	25.16	0.79	19.88	0.60	0.75
20.0	51	0.61	0.93	0.88	1	1	25.16	0.79	19.88	0.79	1.01
30.0	51	0.61	0.93	0.88	1	1	25.16	0.79	19.88	1.19	1.51

Table 5.3: Lifting index calculation for the crate [32].

Object Weight (lbs.)	LC	HM	VM	DM	AM	CM	FIRWL	FM	STRWL	FILI	STLI
2.0	51	0.61	0.98	0.88	1	0.9	23.83	0.79	18.82	0.08	0.11
5.0	51	0.61	0.98	0.88	1	0.9	23.83	0.79	18.82	0.21	0.26
10.0	51	0.61	0.98	0.88	1	0.9	23.83	0.79	18.82	0.42	0.53
15.4	51	0.61	0.98	0.88	1	0.9	23.83	0.79	18.82	0.65	0.82
20.0	51	0.61	0.98	0.88	1	0.9	23.83	0.79	18.82	0.84	1.06
30.0	51	0.61	0.98	0.88	1	0.9	23.83	0.79	18.82	1.26	1.60

Table 5.4: Lifting index calculation for the box [32].

5.2 Results

This section presents the results from the model training process. The ability of machine learning models to identify safe lifts was evaluated, followed by a feature importance study to identify optimal sensor locations on the human body and the top features contributing to the model's performance.

5.2.1 Lifting Assessment

Metrics	SVM	Bagging Trees	AdaBoost	MLP
F1-Score	0.8256 ± 0.0147	0.7829 ± 0.0219	0.7844 ± 0.0199	0.8640 ± 0.0125
Recall	0.7641 ± 0.0269	0.6873 ± 0.0219	0.6896 ± 0.0226	0.8523 ± 0.0131
Precision	0.8986 ± 0.0110	0.9096 ± 0.0267	0.9099 ± 0.0232	0.8761 ± 0.0167

Table 5.5: Evaluation metrics for the best performing models [32].

Seven supervised learning models were developed for this assessment. The models were trained using 5-fold cross-validation (CV). The performance metrics used to assess the model's performance were F1-Score, Precision, and Recall. Precision quantifies the number of correct positive predictions out of all positive predictions. Recall quantifies the number of correct positive predictions made from all the correct positive predictions that could have been made. Recall can indicate the missed positive predictions. Maximizing the precision will minimize the number of false positive errors, whereas maximizing the recall will minimize the number of false negative errors. To identify the

best model that can effectively classify the lifts, a total of 306 features were extracted from the time series data spanning both acceleration and gyroscope sensors across three axes and three locations. For each time series data segment, a total of 17 features were extracted, spanning the time domain, frequency domain, and time-frequency domain. The top four models with the highest performance in identifying risky lifts can be seen in Table 5.5 and the F1-Scores for all seven models can be seen in Figure 5.3. The best-performing models were the Multi-layer Perceptron (MLP) Classifier, Bagging Classifier, AdaBoost Classifier with the bagging classifier as the base estimators, and Support Vector Machines (SVM). It can be seen that using the IMU sensors, the quality of the box lifting activity can be assessed with a maximum F1-Score of 0.8871 ± 0.0086.

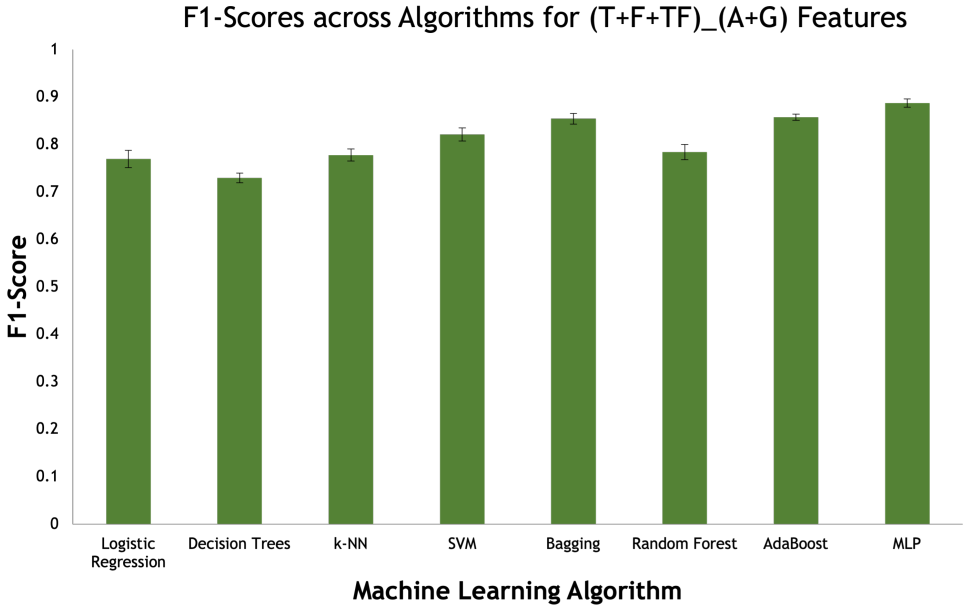
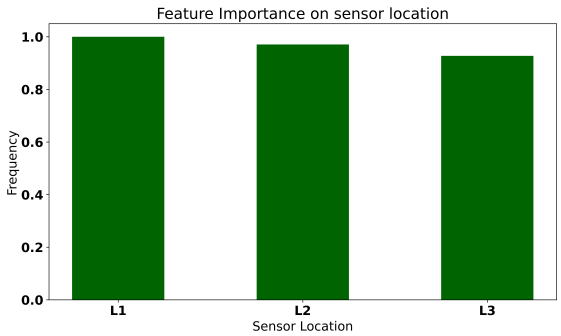
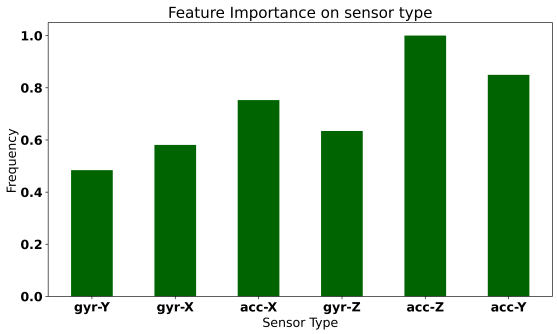


Figure 5.3: Comparison of F1-Scores for the ML models considered in this study [32].

The location of sensors on the human body could significantly affect the performance of ML models in assessing the quality of the lifts [133]. Hence, a sensor location sensitivity study was conducted to identify the ideal regions on the human body that can enable better modeling of the lifting action. The order of importance for the sensor locations was identified using a feature importance study. The feature importance study was repeated for 1000 iterations, considering features from time-domain, frequency-domain, and time-frequency domain features to ensure reliability. The most impactful locations for the sensors were determined to be the wrists, followed by the lumbosacral region (Figure 5.4a). The reason could be twofold: firstly, the most impact from



(a) Most contributing sensor location for lifting risk assessment.



(b) Most contributing sensor type for lifting risk assessment.

Figure 5.4: Feature Importance study to assess the most contributing factor for lifting loads [32].

lifting loads could be felt on the wrists. As the weight of the box directly acted on the wrists, high-frequency signals can be seen in the acceleration data as the weight increases. Secondly, the position of the lumbosacral region as one lifts the load varies depending on the individual performing the lifting task.

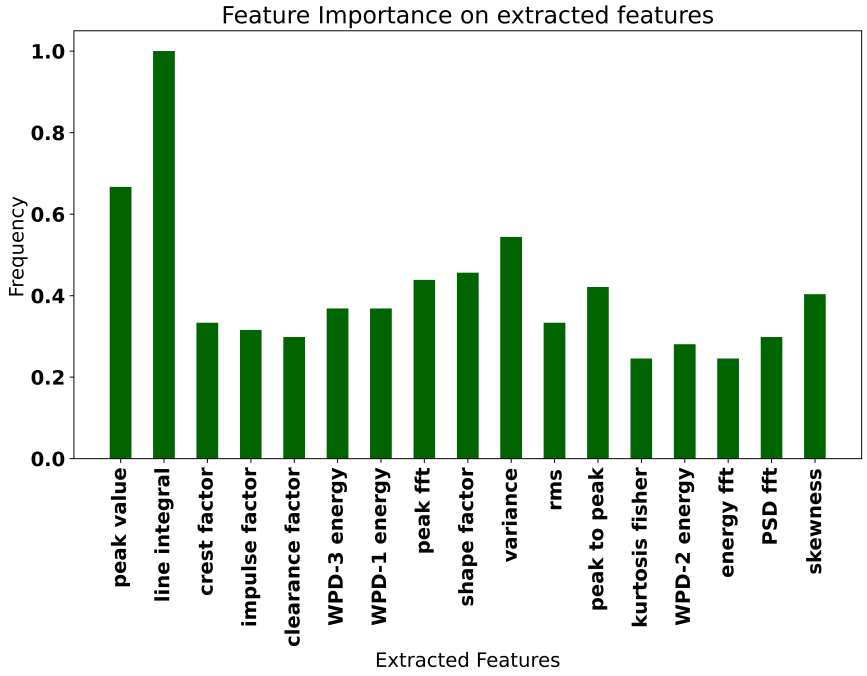
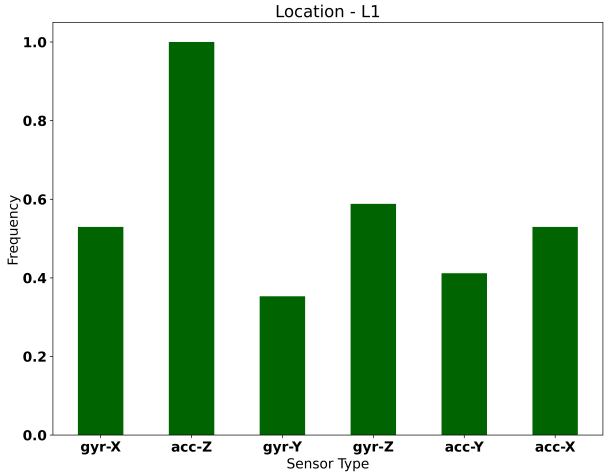
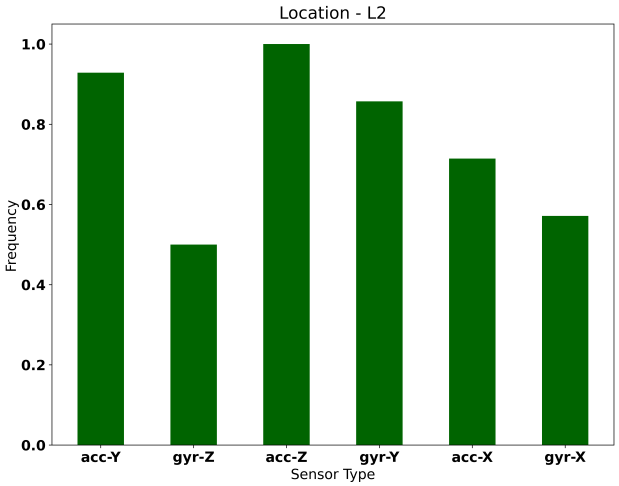


Figure 5.5: Most contributing among the 306 features for lifting assessment [32].

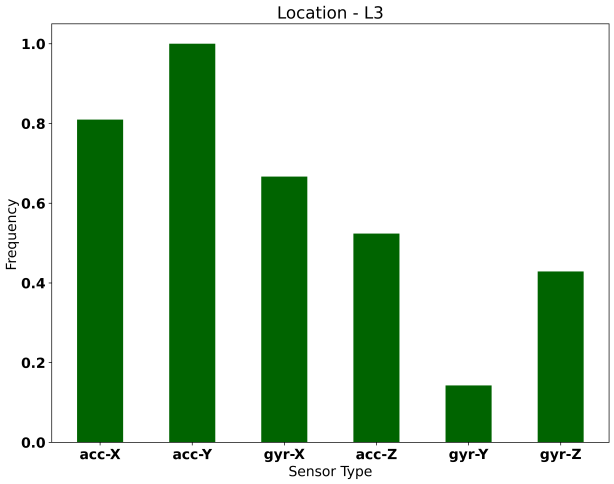
Further, through the feature importance study, the sensor type that had the most impact on the classification of lifts was determined to be acceleration sensors. Among the 306 features considered in this study, the top 10 features that had the most impact on the model's performance



(a) Sensor location L1.

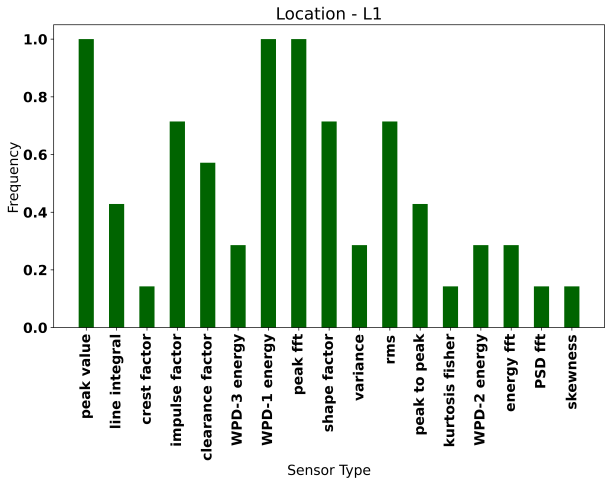


(b) Sensor Location L2.

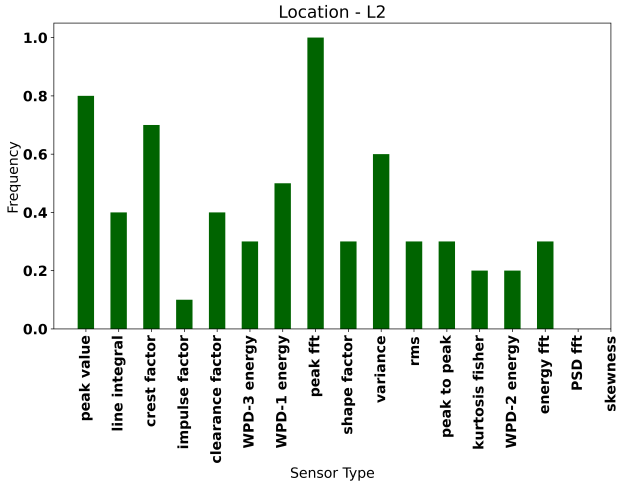


(c) Sensor Location L3.

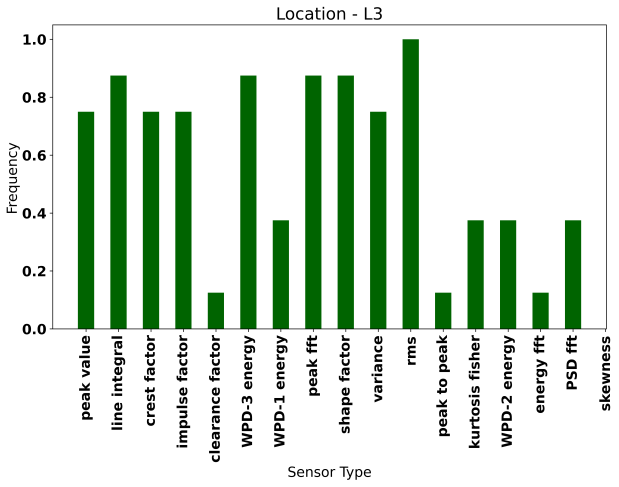
Figure 5.6: Most contributing sensor type categorized by sensor system location [32].



(a) Sensor location L1.



(b) Sensor Location L2.



(c) Sensor Location L3.

Figure 5.7: Most contributing among the extracted features categorized by location [32].

were identified (Figure 5.5). It was noted that the features that had the highest contribution were line integral, followed by peak value and peak-fft. The most contributing feature for model performance could vary depending on the sensor system, as the orientation of the sensor system varies with the sensor location (Figure 5.2). Hence, the feature importance study was broken down to each sensor location, considering the orientation of the sensor. From Figure 5.6, it can be seen that the acceleration sensor type was the most important for all sensor systems, and the axes that go into the frontal or coronal plane were the most important. The coronal or frontal plane is the plane that separates the front (anterior) and back (posterior). The axes flowing into the coronal plane were Y, Z, and Z for the L1, L2, and L3 sensor locations, respectively. Hence, it is important to ensure the weights are lifted in the right way, as was the case in our experiments, where it was ensured that the participants lifted the weights without causing strain on the lower back. Finally, the most contributing feature(s) from the extracted features, considering both acceleration and gyroscope, were determined (Figure 5.7). The feature importance study shows that the most important feature domain was the time domain, as it primarily impacted the performance of the ML models in assessing the quality of lifts.

