machines

MDPI

Article

Data-Driven Techniques for Mechanical Fault Classification

Marc-André Huneault [1], Markus Timusk [1] and Chris Mechefske [2,*]

[1] School of Engineering, Laurentian University, Sudbury, ON P3E 2C6, Canada; mhuneault@laurentian.ca (M.-A.H.); mtimusk@laurentian.ca (M.T.)
[2] Department of Mechanical and Materials Engineering, Smith Engineering, Queen's University, Kingston, ON K7L 3N6, Canada
* Correspondence: chris.mechefske@queensu.ca

Abstract

Data-driven techniques for machine condition monitoring are typically first evaluated on relatively simple experimental datasets, where they often achieve near-perfect classification accuracies. Such results do not establish real-world applicability or whether the associated computational demands are justified. This paper introduces an experimental dataset designed to better approximate selected real-world operating conditions. The dataset is used to evaluate both traditional and novel techniques for machinery fault classification and to assess their suitability for resource-constrained computing environments, such as edge-computing applications on mobile or remote equipment. Four distinct data-driven methods were evaluated using several metrics in addition to classification accuracy. The traditional machine learning techniques struggled to identify fault conditions under unseen speed profiles, whereas the deep learning models performed substantially better. Computational metrics showed that some methods required considerably more memory and processing resources than others. One deep learning model had memory and computational requirements that appear theoretically compatible with real-time implementation on platforms such as the Jetson Nano or Raspberry Pi 5. These results highlight both the potential of continued deep learning development for machine condition monitoring and the limitations of traditional approaches under more variable operating conditions.

Keywords: bi-directional long short-term memory; deep learning; machine learning; multi-scale convolutional neural network; particle swarm optimization; recurrent neural network; support vector machine

Check for updates

Academic Editors: Ahmed Abu-Siada and Maciej Sułowicz

Received: 17 June 2026
Revised: 9 October 2026
Accepted: 21 October 2026
Published: 23 October 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

Downtime and maintenance costs significantly impact industrial, mining, and manufacturing operations. Failures, especially when they occur unexpectedly, can lead to bottlenecks and supply chain disruptions, increase operational and capital expenses, and expose workers to unsafe working conditions. To improve maintenance strategies, real-time monitoring and analysis techniques have been and continue to be developed to detect anomalous behaviour of components during the operation of equipment. Simple applications include using measured vibration amplitude thresholds to deduce the presence of a faulty component. More complex implementations include the use of data-driven machine learning (ML) models to estimate the health status of more complex machinery from those same vibration signals.

Early data-driven fault detection models were typically developed using a few manually selected features (commonly statistical) computed from a labelled dataset representing

different faults embedded into a machine. These models were then used to train a classifier, such as a shallow multi-layered perceptron (MLP) or support vector machine (SVM), with various algorithms and data pre-processing/feature extraction methods to classify unseen or new signals [1]. A shortfall commonly found with these early fault detection models is their inability to handle data complexity due to the dynamic nature of certain machines and the effects of noise in the data. Ongoing development, coupled with enhanced data processing and storage capabilities, has led to the emergence of novel methods to address these limitations. These methods, commonly referred to as deep learning (DL) models, feature more extensive neural network structures with multiple interconnected layers, enabling them to better characterize faulty behaviours. Their increased depth enables them to conduct their own feature extraction, reducing the need for prior knowledge regarding the monitored machine's behaviour [2].

Newly developed models are consistently achieving high prediction accuracies on common fault classification datasets like the Case-Western University Bearing Dataset (CWRU) [3–5]. Such datasets are typically limited in size and are collected from simple experimental setups. They usually involve time-invariant sequences and feature a limited number of fault categories across the entire dataset. It is thus difficult to ascertain how models developed on these datasets would perform on data collected from a real application, which is likely to include more fault cases, noise from various sources, and variance in signal characteristics. Additionally, since many models are reporting excellent results on the same datasets, it can become difficult to select an appropriate model for a specific application. Identifying the limits of a model can showcase where they are likely to be successful, where they can be improved, and which applications they best suit. A further issue is that many approaches struggle when applied to machinery that has few examples of failure data [6].

Significant research has been conducted over the past decade on applications of deep learning to fault detection of rotating machinery, with many limitations being the quality and completeness of the data that is used for training [6].

To strengthen the argument for improving model selection criteria, particularly given the numerous reports of near-perfect performance from models trained and tested on commonly available experimental datasets, it is essential to emphasize the computational cost of training and inferring the proposed methodologies. This information can assist in selecting an appropriate model for a specific task from both hardware cost and computational constraint standpoints.

The goal of this research is to provide an improved analysis of existing and novel fault detection techniques using an experimental dataset collected with the intention of better representing a wider range of electromechanical systems and their operating conditions. The analysis will consider both performance and computational metrics to highlight the suitability of a given technique for real-world applications by better defining their abilities and limitations and whether the computation cost is worthwhile.

2. Materials and Methods

Intelligent machinery monitoring systems have been a topic of research interest for many decades [5]. Data-driven techniques have been developed to detect, classify, and predict the behaviour of machines and their components. Commonly referred to as ML (machine learning) models, these classifiers typically learn, from manually selected features, to identify the state of a monitored system. These models have had many successful implementations in the field of machine condition monitoring, especially in cases where little data and/or limited computational resources are available for training. However, the susceptibility of manually selected signal features to noise and other interference

limits their ability to learn from data collected in unfavourable conditions. Moreover, their restricted capacity to generalize non-linear relationships imposes an upper limit on system complexity that can be effectively monitored. In brief, promising results from traditional ML methods are typically only possible when data are collected from simple experimental apparatuses (featuring few components) subjected to minimal noise and consisting of subsets of short-term, steady-state recordings across discrete changes in operating conditions or fault classes.

Data-driven condition monitoring techniques have evolved due to increasing availability of data sources, computational models, and computational resources. Combined with the availability of abundant training data and computational power, DL (deep learning) classifiers have become larger and more efficient. Moreover, the advent of widely available GPU programming frameworks and APIs has streamlined the utilization of graphical processing units (GPUs) for parallel processing during model training, resulting in significantly accelerated training rates. In many cases in the context of fault detection and classification, these models are capable of learning from signals with little or no data preprocessing and make predictions on system behaviour with high classification accuracy.

However, demonstrating high classification accuracies with a proposed methodology does not always guarantee optimal results. Many of the datasets that researchers succeed with may not demand the characteristics of novel or large DL models. Furthermore, achieving near-perfect results on a particular dataset may not necessarily reflect the technique's suitability in a real-world context, where subtle variations between machines and operational cycles could impact model predictions and performance.

2.1. Background

2.1.1. Benchmark Machine Condition Monitoring Datasets

Many datasets have been developed and published for the evaluation of machine condition monitoring methodologies. Three widely used datasets are the Case Western Reserve University (CWRU) Bearing Dataset [7], the Paderborn University Bearing Data Set [8], and the Mechanical Failure Prevention Technology (MFPT) dataset [9]. These datasets serve as benchmarks for evaluating the performance of fault detection models and have been extensively used in academic research.

The CWRU dataset has been widely adopted for developing and testing machine learning and deep learning algorithms. The dataset was collected using an experimental setup consisting of an electric motor, a torque transducer/encoder, and a dynamometer. Bearings with different fault types and severities were used to simulate various fault conditions. Vibration data was collected using accelerometers placed at different locations on the bearing housing while the drive motor was operated for short times at a constant speed and load. The experiments representing normal (fault free) conditions are around 40 s in duration while the experiments involving faulty bearings are just over 10 s long each.

2.1.2. Machine Learning Methods for Machine Fault Diagnostics

Two popular and effective ML techniques are Support Vector Machines (SVMs) and Artificial Neural Networks (ANNs). The SVM, a well-established binary classification model for the classification of non-separable datasets, is still commonly used because of its ability to effectively generalize non-linear relationships while maintaining relative computational simplicity when compared to DL models. SVMs have frequently been implemented for the task of fault detection and diagnosis for conditions where limited data is available for training [10,11].

Artificial neural networks (ANNs) have been implemented to learn a system's behaviour by using manually selected, or engineered, features derived from physical signals.

These features are used to classify specific conditions and detect anomalies, facilitating the identification of potential faults or failures in the system. This approach leverages the ability of ANNs to model complex, non-linear relationships within the data, for the classification of various operational states and fault conditions [12]. ANNs can be scaled in terms of layers and nodes to increase their resolution and to learn their own features from raw signals. However, since each node has a trainable weight and bias term associated with it for every node of the following layer, this can become computationally expensive.

2.1.3. Deep Learning Methods for Machine Fault Diagnostics

Increased computational power, coupled with access to increasingly large and diverse datasets across various fields such as natural language processing (NLP) and image processing, has led to the development of more sophisticated machine learning models. These models have in some cases diminished the need for extensive data preprocessing or expert knowledge of the target application. This evolution in deep learning (DL) has also significantly impacted the field of machine condition monitoring where autoencoders, recurrent neural networks, convolutional neural networks, and multi-scale convolutional neural networks have been utilized.

Autoencoders are a form of unsupervised learning model trained to reconstruct signals and features from their compressed form. Typically used for anomaly detection tasks in machine condition monitoring applications, autoencoders grade the likelihood of normal or anomalous states from their ability to reconstruct the observed signals or features [13,14]. Should a signal or group of features differ significantly from that which was used to build the model, it can be considered likely that it represents anomalous behaviour. However, as the complexity of the operation of a machine increases, it becomes more difficult for these models to accurately decide if the observed signals are normal or not. As a result, variations in the autoencoder have been developed to overcome these limitations, such as the convolutional autoencoder, which have increased model robustness and are therefore less sensitive to small variations in system behaviour, providing better classification performance.

Recurrent Neural Networks (RNNs) are a type of data-driven model particularly well suited for time-series or sequential data applications. These models process input as a sequence, which is highly valuable in machine condition monitoring, where RNNs and Long Short-Term Memory (LSTM) models are employed to manage time-series data and capture temporal dependencies. This ability to understand the order and timing of events is especially useful for detecting and predicting machinery faults, given the inherently sequential nature of the signals collected and the development of fault conditions over time.

Convolutional neural networks (CNNs) use trainable filters (kernels) that are passed over input data and used to extract localized features that can then be used for classification tasks. Unlike traditional neural networks, CNNs do not require full interconnection between all neurons across layers, which reduces computational complexity. Additionally, CNNs employ pooling operations to reduce the dimensionality of the data, leading to faster convergence, while dropout layers help prevent overfitting by limiting the model's reliance on specific features [15]. The combination of these factors reduces processing and memory requirements [16], while providing effective feature extraction capabilities. CNNs have emerged in the field of machine condition monitoring as a successful tool for denoising and dimensionality reduction in classification tasks [4,5,17].

Multi-scale convolutional neural networks (MSCNNs) have been developed to enhance feature extraction by combining convolutional layers with different kernel sizes [18]. This approach allows features at various scales to be captured, improving model performance in vibration signal analysis. A significant benefit of MSCNNs is their ability to process raw data, such as vibration signals, without the need for extensive feature engi-

neering. However, a limitation is that capturing the required amount of training data in an industrial environment can be challenging, potentially hindering the practicality of their application.

As an input is passed through a convolutional neural network, the "observed" feature, or "perspective" on the original input, is scaled in size after every layer. This same concept can be applied to vibration signals in machine operations where under varying working conditions, vibration patterns can be masked by the changing global operating frequencies of the machine [19–23].

There have been many successful applications of ML and DL implementations in machinery condition monitoring. Rarely reported in the results of these studies are the computational requirements needed to train and infer them, and whether that is practical for the task they are being evaluated with or how that translates to practical use in real-world situations. Issues such as dataset simplicity, limited variability, and potential overfitting challenge the generalizability of many models. More extensive testing of both ML and DL applications on complex, real-world-like datasets, incorporating an analysis of the computational resources required for inference and training, should take place to better highlight their suitability for practical applications.

2.2. Laurentian University Dataset

A new dataset was collected and used for training and testing different condition monitoring techniques applicable to various types of rotating equipment. The dataset includes multiple sets of time-variant operating conditions (in order to address some of the shortcomings in other datasets identified above), a large variety of fault types, a suitable sampling frequency for the different sensors, and an improved overall quantity of data.

