applied sciences

MDPI

Article

Classifying Multivariate Signals in Rolling Bearing Fault Detection Using Adaptive Wide-Kernel CNNs †

Jurgen van den Hoogen [1,2], Stefan Bloemheuvel [1,2] and Martin Atzmueller [3,*]

[1] Department of Cognitive Science and Artificial Intelligence, Tilburg University, 5000 LE Tilburg, The Netherlands; j.o.d.hoogen@jads.nl (J.v.d.H.); s.d.bloemheuvel@jads.nl (S.B.)
[2] Jheronimus Academy of Data Science, 5211 DA 's-Hertogenbosch, The Netherlands
[3] Semantic Information Systems Group, Osnabrück University, 49074 Osnabrück, Germany
* Correspondence: martin.atzmueller@uni-osnabrueck.de
† This paper is a substantially adapted and extended revision of our paper "An Improved Wide-Kernel CNN for Classifying Multivariate Signals in Fault Diagnosis" published in the 2020 IEEE International Conference on Data Mining Workshops Proceedings (ICDMW), Sorrento, Italy, 17–20 November 2020.

Abstract: With the developments in improved computation power and the vast amount of (automatic) data collection, industry has become more data-driven. These data-driven approaches for monitoring processes and machinery require different modeling methods focusing on automated learning and deployment. In this context, deep learning provides possibilities for industrial diagnostics to achieve improved performance and efficiency. These deep learning applications can be used to automatically extract features during training, eliminating time-consuming feature engineering and prior understanding of sophisticated (signal) processing techniques. This paper extends on previous work, introducing one-dimensional (1D) CNN architectures that utilize an adaptive wide-kernel layer to improve classification of multivariate signals, e.g., time series classification in fault detection and condition monitoring context. We used multiple prominent benchmark datasets for rolling bearing fault detection to determine the performance of the proposed wide-kernel CNN architectures in different settings. For example, distinctive experimental conditions were tested with deviating amounts of training data. We shed light on the performance of these models compared to traditional machine learning applications and explain different approaches to handle multivariate signals with deep learning. Our proposed models show promising results for classifying different fault conditions of rolling bearing elements and their respective machine condition, while using a fairly straightforward 1D CNN architecture with minimal data preprocessing. Thus, using a 1D CNN with an adaptive wide-kernel layer seems well-suited for fault detection and condition monitoring. In addition, this paper clearly indicates the high potential performance of deep learning compared to traditional machine learning, particularly in complex multivariate and multi-class classification tasks.

Keywords: fault detection; condition monitoring; multivariate signals; time series analysis; deep learning; industrial application

Citation: van den Hoogen, J.; Bloemheuvel, S.; Atzmueller, M. Classifying Multivariate Signals in Rolling Bearing Fault Detection Using Adaptive Wide-Kernel CNNs. *Appl. Sci.* **2021**, *11*, 11429. https://doi.org/10.3390/app112311429

Academic Editors: Radu Godina and Pedro Espadinha da Cruz

Received: 10 August 2021
Accepted: 13 November 2021
Published: 2 December 2021

1. Introduction

In the current industrial era, manufacturers rely more and more on the use of sensors for data collection and analysis. These developments boost the industry towards newer standards as what is called Industry 4.0 [1,2]. These new approaches for improving performance and increasing production efficiency require scalable methods to process and explain the collected (often complex) data such as multivariate time series. With improved computational power, automated methods are easier to deploy while requiring less background knowledge on the engineering and physics part of the machinery, resulting in more efficient and "intelligent" approaches [3,4]. Due to the improved availability of large datasets derived from sensors, these automated learning techniques, e.g., deep learning, provide strong performance in signal classification tasks. This work focuses on

such approaches and provides specific enhancements for accurately distinguishing signals in fault detection and condition monitoring applications.

Regarding data analysis, in particular industrial applications with heavy machinery can benefit from a better understanding of the underlying processes and the state of the equipment in order to adapt their maintenance strategies. Previous research has shown that depending on the type or size of the machine, rolling bearing element defects account for at least 40% of broken machinery in industrial applications [5,6]. Such parts are used in rotating mechanisms of the machine. Due to frictions when the machine rotates, these bearing elements degrade over time. Traditionally, the condition of a rolling bearing element would be approximated based on historical data of breakdowns. Nowadays, the condition of a bearing element can be measured by mounting sensors on selected sections of the machine to record vibrations [7] or by measuring motor currents of the electrical engine that drives these elements [8]. Especially vibrations have proved to be effective for displaying the underlying state of bearings [9].

As one important prerequisite for an effective analysis, raw signals need to be denoised and preprocessed before analysis—using complex and time-consuming signal processing techniques to retrieve usable information [7,10]. As a result, the general focus has been shifted towards deep learning algorithms that are able to process raw data and can automatically construct features by recognizing patterns in the input data [9]. This automated process saves time, is less prone to human error and may require less domain expertise of a specialized domain expert.

This paper investigates the use of deep learning in the context of fault detection of rolling bearing elements and builds on our earlier research [11], exploring the usage of one-dimensional CNNs for classifying multivariate signals based on data derived from rotating machines. In particular, we propose several architectures that are built on the wide-kernel framework developed by [10]. These models tend to handle high frequency sensor data particularly well, while using a somewhat "shallow" deep learning structure; therefore, these models are easily trainable, can be scaled when the dimensionality of the data increases and are applicable in real-time settings. Due to these reasons, such architectures are then also applicable in different contexts and application domains. We exemplarily investigate those approaches using two datasets, where we demonstrate their efficacy for classifying multivariate signals.

Our contributions are summarized as follows:

1. We investigate the performance of wide-kernel CNNs [10] in several settings and adaptations designed to process multivariate time series derived from various experiments. For this, we demonstrate how high performance in multi-class signal classification tasks can be achieved.
2. We propose a method for implementing an adaptive wide-kernel in the first convolutional layer that is able to transform any form of sequential sensor data without any dimensionality reduction; therefore, abiding the principles of a wide-kernel convolutional layer as proposed in [10,11]. We implement this in two models.
3. We evaluate our model options thoroughly on multiple datasets in deviating contexts in order to show their generalizability, both with and without large amounts of training data.
4. Finally, we illustrate the impact of model settings and architecture adaptations on model performance, resulting in a streamlined model on both the performance side, as well as computational efficiency.

The rest of the paper is structured as follows: We discuss related work in Section 2, where we also briefly summarize the necessary background on deep learning. This is followed by the introduction of the proposed models in Section 3. Next, we describe the experiments and findings in depth in Section 4. Finally, Section 5 concludes with a discussion and summary and also outlines some promising future prospects.

2. Related Work and Background

This section covers relevant work relating to maintenance of industrial machinery connected to fault detection and condition monitoring, and describes related deep learning methods in the section below.

2.1. Maintenance

The usage of machinery is essential in industrial applications. Failure of these machines due to wear of the underlying elements is one of the most prevalent concerns in industry; therefore, equipment maintenance is critical in preventing malfunctions and, as a result, minimizing downtime. According to [12], maintenance expenses range from 15% to 60% of the total cost of the manufactured goods. Around 33% of maintenance expenses are directly connected to redundant and incorrect equipment maintenance within these margins. As a result, lowering the expenses of costly maintenance might substantially reduce overall production costs by improving equipment productivity [13].

According to [14,15] there are three distinct techniques for maintaining equipment: (1) modificative maintenance, in which components are upgraded to improve machine productivity and performance, (2) preventive maintenance, in which a component is replaced just before it fails and (3) break-down corrective maintenance, in which a part is replaced after it fails, leading to downtime of the machine. In this paper, we concentrate on (2) preventive maintenance, which itself is separated into two types: usage-based maintenance (UBM) and condition-based maintenance (CBM).

The UBM method relies entirely on arranging maintenance visits by the engineer when a specific threshold of consumption is achieved. In practice, this implies that visits are scheduled with a certain interval between them, comparable to a yearly automobile inspection. This technique results in relatively little equipment downtime, which is good for production. However, this method has two major disadvantages; the high expenses of maintenance visits and the replacement of parts that are still usable. As a result, in many industrial applications, CBM is the recommended maintenance approach, making use of data-driven methods and approaches, e.g., cf. [14,16,17].

CBM assesses the current state of equipment to identify if maintenance is required. The concept behind CBM is to only execute maintenance when specific parameters, e.g., deviating behavior in the data, indicate a reduction in performance or a predicted rise in failures. This means fewer maintenance visits and more efficient usage of the underlying components, which in turn leads to lower overall maintenance costs.

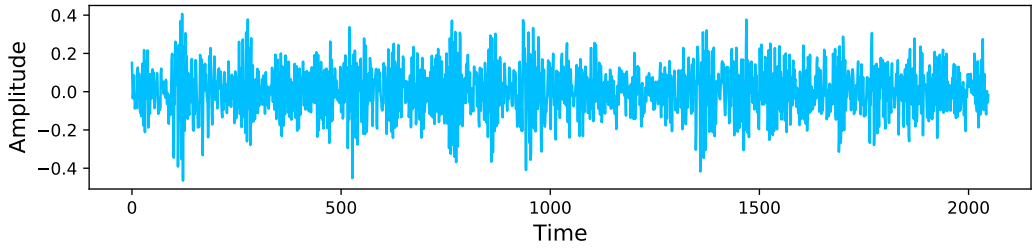
2.2. Fault Detection and Condition Monitoring

Within CBM, fault detection and condition monitoring are common approaches for rotating industrial equipment where faults regularly occur [18]. In the past, this was achieved using physics-based models, which require background knowledge on the underlying processes. These models hardly adapt to changing circumstances and increments in the amount of data and variables [19]. Innovations in data-driven analytics and the advances in Industrial Internet of Things (IOT) have altered the area of fault detection and condition monitoring towards a more intelligent approach [4,20]. These methods allow for automated data processing without prior understanding of technical elements of industrial machinery, while easily adaptable to changing operation conditions.