Extracting features from raw time series data could be time-consuming and, in some cases, computationally expensive, especially for inference at the edge or on resource-constrained devices. Hence, the ML models were evaluated by computing the F1-Score for different feature and sensor type combinations to simplify the deployment of the sensor system for continuous monitoring. In Figure 5.8, where the analysis was performed using only the MLP classifier, it can be seen that the highest F1-Score was attained for the case of features from all domains and the combination of both acceleration and gyroscope sensors, i.e., (T+F+TF)_(A+G). However, the F1-Score of the MLP classifier with only the time domain features and acceleration sensor was very close. Similarly, when all seven models were considered for the evaluation, the F1-Score was the highest when considering all feature domains and sensor types (Figure 5.9). Still, with just the acceleration sensor, the performance was comparable to the best-case scenario across all models. Additionally, it is worth noting that using just the gyroscope sensors leads to the lowest performance across all models. The final analysis involves evaluating the model performance across the three sensor locations. For almost all models, the sensor locations L1 and L3 were the most important, as they had the highest

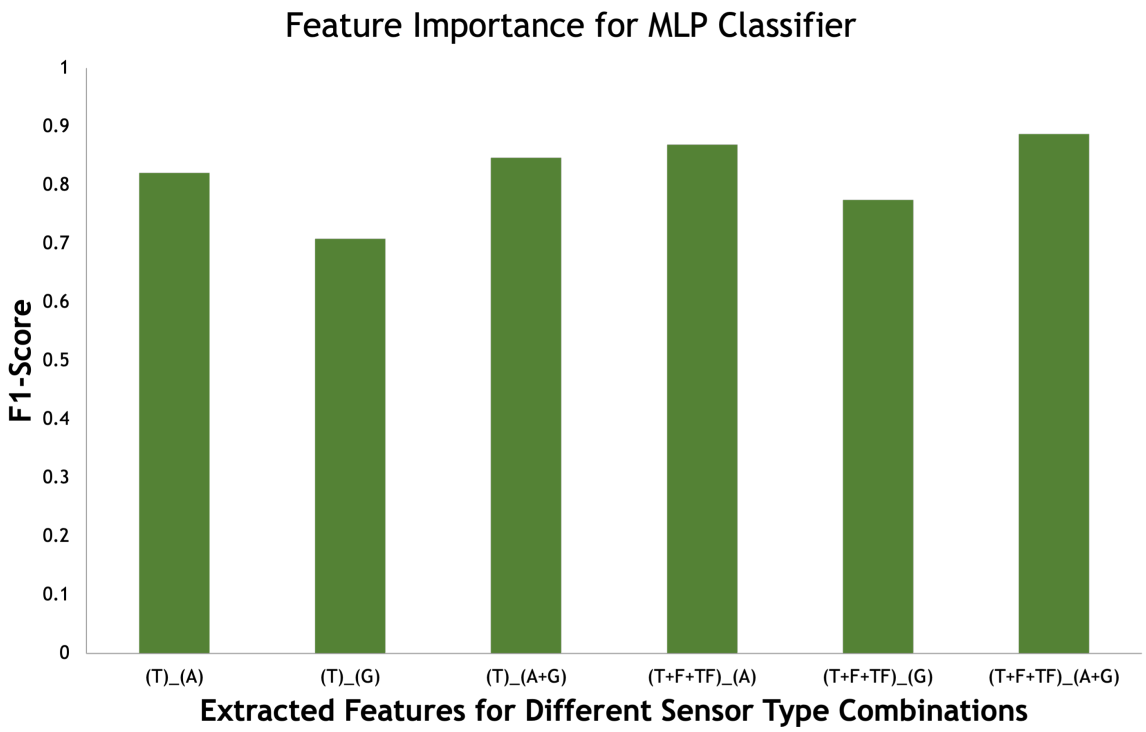


Figure 5.8: Performance of MLP classifier for different extracted features and sensor type combinations [32].

F1-Score when assessing the quality of lifts (Figure 5.10). This validates the insights from Figure 5.4a.

5.2.2 Real-time inference

One of this study's objectives was to enable risk assessment as the lifting activity was performed. In this section, the process of real-time inference to alert the operators of risky lifts is discussed. The inference architecture was designed for both edge and cloud depending on the performance and computational requirements (Figure 5.11). The cloud service used for real-time inference was AWS (Amazon Web Services), and the models trained were hosted using Amazon Sagemaker. At the start of monitoring the sensor systems were continuously polled to collect data synchronously, the data was then transferred to an edge device over BLE where it was preprocessed to extract features from the time, frequency, and time-frequency domains. The extracted features were then sent to the AWS endpoint to assess the quality of lifting tasks. The inference was then sent to

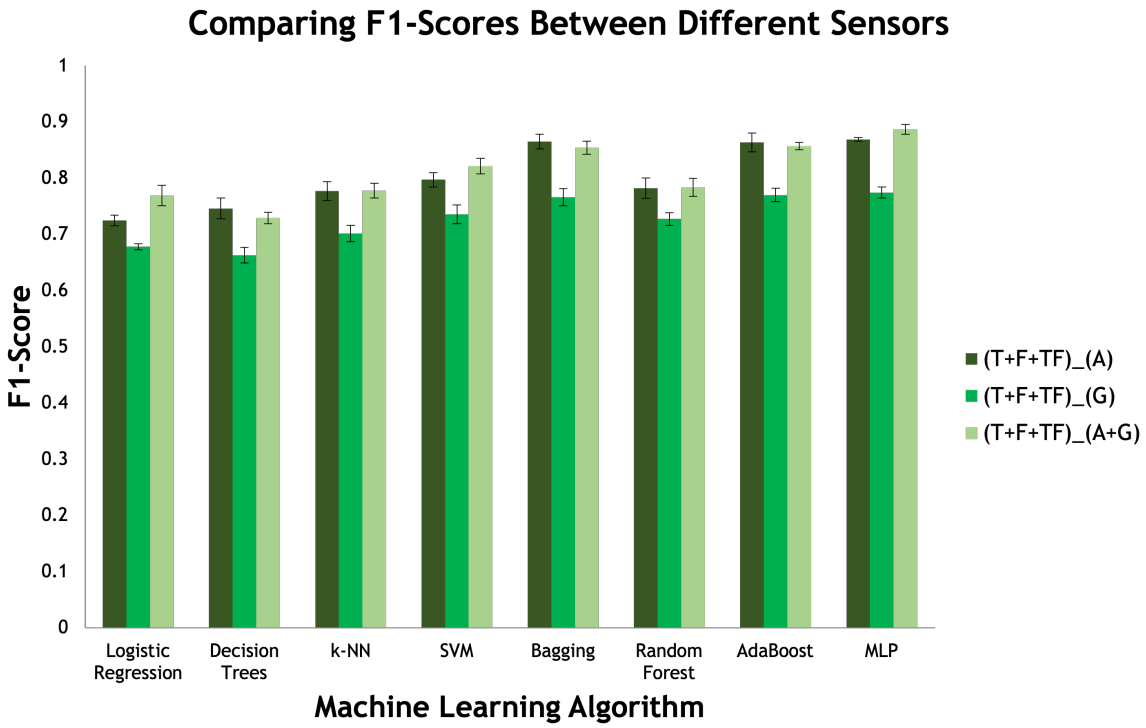


Figure 5.9: Comparison of f1-scores across ML algorithms for different sensor types [32].

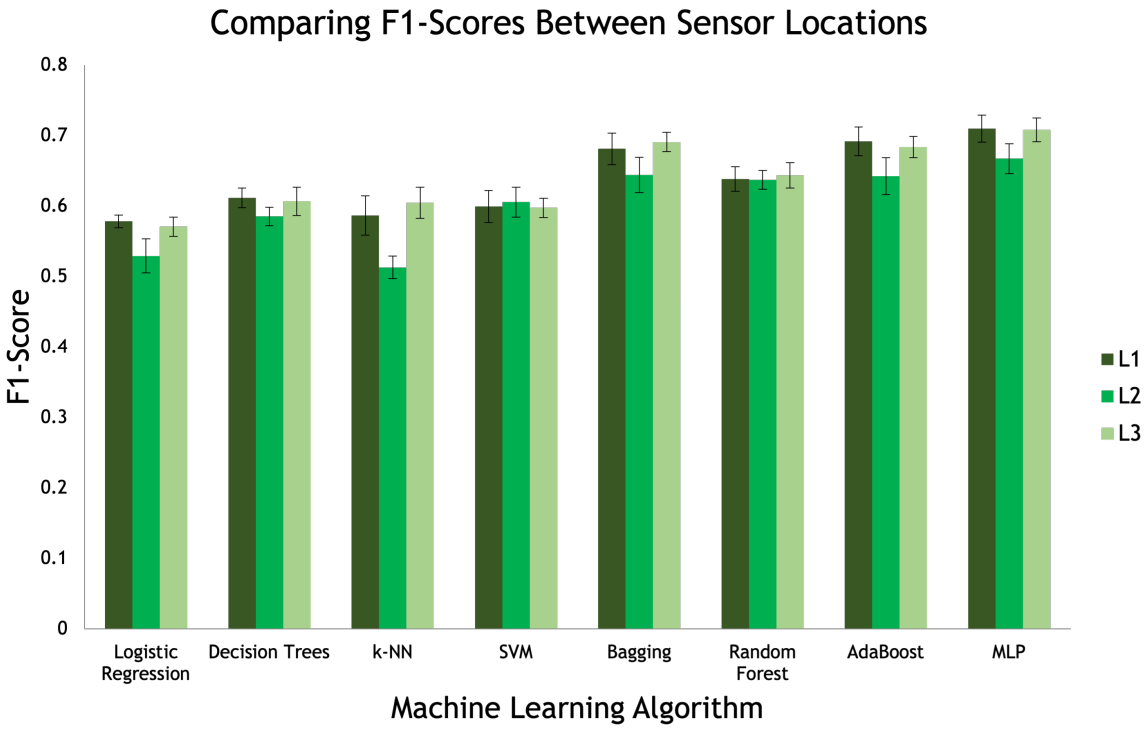


Figure 5.10: Comparison of f1-scores across ML algorithms for different sensor locations [32].

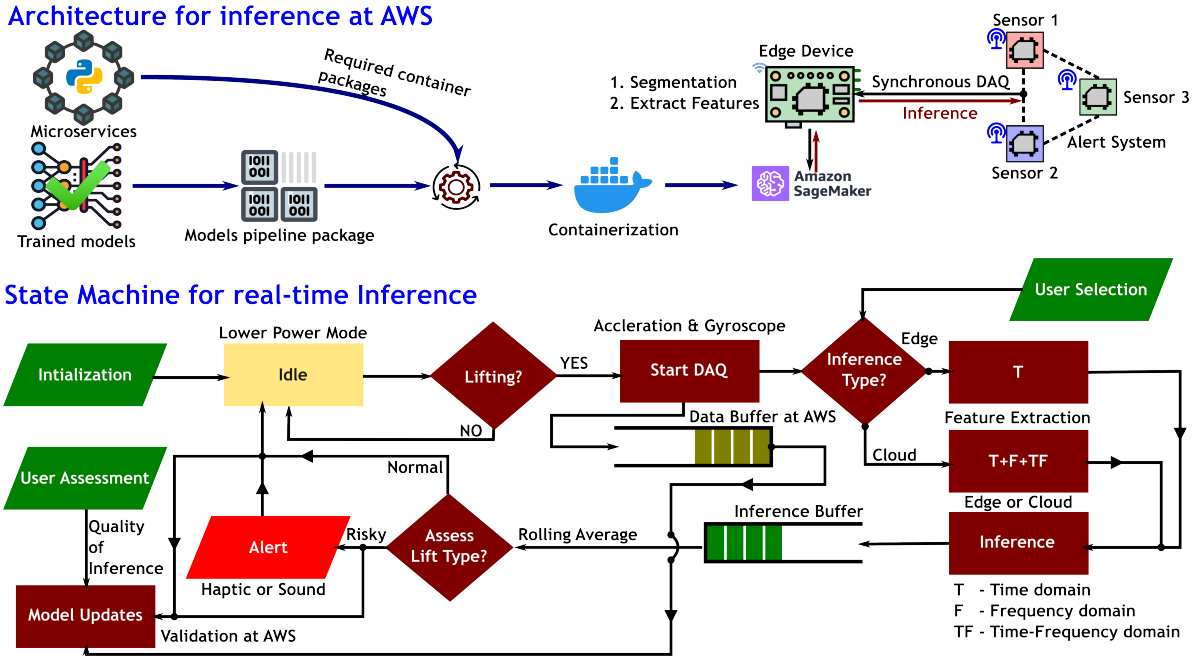


Figure 5.11: Architecture and State Machine (SM) for real-time inference [32].

the edge device, which was transferred to the sensor system that alerts the operator using haptic feedback or a chime. This process was repeated for every lift performed by the operator, and the model inference was performed at the cloud. Cloud-based inference is advantageous regarding resource availability and computation, but it could become expensive if the number of devices requiring ML inference increases. Hence, inference at the edge devices was also implemented as an alternative option. From the feature importance study, it was found that the performance of the lifting assessment models using only the time domain features was comparable to using features from all three domains, as seen in Figure 5.8. Particularly, for the model MLP, by using the only time domain features for acceleration and gyroscope sensor data, the F1-Score was 0.8469, and by using only time domain features for acceleration sensor data, the performance of the MLP model was 0.8210. Whereas the best-case performance obtained by using all three domains of feature extraction across both acceleration and gyroscope sensors was 0.8871. Hence, to reduce the computation load, the inference on the lifting assessment was performed by computing the time domain features on acceleration and gyroscope. The ML model was loaded in the SPI flash of the primary microcontroller, and the inference was made at the primary microcontroller for every lift performed by the operator.

Machine learning is a life-long learning process, in addition to real-time inferencing. The sensor system and the custom-designed application were designed to communicate with the cloud server to update the model parameters. To improve the inference quality, a rolling window segmentation with 75% overlap was performed on the acquired data before extracting the features. The ML model was used to make an inference on each segment, and the results were added to another buffer called the Inference Buffer. The final inference is made by taking a majority vote on the inference buffer. This approach helped improve the reliability of the inferencing process by reducing false positives. Finally, alerts are provided to the operators if the activity performed is risky to prevent further lifts. To continuously update the model parameters, the user feedback on the model assessment of the lifting activity was received through the application or push buttons on the sensor system. The user feedback and sensor data buffer were used to fine-tune the models over time. This was important as the potential risk of lifting activity could vary depending on the individuals, which is not accounted for by the NIOSH equation. The models were fine-tuned for individuals wearing the sensor system through user feedback and continuous updates. Machine learning models have their challenges, the model performance can degrade over time, might require retraining when encountering unseen scenarios, and the performance will largely depend on the quality of the initial training data. But, through the use of the model fine-tuning pipeline in the real-time inference software architecture, the models will be continuously upgraded based on the end-user feedback. We believe that through this approach, a bad model can potentially be improved to make better and more reliable inferences over time.

5.3 Discussion

In this work, a sensor system that can simultaneously and synchronously acquire data from IMU sensors was evaluated. To study the capability and reliability of the sensor system, a case study was conducted to assess the quality of the lifts performed by human operators. The sensor system was designed to be compact and was attached to the wrists and the lumbosacral region of different individuals performing lifting tasks. The inference from this study is as follows:

- The sensor system assessed the quality of lifts, i.e., can say whether a lifting operation is

"safe" or "risky" from acceleration and gyroscope sensor data with a maximum F1-Score of 0.8871. The best-performing model was identified as Multi-Layer Perceptron among the seven models considered for this study.

- Performed elaborate feature importance study to understand better the factors contributing to lifting activity assessment. The study was performed across the extracted features, sensor locations, sensor type, and sensor axes.
- Implemented a real-time monitoring system that can continuously monitor lifting activity performed by individuals and provide feedback to prevent risky lifts.
- Finally, a state machine was designed to receive user feedback and continuously update model parameters. The fine-tuning process would enable the model to better cater to the variations associated with each individual, such as anthropometric variations, etc.

Chapter 6

Conclusion

In this chapter, we start by summarizing the work done in my research followed by setting the stage for future research on a broader perspective.

6.1 Summary

Our research began with the development of a physics-integrated condition monitoring system for manufacturing machines. In the first phase, we collaborated with a cold-forging industry to retrofit a legacy cold heading machine with sensors and custom-designed DAQ systems, the related work can be seen in Chapter 2. These DAQ systems were custom-designed and engineered by us for straightforward retrofitting and designed to support long-term studies. Additionally, we built a complete software infrastructure to streamline the deployment process for the sensing system in industries. This effort resulted in a real-time condition-monitoring system capable of detecting and identifying faults in the cold heading machine, ultimately reducing scrap rates significantly. To promote broader adoption in industrial settings, we developed an anomaly detection system that does not require defective data instances, simplifying the implementation of condition monitoring for manufacturing machines. This work verified our hypothesis that the operating conditions of manufacturing machines, particularly fault conditions, can be reliably identified in real-time.

To test the universal applicability of our approach, the models and data processing techniques were also evaluated against an open-source dataset from Paderborn University. The developed technique was applied to create a condition monitoring system for detecting bearing defects, and both models—the DCNN and the 1DCNN-LSTM—demonstrated high performance in classifying these defects. This validation highlights our approach's capability to be deployed across different systems for effective condition monitoring. Our studies further confirmed that while DL models are effective for condition monitoring applications, a truly robust system must be versatile enough to operate across multiple machines of similar types and processes without requiring retraining or

additional data. In other words, there should not be any requirement to retrain or create data for deploying condition monitoring models to govern a process that has the same underlying physics. This introduces the critical domain of knowledge transferability in our study, which is defined as the capability to transfer knowledge across machines of similar types and processes—essentially the generalization ability of deep learning models. However, constraints in knowledge transfer can arise from stochastic variations related to equipment setup, sensor calibration, and environmental factors such as temperature fluctuations, which affect sensor output sensitivity on the shop floor. Two approaches were explored in our work to enable knowledge transferability. The first involved transfer learning, where data from similar machines was used to fine-tune the deep learning model weights for enhanced knowledge transfer. The second approach focused on domain adaptation, where an adversarial approach for aligning the source and target domain constrained by semi-labeled defect classes (common in fault detection) was developed and evaluated.

