Scalable State Change Detection for Stimulus-Aware Digital Twins

MS (Research) Thesis

by

ABHISHEK TIWARI

DEPARTMENT OF MECHANICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY INDORE

MAY 2025

Scalable State Change Detection for Stimulus-Aware Digital Twins

A THESIS

Submitted in partial fulfillment of the

requirements for the award of the degree

of

Master of Science (Research)

by

ABHISHEK TIWARI

DEPARTMENT OF MECHANICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY INDORE

MAY 2025

INDIAN INSTITUTE OF TECHNOLOGY INDORE

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the thesis entitled **Scalable State Change Detection for Stimulus-Aware Digital Twins** in the fulfillment of the requirements for the award of the degree of **MASTER OF SCIENCE (RESEARCH)** and submitted in the **DISCIPLINE OF MECHANICAL ENGINEERING, Indian Institute of Technology Indore**, is an authentic record of my own work carried out during the time period July 2023 to May 2025 under the supervision of Dr. Vibhor Pandhare, Assistant Professor.

The matter presented in this thesis has not been submitted by me for the award of any other degree of this or any other institute.

[illegible]

Signature of the student

ABHISHEK TIWARI

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

[illegible]

Signature of the Supervisor of
MS (Research) Thesis

DR. VIBHOR PANDHARE

ABHISHEK TIWARI has successfully given his MS (Research) Oral Examination held on 03/07/2025.

[illegible] 04 July 2025
Signature of Chairperson (OEB) with date

[illegible] 4 July, 2025
Signature(s) of Thesis Supervisor(s) with date

[illegible] 04-07-2025
Signature of Convener, DPGC with date

July 4, 2025
Prof. Dhinakaran Shanmugam
[illegible]
Prof Dhinakaran Shanmugam
HoD, MechE
Signature of Head of Discipline with date

ACKNOWLEDGEMENTS

I would like to express my heartfelt gratitude to Dr. Vibhor Pandhare for his exceptional mentorship and steadfast support during my research journey. His insightful guidance, constructive feedback, and encouragement to explore a topic of personal interest have been pivotal in shaping the direction and depth of my work. Dr. Vibhor Pandhare commitment to academic excellence and his belief in my potential have significantly enriched my research experience and have been crucial to my academic and professional development.

I am also immensely grateful to Prof. B.K Lad for his consistent assistance and cooperation. Your willingness to help with my research problems and your insightful feedback have been crucial in overcoming various challenges. Thank you for always being approachable and for your valuable contributions to my work.

I am deeply grateful to the administrative and technical staff of the Department of Mechanical Engineering at IIT Indore. Their prompt assistance and resourcefulness ensured the smooth progress of my research activities. I appreciate their dedication and efficiency.

I want to express my heartfelt gratitude to my family, friends, and especially to my friend Yash Khandelwal. Your unwavering encouragement and belief in me have been my greatest source of motivation. I truly appreciate your continuous support through the highs and lows of this journey; it has been invaluable. I am especially grateful for your understanding and patience during the times when my research required my undivided attention.

Finally, I wish to thank everyone who has directly or indirectly contributed to the completion of this thesis. Your support, encouragement, and belief in my abilities have been an invaluable asset in this academic endeavor. To all the unnamed individuals who have played a part in this journey, your contributions are sincerely appreciated.

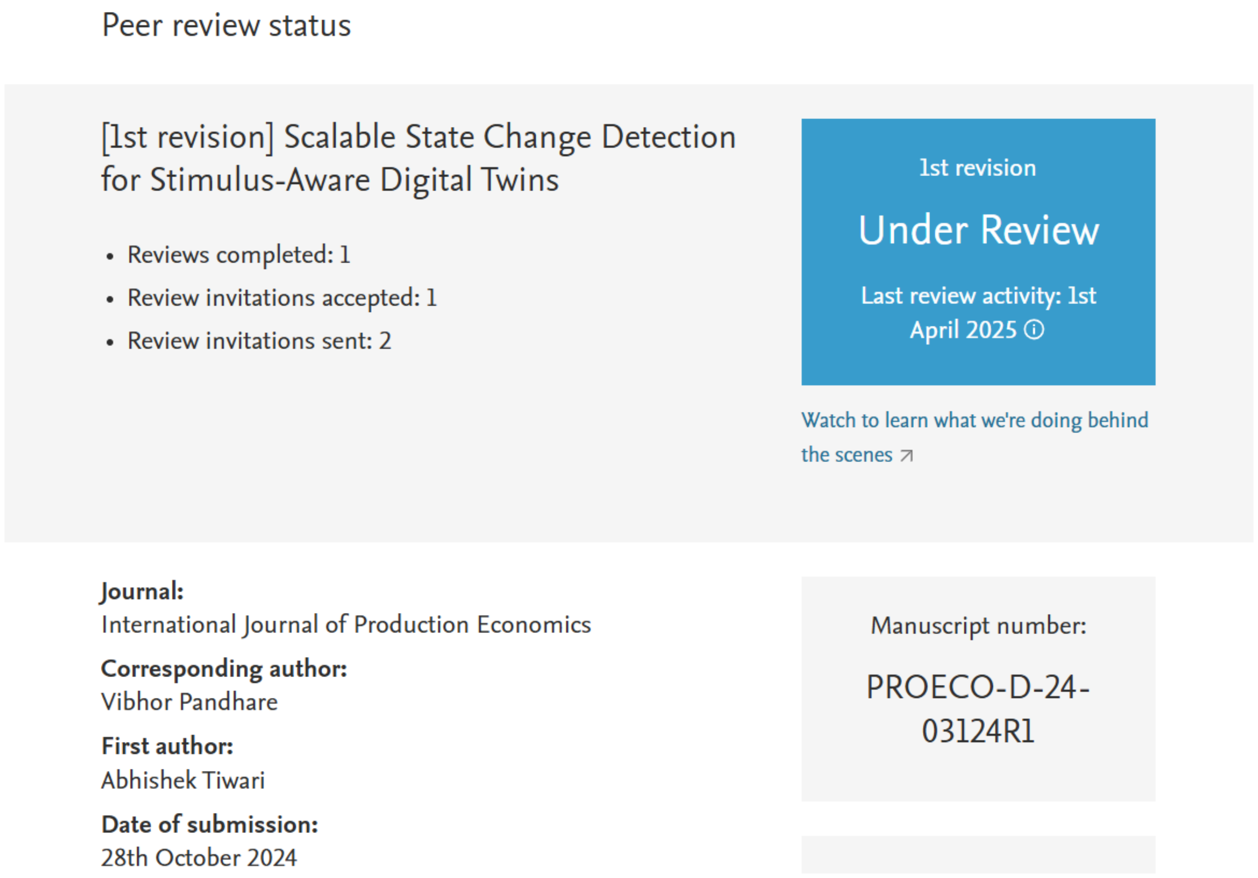
Thank you.

ABSTRACT

One of the fundamental functionalities of a digital twin of any production system is its ability to serve as a ‘twin’, i.e. effective synchronization between the physical state and the digital state. These states can include being idle, working, failure, setup, slowdown, etc. Application-based top-down approaches to developing digital twins generally consider pre-calculated states or known data conditions, limiting their scope. In contrast, real-life production presents a myriad of possible states. This stimulus-awareness serves as the foundation for realistically modeling production behavior in real time, which leads to improved production planning, simulation, execution, and management. Thus, a bottom-up scalable state detection method and a systematic four-phase learning framework using minimal data are proposed for realizing stimulus-aware digital twins at the beginning of equipment operation. The proposed solution uses an unsupervised autoencoder for two-stage pattern recognition and PCA-T*2* for change detection learning using a single state, followed by new state detection and adaptation using a novel implementation of a custom loss function to maximize the distribution discrepancy between the latent representation of the two states. Validation is performed over three real-life datasets, including a newly created gearbox dataset that varies in type of equipment, signal, operating parameters, etc. 80 experiments are conducted across different scenarios and repeated 5 times each, along with statistical testing to benchmark performance with the traditional approach. To further benchmark the performance of the proposed model, Generative Adversarial Networks (GANs) have also been explored. An additional 10 experiments were conducted across various scenarios, with each experiment repeated five times to assess the model's performance. The proposed method detects a change in state with over 97.5% accuracy and an F1 score of more than 95.2%, irrespective of the dataset or the states used for modeling, narrowing the gap between localized success stories and the large-scale use of digital twins in production industries.

LIST OF PUBLICATIONS

Abhishek Tiwari, Vibhor Pandhare, “Scalable State Change Detection for Stimulus-Aware Digital Twins”, *International Journal of Production Economics*. (Manuscript Under Review)



Peer review status

[1st revision] Scalable State Change Detection for Stimulus-Aware Digital Twins

- Reviews completed: 1
- Review invitations accepted: 1
- Review invitations sent: 2

1st revision

Under Review

Last review activity: 1st April 2025

Watch to learn what we're doing behind the scenes

Journal:
International Journal of Production Economics

Corresponding author:
Vibhor Pandhare

First author:
Abhishek Tiwari

Date of submission:
28th October 2024

Manuscript number:

PROECO-D-24-03124R1

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

DT	Digital Twin
CPS	cyber-physical systems
CDT	Cognitive Digital Twins
UML	Unified Modeling Language
MBSE	Model-based systems engineering
MAS	Multi-agent systems
PCA	Principal Component Analysis
AE	Autoencoders
MSE	Mean Squared Error
MMD	Maximum Mean Discrepancy
RKHS	Reproducing Kernel Hilbert Space
AI	Artificial Intelligence
ML	Machine Learning
IOT	Internet of Things
GANs	Generative Adversarial Network
WGAN	Wasserstein Generative Adversarial Network
FFT	Fast Fourier Transform

Chapter 1 Introduction

The modern manufacturing industry operates in an increasingly complex and competitive environment, where agility, efficiency, and precision are essential[1],[2]. However, a critical barrier that manufacturers face is the lack of real-time monitoring and operational visibility. Without a continuous flow of live data from the shop floor, decision-makers struggle to respond swiftly to production anomalies or performance bottlenecks. This lack of insight is often rooted in the widespread use of legacy machines that were not designed to connect digitally, making it difficult to collect and integrate data into centralized systems. The lack of integrated data also limits the adoption of predictive maintenance, forcing many manufacturers to rely on reactive practices that result in unexpected downtimes and costly disruptions[3]. Operational inefficiencies further emerge due to the inability to optimize energy usage and resource consumption, driving up production costs and reducing sustainability. Quality control becomes another critical concern, as outdated inspection processes and limited traceability lead to inconsistent product standards and difficulties in identifying the root causes of defects[4], [5]. Traditional production systems remain rigid and struggle to adapt to evolving customer demands, such as customization and shorter delivery cycles. Ultimately, the absence of standardized, interoperable systems and limited access to actionable data constrain manufacturers from leveraging advanced analytics and predictive technologies, creating a cycle of inefficiency that undermines competitiveness in a rapidly changing market. One major issue is the heavy use of old machines that weren't built to connect digitally, which makes it tough to gather and combine data from them[6]. These challenges highlight the urgent need for digital transformation through the adoption of technologies such as artificial intelligence (AI), sensors, the Internet of Things (IoT), and Digital Twin systems. The growing complexity of modern equipment and manufacturing applications, particularly with digitalization in production processes, consists of numerous diverse components with varying goals and dynamic connections. Managing such systems manually is impractical, so future systems must exhibit advanced autonomy. Industries are undergoing a significant transformation due to the rise of Industry 4.0, which emphasizes the digitalization of traditional industrial and manufacturing practices. This new industrial revolution focuses on leveraging advanced digital technologies to make manufacturing processes smarter, more efficient, and highly interconnected. At the core of Industry 4.0 are technologies such as Artificial Intelligence (AI), Machine Learning (ML), big data analytics, Internet of Things (IoT), and high-performance computing. These tools empower companies to collect and analyze vast amounts of data generated by sensors, machines, and systems in real time. By processing this data, industries can predict operational issues, enhance production processes, and minimize waste. Moreover, the integration of AI and ML enables these systems to learn from historical and real-time data, continuously improving performance and supporting smarter decision-making. Industries are experiencing a significant digital transformation driven by the rise of Industry 4.0. This shift is fueled by the unprecedented growth of technologies, including machine learning, IoT, cyber-physical systems, computational power, information and communication technologies, and advanced sensing capabilities. Digital Twins are emerging as one of the most essential integrations of these technologies due to their potential to optimize process control, prevent quality defects, monitor equipment health, predict failures, and streamline supply chains [7][8].

1.1 Digital Twin Systems

In recent years, numerous efforts have been made to consolidate the definition of digital twins in several review papers [9][10][11]. VanDerHorn et al. [11] state a digital twin as "a virtual representation of a physical system (and its associated environment and processes) that is updated through the exchange of information between the physical and virtual systems". It allows for a dynamic, real-time connection between the digital and physical worlds, enabling organizations to simulate, monitor, and optimize performance throughout the lifecycle of assets or processes. Digital Twins act as a convergence point for multiple Industry 4.0 technologies, combining data from IoT devices, enhanced with AI and ML algorithms, to deliver insights that drive operational excellence.

In manufacturing, Digital Twins are revolutionizing how companies approach process optimization, predictive maintenance, and quality assurance[3], [12]. They allow for simulations that test and validate production processes, reducing downtime and enhancing productivity. In supply chain management, Digital Twins provide end-to-end visibility, support proactive decision-making, and allow for real-time replanning and rescheduling in response to market fluctuations or disruptions[13], [14]. Their ability to continuously learn from operational data means they can not only reflect the current state of a system but also predict future behavior and recommend adjustments. Beyond manufacturing, the benefits of Digital Twins extend to healthcare, automotive, aerospace, smart cities, and agriculture[15], [16][17]. In healthcare, Digital Twins can model individual patients for personalized treatment planning. In the automotive and aerospace sectors, they enable virtual testing and maintenance prediction. For smart cities, they help in urban planning and infrastructure monitoring. As industrial systems become increasingly complex and autonomous, the need for more advanced, intelligent Digital Twins is becoming apparent. These next-generation systems must evolve beyond basic simulations to become cognitive and self-aware, capable of understanding their environment, anticipating changes, and making decisions independently[18]. This evolution will be crucial in achieving the full potential of Industry 4.0, creating systems that are not only automated but also adaptive and intelligent.

1.2 Architecture of Digital Twin Systems

With the rapid advancement of digitalization in manufacturing, modern equipment and production systems have become increasingly complex. These systems comprise numerous interconnected components, each with distinct functions, objectives, and operating conditions. As a result, they generate vast and diverse datasets under varying operational scenarios, making data analysis and interpretation a challenging yet essential task. Managing such systems manually is impractical, so future systems require an intelligent system. These systems are especially valuable in the production industry, where digital twins are integrated with self-awareness principles to enhance prediction and decision-making processes. Based on the level of intelligence integrated into the system, Digital Twins can be classified into the following categories[19]:

i. **Descriptive Digital Twin:** A digital representation that captures and visualizes the current and historical state of a physical system. It provides real-time or recorded data to describe what is happening and has happened, enabling monitoring, reporting, and basic diagnostics. However, it does not offer insights into future states or support decision-making through predictions or simulations.

ii. **Predictive Digital Twin:** Typically utilizes predictive data analytics such as data mining, statistical techniques, and machine learning to extrapolate what might happen in the future. Such a twin only allows reliable extrapolation to scenarios similar to those that produced the historical or training data.

iii. **Prescriptive Digital Twin:** Twins that utilize simulation, support what-if analysis, thus enabling explanations and deep understanding of mechanisms governing system behavior. The exploration of multiple what-if scenarios delivers actionable insights suggesting the best solution given various choices and optimizing the result of future events or risks.

iv. **Cognitive Digital Twin:** These are twins with the highest level of intelligence that can replicate human cognitive processes and execute conscious actions autonomously, with minimal or no human intervention.

As digitalization progresses in production systems, modern machinery and manufacturing setups have grown significantly more complex. These systems consist of many interlinked components, each with unique functions, goals, and operating environments. Due to their dynamic interactions and continual adjustments to shifting demands, manually managing these systems has become increasingly unfeasible. Human operators are no longer able to match the necessary precision, speed, and synchronization needed for efficient and dependable operation. As a result, future manufacturing systems must become more autonomous to effectively cope with this complexity.

A critical aspect of this autonomy lies in the system's ability to be self-aware and self-expressive[19], [20], [21]. Self-awareness refers to the system's capability to monitor its own internal states, understand its performance, and recognize its environment. This allows the system to detect faults, anticipate maintenance needs, and adjust operations to maintain optimal performance. On the other hand, self-expression is the ability of the system to communicate its status, goals, and requirements to other machines, systems, or human operators[21]. This ensures smooth coordination, effective collaboration, and timely responses to any changes or issues within the production environment. Together, self-awareness and self-expression form the foundation for smart, adaptive manufacturing systems. They enable real-time decision-making, reduce the need for human intervention, and increase the system's overall resilience and flexibility. As manufacturing continues to embrace digital transformation, these intelligent capabilities will become essential for achieving efficient, scalable, and sustainable production.

According to the literature, the concept of self-awareness[17], [19], [20], [22] has been seen in multiple disciplines, including psychology, artificial intelligence (AI), systems engineering, and robotics. This multidisciplinary approach contributes to the design of intelligent systems capable of adapting to dynamic environments. In the literature, researchers have defined five principles of self-awareness for digital twins in the following categories[20]:

i. **Stimulus aware:** stimulus-aware digital twin serves as the foundation for all other types of self-aware digital twins. It is primarily reactive in nature, meaning it focuses on detecting events happening and responding to stimuli and events in the monitored environment without engaging in predictive or prescriptive analysis. Unlike simulation or what-if analysis, which use time-based projections to anticipate future outcomes, a stimulus-aware digital twin operates solely in the present. It demonstrates how real-time detection and reaction to environmental changes can influence the state and behavior of the observed system. For instance, in a parcel delivery system utilizing drones, a stimulus-aware digital twin would adjust routing and operations in response to immediate events, such as weather changes or traffic disruptions, according to predefined policies and objectives. Objectives may include minimizing the average delivery time of parcels and maximizing delivery coverage, while constraints could involve maintaining delivery within specified time windows, accounting for environmental factors, and respecting drone flying time limitations. This approach ensures that the system remains adaptive and resilient to real-world changes, maintaining performance without relying on future predictions.

ii. **Interaction aware:** Knowledge of the stimulus of other twins and their relationship to itself.

iii. **Time aware:** Knowledge of the past and likely future stimulus and its impact.

iv. **Goal aware:** Knowledge of current goals, preferences, and constraints.

Incorporating all these levels of awareness in digital twin enhances their intelligence, adaptability, and autonomy, allowing them to self-monitor, optimize processes, and make informed decisions[19]. In [20], [21], [22] discussed the architecture of a self-aware digital twin system along with the different levels of awareness in digital twins. The architecture consists of four main components, as shown in Figure 1. The first component is the physical system connected with sensors; the data acquisition system measures the aspects inside the system. The second component is the updating engine, which is coherent with the self-awareness mechanism in [22]which updates the changes from a physical system to a digital system. The prediction engine [10] is used to predict the future state of the physical system using the prediction engine, which is coherent with the self-expression mechanism that enables decision-making and updating back to the physical system using the actuators. These dimensions are supported by optimization dimensions through optimizing the performance of relevant functionalities like modeling, state estimation, prediction, decision-making, etc. Development and integration of these dimensions qualify an entity as a self-aware digital twin and enable the fulfillment of the enormous benefits spanning its physical counterpart's entire life cycle, including enhanced decision support [23]. These dimensions are supported by optimization

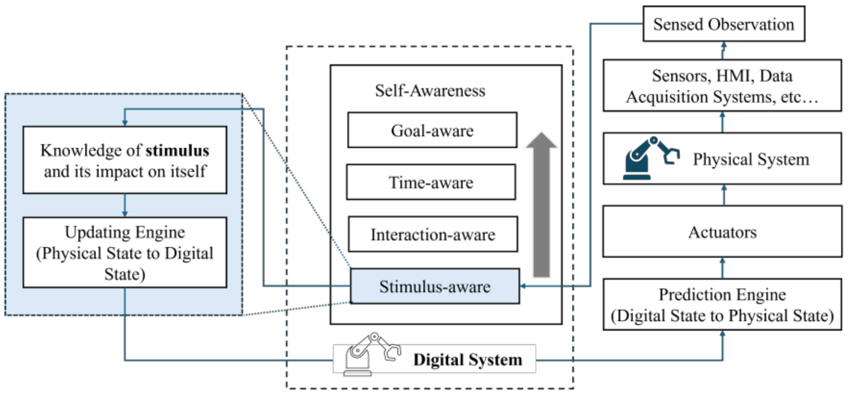

Sensed Observation
Sensors, HMI, Data Acquisition Systems, etc…
Physical System
Actuators
Prediction Engine (Digital State to Physical State)
Self-Awareness
Goal-aware
Time-aware
Interaction-aware
Stimulus-aware
Knowledge of stimulus and its impact on itself
Updating Engine (Physical State to Digital State)
Digital System

Figure 1 Conceptual Framework of Self-Aware Digital Twin

dimensions through optimizing the performance of relevant functionalities like modeling, state estimation, prediction, decision-making, etc.[23]. While the perspectives to categorize the maturity of Digital Twins may differ, the functionalities remain consistent. The fundamental functionality is its ability to serve as a “twin,” i.e., synchronization between the physical state and the digital state. This is given through the updating engine in [23], cognitive capability in [24], and stimulus awareness in [20]. This capability of being stimulus-aware forms the quintessential foundation for the detection of state, optimization, and actuation of the goal-aligned decision to the physical system.

1.3 Concept of State for equipment

[25] Has described the concept of state detection by presenting a probabilistic dynamic decision network that tracks the physical state of the entity S_t, observed by the sensors. At each time step, based on these sensed observations, these physical states are being updated in virtual space as the digital state D_t. [26] have further considered a sequence of unknown digital state vectors that can be represented by $\mathrm{r}_0 : q = \{\mathrm{r}_0, r_1, \cdots, \mathrm{r}_q\}$, and a sequence of noisy measurements or observation vectors, $v_0 : q = \{v_0, v_1, \cdots, v_q\}$. The discrete-time state space model is formulated that takes the following form at time step q which represents the time t_q. F(.)

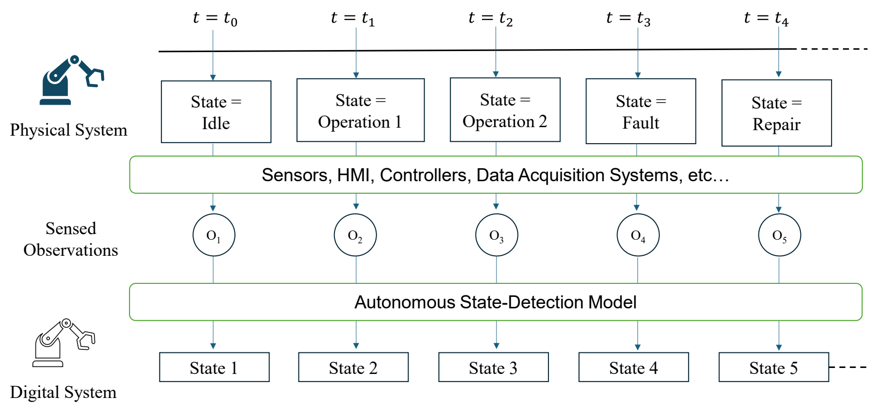


Figure 2 Updating digital states based on observations of physical states in a Digital Twin

and g(.) are further defined as the state transition and measurement function, respectively, that map the transition of the physical state to the digital state based on the observations. Figure 2 describes this ability via representation of the physical states being updated over time and the corresponding change of states in the digital space based on the observations. This becomes extremely important in the production environment to model the behavior of events for cost modeling, make span estimations, labor planning, inventory management, viz., and any behavior modeling for improved planning and execution. In real-life complex manufacturing systems, a system can witness significantly more states. For example, [27] considers a total of five states of the critical components in the target system: 1) healthy and operating; 2) degraded and operating; 3) healthy and idle; 4) degraded and idle; and 5) under maintenance. However, while operating, the system can undergo a change in product, a change in operation like grinding, milling, polishing, etc., and a change in operating conditions like RPM, load, etc. In a degraded health condition, the system can witness multiple failure modes. Figure 3 shows how a combination of all these factors can lead to the exponential growth of several states.

The ability to detect events through the monitoring and analysing equipment states plays a pivotal role in the development and functionality of digital twins. Event detection is not merely about recognizing abnormal behaviours it serves as the foundation for intelligent, responsive, and adaptive systems.

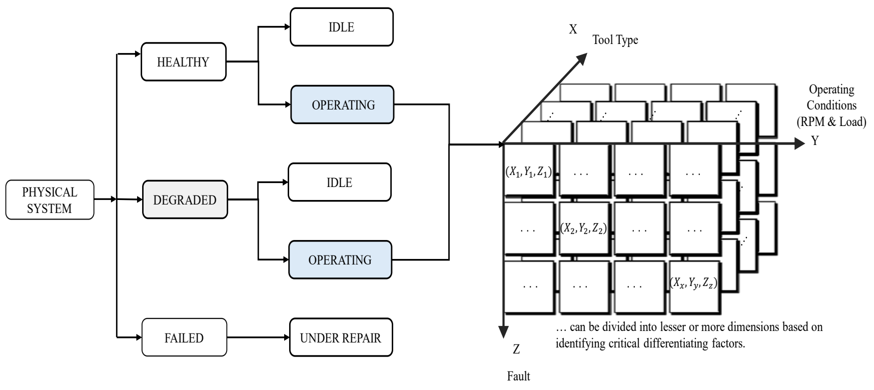


Figure 3 Exponentially expanding machine state combinations

As highlighted in[28], identifying transitions in equipment states enables the modeling of digital twins using knowledge graphs, which can enrich contextual understanding and drive decision-making processes across various industrial domains.