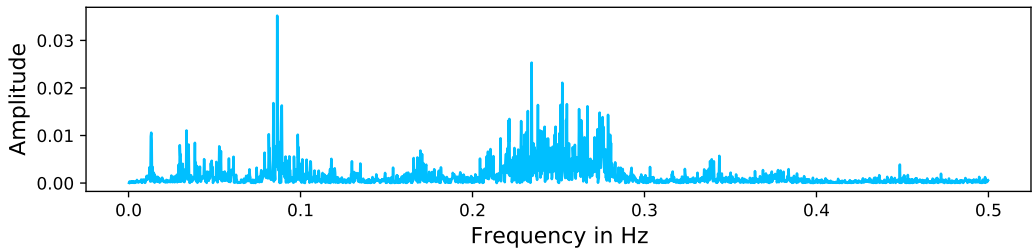
Because of the increased availability of large-scale time series datasets and better processing capacity, the usage of deep learning applications has grown in popularity. These time series are recorded by sensors, which are increasingly being used for fault detection and condition monitoring. When elements of equipment decay over time, for example, the analog metrics of the machine will not immediately reflect this; however, increased power usage (motor current), vibrations or temperature of machine elements monitored with internal and external technologies, such as sensors, might indicate that the underlying parts need to be replaced [4,9,10]. These signals derived from sensors can be converted

into numerical time series data for subsequent study. However, for reliable fault detection, considerable efforts in feature extraction is typically required.

Traditional methods for extracting representative features to classify signals include time-domain analysis (e.g., statistical measures such as mean and standard deviation), frequency-domain analysis (e.g., Fourier transformations [7], see Figure 1) and time-frequency domain analysis, (e.g., wavelet transformations [21,22]). As one could expect, the quantity of features derived from the different domains results in a high-dimensional dataset. Therefore, features are picked [23] and techniques are frequently used to decrease the dimensionality of these features, such as principal component analysis (PCA) [24,25] or linear discriminant analysis (LDA) [26]. Furthermore, [27], for example, utilized information entropy to preprocess the original time series data.



(**a**) Signal in the time-domain



(**b**) Signal in the frequency-domain

Figure 1. Example of the original signal in the time-domain (**a**) transformed to the frequency-domain (**b**) using the fast Fourier transformation algorithm.

Before the final dataset can be supplied to a classifier, the essential preprocessing steps typically take a substantial amount of time and high-level knowledge in signal processing and data processing—regarding standard (non-deep learning) machine learning approaches. Additionally, the feature extraction process is influenced by the type of data gathered from a particular machine or sensor, for instance, vibration sensors require different preprocessing steps than analog sensors.

Within fault detection and condition monitoring, many different classifiers are researched, including k-nearest neighbors (K-NN) [28,29], support vector machines (SVM) [21,30–32], artificial neural networks [33,34] and interpretable machine learning methods such as random forests (RF) [35]. The performance of these techniques varies a lot depending on the data quality, thoroughness of the feature extraction process and complexity of the classification task; therefore, it is often difficult to find the right classifier for the task at hand. In other words, there is not one particular machine learning classifier that is most capable of distinguishing different fault conditions. As a result, a comparison between classifiers is deemed necessary for every fault detection task to find the most optimal model.

This work concentrates on deep learning applications, and more specifically on the use of one-dimensional CNNs in the context of fault detection utilizing time series data, i.e., multivariate signal data. To evaluate our proposed deep learning techniques, we used renowned benchmark datasets for fault detection and condition monitoring in various settings. We look at the generalizability of these techniques, their performance with limited

training data and compare them to traditional machine learning approaches such as k-nearest neighbors, random forests and support vector machines.

2.3. Deep Learning

In general, deep learning methods offer strong processing and learning on complicated data. For instance, automatic feature generation and refinement techniques, such as for complicated classification problems, may typically leverage connections in the data in order to retrieve valuable information from the data into redefined structures. Using complicated multivariate signal data for fault detection and condition monitoring is an example of utilizing these complex data structures. Overall, there has been a lot of interest in utilizing neural networks for such complicated classification problems during the previous decade.

Initially, the multi-layer perceptron (MLP) was used, in which all network layers are fully linked [36]; however, because of the significant increase in calculation time, the depth of these networks is restricted. Thus, in the past years, more advanced neural network architectures were developed to accommodate for this.

The creation of recurrent neural networks (RNN), such as long short-term memory (LSTM) networks, has yielded promising results since they are able to account for time dependencies and therefore can handle time series data and signals very well [37,38]. However, because of memorizing long-term time dependencies, RNNs use vast amounts of memory (RAM). So, these models are less suited for long sequence data due to increased training times. This is especially the case for signal data from sensors, which is often sampled at a high frequency consisting of many data points. To tackle the training issue of RNNs, combined models utilizing autoencoders as feature extractors were created [39]. These models enhance computation but also increase the model's complexity and decrease its interpretability.

Deep learning algorithms have been used to detect faults and monitor machine conditions many times. The MLP [40] was one of the earliest deep learning applications in fault detection and condition monitoring. Later, RNNs [37] and CNNs [4,20,41,42] became more common in fault detection, where they have exhibited significant performance increases. Further, CNN approaches combined with data transformations, e.g., spectrograms, have been proposed several times [3,43]. Ref. [44] was successful in the creation of a 1D CNN that is able to handle raw signals by integrating automated feature extraction with time series classification. These 1D CNNs can also withstand noise effectively and can be trained with small amounts of data [45].

Overview—Convolutional Neural Networks

A convolutional neural network (CNN), in general, is a regularized MLP that specializes in processing two-dimensional inputs such as picture pixels and color channels. CNNs have previously proven to be effective in computer vision tasks including image classification and video identification [46,47].

The main advantages of a CNN, compared with a traditional neural network, such as an MLP, is the use of local receptive fields, weight-sharing and sub-sampling. Especially, the weight-sharing significantly reduces memory requirements and therefore improves algorithmic efficiency [48]. Commonly, a convolutional layer consists of three phases. The first layer performs a number of convolutions, followed by the second phase that consists of an activation function. Afterwards, a pooling function is applied [48].

Before employing a CNN on one-dimensional data, e.g., time series, the data has to be converted using signal processing techniques into a two-dimensional representation in the time-frequency spectrum or using wavelet transforms [22,49,50]. For example, one-dimensional signals can be transformed in two-dimensional spectrograms, which in turn can be fed as an image to the CNN. This approach is not able to process raw signals directly, thus contradicting the advantages of employing deep learning applications over standard machine learning approaches. The one-dimensional (1D) CNN was created to tackle this challenge by integrating automated feature extraction for time series classification tasks [44].

These models are good at handling noise in time series and can be trained with different data sizes, while being less computationally heavy compared to RNNs or MLPs. As a result, 1D CNNs are becoming more and more applied in time series classification tasks such as fault detection and condition monitoring [45].

Convolutional Layer

A convolutional layer convolves the input with filter kernels followed by the activation unit to generate output features. Each of these filters uses the same kernel to extract local features from the input local region, called weight-sharing. Results of the convolutional operations across the input are fed to the activation function that leads to the output features. The convolution operation is described as:

$$y_i^{l+1}(j) = k_i^l * M^l(j) + b_i^l. \quad (1)$$

Here, b_i^l denotes the bias and k_i^l denotes the weights of the i-th filter kernel in layer l. $M^l(j)$ describes the j-th local region in layer l. $(*)$ represents the convolution operation that computes the dot product of the kernel and the local regions. $y_i^{l+1}(j)$ denotes the input of the j-th neuron in feature map i of layer $l+1$.

Activation Function

The activation function is embedded in every convolutional layer to acquire nonlinear features from the input after the convolutional operation. Depending on the input and the task at hand, there are several different activation functions available; however, in recent years, the rectified linear unit (ReLU) has proven to be efficient in its computations and is therefore the most common used activation function. In this study, the ReLU activation function was used in the convolutional layers and can be described as follows:

$$ReLU(x) = max\{0, x + N(0, \sigma(x))\}. \quad (2)$$

Here, x represents the outputs of the convolutional operation $y_i^{l+1}(j)$ and $N(0, \sigma(x))$ represents Gaussian distributed noise with mean 0 and variance $\sigma(x)$, which has proven to make optimization easier [51].

