vibration

MDPI

Article

Exploring the Relationship between Preprocessing and Hyperparameter Tuning for Vibration-Based Machine Fault Diagnosis Using CNNs

Jacob Hendriks * and Patrick Dumond

Department of Mechanical Engineering, Faculty of Engineering, University of Ottawa, Ottawa, ON K1N 6N5, Canada; pdumond@uottawa.ca

* Correspondence: jhend076@uottawa.ca

Abstract: This paper demonstrates the differences between popular transformation-based input representations for vibration-based machine fault diagnosis. This paper highlights the dependency of different input representations on hyperparameter selection with the results of training different configurations of classical convolutional neural networks (CNNs) with three common benchmarking datasets. Raw temporal measurement, Fourier spectrum, envelope spectrum, and spectrogram input types are individually used to train CNNs. Many configurations of CNNs are trained, with variable input sizes, convolutional kernel sizes and stride. The results show that each input type favors different combinations of hyperparameters, and that each of the datasets studied yield different performance characteristics. The input sizes are found to be the most significant determiner of whether overfitting will occur. It is demonstrated that CNNs trained with spectrograms are less dependent on hyperparameter optimization over all three datasets. This paper demonstrates the wide range of performance achieved by CNNs when preprocessing method and hyperparameters are varied as well as their complex interaction, providing researchers with useful background information and a starting place for further optimization.

Keywords: condition monitoring; fault diagnosis; convolutional neural networks; hyperparameters; data representations

Citation: Hendriks, J.; Dumond, P. Exploring the Relationship between Preprocessing and Hyperparameter Tuning for Vibration-Based Machine Fault Diagnosis Using CNNs. *Vibration* **2021**, *4*, 284–309. https://doi.org/10.3390/vibration4020019

Academic Editor: Irina Trendafilova

Received: 18 February 2021
Accepted: 23 March 2021
Published: 3 April 2021

1. Introduction

Data-driven machine health monitoring (MHM) can allow machine operators to improve capitalization of a mechanical asset's useful lifetime and avoid unexpected interruptions to machine operability by detecting mechanical faults in advance of a machine breakdown. Traditionally, signals gathered from machine-mounted sensors are analyzed by system experts to determine whether maintenance action should be taken. This involves first gathering and preprocessing the signals, extracting useful features from the signals, and then analyzing the features to determine whether a fault is present. Current research for advancing MHM is focused on improving automatic feature selection and feature analysis to reduce the dependence on human experts [1].

Bearings and gears are common subjects for MHM since they are nearly ubiquitous in machine design and are accountable for a large proportion of machine failures [2]. Vibration, temperature, current, oil analysis, and acoustic emissions are among the types of signals used in MHM. Vibration signals are the most widely investigated signal type for their convenience and affordability, and because vibration signals can carry useful information about fault type and severity. Machine learning is regarded as a suitable tool for use in vibration-based MHM for its ability to detect complex patterns contained in vibration signals. Classical applications of machine learning using hand-crafted features such as k-nearest neighbors (kNN) and support vector machines (SVM) have been widely explored and perform well in many fault diagnosis problems [3]. The performance of these

shallow networks is highly dependent on whether the extracted features contain useful diagnostic information. Generally, some domain expertise is required in order to select appropriate features [4]. Performance can be improved by automating feature selection, often done by integrating feature selection into the training process [5].

Convolutional neural networks (CNNs) are a type of deep learning algorithm that perform well for abstract tasks such as image recognition and natural language processing [6]. For these types of tasks, the shared weights used in the convolutional layers improve generalization and computational efficiency compared with artificial neural networks, allowing CNNs to efficiently learn features directly from high-dimensional inputs. This can allow them to be trained directly with large, complex inputs including unprocessed raw data, eliminating the need for manual feature engineering. CNNs commonly operate with 2D input spaces such as images or audio spectrograms, but they can also be implemented for tasks with 1D inputs such as raw vibration signals or frequency spectra.

Given the robustness of CNNs in dealing with large inputs, preparing them with raw data presents the most convenient option. However, there are still reasons for preprocessing input data to some extent. Bengio et al. [6] suggest that different representations of the same data can entangle and hide different explanatory factors of variation behind the data. A good data representation can reduce the number of parameters to be trained, reduce the number of training samples required to achieve generalization, quicken network convergence, and improve overall accuracy of the network. Therefore, some investigations have explored which input data representations, obtained with various data transformations, are most suitable for vibration-based fault diagnosis tasks.

Raw vibration signals have been used in 1D CNNs with and without low-pass filters and downsampling to diagnose faults in bearings [7,8] and gearboxes [9,10]. Chen et al. [11] perform bearing fault diagnosis using a low-pass filtered Fourier spectrum as a 1D input. Appana et al. [12] use the envelope spectrum of acoustic measurements for bearing diagnosis at various RPMs. For 2D CNN inputs, Guo et al. [13] use the wavelet transform scalogram to detect bearing faults, Han et al. [14] use an adapted wavelet-based transformation to diagnose gear faults, and Lee et al. [15] use combined spectrograms from multiple accelerometers in a single 2D image. Verstraete et al. [16] compare acceleration spectrograms, scalograms, and the Hilbert–Huang transformation for bearing fault diagnosis and demonstrate the strong influence of network architecture on diagnostic performance. Pandhare et al. [17] compare CNNs trained with raw time-domain, envelope spectrum, and spectrogram inputs against SVM and kNN machines. Liu et al. [18] perform fault diagnosis using a convolutional deep belief network with Fourier spectra inputs.

Most of the works referenced above introduce new combinations of CNN architectures with certain data representations not yet used for fault diagnosis. It is very rare for papers to compare the effectiveness of different data representations within an unchanged CNN architecture. Critically, the majority studies do not address the interaction between data representation and hyperparameter optimization or regard how algorithm performance is influenced by domain changes. None of the works studied have elaborated on the process by which CNN hyperparameter selection was conducted, nor was the sensitivity to changes in hyperparameter values, task domain, and preprocessing methods investigated. In the opinion of the author, too few previous works have validated novel applications of deep learning using multiple datasets available to the public. This work aims to address these shortcomings by investigating the effects of changing hyperparameters, while using various common data representations and data from three popular datasets in the public domain.

Specifically, the experiments repeatedly use three public-domain datasets with four different preprocessing methods to train various CNN configurations. Two-layer CNNs are used throughout, having their input dimension varied independently from their kernel size and stride values. These hyperparameter changes can significantly alter CNN performance with certain input types. Training all CNN configurations with each preprocessing method and for each dataset reveals the complex relationship between all of these variables. Additional details on the CNN architectures used are provided in Section 3.

The results will provide a template framework for presenting the design of CNN-based diagnosis algorithms and data representations, as well as determine whether some representations are universally stronger or stronger only when paired with certain CNN configurations and datasets. The following section reviews the basic concepts involved in bearing and gear fault diagnosis using CNNs, as well as introduces the different input types used to train CNNs.

2. Basic Theory

2.1. Vibration-Based Bearing Fault Diagnosis Fundamentals

This section describes the fundamental nature of the observation data and some of the known factors of variation behind it. Additional useful resources include a model for single-point defect vibration signals given by McFadden and Smith [19] and a detailed bearing fault diagnosis tutorial from Randall and Antoni [20].

Most diagnostic algorithms focus on localized bearing faults since they are more easily distinguished by their vibrational signature and usually precede destructive component failure. Localized faults consist of geometric irregularities, such as pits or spalls that inhibit the normal smooth motion of the bearing components relative to each other. These faults may arise due to any number of factors, such as overheating, lubrication failure, overloading, material defects, misalignment, and through natural material wear or fatigue. Some of the former factors can occur simultaneously and/or behave as catalysts to accelerate the natural rate of fatigue.

As the inner race and outer race rotate relative to each other, the contact zones between the different bearing components pass over the fault. This short-duration contact is similar to an impact and causes a burst of vibrations at frequencies known as the fault characteristic frequencies (FCFs) that propagate through the bearing housing to be detected elsewhere on the machine. The most basic form of bearing fault diagnostics involves checking vibration signals for repeated impacts at the FCFs.

Several other factors combine to obfuscate the diagnostic information within the signal. Among these is the contamination of the signal by background noise, which is a prevalent issue in industrial applications, where many machines interfere with each others' sensors through common transmission paths. As bearing resonant frequencies tend to be quite high, it is generally easiest to find wide frequency bands dominated by bearing signals in the higher end of the spectrum rather than the lower end, as the lower end of the frequency spectrum is more likely to contain contamination from distant background machinery. Another complicating factor is the cyclical variation of transmission paths within the bearing itself caused by the relative rotation of bearing components. The influence of this amplitude modulation is illustrated in Figure 1.

A basic and powerful tool to simplify signal analysis and reveal the underlying shape of the vibration waveform involves taking the envelope of the signal, which effectively smooths the resonance waves induced during impacts against the fault zone into a single wave. This may be regarded as the effective instantaneous amplitude, and it more clearly reveals the FCF than the raw signal. Figure 1 shows how different fault locations involve different amplitude modulation patterns as the fault moves relative to the sensor.

Frequently changing operating conditions pose a challenge for in situ bearing fault diagnosis. A bearing within a machine may be subject to different speed and loading conditions depending on the machine's mode of operation. Varying speed will cause a change in the FCF being activated by a given fault and a change in loading conditions will change the vibration magnitude and can also change the modulation and attenuation characteristics of the vibration response. This motivates the development of diagnostic algorithms that are unaffected by changes in operating conditions or limits them to work on machines with consistent operating conditions.

Although laboratory test benches attempt to simulate real industrial situations, artificially damaged bearings and gears are imperfect approximations of real damage endured gradually through normal wear processes. The resulting difference in fault geometry

induces a different fault signal during operation. Figure 2 shows test bench measurements from identical bearings under the same loading conditions with a seeded fault introduced to the outer race with electron discharge machining (left) and outer race spalling by natural wear occurring after accelerated life testing (right). The raw signals and frequency spectra are somewhat different in appearance, but the differences are minimized in the envelope spectra.

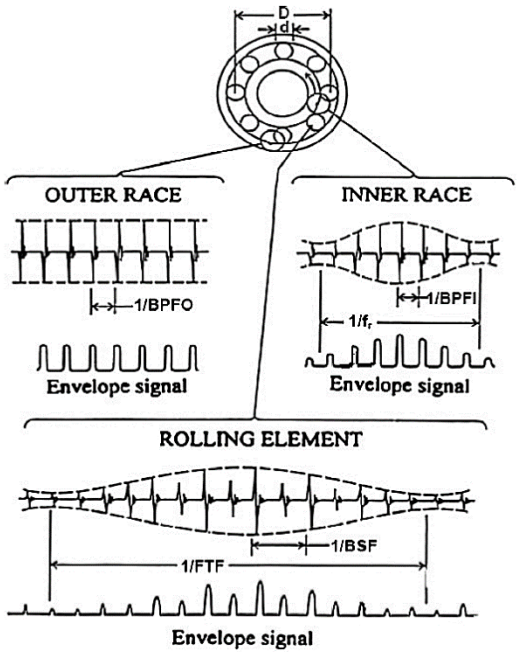

Figure 1. Typical signals and envelope signals from local faults in rolling element bearings [19].

