Study of Deep Learning Architectures for Bearing Fault Diagnosis Using STFT Spectrograms

by

Rajshri Simhadri

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2025

© Rajshri Simhadri 2025

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

This thesis presents a comprehensive study on vibration-based bearing fault type and severity-level detection, this process plays a critical role in predictive maintenance for industrial systems. The proposed framework combines signal processing and image-based representations derived from short-time Fourier transform (STFT) spectrograms to classify ten fault classes encompassing various fault types and severities. Among the evaluated architectures, the pretrained ImageNet model XceptionNet-71, when fine-tuned on grayscale STFTs, achieved the best overall performance, attaining a macro F1-score of 0.9979 and a mean ROC–AUC of 0.99 across all classes. This single-channel model demonstrated superior class separability compared to both flattened 1D STFT inputs and three-channel spectrograms. Which confirms that spectrogram-based representations combined with pretrained convolutional backbones are well-suited for bearing fault diagnosis and real-time deployment.

While prior studies on the CWRU dataset have improved bearing fault classification through handcrafted features, lightweight CNNs, and transformer-based models, they often suffer from dataset leakage and lack systematic benchmarking. This work addresses these gaps through a unified and reproducible framework that compares 1D and 2D CNNs, extends Delta-STFT into a cross-resolution multi-channel representation, and conducts a comprehensive evaluation to classify safe versus unsafe misclassifications, bridging the gap between high accuracy and practical deployability.

Acknowledgements

I would like to express my sincere gratitude to my supervisors, Prof. Kshirasagar Naik and Prof. Mahesh Pandey, for their unwavering support, insightful guidance, and encouragement throughout this entire journey. Their expertise and thoughtful feedback have been instrumental in shaping my research and expanding my academic horizons.

I am especially grateful to my parents, Simhadri Venkateswara Rao and Simhadri Sumathi, for their unconditional love, encouragement, and belief in me. Their strength and sacrifices have been my constant source of motivation throughout this journey. I would also like to thank my beloved brothers Sai Agathiya and Vihaan Sai Ram and my grandparents. I sincerely thank God for granting me the strength and perseverance to complete this work. Additionally, I am thankful to my friends for their friendship and encouragement, which have helped me stay grounded and motivated.

Dedication

This thesis is dedicated to my parents Venkateswara Rao Simhadri and Sumathi Simhadri and my Professors: Prof Kshirasagar Naik and Prof. Mahesh Pandey who belived in my potential and were unwavering support pillars for my strength.

Without their belief in me and constant support, I would not have had the courage and resilience to relocate to a foreign country and start life anew, nor to seize the opportunity to pursue my education at one of the world's most prestigious institutions.

My supervisors and parents were my unwavering pillars of strength and inspiration.

Table of Contents

Author's Declaration ii

Abstract iii

Acknowledgements iv

Dedication v

List of Figures x

List of Tables xiii

1 Introduction 1

1.1 Motivation 1

1.2 Problem Formulation 2

1.3 Research Contributions 3

1.4 Outline of Thesis 4

2 Background and Related Work 6

2.1 Benchmark Datasets for Bearing Fault Diagnosis 6

2.2 Machine Learning in Rotating Machinery Condition Monitoring 9

2.3 Machine Learning Paradigms for Bearing Fault Diagnosis 10

2.4 Time–Frequency Analysis and Spectrogram-Based Features 12
2.5 Deep Learning Architectures for Bearing Fault Diagnosis 13
2.5.1 Convolutional Neural Networks for Vibration Signals 14
2.5.2 Hybrid Architectures (CNN–LSTM, Transformer-based Models, etc.) 14
2.6 Research Gaps 15

3 Problem Description **21**
3.1 System Model 23
3.2 Fault Model and Signal Representation 25
3.2.1 Representative Signal Characteristics 31
3.3 Objective: 40

4 Methodology **41**
4.1 Dataset Preparation 41
4.2 Proposed Methodology 50
4.3 Feature Extraction 50
4.3.1 Single-Channel Spectrograms (2D-CNN) 53
4.4 Classification Model Architecture 57
4.4.1 XceptionNet71 for Single-Channel STFT Features 58

5 Experiments and Results **61**
5.1 Loss Function for Experiments 62
5.2 Evaluation Strategy for Experiments 63
5.3 Evaluation Metrics 64
5.4 Experimental Setup 67
5.5 Flattened STFT Features 69
5.5.1 Per-Fold Validation Macro F1-Score – Dilated 1D CNN 71
5.5.2 Confusion Matrix Analysis of Dilated 1D-CNN 73

5.5.3 ROC-AUC Curve Analysis of Dilated 1D CNN 76
5.5.4 Training Dynamics of Dilated 1D CNN 78
5.6 Single-Channel Spectrograms 80
5.6.1 Per-Fold Validation Macro F1-Score of Single-Channel Spectrograms 80
5.6.2 Confusion Matrix Analysis of XceptionNet71 82
5.6.3 ROC-AUC Curve Analysis of XceptionNet71 86
5.6.4 Training Dynamics of XceptionNet71 88
5.7 Three-Channel Spectrograms 89
5.7.1 Per-Fold Validation Macro F1-Score of Three-Channel Spectrograms 92
5.7.2 Confusion Matrix Analysis of EfficientNetV2-M 94
5.7.3 ROC-AUC Curve Analysis of EfficientNetV2-M 97
5.7.4 Training Dynamics of EfficientNetV2-M 98
5.8 Results and Comparative Analysis 100
5.8.1 Model Evaluation and Comparative Analysis 100
5.8.2 Confusion Matrices and Comparative Analysis 101
5.8.3 ROC-AUC and Comparative Analysis 103
5.8.4 Comparative Analysis of Training Curves 103
5.8.5 Discussion of Results 103

6 Conclusions and Future Work **105**
6.1 Conclusions 105
6.2 Limitations 106
6.3 Future Work 106

References **107**

Appendices **113**

A List of Notations **113**

B Experiment 1: Dilated 1D CNN on Flattened STFT Features **115**

C Experiment 2: XceptionNet-71 on Single-Channel STFT Features **117**

D Experiment 3: EfficientNetV2-M on Multi-Channel Delta-STFT Features **125**

E Live Demonstration **129**

List of Figures

3.1 System Model Diagram 23

3.2 Illustration of a typical ball bearing showing the inner race, outer race, rolling elements (balls), cage, and shaft. 27

3.3 Defects are marked with small notches on the ball : (a) Electrical Discharge Machining (EDM) (b) Ball fault, (c) Outer race fault, (d) Inner race fault . 29

3.4 Time-domain representations of ball faults compared with a healthy condition. 32

3.5 Frequency-domain representations of ball faults compared with a healthy condition. 33

3.6 Time-domain representations of inner-race faults compared with a healthy condition. 34

3.7 Frequency-domain representations of inner-race faults compared with a healthy condition. 35

3.8 Time-domain representations of outer-race faults compared with a healthy condition. 36

3.9 Frequency-domain representations of outer-race faults compared with a healthy condition. 37

3.10 STFT spectrograms of ball faults at different severities compared with a healthy baseline. Fault conditions show localized energy concentrations near the ball-pass frequency (236 Hz), absent in the healthy case. 38

3.11 STFT spectrograms of inner-race faults compared with a healthy baseline. Inner-race defects produce strong, shaft-speed-modulated harmonics, resulting in sharper spectral bands than those observed in ball faults. 39

3.12 STFT spectrograms of outer-race faults compared with a healthy baseline. Outer-race defects exhibit periodic bursts of energy at the outer race ball-pass frequency, distinct from the modulation patterns of inner-race faults. . 40

4.1 Relative class distribution across all fault types and the healthy condition. 46
4.2 Overview of the data preparation pipeline. 49
4.3 Time-domain vibration signal 51
4.4 Frequency-domain signal (FFT) 51
4.5 Feature extraction stage following data preparation. 53
4.6 Single-Channel Spectrogram (2D-CNN). 54
4.7 Xception71 architecture adapted for single-channel spectrogram input. . . . 60

5.1 Workflow of the Experiments 69
5.2 Experiment 1 pipeline: Flattened STFT Vectors with 1D-CNN. 71
5.3 Macro-F1 Performance Across Folds for Dilated 1D CNN 72
5.4 Confusion Matrix Dilated 1D CNN 73
5.5 Multi-Class ROC Curve for Dilated 1D CNN 77
5.6 Dilated 1D CNN loss vs epochs 79
5.7 Dilated 1D CNN accuracy vs epochs 79
5.8 Macro-F1 Performance Across Folds for the XceptionNet71 81
5.9 Confusion Matrix for XceptionNet71 82
5.10 Multi-Class ROC Curve for XceptionNet 71 86
5.11 XceptionNet71 loss vs epochs 88
5.12 XceptionNet71 accuracy vs epochs 88
5.13 Delta-STFT visualizations from a ball fault signal: (Left) short-to-medium differences, (Center) short-to-long differences, (Right) medium-to-long differences. These maps emphasize how fault signatures evolve across spectral resolutions, highlighting transient bursts that appear sharp in short windows but blurred in long windows. 91
5.14 Experiment 3 pipeline: Multi-Channel Spectrograms (Delta-STFT stacks with 2D-CNN). 91
5.15 Macro-F1 Performance Across Folds for the XceptionNet71 93
5.16 Confusion Matrix for EfficientNetV2-M 94

5.17 Multi-Class ROC Curve for Efficient Net V2-M 97
5.18 EfficientNetV2-M loss vs epochs 99
5.19 EfficientNetV2-M accuracy vs epochs 99
5.20 Per Class Performance 102

List of Tables

2.1 Comparative Summary of Machine Learning Approaches for Bearing Fault Diagnosis . . . 17

2.2 Comparison of Representative Studies on Bearing Fault Diagnosis Using the CWRU Dataset . . . 18

3.1 Motor Specifications Used in the Case Western Reserve University (CWRU) Test Setup . . . 24

3.2 Fault type and corresponding fault severity levels. . . . 30

4.1 Sample metadata from the CWRU bearing dataset showing vibration signal arrays for Bearing X282 (Drive End, Fan End, and Base Acceleration sensors). 42

4.2 Mapping of fault types and severities in the CWRU dataset. . . . 43

4.3 Original distribution of samples across fault types and diameters in the CWRU dataset. . . . 45

4.4 Final class distribution after applying all augmentation methods. . . . 49

5.1 Class-wise distribution of samples with train-test split (80:20) . . . 64

5.2 Hyperparameter Settings for Each Experiment . . . 68

5.3 Per-Fold Validation Macro F1-Score – Dilated 1D CNN . . . 72

5.4 Class-wise data distribution and confusion components for Dilated 1D-CNN 74

5.5 Per-class evaluation metrics for Dilated 1D-CNN (Test Set) . . . 75

5.6 Tuple-based representation of misclassifications in the Dilated 1D-CNN experiment. . . . 76

5.7 Per-Fold Validation Macro F1-Score – XceptionNet71 81
5.8 Per-class confusion components for XceptionNet71 (Test Set) 83
5.9 Per-class evaluation metrics for Xception-71 (Test Set) 84
5.10 Tuple-based representation of misclassifications in the XceptionNet-71 experiment. 85
5.11 Per-Fold Validation Macro F1-Score – EfficientNetV2-M 92
5.12 Per-class confusion components for EfficientNetV2-M (Test Set) 95
5.13 Per-class evaluation metrics for EfficientNetV2-M (Test Set) 95
5.14 Tuple-based representation of misclassifications in the EfficientNetV2-M experiment. 96
5.15 Cross-Validation and Final Test Performance Summary 100

A.1 List of Notations Used in This Thesis 113

Chapter 1

Introduction

1.1 Motivation

Rotating machinery forms the backbone of modern industry, powering applications from manufacturing plants and wind farms to railway traction and nuclear power stations. Among their components, rolling element bearings are especially critical: they support radial and axial loads, minimize friction, and enable smooth motion. Yet, despite their simple structure, bearings are highly susceptible to progressive degradation. Industry reports suggest that bearing faults account for 40–50% of all rotating machinery failures [1]. In safety-critical sectors such as aerospace, energy, and transportation, undetected bearing faults can escalate into catastrophic breakdowns, safety hazards, and costly downtime.

With the rise of Industry 4.0, which integrates the Industrial Internet of Things (IIoT) platforms, cyber-physical systems, and advanced analytics [2], there is a growing demand for automated, real-time, and reliable fault diagnosis systems [3].

Vibration analysis is one of the most effective bearing monitoring techniques, as it captures transient and localized fault behavior [4]. However, practical deployment faces challenges such as:

(a) Non-stationary signals under varying speeds and loads

(b) Weak signatures from early-stage defects

(c) Class imbalance in real-world datasets

(d) Large-scale, high-frequency vibration data requiring efficient processing.

Traditional methods relying on hand-crafted statistical features (RMS, kurtosis, envelope spectra) and classifiers such as SVMs or decision trees [5] lack scalability and robustness. Deep learning approaches, particularly CNNs operating on time–frequency representations like STFT spectrograms [6], have shown promise on benchmark datasets such as CWRU. Yet, most studies remain restricted to controlled laboratory settings with fixed motor speeds, balanced data, and shared components between train and test sets, leading to optimistic but non-generalizable performance [7].

This work is motivated by the need to design deep learning-based bearing fault diagnosis systems that are accurate, generalizable, and computationally efficient. The goals of this study are to:

(a) Enable classification under variable operating conditions

(b) Mitigate data imbalance with augmentation and weighted loss functions and diverse sampling strategies

(c) Assess models on based on algorithmic complexity for real-time deployment

1.2 Problem Formulation

Reliable operation of rotating machinery is critical in modern industrial systems, where failures of rolling element bearings are among the leading causes of downtime. With the rise of Industry 4.0 and the Industrial Internet of Things (IIoT), vibration-based fault diagnosis has become a central challenge, requiring automated, real-time, and data-driven methods.

In this thesis, the bearing fault classification problem is formulated as a supervised multi-class classification task, where the objective is to identify both the (i) type and (ii) severity of faults from vibration signal. Each class corresponds to a specific fault *type–severity* combination, including the healthy state.

To effectively train a learning model 'M' for this classification task, it is essential to represent the of the bearing system through measurable signals obtained from sensors in the physical domain.

The sensor data from the physical domain of the system is initially collected in the time-series domain. Let the healthy signal be denoted by s_m, and the corresponding faulty

signal, representing a deviation or distortion from the healthy state, be denoted by δs_m. The complete set of signals acquired from the sensors can thus be expressed as

$$\mathcal{S} = s_m + \delta s_m, \tag{1.1}$$

where δs_m represents the faulty signal collected from the system. In this study, δs_m corresponds to 10 distinct fault conditions (Table 3.2), including the healthy state, for which $\delta s_m = 0$. The class labels are indexed as $k \in \{0, \ldots, K_{\text{cls}}\}$.

This information from the physical system $\mathcal{S}$ is transformed by applying STFT (Eq. 4.7), resulting in X_{ij}. When this feature is passed into the machine learning model M, the predicted output is obtained as shown in Eq. (1.2):

$$\hat{y} = M(X_{ij}), \tag{1.2}$$

where $\hat{y}$ denotes the model's predicted class.

Now this framework is used for predictive maintainence of the early fault detection using the CNN based approach to detect the fault type and the severity level.

1.3 Research Contributions

The key contributions of the thesis are summarized as follows:

- **Handling class imbalance in realistic datasets $\mathbf{C_1}$:** Mitigated the issue of data imbalance by incorporating a augmentation methods, weighted loss function and assigning higher penalties to underrepresented fault classes and considering metrics like F1 score and ROC-AUC metrics. This improved sensitivity to minority fault patterns and ensured balanced classification performance under diverse load and fault scenarios.

- **Efficiency-oriented preprocessing and scalability considerations $\mathbf{C_2}$:** Instead of passing long raw vibration signals directly into the network, the data has been segmented into smaller windows and transformed into compact 2D STFT spectrograms. This design reduces input dimensionality, improves learning efficiency, and enables effective use of modern CNN architectures. Although detailed benchmarking of computational speed, memory footprint, and hardware efficiency was not conducted in this study, these aspects can be addressed in future work to further assess the framework's real-time deployment potential.

- **Time–frequency feature extraction using spectrograms C_3:** Developed a robust time–frequency feature extraction pipeline based on the Short-Time Fourier Transform (STFT) and its differential variants to capture localized fault patterns from vibration signals. Both high-resolution spectrograms and flattened vector representations are explored, enabling a trade-off analysis between feature richness and computational efficiency.

Together, these contributions advance the state of vibration-based fault diagnosis by addressing fundamental challenges bridging the gap between controlled benchmark performance and practical industrial deployment. While prior studies on the CWRU dataset have advanced bearing fault diagnosis through envelope analysis, handcrafted features, lightweight CNNs, physics-guided kernel design, multi-label formulations, and transformer-based models, they remain limited by issues such as dataset leakage, the fact that faults are artificially induced in laboratory settings (rather than naturally evolved in real machinery), and the absence of deployment-oriented evaluation. Our work establishes a new benchmark by introducing a unified STFT pipeline that systematically compares 1D and 2D CNNs, extends Delta-STFT into a cross-resolution multi-channel representation, and applies ImageNet transfer learning on grayscale and RGB STFTs. Furthermore, we haved evaluated safe vs. unsafe misclassifications, bridging the gap between high accuracy and real-world deployability.

1.4 Outline of Thesis

The remaining chapters of the thesis are structured as follows:

Chapter 2: This chapter provides a comprehensive review of the existing literature on traditional and machine learning-based bearing fault diagnosis, including time–frequency analysis, classical signal processing, and deep learning models. It highlights key developments and applications relevant to the proposed research.

Chapter 3: This chapter defines the research problem and introduces the experimental setup used for vibration data acquisition. It describes the physical system, bearing configurations, and underlying fault mechanisms, explaining how various fault types are reflected in vibration responses. The chapter concludes by outlining the key objectives that guide the development of the proposed bearing fault diagnosis framework.

Chapter 4: This chapter presents the overall methodology and proposed framework for bearing fault diagnosis. It describes the data preprocessing pipeline, feature extraction

using time–frequency representations, and the deep learning models developed to perform fault classification. The design rationale, input representation strategy, and integration of the models within the end-to-end diagnostic workflow are discussed in detail.

Chapter 5: This chapter discusses the experimental evaluation and performance analysis of the proposed models. It outlines the experimental design, datasets, and evaluation metrics used to assess model performance. The comparative study among the implemented architectures highlights their strengths, limitations, and generalization capabilities. The chapter concludes with a summary of key observations and insights derived from the results.

Chapter 6: The final chapter summarizes the conclusions, limitations, and future work.

Chapter 2

Background and Related Work

This chapter presents a comprehensive review of the literature on bearing fault detection and classification, with a particular emphasis on the growing role of machine learning approaches in this domain. Bearings are critical components in rotating machinery, and their unexpected failures can result in costly downtime, safety hazards, and significant maintenance expenses. Traditional condition monitoring methods, while effective to a degree, often rely on handcrafted features and domain-specific heuristics, which may not generalize well across varying operating conditions. In contrast, data-driven machine learning techniques offer the promise of scalable, automated, and adaptive solutions capable of handling the complex and non-stationary nature of vibration signals. This chapter aims to contextualize the motivations for machine learning–based bearing fault diagnosis, outline the challenges inherent in industrial applications, and systematically survey the relevant literature. Section 2.1 introduces benchmark datasets widely adopted for fault detection research. Section 2.2 highlights machine learning techniques applied to rotating machinery condition monitoring. Section 2.3 highlights machine learning paradigms for bearing fault diagnosis, while Section 2.4 discusses time–frequency and spectrogram-based feature extraction strategies. Finally, Section 2.5 reviews deep learning architectures and their effectiveness under diverse operational scenarios, setting the stage for the proposed methodology in the following chapters.

2.1 Benchmark Datasets for Bearing Fault Diagnosis

The effectiveness of advanced machine learning models for bearing fault diagnosis depends on the availability of reliable benchmark datasets. Over the past decades, researchers have

relied on publicly available vibration datasets to evaluate and compare the performance of their methods. These datasets provide controlled test environments with known fault classes, well-defined operating conditions, and documented sensor setups, thus enabling reproducibility and fair comparisons across studies. However, while laboratory datasets are indispensable for benchmarking, they often reflect simplified conditions and idealized faults created through controlled experiments, which may not fully represent the complexity of faults encountered in industrial field operations. For example, laboratory datasets generally involve seeded faults with consistent dimensions, speeds, and loads, whereas real-world scenarios exhibit variable fault severities, compound fault types, changing load conditions, and significant environmental noise. Consequently, there is a growing emphasis on understanding the scope and limitations of these benchmark datasets, as well as encouraging complementary use of synthetic data, domain adaptation techniques, and field-collected signals for broader generalizability. The following paragraphs summarize some of the most widely used bearing fault datasets that have formed the backbone of data-driven fault diagnosis research, outlining their structure, acquisition rigs, sampling characteristics, and common adoption in the literature.

- **Case Western Reserve University (CWRU) Dataset:** The CWRU bearing dataset [8, 9, 10] is among the most prominent in the literature. It was collected on a 2-horsepower, three-phase induction motor operating under four discrete load conditions (0, 1, 2, and 3 HP) with speeds ranging from 1730 to 1797 RPM. Artificial faults were introduced via Electrical Discharge Machining (EDM) on the inner race, outer race, and rolling elements, with fault diameters of 0.007"inch, 0.014"inch, and 0.021"inch. Vibration signals were recorded using accelerometers mounted on the drive end and fan end of the motor, with sampling rates of 12,000 samples/s and 48,000 samples/s. Due to its standardized setup and availability of a wide range of fault conditions, the CWRU dataset has become a de facto benchmark for validating fault diagnosis algorithms.

- **Paderborn University (PU) Dataset:** The Paderborn dataset [11, 12] was created on a test rig consisting of a 4-kW electric motor with belt drives, covering 14 faulty classes and 6 healthy classes, including both real and artificial defects. It provides vibration signals sampled at 64 kHz, measured under varying radial loads, with bearing damages induced via electrical discharge and accelerated aging. Its inclusion of naturally degraded bearings and a broader fault diversity make it an important resource for studying domain adaptation and cross-condition generalization.

- **Intelligent Maintenance Systems (IMS) Bearing Dataset:** Collected at the University of Cincinnati's Intelligent Maintenance Systems IMS Center, this dataset [13]

captures run-to-failure experiments on Rexnord bearings under constant load and speed. Vibration data were acquired at 20 kHz while the machine ran continuously until bearing failure, providing a rich degradation progression from healthy to completely failed states. This enables the development of models for predictive maintenance and Remaining Useful Life (RUL) estimation, rather than only classification of fixed faults.

- **Xi'an Jiaotong University and Shenyang Aerospace University (XJTU-SY) Dataset:** The XJTU-SY bearing dataset [14] was recorded on a 6 kW motor under multiple speed and load conditions, with faults seeded in the inner race, outer race, and bearing cage. It uses a 25.6 kHz sampling rate, and provides time series data under realistic compound fault scenarios and variable operating conditions, expanding the possibilities for studying multi-fault and variable-load diagnostic tasks.

- **Jiangnan University (JNU) and Northeast Forestry University (NEFU) Datasets:** These more recent datasets [15] provide vibration signals from different test rigs with varying loads and rotational speeds, covering health states including normal, rolling element faults, cage faults, and complex compound faults. Their primary purpose is to support transfer learning and domain adaptation experiments, since they represent a wider cross-domain distribution.

- **Custom and Synthetic Datasets:** Several recent studies have developed their own laboratory-scale datasets using flexible rotor-bearing rigs [12] or created fully synthetic datasets with dynamic simulation models [16, 13]. These synthetic approaches simulate faults under varied bearing designs, speeds, and loads, and then validate them on public datasets such as CWRU or even aerospace bearing datasets, demonstrating promising cross-domain generalization.

These benchmark datasets: CWRU, PU, IMS, XJTU-SY, JNU, NEFU, and synthetic datasets provide a reliable foundation for reproducible bearing fault diagnosis research. They enable the development and comparison of advanced machine learning approaches, although their laboratory settings motivate future expansion toward field-acquired and industrial datasets.

2.2 Machine Learning in Rotating Machinery Condition Monitoring

Recent years have witnessed an extensive application of machine learning (ML) techniques in the condition monitoring and fault diagnosis of rotating machinery systems. Unlike conventional signal-processing or threshold-based rule systems, ML models can learn from historical data and extract discriminative patterns to identify or predict incipient failures under varying operating conditions. This data-driven paradigm supports more adaptive, scalable, and responsive maintenance practices, essential for modern industrial environments. The surge in ML adoption is largely motivated by the need to achieve higher reliability, reduced maintenance costs, and improved safety across diverse sectors, ranging from manufacturing to energy production [11, 8, 9, 14].

Early efforts in this area primarily relied on traditional statistical and machine learning techniques. These approaches often used handcrafted time-domain features such as RMS, kurtosis, and crest factor, which are computationally simple but struggle to capture the complex, multi-scale patterns typically seen in bearing fault signals. As highlighted in recent reviews [17], such metrics tend to be less effective for identifying early-stage faults, where timely detection is critical for preventing failures. Given the critical role bearings play in motors, turbines, pumps, and other machinery, timely fault detection is essential to enable effective condition-based maintenance and predictive diagnostics [15].

Many of these early methods were evaluated on benchmark datasets, most notably the Case Western Reserve University (CWRU) bearing dataset [8, 9, 10], which has become a standard due to its controlled operating conditions, clearly labeled fault types, and widespread adoption in comparative studies. To push beyond the limitations of CWRU's simplified test environment, other datasets have been introduced. The Paderborn University (PU) dataset [11] incorporates both real-world and artificial faults at varying loads and speeds, while the IMS dataset [13] captures full lifetime degradation, enabling long-term progression analysis. Similarly, the XJTU-SY dataset [14] and the QPZZ-II testbed [15] introduce compound faults and variable speeds, providing more realistic diagnostic challenges as dicussed in the previous Section 2.1

Classical machine learning models such as Support Vector Machines (SVM), Decision Trees (DT), and k-nearest neighbors (k-NN) were widely used with these datasets as baselines. However, by manual feature engineering limited robustness under sensor noise and non-stationary operating conditions. For example, Ni et al. [11] observed that SVM and DT performance deteriorated sharply under low signal-to-noise ratios when compared against a wavelet-integrated CNN. Similarly, Shao et al. [14] reported much lower accuracy for SVM

and DT compared to optimized hybrid deep learning models on the XJTU-SY dataset. These results underscore the limited generalizability of traditional ML methods under realistic industrial fault complexities.

The performance limitations of classical approaches, combined with the growing availability of diverse high-resolution datasets, have catalyzed a shift toward more adaptive, data-driven techniques. This shift is reflected in the dominance of supervised deep learning methods, which reduce dependence on manual feature engineering and show robustness to noise, class imbalance, and variable operating conditions.

Traditionally, supervised learning approaches have dominated this domain, requiring labeled fault data to train classifiers that distinguish healthy states from faulty conditions or categorize specific fault types and severities. In these tasks, deep learning architectures such as Convolutional Neural Networks (CNNs). The performance limitations of classical approaches, combined with the growing availability of diverse high-resolution datasets, have catalyzed a shift toward more adaptive, data-driven techniques. This shift is reflected in the dominance of supervised deep learning methods, which reduce dependence on manual feature engineering and demonstrate greater robustness to noise, class imbalance, and variable operating conditions.

Traditionally, supervised learning approaches have dominated this domain, requiring labeled fault data to train classifiers that distinguish healthy states from faulty conditions or categorize specific fault types and severities. In these tasks, deep learning architectures such as Convolutional Neural Networks (CNNs) [8, 9, 18, 11], spiking neural networks [19], autoencoders, transformers [20, 21], and hybrid models combining temporal and spatial feature extraction [22, 23] have achieved impressive results on benchmark datasets, often surpassing classical machine learning or rule-based techniques.

2.3 Machine Learning Paradigms for Bearing Fault Diagnosis

Traditionally, supervised learning has served as the dominant framework for bearing fault diagnosis, leveraging labeled datasets to develop robust classifiers capable of distinguishing between healthy states and a variety of fault conditions. These supervised paradigms generally involve manually labeled datasets with discrete fault categories, enabling the training of models to recognize and differentiate specific types of damage, their severities, or their locations within the bearing assembly. Within this context, deep learning methods, particularly Convolutional Neural Networks (CNNs) [8, 9, 18, 11], have demonstrated

strong performance by automatically extracting hierarchical and spatial features from raw vibration or current signals, outperforming traditional feature-engineering pipelines. In addition to CNNs, emerging architectures such as spiking neural networks [19], autoencoders, transformers [20, 21], and hybrid models that fuse temporal and spatial feature extraction capabilities [22, 23] have further expanded the capabilities of supervised fault classification systems, achieving state-of-the-art results across diverse benchmark datasets.

Despite their successes, supervised models face inherent challenges in real-world deployment, primarily due to their reliance on accurately labeled training data. In industrial settings, collecting high-quality labeled data is costly, time-consuming, and often impractical, since bearing faults are rare, unpredictable, and may develop over extended equipment lifespans. These constraints have fueled considerable interest in unsupervised and semi-supervised approaches, which seek to minimize or eliminate the need for labeled data while still delivering high diagnostic accuracy. Among these, autoencoder-based anomaly detection frameworks, domain adaptation methods [13, 24], and simulated-data-driven transfer learning strategies [16, 13] have emerged as promising alternatives, aiming to leverage patterns in normal operating data or synthetic data to detect and characterize faults without requiring exhaustive labeled datasets.

Furthermore, beyond discrete fault classification, there has been a growing push toward regression-based models that estimate continuous degradation indicators, such as remaining useful life (RUL) or progressive damage indices, to enable more proactive and predictive maintenance strategies. However, the adoption of these regression-based prognostic frameworks remains relatively sparse within vibration-based bearing monitoring applications, largely due to the absence of reliable long-term datasets capturing gradual degradation under realistic operating scenarios. Several recent survey studies have highlighted this limitation and emphasized the need for new data collection efforts to better support prognostics research in rotating machinery [10, 15].

Another critical barrier to industrial adoption concerns the transferability and generalization of ML models trained on public laboratory datasets, which frequently rely on faults introduced in controlled experimental rigs. These artificially induced fault conditions often fail to reflect the diversity, stochasticity, and complexity of faults encountered in actual industrial equipment. As a result, models trained exclusively on laboratory data may exhibit significant performance degradation when faced with variable loads, speeds, compound faults, or evolving operating environments in practice. Addressing this so-called domain shift problem has become a central focus of recent research, with transfer learning and domain adaptation frameworks showing encouraging results in bridging the gap between laboratory and industrial domains [20, 13, 24, 15].

In summary, while supervised learning continues to drive significant progress in bearing fault diagnosis, there is a clear and growing need to develop methods that are more data-efficient, generalizable, and robust to real-world variability. Unsupervised anomaly detection, transfer learning, domain adaptation, and hybrid models integrating physical knowledge with data-driven techniques represent promising avenues to address these gaps. The remainder of this review explores these methodological advances in more detail, highlighting key research directions, experimental benchmarks, and emerging opportunities to achieve practical, scalable, and explainable bearing fault diagnosis solutions in the industrial landscape.

2.4 Time–Frequency Analysis and Spectrogram-Based Features

Vibration signals originating from rotating machinery, such as rolling element bearings, are inherently non-stationary, with their statistical properties varying over time due to changing loads, speeds, and the development of faults. Traditional time-domain or frequency-domain analyses alone are often insufficient to capture these evolving and transient characteristics. As a result, time–frequency analysis has emerged as a critical tool, providing a joint representation of how signal energy is distributed across both time and frequency domains, which is especially valuable for machine learning applications in fault diagnosis .

Among these methods, the Short-Time Fourier Transform (STFT) is widely adopted due to its simplicity and interpretability. STFT works by segmenting a signal into overlapping windows and applying a local Fourier transform within each segment, resulting in a two-dimensional time–frequency representation known as a spectrogram. This spectrogram reveals how the frequency content of the vibration signal changes over time, highlighting transient impulses and sudden shifts that may indicate early-stage faults such as bearing spalls or cracks [8]. For example, Pavithra and Ramachandran (2025) showed how STFT-based spectrograms, combined with empirical mode decomposition, could effectively train deep convolutional neural networks to classify bearing faults even under noisy conditions.

However, a limitation of STFT is its fixed time–frequency resolution, which is governed by the window size and may not adapt well to signal components occurring across multiple scales. Wavelet-based methods, such as the discrete wavelet transform (DWT) or continuous wavelet transform (CWT), address this challenge by providing multi-resolution analysis. Wavelets use scalable basis functions that allow fine time resolution for high-frequency features and better frequency resolution for low-frequency components. These

advantages have led to wavelet transforms being incorporated into deep learning architectures for fault diagnosis, as demonstrated by Ni et al. (2025), who integrated wavelet-based downsampling into a CNN–ResNet framework to improve robustness against noise [11], and by Shao et al. (2025), who combined variational mode decomposition with DWT to build a hybrid CNN–BiLSTM model for enhanced fault recognition [14].

While wavelet-based approaches offer significant benefits, this study focuses on STFT-derived spectrograms as the primary feature representation. Spectrograms provide a consistent and interpretable format, are well established in the literature, and are directly compatible with standard convolutional and transformer-based neural network architectures. These deep learning models can leverage the two-dimensional spatial structure of spectrogram images to learn discriminative features associated with different fault types and operating conditions [22, 21, 23]. This compatibility, along with their proven performance on benchmark datasets, makes spectrograms a practical and effective choice for scalable bearing fault diagnosis.

In summary, time–frequency analysis methods play a foundational role in transforming complex, non-stationary vibration signals into forms suitable for modern machine learning pipelines. By highlighting transient and evolving patterns, particularly through STFT-based spectrograms, these representations enable deep models to achieve more reliable, interpretable, and robust fault detection and prognosis, while leaving open opportunities to explore wavelet-based enhancements in future research.

2.5 Deep Learning Architectures for Bearing Fault Diagnosis

Deep learning (DL) has become a transformative technology in the field of bearing fault diagnosis, enabling data-driven models to learn complex patterns directly from raw or minimally preprocessed vibration signals. Unlike classical machine learning pipelines that rely on manual feature engineering and domain-specific signal processing, deep learning architectures can automatically discover hierarchical features relevant to fault classification, severity estimation, or prognostics. This capability is especially useful for dealing with non-stationary, high-dimensional, and noisy data that are common in rotating machinery systems. As industrial datasets grow in size and diversity, deep learning provides a scalable framework to generalize across variable loads, speeds, and compound fault modes, potentially improving diagnostic accuracy and reducing the dependency on expert-crafted features. Nevertheless, challenges remain, including high data requirements, explainability,

and robustness to domain shifts between laboratory and real-world environments [10, 24].