From a broader perspective, event detection through state monitoring stands as a cornerstone in the realm of Prognostics and Health Management (PHM) of critical assets. Applications span across a wide range of including electric motors [29], hydraulic pumps [30], air compressor [31], machine tools [3], and renewable energy systems such as wind turbines [32], [33]. Through accurate state recognition, these systems can not only detect early signs of faults but also provide predictive insights, allowing for timely interventions that minimize downtime and extend asset life. One notable domain where state detection has gained traction is in battery technology. Specifically, the quantification of state of change (SoC) and state of health (SoH) in lithium-ion batteries [34][35] is vital for optimizing performance and ensuring safety in energy storage systems. These assessments support informed decisions in applications ranging from electric vehicles to renewable energy storage solutions.

Despite these advancements, the integration of state detection as a core dimension within digital twin architectures remains relatively underexplored. For instance, [36] demonstrated a forward-thinking approach by creating a digital twin of a fixed-wing unmanned aerial vehicle (UAV) that exhibits a degree of self-awareness. This UAV could dynamically detect and adapt to changes in its structural health due to damage or degradation, enabling real-time adjustments in mission planning and execution. Such capabilities are crucial for mission-critical tasks like the delivery of high-priority items [37], especially under constraints like delivery time windows, environmental conditions, and flight endurance limitations [38].

In manufacturing, the incorporation of event detection into digital twins is gradually shaping new paradigms of production control and scheduling. For example, [27] introduced a field-synchronized digital twin framework that responds to real-time fault detection in a flow shop environment, enabling adaptive and uncertainty-resilient production planning.

In essence, event detection via equipment states forms the sensory system of a digital twin, allowing it to perceive, interpret, and act upon the evolving operational context. This capability not only supports predictive maintenance and health monitoring but also empowers the digital twin to serve as a proactive agent, one that evolves with the system, responds to unplanned scenarios, and optimizes performance across the asset lifecycle.

To tackle this practical issue shown in Figure 3, a bottom-up approach is essential, wherein the digital twin learns as the observations become available with time. Additionally, as digital twins can be built for multiple types of assets, scalability is also essential. Thus, a systematic, scalable state detection method for realizing stimulus-aware digital twins is proposed, which forms the primary novelty of the work. It is motivated to bridge the gap between localized success stories and the large-scale usage of digital twins in production industries.

The scope of this work in making digital twins semi-autonomous is defined as its ability to self-learn over the initial operational states of equipment for a given type of observation/signal, with inputs from domain experts. Additionally, the scope of scalability is defined as the quality of being semi-autonomous over types of equipment as well as types of observations. The proposed work is validated across a new dataset collected on an experimental gearbox test rig, along with two public datasets for benchmarking the performance of the method.

1.4 Organization of the Thesis

This thesis is comprised of six chapters, which are as follows:

i. The current chapter introduces the research topic, emphasizes its significance, defines key concepts, and outlines the study's scope.
ii. The second chapter will focus on conducting a comprehensive literature review that emphasizes three main areas: modeling and enabling technologies in the digital twin, state detection methods, and domain adaptation for scalability. The research objectives and proposed contributions will also be defined.
iii. The third chapter will explore the four-state learning methodology that provides insights into developing a stimulus-aware or state-detection capability as a core part of the digital twin.
iv. The fourth chapter presents the experimental gear box test setup, used to perform the experiments, and dataset description, which are used to validate the model.
v. The fifth chapter presents the experimental setup for three real-life datasets and their dataset description, which are used to validate the model.
vi. In the sixth chapter, the results and findings obtained from the research will be presented, analyzing and interpreting the results in relation to the research objectives.
vii. Chapter Seven presents a benchmarking study comparing the performance of GAN-based models with the proposed model. It details the evaluation results, key findings, and discusses the challenges encountered during the benchmarking process.
viii. The eighth chapter will showcase the practical application of the proposed model. The methodology is highly effective for job tracking, monitoring, and scheduling in the production industry.
ix. The ninth chapter will summarize the main conclusions and contributions of the research, discuss its potential areas for future research, provide recommendations for further investigation, and reflect on the overall research experience and lessons learned.

Chapter 2 Literature Review

2.1 Literature Review

In this chapter, we will focus on conducting a comprehensive literature review that emphasizes three main areas: modeling and enabling technologies in the digital twin, state detection methods, and domain adaptation for scalability. The research objectives and proposed contributions will also be defined.

2.2 Modeling technologies in the digital twins

In the growing area of developing digital twins, modeling methods, i.e., the process of creating a digital representation of a physical system, process, or object, play an important role. Their main purpose is to recreate the characteristics, behaviors, and real-time data interactions of a physical asset in a virtual space, allowing computation, analysis, and optimization. Based on purpose and complexity, modeling methods can be classified into the following categories [23]:

1) **Geometric Modelling:** Geometrical modeling, or solid modeling, is a core component of digital twin development. Using computer-aided design (CAD) software, 3D virtual representations of physical objects are created from 2D drawings or measurements. These models capture the shape, structure, and spatial relationships of equipment and systems, allowing engineers to simulate behavior, test configurations, and optimize performance without interfering with actual operations. Common CAD tools used in engineering include SolidWorks, AutoCAD, PTC Creo, and Catia, while platforms like Blender and Unity3D offer accessible free alternatives. Through these tools, individual components can be assembled into full system models for layout analysis, motion simulation, or ergonomic studies. In manufacturing, geometrical models are used to simulate production lines, optimize layout, and

evaluate process timing. Solid modeling also supports human–robot interaction studies, as seen in [28], where ergonomic assessments were performed in a virtual assembly area. These models are vital in ensuring safety and efficiency in shared workspaces. Additionally, solid models are used for quality control tasks such as tolerancing, geometry variation, and defect detection. Though high-fidelity models for physics-based simulations like FEA require significant effort to develop, original equipment manufacturers (OEMs) increasingly provide these models with their machines. This growing trend aligns with the vision outlined by [39], where digital twins come pre-integrated with accurate solid models for predictive maintenance, virtual upgrades, and faster issue resolution. As a result, geometrical modeling forms a foundational pillar in the effective design and application of digital twins in modern manufacturing.

2) **Physics-based Modeling:** Physics-based modeling and simulation in digital twin includes the mathematical model built using partial differential equations. Physics-based modeling and simulation for use in digital twin can be used in solid body structural analysis to know states such as stress levels, fluid flow rate, or temperature, using the finite element analysis. Finite element analysis has so far been an important modeling technique for digital twin [40][41]. Zhang et al. [42] used the finite element analysis and a DT-enhanced dynamic scheduling methodology to overcome the challenges of the machine's unavailability, mainly due to the machine's faulty state and cutting tool wear. Wang et al. [43] developed a digital twin of a battery through a digital model that captures the battery's critical states, such as the State of Charge (SoC), the State of Power (SoP), and the State of Health (SoH), which assesses the battery's overall condition and degradation over time.

3) **Physics-informed Modeling:** Due to the complexities in creating physics-based models, a hybrid approach is used where the physics-based modeling is integrated with machine learning (ML) models. [3] Implemented the hybrid predictive maintenance approach for the CNC milling machine tool driven by a digital twin to monitor the time-varying states of the CNC machine tool during its operation, which are observed by the sensors. Similar hybrid approaches have been attempted for physical system modeling in digital twins for damage detection in structures [44]. While this approach improves generalization and explainability, it lacks a standard methodology for integrating the physics and machine learning models.
4) **Data-driven Modeling:** By far the most used approach, data-driven models are constructed to represent the underlying physical input/output relationship using statistical or machine learning models. These techniques have become popular for modeling the physical system in a digital twin, as they learn from historical and real-time data to make predictions from unseen data [45] [46] [15][47]. Data-driven approaches have also been developed for change detection in industrial applications[48] [49]. Specifically with regard to production and manufacturing, various machine states like idle, working, failure, setup, slow down, off, and others for a shop floor are discussed[50][51]. For advanced machines, controllers can be a promising way to extract operational parameters like load, RPM, and battery capacity that define the state of the equipment. [52] demonstrates the integration of modern PLC programming techniques with the Siemens TIA portal, which enhances the monitoring of induction motors. The proposed integrated IoT-based architecture in [53] incorporates SCADA with an LQR-PID controller to serve as a local control unit or local intelligence for monitoring the flow rate and pressure in fluid pipeline transportation systems. In [28], during the robotic

machining process, the workpiece is monitored for states: chatter, critical, and stable. While modern controllers can provide information without installing additional sensors, the data quality might not be suitable for detecting critical health states or operational anomalies[54] [55]. Moreover, it is not always possible to access data from the controller for all equipment.

5) **System Modeling:** System modeling is very crucial for representing the complex physics production system with Unified Modeling Language (UML), which is a general-purpose language that is commonly used for software development in model-based systems engineering (MSBE)[56]. Its application extends towards the modeling of digital twins, where UML has been used to model the connection between the different mechanical, hydraulic, and electrical components in a CNC machine tool[57]. System modeling has four main components: structure, behavior, properties, and requirements. These components enhance the validation, traceability, and life cycle management, making it highly relevant for digital twin implementations in production systems and shop floor environments. [58][59]. Increasing complexities of digital twins implementations in the production system, driven by simulation and databases, have made the UML and system modeling indispensable for managing system architecture and preventing downtime and implementation delays. A representative usage of UML is showcased by [59] where they considered the digital twin of cyber-physical production systems. As the digital system contained many components and complex relationships between those components, [60]adopted a UML-based approach to model system structure and component interactions associated with different system functions (i.e., monitoring, diagnostics, and predictive maintenance. UML and system modeling face challenges in digital twin modeling related to the software-centric constraints

[61] these techniques also face difficulties when there are large, complex systems [62], and a lack of real-time behavior interaction[60]While the UML improves cross-disciplinary integration, it still requires additional tools for full system connectivity.

6) **Multi-Agent System (MAS):** Multi-Agent Systems (MAS) are increasingly essential in the development of Digital Twins (DTs), especially when modeling complex and dynamic cyber-physical systems. A Digital Twin is a virtual representation of a physical asset or system that operates in real time, often requiring the coordination of multiple subsystems. MAS offers a decentralized framework where autonomous agents, each with defined roles and objectives, interact and collaborate to simulate, monitor, and manage these subsystems effectively. The integration of MAS into DT architectures simplifies development by breaking down complex systems into modular, manageable agents. Each agent can be responsible for specific tasks such as data collection, diagnostics, predictive analytics, or optimization. This structure promotes scalability, resilience, and flexibility, allowing the DT to adapt to changes in its physical counterpart or environment in real time. As noted in [63],applying a design science research methodology alongside MAS architecture supports a systematic approach to building DTs. This methodology emphasizes iterative development and practical application, reducing system complexity and improving the efficiency of the design process. MAS-based DT implementations have been successfully applied in several domains. In smart manufacturing, MAS enables real-time process monitoring, fault prediction, and efficient resource scheduling, leading to reduced downtime and increased productivity [64]. In smart city planning, MAS facilitates the simulation of urban systems such as transportation, energy, and infrastructure, supporting more informed

and adaptive decision-making. In conclusion, MAS provides a robust, flexible foundation for developing Digital Twins, enabling better coordination, scalability, and intelligent system behavior in complex environments.

2.3 State Detection Methods

Specifically with regard to production and manufacturing, various machine states like idle, working, failure, setup, slow down, off, and others for a shop floor are discussed [50][51]. For advanced machines, controllers can be a promising way to extract operational parameters like load, RPM, and battery capacity that define the state of the equipment. Vadi et al. [52] demonstrate the integration of modern PLC programming techniques with the Siemens TIA portal, which enhances the monitoring of induction motors. The proposed integrated IoT-based architecture in [53] incorporates SCADA with an LQR-PID controller to serve as a local control unit or local intelligence for monitoring the flow rate and pressure in fluid pipeline transportation systems. In [28], during the robotic machining process, the workpiece is monitored for states: chatter, critical, and stable. While modern controllers can provide information without installing additional sensors, the data quality might not be suitable for detecting critical health states or operational anomalies. [54],[55]. However, it is not always possible to access data from the controller for all equipment.

External signal-based learning approaches include one-class support vector machines (SVM) [65] and support vector data description (SVDD) [66] amongst others. Xie et al. [67] used a data-driven approach and discussed applications of digital twins at different stages of a cutting tool's life cycle. Peng et al [68]proposed a digital twin model of the life cycle of the bearing, where the health states of the bearing are monitored. Luo et al. [61] have developed the digital twin for the CNC machine tool, where a multi-domain unified modeling technique of DT is established to map the physical and digital space for monitoring the health states and diagnosing faults. Luo et al. [69] also presented a DT for the CNC machine tool, which allows for precise prediction and monitoring of the machining state of a CNC machine tool, as well as real-time compensation of the workpiece's machining contour inaccuracy. Support vector machines (SVM) and Mult-kernel support vector machines are widely employed in detecting bearing classification and different faulty states classifications [70], [71]. Many other machine learning techniques are widely used for the development of digital twins these machine learning algorithms are applied to diagnose rotating machines, such as Principal Component Analysis (PCA) [72], and Kernel Principal Component Analysis (KPCA) [73], decision tree[74], and method like random forest.[75]While the application of data-driven modeling is vast, the approaches are generally developed for a specific equipment and a specific purpose, and they lack generalizability across the different equipment and across the sensors Production systems possess the complexity of changing machine states with time due to varied operational patterns, environmental conditions, as well as varied system configurations. To address generalization across such patterns, deep learning has seen significant success in detecting health states. [76], [77], [78] have summarized and discussed various deep learning techniques that can be used to detect the health states of the various rotating equipment, such as gearboxes, bearings, and pumps. Generative Adversarial Neural Networks [79] can learn the behavior of the system and identify deviations from the learned behavior to be used for the prognostic and health management of the equipment. For rotatory components that play a vital role in various engineering domains, [80] proposed a convolutional neural network for intelligent diagnosis of health states. Zhu et al. [81] discussed the health and operational states, focused on conditional monitoring, and anomaly detection of wind turbines using long short-term memory-fuzzy synthesis (LSTM-FS). Advanced machine learning techniques, such as Generative Adversarial Networks (GANs), Long Short-Term Memory-Fuzzy Synthesis (LSTM-FS), GAN-based Autoencoders (GAN-AE), and traditional

Autoencoders, are widely used for learning and modeling complex data behaviours. These methods excel at automatically extracting meaningful features, identifying intricate patterns, and capturing underlying data distributions. The architecture of GANs is inefficient in handling anomaly and change detection scenarios due to problems such as: Mode collapse, Vanishing gradients, Training instability, and lack of robustness. Autoencoders (AEs) are among the most widely used unsupervised learning techniques due to their ability to learn meaningful data representations and perform dimensionality reduction. As a fundamental building block in deep learning, AEs can be stacked hierarchically to create deep models, enabling the extraction of high-level, non-linear features while efficiently organizing and compressing data [82]. Autoencoders are especially useful for tasks like anomaly detection, predictive modeling, and state recognition across various domains.

2.4 Domain Adaptation Techniques

In complex manufacturing systems, shifts in the domain and data distribution can occur due to changes in operating conditions, equipment, or sensor locations. To further minimize the impact of the shift in the domain or data distribution due to changes in operating conditions, equipment, or sensor locations, domain adaptation techniques can be applied to align the feature distribution between the source and target domains [83],[84]. Domain adaptation is a subsection of transfer learning that seeks to overcome domain differences for algorithms to increase generalizability by learning domain-invariant characteristics. It solves the domain shift problem by determining a mapping from the source distribution to the target distribution. According to [85], the domain adaptation or knowledge transfer in health states diagnosis of a rotating machinery is divided into four categories: domain adaptation across working conditions, across locations, across machines, and across fault types. Pandhare et al. [86] proposes a domain adaptation technique for the knowledge transfer between the sensor locations for monitoring ball screws. Xing et al. [87] show a distribution-invariant Deep Belief Network (DIDBN) across working conditions of machines for the diagnosis of health states. The most widely used approach to tackle the challenges of domain shift is the distribution alignment approach by matching statistics. This includes adaptive batch normalization and automatic domain alignment layers using the statistics of source and target batches at different layers to align the two distributions [88][89]. In the literature review, we have explained various domain adaptation techniques like divergence-based criterion, Wasserstein Discrepancy [90],[91], adversarial domain adaptation [92],[93]. The existing domain adaptation approaches are broadly categorized into methods with shallow and deep architectures. Shallow domain adaptation approaches[94][95], mainly utilize instance-based and feature-based techniques to align the domain distributions. One way of aligning the distributions is by minimizing the distance between domains, including methods such as Wasserstein metric, correlation alignment (CORAL)[96], Kullback-Leibler (KL) divergence[97], and contrastive domain discrepancy (CDD)[98]. The most widely used distance measure in domain adaptation is maximum mean discrepancy (MMD)[99]. The use of the MMD method for domain adaptation to align the domain shift has been widely discussed in [7], [30], [32], [86], [100]. In complex manufacturing systems, a large number of sensors continuously generate complex data, process parameters, and quality metrics. Principal Component Analysis (PCA) plays a crucial role in simplifying this data by reducing its dimensionality while preserving essential information. By eliminating redundant and correlated features. PCA is a multivariate technique that analyses data to extract important information. Its goal is to represent this information as a set of new orthogonal variables known as principal components. Additionally, PCA helps to display the patterns of similarity among the observations and variables. The effectiveness of multivariate analysis in detecting anomalies has been acknowledged in various fields, including industrial monitoring [97],[98],[99],[100]. Although significant research has been conducted in generalization, the development efforts have focused on a top-down approach where, once again, the challenge comes from an application for generalization across specific types of sensors and equipment for their health states. This top-down approach limits the evolution of the digital twin across all dimensions and prevents its existence as a complex, cognitive, and self-aware entity by itself. This is primarily because when the approach is focused on an application, the possible states that can be witnessed by the physical system are pre-calculated, and the model is developed with limited scope. However, in real-life applications, the number of states a system can witness is significantly more. Thus, a scalable method for creating stimulus-aware digital twins is missing.

2.5 Research Gaps

i. First and foremost, most of the work looks at isolated optimization of entities instead of integrated optimization, for example, between production schedule optimization and fault detection. The reason is a lack of focus on work that integrates the aspects of generalized state detection and state sustenance that form the fundamental requirement for realistic and robust prediction and simulation through the digital twin for enhanced decision-making.

ii. Production systems are often categorized as up/healthy (fully operational) or down/faulty (non-operational), overlooking situations where systems may exist in multiple other states that are critical to efficient planning of time, money, and resources. The existing literature does not sufficiently cover the detection of these transitional operational states.

iii. There is a lack of a generalized or scalable methodology for state detection in the digital twin during its operational life cycle. Most of the focus has been on developing methodologies for state detection for specific equipment and for specific use cases. Research is needed to bridge the gap in algorithm generalizability across multiple datasets of multiple signal or equipment types.

iv. The existing methods for change detection or monitoring consider specific data from selected healthy states/faulty states for model training. However, in real life, the state/data available for training may vary significantly as the model is deployed across equipment or use cases. Thus, there has been limited research focused on evaluating the developed model across the various possible training states/data.

2.6 Research Objectives

I. Design and development of a scalable algorithm to autonomously detect changes in a digital twin's state.

II. Creation of new datasets for the development and validation of the algorithm, with two additional public datasets used to benchmark the performance of the proposed method.

Chapter 3 Proposed four-phase learning framework

3.1 Proposed Methodology

Let S_i denote the physical state of the equipment. It can be represented by $S_i\colon \left\{P_1^1, \dots, \dots, P_p^{l_p}\right\}$ where $P_p^{l_p}$ represents the trigger/factor p at level l_p due to which a change in the physical state of the equipment occurs, and i ranges from 1 to n. This means, based on the combinations of p and l_p, there can be n possible states of the equipment. For example, for equipment like a spindle, let factors like load, rotational speed, and health be the identified factors causing a change of states. For the load factor ($p = 1$), let the identified levels be 2kw, 4kw, 5kw, 10kw, thus $l_1 = 4$. For the factor of rotation speed ($p = 2$), if there are two levels identified in which the equipment has operated, for example, 200 RPM and 600 RPM, $l_2 = 2$. Similarly, if the equipment has experienced two health ($p = 3$) modes so far, $l_3 = 2$. Thus, for three factors with three, two, and two levels each, the number of states becomes, $n = 12$. Let samples in each state be represented by $S_i = \left\{s_{ij}\right\}_{j=1}^{j=m_i}$ were m_i represents the number of samples in S_i.

When an equipment initiates operation in a production facility, the initial physical state of the machine can be utilized for model training. The objective is to enable the stimulus-aware capability for the equipment's digital twin that can commence learning from the outset of utilization. As

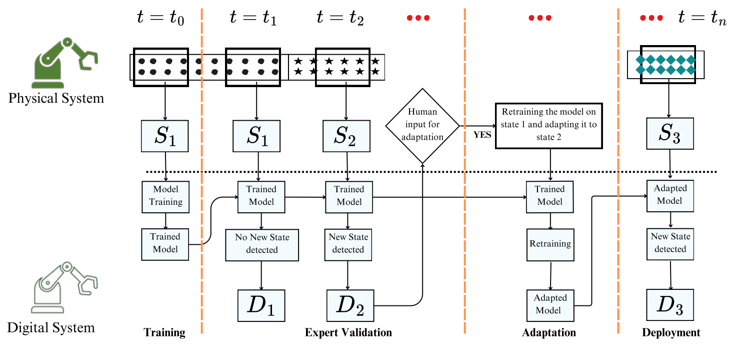


Figure 4 Overview of proposed four-stage learning framework

in real-life complex production systems, the availability of labeled data is generally very limited, and an unsupervised learning approach is used as the foundation for model development. Samples are used from continuous observation in the physical space in a four-phased learning framework. These phases are defined as follows: training, expert validation, adaptation, and deployment, as shown in Figure 4. In the first phase, a two-stage model that includes pattern recognition and change detection model is trained using the first state that occurs. In the second phase, the model is used for detection under expert validation. As a new state is detected and validated by the expert user, it is used as the target state for domain adaptation in phase 3. The adapted model is then deployed in phase 4. Figure 4 further gives an overview of the developed framework to detect changes in the physical state and update the digital state in the digital twin. A detailed discussion of each of the four phases of the framework is provided below.

3.1.1 Training Phase

In the proposed framework, an unsupervised 1D Convolutional Autoencoder with PCA-T^2 is used in the training phase for pattern learning and change detection. Autoencoders are unsupervised feed-forward neural networks that contain an input layer, multiple hidden layers, and an output layer. An autoencoder consists of two modules: an encoder and a decoder.

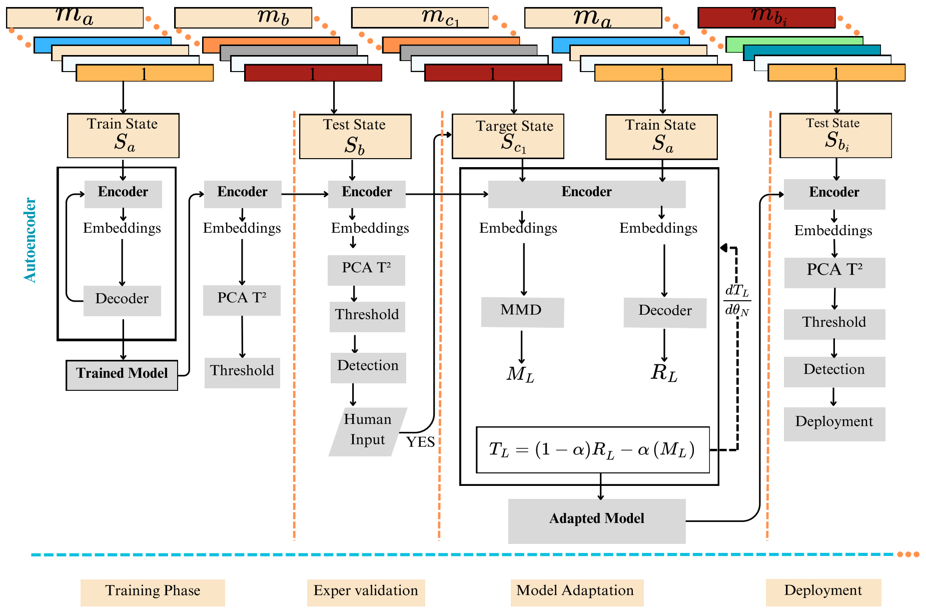


Figure 5 Detailed representation of the proposed four-phase learning framework for scalable state detection

In this study, the autoencoder is represented by f_{ae} and the other two modules, the encoder, and decoder, are represented by f_{en} and f_{de} respectively. The state used for the training of the model is represented by S_a and the state used for testing the model is represented by S_b. Figure 5 shows the schematic of the proposed training pipeline. The sample s_{aj} of the train state S_a is passed through f_{en} that encodes the sample into a lower dimensional latent representation or embedding represented by $Z_{aj} = f_{en}(s_{aj})$. f_{de} reconstructs the latent representation to the reconstructed sample represented by $\hat{s}_{aj} = f_{de}\left(Z_{aj}\right)$. According to the literature, the equation for the loss function can be expressed as $f_{de}[\Delta(s_{aj}, (f_{en}(\hat{s}_{aj}))]$ where f represents the encoder function while g represents the decoder function $\Delta(.)$ is the chosen metric distance. The purpose of the loss function to find the weights in the network that give you the smallest difference according to some metric $\Delta(.)$ between s_{aj} and $\hat{s}_{aj}$ distance. The purpose of the loss function to find the weights in the network that give you the smallest difference according to some metric $\Delta(.)$ between s_{aj} and $\hat{s}_{aj}$.Two loss functions are widely used for autoencoders: Mean Squared Error (MSE) and Binary Cross-Entropy (BCE). For this study MSE is used since the data was continuous and goal was to minimize the squared difference between the inputs and outputs. The loss function is represented by $\sum_{j}^{m_a} ||s_{aj} - \hat{s}_{aj}||_2^2$. Once the autoencoder model is trained, the training samples are again passed to the trained encoder to obtain the latent representation Z_{aj}. These representations are utilized to train a PCA-T^2 model for change detection. The training of the PCA-T^2 model involves calculating the principal component directions in the feature space that captures the most variance. The PCA-T^2 model is applied to the train state latent representation samples to reduce its dimensionality by retaining 95 percent of the variance in the encoded data. A T^2 statistic is then evaluated for trained samples. To classify the test state as a different state from the training state, a chi-square distribution is used, and a critical threshold value is determined for a confidence level of 95 percent. After the model is trained, the expert validation phase follows.