Retrofitting machines for condition monitoring can be expensive, and selecting the optimal sensor locations is essential to ensure effective monitoring. Sensor placement significantly impacts fault detection capability, yet positioning sensors close to the event source can be challenging. In our case, the forces and high temperatures in a cold heading machine make it impractical to place sensors in close proximity, as doing so could damage the sensors or impair their detection capabilities. Additionally, altering the machine's design to accommodate sensors could compromise its structural integrity, leading to potential failures. Therefore, careful consideration of sensor location is crucial for effective retrofitting and condition monitoring. This challenge prompted our work in Chapter 3, where an alternative approach to condition monitoring was explored through monitoring the energy consumption of machines. In this work, we focused on a CNC machine instead of a cold forging machine for several reasons: (i) CNC machines provided greater accessibility, allowing for a more nuanced and detailed study; (ii) the machine's computer control system enabled us to correlate energy consumption patterns with specific machine actions directly.

Phase I of this work involved developing an Equipment State Identification system, which utilizes energy consumption patterns to distinguish between different operating states of the machine. This led to the creation of a custom-designed G-code interpreter application, which we developed to parse G-code and associate it with energy information from DAQ systems. This tool will eventually

support predictive capabilities, enabling the system to forecast energy consumption based on the G-code instructions. Building on the insights from Phase I, Phase II focused on developing an anomaly detection system for manufacturing machines. This system leverages energy consumption data to identify and classify real-time operational anomalies. The Phase II work culminated in a comprehensive anomaly detection system that spans the entire lifecycle—from data collection, model development, and training to evaluation, deployment, and continuous monitoring, thereby facilitating life-long learning. At the time of writing, this system has been deployed to monitor the condition of an ultra-precision CNC machine at the Manufacturing Innovation Network Laboratory (MINLab).

Additionally, across Phases I and II, we conducted a feature importance study to identify key parameters and features in power, energy, and 3-phase AC data that are essential for effective energy monitoring. These findings lay the groundwork for future energy monitoring studies, providing valuable guidance on which data features contribute most to accurate and insightful condition monitoring, energy prediction, and energy optimization.

In the final phase of this research, we examined the potential impact of digitalization on human-centric manufacturing operations, which can be seen in detail in Chapter 4. This study was conducted in collaboration with Foxconn in Mt. Pleasant, WI, USA, where the goal was to monitor a manual assembly process in real-time. Our work involved developing an algorithm called SMIRL, designed to measure step time and cycle time and detect anomalies, such as missed steps and sequence breaks, within an assembly cycle. SMIRL was engineered for efficiency, allowing for quick training and deployment. A custom GUI was created to demonstrate SMIRL‘s capabilities and to support the technology’s deployment at Foxconn.

For action detection and localization, the best-performing model at the time was the Two-Stream approach. This approach uses 3D convolution to process video data, extracting both RGB frame information and Optical Flow information and sending them separately for 3D convolution, thus the “two-stream” designation. While the Two-Stream approach was effective, SMIRL outperformed it for action detection and localization and offered additional features, including anomaly detection and precise time measurement. As of this writing, the SMIRL algorithm has been inte-

grated into Foxconn's product, OPTIMO.

Two key challenges arose during the development of SMIRL. The first was the lack of sufficient data to effectively train the model on or identify NVA (non-value-added) activities within an assembly workstation. A significant issue in human-centric processes is that NVA activities are inherently variable and unpredictable, creating an almost infinite number of possible scenarios. Collecting data on all potential NVA activities is impractical due to the vast domain of possibilities, making it challenging for the model to avoid high false-positive rates when encountering unseen scenarios. To address this, we explored OOD (out-of-distribution) detection techniques in deep learning, developing a method that enables SMIRL to identify NVA activities without requiring explicit training on them. This approach allowed SMIRL to generalize better, recognizing atypical activities as NVA without prior exposure. This capability proved highly beneficial for Foxconn, as it removed the need to collect extensive data on NVA activities, streamlining deployment and reducing data collection costs.

The second challenge was related to the transparency of SMIRL when modeling video data to detect and localize actions. In SMIRL, video frames were enhanced by extracting both RGB and optical flow features, which were then fed into a 3D CNN model. Although this approach was effective for action identification and localization, it resulted in a loss of the underlying physics behind the assembly process. To address this limitation, we explored a graph-based assembly modeling approach. In this approach, video information was processed and converted into spatiotemporal graphs. In the spatial domain, the graphs captured interactions between objects and humans at specific time instants, while in the temporal domain, they represented interactions between objects and humans across past time frames. This approach allowed us to model the physics of the assembly process more accurately, representing it as a dynamic sequence of human-object interactions over time. An additional benefit of this graph-based approach was its computational efficiency. The graph-based model required significantly fewer computational resources than handling entire video frames in SMIRL, enabling faster model training and inference. This efficiency makes it a promising alternative for real-time monitoring in high-demand manufacturing environments, where both speed and interpretability are crucial. The performance of the graph-based modeling approach was also evaluated along with testing several hyperparameters that govern the graph construction

process. It was found that the graph-based modeling approach performed on par with SMIRL, and in certain cases it was found to be more effective than SMIRL.

6.2 Future Work

The future directions have been discussed in detail within each chapter, as outlined in Sections 2.7, 3.7, and 4.9. One overarching challenge that is unique to all the above-mentioned domains of research, at the time of this writing, was the limited availability of structured and cataloged data. Unlike sectors such as finance, computer vision, or healthcare, where structured datasets are often well-defined and widely available, the manufacturing industry presents unique obstacles. Each facility may operate different types of machines from different OEMs, follow distinct process standards, or employ specialized, proprietary equipment that makes data standardization difficult.

The lack of standardization, and non-availability, presents two main issues. First, it impedes the development of generalizable AI models. AI models require high-quality data that is labeled and covers a broad range of operating conditions. However, the data that is currently available is often isolated to a specific machine or a process, and maybe isolated to a laboratory or an industry. This would potentially limit the model's generalizability across different scenarios. Without a cataloged database that spans different machines, operating conditions, and fault types, models remain limited in scope, leading to AI solutions that require extensive retraining for each application.

Secondly, the absence of a structure and accessible repository of manufacturing data impedes the scalability of AI-based condition monitoring, predictive maintenance, and process optimization. Inconsistent data collection practices across industries and academic institutions mean that even when data exists, it may not be directly usable for AI applications. For instance, different sensors, sensor locations, measurement techniques, data acquisition hardware, DAQ hardware configuration, and data logging methods might yield data that is difficult to integrate and interpret without extensive preprocessing. Additionally, the data will only exist within a bubble where AI models or techniques developed might not work outside the bubble, significantly impacting the developed techniques' universal applicability.

The question to answer here is how to enable standardization in the data collection process so industries and academic institutions alike can generate and upload data for manufacturing use cases. Addressing this challenge will require collaborative efforts within the manufacturing industry to establish common data standards and sharing frameworks. Our future research efforts, in the overarching scheme of enhancing or improving data collection for smart manufacturing, would involve developing standardized data protocols, defining machine-specific parameters, and implementing consistent data collection practices across facilities that could significantly enhance data accessibility. Additionally, creating open-source or industry-wide repositories where companies can share anonymized data could accelerate AI innovation and faster adoption of AI by industries.

Appendix A

Appendix - Robust Monitoring of Manufacturing Machines

A.1 Supremus

Supremus was an application developed to simplify the interface with the Wireless Sensor Systems. The application was developed using NodeJS Electron, seen in Figure A.1. Using this application the wireless sensor systems can be completely remotely operated. At the time of this writing, it contains all the features required to remotely control real-time monitoring, data logging, and even Over-the-Air (OTA) firmware updates. The objective behind developing this app was to make it user-friendly to deploy remote DAQ or real-time monitoring systems with minimal expertise in designing and writing firmware for embedded systems. The application development process was carefully constructed to be simple for future lab members to take it forward and improve its capabilities.

The capabilities of the app are listed below in detail:

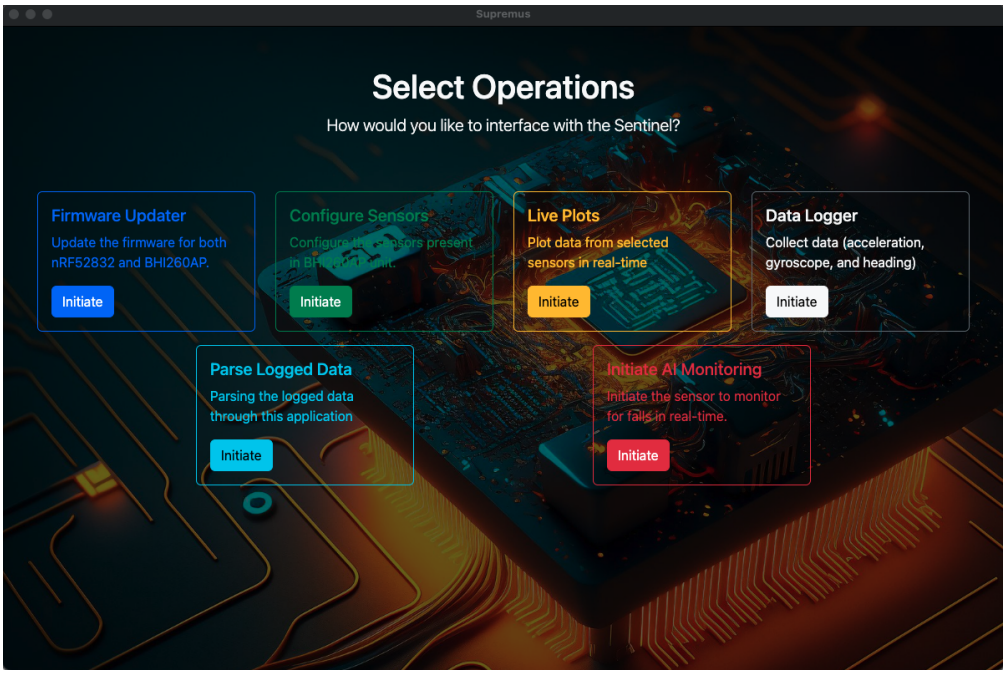


Figure A.1: Application designed to interface with Wireless Sensor Systems - Supremus.

1. **Firmware Updater**: The firmware running on the wireless sensor systems can be updated remotely over BLE.
2. **Configure Sensors**: The sensors on the sensor systems can be configured remotely over BLE. The configuration includes setting the sampling rate, sensitivity, range, etc.
3. **Live Plots**: Live plots can be created for 3-axis acceleration and 3-axis Gyroscope for real-time visualization.

4. **Data Logger**: Depending on sensor configuration, data from acceleration, gyroscope, and orientation sensors can be logged over BLE at a maximum sampling rate of 800Hz, per sensor, per axis.
5. **Parse Logged Data**: To increase throughout over BLE, the data stream from the wireless sensor systems was encoded. This section of the app can be used to decode the data to standard `.csv` format.
6. **Initiate AI monitoring**: ML and DL models can be transferred over BLE to the wireless sensing system for real-time monitoring. At the time of this writing, this section of the app was a work in progress.

The above items correspond to interfacing with a single sensing system. The app was also designed to interface with multiple wireless sensing systems for synchronous DAQ. The app can communicate with multiple (tested on 3) sensing systems for configuration, live plots, and data logging tasks.

A.2 In-situ Vibration Monitoring Sensor System

The schematics, PCB Design, and firmware for the Vibration Monitoring Sensor System are discussed in this section. The board schematics can be seen in Figure A.2 and Figure A.3. For more detailed information on the PCB and the firmware please refer to the repository in GitHub or the zip file in the accompanying USB drive. **Note: On a digital copy of this thesis, you should be able to zoom in to see the contents of the schematics.**

A.3 In-situ Acoustic Monitoring Sensor System

The board schematics can be seen in Figure A.4, A.5, and A.6. For more detailed information on the PCB design and firmware, refer to the repository in GitHub or the zip file in the accompanying USB drive. **Note: On a digital copy of this thesis, you should be able to zoom in to see the contents of the schematics.**

A.4 LabVIEW Program for Experiments at Semblex

The program requirements are listed below:

1. Acquire and log data from two sensors.
2. Collect, log, and plot data at a specified sampling rate without data loss.
3. Output data to a binary ".tdm" file.
4. **Save the data when the collection is in progress, not at the time of stopping the program execution.**

The NI-DAQ configuration for DAQ can be seen in Table A.1. The LabVIEW program can be seen in Figures A.7, A.8. On starting the program, the `while` loop runs indefinitely until

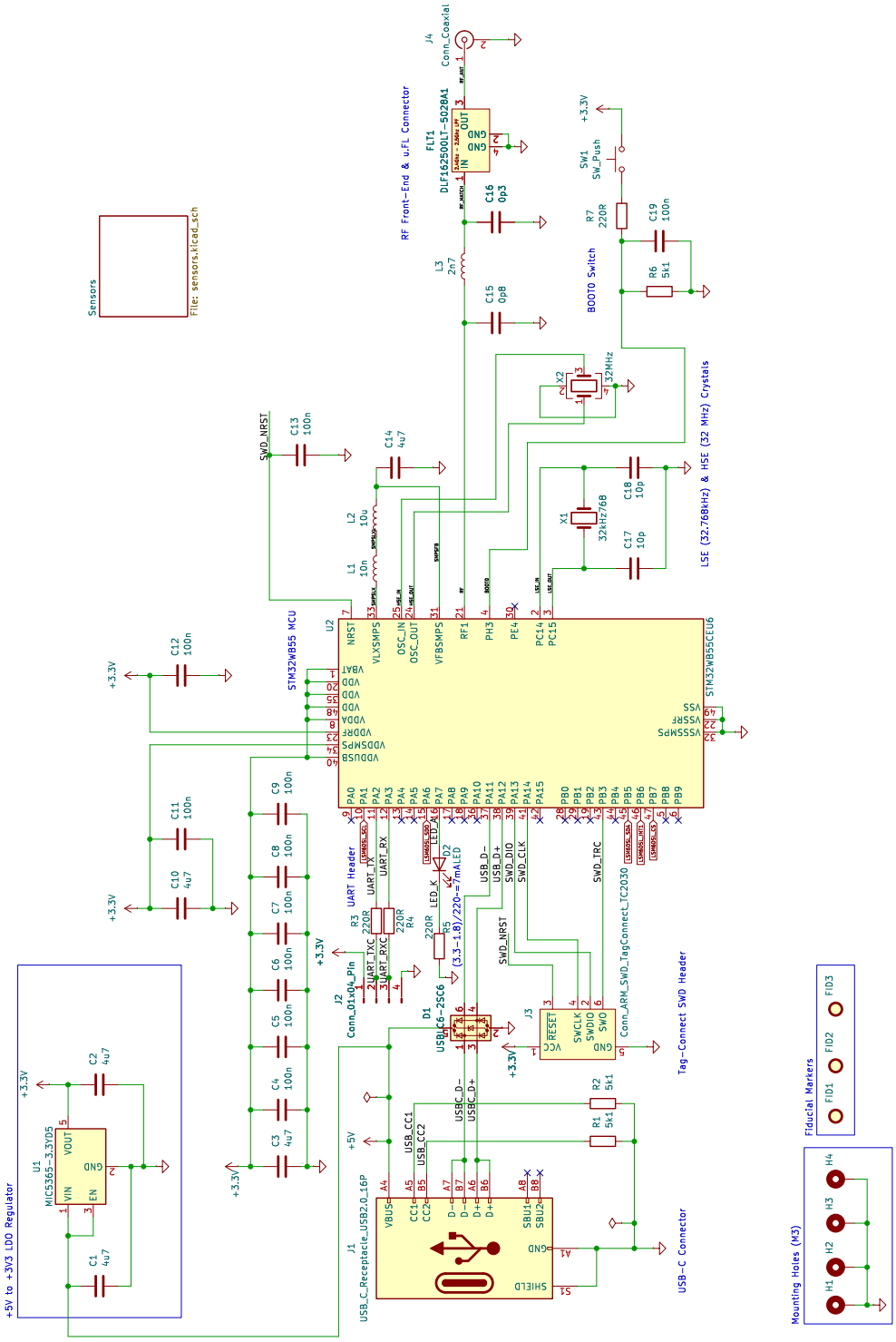


Figure A.2: The main MCU and its schematics (Page-1).

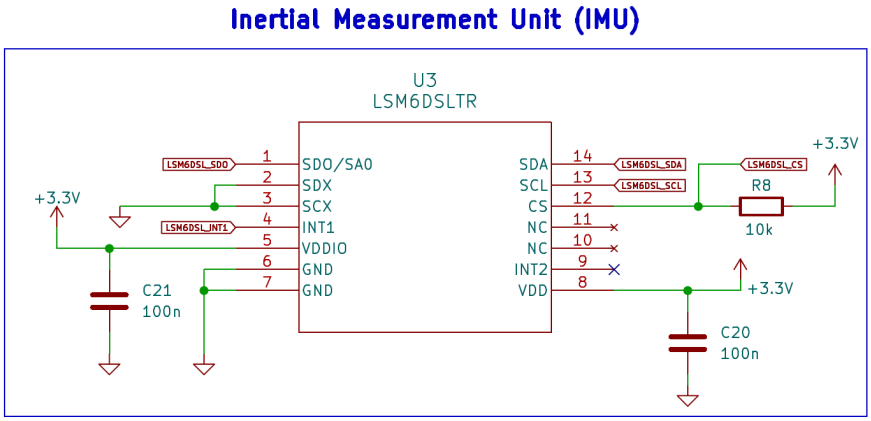


Figure A.3: The sensor's schematics (Page-2).

stopped. The save operation was only done when explicitly pressing the `save` button. Pressing of the `save` button does a data dump, ensuring that the data was logged onto the hard disk. Hence, ensure to press the `save` button to prevent data loss due to errors. The code corresponding to the `save` operation can be seen in Figure A.8. For more detailed information on the PCB design and firmware, refer to the repository in GitHub or the zip file in the accompanying USB drive.