Pooling Layer

The output of the convolutional layer and activation function are usually fed to a pooling layer (also known as sub-sampling layer). This layer reduces the spatial size of the input features by a down-sampling operation and decreases the number of parameters and computations in the network. There are different pooling functions such as max-pooling and average-pooling. The pooling function performs a local operation over the input features resulting in a representation that becomes invariant to small translations of the input. In general, the pooling function can be denoted as:

$$P_i^l = f(\omega_i^l S\left(M_i^{l-1}\right) + b_i^l). \quad (3)$$

$S(\cdot)$ denotes the pooling operation where values of the convolved features are computed on different locations. For every layer l, the i-th weight matrix is denoted as ω_i. M_i represents the outputs of the convolutional layer (feature map) and b_i denotes the bias. These calculations then result in the compressed feature representation p_i^l given above.

3. Method

The design of a CNN and the quality of the data have a significant impact on its classification performance, for instance, sensor signals from industrial machinery regularly contain significant levels of noise. Previous work showed the great performance of using a wide-kernel in the first convolutional layer, followed by smaller kernels in the follow-up layers, for detecting faults and classifying conditions of rotating equipment [10,11].

However, these methods were not able to automatically scale with varying data inputs; therefore, we extend on those approaches in this work by developing an adaptive wide-kernel layer that extracts features and filters out signal noise, transforming the input without any representation loss.

The general idea behind this adaptive layer is that depending on its input data, it will transform the data into an n-dimensional matrix without losing any information, e.g., the dot product of the first layer's output is equal to the dot product of the input data. Hence, the layer functions as a feature extractor without reducing its dimensionality, which results in a feature set extracted from low frequency bands [10].

This adaptive layer does require a set of rules to be adhered to. Otherwise, the output of the first wide-kernel layer does not correspond with the input data. The rules for the adaptive layer are summarized as follows:

- The sequence length and width of the first kernel should be a power term of two.
- The width of the first kernel should be higher than the denominator for calculating the stride in the first layer (default = 4).
- The kernel width of the first layer should not exceed the sequence length.

In this case, we calculate the amount of stride in the first layer, followed by the number of filters used:

$$F^s = \frac{K}{x},$$

$$\delta = \frac{S}{F^s}, \tag{4}$$

where K is the kernel size of the first layer, x is the denominator used to calculate the stride in the first layer F^s with a default setting of 4 and S the sequence length of a signal resulting in penalty function δ. Based on these values, the number of filters can be calculated:

$$F = \frac{S \cdot TS}{\delta}. \tag{5}$$

Here, TS denotes the number of time series available in the data, for example, the number of sensors. The result is the number of filters, denoted as F, used in the first layer of the model.

One of the main properties of this equation is that the number of filters will increase when the kernel size and therefore stride is set to a higher level, while the number of filters decreases with a low set value. This aligns with the idea that if the kernel size reduces the local information, there will be multiple versions of those local combinations. If the kernel size increases, there are less steps that the kernel is applied to per filter. Therefore, increasing the number of filters will equalize this deviation.

We implemented this adaptive layer structure on several fault detection and condition monitoring tasks in the form of a time series classification task. In this section, the proposed 1D CNN models are described in more detail. In addition, a supplementary page (https://github.com/JvdHoogen/Adaptive-WCNN) is made available containing the code for the proposed models.

3.1. Adaptive Wide-Kernel CNN (A-WCNN)

Similar to our previous proposed WDTCNN model [11], the adaptive wide-kernel CNN (A-WCNN) contains five convolutional layers followed by two fully connected layers; however, the A-WCNN is able to adapt its first wide-kernel layer based on the dimensionality of the input data. So, this hybrid version can be easily deployed on different datasets with other dimensionalities, without having to manually adjust the architecture of the model. After each convolutional layer, the model utilizes local average pooling to decrease the vector size of the convolutional output with length T divided by two, resulting in a pooled output length of $\frac{T}{2}$.

In our experiments, we assessed the performance of the A-WCNN under deviating circumstances using multiple sensors. In addition, the model generalizes more easily and allows different kernel sizes, which is set to 64 in this paper. The architecture of the A-WCNN based on a two-dimensional time series input, segmented into sequences of 2048 data points, is shown on the left side of Figure 2.

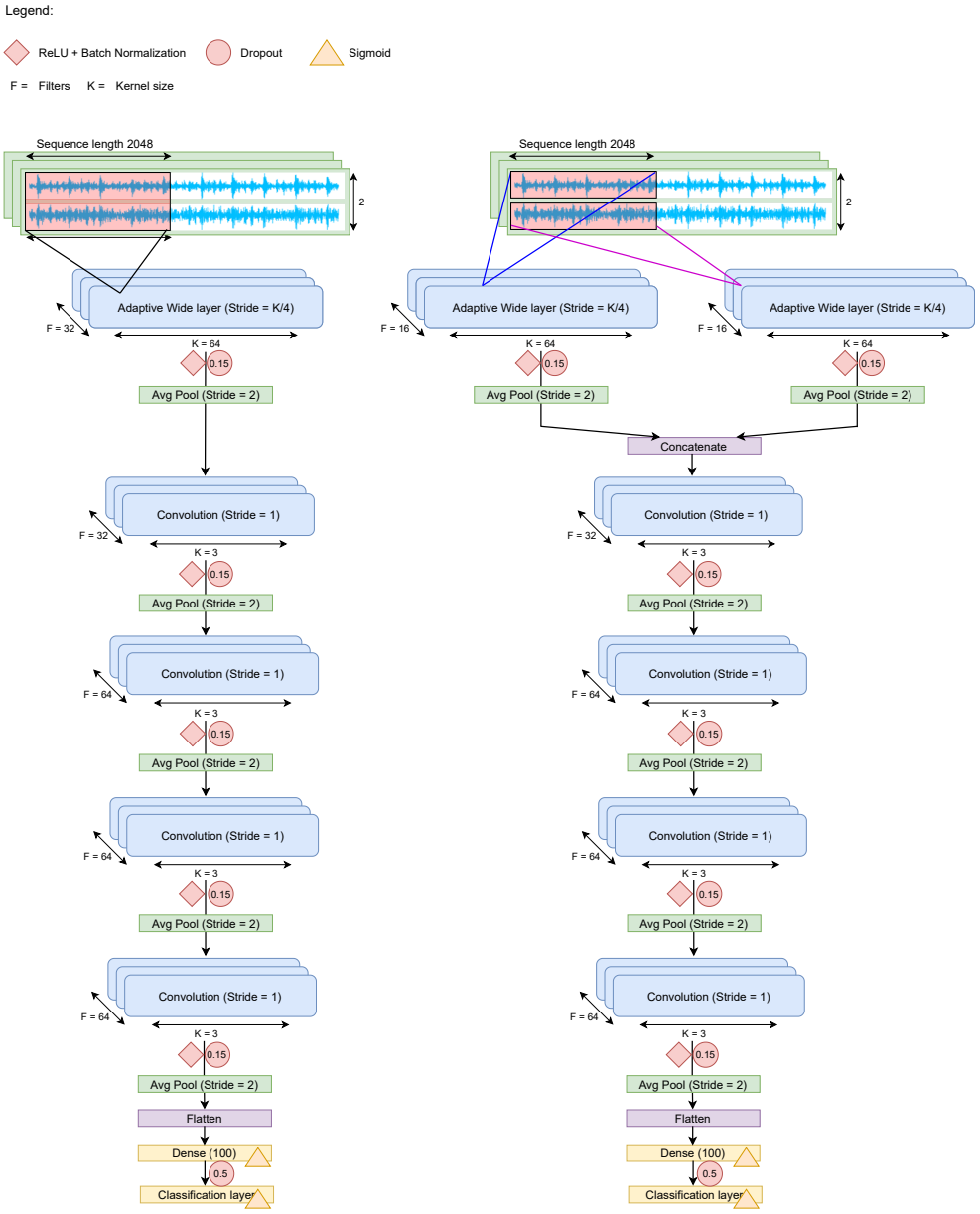


Figure 2. Left: A-WCNN model utilizing the adaptive wide-kernel layer on two time series. **Right**: Ada-WMCNN model that processes two times series in separate channels.

3.2. Adaptive Wide-Kernel Multichannel CNN (Ada-WMCNN)

As we have presented and shown in our previous work, a wide-kernel deep multichannel CNN (WDMTCNN) [11] is able to better process and classify signals due to its separate feature extraction between the different time series dimensions. The model is fed by separate inputs each representing a univariate time series. These time series are processed at the same time by the separate CNNs, which are concatenated in the last stages of the model, e.g., before the fully connected layers. This approach leads to completely independent feature representations of the separate time series, which has proven to perform better in [11]. One main issue with the multichannel approach is that the model is less

scalable than the non-multichannel versions, since the number of separate CNNs increases with every additional feature in the multivariate time series, resulting in a complex model with many more parameters. In many cases this leads to longer processing times while performance gains are not always evident.

We adapted the WDMTCNN by only implementing a separate CNN structure in the first wide-kernel layer. This optimized model is still able to extract these specific characteristics of every individual time series, e.g., features from low-frequency bands, but concatenates directly after, resulting in less parameters and is therefore less computationally heavy. To compare its performance properly with our previous multichannel model (WDMTCNN) that runs completely separate CNNs, which is called the *A-WMCNN* in this paper, we experimented with both model settings to show its behavior across multiple datasets and multiple experiments. In addition, our adapted multichannel model also exploits local average pooling throughout every convolutional layer, reducing the vector size with steps of two. A complete overview on Ada-WMCNN can be seen on the right side of Figure 2.

3.3. Parameter Settings

Besides the models' architecture, parameter settings are vital in finding the most optimal classification performance. In this section, we examine the parameter settings of both models.