All experiments featured in this dataset were conducted with a SpectraQuest Mechanical Fault Simulator apparatus and componentry kit [24]. The simulator features a 1HP electric motor powering a belt-driven gearbox connected to a permanent magnet brake. The apparatus was fitted with multiple accelerometers, current sensors on the motor windings, and a tachometer to measure motor speed (Figure 1).

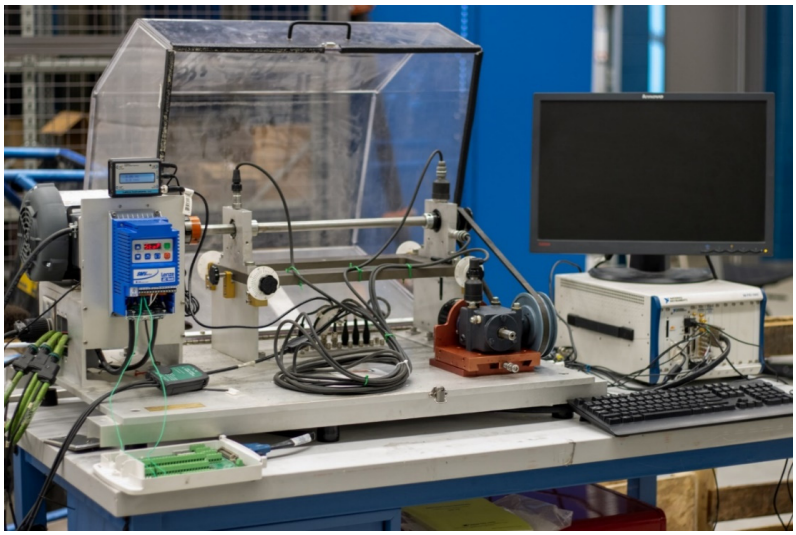


Figure 1. Instrumented mechanical fault simulator.

All signals were sampled at 10 kHz with healthy experiments (experiments with no faulty components) lasting 30 min and faulty experiments (experiments with at least one faulty component) lasting 20 min. Each experiment was conducted twice: once with a square wave speed profile (Figure 2) and then with a sawtooth speed profile (Figure 3). While the original experiments lasted different lengths of time (30 min for the experiments with healthy components, and 20 min each for the experiments with faulty components), the data from all experiments used for training and classification was truncated to 20 min in length.

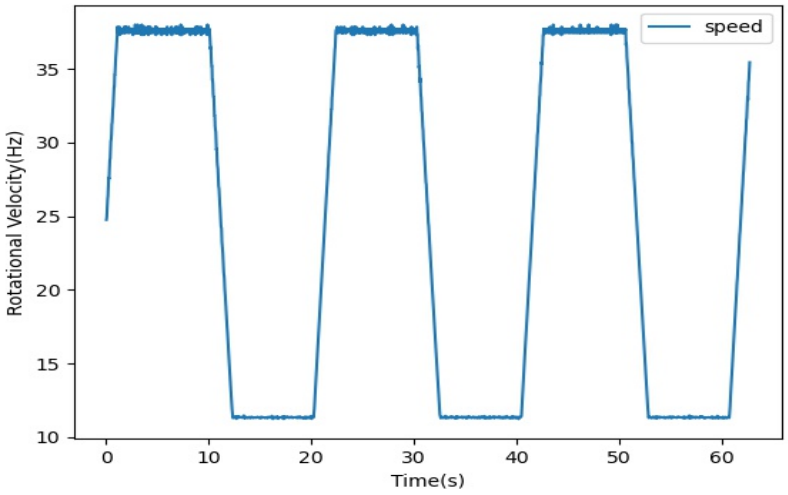


Figure 2. Sample of the square wave speed profile.

Experiments were conducted with single and multiple faults on components, simulating real-world industrial situations. Experiments were also conducted using only healthy components, once before and once after all the faulty experiments. Table 1 shows a full list of the experiments. A more complete description of the experimental apparatus, instrumentation, and data-collection procedures is available from the authors. The complete LU dataset is available from the corresponding author upon reasonable request.

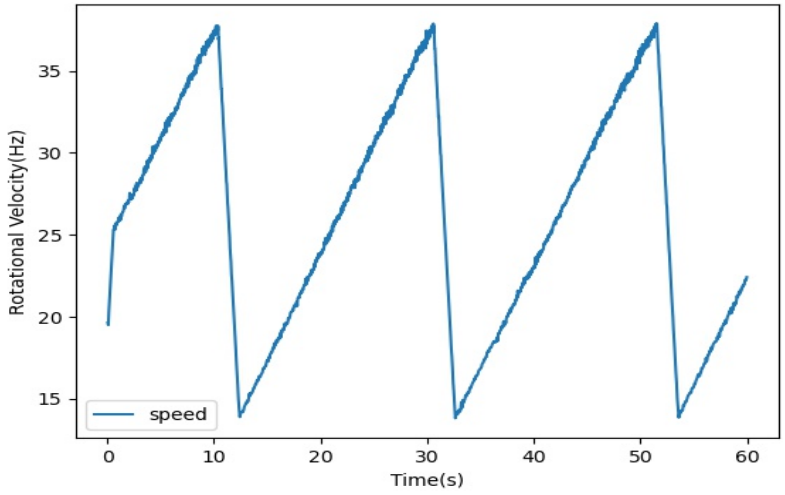


Figure 3. Sample of the sawtooth wave speed profile.

Table 1. List of experiment files-all experiments.

Fault Type	Location	Control Signal	Collection Time (s)	Exp. No.
Healthy 1	NA	Sawtooth	1200	0
Healthy 1	NA	Square	1200	1
Bearing—Inner Race	Motor End	Sawtooth	1200	2
Bearing—Inner Race	Motor End	Square	1200	3
Bearing—Inner Race	Pulley End	Sawtooth	1200	4
Bearing—Inner Race	Pulley End	Square	1200	5
Bearing—Outer Race	Motor End	Sawtooth	1200	6
Bearing—Outer Race	Motor End	Square	1200	7
Bearing—Outer Race	Pulley End	Sawtooth	1200	8
Bearing—Outer Race	Pulley End	Square	1200	9
Bearing—Ball	Motor End	Sawtooth	1200	10
Bearing—Ball	Motor End	Square	1200	11
Bearing—Ball	Pulley End	Sawtooth	1200	12
Bearing—Ball	Pulley End	Square	1200	13
Bearing—Combination	Motor End	Sawtooth	1200	14
Bearing—Combination	Motor End	Square	1200	15
Bearing—Combination	Pulley End	Sawtooth	1200	16
Bearing—Combination	Pulley End	Square	1200	17
Shaft—Cracked	NA	Sawtooth	1200	18
Shaft—Cracked	NA	Square	1200	19
Shaft—Centrally Bent	NA	Sawtooth	1200	20
Shaft—Centrally Bent	NA	Square	1200	21
Shaft—Coupling End Bent	NA	Sawtooth	1200	22
Shaft—Coupling End Bent	NA	Square	1200	23
Shaft—Hub Repair	NA	Sawtooth	1200	24
Shaft—Hub Repair	NA	Square	1200	25
Shaft Imbalance 1	NA	Sawtooth	1200	26
Shaft Imbalance 1	NA	Square	1200	27
Shaft Imbalance 1 + Bearing Inner Race	Motor End	Sawtooth	1200	28
Shaft Imbalance 1 + Bearing Inner Race	Motor End	Square	1200	29
Shaft Imbalance 1 + Bearing Inner Race	Pulley End	Sawtooth	1200	30
Shaft Imbalance 1 + Bearing Inner Race	Pulley End	Square	1200	31
Shaft Imbalance 2	NA	Sawtooth	1200	32
Shaft Imbalance 2	NA	Square	1200	33

Table 1. *Cont.*

Fault Type	Location	Control Signal	Collection Time (s)	Exp. No.
Shaft Imbalance 2 + Bearing Outer Race	Motor End	Sawtooth	1200	34
Shaft Imbalance 2 + Bearing Outer Race	Motor End	Square	1200	35
Shaft Imbalance 2 + Bearing Outer Race	Pulley End	Sawtooth	1200	36
Shaft Imbalance 2 + Bearing Outer Race	Pulley End	Square	1200	37
Shaft Misalignment 1—Parallel	NA	Sawtooth	1200	38
Shaft Misalignment 1—Parallel	NA	Square	1200	39
Shaft Misalign 1—Parallel + Bearing Inner Race	Motor End	Sawtooth	1200	40
Shaft Misalign 1—Parallel + Bearing Inner Race	Motor End	Square	1200	41
Shaft Misalign 1—Parallel + Bearing Inner Race	Pulley End	Sawtooth	1200	42
Shaft Misalign 1—Parallel + Bearing Inner Race	Pulley End	Square	1200	43
Shaft Misalignment 2—Angular	NA	Sawtooth	1200	44
Shaft Misalignment 2—Angular	NA	Square	1200	45
Shaft Misalign 2—Angular +Bearing Outer Race	Motor End	Sawtooth	1200	46
Shaft Misalign 2—Angular +Bearing Outer Race	Motor End	Square	1200	47
Shaft Misalign 2—Angular +Bearing Outer Race	Pulley End	Sawtooth	1200	48
Shaft Misalign 2—Angular +Bearing Outer Race	Pulley End	Square	1200	49
Gear—Chipped Tooth	NA	Sawtooth	1200	50
Gear—Chipped Tooth	NA	Square	1200	51
Gear—Missing Tooth	NA	Sawtooth	1200	52
Gear—Missing Tooth	NA	Square	1200	53
Healthy 2	NA	Sawtooth	1200	54
Healthy 2	NA	Square	1200	55

NA = Not Applicable.

3. Results

3.1. Fault Classification

This section evaluates and compares the performance of an RFC, SVM, CNN, and a Multi-Scale CNN when used to distinguish fault types as would be required in a typical real-world machine condition monitoring application. Each model was initially trained and evaluated on the CWRU dataset using a sample-level training/validation split to obtain benchmark metrics. Using the Laurentian University (LU) dataset, each model was trained on a square wave speed profile and tested on a sawtooth wave speed profile. Multiple instances of each model were evaluated with varying dataset characteristics and

preprocessing methods. The faults simulated are listed in Table 2. Faults that are typical in most industrial settings are represented individually, as well as in different combinations with one another, as is also common in industrial settings. (See Appendixes A–F for computational details.)

The LU dataset was reconfigured in multiple ways to evaluate model performance. The shorter duration used for the SVM implementation was necessary because the feature-selection and SVC optimization procedures become computationally expensive as the number of training samples increases. The SVM datasets were therefore limited to two minutes per experiment to maintain a computationally practical optimization process, whereas the DL models were trained using ten minutes per experiment. Consequently, comparisons between the ML and DL methods should be interpreted as comparisons of the implemented approaches under their respective computational constraints rather than as a controlled comparison using identical training-data volumes. The following datasets were generated for each control signal from two minutes of each fault class and healthy class (four minutes total of healthy data) for the feature datasets of the SVM implementation:

- Sample size: 4000 standardized acceleration data points, Fault Classes: Grouped;
- Sample size: 4000 raw acceleration data points, Fault Classes: Grouped.

The following were generated for each control signal from ten minutes of each fault class and healthy class (twenty minutes total of healthy data) to train and test the DL implementations:

- Sample size: 4000 standardized acceleration data points, Fault Classes: Grouped;
- Sample size: 4000 raw acceleration data points, Fault Classes: Grouped;
- Sample size: 1000 standardized acceleration data points, Fault Classes: Grouped;
- Sample size: 1000 raw acceleration data points, Fault Classes: Grouped;
- Sample size: 4000 standardized acceleration data points, Fault Classes: Fully Separated;
- Sample size: 4000 raw acceleration data points, Fault Classes: Fully Separated;
- Sample size: 1000 standardized acceleration data points, Fault Classes: Fully Separated;
- Sample size: 1000 raw acceleration data points, Fault Classes: Fully Separated.

The Grouped class configuration considers each fault for a given component as the same (all bearing faults for a given location have the same label). The Fully Separated class configuration separates each fault type and fault location as their own class. The class configurations are shown in Table 2. When generating standardized validation or test sets, the mean and standard deviation parameters used in the formulation were taken from the original training set. The class configuration for the CWRU dataset is shown in Table 3.