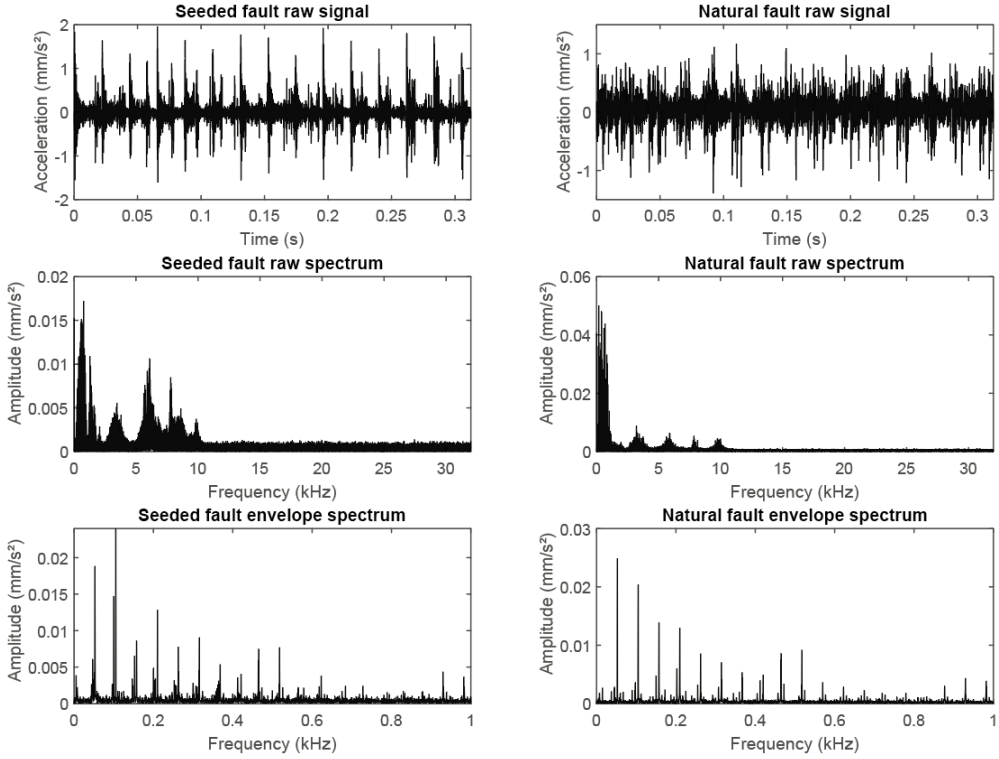

Figure 2. Comparison of natural faults (left column) and real fault signals (right column) with respect to raw signal (top row), raw frequency spectrum (middle row) and envelope spectrum (bottom row) for identical bearings under the same operating conditions.

2.2. Vibration-Based Gear Fault Diagnosis Fundamentals

A useful resource describing gear fault detection using vibration spectral analysis is given by P. D. McFadden [21]. Planetary gearboxes have unique characteristics and are prevalent enough to warrant individual consideration; an informative tutorial is given by Guo et al. [22], which also reviews conventional methods employing time-synchronous averaging and narrowband demodulation.

Unlike bearing vibration signals, which contain no periodic impulses in the healthy state, healthy gear signals contain sharp periodic impulses due to the repeated engagement and disengagement of the meshing teeth. Therefore, gear faults cannot be diagnosed simply by detecting the presence of new periodic fault signals, but must be diagnosed by detecting other changes in the gear signal. The frequency of tooth engagement is known as the gear meshing frequency (GMF) and is calculated as the product of the shaft frequency and the number of teeth on the gear mounted to that shaft.

Gear faults may arise due to several factors, including inadequate lubrication, contamination, installation error, and overloading. These factors may give rise to fatigue pitting on the tooth surfaces or cracks at the base of the tooth where stresses are concentrated. Spectral analysis reveals that sidebands often appear about the GMF and its harmonics when a local tooth defect is present in a gear. In systems with many pairs of meshing gears, it can be very difficult to distinguish the many components in the spectrum, complicating the diagnosis process. Additionally, the presence of sidebands does not always indicate fault severity.

It is not possible to obtain a pure recording of the gear meshing signal. Practical constraints require accelerometers to be placed on the external gearbox housing, this results in a variable transmission path that passes through other gears, shafts, bearings, and the housing itself. The resulting modulation can considerably distort the signal and cause sidebands to arise even for healthy gears.

Figure 3 shows the raw signals and envelope spectra for a two-stage gear reducer when all gears are healthy and when the input gear is chipped.

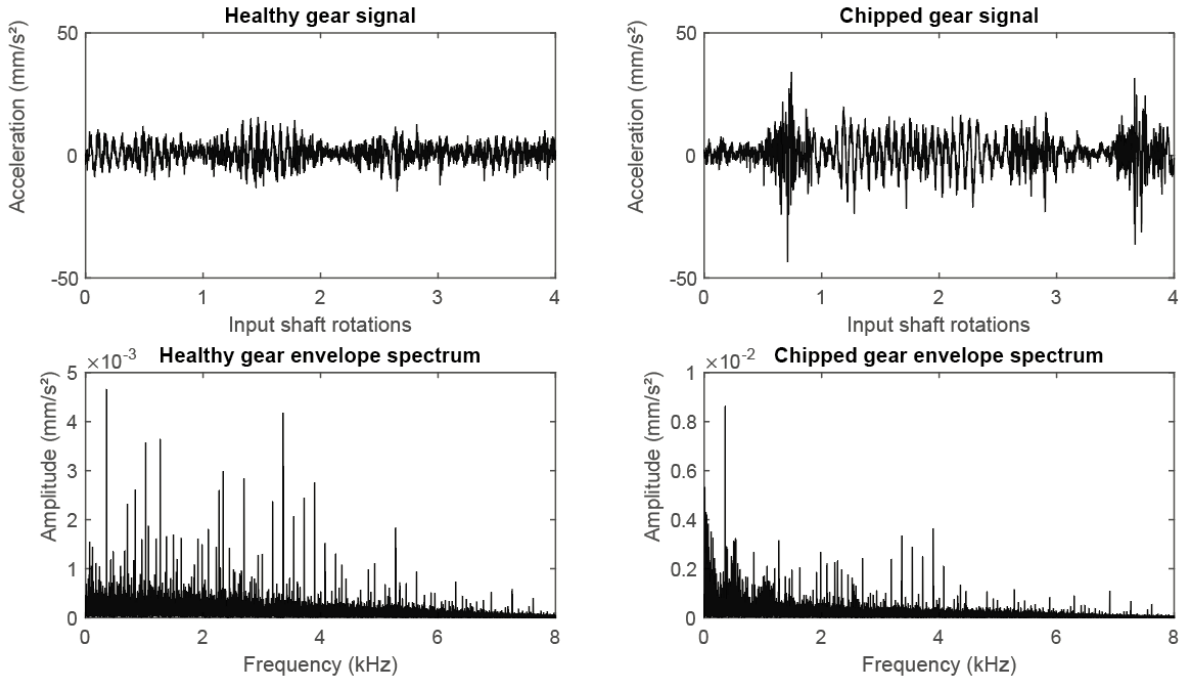

Figure 3. Raw signal (top) and spectrum of the envelope signal (bottom) for a healthy gear train (left) and one with a chipped tooth at the input gear (right).

With respect to gear fault diagnosis, the objective of this paper is to explore whether CNN architectures that are effective for bearing fault diagnosis are also effective for gear fault diagnosis, or whether gear fault diagnosis requires an entirely different configuration of hyperparameters.

2.3. Theory of CNNs

This section gives a general description of how CNNs operate without explaining finer details or providing mathematical formulae. Bouvrie provides useful resources and a detailed discussion of CNNs, along with their mathematical description [23]. Lei et al.'s review of deep learning for machine fault diagnosis also contains a useful visual explanation of the fundamental operations used in CNNs. In general, CNNs consist of three main layer types: convolutional layers, pooling layers, and fully connected layers. A general visual description is presented in Figure 4.

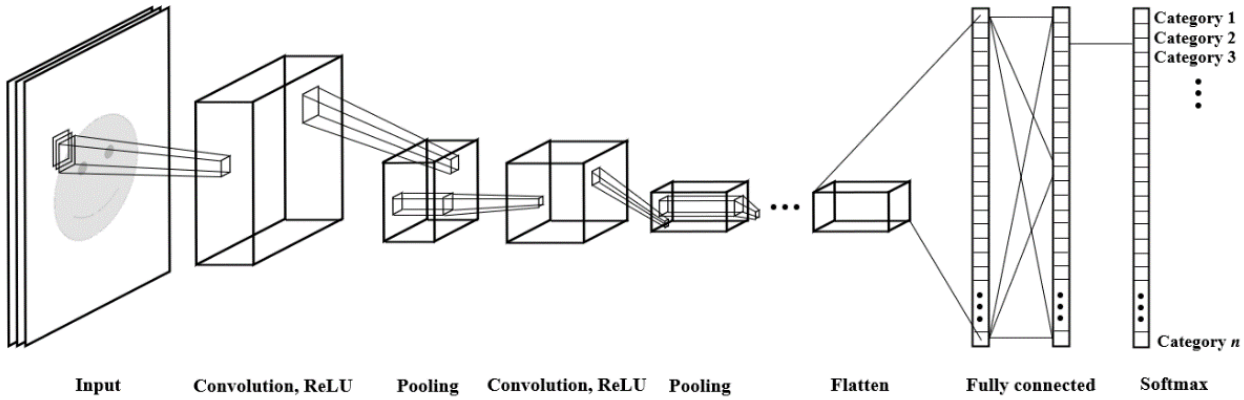

Figure 4. Architecture, or sequence of layers, for a classical CNN classifier.

A convolutional layer contains some number of filter kernels, each of which convolves the previous layer to produce an output called a feature map. The filters may be of any size that fits within the dimensions of the previous layer's output. The convolution operation consists of taking the dot product between a region of the input space and the learnable weights in the kernel. The product is passed through an activation function such as a rectified linear unit (ReLU) and the result is mapped to the corresponding region of the feature map. The filter is swept over the input space using a step size called the stride, taking a dot product at each step to fully populate the feature map. Stride offers CNNs strong translation invariance with respect to the location of features in the input space without the need for many new learnable parameters. A new feature map is generated by each kernel in the convolutional layer. The pooling operation is a downsampling process that reduces the number of training parameters while preserving important information about the input. The pooling layer also involves a filter of some size that is swept over the previous layer at increments determined by its stride. Common pooling operations are max pooling and mean pooling, which keep the maximum and mean of the input, respectively. Using pooling layers after convolutional layers reduces training time, as well as overfitting.

Several repetitions of convolution and pooling layers allow a CNN to learn complex features from the input data. The features can then be flattened into a 1D layer and used as the input to at least one fully connected layer through which the features are mapped to the target class. For supervised learning of machine health states, softmax classification is usually used for the last layer. The parameters within the convolutional kernels and fully connected layers are learned during training with a gradient descent algorithm, as with most neural network training algorithms. The configuration of the CNN, including the sequence of layers, number and size of filters in each layer, learning rate, stride, and other hyperparameters, are hand picked by the network designer and constitute a fully defined CNN architecture.

2.4. Data Augmentation

Data augmentation is a critical tool in machine learning used when the available training data is limited or when it is necessary to improve the network's invariance to certain types of transformations. Data augmentation effectively expands the original training dataset by performing some transformation to the training samples and yields a greater number of unique new training samples. The transformation used must alter the original data in a way that makes it distinguishable from other transformed data, but without totally obscuring the underlying factors of variation. The class labels are usually preserved during data augmentation. For image classification datasets, augmentation techniques include cropping, rotating, and obscuring random subregions of an image, which improve translational and rotational invariance as well as overall generalization. Each of these transformations can involve randomized parameters to allow multiple new images to be generated from each original image.

For the present work, a sliding window is used to extract many short-duration training samples from each original experimental measurement in the time domain. Preprocessing transformations, such as the fast Fourier transform (FFT), are then performed as needed on the extracted sample. This method has several benefits: it is simple and easily reproduced, it offers a many-fold increase in available training samples, it dramatically reduces the dimensionality of the training samples, and it promotes translational invariance in the resulting trained network. The data augmentation technique is illustrated in Figure 5. A sliding window of some prescribed length is used to extract a sample from the original experimental measurement. The window is advanced in time by a set increment to extract the next training sample. Some overlap between windows is provisioned to increase the yield of new training samples, improving translational invariance.

