PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
الجمهـــوريـــــة الجزائـريـــــة الديـمقراطيـــــة الشعبيـــــة
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH
وزارة التعليـم العالـــي والبحـــث العلمـي
HIGHER SCHOOL OF COMPUTER SCIENCE AND DIGITAL TECHNOLOGIES-BEJAIA
المـــدرسة العـــليا في علـــوم وتكنـولوجيـات الاعـــلام الالـــي بجـــاية

Dissertation Submitted to the Department Of Computer Science in Partial Fulfillment of the Requirements for Engineer's Degree in Computer Science

Specialty: Artificial Intelligence and Data Sciences

Submitted By:

BELAYALI Rezkia **OURARI Tinhinane**

Predictive maintenance using Machine learning and sensor data

Supervised by:

Dr. DAOUDI Meroua ESTIN

Members of jury:

▪ **Dr.** MEDJOUDJ Rafik	President	ESTIN
▪ **Dr.** SEBAA Abderrazak	Examiner	ESTIN
▪ **Mrs.** MELCHANE Selestine	Examiner	ESTIN
▪ **Mrs.** AZZOUGUER Dalila	Examiner	ESTIN

Academic year: 2023/2024

Abstract

This thesis explores the relationship between Machine Learning (ML) and sensor data to achieve predictive maintenance (PdM) in Industry 4.0. The focus is on Deep Learning (DL) methods for developing a comprehensive predictive maintenance process using time series data from vibration sensors. The IMS (Center for Intelligent Maintenance Systems) bearing dataset serves as the foundation for this exploration. The process progresses from predicting the machine's next operational state, where the best result was achieved by an LSTM (Long Short-Term Memory) single model in the frequency domain, with an RMSE (Root Mean Square Error) of 0.0005, MAE (Mean Absolute Error) of 0.00004, and an R^2 (coefficient of determination) of 0.93. It then identifies change points indicative of anomalies. Finally, the goal is to predict the remaining useful life (RUL) of the machinery, where the best result was achieved using the hybrid DL model LSTM-CNN (Long Short-Term Memory - Convolutional Neural Networks), with an RMSE of 0.001 and an MAE of 0.0007.

By implementing these techniques, the thesis aims to demonstrate the efficacy of ML and sensor data in establishing proactive maintenance strategies within the industry 4.0 framework, leading to significant cost savings and enhanced operational efficiency.

Key words: Predictive maintenance (PdM), industry 4.0, machine learning (ML), deep learning (DL), vibration data, time-series, sensors, next-state-prediction, remaining useful life (RUL), anomaly detection.

Résumé

Ce mémoire explore la relation entre l'apprentissage automatique et les données des capteurs pour atteindre la maintenance prédictive dans l'industrie 4.0. L'accent est mis sur les méthodes d'apprentissage profond pour développer un processus de maintenance prédictive complet en utilisant des données de séries temporelles provenant de capteurs de vibration. Le jeu de données des roulements du Centre pour les Systèmes de Maintenance Intelligents (IMS) sert de fondation à cette exploration. Le processus progresse de la prédiction du prochain état opérationnel de la machine, où le meilleur résultat a été obtenu par un modèle LSTM (Long Short-Term Memory) dans le domaine fréquentiel, avec une RMSE (erreur quadratique moyenne) de 0.0005, une MAE (erreur absolue moyenne) de 0.00004 et un R^2 (coefficient de détermination) de 0.93. Il identifie ensuite les points de changement indicatifs d'anomalies. Enfin, l'objectif est de prédire la durée de vie restante de la machine, où le meilleur résultat a été obtenu en utilisant le modèle hybride d'apprentissage profond qui est LSTM-CNN (Long Short-Term Memory - Convolutional Neural Networks), avec une RMSE de 0.001 et une MAE de 0.0007.

En mettant en œuvre ces techniques, le mémoire vise à démontrer l'efficacité de l'apprentissage automatique et des données des capteurs dans l'établissement de stratégies de maintenance proactive dans le cadre de l'industrie 4.0, conduisant à des économies de coûts significatives et à une efficacité opérationnelle accrue.

Mots-clés : Maintenance prédictive, Industrie 4.0, apprentissage automatique, apprentissage profond, données de vibration, séries temporelles, capteurs, prédiction d'état suivant, durée de vie restante, détection d'anomalie.

Acknowledgements

First and foremost, we would like to express our deepest gratitude to Allah, the Almighty and the most Merciful, for granting us the strength, health, and perseverance to complete this work.

Secondly, we extend our sincere thanks to all those who have contributed, in one way or another, to the realization of this thesis.

In particular, we would like to warmly thank our supervisor, Dr. M. Daoudi, for her availability, her expert advice, and her invaluable guidance throughout our journey. Her encouragement and trust in us have been a great source of motivation.

We would also like to thank phd student A. Mankouri for her invaluable help and advice. Her contribution has been of great importance to the progress of our work.

We are grateful to all those who have supported and accompanied us throughout this journey especially our school ESTIN.

Dedication

To my dear mother, for your unconditional love, constant support, and sacrifices, thank you for being my source of inspiration and strength, for being strong in weakness, and for encouraging me to always strive for my best.

In memory of my father, who has left us, you remain forever in our hearts and thoughts. Your spirit continues to guide our steps every day.

To my two sisters, for your complicity, affection, and wisdom. You are much more than just sisters, you are pillars in my life, unwavering sources of support and constant comfort.

To my brother, for your friendship, which brightens even the darkest days. Your presence in my life is a constant source of joy and strength, and I am endlessly grateful for the bond we share.

Tinhinane OURARI

To Mom & Dad, my unwavering sources of inspiration. Thank you for the countless sacrifices you've made to give me the opportunity to learn and grow. Your strength and love have guided me through every challenge.

To my siblings, my built-in cheerleaders. Thank you, Sister, for always supporting my decisions, no matter what. Thanks to my eldest brother for being my right hand, and to my youngest siblings for filling my days with laughter and joy.

To my friends, thank you for sharing life's journey with me. Your companionship and support have enriched my life in countless ways.

Finally, to myself, thank you for not giving up. Thank you for working hard and persevering. I am proud of the person you have become.

Rezkia BELAYALI

Table of Contents

List of figures

List of tables

Acronyms

AI: Artificial Intelligence

ANN: Artificial Neural Network

ARIMA: autoregressive integrated moving average model

CNN: Convolutional Neural Network

CPS: Cyber-Physical Systems

db: Daubechies wavelet

DCNN: deep convolutional neural network

DL: Deep Learning

FFT: Fast Fourier Transform

FM: Facilities Management

IMS: Center for Intelligent Maintenance Systems

IIoT: Industrial Internet of Things

IoT: Internet of Things

IT: Information Technology

LSTM: Long Short-Term Memory

MAE: mean absolute error

Max: Maximum

Min: Minimum

ML: Machine learning

PCA: Principal Component Analysis

PdM: predictive maintenance

Pseudo-algo: Pseudo-algorithm

ReLu: Rectified Linear Unit

RMS: Root mean square value

RMSE: root mean squared error

RNN: recurrent neural network architecture

rpm: rotations per minute

RUL: Remaining useful life

SOA: State of the art

SOM: Self-Organizing Maps

Std: Standard deviation

SVM: Support Vector Machine

General Introduction

Industrial plants are critically dependent on the seamless operation of their machinery, with rotating components and their bearings playing a pivotal role (Upadhyay & Kumaraswamidhas, 2018). These vital elements are subjected to continuous stress, wear, and challenging environmental conditions, making them prone to failures. Unexpected bearing failures are a major vulnerability in industrial operations. These crucial components, which account for 40-50% of rotating machinery malfunctions, can disrupt production schedules, lead to substantial repair expenses, and even compromise safety according to **Fact.MR**[1].

Traditionally, maintenance practices have relied on reactive strategies, addressing equipment issues only after failures occur. This approach leads to costly downtime, lost productivity, and potential safety hazards. PdM offers a paradigm shift, aiming to proactively identify and address potential problems before they escalate into catastrophic failures.

Despite their critical role in industrial machinery, less than 30% of bearings reach their full lifespan according to **SKF**[2] **& BDS**[3]. Premature failures, often caused by improper lubrication, contamination, or misalignment, demand immediate attention and inflict significant financial losses. **IBT**[4] estimates these costs can reach a staggering $25,000 to $50,000 per hour due to lost productivity and downtime. Furthermore, traditional reactive maintenance strategies exacerbate these problems. Thus, the critical question becomes: How can we develop effective predictive maintenance systems that anticipate and prevent bearing failures, ultimately enhancing operational efficiency and safety?

This work aims to develop a robust and reliable predictive maintenance system for bearings in industrial settings. The focus will be on the following objectives:

[1] **Fact.MR** : is a fast-growing market research firm specializing in providing in-depth industry reports, data analysis, and strategic advisory services to help businesses make informed decisions across various markets and verticals.

[2] **SKF :** Svenska Kullagerfabriken, 'Swedish Ball Bearing Factory' is a Swedish bearing and seal manufacturing company.

[3] **BDS :** Bearing & Drive Systems : As one of the largest global bearing suppliers to distributors that provides bearing solutions.

[4] **IBT :** is a full-service industrial distributor delivering exceptional service, products, training & consulting to a wide array of industries.

1. **Sensor and Data Selection:** Develop efficient strategies to use vibration data for continuous and reliable monitoring.
2. **Feature Engineering and Anomaly Detection:** Extract meaningful features from the sensor data that represent the health state of the bearings. Develop ML algorithms capable of identifying anomalies and deviations indicative of potential bearing failures.
3. **Remaining useful life Estimation:** Develop a framework to estimate the RUL of bearings.
4. **Evaluation:** Optimize ML models to accurately predict bearing failures and estimate RUL. This involves assessing the performance of these models through various experiments to confirm their generalization capability.

By addressing these objectives and exploring additional considerations, this research has the potential to significantly enhance the efficiency and reliability of industrial operations. The successful conceptualization of a sensor-based PdM system for bearings will not only minimize downtime and costs but also improve overall plant safety and productivity.

The structure of the report was as follows: Chapter 01, titled Background and State of the Art, explores the evolution of PdM alongside the industrial revolution. It categorizes PdM models and examines their integration within Industry 4.0. Related works in the field is reviewed, summarizing key insights for subsequent chapters. Chapter 02, titled Techniques, Tools, and Proposed Solution, summarizes the techniques used, focusing on vibration analysis, feature extraction in time and frequency domains, and denoising methods (Wavelet Transform and Daubechies Wavelet). It outlines our proposed solution for next state prediction, anomaly detection, and RUL prediction using various models such as RNN, LSTM, CNN, and hybrid models like CNN-RNN, RNN-LSTM, and CNN-LSTM. The chapter details evaluation metrics and tools used to enhance PdM effectiveness. Chapter 03, titled Experimental Study, details the experimental study that led to the proposal of our solution. It describes the data used, preprocessing steps, and feature extraction techniques applied in time and frequency domains. Results from experiments on next state prediction and RUL using different models were analyzed. Discussions highlight insights gained, challenges encountered, and recommendations for future research in PdM.

Chapter 01: Background and State of the art

In this initial chapter, we will explore the concept of Maintenance 4.0 within the broader framework of industry 4.0. This examination will encompass an overview of fundamental concepts and the advanced tools used in this field, including Cyber-Physical Systems (CPS), the Internet of Things (IoT), cloud computing, Big Data and machine learning. We will delve into how these technologies converge to enhance PdM strategies and overall operational efficiency. Additionally, this chapter will review related works in the domain, with a particular emphasis on data-driven approaches, both within single models and multimodel frameworks.

I. Background :

In this section of the chapter, we will discuss the primary concepts relevant to industry and maintenance.

I.1. Industrial Revolution

The industrial revolution was a period of significant economic and social change that began in 18th century Britain, transforming the world from an agrarian, handicraft economy to one dominated by industry (Poór et al., 2019).

This journey unfolded across four eras: **Industry 1.0**, the Age of Steam (1760s), revolutionized production with steam power, leading to innovations like steamships and locomotives. **Industry 2.0**, the Rise of Electricity (1870s), introduced mass production with assembly lines powered by electricity, exemplified by Henry Ford's automobile manufacturing. **Industry 3.0**, the Digital Revolution (1970s), brought partial automation through computers, paving the way for fully automated lines. Finally, **Industry 4.0** (2010s) integrates technologies like the Internet of Things and Artificial Intelligence to create smart factories, merging the physical and digital worlds (Poór et al., 2019). This era of CPS emphasizes interconnectedness, leading to more flexible and efficient operations with real-time data analysis and PdM strategies.

I.2. Industrial maintenance:

A combination of all technical, administrative, and managerial actions throughout the life cycle of an item, aimed at maintaining or restoring it to a condition where it can perform its intended function (Bengtsson, 2004). Its types:

I.2.1. Reactive maintenance:

Reactive maintenance, also known as corrective maintenance, is defined by the European standard NF EN 13306 X 60-319 as maintenance performed after detecting a breakdown to restore functionality. This strategy involves delaying repairs until equipment malfunctions, which can be effective when downtime doesn't impact production or costs are low. Reactive maintenance includes curative maintenance, which restores functionality after a breakdown, and palliative maintenance, which prolongs equipment life by addressing wear and tear without resolving underlying issues (Bengtsson, 2004).

I.2.2. Preventive maintenance:

Preventive maintenance, as defined by AFNOR (FD X 60-000), involves scheduled inspections, servicing, and repairs at predetermined intervals to reduce the probability of failures and extend asset lifespan. It aims to prevent unexpected breakdowns and maintain operational efficiency. This maintenance strategy includes systematic maintenance, which is conducted at set intervals based on usage hours or schedules without prior inspection, ensuring proactive repairs but risking premature or delayed maintenance. Conditional maintenance, on the other hand, is based on monitoring equipment operation and parameters, allowing timely intervention when deterioration is detected, thus preventing breakdowns and maintaining equipment condition (Bengtsson, 2004).

I.2.3. Predictive maintenance:

PdM, defined by the European standard (NF EN 13306 X 60-319), involves using extrapolated forecasts from the analysis and evaluation of significant parameters of an asset's degradation. This proactive approach leverages real-time data and condition monitoring through sensors and analytics to predict potential failures before they occur. Unlike traditional time-based schedules or reactive maintenance, PdM identifies early signs of wear and schedules maintenance only when necessary, shifting from a "fix-it-when-it-breaks" mindset to a "prevent-

the-break" approach. This leads to significant improvements in plant performance, cost-effectiveness, and safety (Bengtsson, 2004).

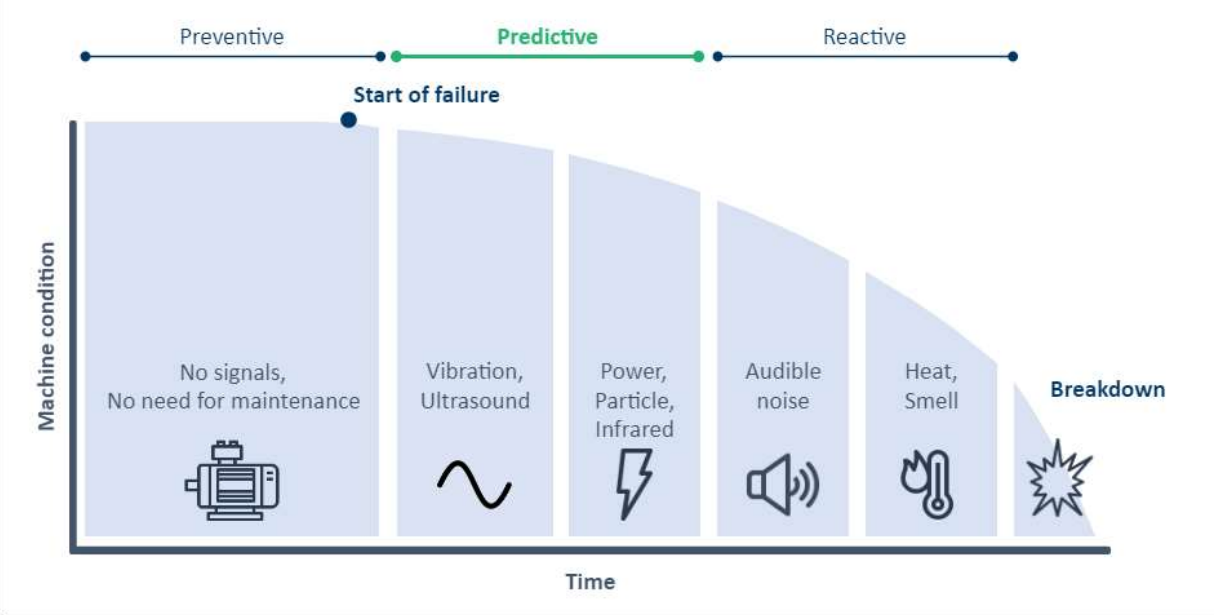


Figure 1: Machine breakdown process (("What Is Predictive Maintenance?," n.d.))

I.3. Types of models in Predictive Maintenance:

Understanding the diverse approaches used in PdM is crucial, as it encompasses various methodologies such as knowledge-based, data-driven, and physics-based models. Each of these approaches refers to an individual technique used to analyze data and predict equipment failures or degradation, offering unique insights and methodologies to facilitate timely maintenance interventions.

I.3.1. Knowledge-based models :

Rely on historical data and expert knowledge to facilitate PdM. They encompass three submodels: rule-based models, case-based models, and fuzzy knowledge-based models. Rule-based models use IF-THEN rules for diagnostic reasoning, while case-based models exploit past cases to solve new problems. Fuzzy knowledge-based models use fuzzy logic to deal with uncertainty in identifying faults, offering an understanding similar to how humans perceive varying levels of truth (Jimenez et al., 2020).

I.3.2. Data-driven models

Harness the available operational data to extract valuable insights for maintenance purposes. They are categorized into statistical models, stochastic models, and machine learning models. Statistical models employ techniques like regression analysis and ARMA models to analyze sensor data for health assessment and Remaining Useful Life (RUL) estimation. Stochastic models use processes like Gaussian processes and Markov chains to model degradation phenomena and forecast future values. ML models, including SVMs and Neural Networks, excel at fault diagnostics and RUL estimation through supervised learning, while unsupervised techniques like self-organizing maps (SOMs) offer additional insights by visualizing hidden relationships within complex data (Jimenez et al., 2020).

I.3.3. Physics-based models

Use physics laws and finite element methods to model degradation phenomena, providing a fundamental understanding of the underlying physical processes. Examples include modeling crack growth and fatigue in various components such as rotor cages, robot health degradation, and lithium-ion batteries. These models offer insights into the mechanisms driving degradation, allowing for more accurate predictions and effective maintenance strategies (Jimenez et al., 2020).

I.3.4. Multi model:

In multi-model approaches, combinations of knowledge-based, data-driven, and physics-based models are employed to overcome individual limitations and enhance PdM capabilities. Examples include integrating fuzzy logic and Markov chains for aero-engine prognostics, using SVM analysis on a health index generated by a physics-based model, and combining a stochastic process with data analysis methods for estimating RUL. These approaches leverage the strengths of each model type, offering comprehensive insights into equipment health and facilitating proactive maintenance actions (Jimenez et al., 2020).

I.4. Predictive maintenance within industry 4.0:

To better explain what makes up Industry 4.0, we need to understand the technologies that make it possible.

I.4.1. Cyber-Physical System:

A CPS is a system that integrates computational algorithms and physical components to monitor and control physical processes. CPS technology connects the physical world and the digital world by enabling real-time communication, data processing, and decision-making. These systems typically consist of interconnected sensors, actuators, and computing devices that work together to monitor, analyze, and respond to changes in the physical environment. CPS plays an essential role in various domains, including Industry 4.0, smart cities, healthcare, transportation, and more, by enabling intelligent and autonomous operations (Meesublak & Klinsukont, 2020).

I.4.2. Internet of things:

The Internet of Things (IoT) refers to a network of interconnected physical devices, vehicles, appliances, and other objects embedded with sensors, software, and connectivity capabilities that enable them to collect and exchange data over the internet. These devices can communicate with each other and with centralized systems to share information, monitor conditions, and perform automated tasks (Meesublak & Klinsukont, 2020).

I.4.3. Industrial Internet of Things:

The Industrial Internet of Things (IIoT) enhances manufacturing processes by integrating sensors into every component, enabling extensive data collection throughout the manufacturing process and product lifecycle. These sensors, along with wireless sensor networks, contribute to the generation of Big Data in smart manufacturing systems. The scale and complexity of this data require advanced computing technologies, with AI playing a crucial role. Unlike traditional programming, AI enables machines to interpret data, learn from it, and adapt. This integration of AI can revolutionize manufacturing processes through applications like predictive quality analytics, automation, and insightful system identification, leading to enhanced efficiency, optimized operations, and increased innovation in the industrial sector (Nguyen et al., 2019).

I.4.4. Big Data:

Big data refers to massive, complex datasets with three key characteristics: volume (huge amounts of data), velocity (rapidly generated), variety (structured, unstructured, and semi-structured formats), Veracity(accuracy, truthfulness, and trustworthiness of data.) and Volatility (the rate of change and lifespan of data)(Nguyen et al., 2019). In manufacturing, big data analytics unlocks the hidden value within this data. This process involves collecting, storing (often in the cloud), cleaning, analyzing, and visualizing the data.

I.4.5. Cloud computing:

Cloud computing offers on-demand availability of computer system resources, particularly data storage and computing power, without direct active management by the user. This technology provides computational and storage facilities by sharing IT infrastructure over the internet, which can be scaled according to dynamic requirements. Generally used to describe data centers accessible to many users online, cloud computing enables companies to access necessary services and resources for Industry 4.0 applications, minimizing up-front IT infrastructure costs. Additionally, it allows enterprises to deploy applications more quickly, with improved manageability and reduced maintenance, enabling IT teams to rapidly adjust resources to meet fluctuating and unpredictable demand (Kim, 2017).

I.4.6. Machine learning:

ML is the field focused on creating algorithms and techniques that allow computers to gain knowledge and enhance their performance by analyzing data. It involves developing systems that can learn and improve from experience (Géron, 2022). They can be categorized based on the level and nature of supervision they receive during the training process into 4 subtypes:

I.4.6.1. Supervised Learning:

Supervised learning involves algorithms that learn patterns from labeled data, where each input is paired with a desired output. The dataset is divided into training and testing subsets, with the training set used to train the model and the test set used to evaluate its performance (Géron, 2022). Common supervised learning algorithms include:

- **Decision Tree:** Visual representation of decisions and their possible consequences, using nodes and branches.

- **SVM:** Used for classification and regression tasks, it can handle both linear and non-linear data through the kernel trick.

I.4.6.2. Unsupervised Learning:

Unsupervised learning algorithms explore data to reveal hidden patterns and structures without predefined labels. They focus on tasks like clustering and feature reduction (Géron, 2022). Common techniques include:

- **Clustering:** Organizes unlabeled data into groups based on similarities, using methods like exclusive, overlapping, hierarchical, and probabilistic clustering.
- **Dimensionality Reduction:** Reduces the number of features in a dataset to simplify analysis, using methods like Principal Component Analysis (PCA) and Autoencoders.

I.4.6.3. Semi-Supervised Learning (SSL):

SSL combines a small amount of labeled data with a large amount of unlabeled data to train models. This approach bridges the gap between supervised and unsupervised learning (Géron, 2022). Types of SSL include:

- **Semi-Supervised Classification:** Classifies data using both labeled and unlabeled data.
- **Semi-Supervised Clustering:** Uses labeled data and constraints to guide the clustering process.

I.4.6.4. Reinforcement Learning (RL):

RL involves an agent interacting with its environment through trial and error to maximize long-term rewards. The agent observes the environment, takes actions, receives rewards or penalties, and learns a policy to determine the best actions to take in various situations (Géron, 2022).

II. Related works:

In this section, we will present several examples from the literature on PdM with a special focus on the data driven methods.

The state-of-the-art in PdM is characterized by a growing trend towards multi-model approaches that integrate diverse techniques to handle the complexities of real-world industrial data. Early research demonstrated the effectiveness of single models, such as ANNs and SVMs, in tasks like predicting tram track widening or detecting anomalies in cyber-physical systems. For example,

ANN and support vector regression (SVR) were used to predict tram track widening with high accuracy for both straight and curved tracks (Amruthnath & Gupta, 2018). Another study effectively used SOM for anomaly detection and localization, leveraging quantization errors to detect and pinpoint degradation in bearings before complete failure (Von Birgelen et al., 2018). Similarly, LSTM models achieved high accuracy in predicting failures in NASA's turbofan engines by handling sequential sensor data (Namuduri et al., 2020). However, when dealing with smaller datasets, simpler traditional machine learning models like K-Nearest Neighbors (KNN) or Decision Trees (DT) often outperformed complex deep learning models such as LSTM and Temporal Convolutional Networks (TCN), particularly in applications like hydraulic test rigs (Silvestrin et al., 2019).

As the complexity and volume of industrial data grew, researchers turned to multi-model strategies. These approaches combine the strengths of different techniques: physics-based models provide deep system understanding, statistical methods extract key data features, and ML algorithms excel at pattern recognition and prediction. For instance, ARIMA models combined with deep neural networks (DNN) were used for trend analysis and failure prediction in slitting machines, demonstrating high accuracy (Kanawaday & Sane, 2017). In railway transportation, ARIMA was combined with PCA and Random Forest (RF) to predict failures (Francis & Mohan, 2019). For diagnosing and predicting issues in complex vending machines, a labeling framework integrated with machine learning models achieved high precision, recall, and F-measure scores (Xiang et al., 2018). Anomaly detection combined with SMOTE (KNN) and SVM effectively detected failures within the printed circuit board (PCB) production line (Rousopoulou et al., 2019). In semiconductor manufacturing, handling imbalanced data through a multi-classifier approach saw SVM outperforming KNN (Susto et al., 2014). Hybrid models also excelled in identifying faults in vibration data using feature extraction, PCA, and clustering methods, with domain experts helping to find optimal cluster numbers (Amruthnath & Gupta, 2018). In PdM within Facility Management systems, data integration and feature selection were combined with SVM and ANN, achieving high accuracy and efficient processing times (Cheng et al., 2020). For classifying tool conditions to integrate old CNC milling machines into Industry 4.0, feature extraction combined with ANN yielded high accuracy, recall, and precision (Hesser & Markert, 2019). Motor failure prediction and RUL estimation used ANN, demonstrating robustness both in real-world applications and cross-validation (Scalabrini Sampaio et al., 2019). Finally, predicting the RUL of bearings involved denoising, feature extraction, and deep convolutional neural networks (DCNN), achieving remarkable results in terms of accuracy and model scores (Ding et al., 2021).

These multi-model approaches not only enhance the accuracy of anomaly detection, fault classification, and RUL prediction but also contribute to more robust, explicable, and adaptable PdM systems. By combining different methodologies, these hybrid models address the intricate challenges of maintaining industrial machinery, paving the way for further advancements in this critical area of Industry 4.0. In the table 01 bellow a summary of the of some application in the literature.