Serial No	Parameters	Setting	Value
1.	Excitation Current	ON	0.002A
2.	Range	ON	+/- 5V
3.	Channels	ON	2 (0 - Die side)
4.	Sensitivity	ON	50 mV/g
5.	Sampling Rate (Required)	SET	8250
6.	Sampling Rate (Actual)	Auto Determined	8533.33
7.	Sample read rate (DAQmx Read)	SET	1000
8.	Coupling Type (DAQmx Channel)	SET	AC (default for IEPE)
9.	Terminal Config (DAQmx Channel)	SET	Pseudo-differential

Table A.1: NI-DAQ configuration.

A.5 Source Code and Other Documents

The source code and documents developed during my research work at MINLab are listed below. This can be found in the USB drive accompanying the thesis document. The USB drive also contains README files describing the content in detail. Additionally, some codebases can also be found on GitHub.

1. In-Situ Vibration Monitoring Sensor System.
   (a) KiCAD PCB design files

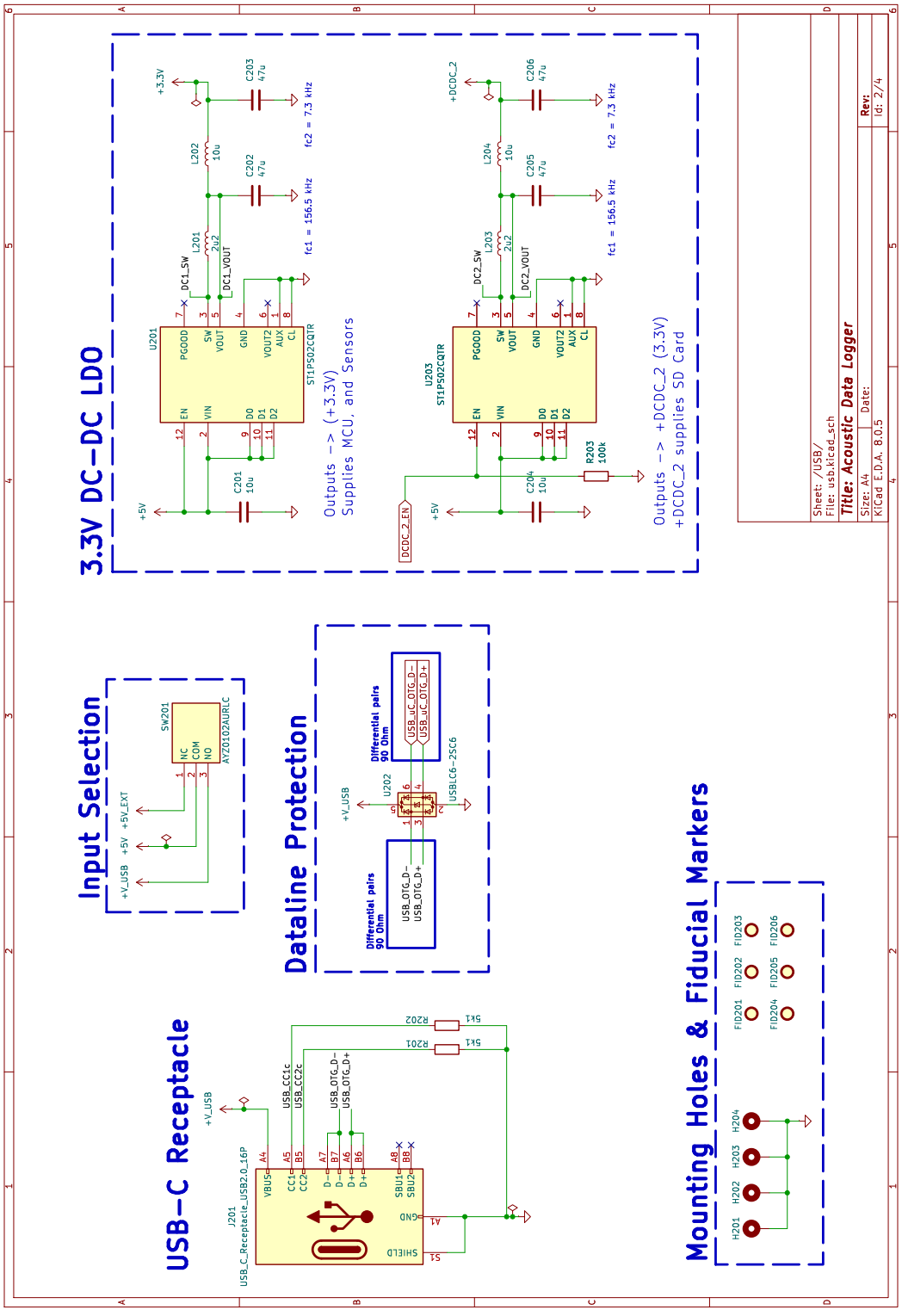


Figure A.4: Schematics for power management (Page-1).

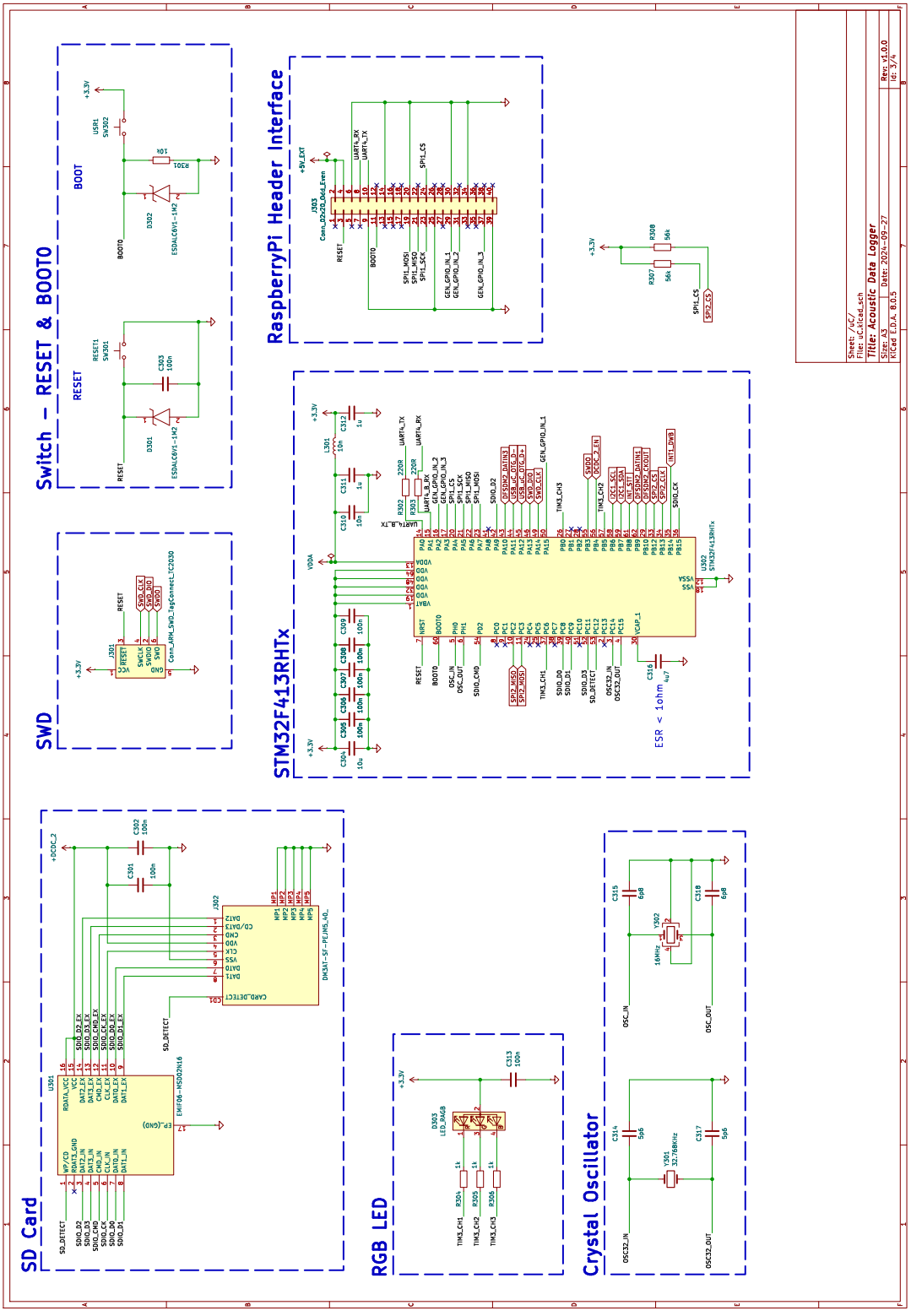


Figure A.5: Schematics for MCU (Page-2).

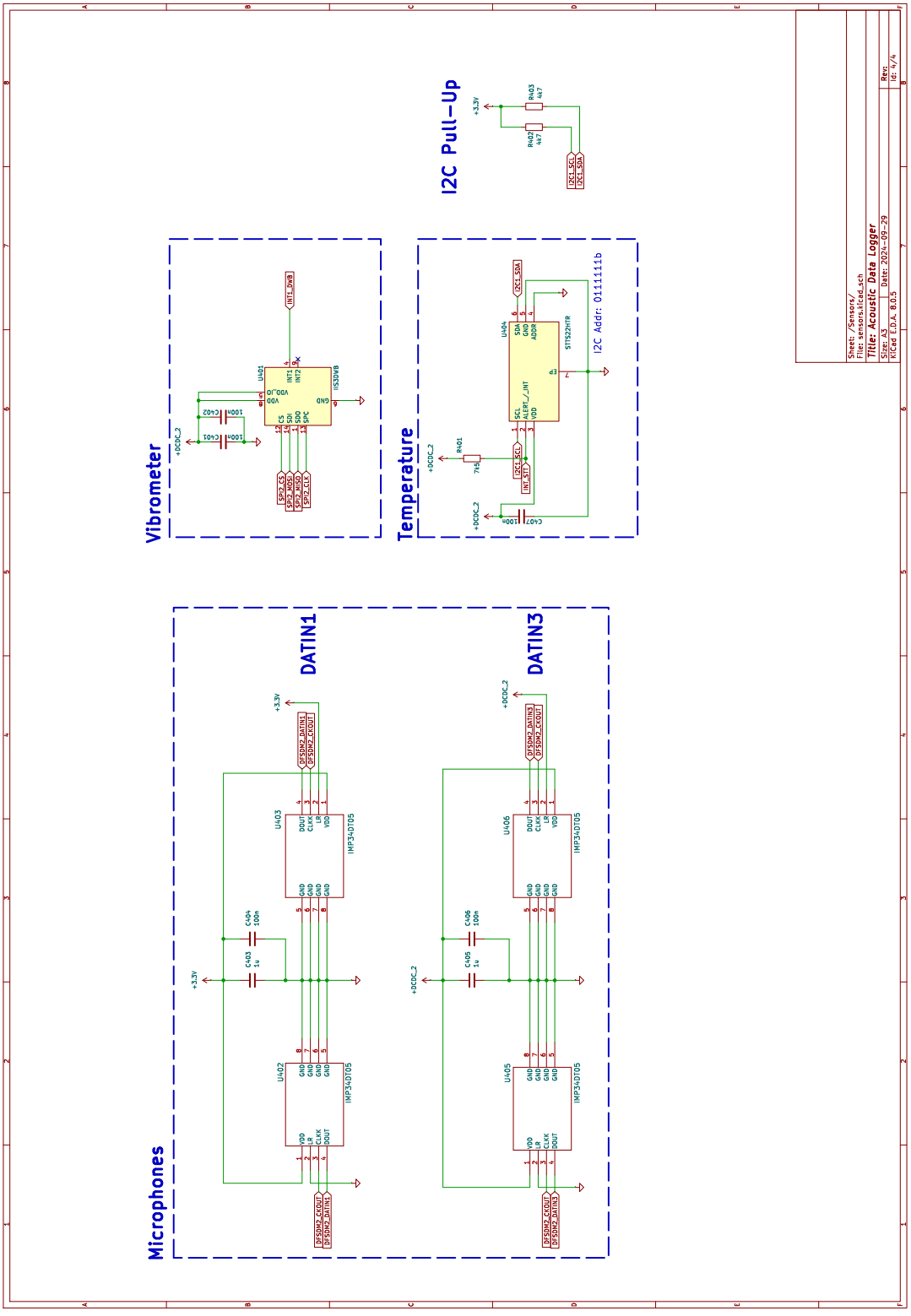


Figure A.6: Schematics for Sensors (Page-3).

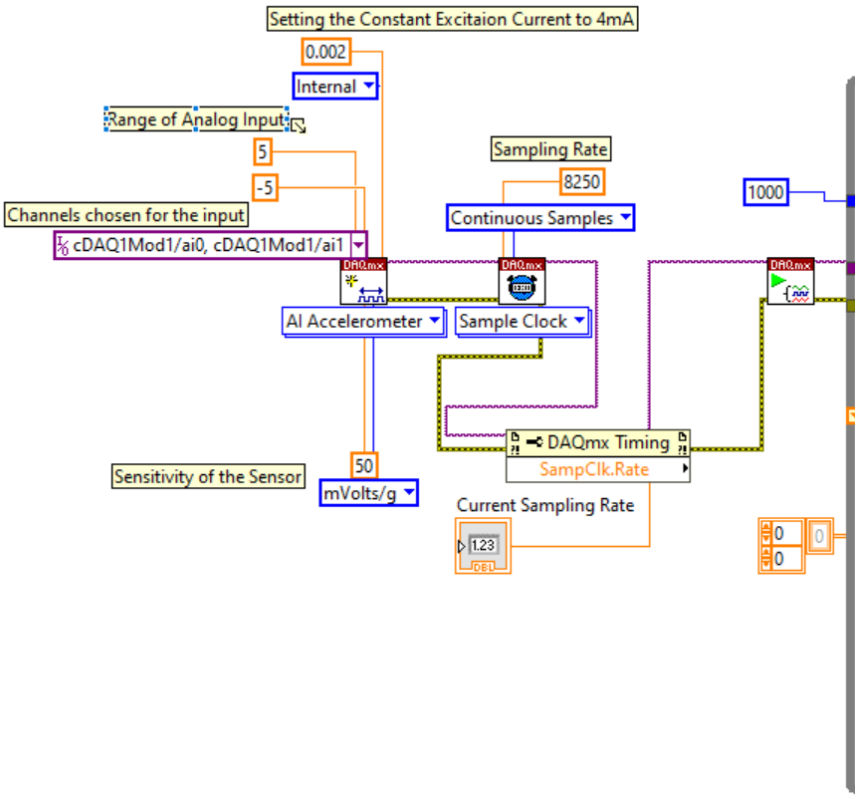


Figure A.7: LabVIEW program for experiments at Semblex - Part 1.

(b) Firmware

2. KiCAD design files for In-situ Acoustic Monitoring Sensor System.
   (a) KiCAD PCB design files
   (b) Firmware
3. Wireless sensor systems for condition monitoring.
   (a) Eagle PCB design files
   (b) Firmware—written using nRFConnectSDK and Zephyr-RTOS
   (c) Supremus - An application to handle multiple wireless sensor nodes for condition monitoring
4. Controlis—An application developed to make the interface with all the hardware developed in my research easy to use.
5. LabVIEW program used for the experiments at Semblex.
6. Stepper motor control firmware for the cold forging mockup designed to replicate the cold forging process at Semblex.
7. DAQ pipeline design for industries. Enabling data collection, logging at remote servers, error handling, and interface with SBCs

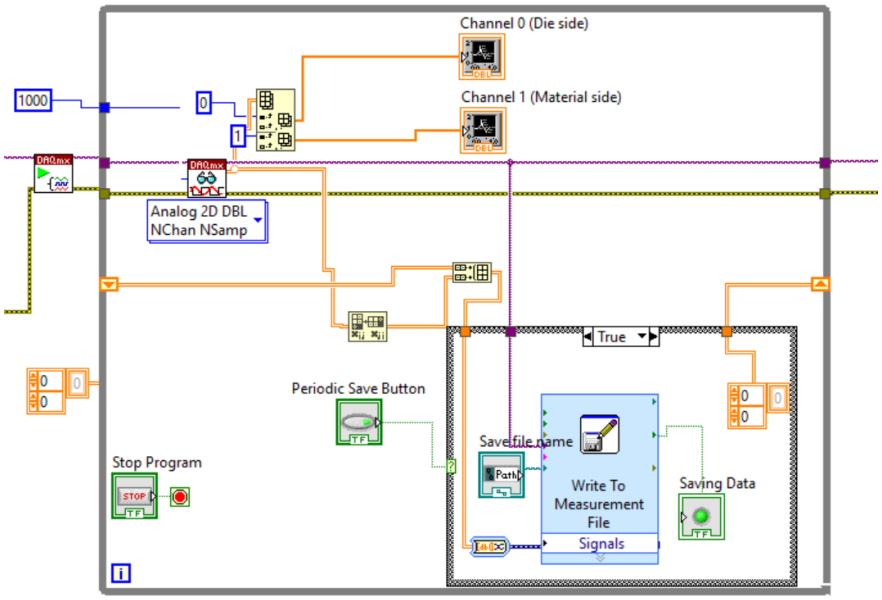


Figure A.8: LabVIEW program for experiments at Semblex - Part 2.