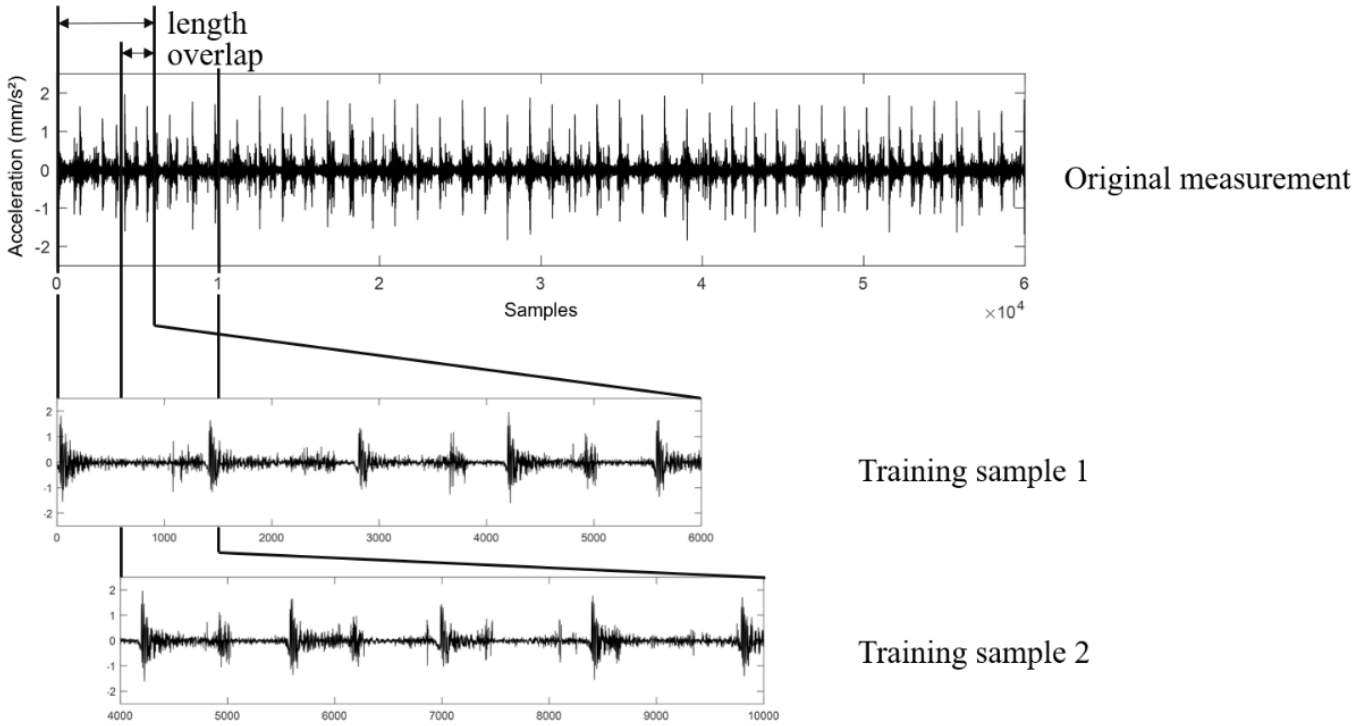

Figure 5. Sliding window using the overlap data augmentation technique with the first two extracted training samples shown.

2.5. CNN Input Types

Since CNNs can theoretically extract useful features directly from raw data, very good accuracy should be achievable without using complex data preprocessing. In practice, however, strong performance may require manipulation of the observation data, especially in cases where data is initially of high dimensionality and the availability of training data is limited. One important cause for this is related to a phenomenon known as "the curse of dimensionality". When the number of dimensions increase, the volume of occupied space increases more quickly and the distance between available datapoints becomes much greater, leading to a sparse dataset. Reducing the number of dimensions used to represent the data while preserving factors of variation that maintain a statistically significant relationship between the observations and their labels (e.g., the health states of the present application) can make it far easier for machine learning algorithms to learn features from the data and overcome sparsity.

Reducing dimensionality is not the only objective of preprocessing. Useful transformations to the raw data can highlight or accentuate explanatory factors, making classification easier. A common example of this is the Fourier transform, which is very useful when the observed data is a time-series measurement, but obvious explanatory factors are present in the frequency spectrum. Transformations may be selected by the algorithm designer using domain-specific expertise, or they may be performed by generic data reduction operations such as principal component analysis or autoencoders. This study is focused on common expert-chosen transformations, since they preserve spatial patterns that will be detectable by the CNN.

The following subsections introduce each of the input types that are used to train CNNs in each of the case studies. The aim is to provide the reader with an intuitive understanding of the transformations used without using detailed mathematical descriptions.

2.5.1. Raw Time Input

As the simplest possible input type, raw time provides only a few parameters that may be chosen by the algorithm designer to characterize this 1D input. Chiefly, one must consider the length or duration of the raw signal that is used as an input feature. In general, a longer signal duration is more likely to contain useful diagnostic information. If a signal is too short, it may not contain enough instances of periodic fault-induced emissions to establish a high probability of those emissions being the result of a fault-related pattern. However, increasing the signal duration also increases the computational power required to train the network. It also increases the probability of the network becoming overfit.

Another motivation for using a shorter window arises when training samples are extracted from a longer experimental measurement via moving window data augmentation. Shorter windows allow more unique training sample to be extracted from the available measured data.

When machine-mounted accelerometers operate at higher sample rates, more data is generated to describe the machine's vibrations over a given span of time. The two signals shown in Figure 6. show how a different number of samples are needed to span a full shaft rotation, and how some fault related signals are manifested over multiple full rotations.

Generally, the resonant frequencies of bearing components are known, so it is possible to determine the upper bound beyond which it is no longer useful to increase the sample rate. However, for this work, no re-sampling of the signals is performed; all benchmark signals will be considered with their native sampling frequencies as stated in each case study.

The experiments conducted here include a study of the influence of input duration against performance in a range of different CNN configurations to determine when it may be advantageous to use a longer or shorter input.

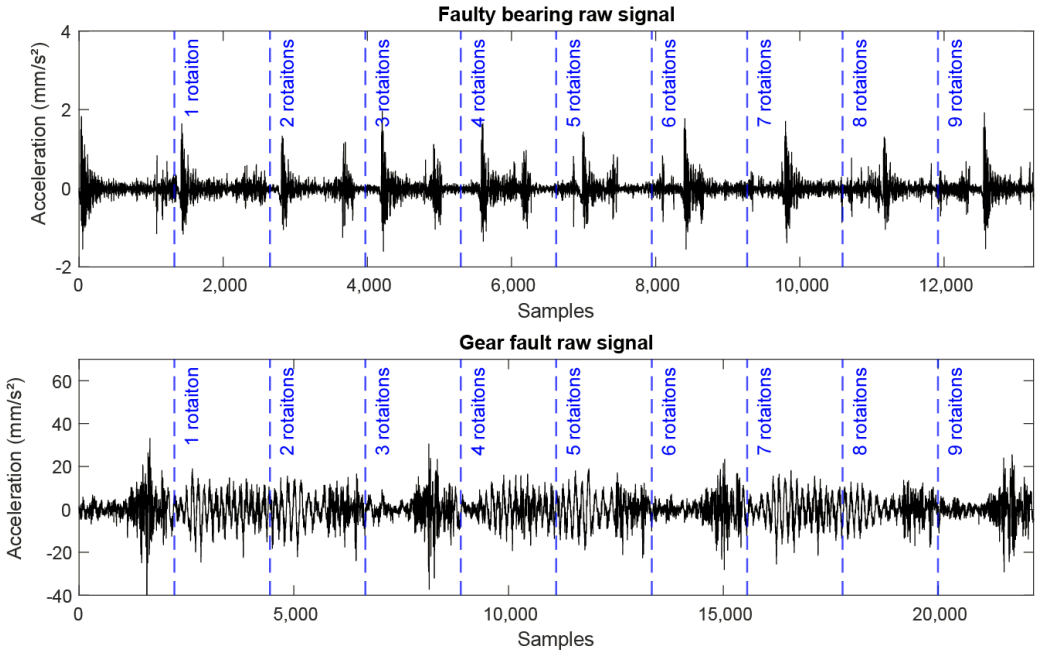

Figure 6. Raw vibration signal for an outer race bearing fault operating at 2900 RPM and sampled at 64 kHz (top) and a chipped tooth fault and a compound fault gear running at 1800 RPM and sampled at 66.6667 kHz.

2.5.2. Frequency Spectrum

Data augmentation is performed prior to using the fast Fourier transform (FFT) to obtain the 1D frequency spectrum samples. The absolute value of the single-sided spectrum is kept. Since a no low-pass filter is used, the upper bound of the spectrum is defined by the Nyquist frequency, itself simply determined by halving the sample rate. The frequency resolution achieved is dependent on the duration of the signal extracted during data augmentation. In this case, the resulting input vector will contain half the number of elements contained within the original time domain window.

The overlapping sliding windows used during data augmentation ensure that each spectrum obtained contains variations of the original signal. As with using a raw time input, the probability that a successful diagnosis can be performed on any given spectrum will depend on whether a window coincides with transient fault-induced signals; a very short window has a low probability of containing such information. Figure 7 shows two such spectra obtained from the same experimental measurement with a 75% overlap. The first sample (bottom left) coincides with an instance of a ball passing over an outer race fault, resulting in a spectrum that more clearly indicates the presence of the fault. Assuming the original measurement is stationary, extracted spectrums become increasingly similar to the spectrum of the whole measurement as the window size is made larger. Therefore, using longer windows to extract data results in a more homogeneous dataset.

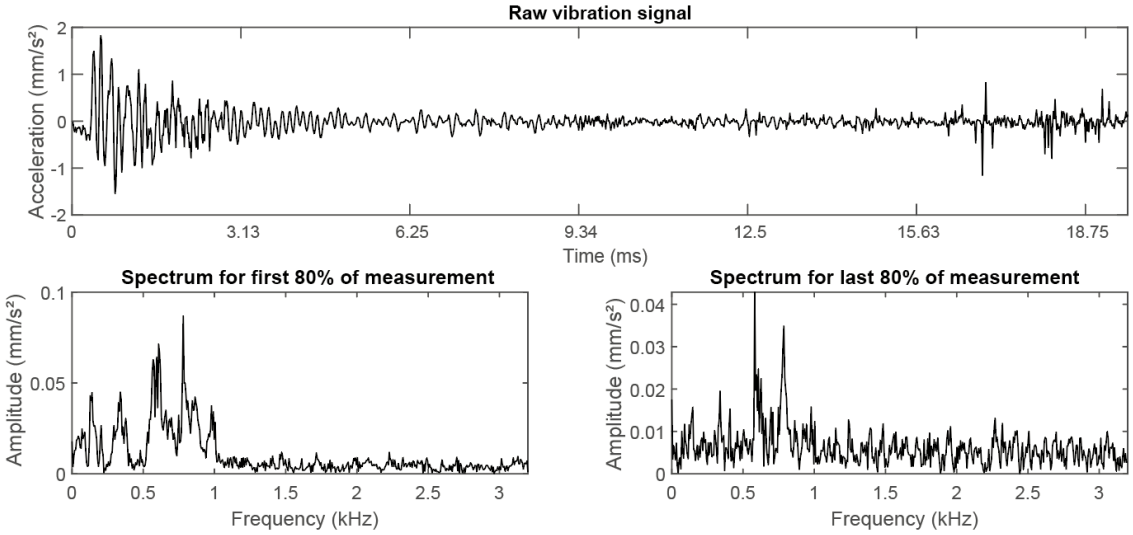

Figure 7. Raw vibration measurement from a bearing fault test bench with two frequency spectra obtained from windows of 1000 samples with 75% overlap.

2.5.3. Envelope Spectrum

The envelope transformation simplifies a signal to reveal the overall shape of the signal. It emphasizes lower frequency modulating signal elements such as low frequency fault related impacts. As a result, the spectrum of the envelope signal can more easily outline FCFs and their harmonics. This eliminates information about the bearing component's resonant frequencies, that may not be useful for the model's ability to learn simpler and more distinct patterns. The transformation can yield an upper and lower envelope. Since vibration signals tend to be symmetric, the upper and lower envelopes will have very similar frequency spectra. This work uses a standard MATLAB function (envspectrum(x,fs)) where x is the signal and fs is the sample rate in Hz) to obtain the envelope spectrum.