3.1.2 Expert validation Phase

In this phase, a test state S_b is passed through the trained encoder $Z_{bj} = f_{en}(s_{bj})$ to obtain the latent representation of that state and these latent representations are passed to the trained PCA model. The PCA-T^2 model uses its principal components to project representations of the data and generate scores for each test sample. These scores indicate how much variation is explained by each principal component for a specific sample. The scores are then squared to calculate T^2 values. These values show how much each test sample deviates from the training data distribution in the latent space. The accuracy is determined by finding the proportion of test

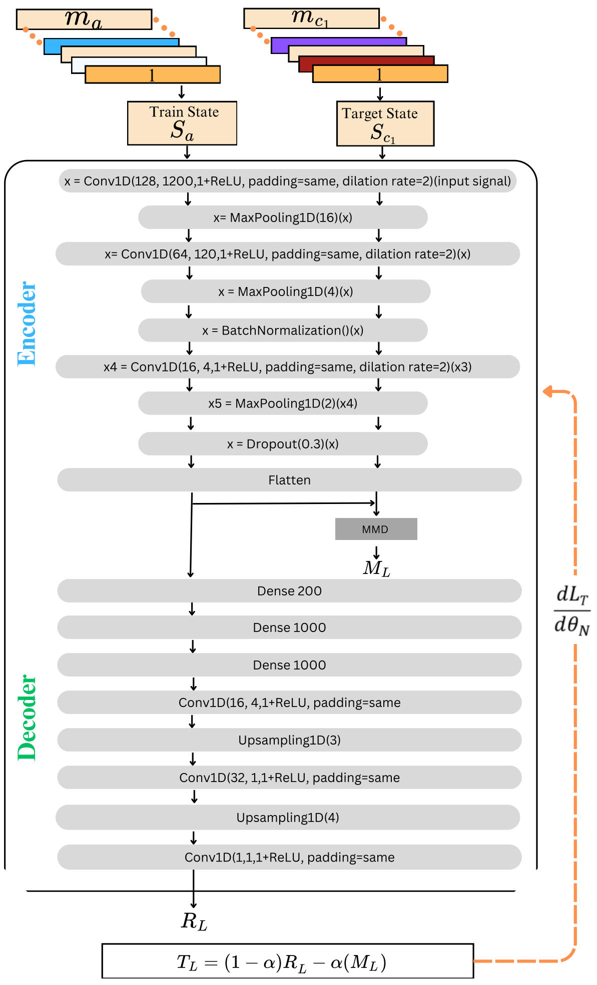

m_a
1
m_{c_1}
1
Train State S_a
Target State S_{c_1}
Encoder
x = Conv1D(128, 1200,1+ReLU, padding=same, dilation rate=2)(input signal)
x= MaxPooling1D(16)(x)
x= Conv1D(64, 120,1+ReLU, padding=same, dilation rate=2)(x)
x = MaxPooling1D(4)(x)
x = BatchNormalization()(x)
x4 = Conv1D(16, 4,1+ReLU, padding=same, dilation rate=2)(x3)
x5 = MaxPooling1D(2)(x4)
x = Dropout(0.3)(x)
Flatten
MMD
M_L
Decoder
Dense 200
Dense 1000
Dense 1000
Conv1D(16, 4,1+ReLU, padding=same
Upsampling1D(3)
Conv1D(32, 1,1+ReLU, padding=same
Upsampling1D(4)
Conv1D(1,1,1+ReLU, padding=same
R_L
$T_L = (1-\alpha)R_L - \alpha(M_L)$
$\frac{dL_T}{d\theta_N}$

Figure 6 Proposed Autoencoder network for domain adaptation

samples whose T^2 values exceed the threshold calculated in the training phase. If the model detects a new state, it is then validated by the expert. This new state is used as the target state and is represented by $S_i = S_{c_1}, S_{c_2}, S_{c_3} \dots\dots S_{c_n}$ and used for domain adaptation in the next phase.

3.1.3 Adaptation Phase

The purpose of the adaptation phase is to make use of the new information that is now available from the previous phase. This new data and its corresponding validated label are used to adapt the model for increased scalability. For performing domain adaptation, a custom loss function is defined that maximizes the discrepancy between the two states based on the learnt knowledge that they are different. The network architecture is shown in Figure 6. It comprises of two aspects: 1) reconstruction loss and 2) Maximum Mean Discrepancy (MMD) loss. The first element of loss represents the reconstruction loss given by: $R_L = \sum_j^{m_a} ||s_{aj} - \hat{s}_{aj}||_2^2$.The symbol $|.|_2^2$ indicates that the norm of a vector is simply the square root of the sum of the squares of the components. This loss function applies in nearly all scenarios, regardless of the output layer activation function choice or the method used for normalizing the input data. To find the minimum of L_{MSE} for $s_{aj} = \hat{s}_{aj}$ the derivative of L_{MSE} with respect to a specific observation j is calculated. The minimum is

found when the condition $\frac{\partial L_{MSE}}{\partial \hat{s}_{aj}} = 0.$ is met for all j = 1, . . ., m_a. Setting this derivative to zero that is

$$= \frac{\partial L_{MSE}}{\partial \hat{s}_{aj}} = -\frac{2}{m_a}\left(s_{aj} - \hat{s}_{aj}\right) = 0$$

$$= (-\frac{2}{m_a}\left(s_{aj} - \hat{s}_{aj}\right)) = 0$$

After solving the critical point is get:

$$= s_{aj} = \hat{s}_{aj}$$

This result confirms that the loss function is minimized when the predicted value $\hat{s}_{aj}$ equals the actual value s_{aj}, that means the model has learned to construct the input data. And to ensure that the critical point is minimum second derivative is calculated that is needed to show that $\frac{\partial^2 L_{MSE}}{\partial \hat{s}_{aj}^{\ 2}} > 0$. This is easily proved as we have $\frac{\partial^2 L_{MSE}}{\partial \hat{s}_{aj}^{\ 2}} = \frac{2}{m_a}$ that is greater than zero, therefore conforming that for $s_{aj} = \hat{s}_{aj}$ we indeed have a minimum of the loss function. The second element of loss represents MMD loss, calculated between the target states, S_{c_1} and the train state S_a. MMD is defined as the squared distance between the kernel embeddings of marginal distributions in the Reproducing Kernel Hilbert Space (RKHS), The RKHS is endowed with a characteristic kernel k that is defined as H. Formally, the MMD is defined as follows[#ref]: $MMD_k(Z_{aj}, Z_{c_1 j})$, is defined as an unbiased estimator of $MMD_k(Z_{aj}, Z_{c_1 j})$, which can be calculated as follows:

$$MMD_k(Z_{aj}, Z_{c_1 j}) = ||E_a[\emptyset(Z_{aj})] - E_{c_1}[\emptyset(Z_{c_1 j})]||^2_{H_k}$$

where $\mathrm{H_k}$ denotes the RKHS endowed with the characteristic kernel k mapping function φ(.). and E_{c_1} [.] is the expectation under the distribution c_1 and similarly E_a [.] is the expectation under the distribution a. $\mathrm{Z}_{c_1 j}$ and Z_{aj} is the latent representation of target states S_{c_1} and S_a respectively.

$$= \frac{1}{min(m_{a,} m_{c_1})^2} [\sum_{j}^{min(m_{a,} m_{c_1})} \sum_{j}^{min(m_{a,} m_{c_1})} k(Z_{ai}, Z_{aj})$$

$$+ \sum_{j}^{min(m_{a,} m_{c_1})} \sum_{i}^{min(m_{a,} m_{c_1})} k(Z_{c_1 i}, Z_{c_1 j})$$

$$+ -2 \sum_{j}^{min(m_{a,} m_{c_1})} \sum_{i}^{min(m_{a,} m_{c_1})} k(Z_{ai}, Z_{c_1 j})]$$

In this study on domain adaptation, the objective is to maximize the latent representation of two states so that the model can learn to distinguish between the two distributions by increasing their discrepancy. To achieve this, the Maximum Mean Discrepancy (MMD) loss $MMD_k(Z_{aj}, Z_{c_1 j})$, can be expressed with a negative sign as $-MMD_k(Z_{aj}, Z_{c_1 j})$. The MMD loss calculated between the two latent representations is represented by $-M_L$. The Total loss function during the domain adaptation is represented by $T_L = (1 - \alpha)R_L + \alpha(-M_L)$ which is minimized during the training loop. The penalty coefficients are represented by α, which plays a crucial role in the trade-off between MMD loss and reconstruction loss. Using the information learned from the previous phase about the two states being different from each other, the goal is to maximize the distribution discrepancy between the latent representation of these states. The addition of the MMD loss in the custom loss has improved the performance of the model. The penalty coefficient, represented by α, represents the trade-off between MMD loss and reconstruction loss. Using the information learned from the previous phase about the two states being different from each other, the goal is to maximize the distribution discrepancy between the latent representation of these states. A learning rate of 0.001 is utilized during training, the penalty coefficient is chosen as 0.2 and the stopping criteria for overall loss change in an epoch of less than 0.1 percent is used.

3.1.4 Deployment Phase

After performing domain adaptation, the adapted model is deployed. When a new state is encountered, the model detects the state and updates the digital system and continues this process throughout its operational life. The next section describes the various experiments performed against multiple cases and a description of the datasets used to validate the model's effectiveness and the practicality of the proposed approach.

Chapter 4 Gearbox Test Rig Setup for Data Acquisition

4.1 Gearbox Test Rig Setup for Data Collection

The concept of 'states' from a stimulus-aware perspective is relatively new, and there are not many datasets that offer sufficient production data in terms of signals needed to interpret the 'states' of a system. In complex production systems, there are many critical components such as gears, spindles, and bearings. Most existing datasets collected from these components focus on fault detection and diagnosis or provide only basic labels for their operational states during production. Consequently, these datasets are often not very useful due to a lack of comprehensive signals.

Thus, we have created a new dataset using the experimental gearbox test rig setup, where we conducted numerous experiments that replicate the real-life operations of gearbox equipment.

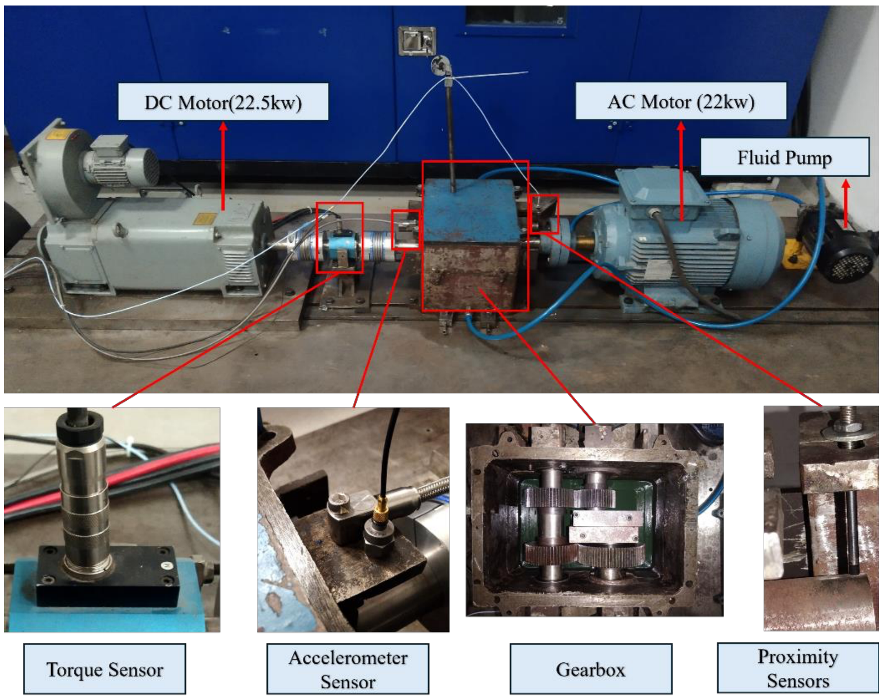


Figure 7 Experimental Gearbox Test Rig Setup

The experimental gearbox test rig was specifically developed to study vibration characteristics of different operating states in gear transmission systems. Gears are critical components in industrial machinery such as wind turbines, automobiles, and manufacturing equipment, where their failure can result in significant operational losses. To replicate real-world conditions, the gearbox test rig was designed to closely simulate the operating parameters typically observed in gearboxes, as illustrated in Figure 7. The primary objective of this experimental setup is to facilitate condition monitoring and system behavior analysis under varying operational conditions. The test rig supports load variation experiments, allowing us to understand the influence of loading conditions on equipment health conditions, and speed variation studies to examine how dynamic responses change with different operating speeds.

The operational procedures of the various components of the test rig are explained in detail below.

4.2 Working Procedure of the Gearbox Test Rig Setup

4.2.1 Drive System

The drive system of the gear test rig is designed to supply the necessary mechanical input power to the gearbox under test. In our experimental setup, a 22 kW AC induction motor serves as the primary driving unit, shown in Figure 8. The specifications of the motor are shown in Table 1.

Table 1 AC Motor Specification

AC MOTOR SPECIFICATION	
Brand Name	ABB
Voltage	415 V
Frequency	50 Hz
Speed	2932 rpm
Power (in kW)	22
Power (in hp)	30
Weight	152 kg

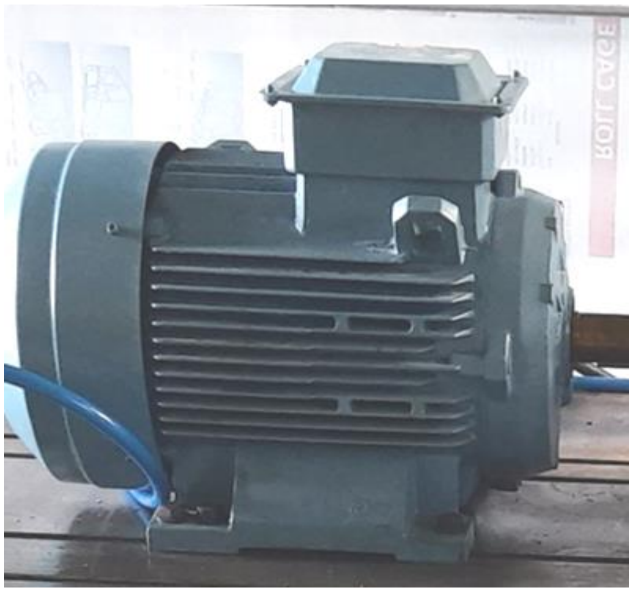


Figure 8 AC Motor Drive

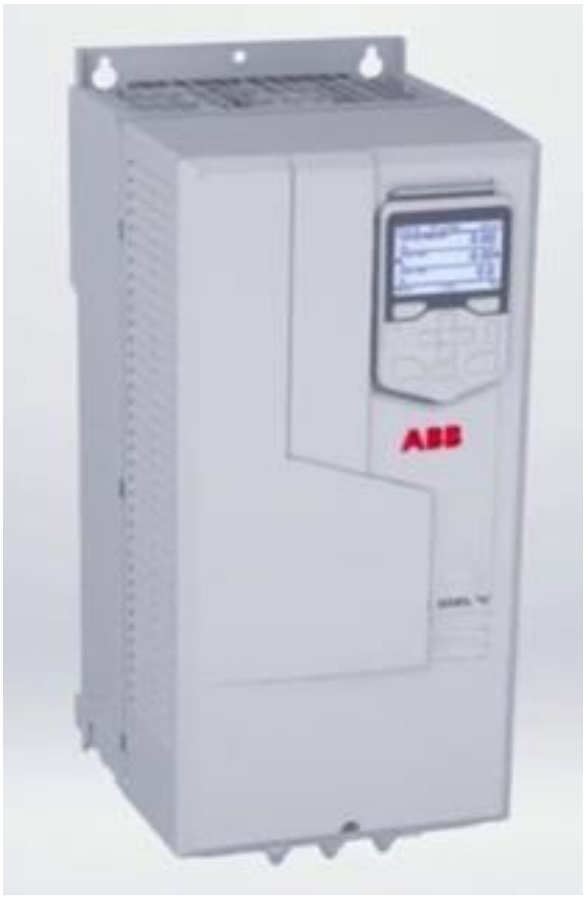


Figure 9 Variable Frequency Drive

This motor provides sufficient torque and speed range required to operate the test gears under various conditions. To enable flexible and precise control over the motor's operation, it is integrated with a Variable Frequency Drive (VFD) as shown in Figure 9, allowing continuous adjustment of motor speed across a broad range. This configuration makes it possible to replicate different real-world operating conditions, including low-speed, high-torque, and high-speed, low-torque scenarios. The AC motor is connected to the test gearbox through a suitable coupling mechanism, ensuring efficient transmission of power with minimal mechanical losses and proper alignment between the driving and driven shafts. The use of an AC motor combined with VFD control offers stable, repeatable, and highly adjustable driving conditions, which are crucial for conducting systematic experimental studies.

4.2.2 Variable Load Control System

The load system plays a crucial role in applying a controlled resisting torque to the gearbox output shaft, thereby simulating the operational loads typically encountered during normal gearbox operation. To achieve this, a DC shunt motor is utilized as the load device, the specifications shown in Table 2. The DC motor operates in generator mode, opposing the motion imparted by the drive system and generating a resistive load on the gearbox. The variable loads are applied through the resistive load bank, as illustrated in Figure A. By adjusting the field current and the load resistance connected to the motor, the resisting torque can be precisely controlled, enabling the simulation of a wide range of loading conditions,

Table 2 DC Motor Specification

DC MOTOR SPECIFICATION	
Brand Name	Crompton Greaves
Armature Voltage	440 V
Field Voltage	220 V
Speed	1500 rpm
Power	22 .5 kW
Weight	170 kg

from light loads to heavy loads and even overload scenarios. This precise control over the load allows for the study of gearbox performance not only under steady-state conditions but also under varying and transient loading conditions. The smooth and accurate variation of the load provided by the DC motor setup is particularly advantageous for investigating the behavior of the gearbox in conditions that closely resemble real-world applications, such as sudden load changes, fluctuating torque demands, and endurance testing. This capability ensures that the gearbox's performance can be thoroughly evaluated across a variety of operational scenarios, providing

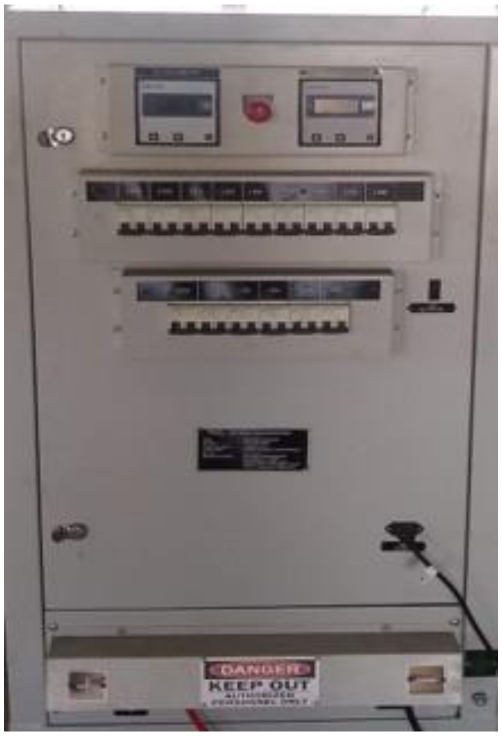


Figure A Variable Load Cell

Table 3 Specification of Gears

Gears	G1	G2	G3	G4
PCD	92.1	121.06	121.06	161.05
Tooth Thickness	4.26	4.5	4.5	4.5
Face Width	40.5	39.9	40.5	40.5
Pressure Angle	20 Degree			
Module	2			

valuable insights into its durability and reliability under different operating conditions.

4.2.3 Description of Gearbox Dimensions

The test rig utilizes a two-stage gearbox, which consists of a total of four spur gears. Data can be collected from the gearbox based on the health condition of these gears. A specific test gear, located close to the sensors, can be selected to ensure a strong signal from the gearbox. The data can be gathered under various operating conditions, such as resistive loads provided by a resistive load bank and different operating speeds that can be adjusted using a variable frequency drive (VFD). The healthy gears can operate until failure, during which vibration signatures can be recorded through sensors. Additionally, artificial faults can be induced for various research purposes. The types of artificial faults added will depend on the specific research objectives. The specifications of the gears used in the system are summarized in Table 3. All gears have an involute gear profile with a module of 2 mm and a pressure angle of 20 degrees. The system incorporates a 2-stage reduction. The individual gear parameters, including pitch circle diameter (PCD), tooth thickness, and face width, are listed for Gears G1, G2, G3, and G4 in Table 3.

4.2.4 Sensors used for data collection

To simulate the behavior of complex manufacturing systems and enable realistic data collection, the gearbox test rig was equipped with multiple types of sensors. In industrial settings, machines typically rely on a variety of sensors to monitor different physical parameters such as vibration, acoustic emissions, torque, and displacement. To replicate this real-life configuration, the experimental setup included a diverse set of sensors, each selected for its ability to capture specific operational characteristics. These sensors are shown in Figure 7 are as follows:

1. Kistler Triaxial IEPE Accelerometer (B & K 4534 B): This sensor is used to capture triaxial vibration data from the system. It operates over a wide frequency range of 0.2 to 12,800 Hz and has a sensitivity of 1 mV/ms^2. Its triaxial measurement capability is essential for understanding complex vibration patterns in rotating machinery, making it ideal for fault detection and condition monitoring.
2. Kistler Piezotron Acoustic Emission Sensor (8152 C1): to detect high-frequency acoustic emissions resulting from material stress, friction, or crack formation, this sensor operates in the 100 to 900 kHz range (±10 dB). With a sensitivity of 48 dB referenced to 1 V/(m/s), it is highly effective in capturing transient acoustic signals that are often early indicators of mechanical faults.
3. Kistler Shaft Torque Sensor (4520A500): This torque sensor measures the dynamic load on the rotating shaft, with a measuring range of 1 to 1000 Nm and a frequency response from 1 to 5 kHz (±3 dB). It is vital for evaluating power transmission efficiency and detecting torque fluctuations due to misalignment, load changes, or mechanical wear.
4. Bentley Nevada Proximity Sensor (3300 XL 5/8 mm): This proximity sensor provides non-contact displacement measurements of the shaft, operating in the 0 to 6.5 kHz frequency range. It features a supply sensitivity of less than 2 mV change in output voltage per volt change in input voltage, enabling precise monitoring of shaft position, imbalance, and runout in real-time.

In this chapter, we have discussed the complete experimental gearbox setup from which we have created the new dataset by conducting numerous experiments that replicate real-life operations of equipment. The created datasets address data gaps in complex manufacturing systems by providing rich signals that represent production states. Most existing datasets focus on fault detection or offer only basic state labels, limiting their usefulness.

Additionally, after a significant search, we arrived at the most relevant real-life datasets that could be used to showcase the benefit of the proposed approach as well as validate it. Two such real-life public datasets have been chosen that span different equipment, data types, data collection frequencies, fault patterns, etc. The description of the datasets are presented in the next section.

Chapter 5 Dataset Description and Validation Methodology

5.1 Dataset Description

In production economics, data is crucial for monitoring and detecting operating states. These data are collected in various forms, including numerical data such as torque, vibration, and pressure, which are gathered from sensors embedded in machinery to detect faults and abnormalities. Textual data, like machine logs and maintenance reports, provides insights into historical performance and supports predictive maintenance. When sensor installation is challenging, visual data [105] Cameras and vision systems help monitor machinery through images or video, assisting in quality control and defect detection. These data sources, including IoT sensors, PLCs, SCADA systems, wearables, and ERP/MES systems, work together to provide real-time insights, improve decision-making, and enhance operational efficiency in manufacturing environments. In this study, the proposed model specifically processes numerical data. The proposed method is validated across three real-life datasets representing a variation in equipment, signals, and fault modes. The descriptions of the datasets used are shown in Table 4. Dataset I is a newly created real-life dataset developed by the authors in the Smart Manufacturing Lab using an experimental test rig. While Datasets II and III are publicly available benchmark datasets. The description of these datasets used are shown in

Table 4 Datasets Used for the validation

Dataset	Equipment Description	Observation Signal	Fault Present
I	Two-stage Gearbox with spur gears and no faults	Torque	No
II	Deep Groove Ball Bearing with multiple operations and multiple faults	Accelerometer	Yes
III	Axial Ball Bearing with multiple fault operations and multiple faults	Accelerometer	Yes

Table 4. Dataset 1 contains only healthy operating states and serves as a baseline for normal condition monitoring. Dataset 2 is the publicly available Case Western Reserve University (CWRU) bearing dataset, which includes both healthy and faulty states. The monitored failure modes in this dataset include ball faults and inner race faults, captured under various operating conditions, providing diverse fault scenarios for validation. Dataset 3 is a bearing degradation dataset in which faults evolve progressively over time, enabling the approach to be tested in a realistic scenario where fault severity increases gradually, reflecting the typical degradation process in industrial equipment.