References	case study	Model	Evaluation	Model type
(Von Birgelen et al., 2018)	anomaly detection, localization	SOM	quantization errors	Single model
(Falamarzi et al., 2019)	predict tram track widening (gauge deviation)	ANN & SVR	straight tracks($R^2 > 0.9$) curved tracks($R^2 > 0.9$)	single model
(Namuduri et al., 2020)	classification of Nasa's turbofan engine for PdM	LSTM	acc = 99.3% AUC = 0.998 precision > 87.3%	single model
(Silvestrin et al., 2019)	multi classification on hydraulic test rig	TCN & LSTM	classification error	single model
(Kanawaday & Sane, 2017)	forecast and classify the time series for milling machine	ARIMA+DNN	accuracy = 98.69%	Multimodel
(Francis & Mohan, 2019)	predict failure in railway transportation	ARIMA + PCA+RF		Multimodel
(Xiang et al., 2018)	diagnosing and predicting issues in complex vending machines	labeling framework+ML model	>80% in precision, recall, and F-measure	Multimodel
(Rousopoulou et al., 2019)	failure detection within the printed circuit board (PCB) production line	anomaly detection+Smote(KNN)+SVM	precision = 76% F1-Score = 86% recall = 100%.	Multimodel
(Susto et al., 2014)	predictive maintenance in semiconductor manufacturing	imbalance data handling+Multi classifier	MC SVM outperformed MC KNN	Multimodel

(Amruthnath & Gupta, 2018)	identify faults in vibration data	FE+PCA &T²+clustering methods	domain expert+elbow & nbCluster to find optimal no of clusters	Multimodel
(Cheng et al., 2020)	predictive maintenance within Facilities Management system	Data integration+FS +SVM & ANN	accuracy = 96.547% AAE = 3.427% processing time (0.09 s and 0.05 s)	Multimodel
(Hesser & Markert, 2019)	classify tool conditions (blades) to integrate old CNC milling machines to industry 4.0	FE + ANN	accuracy = 0.9444 recall = 0.9687 precision = 0.9393	Multimodel
(Scalabrini Sampaio et al., 2019)	motor failure prediction + RUL estimation	ANN	RMSE = 0.0313(real world) and 0.0038(cross validation)	Multimodel
(Ding et al., 2021)	predicting RUL of bearings	denoising + FE+DCNN	RMSE = 0.00525 R^2= 0.99762 Score = 0.11116	Multimodel

Table 1: Summary of the SOA

III. Conclusion :

In this chapter, we have covered the essential concepts needed to understand industry 4.0 and industrial maintenance. Our focus has been on explaining the enabling techniques for PdM and giving an overview of the current SOA in this field over recent years. This thorough examination lays the groundwork for comprehending the approach proposed in the subsequent chapter.

Chapter 02: Techniques, Tools, and Proposed Solution

This chapter looks into the techniques and tools used in the experimental study and for the proposed solution. It begins with an overview of the techniques used, providing brief definitions and explications. These techniques include feature extraction methods such as the Fast Fourier Transform (FFT) and wavelet transforms.

The next section is about the tools and libraries used throughout the study such as NumPy for numerical computations, PyWavelets for wavelet transforms, and TensorFlow for building and training DL models. These tools are essential for implementing the techniques and conducting the experiments efficiently.

Finally, the chapter presents the proposed solution, illustrated with schemas and pseudo algorithms. This solution involves a structured approach starting with feature extraction and denoising followed by next state prediction using DL model, change point detection, and ultimately, the prediction of RUL. The proposed solution aims to provide a comprehensive and effective framework for analyzing and interpreting vibration data.

I. Techniques:

I.1. Vibration definition:

Vibrations are mechanical oscillations that involve the movement of an object back and forth around a central point, characterized by frequency, amplitude, phase, and shape (Rittweger, 2020).

- ➔ Frequency refers to the number of oscillations that occur in a specific time period, indicating how rapidly the object vibrates (Rittweger, 2020).
- ➔ Amplitude describes the maximum distance the object moves from its central position during the vibration, reflecting the intensity of the vibration (Rittweger, 2020).
- ➔ Phase signifies the position of the object at a particular moment within its oscillation cycle, providing information about its current state (Rittweger, 2020).
- ➔ Shape represents the form or pattern of the vibration signal, which can vary depending on the specific characteristics of the vibrating object (Rittweger, 2020).

I.2. Time-series data:

A time series is a sequential arrangement of data points collected at regular intervals, where these data points are typically organized chronologically. Time series analysis involves studying these time-dependent data points to uncover patterns, trends, and relationships, aiming to forecast future values based on past observations (Sen & Das, 2023). Despite the emergence of advanced data analysis techniques, time series analysis remains a fundamental statistical tool, often conducted at fixed time intervals like seconds, minutes, or hours (Kulkarni et al., 2022). Key concepts in time series analysis include trend identification, assessing serial dependence, and ensuring stationarity, with tools like moving averages and autocorrelation aiding in pattern recognition and forecasting (Kulkarni et al., 2022).

I.3. Feature Extraction

Various methods exist for analyzing vibration signals, categorized primarily into three domains: time, frequency, and time-frequency (Jain & Bhosle, 2022).

I.3.1. Time domain:

In time domain different analysis technique are used for vibration analysis which involves (Jain & Bhosle, 2022; Tahir et al., 2018):

- **Maximum value**: The maximum value is the largest data point in the set, representing the highest amplitude or intensity reached by the signal. It indicates the peak of the waveform or the maximum level of the measured quantity (Jain & Bhosle, 2022).

$$MAX(X)$$

- **Minimum value:** The minimum value is the smallest data point in the set, representing the lowest amplitude or intensity observed in the signal. It indicates the trough of the waveform or the minimum level of the measured quantity (Jain & Bhosle, 2022).

$$Min(X)$$

- **Mean value** ($\boldsymbol{\mu}$) **:** consists of calculating the average value of the signal (Tahir et al., 2018).

$$\frac{1}{M}\sum_{t=1}^{M} X(t)$$

- **Standard deviation** ($\boldsymbol{\sigma^2}$) **:** it measures the dispersion of data points around the mean, indicating the signal's variability (Tahir et al., 2018).

$$\frac{1}{M}\sum_{t=1}^{M}(X(t)-\mu)^{2}$$

- **Root mean square value** ($\boldsymbol{RMS}$) **:** it determines the square root of the mean of the squared values of the signal, representing its overall magnitude (Tahir et al., 2018).

$$\sqrt{\frac{1}{M}\sum_{t=1}^{M}[X(t)]^2}$$

- **Skewness :** it assesses the asymmetry of the signal's distribution, indicating whether it is predominantly skewed to the left or right (Tahir et al., 2018).

$$\frac{1}{M}\sum_{t=1}^{M}\left(\frac{X(t)-\mu}{\sigma}\right)^3$$

- **Kurtosis :** it quantifies the peakedness or flatness of the signal's distribution, providing insights into its shape (Tahir et al., 2018).

$$\frac{1}{M}\sum_{t=1}^{M}\left(\frac{X(t)-\mu}{\sigma}\right)^4$$

- **Crest factor :** it calculates the ratio of the peak value of the signal to its RMS value, highlighting its peakiness (Tahir et al., 2018).

$$\frac{MAX\ |X(t)|}{RMS}$$

- **Form factor** : refers to a dimensionless parameter that quantifies the shape or "form" of the waveform (Tahir et al., 2018).

$$\frac{RMS}{\frac{1}{M}\sum_{t=1}^{M}|X(t)|}$$

- **peak_to_peak :** it represents the total magnitude of variation from the highest point to the lowest point within a given time interval (*Maximum-to-Minimum Difference - MATLAB Peak2peak*, n.d.).

$$MAX(X) - Min(X)$$

Where:

- M is the number of samples,
- X(t) is the signal’s instantaneous amplitude.

I.3.2. Frequency domain:

One of the most used methods for feature extraction in the frequency domain is the **Fast Fourier Transform (FFT**) which is a mathematical technique used to extract features from data by analyzing its frequency components efficiently. It converts a signal from its original domain (typically time or space) to a representation in the frequency domain, where it can be analyzed in terms of its frequency components. FFT is particularly useful for identifying periodic patterns

or trends in data that may not be easily visible in the time domain. By decomposing a signal into its frequency components, FFT helps in simplifying complex data sets and highlighting important features for further analysis. In feature extraction, FFT can reveal underlying patterns or structures in the data that can be used for various applications such as signal processing, image analysis, and pattern recognition (Hu et al., 2023).

The formula can be presented as (Zhang et al., 2022):

$$X(k) = \sum_{i=0}^{N-1} x(n).e^{-j\frac{2\pi kn}{N}}$$

where :

- $X(k)$ is the frequency domain representation of the signal,
- N is the total number of samples in the signal,
- k represents the frequency bin.

I.4. Denoising

I.4.1. Wavelets Transform

Wavelets Transform is a powerful tool for signal denoising, offering multi-resolution analysis capabilities to separate useful signals from noise. By decomposing signals into different time scales and wavelet spaces, wavelet transform allows for the identification and removal of noise components while preserving signal characteristics (Bnou et al., 2020; Dey, 2023). There exist various different wavelet families such as Morlet, Symlet, Daubechies, Haar, and wavelet packet transform (Dey, 2023; Sonia & Sumathi, 2022).

I.4.2. Daubechies Wavelet

The Daubechies wavelets are a family of orthogonal wavelets defined by Ingrid Daubechies. They are characterized by their compact support and high regularity. The most common Daubechies wavelets are denoted as dbN, where □ indicates the number of vanishing moments (Smith et al., 2007). These wavelets are known for their optimality with respect to scale- and shift-invariance properties, making them particularly effective in various denoising tasks, including ECG signal processing, medical imaging, and motor speed control systems (Kanungo et al., 2020).

I.5. Models

I.5.1. DL models

DL methods have gained significant attention for time series forecasting and RUL prediction (Arslan, 2023; Lin et al., 2023; Varshney & Sharma, 2023). These methods offer advantages such as capturing temporal dependencies, handling complex nonlinear patterns, and automatically processing trends and seasonality in time series data (Kulkarni et al., 2022).

I.5.1.1. ANN

ANNs are inspired by the brain and mimic its learning process. These networks consist of interconnected processing units called artificial neurons. Each connection between neurons has a weight that determines its influence. By adjusting these weights during training, ANNs learn the relationships between inputs and outputs. This allows them to tackle complex, non-linear problems. ANNs are popular due to their ease of use thanks to widely available libraries and their ability to handle new data beyond the training set, making them highly generalizable (Scalabrini Sampaio et al., 2019).

I.5.1.2. CNN

A CNN is a specific class of feed-forward ANN that integrates convolution operations into at least one of its layers. Inspired from the structure of biological neural networks, CNNs merge ANN principles with discrete convolution techniques primarily for image processing tasks, enabling automatic feature extraction. Consequently, CNNs are tailored for recognizing and analyzing two-dimensional data, such as images and videos, with the advantage of directly accepting images as input, thus eliminating the need for intricate feature extraction and data transformation procedures typically found in conventional image recognition algorithms (Gu et al., 2019).

I.5.1.3. RNN

RNNs are a type of neural network particularly suited for processing sequential inputs like speech and language. Unlike traditional feedforward networks, RNNs process input sequences one element at a time while maintaining a "state vector" in their hidden units, which encodes information about the history of past elements in the sequence. This allows RNNs to capture temporal dependencies within the data (Feng et al., 2017).

I.5.1.4. LSTM

LSTM is a type of recurrent neural network architecture that is well-suited for modeling sequential or time series data; they were introduced to overcome the vanishing gradient problem faced by traditional RNNs when training on long sequences (Elmaz et al., 2021; Goodfellow et al., 2016).

I.6.2. Statistical models

I.6.2.1. ARIMA

An autoregressive integrated moving average model (ARIMA) is a form of regression analysis that gauges the strength of one dependent variable relative to other changing variables (Fattah et al., 2018).

- **autoregressive (AR)**: refers to a model that shows a changing variable that regresses on its own lagged, or prior, values.
- **Integrated (I):** represents the differencing of raw observations to allow the time series to become stationary.
- **Moving Average (MA):** incorporates the dependency between an observation and a residual error from a moving average model applied to lagged observations.

A standard notation for ARIMA would be with p, d, and q, where integer values substitute for the parameters to indicate the type of ARIMA model used. The parameters can be defined as (Fattah et al., 2018):

- ***p*** : the number of autoregressive terms.
- ***d*** : is the number of differences.
- ***q*** : is the number of moving averages.

Autoregressive models assume that Y_t is a linear function of the preceding values and is given by equation (Fattah et al., 2018):

$$Y_t = \alpha_1 Y_{t-1} + \varepsilon_t$$

α_1 in this equation is the self regression coefficient, and ε_t is a random component.

I.7. Next state prediction

Next state prediction with DL involves using various techniques such as Neural Networks, Bayesian networks, Markov models, and state predictors (Petzold et al., 2006). Neural Networks can involve single models such as LSTM, ANN, CNN and RNN …etc. In other hand, Hybrid models, combine different prediction methods or configurations in parallel to reduce configuration overhead and improve accuracy by selecting the most appropriate prediction result from a set of base predictors like ANN with LSTM layers can significantly enhance prediction accuracy, although it comes with increased training time and resource demands (Downey et al., 2017; Rojek, 2010; Violos et al., 2020). Other models can be used as CNN-RNN, RNN-LSTM, CNN-LSTM and more (Elmaz et al., 2021) .

I.8. Anomaly detection

Change point detection is a crucial concept in various fields like materials science, engineering, and anomaly detection. It involves identifying significant changes in a system's behavior or characteristics that cannot be explained by random fluctuations (Feasel, 2022; Park & Ding, 2021). Different methods for change point detection include unsupervised algorithms, Bayesian approaches, and optimization techniques with penalties or regularizations to enhance accuracy and reduce false positives. The goal is to pinpoint the exact moment when a change occurs, aiding in segmenting data, identifying operational phases, or detecting anomalies (Glock et al., 2024).

I.9. RUL prediction

RUL prediction refers to the estimation of the remaining operational lifespan of a component or system before it fails or needs maintenance (Z. Wang et al., 2023). It is widely important for PdM to anticipate when machines will fail, aiding in decision-making and maintenance strategies. Various data-driven approaches have been proposed to enhance RUL prediction accuracy (Shi et al., 2023).

For instance, a neural architecture search technique based on a genetic algorithm optimizes Transformers for RUL predictions, outperforming handcrafted DNNs in terms of RMSE (Mo & Iacca, 2023). Additionally, a self-supervised learning method based on a variational autoencoder improves feature extraction from complex operating conditions and fault modes, enhancing

estimation accuracy (Yu et al., 2024). Furthermore, a three-stage feature selection approach for DL-based RUL prediction models enhances stability and achieves significant improvements in RUL prediction accuracy (Y. Wang & Zhao, 2023). Lastly, a hybrid deep architecture incorporating a deep convolutional variational autoencoder with an attention mechanism shows superior performance in predicting machine failures, providing better spatial feature distributions and interpretability (Remadna et al., 2022).

I.10. Evaluation metrics:

Traditional evaluation metrics for forecasting often focus on measuring the accuracy of forecasting methods without considering the potential costs associated with forecasting errors, leading to limitations in selecting the most suitable forecasting method for aggregate production planning (Choi & Ok, 2015). When it comes to evaluating generative models of behavior learned through imitation learning, metrics that consider longer temporal relationships, unpredictable behavior properties, and biases in behavior distribution are proposed to provide a more comprehensive evaluation and comparison framework (Im et al., 2020).

The root mean squared error (RMSE) and mean absolute error (MAE) are commonly used where each one is optimal for specific error distributions: RMSE for normal errors and MAE for Laplacian errors (Hodson, 2022). For a sample of n observations y $(y_i, 1, 2, \ldots, n)$ and n corresponding model prediction $\hat{y}$ the MAE and RMSE are (Hodson, 2022) :

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y})^2}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}|$$

Additionally, the R^2 metric is commonly used in forecasting models to assess the proportion of variance in the outcome variable explained by the predictor variables (Miles, 2005). R^2 being recommended as a standard metric due to its ability to provide a comprehensive assessment of the model's performance without the interpretability limitations of other metrics like MSE, RMSE, MAE, and MAPE (Ding et al., 2021; Hodson, 2022).

$$R^2 = \frac{\sum_{i=1}^{n}(y_i - \hat{y})^2}{\sum_{i=1}^{n}(\bar{y} - \hat{y})^2}$$

- $\bar{y}$ represents the mean.

II. Tools

The tools used in this thesis primarily involve software libraries designed for data management and manipulation. Python, with Jupyter Notebook from the Anaconda distribution, was the core of the data analytics project. Python's optimized libraries for data analysis were crucial not only for developing prediction models but also for efficiently managing data due to their robust internal structures. Below is a list of the main libraries used, along with a brief description of their functions and the operations they facilitated:

- **NumPy:** is a fundamental library for numerical computing in Python. It provides support for arrays, matrices, and a wide range of mathematical functions to operate on these data structures. NumPy is essential for scientific computing, enabling efficient numerical calculations and serving as a foundation for many other data science and ML libraries (*NumPy* -, n.d.).
- **Pandas:** is a powerful data manipulation and analysis library for Python. It provides data structures like DataFrames and Series, which allow for efficient handling, manipulation, and analysis of structured data. It offers functions for data cleaning, transformation, aggregation, and visualization, making it an essential tool for data scientists and analysts (*Pandas - Python Data Analysis Library*, n.d.).
- **Matplotlib:** is a comprehensive plotting library for Python. It provides tools for creating a wide variety of static, animated, and interactive visualizations, such as line plots, bar charts, scatter plots, and histograms. Matplotlib is highly customizable, allowing users to adjust aspects like colors, labels, and axes, making it a crucial library for data visualization in scientific and analytical applications (*Matplotlib — Visualization with Python*, n.d.).
- **Scikit-learn:** is a robust ML library for Python. It provides simple and efficient tools for data mining and data analysis, including algorithms for classification, regression, clustering, and dimensionality reduction. Scikit-learn also offers utilities for model selection, preprocessing, and evaluation, making it a go-to library for building and testing ML models (*Scikit-Learn: Machine Learning in Python — Scikit-Learn 1.5.0 Documentation*, n.d.).
- **TensorFlow:** is an open-source ML framework developed by Google. It provides a comprehensive ecosystem of tools, libraries, and resources for building and deploying ML models. It is particularly known for its flexibility, scalability, and ease of use, making it

widely adopted in both research and industrial applications. It enables developers to create various types of ML models, including deep neural networks, and efficiently train them on large datasets distributed across multiple CPUs or GPUs (*TensorFlow*, n.d.).

- **Keras :** is a high-level neural networks API written in Python, designed to enable fast experimentation with DL models. It acts as an interface for the TensorFlow library, simplifying the process of building, training, and deploying complex neural network architectures. Keras is known for its user-friendly interface, modularity, and extensibility, which make it accessible for both beginners and advanced users in the field of DL (*Keras: Deep Learning for Humans*, n.d.).
- **PyWavelets:** is an open-source Python library for performing discrete wavelet transforms (DWT) and related wavelet-based computations. It provides tools for signal processing, image processing and data analysis, enabling efficient and flexible handling of wavelet transforms and their applications (G. Lee et al., 2019).

III. Proposed Solution

In this section, we present a comprehensive approach to predictive maintenance that integrates several key processes as presented in **figure 2.** This multi-model approach integrates signal processing, ML, statistical analysis, and knowledge-based techniques, providing a thorough PdM solution that enhances equipment reliability and minimizes downtime.

The steps of this process are as follow:

III.1. Data acquisition:

Our solution starts with data collection using accelerometers which are a type of vibration sensors specifically using ICP Piezoelectric accelerometers which are commonly used due to their wide frequency range and sensitivity and low output noise, although they require contact with the vibrating target for operation. They generate an electrical charge in response to mechanical stress. (Ghemari, Z., et al. 2022)

III.2. Preprocessing:

Feature extraction is performed in the frequency domain using FFT due to its ability to efficiently analyze the frequency components of signals. Then feature selection based on the knowledge

domain, followed by denoising to improve signal clarity using the Daubechies wavelet, known for its capability to accurately capture transient features. This process is presented in **Pseudo-algo 1.**

Preprocessing

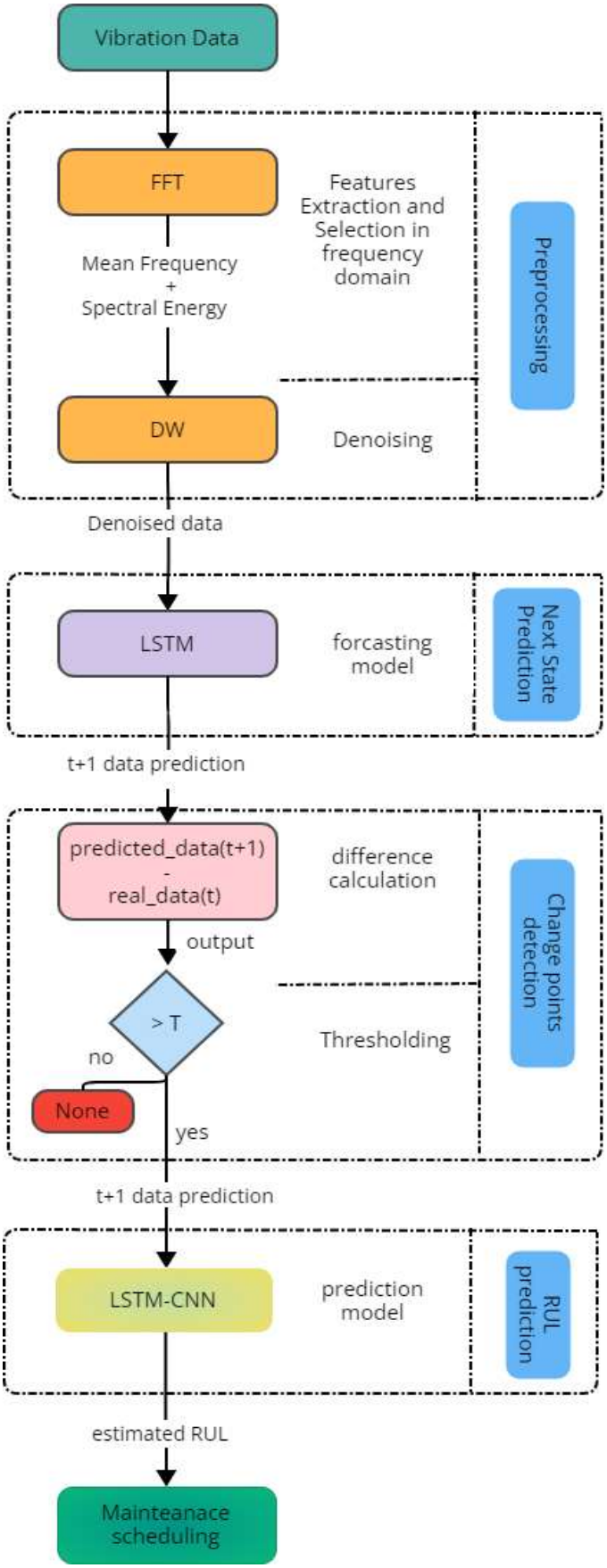
Input : time_series_data
Output : denoised features

```
function extract_features(data):
    fft_result = FFT(data)
    spectral_energy = calculate_spectral_energy(fft_result)
    mean_frequency = calculate_mean_frequency(fft_result)
    return [spectral_energy, mean_frequency]

function denoising_features(spectral_energy, mean_frequency)
   denoised_features = db(spectral_energy, mean_frequency)

   return denoised_features
```

Pseudo-algo 1 : preprocessing



T: Threshold

Figure 2 : proposed solution for predictive maintenance

III.3. Next State prediction:

Now, focusing on predicting the next state of the system using the LSTM model as shown in **Pseudo-algo 2** where in the predict_next_state(time_series_data) function, an LSTM model is initialized with a window size of 5 steps.

For each window of size 5 in the time series, the denoised features after extraction and normalization are stored in a feature vector, which is used as input to predict the next state using the LSTM model to gain time in detecting anomalies without waiting extensively for real-time data, as every second gained is beneficial.

Next State Prediction

```
Input : time_series_data
Output : predicted_next_state

function predict_next_state(features):
    lstm_model = initialize_LSTM(step=5)
    features = []

    for t in range(len(time_series_data) - 5):
        window_data = time_series_data[t:t+5]
        feature_vector = []

        for data in window_data:
            FFT_results = extract_features(time_series_data)
            denoised_data = denoising_features (FFT_results)
             normilized_data = max_min_normalization(denoised_data)

           feature_vector.append(normilized_data)

        features.append(feature_vector)

    predicted_next_state = lstm_model.predict(features)
    return predicted_next_state
```

Pseudo-algo 2: Next State prediction

III.4. Change points detection:

Change points detection and anomaly identification involve statistical analysis to measure discrepancies between predicted and real data as presented in **Pseudo-algo 3** by calculating the difference between the predicted data at time $t+1$ and the real data at time t followed by thresholding informed by empirical studies based on this case using detect_anomalies (predicted_next_state, real_data, threshold). If this difference exceeds a fixed threshold of $17*10^{-6}$. it is considered as a change point indicating an anomaly and is appended to the anomalies list, otherwise, None is appended.

Anomaly Detection

Input: predicted_next_state, real_data, threshold
Output: detected_anomalies

```
function detect_anomalies(predicted_next_state, real_data, threshold):
    anomalies = []

    for t in range(len(predicted_next_state)):
        difference = abs(predicted_next_state[t+1] - real_data[t])

        if difference > threshold:
            anomalies.append(predicted_next_state[t+1])
        else:
            anomalies.append(None)

    return anomalies
```

Pseudo-algo 3: Anomaly Detection

III.5. RUL prediction:

Finally, we estimate the RUL as explained in **Pseudo-algo 4** using detected anomalies and a combined LSTM-CNN model. The predict_RUL(detected_anomalies) function initializes the LSTM-CNN model and iterates through the list of detected anomalies. For each anomaly that is not None, it predicts the RUL using the model and appends the result to the RUL predictions list. If an anomaly is None, it appends None to the list. The function returns the list of predicted RUL values, allowing for effective maintenance scheduling.

RUL Prediction

```
Input: detected_anomalies
Output: predicted_RUL

function predict_RUL(detected_anomalies):
    lstm_cnn_model = initialize_LSTM_CNN()

    RUL_predictions = []
    for anomaly in detected_anomalies:
        if anomaly is not None:
            RUL = lstm_cnn_model.predict(anomaly)
            RUL_predictions.append(RUL)
        else:
            RUL_predictions.append(None)

    return RUL_predictions
```

Pseudo-algo 4 : RUL prediction

IV. Conclusion

This chapter has provided a thorough exploration of the techniques, tools used in the experimental study and the proposed solution. By examining various methodologies, we have laid the groundwork for the subsequent analysis. Additionally, the overview of tools and libraries has equipped us with the necessary resources to implement these techniques effectively. The proposed solution, presented with schemas and pseudo algorithms, offers a structured approach to analyze vibration data and predicting system health. With these foundations in place, the upcoming chapter will delve into the experimentation and analysis phases of the study, with the aim of gaining valuable insights into the behavior of the system under investigation

Chapter 03: Experimental Study

This chapter is about the experimental study that was conducted on a comprehensive exploration of vibration data and its analytical methodologies. It starts with an examination of the vibration data, followed by the application of feature extraction techniques to identify and select the most suitable features for analysis. Once the optimal features are determined, the next state prediction process using both single and hybrid DL models is conducted. Then, the focus is to change point detection to identify significant shifts within the data. Finally, the chapter concludes with predicting the RUL of the system, integrating the insights gained from the previous steps.

I. Data Description

The bearing dataset used in this experiment for contrast and validation improvement is sourced from NASA IMS (J. Lee et al., 2007). The IMS folder contains three testing datasets, each with operational data from monitoring four bearings. The first dataset includes monitoring results for each bearing in both horizontal and vertical directions, while the second and third datasets provide results in a single direction. The bearings are rotated at a constant speed of 2000 rpm by an alternating current motor, with 1-second vibration signal snapshots recorded at specific intervals collected every 10 minutes at a frequency of 20.48 kHz by PCB 353B33 High Sensitivity Quartz ICP accelerometers. Each dataset records the signals from normal operation until the bearing's failure (run-to-fail experiences), with the file name indicating the collection time and each record (row) representing a data point (Ding et al., 2021; Qiu et al., 2006).

This study focuses on the second IMS dataset for training and testing, which includes monitoring information for four bearings. Each bearing is equipped with a single sensor that records vibration data at 20,480 points per collection. This dataset was collected from February 12, 2004, 10:32:39 to February 19, 2004, 06:22:39, and it contains 984 files (Qiu et al., 2006). At the end of this test-to-failure experiment, bearing 1 experienced an outer race failure.