Figure 8 illustrates the transformation in multiple steps: from the original raw signal to the upper envelope signal to the envelope spectrum. The spectrum of the original signal is also shown for comparison. The upper envelope spectrum clearly illustrates the fault characteristic frequency with its harmonics, whereas high frequency vibrations around bearing component resonance frequencies dominate the raw spectrum.

2.5.4. Short Time Fourier Transform Spectrogram

The spectrogram provides a very useful time–frequency representation of the signal; it includes the benefits of spectral analysis as well as time localization of frequency components. The spectrogram is obtained by dividing the original signal into many shorter windows and taking the FFT of each one in sequence. The result is a 2D matrix with time progression along one direction and the frequency scale along the other.

Two parameters used in defining the transformation allow some flexibility in tailoring the spectrogram to suit a given application: window length and overlap. The window length is the result of the original signal length divided by the number of desired samples prior to taking the FFT. A shorter window provides finer division of the signal in time, and thus improves time localization, increasing the time resolution of the spectrogram. The penalty for increased time resolution is decreased frequency resolution due to the innate restrictions of the Fourier transform. The frequency resolution is proportional to the size of the windowed signal and is thus improved by increasing the window length. This creates a tradeoff between time and frequency resolution. The overlap used in obtaining the spectrogram lends some improvement to the time resolution by fitting more windows into the signal, with some samples shared by adjacent windows. For simplicity, and to avoid training CNNs with very large inputs, no overlap is used in this study.

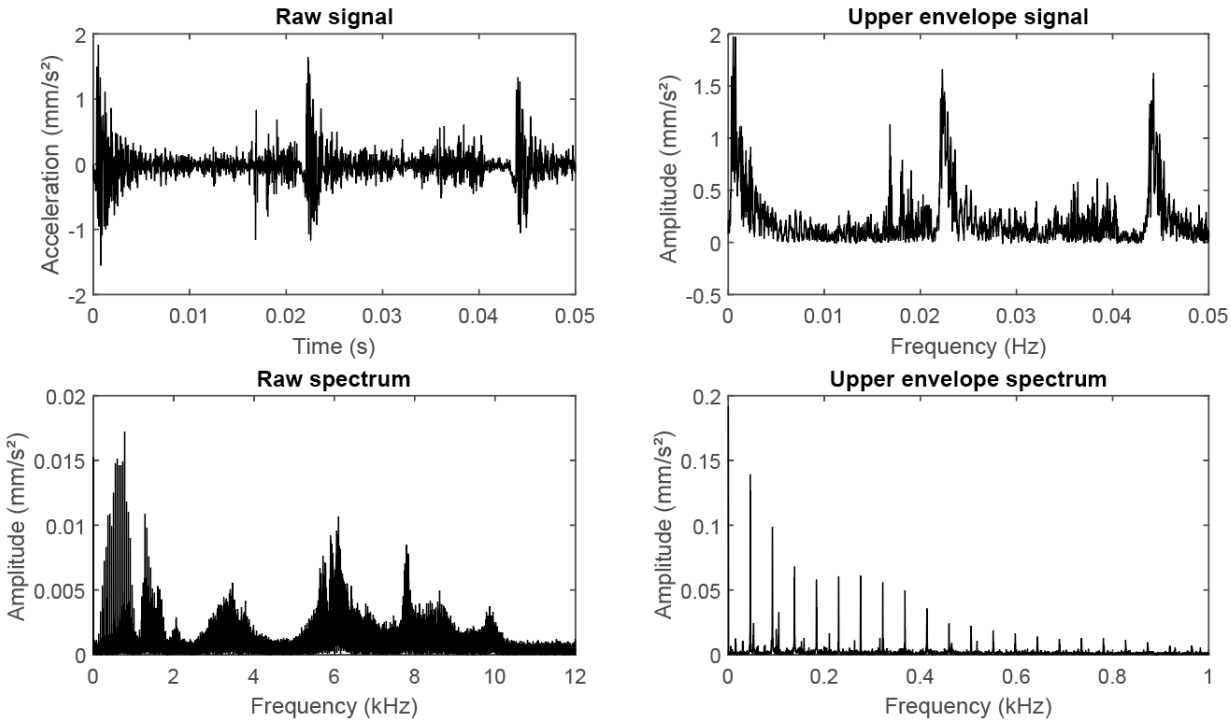

Figure 8. Comparison of raw vibration signal, envelope signal, raw spectrum, and envelope spectrum for an outer race fault sampled at 64 kHz from the Padderborn University dataset.

3. Network Architectures

This paper aims to investigate the relationship between the basic CNN hyperparameters and the choice of signal preprocessing method. Therefore, the same sequence of layers is used in all experiments with experimental variables that include input type, input size, convolutional kernel size and stride. All network configurations use two convolutional layers, each one followed by a ReLU layer, a 50% dropout layer, and a max pooling layer. Additional layers were considered, but a limit of two was selected due to the observed overfitting even in shallow networks. As the number of layers increases, deeper features can be extracted, but the increased number of parameters require a larger training dataset to avoid overfitting, and none of the available open access datasets are of adequate size to allow for the proper training of many-layered CNNs.

The size of the convolutional kernel is varied to investigate its influence. The same kernel size is used in the first and second layers for simplicity and to reduce the number of experimental network configurations. Max pooling for 1D inputs use a 2×1 filter and a 2×1 stride while max pooling for 2D inputs use a 2×2 filter and a 2×2 stride. All convolutional layers have 16 filters. The output from the second max pooling layer is flattened and followed by a fully connected layer with 7 output neurons. A softmax layer is used to classify the output. Here, network configurations are simply denoted by the convolutional kernel size and stride and the input size used. As an example, Table 1 details the network architecture for a network with kernel size of 8×1, stride of 2×1, and input size of 1248×1.

Table 1. Example CNN specifications for a 8 × 1 filter size, a 2 × 1 stride, and a 1248 × 1 input.

#	Layer Type	Activations	Learnables
1	Input layer	1248 × 1 × 1	-
2	Convolutional layer	624 × 1 × 16	Weights 8 × 1 × 1 × 16 Bias 1 × 1 × 16
3	ReLU	624 × 1 × 16	-
4	Dropout (50%)	624 × 1 × 16	-
5	Max pooling	312 × 1 × 16	-
6	Convolutional layer	312 × 1 × 16	Weights 8 × 1 × 16 × 16 Bias 1 × 1 × 16
7	ReLU	312 × 1 × 16	-
8	Dropout (50%)	312 × 1 × 16	-
9	Max pooling	312 × 1 × 16	-
10	Fully connected	1 × 1 × 7	Weights 7 × 4992 Bias 7 × 1
11	Softmax	1 × 1 × 7	-
12	Class output	-	-

In addition to the network architecture, the parameters for training a CNN greatly influence the final accuracy. Unless otherwise indicated, the same training parameters are used across all case studies. Training is ended after eight epochs for Case Study 1, 3 epochs for Case Study 2, and 40 epochs for Case Study 3, since this was observed to provide network convergence across all data types and network configurations. All configurations are trained using the ADAM optimizer for gradient descent. 1D CNNs are trained with an initial learning rate of 0.01, while 2D CNNs are trained with an initial learning rate of 0.001. The number of training samples used for each configuration varies depending on the signal duration used in any given configuration. The duration of the original signal, as well as the dimensions of the processed input data, are presented along with the accuracies for each configuration.

4. Case Study 1: Case Western Reserve University Bearing Fault Dataset

This case study explores the use of CNNs for identifying healthy bearings and bearings having various fault types. Key parameters for preparing the training data and CNN hyperparameters are varied to identify important trends. The objective here is to identify which variables strongly influence diagnostic accuracy and to identify the most accurate combination of input type and CNN configuration.

4.1. Dataset Description

The Case Western Reserve University (CWRU) dataset [24] is commonly used to benchmark algorithms for bearing fault diagnosis [25]. The dataset contains vibration measurements from multiple accelerometers mounted on an electric motor containing bearings in various states of health. Bearing faults were artificially introduced using electro-discharge machining. Different fault types were simulated separately at the inner raceway, outer raceway, or one of the rolling elements. Damaged bearings were installed onto the fan end or drive end positions of the motor's shaft. Thus, seven fault states are created; one for which all bearings are healthy, and three different damage states for each of the two bearing locations. Using these seven states as labels, the trained CNN shall have to accurately diagnose both the fault type and fault location for a prediction to be counted as correct by the error function.

The motor was operated with loads of 0 to 3 HP and speeds ranging from 1720 to 1797 rpm. Three fault severities were simulated by machining defects of different sizes

into the bearings. Vibration data was sampled at 12 kHz for a duration of 10 seconds for each configuration of motor load, fault severity, and fault type. Additional recordings were conducted with outer race faults in which the bearing was installed by placing the static fault at different locations relative to the accelerometer. That is directly below the accelerometer, orthogonal to the accelerometer, or opposite from the accelerometer.

4.2. Data Preparation

Data from different runs of the test bench are divided between training and testing groups prior to performing data augmentation. This ensures an accurate appraisal of the testing accuracy of the trained networks, data from the training set must not overlap with data from the test set, including by being obtained from the same damaged bearing specimen. K-fold cross-validation is used to verify results with k = 4. Thus, each network configuration is trained with one quarter of the original experimental data. Each testing and validation accuracy is the average obtained over the k trained CNNs.

Though the experimental data contains two channels of vibration data, data from only one accelerometer is used. The fault type and faulty bearing location are used as labels, for a total of seven labels. The three different fault severities and four different loading conditions are lumped into each of these seven labels. The goal of the network is to diagnose the fault type and location irrespective of these factors.

The data augmentation method described in Section 2.4 is used for all experiments, with 25% overlap. Six input lengths are chosen, using multiples of the number of samples needed to span a full shaft rotation at the given sample rate and average rotation rate over all experiments. Prior to training, the data is normalized to have a mean of zero and standard deviation of one.

4.3. Results and Discussion

Tables 2–5 show the average training and testing accuracies taken over 4-fold cross-validation for various network input sizes, convolutional filter sizes and strides. The results obtained cover a broad range of accuracies, highlighting significant sensitivity to changing hyperparameters. The best accuracy obtained here (80.1%) still falls short of some other researchers' findings using shallow classical CNNs. This seems likely to be a result of their allowing samples extracted from a given experimental run to exist in both the training and testing dataset.

Table 2. Training and testing accuracies for various CNN architectures with no preprocessing.

Input Type: Raw Time Series													
Training Accuracy: (%)							**Testing Accuracy: (%)**						
Kernel Size, Stride							Kernel Size, Stride						
[4,1], [2,1]	72.7	68.8	64.8	71.9	68.0	72.7	[4,1], [2,1]	50.9	49.4	47.7	49.7	41.2	33.5
[8,1], [2,1]	82.0	86.7	83.6	59.4	65.6	49.2	[8,1], [2,1]	62.6	58.8	57.2	43.2	40.6	31.8
[16,1], [4,1]	89.1	86.7	89.8	82.8	83.6	67.2	[16,1], [4,1]	65.0	66.7	64.4	54.6	58.3	39.4
[32,1], [4,1]	93.8	87.5	97.7	86.7	41.4	46.1	[32,1], [4,1]	66.9	67.5	65.3	63.3	33.1	35.0
[64,1], [4,1]	89.8	**93.8**	96.1	83.6	56.3	50.8	[65,1], [4,1]	65.9	**71.2**	63.3	53.1	37.4	36.6
[128,1], [1,1]	81.3	61.7	54.7	61.7	42.2	31.3	[128,1], [4,1]	60.0	48.2	50.8	47.7	35.9	27.0
[256,1], [4,1]	89.8	87.5	68.8	40.6	28.9	34.4	[256,1], [4,1]	63.1	56.7	49.4	33.1	29.3	35.4
Input length:	416	832	1248	2496	4992	9984	Input length:	416	832	1248	2496	4992	9984
Original signal length:	416	832	1248	2496	4992	9984	Original signal length:	416	832	1248	2496	4992	9984

Table 3. Training and testing accuracies for various CNN architectures with FFT preprocessing.