(a) Synchronized GPS-IMU DAQ unit - Hardware setup and firmware.
(b) DAQ pipeline deployed for Semblex
(c) DAQ pipeline deployed at Heungkuk
(d) Mitsubishi PLC UDP interface for DAQ

8. Condition monitoring model development for manufacturing machines.
   (a) Supervised learning model training and evaluation
   (b) Unsupervised learning model training and evaluation
   (c) Domain Adaptation—Semblex and Paderborn bearing dataset
9. Supervised learning and Unsupervised learning algorithm development for condition monitoring of manufacturing machines.
10. Real-time condition monitoring system deployed at Heungkuk Metaltech Co Ltd.
11. Signal processing module for Orange3 data mining tool–Contains our custom algorithms for machine condition monitoring in an easy to use drag and drop interface.

A.6 Acknowledgement

For the PCB designs developed throughout my research, although the boards were custom-designed to meet specific application needs, many design inspirations were drawn from various high-quality online resources. Notably, some of the design inspirations were derived from Phil's Lab and the evaluation boards provided by STMicroelectronics. I would like to acknowledge these resources for providing valuable insights and guidelines that enabled the development of high-speed signal boards.

Appendix B

Appendix - Monitoring Equipment Energy Consumption to Detect Anomalies

B.1 Energy Monitoring Hardware

The PCB board design for the hardware developed to enable high sampling energy monitoring and interfacing with CNC machines is discussed in this section. The schematics design for the hardware can be seen in Figure B.2, B.3, B.4, B.5, and B.6. The hardware designed services three main purposes:

1. Enable high sampling rate DAQ for capturing and measuring the load transients.
2. Interfacing with CNC machines to synchronize the DAQ process with the CNC controller actions.
3. Long-term data collection. Potentially this hardware would be deployed in industries for continuous and long-term data collection to study the impact of time on the models developed for energy consumption monitoring.

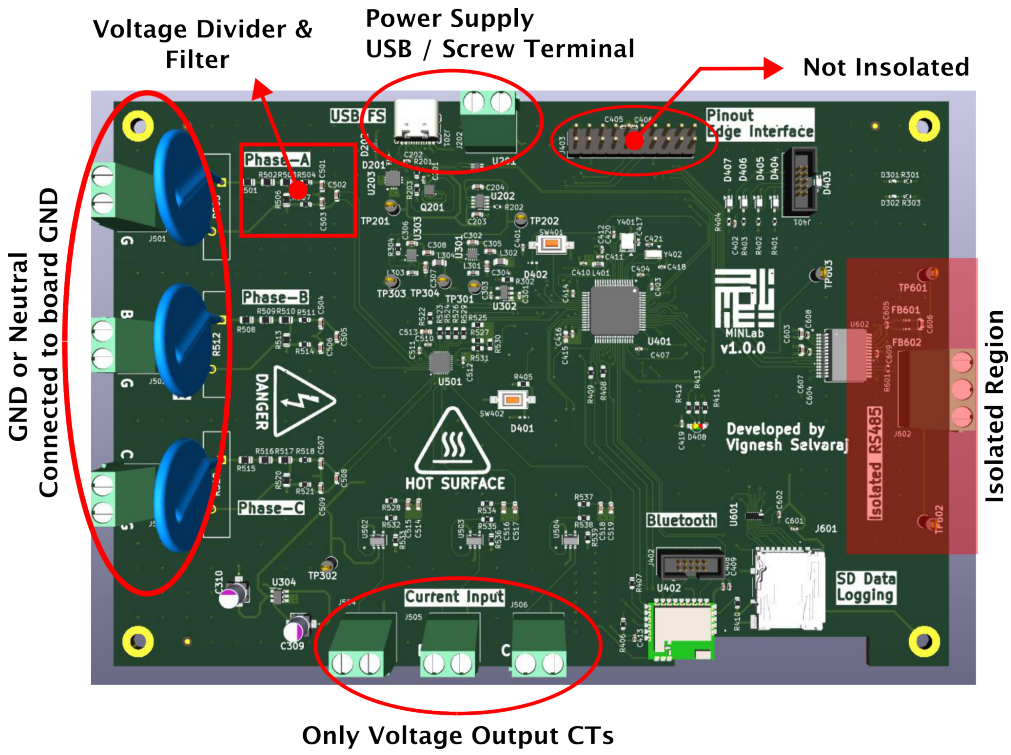


Figure B.1: Safety Instructions for Energy Monitoring Hardware.

B.1.1 Safety Instructions

The hardware developed at the time of this writing was version v1.0.0 and was primarily designed for research and development purposes. Additionally, there are several important safety considerations when working with this device, which are outlined below:

1. High voltage is directly fed into the board, and a voltage divider is used to reduce the input voltage to a measurable range for the ADC.
2. Ensure that the device is NEVER used without the external case to prevent accidental contact with high-voltage regions on the board.
3. The input voltage must NOT exceed 280V.
4. Ensure that the ground and live wires for the input are connected to the correct terminals.
5. Current measurement must be performed using voltage-output CTs (0 - 0.3333 Vac) that do not require burden resistors.
6. The device is powered via a USB port. NEVER plug the USB port into a PC when high-voltage leads are connected to the board. However, if only the CTs are attached, the USB port can be safely connected to a PC.
7. When manufacturing new boards, ensure that the resistors on the high-voltage input meet the required specifications or schematics, and verify their performance.
8. The RS485 interface to this hardware is isolated, meaning the high voltage from the measurement end cannot creep into the RS485 interface. Hence, the RS485 can be used to interface with other computers even when the high-voltage leads are attached to the board.
9. There are two ways to supply power to the board. DO NOT plug in both power supplies, use only one at a time.

The version v1.0.0 of the hardware with the silkscreen indicating the safety considerations can be seen in Figure B.1.

B.2 Application - Machine Whisperer

The application was developed to enhance the way data was collected and labeled. Through this app, the user interacts with the Computer Numerical Control (CNC) machine, and during every interaction, the data from the CNC controller and energy loggers are collected, labeled, and logged. In this section, the application features and the interface were discussed. In Figure B.7, the user section of the app can be seen. The user section contains the user database, where one can select the user who will be operating the machine.

Once the user is selected, the app will enter the tool selection section, which can be seen in Figure B.8. Here, a backend database is maintained on all the available tools. You can either select a tool based on the Tool ID or create a new tool. The new tool creation section will be intuitive to people who have experience operating CNC machines, as it would require information on the

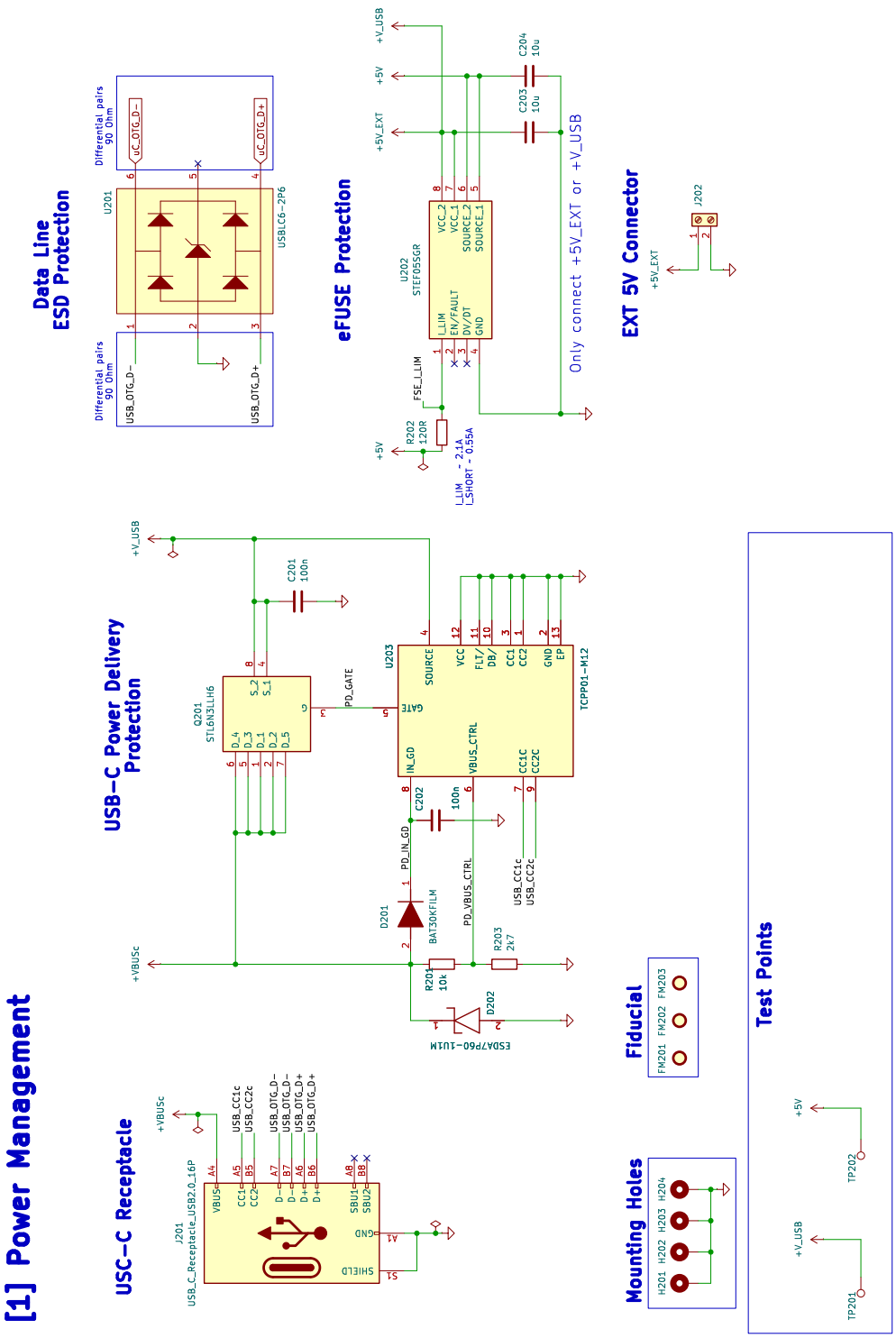


Figure B.2: Schematics for power management (Page-1).

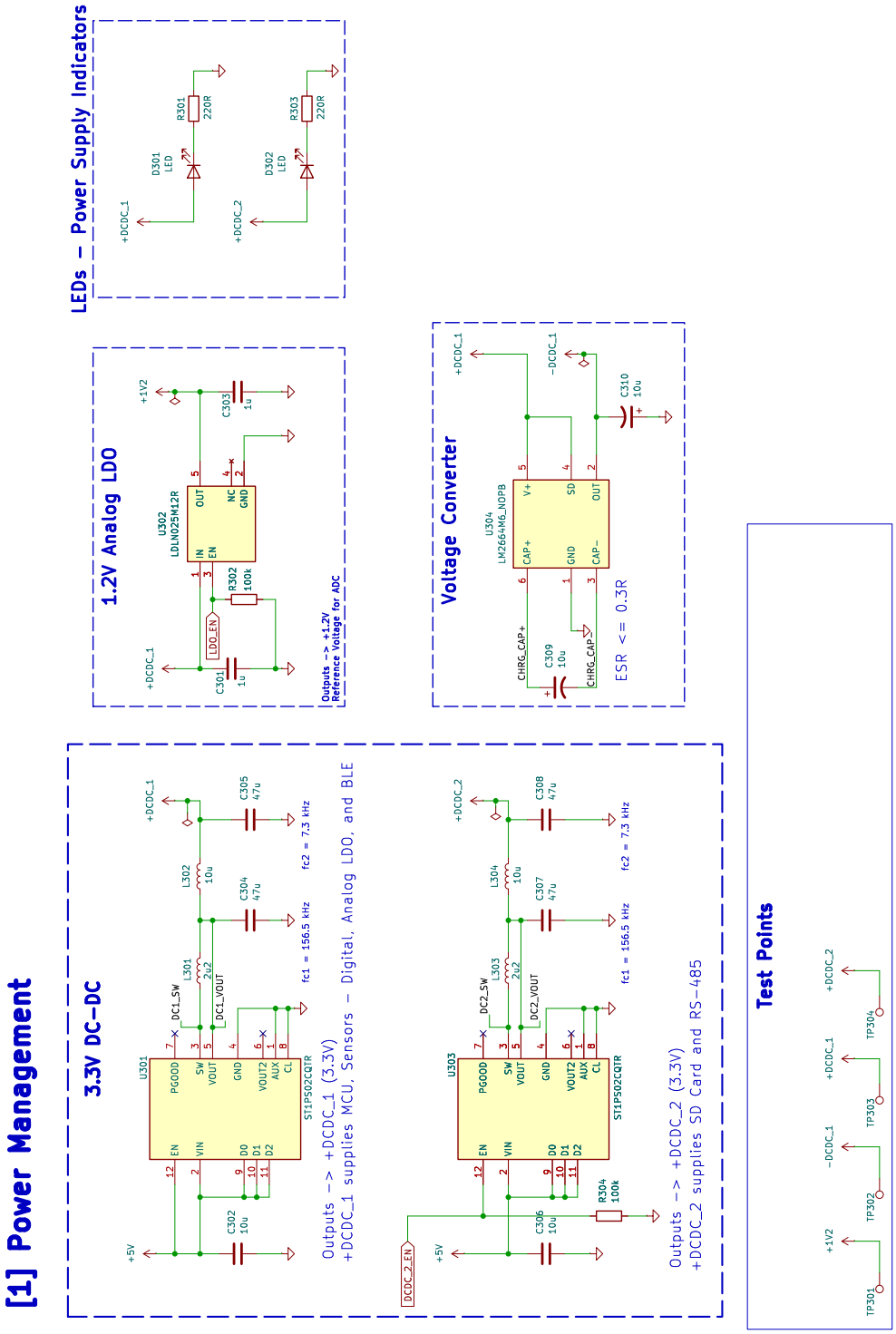


Figure B.3: Schematics for power management (Page-2).

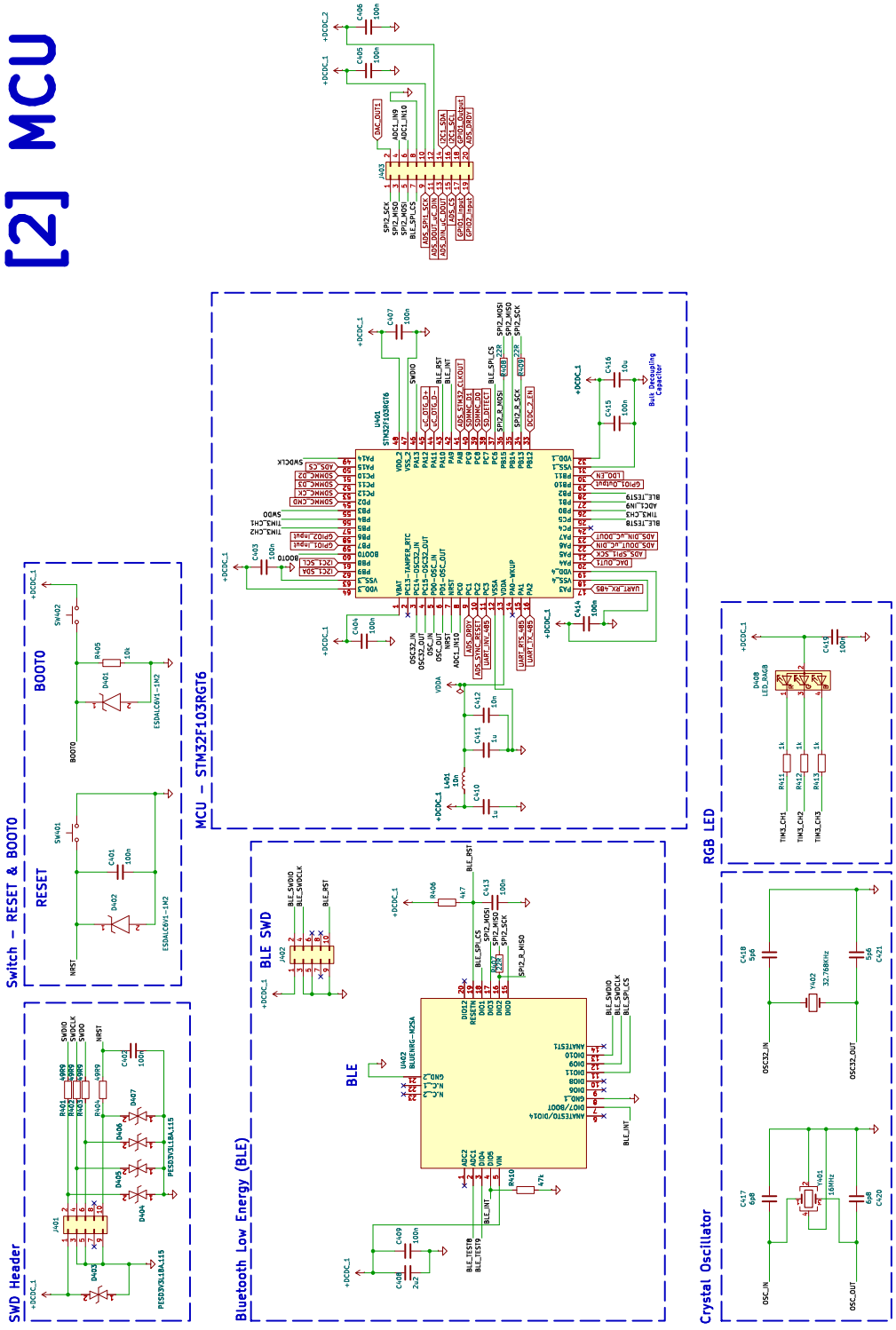


Figure B.4: Schematics for MCU (Page-3).