- Normalization: The distribution of each layer's input varies throughout training, slowing down the process. Batch normalization is a technique for minimizing the influence of internal covariance on the training process, thus speeding up the training process [52]. We imputed batch normalization right after the convolutional layer. For each input x, batch normalization normalizes each dimension k and computes the training set's expectation and variance:

$$\widehat{x}^{(k)} = \frac{x^{(k)} - \mathrm{E}\left[x^{(k)}\right]}{\sqrt{\mathrm{Var}\left[x^{(k)}\right]}}. \tag{6}$$

  When it comes to nonlinear representations, normalizing each input may modify what the layer represents. For each activation, batch normalization has a shift and scale function to account for this constraint:

$$y^{(k)} = p^{(k)}\widehat{x}^{(k)} + q^{(k)}, \tag{7}$$

  where $p^{(k)}$ and $q^{(k)}$ scale and shift each activation $x^{(k)}$, respectively. These parameters are learnt in tandem with the model's initial parameters, restoring the network's representation power.
- Fully Connected Layer: Before the final classification layer, the models are provided with a fully connected layer. This layer is used to identify global compositions of the final convolutional output and is equivalent to a fully connected layer in a multilayer perceptron. The fully connected layer is denoted as follows:

$$FC^{l} = f\left(\omega^{l} * x^{l} + b^{l}\right), \tag{8}$$

  for layer l, where ω^{l} is the weight matrix and x^{l} is the input for the fully connected layer, represented as a flattened vector of the output from the previous pooling layer. A dot product operation is denoted by $*$, while a bias term is denoted by b^{l}. The Sigmoid activation function employed in the layer is represented by $f(\cdot)$.
- Classifier: In the last layer of both models, a Sigmoid classifier is used, which is expressed as:

$$\sigma(x_i) = \frac{e^{x_i}}{1 + e^{x_i}}, \tag{9}$$

where x is the value of the preceding layer's ith output, and e is Euler's mathematical constant. This results in output values that are independent of each other and are not constrained to sum up to 1. Therefore, we chose Sigmoid because it excels in classifying signals of any length with modest or no variations across classes, necessitating the calculation of each probability separately.

- Optimization: An Adam stochastic optimizer is used to optimize the networks. For each individual parameter, it makes use of the power of adaptive learning rates. The optimizer is computationally efficient and memory-light, making it ideal for models with a large number of trainable parameters that process high dimensional data. Adam is particularly well suited for noisy and non-stationary signals [53]. Adam optimization can be denoted as:

$$\begin{aligned} g_t &= \nabla_\theta f^t\left(\theta^{t-1}\right), \\ m_\theta^t &= \beta_1 m_\theta^{t-1} + \left(g_t - \beta_1 g_t\right), \\ v_\theta^t &= \beta_2 v_\theta^{t-1} + \left(g_t^2 - \beta_2 g_t^2\right), \\ \widehat{m}_\theta &= \frac{m_\theta^t}{1-(\beta_1)^t}, \\ \widehat{v}_\theta &= \frac{v_\theta^t}{1-(\beta_2)^t}, \\ \theta^t &= \theta^{t-1} - \alpha \cdot \frac{\widehat{m}_\theta}{\left(\sqrt{\widehat{v}_\theta}+\epsilon\right)}. \end{aligned} \tag{10}$$

  At epoch t, $f(\cdot)$ represents the stochastic objective function with parameters θ^t, yielding gradient g_t. The first and second biased moment estimates are designated as m_θ^t and v_θ^t, respectively, where the decay rates are $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The bias-corrected momentum is represented by $\widehat{m}_\theta$ and $\widehat{v}_\theta$. Using a default learning rate of $\alpha = 0.001$ and $\epsilon = 10^{-8}$, the corrected momentum is utilized to update the parameters θ^t.
- Loss Function: Both networks construct the loss function using mean squared error (MSE), which is more often employed in regression analysis than classification tasks. MSE, on the other hand, may produce fewer differences between values, which we think is beneficial for our experiments, since certain fault circumstances seem to be comparable. The MSE is calculated as the average of the squared differences between the projected $\hat{y}$ and actual y values, with greater discrepancies penalized by the model. MSE is expressed as follows:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2. \tag{11}$$

3.4. Software and Resources

In this research, we used Python combined with Tensorflow and Keras to develop the proposed models. In addition, all other algorithms, such as the machine learning models, are derived from the Sklearn package. Calculations were performed with the support of Numpy and table formatting with Pandas. For optimizing processing time, the data was standardized using the StandardScaler algorithm provided in the Sklearn package. Furthermore, to reduce the overall training time, the models were trained on a dedicated server with two Intel Xeon CPUs (3.2 GHz), 256 GB RAM and a Nvidia Quadro RTX 6000 (24 GB) GPU. After training, the models are fairly small (between 1 and 2 MB) and are deployable on standard PC hardware as well as edge computing platforms.

4. Results

To evaluate the proposed models, we used data from the Case Western Reserve University (CWRU) [54] and Paderborn University [8] reflecting bearing fault experiments

of rotating equipment. In this section, both experiments and datasets are described in more detail. Furthermore, all results are discussed in this paragraph.

4.1. CWRU Bearing Dataset

The CWRU bearing dataset represents a bearing fault experiment where damages were applied to the bearing element. To measure vibration signals, sensors were placed on the drive end and fan end of the machine; we refer to [54,55] for more detail. The test rig is also displayed at: https://engineering.case.edu/bearingdatacenter/apparatus-and-procedures; Figure 3 shows an example of the according signals. These vibrations are digitized into two time series that we segmented into sequences of 2048 data points without overlap. The data from the experiments used in this study is divided into two categories; 12k fan end and 12k drive end, both utilizing a sampling rate of 12 kHz. Both experiments contain different properties, resulting in deviating datasets with fluctuating amounts of classes. Within these datasets, different machine operations are used to measure the vibrations, respectively; 1797, 1750, 1730 and 1772 motor speed (revolutions per minute, RPM). Next to that, there is a normal condition for every motor speed.

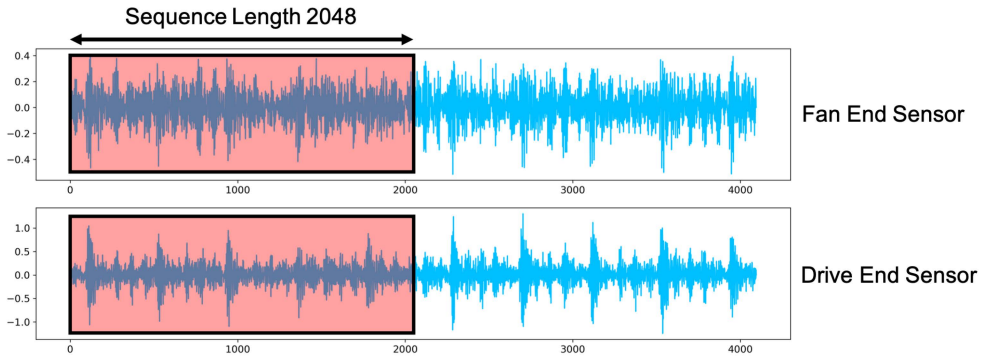


Figure 3. Example signal of the vibrations from both sensors derived from the CWRU Bearing dataset. The red box indicates the length of one data sample (sequence).

Overall, in many fault detection and condition monitoring studies, the CWRU bearing dataset [54,55] has been used, e.g., [10,43,45,56–58]. It can be considered as a de facto benchmark dataset for fault detection because it is publicly available and in principle modeled analogously to important industrial application settings, where data are typically not made openly available. Furthermore, the CWRU dataset is straightforward to interpret and analyze by combining the large number of supplied class labels, which are typically difficult for standard machine learning models to handle.

There are three different depths of damages inflicted to the bearing, 0.007, 0.014 and 0.021 inches, respectively. Each damage type has five distinct bearing fault locations (ball, inner race, outer race opposite, outer race orthogonal and outer race centered). To enhance the tasks' complexity and restrict the number of data samples per class, we opted to count each condition as a single class. Thus, resulting in a large-scale classification task where distinctions are made on fault level and machine condition. The data samples are made by segmenting the two time series into 2048-point sequences with no overlap. This sequence length has been extensively utilized in other research for efficient implementation of the fast Fourier transform (FFT) method, which is known as a powerful baseline [10,45,59]. Every fault condition has approximately the same number of samples per class. Except for the normal conditions, where the number of samples fluctuate. An overview of the dataset with its number of classes and samples can be seen in Table 1.

Table 1. Number of data samples (sequences) for every experiment type, with their respective motor speed (RPM) and the supplementary normal conditions.

Experiment Name	Motor Speed	Nr. of Classes	Samples per Class
12k Fan End	1797	13	59
	1750	12	59
	1730	12	59
	1772	12	59
12k Drive End	1797	14	59
	1750	14	59
	1730	14	59
	1772	14	59
Normal Condition	1797	1	119
	1750	1	236
	1730	1	237
	1772	1	236

Within these experiments, we applied varying combinations of the different motor speeds. Combining these different motor speeds leads to various numbers of samples, but also changes in the number of classes to classify, therefore increasing the complexity of the given classification task. In addition, we experimented with different percentages of training data, e.g., 80%, 40% and 20% training data, to exhibit the behavior of our model under different circumstances. Furthermore, we used k-fold cross validation with k set to 5 to improve the generalizability of the trained models. The classification accuracy of the models is calculated by averaging the accuracy across every fold. However, within every fold, we predicted on the same unseen test set. We chose accuracy as the most important metric due to the multi-class classification task with mostly balanced data (except the normal conditions), as used in many previous studies [10,43,45].