Input Type: Fourier Spectrum													
Training Accuracy: (%)							**Testing Accuracy: (%)**						
Kernel Size, Stride							Kernel Size, Stride						
[4,1] [2,1]	96.9	97.7	**98.4**	100.0	99.2	99.2	[4,1] [2,1]	75.4	74.3	**79.1**	71.4	75.6	66.5
[8,1] [2,1]	99.2	96.9	98.4	99.2	97.7	97.7	[8,1] [2,1]	71.4	71.9	76.1	71.9	70.6	68.4
[16,1] [4,1]	85.9	88.3	93.0	93.8	94.5	69.5	[16,1] [4,1]	68.3	69.0	75.1	74.2	68.5	50.5
[32,1] [4,1]	71.9	88.3	73.4	82.0	79.7	22.7	[32,1] [4,1]	56.2	65.2	53.9	56.8	62.7	25.1
[64,1] [4,1]	40.6	45.3	39.1	45.3	34.4	23.4	[64,1] [4,1]	32.3	32.6	37.0	34.9	30.3	25.1
[128,1] [4,1]	52.3	48.4	47.7	30.5	30.5	20.3	[128,1] [4,1]	45.7	40.7	34.9	25.6	30.1	25.1
[256,1] [4,1]	42.2	39.8	32.0	38.3	31.3	27.3	[256,1] [4,1]	37.0	34.2	26.8	34.1	25.2	25.2
Input length:	208	416	832	1248	2496	4992	Input length:	208	416	832	1248	2496	4992
Original signal length:	416	832	1248	2496	4992	9984	Original signal length:	416	832	1248	2496	4992	9984

The tables are color-formatted to highlight higher accuracies in green with lower accuracies in red. The highest testing accuracy for each table is emboldened, along with its corresponding training accuracy.

Evidently, different input types favor different hyperparameters for maximizing validation accuracy. Using raw temporal measurements appears best paired with mid-sized kernels and a smaller input space. Using FFT preprocessing achieves poor accuracy if large kernels are used and does not show a strong dependence on input size, whereas envelope spectrum preprocessing works best with the largest kernel and mid-sized input. Using the spectrogram input type provides the most consistently strong diagnostic accuracy, with less sensitivity to changes in input size and kernel size and stride than other input types.

Table 4. Training and testing accuracies for various CNN architectures with envelope spectrum preprocessing.

Input Type: Envelope Spectrum													
Training Accuracy: (%)							**Testing Accuracy: (%)**						
Kernel Size, Stride							Kernel Size, Stride						
[4,1] [2,1]	71.1	78.1	82.8	83.6	78.1	64.8	[4,1] [2,1]	59.4	68.1	61.3	62.9	67.6	55.7
[8,1] [2,1]	78.1	85.9	84.4	88.3	82.0	65.6	[8,1] [2,1]	53.9	67.2	64.0	72.0	67.0	49.9
[16,1] [4,1]	61.5	78.1	81.3	83.6	82.0	78.1	[16,1] [4,1]	68.0	67.8	67.6	58.5	65.1	59.7
[32,1] [4,1]	79.7	85.2	91.4	87.5	82.0	72.7	[32,1] [4,1]	65.5	67.5	71.0	66.0	59.7	53.9
[64,1] [4,1]	78.9	90.6	93.0	86.7	82.8	63.3	[64,1] [4,1]	66.9	74.6	72.4	66.9	67.5	47.0
[128,1] [4,1]	75.0	94.5	93.0	95.3	89.1	80.5	[128,1] [4,1]	67.7	73.4	75.7	67.7	68.6	58.5
[256,1] [4,1]	87.5	92.2	96.1	**95.3**	93.0	87.5	[256,1] [4,1]	68.8	76.9	76.7	**78.0**	64.5	66.7
Input length:	208	416	832	1248	2496	4992	Input length:	208	416	832	1248	2496	4992
Original signal length:	416	832	1248	2496	4992	9984	Original signal length:	416	832	1248	2496	4992	9984

Table 5. Training and testing accuracies for various CNN architectures with spectrogram preprocessing.

Input Type: Spectrogram (Window = 104)													
Training Accuracy: (%)							**Testing Accuracy: (%)**						
Kernel Size, Stride							Kernel Size, Stride						
[2,2], [1,1]	100.0	100.0	100.0	100.0	100.0	100.0	[2,2], [1,1]	80.0	76.1	70.5	75.1	75.5	70.9
[4,4], [1,1]	100.0	100.0	100.0	100.0	100.0	100.0	[4,4], [1,1]	75.8	73.7	71.8	71.1	74.8	66.7
[6,6], [1,1]	100.0	**100.0**	100.0	100.0	100.0	100.0	[6,6], [1,1]	80.0	**80.1**	71.2	74.6	71.2	64.7
[8,8], [1,1]	100.0	99.2	100.0	100.0	61.7	53.1	[8,8], [1,1]	75.1	73.2	76.9	71.7	69.7	67.9
Input size:	129 × 7	129 × 15	129 × 23	129 × 47	129 × 95	129 × 191	Input length:	129 × 7	129 × 15	129 × 23	129 × 47	129 × 95	129 × 191
Original signal length:	416	832	1248	2496	4992	9984	Original signal length:	416	832	1248	2496	4992	9984

Testing accuracy is consistently lower than training accuracy, especially for longer input durations, suggesting that overfitting remains a significant problem despite the use of dropout. This might be addressed with a more extensive and more sophisticated data augmentation method. If only the strongest configurations for each input type are taken, the Fourier spectrum, envelope spectrum, and spectrogram seem to perform approximately equally.

Figure 9 shows confusion matrices from the best performing configurations under each input type. Since 4-fold cross-validation is used, four confusion matrices can be produced from each configuration. The summation of these four confusion matrices are presented for each input type.

Considering that practical users of diagnosis algorithms may not be concerned with the fault type and simply need to know whether the bearing is healthy or needs replacing, the best preprocessor appears to be the envelope spectrum. The envelope spectrum preprocessor resulted in no instances of misclassified healthy bearings, nor any faulty bearings falsely classified as healthy.

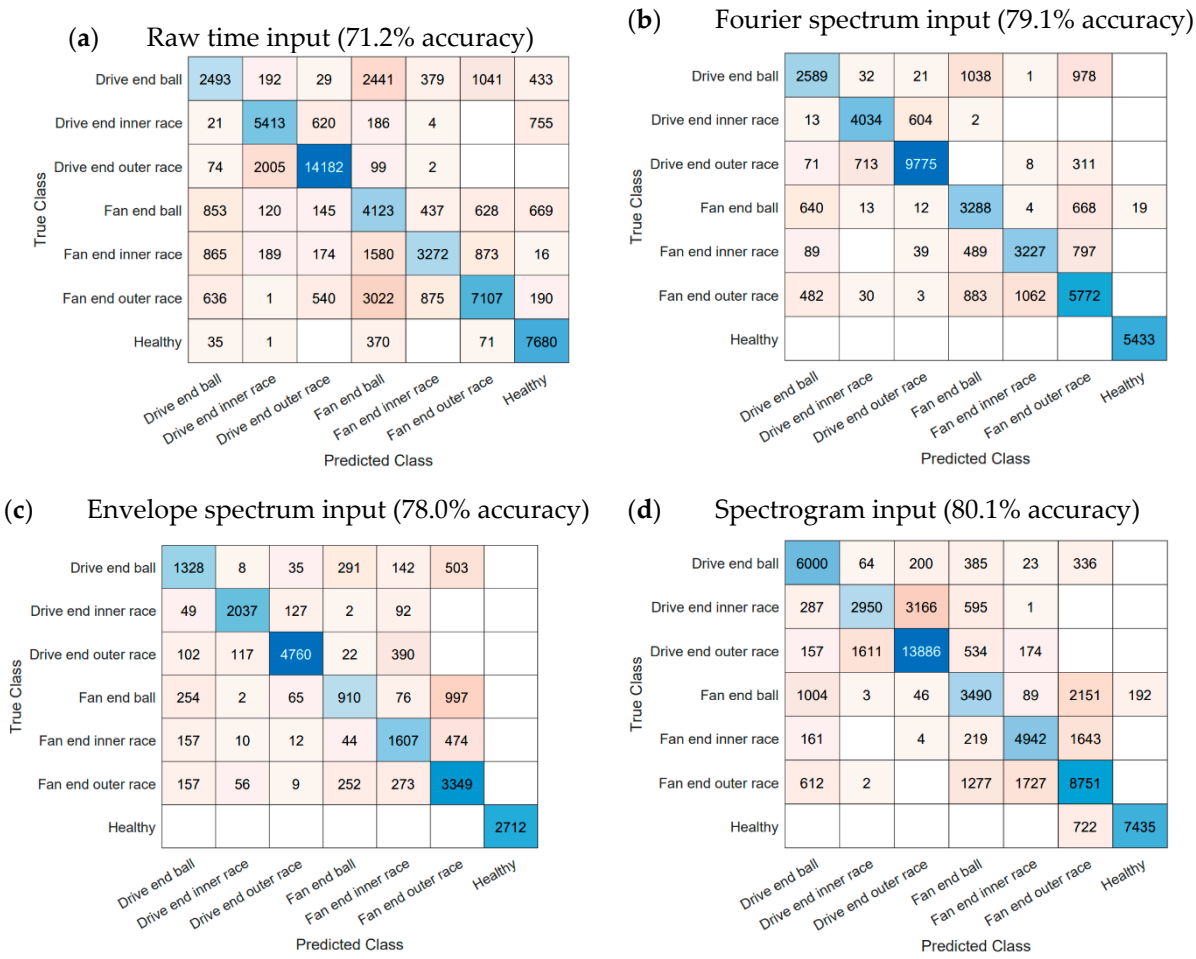

Figure 9. Confusion matrices for (**a**) raw time, (**b**) Fourier spectrum, (**c**) envelope spectrum, and (**d**) spectrogram as inputs to the CNN.

5. Case Study 2: Paderborn University Bearing Fault Dataset

This case study contains two parts. In part 1, the same CNN configurations used in Section 4 are trained with data from artificially damaged bearings and tested with bearings with real damages gained during accelerated lifetime testing. In part 2, these CNN configurations are again used with training and testing datasets both originating from artificially damaged bearing experiments. The purpose of part 1 is to evaluate the cross-domain applicability of the studied CNN configurations in a situation that reflects a real scenario. Part 2 aims to determine the extent to which the inaccuracy observed in part 1 can be attributed to the domain difference. This case study will also reveal whether the trends in hyperparameter selection for each input type are consistent across these two parts.

5.1. Dataset Description

The Paderborn University bearing dataset [26] is another popular benchmarking dataset used for bearing fault diagnosis algorithms. An important differentiating characteristic of this dataset is that it includes bearings with seeded faults as well as bearings with natural faults. Three different methods are used to simulate bearing damage: electric discharge machining, drilling (various diameters), and electric engraving. Outer race and inner race fault types are studied. Training with seeded faults and testing with natural faults provides an analogue for situations when algorithms developed on lab data are deployed in industry where only natural faults exist.

5.2. Data Preparation

For part 1, the faults in the testing set are naturally developed during an accelerated lifetime test before being measured. For part 1, the dataset is split into training and testing with the same scheme as used Chen et al. [11] so that the performance of various traditional CNNs can be directly compared to that of their novel CNN architecture. In the second part, training and testing are both done with bearings having artificial damage. Tables 6 and 7 indicate which experimental runs are included in each dataset for the two parts. The data augmentation procedure described for Case Study 1 is reused for Case Study 2 with the same overlap and input durations. Since the experiments in this dataset use different rotational speeds and sample rates, the input durations used do not correspond with integer numbers of shaft rotations here. However, the reuse of the same input durations allows for direct comparison of the same CNN configurations between datasets.

Table 6. Division of experimental data between training and testing datasets for Case Study 2 part 1.