5.1.1 Dataset I

A new experimental two-stage gearbox test bed with spur gears is designed to validate the proposed method. Figure 10 shows the designed test bed, which comprises an AC motor (22 kW) connected to the gearbox for driving the gearbox, a lubricating pump (3 horsepower) used for circulating oil inside the gearbox, and a DC motor (22.5 kW) used to apply a resistive load to the gearbox shaft. A frequency drive and a variable resistive load bank are used to provide variable operating settings for the gearbox. A Kistler shaft torque sensor is connected to measure the resisting torque under different operating conditions. The sensor (4520A500) has a measuring range of 1-1000 mm and a sensitivity of ±3 dB (1-5 kHz). The

Table 5 Dataset I State Description

State (S_i)	Load (p_1)	RPM (p_2)	Condition (p_3)
S_1	P_1^1 = 2kW	P_2^1 = 300 RPM	P_3^1 = Healthy
S_2	P_1^1	P_2^2 = 600 RPM	P_3^1
S_3	P_1^1	P_2^3 = 900 RPM	P_3^1
S_4	P_1^2 = 4kW	P_2^1	P_3^1
S_5	P_1^2	P_2^2	P_3^1
S_6	P_1^2	P_2^3	P_3^1
S_7	P_1^3 = 6kW	P_2^1	P_3^1
S_8	P_1^3	P_2^2	P_3^1
S_9	P_1^3	P_2^3	P_3^1

sensors are connected to a National Instruments DAQ system that converts the analog signal to a digital signal. Data is collected from nine experiments at varying loads and RPM when all the spur gears are in healthy condition, with a sampling rate of 14 kHz. The setup is shown in Figure 11. The data collected was stored in MATLAB’s .mat file format, enabling seamless integration with MATLAB for further processing. The details of the

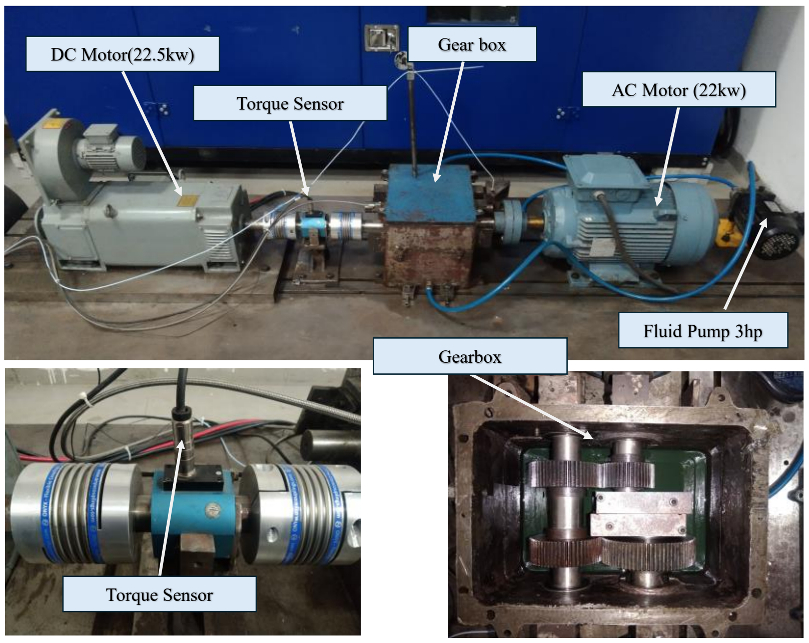


Figure 10 Experimental gearbox test rig setup for a dataset I

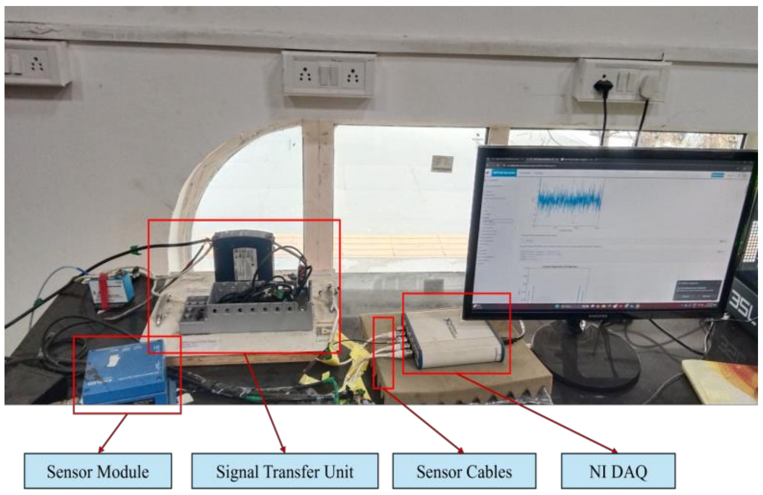


Figure 11 Data collection setup

collected data are provided in Table 5. The torque signal is used as an observation signal for model validation.

5.1.2 Dataset II

Dataset II is a rolling bearing dataset provided by the Bearing Data Center of Case Western Reserve University [106]. The dataset is made up of multi-variate vibration signals generated by a bearing test rig, as shown in Figure 12. The experimental setup is made up of three primary components: a 2-horsepower (hp) motor, a torque transducer/encoder, and a dynamometer. The motor shaft is supported by JEM SKF 6205-2RS bearings. Accelerometers capture bearing data under four load circumstances (0, 1, 2, and 3 hp) at a sample rate of 12 kHz. Depending on the load, the motor rotates at speeds ranging from 1730 to 1797 rpm. The vibration data was collected from the drive end of the motor under different health conditions: normal (H), outer race fault (OF), inner race fault (IF), and ball fault (BF). Electro-discharge machining is used to generate three types of defects with widths of 7 mils, 14 mils, and 21 mils, where 1 mil = 0.001 inch. To validate the concept, this study examined nine different bearing conditions or states. The dataset includes three parameters: Load,

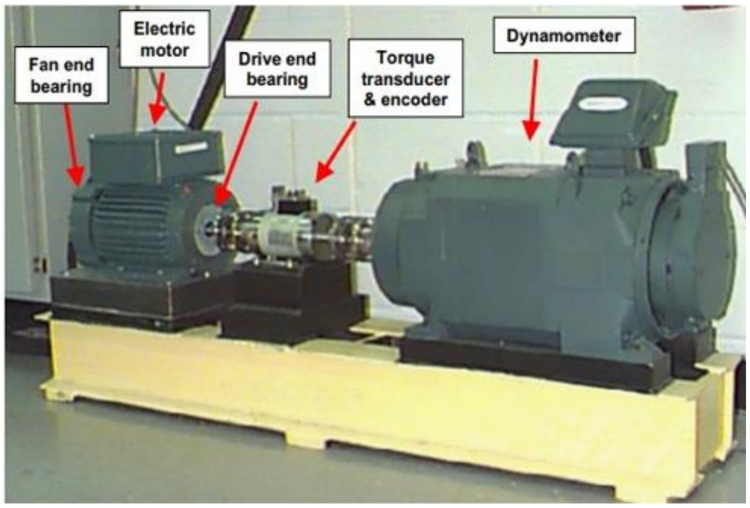


Figure 12 Experimental Setup for Dataset II

RPM, and Fault, which determine the state of the equipment, as shown in Table 6

5.1.3 Dataset III

This axial bearing dataset includes time-series vibration measurements [107]. An experimental test setup [108] based on the FALEX multi-specimen test bench and data acquisition system, comprising an accelerometer (model PCB 356A32), an axial loading mechanism (up to 8.8 kN), and a closed control for the bearing spindle is used. The data is based

Table 7 State description for Dataset III

State (S_i)	Load (p_1)	RPM (p_2)	Fault(p_3)
S1	P_1^1 = 5kW	P_2^1 = 60 RPM	P_3^1
S2	P_1^1	P_2^2 = 500 RPM	P_3^1
S3	P_1^2 = 8.8kW	P_2^2	P_3^1
S4	P_1^2	P_2^2	P_3^2 = Inner Fault (1mm)
S5	P_1^1	P_2^2	P_3^2
S6	P_1^1	P_2^1	P_3^3 = Inner Fault (2.1mm)
S7	P_1^1	P_2^1	P_3^4 = Inner Fault (3.8mm)
S8	P_1^1	P_2^2	P_3^4
S9	P_1^2	P_2^2	P_3^4

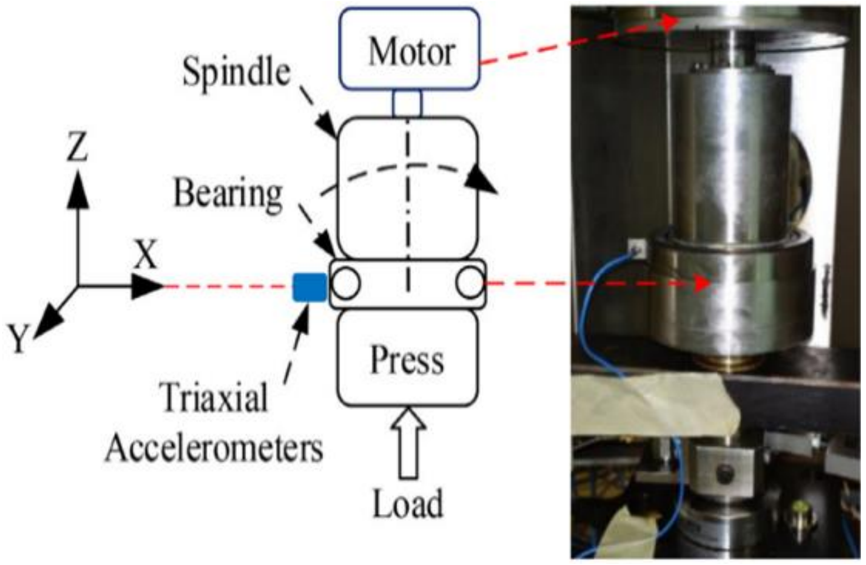


Figure 13 Experimental Setup for the dataset III

on a common aerospace bearing model FAG QJ212TVP. The bearing under test was mounted vertically on the test bench, with a vertical axial load applied to the inner race to match heavy loading at low-speed actuation circumstances in aerospace and energy systems. The fault type is a fatigue spall on the inner race (IF) or outer race (OF). These faults were created by spark-erosion machining. The bearing test rig is shown in Figure 13 with three accelerometers, where a model PCB 356A32 was used to measure the triaxial vibration signal with a sampling rate of 25.6 kHz. Vibration measurement along the x-axis is used in this study. Table 7 provides an overview of the nine states considered for validation, with each state representing a specific operational or fault scenario within the bearing system.

5.2 Validation Methodology

In a complex production system, such as the production systems described before, a change in state can arise from various factors, including operating parameters, environmental factors, equipment health, etc. To evaluate the model's performance, multiple cases are designed as described in Table 8 for dataset I. While the proposed framework includes a training phase first and then a model adaptation phase, the final model to be deployed results from the model adaptation phase itself. This model adaptation phase includes three major elements the state used for representation learning, the state used for modeling change detection, and the state used for domain adaptation. Since the proposed method aims to develop the stimulus-aware digital twin at the beginning of the equipment operation, two important factors need to be considered during validation: 1) minimal data availability for training and 2) the possibility of occurrence of any operating state for

Table 8 Experiments for validation of an algorithm

Dataset	Case No.	f_{ae} Trained	PCA Trained	Domain Adaptation	No. of Target State	No of Subcases
I	1	Train State	Train State	No	0	4
	2 (Proposed)	**Train State**	**Train State**	**Yes**	**1**	**12**
	3	Train State	Target State	Yes	1	12
	4	Train State	Train State	Yes	2	12
	5	Train State	Target State	Yes	2	24

training. Given this context, for the validation design methodology shown in Table 8, the third and fourth column represent the possible states in which the autoencoder and PCA-T2 models can be trained, with varying choices from available states. Further, the fifth column considers the possibility of having a different number of target states for model adaptation. This includes no domain adaptation, domain adaptation with one state, and domain adaptation with two states to explore the impact of using multiple states to improve generalizability. Each of the states is represented by 181 samples, with each sample having (12000, 1), were used to train the autoencoder. The encoder compresses these signals into a lower-dimensional representation of shape (128). These are the encodings generated by the encoder model. These encodings serve as the input to the PCA model. PCA was trained on this 2D array of encoded vectors, and Hotelling's T^2 statistics were computed to capture variations in the data. For PCA, the number of principal components was chosen to retain 95% of the total variance. In Table 8, case 1 assumes the availability of only one state for the entire model training with no domain adaptation. Cases 2 and 3 assume the availability of two states – the train and target states. Case 2 is the proposed method and is highlighted in bold throughout the validation study. Case 3 explores the use of the target state for training the state change detection model PCA-T2. Cases 4 and 5 are similar to cases 2 and 3, considering domain adaptation with two target states. For these two cases, the loss function is modified to include the contribution of two more MMD loss elements from the adaptation between the training and two target states. The loss function is given by $T_L = (1 - \alpha)R_L - \alpha(M_L + M_{L_1} + M_{L_2})$. To incorporate the possibility of occurrence of any operating state for the train and target state, a sampling method is used, followed by statistical hypothesis testing. Since it is difficult to examine the model's performance with each possible state in each case, states are sampled from the dataset to form subcases. The number of subcases for each case is given in the last column of Table 8. These subcases explore the model's ability to detect a state change regardless of the state(s) used for modeling. The performance of each subcase is evaluated by calculating the state change detection accuracy over the dataset on the test states given in the subcase. As this is a sampled approach, each subcases and cases are hypothesized and evaluated for the presence of a statistical difference in the performance metric using t-test. The results and detailed comparison are discussed in the next chapter.

Chapter 6 Results and Discussion

6.1 Analysis of Case 1 for Dataset I

Case 1 represents the case when an autoencoder and PCA model are trained on the same state, and no target state is available/used for model adaptation. Three parameters (load, RPM, health condition) are considered to reflect changes in the equipment's state in dataset I. Table 9 presents the accuracy of the model in distinguishing across different states without domain adaptation. For this case, out of the nine possible states for training Table 9, only four states (S_1, S_4, S_6, S_8) are sampled for performance evaluation, forming four subcases. The autoencoder and the PCA $-T^2$ model is trained on the same state in each subcase. The second column of Table 9 lists these subcases under case 1, while the third and fourth columns represent the state used for modeling. The fifth column of Table 9 represents the model's accuracy in detecting the specified test state. The last column of Table 9 shows the average accuracy for each subcase across all test states. It is important to note that while only one state is considered for training, all other states are used for testing. For example, for subcase 1.1, the f_{ae} is trained on S1, and all states are passed to the encoder module of the trained f_{ae} to generate the latent representation. These latent representations are passed to the PCA model trained on the same state, S_1. The PCA transforms the latent representation of each state into a lower dimension. The top principal component for each state for the given samples is shown in Figure 14. From the perspective of state change detection, the figure illustrates that some encodings exhibit separation, indicating that the model has effectively learned the features necessary to differentiate between the states. However, some state encodings are overlapping, suggesting that the model may face difficulties in differentiating these states. The metric of detection accuracy is used to validate the model performance. In another example, for case 1.2, the f_{ae} and PCA-T^2 are trained on S_4, and the accuracy is then calculated across all the remaining states that can occur in the deployment phase. The

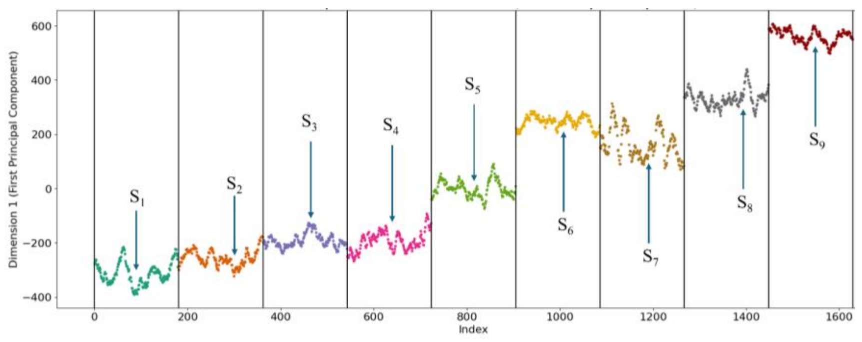


Figure 14 Encoding representation of each state from the dataset I

achieved accuracy for detecting S9 in subcase 1.2 is 0.768, which means the model in subcase 1.2 can detect state S9 with a 76.8 % accuracy. Thus, for case 1.2, the average accuracy with which the model can detect all states is 77.9%, as given in the last column. Two important observations can be made here: 1) the difference in the detection performance across test states for a given training state, and 2) the difference in the detection performance of the test states across training states. This shows that the choice of data available for training has an impact on the model's performance in a real-life environment while creating stimulus-aware digital twins. Thus, an average accuracy across test states is calculated as shown in Table 9, and the achieved mean and standard deviation of the average accuracy are reported for all subcases of case 1.

6.2 Overall Results for Dataset I

6.2.1 Accuracy results for Dataset I

Table 10 presents the overall results for dataset I, which comprises all 64 subcases for all cases repeated 5 times each. The first column represents the case, the second column provides information about the subcases for the specific case. The third and fourth columns show the state on which the autoencoder and the PCA-T^2 model are trained, respectively. The fifth and sixth columns represent the target states available for domain adaptation. The next five columns show that each subcase is repeated five times to calculate the mean accuracy, as explained

Table 9 Results for Dataset I Case 1: No Domain Adaptation

				Test State									
Case	Subcase	f_{ae} Trained	PCA Trained	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	Average accuracy
1	1.1	S_1	S_1	-	0.215	0.95	0.84	1	1	1	1	1	0.876
	1.2	S_4	S_4	1	1	1	-	0.204	0.724	1	0.525	0.768	0.779
	1.3	S_6	S_6	1	1	1	1	0.11	-	1	1	1	0.873
	1.4	S_8	S_8	1	1	1	1	1	1	0.61	-	0.32	0.866
												Mean	**0.849**
												St. Dev.	0.078

in Table 9. This repetition helps evaluate the model's stability and consistency in accurately detecting states. The second last column of the table displays the average accuracy of five repeated evaluations for each subcase. The table's last column shows the model's overall mean accuracy for each case.

As it can be observed from Table 8, the number of subcases has increased for case 2 as compared to case 1, as multiple combinations of train and target state are possible. Keeping the same sampled states for case 2 as for case 1, i.e., S_1, S_4, S_6, S_8, the number of subcases considered is 12. Similar sampling and combination patterns are followed for cases 3, 4, and 5, resulting in the number of subcases as 12, 12, and 24, respectively. Similar to the observations from Table 9, the achieved performance of a model is dependent on the choice of training states. While the variation across trials is comparatively lesser, the variation across subcases is much higher. This further highlights the dependency of the model's performance on the state used for modeling and the configuration.

The overall mean accuracy for case 1 in dataset I, where no target state is used for domain adaptation and both models are trained on the same state, is 85%. In the proposed case 2, the observed state serves as the target state for domain adaptation, resulting in a considerable improvement in the model's performance over case 1. In this proposed case, the autoencoder and PCA-T2 models are trained using the same state and adapted to a different state, resulting in an overall mean accuracy increase to 98.6%. In case 3, where the PCA-T2 model is trained on the target state instead of the train state, the overall mean accuracy is 97.9%, almost similar to Case 2. Case 4 involves using two target states for multi-state domain adaptation. For this case, the PCA-T2 model is trained on the same state at which the autoencoder model is trained. The overall mean accuracy of the model in this case is 98.4%, which is similar to the single-state domain adaptation. Case 5 is also a multi-state domain adaptation, but the PCA-T2 model is trained on the target state instead of the train state, and for this case, the overall accuracy is 98.7%.

Table 10 represents the overall results for the datasets I

Case	Subcase	f_{ae} Trained	PCA Trained	Target state S_{c_1}	Target state S_{c_2}	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Average accuracy	Overall mean accuracy
1	1.1	S_1	S_1	-	-	0.874	0.876	0.876	0.881	0.883	**0.874**	0.850
	1.2	S_4	S_4	-	-	0.778	0.773	0.785	0.778	0.773	**0.773**	
	1.3	S_6	S_6	-	-	0.889	0.887	0.887	0.887	0.885	**0.887**	
	1.4	S_8	S_8	-	-	0.866	0.867	0.864	0.864	0.865	**0.867**	
2	2.1	S_1	S_1	S_4	-	0.920	0.974	0.943	0.999	1.000	**0.967**	**0.986**
	2.2	S_1	S_1	S_6	-	1.000	1.000	1.000	1.000	1.000	**1.000**	
	2.3	S_1	S_1	S_8	-	1.000	1.000	1.000	1.000	1.000	**1.000**	
	2.4	S_4	S_4	S_1	-	1.000	1.000	1.000	1.000	1.000	**1.000**	
	2.5	S_4	S_4	S_6	-	1.000	1.000	1.000	0.929	1.000	**0.986**	
	2.6	S_4	S_4	S_8	-	0.929	0.912	0.931	0.903	0.921	**0.919**	
	2.7	S_6	S_6	S_1	-	1.000	1.000	1.000	1.000	1.000	**1.000**	
	2.8	S_6	S_6	S_4	-	1.000	1.000	1.000	1.000	1.000	**1.000**	
	2.9	S_6	S_6	S_8	-	1.000	1.000	1.000	1.000	1.000	**1.000**	
	2.10	S_8	S_8	S_1	-	0.949	0.972	0.952	0.952	1.000	**0.965**	
	2.11	S_8	S_8	S_4	-	1.000	1.000	0.973	1.000	1.000	**0.995**	
	2.12	S_8	S_8	S_6	-	1.000	1.000	1.000	1.000	1.000	**1.000**	
3	3.1	S_1	S_4	S_4	-	1.000	1.000	1.000	1.000	1.000	**1.000**	0.979
	3.2	S_1	S_6	S_6	-	1.000	1.000	1.000	1.000	1.000	**1.000**	
	3.3	S_1	S_8	S_8	-	1.000	1.000	1.000	1.000	1.000	**1.000**	
	3.4	S_4	S_1	S_1	-	1.000	1.000	1.000	1.000	1.000	**1.000**	
	3.5	S_4	S_6	S_6	-	1.000	1.000	1.000	1.000	1.000	**1.000**	
	3.6	S_4	S_8	S_8	-	1.000	1.000	0.997	1.000	1.000	**0.999**	
	3.7	S_6	S_1	S_1	-	0.984	0.909	0.948	0.972	0.845	**0.932**	

	3.8	S_6	S_4	S_4	-	0.810	1.000	1.000	0.954	0.880	**0.929**	
	3.9	S_6	S_8	S_8	-	0.927	1.000	1.000	0.854	1.000	**0.956**	
	3.10	S_8	S_1	S1	-	0.920	0.981	0.919	0.919	1.000	**0.948**	
	3.11	S_8	S_4	S_4	-	1.000	1.000	1.000	1.000	1.000	**1.000**	
	3.12	S_8	S_6	S_6	-	0.970	0.970	0.997	0.977	1.000	**0.983**	
	4.1	S_1	S_1	S_4	S_6	1.000	1.000	1.000	0.982	1.000	**0.996**	
	4.2	S_1	S_1	S_4	S_8	1.000	1.000	1.000	1.000	1.000	**1.000**	
	4.3	S_1	S_1	S_4	S_8	1.000	0.882	0.952	1.000	0.972	**0.961**	
	4.4	S_4	S_4	S_1	S_6	0.960	1.000	0.923	0.964	1.000	**0.969**	
	4.5	S_4	S_4	S_6	S_8	0.964	1.000	0.923	0.964	1.000	**0.970**	
4	4.6	S_4	S_4	S_1	S_8	0.964	0.977	0.986	0.974	1.000	**0.980**	0.984
	4.7	S_6	S_6	S_1	S_4	0.940	1.000	0.920	1.000	0.980	**0.968**	
	4.8	S_6	S_6	S_4	S_8	1.000	1.000	1.000	1.000	0.991	**0.998**	
	4.9	S_6	S_6	S_1	S_8	1.000	0.983	0.939	1.000	1.000	**0.984**	
	4.10	S_8	S_8	S_1	S_4	0.999	0.969	0.973	0.980	0.969	**0.978**	
	4.11	S_8	S_8	S_1	S_6	1.000	1.000	0.989	1.000	1.000	**0.998**	
	4.12	S_8	S_8	S_4	S_6	1.000	1.000	1.000	1.000	1.000	**1.000**	
	5.1	S_1	S_4	S_4	S_6	1.000	1.000	0.990	1.000	0.979	**0.994**	
	5.2	S_1	S_4	S_4	S_8	1.000	1.000	1.000	1.000	1.000	**1.000**	
	5.3	S_1	S_6	S_6	S_8	1.000	1.000	1.000	1.000	1.000	**1.000**	
	5.4	S_4	S_1	S_1	S_6	1.000	0.998	1.000	0.999	1.000	**0.999**	
	5.5	S_4	S_6	S_6	S_8	1.000	1.000	1.000	1.000	1.000	**1.000**	
	5.6	S_4	S_1	S_1	S_8	0.998	0.998	1.000	0.999	1.000	**0.999**	
	5.7	S_6	S_1	S_1	S_4	1.000	1.000	1.000	1.000	1.000	**1.000**	
	5.8	S_6	S_4	S_4	S_8	0.985	0.963	1.000	0.913	0.921	**0.956**	
	5.9	S_6	S_1	S_1	S_8	0.981	0.999	1.000	0.982	1.000	**0.992**	
	5.10	S_8	S_1	S_1	S_4	0.959	0.838	0.855	0.890	0.886	**0.886**	
	5.11	S_8	S_1	S_1	S_6	0.883	0.930	0.898	0.900	0.892	**0.901**	
5	5.12	S_8	S_4	S_4	S_6	1.000	1.000	1.000	1.000	1.000	**1.000**	0.987
	5.13	S_1	S_6	S_4	S_6	1.000	1.000	1.000	1.000	1.000	**1.000**	
	5.14	S_1	S_8	S_4	S_8	1.000	1.000	0.997	1.000	0.975	**0.994**	
	5.15	S_1	S_8	S_6	S_8	1.000	1.000	1.000	1.000	1.000	**1.000**	
	5.16	S_4	S_6	S_1	S_6	1.000	1.000	1.000	1.000	1.000	**1.000**	
	5.17	S_4	S_8	S_6	S_8	1.000	1.000	1.000	0.990	0.998	**0.998**	
	5.18	S_4	S_8	S_1	S_8	1.000	1.000	1.000	1.000	1.000	**1.000**	
	5.19	S_6	S_4	S_1	S_4	1.000	0.999	1.000	1.000	1.000	**1.000**	
	5.20	S_6	S_8	S_4	S_8	1.000	1.000	1.000	1.000	1.000	**1.000**	
	5.21	S_6	S_8	S_1	S_8	1.000	1.000	1.000	1.000	1.000	**1.000**	
	5.22	S_8	S_4	S_1	S_4	0.964	1.000	1.000	0.965	0.984	**0.983**	
	5.23	S_8	S_6	S_1	S_6	0.974	0.991	0.973	0.979	0.974	**0.978**	
	5.24	S_8	S_6	S_4	S_6	1.000	1.000	1.000	1.000	1.000	**1.000**	