2.5.1 Convolutional Neural Networks for Vibration Signals

Convolutional Neural Networks (CNNs) have become a cornerstone of bearing fault diagnosis due to their strong capacity to learn local and hierarchical features from vibration signals. Typically, vibration signals are transformed into time–frequency images (e.g., spectrograms) that serve as inputs to 2D CNNs. For example, Pavithra and Ramachandran (2025) employed STFT-based spectrograms of noisy vibration data and trained a deep CNN, achieving robust performance even in low-SNR conditions [8]. Other researchers, such as Sonmez et al. (2023), explored wide-kernel CNN architectures to improve the receptive field and capture more global information from 1D vibration sequences [18, 25].

In addition, CNNs have been enhanced with domain adaptation and denoising strategies to handle the limitations of laboratory-based datasets. [11] has proposed replacing standard pooling operations with discrete wavelet downsampling layers within a CNN-residual network to preserve low-frequency fault features and improve generalization under noisy and variable-load conditions. CNNs are also frequently integrated with empirical mode decomposition (EMD), variational mode decomposition (VMD), and blind deconvolution to remove noise and boost discriminative features before feeding signals into CNN architectures [14, 12]. Overall, CNNs provide a powerful baseline for vibration-based fault diagnosis but may struggle to fully exploit temporal dependencies in sequential data without hybridization.

2.5.2 Hybrid Architectures (CNN–LSTM, Transformer-based Models, etc.)

Hybrid deep learning models aim to capture both spatial patterns in time–frequency representations and temporal dependencies in vibration signals. These frameworks typically combine CNNs with sequential modules such as LSTMs, GRUs, or Transformer-based neural networks. Here, **Transformer refers to self-attention architectures originally introduced in natural language processing, which are now adapted to vibration analysis for their ability to capture long-range dependencies** [20, 21].

Wang and Zhao (2025) proposed a CNN–LSTM–attention ensemble that outperformed conventional CNNs under varying loads and fault types [22]. Liu et al. (2025) developed an interpretable domain adaptation Transformer that achieved strong cross-domain accuracy while highlighting informative signal segments [20]. Snyder et al. (2025) integrated

Transformers with CNNs to enhance spectrotemporal modeling [21]. Beyond sequential hybrids, Chennana et al. (2025) fused CNN features with handcrafted descriptors, achieving robustness to noise and load variation [23].

While these hybrid frameworks generally surpass CNN-only baselines by leveraging both spatial and sequential information, they increase computational cost and demand larger, more diverse datasets. Addressing challenges of efficiency, interpretability, and deployment feasibility remains essential for real-world adoption.

2.6 Research Gaps

Based on the existing literature, a thorough examination reveals that there are significant gaps that need to be addressed to enhance the robustness and industrial applicability of machine learning approaches for bearing fault diagnosis. While recent years have seen rapid progress in supervised and deep learning methods, several limitations persist that restrict their direct deployment in real-world scenarios.

Although CNN-based models have demonstrated strong performance in bearing fault classification, particularly on benchmark datasets such as the Case Western Reserve University (CWRU) dataset, a number of research gaps remain unaddressed:

- **Idealized evaluation settings:** Many studies conduct experiments under controlled laboratory conditions with fixed motor speeds, low noise levels, and balanced class distributions. These conditions fail to capture the variability, noise, and operating uncertainties of industrial environments.
- **Data leakage and limited generalization:** It is observed that the same bearings are used for both training and testing. While this may inflate reported accuracy, it limits the model's ability to generalize to unseen bearings or fault conditions.
- **Underreporting of deployment-critical metrics:** Key indicators such as false alarm rate, inference latency, and computational efficiency are frequently omitted, making it difficult to assess practical feasibility for real-time deployment.
- **Limited exploration of multi-resolution and multi-channel spectrogram analysis:** Most prior studies rely on time–frequency representation as the model input, this restricts the ability to capture complementary patterns that exist across different time–frequency resolutions. In particular, the combined use of multi-resolution

STFTs and their differential representations (Delta-STFTs) as stacked input channels remains underexplored in the bearing fault diagnosis literature, despite its potential to enrich spatiotemporal feature extraction.

- **Underutilization of transfer learning with large-scale pretrained CNNs:** Although CNNs have been widely applied to spectrograms, most studies train these models from scratch using only vibration datasets. The potential of transfer learning from large-scale image classification networks (e.g., ImageNet-pretrained models such as XceptionNet or EfficientNet) has not been systematically examined. Demonstrating that such models can be effectively fine-tuned for fault spectrograms provides a strong approach to reduce training cost, mitigate data scarcity, and accelerate convergence yet this remains largely absent from existing research.

Table 2.1 provides a comparative summary of representative studies, highlighting the strengths and weaknesses of current machine learning approaches for bearing fault diagnosis. The table illustrates that although reported accuracies are above 95%, the limitations outlined above remain prevalent and effect negatively industrial adoption. While the 2.2 provides a compartitive summary of the studies specific to the Case Western Reserve University's Bearing Fault Diagnosis Data.

Table 2.1: Comparative Summary of Machine Learning Approaches for Bearing Fault Diagnosis

Study	**Year**	**Method**	**Key Contribution**	**Accuracy**	**Limitation**
Fan et al.	2024	MPEG + SVM	Graph-based entropy features improve robustness.	98.3% (CWRU), 96.7% (Lab)	High computation; parameter sensitivity.
Ni et al.	2025	Baseline ML vs. DWCResNet	Wavelet-enhanced CNN outperforms ML under noise.	SVM: ∼88%, DWCResNet: >99%	Baselines degrade under noise.
Shao et al.	2025	ML vs. CNN-BiLSTM	Hybrid model improves accuracy with temporal features.	SVM/DT: ∼90%, CNN-BiLSTM: >99%	ML struggles with temporal features.
Zhang & Liu	2025	MSLSTM + TL	Transfer learning effective with low labels.	>93%	Baseline results not reported.
Wang et al.	2025	CNN + IATL	Synthetic data reduces domain gap.	99.89% (CWRU→IMS)	Complex, limited real-time use.
Neupane & Seok	2020	CNN, DBN, AE, GAN	Review of DL models on CWRU.	CNN: 99.9%, DBN: 98.5%	Large data needs, imbalance issues.
Ruan et al.	2023	Physics-Guided CNN	Incorporates physics into CNN.	99.95% (CWRU), 99.79% (PU)	Limited to modeled faults.
Pavithra & Ramachandran	2025	DCNN + STFT + HHT	EMD/HHT spectrograms for noise-robust classification.	100% (CWRU up to 10 dB)	Requires EMD/HHT; sensitive to impulse noise.
Wang & Zhao	2025	1D2D-EDL	CNN+LSTM+attention ensemble.	99.47% (UOttawa)	High complexity, tuning-sensitive.
Liu et al.	2025	IDAT (Transformer)	Interpretable domain adaptation via attention.	96.9% (cross-cond.), 93.3% (cross-mach.)	Needs labels, high cost.
Kiakojouri & Wang	2025	CNN + Hybrid Features	Synthetic training for domain generalization.	99.09% (CWRU), 86.03% (Safran)	Weak on real-world faults.
Li et al.	2025	MKWJDAN (1D CNN + MKMMD)	Adaptive transfer with pseudo-labeling.	98.96% (H2→H1)	Needs careful pseudo-label tuning.
Chennana et al.	2025	CNN + MBH-LPQ	Combines deep+shallow features for load invariance.	98.87% (PU)	Limited to spectrogram-based inputs.
Liao et al.	2025	ClassBD + WDCNN	Joint denoising-classification robust to noise.	99.7% (PU)	Complex optimization.

Table 2.2: Comparison of Representative Studies on Bearing Fault Diagnosis Using the CWRU Dataset

Study	**Year**	**Method**	**Key Contribution**	**Performance (CWRU)**	**Limitation**
Zhang & Zhou [26]	2021	DCNN + TL (ImageNet) on grayscale STFT	1D → 2D grayscale transformation; ImageNet-pretrained CNNs validated on CWRU, SPCP, and Paderborn datasets.	Up to 100% (DenseNet121, MobileNet)	Only grayscale mapping; no multi-channel features; limited interpretability.
Balamurugan et al. [27]	2024	CNN + BiLSTM (RUL Estimation)	Hybrid CNN–BiLSTM for Remaining Useful Life (RUL) prediction using CWRU signals; focused on prognostics rather than classification.	Improved RUL prediction vs. baselines	Limited to prognostics; not evaluated for classification or domain generalization.
Yoo et al. [28]	2023	Lite CNN + Downsampled STFT	Lightweight CNN architecture using downsampled STFT representations for reduced computational complexity.	99.96–100%	Tailored to CWRU; lacks transfer learning or cross-dataset validation.
Alonso-González et al. [29]	2023	Envelope Analysis + ML (KNN, DT, NB)	Traditional envelope features combined with ML classifiers demonstrated strong accuracy on CWRU.	KNN/DT: 100%; NB: 94.4%	Relies on handcrafted features; limited robustness and deployment validation.
Ruan et al. [30]	2023	Physics-Guided CNN (PGCNN)	CNN input and kernel design derived from bearing physics; validated on CWRU and Paderborn datasets.	99.95%	Physics-specific tuning; limited generalization beyond lab conditions.
Raj et al. [31]	2024	MLP, LSTM, CNNs on Raw Signals	End-to-end training on raw accelerometer signals; separability explored via PCA/t-SNE visualization.	2D-CNN: 99.32% (0 hp), 91.16% (1 hp), 94.97% (2 hp)	Accuracy drops with load variation; lacks spectrogram or transfer validation.

Comparison of representative studies on bearing fault diagnosis using the CWRU dataset (cont.)

Rosa et al. [32]	2024	Multi-label DL (ResNet18, WDCNN, EfficientNet, Transformers)	Proposed multi-label + leakage-free split; benchmarked CNNs/Transformers	AUROC: 0.911 (ResNet18 spectrogram); up to 0.99 for some	Limited to CWRU; artificially induced EDM faults; inner/outer fault confusion
Jang et al. [33]	2024	DAE + Vision Transformer (ViT)	Denoising autoencoder preprocessing; ViT classification robust to noise	Raw ViT: 99.46%; DAE+ViT: 100%	Focused on noise; Transformers compute-heavy
Łuczak et al. [34]	2024	IMU6DoF-SST-CNN (FSST/WSST)	Synchrosqueezed features sharper than STFT; IMU data fusion; tested on fan + CWRU	$\geq$90% test; up to 100% train/-val (13-class)	Relies on IMU; costly fusion; risk of overfitting
Mohammad-Alikhani et al. [35]	2025	Differential STFT + Channel-Regulated CNN	Multi-resolution differential STFT + channel regulation; noise-robust	$\sim$30% noise MSE reduction; strong accuracy	Needs tuning; focus on noise suppression; not a full benchmark
Huang et al. [36]	2025	STFT + Multi-Scale Channel Attention (MsCCA)	STFT + multi-scale convs with channel attention; outperformed CNN-LSTM/GRU/ResNet	Best: 99.87%; Avg: 99.35%	Only CWRU; single faults; added complexity; no deployment metrics
van den Hoogen et al. [25]	2020	Wide-Kernel CNN (WDTCNN, WDMTCNN)	Extended WDCNN with wide/multi-channel kernels; robust to noise	Up to 99.9%	Tailored to CWRU; no cross-domain validation; higher cost
Han & Jeong [37]	2020	Weighted CNN Ensemble	CNN ensemble with weighted fusion; effective with small training sets	$\sim$99%	Inference cost higher; no deployment validation
Hendriks et al. [10]	2022	Benchmark critique of CWRU	Exposed flaws in cross-load evaluation; proposed independent bearing splits; tested CNNs	Accuracy dropped $\sim$45% under stricter split	Many CWRU studies overestimate generalization; still no field data
Proposed Approach	**2025**	1D-CNN + Xception-Net71 + EfficientNetV2-M	Unified framework with multiple input reps (1D STFT, spectrograms, Delta-STFT) and tailored deep architectures.	**99.76% (CWRU)**	Cross-domain validation pending.

By systematically identifying these gaps, the present work addresses them through the design of stratified cross-fold validation protocols, comparative evaluation of both 1D and 2D CNN-based models, and explicit consideration of model complexity alongside classification accuracy. While prior studies on the CWRU dataset have advanced bearing fault diagnosis through envelope analysis, handcrafted features, lightweight CNNs, physics-guided kernel design, multi-label formulations, and transformer-based models, they remain limited by issues such as dataset leakage, the fact that faults are artificially induced in laboratory settings (rather than naturally evolved in real machinery), and the absence of deployment-oriented evaluation. Our work establishes a new benchmark by introducing a unified STFT pipeline that systematically compares 1D and 2D CNNs, extends Delta-STFT into a cross-resolution multi-channel representation, and applies ImageNet transfer learning for the first time on grayscale and RGB-like STFTs. Furthermore, we uniquely evaluate safe vs. unsafe misclassifications, bridging the gap between high accuracy and real-world deployability.

Chapter 3

Problem Description

This chapter is organized as follows: Section 3.1 describes the layout of the physical system model and the data acquisition setup. Section 3.2 explains the manifestation of fault types and severities over time. Section 3.2.1 presents representative vibration signals illustrating these fault types and severity levels and Section 3.3 describes the objective of the thesis.

This chapter addresses the core problem of this thesis—the robust detection and classification of bearing faults from vibration signals collected under varying operating conditions. To achieve this, it is essential to first understand the physical setup of the machinery from which these signals originate. As illustrated in Fig. 3.1, the system under study consists of a rotating machinery test rig that includes an induction motor, a shaft, and a bearing assembly operating under controlled load and speed conditions. This configuration closely replicates a typical industrial drivetrain, enabling the acquisition of vibration signals from the bearing housings during both healthy and faulty states. These signals form the foundation for the AI-based fault detection model discussed in the subsequent sections.

Over time, the mechanical integrity of the bearing components deteriorates due to continuous operation, misalignment, or material fatigue, and sometimes due to environmental factors such as corrosion, contamination, or temperature variations. As a result, microscopic cracks or pits begin to form on the raceways or rolling elements, which grow with repeated stress cycles. These localized defects disrupt the uniform transmission of forces within the rotating system, causing periodic impacts whenever a rolling element passes over the damaged region. The repeated impacts modulate the vibration response of the machine, gradually altering its characteristic waveform as the fault progresses. As the damage worsens, the deviations from the nominal healthy signature become increasingly evident—manifesting as amplitude fluctuations, sideband frequencies, and broadband noise

components in the frequency spectrum.

Such progressive degradation in bearing health has a direct impact on industrial productivity and operational safety. Even minor defects can lead to increased vibration, heat generation, and energy losses, reducing the overall efficiency of rotating equipment. If left undetected, these faults may propagate to adjacent components such as shafts, gears, or motors, resulting in unplanned downtime, costly repairs, and production delays. In critical applications, catastrophic bearing failure can cause complete machine shutdowns or safety hazards, emphasizing the need for timely fault detection and condition-based maintenance strategies.

The data used to study this problem are obtained from controlled laboratory experiments using the Case Western Reserve University (CWRU) Bearing Data Center test rig. The physical setup, shown in Fig. 3.1, and its instrumentation are described in detail in the following section. The dataset includes vibration signals measured under various load and fault conditions, providing a standardized benchmark for developing and evaluating machine learning models for bearing fault diagnosis.

By continuously monitoring the performance of the motor through vibration-based sensing and predictive analytics, it becomes possible to detect early signs of degradation before they escalate into major faults. Predictive maintenance strategies, supported by AI-driven fault diagnosis models, enable data-driven assessment of bearing health and facilitate timely intervention. By accurately identifying anomalies in motor behavior and forecasting potential failures, industries can minimize unplanned downtime, extend equipment lifespan, and ensure safer, more reliable operation of rotating machinery—effectively preventing catastrophic bearing failures.

3.1 System Model

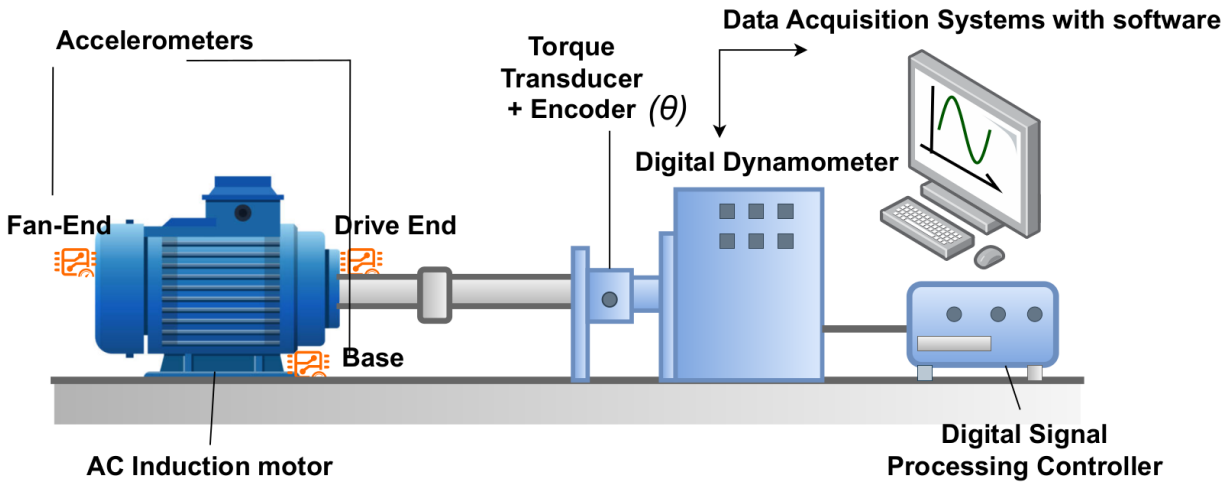


Figure 3.1: System Model Diagram

Table 3.1: Motor Specifications Used in the Case Western Reserve University (CWRU) Test Setup

Parameter	Value
Motor Type	3-phase AC induction motor
Manufacturer	Reliance Electric
Rated Power	2 horsepower (HP), approximately 1.5 kW
Rated Voltage	208–230 V / 460 V (dual-voltage configuration)
Rated Current	6.4 A at 230 V, 3.2 A at 460 V
Motor Frequency	60 Hz (nominal line frequency)
Synchronous Speed	1800 RPM (4-pole motor at 60 Hz)
Operating Speeds in CWRU	1797, 1772, 1750, 1730 RPM (measured under experimental loads)
Number of Poles	4
Rated Efficiency	85–87% (typical for 2 HP motors)

The physical model diagram Figure 3.1 illustrates the test setup designed to evaluate the vibration response of a 3-phase AC induction motor mounted on a test bench rated at 2 horsepower and operating at 60 Hz. The motor specifications used in this setup are summarized in Table 3.1. It was tested under controlled mechanical conditions, with variations in operating conditions and achieved by mechanically loading the shaft through a controllable dynamometer. The mechanical output of the motor is transmitted to a torque transducer and encoder mounted directly in line along the motor shaft. The torque transducer measures real-time torque output, while the encoder provides accurate readings of rotational speed (RPM) and angular position (θ). Here, θ refers to the instantaneous rotational displacement of the shaft relative to a fixed reference point. Measurement of angular position enables the synchronization of vibration signals with shaft rotation, which is essential for accurate bearing fault analysis. Although the test setup includes this instrumentation, the publicly available CWRU dataset does not include these torque and speed measurements instead, estimates are inferred from test configuration files, load la-

bels, and metadata. Therefore, the present analysis relies primarily on vibration signals collected from accelerometers placed at the drive end, fan end, and base of the motor. The mechanical output of the motor, characterized by torque and rotational speed, is delivered to the digital dynamometer, which acts as a controllable load by applying variable resistive torque to simulate different operating conditions during the experiments.

To monitor the health of the system, three accelerometers are installed on the motor. The drive-end (DE) accelerometer captures vibrations near the output shaft bearing, where load-transmitting components are located. The fan-end (FE) accelerometer measures vibrations at the non-drive side, often associated with imbalance or cooling-related issues. The base (BA) accelerometer, mounted on the motor's foot or base structure, detects structure-borne vibrations and resonance, giving insight into mounting stability and system-level behavior.

All sensor outputs, including vibration signals, as well as torque and speed measurements from the bench instrumentation, are routed to a dedicated data acquisition (DAQ) system. This DAQ platform performs synchronized, timestamped signal sampling and organizes the measurements for subsequent computational processing. These data streams are used to train and validate machine learning models for bearing fault detection.

We will discuss about the fault model and signal representation in the next section 3.2.

3.2 Fault Model and Signal Representation

The rolling elements called bearings, are critical components in rotating machinery, designed to support shafts and reduce friction between moving parts. A typical bearing in this study refers to a single-row, deep-groove ball bearing, which consists of an inner race, an outer race, a cage, and rolling elements (balls), as illustrated in Figure 3.2. Faults can develop in any of these regions: rolling elements may suffer surface pitting or spalling, the inner ring can experience localized damage along its rotating surface in contact with the balls, and the outer race may develop defects on its fixed track that produce repetitive impacts as the balls pass over them. Such structural faults alter the dynamic behavior of the bearing, leading to vibration anomalies and potentially resulting in mechanical failure. The "normal" bearing condition refers to a healthy bearing without any artificially induced damage.

Figure 3.2 illustrates the main components and geometric parameters of a typical ball bearing. The bearing assembly consists of an inner race, an outer race, and rolling elements (balls). In practice, these balls are held in place and evenly spaced by a cage (also called

a retainer or separator). It is important to note that the cage refers only to this guiding component, not to the entire bearing assembly. The outer diameter of the bearing is denoted by D, while the bore diameter of the inner race is d. The pitch diameter, d_m, represents the diameter of the circle passing through the centers of the rolling elements. Each rolling element has a ball diameter d_b, and the angular position of a ball relative to the bearing axis is defined by the contact angle β. The raceway has a finite width w, while the defect slot (when present) is characterized by its length L and depth h. In addition to the geometric dimensions of the bearing, the CWRU dataset introduces controlled artificial defects with predefined sizes, referred to as the *fault diameter*. Its role in defining severity levels can be summarized as follows:

- **Definition**: The *fault diameter* specifies the size of the machined pit or spall introduced on a bearing component.
- **Fault Severity**: Three levels of severity levels are used in the dataset with $0.007''$ ($\approx$0.18 mm, *small/low severity*), $0.014''$ ($\approx$0.36 mm, *medium/moderate severity*), $0.021''$ ($\approx$0.53 mm, *large/high severity*).
- **Fault Locations**: These defects, machined using Electrical Discharge Machining (EDM) are introduced on the ball, inner raceway, or outer raceway.
- **Healthy Baseline**: A bearing without any machined defect corresponds to the *healthy class*, serving as the reference for severity comparison.

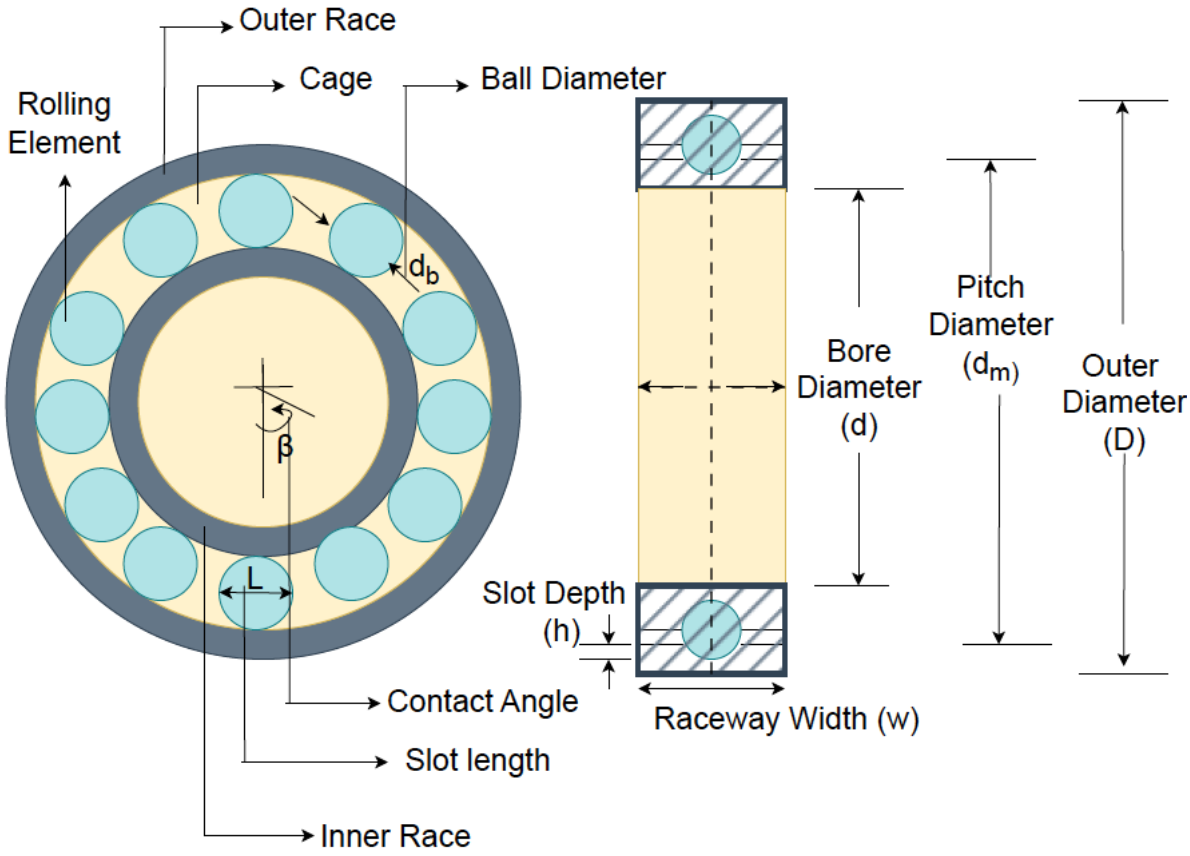


Figure 3.2: Illustration of a typical ball bearing showing the inner race, outer race, rolling elements (balls), cage, and shaft.

In the CWRU dataset, bearing faults are classified based on their position within the bearing structure and their failure mode:

- **Ball Fault**: As shown in Figure 3.3b ball fault refers to a localized defect such as a pit, spall, or crack on the surface of a rolling element (ball). The ball bearing remains in continuous rolling contact with both the inner and outer raceways, the defect repeatedly strikes or slides against these surfaces during rotation. This interaction generates periodic impact forces that give rise to distinctive vibration signatures.It typically produces vibrations at a moderate frequency range of about 150–250 Hz.
- **Outer Race Fault**: As illustrated in Figure 3.3c, an outer race fault refers to a

localized defect such as a pit, spall, or crack on the raceway surface of the stationary outer ring. Since the outer race does not rotate, the defect remains fixed in position while the rolling elements pass over it. Each time a ball traverses the defect, a periodic impact is generated at the ball-pass frequency of the outer race, giving rise to a characteristic vibration pattern. Unlike inner race faults, the resulting impacts are not directly modulated by shaft speed but are instead governed by the relative motion of the rolling elements across the stationary defect. An outer race fault forms on the stationary outer ring, producing vibrations at a lower frequency range of about 100–200 Hz, as the defect remains fixed while the rolling elements pass over it.

- **Inner Race Fault**: Figure 3.3d illustrates an inner race fault, which corresponds to a localized defect such as a pit, spall, or crack on the raceway surface of the rotating inner ring. Because the inner ring spins in direct synchrony with the shaft, the defect repeatedly interacts with the rolling elements at a relatively high frequency. Each rolling element striking the defect produces sharp impacts whose repetition rate is strongly governed by the shaft speed. Consequently, inner race faults are characterized by well-defined fault frequencies that are directly modulated by the rotational speed of the shaft. An inner race fault develops on the rotating inner ring, generating vibrations at a higher frequency range of about 250–400 Hz, since the defect moves with the shaft rotation.

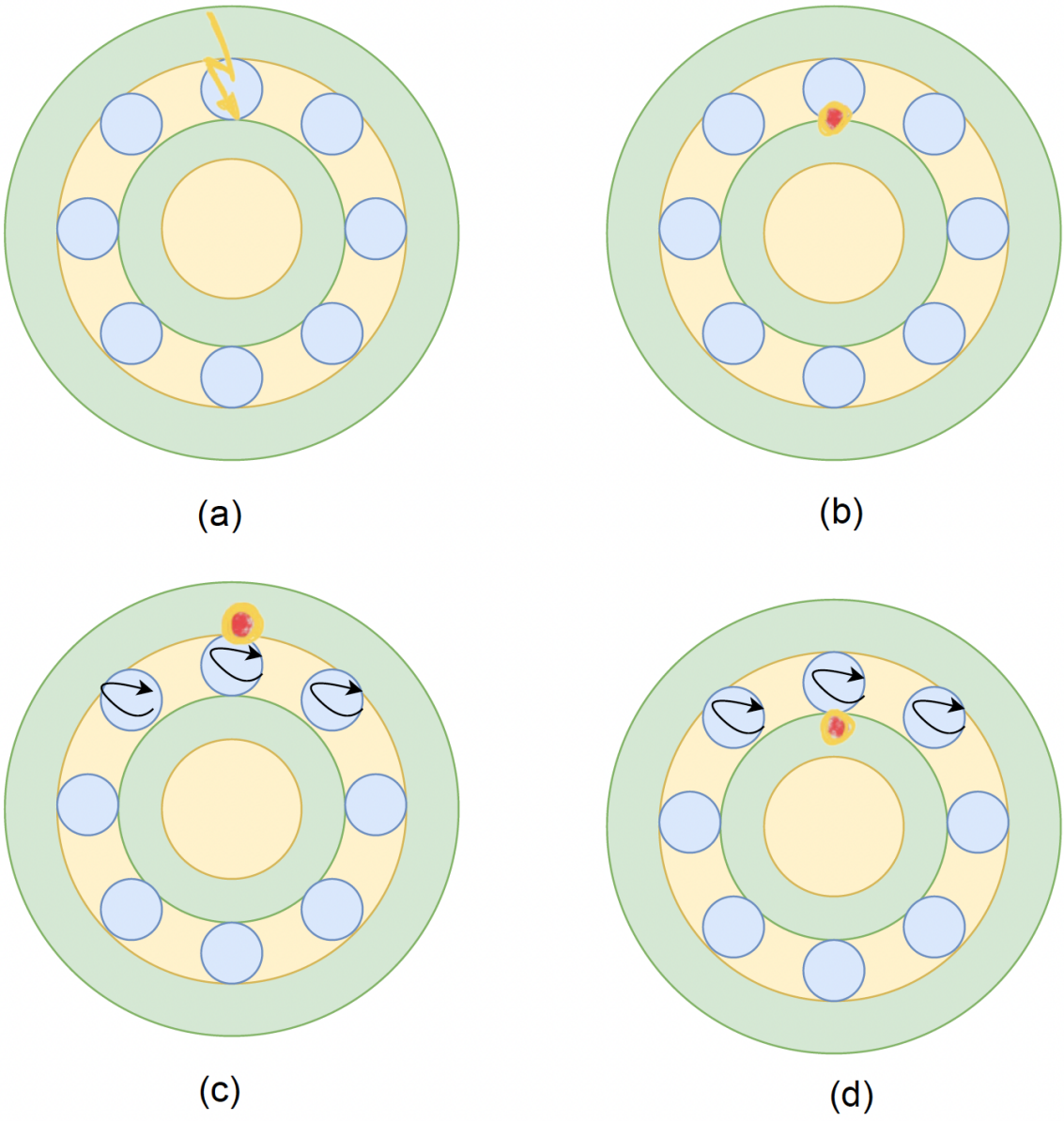


Figure 3.3: Defects are marked with small notches on the ball : (a) Electrical Discharge Machining (EDM) (b) Ball fault, (c) Outer race fault, (d) Inner race fault

Each of the fault types shown in Figure 3.3 is represented in the CWRU dataset with

three predefined defect sizes created using Electrical Discharge Machining (EDM), a precision process that removes material with controlled electrical sparks to produce small, accurately placed pits or spalls on the bearing surface. In addition to the healthy (fault-free) condition, defects of diameter 0.007″, 0.014″, and 0.021″ which are basically the level of fault severity being introduced, corresponding to *small*, *medium*, and *large* fault severities, respectively, relative to the normal bearing.

Table 3.2[1] lists the mapping between each fault type–severity combination and its unique class ID, along with an additional class representing healthy bearings.

Table 3.2: Fault type and corresponding fault severity levels.

Fault Type	**Fault Diameter (in)**
Ball Fault	0.007″
Ball Fault	0.014″
Ball Fault	0.021″
Inner Race Fault	0.007″
Inner Race Fault	0.014″
Inner Race Fault	0.021″
Outer Race Fault	0.007″
Outer Race Fault	0.014″
Outer Race Fault	0.021″
Normal (Healthy)	–

[1] *Fault diameters indicate the sizes of EDM-induced defects on the specified bearing component (ball, inner race, or outer race) to represent different damage severities. These values are not to be confused with the rolling element (ball) diameter or the pitch diameter of the bearing.*

3.2.1 Representative Signal Characteristics

This section illustrates representative examples of vibration measurements for healthy and faulty bearings in both the time and frequency domains.

Figure 3.4 presents raw time-domain signals for a healthy bearing and three ball fault severities ($0.007''$, $0.014''$, and $0.021''$). As defect size increases, the signals exhibit stronger periodic impulses with higher amplitudes, corresponding to repeated impacts of the damaged ball on the raceways.

Complementarily, Figure 3.5 shows the corresponding Fast Fourier Transform (FFT) spectra. Here, the presence of harmonics and sidebands around the ball-pass frequency becomes more evident with increasing defect size, distinguishing faulty conditions from the healthy baseline.

Together, these visualizations highlight the discriminative fault signatures that are exploited by time–frequency domain analysis in subsequent feature extraction steps.

- **Time domain:** As defect size increases, the signals exhibit more pronounced periodic impulses with higher amplitudes. These impacts reflect repetitive mechanical shocks caused by the defect striking the raceways.
- **Frequency domain:** The FFT spectra reveal fault-specific harmonics and sidebands emerging around the ball-pass frequency. With increasing severity, these spectral features become stronger and more distinct, distinguishing faulty states from the healthy baseline.

Together, these visualizations highlight the discriminative fault signatures captured in the CWRU dataset, reinforcing the importance of time–frequency analysis for bearing fault diagnosis.