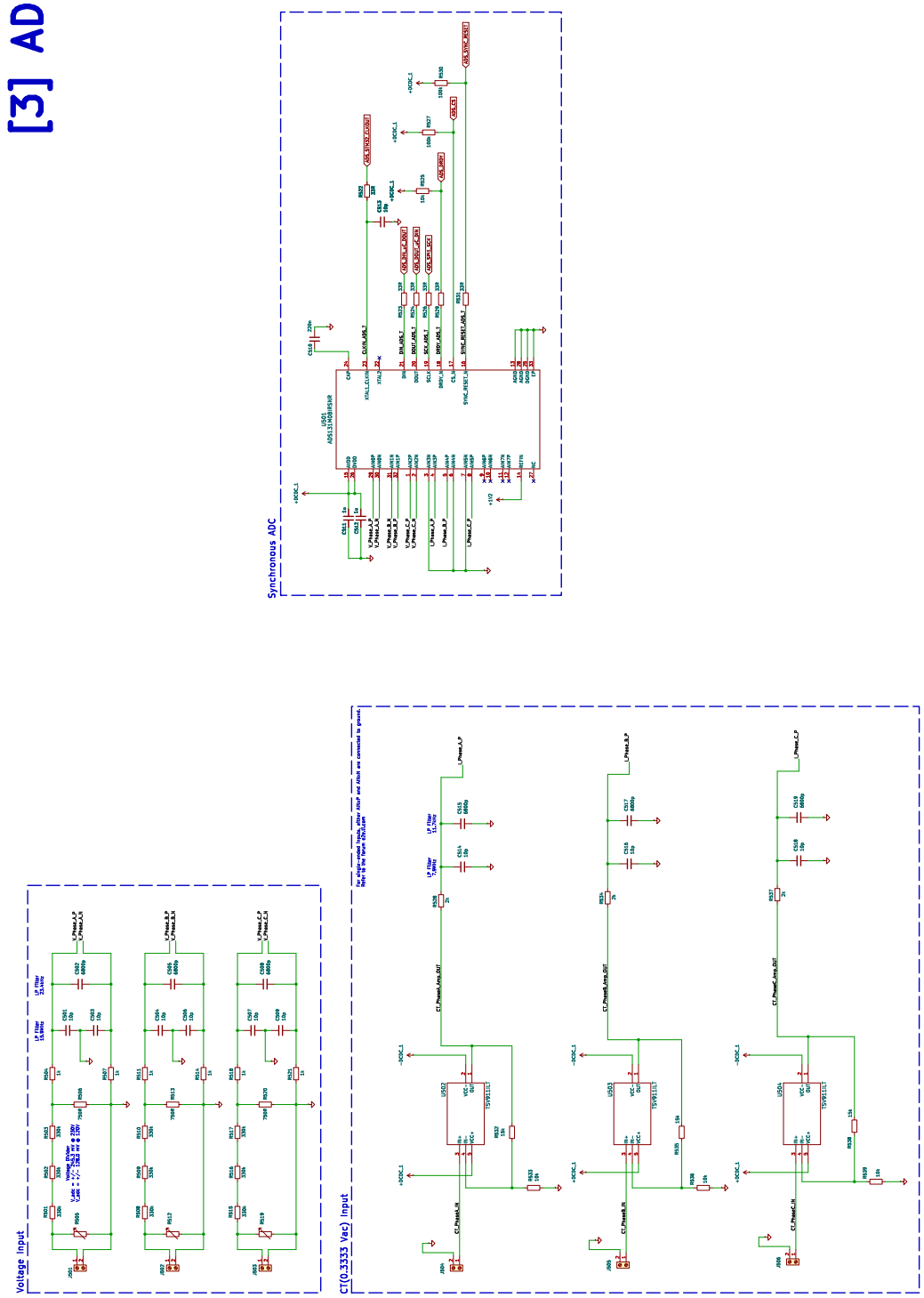


Figure B.5: Schematics for ADC Interface (Page-4).

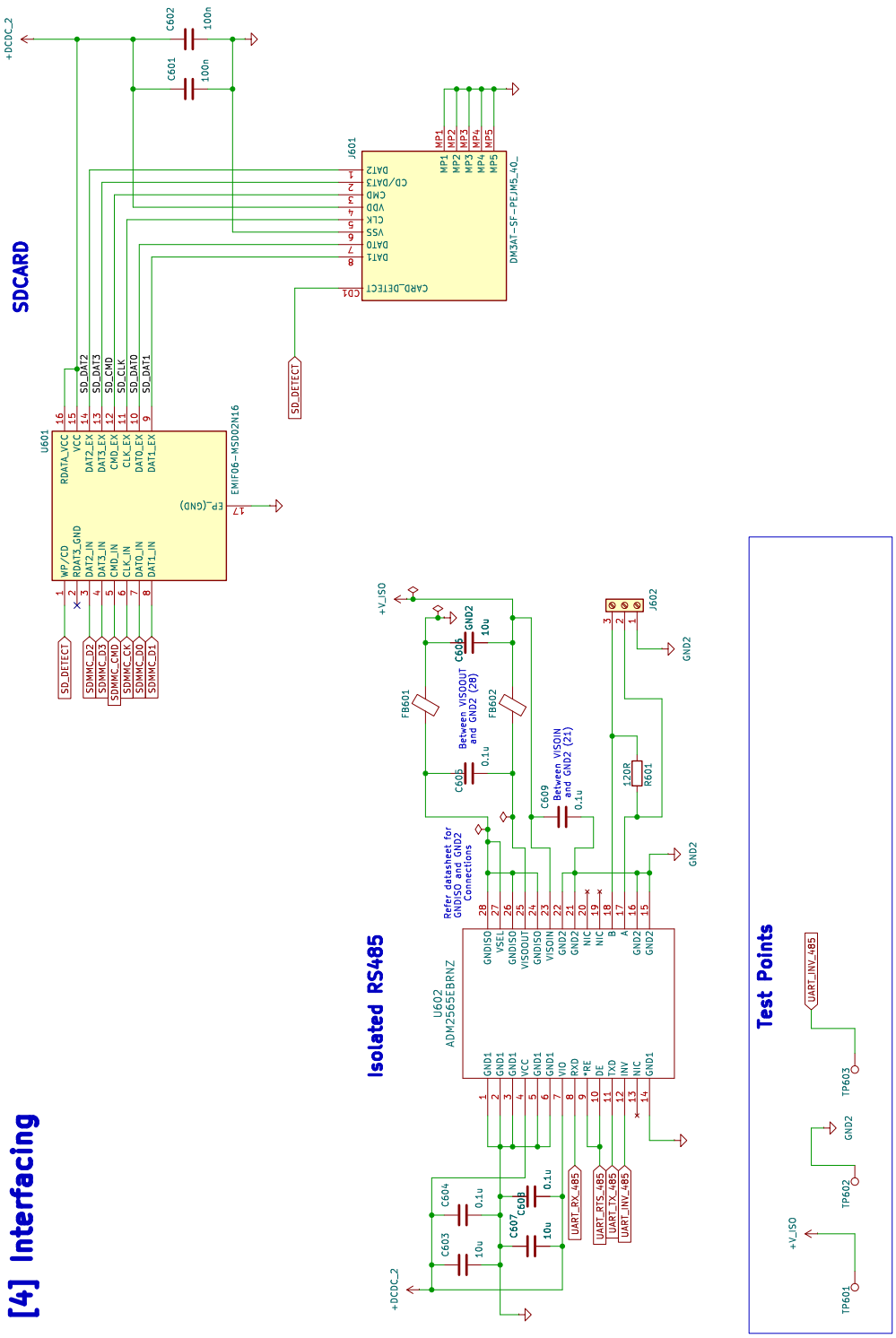


Figure B.6: Schematics for SDCard and Isolated RS485 Interfacing (Page-5).

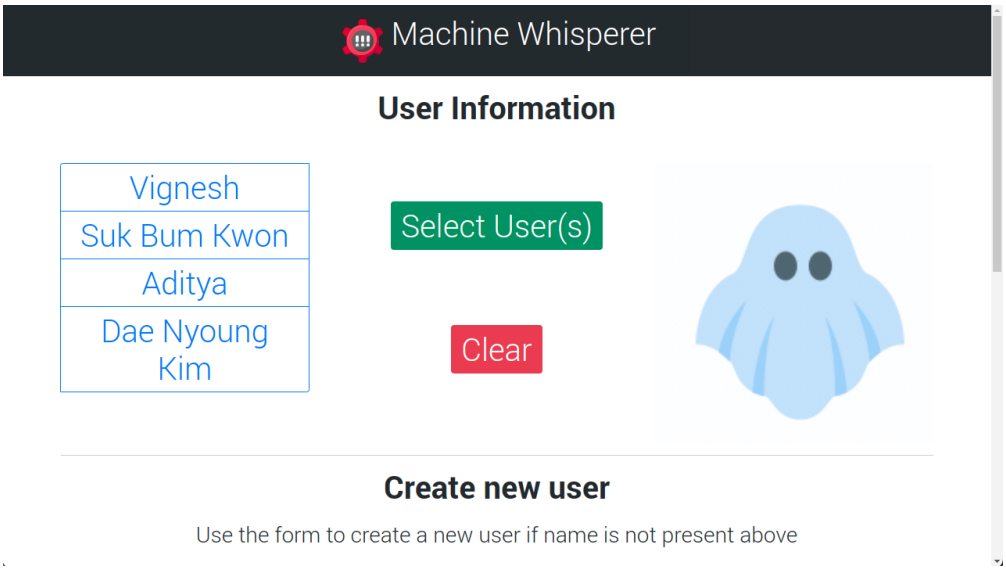


Figure B.7: User selection/creation screen of the application.

tool parameters, materials, manufacturer, etc., as from our studies these parameters will impact the energy consumption of the CNC machine tool.

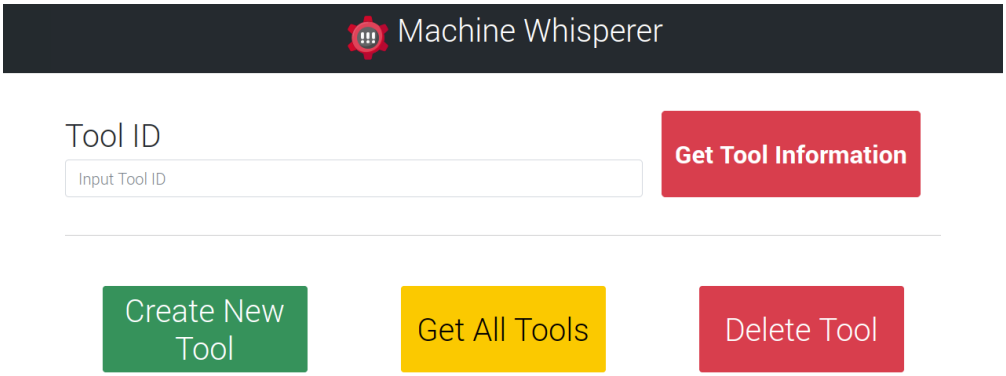


Figure B.8: Tool selection/creation screen of the application.

After tool information has been selected the app will lead to the workpiece information entry followed by operation information entry pages, which can be seen in Figure B.9 and B.10. The workpiece information section of the app can be used to provide information on the workpiece to be machined, optionally, it has provisions to upload the CAD models of the finished part. The operation information entry page can be used to describe the machine axis configuration and workpiece fixture information. Studies have shown that the workpiece location affects the effective energy consumption of the CNC machines. Additionally, the operation information section of the app can be used to upload G-codes if required.

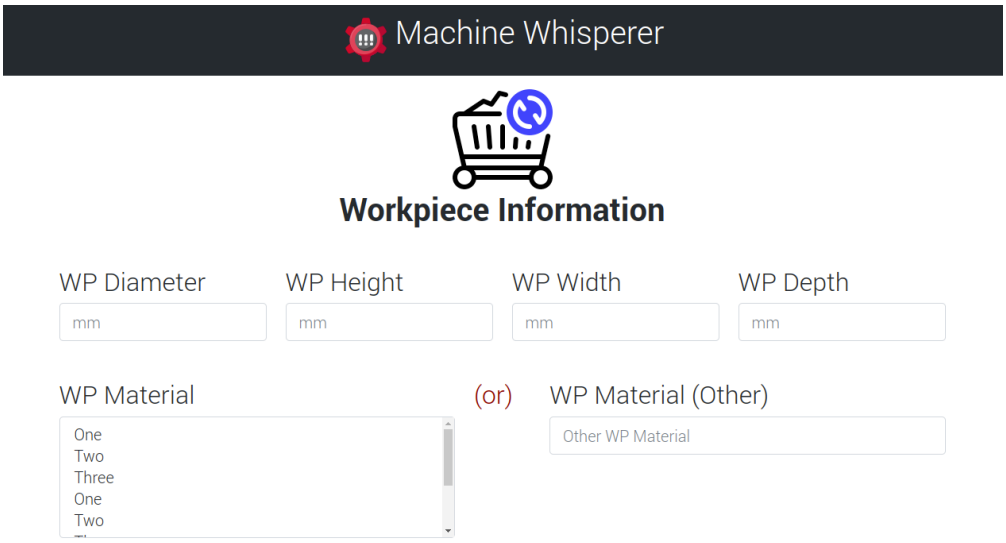


Figure B.9: Workpiece information input screen of the application.

It is imperative to note that the application was developed using NodeJS, but can only be compiled to run on Debian Linux. For our work, it was tested on SBCs like RaspberryPi running Debian Linux.

B.3 Source Code and Other Documents

The source code and documents developed during my research work at MINLab are listed below. This can be found in the USB drive accompanying the thesis document. The USB drive also contains README files describing the content in detail. Additionally, some codebases can be also found on GitHub.

1. Energy data logging and CNC interface hardware
   (a) KiCAD PCB design files
   (b) Firmware—Device functionality, CNC interface, and RS485 interface
2. MTConnect Setup
   (a) Adapter for FANUC ROBONANO $\alpha 0iB$
   (b) Machine status update deployment at MINLab
   (c) Adapter for HAAS
3. LabVIEW program to interface with Fluke Norma4000 power analyzers.
4. G-code interpreter application.
5. MODBUS library for WattNode (from Continential Control Systems) interface with RaspberryPi over RS485.
6. ROBONANO temperature monitoring system deployment.

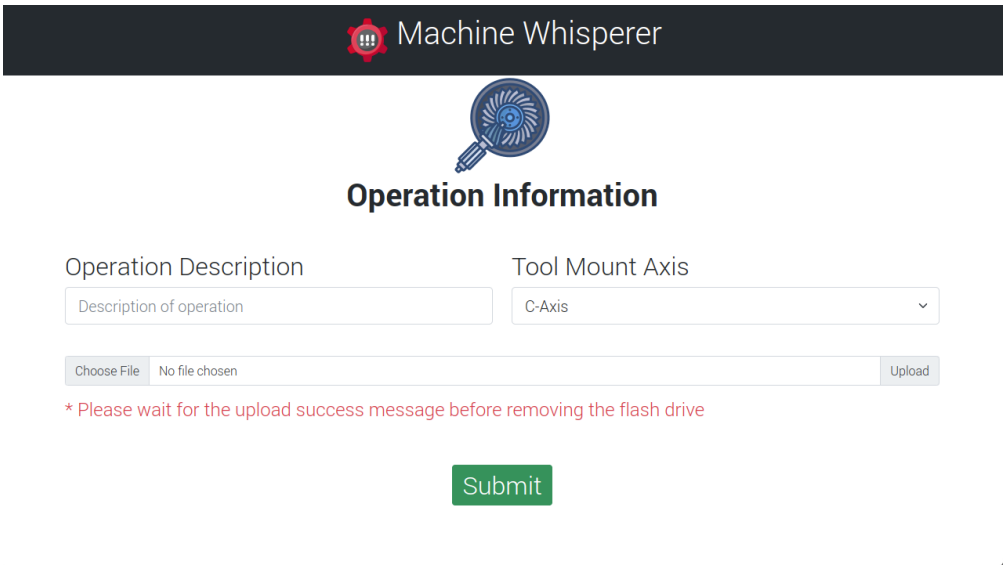


Figure B.10: Operation information input screen of the application.

7. Model development and evaluation for machine fault identification on ROBONANO.
8. Model development and evaluation for state identification on ROBONANO.
9. The source code for the "Machine Whisperer" application.
10. Interface with FANUC CNC controllers over FOCAS2 for data logging and monitoring.

Appendix C

Appendix - Monitoring Of Human-Centric Assembly Processes

The content related to Chapter 4 can be found in this Appendix.