6.2.2 t-test Analysis for Dataset I Accuracy Results

As each sub-case is a sample from all possible combinatorial occurrences, a statistical significance test is performed to eliminate the chance of achieving the results due to the choice of the states. In this study, pairwise comparisons were conducted using the t-test to evaluate the presence of statistically significant differences between the cases, and the results are summarized in Table 11. Each test is performed for case X vs. case Y, where X and Y are cases from the analysis performed. The t-test was conducted with a significance threshold of 0.05, meaning any p-value below this threshold would indicate a statistically significant difference between the results of the cases being compared. The 'Result' column shows 1 when there is a significant difference present and 0 when there is no significant difference present. The first four tests are designed to evaluate the benefit of model adaptation using target state(s). The results show that case 1 displayed a significant difference when compared with case 2, case 3, case 4, and case 5 with p-values of 0.01103, 0.01165, 0.01250, and 0.01122, respectively. Since these p-values are below the threshold of 0.05, it can be said that case 1 (no domain adaptation) is significantly different from cases 2 and 3 (single-state domain adaptation) and cases 4 and 5 (two-state domain adaptation). These results validate that the achieved accuracy in the last columns of Table 11 is statistically higher for cases with domain adaptation. While a significant difference is visible when comparing the accuracy of case 1 with others, this difference is faded when comparing the performance of cases 2, 3, 4, and 5. Thus, to statistically validate the difference between the performance of other cases, the next three tests are

Table 11 t-test analysis on results for dataset I

Test No.	Case		p-value for X vs. Y	Threshold	Result	Overall Mean Accuracy for	
	X	Y				X	Y
1	Case 1	**Case 2**	0.01103	0.05	**1**	0.85	**0.986**
2	Case 1	Case 3	0.01165	0.05	1	0.85	0.979
3	Case 1	Case 4	0.0125	0.05	1	0.85	0.984
4	Case 1	Case 5	0.01122	0.05	1	0.85	0.987
5	**Case 2**	Case 3	0.59896	0.05	**0**	**0.986**	0.979
6	**Case 2**	Case 4	0.9842	0.05	**0**	**0.986**	0.984
7	**Case 2**	Case 5	0.97252	0.05	**0**	**0.986**	0.987

Table 12 Table F1 Score result for dataset I

Case	Subcase	f_{ae} Model trained	PCA	T1	T2	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Average F1 Score	Overall mean F1 Score
1	1.1	S_1	S_1	-	-	0.874	0.876	0.876	0.881	0.883	0.878	0.852
	1.2	S_4	S_4	-	-	0.778	0.773	0.785	0.778	0.773	0.777	
	1.3	S_6	S_6	-	-	0.889	0.887	0.887	0.887	0.885	0.887	
	1.4	S_8	S_8	-	-	0.866	0.867	0.864	0.864	0.865	0.865	
2	2.1	S_1	S_1	S_4	-	0.976	0.976	0.975	0.942	0.976	0.976	**0.960**
	2.2	S_1	S_1	S_6	-	0.976	0.976	0.972	0.961	0.967	0.976	
	2.3	S_1	S_1	S_8	-	0.975	0.976	0.982	0.936	0.912	0.975	
	2.4	S_4	S_4	S_1	-	0.941	0.976	0.976	0.976	0.975	0.941	
	2.5	S_4	S_4	S_6	-	0.963	0.976	0.972	0.854	0.956	0.963	
	2.6	S_4	S_4	S_8	-	0.925	0.916	0.923	0.895	0.915	0.925	
	2.7	S_6	S_6	S_1	-	0.976	0.963	0.976	0.978	0.976	0.976	
	2.8	S_6	S_6	S_4	-	0.976	0.976	0.976	0.966	0.972	0.976	
	2.9	S_6	S_6	S_8	-	0.961	0.934	0.873	0.976	0.964	0.961	
	2.1	S_8	S_8	S_1	-	0.972	0.965	0.947	0.958	0.972	0.972	
	2.11	S_8	S_8	S_4	-	0.974	0.955	0.975	0.965	0.974	0.974	
	2.12	S_8	S_8	S_6	-	0.975	0.973	0.976	0.976	0.976	0.975	

designed keeping case X = case 2, i.e., the proposed method. The p-values show that there is no significant difference between these cases. This highlights two important results. First, there is no significant benefit in performing domain adaptation using more than one target state. Second, there is no significant benefit in using different states for training the representation learning model and the state change detection model. Thus, the proposed method performs best from a practical perspective as it needs a minimal amount of data, i.e. data from only one state for training and one different state for adaptation. To further assess the effectiveness of the proposed method and validate the key observation from the t-test analysis, F1 score are used as evaluation metrics.

6.2.3 F1 Score Results for Dataset I

Following the same methodology as for obtaining Table 10. The F1 score for dataset I is shown in Table 12 for case 1 and case 2. In case 1 (no domain adaptation), the overall F1 score is approximately 85.2%. However, in the proposed method, where the observed state is used as the target state for domain adaptation, the F1 score increases to 96%. Based on these observations, from the t-test analysis and with the F1 Score, the other two datasets are evaluated for case 1 and case 2.

6.3 Results for Dataset II and Dataset III

Following the same methodology as for obtaining Table 10, Table 13 presents the performance of dataset II. In case 1 (no domain adaptation), the overall accuracy is approximately 89.9%. However, in the proposed method, where the observed state is used as the target state for domain adaptation, the accuracy increases to 97.4%.

Similarly, Table 14 presents the model's performance for dataset III. For dataset III as well, the proposed method, case 2, shows an improvement in performance compared to case 1. The proposed method has an overall accuracy of 98.4% compared to 96.6% for case 1.

Table 13 Achieved average accuracy for cases in Dataset II

Case	Subcase	f_{ae} Trained	PCA Trained	Target state S_{c_1}	Target state S_{c_2}	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Average Accuracy	Overall mean Accuracy
1	1.1	S_1	S_1	-	-	1.000	1.000	1.000	1.000	1.000	**1.000**	0.899
	1.2	S_2	S_2	-	-	0.972	0.972	0.971	0.972	0.972	**0.972**	
	1.3	S_3	S_3	-	-	0.728	0.728	0.727	0.727	0.727	**0.728**	
2	2.1	S_1	S_1	S_2	-	1.000	0.999	0.999	1.000	0.998	**0.999**	**0.974**
	2.2	S_1	S_1	S_3	-	0.946	0.926	0.951	0.885	0.941	**0.930**	
	2.3	S_2	S_2	S_1	-	0.975	0.967	0.974	0.966	0.978	**0.972**	
	2.4	S_2	S_2	S_3	-	0.988	0.875	0.98	0.965	0.963	**0.954**	
	2.5	S_3	S_3	S_1	-	0.998	0.992	1.000	1.000	1.000	**0.998**	
	2.6	S_3	S_3	S_2	-	1.000	1.000	1.000	1.000	1.000	**1.000**	

Table 14 Achieved average accuracy for cases in Dataset III

Case	Subcase	f_{ae} Trained	PCA Trained	Target state S_{c_1}	Target state S_{c_2}	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Average Accuracy	Overall mean accuracy
1	1.1	S_1	S_1	-	-	0.932	0.929	0.928	0.928	0.93	**0.929**	0.966
	1.2	S_2	S_2	-	-	0.968	0.968	0.968	0.968	0.969	**0.968**	
	1.3	S_3	S_3	-	-	1.000	1.000	1.000	1.000	1.000	**1.000**	
2	2.1	S_1	S_1	S_2	-	0.99	1.000	0.934	1.000	0.948	**0.974**	**0.986**
	2.2	S_1	S_1	S_3	-	1.000	1.000	1.000	1.000	1.000	**1.000**	
	2.3	S_2	S_2	S_1	-	1.000	1.000	1.000	1.000	1.000	**1.000**	
	2.4	S_2	S_2	S_3	-	0.939	0.954	0.842	0.948	0.971	**0.931**	
	2.5	S_3	S_3	S_1	-	1.000	1.000	1.000	1.000	1.000	**1.000**	
	2.6	S_3	S_3	S_2	-	0.997	1.000	1.000	1.000	1.000	**0.999**	

6.4 Analysis of Model Training Performance

Model performance is validated using three real-life datasets, and variations in model training and its performance are illustrated in Figure 15, Figure 17 and Figure 16 for Dataset I, Dataset II, and Dataset III, respectively. For Case 1 with no domain adaptation, in each figure the loss function is represented by: $T_L = R_L = \sum_{j}^{m_a} \left|\left| s_{aj} - \hat{s}_{aj} \right|\right|_2^2$. The early convergence of the loss function suggests that, without domain adaptation, the model learns only from the training data and struggles to generalize. In contrast, for Cases 2 and 3, which illustrate single target state domain adaptation, the loss function is modified and represented by $T_L = (1-\alpha)R_L - \alpha(M_L)$. These plots demonstrate a more gradual decline in loss over more epochs, indicating enhanced learning. For Cases 4 and 5,

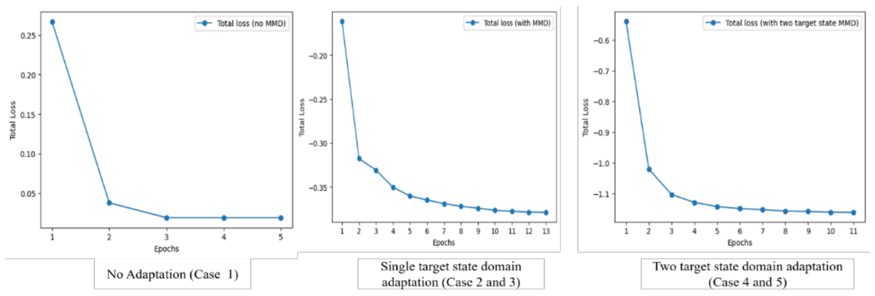


Figure 15 Comparison of model convergence with and without custom loss function (domain adaptation) for Dataset I

representing two target state domain adaptation, the loss function is represented by: $T_L = (1-\alpha)R_L - \alpha(M_L + M_{L_1} + M_{L_2})$, follows a similar trend, showing no significant difference in the rate of loss reduction when compared to single state adaptation. This suggests that while domain adaptation enhances model performance relative to no adaptation, increasing the number of target states does not provide a substantial additional benefit in reducing the overall loss. This observation of not having significant differences is further highlighted in t-test analysis in the manuscript. These observations are common to all three datasets. It is also important to note that while the model takes longer to converge, the increment in the number of epochs are still smaller, allowing for quicker training times. The effectiveness of integrating the MMD loss is validated using accuracy as a metric.

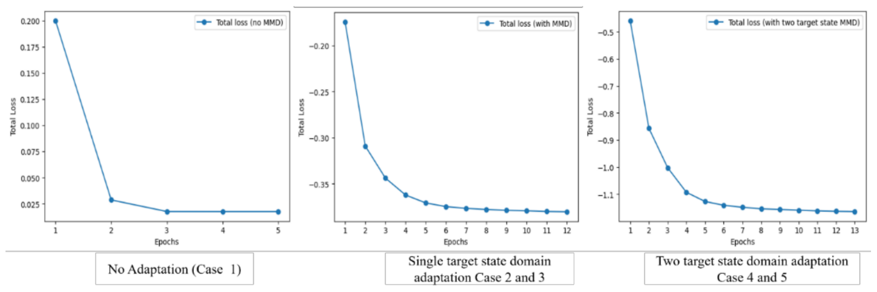


Figure 17 Comparison of model convergence with and without custom loss function (domain adaptation) for Dataset II

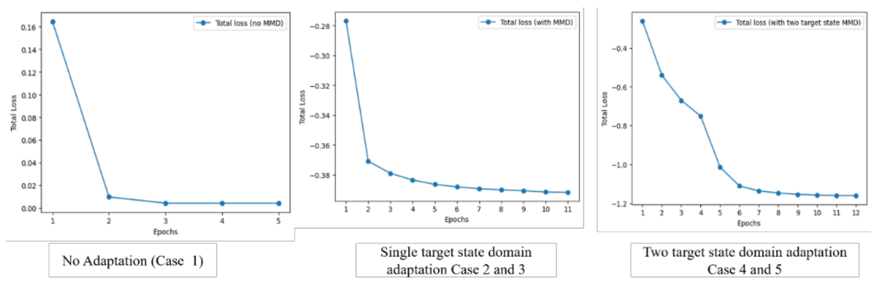


Figure 16 Comparison of model convergence with and without custom loss function (domain adaptation) for Dataset III

6.5 Impact of Data Quantity on Model Training Performance

The results shown so far use data of time duration over 10 seconds (120000 values) for model training. While this time interval of 10 seconds can be considered small, further experiments are conducted to evaluate the performance of the proposed method under smaller time intervals, i.e. lesser data quantity. The following Table 15 compares the performance of the proposed method as described in Table 10 of the manuscript across data quantities from 12000 points (10 secs) to 48000 points (4 secs). The performance is compared using metrics of accuracy and F1 score. It could be clearly seen that the proposed model performs significantly better across all data quantities as compared to using the method without adaptation. While there is an expected reduction in the performance with a reduction in the data quantity within the proposed method, the accuracy at the lowest data quantity of 48000 points (4 secs) is more than 95%. This shows that even with less data for model training, the efficiency of state detection is maintained.

The following observations could be made by comparing the performance across different datasets. Each dataset is different in terms of the equipment, sensor, sensor location, fault type, operating conditions, data sampling rate, etc. This represents a real-life scenario for developing digital twins using a bottom-up, scalable approach. The results highlight the variation in

Table 15 The impact of data quantity used for model training on performance of the proposed approach

Data Collection Time Interval	Data Quantity for Training	Accuracy		F1 Score	
		Case 1 (No MMD)	**Case 2 (Proposed)**	Case 1 (No MMD)	**Case 2 (Proposed)**
10 sec	120000	0.850	**0.986**	0.865	**0.962**
8 sec	84000	0.874	**0.969**	0.859	**0.954**
5 sec	60000	0.818	**0.952**	0.819	**0.937**
4 sec	48000	0.808	**0.951**	0.827	**0.934**

performance due to these differences. For example, comparing case 1 across datasets I, II, and III, it can be observed that dataset II has the highest accuracy of 96.6% compared to 89.9% and 85% for III and I, respectively. However, using the proposed method (case 2), the achieved performance is more than 97.5% irrespective of the above factors as well as the state used for model training in the digital twin. Further, the proposed method allows the creation of stimulus-aware digital twins using only two states at the beginning of equipment operation.

Chapter 7 Benchmarking the Autoencoder Model

In this chapter, we present a comprehensive comparison between the autoencoder model and the Wasserstein Generative Adversarial Network (WGAN). The chapter begins with a detailed literature review on GANs, followed by an introduction to their underlying principles and implementation techniques, and the challenges with traditional GANs. We then describe how WGANs were applied to address the objectives of our study. Currently, we have implemented WGANs without domain adaptation, as integrating domain adaptation into WGANs is a relatively new development that requires significant research and investigation. This will be considered for future scope.

Thus, a comparative analysis of the results obtained from the WGANs without domain adaptation and the autoencoder model without domain adaptation is provided to highlight the strengths and limitations of WGANs

7.1 Introduction to GANs

Data-driven approaches have gained significant popularity in the field of equipment behavior monitoring due to their ability to utilize real-time data to predict and simulate system behavior. These methods, often based on machine learning algorithms, are particularly valuable because they can model complex systems without the need for explicit mathematical formulations. This is especially important in industries such as manufacturing, energy, and aerospace, where the dynamics of equipment are often nonlinear, highly interconnected, and difficult to describe using traditional physics-based models. The increasing complexity and volume of industrial data presents major challenges for conventional modeling techniques. In contrast, data-driven methods can effectively handle heterogeneous and high-dimensional datasets, enabling accurate and timely detection of equipment states. Furthermore, these approaches are adaptable and capable of learning from new data, which supports real-time monitoring, early fault detection, and predictive maintenance. As a result, data-driven techniques are becoming essential tools for improving reliability, efficiency, and decision-making in modern industrial systems.

In recent years, the application of machine learning techniques one-class support vector machines (SVM) [65] and support vector data description (SVDD) [66], has been widely used for health and condition monitoring. Xie et al. [67] used a data-driven approach and discussed applications of digital twins at different stages of a cutting tool's life cycle. Peng et al. [68]proposed a digital twin model of the life cycle of the bearing, where the health states of the bearing are monitored. Luo et al. [61] have developed the digital twin for the CNC machine tool, where a multi-domain unified modeling technique of DT is established to map the physical and digital space for monitoring the health states and diagnosing faults. Luo et al. [69] also presented a DT for the CNC machine tool, which allows for precise prediction and monitoring of the machining state of a CNC machine tool, as well as real-time compensation of the workpiece's machining contour inaccuracy. Support vector machines (SVM) and Mult-kernel support vector machines are widely employed in detecting the bearing classification and different faulty states classifications[70], [71]. Many other machine-learning techniques are widely used for the development of digital twins.

[75]While the application of data-driven modeling is vast, the approaches are generally developed for specific equipment and for specific purposes, and they lack generalizability across different equipment and across different sensors. Production systems possess the complexity of changing machine states with time due to varied operational patterns, environmental conditions, as well as varied system configurations. To address generalization across such patterns, deep learning has seen significant success in detecting health states. [76], [77], [78] have summarized and discussed various deep learning techniques that can be used to detect the health states of the various rotating equipment such as gearboxes, bearings, and pumps.

In recent research, Advanced machine learning techniques, such as Traditional Generative Adversarial Networks (GANs), Long Short-Term Memory-Fuzzy Synthesis (LSTM-FS), GAN-based Autoencoders (GAN-AE), and traditional Autoencoders, are widely used for learning and modeling complex data behaviours. These methods excel at automatically extracting meaningful features, identifying intricate patterns, and capturing underlying data distributions.

Although the concept of the GAN is exciting and promises many novel applications, it is very difficult to train a GAN model. In some cases, the loss of the generator and discriminator may exhibit an oscillating pattern, rather than converging to some value over the course of the training. Due to this oscillating loss problem, the model parameters do not stabilize, and the model does not provide good output. Mode collapse is another common challenge in GAN training, which occurs when the generator settles itself into generating a small number of similar outputs. In this case, the generator might make the discriminator believe the output is real output, but the generator is not able to generate a real-data distribution, which is more diverse and complex, rather than the limited set it generates when mode collapse occurs. Additionally, Vanishing gradient is a phenomenon in GAN training where the generator receives gradients that are too small to effectively update its weights. This often occurs when the discriminator becomes highly accurate at distinguishing between real and generated data early in the training process. As the discriminator’s output becomes highly confident, the gradients passed back to the generator diminish, resulting in minimal updates and a stagnation or divergence of the generator's loss, commonly referred to as generator saturation.

To provide stable GAN training, several approaches have been proposed over the past few years. WGAN (WGAN) [109], [110], which proved efficient in many complex GAN applications. WGANs are especially useful for anomaly detection, predictive modeling, and state recognition across various domains. They have broad applicability in real-world scenarios, particularly in complex production systems where data is collected in diverse forms and from multiple sources. In many cases, especially in manufacturing environments, there is a significant imbalance in the availability of data. Large volumes of data can be collected during normal or healthy operating conditions, while data from rare or faulty states is often limited or unavailable. WGANs address this challenge by generating realistic synthetic data, making them particularly valuable for augmenting datasets where rare conditions are underrepresented. As a result, WGANs have become one of the most widely used algorithms for processing both numerical and visual data in applications such as system monitoring, predictive maintenance, and digital twin development within industrial production systems.

In this work, we describe how WGANs were applied to address the specific objectives of our study. Finally, a comparative analysis of the results obtained from the WGANs and the autoencoder model is provided to highlight the strengths and limitations of WGANs. WGANs hold the potential to tackle the challenges of data scarcity and unsupervised learning, making them a promising candidate for the generalized and autonomous modeling of equipment behavior. Generative Adversarial Networks (WGANs) were first introduced as a novel framework for estimating generative models through an adversarial training process. The WGANs are unsupervised generative models that automatically locate the data distribution. In this framework, two models are trained simultaneously: a generator G is a type of neural network, a convolutional network, that aims to capture the underlying data distribution and is responsible for generating the data, while another type of network, a deconvolutional network, is the discriminator D That attempts to differentiate between real data samples and those produced by G. The generator's task is to create data that closely resembles the actual dataset, while the discriminator assesses the authenticity of the samples, predicting whether a given instance comes from the true data distribution or the generator.

In the traditional GANs that use the Jensen-Shannon (JS) divergence to measure the difference between the real and generated data distributions, WGAN employs the Earth-Mover (Wasserstein-1) distance, which provides a smoother and more meaningful loss metric even when the supports of the distributions do not overlap.

The generator $G$is responsible for producing synthetic data samples by transforming random noise vectors z, drawn from a prior distribution like (uniform or Gaussian) $P_z(Z)$, into data space samples that ideally resemble those from the real data space as $G(z;\theta_g)$. Instead of a traditional discriminator, the framework includes the critic function, f_w, parameterized by w, which assigns real-valued scores rather than probabilities. This critic estimated the Wasserstein distance between the real data and generated distributions.

The G and f_w play the following two-player minimax game with value function $W\left(P_g, \mathbb{P}_g\right) = sup_{||f||_{l\leq 1}}\ \mathbb{E}_{s_{aj}\sim\mathbb{P}_r}[\mathrm{f}\left((\mathrm{z})\right) - \mathbb{E}_{r\sim p(z)}[\mathrm{f}\left(\mathrm{G}(\mathrm{z})\right)]$

$$||f||_L \leq 1$$

Where f is a 1-Lipschitz function.

The generator loss is $-\mathbb{E}_{r\sim p(z)}[\mathrm{f}\left(\mathrm{G}(\mathrm{z})\right)]$. Where P_g is real data distribution the data from the actual training samples. While $\mathbb{P}_g$ represents the distribution of generated samples $\mathrm{G}(\mathrm{z})$.

7.2 Implementation of WGANs

In a complex production system, changes in state can occur due to various factors such as operating parameters, environmental influences, and equipment health. This study aims to establish a stimulus-aware capability for the digital twin of the equipment, allowing it to begin learning from the moment it is put into use. This study aims to establish a stimulus-aware capability for the digital twin of the equipment, allowing it to begin learning from the moment it is put into use. The framework consists of two phases:

a training phase followed by a detection phase. Since our proposed method focuses on developing the stimulus-aware digital twin right from the start of the equipment's operation, two key factors must be considered during validation: 1) minimal data availability for training, and 2) the potential occurrence of any operating state during training. The Schematic training Pipeline is shown in Figure 18, which is a combination of two phases. The two phases of this framework are as follows:

7.2.1 Training Phase

The framework is an unsupervised 3D convolutional WGANs used for the training phase for pattern learning. The WGANs are unsupervised feed-forward neural networks that contain an input layer, multiple hidden layers, and an output layer. The framework consists of two modules: a generator and a discriminator. WGANs consist of two modules: a generator and a discriminator. In this study, the encoder, and decoder, are represented by G and D respectively. The state used for the training of the model is represented by S_a and the state used for testing the model is represented by S_b.The generator $G$is responsible for producing synthetic data samples by transforming random noise vectors z, drawn from a prior distribution like (uniform or Gaussian) $P_z(Z)$, into data space samples that ideally resemble

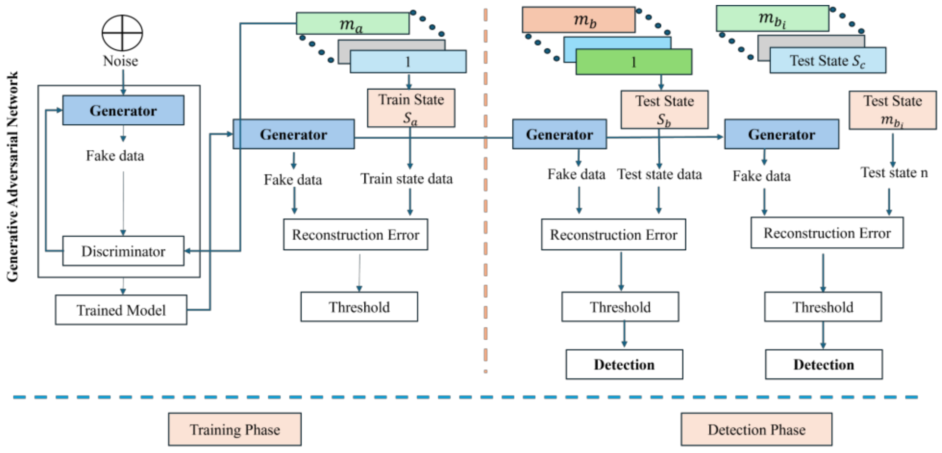


Figure 18 Schematic Training Pipeline For WGANs

those from the real data space as $G(z; \theta_g)$. The G and f_w play the following two-player minimax game with value function

$$W(S_a, \mathbb{P}_g) = sup_{||f||_{l \leq 1}} \mathbb{E}_{s_{aj} \sim \mathbb{P}_r}[\mathrm{f}\big((\mathrm{z})\big) - \mathbb{E}_{r \sim p(z)}[\mathrm{f}\big(\mathrm{G(z)}\big)]$$

$$||f||_L \leq 1$$

Where f is a 1-Lipschitz function. The generator loss is $-\mathbb{E}_{r \sim p(z)}[\mathrm{f}\big(\mathrm{G(z)}\big)]$. Where S_a is real data distribution the data from the actual training samples. While $\mathbb{P}_g$ represents the distribution of generated samples $\mathrm{G(z)}$. Once training is complete, the generator learns to produce samples that mimic real data well enough to minimize the adversarial loss. To validate the performance of the trained model, we assess how well it generates synthetic data that closely replicates the behavior of the real data. This assessment is conducted using Fast Fourier Transform (FFT), as well as analyses in both the time and frequency domains. Based on these promising results, we moved forward with deploying the trained algorithm for state detection across multiple states using a reconstruction-error-based approach.