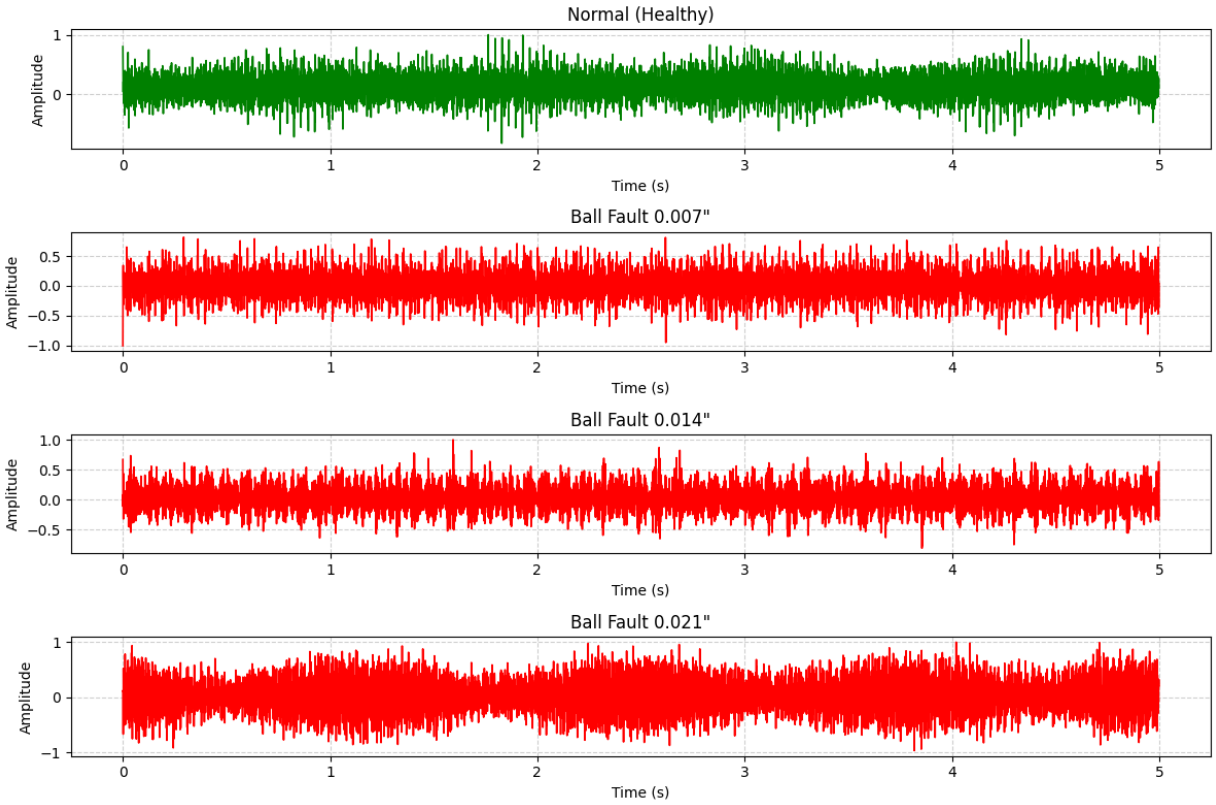


Figure 3.4: Time-domain representations of ball faults compared with a healthy condition.

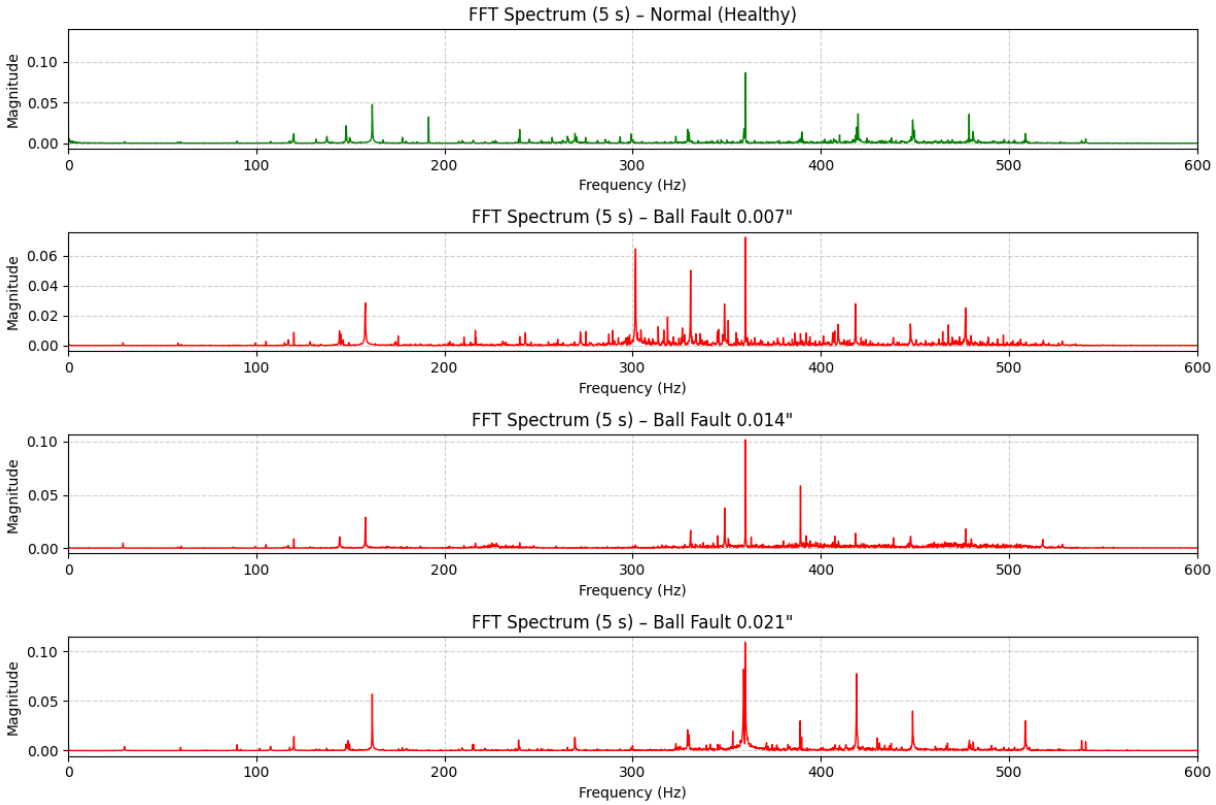


Figure 3.5: Frequency-domain representations of ball faults compared with a healthy condition.

Similar illustrations can be constructed for inner-race and outer-race faults, highlighting differences in how each defect type excites the bearing structure. For example, inner-race faults are strongly modulated by shaft speed, producing distinct high-frequency harmonics, whereas outer-race faults exhibit repeated impacts tied to the ball-pass frequency of the outer race. These qualitative differences form the physical basis for subsequent feature extraction using time–frequency analysis (see Section 4.3).

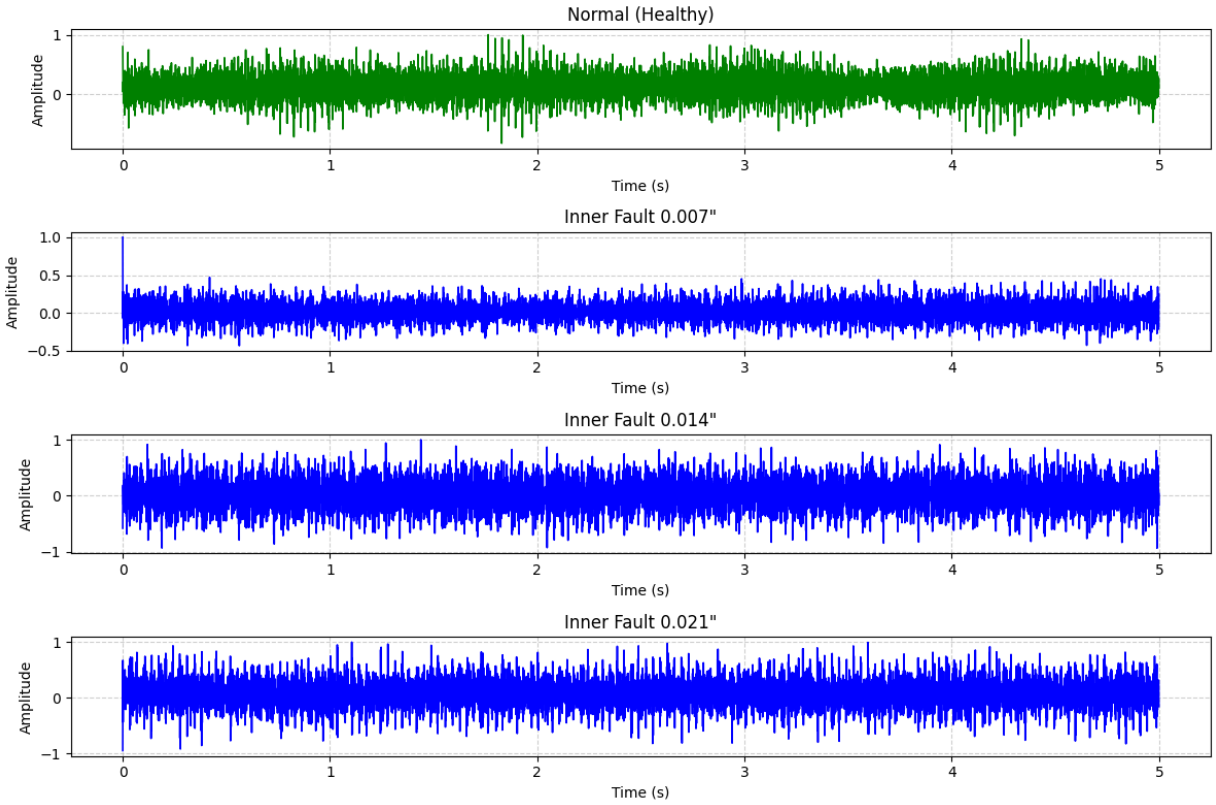


Figure 3.6: Time-domain representations of inner-race faults compared with a healthy condition.

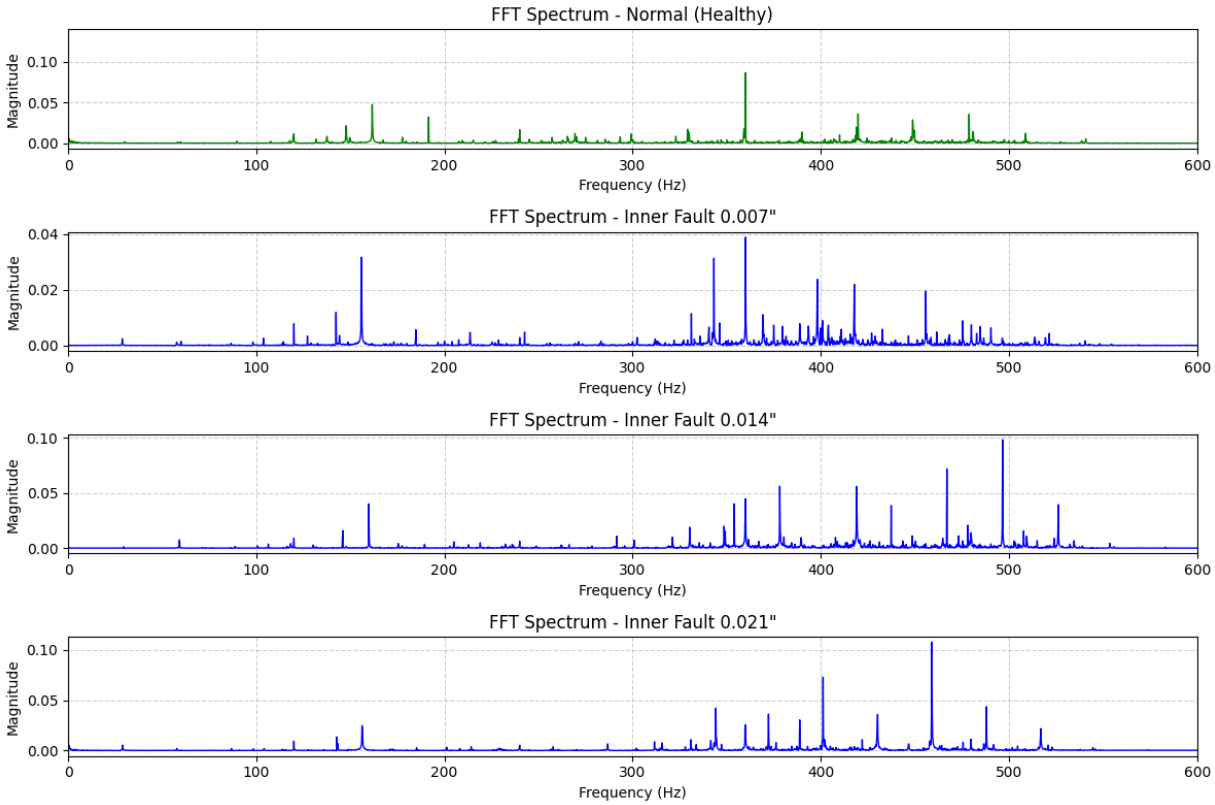


Figure 3.7: Frequency-domain representations of inner-race faults compared with a healthy condition.

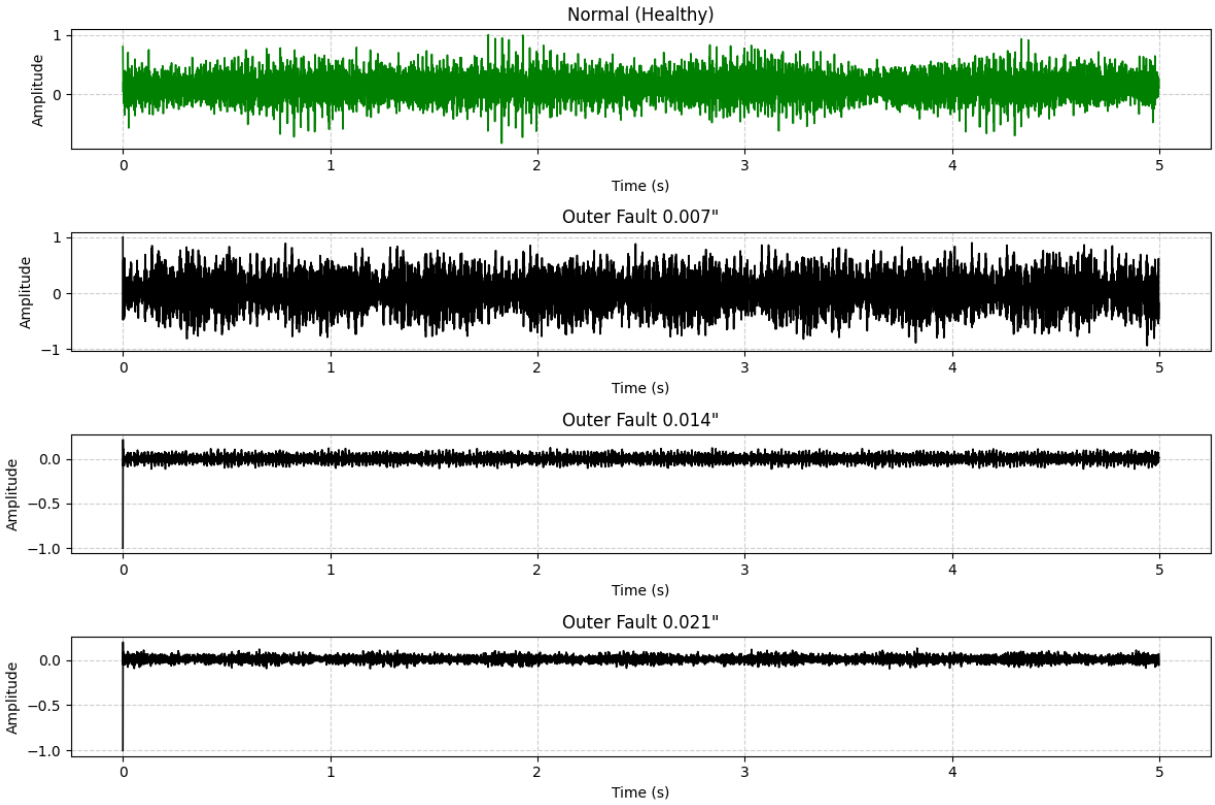


Figure 3.8: Time-domain representations of outer-race faults compared with a healthy condition.

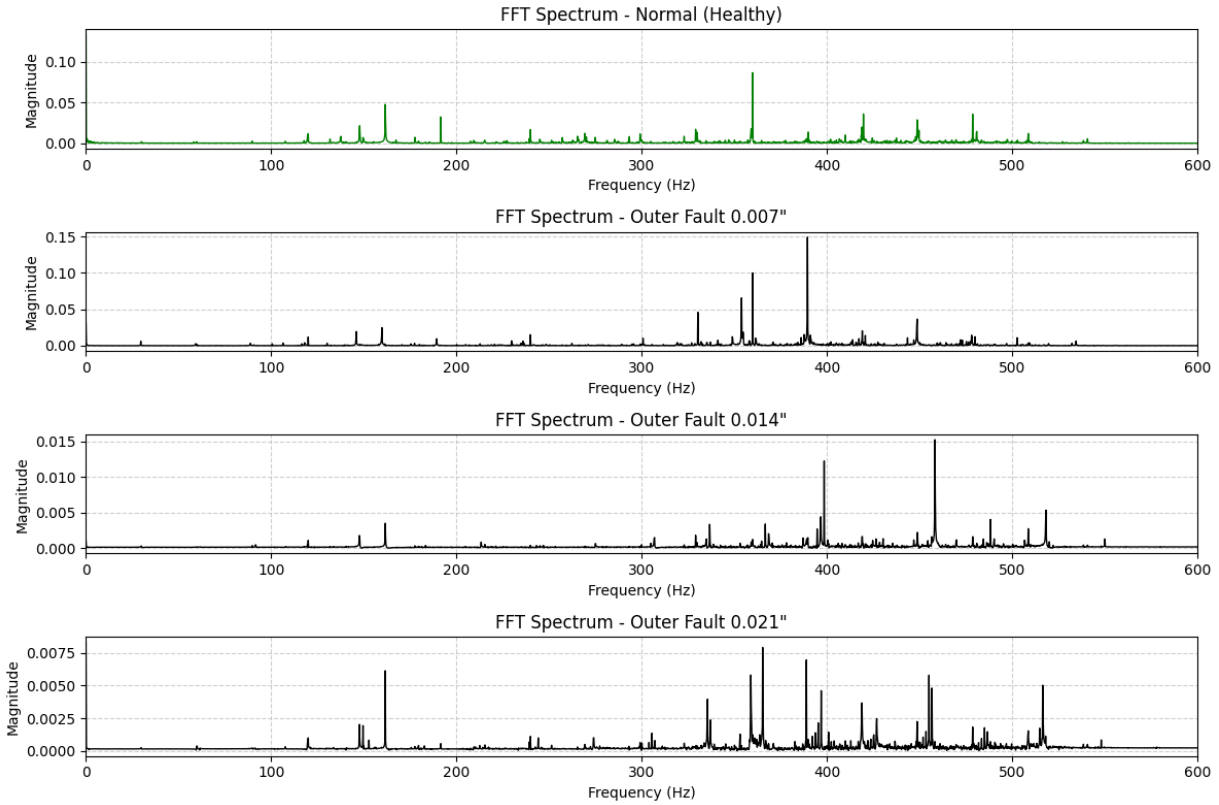


Figure 3.9: Frequency-domain representations of outer-race faults compared with a healthy condition.

While the time- and frequency-domain views provide complementary insights, they do not fully capture how spectral content evolves over time. To address this, Short-Time Fourier Transform (STFT) spectrograms are used to generate time–frequency representations of vibration signals. These spectrograms highlight how transient fault signatures distribute energy across both time and frequency, enabling better discrimination between healthy and faulty conditions.

The healthy condition exhibits relatively low and diffuse spectral energy, with no dominant fault-related bands. In contrast, faulty signals show distinct localized energy concentrations near the ball-pass frequency (approximately 236 Hz), whose intensity and spread increase with defect size. This progressive amplification provides strong evidence of fault severity, and highlights the discriminative power of time–frequency analysis for bearing fault detection.

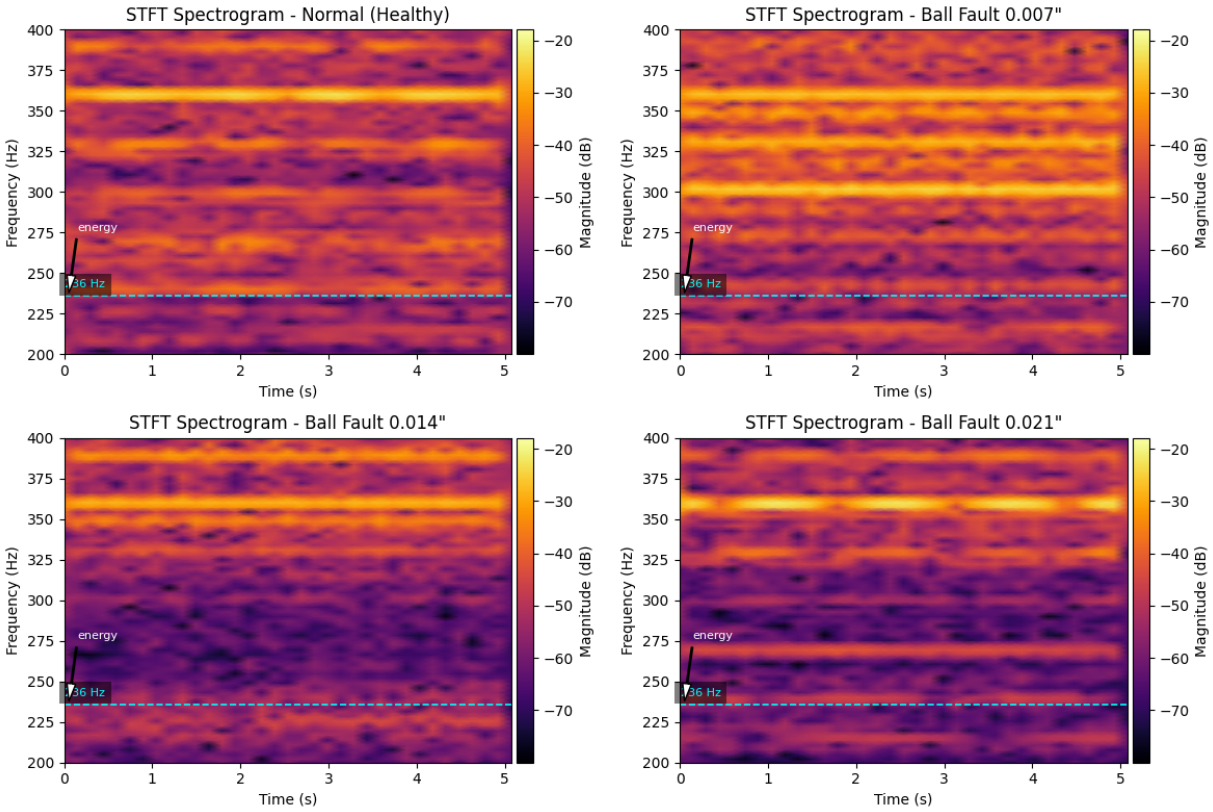


Figure 3.10: STFT spectrograms of ball faults at different severities compared with a healthy baseline. Fault conditions show localized energy concentrations near the ball-pass frequency (236 Hz), absent in the healthy case.

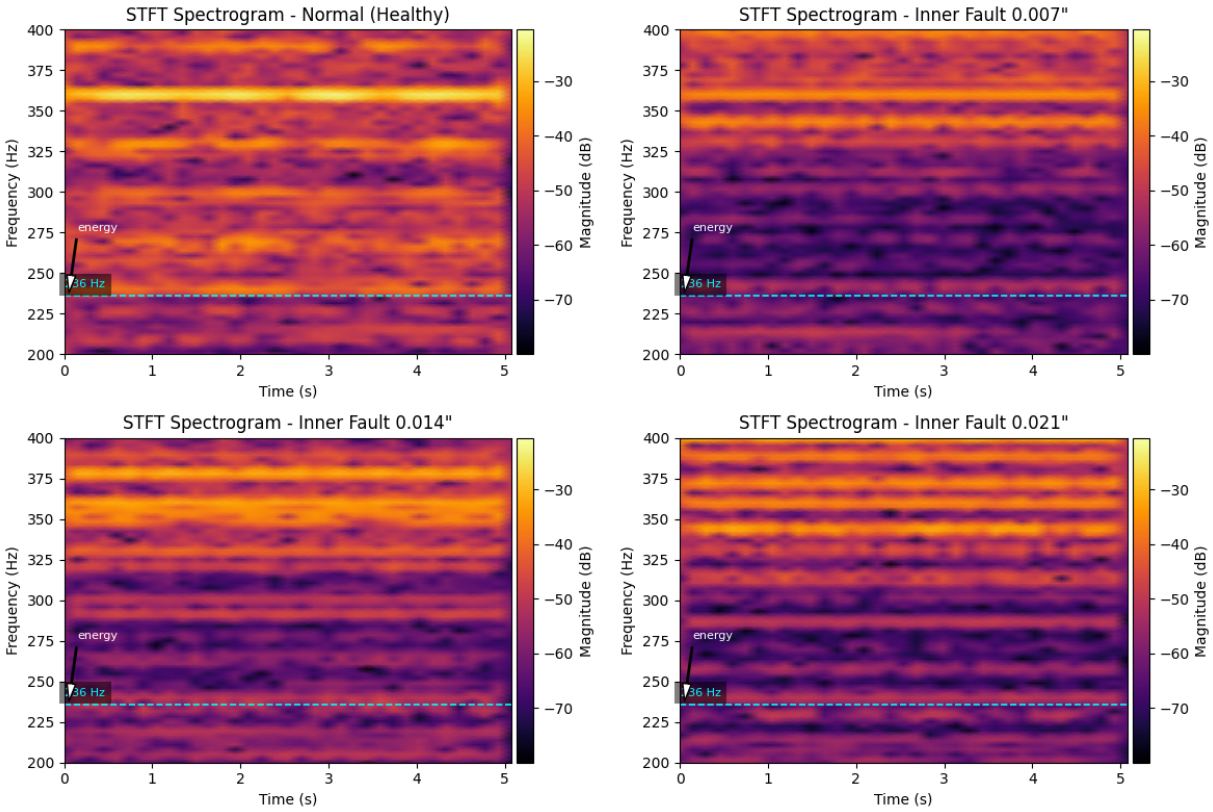


Figure 3.11: STFT spectrograms of inner-race faults compared with a healthy baseline. Inner-race defects produce strong, shaft-speed-modulated harmonics, resulting in sharper spectral bands than those observed in ball faults.

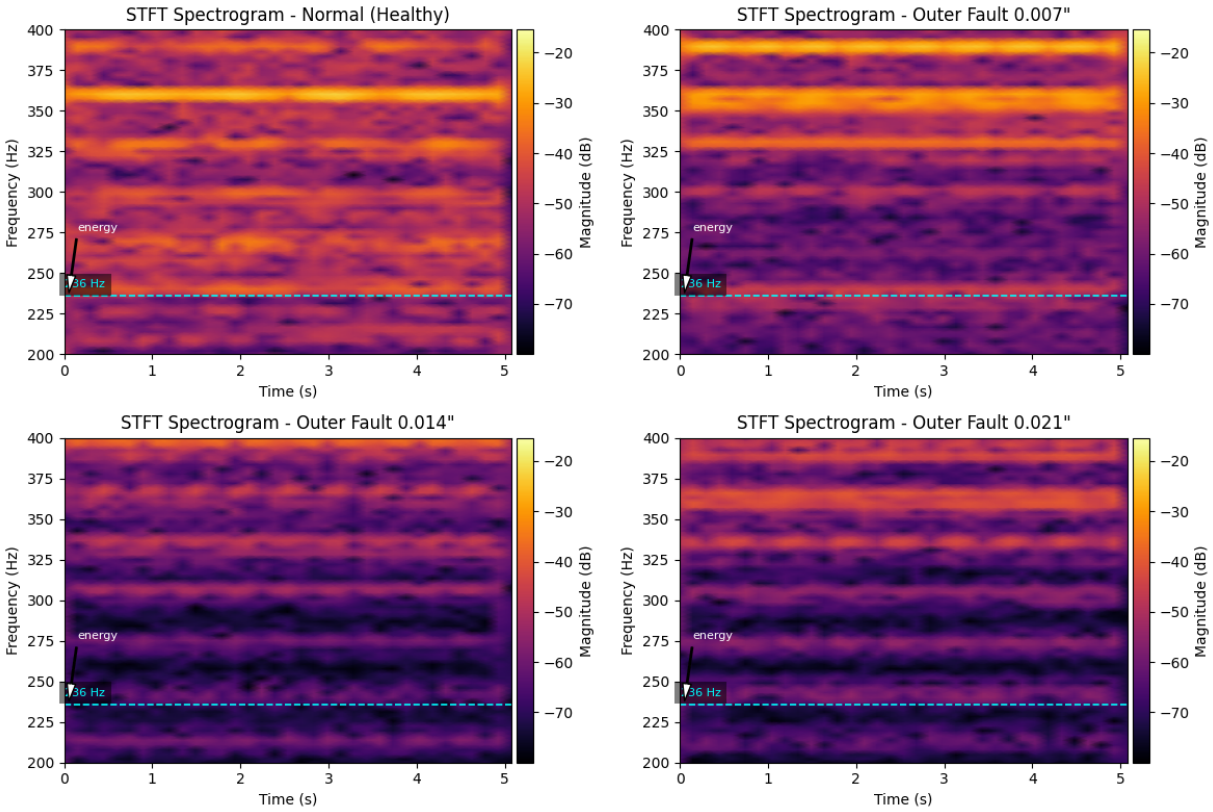


Figure 3.12: STFT spectrograms of outer-race faults compared with a healthy baseline. Outer-race defects exhibit periodic bursts of energy at the outer race ball-pass frequency, distinct from the modulation patterns of inner-race faults.

The representative spectrograms reveal distinct excitation patterns for each fault type: ball faults show localized energy near the ball-pass frequency, inner-race faults display shaft-speed–modulated harmonics, and outer-race faults exhibit periodic energy bursts at the outer-race frequency, while the healthy condition shows diffuse, uniform energy. These distinctions highlight the discriminative nature of vibration signatures and justify using STFT-based time–frequency representations for feature extraction and subsequent deep learning analysis (Section 4.3).

3.3 Objective:

The objective is to develop and analyse an efficient method for early fault detection using feature extraction technique(s) and transfer learning of CNN model(s).

Chapter 4

Methodology

The chapter is organized as follows. Section 4.1 outlines the data preparation steps applied to vibration signals obtained from the physical setup. Section 4.2 describes the overall methodology, while Section 4.3 explains the feature extraction process and the construction of input representations, including the single-channel spectrogram described in Section 4.3.1. Finally, Section 4.4 presents the classification model architecture based on the proposed single-channel spectrogram. Together, these sections provide a comprehensive overview of the methodology employed in this study.

4.1 Dataset Preparation

The dataset used in this study is the benchmark Case Western Reserve University (CWRU) bearing fault dataset. It was collected by the CWRU Bearing Data Center using a controlled test rig comprising a 2 hp induction motor, torque transducer, and dynamometer, where faults were artificially introduced on the bearing components through electrical discharge machining (EDM). Vibration signals were recorded from accelerometers mounted on the drive end, fan end, and base housing under various motor loads and speeds (0–3 hp, 1797–1730 rpm) at two different sampling rates of 12 kHz and 48 kHz.

The downloaded dataset is organized as a collection of MATLAB `.mat` files, each containing time-series recordings from three locations: the drive end (DE), fan end (FE), and base acceleration (BA) points of the motor housing. Every file corresponds to a specific combination of operating load, fault type, and fault severity. Each vibration recording has an approximate duration of two minutes and the structure of the data is as shown in the

4.1. Each variable name adopts the format `X_DE_time`, `X_FE_time`, or `X_BA_time`, where `X` denotes the file index corresponding to a particular experimental configuration.

Table 4.1: Sample metadata from the CWRU bearing dataset showing vibration signal arrays for Bearing X282 (Drive End, Fan End, and Base Acceleration sensors).

X282_DE_time	**X282_FE_time**	**X282_BA_time**
-0.16812035928143700	0.31966649350649400	0.017306330365974300
0.18127760479041900	0.32617038961039000	-0.0037027497527200800
0.04434479041916170	-0.26048103896103900	-0.16911907022749800
-0.2704544910179640	0.031056103896103900	-0.06946680514342240

Note: Although the dataset stores high-precision floating-point samples, the training pipeline operates with 32-bit (FP32) or mixed-precision (FP16) formats.

The collected vibration signals are represented as

$$\mathcal{S} = \{s_1, s_2, \ldots, s_M\}, \tag{4.1}$$

where each s_i denotes the i-th recorded signal instance.

Since the collected data $\mathcal{S}$ consists of signals sampled at two different frequencies 12 kHz and 48 kHz all signals were uniformly resampled to a common target frequency of f_s^{target}= 12 kHz in this study to ensure consistency across all operating conditions. This process is referred to as resampling.

Resampling is one of the most important preprocessing steps. In practice, signals being initally collected from the systems is considered as are often recorded at different sampling frequencies. This inconsistency makes it challenging to compare or combine signals collected under varying conditions. To ensure uniformity, each s_i is resampled to a common target sampling frequency f_s^{target}.

If $s_i(t)$ denotes the continuous-time version of the discrete signal s_i, the uniformly resampled sequence is expressed as

$$s_{i,r}[n] = s_i\left(\frac{n}{f_s^{\text{target}}}\right), \quad n = 0, 1, 2, \ldots \tag{4.2}$$

where n denotes the discrete-time sample index of the resampled signal $s_{i,r}[n]$, corresponding to time instants spaced at intervals of $1/f_s^{\text{target}}$ seconds. Where $s_{i,r}[n]$ represents the resampled discrete-time sequence indexed by n. Here, $f_{s,i}^{\text{orig}}$ is the original sampling frequency at which s_i was acquired, and f_s^{target} is the common target sampling frequency to which all

signals are resampled. This process yields a consistent dataset $\mathcal{S}_r = \{s_{1,r}, s_{2,r}, \ldots, s_{M,r}\}$, where each $s_{i,r}$ has a uniform sampling rate and can be directly compared or used in subsequent feature extraction and learning stages.

$$\mathcal{S}_r = \{s_{1,r}, s_{2,r}, \ldots, s_{M,r}\}, \tag{4.3}$$

where each $s_{i,r}$ is obtained through the resampling operation defined in Eq. (4.2).

Resampling provides two main benefits:

- **Standardization**: ensures that all signals s_i share the same temporal resolution, enabling consistent segmentation and feature extraction across the dataset.
- **Efficiency**: avoids redundant information from overly high sampling rates while still preserving the essential fault-related dynamics.

Each recording was associated with metadata describing its fault type, severity, load condition, and sensor position, enabling systematic class labeling. In this work, all vibration recordings were grouped into ten distinct fault classes, corresponding to the combinations of fault type (ball, inner race, outer race) and severity levels (0.007″, 0.014″, 0.021″)/(low, medium, high), along with the healthy condition. A detailed taxonomy of these classes is presented in Section 3.2 (see Table 4.2).