C.1 GUI designed for real-time assembly monitoring

The GUI was designed to showcase the SMIRL in action. You start the GUI by loading the inference machine - A custom trained action detection algorithm. The GUI contains the state machine integrated within its codebase. Figure C.1, shows the starting screen for the GUI. The sections labeled 1, 2, and 3, help load the video, inference machine, and assembly type, and initialize the entire inference module - Both the inference machine and the state machine. The hyperparameters that can be controlled were the inference length, state transition time, and cycle reset time. These parameters govern the sensitivity of the inference module for continuous assembly monitoring. Finally, on pressing the `State Inference` button, the GUI will start streaming the camera feed or load the data from dive to run inference.

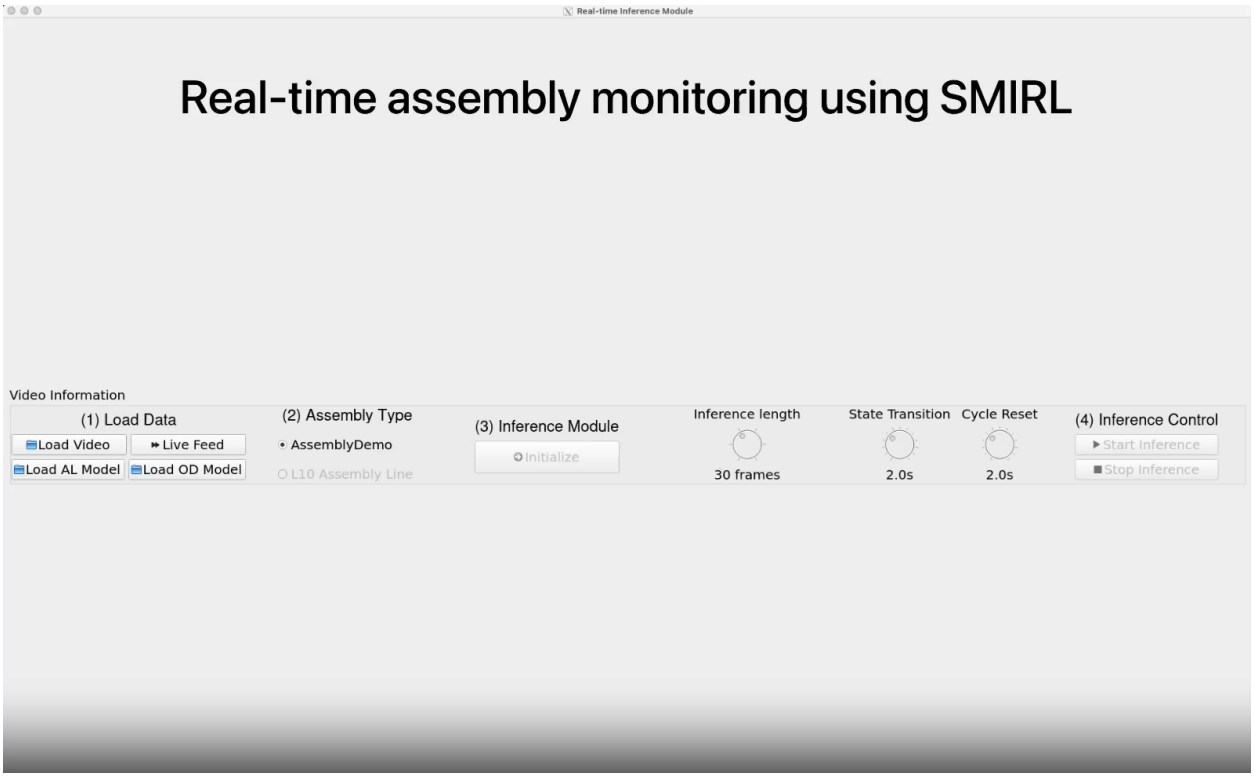


Figure C.1: Working with the GUI developed for real-time assembly monitoring.

The GUI was designed to enable both live streaming of the camera feed and run inference continuously on the recorded data stream. At the end of every cycle, the step time plot was reset to keep the inference process running. An SQL database was implemented to record the inference at every cycle. The database stores the step time for each assembly cycle, the overall cycle time, and any anomalies or missed steps happening within a cycle. The intended use case for the GUI

was to showcase the algorithm's capability and to conduct time studies on assembly workstations. **Note:** To run this application, it is required to have a dedicated GPU with at least 26GB of VRAM. At the time of testing, the GPU used to run this app was an RTX6000.

C.2 Source Code and Other Documents

The source code and documents developed during my research work at MINLab are listed below. This can be found in the USB drive accompanying the thesis document. The USB drive also contains README files describing the content in detail. Additionally, some codebases can be also found on GitHub.

1. SMIRL algorithm development
    (a) C3D-OpticalFlow model training and evaluation
    (b) VGG16 model training and evaluation
    (c) Two-Stream configuration for the inference machine—Model development, training, and evaluation.
    (d) Integration of State Machine with Inference Machine within the Inference Module
    (e) Detection of NVA activities as OOD instances
    (f) Integration of object detection with SMIRL
2. Graph modeling of assembly operations
    (a) Detectron2 training and feature extraction
    (b) 3D model training and feature extraction
    (c) Graph Construction, GCNN training, and Evaluation
3. GUI - Showcasing the capabilities of SMIRL.

Bibliography

[1] Yixiong Feng, Zhaoxi Hong, Zhiwu Li, Hao Zheng, and Jianrong Tan. "Integrated intelligent green scheduling of sustainable flexible workshop with edge computing considering uncertain machine state". In: *Journal of Cleaner Production* 246 (2020), p. 119070.

[2] YC Liang, X Lu, WD Li, and S Wang. "Cyber Physical System and Big Data enabled energy efficient machining optimisation". In: *Journal of cleaner Production* 187 (2018), pp. 46–62.

[3] Jinjiang Wang, Lunkuan Ye, Robert X Gao, Chen Li, and Laibin Zhang. "Digital Twin for rotating machinery fault diagnosis in smart manufacturing". In: *International Journal of Production Research* 57.12 (2019), pp. 3920–3934.

[4] Hyoung Seok Kang, Ju Yeon Lee, SangSu Choi, Hyun Kim, Jun Hee Park, Ji Yeon Son, Bo Hyun Kim, and Sang Do Noh. "Smart manufacturing: Past research, present findings, and future directions". In: *International journal of precision engineering and manufacturing-green technology* 3 (2016), pp. 111–128.

[5] Arno Schmetz, Tae Hun Lee, Maximilian Hoeren, Marvin Berger, Susanne Ehret, Daniel Zontar, Soo-Hong Min, Sung-Hoon Ahn, and Christian Brecher. "Evaluation of industry 4.0 data formats for digital twin of optical components". In: *International Journal of Precision Engineering and Manufacturing-Green Technology* 7 (2020), pp. 573–584.

[6] Robert X Gao, Lihui Wang, Moneer Helu, and Roberto Teti. "Big data analytics for smart factories of the future". In: *CIRP annals* 69.2 (2020), pp. 668–692.

[7] Heiner Lasi, Peter Fettke, Hans-Georg Kemper, Thomas Feld, and Michael Hoffmann. "Industry 4.0". In: *Business & information systems engineering* 6.4 (2014), pp. 239–242.

[8] Pai Zheng, Zhiqian Sang, Ray Y Zhong, Yongkui Liu, Chao Liu, Khamdi Mubarok, Shiqiang Yu, Xun Xu, et al. "Smart manufacturing systems for Industry 4.0: Conceptual framework, scenarios, and future perspectives". In: *Frontiers of Mechanical Engineering* 13.2 (2018), pp. 137–150.

[9] Jim Davis, Thomas Edgar, James Porter, John Bernaden, and Michael Sarli. "Smart manufacturing, manufacturing intelligence and demand-dynamic performance". In: *Computers & Chemical Engineering* 47 (2012), pp. 145–156.

[10] Andrew Kusiak. "Convolutional and generative adversarial neural networks in manufacturing". In: *International Journal of Production Research* 58.5 (2020), pp. 1594–1604.

[11] Chuan Li, Shaohui Zhang, Yi Qin, and Edgar Estupinan. "A systematic review of deep transfer learning for machinery fault diagnosis". In: *Neurocomputing* 407 (2020), pp. 121–135.

[12] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *nature* 521.7553 (2015), pp. 436–444.

[13] Shuhui Wang, Jiawei Xiang, Yongteng Zhong, and Yuqing Zhou. "Convolutional neural network-based hidden Markov models for rolling element bearing fault identification". In: *Knowledge-Based Systems* 144 (2018), pp. 65–76.

[14] Xiang Li, Wei Zhang, and Qian Ding. "A robust intelligent fault diagnosis method for rolling element bearings based on deep distance metric learning". In: *Neurocomputing* 310 (2018), pp. 77–95.

[15] Huaqing Wang, Shi Li, Liuyang Song, and Lingli Cui. "A novel convolutional neural network based fault recognition method via image fusion of multi-vibration-signals". In: *Computers in Industry* 105 (2019), pp. 182–190.

[16] Te Han, Dongxiang Jiang, Yankui Sun, Nanfei Wang, and Yizhou Yang. "Intelligent fault diagnosis method for rotating machinery via dictionary learning and sparse representation-based classification". In: *Measurement* 118 (2018), pp. 181–193.

[17] Haedong Jeong, Seungtae Park, Sunhee Woo, and Seungchul Lee. "Rotating machinery diagnostics using deep learning on orbit plot images". In: *Procedia Manufacturing* 5 (2016), pp. 1107–1118.

[18] Turker Ince, Serkan Kiranyaz, Levent Eren, Murat Askar, and Moncef Gabbouj. "Real-time motor fault detection by 1-D convolutional neural networks". In: *IEEE Transactions on Industrial Electronics* 63.11 (2016), pp. 7067–7075.

[19] Guokai Liu, Weiming Shen, Liang Gao, and Andrew Kusiak. "Knowledge transfer in fault diagnosis of rotary machines". In: *IET Collaborative Intelligent Manufacturing* 4.1 (2022), pp. 17–34.

[20] Mario Aehnelt, Enrico Gutzeit, Bodo Urban, et al. "Using activity recognition for the tracking of assembly processes: Challenges and requirements". In: *WOAR* 2014 (2014), pp. 12–21.

[21] Andrew Glaeser, Vignesh Selvaraj, Sooyoung Lee, Yunseob Hwang, Kangsan Lee, Namjeong Lee, Seungchul Lee, and Sangkee Min. "Applications of deep learning for fault detection in industrial cold forging". In: *International Journal of Production Research* 59.16 (2021), pp. 4826–4835.

[22] K Lange, L Cser, Mm Geiger, and JAG Kals. "Tool life and tool quality in bulk metal forming". In: *CIRP annals* 41.2 (1992), pp. 667–675.

[23] Nader Asnafi. "On tool stresses in cold heading of fasteners". In: *Engineering Failure Analysis* 6.5 (1999), pp. 321–335.

[24] Victor Vazquez, Daniel Hannan, and Taylan Altan. "Tool life in cold forging–an example of design improvement to increase service life". In: *Journal of Materials Processing Technology* 98.1 (2000), pp. 90–96.

[25] Peder Skov-Hansen, Niels Bay, Jens Grønbæk, and Povl Brøndsted. "Fatigue in cold-forging dies: tool life analysis". In: *Journal of Materials Processing Technology* 95.1-3 (1999), pp. 40–48.

[26] B-A Behrens, A Santangelo, and Christian Buse. "Acoustic emission technique for online monitoring during cold forging of steel components: a promising approach for online crack detection in metal forming processes". In: *Production Engineering* 7.4 (2013), pp. 423–432.

[27] Ying-jun Li, Chang-sheng Ai, Xiu-hua Men, Cheng-liang Zhang, and Qi Zhang. "Research on on-line monitoring technology for steel ball's forming process based on load signal analysis method". In: *Mechanical Systems and Signal Processing* 36.2 (2013), pp. 317–331.

[28] Ali Imran Ansari, Santosh J Chauhan, and Pallavi Khaire. "Effect of crack on natural frequency in rotor system". In: *AIP Conference Proceedings*. Vol. 1859. 1. AIP Publishing LLC. 2017, p. 020101.

[29] Hailong Xu, Zhongsheng Chen, Yongmin Yang, Limin Tao, and Xuefeng Chen. "Effects of crack on vibration characteristics of mistuned rotated blades". In: *Shock and Vibration* 2017 (2017).

[30] F Ismail, A Ibrahim, and HR Martin. "Identification of fatigue cracks from vibration testing". In: *Journal of Sound and Vibration* 140.2 (1990), pp. 305–317.

[31] Andrew Glaeser, Vignesh Selvaraj, Kangsan Lee, Namjeong Lee, Yunseob Hwang, Sooyoung Lee, Seungchul Lee, and Sangkee Min. "Remote machine mode detection in cold forging using vibration signal". In: *Procedia Manufacturing* 48 (2020), pp. 908–914.

[32] Vignesh Selvaraj, Aditya Nagaraj, Benjamin Gregory Whiffen, and Sangkee Min. "Development of a wireless smart sensor system and case study on lifting risk assessment". In: *Manufacturing Letters* 41 (2024). 52nd SME North American Manufacturing Research Conference (NAMRC 52), pp. 229–240. ISSN: 2213-8463. DOI: https://doi.org/10.1016/j.mfglet.2024.09.027. URL: https://www.sciencedirect.com/science/article/pii/S2213846324000890.

[33] Tahera Kalsoom, Naeem Ramzan, Shehzad Ahmed, and Masood Ur-Rehman. "Advances in sensor technologies in the era of smart factory and industry 4.0". In: *Sensors* 20.23 (2020), p. 6783.

[34] Andreas Schütze, Nikolai Helwig, and Tizian Schneider. "Sensors 4.0–smart sensors and measurement technology enable Industry 4.0". In: *Journal of Sensors and Sensor systems* 7.1 (2018), pp. 359–371.

[35] Huitaek Yun, Hanjun Kim, Eunseob Kim, and Martin BG Jun. "Development of internal sound sensor using stethoscope and its applications for machine monitoring". In: *Procedia Manufacturing* 48 (2020), pp. 1072–1078.

[36] Yang Fu, Yun Zhang, Yuan Gao, Huang Gao, Ting Mao, Huamin Zhou, and Dequn Li. "Machining vibration states monitoring based on image representation using convolutional neural networks". In: *Engineering Applications of Artificial Intelligence* 65 (2017), pp. 240–251.

[37] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. "Densely connected convolutional networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 4700–4708.

[38] Gary Marcus. "Deep learning: A critical appraisal". In: *arXiv preprint arXiv:1801.00631* (2018).

[39] Leslie N Smith and Nicholay Topin. "Deep convolutional neural network design patterns". In: *arXiv preprint arXiv:1611.00847* (2016).

[40] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[41] Leslie N Smith. "Cyclical learning rates for training neural networks". In: *2017 IEEE winter conference on applications of computer vision (WACV)*. IEEE. 2017, pp. 464–472.

[42] Christian Lessmeier, James Kuria Kimotho, Detmar Zimmer, and Walter Sextro. "Condition monitoring of bearing damage in electromechanical drive systems by using motor current signals of electric motors: A benchmark data set for data-driven classification". In: *PHM Society European Conference*. Vol. 3. 1. 2016.

[43] Shen Zhang, Shibo Zhang, Bingnan Wang, and Thomas G Habetler. "Deep learning algorithms for bearing fault diagnostics—A comprehensive review". In: *IEEE Access* 8 (2020), pp. 29857–29881.

[44] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. "How transferable are features in deep neural networks?" In: *Advances in neural information processing systems* 27 (2014).

[45] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. "Deep domain confusion: Maximizing for domain invariance". In: *arXiv preprint arXiv:1412.3474* (2014).

[46] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. "Learning transferable features with deep adaptation networks". In: *International conference on machine learning*. PMLR. 2015, pp. 97–105.

[47] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. "Adversarial discriminative domain adaptation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 7167–7176.

[48] Weining Lu, Bin Liang, Yu Cheng, Deshan Meng, Jun Yang, and Tao Zhang. "Deep model based domain adaptation for fault diagnosis". In: *IEEE Transactions on Industrial Electronics* 64.3 (2016), pp. 2296–2305.

[49] Artem Rozantsev, Mathieu Salzmann, and Pascal Fua. "Beyond sharing weights for deep domain adaptation". In: *IEEE transactions on pattern analysis and machine intelligence* 41.4 (2018), pp. 801–814.

[50] Erik Englesson and Hossein Azizpour. "Consistency regularization can improve robustness to label noise". In: *arXiv preprint arXiv:2110.01242* (2021).

[51] Wai Sze Yip, Suet To, and Hongting Zhou. "Current status, challenges and opportunities of sustainable ultra-precision manufacturing". In: *Journal of Intelligent Manufacturing* (2021), pp. 1–13.

[52] Weili Cai, Wenjuan Zhang, Xiaofeng Hu, and Yingchao Liu. "A hybrid information model based on long short-term memory network for tool condition monitoring". In: *Journal of Intelligent Manufacturing* 31 (2020), pp. 1497–1510.

[53] Joost R Duflou, John W Sutherland, David Dornfeld, Christoph Herrmann, Jack Jeswiet, Sami Kara, Michael Hauschild, and Karel Kellens. "Towards energy and resource efficient manufacturing: A processes and systems approach". In: *CIRP annals* 61.2 (2012), pp. 587–609.

[54] Thomas Behrendt, Andre Zein, and Sangkee Min. "Development of an energy consumption monitoring procedure for machine tools". In: *CIRP annals* 61.1 (2012), pp. 43–46.