7.2.2 Detection Phase

To carry out the state detection, we calculate the reconstruction error for each input sample by measuring the mean squared error between the original state input data and the synthetic data. The loss can be represented by:

$$R_E = \sum_j^{m_a} \big|\big|s_{aj} - \mathrm{G(z)}\big|\big|_2^2.$$

After calculating the reconstruction error using the training data, a threshold is defined by calculating the mean of its reconstruction errors and then adding three times the standard deviation. This statistical formula is based on the three-sigma rule, which assumes that for a normally distributed variable, nearly all (about 99.7%) of the data points fall within three standard deviations from the mean. These training data points will mostly lie within this range, and any data point with a reconstruction error exceeding this threshold is considered a separate state relative to that particular state. For detection, this threshold is then tested against the reconstruction errors of all other states. For each test state, the number of data points with errors above the threshold is counted and divided by the total number of points in that state. This results in a detection accuracy score, which reflects how different the test state is compared to the training (threshold-setting) state. A high accuracy implies that many points in the test state exceeded the threshold and are thus detected as separate state samples. Conversely, a low accuracy suggests that the test state is similar to the training state.

7.3 Results and Discussion

In a complex production system, such as the production systems described in Figure 3, a change in state can arise from various factors, including operating parameters, environmental factors, equipment health, etc. To assess the performance of the autoencoder model, we created several cases, as outlined in Table 8. Furthermore, to benchmark the autoencoder model's performance without domain adaptation (case 1) is compared with a Generative Adversarial Network (WGAN).

Table 16 WGANs Results for Dataset I

					Test State									
Methodology	Case	Subcase	Model Trained	Threshold State	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	Average accuracy
GANs	1	1.1	S_1	S_1	-	0.01	0.01	0.99	1.00	1.00	1.00	1.00	1.00	0.751
		1.2	S_4	S_4	0.03	0.02	-	0.99	0.01	0.49	1.00	1.00	1.00	0.567
		1.3	S_6	S_6	0.01	0.01	0.32	0.00	0.02	-	0.79	0.72	0.97	0.355
		1.4	S_8	S_8	0.02	0.01	0.01	0.01	0.00	0.32	0.01	-	0.00	0.047
													Mean	**0.430**
													St. Dev.	**0.302**

A similar validation methodology is applied to the WGANs methodology. However, only the first case was considered, as domain adaptation techniques are not employed in the WGANs approach. Furthermore, unlike the autoencoder model, which utilizes a PCA-based method for threshold setting and state change detection, the WGANs model relies on the three-sigma statistical method for state change detection. Each of the states is represented by 181 samples, with each sample having 12,000 data points. For this case, out of the nine possible states for training only four states (S_1, S_4, S_6, S_8) are sampled for performance evaluation, forming four subcases. Both the WGANs training and the threshold for detection are determined using the same state. The third column of Table 16 lists these subcases under case 1, while the fourth and fifth columns represent the state used for modeling. The sixth column of Table 16 represents the model's accuracy in detecting the specified test state. The last column of Table 16 shows the average accuracy for each subcase across all test states. It is important to note that while only one state is used for training, all other states are employed for testing. The trained GANs generator module produces synthetic data, which is then compared with the actual test state data to compute the reconstruction error. The reconstruction error from the training state is subsequently passed through the three-sigma method to determine the detection threshold.

We have compared the learning of the autoencoder model with that of the WGANs model by evaluating how well each learns the behavior of the data using four parameters.

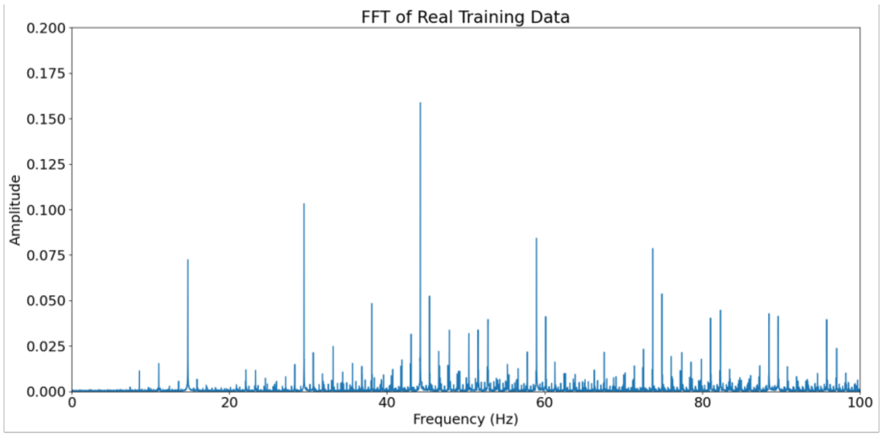
- Fast Fourier Transform: The FFT (Fast Fourier Transform) of real training data is compared with the FFT of generated data (using WGANs) and the FFT of reconstructed data (using the autoencoder model).
- Time domain analysis: Time domain analysis of real training data is compared with time domain analysis of generated data (using

WGANs) and time domain analysis of reconstructed data (using the autoencoder model).

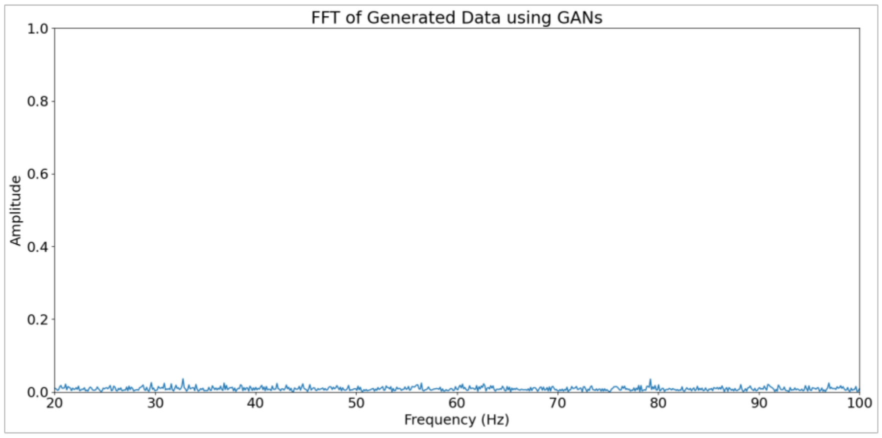
- Based on the Representation: Reconstruction error (using WGANs) is compared with the encoding representation (using the autoencoder Model).
- Accuracy: Additionally, the performance of the WGAN model is also compared with the autoencoder model.

For example, in subcase 1.1, the WGANs model is trained on state S_1. All states are then passed through the generator module of the trained WGANs model to generate synthetic data. The reconstruction error is calculated for each state as the mean squared error (MSE) between the real state data and the corresponding data generated by the model. The threshold is determined based on the reconstruction errors of the training state, using the 3-sigma method.

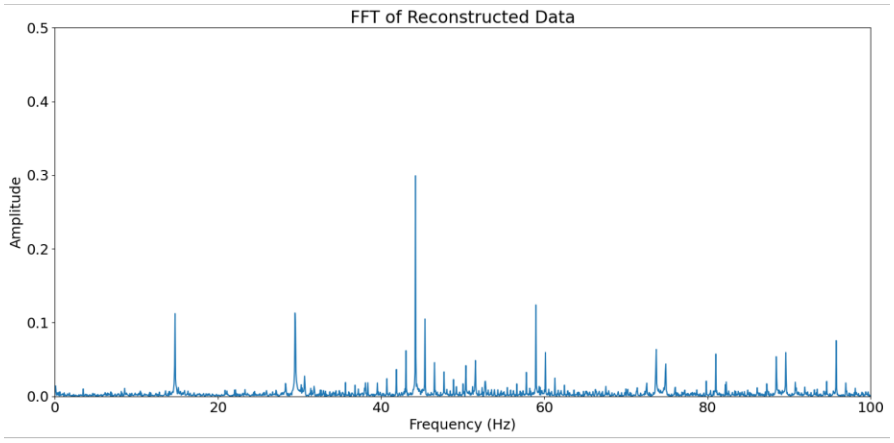
For example, in subcase 1.1, the pattern learning capability of both models is analyzed by comparing the Fast Fourier Transform (FFT) plots, time-domain signals, and the visualizations of reconstruction error and encoded representations, as illustrated in Figure 19. The goal is to evaluate how effectively the model learns and replicates the statistical and temporal characteristics of the real training vibration signals. The FFT plot (a) is of real training data shown in Figure 19, revealing distinct amplitude peaks at certain frequencies, indicating the presence of dominant periodic components that are characteristic of S1 vibration signals. The FFT of the generated data using GANs is shown in plot (b), which represents a nearly flat spectrum with very low amplitude and no prominent frequency components, suggesting that the model fails to capture the underlying frequency characteristics of the real data. While the plot (c) shows the FFT plot of reconstructed data constructed by the autoencoder model, closely mirroring the amplitude peaks of the real training data. This spectral similarity indicates that the autoencoder has effectively learned the frequency distribution of the real signal, which strongly suggests that the model can produce realistic reconstructed vibration data peaks, though the amplitudes may be slightly lower. This spectral similarity suggests that the



(a) Real Training Data



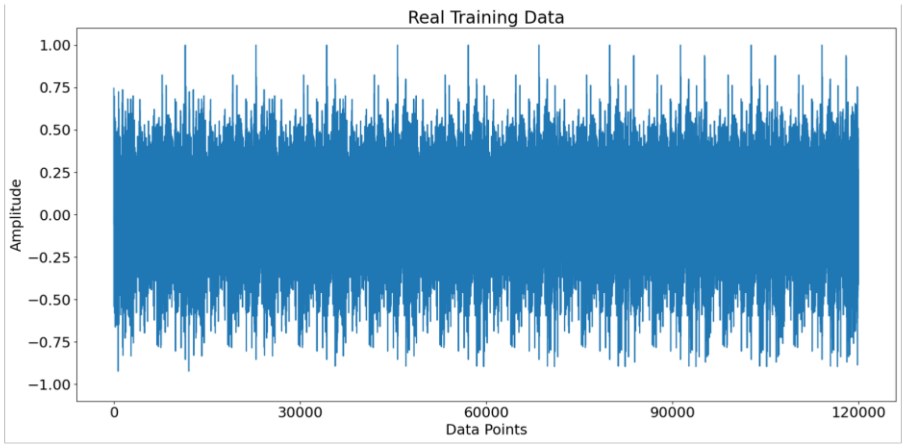
(b) Generated Data



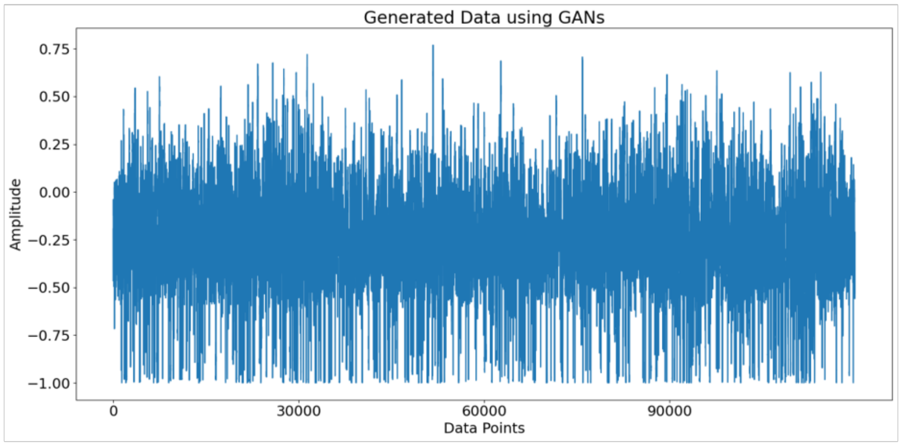
(c) Reconstructed Data

Figure 19 FFT Trend comparison between (a) Real Training Data, (b) Generated data using the WGANs, (c) Reconstructed data using the Autoencoder Model

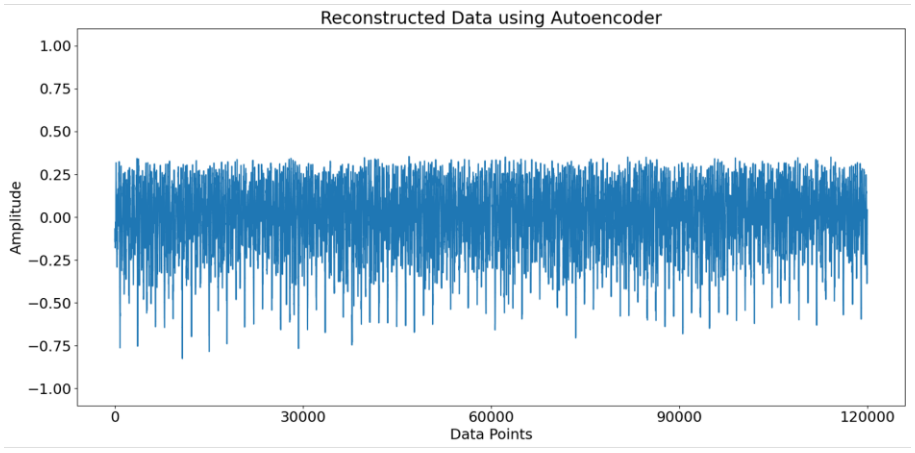
autoencoder has successfully learned the frequency distribution of the real



(a) Real Training Data



(b) Generated Data



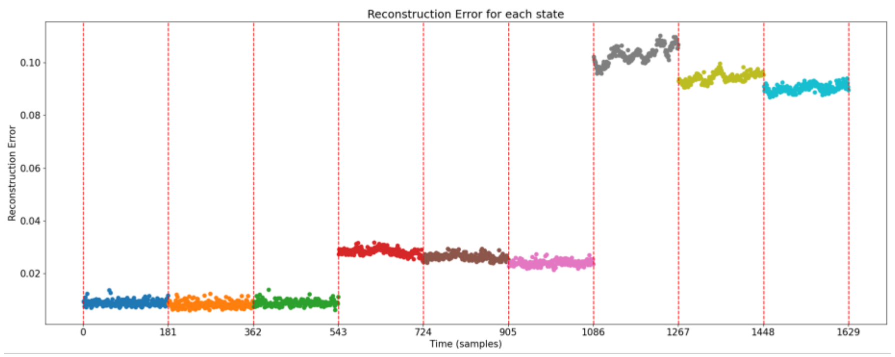
(c) Reconstructed Data

Figure 20 Time Domain Analysis Trend between the (a) Real Training Data, (b) Generated data using WGANs and (c) Reconstructed data using Autoencoder Model

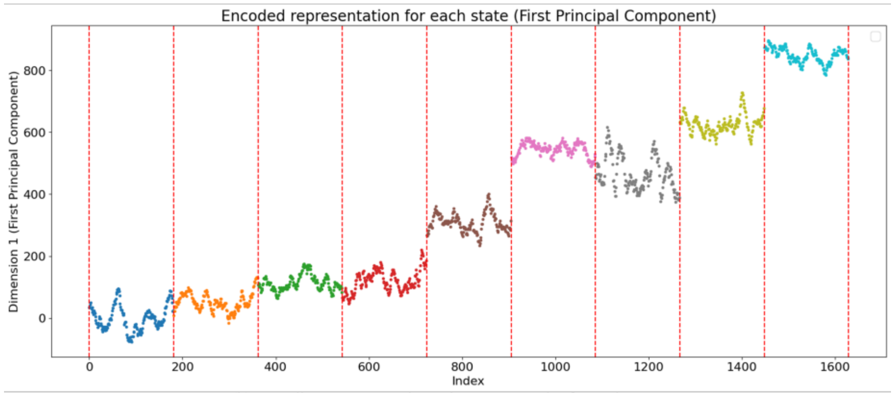
signal, which is a strong indicator of the model's ability to produce realistic reconstructed vibration data. Figure 20 shows the time domain analysis, where plot (a) represents the real training data, showing a signal centered around zero with consistent amplitude and typical noise patterns. The plot (b) shows the time domain analysis of data that is generated using the WGANs model, which struggles with consistent variability across the sequence, which potentially indicates the problem of model collapse and training instability.

While the autoencoder model is capable of constructing data with similar patterns, indicating that the autoencoder has effectively learned the complex patterns from the intricate manufacturing data shown with plot (c). The reconstructed data using the autoencoder appears to be more aligned with the real training data compared to the data generated using WGANs. While both the WGAN-generated and autoencoder-reconstructed data attempt to replicate the characteristics of the original signal, the reconstructed data maintains a more consistent amplitude distribution and structural pattern throughout the time series. In contrast, the WGAN-generated data shows noticeable irregularities, particularly a reduction in amplitude variability and dynamic range in the latter half of the series, which suggests instability in the generation process.

Following the training of WGANs, the reconstruction error is plotted is shown with plot (a) of Figure 21. The reconstruction error for each state sample reveals how different samples deviate in terms of reconstruction quality. We have observed that some state samples exhibit significantly higher reconstruction errors, indicating they are likely different states. These deviations validate the model's ability to detect a shift in data distribution, effectively identifying different operational states. However, some sample groups, despite belonging to different states, show overlapping or similar reconstruction error levels, which implies that the model struggles to distinguish between certain states. This overlap creates a challenge in clearly classifying all states.



(a) Reconstruction errors using Wasserstein Generative Adversarial Networks (WGANs) for each state.



(b) Encoding Representation using Autoencoder for each state.

Figure 21 (a) Reconstruction error calculated using WGANs vs Encodings Generated using Autoencoder

However, in the autoencoder model, the latent representations are generated using the encoder module. These latent representations are passed to the PCA model trained on the same state, S_1. The PCA transforms the latent representation of each state into a lower dimension. The top principal component for each state for the given samples is shown with plot (b) of Figure 21. From the perspective of state change detection, the figure illustrates that some encodings exhibit separation, indicating that the model has effectively learned the features necessary to differentiate between the states. However, some state encodings are overlapping, suggesting that the model may face difficulties in differentiating these states.

The accuracy for Dataset I is presented in Table 16. The first column of the table represents the methodology used. The third column of lists these subcases under case 1, while the fourth and fifth columns represent the state used for modeling. The sixth column of Table 16 represents the model's accuracy in detecting the specified test state. The last column of Table 16 shows the average accuracy for each subcase across all test states. For instance, in case 1.2, the achieved accuracy for detecting S_6 in subcase 1.2 is 0.49, which means the model in subcase 1.2 can detect state S_6 with a 49 % accuracy. Thus, for case 1.2, the average accuracy with which the model can detect all states is 56.7%, as given in the last column. presents the overall result for dataset I calculated using the WGANs and Autoencoder Model, which comprises four subcases, and where each sub-case is repeated five times to calculate the mean accuracy. This repetition helps evaluate the model's stability and consistency in detecting states. The second last column of the table displays the average accuracy of five repeated evaluations for each subcase. The table's last column shows the model's overall mean accuracy for each case. The overall mean accuracy calculated using the WGANs for case 1 in dataset I, is 43.9%. Table 17 presents the overall result for dataset I calculated using the WGANs and Autoencoder Model, which comprises four subcases, and where each sub-case is repeated five times to calculate the mean accuracy, as explained in Table 16. The second last column of the table displays the average accuracy of five repeated evaluations for each subcase. The table's last column shows the model's

Table 17 Accuracy Analysis for Dataset I using WGANs and Autoencoders

Methodology	Case	Subcase	Model Trained	Threshold State	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Average Accuracy	Overall mean accuracy
GANs	1	1.1	S_1	S_1	0.741	0.721	0.761	0.731	0.737	0.738	0.439
		1.2	S_4	S_4	0.567	0.51	0.512	0.551	0.521	0.532	
		1.3	S_6	S_6	0.351	0.312	0.281	0.381	0.321	0.329	
		1.4	S_8	S_8	0.042	0.52	0.12	0.051	0.047	0.156	
Autoencoder Model	1	1.1	S_1	S_1	0.874	0.876	0.876	0.881	0.883	**0.874**	**0.85**
		1.2	S_4	S_4	0.778	0.773	0.785	0.778	0.773	**0.773**	
		1.3	S_6	S_6	0.889	0.887	0.887	0.887	0.885	**0.887**	
		1.4	S_8	S_8	0.866	0.867	0.864	0.864	0.865	**0.867**	

overall mean accuracy for each case. The overall mean accuracy calculated using the GANs for case 1 in dataset I, is 43.9%.While the autoencoder model resulted in a significant improvement in the model's performance compared to the WGANs method. The overall mean accuracy of the autoencoder model is around 85 % shown in Table 17 .

7.4 Results Analysis for Dataset II and Dataset III

Using the same methodology as for obtaining results in Table 17. Table 18 presents the overall accuracy comparison between the WGANs and the Autoencoder model for Dataset II. The overall accuracy using WGANs is

Table 18 Accuracy Analysis of WGANs and Autoencoders on Dataset II

Methodology	Case	Subcase	Model Trained	Threshold State	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Average Accuracy	Overall mean accuracy
GANs	1	1.1	S_1	S_1	0.574	0.53	0.512	0.536	0.486	**0.528**	0.595
		1.2	S_2	S_2	0.669	0.61	0.612	0.684	0.623	**0.639**	
		1.3	S_3	S_3	0.623	0.61	0.612	0.591	0.652	**0.618**	
Autoencoder Model	1	1.1	S_1	S_1	1	1	1	1	1	**1**	0.899
		1.2	S_2	S_2	0.972	0.972	0.971	0.972	0.972	**0.972**	
		1.3	S_3	S_3	0.728	0.728	0.727	0.727	0.727	**0.728**	

Table 19 Accuracy Analysis of WGANs and Autoencoders on Dataset III

Methodology	Case	Subcase	Model Trained	Threshold State	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Average Accuracy	Overall mean accuracy
GANs	1	1.1	S_1	S_1	0.374	0.31	0.374	0.362	0.352	0.355	0.462
		1.2	S_2	S_2	0.452	0.41	0.469	0.475	0.432	0.448	
		1.3	S_3	S_3	0.581	0.61	0.621	0.578	0.526	0.584	
Autoencoder Model	1	1.1	S_1	S_1	0.932	0.929	0.928	0.928	0.93	0.929	**0.966**
		1.2	S_2	S_2	0.968	0.968	0.968	0.968	0.969	0.968	
		1.3	S_3	S_3	1	1	1	1	1	1	

approximately 59.5%. However, with the autoencoder model, the accuracy is increased to 89.9%. The model's performance for dataset III is presented similarly in Table 19. For this dataset, the autoencoder method demonstrates a significant improvement compared to the WGANs model, achieving an overall accuracy of 96.6% compared to 46.2% with WGANs.

7.5 Challenges Faced by the WGANs Models

1. Manufacturing data often includes multivariate time-series signals, high-frequency sensor readings, and various interdependent variables. The GAN model struggles to capture the intricate temporal and spatial relationships within this data. As reflected in the FFT analysis, the generator fails to replicate the spectral properties of the real data, indicating its inability to learn complex patterns effectively.
2. Models trained on one dataset do not generalize well to others, even when using unsupervised learning. This suggests that the GAN is overfitting to specific data distributions and lacks the robustness required to handle variability between different machines, production lines, or operational modes.
3. GANs inherently require large volumes of diverse data to generate realistic outputs. However, in real-world manufacturing environments, data scarcity is a common issue especially for rare events such as faults or anomalies. The model's performance significantly degrades with limited data, making it unsuitable for domains where comprehensive data collection is impractical.
4. Even with the integration of the Wasserstein loss function, which is designed to mitigate training instability and mode collapse, these issues continue to occur. The generator frequently produces repetitive or uniform outputs, indicating it has collapsed to a narrow subset of the data distribution.

Chapter 8 Exemplary Use Case of Proposed Work in Real Life

Digital twin technology is rapidly transforming production economics by enabling real-time decision-making. It plays a crucial role in various aspects of production management, including production planning and control[111], fault detection and diagnosis[112], condition monitoring[61], predictive maintenance, and asset management[113]. The first and most crucial step in decision-making within this process is ensuring that the digital twin is stimulus-aware, meaning it must have an understanding of the state in which it exists. Without accurate real-time awareness of its environment, it becomes very difficult to effectively support decision-making for production planning, scheduling, condition monitoring, predictive maintenance, or asset management. This foundational capability enables the digital twin to respond dynamically to changes, optimize processes, and drive intelligent automation. The developed algorithm enables real-time state detection in a digital twin, starting from the very beginning of production (t = 0). The methodology consists of four phases: the training phase, the testing phase, the adaptation phase, and the deployment phase. The adaptation phase is particularly important, as it involves crucial decisions made by the manager. During this phase, the manager utilizes the detected state information for domain adaptation. After adaptation, it is deployed for the upcoming state detection. Figure 22 shows the state detection of a digital twin from t = 0 to its operational. The y-axis shows the different states that the digital twin has changed, while the x-axis represents time. For instance, let's consider a machine that is producing part Q_1, in various batches that include three states: operation 1, operation 2, and operation 3, shown in Figure 22. By utilizing real-time state detection, this methodology enables the manager to accurately determine the exact production time for product Q_1. Additionally, the proposed method tracks state changes during production by capturing when each state transition occurs and how long the process remains in each state. This methodology provides an overall view of the equipment's behavior from the start (t = 0) until it reaches its operating condition, which is crucial for analyzing machine performance and degradation over time. When a machine starts to slow down, managers can take immediate action to address the issue. The novelty of this methodology lies in its scalability; the same approach can be applied to different machines and across various sensors, regardless of their location. Additionally, the developed methodology for creating a stimulus-aware digital twin includes an updating capability that can be improved over time. When the initial model fails to detect a state or demonstrates low accuracy in state detection, a human intervenes to label the model and clarify the specific state. This process allows the model to gain knowledge about the state, enabling it to detect the same state accurately in the future. In modern manufacturing, even slight variations in machine performance can indicate potential issues. Identifying these changes early enables managers to make informed decisions, avoid costly breakdowns, enhance efficiency, and improve production planning. In the scenario shown in Figure 22 where, a piece of equipment typically remains in a specific state that is operation 3 for 20 seconds during batch 1 production. In the next batch, this duration increases to 23 seconds, an increase with time that might seem insignificant. However, if the duration continues to rise to 25 seconds and beyond, and the efficiency and quality of production also get degraded, it becomes a clear sign of inefficiency or potential wear. At this point, managers can take real-time action by either pausing the process for further investigation or sending the machine for maintenance before a more severe failure occurs. Once maintenance is performed, the root cause of the increased duration can be identified, along with the number of parts produced before the issue arose and the amount of increase in the cycle time.