The third dataset is used for generalization testing. This one was collected from March 4, 2004 09:27:46 to April 4, 2004 19:01:57, and contains 6 324 files, where at the end of this experiment, outer race failure occurred in bearing 3 (Qiu et al., 2006).

II. Experimental results and analysis

II.1. Data Exploration and preprocessing

Figures **3 and 4** display the vibration signals over 1 second for each data point and bearing. Where Figure **3**, representing the normal state, the signal shows a consistent pattern. However, in Figure **4**, depicting the abnormal state, significant irregularities and disruptions are evident, indicating disorder and deviation from the normal operating condition. The abscissa in both Figs represents the index of the data point, while the ordinate represents the value of the vibration signal.

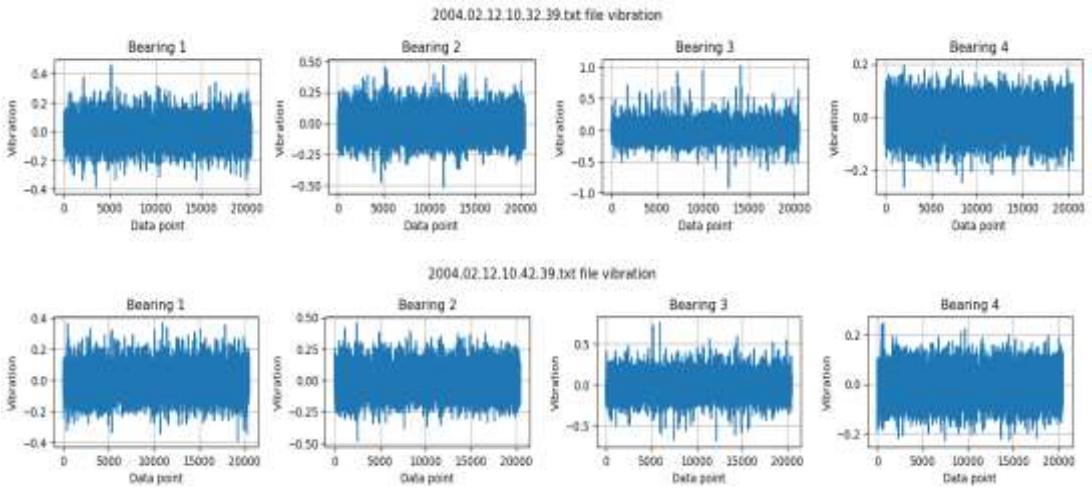


Figure 3 : Vibration Signals in Normal Operating Condition

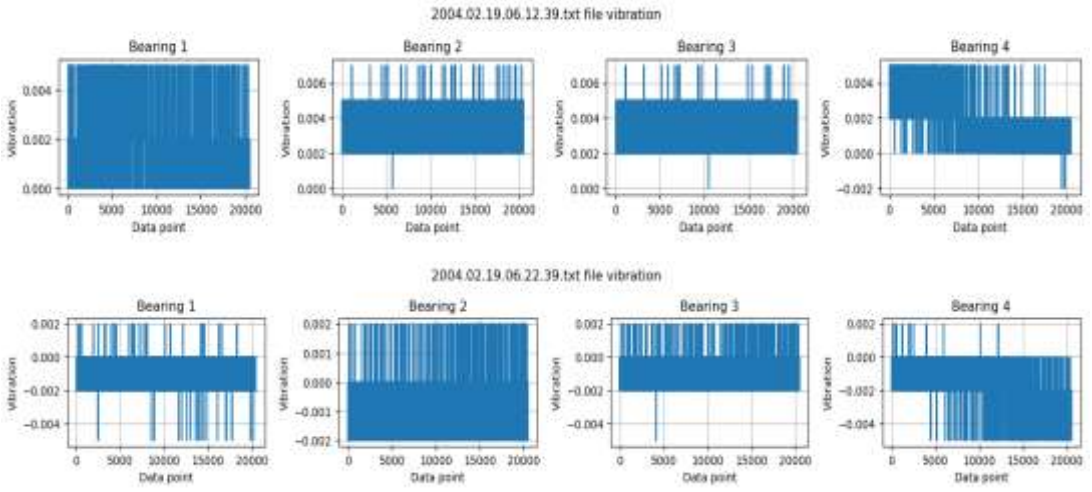


Figure 4 : Vibration Signals in abnormal Operating Condition

II.1.1. Feature Extraction

Previously mentioned, our dataset comprises 984 files, each containing 1-second vibration signals with 20480 data points. To effectively represent this data, we aim to condense each 1-second vibration signal into a single row encapsulating all information from each file. This process involves feature extraction from the vibration data. Two primary methods are employed: feature extraction in the time domain and feature extraction in the frequency domain.

II.1.1.1. Time Domaine:

To conduct feature extraction in the time domain, we, initially, define functions to extract dates from file names and calculate statistical measures from data columns. Upon processing each file within a specified folder, we load the data into a DataFrame and select the appropriate column. Subsequently, we calculate a range of statistical measures, including the maximum value, minimum value, mean value, standard deviation, root mean square value, skewness, kurtosis, crest factor, form factor, and peak-to-peak value. These measures encapsulate essential characteristics of the vibration signals. The processed data is then organized into a structured DataFrame for analysis. As a result of this process, we obtain four datasets, each representing data from one bearing, with 984 rows each and 10 columns. The features plot for each dataset is presented in **Figure 5**.

II.1.1.2. Frequency Domain:

To perform feature extraction in the frequency domain, we start by defining functions to extract dates from file names and calculate spectral measures from data columns. For each file in a specified folder, we load the data and perform an FFT on the relevant column. We then compute spectral features, including maximum frequency, spectral energy, and mean frequency, to capture the key characteristics of the vibration signals. The processed data is organized into a structured DataFrame for analysis. This method produces four datasets, each representing data from one bearing, with 984 rows and three columns containing the calculated spectral features. The feature plots for each dataset are presented **Figure 6**.

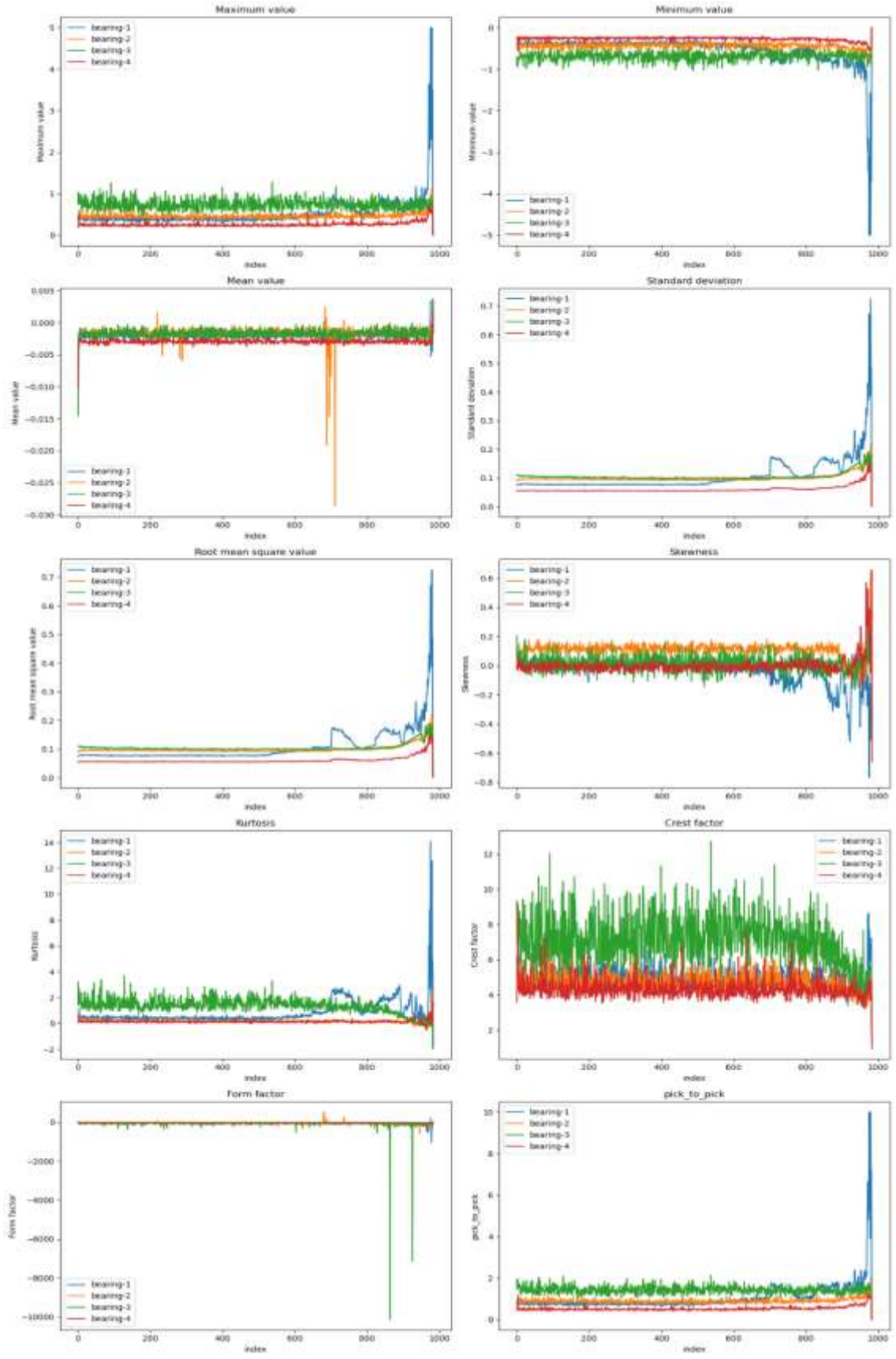


Figure 5: Time domain features

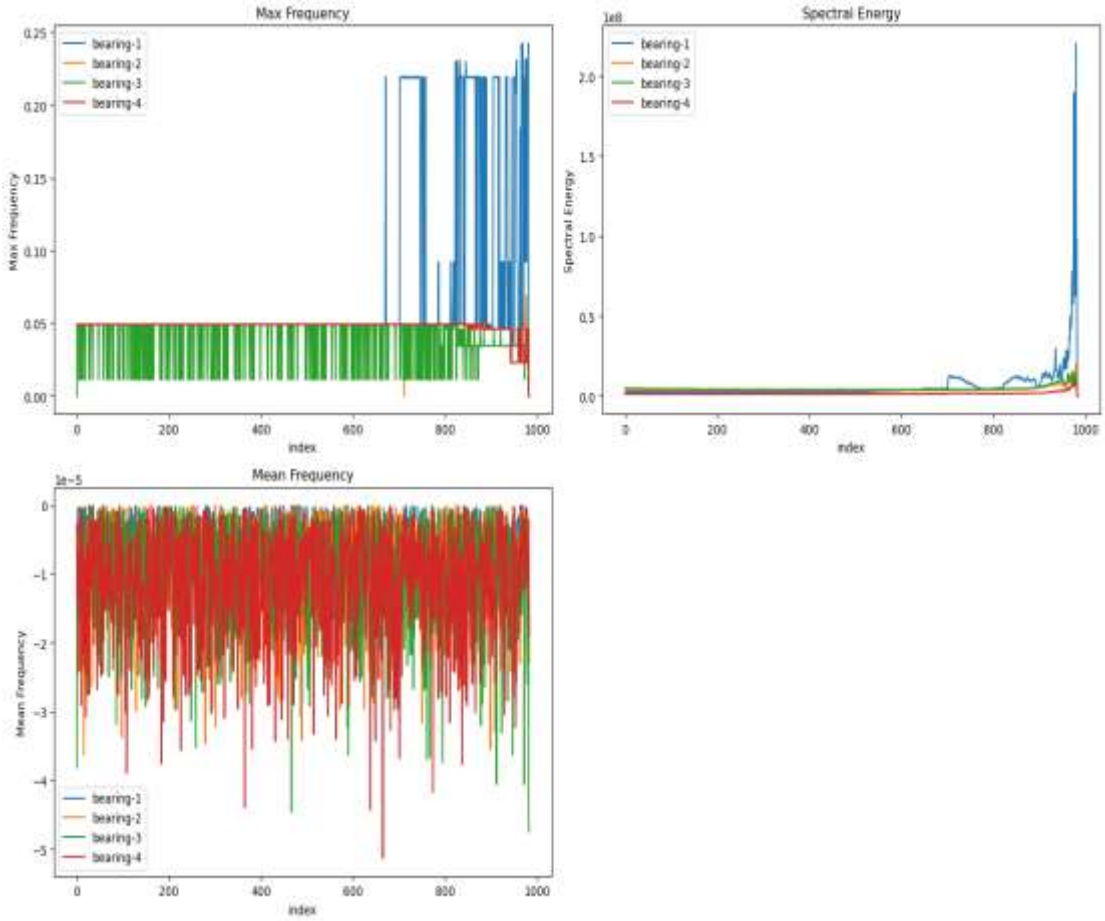


Figure 6: Frequency domain features

II.1.2. Features selection

To ensure a comprehensive and robust approach to the needed analysis, we selected specific features from both the time and frequency domains. These features capture a wide range of signal characteristics and were chosen using domain knowledge to identify relevant features while reducing computational costs and enhancing performances (Rustamov et al., 2023; Z. Wang et al., 2021). By leveraging spectral energy and mean frequency in frequency domain (Rustamov et al., 2023; Zheng, 2017), and by selecting Standard Deviation and Kurtosis in time domain (Wibawa et al., 2022), meaningful information from complex datasets are extracted, leading to more accurate outcomes.

- ➔ **Time Domain Features:** By selecting Standard Deviation and Kurtosis as features, we gain critical insights into the distributional characteristics of the vibration signals, enhancing our ability to detect and characterize faults (Wibawa et al., 2022):

Standard Deviation:

- **Variability Measurement:** In the context of bearing condition monitoring, increased variability often signals changes in the operational state, potentially indicating the onset of faults.
- **Sensitivity to Anomalies:** It is highly sensitive to changes in the signal caused by mechanical defects or wear, making it a reliable indicator of the health state of bearings.

Kurtosis:

- **Sensitivity to Impulses**: Kurtosis is highly sensitive to impulsive events in the signal, making it a valuable feature for detecting the presence of faults that cause sudden, sharp impacts. These impacts are typically not captured effectively by other statistical measures like mean.
- **Fault Severity Assessment**: Higher kurtosis values can indicate more severe fault conditions, providing a means to assess the severity of detected anomalies.

➔ **Frequency Domain Features:** By selecting Spectral Energy and Mean Frequency as features in the frequency domain, we enhance our ability to detect and diagnose faults in bearings by capturing critical aspects of the frequency content of the vibration signals (Rustamov et al., 2023; Zheng, 2017):

Spectral Energy:

- **Intensity and Distribution of Frequencies:** Spectral Energy captures the total energy present in the signal's frequency components. It reflects the intensity of vibrations over a range of frequencies. In bearing monitoring, an increase in spectral energy often correlates with increased vibration activity, which can be indicative of developing faults or mechanical issues.
- **Detection of Anomalies**: Spectral energy is highly sensitive to changes in the vibration signal caused by mechanical faults. A higher spectral energy can indicate abnormal vibrations due to defects such as misalignment, imbalance, or wear. So, it leads to early fault detection.

Mean Frequency:

- **Central Tendency of Frequency Components:** Mean frequency calculates the average frequency of the vibration signal, providing a central tendency of the frequency distribution. Shifts in the mean frequency can indicate changes in the vibration pattern, which are often associated with different types of bearing faults. For instance, an increase in mean frequency might suggest higher rotational speeds or increased friction.
- **Detection of Frequency Shifts**: Mean frequency is sensitive to changes in the overall frequency content of the signal. This makes it a valuable feature for detecting faults that cause frequency shifts, such as imbalance or misalignment. By analyzing the mean frequency, we can assess the severity of the detected faults, as significant deviations from the normal mean frequency may indicate more severe conditions.

II.1.3. Denoising

To eliminate the noise in our data, we applied a wavelet transform. The process involves decomposing a signal into wavelet coefficients, thresholding these coefficients to remove noise, and then reconstructing the signal. The Python library pywt (PyWavelets) was used for performing wavelet transforms. We created a function, that takes a signal and applies the 'db8' (Daubechies wavelet with 8 levels of decomposition). The signal is decomposed into wavelet coefficients at different scales using pywt.wavedec, with the decomposition level specifying the depth of the wavelet decomposition. The noise level (σ) is estimated using the robust median absolute deviation (MAD) method, and the universal threshold is calculated as $\sigma\sqrt{2log(N)}$, where N is the length of the signal. The detail coefficients are thresholded using soft thresholding to remove noise, and pywt.waverec reconstructs the denoised signal from the thresholded coefficients. The function that denoise the signal is applied to each column of the vibration data. The results of before and after denoising of each bearing in the time domain are shown in **Figure 7** and in frequency domain are shown in **Figure 8**.

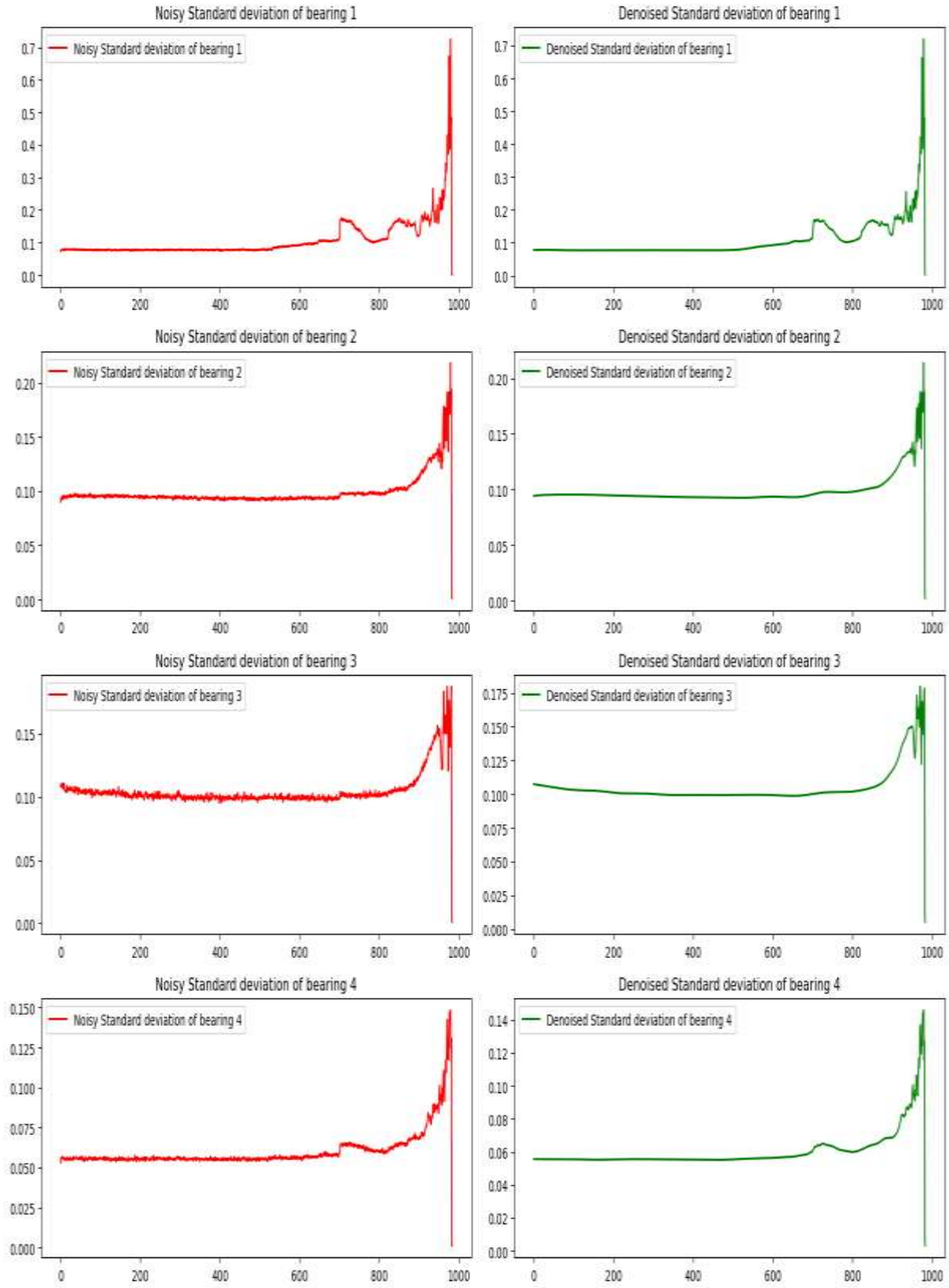


Figure 7: Standard Deviation plot before and after denoising (time domain)

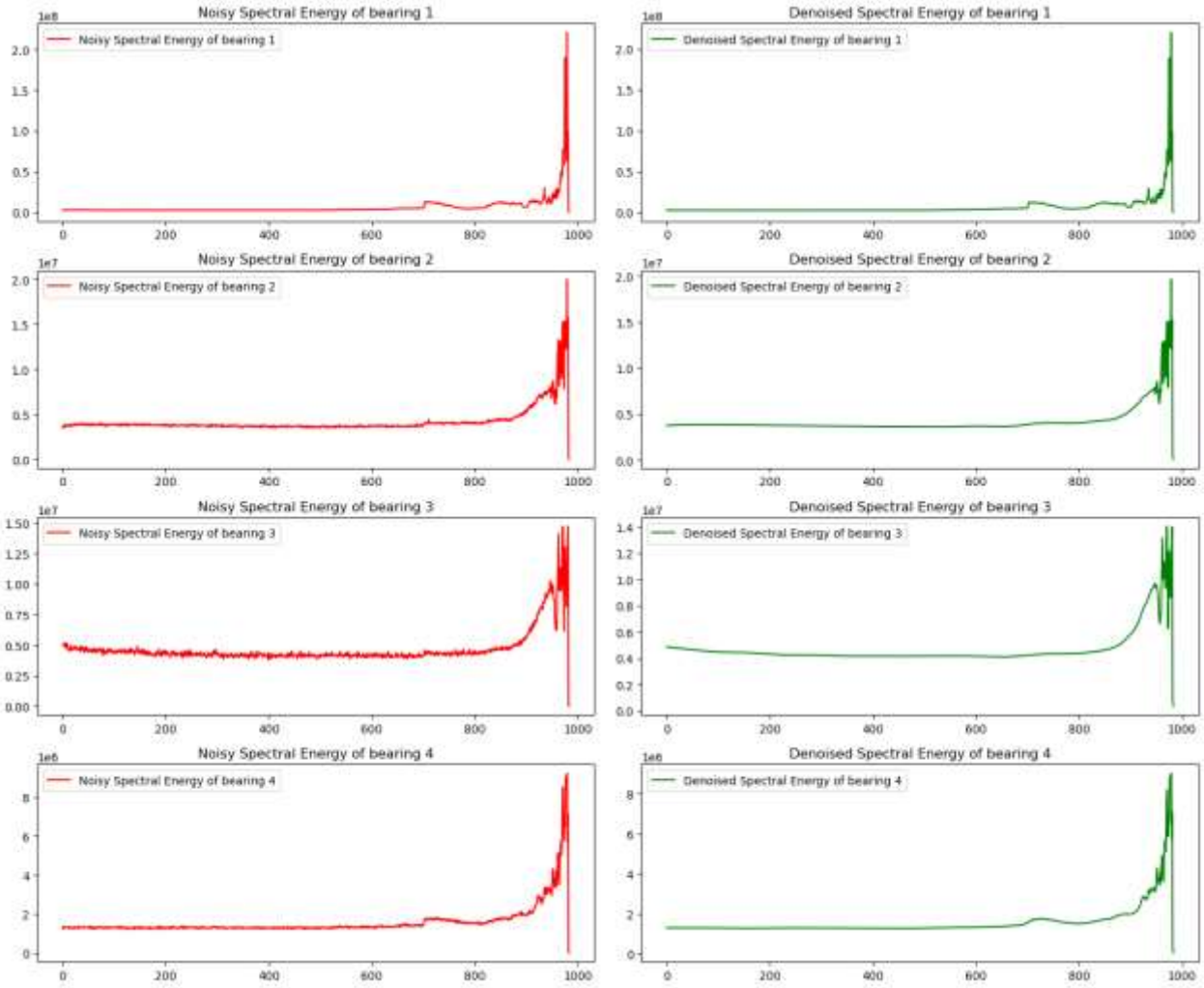


Figure 8: Spectral Energy plot before and after denoising (frequency domain)

II.2. Next State Prediction:

For the next state prediction, the data from bearing 1 in the second test was used for training and for models evaluation, data from the second, third, and fourth bearings from the same test were used. To test the generalization ability of the models, data from the first and third bearings in the third test were used.

Note: For experiment 01 and 02, the graphs representing the prediction of the next state use the x-axis to denote time in minutes where each point represent 10 min, and the y-axis to show the predicted standard deviation in the time domain and spectral energy in the frequency domain. In

these graphs, the blue color represents the actual next state, while the orange color indicates the predicted next state.

II.2.1. Experiment 01:

In this experiment, our objective is to compare the forecasting performance of hybrid DL models with that of single DL models in both frequency and time domains, with noisy and denoised data, to choose the best model in the end where the process of this experiment is presented in **figure 9.** The results for the time domain are presented in **Table 02**, and the results for the frequency domain are in **Table 03**.