Every experiment featured in the CWRU dataset was compiled into binary files with sample sizes of 1000 and 4000 acceleration data points. As was done with the LU dataset, the list of file paths to each binary file was shuffled and split for training and validating. Because this split was performed at the sample level rather than at the physical-recording level, samples originating from the same CWRU recording may be present in both the training and validation sets. The CWRU results should therefore be interpreted as validation-based benchmark results rather than as performance on an independent recording. The entire dataset was reprocessed for both the SVM (only sample sizes of 4000 used for SVM) and DL implementation as follows:

- Sample size: 4000 standardized acceleration data points;
- Sample size: 4000 raw acceleration data points;
- Sample size: 1000 standardized acceleration data points;
- Sample size: 1000 raw acceleration data points.

Table 2. List of experiment files- fully separated class configuration.

LU Full Class Separation	LU Grouped Classes
Healthy	Healthy
Bearing—Inner Race—Motor End	Bearing –Motor End
Bearing—Inner Race—Pulley End	Bearing –Pulley End
Bearing—Outer Race—Motor End	Shaft—Cracked
Bearing—Outer Race—Pulley End	Shaft—Bent
Bearing—Ball—Motor End	Shaft—Hub Repair
Bearing—Ball—Pulley End	Shaft—Imbalance
Bearing—Combination—Motor End	Shaft—Imbalance with Bearing—Motor End
Bearing—Combination—Pulley End	Shaft—Imbalance with Bearing—Pulley End
Shaft—Cracked	Shaft—Misalignment
Shaft—Centrally Bent	Shaft—Misalignment with Bearing—Motor End
Shaft—Coupling End Bent	Shaft—Misalignment with Bearing—Pulley End
Shaft—Hub Repair	Gear
Shaft—Imbalance 1	
Shaft—Imbalance 1 with Bearing—Inner Race—Motor End	
Shaft—Imbalance 1 with Bearing—Inner Race—Pulley End	
Shaft—Imbalance 2	
Shaft—Imbalance 2 with Bearing—Outer Race—Motor End	
Shaft—Imbalance 2 with Bearing—Outer Race—Pulley End	
Shaft—Misalignment 1—Parallel	
Shaft—Misalignment 1—Parallel with Bearing—Inner Race—Motor End	
Shaft—Misalignment 1—Parallel with Bearing—Inner Race—Pulley End	
Shaft—Misalignment 2—Angular	
Shaft—Misalignment 2—Angular with Bearing—Outer Race—Motor End	
Shaft—Misalignment 2—Angular with Bearing—Outer Race—Pulley End	
Gear—Chipped Tooth	
Gear—Missing Tooth	

Table 3. CWRU dataset class configurations.

Class Label	Fault Type	Fault Location
Normal Data	NA	NA
Inner Race, Drive End	Inner Raceway Fault	Drive End of the Motor
Inner Race, Fan End	Inner Raceway Fault	Fan End of the Motor
Outer Race, Drive End	Outer Raceway Fault	Drive End of the Motor
Outer Race, Fan End	Outer Raceway Fault	Fan End of the Motor
Ball, Drive End	Rolling Element Fault	Drive End of the Motor
Ball, Fan End	Rolling Element Fault	Fan End of the Motor

NA = Not Applicable.

3.1.1. Feature Dataset for the Support Vector Machine Implementation

Multiple datasets consisting of multiple time-series statistical features extracted from each sample of acceleration data were developed for the SVM implementation. These feature datasets were built from the segmented LU datasets. The computed statistical features were Mean, Variance, Root Mean Square, Gaussian Entropy, Standard Deviation, Fisher-Pearson Skewness, Sample Excess Kurtosis, Shape Factor, Crest Factor, Impulse Factor, Peak-to-Peak magnitude, Clearance Factor, Histogram Delta, Upper Bound of Histogram, Lower Bound of Histogram [11], and Moments of Degree (5, 6, and 7) [10]. The SVM's hyperparameter selection scheme followed the methods presented in [11]. A graphical representation of this process is illustrated in Figure 4.

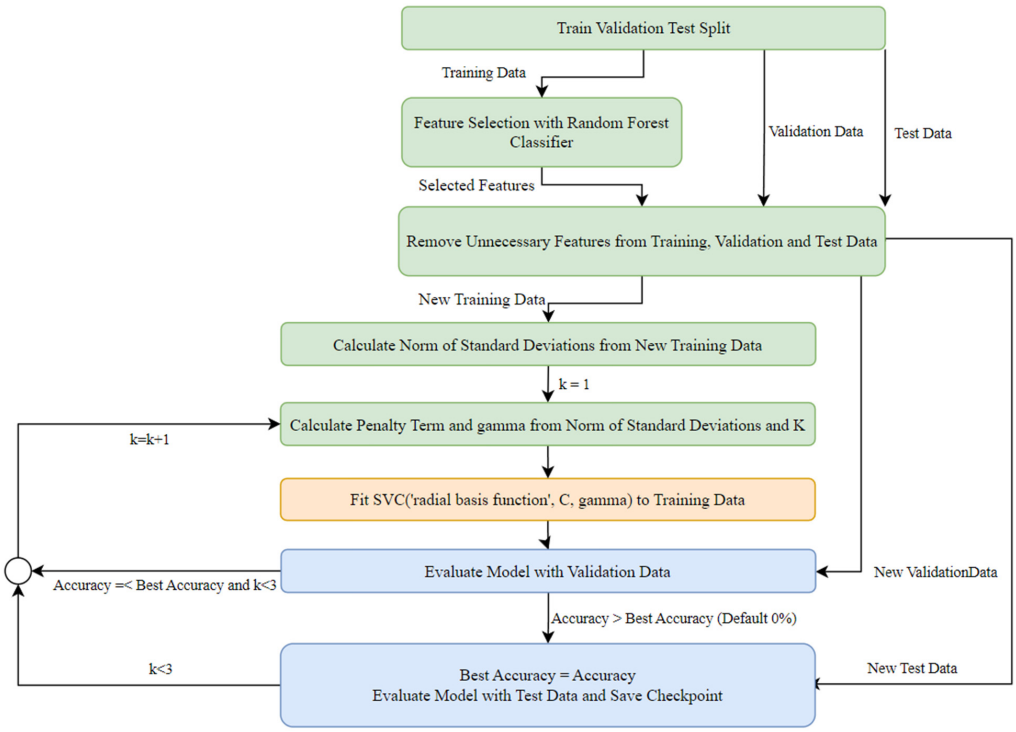


Figure 4. SVM model and feature selection scheme.

3.1.2. Convolutional Neural Network Implementation

The CNN used in this study was based upon [17], which has reportedly reached accuracy nearing 100 percent on the CWRU dataset. The model was trained in batches of 64 samples with the Adam optimization algorithm and a learning rate of 0.001. The loss criterion was cross entropy loss. The only difference between the model used in [17] and

this research was the input size, the fully connected classifier, and output size. The input size varied between 1000 or 4000 acceleration data points. The fully connected classifier input size was dependent on the input to the model and thus varied between (batch size, 216) and (batch size, 968). The output size of the classifier was dependent on the class configuration of the dataset, and thus varied between (batch size, 7) and (batch size, 27).

3.1.3. Multi-Scale CNN Implementation

The multi-scale convolutional neural-network implementation was based on the architecture used in [18]. Instead of using the time-frequency representation of the acceleration data, a one-dimensional vector of the raw or standardized acceleration data was used to capitalize on the feature extraction capabilities of the model. Additionally, the attention-based mechanism was replaced by a Convolutional Block Attention Module (CBAM) [25]. A detailed representation of this architecture can be found in Figure 5.

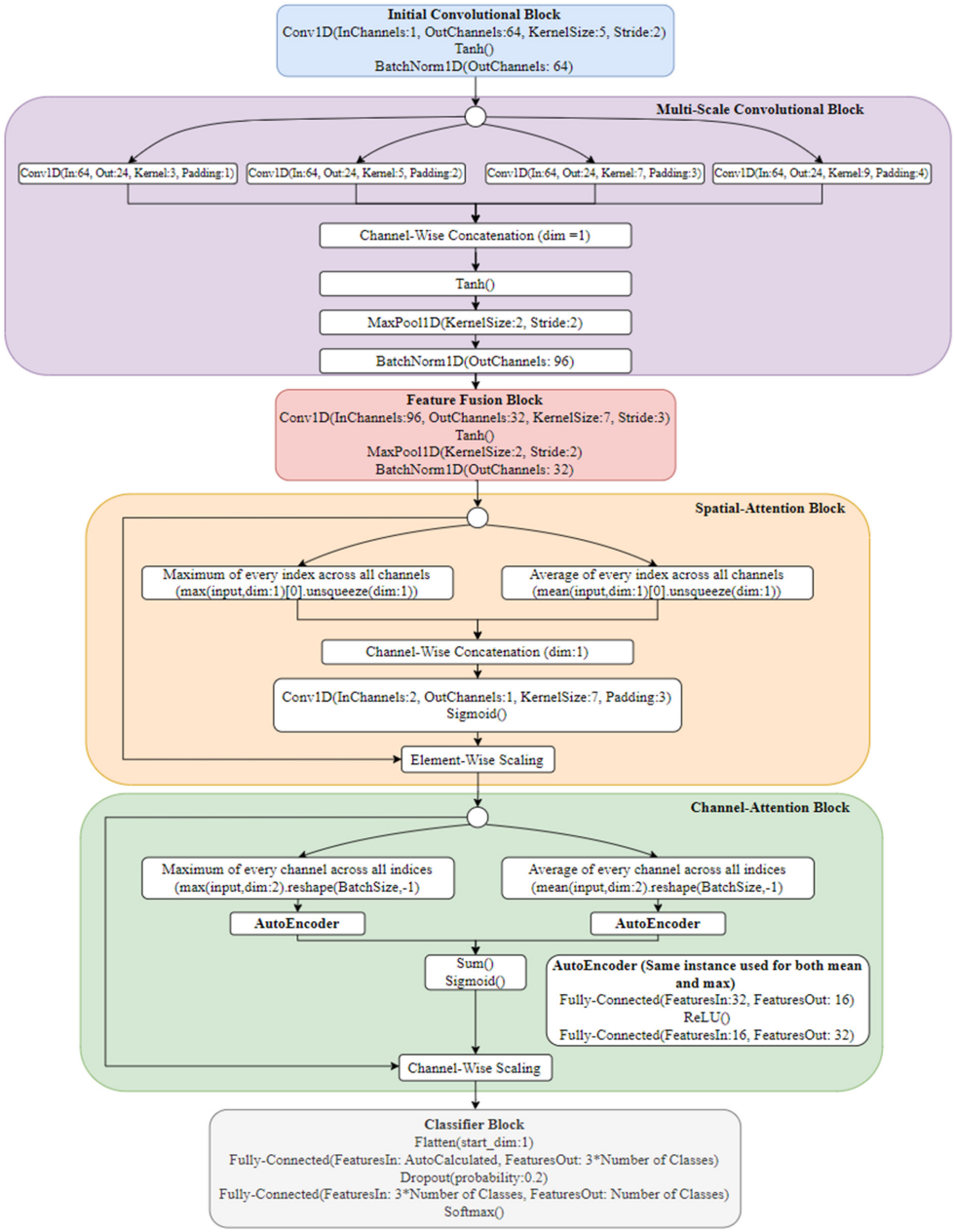


Figure 5. Multi-scale convolutional neural network architecture.

The following modifications were made to the model architecture:

- The number of output channels from each convolutional layer in the Multi-Scale Convolutional Block was reduced to 24, resulting in a concatenated output of 96 channels;
- A feature fusion block was added before the attention blocks to reduce the dimensionality of the input features and the size requirement of the attention module and classifier layer;
- Hyperbolic tangent activation functions and batch-normalization layers [26] were added after each convolutional block to stabilize the input distribution to subsequent layers;
- The number of channels of the convolution layer within the Initial Convolutional Block was increased to 64, its kernel size was reduced to five, and its stride increased to two. The reduction in kernel size was to reduce feature scaling due to the added convolutional layers in the Feature Fusion Block. A stride of two was implemented to reduce the dimensionality of the acceleration data while retaining localized information within each feature passed to the multi-scale convolutional layer;
- The Classifier Block was modified such that the number of output features of the first fully connected sub-layer was three times the number of classes of the dataset it is trained on. The dropout probability was reduced to 0.2 to compensate for the reduced number of intermediary features and the addition of batch normalization layers.