By incorporating this data into the digital twin model, future deviations can be more accurately analysed. Instead of automatically sending the machine for maintenance, managers can address the exact issue directly, preventing unnecessary downtime and improving production economics. This proactive approach enhances production planning by allowing managers to make smarter, data-driven, real-time decisions, ultimately reducing operational costs and improving delivery timelines. With real-time state detection, managers can ensure a more efficient, cost-effective, and robust production process. This proactive approach enhances production planning by allowing managers to make smarter, data-driven, real-time decisions, ultimately reducing operational costs and improving delivery timelines. With real-time state detection, managers can ensure a more efficient, cost-effective, and robust production process. Whenever the states change, accurate detection within less time becomes essential for effective decision-making in production planning, scheduling, condition monitoring, predictive maintenance, and asset management. However, a trade-off exists

Table 20 Sample Product Description

Product ID	Comprising states (in sequence of occurrence)			
1	S_5	S_4	S_6	S_8
2	S_6	S_9	S_8	S_5
3	S_7	S_3	S_4	S_2
4	S_1	S_2	S_7	S_6

between how much data should be considered for calculating the deviation

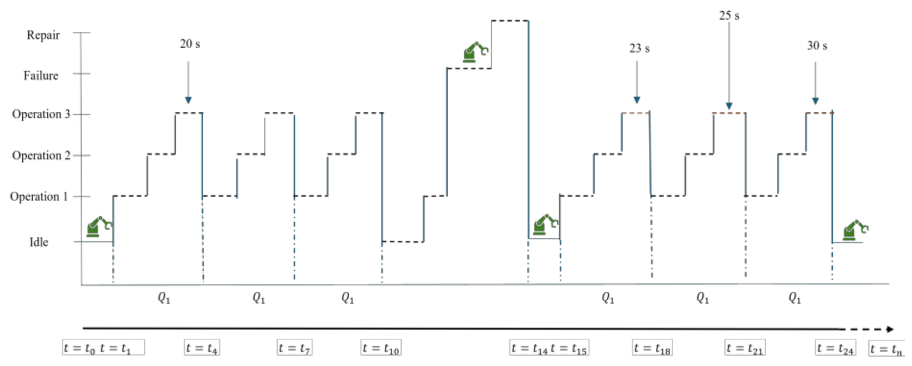


Figure 22 Illustration of stimulus-aware digital twins

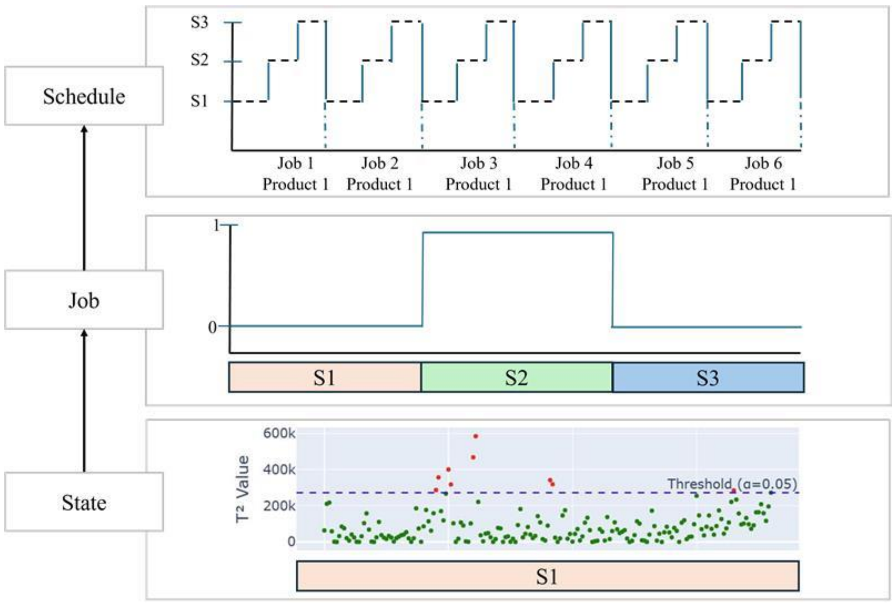


Figure 23 Concept of State Change used to Describe the Job and Schedule

metric for comparison against the threshold. The more data is used, the longer it takes to detect the state change. On the other hand, using less data can lead to errors in state change detection, decreasing the performance and reliability of the method. To evaluate the impact of data quantity on state change detection performance, a new set of experiments has been designed that simulate time-series behaviour similar to real-life production environments. Figure 23 describes the concept of state and state change and its use to describe a job and further to describe a schedule. A job is described as a pattern of state changes, and a schedule is a repeating pattern of jobs. It becomes important to detect the change of state as soon as it occurs for accurate modeling of the digital twin. The following Table 20 shows sample products considering various states in the sequence of occurrence. For example, Product 1 can be represented as an occurrence of states in a sequence of S_5, S_4, S_6, S_8, with each state existing for a given time duration. In the proposed architecture, during the detection step, if the T^2 metric remains above the threshold for a certain number of samples corresponding to a detection time t_d, a change in state is declared. Experiments are conducted to evaluate the impact of this detection time t_d (corresponding to different data quantity) in the detection step. As shown in Table 21, four values of detection times are evaluated: 2.21 seconds, 3.31 seconds, 4.42 seconds, and 5.52 seconds. For performance evaluation, the detected number of state changes is compared with the actual number of state changes in the job. The following table compares the performance of the proposed approach under the four levels of detection time for a Product ID.

The results show the proposed approach's robustness in terms of state change detection accuracy across reductions in detection time. Without using domain adaptation, it can be observed that the state change detection percentage improves with an increase in the detection time (and thus data quantity). However, the proposed approach shows a superior performance for a small detection time of 2.2 seconds.

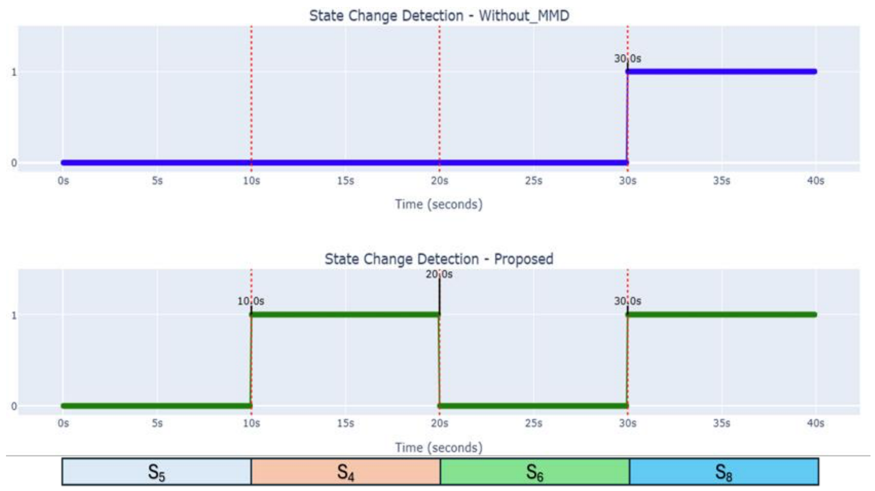


Figure 24 Visualization of state change detection performance for a detection time of 2.2 seconds for a sample product 1

Table 21 Impact of data quantity used for detection on performance of the proposed approach for a sample product

			Case 1 (No MMD)		**Case 2 (Proposed)**	
Product ID	Detection Time	True State Changes	Detected State Changes	Detection Percentage	Detected State Changes	Detection Percentage
1	5.52	3	3	100	3	100
	4.42	3	2	66.67	3	**100**
	3.31	3	2	66.67	3	**100**
	2.21	3	1	33.33	3	**100**

Table 22 Sample Product Description

Product ID	Comprising states (in sequence of occurrence)				Cycle Time (s)
1	S_5	S_4	S_6	S_8	40
2	S_6	S_9	S_8	S_5	40
3	S_7	S_3	S_4	S_2	40
4	S_1	S_2	S_7	S_6	40

Figure 24 illustrates the example further. Further, the experiments are repeated using the detection time of 2.2 seconds for multiple products and multiple training states to evaluate the robustness, and are shown in Table 23. The proposed approach is able to showcase a superior performance of 88.9% detection percentage compared to 30.6% without using domain adaptation across variation in training states, as well as the type of products. This further shows that even with less data for detection, as well, the efficiency of state detection is maintained. This efficient state detection can be highly beneficial for production monitoring, which is very essential for effective management of production and resources, as well as improving planning. The integration of state change detection with the production monitoring use case allows for timely monitoring of shop floor performance by managers and operators. It allows the shop floor managers to have a synchronized status view of the shop floor with enhanced insights about the asset states and condition, ease of deployment through scalability and reduced costs, and opportunity to take quick actions possible with the requirement of less data. State detection facilitates the overall monitoring of equipment behavior, enabling the tracking of various metrics, such as the time required to manufacture a specific product and the average cycle time. We believe this approach is a vital component of Industry 4.0-driven production economics. To show the production monitoring use case, the time-series behavior of real-life production is created. Figure 23 describes the concept of state and state change and its use to describe a job and further to describe a production schedule. A job is described as a pattern of state changes, and a schedule is a repeating pattern of jobs. It becomes important to detect the state change as soon as it occurs for accurate modeling of the digital twin. The objective of the use case is to effectively capture the true cycle time of jobs in a production schedule. This would allow production managers to monitor shopfloor production for better estimation of production capacity and delivery times. Table 22 shows the products considering various states, representing various operations, in the sequence of occurrence that would be used in the use case. For example, Product 1 can be represented as an occurrence of states in a sequence of S_5, S_4, S_6, and

Table 23 Performance comparison of the proposed approach for a detection time of 2.2 seconds across varying products and varying training states.

Training State	Product ID	True State Changes	Case 1 (No MMD)		**Case 2 (Proposed)**	
			Detected State Changes	Detection Percentage	Detected State Changes	Detection Percentage
S_1	1	3	1	66.67	2	66.67
	2	3	3	0	3	100
	3	3	1	66.67	3	100
	4	3	2	33.33	2	66.67
S_8	1	3	3	0	3	100
	2	3	2	33.33	3	100
	3	3	2	33.33	4	66.67
	4	3	2	33.33	3	100
S6	1	3	3	0	3	100
	2	3	1	66.67	3	100
	3	3	3	0	3	100
	4	3	2	33.33	2	66.67
			Average	30.56	**Average**	**88.89**

Table 24 Comparison of Makespan estimation for production schedules

Production Schedule (Product IDs)	True Makespan (s)	Case 1 (No MMD)		**Case 2 (Proposed)**	
		Estimated Makespan (s)	Estimation Error (s)	Estimated Makespan (s)	Estimation Error (s)
[1, 4, 1, 1, 1, 4]	240	397.6	157.6	**280**	**40**
[3, 4, 3, 4, 4, 3]	240	310	70	**260**	**20**

S_8, with each state existing for a given time duration. In the following table, Column 1 represents the Product ID, which is the unique name for each product. The remaining column represents the operating states through which each specific product is produced. Each job is a combination of these different operating states; for example, S_7 operates at 400 RPM and a 2-kW load when the equipment is in healthy condition. Dataset I is used in this use case. Each product is given a true cycle time of 40 seconds, distributed equally for each state. Table 25 presents the results comparing the estimated cycle time for each product under different training states for the proposed approach vs. without using domain adaptation. It can be noted that for each product under each scenario, the proposed method produces a better or on-par estimate of the cycle time. The average error for the proposed method is 4.43 seconds (11%) compared to 13.45 seconds (33.6%). The proposed method also shows much less variation in the estimation across products, as can be seen through the standard deviation. This shows the superiority of the proposed method in the realistic estimation of product cycle time using

Table 25 Comparison of Cycle Time Estimation across products and training states

Training State	Product ID	True Product Cycle Time (s)	Case 1 (No MMD)		**Case 2 (Proposed)**	
			Estimated Cycle Time (s)	Estimation Error (s)	Estimated Cycle Time (s)	Estimation Error (s)
S_1	1	40	70	30	50	10
	2	40	40	0	40	0
	3	40	60	20	40	0
	4	40	60.22	20.22	40	0
S_8	1	40	40	0	40	0
	2	40	51.71	11.71	50	10
	3	40	50	10	50	10
	4	40	66.68	26.68	50	10
S_6	1	40	43.53	3.53	40	0
	2	40	69.32	29.32	42.1	2.1
	3	40	40	0	40	0
	4	40	50	10	50	10
		Mean	53.45	13.45	**44.34**	**4.34**
		Std. Dev.		11.51		**5.03**

minimal data. These errors per product can further accumulate over production schedules, impacting the production completion time estimation or makespan estimation. Further, two production schedules have been defined, as shown in Table 24. The first job in each production schedule begins at time t = 0. The cycle time is defined as the completion time of the last job that is completed. The schedule was generated for six randomly selected products from the previous table. The cycle time and deviation in state detection were calculated in two scenarios: without domain adaptation and with the proposed method. The results show a similar performance compared to cycle time estimation. The proposed method shows a superior performance with an error of less than 17% compared to more than 30% for the method without domain adaptation against the true makespan for both production schedules. Figure 25 illustrates the results further. This allows for a more realistic and accurate prediction of completion time for improved resource planning and management for the shop floor managers.

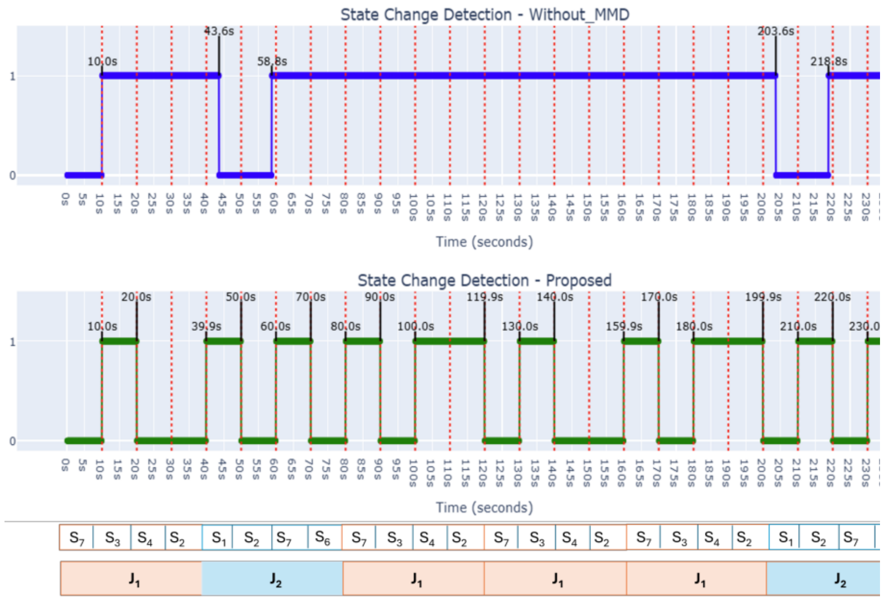


Figure 25 Visualization of State change detection in a sample production schedule for makespan estimation

Chapter 9 Conclusion and Future Scope

9.1 Conclusion

The challenge of developing stimulus-aware digital twins for complex production systems is addressed in this work. A four-phase semi-autonomous learning framework is proposed to create stimulus-aware digital twins of production equipment from the beginning of asset operation. An unsupervised 1D Convolutional Autoencoder with PCA-T^2 is used for initial model training. Upon detection and validation of a new state, the model is adapted by using a novel implementation of a custom loss function to maximize the distribution discrepancy between the latent representation of these states. The proposed method is validated over three real-life datasets, including a newly created gearbox test rig dataset for this study. The datasets span over variation in equipment type, signal type, operating conditions, etc., representing the challenge of scalable creation of digital twins in real-life production scenarios. The validation study included conducting 84 experiments of model training and testing, repeated 5 times each, across different scenarios. The results observed a dependency of the performance of an autoencoder model without domain adaptation on the choice of the training state, as well as the equipment and its associated parameters, as expected. To further benchmark the performance of the autoencoder model learning, Generative Adversarial Networks (GANs) have also been explored. An additional 10 experiments were conducted across various scenarios, with each experiment repeated five times to assess the model's performance. The WGANs model is not performing well across all the datasets. However, the proposed method was able to achieve the performance of being able to detect a new state with an accuracy of more than 97.5%, irrespective of the equipment or the training state. Further, the proposed method allows the creation of stimulus-aware digital twins using only two states at the beginning of equipment operation. Such a scalable approach is expected to achieve generalized state detection and state sustenance knowledge capability that form the fundamental requirement for realistic and robust prediction and simulation through the digital twin for enhanced decision-making. The ability of the approach to detect operational states beyond the heavily focused faulty states allows for effective modeling of production behavior critical to efficient planning of time, money, and resources, ultimately impacting production economics.

9.2 Future Scope

With the current findings as a stepping stone, several future research opportunities arise, such as:

1. Future research can explore the development of a unified framework that integrates multiple types of observational data, numerical, visual (e.g., images or video), and textual (e.g., maintenance reports, operator logs). Such a multimodal approach could enhance the digital twin's ability to understand system states and anomalies by capturing diverse perspectives of the physical environment, especially in scenarios where numerical sensor data alone is insufficient or unavailable.
2. In situations where sensor installation is impractical, such as in rotating or hard-to-reach machinery, future studies can focus on utilizing computer vision techniques for state detection and event recognition. This includes deploying cameras to monitor operational behavior and applying AI-based image analysis methods (e.g., object detection, motion analysis, thermal imaging) to extract meaningful insights for digital twin updates and predictive maintenance.
3. Further research can be explored to enable autonomous learning for the developed model based on the detected state. This study implemented domain adaptation right after the model made its detections, significantly enhancing the model's performance. Based

on this performance, humans will decide when to reapply domain adaptation to address any issues that arise.

4. Additionally, we can explore some quantification methods to identify the accumulation of significant new information. This could help us automate the process of re-triggering domain adaptation during the operation of production equipment, further enhancing performance. Further integration of the role of human inputs in the development and maintenance of digital twins, especially when initial training performance is not acceptable, can be explored.
5. The current methodology has been developed to create a stimulus-aware digital twin capable of detecting changes in system state. However, further research is needed to classify the specific types of state changes, which would enable a deeper understanding of the underlying causes and improve diagnostic capabilities.
6. While current methodologies equip digital twins with stimulus-awareness, enabling them to perceive and respond to changes in their own states, they remain largely isolated in their interpretation of system-wide dynamics. A critical future direction lies in developing interaction-aware capabilities, where a digital twin can dynamically assess and interpret the influence of changes occurring in other interconnected twins. Such advancements would not only enable seamless information exchange but also give the way for intelligent, collaborative behavior across complex digital ecosystems. This shift from isolated perception to mutual awareness represents a foundational step toward fully autonomous and adaptive digital twin networks.

   It addresses key challenges in modern manufacturing, including the lack of system-wide visibility, delayed response to interdependent failures, and inefficient data silos. By facilitating seamless information flow and mutual context-awareness among machines, systems, and processes, interaction-aware digital twins can drive the transition toward the autonomous, self-optimizing smart factories.

7. The integration of domain adaptation into GANs required substantial research and investigation. In the current work, GANs have been applied without domain adaptation; however, future research could explore the impact of incorporating domain adaptation on model performance.
8. Future work could involve validating the proposed approach with datasets that contain multiple simultaneous failure modes. This would help demonstrate the model's effectiveness in more complex and realistic industrial scenarios.

References

[1] K. J. Wang, Y. H. Lee, and S. Angelica, “Digital twin design for real-time monitoring–a case study of die cutting machine,” *Int J Prod Res*, vol. 59, no. 21, pp. 6471–6485, 2021, doi: 10.1080/00207543.2020.1817999.

[2] J. Morgan and G. E. O’Donnell, “Cyber physical process monitoring systems,” *J Intell Manuf*, vol. 29, no. 6, pp. 1317–1328, Aug. 2018, doi: 10.1007/s10845-015-1180-z.

[3] W. Luo, T. Hu, Y. Ye, C. Zhang, and Y. Wei, “A hybrid predictive maintenance approach for CNC machine tool driven by Digital Twin,” *Robot Comput Integr Manuf*, vol. 65, Oct. 2020, doi: 10.1016/j.rcim.2020.101974.

[4] X. Tong, Q. Liu, S. Pi, and Y. Xiao, “Real-time machining data application and service based on IMT digital twin,” *J Intell Manuf*, vol. 31, no. 5, pp. 1113–1132, Jun. 2020, doi: 10.1007/s10845-019-01500-0.

[5] V. Pandhare, J. Singh, and J. Lee, “Convolutional Neural Network Based Rolling-Element Bearing Fault Diagnosis for Naturally Occurring and Progressing Defects Using Time-Frequency Domain Features,” in *Proceedings - 2019 Prognostics and System Health Management Conference, PHM-Paris 2019*, Institute of Electrical and Electronics Engineers Inc., May 2019, pp. 320–326. doi: 10.1109/PHM-Paris.2019.00061.

[6] K. T. Park *et al.*, “Design and implementation of a digital twin application for a connected micro smart factory,” *Int J Comput Integr Manuf*, vol. 32, no. 6, pp. 596–614, Jun. 2019, doi: 10.1080/0951192X.2019.1599439.

[7] Z. Lai, C. Yang, S. Lan, L. Wang, W. Shen, and L. Zhu, “BearingFM: Towards a foundation model for bearing fault diagnosis by domain knowledge and contrastive learning,” *Int J Prod Econ*, vol. 275, Sep. 2024, doi: 10.1016/j.ijpe.2024.109319.

[8] T. Nguyen, Q. H. Duong, T. Van Nguyen, Y. Zhu, and L. Zhou, “Knowledge mapping of digital twin and physical internet in Supply Chain

Management: A systematic literature review," *Int J Prod Econ*, vol. 244, Feb. 2022, doi: 10.1016/j.ijpe.2021.108381.

[9] B. R. Barricelli, E. Casiraghi, and D. Fogli, "A survey on digital twin: Definitions, characteristics, applications, and design implications," 2019, *Institute of Electrical and Electronics Engineers Inc.* doi: 10.1109/ACCESS.2019.2953499.

[10] D. Jones, C. Snider, A. Nassehi, J. Yon, and B. Hicks, "Characterising the Digital Twin: A systematic literature review," *CIRP J Manuf Sci Technol*, vol. 29, pp. 36–52, May 2020, doi: 10.1016/j.cirpj.2020.02.002.

[11] E. VanDerHorn and S. Mahadevan, "Digital Twin: Generalization, characterization and implementation," *Decis Support Syst*, vol. 145, Jun. 2021, doi: 10.1016/j.dss.2021.113524.

[12] M. G. Kapteyn, "Mathematical and Computational Foundations to Enable Predictive Digital Twins at Scale."

[13] D. Ivanov, "Intelligent digital twin (iDT) for supply chain stress-testing, resilience, and viability," *Int J Prod Econ*, vol. 263, Sep. 2023, doi: 10.1016/j.ijpe.2023.108938.

[14] K. Y. H. Lim, L. Van Dang, and C. H. Chen, "Incorporating supply and production digital twins to mitigate demand disruptions in multi-echelon networks," *Int J Prod Econ*, vol. 273, Jul. 2024, doi: 10.1016/j.ijpe.2024.109258.

[15] H. Elayan, M. Aloqaily, and M. Guizani, "Digital Twin for Intelligent Context-Aware IoT Healthcare Systems," *IEEE Internet Things J*, vol. 8, no. 23, pp. 16749–16757, Dec. 2021, doi: 10.1109/JIOT.2021.3051158.

[16] A. Croatti, M. Gabellini, S. Montagna, and A. Ricci, "On the Integration of Agents and Digital Twins in Healthcare", doi: 10.1007/s10916-020-01623-5.

[17] D. Allaire, G. Biros, J. Chambers, O. Ghattas, D. Kordonowy, and K. Willcox, "Dynamic data driven methods for self-aware aerospace vehicles," in *Procedia Computer Science*, Elsevier B.V., 2012, pp. 1206–1210. doi: 10.1016/j.procs.2012.04.130.

[18] M. Abdullah, A. Faruque, D. Muthirayan, S.-Y. Yu, and P. P. Khargonekar, *Cognitive Digital Twin for Manufacturing Systems*.

[19] N. Zhang, R. Bahsoon, and G. Theodoropoulos, "Towards Engineering Cognitive Digital Twins with Self-Awareness," *Conf Proc IEEE Int Conf Syst Man Cybern*, vol. 2020-October, pp. 3891–3896, Oct. 2020, doi: 10.1109/SMC42975.2020.9283357.