4.2. Paderborn Bearing Dataset

The Paderborn bearing dataset can also be seen as a benchmark for fault detection and condition monitoring of damaged rolling bearing elements of rotating equipment, cf. [20,42]. The dataset represents motor current signals of an electromechanical drive system and vibrations of the housing [8]. The signals can be extracted in the existing frequency inverters. Therefore, no additional sensors needed to be placed on the system, as was the case in the CWRU bearing experiments, resulting in more resource-efficient experimentation, thus less expensive. Monitoring damages in external bearings, which are positioned in the drive system but outside the electric motor, is a unique feature of the current method. Regardless, the motor current signal was employed as an input for fault detection.

In total, the data derived from the experiments represents “healthy”, “real damaged” and “artificially damaged” bearings. Data were recorded for approximately 4 s with a sampling rate of 64 kHz, resulting in many separate data files with approximately 256 thousand data points. We concatenated the data files based on their specific bearing conditions, e.g., “healthy”, “real damaged” and “artificially damaged” bearings. Due to the many different machine settings, we decided not to deviate between these, and only look at the specific bearing fault condition. This resulted in a large multi-class classification task and a large-scale dataset containing many sequences of length 2048, similar to the CWRU data.

However, in contrast to the CWRU data, the number of classes is a bit lower, the data between every single class are more balanced and due to the high number of sequences, the training time of the models is much higher. Furthermore, the Paderborn Bearing dataset contains three time series, e.g., two motor current signals and one vibration signal, that are taken into account during processing. Therefore, the first wide-kernel layer adapts itself to

the input data, resulting in a slightly changed architecture compared with our models on the CWRU data.

For this experiment we also used k-fold cross validation with $k = 5$ to improve the generalizability of the trained models and tested models' performance with different train splits. However, due to the larger number of available samples, we attempted to complicate the classification task by lowering the train split even further, e.g., with only 10% and 5% training data. Table 2 provides an overview on the different conditions with their respective number of samples.

Table 2. Number of data samples (sequences) for every bearing condition, with their respective number of classes and samples per class.

Bearing Condition	Nr. of Classes	Samples per Class
Healthy	6	10.000
Real damage	14	10.000
Artificial damage	12	≈10.000

4.3. Machine Learning Algorithms

Our proposed models are compared with traditional machine learning (ML) algorithms such as k-nearest neighbors (K-NN), random forest (RF) and support vector machine (SVM). These models do not lend themselves for processing raw signals, which means that we preprocessed the data into a set of features in both the time and frequency domain. The features used to feed the ML models are derived from various studies, e.g., [60,61] and are described in detail in Table 3. All of these features were used in the ML models and are calculated across the sequences (with length 2048). Features in the frequency-domain are calculated after transforming the time series to the frequency spectrum using the FFT algorithm that computes the one-dimensional discrete Fourier transform (DFT) with backward normalization, which is known as a potent baseline [10,45,59]. In total, the feature set consists of nine different features for every one of the separate time series.

Table 3. Set of features used for the ML algorithms with their formula and description.

Features	Formula	Description
Time-Domain		
Mean	$\mu_x = \frac{1}{t}\left(\sum_{i=1}^{t} x_i\right)$	The average value of sequence $x = (x_i), i = 0 \dots t-1$.
Standard Deviation	$\sigma_x = \frac{1}{t-1}\sum_{i=0}^{t-1}(x_i - \mu)^2$	The standard deviation of sequence $x = (x_i), i = 0 \dots t-1$.
Variance	$\sum_{i=1}^{t} \frac{x_i - \mu_x}{t}$	Square of standard deviation.
Median	$\text{Median}_x = x_{(n+1)/2}$	Median value of a sequence x given by x_i as above.
Minimum	Min_x	Minimum value of sequence x given by x_i as above.
Maximum	Max_x	Maximum value of sequence x given by x_i as above.
Range	$\text{Range} = \text{Max}_x - \text{Min}_x$	Difference between the maximum and minimum value.
Frequency-Domain		
Signal Energy	$\text{Peng} = \sum(\text{fft}\, x_i)^2$	Energy of a signal calculated using FFT.
Signal Power	$\text{Psig} = \sum \frac{(\text{fft}\, x_i)^2}{\sum_i^t}$	Power of a signal calculated using FFT.

For all ML models, we used grid search optimization with five-fold cross-validation derived from the Sklearn package to assess which model performs best for every experiment. The parameter settings of the best performing model vary throughout the different experiments and datasets used. In Table 4 the properties of the grid search optimization are described. In addition, we compared the models with the standard WDCNN models as proposed in [10]. The results are described per dataset since these sets have different properties and classification conditions. Further, we distinguished between different percentages of training data to see how well the ML models perform under varying circumstances regarding data availability.

Table 4. Overview of parameter settings used in grid search optimization for every ML model using Sklearn package.

Model	Parameter	Grid Search Range
K-NN	K	1–20
	Weight Options	Uniform or Distance
RF	Nr of Estimators	100–1000 (100 per step)
	Feauture Estimation	Square Root or Log2
SVM	C	10–40 (5 per step)
	Gamma	0.0001, 0.001
	Kernel	Linear or Radial Basis Function

4.4. Results CWRU Dataset

In general, the results on the CWRU dataset in Table 5 clearly show that in almost every case the deep learning applications outperform traditional ML approaches, therefore emphasizing on the power of automated learning approaches in fault detection and condition monitoring. Overall, *Ada-WMCNN* shows the best performance on the different experimental settings suggesting that using a multichannel structure is particularly useful for the *first* wide-kernel layer. However, compared to the *A-WMCNN*, the margins are really close. Therefore, choosing a multichannel approach seems to be well-suited for the task at hand. In that case, one should look at the amount of computational resources needed for the model, which is significantly less for the *Ada-WMCNN* due to its fewer trainable parameters. This model merges the separate inputs in an earlier stage, compared to the *A-WMCNN*, resulting in less convolutional operations.

Table 5. Test accuracy scores (%) averaged across the 5-fold cross validation for each of the models with their respective train split on the CWRU Bearing dataset.

Experiment Type	Motor Speed	Number of Sequences	Training Data (%)	Accuracy						
				K-NN	**RF**	**SVM**	**WDCNN**	**A-WCNN**	**A-WMCNN**	**Ada-WMCNN**
Drive End	1797/1750	1.889	80	85.71	97.35	97.62	99.38	**100**	**100**	**100**
			40	82.98	94.27	91.62	95.62	97.72	97.34	**97.74**
			20	73.88	**93.65**	87.37	56.53	76.79	93.20	93.44
	1797/1750 1772	2.892	80	63.56	81.87	81.69	87.89	**89.86**	89.46	89.59
			40	57.89	80.13	77.82	80.35	70.13	**85.31**	84.93
			20	52.33	77.18	69.88	66.20	77.39	**77.56**	65.22
	1797/1750 1730	2.894	80	74.44	94.13	93.61	99.69	99.97	99.80	**100**
			40	67.47	89.81	87.16	91.13	96.07	96.29	**96.79**
			20	58.38	86.57	78.76	68.10	87.46	87.41	**90.32**
	1797/1750 1730/1772	3.897	80	58.59	81.28	81.15	91.99	92.68	**93.31**	91.67
			40	53.95	78.58	75.07	81.51	88.44	**89.58**	86.55
			20	47.47	**74.15**	66.04	39.47	42.04	65.63	71.58
Fan End	1797/1750	1.710	80	83.33	90.64	87.43	99.43	**99.94**	99.83	**99.94**
			40	76.02	87.91	83.43	97.69	99.25	99.19	**99.87**
			20	69.74	84.21	77.63	40.55	60.98	**90.56**	72.66
	1797/1750 1772	2.593	80	61.85	73.60	71.10	85.27	89.92	89.28	**90.15**
			40	55.27	70.05	64.91	77.89	84.16	**86.61**	85.57
			20	49.69	66.22	59.18	53.43	68.03	**77.70**	55.85
	1797/1750 1730	2.595	80	69.36	83.82	80.73	99.39	99.85	99.92	**99.96**
			40	56.07	82.66	74.63	94.40	98.34	98.07	**99.49**
			20	54.87	79.82	64.98	59.07	86.88	91.67	**92.46**
	1797/1750 1730/1772	3.478	80	55.32	73.85	69.97	91.75	92.85	92.15	**92.94**
			40	47.29	69.81	64.54	82.96	88.86	89.87	**90.76**
			20	40.71	66.12	52.71	38.04	54.08	55.64	**68.37**

In addition, the results for all models demonstrate that lowering the amount of training data will almost always result in a lower performance on the test set, especially for the more complicated classification tasks e.g., with all types of motor speed (RPM). These results clearly indicate the importance of sufficient training samples for fault detection and condition monitoring. For this particular dataset (CWRU), we expected this behavior due

to the sparsity of the amount of sequences available. Especially with just 20% train data, the number of samples per class is in the most complex case lower than 20 samples resulting in major performance drops, sometimes even below 50% accuracy. This phenomenon can be better observed in Figure 4 where the results of the CNNs are averaged across all the CWRU bearing experiments for their given train split.