Dataset	Class	Fault Origin	Bearing code
Training	Healthy	None	K002
Training	IR damage	Artificial damage	KI01
Training	IR damage	Artificial damage (electric engraver)	KI05
Training	IR damage	Artificial damage (electric engraver)	KI07
Training	OR damage	Artificial damage (EDM machining)	KA01
Training	OR damage	Artificial damage (electric engraver)	KA05
Training	OR damage	Artificial damage (drilled)	KA07
Testing	Healthy	None	KA001
Testing	IR damage	Overload, wrong viscosity, contamination	KI14
Testing	IR damage	Overload, wrong viscosity, contamination	KI16
Testing	IR damage	Overload, wrong viscosity, contamination	KI17
Testing	IR damage	Overload, wrong viscosity, contamination	KI18
Testing	IR damage	Overload, wrong viscosity, contamination	KI21
Testing	OR damage	Overload, wrong viscosity, contamination	KA04
Testing	OR damage	Overload, wrong viscosity, contamination	KA15
Testing	OR damage	Overload, wrong viscosity, contamination	KA16
Testing	OR damage	Overload, wrong viscosity, contamination	KA22
Testing	OR damage	Overload, wrong viscosity, contamination	KA30

Table 7. Division of experimental data between training and testing datasets for Case Study 2 part 2.

Dataset	Class	Fault Origin	Bearing code
Training	Healthy	None	K004
Training	Healthy	None	K005
Training	Healthy	None	K006
Training	IR damage	Artificial damage	KI01
Training	IR damage	Artificial damage (electric engraver)	KI03
Training	IR damage	Artificial damage (electric engraver)	KI05
Training	OR damage	Artificial damage (electric engraver)	KA06
Training	OR damage	Artificial damage (drilled)	KA07
Training	OR damage	Artificial damage (drilled)	KA08
Testing	Healthy	None	K001
Testing	Healthy	None	K002
Testing	Healthy	None	K003

Table 7. *Cont.*

Dataset	Class	Fault Origin	Bearing code
Testing	IR damage	Artificial damage (electric engraver)	KI07
Testing	IR damage	Artificial damage (electric engraver)	KI08
Testing	OR damage	Artificial damage (EDM machining)	KA01
Testing	OR damage	Artificial damage (electric engraving)	KA03
Testing	OR damage	Artificial damage (electric engraving)	KA05

5.3. Results and Discussion

5.3.1. Part 1: Artificial to Natural Damage

The results in Tables 8–11 show the accuracy of the previously studied CNN configurations for the cross-domain task described above.

Table 8. Training and testing accuracies for various CNN architectures with no preprocessing.

Input Type: Raw Time Series													
	Training Accuracy: (%)							**Testing Accuracy: (%)**					
Kernel Size, Stride							Kernel Size, Stride						
[4,1], [2,1]	62.5	68.8	87.5	81.3	84.4	87.5	[4,1], [2,1]	47.0	46.1	47.1	37.8	43.5	35.9
[8,1], [2,1]	81.3	75.0	81.3	78.1	84.4	90.6	[8,1], [2,1]	49.7	46.6	49.3	48.6	48.0	31.5
[16,1], [4,1]	81.3	93.8	87.5	81.3	84.4	93.8	[16,1], [4,1]	49.6	50.2	51.7	49.5	52.2	52.5
[32,1], [4,1]	75.0	75.0	87.5	93.8	87.5	87.5	[32,1], [4,1]	47.7	49.0	48.7	49.4	51.5	52.7
[64,1], [4,1]	84.4	84.4	81.3	100.0	**100.0**	93.8	[64,1], [4,1]	47.9	49.2	49.1	49.4	**55.0**	50.0
[128,1], [4,1]	81.3	87.5	93.8	100.0	100.0	100.0	[128,1], [4,1]	45.7	45.5	46.9	46.6	47.9	51.8
[256,1], [4,1]	75.0	84.4	81.3	96.9	90.6	100.0	[256,1], [4,1]	45.8	45.4	48.7	49.0	48.9	49.7
Input length:	416	832	1248	2496	4992	9984	Input length:	416	832	1248	2496	4992	9984
Original signal length:	416	832	1248	2496	4992	9984	Original signal length:	416	832	1248	2496	4992	9984

Table 9. Training and testing accuracies for various CNN architectures with FFT preprocessing.

Input Type: Fourier Spectrum													
	Training Accuracy: (%)							**Testing Accuracy: (%)**					
Kernel Size, Stride							Kernel Size, Stride						
[4,1] [2,1]	75.0	93.8	87.5	90.6	90.6	90.6	[4,1] [2,1]	37.8	39.0	41.8	38.9	39.1	41.3
[8,1] [2,1]	75.0	90.6	90.6	90.6	93.8	93.8	[8,1] [2,1]	38.0	36.3	40.5	43.3	37.4	43.1
[16,1] [4,1]	68.8	84.4	84.4	96.9	96.9	96.9	[16,1] [4,1]	36.9	42.8	40.7	34.7	48.4	44.0
[32,1] [4,1]	78.1	84.4	84.4	96.9	87.5	90.6	[32,1] [4,1]	37.3	35.2	42.1	41.0	48.3	46.9
[64,1] [4,1]	75.0	87.5	90.6	93.8	84.4	**90.6**	[64,1] [4,1]	38.2	43.4	40.7	39.5	32.3	**55.8**
[128,1] [4,1]	84.4	87.5	75.0	84.4	90.6	87.5	[128,1] [4,1]	40.2	41.1	42.3	41.6	39.7	41.0
[256,1] [4,1]	90.6	78.1	81.3	87.5	90.6	75.0	[256,1] [4,1]	37.8	42.2	37.3	41.4	40.8	50.5
Input length:	208	416	832	1248	2496	4992	Input length:	208	416	832	1248	2496	4992
Original signal length:	416	832	1248	2496	4992	9984	Original signal length:	416	832	1248	2496	4992	9984

Table 10. Training and testing accuracies for various CNN architectures with envelope spectrum preprocessing.

Input Type: Envelope Spectrum													
Training Accuracy: (%)							**Testing Accuracy: (%)**						
Kernel Size, Stride							Kernel Size, Stride						
[4,1] [2,1]	53.1	71.9	68.8	84.4	78.1	87.5	[4,1] [2,1]	42.2	39.3	37.0	37.2	36.7	36.4
[8,1] [2,1]	**59.4**	71.9	59.4	75.0	78.1	78.1	[8,1] [2,1]	**42.3**	40.3	38.1	36.8	37.4	36.5
[16,1] [4,1]	65.6	68.8	62.5	78.1	87.5	90.6	[16,1] [4,1]	42.0	39.5	38.2	37.5	37.1	36.7
[32,1] [4,1]	56.3	71.9	81.3	81.3	71.9	90.6	[32,1] [4,1]	41.9	39.8	37.5	36.7	36.9	36.6
[64,1] [4,1]	75.0	59.4	68.8	75.0	84.4	75.0	[64,1] [4,1]	41.9	40.0	37.2	37.2	36.5	36.6
[128,1] [4,1]	53.1	90.6	78.1	84.4	71.9	81.3	[128,1] [4,1]	41.9	40.2	37.6	37.4	36.8	36.5
[256,1] [4,1]	59.4	78.1	75.0	78.1	90.6	78.1	[256,1] [4,1]	41.7	39.6	37.0	36.9	36.5	36.6
Input length:	208	416	832	1248	2496	4992	Input length:	208	416	832	1248	2496	4992
Original signal length:	416	832	1248	2496	4992	9984	Original signal length:	416	832	1248	2496	4992	9984

Table 11. Training and testing accuracies for various CNN architectures with spectrogram preprocessing.

Input Type: Spectrogram (window = 104)													
Training Accuracy: (%)							**Testing Accuracy: (%)**						
Kernel Size, Stride							Kernel Size, Stride						
[2,2], [1,1]	81.3	87.5	84.4	78.1	93.8	96.9	[2,2], [1,1]	39.5	40.3	38.4	36.6	48.9	30.4
[4,4], [1,1]	78.1	78.1	84.4	87.5	96.9	96.9	[4,4], [1,1]	39.5	40.1	34.6	32.8	49.2	49.7
[6,6], [1,1]	68.8	84.4	93.8	90.6	93.8	90.6	[6,6], [1,1]	39.1	40.4	32.0	30.5	51.9	46.6
[8,8], [1,1]	75.0	87.5	84.4	84.4	90.6	**96.9**	[8,8], [1,1]	43.5	37.4	39.0	49.6	50.0	**52.2**
Input size:	129 ×	129 × 15	129 × 23	129 × 47	129 × 95	129 × 191	Input length:	129 × 7	129 × 15	129 × 23	129 × 47	129 × 95	129 × 191
Original signal length:	416	832	1248	2496	4992	9984	Original signal length:	416	832	1248	2496	4992	9984

All CNN configurations have significantly lower testing accuracy compared to those found using the CWRU dataset though training accuracy remains high. While this is certainly attributable, to some extent, to the disparity between measurements of artificial and natural bearing damage, other factors might make the Paderborn dataset more difficult to learn from compared to the CWRU dataset. Foremost among these factors is the reduced number of experimental runs from which to learn, as this leads to dataset sparsity.

Unlike Case Study 1, the envelope spectrum appears to be the poorest choice in preprocessor based on overall accuracy. The remaining three preprocessing methods seem to be approximately equal, though all seem too poor to be particularly useful.

The results of Part 1 give broader confirmations of the findings of Chen et al. [11], who demonstrate that classic implementations of CNNs are not able to learn enough useful features from the artificially damaged bearings to accurately diagnose real faults.

5.3.2. Part 2: Artificial to Artificial Damage

The results in Tables 12–15 demonstrate the improvement in accuracy when the training and testing data both relate to artificially damaged bearings.

Table 12. Training and testing accuracies for various CNN architectures with no preprocessing.

Input Type: Raw Time Series													
Training Accuracy: (%)							**Testing Accuracy: (%)**						
Kernel Size, Stride							Kernel Size, Stride						
[4,1], [2,1]	81.3	71.9	81.3	93.8	96.9	87.5	[4,1], [2,1]	33.8	43.2	39.7	48.5	52.3	53.6
[8,1], [2,1]	71.9	71.9	87.5	90.6	90.6	96.9	[8,1], [2,1]	43.8	50.3	52.5	52.0	53.2	48.8
[16,1], [4,1]	87.5	87.5	90.6	93.8	100.0	100.0	[16,1], [4,1]	46.3	50.6	50.1	56.2	51.7	48.3
[32,1], [4,1]	87.5	100.0	96.9	100.0	96.9	100.0	[32,1], [4,1]	50.6	57.3	60.5	54.2	59.6	52.1
[64,1], [4,1]	87.5	87.5	96.9	100.0	100.0	100.0	[64,1], [4,1]	55.0	55.1	57.6	64.2	60.1	47.2
[128,1], [4,1]	90.6	96.9	**90.6**	100.0	100.0	100.0	[128,1], [4,1]	53.6	57.1	**70.0**	62.1	54.3	58.7
[256,1], [4,1]	90.6	96.9	100.0	100.0	96.9	96.9	[256,1], [4,1]	52.5	58.7	57.1	64.2	62.5	51.7
Input length:	416	832	1248	2496	4992	9984	Input length:	416	832	1248	2496	4992	9984
Original signal length:	416	832	1248	2496	4992	9984	Original signal length:	416	832	1248	2496	4992	9984

Table 13. Training and testing accuracies for various CNN architectures with FFT preprocessing.

Input Type: Fourier Spectrum													
Training Accuracy: (%)							**Testing Accuracy: (%)**						
Kernel Size, Stride							Kernel Size, Stride						
[4,1] [2,1]	68.8	90.6	87.5	96.9	96.9	100.0	[4,1] [2,1]	44.7	55.6	58.9	63.7	66.1	62.2
[8,1] [2,1]	65.6	87.5	93.8	96.9	100.0	100.0	[8,1] [2,1]	43.9	55.3	59.4	68.3	61.9	65.2
[16,1] [4,1]	56.3	87.5	93.8	93.8	**96.9**	100.0	[16,1] [4,1]	46.2	52.0	56.0	63.9	**68.7**	67.8
[32,1] [4,1]	75.0	87.5	96.9	100.0	100.0	93.8	[32,1] [4,1]	47.9	55.7	58.9	61.8	66.4	63.6
[64,1] [4,1]	84.4	84.4	87.5	96.9	96.9	100.0	[64,1] [4,1]	47.1	50.3	58.0	55.2	65.2	69.1
[128,1] [4,1]	78.1	87.5	90.6	90.6	93.8	100.0	[128,1] [4,1]	47.9	57.7	55.2	53.9	47.2	56.0
[256,1] [4,1]	68.8	90.6	90.6	100.0	96.9	100.0	[256,1] [4,1]	42.4	57.6	61.9	63.5	64.0	49.8
Input length:	208	416	832	1248	2496	4992	Input length:	208	416	832	1248	2496	4992
Original signal length:	416	832	1248	2496	4992	9984	Original signal length:	416	832	1248	2496	4992	9984

Table 14. Training and testing accuracies for various CNN architectures with envelope spectrum preprocessing.