Table 4.2: Mapping of fault types and severities in the CWRU dataset.

Class ID	**Fault Type**	**Fault Diameter (in)**
0	Ball Fault	0.007″
1	Ball Fault	0.014″
2	Ball Fault	0.021″
3	Inner Race Fault	0.007″
4	Inner Race Fault	0.014″
5	Inner Race Fault	0.021″
6	Outer Race Fault	0.007″
7	Outer Race Fault	0.014″
8	Outer Race Fault	0.021″
9	Normal (Healthy)	–

As discussed earlier in Section 3, Figures 3.4–3.9 and Figures 3.10–3.12 illustrate representative time-, frequency-, and time–frequency-domain signatures for various fault locations and severities in the CWRU dataset. These plots highlight how vibration energy

varies over characteristic fault cycles, typically spanning a few seconds of recording. Accordingly, each signal in this study was divided into 5-second segments i.e., 60,000 samples per window, as the total duration of the start- steady and off state this segment length is sufficiently long to capture at least one full vibration cycle and with a 1-second hop with 12,000-sample stride between consecutive windows, ensuring dense temporal coverage and preventing any portion of the signal from being missed this chosen hop size preserves temporal continuity addressing the $\mathbf{C_2}$

Overall, this configuration balances temporal context with computational efficiency, ensuring that localized spectral patterns associated with specific fault types are retained within each analysis window.

Segmentation: In the multi-sensor configuration used in this study, vibration signals were simultaneously collected from three accelerometers mounted along single axes at the *drive end (DE)*, *fan end (FE)*, and *base (BA)* locations of the motor housing. Each resampled signal corresponding to a given sensor is denoted by $s_{i,r}^{(c)}$, where $c \in \{\mathrm{DE}, \mathrm{FE}, \mathrm{BA}\}$ identifies the accelerometer channel. Following the segmentation process, the resulting 5-second segments $x_{ij}^{(c)}$ from each channel are temporally aligned such that the j-th segment from every sensor corresponds to the same time interval. When data from all three sensors are employed, these aligned segments are stacked along a new channel dimension to form a three-channel tensor:

$$x_{ij} = [x_{ij}^{(\mathrm{DE})}, x_{ij}^{(\mathrm{FE})}, x_{ij}^{(\mathrm{BA})}]^\top \in \mathbb{R}^{3\times N_{\mathrm{seg}}}, \tag{4.4}$$

which serves as the synchronized input structure for the machine learning models. This tensor formulation ensures that each training sample contains temporally aligned 5-second vibration information from all three sensor locations, allowing the network to jointly learn cross-sensor dependencies and spatial relationships among sensors.

Note. The superscript c denotes the *physical accelerometer channel* (DE, FE, BA) used during data acquisition. This is distinct from later CNN-level notations, where d refers to the *input-channel index* in convolutional layers, and C_f denotes the *number of learned feature channels* produced by the final convolutional block.

To capture localized temporal variations within each synchronized signal, the resampled signals from the set $s_{i,r}[n]$ are divided into overlapping windows. Building on the 5-second segment and 1-second hop configuration described earlier, each signal is segmented into smaller windows to preserve both short-term and long-term temporal patterns. Each window has a fixed length of N_{seg} samples and a hop size of H_{hop} samples, where $H_{\mathrm{hop}} < N_{\mathrm{seg}}$. Here, x'_{ij} denotes the windowed signal segment itself, while N_{seg} specifies the segment length in samples.

This segmentation process arises from the multi-sensor representation in Eq. 4.4, where each channel-specific signal contributes to the final synchronized tensor. The j-th segment extracted from $s_{i,r}[n]$ is mathematically defined as

$$x'_{ij}[n] = s_{i,r}[n + jH_{\text{hop}}], \quad 0 \leq n < N_{\text{seg}}, \tag{4.5}$$

where $x'_{ij}[n]$ represents the j-th segment of length N_{seg} extracted from the i-th resampled signal $s_{i,r}[n]$, with n indexing the samples within the segment. This segmentation step increases the number of training samples and preserves temporal continuity, which is particularly useful in fault diagnosis since fault signatures may appear intermittently within long vibration recordings.

This segmentation step increases the number of training samples and preserves temporal continuity, which is particularly useful in fault diagnosis since fault signatures may appear intermittently within long vibration recordings.

Table 4.3: Original distribution of samples across fault types and diameters in the CWRU dataset.

Class ID	**Count**
0	182
1	182
2	182
3	182
4	174
5	182
6	470
7	90
8	398
9	38
Total	**2,080**

As seen in Table 4.3, the dataset is not uniformly distributed across fault categories, which has implications for the design and evaluation of classification models. To provide a clearer visual understanding of this imbalance, Figure 4.1 illustrates the relative proportion of samples corresponding to each fault type and severity level, including the healthy condition. The dominance of outer race faults—particularly those with smaller defect diameters— is evident from the pie chart, whereas the normal condition constitutes only a small fraction of the total dataset.

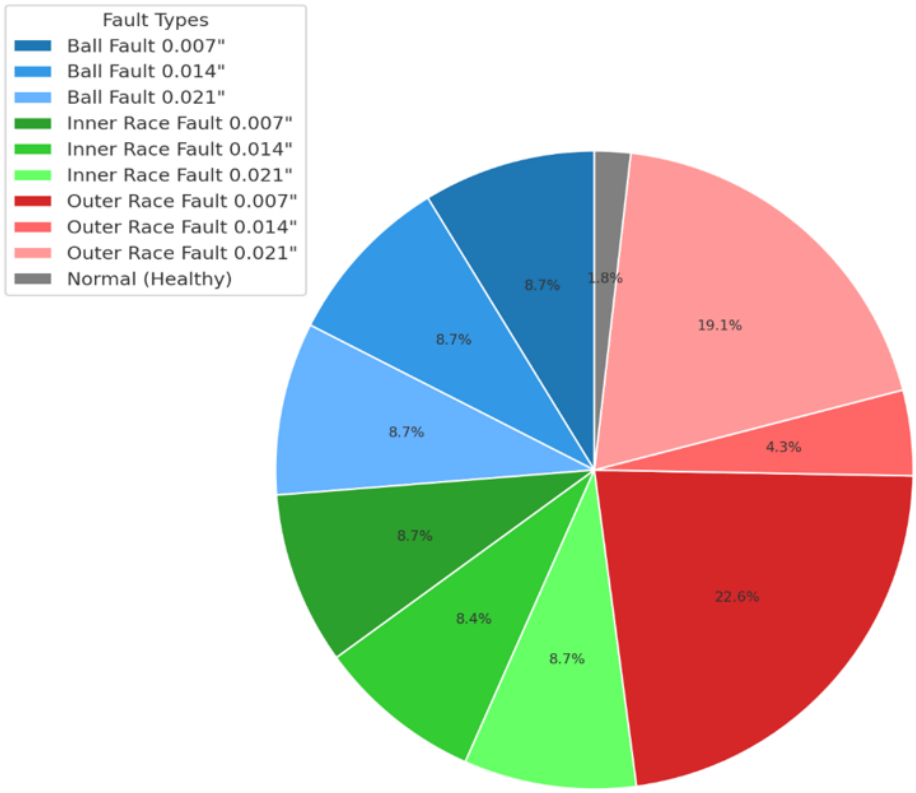


Figure 4.1: Relative class distribution across all fault types and the healthy condition.

To further improve model robustness, data augmentation is applied as the data is not sufficient enough for the model training. This step also simulates variations that commonly occur in real-world operating environments, such as environmental noise, sensor misalignment, or fluctuations in load and speed conditions. Moreover, as evident from the original class distribution (Table 4.3), the dataset contains a relatively small number of samples with noticeable imbalance across different fault classes—particularly between outer race faults of varying severities and the healthy condition. To mitigate these limitations and enhance the generalization capability of the model, data augmentation is employed to synthetically expand the dataset and achieve a more balanced representation of all fault categories.

Following segmentation, data augmentation was applied to the vibration signal segments to mitigate overfitting and improve generalization, especially given the limited sample count for certain fault classes such as the healthy (class 9) and medium-severity outer

race faults (class 7). This process enriches the training set by simulating realistic variations in amplitude, timing, and noise conditions that commonly occur in industrial machinery.

A generalized augmentation process can be expressed as:

$$\tilde{x}'_{ij}[n] = \alpha \cdot x'_{ij}[n - \delta] + \eta[n], \tag{4.6}$$

where $\tilde{x}'_{ij}[n]$ represents the augmented version of the multi-sensor segment $x'_{ij}[n]$ derived from the j-th window of the i-th resampled signal $s_{i,r}[n]$. Here, α is an amplitude-scaling factor, δ is a temporal shift, and $\eta[n]$ denotes additive noise. These operators collectively define the augmentation framework applied to the synchronized multi-sensor segments.

Implemented Augmentation Methods. Five complementary techniques were employed to generate realistic signal variations:

- **Gaussian noise injection:** The additive noise term $\eta[n]$ in Eq. 4.6 was modeled as white Gaussian noise with zero mean and a standard deviation of 0.01. This simulates electrical and mechanical interference often present in real vibration measurements, forcing the model to extract features robust to ambient or sensor-induced noise.
- **Random masking:** Approximately 10% of samples within each segment $\tilde{x}'_{ij}[n]$ were randomly zeroed out to emulate sensor disconnections, loose wiring, or transient measurement dropouts. This encourages the network to learn invariant representations even when partial data are missing.
- **Time translation:** To simulate asynchronous sampling or phase shifts in vibration signals, each segment was circularly shifted by a random temporal offset δ. This preserves the periodic structure of fault-induced impacts while reducing over-dependence on precise temporal alignment.
- **Amplitude scaling:** Each segment $\tilde{x}'_{ij}[n]$ was multiplied by a random scaling factor $\alpha \in [0.9, 1.1]$ to simulate variations in sensor gain or amplitude drift that commonly occur across different operating conditions. This transformation ensures that the classifier remains insensitive to absolute signal magnitudes and focuses instead on relative spectral features.
- **Temporal shifting:** Segments were circularly shifted in time according to the offset parameter δ defined in Eq. 4.6. This simulates small temporal misalignments due to asynchronous sampling or mechanical phase differences, while preserving the overall periodic structure of the fault impacts. By incorporating such shifts, the model learns to recognize fault patterns independent of their temporal position within the signal.

Collectively, these augmentation techniques introduce stochastic variations in amplitude, phase, and frequency content, thereby enhancing the network's ability to generalize across unseen operating conditions and preventing overfitting to specific experimental noise characteristics.

Global Augmentation Strategy: In standard deep learning pipelines, augmentation is typically applied only to the training set. However, for small-scale vibration datasets, such a restriction may not provide enough diversity for meaningful model training and evaluation. To overcome this, a global augmentation strategy was adopted—applying all augmentation operators to the entire dataset prior to the train–test split—to achieve a larger and more representative sample distribution across all classes.

This approach aligns with prior work in fault diagnosis. Afrasiabi *et al.* [38] injected Gaussian noise into both training and testing data to evaluate model robustness under real-time variations, while Lee *et al.* [39] introduced synthetic variability during preprocessing to improve fault severity estimation, effectively augmenting both training and evaluation sets. Such methods prioritize robustness and reliability over strict benchmarking and are well-suited to constrained-data scenarios like the CWRU dataset.

Each augmented segment $\tilde{x}_{ij}$ retained its corresponding class label y_{ij}, and the process resulted in a significantly expanded dataset.

Class Distribution After Augmentation. After applying all augmentation methods globally, the dataset expanded to 12,480 labeled segments across 10 fault classes, as summarized in Table 4.4.

Table 4.4: Final class distribution after applying all augmentation methods.

Fault Class ID	Segment Count
0	1,092
1	1,092
2	1,092
3	1,092
4	1,044
5	1,092
6	2,820
7	540
8	2,388
9	228
Total	12,480

This global augmentation approach alleviated class imbalance, diversified the feature space, and provided a robust foundation for supervised model training.

The complete preprocessing pipeline adopted in this work (as illustrated in Fig. 5.1) is summarized in Fig. 4.2. Starting from the raw vibration recordings, the following sequential steps were performed:

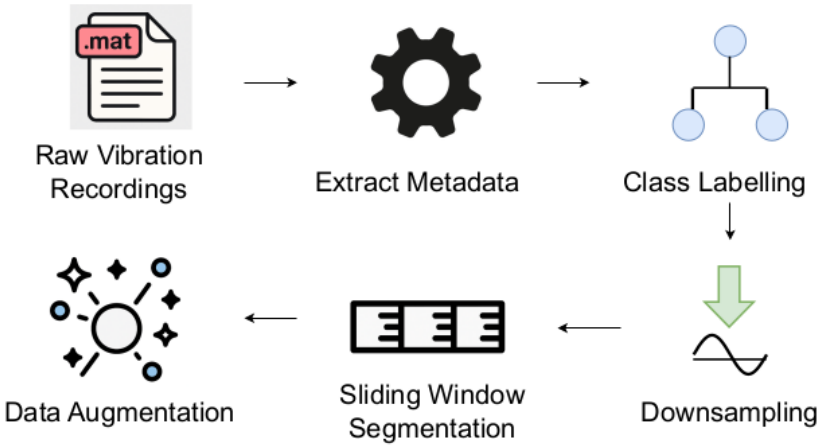


Figure 4.2: Overview of the data preparation pipeline.

4.2 Proposed Methodology

The first stage in the proposed approach is signal preprocessing, which prepares raw vibration data for subsequent analysis. Since vibration recordings can vary depending on acquisition systems, sensor placement, and operating conditions, preprocessing ensures that the input signals are consistent, comparable, and suitable for feature extraction.

The proposed approach leverages time–frequency domain representations of raw vibration signals, enabling the extraction of discriminative features that characterize both stationary and transient fault patterns. The time-domain vibrational signal one-dimensional are transformed into spectrograms using the Short-Time Fourier Transform (STFT), which provides a study of spatio-temporal aspects. These spectrograms are then used feature representations: single-channel image features which is futher described in Section 4.3.1. A transfer-learning approach with the XceptionNet-71 is adopted to study 4.4.1

4.3 Feature Extraction

Vibration signals collected from rotating machinery are inherently non-stationary, with their spectral properties changing over time due to varying load conditions, mechanical wear, or the presence of faults. Analyzing these signals purely in the time domain can obscure critical fault-related features. Therefore, a time–frequency representation using the Short-Time Fourier Transform (STFT) is adopted to preserve both temporal and spectral characteristics.

To better understand the limitations of time-domain analysis, Figure 4.3 shows a vibration signal from a bearing fault recorded over 10 seconds. The waveform appears noisy and stochastic, making it difficult to visually identify patterns or anomalies. Although some irregularities might be present, they are not easily distinguishable without further processing.

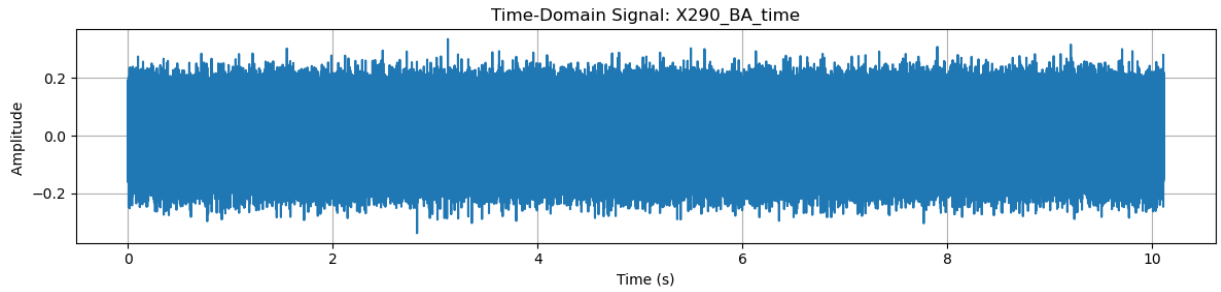


Figure 4.3: Time-domain vibration signal

To uncover the underlying frequency components, the Fast Fourier Transform (FFT) is applied to the same signal. As shown in Figure 4.4, several dominant spectral peaks become visible, particularly around 3700–4200 Hz. However, the FFT only provides a global frequency summary and does not indicate when these frequencies occur. This lack of temporal localization limits its effectiveness for analyzing transient or evolving fault signatures in rotating machinery.

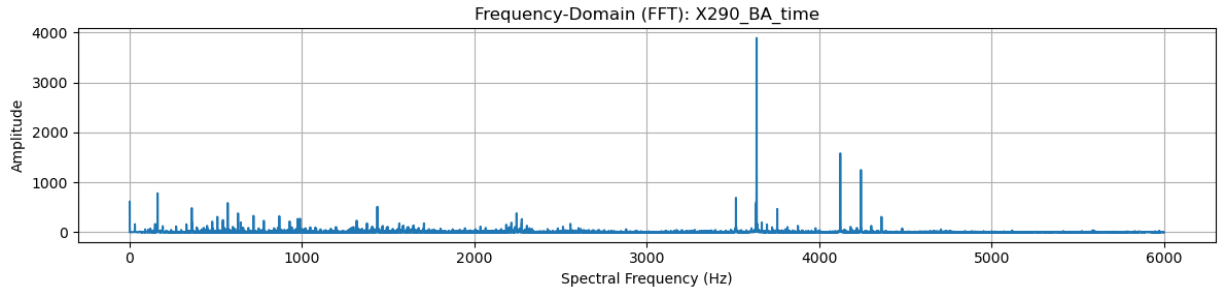


Figure 4.4: Frequency-domain signal (FFT)

These limitations motivate the use of time–frequency methods. The segmented and augmented vibration signals are therefore transformed into the time–frequency domain to capture both spectral and temporal characteristics. This transformation follows the overall data-processing pipeline:

To overcome this limitation, the *Short-Time Fourier Transform (STFT)* [35] is employed. The STFT applies the Fourier Transform within short, overlapping windows, thereby producing a two-dimensional representation that captures both temporal and spectral characteristics. This makes it particularly suitable for vibration-based fault diagnosis,

where localized defects manifest as non-stationary events. Formally, for a discrete-time segment $\tilde{x}'_{ij}$, the STFT is defined as

$$X_{ij}(\ell, \omega) = \sum_{n=-\infty}^{\infty} \tilde{x}'_{ij}[n]\, w[n - \ell]\, e^{-j\omega n}, \tag{4.7}$$

Each augmented multi-sensor segment $\tilde{x}'_{ij}$ corresponds to a finite-length discrete-time signal, i.e.,

$$\tilde{x}'_{ij} = [\tilde{x}'_{ij}[0],\, \tilde{x}'_{ij}[1],\, \ldots,\, \tilde{x}'_{ij}[N_{\text{seg}} - 1]].$$

The key components involved in the STFT operation are summarized as follows:

- n: discrete-time sample index within the segment $\tilde{x}'_{ij}$. It iterates over all time samples within the 5-second window.
- ℓ: frame index corresponding to the sliding window position. Shifting ℓ slides the analysis window along the signal, enabling localized spectral analysis at successive time instants.
- $w[\cdot]$: analysis window function (e.g., Hamming or Hann) that selects a localized subset of samples around the current frame position ℓ.
- ω: normalized angular frequency (in radians per sample) representing the frequency axis in the time–frequency plane.

The window function $w[n - \ell]$ therefore extracts a local segment centered around ℓ, and the resulting $X_{ij}(\ell, \omega)$ is a complex-valued function representing the joint time–frequency distribution of the segment $\tilde{x}'_{ij}$.

The time–frequency representation obtained from the STFT in Eq. 4.7 is converted into a feature map by taking the magnitude (or power) of the complex-valued coefficients:

$$X_{ij}(\ell, \omega) = \left|X_{ij}(\ell, \omega)\right|^2, \tag{4.8}$$

yielding a two-dimensional time–frequency energy map in which each coordinate (ℓ, ω) encodes the signal energy at a specific time frame and frequency bin.

To enhance visual contrast and emphasize low-energy components, a logarithmic compression is applied:

$$X_{ij}(\ell, \omega) \leftarrow \log\big(1 + X_{ij}(\ell, \omega)\big), \tag{4.9}$$

which reduces the dynamic range of spectral magnitudes while preserving discriminative fault-related patterns.

Finally, each X_{ij} is normalized and resized to a fixed resolution for compatibility with the input layer of deep learning models. Mean–variance normalization is commonly adopted:

$$X_{ij}(\ell, \omega) \leftarrow \frac{X_{ij}(\ell, \omega) - \mu}{\sigma}, \tag{4.10}$$

where μ and σ denote the mean and standard deviation of all spectrogram values within a segment. The resulting standardized feature tensors X_{ij} form the model-ready representations used for supervised fault classification.

The given figure below summarizes the feature extraction phase of this study, illustrating the transformation pipeline from raw vibration signals to model-ready feature tensors. Starting from the augmented multi-sensor segments $\tilde{x}'_{ij}$, the Short-Time Fourier Transform (STFT) (Eq. 4.7) is applied to obtain localized time–frequency representations. These are then converted into magnitude spectrograms, logarithmically compressed, and normalized (Eqs. 4.9–4.10) to produce the standardized feature maps X_{ij} that serve as the inputs to the deep learning models.

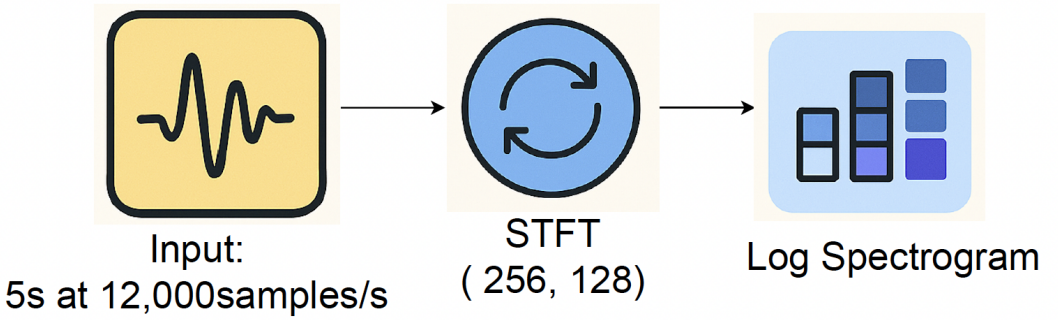


Figure 4.5: Feature extraction stage following data preparation.

4.3.1 Single-Channel Spectrograms (2D-CNN)

This approach preserves the native two-dimensional format of the spectrograms, enabling convolutional neural networks (2D-CNNs) to directly learn localized time–frequency features. Each augmented multi-sensor segment $\tilde{x}'_{ij}$ is transformed using the Short-Time

Fourier Transform (STFT) defined in Eq. 4.7, yielding a time–frequency representation and The resulting normalized spectrogram tensor,

$$X_{ij} \in \mathbb{R}^{128\times64\times1}, \tag{4.11}$$

where 128 corresponds to the number of discrete frequency bins, 64 to the number of time frames, and the single channel encodes the spectral magnitude.

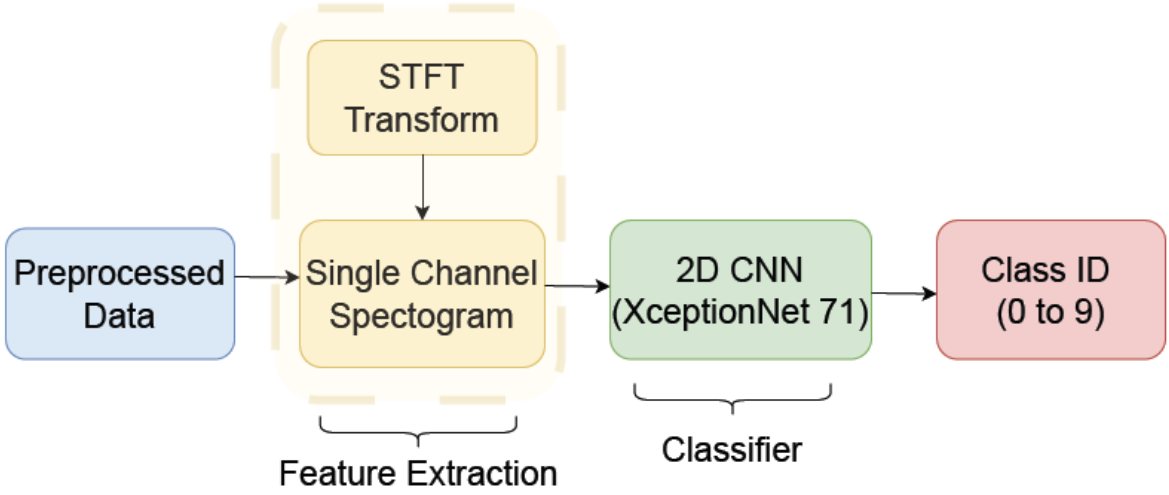


Figure 4.6: Single-Channel Spectrogram (2D-CNN).

Figure 4.6 illustrates the overall flow from STFT-based feature generation to classification via a 2D-CNN. By preserving spatial locality in the time–frequency domain, this setup enables the model to learn correlated patterns.

Algorithm 1 Inference Pipeline for Bearing Fault Classification

function LOAD_SIGNAL_AUTO($\mathcal{S}, f_{s,i}^{\text{orig}}$)
  for each signal $s_i \in \mathcal{S}$ **do**
    Identify file type corresponding to s_i
    if file type = '.mat' **then**
      Load MATLAB structure via `scipy.io.loadmat()`
      Retrieve original sampling frequency $f_{s,i}^{\text{orig}}$
      if $|f_{s,i}^{\text{orig}} - f_s^{\text{target}}| > \epsilon$ **then**
        Compute resampled discrete-time sequence:

$$s_{i,r}[n] = s_i\left(\frac{n}{f_s^{\text{target}}}\right), \quad n = 0, 1, 2, \ldots \qquad \text{(from Eq. 4.2)}$$

      else
        Set $s_{i,r}[n] = s_i[n]$
      end if
    end if
  end for
  Form the uniformly sampled dataset, where each $s_{i,r}$ is resampled to the target frequency f_s^{target}:

$$\mathcal{S}_r = \{s_{1,r}, s_{2,r}, \ldots, s_{M,r}\} \qquad \text{(Eq. 4.3)}$$

  return $\mathcal{S}_r$
end function

function SEGMENT_SIGNAL($\mathcal{S}_r, f_s^{\text{target}}$)
  for each resampled signal $s_{i,r}[n] \in \mathcal{S}_r$ **do**
    Define segment length and hop size:
    $N_{\text{seg}} = 5 f_s^{\text{target}}, \quad H_{\text{hop}} = 1 f_s^{\text{target}}$
    Divide $s_{i,r}[n]$ into overlapping windows according to Eq. 4.5:
    $x_{ij}[n] = s_{i,r}[n + jH_{\text{hop}}], \quad 0 \le n < N_{\text{seg}}$
    (from Eq. 4.5)
    Repeat for all valid j until the end of $s_{i,r}[n]$
  end for
  Each resulting $x_{ij}[n]$ corresponds to a 5-second segment extracted from the resampled signal.
end function

Algorithm 1 Inference Pipeline for Bearing Fault Classification (continued)

function FEATURE_EXTRACTION($\{x_{ij}\}$)
    for each segment x_{ij} obtained from segmentation **do**
        Compute STFT to obtain $X_{ij}(\ell, \omega)$ 4.7
        Convert to magnitude spectrogram 4.8
        Apply logarithmic compression 4.9
        Normalize using mean–variance normalization 4.10
        Resize or pad X_{ij} to the fixed dimension $X_{ij} \in \mathbb{R}^{128\times64\times1}$ 4.11
    end for
    Each X_{ij} serves as the time–frequency feature representation for classification
end function
function GET_XCEPTION71(K_{cls})
    Instantiate the model:
M = `timm.create_model("xception71", pretrained=False, num_classes` = K_{cls})
    Load pretrained weights from the saved checkpoint
    Set M to evaluation mode
    return M
end function
function CLASSIFICATION($M, \{X_{ij}\}$)
    for each input spectrogram X_{ij} **do**
        Obtain predicted output using the trained model:
$\hat{y} = M(X_{ij})$ 1.2
        Determine the predicted class index:
$\hat{k} = \arg\max(\hat{y})$
        Store $\hat{k}$
    end for
    Determine the final predicted class by majority voting across all segments:
$\hat{y}_{\text{final}} = \arg\max_k \text{count}(\hat{k} = k)$
    return $\hat{y}_{\text{final}}$
end function

Algorithm 1 Inference Pipeline for Bearing Fault Classification (continued)

function MAIN
    Load the pretrained model for $K_{\text{cls}} = 10$ bearing classes:
    $M \leftarrow \text{GET_XCEPTION71}(K_{\text{cls}})$
    Load and resample raw vibration signals from the dataset:
    $\mathcal{S}_r \leftarrow \text{LOAD_SIGNAL_AUTO}(\mathcal{S}, f_{s,i}^{\text{orig}})$
    Segment each resampled signal into fixed-length windows:
    $\{x_{ij}\} \leftarrow \text{SEGMENT_SIGNAL}(\mathcal{S}_r, f_s^{\text{target}})$
    Extract time–frequency features via STFT:
    $\{X_{ij}\} \leftarrow \text{FEATURE_EXTRACTION}(\{x_{ij}\})$
    Perform model inference and obtain predicted fault class:
    $\hat{y}_{\text{final}} \leftarrow \text{CLASSIFICATION}(M, \{X_{ij}\})$
    print $\hat{y}_{\text{final}}$ ▷ Predicted bearing fault label as per Table 3.2
end function

4.4 Classification Model Architecture

This section presents the neural network architecture employed for classifying bearing fault conditions using single channel input representations: single-channel 2D CNN STFT spectrograms. The input modality is paired with a deep learning architecture tailored to its dimensional structure and signal characteristics, ensuring efficient and discriminative feature extraction.

Recent advancements in deep learning for time–frequency analysis have introduced a spectrum of model architectures—ranging from convolutional networks to encoder–decoder structures and transfer learning–based attention frameworks. These architectures differ in their capacity to capture hierarchical abstractions, multiscale dependencies, and global context from spectrogram-like representations.

To systematically analyze the impact of architectural choices on classification performance, we categorize our models into three broad families: Encoder–Decoder architectures, Dilated Convolutional Networks, and Transfer Learning–based Attention models. This taxonomy is inspired by standard segmentation literature and is adapted here for the task of vibration-based fault diagnosis.

- **Transfer Learning and Attention-based Architectures.** Here pretrained convolutional backbones XceptionNet71 which were originally trained on large-scale im-

age corpora like ImageNet21k. These models are adapted to our fault classification task via transfer learning, achieving parameter efficiency using depthwise separable convolutions. This demonstrates strong generalization capabilities in low-data regimes and exploit the spatial structure of STFT-based inputs.

This architectural framework provides the foundation for the experimental design.

4.4.1 XceptionNet71 for Single-Channel STFT Features

Each vibration segment is transformed via the Short-Time Fourier Transform (STFT) into a grayscale spectrogram (Eq. 4.11). These spectrograms are resized to $224 \times 224 \times 1$ to fit the input requirements of two-dimensional convolutional backbones, enabling the network to extract spatially correlated features that are crucial for fault discrimination.

We employ the XceptionNet71 architecture from the `timm` library as the core feature extractor. Xception71 is a deep CNN that extends the Inception-style design by using only depthwise separable convolutions, thereby reducing the parameter count while improving accuracy. To leverage prior knowledge from large-scale datasets, transfer learning is applied by initializing the model with pretrained ImageNet weights. The model's original classification head is changed as per our number of classes, and the input channel is adapted to accept grayscale inputs by setting `in_chans = 1`. This configuration allows the model to study the generic low- and mid-level feature detectors (e.g., edges, contours, frequency ridges), accelerating convergence and improving generalization on limited industrial data.

The adapted Xception71 architecture consists of an entry flow, eleven middle-flow blocks, and an exit flow, each composed of depthwise separable convolutional layers and residual connections. The middle-flow module can be expressed as

$$\text{SeparableConv2D}(x) = \text{PointwiseConv}(\text{DepthwiseConv}(x)), \tag{4.12}$$

where the depthwise convolution performs spatial filtering per channel and the pointwise (1×1) convolution fuses cross-channel information. The exit flow concludes with global average pooling and a fully connected classification layer.

Given a normalized single-channel spectrogram $X_{ij} \in \mathbb{R}^{1\times 224\times 224}$, the feature extractor $\phi(\cdot)$ produces a high-level representation in the form of a feature tensor:

$$\phi(X_{ij}) \in \mathbb{R}^{C_f \times H' \times W'}, \tag{4.13}$$

where C_f denotes the number of learned feature channels, and H' and W' represent the reduced spatial dimensions after convolution and pooling operations.

A global average pooling layer aggregates spatial information as:

$$F = \frac{1}{H'W'} \sum_{i=1}^{H'} \sum_{j=1}^{W'} \phi_{:,i,j}, \quad F \in \mathbb{R}^{C_f}. \tag{4.14}$$

Here, F represents the pooled feature vector, which is passed through a fully connected layer to produce logits:

$$z = WF + b, \quad \hat{y}_k = \frac{e^{z_k}}{\sum_{j=1}^{K_{\text{cls}}} e^{z_j}}, \quad k \in \{1, \ldots, K_{\text{cls}}\}, \tag{4.15}$$

where $W \in \mathbb{R}^{K_{\text{cls}} \times C_f}$, z_k denotes the logit corresponding to class k, and $K_{\text{cls}} = 10$ is the total number of bearing condition classes. The softmax output $\hat{y}_k$ represents the predicted probability that the input spectrogram X_{ij} belongs to class k, i.e.,

$$\hat{y}_k = P(\text{class} = k \mid X_{ij}).$$

During training, these predicted probabilities are compared against the one-hot encoded ground-truth labels y_k using the categorical cross-entropy loss function introduced later in Section 5.1.

Xception71's depthwise-separable design is particularly effective for spectrogram-based fault diagnosis, as it enables efficient extraction of hierarchical spatial features that capture both harmonic and transient fault patterns. Transfer learning from ImageNet improves convergence and robustness on limited industrial datasets, providing a balance between interpretability, accuracy, and computational efficiency.