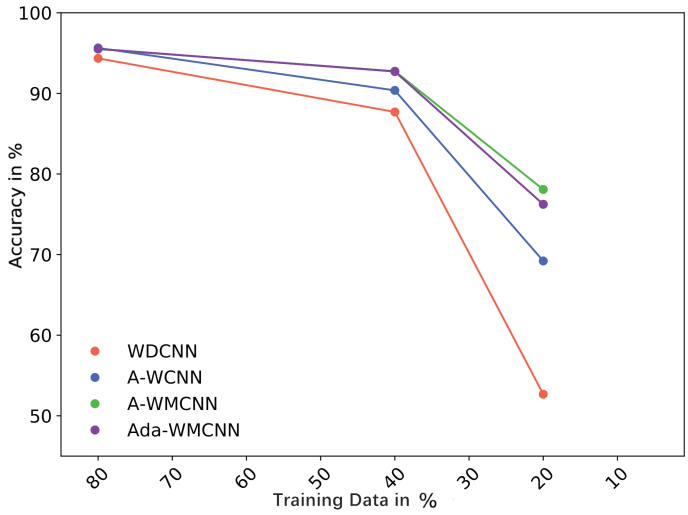


Figure 4. Averaged accuracy scores (%) across all CWRU experiments with their respective train split (%).

4.5. Results Paderborn Dataset

As can be seen in Table 6, the results on the Paderborn dataset show different characteristics compared to the results on the CWRU dataset. One of the most prominent observations is the consistency in the results across multiple classification tasks with different train splits, which can also be observed when averaging the accuracy scores of the CNNs across all experiments (see Figure 5). This particular behavior is ought to be caused by three factors: first, the amount of data available is much higher than for the CWRU experiments, even with lower train splits; second, there are fewer classes to choose from, indicating that the classification task at hand is slightly easier; third, the data consist of three signals instead of two representing motor currents and vibrations.

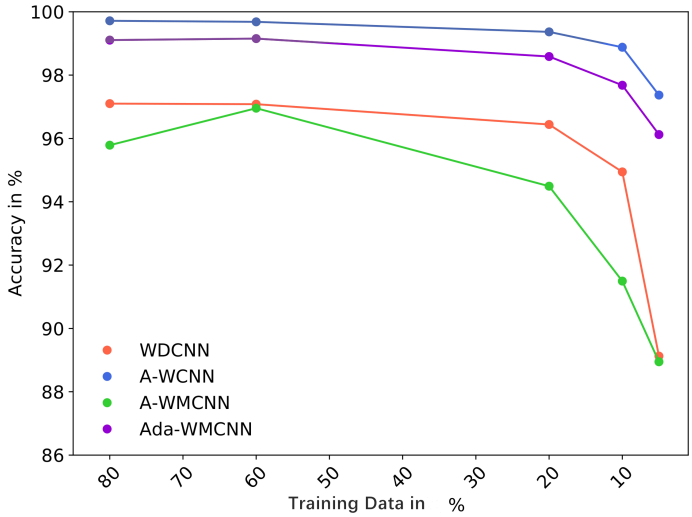


Figure 5. Averaged accuracy scores (%) across all Paderborn experiments with their respective train split (%).

Another remarkable observation is the much higher performance of the traditional ML approaches compared to the results in the CWRU experiments. It seems that the extracted features are particularly well-suited for motor current signals when detecting bearing faults, especially in the case of the Random Forest algorithm. However, the results clearly reveal that in this case the *A-WCNN* model is the favorite over the other models, showing the highest accuracy performance in every single experiment, followed by the *Ada-WMCNN* that runs closely behind. This result suggests that having an overarching first wide-kernel layer with many filters performs slightly better compared to the multichannel separation in the *Ada-WMCNN* model, while completely separating the signals and feed them as univariate time series results in lower performance (results from the *A-WMCNN* model). A reason for this might lie in the nature of the data where motor current signals tend to be more correlated than separately installed vibration sensors as used in the CWRU bearing experiments.

Table 6. Test accuracy scores (%) averaged across the 5-fold cross validation for each of the models with their respective train split on the Paderborn Bearing dataset.

Experiment Combination	Number of Sequences	Training Data (%)	Accuracy						
			K-NN	RF	SVM	WDCNN	A-WCNN	A-WMCNN	Ada-WMCNN
Real/Healthy	200.000	80	97.34	98.25	90.14	99.24	**99.96**	98.99	99.67
		60	97.09	98.00	90.02	99.31	**99.94**	98.99	99.71
		20	95.72	97.20	89.49	98.75	**99.75**	98.56	99.55
		10	94.44	96.50	88.90	97.84	**99.61**	95.78	99.06
		5	92.40	95.39	87.82	97.23	**99.03**	93.70	98.10
Real/Artificial	259.875	80	90.61	93.01	77.18	97.53	**99.80**	96.58	99.45
		60	89.91	92.43	77.10	97.05	**99.74**	96.18	99.35
		20	86.14	90.21	76.53	96.75	**99.36**	92.82	98.56
		10	83.34	88.53	75.79	95.99	**98.82**	91.42	97.39
		5	79.50	86.71	74.50	93.36	**97.60**	87.85	97.21
Healthy/Artificial	179.875	80	89.19	92.28	77.50	97.01	**99.63**	95.41	98.55
		60	88.37	91.67	77.68	96.20	**99.69**	98.04	98.96
		20	85.21	89.68	77.15	96.30	**99.20**	91.33	98.06
		10	82.52	87.92	76.81	93.92	**98.46**	90.29	97.51
		5	78.71	85.90	76.58	91.50	**95.87**	90.08	93.86
Real/Artificial/Healthy	319.875	80	90.64	93.16	78.17	94.63	**99.47**	92.17	98.75
		60	89.98	92.69	78.20	95.77	**99.36**	94.61	98.59
		20	86.85	90.78	77.80	93.96	**99.15**	95.26	98.18
		10	83.98	89.25	77.00	92.02	**98.63**	88.49	96.76
		5	80.58	87.14	75.70	74.41	**96.98**	84.16	95.32

4.6. Overall Model Performance

The results displayed in Tables 5 and 6 show the performance of the models for every different experiment. However, to obtain a better understanding in general, we calculated the mean accuracy score across all experiments for both datasets to give an indication of the overall performance, regardless of the experiment combinations related to train splits and machine conditions. In addition, we describe the standard deviation to explain the stability of the models' performance, therefore making some assumptions regarding the generalizability of the models. Furthermore, for the CNNs, the model size in terms of parameters and computation speed based on GPU acceleration for every epoch is described in Table 7.

As can be seen in Table 8, the proposed adaptive models clearly outperform any other model. However, it seems that it depends on the properties of the dataset which model performs best. For example, the *A-WMCNN* performs most optimal on the CWRU dataset, also with the second lowest standard deviation, while having a lower performance on the Paderborn dataset, with a relatively high standard deviation. Vice versa, we see similar behavior for the *A-WCNN* model.

Table 7. Number of parameters and computation speed, expressed in milliseconds per epoch (Ms/step) using GPU acceleration, for the CNN models.

Model	Experiment	Parameters	Ms/step
WDCNN	CWRU	54.804	15.50
	Paderborn	55.828	16.25
A-WCNN	CWRU	58.468	15.88
	Paderborn	65.204	16.50
A-WMCNN	CWRU	107.460	23
	Paderborn	161.140	29.25
Ada-WMCNN	CWRU	56.420	17
	Paderborn	59.060	18.75

Another interesting observation is the performance of the optimized multichannel model (*Ada-WMCNN*). While performing a bit worse than the best performing model in both experiments, its averaged overall performance is better. Further, as Table 7 suggests, this model has less parameters and a competitive computation speed compared to the other adaptive CNNs. Therefore, according to this study, the *Ada-WMCNN* is well-suited for a variety of fault detection and condition monitoring tasks.

Finally, it is worth mentioning that even though the ML models perform considerably worse than the deep learning approaches, the random forest (RF) algorithm seems to mark the highest performance with a fairly low standard deviation, indicating that this ML model is surprisingly stable under varying circumstances. The RF algorithm clearly outperforms the other ML models, which may suggest that tree-based algorithms are well-suited for fault detection and condition monitoring tasks—provided that the data are preprocessed into a rich feature set. This is one of the directions to pursue in future work.

Table 8. Averaged test accuracy scores (%) for both datasets (with standard deviation).

Experiment	K-NN	RF	SVM	WDCNN	A-WCNN	A-WMCNN	Ada-WMCNN
CWRU	62.34	81.57	76.63	78.24	85.07	**89.39**	88.16
	(12.34)	*(9.09)*	*(11.32)*	*(20.54)*	*(15.85)*	*(10.99)*	*(12.49)*
Paderborn	88.13	91.84	80.00	94.94	**99.00**	93.54	98.13
	(5.65)	*(3.78)*	*(5.58)*	*(5.31)*	*(1.07)*	*(4.06)*	*(1.49)*
Overall	74.06	86.24	78.16	85.83	91.40	91.27	**92.69**
	(16.26)	*(8.79)*	*(9.23)*	*(17.58)*	*(13.57)*	*(8.73)*	*(10.47)*

5. Discussion

This work extends on our previous research [11] to investigate the performance of our proposed models utilizing an adaptive wide-kernel in the first convolutional layer in fault detection and condition monitoring tasks. This adaptive layer is able to scale towards varying dimensionalities of time series data, while maintaining its core task for extracting valuable features without representation loss.