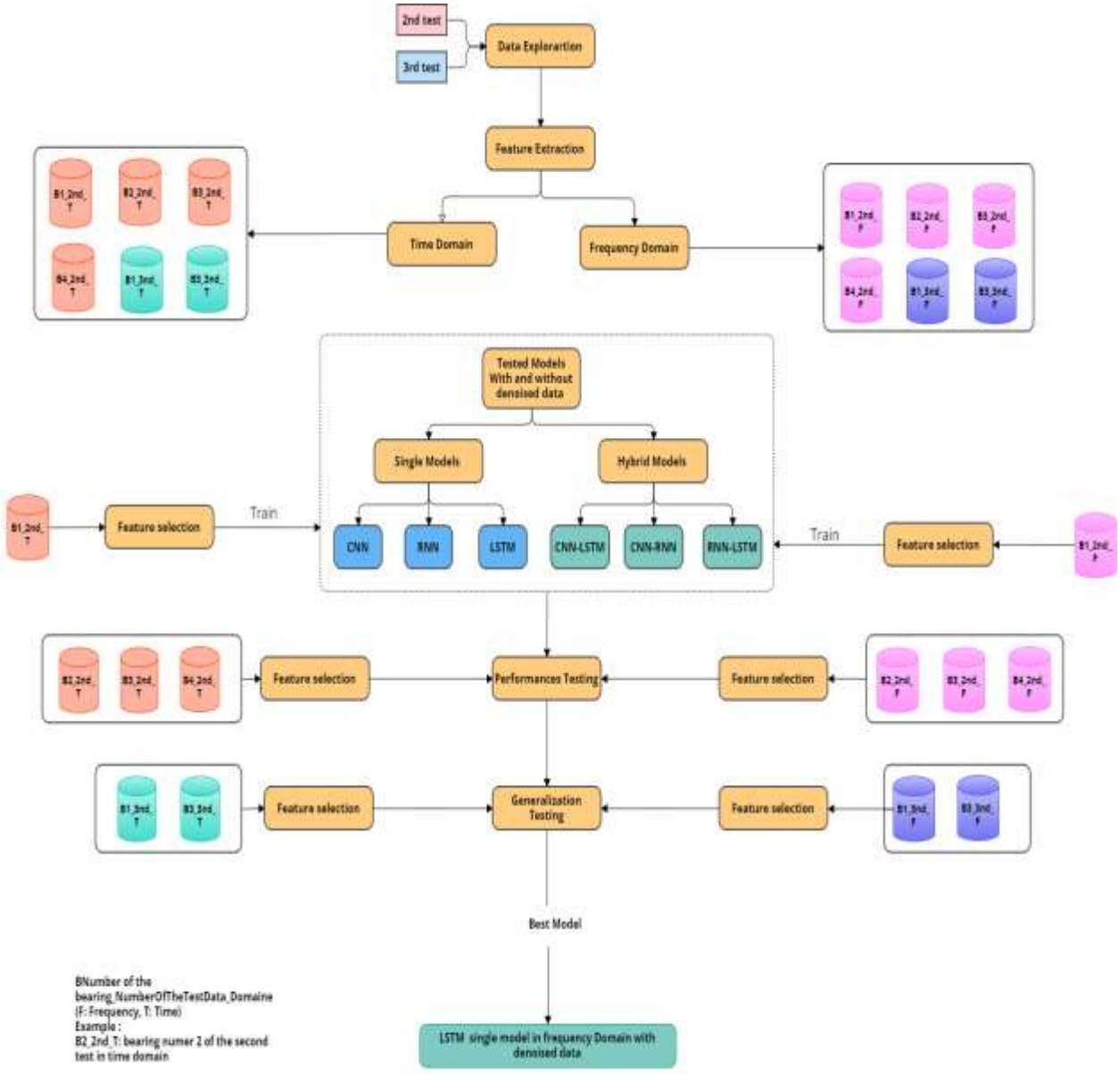


Figure 9: Experiment 01 for next state prediction

			testing data						Generalization data			
			noised data			denoised data			noised data		denoised data	
			B2	B3	B4	B2	B3	B4	B1	B3	B1	B3
Single models	**RNN**	RMSE	0.010	0.010	0.006	0.010	0.009	0.005	0.003	0.013	0.002	0.013
		MAE	0.003	0.003	0.002	0.001	0.001	0.001	0.001	0.002	0.0003	0.0008
		R^2	0.73	0.71	0.84	0.74	0.75	0.87	0.86	0.90	0.91	0.90
	LSTM	RMSE	0.011	0.10	0.008	0.011	0.009	0.007	0.003	0.013	0.002	0.012
		MAE	0.002	0.003	0.004	0.001	0.001	0.002	0.001	0.002	0.001	0.001
		R^2	0.68	0.67	0.72	0.70	0.71	0.79	0.87	0.89	0.90	0.90
	CNN	RMSE	0.012	0.011	0.009	0.012	0.010	0.009	0.012	0.012	0.013	0.011
		MAE	0.002	0.003	0.002	0.001	0.001	0.003	0.003	0.004	0.002	0.004
		R^2	0.61	0.62	0.67	0.61	0.65	0.68	0.59	0.57	0.56	0.60
Hybrid models	**CNN-RNN**	RMSE	0.013	0.013	0.009	0.014	0.012	0.008	0.003	0.01	0.0025	0.01
		MAE	0.02	0.003	0.002	0.002	0.001	0.002	0.001	0.002	0.001	0.001
		R^2	0.54	0.51	0.69	0.51	0.56	0.70	0.84	0.91	0.91	0.92
	RNN-LSTM	RMSE	0.011	0.010	0.007	0.010	0.009	0.006	0.002	0.01	0.003	0.01
		MAE	0.002	0.002	0.003	0.001	0.001	0.002	0.001	0.002	0.001	0.003
		R^2	0.69	0.66	0.78	0.72	0.71	0.82	0.84	0.88	0.86	0.90
	CNN-LSTM	RMSE	0.013	0.012	0.010	0.019	0.011	0.009	0.003	0.014	0.002	0.011
		MAE	0.002	0.003	0.005	0.002	0.002	0.002	0.001	0.002	0.0004	0.001
		R^2	0.55	0.55	0.60	0.58	0.63	0.69	0.87	0.88	0.91	0.92

Table 2: Next State Prediction in time-domain results

			testing data						Generalization data			
			noised data			denoised data			noised data		denoised data	
			B2	B3	B4	B2	B3	B4	B1	B3	B1	B3
Single models	**RNN**	RMSE	0.003	0.003	0.002	0.003	0.002	0.001	0.0007	0.015	0.0006	0.015
		MAE	0.0008	0.0009	0.0005	0.0006	0.0005	0.0002	0.0003	0.0009	0.0001	0.0007
		R^2	0.75	0.75	0.87	0.77	0.82	0.89	0.89	0.68	0.91	0.65
	LSTM	RMSE	0.003	0.003	0.0015	0.003	0.002	0.0014	0.0007	0.014	0.0005	0.014
		MAE	0.0016	0.001	0.001	0.0007	0.0006	0.0002	0.0003	0.0009	0.00004	0.0005
		R^2	0.71	0.70	0.80	0.76	0.79	0.87	0.89	0.73	0.93	0.74
	CNN	RMSE	0.003	0.003	0.003	0.003	0.003	0.004	0.001	0.014	0.0007	0.012
		MAE	0.0008	0.001	0.002	0.0008	0.0006	0.003	0.001	0.003	0.0003	0.001
		R^2	0.71	0.71	0.38	0.72	0.76	-0.28	0.52	0.71	0.90	0.78
Hybrid models	**CNN-RNN**	RMSE	0.003	0.0036	0.003	0.003	0.0032	0.002	0.0008	0.012	0.0007	0.013
		MAE	0.001	0.0012	0.003	0.0005	0.0004	0.002	0.00042	0.001	0.00041	0.001
		R^2	0.68	0.68	0.25	0.71	0.72	0.57	0.88	0.78	0.90	0.77
	RNN-LSTM	RMSE	0.003	0.003	0.001	0.003	0.002	0.001	0.0009	0.013	0.0006	0.01
		MAE	0.001	0.001	0.0005	0.0006	0.0006	0.0003	0.0006	0.001	0.00003	0.0006
		R^2	0.74	0.74	0.87	0.75	0.77	0.89	0.83	0.74	0.92	0.72
	CNN-LSTM	RMSE	0.004	0.004	0.005	0.003	0.003	0.003	0.001	0.013	0.0009	0.013
		MAE	0.002	0.002	0.004	0.0007	0.0005	0.003	0.0007	0.002	0.0006	0.002
		R^2	0.60	0.54	-0.78	0.70	0.72	0.16	0.79	0.75	0.83	0.76

Table 3: Next State Prediction in frequency-domain results

II.2.1.1. single model

The models assessed in this section are:

II.2.1.1.1. RNN

This model was trained using:

- **RNN Layers:** it consists of SimpleRNN with 50 units which allows the network to have a memory of previous inputs, making it suitable for sequence data.
- **Dense Layers:** Dense layer with 2 units that serves as the output layer, producing predictions for the selected features.
- **Optimizer:** Adam is used for efficient training
- **Loss Function:** MSE is chosen as the loss function to minimize the prediction errors
- **Training**: The model is trained over 100 epochs with a batch size of 32, with validation split of 10% allowing the model to learn from small subsets of the data iteratively.

Results :

➔ **Time domain:** We observe that the best results for testing data using RNN are achieved with denoised data for bearing 4, where the metric values are: **RMSE: 0.005, MAE: 0.001, and R^2: 0.87**. In contrast, the best results for generalization data using RNN are achieved with denoised data for bearing 1, with metric values of **RMSE: 0.002, MAE: 0.0003, and R^2: 0.91**. According to these results, we observe that the generalization performance of the model is superior to the test performance, indicating that the model generalizes well. The results are presented in Figures **10, 11, 12 and 13**.

➔ **Frequency domain:** The RNN model demonstrates good performance on the test data, achieving its best results with denoised data, an **RMSE of 0.001, MAE of 0.0002, and R^2 of 0.89**. The generalization test further underscores the model's efficacy with denoised data, yielding an **RMSE of 0.0006, MAE of 0.0001, and R^2 of 0.91.**

The results are presented in Figures **14, 15, 16 and 17**.

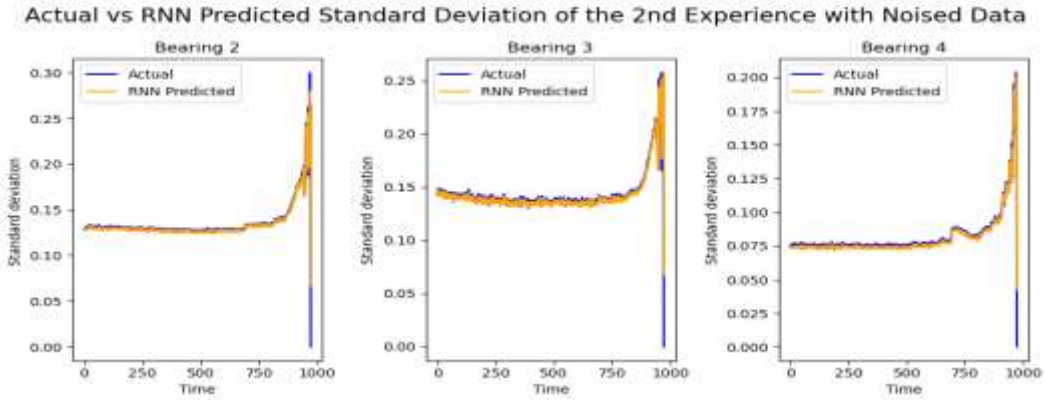


Figure 10: RNN Predictions of Standard Deviation on Test Noised Data (Time domain)

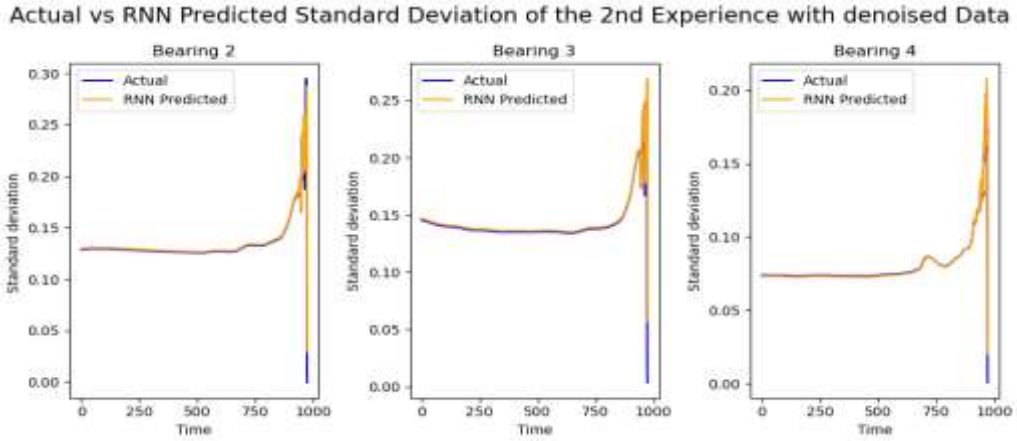


Figure 11: RNN Predictions of Standard Deviation on Test denoised Data

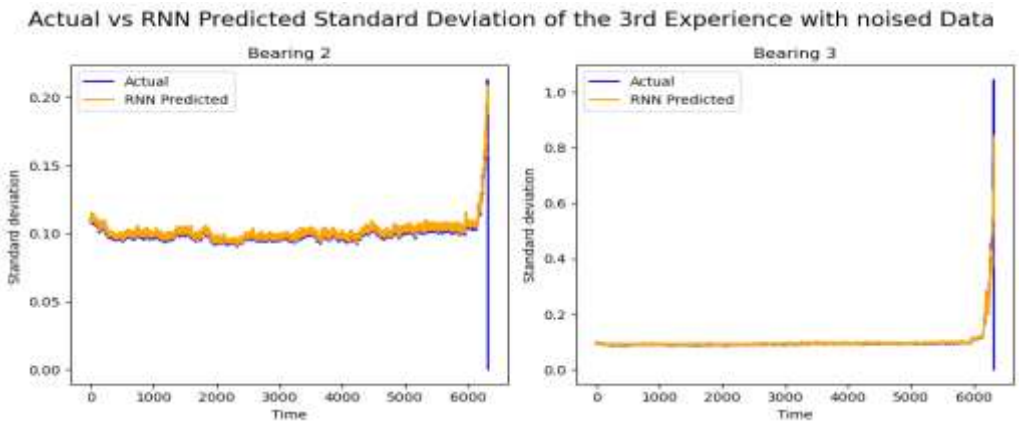


Figure 12: RNN Predictions of Standard Deviation on generalization noised Data

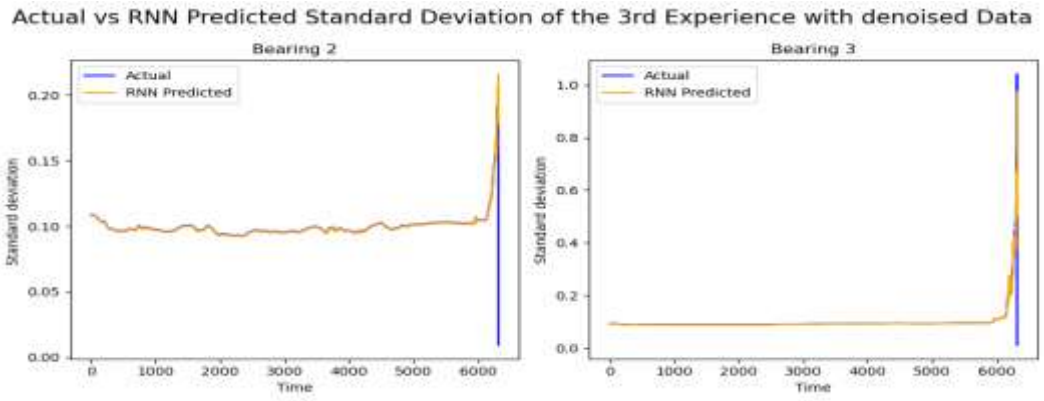


Figure 13: RNN Predictions of Standard Deviation on generalization denoised Data

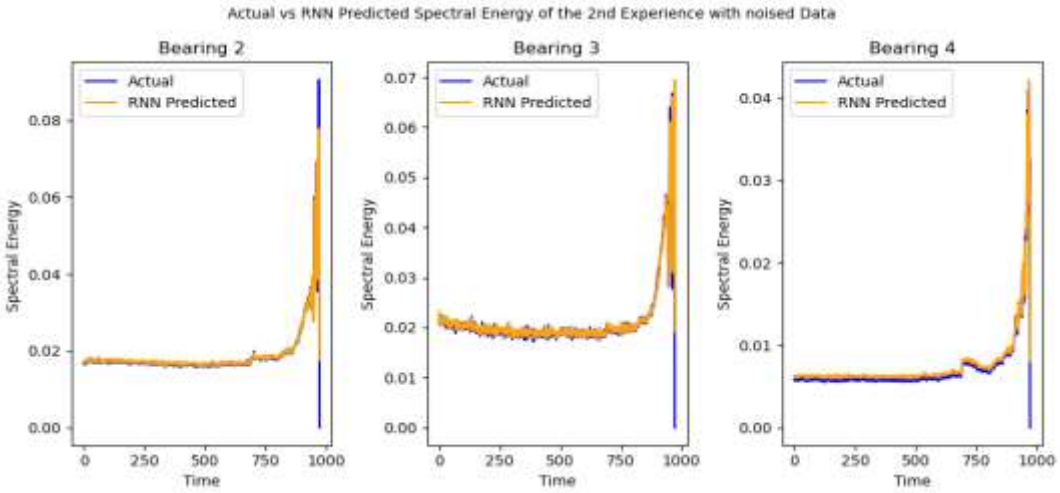


Figure 14: RNN Predictions of Spectral Energy on generalization noised Data

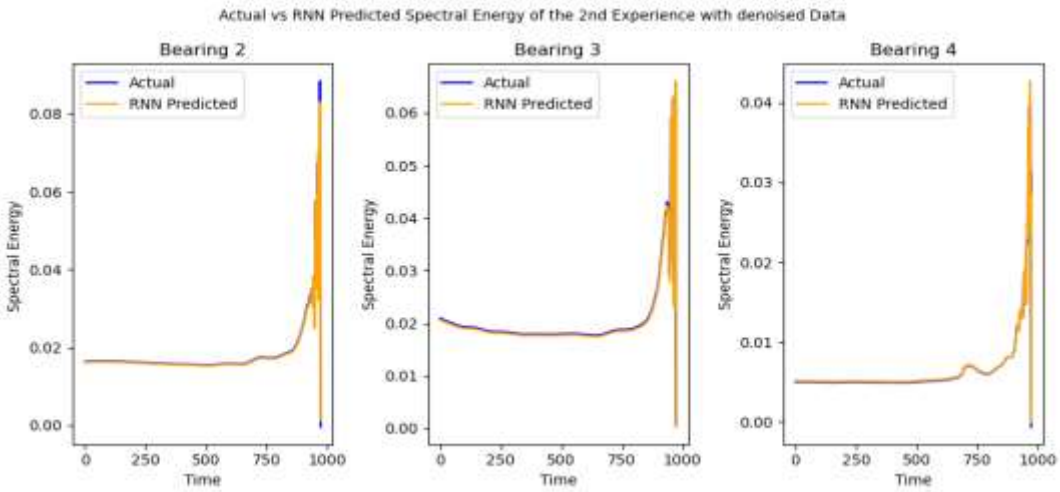


Figure 15: RNN Predictions of Spectral Energy on generalization denoised Data

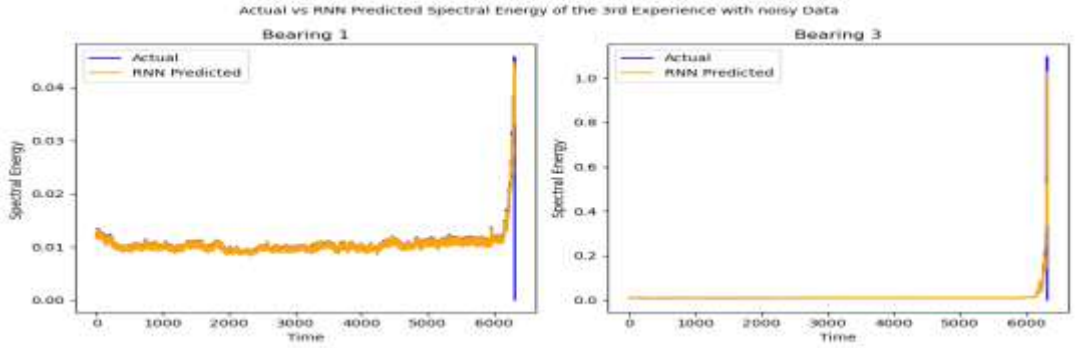


Figure 16: RNN Predictions of Spectral Energy on generalization noised Data

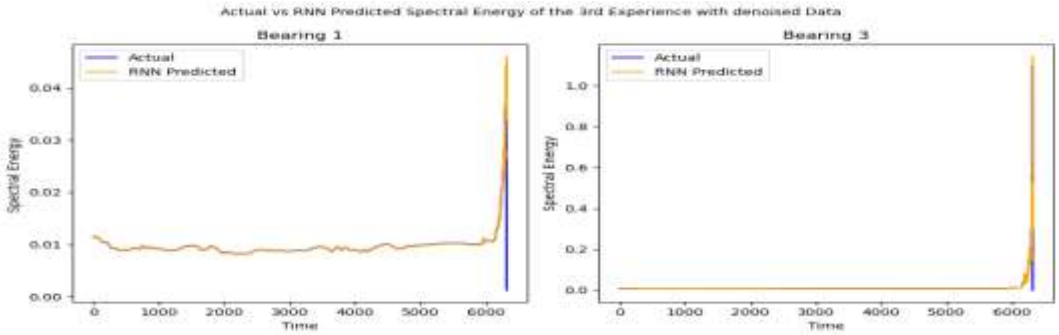


Figure 17: RNN Predictions of Spectral Energy on generalization denoised Data

II.2.1.1.2. LSTM

This model consist of 2 LSTM layers and a dense layer:

- **First LSTM Layer:** Consists of 50 units and returns sequences, allowing the next LSTM layer to process the entire sequence.
- **Second LSTM Layer:** Also with 50 units but does not return sequences, outputting only the final state.
- **Dense Layer:** A fully connected layer that outputs predictions based on the processed sequences.

The model is compiled using the Adam optimizer and MSE as a loss function. It is trained on the prepared data with 100 epochs and a batch size of 32.

Results :

➔ **Time domain:** The model demonstrates superior forecasting performance with denoised data. The best test results are **RMSE: 0.007, MAE: 0.001, and R^2: 0.79**. For the generalization test, the results improve further with **RMSE: 0.002, MAE: 0.001,**

and R^2: 0.90. It is evident that the generalization results are better, indicating the model's robustness and effectiveness.

The results are presented in Figures **18, 19, 20, 21.**

➔ **Frequency domain:** The model shows good performance on the test data, achieving its best results with denoised data: an **RMSE of 0.0014, MAE of 0.0002, and R^2 of 0.87.** The generalization test further highlights the model's effectiveness with denoised data, showing an **RMSE of 0.0005, MAE of 0.00004, and R^2 of 0.93**.

The results are presented in Figures **22, 23, 24, 25.**

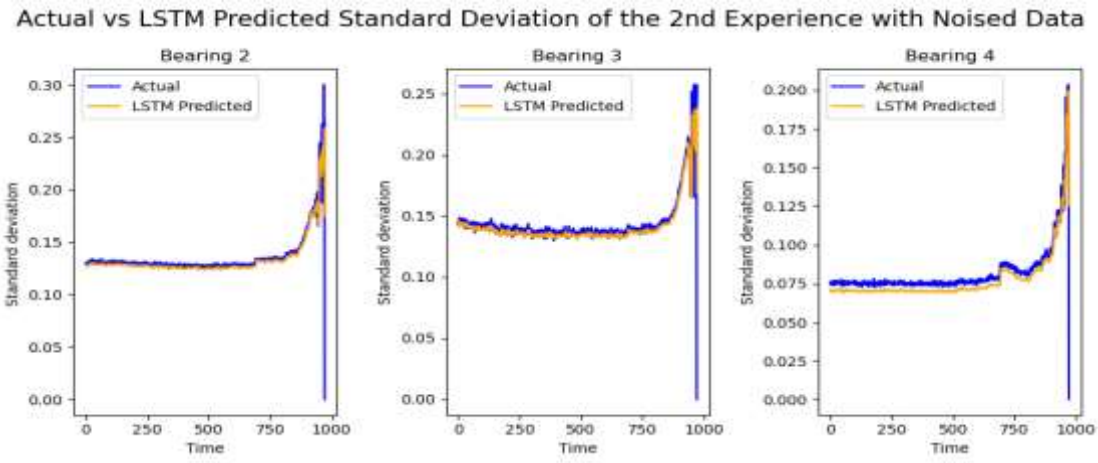


Figure 18: LSTM Predictions of Standard Deviation on test noised Data

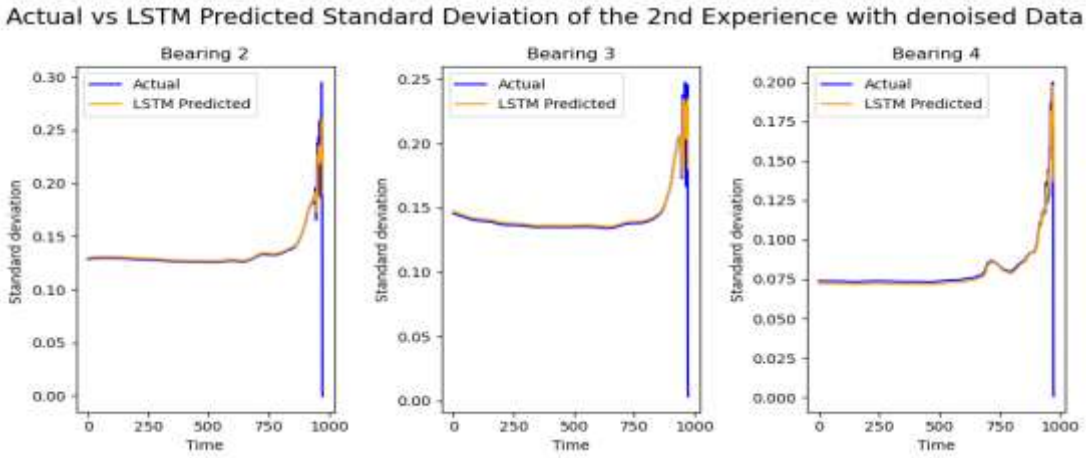


Figure 19: LSTM Predictions of Standard Deviation on test denoised Data

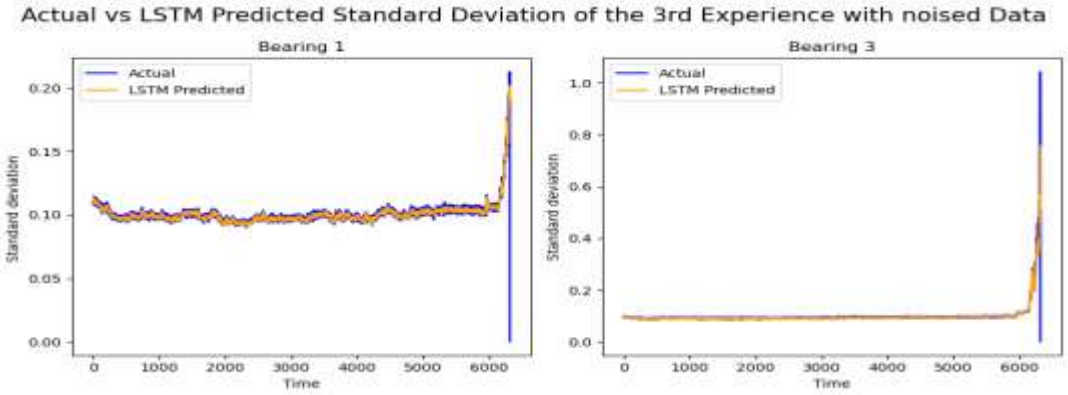


Figure 20: LSTM Predictions of Standard Deviation on generalization noised Data

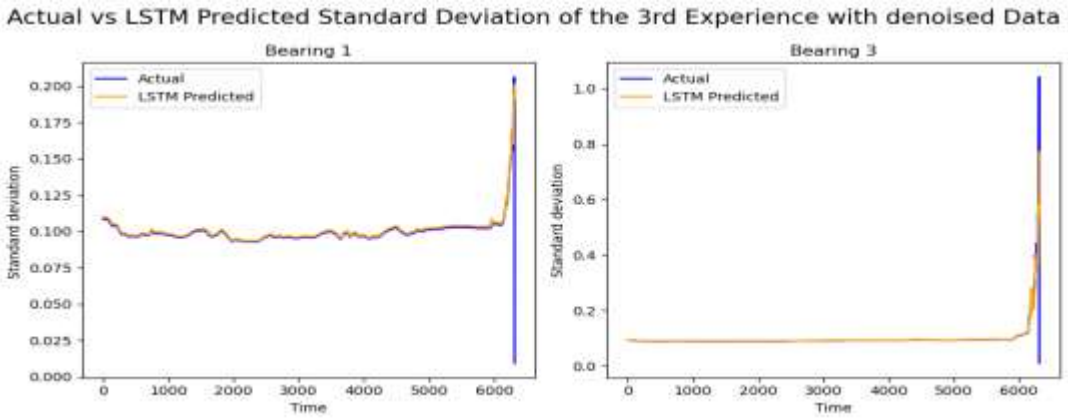


Figure 21: LSTM Predictions of Standard Deviation on generalization denoised Data