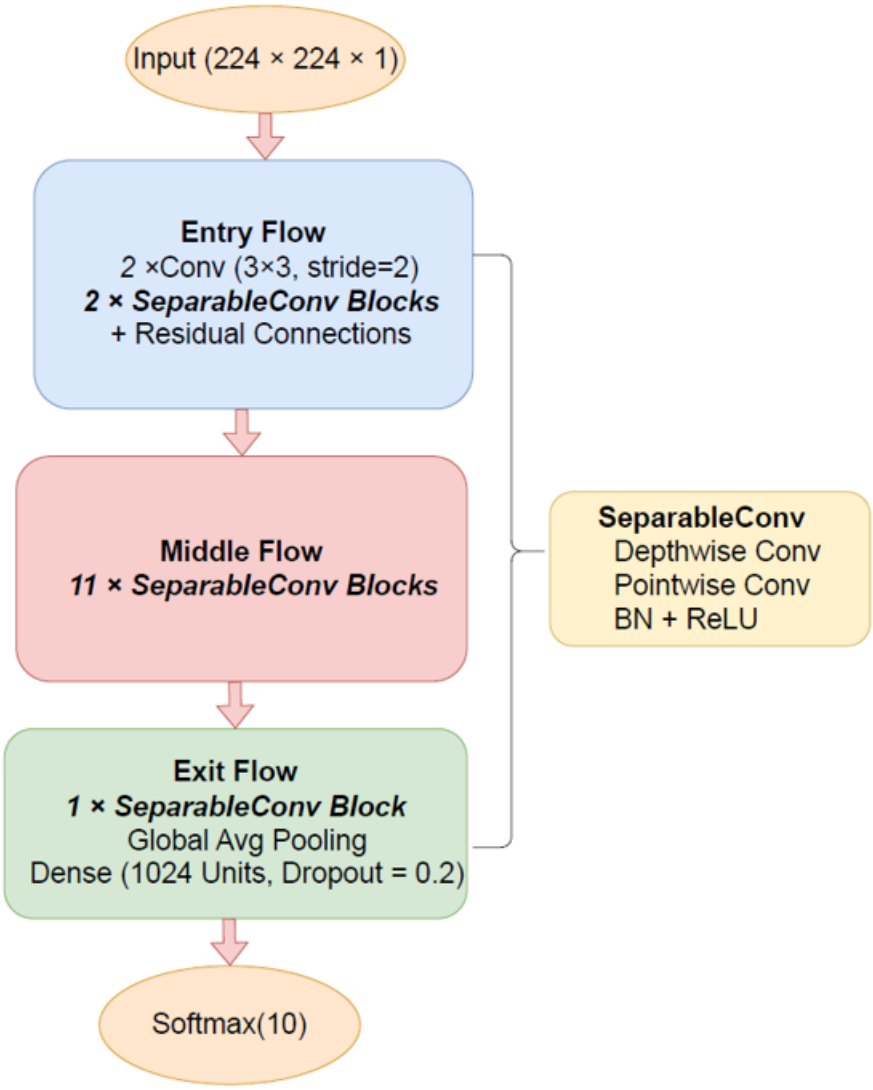


Figure 4.7: Xception71 architecture adapted for single-channel spectrogram input.

The XceptionNet-71 architecture employed in this work is derived from the original Xception model proposed by Chollet [40].

Chapter 5

Experiments and Results

This chapter outlines the experimental design and compares three deep learning model architectures developed for bearing fault classification. The objective is to evaluate how input representation and model complexity influence classification performance under a unified training pipeline.

The chapter is organized as follows: Section 5.1 describes the loss function used in all experiments, Section 5.2 outlines the evaluation strategy, and Section 5.3 explains the performance metrics adopted. Section 5.4 details the overall experimental setup. The subsequent sections present the three experiments conducted in this study.

Experiment 1 investigates the performance of the Dilated 1D CNN using flattened STFT signals as input. Section 5.5 introduces the input representation, and Section 5.5.1 outlines the experiment workflow. Sections 5.5.2, 5.5.3, and 5.5.4 discuss the per-fold macro F1-scores, ROC–AUC curves, and training dynamics of the Dilated 1D CNN.

Experiment 2 evaluates the pretrained XceptionNet-71 model using single-channel STFT spectrograms. Section 5.6 describes the grayscale spectrogram preprocessing, followed by the performance results and evaluation metrics in Section 5.6.2 and Table 5.9. Section 5.6.3 presents the ROC–AUC analysis, while Section 5.6.4 examines the training and validation behavior of the model.

Experiment 3 analyzes the EfficientNetV2-M model trained on three-channel Delta-STFT spectrograms. Section 5.7 explains the construction of multi-resolution, three-channel inputs and preprocessing workflow. Section 5.7.1 presents the per-fold validation metrics, while Section 5.7.2 summarizes the test performance. The ROC–AUC and training dynamics are discussed in Sections 5.7.3 and 5.7.4, respectively.

This section presents a comprehensive comparative analysis of the three deep learning models—Dilated 1D CNN, XceptionNet-71, and EfficientNetV2-M—developed for vibration-based bearing fault classification using the CWRU dataset contributing to $\mathbf{C_3}$.The structure of this section is as follows: Section 5.8.1 summarizes the overall model evaluation and cross-validation performance, consolidating the key findings across all architectures. Section 5.8.2 compares the confusion matrices to analyze class-wise fault separability and identify common misclassification patterns. Section 5.8.3 presents the ROC–AUC analysis, highlighting the discriminative capability of each model across the ten fault categories. Section 5.8.4 examines the training and validation dynamics to assess optimization stability and convergence behavior. Finally, Section 5.8.5 synthesizes the main insights, discussing comparative performance, model reliability, and implications for real-world deployment in predictive maintenance applications.

5.1 Loss Function for Experiments

The time–frequency feature representation X_{ij} through preprocessing and feature extraction using the STFT defined in Eq. 4.7. Subsequent logarithmic compression and normalization steps (Eqs. 4.9–4.10) refine these representations to enhance contrast and ensure compatibility with deep learning models.

The resulting labeled dataset is expressed as

$$\mathcal{D} = \{(X_{ij}, y_{ij})\} \iff \{(x^{(n)}, y^{(n)})\}_{n=1}^{N_{\text{tot}}}, \tag{5.1}$$

where N_{tot} is the total number of samples and $y^{(n)}$ denotes the corresponding ground-truth class label.

so, for a supervised learning problem, we consider

$$\mathcal{D} = \{(x^{(n)}, y^{(n)})\}_{n=1}^{N_{\text{tot}}}, \tag{5.2}$$

where N_{tot} is the total number of samples and $y^{(n)}$ denotes the ground-truth label

Here, $y^{(n)}$ represents the **ground-truth** class label of the n-th training example, while $\hat{y}^{(n)}$ (introduced in the learning objective) denotes the **predicted** class label produced by the model $f_{\boldsymbol{\psi}}(x^{(n)})$.

Learning Objective. The goal of supervised learning is to approximate a mapping

$$f_{\boldsymbol{\psi}} : x^{(n)} \mapsto \hat{y}^{(n)},$$

parameterized by the model weights and biases $\boldsymbol{\psi}$, that predicts the class label $\hat{y}^{(n)}$ given the input segment $x^{(n)}$. Here, $y^{(n)}$ denotes the true (ground-truth) class label, and $\hat{y}^{(n)}$ represents the predicted output of the model.

Training is guided by the *categorical cross-entropy loss*, which quantifies the discrepancy between the predicted probabilities and the ground-truth labels:

$$\mathcal{L}_{\mathrm{CE}}(\boldsymbol{\psi}) = -\frac{1}{N_{\mathrm{tot}}} \sum_{n=1}^{N_{\mathrm{tot}}} \sum_{k=1}^{K_{\mathrm{cls}}} \mathbf{1}\{y^{(n)} = k\} \log\left(\hat{y}_k^{(n)}\right), \tag{5.3}$$

- $\mathbf{1}\{\cdot\}$ denotes the indicator function, which outputs 1 when the condition inside the braces is true and 0 otherwise;
- $\mathbf{1}\{y^{(n)} = k\}$: equals 1 if sample n belongs to class k, and 0 otherwise;
- $\hat{y}_k^{(n)} = f_{\boldsymbol{\psi}}(x^{(n)})_k$ denotes the predicted probability assigned to class k;
- $y^{(n)}$ and $\hat{y}^{(n)}$ represent the ground-truth and predicted class labels, respectively, with $\hat{y}_k^{(n)} = \arg\max_k \hat{y}_k^{(n)}$.

5.2 Evaluation Strategy for Experiments

For conducting these experiments a stratified 80/20 split was subsequently applied to ensure that the relative proportions of all fault classes were preserved in both subsets, resulting in 9,984 training and 2,496 testing samples. To mitigate this effect, all models were trained and evaluated using **10-fold stratified cross-validation**, which preserves the original class distribution in each fold Table 5.1 summarizes the class-wise distribution of fault types along with the number of training and test samples in each class after the split. This approach reduces the risk of imbalance-induced bias while providing a more reliable estimate of model performance. For each fold, performance metrics were computed and averaged across folds.

These architectures were chosen to explore different inductive biases: 1D convolutions for raw temporal patterns, and 2D convolutions for time-frequency images.

All the experiments were trained using a unified pipeline implemented in PyTorch and executed on an NVIDIA GeForce RTX 3060 Ti GPU. The core training setup was consistent across all experiments.

Table 5.1: Class-wise distribution of samples with train-test split (80:20)

Class ID	Train Samples	Test Samples
0	874	218
1	874	218
2	874	218
3	873	219
4	835	209
5	874	218
6	2256	564
7	432	108
8	1910	478
9	182	46
Total	**9984**	**2296**

5.3 Evaluation Metrics

Within this framework, the macro-averaged F1-score was adopted as the primary evaluation metric. Unlike overall accuracy which is inflated by correct predictions on majority classes—macro F1 assigns equal weight to all classes by averaging their individual F1 size [41]. This makes it particularly suitable for imbalanced multi-class classification tasks such as bearing fault diagnosis. Additionally, reporting class-wise F1-scores provides deeper insight into which fault categories are most challenging to predict and whether performance is skewed toward specific subsets of classes contributing to $\mathbf{C_1}$.

Beyond F1-based measures, a complementary suite of evaluation metrics was employed to provide a more holistic understanding of model behavior under imbalance. These include overall accuracy, precision, recall, specificity, and the area under the ROC curve (AUROC). Together, they capture not only general correctness but also the model's ability to avoid false alarms, detect rare faults, and maintain balanced performance across all fault categories [41].

Confusion Matrix Formulation. All performance metrics derive from the confusion ma-

trix, which summarizes the model's prediction outcomes for each class $k \in \{1, 2, \ldots, K_{\text{cls}}\}$:

$$TP_k = \sum_{i=1}^{N_{\text{test}}} \mathbf{1}\{y_i = k \wedge \hat{y}_i = k\}, \quad \text{True Positives (correct predictions of class } k), \tag{5.4}$$

$$FP_k = \sum_{i=1}^{N_{\text{test}}} \mathbf{1}\{y_i \neq k \wedge \hat{y}_i = k\}, \quad \text{False Positives (incorrectly predicted as class } k), \tag{5.5}$$

$$FN_k = \sum_{i=1}^{N_{\text{test}}} \mathbf{1}\{y_i = k \wedge \hat{y}_i \neq k\}, \quad \text{False Negatives (missed predictions of class } k), \tag{5.6}$$

$$TN_k = \sum_{i=1}^{N_{\text{test}}} \mathbf{1}\{y_i \neq k \wedge \hat{y}_i \neq k\}, \quad \text{True Negatives (all other classes correctly identified).} \tag{5.7}$$

Here, y_i and $\hat{y}_i$ denote the ground-truth and predicted labels for the i-th test sample, N_{test} is the total number of test samples, and $\mathbf{1}\{\cdot\}$ is the indicator function.

Evaluation Metrics. The primary metrics derived from the confusion matrix are summarized below:

- **Accuracy:** This metric quantifies the proportion of total predictions that the model correctly identifies, encompassing both the positive and negative classes. It is a straightforward and commonly used performance indicator in classification problems. However, in datasets with significant class imbalance, accuracy can be misleading, as a model may achieve high accuracy by merely predicting the dominant class correctly while performing poorly on minority classes.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.8}$$

- **Precision:** Precision evaluates the correctness of positive predictions by computing the ratio of true positives to all instances predicted as positive. It reflects how well the model avoids false positives. In imbalanced datasets, precision becomes especially relevant when the cost of false alarms is high, as it emphasizes the reliability of the model's positive predictions.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{5.9}$$

- **Recall (Sensitivity):** Recall quantifies the model's ability to correctly identify all instances of the positive class. It is defined as the ratio of true positives to the total number of actual positive instances. A high recall indicates that the model is effective at minimizing false negatives, making it particularly important in fault detection scenarios where missing a faulty instance could have severe consequences.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{5.10}$$

- **F1-Score / Dice Similarity Coefficient (DSC):** The F1-score is the harmonic mean of precision and recall, offering a balanced measure that considers both false positives and false negatives. It is particularly useful in imbalanced classification problems where a high precision or recall alone may be misleading. The Dice Similarity Coefficient (DSC) is mathematically equivalent to the F1-score and is commonly used in domains like medical imaging. In this study, macro-averaged F1-score computed by averaging F1-scores across all classes without weighting—served as the principal evaluation criterion due to the class imbalance in the dataset.

$$\text{F1-Score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5.11}$$

- **AUROC (Area Under the Receiver Operating Characteristic Curve):** AUROC evaluates the model's discriminative capability by plotting the True Positive Rate (TPR) against the False Positive Rate (FPR) at varying classification thresholds. It reflects how well the model ranks positive instances higher than negative ones. Although AUROC is commonly used for binary classification, in multi-class scenarios, it can be extended using a one-vs-rest (OvR) strategy, where the AUROC is computed independently for each class against all others and then averaged. A higher AUROC indicates stronger separability between classes.

$$\text{AUROC} = \int_0^1 \text{TPR}(FPR)\, d(\text{FPR}) \tag{5.12}$$

- **AUPRC (Area Under the Precision-Recall Curve):** AUPRC quantifies the area under the curve formed by plotting Precision against Recall at different classification thresholds. It is particularly informative in class-imbalanced settings, where AUROC may present an overly optimistic view. AUPRC emphasizes the model's ability to correctly identify positive instances while penalizing false positives more

explicitly. Like AUROC, AUPRC is extended to multi-class classification via a one-vs-rest scheme and averaged across all classes.

$$\text{AUPRC} = \int_0^1 \text{Precision}(\text{Recall})\, d(\text{Recall}) \tag{5.13}$$

- **Macro-F1:** The Macro-F1 score aggregates performance across classes by computing the unweighted average of per-class F1-scores. This metric treats all classes equally, making it particularly suitable for imbalanced datasets where minority classes must not be overshadowed by majority ones.

$$\text{Macro-F1} = \frac{1}{K} \sum_{k=1}^{K} \text{F1}_k \tag{5.14}$$

- **Micro-F1:** The Micro-F1 score is computed globally by aggregating true positives, false positives, and false negatives across all classes before calculating the F1-score. Unlike Macro-F1, it weights each class proportionally to its number of samples, and thus reflects overall classification performance more strongly influenced by majority classes.

$$\text{Micro-F1} = \frac{2 \cdot \text{TP}_{\text{micro}}}{2 \cdot \text{TP}_{\text{micro}} + \text{FP}_{\text{micro}} + \text{FN}_{\text{micro}}}, \tag{5.15}$$

where

$$\text{TP}_{\text{micro}} = \sum_{k=1}^{K} TP_k, \quad \text{FP}_{\text{micro}} = \sum_{k=1}^{K} FP_k, \quad \text{FN}_{\text{micro}} = \sum_{k=1}^{K} FN_k.$$

5.4 Experimental Setup

The experimental pipeline (Fig. 5.1) consists of three main stages: (a) *Preprocessing* detailed in so and so section , (b) *Feature Extraction* as detailed in so and so section, and (c) *Classification*.

This table 5.2 summarizes the core architectural choices and training hyperparameters used across the three experiments.

Table 5.2: Hyperparameter Settings for Each Experiment

Parameter	**Exp 1: Dilated 1D CNN**	**Exp 2: Xception71**	**Exp 3: EfficientNetV2-M**
Input Type	Raw 1D Signal	STFT Spectrogram	Delta-STFT Spectrogram
Input Shape	8192 samples	224 × 224	224 × 224
Input Channels	1	1 (Grayscale)	3 (RGB)
Architecture	Custom Dilated 1D CNN	`xception71` (pretrained)	`tf_efficientnetv2_m` (pretrained)
Loss Function	CrossEntropyLoss	CrossEntropyLoss	CrossEntropyLoss
Optimizer	Adam	Adam	Adam
Learning Rate	1e-4	1e-4	1e-4
Weight Decay	1e-5	1e-5	1e-5
Batch Size	16 (AMP Enabled)	64 (Grad Accum)	64 (Grad Accum)
Gradient Accumulation	No	Yes (8 steps)	Yes (8 steps)
LR Scheduler	ReduceLROnPlateau (factor 0.5)	Same	Same
Epochs per Fold	10	10	10
Early Stopping	Yes (patience = 10)	Yes	Yes
Cross-Validation	10-fold Stratified K-Fold	Same	Same
Evaluation Metric	Macro F1-Score	Macro F1-Score	Macro F1-Score
Device	NVIDIA RTX 3060 Ti	Same	Same

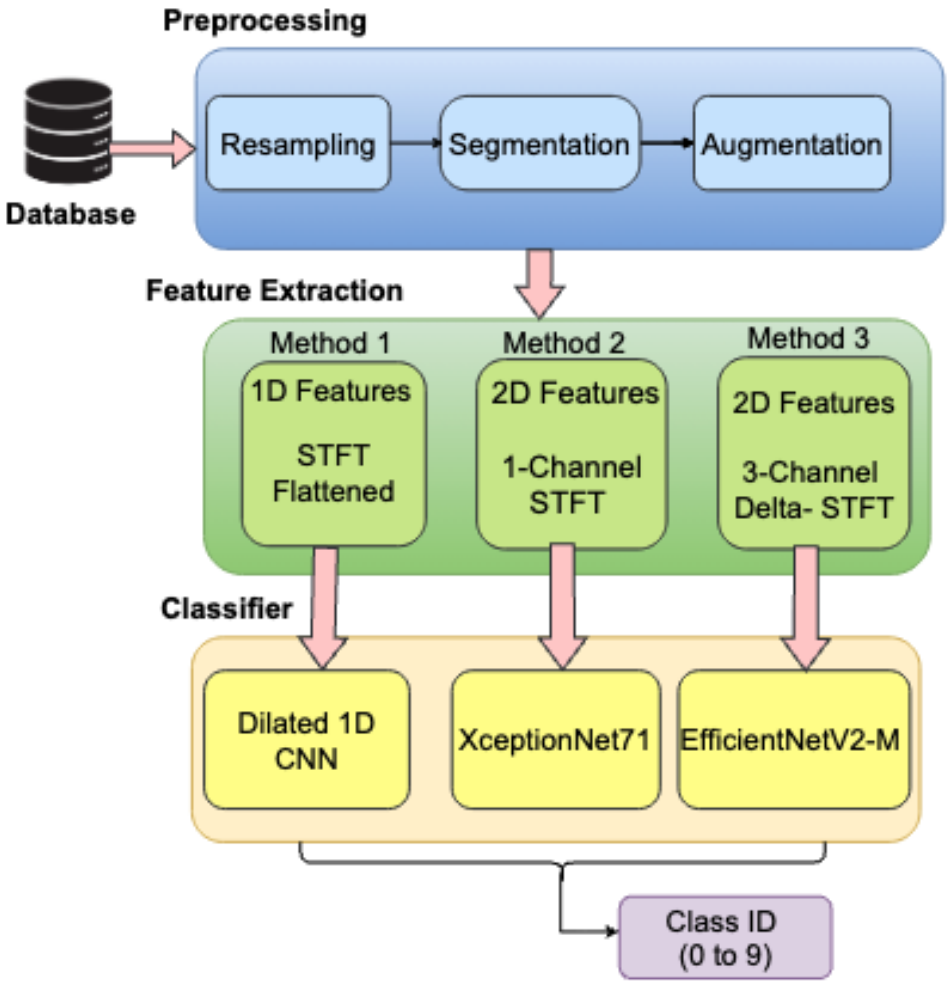


Figure 5.1: Workflow of the Experiments

5.5 Flattened STFT Features

As a foundational step in the modeling process, **Experiment 1** explores a simplified yet effective approach for capturing time–frequency patterns using flattened representations

of Short-Time Fourier Transform (STFT) spectrograms. The motivation behind this experiment is twofold: (i) to establish a computationally lightweight and interpretable baseline using one-dimensional convolutional neural networks (1D-CNNs), and (ii) to examine whether discriminative fault-related features can be effectively extracted without explicitly preserving the two-dimensional spatial structure of spectrograms.

After applying the STFT to each segmented vibration signal, the resulting spectrogram representation for the j-th segment of the i-th signal is denoted as

$$X_{ij} \in \mathbb{R}^{128\times 64},$$

where each row corresponds to a discrete frequency bin and each column represents a localized time frame. This time–frequency distribution captures transient fault impulses and harmonic modulations that are often obscured in raw time-domain signals.

To ensure compatibility with 1D-CNN architectures, each spectrogram X_{ij} is reshaped into a flattened one-dimensional vector:

$$X_{\text{flat}}^{(n)} \in \mathbb{R}^{8192},$$

where all 128×64 elements are concatenated.

The simplicity of the flattened representation offers computational advantages during both training and inference. But, in this study the spatial context is not preserved.

In this study, flattened STFT vectors form the basis for Experiment 1, serving as a baseline for the further two experiments two-dimensional representations single-channel and three-channel spectrograms. A visual overview of this pipeline is presented in Figure 5.2.

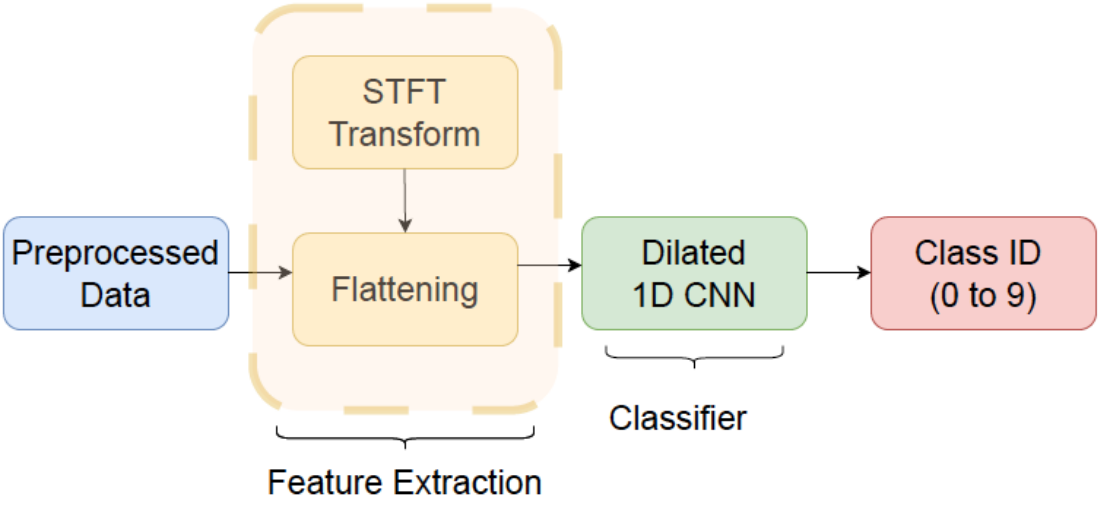


Figure 5.2: Experiment 1 pipeline: Flattened STFT Vectors with 1D-CNN.

5.5.1 Per-Fold Validation Macro F1-Score – Dilated 1D CNN

Table 5.3 summarises the macro F1-score for the dilated 1d CNN and the figure 5.3 the representative figure of the performance for the dilated 1d CNN model.

Table 5.3: Per-Fold Validation Macro F1-Score – Dilated 1D CNN

Fold	**Val Macro F1**
Fold 1	0.9891
Fold 2	0.9924
Fold 3	0.9903
Fold 4	0.9963
Fold 5	0.9931
Fold 6	0.9850
Fold 7	0.9909
Fold 8	0.9949
Fold 9	0.9805
Fold 10	0.9695
Mean ± Std	**0.9870 ± 0.0083**

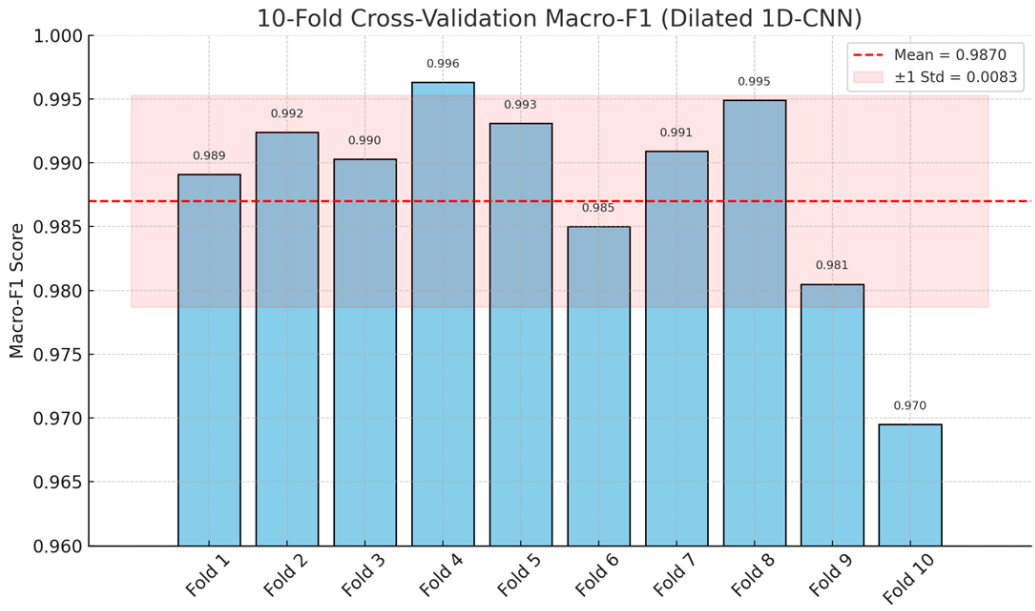


Figure 5.3: Macro-F1 Performance Across Folds for Dilated 1D CNN

Evaluation Metric for Dilated 1D CNN The classification performance was evaluated using the **Macro-F1 score**, which accounts for class imbalance across the 10 fault categories. The results are summarized as follows:

- **Macro-F1 (Mean ± Std):** 0.9870 ± 0.0083, indicating high accuracy with low variance across folds.
- **Consistency:** No fold exhibited a significant drop, confirming model stability and generalization.
- **Validation Strategy:** A 10-fold stratified cross-validation protocol was employed to ensure robustness of the reported results.

5.5.2 Confusion Matrix Analysis of Dilated 1D-CNN

To better understand class-wise performance, we present the confusion matrices for the three best-performing models. These matrices provide insights into which fault types were correctly identified and which were occasionally misclassified. All matrices are based on test set predictions or the best validation fold, depending on the model.

Figure 5.4 shows the confusion matrix for the Dilated 1D CNN.

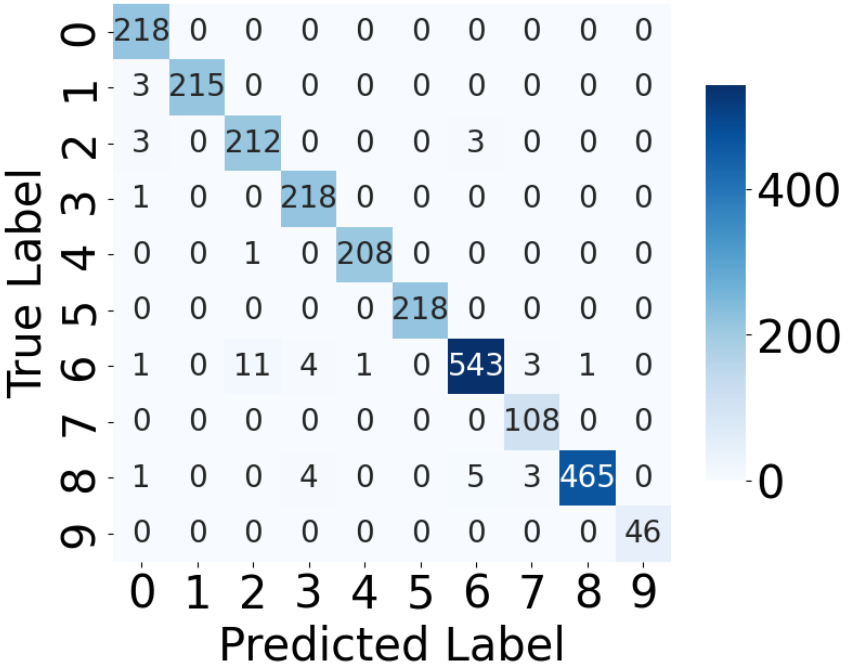


Figure 5.4: Confusion Matrix Dilated 1D CNN

The performance of the Dilated 1D-CNN model is evaluated through per-class metrics and the confusion matrix. Together, these provide complementary insights into class-wise reliability, error structure, and operational significance.

Table 5.5 reports precision, recall, F1-score, specificity, and confusion counts for each class. Classes 0 through 3 and Class 9 (Normal) achieve exceptionally high F1-scores (> 0.98), reflecting strong separability between early-stage faults and healthy conditions. Class 9 is classified perfectly with zero errors, confirming the robustness of the model on the healthy baseline. Class 4 achieves perfect precision but slightly lower recall (0.9761) due to a few false negatives. Class 6, despite having the largest sample count, shows a modest reduction in recall (0.9823), likely because of spectral feature overlap with neighboring fault classes. Overall, specificity exceeds 0.995 across all classes, indicating very few false positives.

Table 5.4: Class-wise data distribution and confusion components for Dilated 1D-CNN

Class ID	**Train**	**Test**	$\mathbf{TP}_k$	$\mathbf{FP}_k$	$\mathbf{FN}_k$	$\mathbf{TN}_k$
0	874	218	218	9	0	2069
1	874	218	215	0	3	2078
2	874	218	212	12	6	2066
3	873	219	218	8	1	2069
4	835	209	208	1	1	2086
5	874	218	218	0	0	2078
6	2256	564	543	8	21	1724
7	432	108	108	6	0	2282
8	1910	478	465	1	13	1817
9	182	46	46	0	0	2250
Total	9984	2296	2443	45	45	20,519

Table 5.5: Per-class evaluation metrics for Dilated 1D-CNN (Test Set)

Class ID	**Precision**$_k$	**Recall**$_k$	**F1**$_k$	**Specificity**$_k$
0	0.9603	1.0000	0.9797	0.9957
1	1.0000	0.9862	0.9930	1.0000
2	0.9464	0.9725	0.9593	0.9942
3	0.9646	0.9954	0.9798	0.9961
4	0.9952	0.9952	0.9952	0.9995
5	1.0000	1.0000	1.0000	1.0000
6	0.9628	0.9628	0.9628	0.9879
7	0.9474	1.0000	0.9729	0.9973
8	0.9728	0.9728	0.9728	0.9929
9	1.0000	1.0000	1.0000	1.0000
Macro-F1	0.9816			
Micro-F1	0.9819			

The confusion matrix (Fig. 5.4) further highlights the error patterns. Five key conclusions can be drawn:

1. **Fault-Type Preservation:** Ball, Inner Race, and Outer Race faults are almost never confused with one another. This is crucial for real-world reliability, as cross-type errors could misdirect maintenance.

2. **Severity-Level Misclassifications (Safe):** Most errors occur within the same fault family, such as Ball $0.021''$ $\rightarrow$ Ball $0.014''$. These represent severity downgrades or upgrades, but remain operationally safe since the correct subsystem is still flagged.

3. **Minimal Cross-Type Errors:** A few rare tuples, such as Ball $0.021''$ $\rightarrow$ Inner $0.007''$, were observed. While undesirable, these occur infrequently and do not undermine overall robustness.

4. **Normal vs. Fault Separation:** The Normal class (Class 9) is consistently distinguished from fault classes. False alarms (Normal $\rightarrow$ Fault) and missed detections (Fault $\rightarrow$ Normal) are negligible, reducing both unnecessary downtime and risk of missed faults.

5. **Operational Reliability by Fault Type:** Ball faults (easier to replace) are consistently identified, Inner Race faults are correctly captured with only minor severity confusion, and Outer Race faults—the most critical case—are reliably classified as outer race. This ensures that high-risk conditions are not overlooked.

Representative tuples of the form (True Class → Predicted Class) are grouped in Table 5.6 to make the misclassification structure explicit.

Table 5.6: Tuple-based representation of misclassifications in the Dilated 1D-CNN experiment.

Group	**Representative Tuples**
Safe (within-type severity confusion)	$(0 \to 1), (1 \to 2), (2 \to 1), (3 \to 4), (4 \to 5), (6 \to 7), (7 \to 6), (8 \to 6)$
Unsafe (cross-type, rare)	$(2 \to 3)$
Normal-related	Negligible; very few cases of $(9 \to X)$ or $(X \to 9)$

In summary, the Dilated 1D-CNN achieves high accuracy across all classes, with errors largely confined to safe severity-level confusions. The absence of significant cross-type or Normal-related errors underscores the model's suitability for real-world predictive maintenance deployment.

5.5.3 ROC-AUC Curve Analysis of Dilated 1D CNN

To further evaluate the discriminative capability of the Dilated 1D CNN model across all fault categories, we computed the one-vs-rest Receiver Operating Characteristic (ROC) curves for each class. Figure 5.5 presents the resulting multi-class ROC plot.

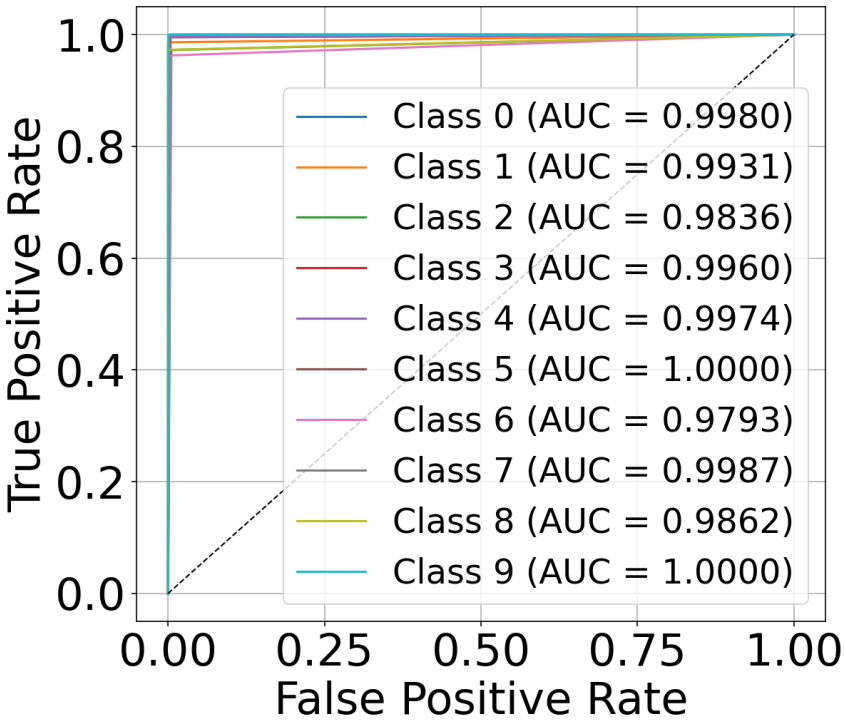


Figure 5.5: Multi-Class ROC Curve for Dilated 1D CNN

The ROC–AUC curves for the Dilated 1D-CNN (Fig. 5.5) demonstrate the model's strong discriminative ability across all fault categories. All classes achieve AUC values above 0.97, with several classes approaching or reaching perfect separation. This analysis provides the following qualitative insights:

- **High separability across all faults:** Since all AUC values are close to 1.0, the model demonstrates excellent capacity to distinguish between faulty and healthy states. In practice, this means operators can rely on the model to consistently identify fault signatures with minimal ambiguity.
- **Normal state is highly reliable:** The Normal class (Class 9) achieves a perfect AUC of 1.0000, confirming that healthy bearings are never misclassified as faulty. This is crucial in predictive maintenance, as it minimizes false alarms and avoids unnecessary downtime or maintenance costs.
- **Safety-critical faults are well captured:** Outer Race faults, which are the most critical because they can propagate into catastrophic machine failure, achieve very

high AUC values. This ensures strong protection against overlooking dangerous conditions.