With this adaptive layer we optimized our previous proposed models as described in [11] resulting in two models, respectively *A-WCNN* and *Ada-WMCNN*. The first model initializes the adaptive wide-kernel layer followed by small kernel layers, similar to the previous proposed WDTCNN model. The *Ada-WMCNN* is able to process multivariate time series in a univariate way in the first wide-kernel layer. This model has a similar principle as the WDMTCNN model proposed in [11], but is more computationally efficient due to the lower number of trainable parameters and higher computation speed, as can be seen in Table 7. We compared the models with many traditional machine learning applications (optimized with a grid search) as well as with the original WDCNN proposed by [10] and our completely separate multichannel CNN (*A-WMCNN*).

The proposed models are tested in a wide range of experiments on two well-known bearing fault datasets; the CWRU Bearing dataset, a dataset with few samples and many different fault and machine conditions and the Paderborn Bearing dataset, a large-scale dataset with a vast number of samples and fewer fault conditions. Compared to the ML models and the original WDCNN, our optimized models demonstrated a better performance; therefore, concluding that an adaptive wide-kernel layer is a valuable addition to existing 1D CNN architectures. However, between the different settings for the adaptive models, it seems that highest performance depends on the properties of the dataset, e.g., the best performing model deviates per dataset. Overall, we can conclude that the *Ada-WMCNN* model performs best throughout all experiments suggesting that a separate multichannel in the first wide-kernel layer is effective and adaptable to changing environments. Furthermore, both models handle raw signals directly and are quite good in detecting faults and distinguish between machine conditions, even in cases with minimal training data.

On another note, in this research our models are only trained on experimental data that are carefully recorded and annotated, which is usually difficult to obtain in real-world settings. Thus, generalization of the trained models in real-world settings is difficult to assess, also due to scarcity of available data. Furthermore, it seems in many cases, especially in the results on the Paderborn dataset, that the classification task is fairly easy for the models to perform. Therefore, for future research, we intend to apply transfer learning by training the models on experiment data and test their performance on real-world data to assess the generalizability of the trained models in other contexts. However, in this particular situation, the properties of the data should be similar, e.g., data derived from vibration sensors or motor current signals. Further, we aim to investigate the performance of the adaptive models in a context with variable sequence lengths. In addition, we plan to research the application of attention mechanisms in time series classification tasks as described in [62,63] and implement these mechanisms in fault detection and condition monitoring tasks.

Author Contributions: Conceptualization, J.v.d.H. and M.A.; methodology, J.v.d.H.; software, J.v.d.H.; validation, J.v.d.H.; formal analysis, J.v.d.H.; investigation, J.v.d.H.; resources, J.v.d.H.; data curation, J.v.d.H.; writing—original draft preparation, J.v.d.H.; writing—review and editing, J.v.d.H., S.B. and M.A.; visualization, J.v.d.H. and S.B.; supervision, M.A.; funding acquisition, M.A. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been funded by the Interreg North-West Europe program (Interreg NWE), project Di-Plast—Digital Circular Economy for the Plastics Industry (NWE729).

Data Availability Statement: In this research, the following two publicly available datasets were used; the Case Western Reserve University Bearing data (https://csegroups.case.edu/bearingdatacenter/pages/download-data-file), accessed on 14 January 2020 and the Paderborn University Bearing data (https://mb.uni-paderborn.de/en/kat/main-research/datacenter/bearing-datacenter/data-sets-and-download), accessed on 20 May 2021.

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

Abbreviations

The following abbreviations are used in this manuscript:

A-WCNN	Adaptive Wide-Kernel CNN
Ada-WMCNN	Adaptive Wide-Kernel Multichannel CNN
CBM	Condition-based Maintenance
CNN	Convolutional Neural Network
CWRU	Case Western Reserve University
K-NN	K-Nearest Neightbor

LSTM	Long Short-term Memory
ML	Machine Learning
MLP	Multi-layer Perceptron
1D	One-dimensional
ReLU	Rectified Linear Unit
RF	Random Forest
RNN	Recurrent Neural Network
RPM	Revolutions per Minute
SVM	Support Vector Machine
UBM	Usage-based Maintenance
WDCNN	Wide-kernel Deep CNN
WDMTCNN	Wide-kernel Deep Multichannel Temporal CNN
WDTCNN	Wide-kernel Deep Temporal CNN

References

1. Lasi, H.; Fettke, P.; Kemper, H.G.; Feld, T.; Hoffmann, M. Industry 4.0. *Bus. Inf. Syst. Eng.* **2014**, *6*, 239–242. [CrossRef]
2. Atzmueller, M.; Kloepper, B.; Mawla, H.A.; Jäschke, B.; Hollender, M.; Graube, M.; Arnu, D.; Schmidt, A.; Heinze, S.; Schorer, L.; et al. Big Data Analytics for Proactive Industrial Decision Support. *atp Ed.* **2016**, *58*, 62–74. [CrossRef]
3. Zhao, D.; Wang, T.; Chu, F. Deep convolutional neural network based planet bearing fault classification. *Comput. Ind.* **2019**, *107*, 59–66. [CrossRef]
4. Zhao, R.; Yan, R.; Chen, Z.; Mao, K.; Wang, P.; Gao, R.X. Deep learning and its applications to machine health monitoring. *Mech. Syst. Signal Process.* **2019**, *115*, 213–237. [CrossRef]
5. Kim, S.; An, D.; Choi, J.H. Diagnostics 101: A Tutorial for Fault Diagnostics of Rolling Element Bearing Using Envelope Analysis in MATLAB. *Appl. Sci.* **2020**, *10*, 7302. [CrossRef]
6. Immovilli, F.; Bianchini, C.; Cocconcelli, M.; Bellini, A.; Rubini, R. Bearing fault model for induction motor with externally induced vibration. *IEEE Trans. Ind. Electron.* **2012**, *60*, 3408–3418. [CrossRef]
7. Lei, Y.; Jia, F.; Lin, J.; Xing, S.; Ding, S.X. An intelligent fault diagnosis method using unsupervised feature learning towards mechanical big data. *IEEE Trans. Ind. Electron.* **2016**, *63*, 3137–3147. [CrossRef]
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. In Proceedings of the PHM Society European Conference, Bilbao, Spain, 5–8 July 2016; Volume 3.
9. 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]
10. Zhang, W.; Peng, G.; Li, C.; Chen, Y.; Zhang, Z. A new deep learning model for fault diagnosis with good anti-noise and domain adaptation ability on raw vibration signals. *Sensors* **2017**, *17*, 425. [CrossRef]
11. van den Hoogen, J.; Bloemheuvel, S.; Atzmueller, M. An Improved Wide-Kernel CNN for Classifying Multivariate Signals in Fault Diagnosis. In Proceedings of the 2020 International Conference on Data Mining Workshops (ICDMW), Sorrento, Italy, 17–20 November 2020; pp. 275–283. [CrossRef]
12. Mobley, R.K. *An Introduction to Predictive Maintenance*; Elsevier: Amsterdam, The Netherlands, 2002.
13. Alsyouf, I. The role of maintenance in improving companies' productivity and profitability. *Int. J. Prod. Econ.* **2007**, *105*, 70–78. [CrossRef]
14. Jardine, A.K.; Lin, D.; Banjevic, D. A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mech. Syst. Signal Process.* **2006**, *20*, 1483–1510. [CrossRef]
15. Arts, J. *Maintenance Modeling and Optimization*; TU/e Eindhoven: Eindhoven, The Netherlands , 2017; Volume 526.
16. Vollert, S.; Atzmueller, M.; Theissler, A. Interpretable Machine Learning: A brief survey from the predictive maintenance perspective. In Proceedings of the IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2021), Vasteras, Sweden, 7–10 September 2021.
17. Vollert, S.; Theissler, A. Challenges of machine learning-based RUL prognosis: A review on NASA's C-MAPSS data set. In Proceedings of the IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2021), Vasteras, Sweden, 7–10 September 2021.
18. Motor Reliability Working Group. Report of large motor reliability survey of industrial and commercial installations, Part I. *IEEE Trans. Ind. Appl.* **1985**, *1*, 865–872.
19. Yin, S.; Li, X.; Gao, H.; Kaynak, O. Data-based techniques focused on modern industry: An overview. *IEEE Trans. Ind. Electron.* **2014**, *62*, 657–667. [CrossRef]
20. 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 (PHM-Paris), Paris, France, 2–5 May 2019; pp. 320–326. [CrossRef]
21. You, D.; Gao, X.; Katayama, S. WPD-PCA-based laser welding process monitoring and defects diagnosis by using FNN and SVM. *IEEE Trans. Ind. Electron.* **2014**, *62*, 628–636. [CrossRef]