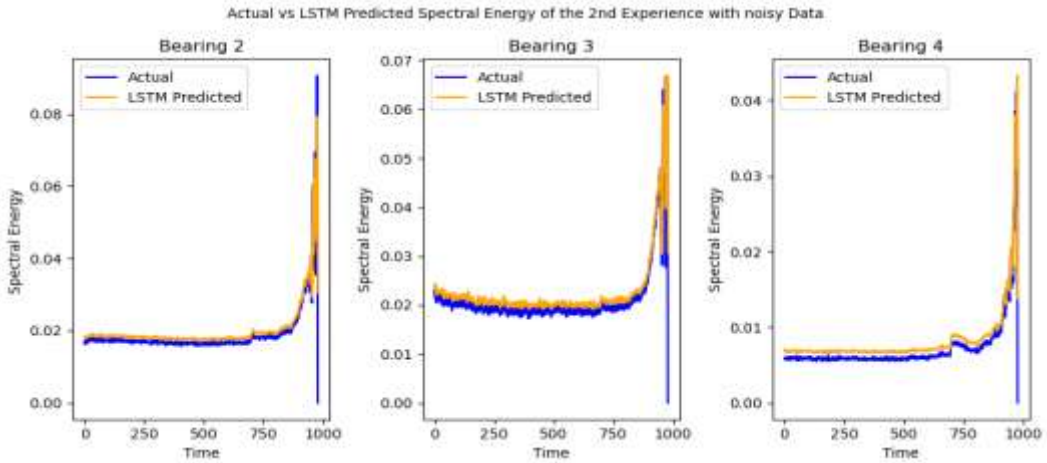


Figure 22: LSTM Predictions of Spectral Energy on test noised Data

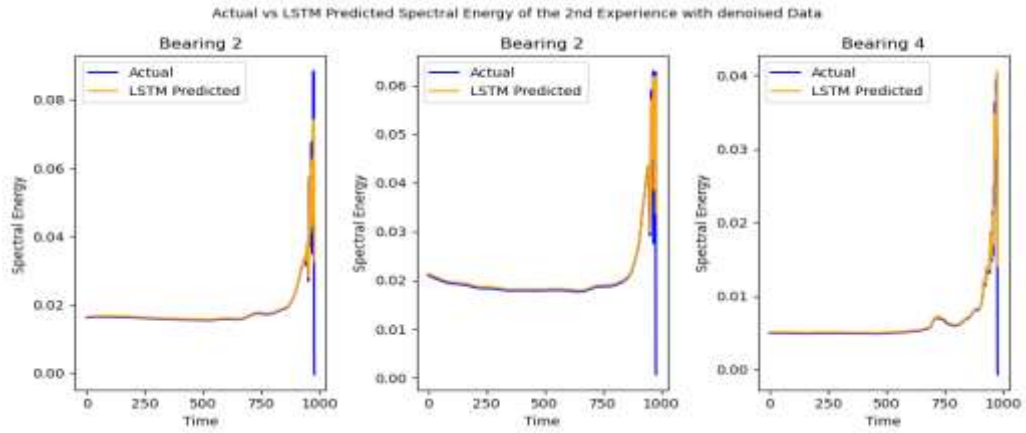


Figure 23: LSTM Predictions of Spectral Energy on test denoised Data

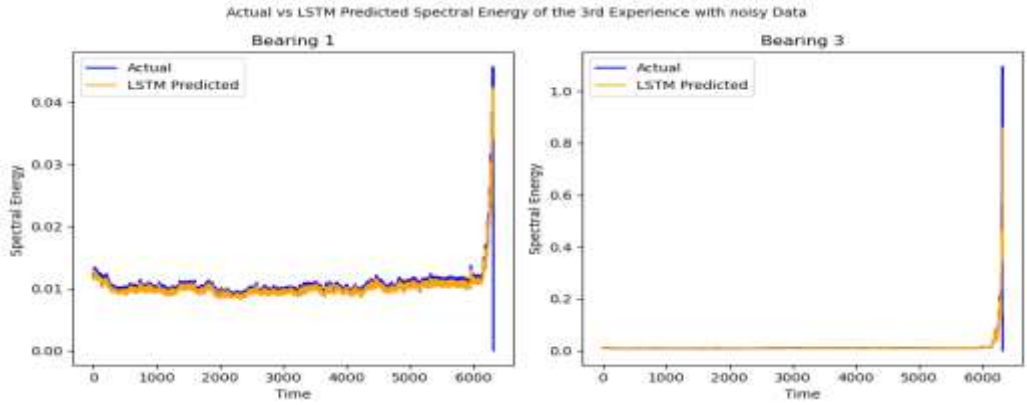


Figure 24: LSTM Predictions of Spectral Energy on generalization noised Data

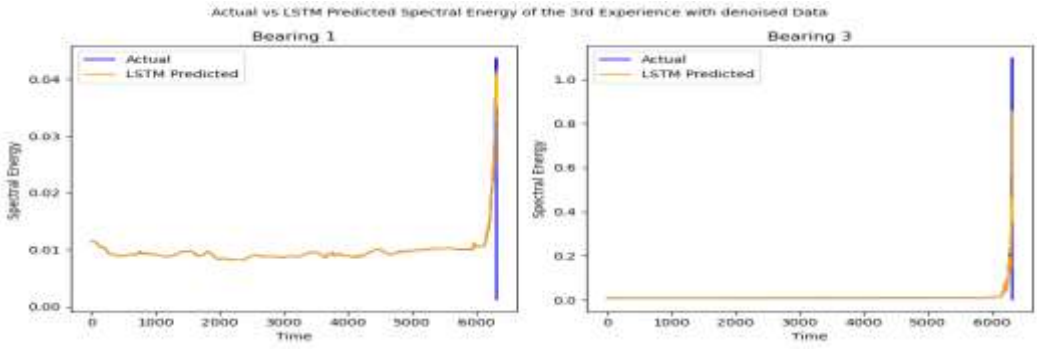


Figure 25: LSTM Predictions of Spectral Energy on generalization denoised Data

II.2.1.1.3. CNN

The CNN model consists of two Conv1D layers with 64 filters each, a kernel size of 2, and ReLU activation, followed by MaxPooling1D layers with a pool size of 2. After flattening the output, the model includes a Dense layer with 50 units and ReLU activation, and a final Dense layer with units corresponding to the number of selected features. The model is compiled with the Adam optimizer and MSE loss function. It is trained for 100 epochs with a batch size of 32, using 10% of the training data for validation.

Results:

- ➔ **Time domain:** The CNN model demonstrates superior forecasting performance with denoised data for the test data when evaluated using MAE and R^2 metrics, and comparable performance using the RMSE metric. For the generalization data, the model performs better with denoised data across all metrics. The best test results are **RMSE: 0.009, MAE: 0.001, and R^2: 0.68**. In contrast, for the generalization test, the results are **RMSE: 0.011, MAE: 0.002, and R^2: 0.60.** These results indicate that the model performs better on the test data but **lacks robustness in generalization**.

  The results are presented in Figures **26, 27, 28 and 29**.

- ➔ **Frequency domain:** The CNN model shows comparable results for bearing 1 with both noisy and denoised data, with only a slight difference in the R^2 metric. For bearing 2, the model performs better with denoised data, whereas for bearing 3, it performs better with noisy data. However, the model presents a negative R^2 value with denoised data for bearing 3, indicating inexplicable results and a lack of stability in the test data.

  For the generalization results, the model demonstrates strong performance with denoised data, achieving an **RMSE of 0.0007, MAE of 0.0003, and R^2 of 0.90.**

  The results are presented in Figures **30, 31, 32 and 33**.

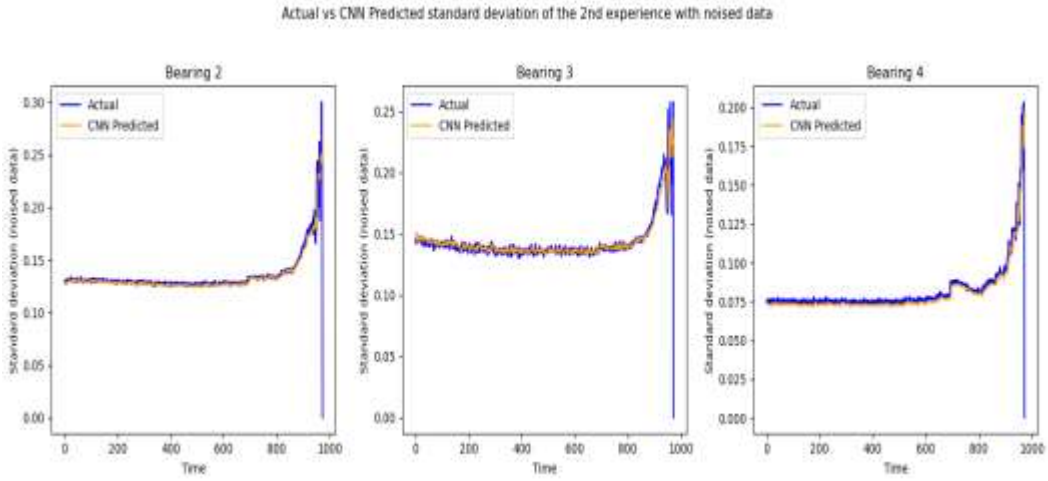


Figure 26: CNN Predictions of Standard Deviation on test noised Data

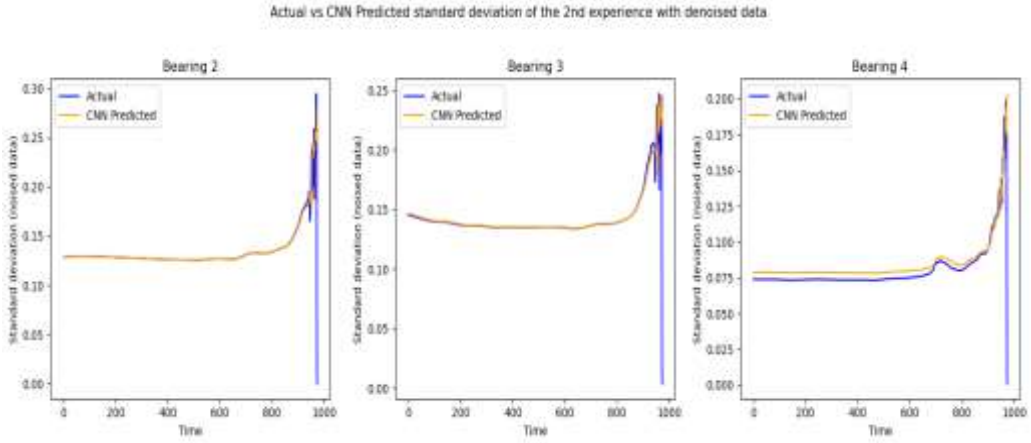


Figure 27: CNN Predictions of Standard Deviation on test denoised Data

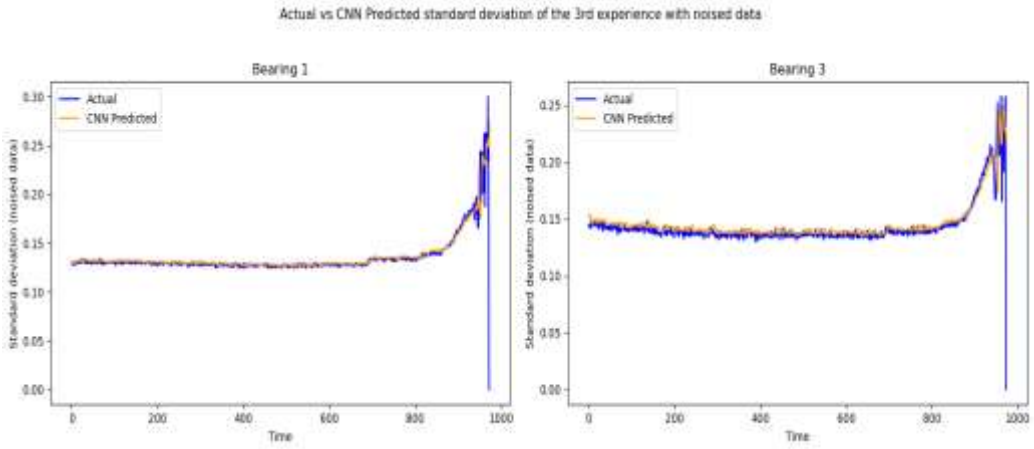


Figure 28: CNN Predictions of Standard Deviation on generalization noised Data

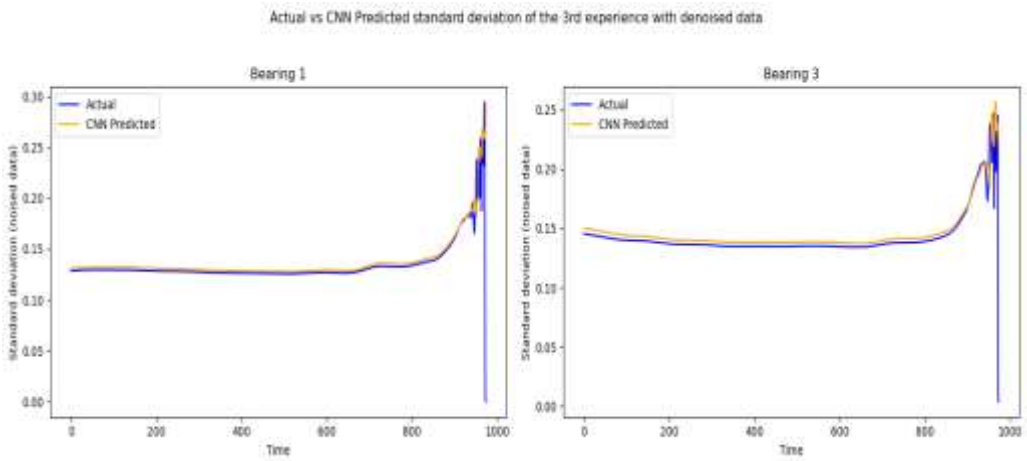


Figure 29: CNN Predictions of Standard Deviation on generalization denoised Data

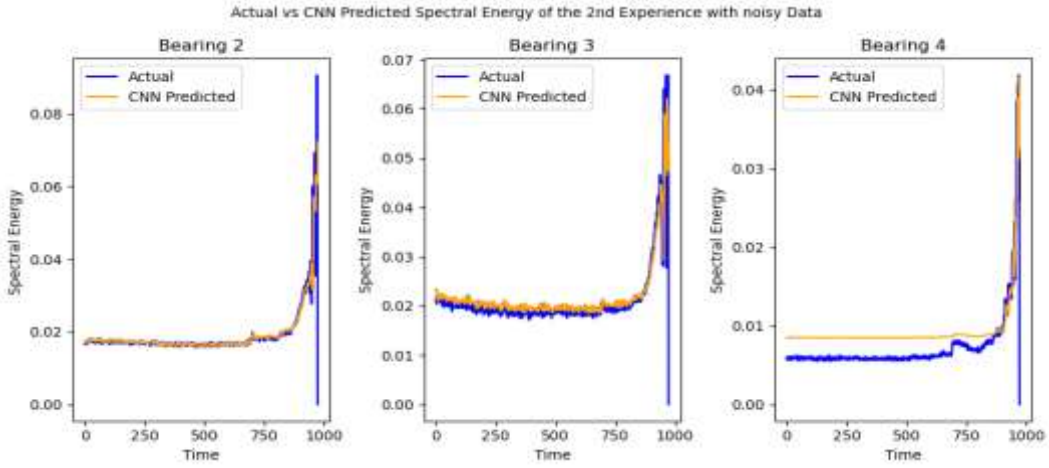


Figure 30: CNN Predictions of Spectral Energy on test noised Data

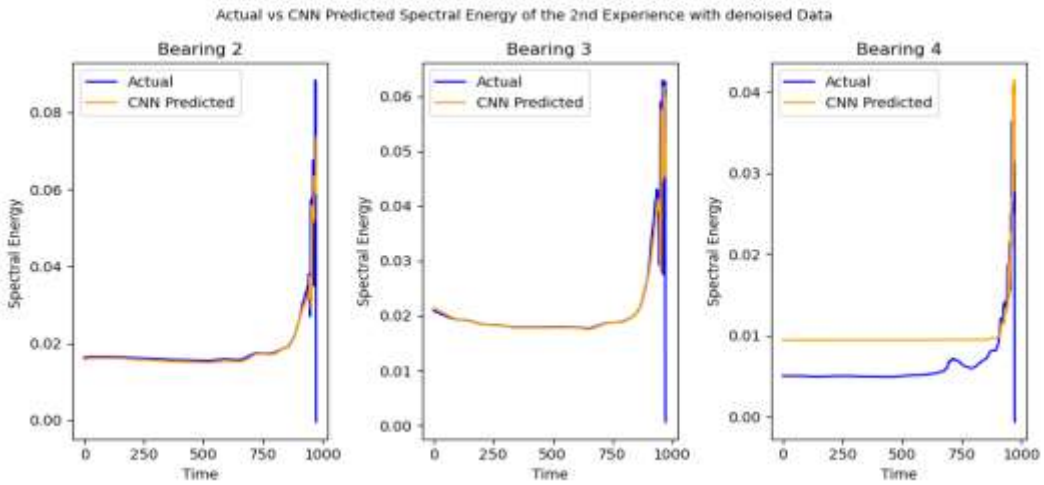


Figure 31: CNN Predictions of Spectral Energy on test denoised Data

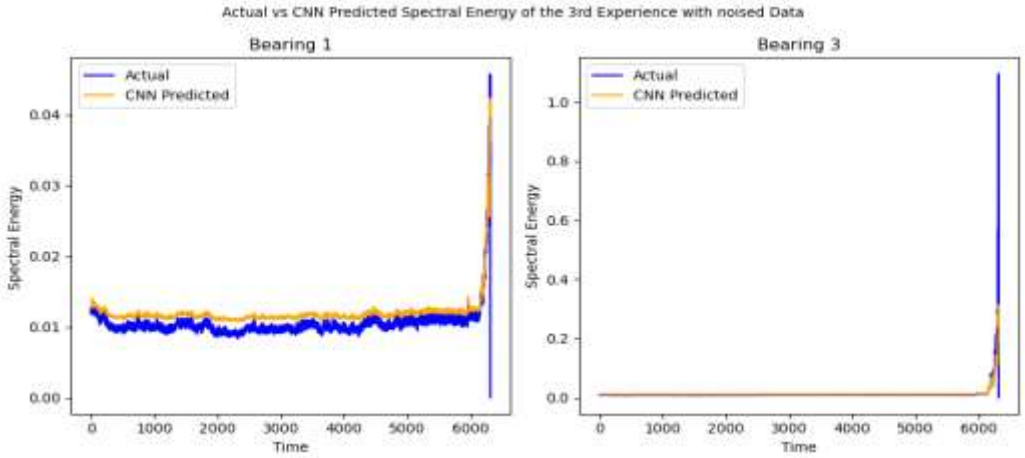


Figure 32: CNN Predictions of Spectral Energy on generalization noised Data

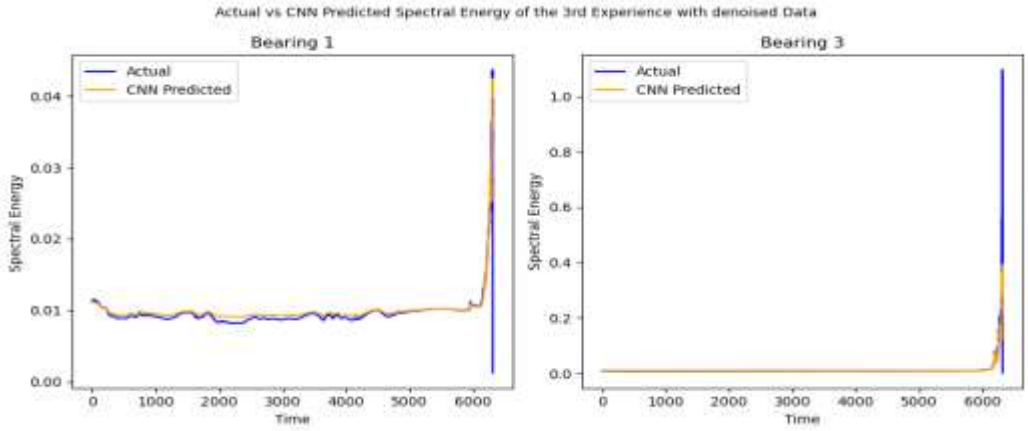


Figure 33: CNN Predictions of Spectral Energy on generalization denoised Data

II.2.1.2. Hybrid Models

In this section, we evaluated the following hybrid models:

II.2.1.2.1. CNN-RNN

The model is a hybrid architecture consisting of convolutional layers (Conv1D), pooling layers (MaxPooling1D), a TimeDistributed layer followed by a SimpleRNN layer, and Dense layers:

- **Conv1D Layer:** This layer applies 1D convolution to the input data with 64 filters and a kernel size of 2, using the ReLU activation function.
- **MaxPooling1D Layer:** This layer performs max pooling on the output of the convolutional layer, reducing the spatial dimensions of the input.

- **TimeDistributed Layer with Dense:** This layer applies a dense transformation to each time step of the input sequence independently. It consists of 50 units with the ReLU activation function.
- **SimpleRNN Layer:** This layer is a basic RNN with 50 units and returns only the last output of the sequence.
- **Dense Layer:** This layer is the final output layer, producing predictions based on the features learned by the preceding layers.

The model is compiled using the Adam optimizer and MSE loss function. It is then trained on the training data for 100 epochs with a batch size of 32, using 10% of the training data for validation.

Results:

➔ **Time domain:** this model showcases superior forecasting performance with denoised test data, achieving equality in the MAE metric between noisy and denoised data with generalization data. Notably, the best test results include an **RMSE of 0.008, MAE of 0.001, and R^2 of 0.70**. Subsequently, the generalization test yields even better results, with an **RMSE of 0.002, MAE of 0.001, and R^2 of 0.92**. These findings underscore the model's robustness and effectiveness, particularly evident in its **superior performance during the generalization phase**.

The results are presented in Figures **34, 35, 36 and 37.**

➔ **Frequency domain:** the model shows better performance with denoised data for both test and generalization datasets. The best test results are an **RMSE of 0.002, MAE of 0.0004, and R^2 of 0.72**. For the generalization test, the results are even better, with an **RMSE of 0.0007, MAE of 0.00041, and R^2 of 0.90**, indicating superior generalization performance compared to the test results.

The results are presented in Figures **38, 39, 40 and 41**.

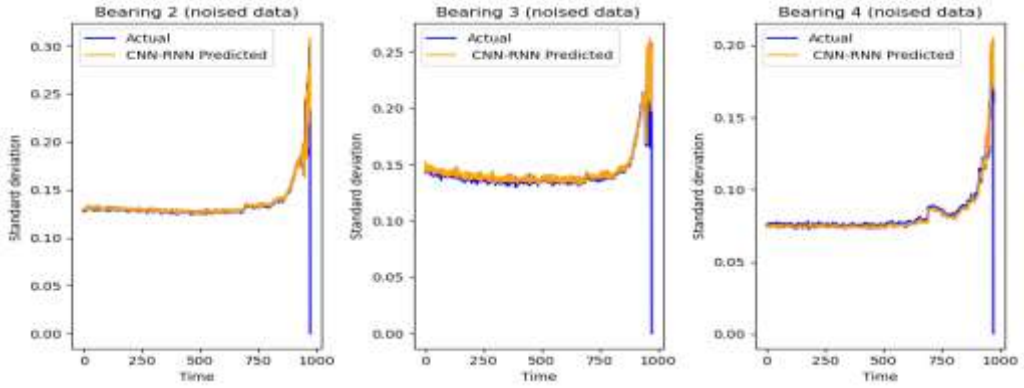


Figure 34: CNN-RNN Predictions of Standard Deviation on test noised Data

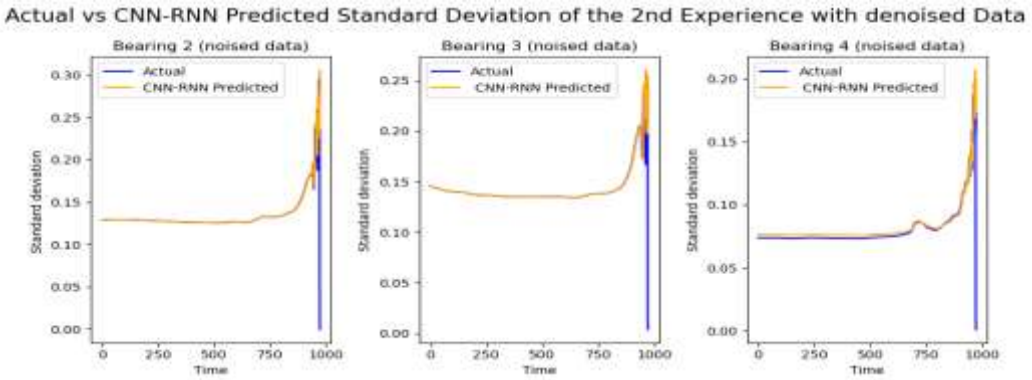


Figure 35: CNN-RNN Predictions of Standard Deviation on test denoised Data

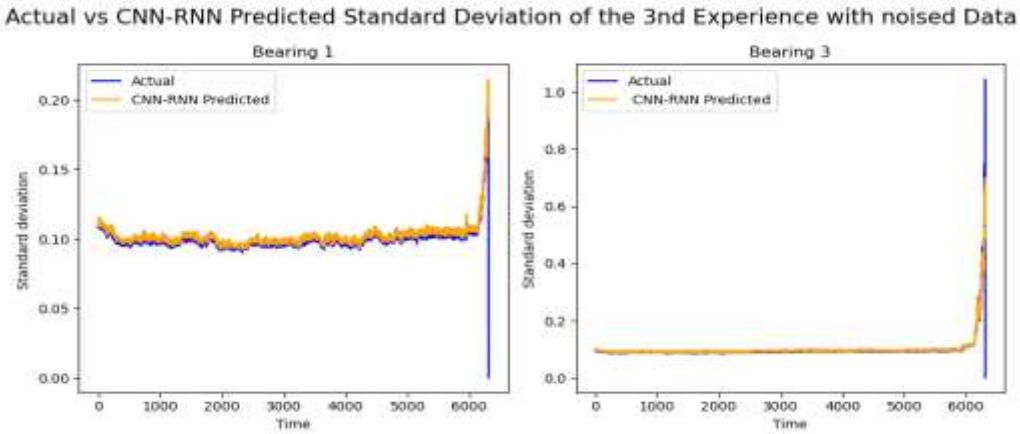


Figure 36: CNN-RNN Predictions of Standard Deviation on generalization noised Data