- **Minor weaknesses in severity gradation:** The few classes with slightly lower AUCs (e.g., Ball Fault 0.021″, Outer Race Fault 0.007″) reflect difficulty in distinguishing between fault severities of the same type. From an operational perspective, this is acceptable: identifying that an outer race fault exists is more important than precisely estimating the spall size. These errors therefore represent safe degradations rather than unsafe misdiagnoses.
- **Consistency across the board:** The tight clustering of AUC values (0.98–1.0) shows that no single class is a weak link. This consistency is essential for deployment in real-world systems, where overlooking even a rare fault can carry high costs.

In summary, the ROC–AUC results confirm that the Dilated 1D-CNN provides highly reliable, safe, and consistent classification across all fault categories, with especially strong performance on the Normal state and safety-critical Outer Race faults.

5.5.4 Training Dynamics of Dilated 1D CNN

Figures 5.7 and 5.6 illustrate the training and validation behavior of the Dilated 1D-CNN model over 10 epochs.

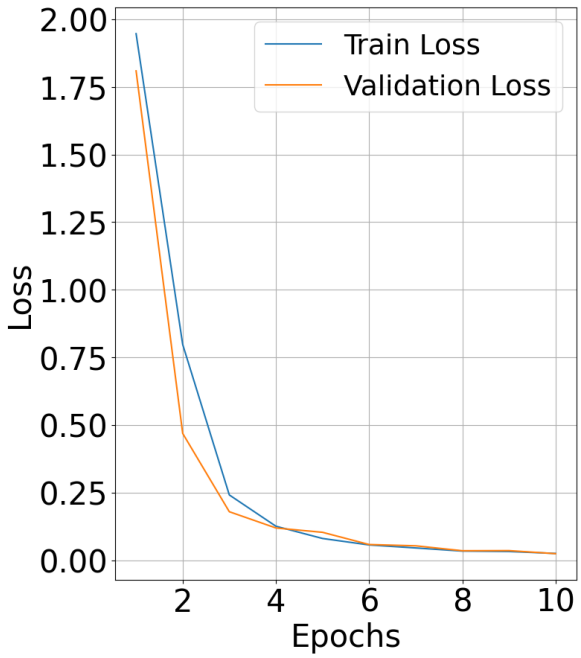


Figure 5.6: Dilated 1D CNN loss vs epochs

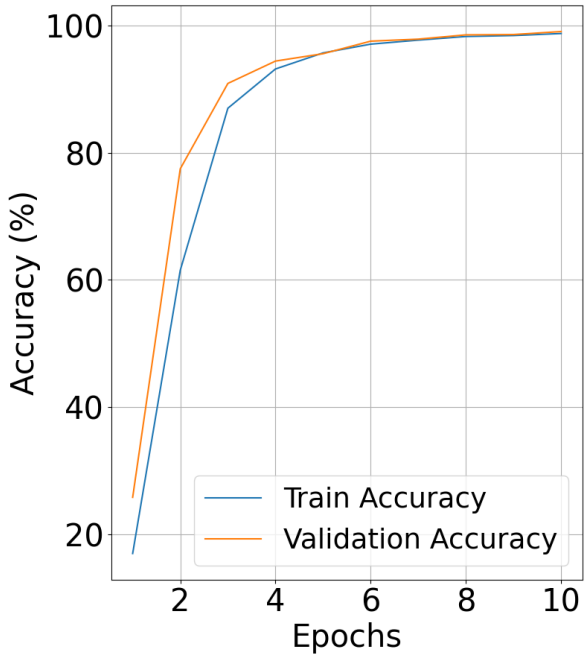


Figure 5.7: Dilated 1D CNN accuracy vs epochs

Several important observations can be made which are states as follows:

- **Rapid convergence:** Both accuracy and loss curves demonstrate steep improvements within the first 3 epochs. Training accuracy rises sharply from below 20% to above 80%, while loss drops from nearly 2.0 to below 0.2. This indicates that the model quickly learns discriminative patterns from the vibration data.
- **Stability after initial epochs:** Beyond epoch 4, both training and validation curves begin to flatten, showing incremental improvements until convergence. This suggests that the model achieves stable optimization without oscillations or divergence.
- **High final accuracy:** By epoch 10, training accuracy approaches 100%, while validation accuracy stabilizes around 97–98%. This confirms strong generalization performance, consistent with the high F1-scores reported in the confusion matrix analysis.

- **Minimal overfitting:** The gap between training and validation curves is small throughout training. Validation loss closely follows training loss, and validation accuracy tracks training accuracy, indicating that the model does not overfit despite its high capacity.
- **Smooth generalization behavior:** The gradual alignment of validation and training curves across both metrics reflects effective learning dynamics, with the model generalizing well across unseen samples.

In summary, the training and validation curves demonstrate that the Dilated 1D-CNN converges rapidly, achieves high accuracy, and maintains consistent generalization without overfitting. These findings reinforce the reliability of the model's performance on fault classification tasks.

5.6 Single-Channel Spectrograms

This experiment has been briefly described in so and so Section 4.3.1. The Section 4.4 presents the classification model architecture based on the proposed single-channel spectrogram. Together, these sections provide a comprehensive overview of the methodology employed in this study.

5.6.1 Per-Fold Validation Macro F1-Score of Single-Channel Spectrograms

Table 5.7 summarises the macro F1-score for the XceptionNet71 and the figure 5.8 the representative figure of the performance for the XceptionNet71.

Table 5.7: Per-Fold Validation Macro F1-Score – XceptionNet71

Fold	Val Macro F1
Fold 1	0.9869
Fold 2	0.9924
Fold 3	0.9881
Fold 4	0.9698
Fold 5	0.9881
Fold 6	0.9897
Fold 7	0.9904
Fold 8	0.9982
Fold 9	0.9854
Fold 10	0.9777
Mean ± Std	**0.9852 ± 0.0082**

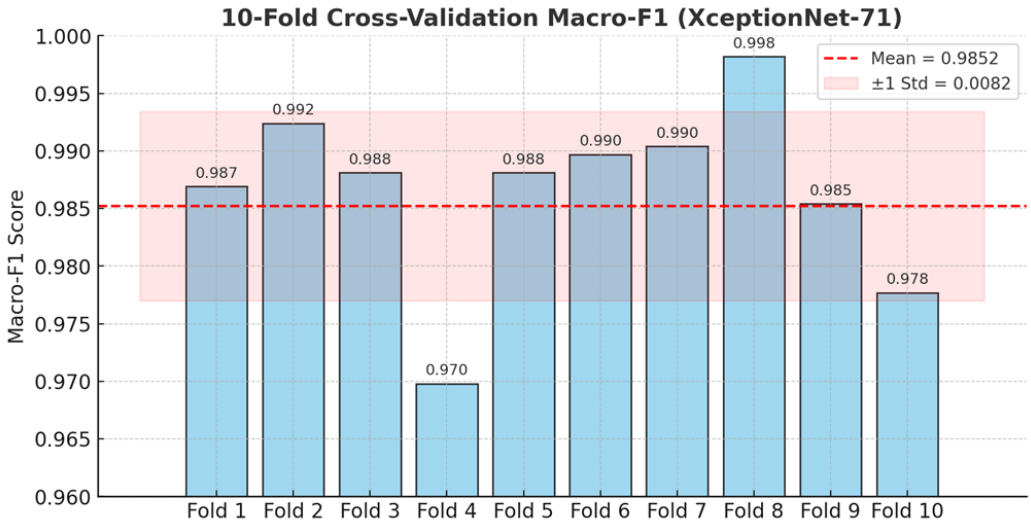


Figure 5.8: Macro-F1 Performance Across Folds for the XceptionNet71

Evaluation Metric. The classification performance was assessed using the **Macro-F1 score**, selected to address class imbalance across the 10 fault categories. The key observations are:

- **Metric Used:** Macro-F1 score, chosen to account for imbalance across all 10 classes.

- **Mean ± Std:** 0.9870 ± 0.0083, showing high accuracy with low variance and confirming stability.
- **Consistency Across Folds:** No drastic performance drop; even the weakest fold (Fold 10, 0.9695) remains above 0.96.
- **Strong Generalization:** Model adapts well across data splits, indicating results are not biased by any particular fold.
- **Robust Validation:** A 10-fold stratified cross-validation protocol ensures fairness across all classes, making the evaluation more reliable.

5.6.2 Confusion Matrix Analysis of XceptionNet71

Figure 5.9 shows the confusion matrix for the XceptionNet-71 model

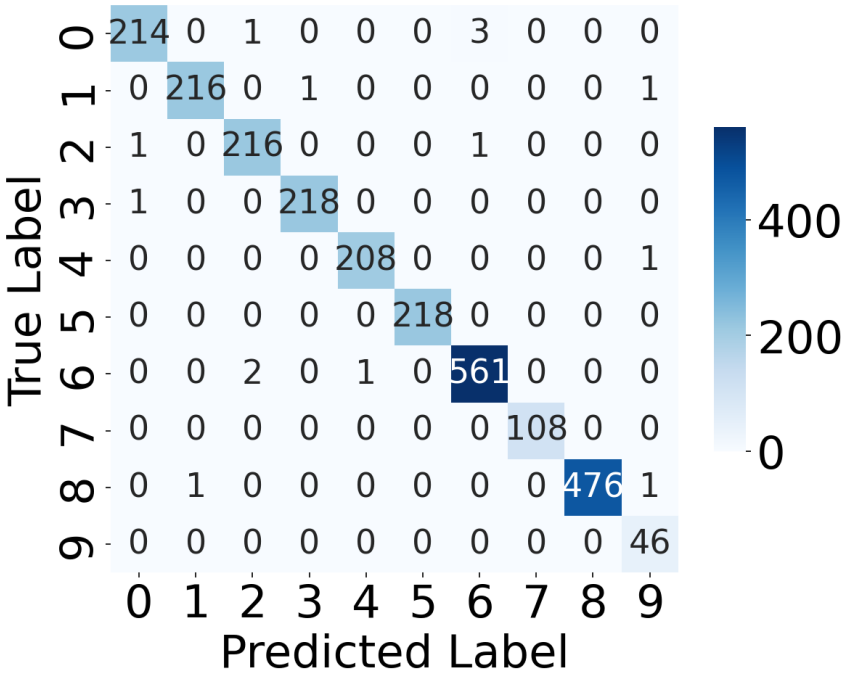


Figure 5.9: Confusion Matrix for XceptionNet71

The performance of the XceptionNet-71 model is evaluated through per-class metrics

and the confusion matrix. Together, these provide complementary insights into class-wise reliability, error structure, and operational significance.

Table 5.9 reports precision, recall, F1-score, specificity, and confusion counts for each class. Nearly all classes achieve exceptionally high F1-scores (> 0.997), reflecting near-perfect separability between different fault severities and healthy conditions. Class 9 (Normal) is classified perfectly with zero errors, confirming the robustness of the model on the healthy baseline. Class 6 (Outer Race Fault, $0.007''$) exhibits a minor recall drop (0.9965) due to two false negatives, while Class 8 (Outer Race Fault, $0.021''$) shows a single misclassification into Class 5. All other classes, including Class 4 (Inner Race Fault, $0.014''$) and Class 7 (Outer Race Fault, $0.014''$), are classified with perfect accuracy. Overall, specificity remains above 0.999 for all classes, indicating extremely low false positive rates.

Table 5.8: Per-class confusion components for XceptionNet71 (Test Set)

Class ID	**TP**$_k$	**FP**$_k$	**FN**$_k$	**TN**$_k$
0	214	2	4	2076
1	216	1	2	2077
2	216	3	2	2075
3	218	1	1	2076
4	208	1	1	2086
5	218	0	0	2078
6	561	4	3	1728
7	108	0	0	2188
8	476	0	2	1818
9	46	3	0	2247
Total	2481	15	15	20,449

Table 5.9: Per-class evaluation metrics for Xception-71 (Test Set)

Class ID	**Precision**$_k$	**Recall**$_k$	**F1**$_k$	**Specificity**$_k$
0	0.9908	0.9816	0.9862	0.9990
1	0.9954	0.9908	0.9931	0.9995
2	0.9863	0.9908	0.9885	0.9986
3	0.9954	0.9954	0.9954	0.9995
4	0.9952	0.9952	0.9952	0.9995
5	1.0000	1.0000	1.0000	1.0000
6	0.9947	0.9947	0.9947	0.9983
7	1.0000	1.0000	1.0000	1.0000
8	0.9958	0.9958	0.9958	0.9989
9	0.9388	1.0000	0.9684	0.9987
Macro-F1	0.9917			
Micro-F1	0.9940			

The confusion matrix (Fig. 5.9) further highlights the error patterns. Five key conclusions can be drawn:

1. **Fault-Type Preservation:** Ball, Inner Race, and Outer Race faults are consistently distinguished, with no significant cross-type errors. This ensures reliability for subsystem-level diagnosis.

2. **Severity-Level Misclassifications (Safe):** The few errors are confined to severity overlaps within the same fault family, such as Outer $0.007''$ $\rightarrow$ Outer $0.021''$ and Outer $0.021''$ $\rightarrow$ Inner $0.021''$. These remain operationally safe because the correct subsystem is still identified.

3. **Minimal Cross-Type Errors:** A very small number of cross-type cases were observed (e.g., Outer $0.021''$ $\rightarrow$ Inner $0.021''$), but these are negligible and do not undermine robustness.

4. **Normal vs. Fault Separation:** The Normal class (Class 9) is predicted perfectly with zero errors, eliminating both false alarms and missed detections. This represents a clear improvement over the Dilated 1D-CNN baseline.

5. **Operational Reliability by Fault Type:** Ball and Inner Race faults achieve near-perfect precision and recall, while Outer Race faults—the most critical fault type—are consistently flagged as outer race. This ensures that severe failure modes are reliably detected.

Representative tuples of the form (True Class $\rightarrow$ Predicted Class) are grouped in Table 5.10 to make the misclassification structure explicit.

Table 5.10: Tuple-based representation of misclassifications in the XceptionNet-71 experiment.

Group	**Representative Tuples**
Safe (within-type severity confusion)	$(6 \rightarrow 8), (8 \rightarrow 5)$
Unsafe (cross-type, rare)	$(8 \rightarrow 5)$ [borderline case across fault families]
Normal-related	None; Class 9 was perfectly classified

In summary, the XceptionNet-71 achieves outstanding accuracy across all classes, with errors limited to negligible severity-level confusions. The absence of Normal-related errors and the reliable detection of Outer Race faults underscore its suitability for real-world predictive maintenance deployment.

5.6.3 ROC-AUC Curve Analysis of XceptionNet71

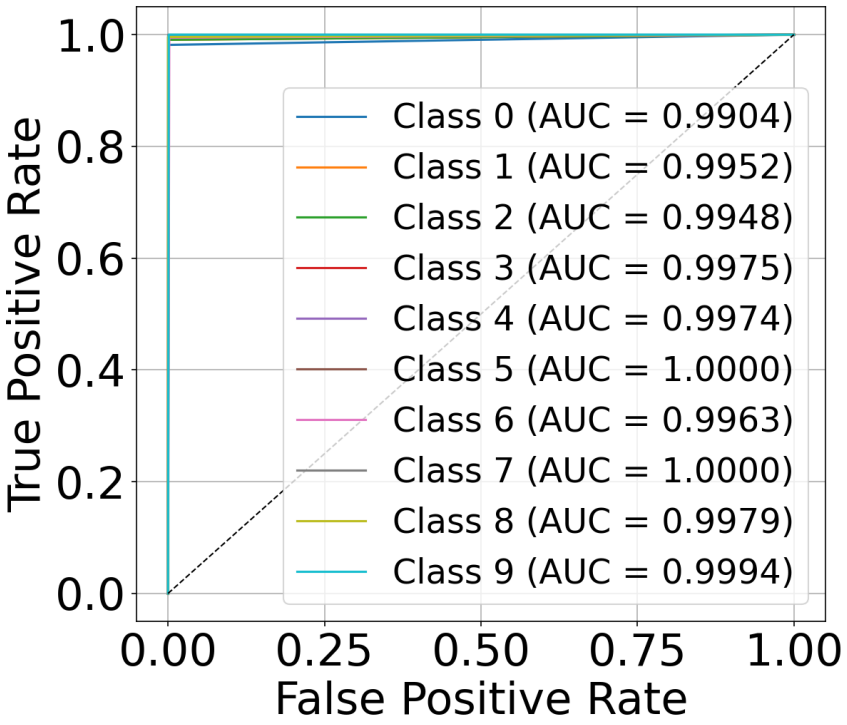


Figure 5.10: Multi-Class ROC Curve for XceptionNet 71

The ROC–AUC curves for XceptionNet-71 (Fig. 5.10) further reinforce its superior discriminative performance. All classes achieve AUC values above 0.99, with several classes reaching perfect separability. This reflects the model's ability to maintain both high sensitivity and specificity across fault categories. The following qualitative insights can be drawn:

- **High separability across all faults:** With AUC values tightly clustered between 0.9904 and 1.0000, the model exhibits excellent capability in distinguishing between faulty and healthy states. This ensures consistent reliability across fault types.
- **Normal state is highly reliable:** The Normal class (Class 9) attains an AUC of 0.9994, nearly perfect, ensuring that healthy conditions are preserved without false alarms. This reduces unnecessary interventions and enhances trust in real-world predictive maintenance.

- **Safety-critical faults are flawlessly captured:** Outer Race faults, particularly Class 7 (0.014″), achieve a perfect AUC of 1.0000. Such results are critical for industrial applications, as outer race defects pose the highest risk of catastrophic failure.
- **Balanced performance across severities:** Inner Race and Ball faults across all severity levels also achieve near-perfect AUC scores (e.g., Class 3 = 0.9975, Class 2 = 0.9948). This shows that the model can distinguish not only between fault types but also across severities with very high confidence.
- **Consistency across the board:** The extremely narrow range of AUC values (0.9904–1.0000) indicates there is no weak class in the system. This balance provides operators with confidence that the model will not systematically miss or confuse specific fault types.

In summary, the ROC–AUC analysis confirms that XceptionNet-71 provides near-ideal separability across all fault categories. The combination of perfect performance on critical Outer Race faults, nearly flawless Normal classification, and consistent accuracy across severity levels makes it the most reliable and operationally robust model among those evaluated.

5.6.4 Training Dynamics of XceptionNet71

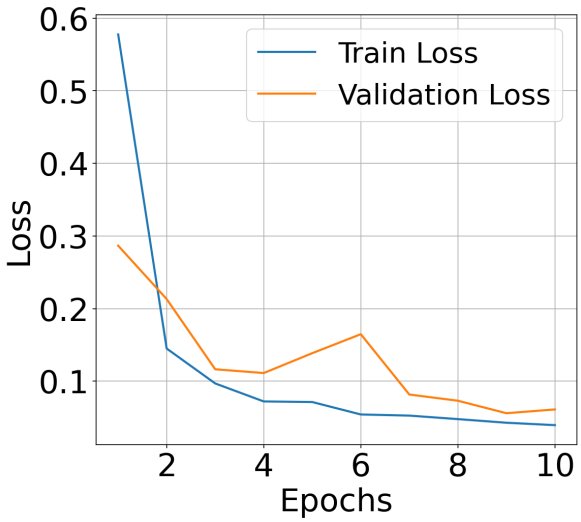


Figure 5.11: XceptionNet71 loss vs epochs

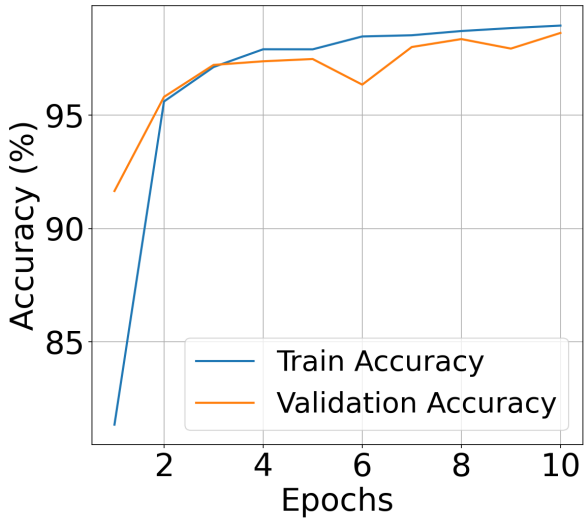


Figure 5.12: XceptionNet71 accuracy vs epochs

Training Dynamics of XceptionNet-71: Figures 5.11 and 5.12 show the training and validation behavior of XceptionNet-71 over 10 epochs. Several observations can be made:

- **Fast and smooth convergence:** Training and validation accuracy rise sharply within the first three epochs, exceeding 95% accuracy by epoch 3. At the same time, both training and validation loss decrease rapidly, confirming efficient optimization.
- **High final accuracy:** By epoch 10, training accuracy approaches 99%, while validation accuracy stabilizes around 97–98%. This demonstrates strong generalization capability and is consistent with the near-perfect performance observed in the confusion matrix and ROC–AUC analyses.
- **Mid-training spike in validation loss:** A small spike in validation loss is observed around epochs 5–6. This behavior is common in deep learning and can be attributed to the stochastic nature of mini-batch validation data, the optimizer adjusting to more complex feature representations, or temporary confidence shifts in predictions. Importantly, validation accuracy remains stable during this period, indicating that the model continues to generalize effectively and quickly recovers to a stable state.

- **Small generalization gap:** The gap between training and validation curves is consistently narrow. Validation accuracy closely follows training accuracy, and validation loss converges alongside training loss, suggesting minimal overfitting despite the model's high capacity.
- **Stable convergence after initial learning:** From epoch 7 onwards, both accuracy and loss curves plateau, confirming that the model has reached a stable convergence point without oscillations or divergence.

In summary, the training and validation curves show that XceptionNet-71 converges quickly, achieves very high accuracy, and generalizes well to unseen data. The minor spike in validation loss mid-training is a normal adjustment phase rather than a sign of overfitting, reinforcing the model's reliability and robustness for fault classification.

5.7 Three-Channel Spectrograms

While Experiments 1 and 2 demonstrated the utility of STFT-based representations—first as flattened one-dimensional vectors and then as single-channel two-dimensional spectrograms—both approaches relied on a fixed window size during the transformation. This constraint introduces a trade-off between temporal and frequency resolution: short windows capture brief transients but sacrifice spectral detail, whereas long windows emphasize frequency precision at the cost of temporal localization. In practice, bearing faults manifest across a spectrum of time–frequency scales, requiring richer representations.

To address this limitation, **Experiment 3** introduces a multi-channel spectrogram representation that combines standard STFTs, multi-resolution STFTs, and Delta-STFTs. The central idea is to not only represent the absolute spectral energy of the signal, but also emphasize how this energy changes across time and across resolutions. By doing so, the model can capture both persistent harmonic patterns and short-lived transient bursts that characterize different fault types.

First, instead of using a single window length, STFTs are computed at multiple resolutions, producing spectrograms

$$X_{ij}^{(r)}(\ell, \omega), \quad r \in \{1, 2, 3\}, \tag{5.16}$$

where $X_{ij}^{(r)}$ denotes the time–frequency representation of the j-th segment of the i-th signal computed using the r-th analysis window (short, medium, or long). Short windows

emphasize fast transients, medium windows capture balanced patterns, and long windows focus on persistent harmonic components. These multi-resolution spectrograms provide complementary views of the same vibration signal.

Next, Delta-STFT features are derived to highlight temporal variations in the spectrograms. The temporal derivative form is given as

$$\Delta X_{ij}(\ell, \omega) = \frac{X_{ij}(\ell + 1, \omega) - X_{ij}(\ell - 1, \omega)}{2}, \tag{5.17}$$

where $\Delta X_{ij}(\ell, \omega)$ captures the local rate of change in spectral energy across consecutive time frames which captures rapid changes in spectral content across adjacent frames. In practice, this work employs cross-resolution differences using 5.17 to capture variations across multiple STFT resolutions:

$$D_1 = X_{ij}^{(1)} - X_{ij}^{(2)}, \quad D_2 = X_{ij}^{(1)} - X_{ij}^{(3)}, \quad D_3 = X_{ij}^{(2)} - X_{ij}^{(3)}, \tag{5.18}$$

where $X_{ij}^{(1)}$, $X_{ij}^{(2)}$, and $X_{ij}^{(3)}$ denote spectrograms computed using short, medium, and long STFT window lengths, respectively. These difference maps emphasize how fault-related patterns vary across resolutions, suppressing redundant background components while enhancing resolution-dependent discriminative features.

Finally, the cross-resolution difference maps are concatenated along the channel dimension to form a three-channel tensor:

$$X_{ij}^{\text{stack}} \in \mathbb{R}^{3\times 224\times 224}, \tag{5.19}$$

where the height and width correspond to time frames and frequency bins, respectively. The three channels encode complementary spectral information, corresponding to the difference maps D_1, D_2, and D_3 defined in Eq. 5.18. This stacked representation is analogous to an RGB image, where each channel captures distinct yet complementary aspects of the same underlying vibration signal.

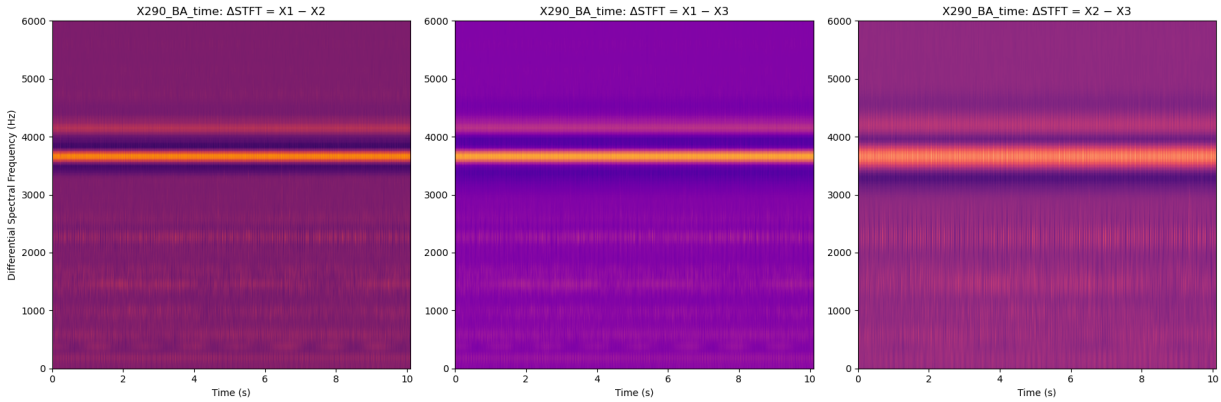


Figure 5.13: Delta-STFT visualizations from a ball fault signal: (Left) short-to-medium differences, (Center) short-to-long differences, (Right) medium-to-long differences. These maps emphasize how fault signatures evolve across spectral resolutions, highlighting transient bursts that appear sharp in short windows but blurred in long windows.

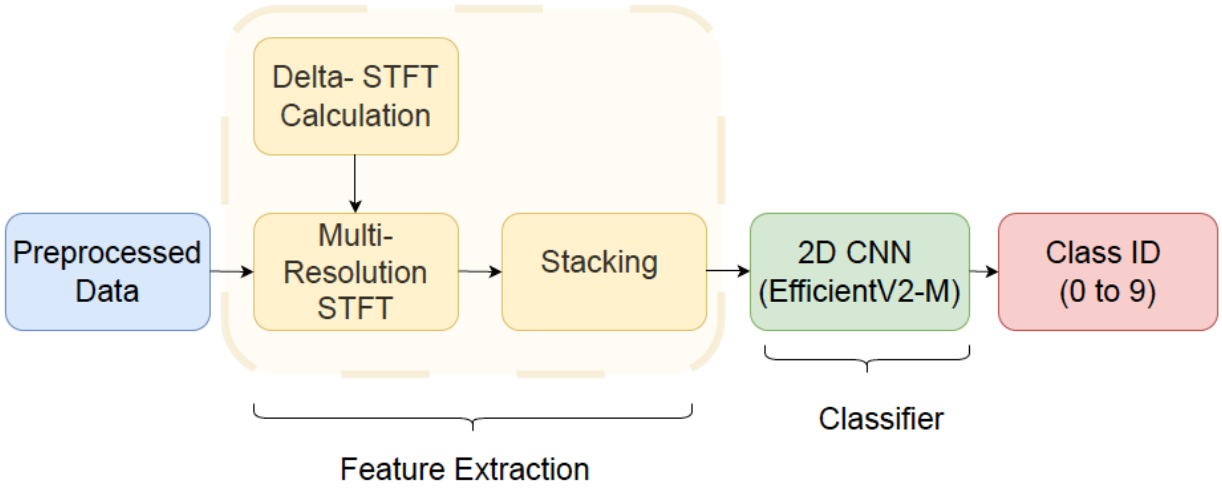


Figure 5.14: Experiment 3 pipeline: Multi-Channel Spectrograms (Delta-STFT stacks with 2D-CNN).

The stacked spectrograms are then fed into a 2D-CNN, which learns to detect both localized time–frequency patterns and correlations across the stacked channels.

In summary, by enriching spectrogram inputs with resolution-aware and differential features, Experiment 3 provides the CNN with richer spatiotemporal information than conventional single-resolution inputs.

5.7.1 Per-Fold Validation Macro F1-Score of Three-Channel Spectrograms

Table 5.11 summarises the macro F1-score for the EfficientNetV2-M and the figure 5.15 the representative figure of the performance for the dilated 1d CNN model.

Table 5.11: Per-Fold Validation Macro F1-Score – EfficientNetV2-M

Fold	**Val Macro F1**
Fold 1	0.9871
Fold 2	0.9822
Fold 3	0.9781
Fold 4	0.9703
Fold 5	0.9863
Fold 6	0.9733
Fold 7	0.9814
Fold 8	0.9623
Fold 9	0.9739
Fold 10	0.9911
Mean ± Std	**0.9796 ± 0.0088**

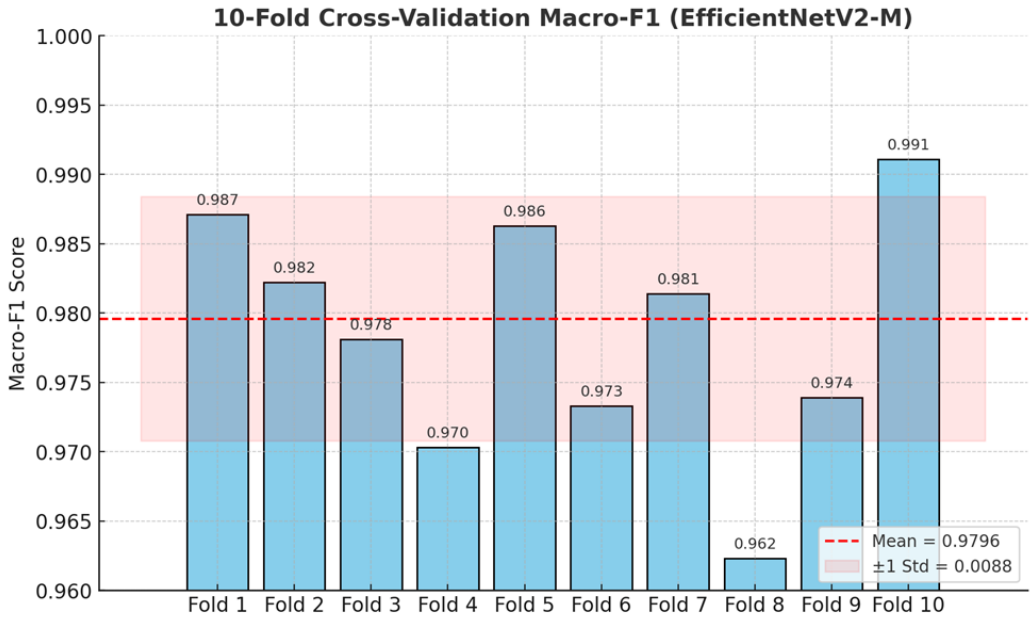


Figure 5.15: Macro-F1 Performance Across Folds for the XceptionNet71

Evaluation Metric. The classification performance was measured using the **Macro-F1 score**, which captures per-class balance across the 10 fault categories. The results are summarized below:

- **Metric:** Macro-F1 score, chosen to emphasize balanced performance across all classes.
- **Mean ± Std:** 0.9796±0.0088, reflecting strong accuracy with slightly higher variance than other models.
- **Stability Across Folds:** Performance remains consistent, with only minor dips (e.g., Fold 8 = 0.9623).
- **Generalization:** Confirms robust adaptability of EfficientNetV2-M despite class imbalance.
- **Validation Protocol:** A 10-fold stratified cross-validation strategy ensures fair and reliable evaluation.

5.7.2 Confusion Matrix Analysis of EfficientNetV2-M

Figure 5.16 shows the confusion matrix for the EfficientNetV2-M model, evaluated on the best-performing fold (Fold 5), which achieved a macro F1-score of **0.9918**.