Despite the inclusion of batch-normalization layers, early training sessions with a learning rate of 0.001 demonstrated a strong bias toward certain classes in the fully separated LU dataset. The learning rate was therefore reduced to 0.0001 to allow finer optimization of the subtle differences between classes. The acceleration data samples describing the LU and CWRU datasets were shuffled and then split with 80 percent of samples used for training and 20 percent for validation. In the case where a trained model was to be tested on the sawtooth control signal features in the LU dataset, they were tested on the entire ten minutes of data for each class, or two minutes for the SVM configuration. As for the CWRU dataset, the entire dataset was used for all applications including the SVM. For CWRU, final reported classification metrics were obtained from the validation set. For LU, validation results were obtained from the square-wave data and test results from the unseen sawtooth speed profile.

The following metrics were collected from the DL model implementations:

- The number of floating-point operations (FLOP) required for model inference;
- The number of parameters in the model;
- The average time duration of a training epoch;
- The time duration for parsing through the LU test or CWRU validation datasets;
- The accuracies per epoch on the validation datasets;
- Confusion matrices of the best trained model on the test data;
- A confusion matrix of the best trained model on the validation data;
- Precision, recall and f1-score of the predictions of each class in the validation and test datasets;
- Accuracy on the validation and test datasets.

The following metrics were collected from the SVM model implementations:

- The time duration for selecting features;
- The time duration for fitting the SVC;
- The time duration for parsing through the LU test or CWRU validation datasets;
- Number of iterations run by the optimization routine to fit the model;
- The total number of support vectors in the SVC;
- The total number of parameters (coefficients and intercepts) in the SVC;
- Confusion matrix(ces) of the best trained model on the test data;

- A confusion matrix of the best trained model on the validation data;
- Precision, recall and f1-score of the predictions of each class in the validation and test datasets;
- Accuracy on the validation and test datasets.

The same metrics collected from the SVC implementation were also collected for the Random Forest Classifier and default SVC configuration mentioned. From these metrics, a comparison can be made not only of performance, but for the computational requirements needed to implement them and whether that cost is worth the added model complexity.

4. Discussion

This section focuses on whether the selected ML and DL methodologies can successfully predict faults from a dataset representing a machine with two different transient operating states and a greater quantity of fault categories than is commonly found in public datasets. Assessment of the classification results and computational requirements was used to distinguish model suitability for embedded applications (implemented on relatively low-cost, low-power edge-computing devices).

4.1. Fault Classification Performance

4.1.1. CWRU Dataset

As illustrated in Figure 6, the deep learning (DL) models were trained using 1000 standardized acceleration data points per sample, while the highest-performing traditional machine learning (ML) models utilized features extracted from samples of 4000 raw acceleration data points. It is important to note that the CWRU validation datasets contained a total of 2880 or 722 samples, depending on whether they were divided into segments of 1000 or 4000 data points per sample, respectively.

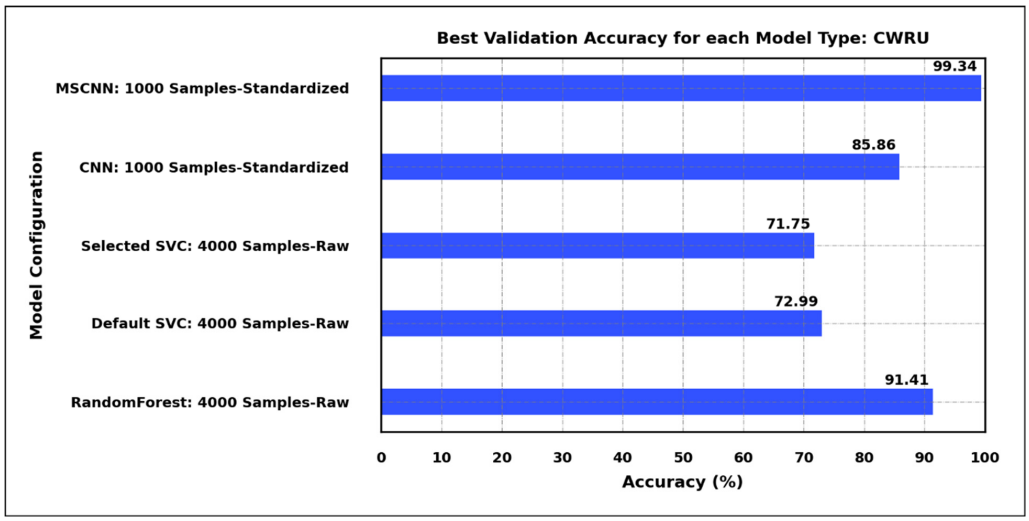


Figure 6. Best model accuracies on the validation set from the CWRU dataset.

Among the DL models, the MSCNN achieved the highest overall classification accuracy on the validation set, correctly classifying nearly 100 percent of the samples. In contrast, the RFC achieved the highest accuracy among the ML models, with a maximum accuracy of approximately 92 percent. Note that CWRU scores reflect validation and the windows used from the same recording may span training and validation. The sample-level split does not establish independence between recordings. Accordingly, these CWRU accuracies are intended primarily for benchmark comparison and may be higher than would be obtained using an independent recording-level evaluation. The confusion matrices (not included in

this paper) indicate that the CNN implementation had difficulty discerning the inner race bearing faults at the drive end location, mistaking them for outer race faults. Also, some of the samples from experiments featuring faults embedded in the ball (rolling element) of the bearing placed at the drive end location were partially misclassified as outer race faults. The discrepancy between the accuracies obtained in [17], from which the CNN was replicated, may be attributed to differences in the organization of fault types and the balance of class sizes.

Similarly, the confusion matrices indicate that the SVM models struggled to accurately predict faults in the outer race and rolling elements of the bearings. This difficulty may be due to the subtle nature of these faults and their influence on the selected features used for training the models. However, the RFC implementation did not exhibit similar challenges and only encountered difficulty in predicting rolling element faults located in the fan end bearing.

4.1.2. LU Dataset—Fully Separated Class Configuration

Only the DL implementations were evaluated with the fully separated class configuration of the LU dataset due to the computational constraints dictated by the optimization of an SVM or RFC. As shown in Figures 7 and 8, the MSCNN implementation outperforms the standard CNN implementation by roughly 20 percent in accuracy on both the validation and test sets when distinguishing the fault types across various components and their locations. The MSCNN exhibits a reduction of 9.8 percent in classification accuracy from the validation set (square wave speed profile used for training) to the test set (unseen sawtooth speed profile), whereas the CNN implementation experiences a reduction of 10.1 percent. The confusion matrices demonstrated that the multiscale CNN implementation had difficulty distinguishing the outer race faults in the bearing located at the motor end housing from those in the pulley-end bearing housing. However, this difficulty did not occur in the reverse scenario.

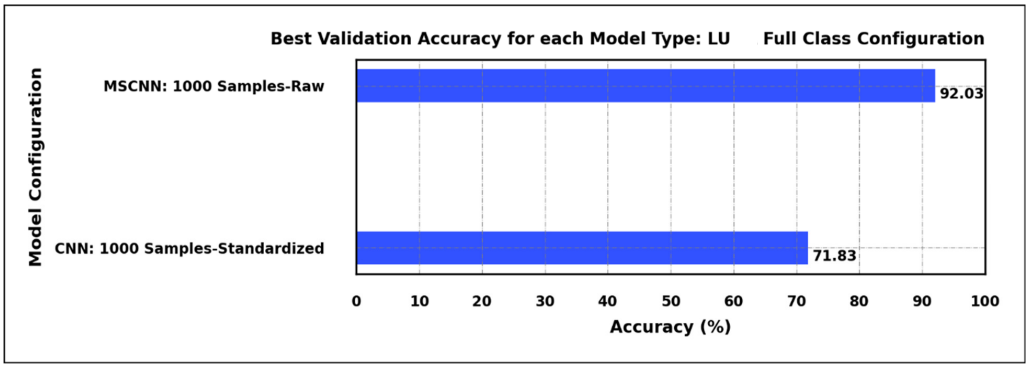


Figure 7. Best model accuracies on the validation set (square wave speed profile) from the LU dataset, fully separated class configuration.

Additionally, the model encountered challenges distinguishing between the two types of bent shaft faults and frequently mislabeled the second shaft imbalance experiment as either the first imbalance or the chipped-tooth gear fault. Furthermore, the missing tooth experiment was predominantly misclassified as healthy data, with a smaller portion being mislabeled as part of the chipped tooth experiment.

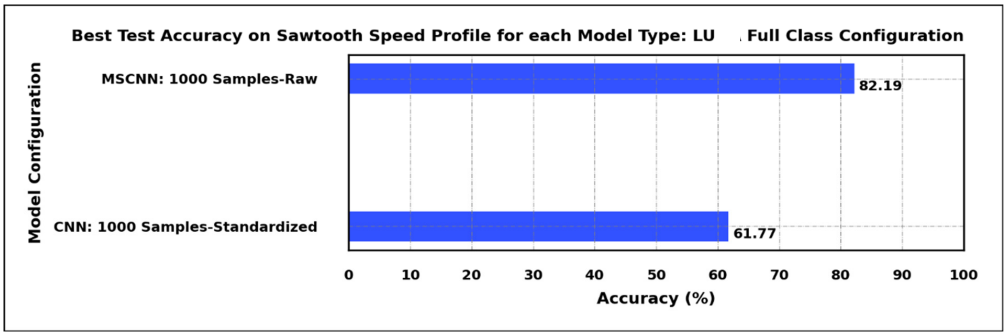


Figure 8. Best model accuracies on the test set (sawtooth wave speed profile) from the LU dataset, fully separated class configuration.

The standard CNN model failed to correctly classify any gear faults or parallel misalignment faults in the fully separated class configuration. It also struggled with identifying cracked-shaft faults and differentiating the types of bearing faults in the motor-end housing. These results suggest that the CNN faced difficulties in scaling signal sources to identify the exact location of the faults, particularly with the level of resolution needed to detect small faults, such as a minor wedge section missing from the shaft. The increased number of classes may have introduced excessive complexity, hindering the CNN's ability to detect the subtle characteristics of gear faults, which it managed to identify more effectively in the grouped class configuration.

4.1.3. LU Dataset—Grouped Class Configuration

When all fault types for a given component placed in each location (where applicable) were consolidated to the same class label (i.e., all bearing fault types at the motor end bearing housing have the same class label), the DL implementations had similar results in accuracy over the validation set and the test set. (see Figures 9 and 10).

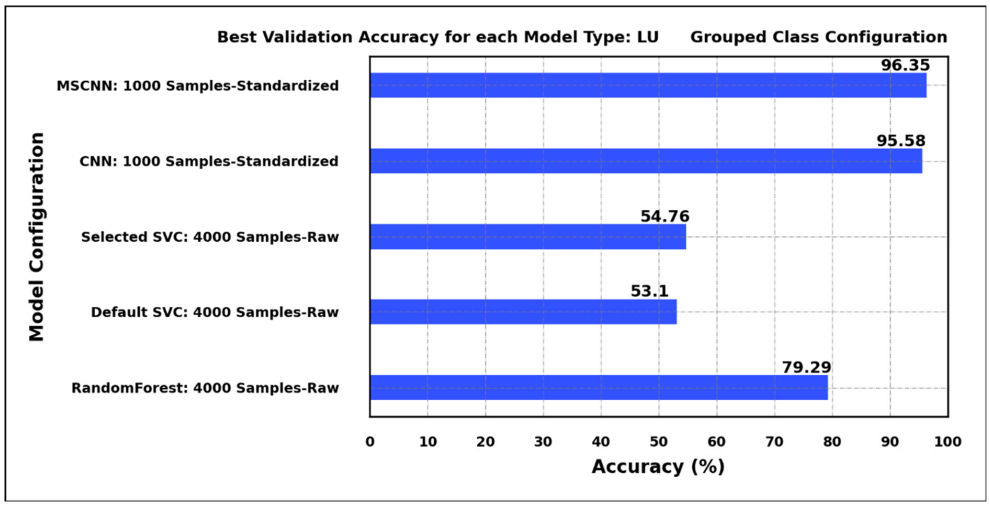


Figure 9. Best model accuracies on the validation set (square wave speed profile) from the LU dataset, grouped class configuration.

Regarding the ML implementations, the RFC seems to have greatly outperformed the SVC implementations on the validation set but had a similar classification accuracy over the test set. This could indicate the RFC overfit to the training data (square wave speed profile) and the variation in features between the training set and the test set deriving from different speed profiles affected its ability to accurately predict the fault conditions.