[55] Athulan Vijayaraghavan and David Dornfeld. "Automated energy monitoring of machine tools". In: *CIRP annals* 59.1 (2010), pp. 21–24.

[56] Makoto Fujishima, Katsuhiko Ohno, Shizuo Nishikawa, Kimiyuki Nishimura, Masataka Sakamoto, and Kengo Kawai. "Study of sensing technologies for machine tools". In: *CIRP Journal of Manufacturing Science and Technology* 14 (2016), pp. 71–75.

[57] Makoto Fujishima, Masahiko Mori, Kimiyuki Nishimura, Masakazu Takayama, and Yasutaka Kato. "Development of sensing interface for preventive maintenance of machine tools". In: *Procedia CIRP* 61 (2017), pp. 796–799.

[58] Vignesh Selvaraj, Zhicheng Xu, and Sangkee Min. "Intelligent operation monitoring of an ultra-precision cnc machine tool using energy data". In: *International Journal of Precision Engineering and Manufacturing-Green Technology* 10.1 (2023), pp. 59–69.

[59] Zhicheng Xu, Vignesh Selvaraj, and Sangkee Min. "State identification of a 5-axis ultra-precision CNC machine tool using energy consumption data assisted by multi-output densely connected 1D-CNN model". In: *Journal of Intelligent Manufacturing* (2022), pp. 1–14.

[60] Zhicheng Xu, Vignesh Selvaraj, and Sangkee Min. "Intelligent G-code-based power prediction of ultra-precision CNC machine tools through 1DCNN-LSTM-Attention model". In: *Journal of Intelligent Manufacturing* (2024), pp. 1–24.

[61] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *International conference on machine learning*. PMLR. 2015, pp. 448–456.

[62] Tom O'Malley, Elie Bursztein, James Long, François Chollet, Haifeng Jin, and Luca Invernizzi. *KerasTuner*. `https://github.com/keras-team/keras-tuner`. 2019.

[63] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. In: *Deep learning*. MIT press, 2016, pp. 12–13.

[64] Vignesh Selvaraj and Sangkee Min. "Real-time fault identification system for a retrofitted ultra-precision CNC machine from equipment's power consumption data: A case study of an implementation". In: *International Journal of Precision Engineering and Manufacturing-Green Technology* (2023), pp. 1–17.

[65] German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. "Continual lifelong learning with neural networks: A review". In: *Neural networks* 113 (2019), pp. 54–71.

[66] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. "Isolation forest". In: *2008 eighth ieee international conference on data mining*. IEEE. 2008, pp. 413–422.

[67] Bartosz Krawczyk. "Learning from imbalanced data: open challenges and future directions". In: *Progress in artificial intelligence* 5.4 (2016), pp. 221–232.

[68] Marina Sokolova and Guy Lapalme. "A systematic analysis of performance measures for classification tasks". In: *Information processing & management* 45.4 (2009), pp. 427–437.

[69] Thomas Weber, Johannes Sossenheimer, Steffen Schäfer, Moritz Ott, Jessica Walther, and Eberhard Abele. "Machine learning based system identification tool for data-based energy and resource modeling and simulation". In: *Procedia CIRP* 80 (2019), pp. 683–688.

[70] Vignesh Selvaraj, Md Al-Amin, Xuyong Yu, Wenjin Tao, and Sangkee Min. "Real-time action localization of manual assembly operations using deep learning and augmented inference state machines". In: *Journal of Manufacturing Systems* 72 (2024), pp. 504–518.

[71] Vignesh Selvaraj, Md Al-Amin, Wenjin Tao, and Sangkee Min. "Intelligent assembly operations monitoring with the ability to detect non-value-added activities as out-of-distribution (OOD) instances". In: *CIRP Annals* 72.1 (2023), pp. 413–416.

[72] Vignesh Selvaraj and Sangkee Min. "AI-assisted monitoring of human-centered assembly: A comprehensive review". In: *International Journal of Precision Engineering and Manufacturing-Smart Technology* 1.2 (2023), pp. 201–218.

[73] Takuya Maekawa, Daisuke Nakai, Kazuya Ohara, and Yasuo Namioka. "Toward practical factory activity recognition: unsupervised understanding of repetitive assembly work in a factory". In: *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. 2016, pp. 1088–1099.

[74] Baicun Wang, Pai Zheng, Yue Yin, Albert Shih, and Lihui Wang. "Toward human-centric smart manufacturing: A human-cyber-physical systems (HCPS) perspective". In: *Journal of Manufacturing Systems* 63 (2022), pp. 471–490.

[75] Xun Xu, Yuqian Lu, Birgit Vogel-Heuser, and Lihui Wang. "Industry 4.0 and Industry 5.0—Inception, conception and perception". In: *Journal of Manufacturing Systems* 61 (2021), pp. 530–535.

[76] Joseph Korpela, Kazuyuki Takase, Takahiro Hirashima, Takuya Maekawa, Julien Eberle, Dipanjan Chakraborty, and Karl Aberer. "An energy-aware method for the joint recognition of activities and gestures using wearable sensors". In: *Proceedings of the 2015 ACM International Symposium on Wearable Computers*. 2015, pp. 101–108.

[77] Takuya Maekawa, Yutaka Yanagisawa, Yasue Kishino, Katsuhiko Ishiguro, Koji Kamei, Yasushi Sakurai, and Takeshi Okadome. "Object-based activity recognition with heterogeneous sensors on wrist". In: *International Conference on Pervasive Computing*. Springer. 2010, pp. 246–264.

[78] Juhi Ranjan and Kamin Whitehouse. "Object hallmarks: Identifying object users using wearable wrist sensors". In: *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. 2015, pp. 51–61.

[79] Takuya Maekawa, Yasue Kishino, Yasushi Sakurai, and Takayuki Suyama. "Activity recognition with hand-worn magnetic sensors". In: *Personal and ubiquitous computing* 17.6 (2013), pp. 1085–1094.

[80] Ferhat Attal, Samer Mohammed, Mariam Dedabrishvili, Faicel Chamroukhi, Latifa Oukhellou, and Yacine Amirat. "Physical human activity recognition using wearable sensors". In: *Sensors* 15.12 (2015), pp. 31314–31338.

[81] Trung Thanh Ngo, Yasushi Makihara, Hajime Nagahara, Yasuhiro Mukaigawa, and Yasushi Yagi. "Similar gait action recognition using an inertial sensor". In: *Pattern Recognition* 48.4 (2015), pp. 1289–1301.

[82] Heli Koskimaki, Ville Huikari, Pekka Siirtola, Perttu Laurinen, and Juha Roning. "Activity recognition using a wrist-worn inertial measurement unit: A case study for industrial assembly lines". In: *2009 17th mediterranean conference on control and automation*. IEEE. 2009, pp. 401–405.

[83] Thomas Stiefmeier, Georg Ogris, Holger Junker, Paul Lukowicz, and Gerhard Troster. "Combining motion sensors and ultrasonic hands tracking for continuous activity recognition in a maintenance scenario". In: *2006 10th IEEE international symposium on wearable computers*. IEEE. 2006, pp. 97–104.

[84] Wenjin Tao, Ze-Hao Lai, Ming C Leu, and Zhaozheng Yin. "Worker activity recognition in smart manufacturing using IMU and sEMG signals with convolutional neural networks". In: *Procedia Manufacturing* 26 (2018), pp. 1159–1166.

[85] Wenjin Tao, Haodong Chen, Md Moniruzzaman, Ming C Leu, Zhaozheng Yi, and Ruwen Qin. "Attention-based sensor fusion for human activity recognition using IMU signals". In: *arXiv preprint arXiv:2112.11224* (2021).

[86] Wenjin Tao, Ming C Leu, and Zhaozheng Yin. "Multi-modal recognition of worker activity for human-centered intelligent manufacturing". In: *Engineering Applications of Artificial Intelligence* 95 (2020), p. 103868.

[87] Chen Chen, Roozbeh Jafari, and Nasser Kehtarnavaz. "A survey of depth and inertial sensor fusion for human action recognition". In: *Multimedia Tools and Applications* 76.3 (2017), pp. 4405–4425.

[88] Shibo Zhang, Yaxuan Li, Shen Zhang, Farzad Shahabi, Stephen Xia, Yu Deng, and Nabil Alshurafa. "Deep learning in human activity recognition with wearable sensors: A review on advances". In: *Sensors* 22.4 (2022), p. 1476.

[89] Huifen Xia and Yongzhao Zhan. "A survey on temporal action localization". In: *IEEE Access* 8 (2020), pp. 70477–70487.

[90] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. "Large-scale video classification with convolutional neural networks". In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2014, pp. 1725–1732.

[91] Karen Simonyan and Andrew Zisserman. "Two-stream convolutional networks for action recognition in videos". In: *Advances in neural information processing systems* 27 (2014).

[92] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. "Learning spatiotemporal features with 3d convolutional networks". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 4489–4497.

[93] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. "Long-term recurrent convolutional networks for visual recognition and description". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 2625–2634.

[94] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. "Beyond short snippets: Deep networks for video classification". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 4694–4702.

[95] Joao Carreira and Andrew Zisserman. "Quo vadis, action recognition? a new model and the kinetics dataset". In: *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 6299–6308.

[96] Zheng Shou, Dongang Wang, and Shih-Fu Chang. "Temporal action localization in untrimmed videos via multi-stage cnns". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 1049–1058.

[97] Yuanjun Xiong, Yue Zhao, Limin Wang, Dahua Lin, and Xiaoou Tang. "A pursuit of temporal accuracy in general activity detection". In: *arXiv preprint arXiv:1703.02716* (2017).

[98] Tianwei Lin, Xiao Liu, Xin Li, Errui Ding, and Shilei Wen. "Bmn: Boundary-matching network for temporal action proposal generation". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 3889–3898.

[99] Huijuan Xu, Abir Das, and Kate Saenko. "R-c3d: Region convolutional 3d network for temporal activity detection". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 5783–5792.

[100] Runhao Zeng, Wenbing Huang, Mingkui Tan, Yu Rong, Peilin Zhao, Junzhou Huang, and Chuang Gan. "Graph convolutional networks for temporal action localization". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 7094–7103.

[101] Qianqian Xiong, Jianjing Zhang, Peng Wang, Dongdong Liu, and Robert X Gao. "Transferable two-stream convolutional neural network for human action recognition". In: *Journal of Manufacturing Systems* 56 (2020), pp. 605–614.

[102] Marcello Urgo, Marco Tarabini, and Tullio Tolio. "A human modelling and monitoring approach to support the execution of manufacturing operations". In: *CIRP Annals* 68.1 (2019), pp. 5–8.

[103] Chengjun Chen, Tiannuo Wang, Dongnian Li, and Jun Hong. "Repetitive assembly action recognition based on object detection and pose estimation". In: *Journal of Manufacturing Systems* 55 (2020), pp. 325–333.

[104] Joseph Redmon and Ali Farhadi. "Yolov3: An incremental improvement". In: *arXiv preprint arXiv:1804.02767* (2018).

[105] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. "Convolutional pose machines". In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2016, pp. 4724–4732.

[106] Ping Lou, Ji Li, YuHang Zeng, Bing Chen, and Xiaomei Zhang. "Real-time monitoring for manual operations with machine vision in smart manufacturing". In: *Journal of Manufacturing Systems* 65 (2022), pp. 709–719.

[107] Jihong Yan and Zipeng Wang. "YOLO V3+ VGG16-based automatic operations monitoring and analysis in a manufacturing workshop under Industry 4.0". In: *Journal of Manufacturing Systems* 63 (2022), pp. 134–142.

[108] Chengjun Chen, Chunlin Zhang, Changzhi Li, and Jun Hong. "Assembly Monitoring Using Semantic Segmentation Network Based on Multiscale Feature Maps and Trainable Guided Filter". In: *IEEE Transactions on Instrumentation and Measurement* 71 (2022), pp. 1–11.

[109] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors". In: *nature* 323.6088 (1986), pp. 533–536.

[110] Weitang Liu, Xiaoyun Wang, John Owens, and Yixuan Li. "Energy-based out-of-distribution detection". In: *Advances in neural information processing systems* 33 (2020), pp. 21464–21475.

[111] Justin Johnson, Ranjay Krishna, Michael Stark, Li-Jia Li, David Shamma, Michael Bernstein, and Li Fei-Fei. "Image retrieval using scene graphs". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3668–3678.

[112] Hang Qi, Yuanlu Xu, Tao Yuan, Tianfu Wu, and Song-Chun Zhu. "Scene-centric joint parsing of cross-view videos". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018.

[113] Ruize Wang, Zhongyu Wei, Piji Li, Qi Zhang, and Xuanjing Huang. "Storytelling from an image stream using scene graphs". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 05. 2020, pp. 9185–9192.

[114] Xiaolong Wang and Abhinav Gupta. "Videos as space-time region graphs". In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 399–417.

[115] Jingwei Ji, Ranjay Krishna, Li Fei-Fei, and Juan Carlos Niebles. "Action genome: Actions as compositions of spatio-temporal scene graphs". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 10236–10247.

[116] Chengjun Chen, Xicong Zhao, Jinlei Wang, Dongnian Li, Yuanlin Guan, and Jun Hong. "Dynamic graph convolutional network for assembly behavior recognition based on attention mechanism and multi-scale feature fusion". In: *Scientific Reports* 12.1 (2022), p. 7394.

[117] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. "Mask r-cnn". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2961–2969.

[118] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. "Faster R-CNN: Towards real-time object detection with region proposal networks". In: *IEEE transactions on pattern analysis and machine intelligence* 39.6 (2016), pp. 1137–1149.

[119] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. *Detectron2*. `https://github.com/facebookresearch/detectron2`. 2019.

[120] Anton Milan, Laura Leal-Taixé, Ian Reid, Stefan Roth, and Konrad Schindler. "MOT16: A benchmark for multi-object tracking". In: *arXiv preprint arXiv:1603.00831* (2016).

[121] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. "How powerful are graph neural networks?" In: *arXiv preprint arXiv:1810.00826* (2018).

[122] National Safety Council. *Work injuries and illnesses by part of body*. https://injuryfacts.nsc.org/work/industry-incidence-rates/work-injuries-and-illnesses-by-part-of-body/. 2023.

[123] Xuemei Luo, Ricardo Pietrobon, Shawn X Sun, Gordon G Liu, and Lloyd Hey. "Estimates and patterns of direct health care expenditures among individuals with back pain in the United States". In: *Spine* 29.1 (2004), pp. 79–86.

[124] Ming-Lun Lu, Menekse S Barim, Shuo Feng, Grant Hughes, Marie Hayden, and Dwight Werren. "Development of a wearable IMU system for automatically assessing lifting risk factors". In: *Digital Human Modeling and Applications in Health, Safety, Ergonomics and Risk Management. Posture, Motion and Health: 11th International Conference, DHM 2020, Held as Part of the 22nd HCI International Conference, HCII 2020, Copenhagen, Denmark, July 19–24, 2020, Proceedings, Part I 22*. Springer. 2020, pp. 194–213.

[125] Thomas R Waters, Vern Putz-Anderson, Arun Garg, and Lawrence J. Fine. "Revised NIOSH Equation for the Design and Evaluation of Manual Lifting Tasks". In: *Ergonomics* (July 1993). (Visited on 10/17/2023).

[126] Thomas R Waters, Vern Putz-Anderson, and Arun Garg. *Applications manual for the revised NIOSH lifting equation*. Revised, 2021. U. S. Department of Health, Human Services, National Institute for Occupational Safety, and Health, Cincinnati OH. 1994.

[127] Steven D Hlucny and Domen Novak. "Characterizing human box-lifting behavior using wearable inertial motion sensors". In: *Sensors* 20.8 (2020), p. 2323.

[128] Srimantha E Mudiyanselage, Phuong Hoang Dat Nguyen, Mohammad Sadra Rajabi, and Reza Akhavian. "Automated workers' ergonomic risk assessment in manual material handling using sEMG wearable sensors and machine learning". In: *Electronics* 10.20 (2021), p. 2558.

[129] Leandro Donisi, Giuseppe Cesarelli, Armando Coccia, Monica Panigazzi, Edda Maria Capodaglio, and Giovanni D'Addio. "Work-related risk assessment according to the revised NIOSH lifting equation: A preliminary study using a wearable inertial sensor and machine learning". In: *Sensors* 21.8 (2021), p. 2593.

[130] Brennan Thomas, Ming-Lun Lu, Rashmi Jha, and Joseph Bertrand. "Machine Learning for Detection and Risk Assessment of Lifting Action". In: *IEEE Transactions on Human-Machine Systems* 52.6 (2022), pp. 1196–1204.

[131] T Waters, E Occhipinti, D Colombini, E Alvarez-Casado, and A Hernandez-Soto. "The Variable Lifting Index (VLI): a new method for evaluating variable lifting tasks using the Revised NIOSH Lifting Equation". In: *Proceedings of the 17th Triennial Congress of the International Ergonomics Association*. 2009, pp. 1–3.

[132] Natale Battevi, Monica Pandolfi, and Ivan Cortinovis. "Variable lifting index for manual-lifting risk assessment: a preliminary validation study". In: *Human Factors* 58.5 (2016), pp. 712–725.

[133] Ilaria Conforti, Ilaria Mileti, Zaccaria Del Prete, and Eduardo Palermo. "Measuring biomechanical risk in lifting load tasks through wearable system and machine-learning approach". In: *Sensors* 20.6 (2020), p. 1557.