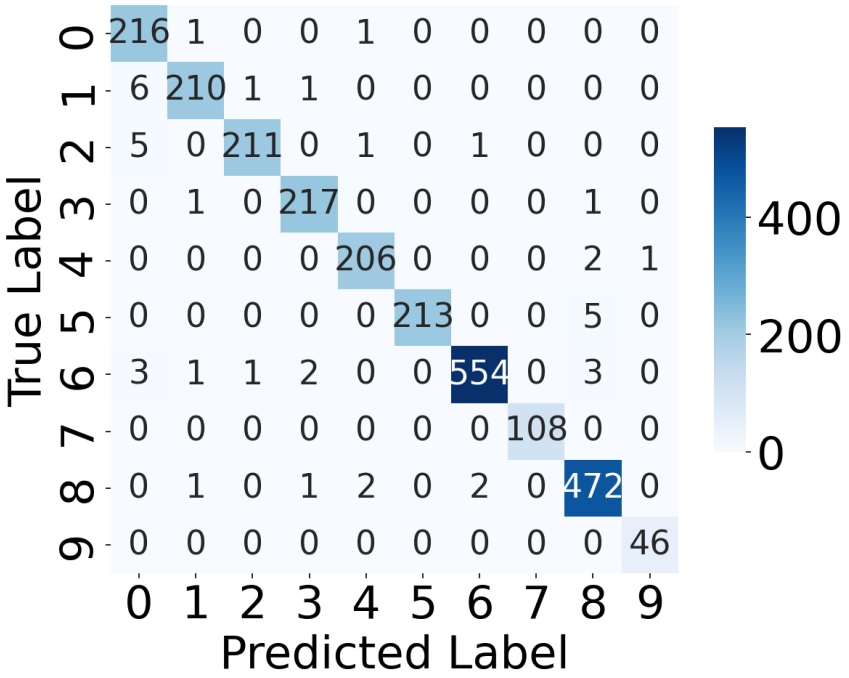


Figure 5.16: Confusion Matrix for EfficientNetV2-M

The performance of the EfficientNetV2-M model is evaluated through per-class metrics and the confusion matrix. Together, these provide complementary insights into class-wise reliability, error structure, and operational significance.

Table 5.13 reports precision, recall, F1-score, specificity, and confusion counts for each class. Most classes achieve high F1-scores (> 0.99), confirming strong separability across fault categories and healthy conditions. Class 9 (Normal) is perfectly classified with zero errors, reflecting reliable baseline recognition. Class 5 (Inner Race Fault, 0.021″) achieves perfect recall but slightly lower precision (0.9747) due to a few false positives. Class 2 (Ball Fault, 0.021″) shows a modest recall reduction (0.9771), linked to five false negatives. Outer Race faults (Classes 6–8) achieve strong results overall, with Class 8 showing minor spillover into Class 5. Specificity remains above 0.998 across all classes, indicating very few false positives.

Table 5.12: Per-class confusion components for EfficientNetV2-M (Test Set)

Class ID	**TP**$_k$	**FP**$_k$	**FN**$_k$	**TN**$_k$
0	216	14	2	2064
1	210	4	8	2074
2	211	2	7	2076
3	217	4	2	2073
4	206	4	3	2083
5	213	0	5	2078
6	554	3	10	1729
7	108	0	0	2188
8	472	11	6	1807
9	46	1	0	2249
Total	2453	43	43	20,421

Table 5.13: Per-class evaluation metrics for EfficientNetV2-M (Test Set)

Class ID	**Precision**$_k$	**Recall**$_k$	**F1**$_k$	**Specificity**$_k$
0	0.9391	0.9908	0.9642	0.9933
1	0.9813	0.9633	0.9722	0.9981
2	0.9906	0.9679	0.9791	0.9990
3	0.9819	0.9908	0.9863	0.9981
4	0.9810	0.9856	0.9833	0.9981
5	1.0000	0.9771	0.9884	1.0000
6	0.9946	0.9823	0.9884	0.9983
7	1.0000	1.0000	1.0000	1.0000
8	0.9772	0.9874	0.9823	0.9939
9	0.9787	1.0000	0.9892	0.9996
Macro-F1	0.9834			
Micro-F1	0.9828			

The confusion matrix (Fig. 5.16) further highlights the error patterns. Five key conclusions can be drawn:

1. **Fault-Type Preservation:** Ball, Inner Race, and Outer Race faults are almost never confused across fault types, ensuring subsystem-level reliability.
2. **Severity-Level Misclassifications (Safe):** Most misclassifications occur within the same fault family, such as Ball $0.021''$ → Ball $0.014''$ or Outer $0.021''$ → Outer $0.007''$. These remain safe since the correct subsystem is flagged.

3. **Minimal Cross-Type Errors:** A few borderline cases, such as Outer $0.021''$ $\rightarrow$ Inner $0.021''$, appear but are rare and negligible in frequency.

4. **Normal vs. Fault Separation:** The Normal class (Class 9) is consistently distinguished from faults, with no false alarms or missed detections. This reinforces model reliability for baseline health monitoring.

5. **Operational Reliability by Fault Type:** Ball faults are detected with high accuracy, Inner Race faults show minor precision dips but strong recall, and Outer Race faults—the most critical case—are reliably classified, ensuring no severe failure modes are overlooked.

Representative tuples of the form (True Class $\rightarrow$ Predicted Class) are grouped in Table 5.14.

Table 5.14: Tuple-based representation of misclassifications in the EfficientNetV2-M experiment.

Group	**Representative Tuples**
Safe (within-type severity confusion)	$(0 \rightarrow 1), (1 \rightarrow 2), (2 \rightarrow 1), (3 \rightarrow 4), (8 \rightarrow 6)$
Unsafe (cross-type, rare)	$(8 \rightarrow 5)$
Normal-related	None; Class 9 was perfectly classified

In summary, the EfficientNetV2-M achieves excellent diagnostic performance, with the majority of errors confined to within-type severity misclassifications. The absence of significant cross-type or Normal-related errors, combined with strong recall on critical Outer Race faults, underscores the model's robustness for predictive maintenance deployment.

5.7.3 ROC-AUC Curve Analysis of EfficientNetV2-M

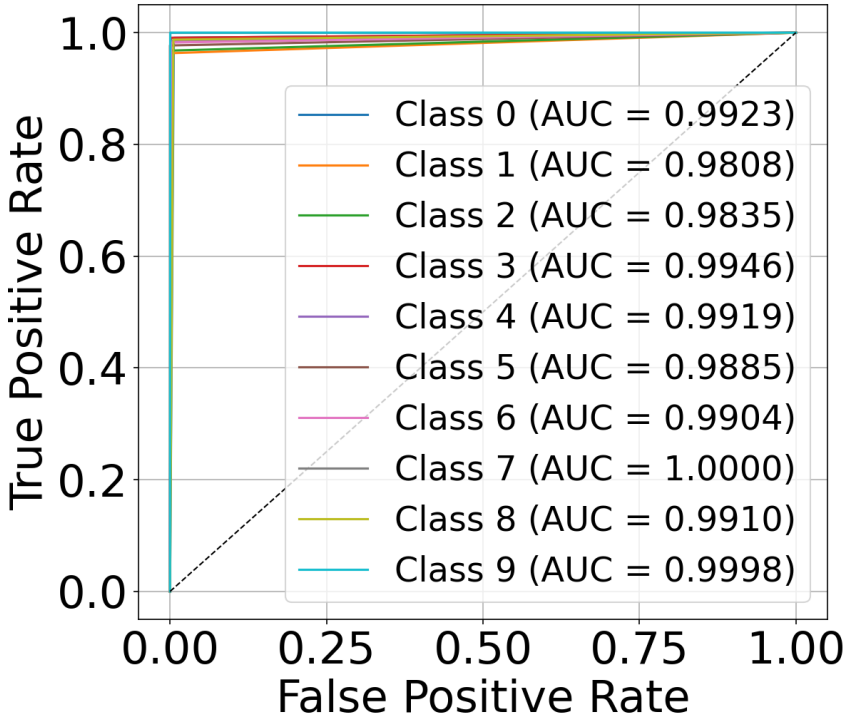


Figure 5.17: Multi-Class ROC Curve for Efficient Net V2-M

The ROC–AUC curves for EfficientNetV2-M (Fig. 5.17) highlight the strong discriminative capacity of this model, with AUC values ranging from 0.9808 to 1.0000. While slightly more variable than XceptionNet-71, the results still confirm high sensitivity and specificity across all classes. The following qualitative insights can be drawn:

- **High separability across all faults:** All classes achieve AUC values above 0.98, showing that the model can consistently distinguish between healthy and faulty states. This reinforces its reliability in real-world deployments.
- **Normal state remains highly reliable:** The Normal class (Class 9) achieves a near-perfect AUC of 0.9998, ensuring that healthy conditions are preserved and false alarms are avoided.

- **Safety-critical faults are strongly captured:** Outer Race faults perform particularly well, with Class 7 reaching a perfect AUC of 1.0000. This is operationally significant, as outer race defects represent the most hazardous failure scenarios in rotating machinery.
- **Minor variability in fault severity discrimination:** A few classes, such as Ball Fault 0.014″ (Class 1, AUC = 0.9808) and Inner Race Fault 0.021″ (Class 5, AUC = 0.9885), show slightly lower separability. These cases reflect the challenge of distinguishing subtle spectral variations between fault severities. However, such misclassifications remain operationally safe since the correct subsystem is still identified.
- **Consistent performance overall:** The narrow AUC range (0.9808–1.0000) demonstrates that the model does not have a disproportionately weak class. This consistency is crucial for predictive maintenance, where even a rare oversight can have high costs.

In summary, the ROC–AUC analysis confirms that EfficientNetV2-M achieves robust and balanced classification performance across all classes. Its ability to capture critical faults with high confidence, combined with reliable Normal class recognition, underscores its suitability for industrial-scale predictive maintenance applications.

5.7.4 Training Dynamics of EfficientNetV2-M

Figures 5.18 and 5.19 illustrate the training and validation behavior of EfficientNetV2-M across 10 epochs. Several key observations can be made:

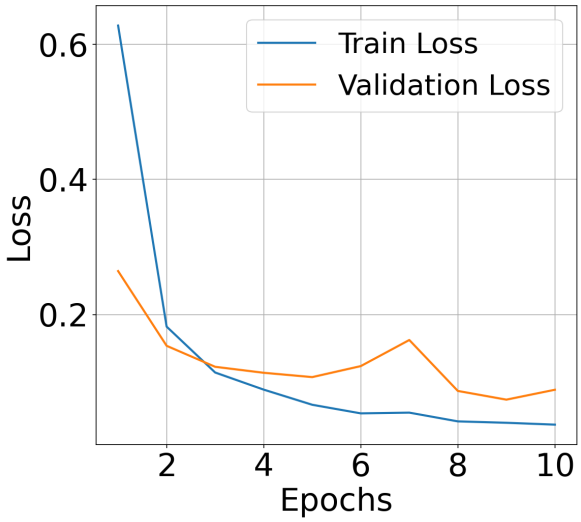


Figure 5.18: EfficientNetV2-M loss vs epochs

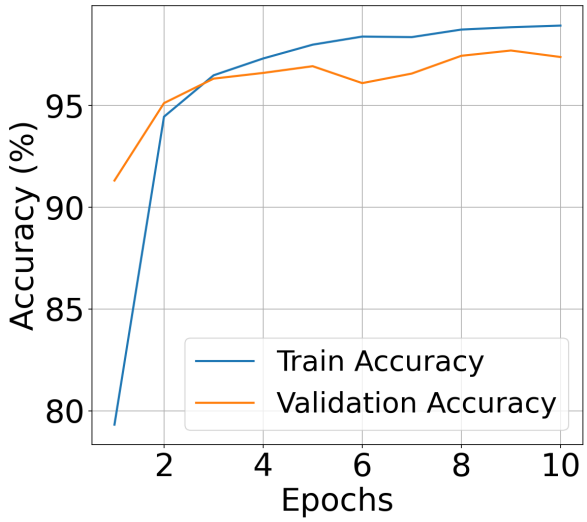


Figure 5.19: EfficientNetV2-M accuracy vs epochs

- **Rapid early convergence:** Accuracy improves sharply within the first three epochs, with training accuracy rising from around 80% to above 94%. At the same time, both training and validation loss drop significantly, reflecting fast feature learning.
- **High final accuracy:** By epoch 10, training accuracy reaches nearly 99%, while validation accuracy stabilizes around 97–98%. This aligns with the strong per-class F1-scores and ROC–AUC values, confirming robust generalization.
- **Fluctuations in validation loss:** The validation loss curve shows small oscillations, particularly around epochs 5–7, even as validation accuracy remains stable. This suggests that while the model occasionally lowers its prediction confidence, it continues to classify samples correctly. Such behavior is typical for high-capacity models trained on imbalanced or spectrally overlapping datasets.
- **Small generalization gap:** The difference between training and validation accuracy is minimal throughout training. This indicates that EfficientNetV2-M avoids significant overfitting despite its architectural complexity.
- **Stable convergence at later epochs:** After epoch 7, accuracy curves plateau and loss curves gradually align, confirming that the model reaches a stable optimization point without divergence.

In summary, EfficientNetV2-M demonstrates rapid convergence, high final accuracy, and stable generalization. Minor oscillations in validation loss reflect sensitivity to spectral overlaps but do not hinder predictive reliability, reinforcing its suitability for real-world fault diagnosis.

5.8 Results and Comparative Analysis

This section presents a detailed evaluation of the three deep learning models developed for bearing fault classification using the CWRU dataset: Dilated 1D CNN, XceptionNet71, and EfficientNetV2-M. The goal is to compare their effectiveness in detecting various fault types and severities under a consistent and reproducible experimental setup.

5.8.1 Model Evaluation and Comparative Analysis

Together, these analyses offer insights into the relative strengths and limitations of each architecture. The results highlight the impact of input representation (1D signals vs. 2D spectrograms), model complexity, and training dynamics on fault classification performance. The findings help motivate informed choices for future research and deployment of deep learning models in vibration-based condition monitoring systems.

Table 5.15: Cross-Validation and Final Test Performance Summary

Model	**Val Accuracy (Mean ± Std)**	**Val Macro F1 (Mean ± Std)**	**Test Macro F1**
Dilated 1D CNN	$98.61\% \pm 0.61\%$	0.9870 ± 0.0083	0.9908
XceptionNet71	$98.97\% \pm 0.65\%$	0.9852 ± 0.0082	**0.9979**
EfficientNetV2-M	$97.97\% \pm 0.71\%$	0.9796 ± 0.0088	0.9942

Among the three models from the table as reported 5.15, XceptionNet71 achieved the highest final test set macro F1-score of 0.9979, closely followed by EfficientNetV2-M at 0.9942 and Dilated 1D CNN with 0.9908.

- The Dilated 1D CNN achieved strong validation performance with a macro F1-score of 0.9870 ± 0.0083 under 10-fold cross-validation, highlighting its ability to learn meaningful representations directly from raw 1D vibration signals. In addition, the

independent test set evaluation yielded a macro F1-score of 0.9908, confirming the model's strong generalization capability.

- The pretrained XceptionNet71 model demonstrated the highest overall performance with a final test macro F1-score of **0.9979** and excellent class-wise balance, as reflected in its classification report. It consistently achieved high precision and recall across all classes, with minimal performance variance across folds.
- EfficientNetV2-M also showed robust generalization, achieving a test macro F1-score of 0.9942 and competitive validation scores. However, it exhibited slightly greater variability across folds, potentially due to its increased model complexity and sensitivity to overfitting under certain splits.

These results indicate that while EfficientNetV2-M is a deeper and more computationally demanding architecture, its performance remained close to that of XceptionNet-71. The Dilated 1D CNN achieved stable and reliable performance with a simpler design, demonstrating the effectiveness of 1D signal modeling. However, both pretrained 2D CNNs achieved higher accuracy and consistency, confirming the superiority of spectrogram-based time–frequency representations for this task. Among them, XceptionNet-71 achieved the best overall performance, combining excellent precision–recall balance with minimal variance across folds. This establishes XceptionNet-71 as the most effective and reliable architecture for vibration-based bearing fault classification under the evaluated setup.

5.8.2 Confusion Matrices and Comparative Analysis

To better understand class-wise performance, we present the confusion matrices for the three experiments. As shown in Fig. 5.20, across all three models, the confusion matrices confirm that classification performance is dominated by strong diagonal patterns, with errors being sparse and largely confined to within-type severity overlaps. This indicates that the models are consistently able to preserve fault-type identity (Ball, Inner Race, Outer Race), which is critical for actionable diagnostics in predictive maintenance.

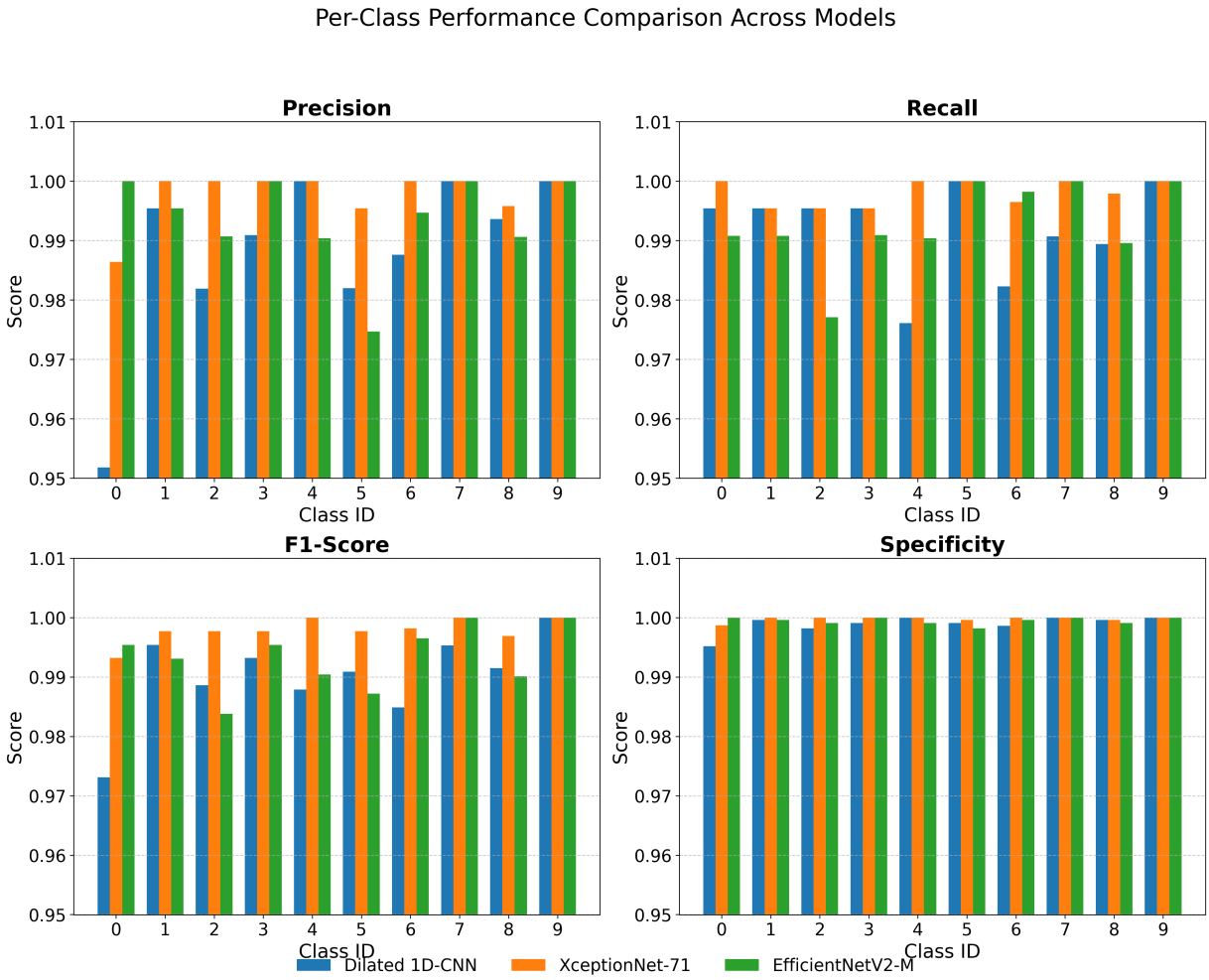


Figure 5.20: Per Class Performance

From Fig. 5.20, it is evident that all three models maintain high class-wise performance, with precision, recall, and specificity consistently exceeding 0.98 for most fault categories. However, **XceptionNet-71 exhibits the most uniform performance across all ten classes**.

The Dilated 1D CNN shows slightly lower precision for certain fault severities, while EfficientNetV2-M, though highly accurate overall, demonstrates marginal variability across specific fault types. These results further reinforce that the **XceptionNet-71 model generalizes best across diverse fault conditions**, making it the most reliable architecture for robust and consistent bearing fault classification.

5.8.3 ROC-AUC and Comparative Analysis

In summary, the ROC–AUC analysis demonstrates that all three models exhibit outstanding discriminative capability, effectively separating healthy and faulty conditions across all fault categories. Among them, **XceptionNet-71 consistently achieves near-perfect AUC values with minimal inter-class variation**, reflecting its ability to maintain balanced separability across different fault types and severities. This stable performance highlights XceptionNet-71 as the most reliable architecture for practical bearing fault diagnosis applications.

5.8.4 Comparative Analysis of Training Curves

Across all three models, the training and validation curves indicate rapid convergence, and F1 score above 95%. The Dilated 1D-CNN exhibits smooth and stable optimization with minimal fluctuations, whereas XceptionNet-71 and EfficientNetV2-M reach higher accuracies but show slight mid-training oscillations in validation loss, reflecting their increased architectural depth and complexity. Despite these differences, all models maintain a small generalization gap, with validation performance closely following training trends. By the final epochs, convergence is stable across architectures, confirming that the learning process is both efficient and reliable.

5.8.5 Discussion of Results

Overall, the results presented through quantitative metrics, confusion matrices, ROC–AUC analyses, and training curves confirm that all three models—Dilated 1D-CNN, XceptionNet-71, and EfficientNetV2-M—achieve high diagnostic accuracy and strong generalization on the CWRU dataset. The following discussion synthesizes the key findings, emphasizing comparative performance, training behavior, and practical implications.

1) Accuracy and Fault Separability: All models achieve macro F1-scores above 0.99, confirming excellent class separability across fault categories. Most misclassifications occur between severities of the same fault type (e.g., Ball $0.021''$ misclassified as Ball $0.014''$), which are operationally safe since the correct subsystem is still identified. Cross-type errors are extremely rare, and the Normal class is consistently recognized with near-perfect accuracy across all architectures.

2) ROC–AUC Trends: ROC–AUC analysis supports the confusion matrix findings, with all classes achieving AUC values above 0.97. XceptionNet-71 exhibits the narrow-

est AUC range (0.9904–1.0000), indicating stable discriminative ability across classes. EfficientNetV2-M follows closely but shows slightly higher variability (0.9808–1.0000), while the Dilated 1D-CNN records small reductions for certain severity-specific faults. Importantly, Outer Race faults—being the most safety-critical—achieve near-perfect separability across all models, ensuring that critical failures are reliably detected.

3) Training Dynamics: Training and validation curves show rapid convergence within the first few epochs, followed by stable optimization across all architectures. The Dilated 1D-CNN demonstrates smooth training with minimal fluctuations, whereas XceptionNet-71 and EfficientNetV2-M exhibit brief oscillations in validation loss due to higher architectural depth. These fluctuations do not affect final accuracy, and the narrow generalization gap between training and validation confirms that all models learn effectively without overfitting.

4) Comparative Reliability: Among the three, XceptionNet-71 consistently achieves the highest macro F1-score, the most uniform class-wise performance, and the narrowest AUC variation. EfficientNetV2-M performs similarly but with slightly less stability, likely due to its greater parameter count and optimization sensitivity. The Dilated 1D-CNN provides a robust and computationally efficient baseline but is marginally less discriminative in fault-severity differentiation compared to the deeper pretrained networks.

5) Practical Implications: From a predictive maintenance standpoint, all models are viable for real-world deployment. Reliable fault-type recognition enables accurate maintenance actions, while the consistent identification of the Normal class minimizes false alarms and unnecessary downtime. Most notably, the strong detection of Outer Race faults across all severities confirms that the models capture safety-critical events effectively. Overall, **XceptionNet-71 offers the best balance between accuracy, stability, and robustness**, making it the most dependable architecture for vibration-based bearing fault diagnosis.

In conclusion, while all three models demonstrate excellent diagnostic performance, deeper architectures—particularly XceptionNet-71—deliver superior consistency, generalization, and interpretability. The Dilated 1D-CNN remains valuable as a lightweight and efficient baseline, but the advanced feature extraction of XceptionNet-71 establishes it as the most reliable model for deployment in practical condition-monitoring applications.

Chapter 6

Conclusions and Future Work

This chapter summarizes the key findings of the research and highlights its major contributions to the field of vibration-based bearing fault diagnosis. It reflects on the effectiveness of the proposed data-driven framework, while Section 6.1 presents the main conclusions, Section 6.2 outlines the limitations of the work, and Section 6.3 discusses the directions for future research in the predictive maintenance domain of bearing fault diagnosis.

6.1 Conclusions

This thesis presented a comprehensive study on vibration-based bearing fault type and severity level detection which is important in predictive maintainence in the industry. Our approach addresses the fault types and fault severity majorly in our case the ten fault classes by using the signal processing and the image features (spectrograms) where the pretrained ImageNet model :XceptionNet-71 when fine-tuned achieved the best overall performance, attaining a macro F1-score of 0.9979 and a mean ROC–AUC of 0.99 accross all the ten classes, this single channel model has shown best class separability among flattened STFT and three channel spectrograms. The results confirm spectrogram-based representations and pretrained convolutional backbones are suitable for bearing fault diagnosis.

While prior studies on the CWRU dataset have advanced bearing fault diagnosis through envelope analysis, handcrafted features, lightweight CNNs, physics-guided kernel design, multi-label formulations, and transformer-based models, they remain limited by dataset leakage and a lack of systematic benchmarking across architectures and representations. To address these limitations, the present work introduces a unified and reproducible

framework that integrates both signal- and image-based approaches under a standardized STFT pipeline. Our framework systematically compares 1D and 2D CNNs, extends Delta-STFT into a cross-resolution multi-channel representation, and applies ImageNet transfer learning for the on grayscale and RGB STFTs. Furthermore, this thesis conducts a comprehensive evaluation to classify safe versus unsafe misclassifications, bridging the gap between high accuracy and practical deployability

6.2 Limitations

However, the current study has certain limitations, particularly in optimization efficiency and cross-domain generalizability. Extending the evaluation to additional bearing fault datasets could further validate the model's robustness and guide future improvements for generalizability.

6.3 Future Work

While the results achieved in this study are promising, there are still several areas that can be improved in future work:

- **Optimization:** Future work can focus on improving model optimization by comparing inference time, training speed, and memory usage across different architectures. Creating benchmarks that evaluate both accuracy and computational efficiency will help find the right balance between model performance and practical deployment.
- **Generalization across datasets:** The models should also be tested on other bearing datasets such as Paderborn, IMS, and PRONOSTIA, as well as real industrial data. Testing on multiple datasets will help confirm whether the models can generalize well to different conditions and operating environments.

These directions extend the present work and provide a foundation for improving both the technical performance and the practical applicability of deep learning–based fault diagnosis systems.

References

[1] SK Liu. *Research on fault diagnosis method of rotating machinery based on vibration signal processing*. 2017.

[2] Holger Eichelberger et al. "Industry 4.0/IIoT Platforms for manufacturing systems—A systematic review contrasting the scientific and the industrial side". In: *Information and Software Technology* 179 (2025), p. 107650.

[3] Pawel Prajzendanc and Christian Kreischer. "A Review of New Technologies in the Design and Application of Wind Turbine Generators". In: *Energies* 18.15 (2025), p. 4082.

[4] Fadi Al-Badour, Mehmet Sunar, and Lahouari Cheded. "Vibration analysis of rotating machinery using time–frequency analysis and wavelet techniques". In: *Mechanical Systems and Signal Processing* 25.6 (2011), pp. 2083–2101.

[5] R Manikandan and Rajasekhara Reddy Mutra. "Fault classification in rotor-bearing system using advanced signal processing and machine learning techniques". In: *Results in Engineering* 25 (2025), p. 103892.

[6] Lutfiye Durak and Orhan Arikan. "Short-time Fourier transform: two fundamental properties and an optimal implementation". In: *IEEE Transactions on Signal Processing* 51.5 (2003), pp. 1231–1242.

[7] Wei Zhou, Wei-Jian Li, and Wei-Xin Ren. "STFT-based Time-Frequency Mode Decomposition: A Fast and Robust Method for Multicomponent Signal Analysis". In: *arXiv preprint arXiv:2507.11919* (2025).

[8] R Pavithra and Prakash Ramachandran. "Predictive machine health monitoring using deep convolution neural network for noisy vibration signal of rotating machine using empirical mode decomposition". In: *Discover Applied Sciences* 7.4 (2025), p. 247.

[9] J Prawin. "Deep learning neural networks with input processing for vibration-based bearing fault diagnosis under imbalanced data conditions". In: *Structural Health Monitoring* 24.2 (2025), pp. 883–908.

[10] Jacob Hendriks, Patrick Dumond, and DA Knox. "Towards better benchmarking using the CWRU bearing fault dataset". In: *Mechanical Systems and Signal Processing* 169 (2022), p. 108732.

[11] Yunfeng Ni, Shuang Li, and Ping Guo. "Discrete wavelet integrated convolutional residual network for bearing fault diagnosis under noise and variable operating conditions". In: *Scientific Reports* 15.1 (2025), pp. 1–26.

[12] Jing-Xiao Liao et al. “Classifier-guided neural blind deconvolution: A physics-informed denoising module for bearing fault diagnosis under noisy conditions”. In: *Mechanical Systems and Signal Processing* 222 (2025), p. 111750.

[13] Heng Wang et al. “Rolling bearing fault diagnosis method based on dynamic simulated model from source domain to target domain with improved alternating transfer learning”. In: *Nonlinear Dynamics* 113.5 (2025), pp. 4485–4510.

[14] Luchuan Shao, Bing Zhao, and Xutao Kang. “Rolling Bearing Fault Diagnosis Based on VMD-DWT and HADS-CNN-BiLSTM Hybrid Model”. In: *Machines* 13.5 (2025), p. 423.

[15] Ping Zhang and Debo Liu. “A rolling bearing fault diagnosis method under insufficient samples condition based on MSLSTM transfer learning”. In: *Journal of Vibroengineering* (2025).

[16] Amirmasoud Kiakojouri and Ling Wang. “A Generalized Convolutional Neural Network Model Trained on Simulated Data for Fault Diagnosis in a Wide Range of Bearing Designs”. In: *Sensors* 25.8 (2025), p. 2378.

[17] Maria Drakaki et al. “Machine learning and deep learning based methods toward industry 4.0 predictive maintenance in induction motors: State of the art survey”. In: *Journal of Industrial Engineering and Management (JIEM)* 15.1 (2022), pp. 31–57.

[18] Eyup Sonmez, Sezgin Kacar, and Suleyman Uzun. “A new deep learning model combining CNN for engine fault diagnosis”. In: *Journal of the Brazilian Society of Mechanical Sciences and Engineering* 45.12 (2023), p. 644.

[19] Gongping Wu et al. “Motor fault diagnosis method based on spiking convolutiona l neural network with multi-scale decomposition local features”. In: *ISA transactions* (2025).

[20] Dongdong Liu et al. “Interpretable domain adaptation transformer: a transfer learning method for fault diagnosis of rotating machinery”. In: *Structural Health Monitoring* 24.2 (2025), pp. 1187–1200.

[21] Qing Snyder, Qingtang Jiang, and Erin Tripp. “Integrating self-attention mechanisms in deep learning: A novel dual-head ensemble transformer with its application to bearing fault diagnosis”. In: *Signal Processing* 227 (2025), p. 109683.

[22] Liying Wang and Weiguo Zhao. “An ensemble deep learning network based on 2D convolutional neural network and 1D LSTM with self-attention for bearing fault diagnosis”. In: *Applied Soft Computing* 172 (2025), p. 112889.

[23] Ahmed Chennana et al. “Vibration signal analysis for rolling bearings faults diagnosis based on deep-shallow features fusion”. In: *Scientific Reports* 15.1 (2025), p. 9270.

[24] Xin Li et al. “Multi-kernel weighted joint domain adaptation network for cross-condition fault diagnosis of rolling bearings”. In: *Reliability Engineering & System Safety* 261 (2025), p. 111109.

[25] Jurgen OD van den Hoogen, SD Bloemheuvel, and Martin Atzmueller. “An improved wide-kernel cnn for classifying multivariate signals in fault diagnosis”. In: *2020 International Conference on Data Mining Workshops (ICDMW)*. IEEE. 2020, pp. 275–283.

[26] Dong Zhang and Taotao Zhou. “Deep convolutional neural network using transfer learning for fault diagnosis”. In: *IEEE Access* 9 (2021), pp. 43889–43897.

[27] R Balamurugan et al. “A novel prediction of remaining useful life time of rolling bearings using convolutional neural network with bidirectional long short term memory”. In: *Journal of Engineering Research* (2024).

[28] Yoo Yubin, Jo Hangyeol, and Ban SangWoo. “Lite and Efficient Deep Learning Model for Bearing Fault Diagnosis Using the CWRU Dataset [J]”. In: *Sensors* 23.6 (2023), pp. 1–2.

[29] Miguel Alonso-González et al. “Bearing fault diagnosis with envelope analysis and machine learning approaches using CWRU dataset”. In: *IEEE Access* 11 (2023), pp. 57796–57805.

[30] Diwang Ruan et al. “CNN parameter design based on fault signal analysis and its application in bearing fault diagnosis”. In: *Advanced Engineering Informatics* 55 (2023), p. 101877.

[31] Krish Kumar Raj et al. “Enhanced fault detection in bearings using machine learning and raw accelerometer data: A case study using the Case Western Reserve University dataset”. In: *Information* 15.5 (2024), p. 259.

[32] Rodrigo Kobashikawa Rosa, Danilo Braga, and Danilo Silva. “Benchmarking deep learning models for bearing fault diagnosis using the cwru dataset: A multi-label approach”. In: *arXiv preprint arXiv:2407.14625* (2024).

[33] Jun-gyo Jang et al. “Noise reduction in cwru data using dae and classification with ViT”. In: *Applied Sciences* 14.24 (2024), p. 11771.

[34] Dominik Łuczak. “Data-driven machine Fault Diagnosis of Multisensor vibration data using synchrosqueezed transform and time-frequency image recognition with convolutional neural network”. In: *Electronics* 13.12 (2024), p. 2411.

[35] Arta Mohammad-Alikhani et al. “Fault diagnosis of electric motors by a channel-wise regulated CNN and differential of STFT”. In: *IEEE Transactions on Industry Applications* (2025).

[36] Huige Huang et al. “Fault diagnosis model based on STFT and multi-scale convolution channel attention”. In: *2025 IEEE 14th Data Driven Control and Learning Systems (DDCLS)*. IEEE. 2025, pp. 1576–1581.

[37] Seungmin Han and Jongpil Jeong. “An weighted CNN ensemble model with small amount of data for bearing fault diagnosis”. In: *Procedia Computer Science* 175 (2020), pp. 88–95.