The confusion matrices indicated that the multiscale CNN implementation had difficulty distinguishing the gear faults from the healthy condition data. This was expected based on the separation of the gearbox from the accelerometer by the belt and pulley drive. Some of the motor end bearing faults were considered pulley end bearing faults and some of the misalignment configurations were mistaken for gear faults or imbalance faults. The same can be said for the CNN implementation.

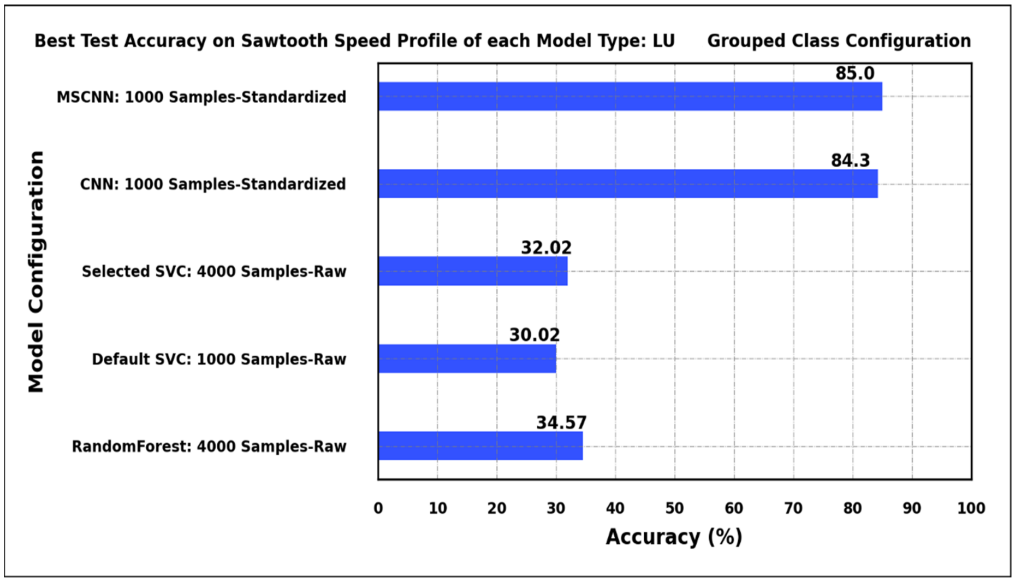


Figure 10. Best model accuracies on the test set (sawtooth wave speed profile) from the LU dataset, grouped class configuration.

4.2. Model Computational Requirements

One of the motivations of this work was to consider the computational resources required for training and inference. This assessment aims to provide further justification for the effectiveness of each method in comparison to the others. Assuming all trainable parameters are represented as 32-bit floating-point values, the number of parameters in each model was multiplied by 4 bytes and divided by 1024 to estimate the memory required to store the model parameters in kilobytes. While these values may not represent exact approximations of the actual memory allocation required to host and infer each model, they provide a reference for understanding size differences between them.

To approximate the number of parameters in the SVC models, the number of support vectors (data points in the training set with non-zero Lagrange multipliers) was multiplied by the number of selected features to account for the stored support vector coordinates. This was then summed with the number of decision intercepts to obtain the total number of parameters. For the RFC, the number of parameters was approximated as the total number of nodes in the set of trees. For the DL models, the number of parameters in each model was computed by summing the elements within the model. Since all variations in the DL models contained negligible amounts of buffers, the approximate model memory requirement was computed with the number of parameters only as mentioned above. The number of floating-point operations (FLOPs) required to infer the models was computed using a python module and is represented in millions of FLOPs (MFLOPs) in Tables 4 and 5. The number of FLOPs per second (FLOPs/s) required to infer each model in the time it takes to collect the input size at the sample frequency of collection was calculated to further represent the computational needs of each model.

Table 4. Computational requirements of the selected machine learning models.

Model Configuration	N_{SV}	N_f	Acc	Params	N_{ITER}	# Train Samples	N_{MFLOP}	Mem Req	$N_{MFLOP/s}$ Real-Time
CWRU-Default SVC-4000 Raw	1845	9	73.0	16,626	5652	2888	0.0387	64.9	0.12
CWRU-Random Forest-4000 Raw	NA	9	91.6	213,582	NA	2888	0.2144	834.3	0.64
CWRU-Selected SVC-4000 Raw	1762	9	71.8	15,879	27,391	2888	0.0370	62.0	0.11
LU-Grouped-Default SVC-1000 Raw	11,684	7	30.0	81,866	53,338	67,200	0.2220	319.8	2.22
LU-Grouped-Random Forest-4000 Raw	NA	8	34.7	600,602	NA	16,800	0.5985	2346.1	1.50
LU-Grouped-Selected SVC-4000 Raw	2808	8	32.0	22,542	134,847	16,800	0.0562	88.1	0.14

NA = Not Applicable.

Table 5. Computational requirements of the selected deep learning models.

Model Configuration	Acc	Params	# Train Epochs	# Train Samples	N_{MFLOP}	Mem Req	$N_{MFLOP/s}$ Real-Time
CWRU-CNN-1000 Standardized	85.9	80,103	21	11,572	36.03	312.9	432.36
CWRU-SpatialChannelMSCNN-1000 Standardized	99.3	87406	18	11,572	20.52	341.4	246.24
LU-Full-CNN-1000 Standardized	61.8	84443	17	720,000	36.03	329.9	360.3
LU-Full-SpatialChannelMSCNN-1000 Raw	82.2	166,326	18	720,000	20.6	649.7	206
LU-Grouped-CNN-1000 Standardized	84.3	81,405	28	336,000	36.03	318.0	360.3
LU-Grouped-SpatialChannelMSCNN-1000 Standardized	85.0	110,830	23	336,000	20.54	432.9	205.4

The columns in Tables 4 and 5 are defined as follows:

- N_{sv}: Number of support vectors;
- N_f: Number of selected features;
- *Acc*: Accuracy in percentage on the evaluation set (validation set for CWRU; test set for LU)
- *Params*: Number of parameters in the model;
- N_{ITER}: Number of optimization iterations of the support vector machine;
- *#Train Epochs*: Number of training set epochs required for the DL models;
- *#Train Samples*: Number of samples in the training set of corresponding size;
- N_{MFLOP}: Number of millions of FLOP required to infer a model;

- ***Mem Req***: Approximate amount of memory required to host the model in kB;
- $N_{MFLOP/s}$ ***Real-Time***: Approximated number of MFLOPs per second required to infer each model in a real-time fault detection implementation using the same sample rate (10 kHz) as the LU dataset.

4.2.1. Performance

The best performing model on all datasets was the Spatial-Channel Multi-Scale CNN with an overall accuracy of 99.3 percent on the CWRU dataset, 85.0 percent on the Grouped class configuration test set of the LU dataset, and 82.2 percent on the Fully Separated class configuration test set of the LU dataset. The near 20 percent misclassified LU samples related primarily to the gear faults. The CNN model did similarly to the MSCNN on the Grouped class configuration of the LU dataset (within 1 percent). However, it misclassified nearly 15 percent and 21 percent more samples on the CWRU and Fully Separated class configuration of the LU dataset than the MSCNN, respectively.

Both SVC implementations did similarly on both datasets with overall accuracies nearing 72 percent on the CWRU dataset and 31 percent on the Grouped class configuration of the LU dataset. Because the SVM implementations used less training data than the DL implementations, differences in classification accuracy cannot be attributed solely to model architecture. These results demonstrate that the SVM could be used on a simpler application, but anything more than steady-state machinery with an array of fault classes similar to the LU dataset would not result in a favourable implementation. The random forest classifier, consisting of 500 trees, performed comparably to the deep learning models on the CWRU dataset and LU validation set. However, when tested on the LU test set, which featured a different speed profile from the training set, it slightly outperformed the SVC models, achieving an overall accuracy of 34.7 percent. As noted above, the CWRU scores reflect validation and the windows used from the same recording may span training and validation. The sample-level split does not establish independence between recordings.

4.2.2. Model Size and Computational Demands

The largest model in each implementation was the random forest classifier. The smallest random forest classifier implementation required 28.4 percent more memory allocation for parameters than the largest of the DL models. For the CWRU random forest model, Table 4 reports 0.2144 MFLOP per inference and an estimated real-time processing requirement of 0.64 MFLOP/s. Despite their poorer results in classification accuracy, the SVM implementations had the lowest computational requirement in terms of FLOPs required to infer them. A computing source capable of 140,000 FLOPs per second would be needed to process the best performing SVM on the LU dataset in a real-time detection application. This is nearly 1500 times less than the best performing model; the spatial-channel multi-scale CNN. Additionally, the largest of the best performing SVCs on the CWRU dataset required roughly 64.9 kB in memory to host it whereas the best performing random forest classifier required 834.3 kB. The random forest classifier implementation scored well on the validation set from the CWRU dataset and the LU dataset, however when faced with the variation in speed profile between the LU training and test sets, it performed as poorly as the other ML methods (near 30 percent overall classification accuracy).

4.2.3. Edge Computing Considerations

To help understand the computational requirements for inferring each model featured in this research, the specifications and costs of an Arduino Mega 2560 REV3, a Raspberry Pi 5, and a Jetson Nano are considered. It is important to note that these embedded devices may not meet the safety or weatherproofing standards necessary for many industrial environments. They were selected to illustrate potential capabilities rather than represent

optimal use cases. An Arduino Mega would not be a practical option for data-driven embedded applications and is only listed to demonstrate the computational capabilities of a common, low-cost ECU.

The Raspberry Pi 5 and Jetson Nano appear capable of real-time inference for the models considered here, although training deep learning models on these devices would likely be slower and less practical than training on a workstation or GPU-enabled system [27]. These relatively low-cost controllers demonstrate that there is room for model development and scale, further enabling them to predict behaviour in machinery. (See Table 6).

Table 6. Potential edge-computing devices (theoretical).

Specification	Arduino Mega 2560	Raspberry Pi 5	Jetson Nano
Memory	8 kB SRAM, 248 kB flash	4 GB or 8 GB	4 GB 64-bit LPDDR4, 1600 MHz 25.6 GB/s
Controller/CPU	ATmega2560—8-bit AVR® RISC-based—16 MHz	Broadcom BCM2712 2.4 GHz quad-core 64-bit Arm Cortex-A76 CPU	NVIDIA Maxwell architecture with 128 NVIDIA CUDA® cores
Compute Performance	160 kFLOP/s (Assuming 1/100 FLOP per cycle)	9.6 GFLOP/s (Assuming 4 FLOP per cycle)	472 GFLOP/s (as claimed by NVIDIA) [28]
Storage	-	128 GB MicroSD	16 GB eMMC 5.1
Additional Features	-	-	High-rate communication and GPIO pins

5. Conclusions

The results of this work demonstrate that, with further application-specific model tuning, the traditional techniques studied could be suitably applied to a system with complexity similar to that found in the CWRU dataset. However, they had significantly greater difficulty classifying the vibration signatures from the unseen speed profile in the LU dataset compared to the DL models. The novel DL technique (MSCNN) outperformed every other model on each dataset configuration. The CNN implementation performed similarly to the MSCNN on the grouped class configuration but could not distinguish one bearing fault type from another within the CWRU dataset and struggled with the greater number of classes in the fully separated class configuration of the LU dataset.

Computationally, the approximated memory and processing power required to infer the ML implementations (aside from the random forest classifier in terms of memory), were significantly less than would be needed to host the DL models. However, it is also demonstrated that the DL models are within reasonable bounds for processing on relatively low-cost, accessible embedded devices. The MSCNN requires more memory usage than the CNN but requires fewer floating-point operations and therefore less processing resources for inference, making a case that the MSCNN is a better choice for both performance and computational demand.

In conclusion, these results suggest that the further development of DL models for condition monitoring and fault detection applications has merit. However, further investment into experimental dataset generation should be considered to refine the selection criteria.

Author Contributions: Conceptualization, M.-A.H., M.T. and C.M.; Methodology, M.-A.H., M.T. and C.M.; Software, M.-A.H.; Validation, M.-A.H.; Formal analysis, M.-A.H., M.T. and C.M.; Resources, M.T. and C.M.; Data curation, M.-A.H.; Writing—original draft, M.-A.H.; Writing—review & editing, M.T. and C.M.; Visualization, M.-A.H.; Supervision, M.T. and C.M.; Project administration, M.T. and C.M.; Funding acquisition, M.T. and C.M. All authors have read and agreed to the published version of the manuscript.