22. 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]
23. Atzmueller, M.; Hayat, N.; Schmidt, A.; Klöpper, B. Explanation-Aware Feature Selection using Symbolic Time Series Abstraction: Approaches and Experiences in a Petro-Chemical Production Context. In Proceedings of the IEEE International Conference on Industrial Informatics, Emden, Germany, 24–26 July 2017; IEEE: Boston, MA, USA, 2017.
24. Malhi, A.; Gao, R.X. PCA-based feature selection scheme for machine defect classification. *IEEE Trans. Instrum. Meas.* **2004**, *53*, 1517–1525. [CrossRef]
25. Zhang, X.; Xu, R.; Kwan, C.; Liang, S.Y.; Xie, Q.; Haynes, L. An integrated approach to bearing fault diagnostics and prognostics. Proceedings of the 2005 IEEE American Control Conference, Portland, OR, USA, 8–10 June 2005; pp. 2750–2755.
26. Jin, X.; Zhao, M.; Chow, T.W.; Pecht, M. Motor bearing fault diagnosis using trace ratio linear discriminant analysis. *IEEE Trans. Ind. Electron.* **2013**, *61*, 2441–2451. [CrossRef]
27. Jiang, Q.; Shen, Y.; Li, H.; Xu, F. New fault recognition method for rotary machinery based on information entropy and a probabilistic neural network. *Sensors* **2018**, *18*, 337. [CrossRef]
28. Pandya, D.; Upadhyay, S.; Harsha, S.P. Fault diagnosis of rolling element bearing with intrinsic mode function of acoustic emission data using APF-KNN. *Expert Syst. Appl.* **2013**, *40*, 4137–4145. [CrossRef]
29. Zhou, Z.; Wen, C.; Yang, C. Fault Detection Using Random Projections and k-Nearest Neighbor Rule for Semiconductor Manufacturing Processes. *IEEE Trans. Semicond. Manuf.* **2015**, *28*, 70–79. [CrossRef]
30. Huang, J.; Hu, X.; Yang, F. Support vector machine with genetic algorithm for machinery fault diagnosis of high voltage circuit breaker. *Measurement* **2011**, *44*, 1018–1027. [CrossRef]
31. Konar, P.; Chattopadhyay, P. Bearing fault detection of induction motor using wavelet and Support Vector Machines (SVMs). *Appl. Soft Comput.* **2011**, *11*, 4203–4211. [CrossRef]
32. Santos, P.; Villa, L.F.; Reñones, A.; Bustillo, A.; Maudes, J. An SVM-based solution for fault detection in wind turbines. *Sensors* **2015**, *15*, 5627–5648. [CrossRef]
33. Chow, M..; Mangum, P.M.; Yee, S.O. A neural network approach to real-time condition monitoring of induction motors. *IEEE Trans. Ind. Electron.* **1991**, *38*, 448–453. [CrossRef]
34. Cococcioni, M.; Lazzerini, B.; Volpi, S.L. Robust Diagnosis of Rolling Element Bearings Based on Classification Techniques. *IEEE Trans. Ind. Inform.* **2013**, *9*, 2256–2263. [CrossRef]
35. Wang, Z.; Zhang, Q.; Xiong, J.; Xiao, M.; Sun, G.; He, J. Fault Diagnosis of a Rolling Bearing Using Wavelet Packet Denoising and Random Forests. *IEEE Sens. J.* **2017**, *17*, 5581–5588. [CrossRef]
36. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. Learning representations by back-propagating errors. *Nature* **1986**, *323*, 533–536. [CrossRef]
37. Malhotra, P.; Ramakrishnan, A.; Anand, G.; Vig, L.; Agarwal, P.; Shroff, G. LSTM-based encoder-decoder for multi-sensor anomaly detection. *arXiv* **2016**, arXiv:1607.00148
38. Yao, P.; Yang, S.; Li, P. Fault Diagnosis Based on RseNet-LSTM for Industrial Process. In Proceedings of the 2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), Chongqing, China, 12–14 March 2021; Volume 5, pp. 728–732. [CrossRef]
39. Liu, H.; Zhou, J.; Zheng, Y.; Jiang, W.; Zhang, Y. Fault diagnosis of rolling bearings with recurrent neural network-based autoencoders. *ISA Trans.* **2018**, *77*, 167–178. [CrossRef]
40. Hajnayeb, A.; Ghasemloonia, A.; Khadem, S.; Moradi, M. Application and comparison of an ANN-based feature selection method and the genetic algorithm in gearbox fault diagnosis. *Expert Syst. Appl.* **2011**, *38*, 10205–10209. [CrossRef]
41. Zhang, W.; Peng, G.; Li, C. Rolling element bearings fault intelligent diagnosis based on convolutional neural networks using raw sensing signal. In *Advances in Intelligent Information Hiding and Multimedia Signal Processing*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 77–84.
42. Liu, Y.; Yan, X.; Zhang, C.A.; Liu, W. An Ensemble Convolutional Neural Networks for Bearing Fault Diagnosis Using Multi-Sensor Data. *Sensors* **2019**, *19*, 5300. [CrossRef]
43. Chen, R.; Huang, X.; Yang, L.; Xu, X.; Zhang, X.; Zhang, Y. Intelligent fault diagnosis method of planetary gearboxes based on convolution neural network and discrete wavelet transform. *Comput. Ind.* **2019**, *106*, 48–59. [CrossRef]
44. Ince, T.; Kiranyaz, S.; Eren, L.; Askar, M.; Gabbouj, M. Real-time motor fault detection by 1-D convolutional neural networks. *IEEE Trans. Ind. Electron.* **2016**, *63*, 7067–7075. [CrossRef]
45. Zhang, A.; Li, S.; Cui, Y.; Yang, W.; Dong, R.; Hu, J. Limited data rolling bearing fault diagnosis with few-shot learning. *IEEE Access* **2019**, *7*, 110895–110904. [CrossRef]
46. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
47. LeCun, Y.; Bengio, Y. Convolutional networks for images, speech, and time series. In *The Handbook of Brain Theory and Neural Networks*; MIT Press: Cambridge, MA, USA, 1995; Volume 3361.
48. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
49. Vidaña-Vila, E.; Navarro, J.; Borda-Fortuny, C.; Stowell, D.; Alsina-Pagès, R.M. Low-Cost Distributed Acoustic Sensor Network for Real-Time Urban Sound Monitoring. *Electronics* **2020**, *9*, 2119. [CrossRef]
50. Hoang, D.T.; Kang, H.J. Rolling element bearing fault diagnosis using convolutional neural network and vibration image. *Cogn. Syst. Res.* **2019**, *53*, 42–50. [CrossRef]

51. Gulcehre, C.; Moczulski, M.; Denil, M.; Bengio, Y. Noisy activation functions. In Proceedings of the International Conference on Machine Learning, PMLR, New York, NY, USA, 20–22 June 2016; pp. 3059–3068.
52. Ioffe, S.; Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv* **2015**, arXiv:1502.03167.
53. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
54. CWRU Dataset; Case Western Reserve University Bearing Data Center. Available online: https://csegroups.case.edu/bearingdatacenter/home (accessed on 14 January 2020).
55. Ocak, H.; Loparo K. A. Estimation of the Running Speed and Bearing Defect Frequencies of an Induction Motor from Vibration Data. *Mech. Syst. Signal Process.* **2004**, *8*, 515–533. [CrossRef]
56. Neupane, D.; Seok, J. Bearing Fault Detection and Diagnosis Using Case Western Reserve University Dataset with Deep Learning Approaches: A Review. *IEEE Access* **2020**, *8*, 93155–93178. [CrossRef]
57. Liu, X.; Huang, H.; Xiang, J. A personalized diagnosis method to detect faults in a bearing based on acceleration sensors and an FEM simulation driving support vector machine. *Sensors* **2020**, *20*, 420. [CrossRef]
58. Piltan, F.; Kim, J.M. SVM-Based Hybrid Robust PIO Fault Diagnosis for Bearing. In Proceedings of the International Conference on Intelligent and Fuzzy Systems, Istanbul, Turkey, 21–23 July 2020; Springer: Berlin/Heidelberg, Germany, 2020; pp. 858–866.
59. Tong, Z.; Li, W.; Zhang, B.; Zhang, M. Bearing fault diagnosis based on domain adaptation using transferable features under different working conditions. *Shock Vib.* **2018**, *2018*. [CrossRef]
60. Mazilu, S.; Calatroni, A.; Gazit, E.; Roggen, D.; Hausdorff, J.M.; Tröster, G. Feature learning for detection and prediction of freezing of gait in Parkinson's disease. In Proceedings of the International Workshop on Machine Learning and Data Mining in Pattern Recognition, New York, NY, USA, 19–25 July 2013; Springer: Berlin/Heidelberg, Germany, 2013; pp. 144–158.
61. Masiala, S.; Huijbers, W.; Atzmueller, M. Feature-Set-Engineering for Detecting Freezing of Gait in Parkinson's Disease using Deep Recurrent Neural Networks. *arXiv* **2019**, arXiv:1909.03428.
62. Tang, Y.; Xu, J.; Matsumoto, K.; Ono, C. Sequence-to-Sequence Model with Attention for Time Series Classification. In Proceedings of the 2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW), Barcelona, Spain, 12–15 December 2016; pp. 503–510. [CrossRef]
63. Du, Q.; Gu, W.; Zhang, L.; Huang, S.L. Attention-based LSTM-CNNs for time-series classification. In Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems, Shenzhen, China, 4–7 November 2018; pp. 410–411.