Input Type: Envelope Spectrum													
Training Accuracy: (%)							**Testing Accuracy: (%)**						
Kernel Size, Stride							Kernel Size, Stride						
[4,1] [2,1]	46.9	50.0	71.9	78.1	71.9	84.4	[4,1] [2,1]	31.8	25.6	22.2	23.8	27.0	31.5
[8,1] [2,1]	65.6	56.3	68.8	84.4	87.5	90.6	[8,1] [2,1]	28.8	26.5	18.2	20.5	19.5	15.8
[16,1] [4,1]	59.4	78.1	78.1	75.0	93.8	84.4	[16,1] [4,1]	32.2	30.6	32.3	21.9	23.8	31.1
[32,1] [4,1]	65.6	59.4	65.6	87.5	81.3	81.3	[32,1] [4,1]	23.5	23.8	21.3	32.2	38.3	38.3
[64,1] [4,1]	65.6	68.8	78.1	81.3	84.4	78.1	[64,1] [4,1]	29.8	32.4	29.4	37.9	38.3	37.6
[128,1] [4,1]	53.1	59.4	81.3	87.5	90.6	96.9	[128,1] [4,1]	25.6	27.5	30.9	32.8	31.1	31.2
[256,1] [4,1]	62.5	53.1	68.8	84.4	81.3	**87.5**	[256,1] [4,1]	25.2	36.7	24.7	39.2	32.5	**51.6**
Input length:	208	416	832	1248	2496	4992	Input length:	208	416	832	1248	2496	4992
Original signal length:	416	832	1248	2496	4992	9984	Original signal length:	416	832	1248	2496	4992	9984

Table 15. Training and testing accuracies for various CNN architectures with spectrogram preprocessing.

Input Type: Spectrogram (window = 104)													
Training Accuracy: (%)							Testing Accuracy: (%)						
Kernel Size, Stride							Kernel Size, Stride						
[2,2], [1,1]	90.6	96.9	100.0	100.0	84.4	96.9	[2,2], [1,1]	49.1	57.3	58.7	56.8	59.7	60.9
[4,4], [1,1]	84.4	90.6	96.9	96.9	100.0	**100.0**	[4,4], [1,1]	49.9	57.9	62.8	62.8	60.9	**69.5**
[6,6], [1,1]	93.8	100.0	100.0	100.0	100.0	100.0	[6,6], [1,1]	50.4	57.6	60.6	61.5	60.1	63.6
[8,8], [1,1]	96.9	93.8	100.0	100.0	100.0	100.0	[8,8], [1,1]	49.6	57.5	60.8	66.8	58.5	69.0
Input size:	129 × 7	129 × 15	129 × 23	129 × 47	129 × 95	129 × 191	Input length:	129 × 7	129 × 15	129 × 23	129 × 47	129 × 95	129 × 191
Original signal length:	416	832	1248	2496	4992	9984	Original signal length:	416	832	1248	2496	4992	9984

Part 2 eliminates the underlying domain difference between natural and artificial bearing damage measurements by using artificially damaged bearings for training and validation. Validation accuracy is improved overall with the notable exception of CNNs trained with envelope spectrum data. This indicates that the inherent difference between artificial and natural bearing damage are a significant, but not sole, contributor to the poor accuracy achieved in Part 1.

As with Case Study 1, results obtained using spectrogram preprocessing appear to be the least sensitive to changes in kernel and input sizes. Other trends linking accuracy and hyperparameter values differ between case study 1 and case study 2. This suggests that different underlying factors including experimental procedure and physical setup influence hyperparameter optimization. This implies that CNNs may need to be tuned for different industrial applications if a universally applicable architecture and training scheme is not developed.

6. Case Study 3: 2009 PHM Challenge Gear Fault Dataset

This case study explores the effectiveness of the previously described CNN configurations for diagnosing various health conditions of a two-stage gear box using the 2009 PHM Challenge dataset [27]. The objective of this case study is to determine whether architectures apparently useful for bearing fault detection by CNNs are also able to perform gearbox fault diagnosis.

6.1. Dataset Description

This dataset contains eight unique health states for the gearbox, each having a different combination of subcomponents that are either healthy or artificially damaged. This gives rise to health states with multiple faults, leading to a more complicated diagnosis problem. Table 16 summarizes the states of the various gears, bearings, and shafts for each of the health states. The dataset contains two channels of vibration measurements and a tachometer signal. For this experiment, only the first channel is used. For all states, four seconds are sampled at a sampling frequency of 66.67 kHz.

Table 16. Summary of gearbox component condition in various labeled states [4].

	Gear				Bearing						Shaft	
Case	**32T**	**96T**	**48T**	**80T**	**IS:IS**	**ID:IS**	**OS:IS**	**IS:OS**	**ID:OS**	**OS:OS**	**Input**	**Output**
Spur 1	Good	Good	Good	Good	Good	Good	Good	Good	Good	Good	Good	Good
Spur 2	Chipped	Good	Eccentric	Good	Good	Good	Good	Good	Good	Good	Good	Good
Spur 3	Good	Good	Eccentric	Good	Good	Good	Good	Good	Good	Good	Good	Good
Spur 4	Good	Good	Eccentric	Broken	Ball	Good	Good	Good	Good	Good	Good	Good
Spur 5	Chipped	Good	Eccentric	Broken	Inner	Ball	Outer	Good	Good	Good	Good	Good
Spur 6	Good	Good	Good	Broken	Inner	Ball	Outer	Good	Good	Good	Imbalance	Good
Spur 7	Good	Good	Good	Good	Inner	Good	Good	Good	Good	Good	Good	Keyway Sheared
Spur 8	Good	Good	Good	Good	Good	Ball	Outer	Good	Good	Good	Imbalance	Good

IS = Input Shaft; IS = Input Side; ID = Idler Shaft; OS = Ouput Side; OS = Output Shaft.

6.2. Data Preparation

The same data augmentation procedure described above is used here to generate many more training and testing samples of different sizes form the original measurements. Again, k-fold cross validation is used, with k = 3. One difference in procedure was mandated for the process using envelope spectrum preprocessing; data was normalized to have a zero mean as well as having a standard deviation of one. This was necessary to achieve network convergence under the same learning parameters as all other preprocessing methods.

Unlike the datasets used in the previous two case studies, there is only one measurement obtained for each health state. This means that training and testing data cannot be obtained from different sets of damaged components. This leads to a simpler machine learning problem for which less generalization is needed to achieve high testing accuracies. The results below indeed show that the testing accuracies achieved here are much higher than the previous case studies.

6.3. Results and Discussion

Tables 17–20 present the results for Case Study 3 in the same format used above.

Table 17. Training and testing accuracies for various CNN architectures with no preprocessing.

Input Type: Raw Time Series													
	Training Accuracy: (%)							**Testing Accuracy: (%)**					
Kernel Size, Stride							Kernel Size, Stride						
[4,1], [2,1]	41.7	56.3	54.2	89.6	100.0	100.0	[4,1], [2,1]	34.3	27.7	31.4	27.1	27.2	20.3
[8,1], [2,1]	45.8	77.1	83.3	100.0	100.0	100.0	[8,1], [2,1]	44.7	53.5	52.2	46.4	31.5	24.7
[16,1], [4,1]	39.6	66.7	79.2	85.4	95.8	100.0	[16,1], [4,1]	53.4	64.0	67.7	68.1	55.3	49.2
[32,1], [4,1]	68.8	85.4	81.3	95.8	100.0	100.0	[32,1], [4,1]	64.2	75.7	80.0	74.4	80.2	51.6
[64,1], [4,1]	72.9	93.8	85.4	93.8	100.0	100.0	[64,1], [4,1]	73.2	84.9	93.8	93.1	83.6	59.7
[128,1], [4,1]	77.1	97.9	97.9	93.8	100.0	100.0	[128,1], [4,1]	78.2	90.9	96.1	91.7	93.2	69.0
[256,1], [4,1]	89.6	97.9	100.0	**97.9**	100.0	100.0	[256,1], [4,1]	79.2	91.5	96.4	**98.1**	96.7	92.6
Input length:	416	832	1248	2496	4992	9984	Input length:	416	832	1248	2496	4992	9984
Original signal length:	416	832	1248	2496	4992	9984	Original signal length:	416	832	1248	2496	4992	9984

Table 18. Training and testing accuracies for various CNN architectures with FFT preprocessing.

	Training Accuracy: (%)							**Testing Accuracy: (%)**					
Kernel Size, Stride							Kernel Size, Stride						
[4,1] [2,1]	85.4	95.8	100.0	100.0	100.0	**100.0**	[4,1] [2,1]	81.3	94.1	97.5	99.7	99.9	**100.0**
[8,1] [2,1]	79.2	95.8	100.0	100.0	100.0	**100.0**	[8,1] [2,1]	82.3	95.1	97.9	99.6	99.8	**100.0**
[16,1] [4,1]	77.1	97.9	100.0	100.0	100.0	100.0	[16,1] [4,1]	81.4	93.8	98.6	99.2	99.0	99.8
[32,1] [4,1]	77.1	91.7	97.9	100.0	100.0	100.0	[32,1] [4,1]	84.4	95.1	98.3	99.3	99.5	99.8
[64,1] [4,1]	81.3	95.8	97.9	100.0	100.0	100.0	[64,1] [4,1]	85.6	96.2	98.8	99.6	99.7	99.0
[128,1] [4,1]	87.5	95.8	100.0	100.0	100.0	100.0	[128,1] [4,1]	87.3	96.7	98.9	99.0	99.0	99.9
[256,1] [4,1]	83.3	95.8	100.0	97.9	95.8	100.0	[256,1] [4,1]	88.4	97.0	99.1	98.0	98.0	99.7
Input length:	208	416	832	1248	2496	4992	Input length:	208	416	832	1248	2496	4992
Original signal length:	416	832	1248	2496	4992	9984	Original signal length:	416	832	1248	2496	4992	9984

Table 19. Training and testing accuracies for various CNN architectures with envelope spectrum preprocessing.

Input Type: Envelope Spectrum													
	Training Accuracy: (%)							**Testing Accuracy: (%)**					
Kernel Size, Stride							Kernel Size, Stride						
[4,1] [2,1]	31.3	52.1	60.4	89.6	100.0	100.0	[4,1] [2,1]	20.8	24.9	28.1	34.3	43.0	58.6
[8,1] [2,1]	20.8	39.6	56.3	97.9	100.0	100.0	[8,1] [2,1]	21.9	24.6	27.3	32.5	40.5	55.6
[16,1] [4,1]	31.3	31.3	56.3	89.6	100.0	100.0	[16,1] [4,1]	21.8	26.2	29.6	35.8	46.8	58.8
[32,1] [4,1]	31.3	52.1	45.8	81.3	97.9	100.0	[32,1] [4,1]	22.8	27.3	29.8	37.0	46.0	59.1
[64,1] [4,1]	31.3	47.9	54.2	77.1	100.0	100.0	[64,1] [4,1]	22.7	27.1	30.8	36.3	46.0	63.0
[128,1] [4,1]	25.0	47.9	52.1	83.3	97.9	**100.0**	[128,1] [4,1]	21.7	25.8	30.1	37.8	45.4	**65.4**
[256,1] [4,1]	35.4	54.2	70.8	89.6	95.8	100.0	[256,1] [4,1]	19.5	26.0	29.1	36.6	46.1	63.3
Input length:	208	416	832	1248	2496	4992	Input length:	208	416	832	1248	2496	4992
Original signal length:	416	832	1248	2496	4992	9984	Original signal length:	416	832	1248	2496	4992	9984

Table 20. Training and testing accuracies for various CNN architectures with spectrogram preprocessing.