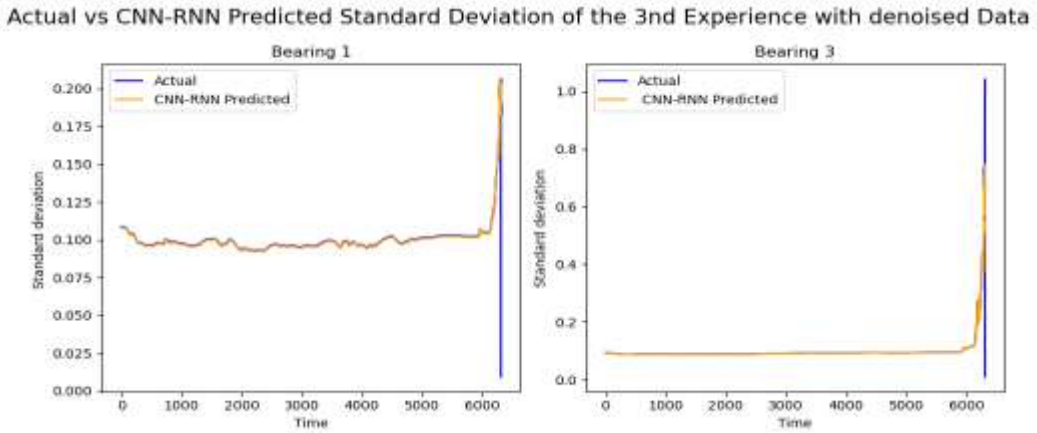


Figure 37: CNN-RNN Predictions of Standard Deviation on generalization denoised

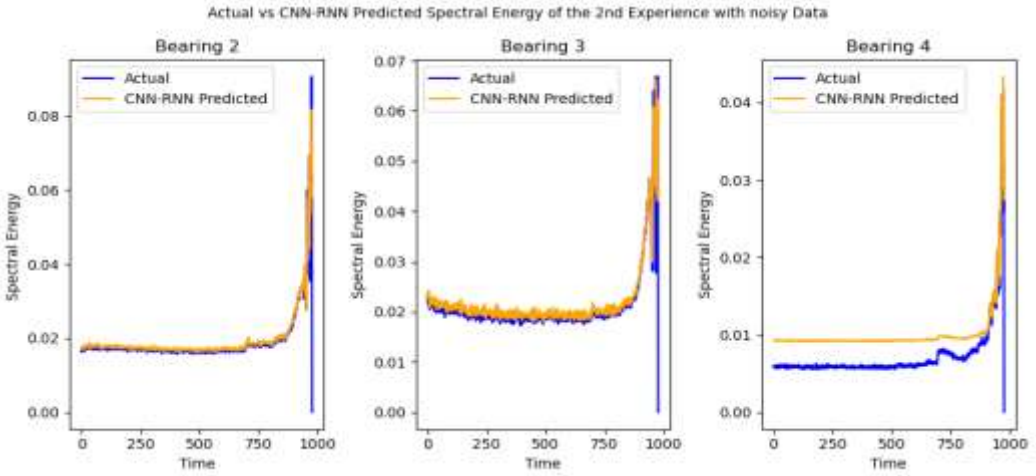


Figure 38: CNN-RNN Predictions of Spectral Energy on test noised Data

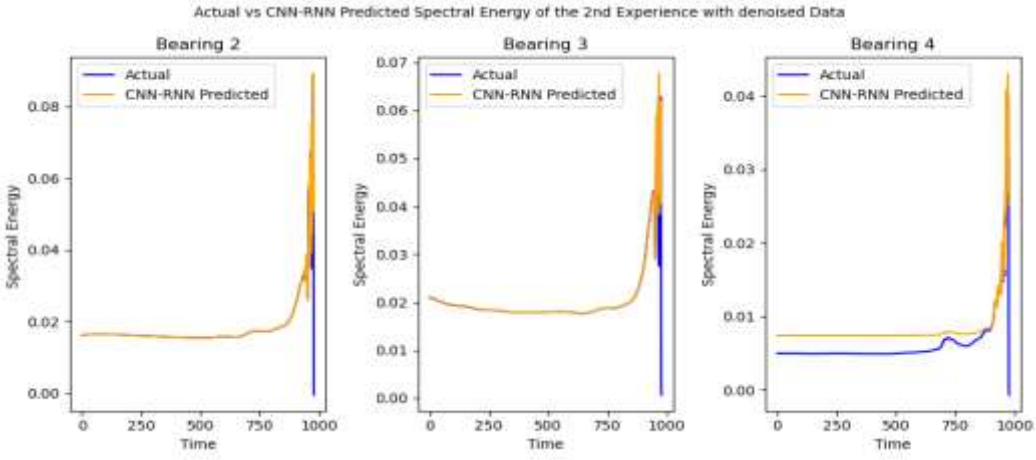


Figure 39: CNN-RNN Predictions of Spectral Energy on test denoised Data

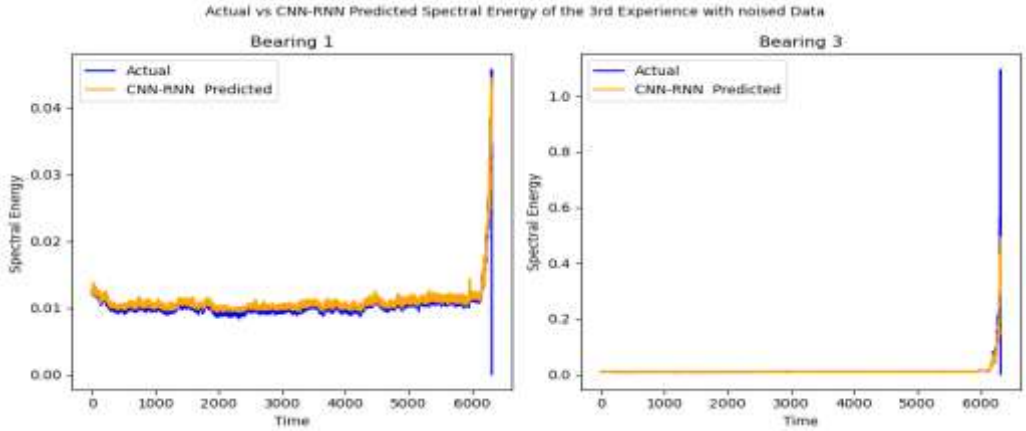


Figure 40: CNN-RNN Predictions of Spectral Energy on generalization noised Data

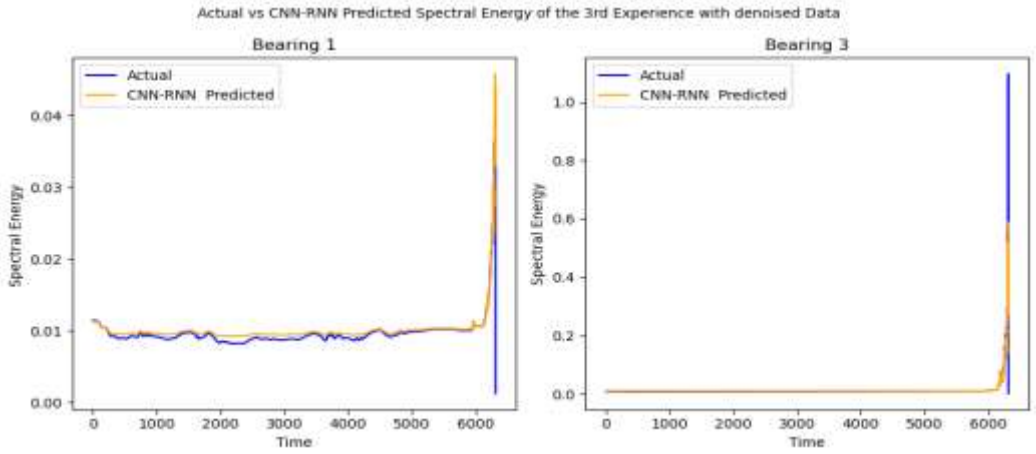


Figure 41: CNN-RNN Predictions of Spectral Energy on generalization denoised Data

II.2.1.2.2. RNN-LSTM

The RNN-LSTM model is constructed using the Sequential API in Keras. It comprises three main layers: a **SimpleRNN layer** with 50 units and returning sequences, followed by **two LSTM layers**, each also with 50 units and returning sequences for the first layer. The second LSTM layer returns only the final output sequence. The model concludes with a **Dense layer** to generate predictions. After compiling the model with the Adam optimizer and MSE loss function, it undergoes training using the provided training data for 100 epochs with a batch size of 32, using 10% of the data for validation during training.

Results:

- **Time domain:** RNN-LSTM model demonstrates remarkable forecasting performance, especially with denoised data. Notably, the model achieves impressive test results with an **RMSE of 0.006, MAE of 0.001, and R^2 of 0.82**. Furthermore, during the generalization test, it performs even better, with an **RMSE of 0.003, MAE of 0.001, and R^2 of 0.90**. These results underscore the model's robustness and effectiveness, particularly evident in its exceptional performance during the generalization phase.
  The results are presented in Figures **42, 43, 44 and 45**.
- **Frequency domain:** This model shows better performance with denoised data for both test and generalization datasets. The best test results are an **RMSE of 0.001, MAE of 0.0003, and R^2 of 0.89**. For the generalization test, the results are even better, with an **RMSE of 0.0006, MAE of 0.00003, and R^2 of 0.92**, indicating superior generalization performance compared to the test results.
  The results are presented in Figures **46, 47,48 and 49.**

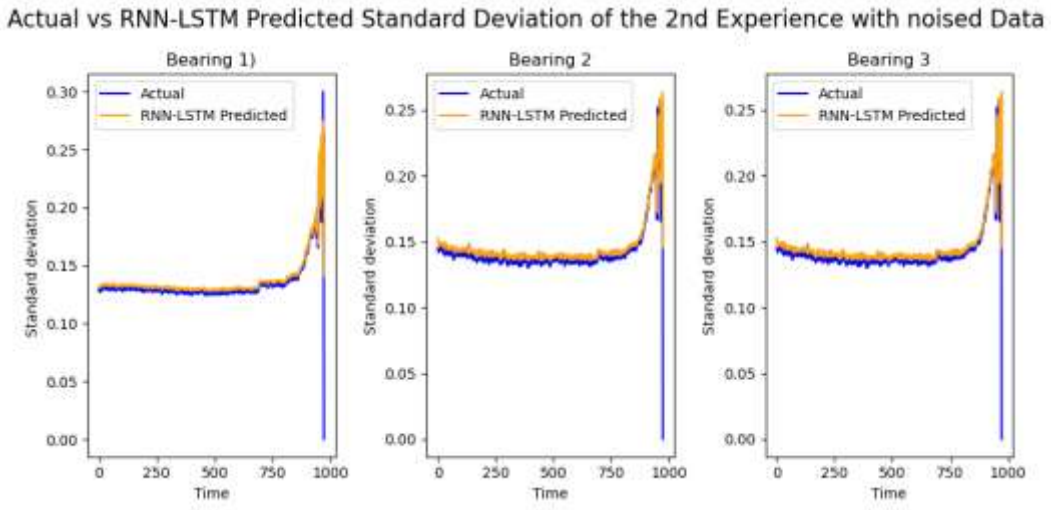


Figure 42: RNN-LSTM Predictions of Standard Deviation on test noised Data

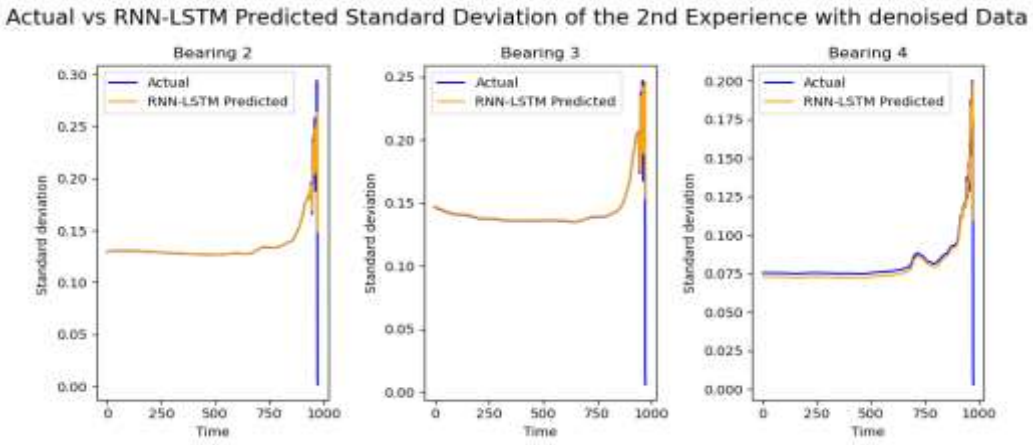


Figure 43: RNN-LSTM Predictions of Standard Deviation on test denoised Data

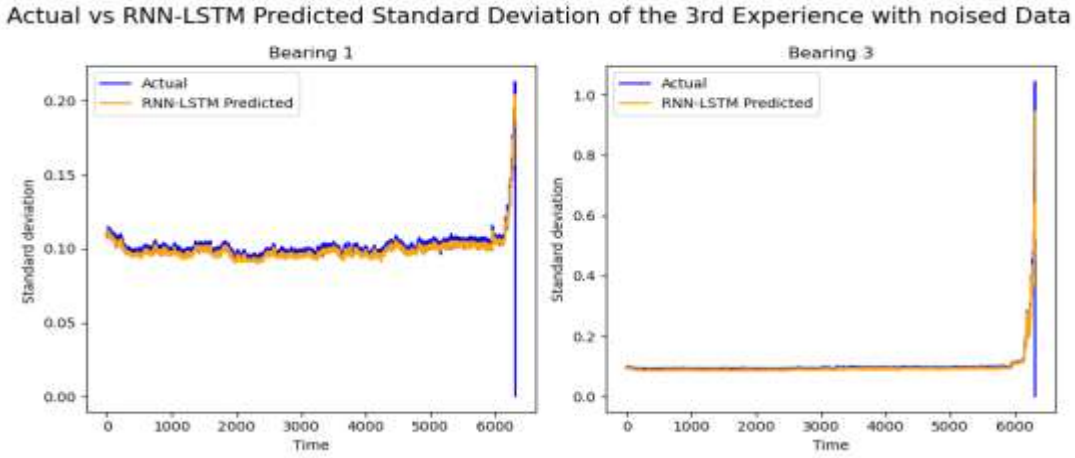


Figure 44: RNN-LSTM Predictions of Standard Deviation on test denoised Data

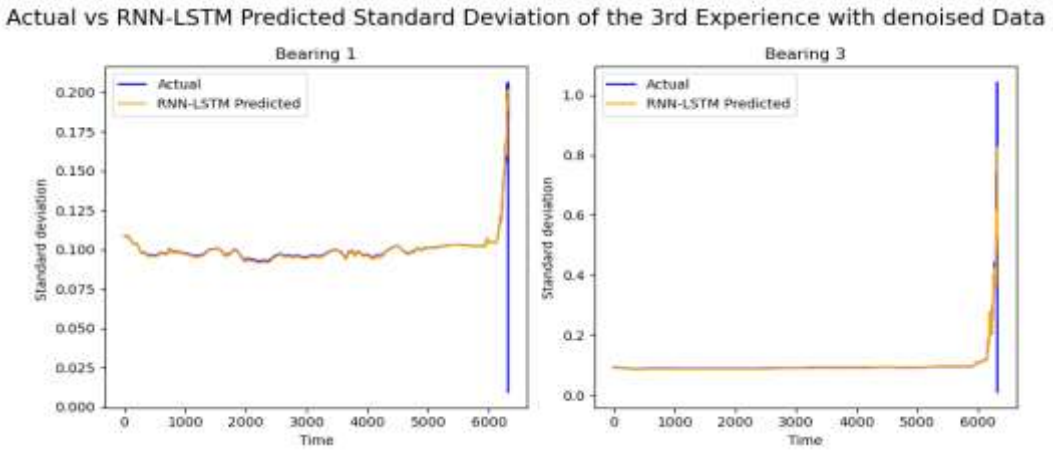


Figure 45: RNN-LSTM Predictions of Standard Deviation on test denoised Data

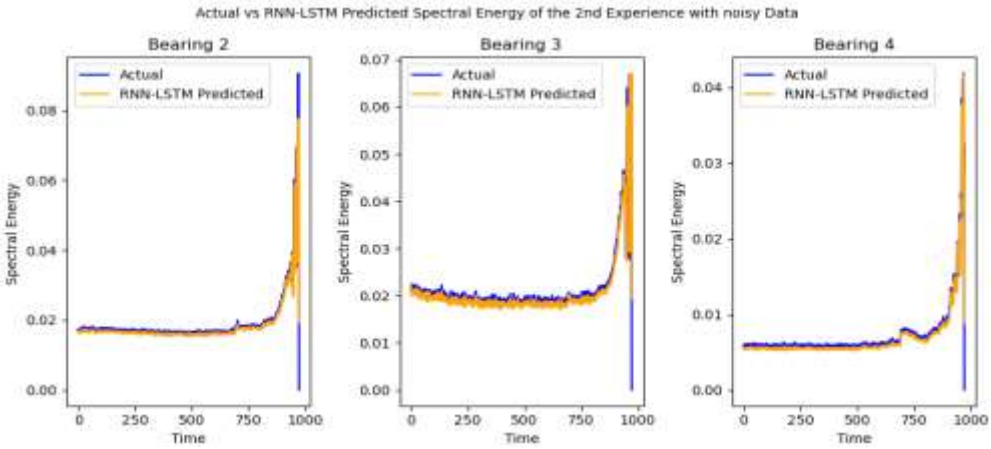


Figure 46: RNN-LSTM Predictions of Spectral Energy on test noised Data

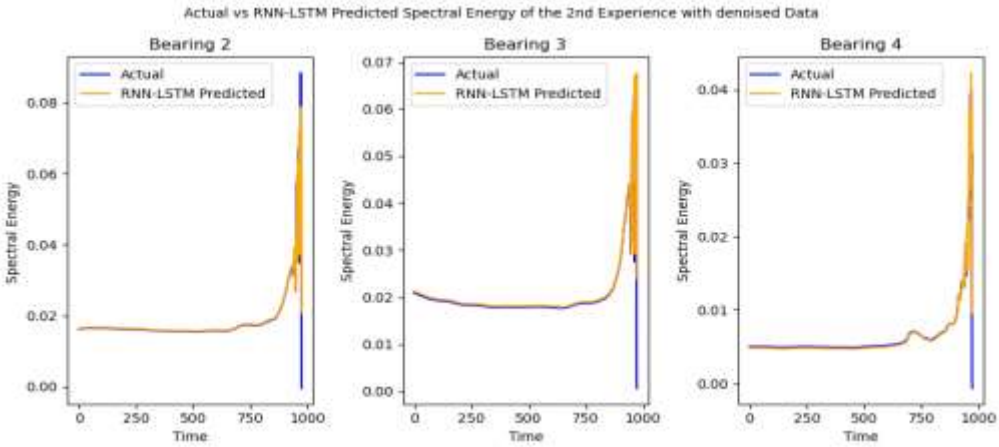


Figure 47: RNN-LSTM Predictions of Spectral Energy on test denoised Data

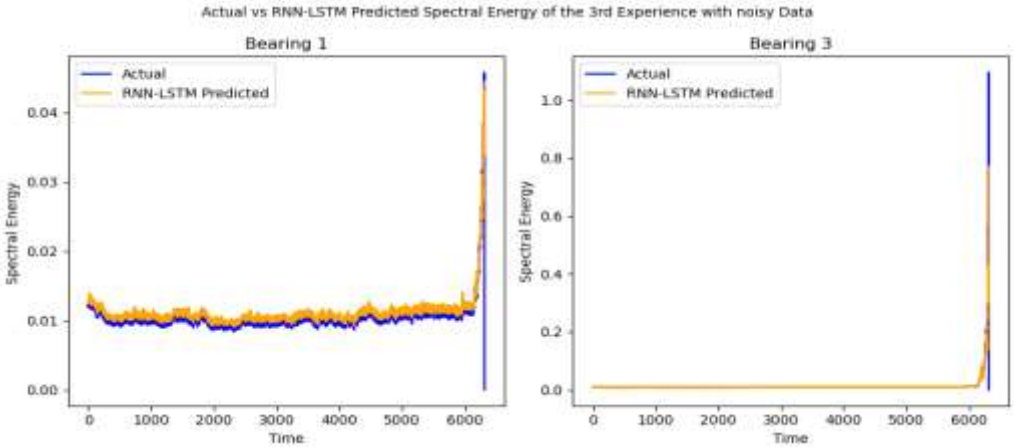


Figure 48: RNN-LSTM Predictions of Spectral Energy on generalization noised Data

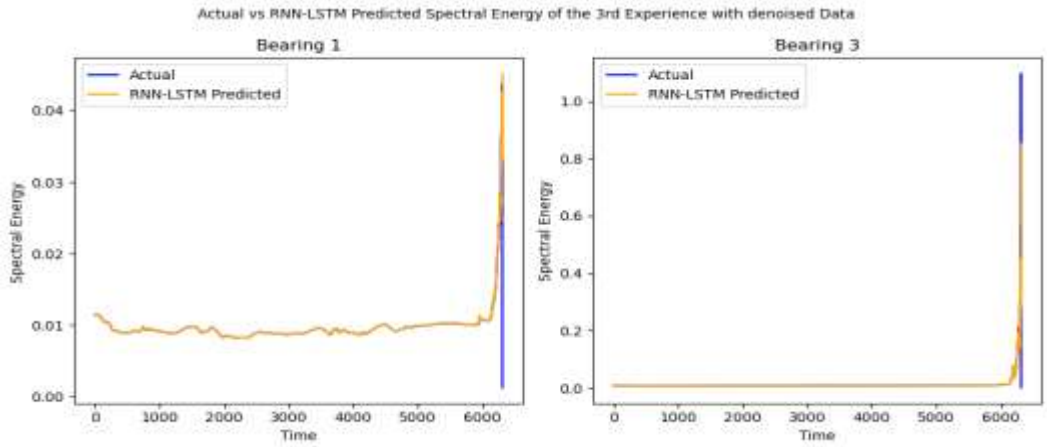


Figure 49: RNN-LSTM Predictions of Spectral Energy on generalization denoised Data

II.2.1.2.3. CNN-LSTM

The CNN-LSTM model comprises several key layers. Initially, a **Conv1D layer** with 64 filters and a kernel size of 2 is applied to the input data, followed by **max-pooling** to reduce spatial dimensions. Subsequently, a **TimeDistributed layer** with 50 units and ReLU activation is used to apply dense transformations to each time step independently. **Two LSTM layers** with 50 units each are then employed, the first returning sequences and the second providing the final output. Finally, a **Dense layer** generates predictions based on the features learned by the preceding layers. The model is compiled using the Adam optimizer and MSE loss function, and subsequently trained on the provided training data for 100 epochs with a batch size of 32, using 10% of the data for validation during training.

Results:

- ➔ **Time domain:** The CNN-LSTM model exhibits good forecasting performance, particularly with denoised data, where an equivalent MAE metric is observed between the test data with noisy and denoised inputs. Noteworthy are the commendable test results, boasting an **RMSE of 0.009, MAE of 0.002, and R^2 of 0.69**. Furthermore, during the generalization test, it demonstrates superior performance, with an **RMSE of 0.002, MAE of 0.0004, and R^2 of 0.92.** These findings underscore the robustness and efficacy of the model, notably evident in its **exceptional performance during the generalization phase**.

The results are presented in Figures **50, 51, 52 and 53.**

➔ **Frequency domain:** This model demonstrates improved performance with denoised data for both the test and generalization datasets. The best test results include an **RMSE of 0.003, MAE of 0.0005, and R^2 of 0.72**. The generalization results are even more impressive, achieving an **RMSE of 0.0009, MAE of 0.0006, and R^2 of 0.83**, highlighting the model's superior generalization capability compared to the test performance.

The results are presented in Figures **54, 55, 56 and 57.**

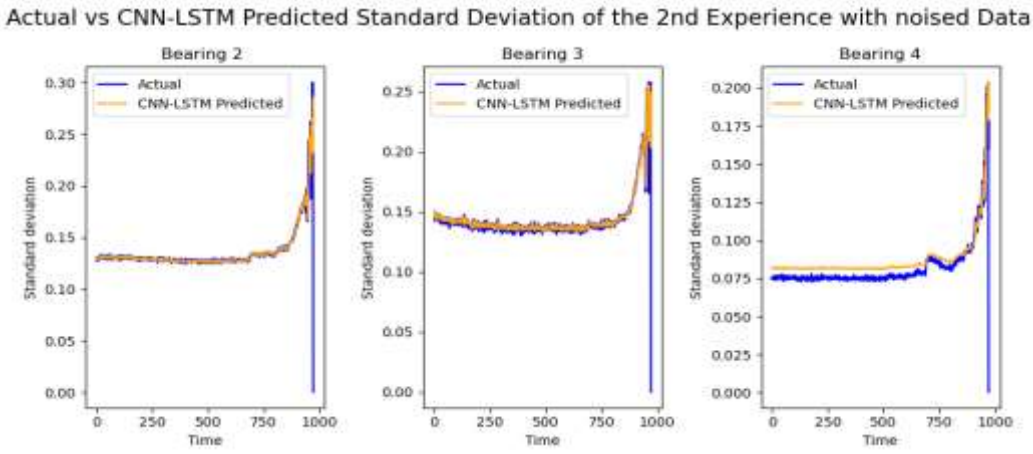


Figure 50: CNN-LSTM Predictions of Standard Deviation on test noised Data

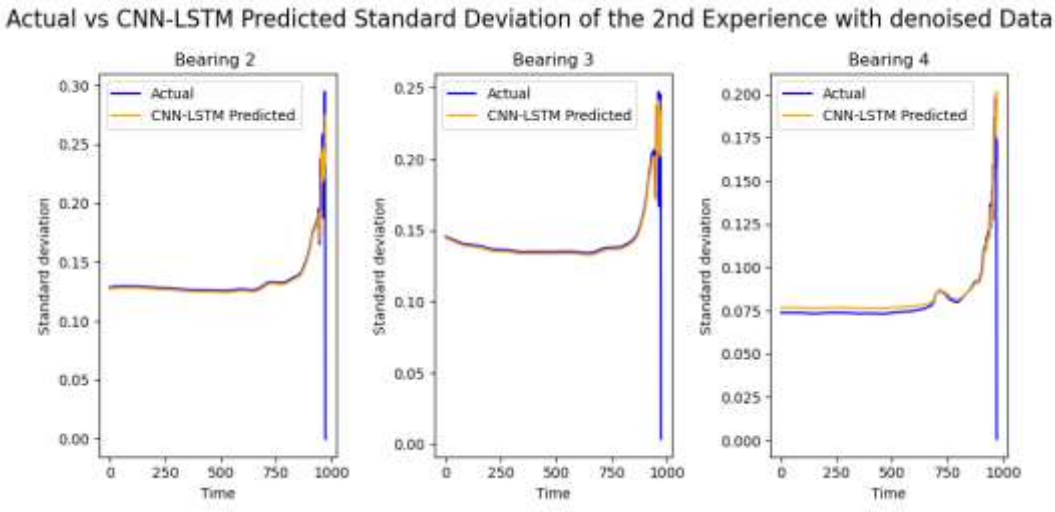


Figure 51: CNN-LSTM Predictions of Standard Deviation on test denoised Data

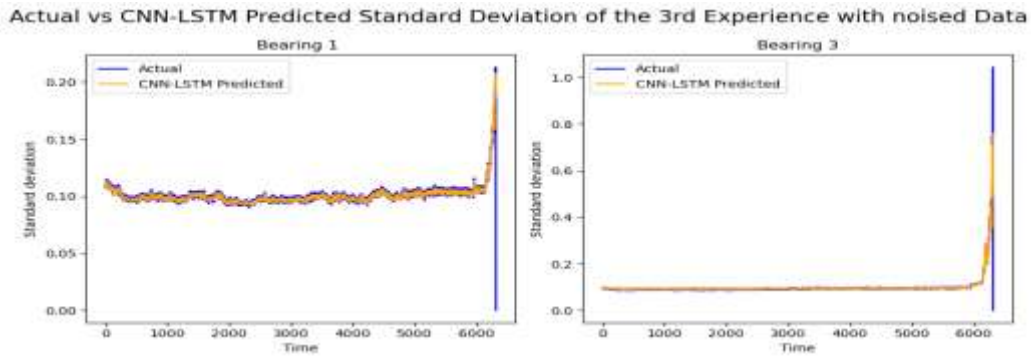


Figure 52: CNN-LSTM Predictions of Standard Deviation on generalization noised Data (Time domain)

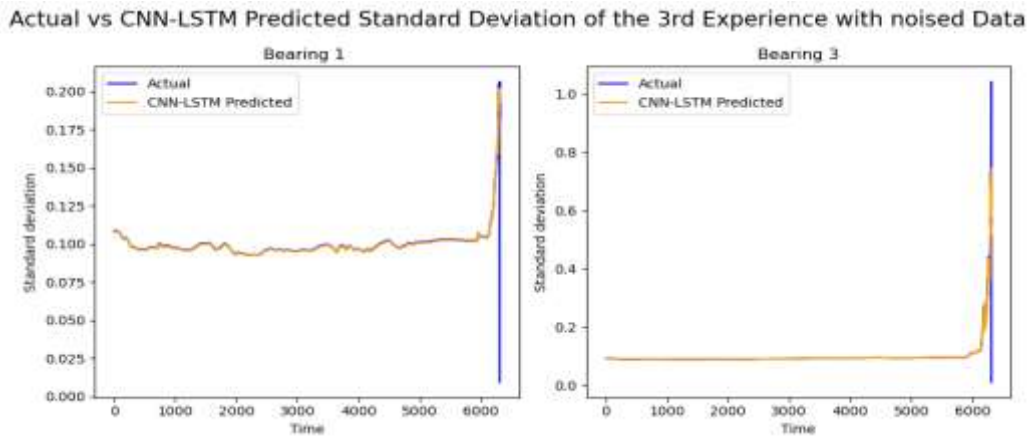


Figure 53: CNN-LSTM Predictions of Standard Deviation on generalization denoised Data (Time domain)