Funding: Natural Sciences and Engineering Research Council of Canada: RGPION-2025-06174.

Data Availability Statement: The datasets presented in this article are not readily available because the data are part of an ongoing study. Requests to access the datasets should be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

List of Abbreviations

ANN	Artificial Neural Network
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CWRU	Case-Western Reserve University Bearing Dataset
DL	Deep Learning
FLOP	Floating-Point Operation(s)
GB	Gigabyte(s)
LU	Laurentian University
LSTM	Long Short-Term Memory
ML	Machine Learning
MLP	Multi-Layered Perceptron
MSCNN	Multi-Scale Convolutional Neural Network
SVC	Support Vector Classifier
SVM	Support Vector Machine

Appendix A

The SVM's hyperparameter selection scheme followed the process outlined below. For a given parameter k, the penalty term C is calculated as:

$$\text{If} \|\underline{\sigma}\| >= 1 \text{ then}, \quad C = \|\underline{\sigma}\|^k$$

$$\text{Else} \quad C = \|\underline{\sigma}\|^{\frac{1}{k}}$$

where $\|\underline{\sigma}\|$ is the Euclidean norm of the standard deviations over each feature across the training dataset. The γ parameter used in the kernel formulation is defined as C^{-1}, where C scales the L2 penalty calculation. The development of the Support Vector Classifier (SVC) implementation involved three stages: the selection of features, the selection of the hyper-parameter k, and eventually, solving the model. A graphical representation of this is illustrated in Figure 4.

The feature dataset for the LU dataset was originally intended to have the same characteristics (sample size and number of samples) as those used to train the CNN or Multi-Scale CNN. However, solving a SVM is a quadratic problem of $O(n_{features} \times n^2{}_{samples})$. Therefore, with 16 features, an input size of 4000 acceleration data points, and with 290 min of data across 27 different experiments in the LU dataset sampled at 10 kHz, would account for roughly 1×10^{14} operations without considering the multiple iterations required for hyperparameter and feature selection. To put that into perspective, with the assumption of one operation per clock cycle of a CPU rated for 5 GHz, solving the optimization problem could take 5.4 h to compute. The time duration sampled from each experiment was thus reduced to two minutes. Which, by the same metrics, would account for roughly 4×10^{12} operations or 12.8 min.

The method used to select features for the SVC implementation involved selecting the features that represent the greatest Gini importance (mean decrease in impurity) in a random forest classifier of 500 trees. The Gini importance of a given feature is the normalized total reduction in the impurity brought by said feature. The scikit-learn python

module contains a class which returns an input sized vector of Boolean values that are set to true if the Gini importance of the feature at that index is greater than the average across all features.

Once the features are selected from a particular training set, the unnecessary features are pruned from the training set, validation set and test sets. The Euclidean norm of the standard deviations from each feature in the pruned training set is used along with the given iteration of the value of k to calculate C and γ. For each value of $k = \{1, 2, 3\}$, an SVC is fitted to the training set and evaluated with the validation set. If the accuracy on the validation set is greater than the running best or $k = 1$, then the model is evaluated on the test sets and saved as a checkpoint.

A random forest classifier of the same size (500 trees) as the one used to select features for the SVC was trained and tested on the pruned datasets to offer a performance comparison on the feature-based implementation. Additionally, an SVC with a radial basis function kernel and default parameters for the penalty term and γ was also trained and tested to provide a baseline to contrast the performance of the model with the parameter selection scheme mentioned above. Scikit-learn's SVC module [29] uses default values: $C = 1.0$ and $\gamma = (\text{number of features})^{-1}$ when set to 'auto'. The CNN used in this study is defined as shown in Figures A1 and A2.

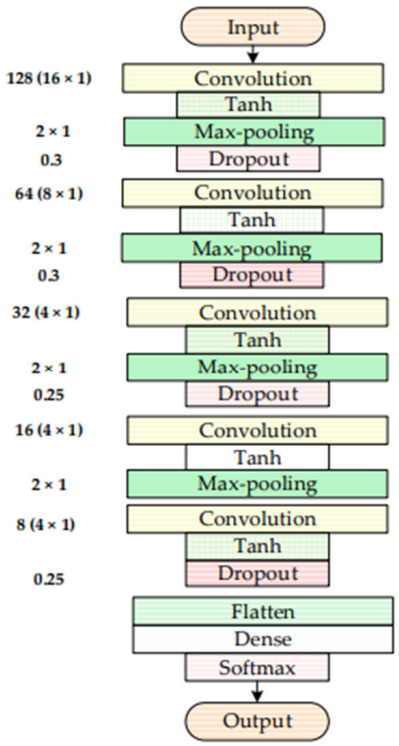


Figure A1. CNN architecture.

The model was trained in batches of 64 samples with the Adam optimization algorithm and a learning rate of 0.001. The loss criterion was cross entropy loss. The input size varied between 1000 or 4000 acceleration data points. The fully connected classifier's input size was dependent on the input to the model and thus varied between (batch size, 216) and (batch size, 968). The output size of the classifier was dependent on the class configuration of the dataset, and thus varied between (batch size, 7) and (batch size, 27).

The multi-scale convolutional neural-network implementation using a one-dimensional vector of the raw or standardized acceleration data was used instead to capitalize on the

feature extraction capabilities of the model. A generalized architecture can be found in Figure A3 below, and a more detailed representation can be found in Figure 5.

Network Layer	Output Characteristic	Specific Settings
Input layer	1024 × 1	1024 pieces of vibration data
Conv1 layer	1009 × 128	128 @ 16 × 1, stride = 1
Pool1 layer	504 × 128	pool size is 2 × 1, stride = 2
Conv2 layer	497 × 64	64 @ 8 × 1, stride = 1
Pool2 layer	248 × 64	pool size is 2 × 1, stride = 2
Conv3 layer	245 × 32	32 @ 4 × 1, stride = 1
Pool3 layer	122 × 32	pool size is 2 × 1, stride = 2
Conv4 layer	119 × 16	16 @ 4 × 1, stride = 1
Pool4 layer	59 × 16	pool size is 2 × 1, stride = 2
Conv5 layer	56 × 8	8 @ 4 × 1, stride = 1
Flatten	1 × 448	448 neurons
Dense	1 × 10	10 neurons

Figure A2. CNN architecture layer parameters.

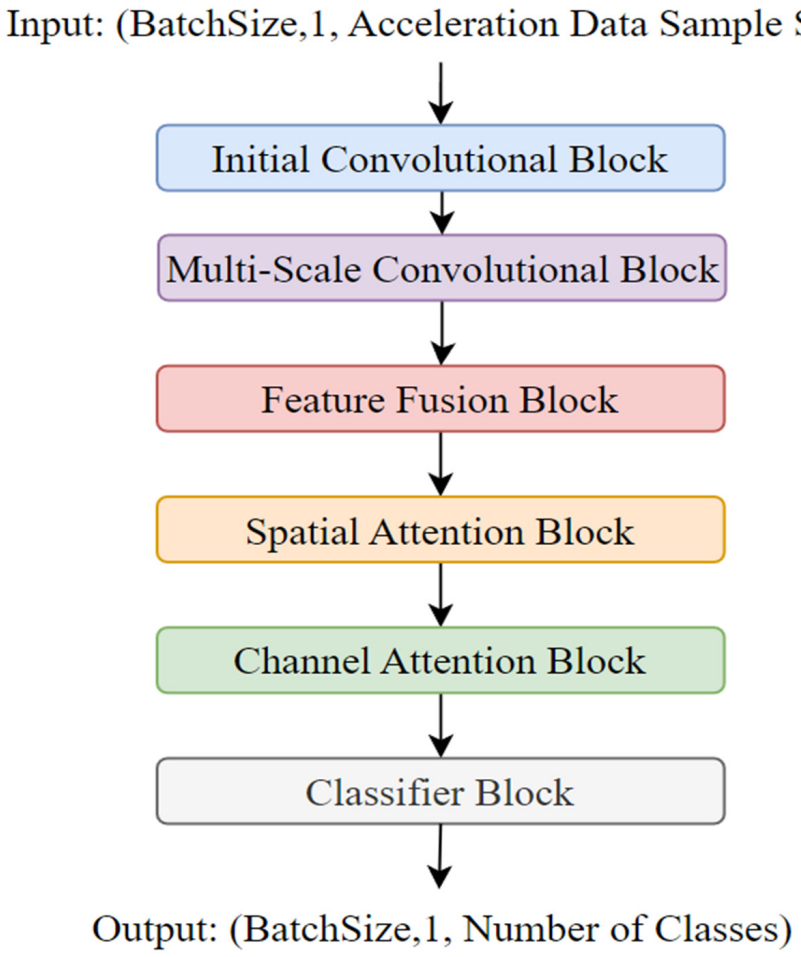


Figure A3. MSCNN implementation flowchart.

Appendix B

Confusion Matrices from the Fully Separated Class Configuration of the LU Datasets.

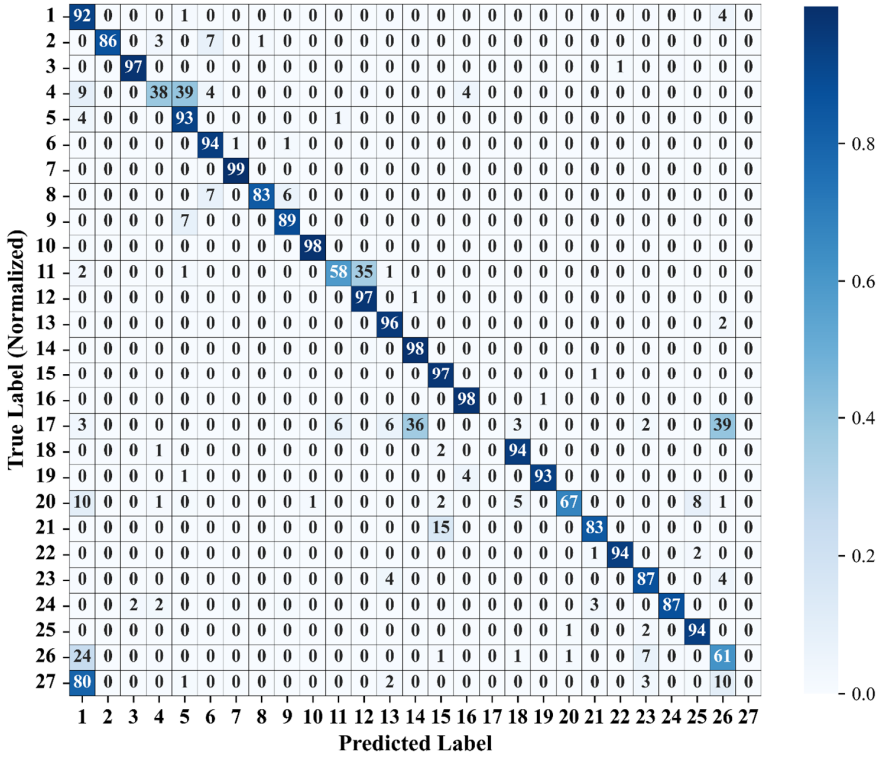


Figure A4. Confusion matrix for the best performing spatial-channel MSCNN evaluated on the fully separated class configuration test set (sawtooth speed profile).