[20] T. Chen *et al.*, "The Handbook of Engineering Self-Aware and Self-Expressive Systems," Sep. 2014, [Online]. Available: http://arxiv.org/abs/1409.1793

[21] N. Zhang, … R. B.-2020 I. I., and undefined 2020, "Towards engineering cognitive digital twins with self-awareness," *ieeexplore.ieee.orgN Zhang, R Bahsoon, G Theodoropoulos2020 IEEE International Conference on Systems, Man, and, 2020•ieeexplore.ieee.org*, Accessed: Jan. 21, 2025. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9283357/

[22] P. R. Lewis *et al.*, "5 / $ 3 1. 0 0 © 2 0 1 5 I E E E Architectural Aspects of Self-Aware and Self-Expressive Computing Systems: From Psychology to Engineering." [Online]. Available: www.cs.pdx.edu/~mm/self-awareness.pdf.

[23] A. Thelen *et al.*, "A comprehensive review of digital twin — part 1: modeling and twinning enabling technologies," Dec. 01, 2022, *Springer Science and Business Media Deutschland GmbH*. doi: 10.1007/s00158-022-03425-4.

[24] X. Zheng, J. Lu, and D. Kiritsis, "The emergence of cognitive digital twin: vision, challenges and opportunities," *Int J Prod Res*, vol. 60, no. 24, pp. 7610–7632, 2022, doi: 10.1080/00207543.2021.2014591.

[25] M. G. Kapteyn, J. V. R. Pretorius, and K. E. Willcox, "A Probabilistic Graphical Model Foundation for Enabling Predictive Digital Twins at Scale."

[26] A. Thelen *et al.*, "A comprehensive review of digital twin — part 1: modeling and twinning enabling technologies," Dec. 01, 2022, *Springer*

Science and Business Media Deutschland GmbH. doi: 10.1007/s00158-022-03425-4.

[27] V. Pandhare, E. Negri, L. Ragazzini, L. Cattaneo, M. Macchi, and J. Lee, "Digital twin-enabled robust production scheduling for equipment in degraded state," *J Manuf Syst*, vol. 74, pp. 841–857, Jun. 2024, doi: 10.1016/j.jmsy.2024.04.027.

[28] X. Zhang, L. Zheng, W. Fan, W. Ji, L. Mao, and L. Wang, "Knowledge graph and function block based Digital Twin modeling for robotic machining of large-scale components," *Robot Comput Integr Manuf*, vol. 85, Feb. 2024, doi: 10.1016/j.rcim.2023.102609.

[29] H. Fan, Z. Ren, X. Zhang, X. Cao, H. Ma, and J. Huang, "A gray texture image data-driven intelligent fault diagnosis method of induction motor rotor-bearing system under variable load conditions," *Measurement (Lond)*, vol. 233, Jun. 2024, doi: 10.1016/j.measurement.2024.114742.

[30] A. Kumar, R. Kumar, J. Xiang, Z. Qiao, Y. Zhou, and H. Shao, "Digital twin-assisted AI framework based on domain adaptation for bearing defect diagnosis in the centrifugal pump," *Measurement (Lond)*, vol. 235, Aug. 2024, doi: 10.1016/j.measurement.2024.115013.

[31] J. Wang, L. Ye, R. X. Gao, C. Li, and L. Zhang, "Digital Twin for rotating machinery fault diagnosis in smart manufacturing," *Int J Prod Res*, vol. 57, no. 12, pp. 3920–3934, Jun. 2019, doi: 10.1080/00207543.2018.1552032.

[32] "Intelligent cross-machine fault diagnosis approach with deep auto-encoder and domain adaptation".

[33] W. Luo, T. Hu, C. Zhang, and Y. Wei, "Digital twin for CNC machine tool: modeling and using strategy," *J Ambient Intell Humaniz Comput*, vol. 10, no. 3, pp. 1129–1140, Mar. 2019, doi: 10.1007/s12652-018-0946-5.

[34] C. Hu, B. D. Youn, and J. Chung, "A multiscale framework with extended Kalman filter for lithium-ion battery SOC and capacity estimation," *Appl Energy*, vol. 92, pp. 694–704, 2012, doi: 10.1016/j.apenergy.2011.08.002.

[35] W. Li, M. Rentemeister, J. Badeda, D. Jöst, D. Schulte, and D. U. Sauer, "Digital twin for battery systems: Cloud battery management system with

online state-of-charge and state-of-health estimation,” *J Energy Storage*, vol. 30, Aug. 2020, doi: 10.1016/j.est.2020.101557.

[36] M. G. Kapteyn, D. J. Knezevic, D. B. P. Huynh, M. Tran, and K. E. Willcox, “Data-driven physics-based digital twins via a library of component-based reduced-order models,” *Int J Numer Methods Eng*, vol. 123, no. 13, pp. 2986–3003, Jul. 2022, doi: 10.1002/nme.6423.

[37] *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2020.

[38] V. Singh and K. E. Willcox, “Methodology for path planning with dynamic data-driven flight capability estimation,” in *17th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, American Institute of Aeronautics and Astronautics Inc, AIAA, 2016. doi: 10.2514/6.2016-4124.

[39] J. Vickers and M. Grieves, “Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems,” 2017, doi: 10.1007/978-3-319-38756-7_4.

[40] G. Angjeliu, D. Coronelli, and G. Cardani, “Development of the simulation model for Digital Twin applications in historical masonry buildings: The integration between numerical and experimental reality,” *Comput Struct*, vol. 238, Oct. 2020, doi: 10.1016/j.compstruc.2020.106282.

[41] M. Wang, S. Feng, A. Incecik, G. Królczyk, and Z. Li, “Structural fatigue life prediction considering model uncertainties through a novel digital twin-driven approach,” *Comput Methods Appl Mech Eng*, vol. 391, Mar. 2022, doi: 10.1016/j.cma.2021.114512.

[42] M. Zhang, F. Tao, and A. Y. C. Nee, “Digital Twin Enhanced Dynamic Job-Shop Scheduling,” *J Manuf Syst*, vol. 58, pp. 146–156, Jan. 2021, doi: 10.1016/j.jmsy.2020.04.008.

[43] Y. Wang, R. Xu, C. Zhou, X. Kang, and Z. Chen, “Digital twin and cloud-side-end collaboration for intelligent battery management system,” *J Manuf Syst*, vol. 62, pp. 124–134, Jan. 2022, doi: 10.1016/j.jmsy.2021.11.006.

[44] T. G. Ritto and F. A. Rochinha, "Digital twin, physics-based model, and machine learning applied to damage detection in structures," *Mech Syst Signal Process*, vol. 155, Jun. 2021, doi: 10.1016/j.ymssp.2021.107614.

[45] D. Bzdok, N. Altman, and M. Krzywinski, "Points of Significance: Statistics versus machine learning," Apr. 03, 2018, *Nature Publishing Group*. doi: 10.1038/nmeth.4642.

[46] C. Liu, L. Le Roux, C. Körner, O. Tabaste, F. Lacan, and S. Bigot, "Digital Twin-enabled Collaborative Data Management for Metal Additive Manufacturing Systems," *J Manuf Syst*, vol. 62, pp. 857–874, Jan. 2022, doi: 10.1016/j.jmsy.2020.05.010.

[47] M. Tang, Y. Liu, and L. J. Durlofsky, "A deep-learning-based surrogate model for data assimilation in dynamic subsurface flow problems," *J Comput Phys*, vol. 413, Jul. 2020, doi: 10.1016/j.jcp.2020.109456.

[48] M. A. F. Pimentel, D. A. Clifton, L. Clifton, and L. Tarassenko, "A review of novelty detection," Jun. 2014. doi: 10.1016/j.sigpro.2013.12.026.

[49] L. Ruff *et al.*, "A Unifying Review of Deep and Shallow Anomaly Detection," *Proceedings of the IEEE*, vol. 109, no. 5, pp. 756–795, May 2021, doi: 10.1109/JPROC.2021.3052449.

[50] M. Taisch, B. Stahl, F. Vaccari, and A. Cataldo, "A Production-State Based Approach for Energy Flow Simulation in Manufacturing Systems A Production-State Based Approach for Energy Flow Simulation in Manufacturing," p. 10, 2013, doi: 10.1007/978-3-642-41266.

[51] S. Mousavi, S. Thiede, W. Li, S. Kara, and C. Herrmann, "An integrated approach for improving energy efficiency of manufacturing process chains," *International Journal of Sustainable Engineering*, vol. 9, no. 1, pp. 11–24, Jan. 2016, doi: 10.1080/19397038.2014.1001470.

[52] S. Vadi, R. Bayindir, Y. Toplar, and I. Colak, "Induction motor control system with a Programmable Logic Controller (PLC) and Profibus communication for industrial plants — An experimental setup," *ISA Trans*, vol. 122, pp. 459–471, Mar. 2022, doi: 10.1016/j.isatra.2021.04.019.

[53] E. B. Priyanka, C. Maheswari, S. Thangavel, and M. P. Bala, “Integrating IoT with LQR-PID controller for online surveillance and control of flow and pressure in fluid transportation system,” *J Ind Inf Integr*, vol. 17, Mar. 2020, doi: 10.1016/j.jii.2020.100127.

[54] J. Tao *et al.*, “Timely chatter identification for robotic drilling using a local maximum synchrosqueezing-based method,” *J Intell Manuf*, vol. 31, no. 5, pp. 1243–1255, Jun. 2020, doi: 10.1007/s10845-019-01509-5.

[55] Y. Wang, M. Zhang, X. Tang, F. Peng, and R. Yan, “A kMap optimized VMD-SVM model for milling chatter detection with an industrial robot,” *J Intell Manuf*, vol. 33, no. 5, pp. 1483–1502, Jun. 2022, doi: 10.1007/s10845-021-01736-9.

[56] “IEEE Xplore Full-Text PDF:” Accessed: Mar. 09, 2025. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5722047

[57] W. Luo, T. Hu, C. Zhang, and Y. Wei, “Digital twin for CNC machine tool: modeling and using strategy,” vol. 10, pp. 1129–1140, 2019, doi: 10.1007/s12652-018-0946-5.

[58] “IEEE Xplore Full-Text PDF:” Accessed: Mar. 09, 2025. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8835120

[59] J. Liu, J. Liu, C. Zhuang, Z. Liu, and T. Miao, “Construction method of shop-floor digital twin based on MBSE,” *J Manuf Syst*, vol. 60, pp. 93–118, Jul. 2021, doi: 10.1016/J.JMSY.2021.05.004.

[60] M. Glatt, C. Sinnwell, L. Yi, S. Donohoe, … B. R.-J. of M., and undefined 2021, “Modeling and implementation of a digital twin of material flows based on physics simulation,” *Elsevier*, Accessed: Mar. 09, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0278612520300595

[61] W. Luo, T. Hu, C. Zhang, and Y. Wei, “Digital twin for CNC machine tool: modeling and using strategy,” *J Ambient Intell Humaniz Comput*, vol. 10, no. 3, pp. 1129–1140, Mar. 2019, doi: 10.1007/S12652-018-0946-5/FIGURES/12.

[62] "SYSMOD - The Systems Modeling Toolbox: Pragmatic MBSE with SysML - Tim Weilkiens - Google Books." Accessed: Mar. 09, 2025. [Online]. Available: https://books.google.co.in/books?hl=en&lr=&id=I_3xEAAAQBAJ&oi=fnd&pg=PP3&dq=(Weilkiens,+2011),&ots=aDPUlXwE1z&sig=7qCHS_NuqqERfmpobIQM3JTcv-Q&redir_esc=y#v=onepage&q=(Weilkiens%2C%202011)%2C&f=false

[63] H. Marah, "Intelligent Agents and Multi Agent Systems for Modeling Smart Digital Twins".

[64] V. Gorodetsky, S. Kozhevnikov, … D. N.-I. A. of, and undefined 2019, "The framework for designing autonomous cyber-physical multi-agent systems for adaptive resource management," *SpringerVI Gorodetsky, SS Kozhevnikov, D Novichkov, PO SkobelevIndustrial Applications of Holonic and Multi-Agent Systems: 9th International, 2019•Springer*, vol. 11710 LNAI, pp. 52–64, 2019, doi: 10.1007/978-3-030-27878-6_5.

[65] L. Clifton, D. A. Clifton, Y. Zhang, P. Watkinson, L. Tarassenko, and H. Yin, "Probabilistic novelty detection with support vector machines," *IEEE Trans Reliab*, vol. 63, no. 2, pp. 455–467, 2014, doi: 10.1109/TR.2014.2315911.

[66] D. M. J. Tax and R. P. W. Duin, "Support Vector Data Description," 2004.

[67] Y. Xie, K. Lian, Q. Liu, C. Zhang, and H. Liu, "Digital twin for cutting tool: Modeling, application and service strategy," *J Manuf Syst*, vol. 58, pp. 305–312, Jan. 2021, doi: 10.1016/J.JMSY.2020.08.007.

[68] F. Peng, L. Zheng, Y. Peng, C. Fang, and X. Meng, "Digital Twin for rolling bearings: A review of current simulation and PHM techniques," *Measurement*, vol. 201, p. 111728, Sep. 2022, doi: 10.1016/J.MEASUREMENT.2022.111728.

[69] W. Luo, T. Hu, Y. Ye, C. Zhang, and Y. Wei, "A hybrid predictive maintenance approach for CNC machine tool driven by Digital Twin," *Robot Comput Integr Manuf*, vol. 65, p. 101974, Oct. 2020, doi: 10.1016/J.RCIM.2020.101974.

[70] X. Liu, L. Bo, H. L.- Measurement, and undefined 2015, “Bearing faults diagnostics based on hybrid LS-SVM and EMD method,” *ElsevierX Liu, L Bo, H LuoMeasurement, 2015•Elsevier*, Accessed: Mar. 09, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0263224114004175

[71] D. Fernández-Francos, D. Marténez-Rego, O. Fontenla-Romero, and A. Alonso-Betanzos, “Automatic bearing fault diagnosis based on one-class ν-SVM,” *Comput Ind Eng*, vol. 64, no. 1, pp. 357–365, Jan. 2013, doi: 10.1016/J.CIE.2012.10.013.

[72] L. Jiang, X. Fu, J. Cui, Z. L.-2012 24th C. C. and, and undefined 2012, “Fault detection of rolling element bearing based on principal component analysis,” *ieeexplore.ieee.orgL Jiang, X Fu, J Cui, Z Li2012 24th Chinese Control and Decision Conference (CCDC), 2012•ieeexplore.ieee.org*, Accessed: Mar. 09, 2025. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6243071/

[73] F. Wang, J. Cao, B. C.-2007 I. C. on, and undefined 2007, “Nonlinear feature fusion scheme based on kernel PCA for machine condition monitoring,” *ieeexplore.ieee.orgF Wang, J Cao, B Cao2007 International Conference on Mechatronics and Automation, 2007•ieeexplore.ieee.org*, Accessed: Mar. 09, 2025. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/4303615/

[74] R. Zhao *et al.*, “Decision tree based parameter identification and state estimation: Application to Reactor Operation Digital Twin,” *Nuclear Engineering and Technology*, p. 103527, Feb. 2025, doi: 10.1016/J.NET.2025.103527.

[75] K. Guo, X. Wan, L. Liu, Z. Gao, and M. Yang, “Fault Diagnosis of Intelligent Production Line Based on Digital Twin and Improved Random Forest,” *Applied Sciences 2021, Vol. 11, Page 7733*, vol. 11, no. 16, p. 7733, Aug. 2021, doi: 10.3390/APP11167733.

[76] X. Zhang, C. Sheng, W. Ouyang, and L. Zheng, “Fault diagnosis of marine electric thruster bearing based on fusing multi-sensor deep learning

models," *Measurement (Lond)*, vol. 214, Jun. 2023, doi: 10.1016/j.measurement.2023.112727.

[77] Y. Li, G. Cheng, C. Liu, and X. Chen, "Study on planetary gear fault diagnosis based on variational mode decomposition and deep neural networks," *Measurement (Lond)*, vol. 130, pp. 94–104, Dec. 2018, doi: 10.1016/j.measurement.2018.08.002.

[78] L. Xiang, X. Zhang, Y. Zhang, A. Hu, and H. Bing, "A novel method for rotor fault diagnosis based on deep transfer learning with simulated samples," *Measurement (Lond)*, vol. 207, Feb. 2023, doi: 10.1016/j.measurement.2022.112350.

[79] R. Chalapathy, A. K. Menon, and S. Chawla, "Robust, Deep and Inductive Anomaly Detection," Apr. 2017, [Online]. Available: http://arxiv.org/abs/1704.06743

[80] S. Tang, S. Yuan, and Y. Zhu, "Convolutional Neural Network in Intelligent Fault Diagnosis Toward Rotatory Machinery," *IEEE Access*, vol. 8, pp. 86510–86519, 2020, doi: 10.1109/ACCESS.2020.2992692.

[81] Y. Zhu, C. Zhu, J. Tan, Y. Tan, and L. Rao, "Anomaly detection and condition monitoring of wind turbine gearbox based on LSTM-FS and transfer learning," *Renew Energy*, vol. 189, pp. 90–103, Apr. 2022, doi: 10.1016/j.renene.2022.02.061.

[82] L.- Rubio, E. Palomo, E. Domínguez, S. Chen, and W. Guo, "Auto-encoders in deep learning—a review with new perspectives," *mdpi.comS Chen, W GuoMathematics, 2023•mdpi.com*, 2023, doi: 10.3390/math11081777.

[83] O. Fink, Q. Wang, M. Svensén, P. Dersin, W. J. Lee, and M. Ducoffe, "Potential, challenges and future directions for deep learning in prognostics and health management applications," *Eng Appl Artif Intell*, vol. 92, Jun. 2020, doi: 10.1016/j.engappai.2020.103678.

[84] W. Li *et al.*, "A perspective survey on deep transfer learning for fault diagnosis in industrial scenarios: Theories, applications and challenges,"

Mech Syst Signal Process, vol. 167, Mar. 2022, doi: 10.1016/j.ymssp.2021.108487.

[85] Z. Zhu *et al.*, "A review of the application of deep learning in intelligent fault diagnosis of rotating machinery," Jan. 01, 2023, *Elsevier B.V.* doi: 10.1016/j.measurement.2022.112346.

[86] V. Pandhare, X. Li, M. Miller, X. Jia, and J. Lee, "Intelligent Diagnostics for Ball Screw Fault through Indirect Sensing Using Deep Domain Adaptation," *IEEE Trans Instrum Meas*, vol. 70, 2021, doi: 10.1109/TIM.2020.3043512.

[87] S. Xing, Y. Lei, S. Wang, and F. Jia, "Distribution-Invariant Deep Belief Network for Intelligent Fault Diagnosis of Machines under New Working Conditions," *IEEE Transactions on Industrial Electronics*, vol. 68, no. 3, pp. 2617–2625, Mar. 2021, doi: 10.1109/TIE.2020.2972461.

[88] Z. Yang, W. W. Cohen, and R. Salakhutdinov, "Revisiting Semi-Supervised Learning with Graph Embeddings," 2016.

[89] F. M. Cariucci, L. Porzi, B. Caputo, E. Ricci, and S. R. Bulo, "AutoDIAL: Automatic Domain Alignment Layers," in *Proceedings of the IEEE International Conference on Computer Vision*, Institute of Electrical and Electronics Engineers Inc., Dec. 2017, pp. 5077–5085. doi: 10.1109/ICCV.2017.542.

[90] H. Yan, Y. Ding, P. Li, Q. Wang, Y. Xu, and W. Zuo, "Mind the Class Weight Bias: Weighted Maximum Mean Discrepancy for Unsupervised Domain Adaptation."

[91] C.-Y. Lee, T. Batra, M. Haris Baig, and D. Ulbricht, "Sliced Wasserstein Discrepancy for Unsupervised Domain Adaptation."

[92] Y. Ganin, V. Lempitsky, and L. Ru, "Unsupervised Domain Adaptation by Backpropagation."

[93] Y. Ganin *et al.*, "Domain-Adversarial Training of Neural Networks," 2016.

[94] A. Gretton, A. Smola, J. Huang, M. Schmittfull, K. Borgwardt, and B. Schölkopf, "Covariate shift by kernel mean matching," *Citeseer*, Accessed: Mar. 05, 2025. [Online]. Available:

https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=06cfa748f8b548b55a1d7816f323029e0f69f4d1

[95] S. Pan, I. Tsang, … J. K.-I. transactions on, and undefined 2010, "Domain adaptation via transfer component analysis," *ieeexplore.ieee.orgSJ Pan, IW Tsang, JT Kwok, Q YangIEEE transactions on neural networks, 2010•ieeexplore.ieee.org*, Accessed: Mar. 05, 2025. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/5640675/

[96] B. Sun, J. Feng, and K. Saenko, "Correlation alignment for unsupervised domain adaptation," *Advances in Computer Vision and Pattern Recognition*, no. 9783319583464, pp. 153–171, 2017, doi: 10.1007/978-3-319-58347-1_8.

[97] S. Kullback, R. L.-T. annals of mathematical statistics, and undefined 1951, "On information and sufficiency," *JSTORS Kullback, RA LeiblerThe annals of mathematical statistics, 1951•JSTOR*, Accessed: Mar. 05, 2025. [Online]. Available: https://www.jstor.org/stable/2236703

[98] G. Kang, L. Jiang, … Y. Y.-P. of the, and undefined 2019, "Contrastive adaptation network for unsupervised domain adaptation," *openaccess.thecvf.comG Kang, L Jiang, Y Yang, AG HauptmannProceedings of the IEEE/CVF conference on computer vision and, 2019•openaccess.thecvf.com*, Accessed: Mar. 05, 2025. [Online]. Available: http://openaccess.thecvf.com/content_CVPR_2019/html/Kang_Contrastive_Adaptation_Network_for_Unsupervised_Domain_Adaptation_CVPR_2019_paper.html

[99] A. Gretton, K. M. Borgwardt, M. Rasch, B. Schölkopf, and A. J. Smola, "A Kernel Method for the Two-Sample-Problem." [Online]. Available: www.kyb.mpg.de/bs/people/arthur/mmd.htm

[100] Z. Tong, W. Li, B. Zhang, F. Jiang, and G. Zhou, "Bearing Fault Diagnosis under Variable Working Conditions Based on Domain Adaptation Using Feature Transfer Learning," *IEEE Access*, vol. 6, pp. 76187–76197, 2018, doi: 10.1109/ACCESS.2018.2883078.

[101] L. Huang, X. Nguyen, M. Garofalakis, M. I. Jordan, A. Joseph, and N. Taft, “In-network PCA and anomaly detection,” *proceedings.neurips.ccL Huang, XL Nguyen, M Garofalakis, M Jordan, A Joseph, N TaftAdvances in neural information processing systems, 2006•proceedings.neurips.cc*, Accessed: Mar. 05, 2025. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2006/hash/2227d753dc18505031869d44673728e2-Abstract.html

[102] D. Brauckhoff, K. Salamatian, and M. May, “Applying PCA for traffic anomaly detection: Problems and solutions,” *Proceedings - IEEE INFOCOM*, pp. 2866–2870, Jan. 2009, doi: 10.1109/INFCOM.2009.5062248.

[103] “Chen: Analysis and monitoring of batch processes... - Google Scholar.” Accessed: Mar. 05, 2025. [Online]. Available: https://scholar.google.com/scholar_lookup?title=Analysis%20and%20monitoring%20of%20batch%20processes%20using%20projection%20methods%3A%20an%20evaluation%20of%20alternative%20approaches&author=Y.%20Chen&publication_year=2002

[104] J. Camacho, J. Picó, and A. Ferrer, “The best approaches in the on-line monitoring of batch processes based on PCA: Does the modelling structure matter?,” *Anal Chim Acta*, vol. 642, no. 1–2, pp. 59–68, May 2009, doi: 10.1016/J.ACA.2009.02.001.

[105] Y. Ham, K. Han, J. Lin, M. G.-F.-V. in Engineering, and undefined 2016, “Visual monitoring of civil infrastructure systems via camera-equipped Unmanned Aerial Vehicles (UAVs): a review of related works,” *Springer*, vol. 4, no. 1, Dec. 2011, doi: 10.1186/s40327-015-0029-z.

[106] W. A. Smith and R. B. Randall, “Rolling element bearing diagnostics using the Case Western Reserve University data: A benchmark study,” Dec. 01, 2015, *Academic Press*. doi: 10.1016/j.ymssp.2015.04.021.

[107] M. Ismail, J. Windelberg, A. Bierig, I. Bravo-imaz, and A. Arnaiz, “Vibration Data for Axial Ball Bearings and Spall Faults,” vol. 2, 2022, doi: 10.17632/CHWHH9N3BF.2.

[108] M. A. A. Ismail, J. Windelberg, A. Bierig, I. Bravo, and A. Arnaiz, "Ball bearing vibration data for detecting and quantifying spall faults," *Data Brief*, vol. 47, p. 109019, Apr. 2023, doi: 10.1016/J.DIB.2023.109019.

[109] X. Wei, B. Gong, Z. Liu, W. Lu, and L. Wang, "IMPROVING THE IMPROVED TRAINING OF WASSERSTEIN GANS: A CONSISTENCY TERM AND ITS DUAL EFFECT".

[110] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved Training of Wasserstein GANs," *Adv Neural Inf Process Syst*, vol. 30, 2017.

[111] Y. Pan, R. Y. Zhong, T. Qu, L. Ding, and J. Zhang, "Multi-level digital twin-driven kitting-synchronized optimization for production logistics system," *Int J Prod Econ*, vol. 271, p. 109176, May 2024, doi: 10.1016/J.IJPE.2024.109176.

[112] M. Xia, H. Shao, D. Williams, S. Lu, L. Shu, and C. W. de Silva, "Intelligent fault diagnosis of machinery using digital twin-assisted deep transfer learning," *Reliab Eng Syst Saf*, vol. 215, p. 107938, Nov. 2021, doi: 10.1016/J.RESS.2021.107938.

[113] K. Y. H. Lim, P. Zheng, and C. H. Chen, "A state-of-the-art survey of Digital Twin: techniques, engineering product lifecycle management and business innovation perspectives," Aug. 01, 2020, *Springer*. doi: 10.1007/s10845-019-01512-w.