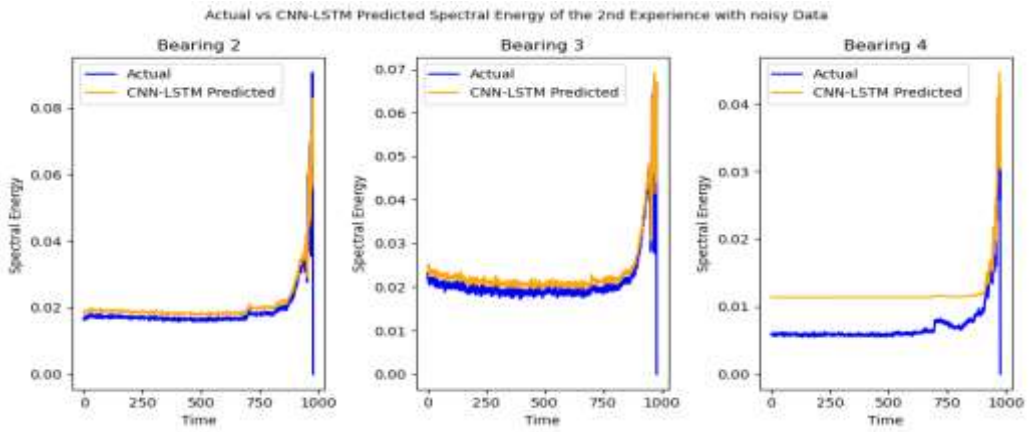


Figure 54: CNN-LSTM Predictions of Spectral Energy on test noised data

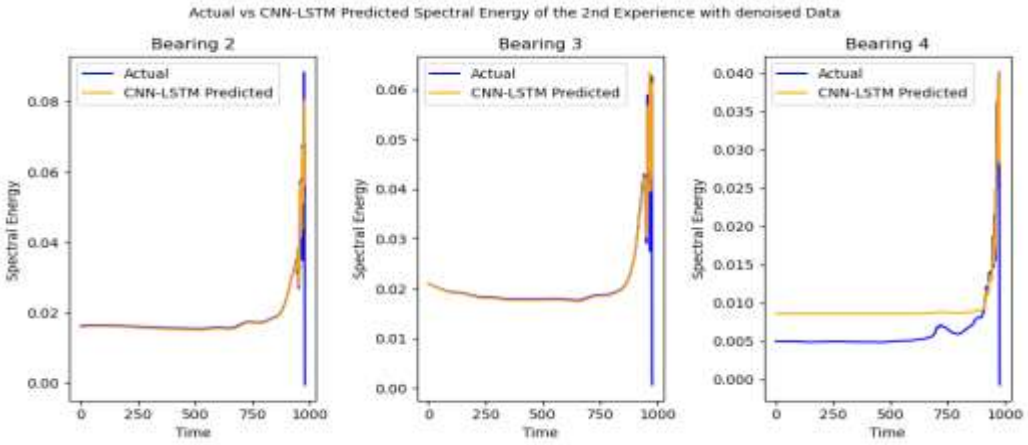


Figure 55: CNN-LSTM Predictions of Spectral Energy on test denoised data

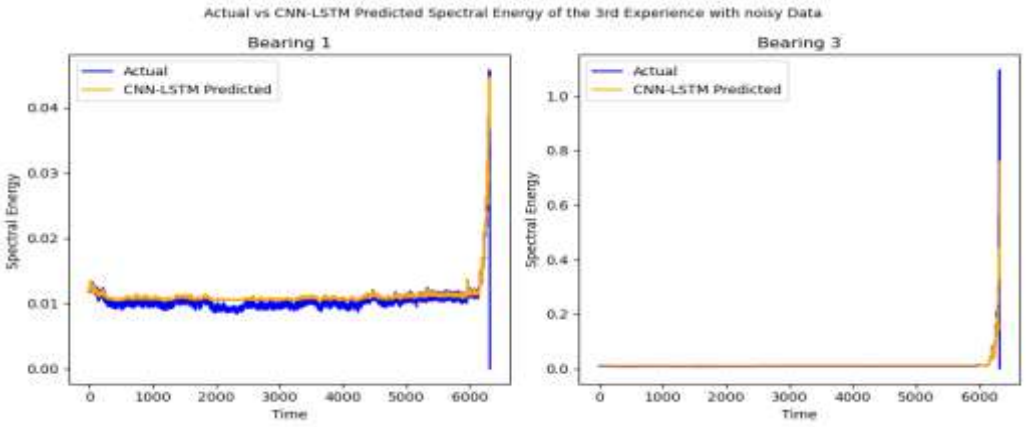


Figure 56: CNN-LSTM Predictions of Spectral Energy on generalization noised data (Frequency Domain)

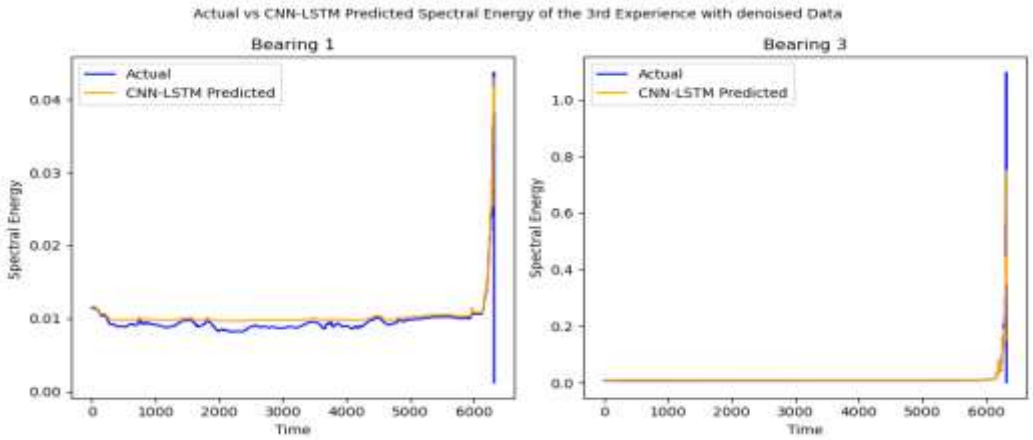


Figure 57: CNN-LSTM Predictions of Spectral Energy on generalization denoised data (Frequency Domain)

II.2.1.3. Experiment Conclusion

Results conclusion of single models

Based on the results, the models generally perform better with denoised data in both the frequency and time domains. Both LSTM and RNN show good generalization abilities, while CNN does not, resulting in less stable outcomes. Additionally, RNN surpasses LSTM in generalization within the time domain. In contrast, in the frequency domain, LSTM outperforms RNN in generalization, achieving an RMSE of 0.0005, an MAE of 0.00004, and an R^2 of 0.93.

Results conclusion of hybrid models:

After analyzing the results of the tested hybrid models in both the time and frequency domains, we notice that they achieve superior performance with denoised data during generalization testing. In the time domain, the CNN-LSTM model emerged as the top performer, with an RMSE of 0.002, an MAE of 0.0004, and an R^2 of 0.92. Similarly, in the frequency domain, the models generally performed better with denoised data and demonstrated good generalization abilities. Among these, the RNN-LSTM model outperformed the others, achieving an RMSE of 0.0006, an MAE of 0.00003, and an R^2 of 0.92.

General Results Conclusion for time and frequency domain:

Comparing the best results of a single model RNN with those of a hybrid model CNN-LSTM in the time domain reveals that their performances are very close. Both models achieve the same RMSE of 0.002. The RNN model performs slightly better in terms of MAE, with a score of 0.0003 compared to CNN-LSTM’s 0.0004. However, the CNN-LSTM model outperforms the RNN in R^2, achieving 0.92 versus the RNN's 0.91. In the frequency domain, after comparing the best results between the single LSTM model and the hybrid RNN-LSTM model, the single LSTM model is chosen for its overall better performance, it slightly outperforms in RMSE and demonstrates a superior R^2 value, indicating its stronger ability to capture the data's variance. Despite RNN-LSTM's slight advantage in MAE, the higher R^2 value for the LSTM model signifies its better reliability in understanding the underlying patterns, making it the preferred choice for forecasting tasks.

Conclusion for the experiment 01 (Best model Results):

Based on the results from both the frequency and time domains, we observe that denoised data consistently yields better outcomes. Furthermore, by comparing the two domains, we observe that the frequency domain exhibits significantly superior results. Consequently, the **LSTM model** applied to **denoised data in the frequency domain** emerges as the optimal choice for forecasting in our experiment with an **RMSE of 0.0005, an MAE of 0.00004, and an R^2 of 0.93.**

II.2.2. Experiment 02

The objective of this experiment is to ameliorate the results of the chosen LSTM model by correcting its predictions using an ARIMA model for residual correction with denoised data in the frequency domain. For this purpose, we calculated the residuals between the LSTM predictions and the real data of bearing 2 of the 2nd test, which were then passed to ARIMA as training data in order to train to predict the residuals. The final test of the hybrid model was conducted using data from bearings 3 and 4 of the 2nd test. To evaluate generalization, we used data from bearings 1 and 3 of the 3rd test. The process of this experiment is presented if **Figure 58**.

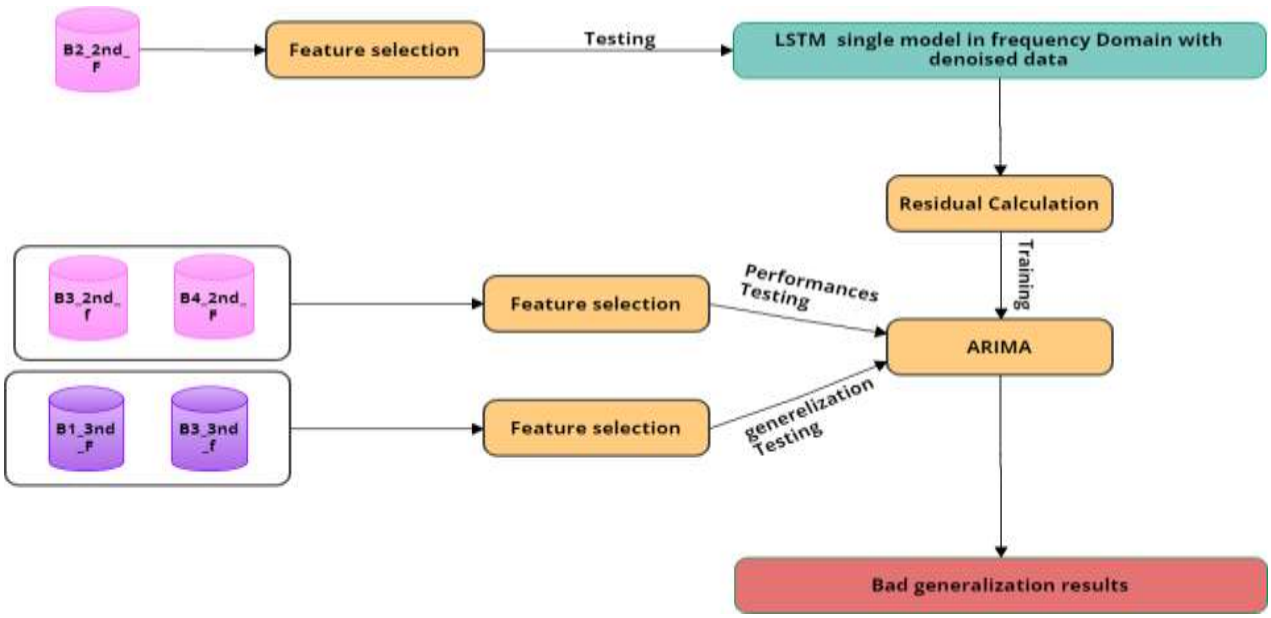


Figure 58: Experiment 02 for next state prediction

The results for the testing data are presented in **Figures 59 and 60**, with the corresponding metrics detailed in **Table 4**. The generalization results are illustrated in **Figures 61 and 62**, with the associated metrics provided in **Table 5**. where we can constat after analyzing the results, that the LSTM model followed by ARIMA for residual correction performs well with the testing data, but **demonstrates poor generalization**. So, we conclude that a single **LSTM model is better for the next state prediction.**

Testing data	**Evaluation metrics**	**LSTM then ARIMA model**
Bearing 3	$RMSE$	**0.007**
	MAE	**0.001**
	R^2	**0.84**
Bearing 4	$RMSE$	0.007
	MAE	0.003
	R^2	0.79

Table 4: LSTM then ARIMA testing results

generalization data	**Evaluation metrics**	**LSTM then ARIMA model**
Bearing 1	$RMSE$	0.58
	MAE	0.52
	R^2	-4451.28
Bearing 3	$RMSE$	0.40
	MAE	0.35
	R^2	-87.01

Table 5: LSTM then ARIMA generalization results

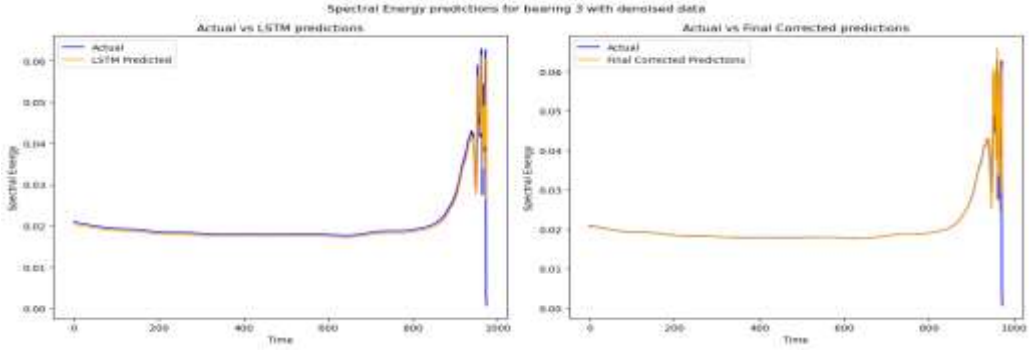


Figure 59: LSTM then ARIMA Spectral Energy predictions for bearing 3 of testing data

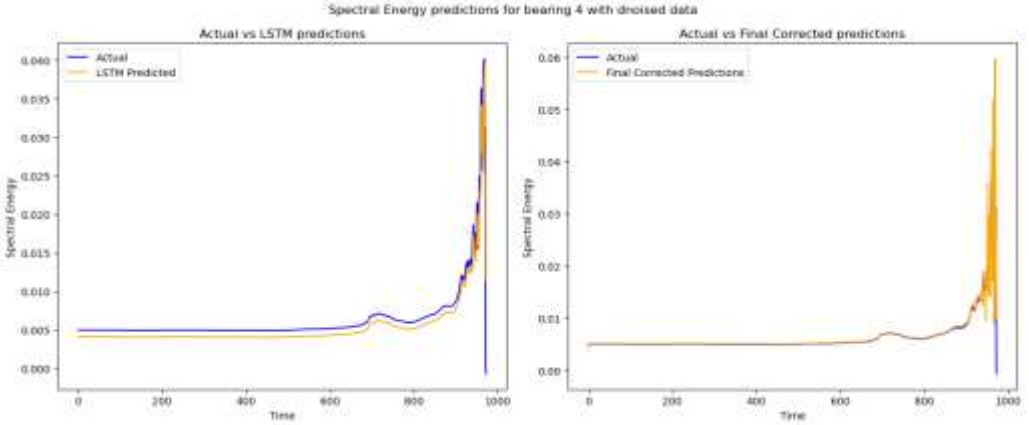


Figure 60: LSTM then ARIMA Spectral Energy predictions for bearing 4 of testing data

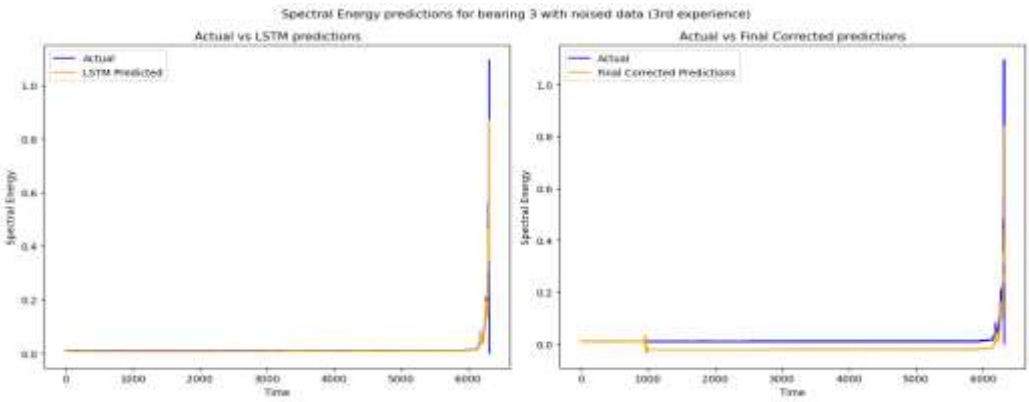


Figure 61: LSTM then ARIMA Spectral Energy predictions for bearing 3 of generalization data

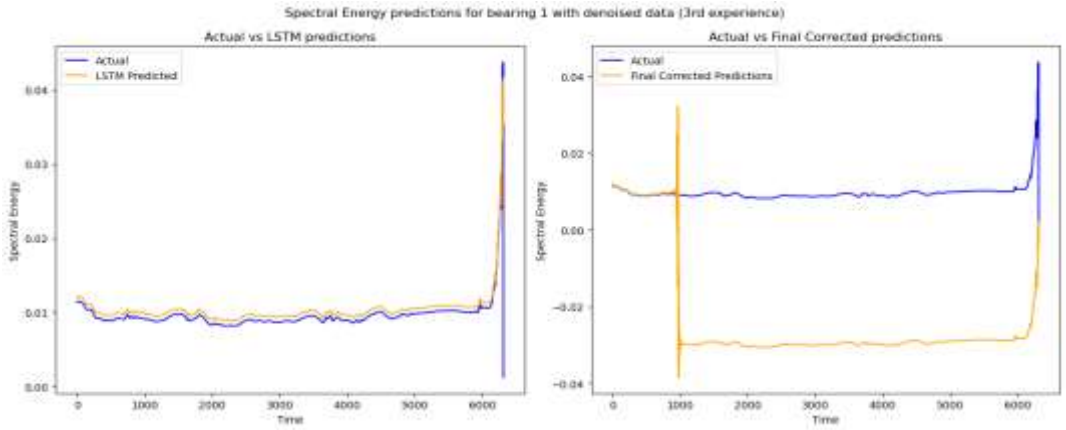


Figure 62: LSTM then ARIMA Spectral Energy predictions for bearing 1 of generalization data

II.3. Anomaly Detection

After predicting the next state, our goal is to detect change points that indicate abnormal behavior. To achieve this, we defined a Change Point Detection function that compares the predicted point at □+1 with the real point at time □. This comparison involves calculating the difference between them. If this difference exceeds a certain threshold, it is flagged as a change point as presented in **figure 63**.

Through extensive testing with various thresholds, we found that the thresholds $16 * 10^{-6}, 17 * 10^{-6}, 18 * 10^{-6}$ yielded similar results. Therefore, we selected the median threshold of $17 * 10^{-6}$. We validated this threshold using data from bearings 2, 3, and 4 of the 2nd test. Additionally, we tested its generalization using the data from bearing 3 of the 3rd test. The results are shown in Figures **64, 65 and 66.**

Note: The graphs representing the detection of change points use the x-axis to denote time in minutes where each point represent 10 min, and the y-axis to show the predicted spectral energy in the frequency domain. In these graphs, the blue color represents the actual next state, while the orange color indicates the predicted next state and red lines indicate the detected change points using the defined threshold.

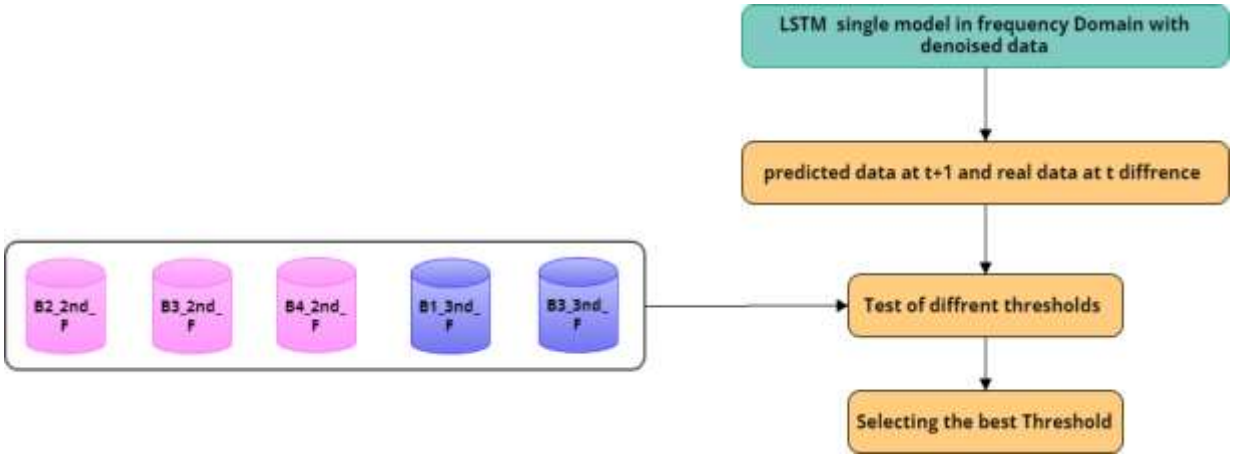


Figure 63: Change point detection process

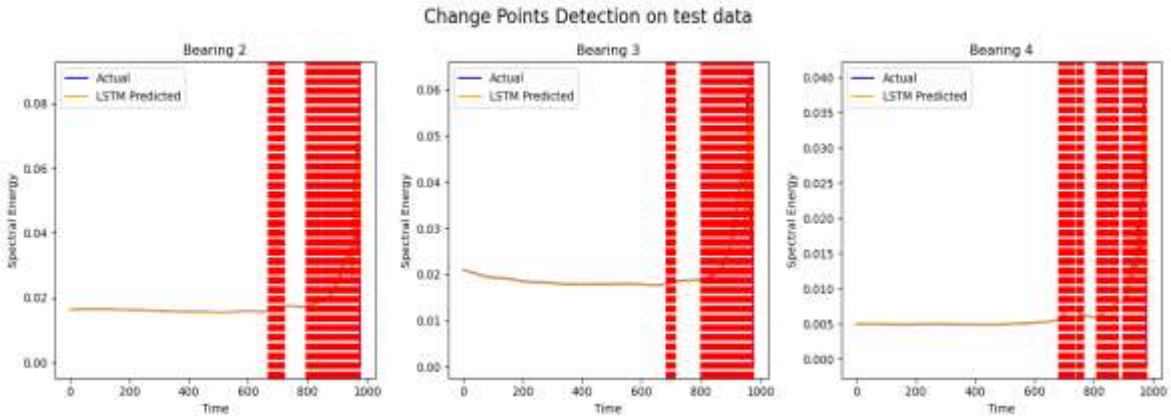


Figure 64: detected change points on test data

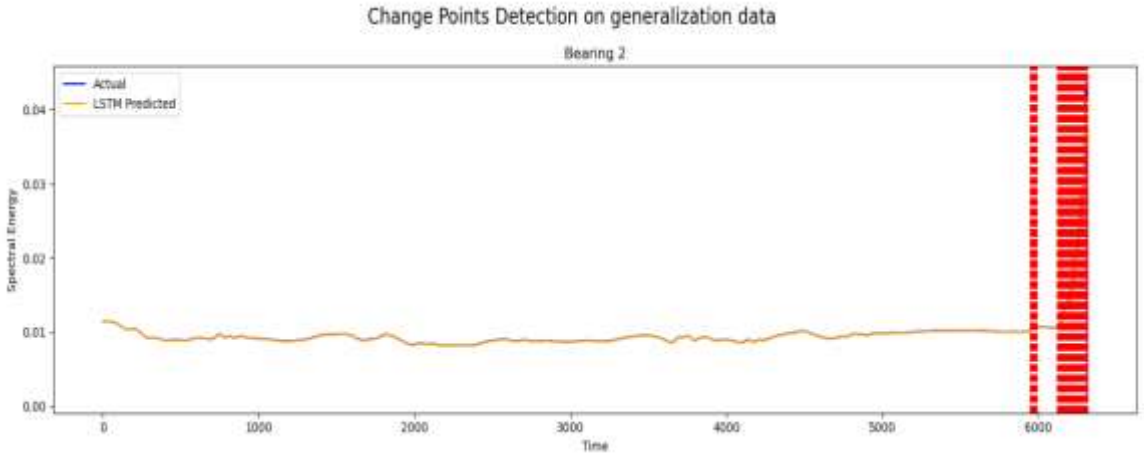


Figure 65: detected change points on generalization data (bearing 2)

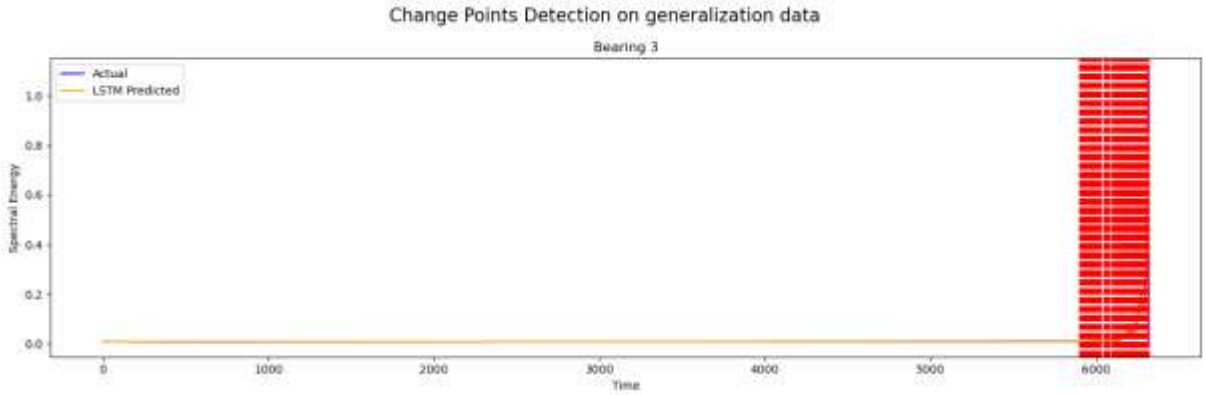


Figure 66: detected change points on generalization data (bearing 3)

II.4. RUL prediction

After detecting the change points, our objective is to predict the RUL of the bearing to enable proactive maintenance scheduling. To achieve this, we trained various models, including single approaches which encompass both traditional ML and DL models, as well as hybrid approaches using DL models. The training was conducted using data from bearing 1 of the second test after feature selection and adding the RUL feature, which was calculated by subtracting the actual index from the last index, followed by normalization. For testing, we excluded every sixth data point to evaluate the model's ability to predict the RUL across different stages as presented in **figure 67**. The results presented in **table 06** show that the **LSTM-CNN** hybrid model outperforms the other models (both hybrid and single), achieving a **RMSE of 0.001 and a MAE of 0.0007.** The results of the hybrid models are shown in **Figures 66, 68 and 69.**

Note: The graphs representing the estimated RUL use the x-axis to denote time in minutes where each point represent 10 min, and the y-axis to show the predicted RUL in the frequency domain. In these graphs, the blue color represents the real RUL, while the orange color indicates the estimated RUL.