[38] Shahabodin Afrasiabi et al. “Real-time bearing fault diagnosis of induction motors with accelerated deep learning approach”. In: *2019 10th international power electronics, drive systems and technologies conference (PEDSTC)*. IEEE. 2019, pp. 155–159.

[39] Sang Kyung Lee et al. “Hybrid approach of xgboost and rule-based model for fault detection and severity estimation in spacecraft propulsion system”. In: *Phm society asia-pacific conference*. Vol. 4. 1. 2023.

[40] François Chollet. “Xception: Deep learning with depthwise separable convolutions”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1251–1258.

[41] Marina Sokolova and Guy Lapalme. “A systematic analysis of performance measures for classification tasks”. In: *Information processing & management* 45.4 (2009), pp. 427–437.

[42] T Narendiranath Babu et al. “Rolling element bearing fault diagnosis using discrete mayer wavelet and fault classification using machine learning algorithms”. In: *Journal of Vibration Engineering & Technologies* 13.1 (2025), p. 87.

[43] Huijuan Guo et al. “Fault Diagnosis Method of Rolling Bearing Based on 1D Multi-Channel Improved Convolutional Neural Network in Noisy Environment”. In: *Sensors* 25.7 (2025), p. 2286.

[44] Junning Li, Wenguang Luo, and Mengsha Bai. “Review of research on signal decomposition and fault diagnosis of rolling bearing based on vibration signal”. In: *Measurement Science and Technology* 35.9 (2024), p. 092001.

[45] Xiao Zhang, Boyang Zhao, and Yun Lin. “Machine learning based bearing fault diagnosis using the case western reserve university data: A review”. In: *Ieee Access* 9 (2021), pp. 155598–155608.

[46] Haihong Tang et al. “Feature extraction of multi-sensors for early bearing fault diagnosis using deep learning based on minimum unscented kalman filter”. In: *Engineering Applications of Artificial Intelligence* 127 (2024), p. 107138.

[47] Bodi Cui, Yang Weng, and Ning Zhang. “A feature extraction and machine learning framework for bearing fault diagnosis”. In: *Renewable Energy* 191 (2022), pp. 987–997.

[48] Jun Li, Yongbao Liu, and Qijie Li. “Intelligent fault diagnosis of rolling bearings under imbalanced data conditions using attention-based deep learning method”. In: *Measurement* 189 (2022), p. 110500.

[49] Guoyu Li and Xuanhao Cui. “Mechanical and structural optimisation of transverse and radial load bearing characteristics of reed straw based on finite elements”. In: *Latin American Journal of Solids and Structures* 22.4 (2025), e8385.

[50] Case School of Engineering. *CWRU bearing test data (Case Western Reserve University)*. Case Western Reserve University Bearing Data Center [Online]. Accessed: 2025-07-28. n.d. URL: https://engineering.case.edu/bearingdatacenter/download-data-file.

[51] Mohammed M Farag. “Towards a Standard Benchmarking Framework for Domain Adaptation in Intelligent Fault Diagnosis”. In: *IEEE Access* (2025).

[52] Paulo Almeida et al. “Spectral Entropy-Enhanced Multifractal Analysis for Real Bearing Fault Progression: A Comprehensive Study Using Paderborn and Cwru Datasets”. In: *Available at SSRN 5356009* ().

[53] Zhiying Qin et al. “Fault diagnosis method for rolling bearings under a small sample dataset based on improved WGAN-GP”. In: *Measurement* (2025), p. 118456.

[54] Juying Dai et al. “Fault diagnosis of rolling bearing based on multiscale intrinsic mode function permutation entropy and a stacked sparse denoising autoencoder”. In: *Applied Sciences* 9.13 (2019), p. 2743.

[55] Yinsheng Chen et al. “A novel rolling bearing fault diagnosis and severity analysis method”. In: *Applied Sciences* 9.11 (2019), p. 2356.

[56] Tao Lu et al. “A generic intelligent bearing fault diagnosis system using convolutional neural networks with transfer learning”. In: *Ieee Access* 8 (2020), pp. 164807–164814.

[57] Yanli Yang, Peiying Fu, and Yichuan He. “Bearing fault automatic classification based on deep learning”. In: *Ieee Access* 6 (2018), pp. 71540–71554.

[58] Levent Eren, Turker Ince, and Serkan Kiranyaz. "A generic intelligent bearing fault diagnosis system using compact adaptive 1D CNN classifier". In: *Journal of signal processing systems* 91.2 (2019), pp. 179–189.

[59] Francesco Caliva et al. "A deep learning approach to anomaly detection in nuclear reactors". In: *2018 International joint conference on neural networks (IJCNN)*. IEEE. 2018, pp. 1–8.

[60] Dhiraj Neupane and Jongwon Seok. "Bearing fault detection and diagnosis using case western reserve university dataset with deep learning approaches: A review". In: *IEEE Access* 8 (2020), pp. 93155–93178.

[61] Zewen Li et al. "A survey of convolutional neural networks: analysis, applications, and prospects". In: *IEEE transactions on neural networks and learning systems* 33.12 (2021), pp. 6999–7019.

[62] Patrik Reizinger et al. "Cross-entropy is all you need to invert the data generating process". In: *arXiv preprint arXiv:2410.21869* (2024).

[63] Sofyan Pariyasto, Vicky Arfeni Warongan, and Suryani Suryani. "Optimalisasi klasifikasi citra medis menggunakan CNN dan Adam optimizer dengan paramater minimum". In: *Journal of Science and Social Research* 8.1 (2025), pp. 691–701.

[64] Haigen Hu et al. "DDConv: Dynamic Dilated Convolution". In: *IEEE Transactions on Artificial Intelligence* (2025).

[65] Feifei Pang et al. "Time-frequency analysis based on multi-resolution synchroextracting Chirplet transform in reverberant environments". In: *Applied Acoustics* 231 (2025), p. 110526.

[66] Mustafa Sameer and Bharat Gupta. "Time–frequency statistical features of delta band for detection of epileptic seizures". In: *Wireless Personal Communications* 122.1 (2022), pp. 489–499.

[67] Santosh Kumar Satapathy et al. "Machine learning with ensemble stacking model for automated sleep staging using dual-channel EEG signal". In: *Biomedical Signal Processing and Control* 69 (2021), p. 102898.

[68] Yuan Gong, Yu-An Chung, and James Glass. "Ast: Audio spectrogram transformer". In: *arXiv preprint arXiv:2104.01778* (2021).

Appendices

A List of Notations

Table A.1: List of Notations Used in This Thesis

Notation	Description
s_i	i-th vibration signal measured from the bearing test rig
δs_i	Fault-induced distortion component in the i-th signal; $\delta s_i = 0$ for healthy state
$\mathcal{S}$	Set of all collected vibration signals $\{s_1, s_2, \ldots, s_M\}$
$f_{s,i}^{\text{orig}}$	Original sampling frequency of signal s_i
f_s^{target}	Target sampling frequency after resampling (12 kHz)
$s_{i,r}[n]$	Resampled discrete-time signal corresponding to s_i (Eq. 4.2)
$\mathcal{S}_r$	Set of all uniformly resampled signals (Eq. 4.3)
N_{seg}	Segment (window) length in samples ($5\text{ s} \times f_s^{\text{target}}$)
H_{hop}	Hop size in samples ($1\text{ s} \times f_s^{\text{target}}$)
$x_{ij}[n]$	j-th segment of length N_{seg} extracted from resampled signal $s_{i,r}[n]$ (Eq. 4.5)
$s_{i,r}^{(c)}$	Resampled signal from accelerometer channel $c \in \{\text{DE}, \text{FE}, \text{BA}\}$
$x_{ij}^{(c)}$	Segment extracted from the c-th sensor channel (DE, FE, or BA)
x'_{ij}	Stacked multi-sensor segment tensor $[x_{ij}^{(\text{DE})}, x_{ij}^{(\text{FE})}, x_{ij}^{(\text{BA})}]^\top \in \mathbb{R}^{3\times N_{\text{seg}}}$ (Eq. 4.4)
$\tilde{x}'_{ij}[n]$	Augmented multi-sensor segment (Eq. 4.6)
α	Amplitude scaling factor for augmentation ($0.9 \leq \alpha \leq 1.1$)
δ	Temporal shift parameter used for time translation
$\eta[n]$	Additive white Gaussian noise term in augmentation

Table A.1 (continued): List of Notations Used in This Thesis

Notation	Description
$X_{ij}(\ell, \omega)$	Short-Time Fourier Transform of segment x_{ij} (Eq. 4.7)
X_{ij}	Magnitude (log-scaled, normalized) spectrogram of segment x_{ij} (Eqs. 4.8–4.11)
$w[\cdot]$	Analysis window (e.g., Hann) used in STFT
ℓ	Frame index corresponding to sliding window position in STFT
ω	Normalized angular frequency (radians/sample) in STFT
μ, σ	Mean and standard deviation for spectrogram normalization (Eq. 4.10)
M	Deep learning model used for classification (e.g., Xception-71)
K_{cls}	Number of bearing condition classes ($K_{\text{cls}} = 10$)
$\phi(\cdot)$	CNN feature extraction function mapping X_{ij} to learned feature maps (Eq. 4.13)
C_f	Number of learned feature channels produced by the CNN (Eq. 4.13)
H', W'	Reduced spatial dimensions of feature maps after pooling (Eq. 4.13)
F	Global feature vector after global average pooling (Eq. 4.14)
W, b	Weight matrix and bias vector of final fully connected layer (Eq. 4.15)
z_k	Logit corresponding to class k before softmax (Eq. 4.15)
$\hat{y}_k$	Predicted probability that input belongs to class k (Eq. 4.15)
$\hat{y}$	Model output vector of predicted probabilities (Eq. 1.2)
y_k	One-hot encoded ground-truth label for class k
$x^{(n)}, y^{(n)}$	n-th input sample and its ground-truth label (Eq. 5.1)
$\hat{y}^{(n)}$	Predicted class label for $x^{(n)}$
$f_{\boldsymbol{\psi}}(\cdot)$	Parametric model with learnable parameters $\boldsymbol{\psi}$
$\boldsymbol{\psi}$	Set of all trainable parameters (weights and biases) of the model
N_{tot}	Total number of samples in the supervised dataset (Eq. 5.2)
$\mathbf{1}\{y^{(n)} = k\}$	Indicator function equal to 1 if sample n belongs to class k, else 0 (Eq. 5.3)
$\mathcal{L}_{\text{CE}}$	Categorical cross-entropy loss function (Eq. 5.3)

B Experiment 1: Dilated 1D CNN on Flattened STFT Features

This appendix presents the unique implementation details of **Experiment 1**, which investigates bearing fault classification using flattened one-dimensional STFT vectors and a custom *Dilated 1D Convolutional Neural Network (1D CNN)* architecture. All preprocessing, data augmentation, and cross-validation procedures follow the same structure described in Appendix C, except for the feature representation and network design specified below.

Note. Only the modified feature-extraction and model-definition sections are shown here; all other pipeline components (e.g., training loop, evaluation, and cross-validation) are identical to those used in Appendix C.

```
# Load augmented vibration data
df_aug = pd.read_pickle('data_augmented.pkl')

# STFT parameters
fs = 12000
fi = 100
Nseg = int(fs / fi)     # 120 samples per segment
target_shape = (128, 64, 1)

# Compute STFT and flatten to 1D
def compute_stft(x, Nseg, Nfft):
    overlap_ratio = 0.9
    Nover = int(Nseg * overlap_ratio)
    Nstep = Nseg - Nover
    Nframe = (len(x) - Nseg) // Nstep
    X = np.zeros((Nframe, Nfft))
    window = hann(Nseg)
    for n in range(Nframe):
        start, end = n * Nstep, n * Nstep + Nseg
        spectrum = np.abs(fft(x[start:end] * window, Nfft))
        X[n, :] = spectrum
    return X.T

def normalize_tensor(tensor):
    tensor = np.log1p(np.abs(tensor))
    return tensor / np.max(tensor)
```

```
27
28 processed_data = []
29 for _, row in df_aug.iterrows():
30     signal, label = row['signal'], row['fault_class_id']
31     X = compute_stft(signal, Nseg, Nseg)[:128, :64]
32     normalized = normalize_tensor(X[:, :, np.newaxis])
33     flattened = normalized.flatten()            # Shape: (8192,)
34     processed_data.append({'tensor': flattened, 'label': label})
```

Listing B.1: Flattened STFT feature extraction for Experiment 1

```
1  class STFTFlattenedDataset(Dataset):
2      def __init__(self, tensors, labels):
3          self.data = tensors
4          self.labels = labels
5      def __len__(self):
6          return len(self.data)
7      def __getitem__(self, idx):
8          vector = torch.tensor(self.data[idx], dtype=torch.float32)
               .unsqueeze(0)
9          label = torch.tensor(self.labels[idx], dtype=torch.long)
10         return vector, label
11
12 class Custom1DCNN_8192(nn.Module):
13     def __init__(self, input_channels=1, input_length=8192,
           num_classes=10):
14         super().__init__()
15         # Five convolutional blocks with increasing filters and
               pooling
16         self.block1 = nn.Sequential(
17             nn.Conv1d(1, 32, 9, padding=4), nn.ReLU(),
18             nn.Conv1d(32, 32, 9, padding=4), nn.ReLU(),
19             nn.MaxPool1d(2))
20         self.block2 = nn.Sequential(
21             nn.Conv1d(32, 64, 7, padding=3), nn.ReLU(),
22             nn.Conv1d(64, 64, 7, padding=3), nn.ReLU(),
23             nn.MaxPool1d(2))
24         self.block3 = nn.Sequential(
25             nn.Conv1d(64, 128, 5, padding=2), nn.ReLU(),
26             nn.Conv1d(128, 128, 5, padding=2), nn.ReLU(),
27             nn.MaxPool1d(2))
```

```
        self.block4 = nn.Sequential(
            nn.Conv1d(128, 256, 3, padding=1), nn.ReLU(),
            nn.Conv1d(256, 256, 3, padding=1), nn.ReLU(),
            nn.MaxPool1d(2))
        self.block5 = nn.Sequential(
            nn.Conv1d(256, 512, 3, padding=1), nn.ReLU(),
            nn.Conv1d(512, 512, 3, padding=1), nn.ReLU(),
            nn.MaxPool1d(2))
        final_len = input_length // (2 ** 5)  # 8192/32=256
        self.fc = nn.Sequential(
            nn.Linear(512 * final_len, 1024), nn.ReLU(), nn.
               Dropout(0.2),
            nn.Linear(1024, 512), nn.ReLU(), nn.Dropout(0.4),
            nn.Linear(512, 256), nn.ReLU(), nn.Dropout(0.5),
            nn.Linear(256, num_classes))
    def forward(self, x):
        x = self.block1(x)
        x = self.block2(x)
        x = self.block3(x)
        x = self.block4(x)
        x = self.block5(x)
        x = x.view(x.size(0), -1)
        return self.fc(x)
```

Listing B.2: Dilated 1D CNN model architecture for flattened STFT input

The above model receives a 1D flattened STFT vector (1×8192) per sample, captures global frequency–time information through dilated 1D CNN. Training, optimization, and evaluation steps follow the same cross-validation structure used in the baseline XceptionNet-71 C.

C Experiment 2: XceptionNet-71 on Single-Channel STFT Features

This appendix provides the complete Python implementation corresponding to **Experiment 2** described in Section 5.6. In this experiment, the pretrained `XceptionNet-71` model is utilized for bearing fault classification using single-channel spectrograms derived from the Short-Time Fourier Transform (STFT).

The following code is organized into four major stages—*Data Preparation*, *Feature Extraction*, *Model Construction*, and *Training and Evaluation*—mirroring the methodological framework detailed in the main text.

```
# ================================================================
# 1. Data Preparation
# ================================================================
import os
os.environ["PYTORCH_CUDA_ALLOC_CONF"] = "expandable_segments:True"

import pandas as pd
import numpy as np
from scipy.signal.windows import hann
from numpy.fft import fft
from torchvision import transforms
import torch, torch.nn as nn, timm, gc
import matplotlib.pyplot as plt
from sklearn.model_selection import StratifiedKFold,
    train_test_split
from sklearn.metrics import accuracy_score, f1_score,
    classification_report, confusion_matrix, roc_curve, auc
from sklearn.preprocessing import label_binarize
from sklearn.utils.class_weight import compute_class_weight
from torch.utils.data import DataLoader, Dataset,
    WeightedRandomSampler
from torch.cuda.amp import autocast, GradScaler
from tqdm import tqdm
from PIL import Image
import seaborn as sns

print("CUDA available:", torch.cuda.is_available())
print("Device name:", torch.cuda.get_device_name(0) if torch.cuda.
    is_available() else "CPU only")

# STFT Parameters
fs, fi, Nfft = 12000, 100, 256
Nseg = int(fs / fi)                    # 120 samples per segment
target_shape = (128, 64, 1)            # (Freq, Time, Channels)
df_aug = pd.read_pickle('data_augmented.pkl')

```

```
# Image transform
transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.5], std=[0.5])
])

# ================================================================
# 2. Feature Extraction
# ================================================================
def compute_stft(x, Nseg, Nfft):
    overlap_ratio = 0.9
    Nover = int(Nseg * overlap_ratio)
    Nstep = Nseg - Nover
    Nframe = (len(x) - Nseg) // Nstep
    X = np.zeros((Nframe, Nfft))
    window = hann(Nseg)
    for n in range(Nframe):
        start, end = n * Nstep, n * Nstep + Nseg
        X[n, :] = np.abs(fft(x[start:end] * window, Nfft))
    return X.T

def normalize_tensor(tensor):
    tensor = np.log1p(np.abs(tensor))
    return tensor / np.max(tensor)

def fix_tensor_shape(tensor, target_shape):
    padded = np.zeros(target_shape, dtype=np.float32)
    padded[:tensor.shape[0], :tensor.shape[1], :tensor.shape[2]] =
        tensor
    return padded

class STFTSingleChannelDataset(Dataset):
    def __init__(self, tensors, labels, transform=None):
        self.data, self.labels, self.transform = tensors, labels,
            transform
    def __len__(self): return len(self.data)
    def __getitem__(self, idx):
        img = (self.data[idx].squeeze(-1) * 255).astype(np.uint8)
        image = Image.fromarray(img, mode='L')
```

```
        if self.transform: image = self.transform(image)
        return image, torch.tensor(self.labels[idx], dtype=torch.long)

# Extract STFT features
extracted_features = []
for _, row in df_aug.iterrows():
    X = compute_stft(row['signal'], Nseg, Nseg)[:128, :64]
    tensor = fix_tensor_shape(normalize_tensor(X[:, :, np.newaxis]), target_shape)
    extracted_features.append({'tensor': tensor, 'label': row['fault_class_id']})

X = [d['tensor'] for d in extracted_features]
y = [d['label'] for d in extracted_features]
X_trainval, X_test, y_trainval, y_test = train_test_split(X, y, test_size=0.2,
                                                          stratify=y, random_state=42)

# ============================================================
# 3. Model Construction
# ============================================================
def get_timm_model(model_name, num_classes=10):
    return timm.create_model(model_name, pretrained=True, in_chans=1, num_classes=num_classes)

def get_xception71_bundle(num_classes=10, lr=1e-4, weight_decay=1e-5):
    torch.cuda.empty_cache(); gc.collect()
    model = get_timm_model("xception71", num_classes)
    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
    print(f"Training on device: {device}")
    model = model.to(device)
    criterion = nn.CrossEntropyLoss()
    optimizer = torch.optim.Adam(model.parameters(), lr=lr, weight_decay=weight_decay)
```

```
    return model, device, criterion, optimizer

# ================================================================
# 4. Training and Cross-Validation
# ================================================================
def train_model(model, train_loader, val_loader, criterion, optimizer,
                device, num_epochs=10, fold=None):
    scaler, best_val_f1 = GradScaler(), 0.0
    best_model_path = f"xception_net71_best_model.pth"
    history = {'train_loss': [], 'val_loss': [], 'train_acc': [], 'val_acc': [],
               'val_f1_macro': [], 'val_classwise_f1': []}
    for epoch in range(num_epochs):
        # ---- Training ----
        model.train(); train_loss, correct, total = 0, 0, 0
        for Xb, yb in tqdm(train_loader, desc=f"[Fold {fold+1}] Epoch {epoch+1} Training", leave=False):
            Xb, yb = Xb.to(device), yb.to(device)
            optimizer.zero_grad()
            with autocast(device_type='cuda', enabled=(device.type == 'cuda')):
                outputs = model(Xb)
                loss = criterion(outputs, yb)
            scaler.scale(loss).backward(); scaler.step(optimizer); scaler.update()
            train_loss += loss.item() * Xb.size(0)
            correct += (outputs.argmax(1) == yb).sum().item(); total += Xb.size(0)
        train_loss /= total; train_acc = 100. * correct / total

        # ---- Validation ----
        model.eval(); val_loss, correct, total = 0, 0, 0
        all_preds, all_labels = [], []
        with torch.no_grad():
            for Xb, yb in tqdm(val_loader, desc=f"[Fold {fold+1}] Epoch {epoch+1} Validation", leave=False):
                Xb, yb = Xb.to(device), yb.to(device)
                with autocast(device_type='cuda', enabled=(device.type == 'cuda')):
```

```
                    outputs = model(Xb); loss = criterion(outputs, yb)
                val_loss += loss.item() * Xb.size(0)
                preds = outputs.argmax(1)
                correct += (preds == yb).sum().item(); total += Xb.size(0)
                all_preds.extend(preds.cpu().numpy()); all_labels.extend(yb.cpu().numpy())
        val_loss /= total; val_acc = 100. * correct / total
        macro_f1 = f1_score(all_labels, all_preds, average='macro', zero_division=0)
        if macro_f1 > best_val_f1:
            best_val_f1 = macro_f1; torch.save(model.state_dict(), best_model_path)
            print(f"[Fold {fold+1}] New best model saved ({macro_f1:.4f})")
        history['train_loss'].append(train_loss); history['val_loss'].append(val_loss)
        history['train_acc'].append(train_acc); history['val_acc'].append(val_acc)
        history['val_f1_macro'].append(macro_f1)
    return history, best_model_path

def cross_validate_and_train(X_trainval, y_trainval, skf, num_epochs=10, batch_size=32):
    cv_history, all_fold_hist, saved_paths = {}, [], []
    for fold, (train_idx, val_idx) in enumerate(skf.split(X_trainval, y_trainval)):
        print(f"\n--- Fold {fold + 1} ---")
        X_train = [X_trainval[i] for i in train_idx]; y_train = [y_trainval[i] for i in train_idx]
        X_val = [X_trainval[i] for i in val_idx]; y_val = [y_trainval[i] for i in val_idx]
        class_wts = compute_class_weight('balanced', classes=np.unique(y_train), y=y_train)
        w_map = {cls: w for cls, w in zip(np.unique(y_train), class_wts)}
        sample_wts = [w_map[y] for y in y_train]
        sampler = WeightedRandomSampler(sample_wts, len(sample_wts), replacement=True)
```

```
157         train_loader = DataLoader(STFTSingleChannelDataset(X_train
                , y_train, transform=transform),
158                                   batch_size=batch_size, sampler=
                                      sampler, num_workers=2)
159         val_loader = DataLoader(STFTSingleChannelDataset(X_val,
                y_val, transform=transform),
160                                 batch_size=batch_size, shuffle=
                                    False, num_workers=2)
161         model, device, crit, opt = get_xception71_bundle(
                num_classes=10)
162         hist, best_model_path = train_model(model, train_loader,
                val_loader, crit, opt, device, num_epochs, fold)
163         all_fold_hist.append(hist); saved_paths.append(
                best_model_path)
164         del model; torch.cuda.empty_cache(); gc.collect()
165     print("\nAll folds completed.")
166     return all_fold_hist, saved_paths
167
168 skf = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)
169 cv_history, saved_models = cross_validate_and_train(X_trainval,
        y_trainval, skf, num_epochs=10, batch_size=32)
170
171 # ================================================================
172 # 5. Final Evaluation and Visualization
173 # ================================================================
174 best_model_path = "xception_net71_best_model.pth"
175 device = torch.device("cuda" if torch.cuda.is_available() else "
        cpu")
176 model = get_timm_model("xception71", num_classes=10)
177 model.load_state_dict(torch.load(best_model_path, map_location=
        device))
178 model = model.to(device).eval()
179
180 test_dataset = STFTSingleChannelDataset(X_test, y_test, transform=
        transform)
181 test_loader = DataLoader(test_dataset, batch_size=64, shuffle=
        False)
182
183 # Evaluate on test data
184 all_preds, all_labels, all_probs = [], [], []
```

```
with torch.no_grad():
    for Xb, yb in test_loader:
        Xb = Xb.to(device)
        probs = torch.softmax(model(Xb), dim=1)
        preds = torch.argmax(probs, dim=1)
        all_probs.extend(probs.cpu().numpy())
        all_preds.extend(preds.cpu().numpy())
        all_labels.extend(yb.numpy())

# Compute metrics
acc = accuracy_score(all_labels, all_preds)
macro_f1 = f1_score(all_labels, all_preds, average='macro')
print(f"Accuracy: {acc:.4f}, Macro F1: {macro_f1:.4f}")
print(classification_report(all_labels, all_preds, digits=4))

# Confusion Matrix
cm = confusion_matrix(all_labels, all_preds)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted'); plt.ylabel('True'); plt.show()

# ROC Curves
y_true_bin = label_binarize(all_labels, classes=list(range(10)))
for i in range(10):
    fpr, tpr, _ = roc_curve(y_true_bin[:, i], np.array(all_probs)
        [:, i])
    plt.plot(fpr, tpr, label=f'Class {i} (AUC={auc(fpr,tpr):.4f})'
        )
plt.legend(); plt.grid(True); plt.show()
```

Listing C.1: Python implementation of Experiment 2: XceptionNet-71 on Single-Channel STFT Features

D Experiment 3: EfficientNetV2-M on Multi-Channel Delta-STFT Features

This appendix presents the implementation of **Experiment 3**, which extends the single-channel STFT framework (see Section 5.7) by constructing a *multi-channel Delta-STFT* representation that captures the temporal evolution of vibration spectra across multiple time resolutions.

A pretrained `EfficientNetV2-M` backbone from the `timm` library is fine-tuned for ten-class bearing fault classification.

The experimental pipeline follows the same data preparation, augmentation, and cross-validation framework outlined in Appendix C, with the following key modifications:

- Multi-resolution STFT computation using three window sizes (N_{seg1}, N_{seg2}, N_{seg3}).
- Construction of Delta-STFT channels from inter-resolution spectral differences (D12, D13, D23).
- Adapted tensor dimensionality of $128 \times 64 \times 3$, analogous to RGB channels for CNN input.
- Integration of `EfficientNetV2-M` with a customized 10-class classification head.

Note. Only the modified feature extraction, dataset definition, and model-construction code blocks are included here. The training and evaluation follow the same structure as in Appendix C.

```
import pandas as pd
import numpy as np
from scipy.signal.windows import hann
from numpy.fft import fft

# Load augmented vibration data
df_aug = pd.read_pickle('data_augmented.pkl')

# Multi-resolution STFT parameters
```

```
fs = 12000
fi = 100
Nseg1 = int(fs / fi)
Nseg2 = Nseg1 // 2
Nseg3 = Nseg1 // 4
Nfft = 256
target_shape = (128, 64, 3)  # Three Delta-STFT channels

def compute_stft(x, Nseg, Nfft):
    """Compute STFT magnitude for a signal with given segment
       length."""
    overlap_ratio = 0.9
    Nover = int(Nseg * overlap_ratio)
    Nstep = Nseg - Nover
    Nframe = (len(x) - Nseg) // Nstep
    X = np.zeros((Nframe, Nfft))
    window = hann(Nseg)
    for n in range(Nframe):
        seg = x[n*Nstep:n*Nstep+Nseg] * window
        X[n, :] = np.abs(fft(seg, Nfft))
    return X.T

def normalize_tensor(tensor):
    """Apply log scaling and normalize to [0,1]."""
    tensor = np.log1p(np.abs(tensor))
    return tensor / np.max(tensor)

processed_data = []
for _, row in df_aug.iterrows():
    signal, label = row['signal'], row['fault_class_id']
    # STFTs at three resolutions
    X1 = compute_stft(signal, Nseg1, Nseg1)
    X2 = compute_stft(signal, Nseg2, Nseg2)
    X3 = compute_stft(signal, Nseg3, Nseg3)

    # Align shapes
    min_shape = (min(X1.shape[0], X2.shape[0], X3.shape[0]),
                 min(X1.shape[1], X2.shape[1], X3.shape[1]))
    X1, X2, X3 = X1[:min_shape[0], :min_shape[1]], \
                 X2[:min_shape[0], :min_shape[1]], \
```

```
                X3[:min_shape[0], :min_shape[1]]

    # Compute Delta-STFT difference channels
    D12, D13, D23 = X1 - X2, X1 - X3, X2 - X3

    # Stack into 3-channel tensor
    raw_tensor = np.stack([D12[:128], D13[:128], D23[:128]], axis
       =-1)
    normalized = normalize_tensor(raw_tensor)
    processed_data.append({'tensor': normalized, 'label': label})
```

Listing D.1: Computation of multi-channel Delta-STFT features for Experiment 3

```
from PIL import Image
from torch.utils.data import Dataset, DataLoader
from torchvision import transforms
import torch

class DeltaSTFTDataset(Dataset):
    """Dataset for three-channel Delta-STFT spectrograms."""
    def __init__(self, tensors, labels, transform=None):
        self.data = tensors
        self.labels = labels
        self.transform = transform
    def __len__(self):
        return len(self.data)
    def __getitem__(self, idx):
        image = Image.fromarray((self.data[idx] * 255).astype(np.
           uint8))
        if self.transform:
            image = self.transform(image)
        return image, torch.tensor(self.labels[idx], dtype=torch.
           long)

# Image preprocessing compatible with EfficientNetV2-M
transform = transforms.Compose([
    transforms.Resize((299, 299)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406],
                         std=[0.229, 0.224, 0.225])
])
```

Listing D.2: Delta-STFT dataset and preprocessing pipeline

```
import timm
import torch.nn as nn
import torch

def get_timm_model(model_name, num_classes):
    """Load pretrained CNN and replace final classification head."""
    model = timm.create_model(model_name, pretrained=True)
    if hasattr(model, 'classifier'):
        in_features = model.classifier[-1].in_features
        model.classifier[-1] = nn.Linear(in_features, num_classes)
    return model

def get_effnetv2_m_bundle(num_classes=10, lr=1e-4, weight_decay=1e-5):
    """Bundle model, device, loss, and optimizer for training."""
    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
    model = get_timm_model("tf_efficientnetv2_m.in21k_ft_in1k", num_classes).to(device)
    criterion = nn.CrossEntropyLoss()
    optimizer = torch.optim.Adam(model.parameters(), lr=lr, weight_decay=weight_decay)
    return model, device, criterion, optimizer
```

Listing D.3: Initialization and configuration of EfficientNetV2-M for Delta-STFT input

```
# 10-Fold Cross-Validation Training
cv_history, all_fold_histories, saved_model_paths, fold_preds = cross_validate_and_train(
    X_trainval,
    y_trainval,
    skf,
    num_epochs=10,
    batch_size=32
)

```

```
# After each fold
torch.cuda.empty_cache()
gc.collect()
```

Listing D.4: Simplified 10-fold cross-validation training loop

The model operates on Delta-STFT tensors of size $3 \times 299 \times 299$, where each channel encodes inter-resolution spectral differences (D12, D13, D23). Using pretrained `EfficientNetV2-M` weights enables faster convergence and improved generalization to unseen bearing conditions.

E Live Demonstration

This appendix provides the complete Python script used for the live demonstration of the bearing fault classification pipeline (see Section 4.3.1). The demonstration replicates the preprocessing, feature extraction, and classification steps, but operates on a single vibration file to illustrate end-to-end inference. The trained XceptionNet-71 model (loaded via the `timm` library) is used for classification.

```
import scipy.io
import numpy as np
from scipy.signal import resample, hann
from scipy.fft import fft
import torch
import torch.nn as nn
import timm
from torchvision import transforms

# -------------------------
# 1. Load .mat data file
# -------------------------
def load_data_mat_file(file_path):
    mat_data = scipy.io.loadmat(file_path)
    file_num = file_path.split('/')[-1].split('.')[0].split('_')
        [0]
    if '48k' in file_path or 'normal' in file_path:
        original_fs = 48000
    else:
```

```
        original_fs = 12000
    return mat_data, original_fs, file_num

# ------------------------
# 2. Preprocessing
# ------------------------
def segmentation(window_sec, hop_sec, fs, signal):
    window_size = window_sec * fs
    hop_size = hop_sec * fs
    segments = []
    for start in range(0, len(signal) - window_size + 1, hop_size):
        segment = signal[start:start + window_size]
        segments.append(segment)
    return np.array(segments)

def compute_stft(x, Nseg, Nfft):
    overlap_ratio = 0.9
    Nover = int(Nseg * overlap_ratio)
    Nstep = Nseg - Nover
    Nframe = (len(x) - Nseg) // Nstep
    X = np.zeros((Nframe, Nfft))
    window = hann(Nseg)
    for n in range(Nframe):
        segment = x[n*Nstep:n*Nstep+Nseg] * window
        spectrum = np.abs(fft(segment, Nfft))
        X[n, :] = spectrum
    return X.T

# ------------------------
# 3. Load trained XceptionNet-71
# ------------------------
def get_xception71(num_classes, model_path):
    model = timm.create_model("xception71", pretrained=False,
        num_classes=num_classes)
    checkpoint = torch.load(model_path, map_location="cpu")
    model.load_state_dict(checkpoint)
    model.eval()
    return model

```

```
# ------------------------
# 4. Inference
# ------------------------
def classify(model, features):
    transform = transforms.Compose([
        transforms.ToTensor(),
        transforms.Resize((224, 224)),   # required input size for XceptionNet-71
        transforms.Normalize(mean=[0.485, 0.456, 0.406],
                             std=[0.229, 0.224, 0.225])
    ])
    with torch.no_grad():
        for segment in features:
            input_tensor = transform(segment).unsqueeze(0)
            outputs = model(input_tensor)
            _, pred = torch.max(outputs, 1)
            print("Predicted class:", pred.item())

# ------------------------
# Main Execution
# ------------------------
if __name__ == "__main__":
    file_path = "path/to/sample.mat"
    model_path = "path/to/xception71_trained.pth"
    model = get_xception71(num_classes=10, model_path=model_path)
    mat_data, original_fs, file_num = load_data_mat_file(file_path)
    # Preprocessing + STFT + classification steps would be added here
```

Listing E.1: Python implementation of the live demonstration pipeline using XceptionNet-71