Input Type: Spectrogram (window = 104)													
	Training Accuracy: (%)							**Testing Accuracy: (%)**					
Kernel Size, Stride							Kernel Size, Stride						
[2,2], [1,1]	95.8	97.9	100.0	100.0	100.0	100.0	[2,2], [1,1]	75.9	87.4	95.8	96.1	98.4	97.2
[4,4], [1,1]	95.8	97.9	100.0	100.0	100.0	100.0	[4,4], [1,1]	81.7	92.5	97.3	98.8	98.6	99.0
[6,6], [1,1]	95.8	100.0	100.0	100.0	**100.0**	100.0	[6,6], [1,1]	82.3	93.6	97.8	99.2	**99.2**	98.2
[8,8], [1,1]	91.7	97.9	100.0	100.0	100.0	100.0	[8,8], [1,1]	83.9	92.2	97.2	99.1	98.7	92.8
Input size:	129 × 7	129 × 15	129 × 23	129 × 47	129 × 95	129 × 191	Input length:	129 × 7	129 × 15	129 × 23	129 × 47	129 × 5	129 × 191
Original signal length:	416	832	1248	2496	4992	9984	Original signal length:	416	832	1248	2496	4992	9984

Despite the increased complexity of the mechanical system studied, the diagnostic abilities of the studied CNNs appear much greater for this dataset. This is almost certainly a result of the fact that the data from all experimental runs appear in both training and testing, even if unique samples created during data augmentation do not appear in both datasets. As with the other case studies, different preprocessing methods yield different patterns in which kernel sizes and input sizes perform best.

Using the raw time input type appears to only be successful when larger kernels are used. Moreover, performance is somewhat improved by using mid-sized inputs. FFT input types appear to work very well irrespective of kernel size and benefit somewhat by using larger inputs. The results from envelope spectrum contrast starkly with those of other preprocessing types–the average testing accuracy achieved is much poorer for the configurations trained here. It appears that a larger input duration than studied here would be needed for the envelope spectrum to be accurate. This may be due to the longer period involved for repeating gear meshing envelope signatures. Training accuracies improved with larger input durations for all input types and kernel sizes, though corresponding validation accuracies did not necessarily follow. Raw time and envelope spectrum showed the most significant overfitting, especially with long inputs durations and small kernels. Spectrogram preprocessing gives strong results overall with performance peaking when paired with mid-sized kernels and larger inputs.

7. Discussion

Few insights can be extrapolated with respect to which CNN configurations are best suited to each input type. The only commonality between all three datasets studied is the low sensitivity to hyperparameter selection in achieving high accuracy for models trained with spectrograms. This makes them the safest choice if extensive hyperparameter optimization is not possible. This is not necessarily revealed when only considering the best overall configurations from each case study, as in Table 21, though a wide range of input durations and kernel sizes are represented here.

Table 21. Details of best CNNs from each case study.

	Validation Accuracy (%)	Preprocessor	Original Signal Length	Kernel Size/Stride
Case Study 1	80.1	Spectrogram	832	[6,6]/[1,1]
Case Study 2–part 1	55.8	FFT	9984	[64,1]/[4,1]
Case Study 2–part 2	70.0	None	1248	[128,1]/[4,1]
Case Study 3	99.2	Spectrogram	4992	[6,6]/[1,1]

The three datasets studied yield different patterns and vastly different accuracies from the same CNNs. There is a complex relationship between input type and hyperparameter optimization that varies for each dataset. Case Study 2 highlights the ineffectiveness of classical CNNs for cross-domain problems involving training with artificially damaged bearings and testing with real damage, suggesting that they would not be adequate for real industrial applications, highlighting this as an area for additional future work.

The comparatively high performance of the studied CNNs with the PHM 2009 dataset in case study 3 show how extracting training and testing samples from the same experimental runs leads to a far easier problem and inflates validation accuracies. This supports the findings of Pandhare et al. [17], who demonstrate that diagnostic accuracy can drop from 95 to 100% when experimental data is mixed to approximately 60% when experimental data contained in training and testing datasets is mutually exclusive. It seems probable that researchers finding greater accuracies on the CWRU dataset using classically shallow CNNs achieve such results by "contaminating" validation datasets in this way. A frequent

claim in papers such as this is that the application of deep learning for machine fault diagnosis will eliminate the need for a human expert, presumably an expert on the mechanical system being diagnosed. However, true that claim might be, it neglects the fact that a different sort of expert is needed to create a viable solution using deep learning methods. Clearly, useful diagnoses cannot be achieved without a preprocessor and CNN architecture that is well suited for the target domain. It is also clear that misleadingly high diagnosis accuracies can be achieved if data augmentation is performed on the experimental data before the data is randomly shuffled and split into training and testing datasets, leading to contamination of the testing set.

If the diagnostic problem is fairly constructed to reflect real-world challenges, it appears that classically shallow CNNs are, irrespective of kernel size and input size, not able to perform in a manner that would motivate industrial implementation. However, the present study is not an exhaustive exploration of CNNs; other hyperparameters can be altered to give many more network architectures. Future work should focus on designing algorithms that can learn from simulated faults to provide accurate diagnosis of real faults and addressing overfitting problems.

Author Contributions: Conceptualization, J.H. and P.D.; Data curation, J.H.; Investigation, J.H.; Project administration, P.D.; Software, P.D.; Supervision, P.D.; Visualization, J.H.; Writing—original draft, J.H.; Writing—review & editing, J.H. and P.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Natural Sciences and Engineering Research Council of Canada.

Data Availability Statement: This work uses publicly available data. The Case Western Reserve University bearing fault dataset [24], Paderborn University bearing fault dataset [26], and PHM 2009 Data Challenge dataset [27].

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

References

1. Stetco, A.; Dinmohammadi, F.; Zhao, X.; Robu, V.; Flynn, D.; Barnes, M.; Keane, J.; Nenadic, G. Machine learning methods for wind turbine condition monitoring: A review. *Renew. Energy* **2019**, *133*, 620–635. [CrossRef]
2. Nandi, S.; Toliyat, H.A.; Li, X. Condition Monitoring and Fault Diagnosis of Electrical Motors—A Review. *IEEE Trans. Energy Convers.* **2005**, *20*, 719–729. [CrossRef]
3. Lei, Y.; Yang, B.; Jiang, X.; Jia, F.; Li, N.; Nandi, A.K. Applications of machine learning to machine fault diagnosis: A review and roadmap. *Mech. Syst. Signal Process.* **2020**, *138*, 106587. [CrossRef]
4. Jing, L.; Zhao, M.; Li, P.; Xu, X. A convolutional neural network based feature learning and fault diagnosis method for the condition monitoring of gearbox. *Measurement* **2017**, *111*, 1–10. [CrossRef]
5. Bolón-Canedo, V.; Sánchez-Maroño, N.; Alonso-Betanzos, A. A review of feature selection methods on synthetic data. *Knowl. Inf. Syst.* **2013**, *34*, 483–519. [CrossRef]
6. Bengio, Y.; Courville, A.; Vincent, P. Representation Learning: A Review and New Perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.* **2013**, *35*, 1798–1828. [CrossRef] [PubMed]
7. Eren, L. Bearing Fault Detection by One-Dimensional Convolutional Neural Networks. *Math. Probl. Eng.* **2017**, *2017*, 8617315. [CrossRef]
8. Eren, L.; Ince, T.; Kiranyaz, S. A Generic Intelligent Bearing Fault Diagnosis System Using Compact Adaptive 1D CNN Classifier. *J. Sign. Process Syst.* **2019**, *91*, 179–189. [CrossRef]
9. Jiang, G.; He, H.; Yan, J.; Xie, P. Multiscale Convolutional Neural Networks for Fault Diagnosis of Wind Turbine Gearbox. *IEEE Trans. Ind. Electron.* **2019**, *66*, 3196–3207. [CrossRef]
10. Huang, R.; Liao, Y.; Zhang, S.; Li, W. Deep Decoupling Convolutional Neural Network for Intelligent Compound Fault Diagnosis. *IEEE Access* **2019**, *7*, 1848–1858. [CrossRef]
11. Chen, Y.; Peng, G.; Xie, C.; Zhang, W.; Li, C.; Liu, S. ACDIN: Bridging the gap between artificial and real bearing damages for bearing fault diagnosis. *Neurocomputing* **2018**, *294*, 61–71. [CrossRef]
12. Appana, D.K.; Prosvirin, A.; Kim, J.-M. Reliable fault diagnosis of bearings with varying rotational speeds using envelope spectrum and convolution neural networks. *Soft Comput.* **2018**, *22*, 6719–6729. [CrossRef]
13. Guo, S.; Yang, T.; Gao, W.; Zhang, C. A Novel Fault Diagnosis Method for Rotating Machinery Based on a Convolutional Neural Network. *Sensors* **2018**, *18*, 1429. [CrossRef]

14. Han, Y.; Tang, B.; Deng, L. Multi-level wavelet packet fusion in dynamic ensemble convolutional neural network for fault diagnosis. *Measurement* **2018**, *127*, 246–255. [CrossRef]
15. Lee, W.J. Learning via acceleration spectrograms of a DC motor system with application to condition monitoring. *Int. J. Adv. Manuf. Technol.* **2020**, *14*. [CrossRef]
16. Verstraete, D.; Ferrada, A.; Droguett, E.L.; Meruane, V.; Modarres, M. Deep Learning Enabled Fault Diagnosis Using Time-Frequency Image Analysis of Rolling Element Bearings. *Shock Vib.* **2017**, *2017*, 5067651. [CrossRef]
17. Pandhare, V.; Singh, J.; Lee, J. Convolutional Neural Network Based Rolling-Element Bearing Fault Diagnosis for Naturally Occurring and Progressing Defects Using Time-Frequency Domain Features. In Proceedings of the 2019 Prognostics and System Health Management Conference (IEEE PHM-Paris), Paris, France, 2–5 May 2019; pp. 320–326. [CrossRef]
18. Liu, S.; Xie, J.; Shen, C.; Shang, X.; Wang, D.; Zhu, Z. Bearing Fault Diagnosis Based on Improved Convolutional Deep Belief Network. *Appl. Sci.* **2020**, *10*, 6359. [CrossRef]
19. McFadden, P.D.; Smith, J.D. Model for the vibration produced by a single point defect in a rolling element bearing. *J. Sound Vib.* **1984**, *96*, 69–82. [CrossRef]
20. Randall, R.B.; Antoni, J. Rolling element bearing diagnostics—A tutorial. *Mech. Syst. Signal Process.* **2011**, *25*, 485–520. [CrossRef]
21. McFadden, P.D. Detecting Fatigue Cracks in Gears by Amplitude and Phase Demodulation of the Meshing Vibration. *J. Vib. Acoust.* **1986**, *108*, 165–170. [CrossRef]
22. Guo, Y.; Zhao, L.; Wu, X.; Na, J. Vibration separation technique based localized tooth fault detection of planetary gear sets: A tutorial. *Mech. Syst. Signal Process.* **2019**, *129*, 130–147. [CrossRef]
23. Bouvrie, J. *Notes on Convolutional Neural Networks*; 2006; p. 8. Available online: https://core.ac.uk/reader/86960 (accessed on 12 March 2021).
24. Case Western Reserve University Bearing Data Center Website. Available online: http://csegroups.case.edu/bearingdatacenter/home (accessed on 8 May 2020).
25. Smith, W.A.; Randall, R.B. Rolling element bearing diagnostics using the Case Western Reserve University data: A benchmark study. *Mech. Syst. Signal Process.* **2015**, *64–65*, 100–131. [CrossRef]
26. 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. In Proceedings of the European Conference of the Prognostics and Health Management Society, Bilbao, Spain, 5–8 July 2016; p. 17.
27. Public Data Sets | PHM Society. Available online: https://www.phmsociety.org/references/datasets (accessed on 24 August 2020).