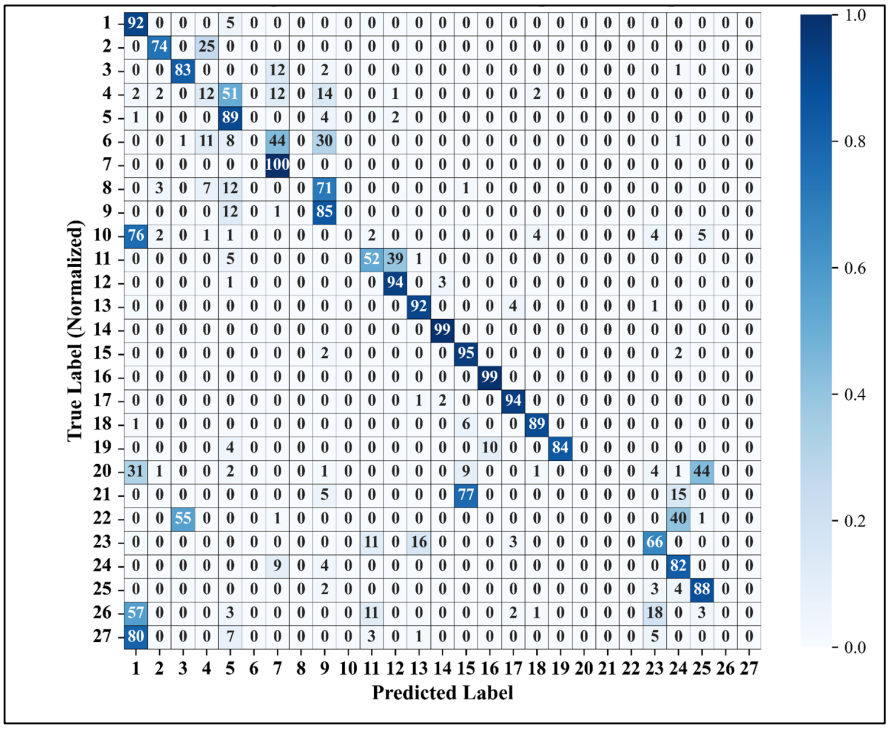
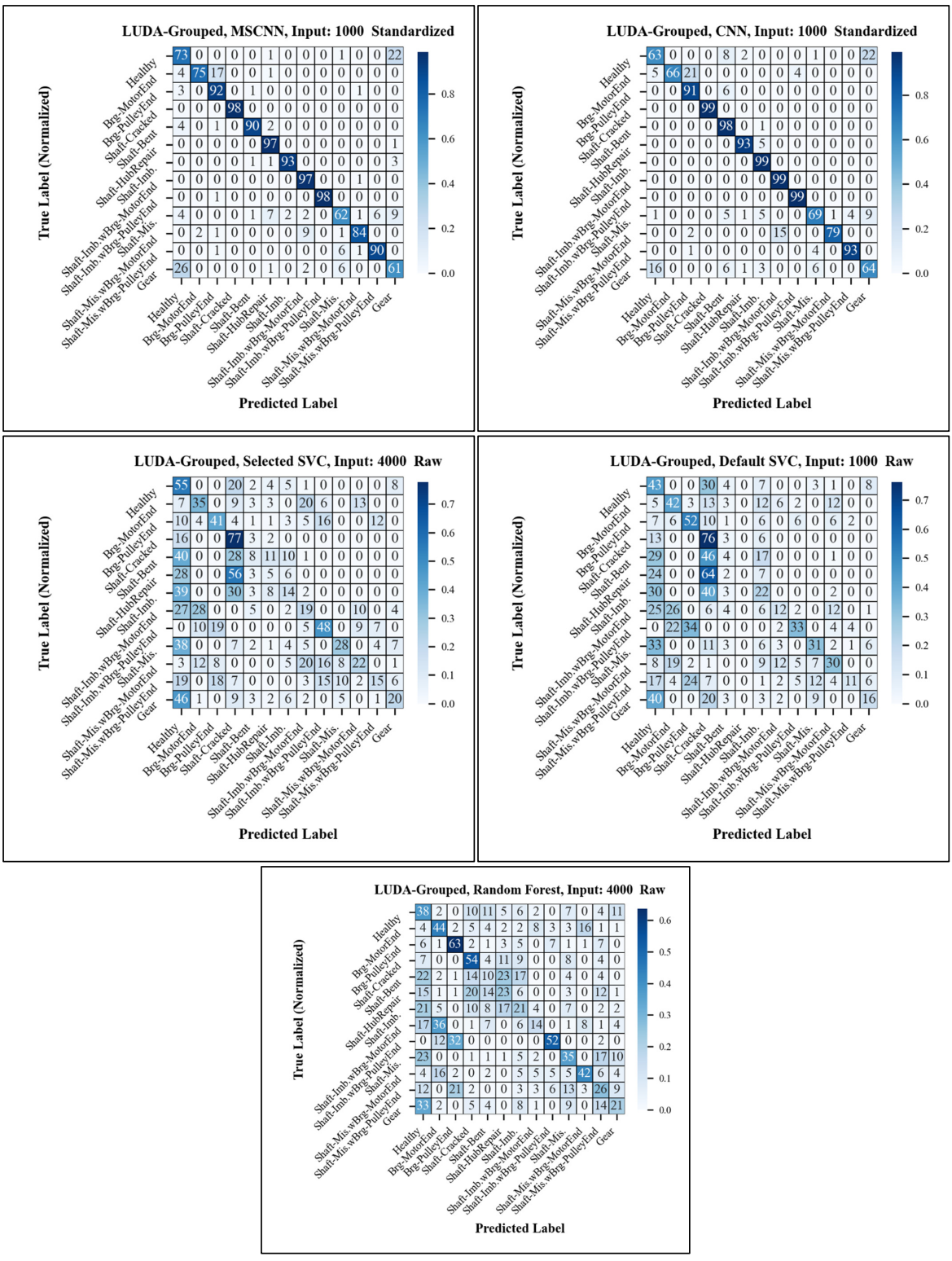


Figure A5. Confusion matrix for the best performing CNN evaluated on the fully separated class configuration test set (sawtooth speed profile).

Appendix C

Confusion Matrices from the Grouped Class Configuration of the LU Datasets.



Appendix D

Classification reports from the CWRU dataset.

The support column contains the number of samples present from each class within the CWRU validation set. The CWRU scores reflect validation and the windows used from the same recording may span training and validation. The sample-level split does not establish independence between recordings.

MSCNN, CWRU, Input Size: 1000 Samples-Standardized

Class	precision	recall	f1-score	support
InnerRace Drive End	1.0	0.99	1.0	396
Outer Fan End	0.99	0.99	0.99	479
Outer Drive End	1.0	1.0	1.0	678
Ball Drive End	1.0	1.0	1.0	413
Ball Fan End	0.99	0.97	0.98	283
InnerRace Fan End	0.97	0.99	0.98	278
Normal Data	1.0	1.0	1.0	366

CNN, CWRU, Input Size: 1000 Samples-Standardized

Class	precision	recall	f1-score	support
InnerRace Drive End	0.0	0.0	0.0	396
Outer Fan End	0.99	0.99	0.99	479
Outer Drive End	0.64	1.0	0.78	678
Ball Drive End	0.96	0.99	0.97	413
Ball Fan End	0.99	0.98	0.99	283
InnerRace Fan End	0.99	1.0	0.99	278
Normal Data	1.0	1.0	1.0	366

Selected SVC, CWRU, Input Size: 4000 Samples-Raw

Class	precision	recall	f1-score	support
InnerRace Drive End	0.7	0.99	0.82	78
Outer Fan End	0.45	0.8	0.58	122
Outer Drive End	0.94	0.79	0.86	165
Ball Drive End	0.95	0.38	0.55	107
Ball Fan End	0.36	0.12	0.18	86
InnerRace Fan End	0.76	0.96	0.85	70
Normal Data	0.97	1.0	0.98	94

Default SVC, CWRU, Input Size: 4000 Samples-Raw

Class	precision	recall	f1-score	support
InnerRace Drive End	0.85	0.96	0.9	78
Outer Fan End	0.47	0.8	0.59	122
Outer Drive End	0.93	0.83	0.88	165
Ball Drive End	0.94	0.48	0.63	107
Ball Fan End	0.25	0.06	0.09	86
InnerRace Fan End	0.67	0.97	0.8	70
Normal Data	0.9	1.0	0.94	94

Random Forest, CWRU, Input Size: 4000 Samples-Raw

Class	precision	recall	f1-score	support
InnerRace Drive End	0.96	0.99	0.97	78
Outer Fan End	0.8	0.89	0.84	122
Outer Drive End	0.96	0.93	0.94	165
Ball Drive End	0.98	0.95	0.97	107
Ball Fan End	0.82	0.67	0.74	86
InnerRace Fan End	0.9	0.99	0.94	70
Normal Data	0.99	1.0	0.99	94

Appendix E

Classification reports from the LU dataset (Fully Separated Configuration).

The support column contains the number of samples present from each class within the test set.

MSCNN, LUDataset Full, Input Size: 1000 Samples-Raw

Class	precision	recall	f1-score	support
Healthy-NA	0.73	0.93	0.82	24000
Brg-Inner-MotorEnd	0.98	0.87	0.92	6000
Brg-Inner-PulleyEnd	0.96	0.98	0.97	6000
Bearing-Outer-MotorEnd	0.78	0.38	0.51	6000
Bearing-Outer-PulleyEnd	0.61	0.93	0.74	6000
Brg- Ball-MotorEnd	0.83	0.95	0.88	6000
Brg- Ball-PulleyEnd	0.96	1.0	0.98	6000
Brg- Comb.-MotorEnd	0.97	0.84	0.9	6000
Brg- Comb.-PulleyEnd	0.89	0.89	0.89	6000
Shaft - Cracked-NA	0.97	0.99	0.98	6000
Shaft - Centrally Bent-NA	0.84	0.59	0.69	6000
Shaft - Coupling End Bent-NA	0.71	0.97	0.82	6000
Shaft - Hub Repair-NA	0.85	0.96	0.9	6000
Shaft Imb. 1-NA	0.72	0.99	0.83	6000
Shaft Imb. 1 w Brg-Inner-MotorEnd	0.82	0.97	0.89	6000
Shaft Imb. 1 w Brg-Inner-PulleyEnd	0.91	0.98	0.94	6000
Shaft Imb. 2-NA	0.0	0.0	0.0	6000
Shaft Imb. 2 w Brg-Outer -MotorEnd	0.9	0.95	0.92	6000
Shaft Imb. 2 w Brg-Outer -PulleyEnd	0.97	0.94	0.96	6000
Shaft Mis. 1-Par.-NA	0.94	0.68	0.79	6000
Shaft Mis. 1-Par. w Brg-Inner-MotorEnd	0.91	0.84	0.87	6000
Shaft Mis. 1-Par. w Brg-Inner-PulleyEnd	0.98	0.95	0.96	6000
Shaft Mis. 2-Ang.-NA	0.83	0.87	0.85	6000
Shaft Mis. 2-Ang. w Brg-Outer -MotorEnd	0.96	0.87	0.91	6000
Shaft Mis. 2-Ang. w Brg-Outer -PulleyEnd	0.87	0.94	0.9	6000
Gear - Chipped tooth-NA	0.44	0.62	0.52	6000
Gear - Missing tooth-NA	0.0	0.0	0.0	6000

CNN, LUDataset Full, Input Size: 1000 Samples-Standardized

Class	precision	recall	f1-score	support
Healthy-NA	0.59	0.92	0.72	24000
Brg-Inner-MotorEnd	0.87	0.74	0.8	6000
Brg-Inner-PulleyEnd	0.58	0.83	0.68	6000
Bearing-Outer-MotorEnd	0.21	0.13	0.16	6000
Bearing-Outer-PulleyEnd	0.4	0.9	0.55	6000
Brg- Ball-MotorEnd	0.0	0.0	0.0	6000
Brg- Ball-PulleyEnd	0.54	1.0	0.7	6000
Brg- Comb.-MotorEnd	0.0	0.0	0.0	6000
Brg- Comb.-PulleyEnd	0.38	0.85	0.52	6000
Shaft - Cracked-NA	0.0	0.0	0.0	6000
Shaft - Centrally Bent-NA	0.6	0.52	0.56	6000
Shaft - Coupling End Bent-NA	0.68	0.94	0.79	6000
Shaft - Hub Repair-NA	0.79	0.92	0.85	6000
Shaft Imb. 1-NA	0.95	0.99	0.97	6000
Shaft Imb. 1 w Brg-Inner-MotorEnd	0.49	0.95	0.65	6000
Shaft Imb. 1 w Brg-Inner-PulleyEnd	0.89	0.99	0.94	6000
Shaft Imb. 2-NA	0.89	0.94	0.92	6000
Shaft Imb. 2 w Brg-Outer -MotorEnd	0.87	0.9	0.88	6000
Shaft Imb. 2 w Brg-Outer -PulleyEnd	0.99	0.84	0.91	6000
Shaft Mis. 1-Par.-NA	0.0	0.0	0.0	6000
Shaft Mis. 1-Par. w Brg-Inner-MotorEnd	0.0	0.0	0.0	6000
Shaft Mis. 1-Par. w Brg-Inner-PulleyEnd	0.0	0.0	0.0	6000
Shaft Mis. 2-Ang.-NA	0.62	0.67	0.64	6000
Shaft Mis. 2-Ang. w Brg-Outer -MotorEnd	0.55	0.83	0.66	6000
Shaft Mis. 2-Ang. w Brg-Outer -PulleyEnd	0.61	0.88	0.72	6000
Gear - Chipped tooth-NA	0.0	0.0	0.0	6000
Gear - Missing tooth-NA	0.0	0.0	0.0	6000

Appendix F

Classification reports from the LU dataset (Grouped Configuration).