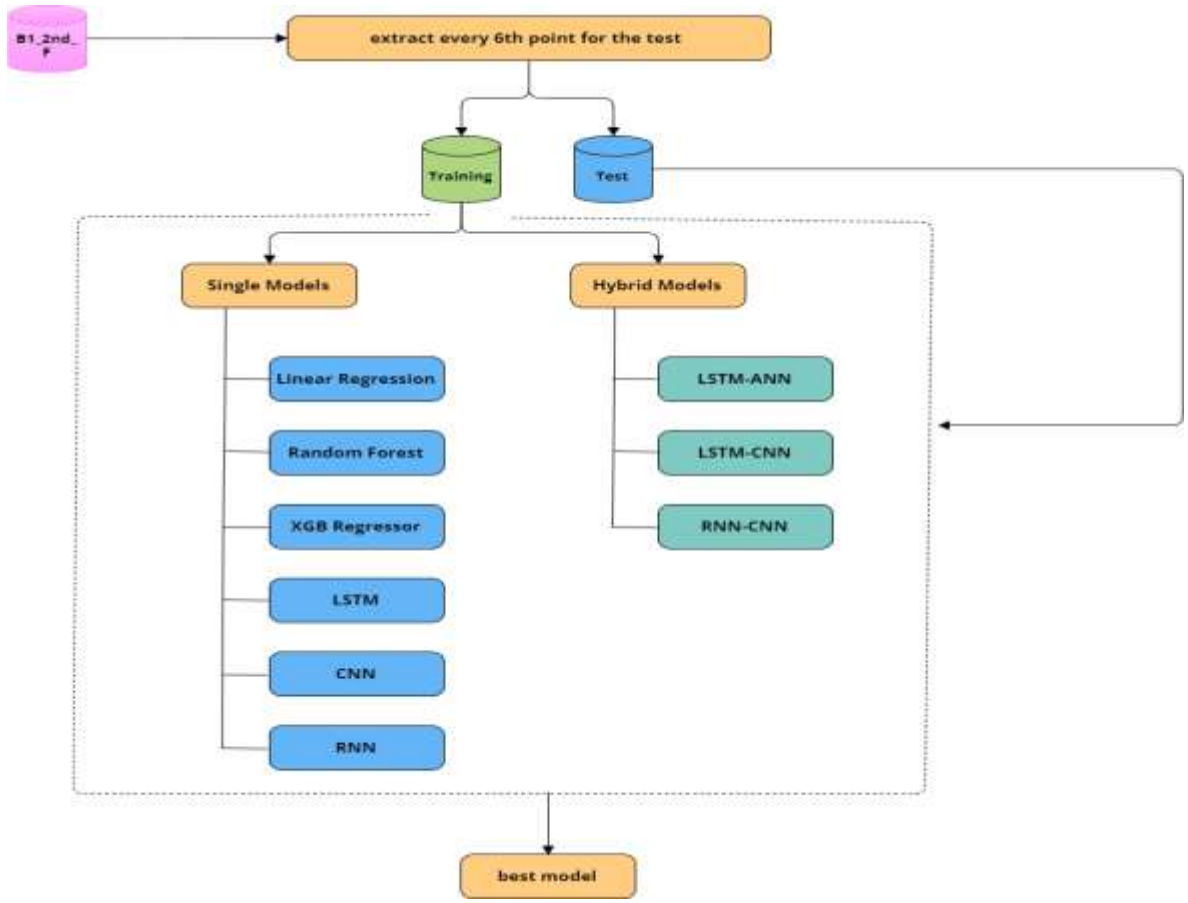


Figure 67: RUL prediction process

	Model	MAE	RMSE
Single models	Linear Regression	0.21	0.25
	Random Forest	0.45	0.51
	XGB Regressor	0.35	0.39
	LSTM	0.27	0.07
	CNN	0.25	0.32
	RNN	0.26	0.38
Hybrid models	LSTM-ANN	0.02	0.06
	LSTM-CNN	**0.0007**	**0.001**
	RNN-CNN	0.006	0.016

Table 6: RUL prediction results

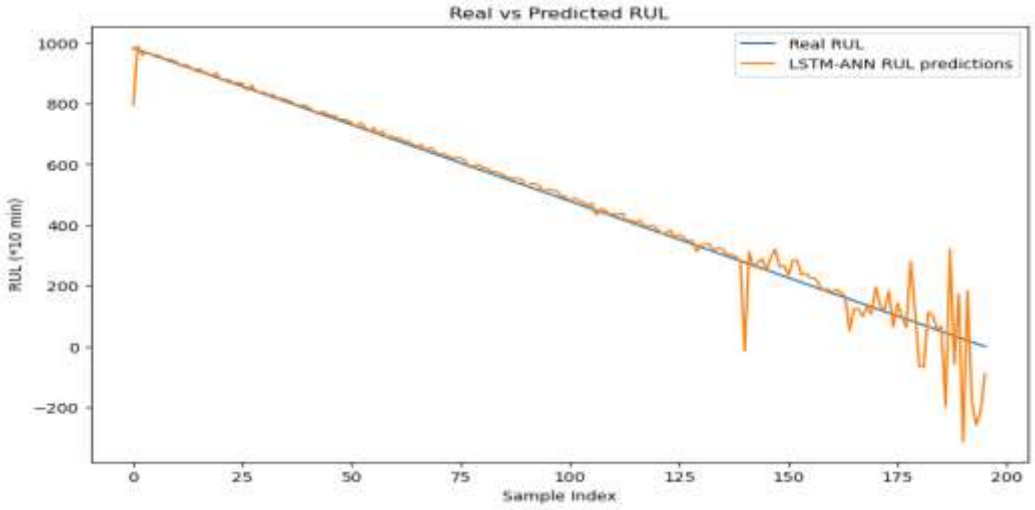


Figure 68: LSTM-ANN RUL predictions

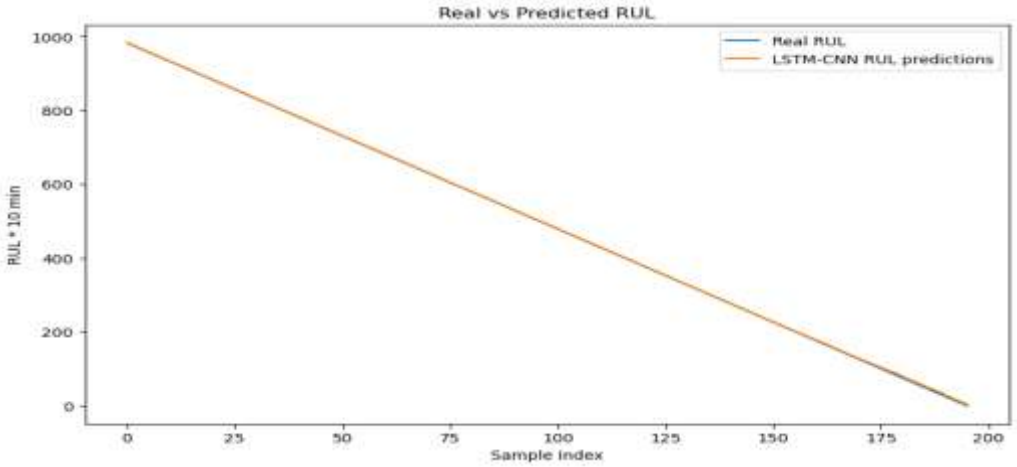


Figure 69: LSTM-CNN RUL predictions

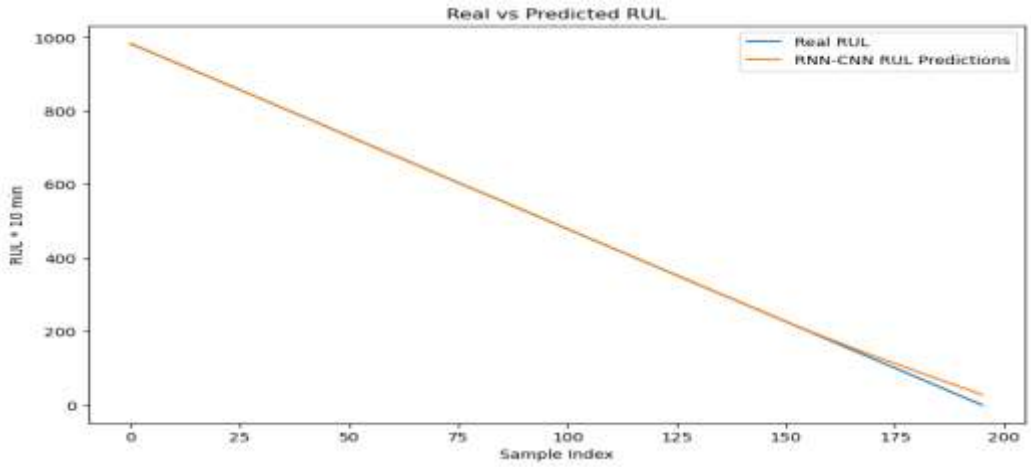


Figure 70: RNN-CNN RUL predictions

III. Discussion

According to our experiment, the single model LSTM demonstrated the best results for next-state prediction with generalization and denoised data in the frequency domain. Specifically, it achieved an **RMSE of 0.0005, MAE of 0.00004, and an R^2 of 0.93**. Notably, most models demonstrated improved performance in the frequency domain when the data was denoised using wavelet transform. Consequently, the experiment proceeded in the frequency domain with denoised data.

For change point detection, a thresholding technique was employed by examining the difference between the predicted state at $t + 1$ and the actual state at t. A threshold of $17 * 10^{-6}$ yielded good results on both test and generalization data.

Regarding RUL prediction, the hybrid model LSTM-CNN outperformed other models, achieving an **RMSE of 0.001 and an MAE of 0.0007**. This performance is notably better compared to the results of (Ding H. et al. (2021)) in their study on RUL prediction for bearings using deep neural networks. Their best result for the same dataset, using a DCNN with a denoising method based on the 3-sigma criterion in the time-frequency domain, achieved an **RMSE of 0.00525**.

In summary, our findings underscore the efficacy of LSTM models for next-state prediction and the superiority of the LSTM-CNN hybrid model for RUL prediction. The application of wavelet transform for denoising and the focus on the frequency domain have proven to be beneficial in enhancing model performance. Our approach to change point detection also demonstrated robustness and reliability, further contributing to the field of PdM.

IV. Conclusion

In this chapter, we investigated the performance of hybrid and single models for next state prediction and RUL estimation. We then used change point detection by calculating the difference between the predicted state at t+1 and the real state at □, setting a threshold to distinguish between normal and abnormal behavior.

For RUL prediction, the hybrid models proved to be the most effective. These results highlight the potential of advanced machine learning models in PdM, with LSTM excelling in next state prediction and LSTM-CNN in RUL estimation.

General Conclusion:

The objective of this final project was to develop a system that supports PdM activities within Industry 4.0. Our PdM framework integrates advanced ML models to improve predictive accuracy and reliability. We focused on next-state prediction using both single and hybrid model approaches, change point detection with multiple thresholds, and RUL prediction across time and frequency domains.

Through experimentation, we found that vibration data, denoised with wavelet transforms and processed in the frequency domain, consistently yielded superior results across most models. Specifically, our single model LSTM achieved impressive metrics with an extremely low RMSE, an exceptionally small MAE, and an R^2 of 0.93 for next-state prediction. Furthermore, our hybrid LSTM-CNN model outperformed other approaches in RUL prediction, achieving a remarkably low RMSE and a very small MAE. The results are good in both test and generalization scenarios, demonstrating robust performance with datasets from the same experiment used for testing and datasets from another experiment used for generalization.

Moving forward, future research should prioritize the development of seamless data integration frameworks, reducing computational burdens through parallel computing and efficient algorithms. Designing scalable PdM solutions that are adaptable to diverse industrial contexts is crucial. Our perspective also involves establishing anomaly localization for specific bearing failure identification, integrating with existing plant data infrastructure for real-time monitoring, and designing intuitive interfaces to improve interpretation and maintenance decision-making.

Continued advancements in these areas will be pivotal in realizing PdM's potential for enhanced reliability, reduced downtime, and improved operational efficiency.

Bibliography:

Amruthnath, N., & Gupta, T. (2018). A research study on unsupervised machine learning algorithms for early fault detection in predictive maintenance. *2018 5th International Conference on Industrial Engineering and Applications (ICIEA)*, 355–361.

Arslan, S. (2023). Gated recurrent unit network-based fuzzy time series forecasting model. *Afyon Kocatepe Üniversitesi Fen Ve Mühendislik Bilimleri Dergisi*, *23*(3), 677–692.

Bengtsson, M. (2004). *Condition based maintenance systems: An investigation of technical constituents and organizational aspects*. Mälardalen University Eskilstuna, Sweden.

Bnou, K., Raghay, S., & Hakim, A. (2020). A wavelet denoising approach based on unsupervised learning model. *EURASIP Journal on Advances in Signal Processing*, *2020*(1), 36. https://doi.org/10.1186/s13634-020-00693-4

Cheng, J. C., Chen, W., Chen, K., & Wang, Q. (2020). Data-driven predictive maintenance planning framework for MEP components based on BIM and IoT using machine learning algorithms. *Automation in Construction*, *112*, 103087.

Choi, D.-I., & Ok, C.-S. (2015). A New Metric for Evaluation of Forecasting Methods: Weighted Absolute and Cumulative Forecast Error. *Journal of the Society of Korea Industrial and Systems Engineering*, *38*(3), 159–168.

Dey, I. (2023). Wavelet Transform for IoT--A Signal Processing Perspective. *Authorea Preprints*.

Ding, H., Yang, L., Cheng, Z., & Yang, Z. (2021). A remaining useful life prediction method for bearing based on deep neural networks. *Measurement*, *172*, 108878.

Downey, C., Hefny, A., Boots, B., Gordon, G. J., & Li, B. (2017). Predictive state recurrent neural networks. *Advances in Neural Information Processing Systems*, *30*.

Elmaz, F., Eyckerman, R., Casteels, W., Latré, S., & Hellinckx, P. (2021). CNN-LSTM architecture for predictive indoor temperature modeling. *Building and Environment*, *206*, 108327.

Falamarzi, A., Moridpour, S., Nazem, M., & Cheraghi, S. (2019). Prediction of tram track gauge deviation using artificial neural network and support vector regression. *Australian Journal of Civil Engineering*, *17*(1), 63–71.

Fattah, J., Ezzine, L., Aman, Z., El Moussami, H., & Lachhab, A. (2018). Forecasting of demand using ARIMA model. *International Journal of Engineering Business Management*, *10*, 1847979018808673.

Feasel, K. (2022). Change Point Detection. In *Finding Ghosts in Your Data: Anomaly Detection Techniques with Examples in Python* (pp. 247–261). Apress. https://doi.org/10.1007/978-1-4842-8870-2_14

Feng, W., Guan, N., Li, Y., Zhang, X., & Luo, Z. (2017). Audio visual speech recognition with multimodal recurrent neural networks. *2017 International Joint Conference on Neural Networks (IJCNN)*, 681–688.

Francis, F., & Mohan, M. (2019). Arima model based real time trend analysis for predictive maintenance. *2019 3rd International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, 735–739.

Géron, A. (2022). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly Media, Inc.

Ghemari, Z., Belkhiri, S., & Saad S. (2022). *Improvement of the relative sensitivity for obtaining a high performance piezoelectric sensor*.

Glock, A.-C., Sobieczky, F., Fürnkranz, J., Filzmoser, P., & Jech, M. (2024). Predictive change point detection for heterogeneous data. *Neural Computing and Applications*, 1–26.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.

Gu, H., Wang, Y., Hong, S., & Gui, G. (2019). Blind channel identification aided generalized automatic modulation recognition based on deep learning. *IEEE Access*, *7*, 110722–110729.

Hesser, D. F., & Markert, B. (2019). Tool wear monitoring of a retrofitted CNC milling machine using artificial neural networks. *Manufacturing Letters*, *19*, 1–4.

Hodson, T. O. (2022). Root mean square error (RMSE) or mean absolute error (MAE): When to use them or not. *Geoscientific Model Development Discussions*, *2022*, 1–10.

Hu, J., Jia, F., & Liu, W. (2023). Application of Fast Fourier Transform. *Highlights in Science, Engineering and Technology*, *38*, 590–597. https://doi.org/10.54097/hset.v38i.5888

Im, D. J., Kwak, I., & Branson, K. (2020). Evaluation metrics for behaviour modeling. *arXiv Preprint arXiv:2007.12298*.

Jain, P. H., & Bhosle, S. P. (2022). Analysis of vibration signals caused by ball bearing defects using time-domain statistical indicators. *International Journal of Advanced Technology and Engineering Exploration*, *9*(90), 700.

Jimenez, J. J. M., Schwartz, S., Vingerhoeds, R., Grabot, B., & Salaün, M. (2020). Towards multi-model approaches to predictive maintenance: A systematic literature survey on diagnostics and prognostics. *Journal of Manufacturing Systems*, *56*, 539–557.

Kanawaday, A., & Sane, A. (2017). Machine learning for predictive maintenance of industrial machines using IoT sensor data. *2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, 87–90.

Kanungo, A., Mittal, M., & Dewan, L. (2020). Comparison of Haar and Daubechies wavelet based denoising for speed control of DC motor. *2020 First IEEE International Conference on Measurement, Instrumentation, Control and Automation (ICMICA)*, 1–4.

Keras: Deep Learning for humans. (n.d.). Retrieved June 18, 2024, from https://keras.io/

Kim, J. H. (2017). A review of cyber-physical system research relevant to the emerging IT trends: Industry 4.0, IoT, big data, and cloud computing. *Journal of Industrial Integration and Management*, *2*(03), 1750011.

Kulkarni, A. R., Shivananda, A., Kulkarni, A., & Krishnan, V. A. (2022). Deep Learning–based Time Series Forecasting. In *Time Series Algorithms Recipes: Implement Machine Learning and Deep Learning Techniques with Python* (pp. 127–168). Springer.

Kulkarni, A. R., Shivananda, A., Kulkarni, A., & Krishnan, V. A. (2022). Getting Started with Time Series. In *Time Series Algorithms Recipes: Implement Machine Learning and Deep Learning Techniques with Python* (pp. 1–31). Springer.

Lee, G., Gommers, R., Waselewski, F., Wohlfahrt, K., & O'Leary, A. (2019). PyWavelets: A Python package for wavelet analysis. *Journal of Open Source Software*, *4*(36), 1237. https://doi.org/10.21105/joss.01237

Lee, J., Qiu, H., Yu, G., & Lin, J. (2007). IMS,"University of Cincinnati."Bearing Data Set", NASA Ames Prognostics Data Repository. *NASA Ames Research Center, Moffett Field, CA*.

Lin, Y.-H., Guan, L.-X., Chang, L., & Zio, E. (2023). A semi-supervised deep hybrid multi-task model for RUL prediction. *IEEE Transactions on Instrumentation and Measurement*.

Matplotlib—Visualization with Python. (n.d.). Retrieved June 18, 2024, from https://matplotlib.org/

Maximum-to-minimum difference—MATLAB peak2peak. (n.d.). Retrieved June 19, 2024, from https://www.mathworks.com/help/signal/ref/peak2peak.html

Meesublak, K., & Klinsukont, T. (2020). A cyber-physical system approach for predictive maintenance. *2020 Ieee International Conference on Smart Internet of Things (Smartiot)*, 337–341.

Miles, J. (2005). R-squared, adjusted R-squared. *Encyclopedia of Statistics in Behavioral Science*.

Mo, H., & Iacca, G. (2023). Evolutionary neural architecture search on transformers for RUL prediction. *Materials and Manufacturing Processes*, *38*(15), 1881–1898.

Namuduri, S., Narayanan, B. N., Davuluru, V. S. P., Burton, L., & Bhansali, S. (2020). Deep learning methods for sensor based predictive maintenance and future perspectives for electrochemical sensors. *Journal of The Electrochemical Society*, *167*(3), 037552.

Nguyen, H. D., Tran, K. P., Zeng, X., Koehl, L., Castagliola, P., & Bruniaux, P. (2019). Industrial Internet of Things, big data, and artificial intelligence in the smart factory: A survey and perspective. *ISSAT International Conference on Data Science in Business, Finance and Industry*, 72–76.

NumPy -. (n.d.). Retrieved June 18, 2024, from https://numpy.org/

pandas—Python Data Analysis Library. (n.d.). Retrieved June 18, 2024, from https://pandas.pydata.org/

Park, C., & Ding, Y. (2021). Change Point Detection. In *Data Science for Nano Image Analysis* (pp. 241–275). Springer International Publishing. https://doi.org/10.1007/978-3-030-72822-9_9

Petzold, J., Bagci, F., Trumler, W., & Ungerer, T. (2006). Hybrid predictors for next location prediction. *Ubiquitous Intelligence and Computing: Third International Conference, UIC 2006, Wuhan, China, September 3-6, 2006. Proceedings 3*, 125–134.

Poór, P., Ženíšek, D., & Basl, J. (2019). Historical overview of maintenance management strategies: Development from breakdown maintenance to predictive maintenance in accordance with four industrial revolutions. *Proceedings of the International Conference on Industrial Engineering and Operations Management, Pilsen, Czech Republic*, 23–26.

Qiu, H., Lee, J., Lin, J., & Yu, G. (2006). Wavelet filter-based weak signature detection method and its application on rolling element bearing prognostics. *Journal of Sound and Vibration*, *289*(4–5), 1066–1090.

Remadna, I., Terrissa, L. S., Al Masry, Z., & Zerhouni, N. (2022). RUL prediction using a fusion of attention-based convolutional variational autoencoder and ensemble learning classifier. *IEEE Transactions on Reliability*, *72*(1), 106–124.

Rittweger, J. (2020). *Manual of vibration exercise and vibration therapy*. Springer.

Rojek, I. (2010). Hybrid neural networks as prediction models. *Artifical Intelligence and Soft Computing: 10th International Conference, ICAISC 2010, Zakopane, Poland, June 13-17, 2010, Part II 10*, 88–95.

Rousopoulou, V., Nizamis, A., Giugliano, L., Haigh, P., Martins, L., Ioannidis, D., & Tzovaras, D. (2019). Data analytics towards predictive maintenance for industrial ovens: A case study based on data analysis of various sensors data. *Advanced Information Systems Engineering Workshops: CAiSE 2019 International Workshops, Rome, Italy, June 3-7, 2019, Proceedings 31*, 83–94.

Rustamov, Z., Rustamov, J., Zaki, N., Turaev, S., Sultana, M. S., Tan, J. Y., & Balakrishnan, V. (2023). *Enhancing Cardiovascular Disease Prediction: A Domain Knowledge-Based Feature Selection and Stacked Ensemble Machine Learning Approach*.

Scalabrini Sampaio, G., Vallim Filho, A. R. de A., Santos da Silva, L., & Augusto da Silva, L. (2019). Prediction of motor failure time using an artificial neural network. *Sensors*, *19*(19), 4342.

Scikit-learn: Machine learning in Python—Scikit-learn 1.5.0 documentation. (n.d.). Retrieved June 18, 2024, from https://scikit-learn.org/stable/

Sen, R., & Das, S. (2023). Time series. In *Computational Finance with R. pp 279–292*. https://link.springer.com/chapter/10.1007/978-981-19-2008-0_18.

Shi, J., Gao, J., & Xiang, S. (2023). Adaptively Lightweight Spatiotemporal Information-Extraction-Operator-Based DL Method for Aero-Engine RUL Prediction. *Sensors*, *23*(13), 6163.

Silvestrin, L. P., Hoogendoorn, M., & Koole, G. (2019). A Comparative Study of State-of-the-Art Machine Learning Algorithms for Predictive Maintenance. *SSCI*, 760–767.

Smith, C., Akujuobi, C. M., Hamory, P., & Kloesel, K. (2007). An approach to vibration analysis using wavelets in an application of aircraft health monitoring. *Mechanical Systems and Signal Processing*, *21*(3), 1255–1272.

Sonia, M. S., & Sumathi, S. (2022). A Comparative Analysis of Wavelet Transforms for Denoising MRI Brain Images. *2022 International Conference on Power, Energy, Control and Transmission Systems (ICPECTS)*, 1–5.

Susto, G. A., Schirru, A., Pampuri, S., McLoone, S., & Beghi, A. (2014). Machine learning for predictive maintenance: A multiple classifier approach. *IEEE Transactions on Industrial Informatics*, *11*(3), 812–820.

Tahir, M. M., Badshah, S., Hussain, A., & Khattak, M. A. (2018). Extracting accurate time domain features from vibration signals for reliable classification of bearing faults. *International Journal of Advanced and Applied Sciences*, *5*(1), 156–163.

TensorFlow. (n.d.). Retrieved June 18, 2024, from https://www.tensorflow.org/?hl=fr

Upadhyay, R. K., & Kumaraswamidhas, L. A. (2018). Bearing failure issues and corrective measures through surface engineering. In *Handbook of Materials Failure Analysis* (pp. 209–233). Elsevier.

Varshney, R. P., & Sharma, D. K. (2023). A Novel Deep Learning Framework with Multi-Stage Feature Selection for Time Series Forecasting. *2023 10th International Conference on Computing for Sustainable Global Development (INDIACom)*, 200–205.

Violos, J., Tsanakas, S., Androutsopoulou, M., Palaiokrassas, G., & Varvarigou, T. (2020). Next position prediction using LSTM neural networks. *11th Hellenic Conference on Artificial Intelligence*, 232–240.

Von Birgelen, A., Buratti, D., Mager, J., & Niggemann, O. (2018). Self-organizing maps for anomaly localization and predictive maintenance in cyber-physical production systems. *Procedia Cirp*, *72*, 480–485.

Wang, Y., & Zhao, Y. (2023). Three-stage feature selection approach for deep learning-based RUL prediction methods. *Quality and Reliability Engineering International*, *39*(4), 1223–1247.

Wang, Z., Li, Y., Dong, L., Li, Y., & Du, W. (2023). A RUL prediction of bearing using fusion network through feature cross weighting. *Measurement Science and Technology*, *34*(10), 105908.

Wang, Z., Wang, Y., & Wang, Y. (2021). Implanting Domain Knowledge into Feature Selection for Effective Outlier Detection in Network Traffic Data. *2021 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/IOP/SCI)*, 115–122.

What is predictive maintenance? (Benefits and examples) - Sensorfact - smart monitoring for industry. (n.d.). *Sensorfact*. Retrieved June 19, 2024, from https://www.sensorfact.eu/blog/what-is-predictive-maintenance-benefits-and-examples/

Wibawa, A. D., Fatih, N., Pamungkas, Y., Pratiwi, M., & Ramadhani, P. A. (2022). Time and Frequency Domain Feature Selection Using Mutual Information for EEG-based Emotion Recognition. *2022 9th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)*, 19–24.

Xiang, S., Huang, D., & Li, X. (2018). A generalized predictive framework for data driven prognostics and diagnostics using machine logs. *TENCON 2018-2018 IEEE Region 10 Conference*, 0695–0700.

Yu, Z., Lei, N., Mo, Y., Xu, X., Li, X., & Huang, B. (2024). Feature Extraction Based on Self-Supervised Learning for Remaining Useful Life Prediction. *Journal of Computing and Information Science in Engineering*, *24*(2).

Zhang, M., Hua, Y., Chen, C., Chu, C., & Zhang, X. (2022). Research on Feature Extraction Method of Fiber Bragg Grating Vibration Monitoring Based on FFT. *Journal of Engineering Research and Sciences*, *1*(7), 44–47. https://doi.org/10.55708/js0107007

Zheng, L. (2017). *Feature Grouping-based Feature Selection*. Aberystwyth University.