The support column contains the number of samples present from each class within the test set.

MSCNN, LUDataset Grouped, Input Size: 1000 Samples-Standardized

Class	precision	recall	f1-score	support
Healthy	0.76	0.73	0.75	12000
Bearing-MotorEnd	0.96	0.75	0.84	6000
Bearing-PulleyEnd	0.79	0.92	0.85	6000
Shaft-Cracked	0.99	0.99	0.99	6000
Shaft-Bent	0.94	0.91	0.92	6000
Shaft-HubRepair	0.85	0.98	0.91	6000
Shaft-Imb.	0.96	0.94	0.95	6000
Shaft-Imb.wBearing-MotorEnd	0.86	0.97	0.92	6000
Shaft-Imb.wBearing-PulleyEnd	0.99	0.98	0.99	6000
Shaft-Mis.	0.77	0.63	0.69	6000
Shaft-Mis.wBearing-MotorEnd	0.93	0.85	0.88	6000
Shaft-Mis.wBearing-PulleyEnd	0.92	0.9	0.91	6000
Gear	0.5	0.62	0.55	6000

CNN, LUDataset Grouped, Input Size: 1000 Samples-Standardized

Class	precision	recall	f1-score	support
Healthy	0.84	0.64	0.72	12000
Bearing-MotorEnd	0.98	0.66	0.79	6000
Bearing-PulleyEnd	0.76	0.91	0.83	6000
Shaft-Cracked	0.98	0.99	0.99	6000
Shaft-Bent	0.73	0.98	0.84	6000
Shaft-HubRepair	0.93	0.94	0.93	6000
Shaft-Imb.	0.85	0.99	0.92	6000
Shaft-Imb.wBearing-MotorEnd	0.84	0.99	0.91	6000
Shaft-Imb.wBearing-PulleyEnd	0.95	0.99	0.97	6000
Shaft-Mis.	0.84	0.69	0.76	6000
Shaft-Mis.wBearing-MotorEnd	0.96	0.8	0.87	6000
Shaft-Mis.wBearing-PulleyEnd	0.94	0.93	0.93	6000
Gear	0.53	0.64	0.58	6000

Selected SVC, LUDataset Grouped, Input Size: 4000 Samples-Raw

Class	precision	recall	f1-score	support
Healthy	0.29	0.56	0.38	600
Bearing-MotorEnd	0.37	0.35	0.36	300
Bearing-PulleyEnd	0.47	0.41	0.44	300
Shaft-Cracked	0.29	0.78	0.42	300
Shaft-Bent	0.21	0.08	0.12	300
Shaft-HubRepair	0.12	0.06	0.08	300
Shaft-Imb.	0.22	0.15	0.18	300
Shaft-Imb.wBearing-MotorEnd	0.22	0.19	0.2	300
Shaft-Imb.wBearing-PulleyEnd	0.46	0.48	0.47	300
Shaft-Mis.	0.51	0.28	0.36	300
Shaft-Mis.wBearing-MotorEnd	0.38	0.23	0.28	300
Shaft-Mis.wBearing-PulleyEnd	0.35	0.16	0.22	300
Gear	0.36	0.21	0.26	300

Default SVC, LUDataset Grouped, Input Size: 1000 Samples-Raw

Class	precision	recall	f1-score	support
Healthy	0.27	0.43	0.33	2400
Bearing-MotorEnd	0.34	0.42	0.38	1200
Bearing-PulleyEnd	0.44	0.52	0.48	1200
Shaft-Cracked	0.21	0.76	0.33	1200
Shaft-Bent	0.12	0.05	0.07	1200
Shaft-HubRepair	0.12	0.0	0.01	1200
Shaft-Imb.	0.19	0.22	0.21	1200
Shaft-Imb.wBearing-MotorEnd	0.28	0.13	0.17	1200
Shaft-Imb.wBearing-PulleyEnd	0.59	0.33	0.42	1200
Shaft-Mis.	0.45	0.31	0.37	1200
Shaft-Mis.wBearing-MotorEnd	0.38	0.31	0.34	1200
Shaft-Mis.wBearing-PulleyEnd	0.59	0.12	0.19	1200
Gear	0.34	0.17	0.23	1200

Random Forest, LUDataset Grouped, Input Size: 4000 Samples-Raw

Class	precision	recall	f1-score	support
Healthy	0.31	0.38	0.34	600
Bearing-MotorEnd	0.34	0.44	0.39	300
Bearing-PulleyEnd	0.5	0.64	0.56	300
Shaft-Cracked	0.39	0.54	0.46	300
Shaft-Bent	0.13	0.1	0.11	300
Shaft-HubRepair	0.25	0.23	0.24	300
Shaft-Imb.	0.21	0.21	0.21	300
Shaft-Imb.wBearing-MotorEnd	0.31	0.14	0.2	300
Shaft-Imb.wBearing-PulleyEnd	0.7	0.52	0.6	300
Shaft-Mis.	0.34	0.36	0.35	300
Shaft-Mis.wBearing-MotorEnd	0.57	0.42	0.49	300
Shaft-Mis.wBearing-PulleyEnd	0.24	0.27	0.25	300
Gear	0.27	0.21	0.24	300

References

1. Jardine, A.K.S.; Lin, D.; Banjevic, D. A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mech. Syst. Signal Process* **2006**, *20*, 1483–1510. [CrossRef]
2. Tiboni, M.; Remino, C.; Bussola, C.; Amici, C. A Review on Vibration-Based Condition Monitoring of Rotating Machinery. *Appl. Sci.* **2022**, *12*, 972. [CrossRef]
3. Wang, W.; Taylor, J.; Rees, R.J. Recent Advancement of Deep Learning Applications to Machine Condition Monitoring Part 2: Supplement Views and a Case Study. *Acoust. Aust.* **2021**, *49*, 221–228. [CrossRef]
4. Wang, W.; Taylor, J.; Rees, R.J. Recent Advancement of Deep Learning Applications to Machine Condition Monitoring Part 1: A Critical Review. *Acoust. Aust.* **2021**, *49*, 207–219. [CrossRef]
5. Wang, F.; Wang, K. Intelligent condition monitoring and diagnosis system: A computational intelligence approach. In *Frontiers in Artificial Intelligence and Applications*; IOS Press: Amsterdam/Berlin, Germany, 2003; Volume 93.
6. Matania, O.; Dattner, I.; Bortman, J.; Kenett, R.S.; Parmet, Y. A systematic literature review of deep learning for vibration-based fault diagnosis of critical rotating machinery: Limitations and challenges. *J. Sound Vib.* **2024**, *590*, 118562. [CrossRef]
7. Case Western Reserve University, Case Western Reserve University Bearing Data. Available online: https://engineering.case.edu/bearingdatacenter (accessed on 30 April 2024).
8. Lessmeier, C.; Kimotho, J.K.; Zimmer, D.; Sextro, W. Condition Monitoring of Bearing Damage in Electromechanical Drive Systems by Using Motor Current Signals of Electric Motors: A Benchmark Data Set for Data-Driven Classification. *PHM Soc. Eur. Conf.* **2016**, *3*, 1577. [CrossRef]
9. Bechhoefer, E. Condition Based Maintenance Fault Database for Testing of Diagnostic and Prognostics Algorithms. Available online: https://www.mfpt.org/fault-data-sets/ (accessed on 20 July 2022).

10. Gangsar, P.; Tiwari, R. A support vector machine based fault diagnostics of Induction motors for practical situation of multi-sensor limited data case. *Measurement* **2019**, *135*, 694–711. [CrossRef]
11. Rojas, A.; Nandi, A.K. Detection and Classification of Rolling-Element Bearing Faults using Support Vector Machines. In Proceedings of the IEEE Workshop on Machine Learning for Signal Processing, Mystic, CT, USA, 28–30 September 2005; pp. 153–158.
12. Samanta, B.; Al-Balushi, K.R. Artificial neural network based fault diagnostics of rolling element bearings using time-domain features. *Mech. Syst. Signal Process* **2003**, *17*, 317–328. [CrossRef]
13. Jana, D.; Patil, J.; Herkal, S.; Nagarajaiah, S.; Duenas-Osorio, L. CNN and Convolutional Autoencoder (CAE) based real-time sensor fault detection, localization, and correction. *Mech. Syst. Signal Process* **2022**, *169*, 108723. [CrossRef]
14. Arellano-Espitia, F.; Delgado-Prieto, M.; Martinez-Viol, V.; Saucedo-Dorantes, J.J.; Osornio-Rios, R.A. Deep-Learning-Based Methodology for Fault Diagnosis in Electromechanical Systems. *Sensors* **2020**, *20*, 3949. [CrossRef]
15. Li, Z.; Liu, F.; Yang, W.; Peng, S.; Zhou, J. A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, *33*, 6999–7019. [CrossRef]
16. Zhang, S.; Zhang, S.; Wang, B.; Habetler, T.G. Deep Learning Algorithms for Bearing Fault Diagnostics—A Comprehensive Review. *IEEE Access* **2020**, *8*, 29857–29881. [CrossRef]
17. Chen, C.C.; Liu, Z.; Yang, G.; Wu, C.C.; Ye, Q. An Improved Fault Diagnosis Using 1D-Convolutional Neural Network Model. *Electronics* **2020**, *10*, 59. [CrossRef]
18. Huang, T.; Fu, S.; Feng, H.; Kuang, J. Bearing Fault Diagnosis Based on Shallow Multi-Scale Convolutional Neural Network with Attention. *Energies* **2019**, *12*, 3937. [CrossRef]
19. Kim, Y.; Na, K.; Youn, B.D. A health-adaptive time-scale representation (HTSR) embedded convolutional neural network for gearbox fault diagnostics. *Mech. Syst. Signal Process.* **2022**, *167*, 108575. [CrossRef]
20. He, J.; Wu, P.; Tong, Y.; Zhang, X.; Lei, M.; Gao, J. Bearing Fault Diagnosis via Improved One-Dimensional Multi-Scale Dilated CNN. *Sensors* **2021**, *21*, 7319. [CrossRef]
21. Xiao, Y.; Shao, H.; Yan, S.; Wang, J.; Peng, Y.; Liu, B. Domain generalization for rotating machinery fault diagnosis: A survey. *Adv. Eng. Inform.* **2025**, *64*, 103063. [CrossRef]
22. Pang, B.; Liu, Q.; Sun, Z.; Xu, Z.; Hao, Z. Time-frequency supervised contrastive learning via pseudo-labeling: An unsupervised domain adaptation network for rolling bearing fault diagnosis under time-varying speeds. *Adv. Eng. Inform.* **2024**, *59*, 102304. [CrossRef]
23. Gao, T.; Yang, J.; Tang, Q. A multi-source domain information fusion network for rotating machinery fault diagnosis under variable operating conditions. *Inf. Fusion* **2024**, *106*, 102278. [CrossRef]
24. SpectraQuest Machinery Fault Simulator. Available online: https://spectraquest.com/machinery-fault-simulators/ (accessed on 16 March 2021).
25. Woo, S.; Park, J.; Lee, J.-Y.; Kweon, I.S. CBAM: Convolutional Block Attention Module. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 3–19.
26. Ioffe, S.; Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv* **2015**. [CrossRef]
27. Duan, Z.; Zhang, W.; Zhang, H.; Yang, F. Research on Intelligent Fault Diagnosis of Rotating Machinery for Edge Computing Platforms. *Mech. Syst. Signal Process.* **2025**, *237*, 113101. [CrossRef]
28. Franklin, D. Jetson Nano Brings AI Computing to Everyone. NVIDIA Technical Blog. Available online: https://developer.nvidia.com/blog/jetson-nano-ai-computing/ (accessed on 12 April 2024).
29. Pedregosa, F. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.