

Latent Dimensions of Auto-Encoder as Robust Features for Inter-Conditional Bearing Fault Diagnosis

Chandrakanth R. Kancharla ^{1,*}, Jens Vankeirsbilck ¹, Dries Vanoost ², Jeroen Boydens ¹ and Hans Hallez ¹

¹ M-Group, DistriNet, Department of Computer Science, KU Leuven Bruges Campus, 8200 Bruges, Belgium; jens.vankeirsbilck@kuleuven.be (J.V.); jeroen.boydens@kuleuven.be (J.B.); hans.hallez@kuleuven.be (H.H.)

² M-Group, WaveCoRE, Department of Electrical Engineering, KU Leuven Bruges Campus, 8200 Bruges, Belgium; dries.vanoost@kuleuven.be

* Correspondence: chandu.kancharla@kuleuven.be

Abstract: Condition-based maintenance (CBM) is becoming a necessity in modern manufacturing units. Particular focus is given to predicting bearing conditions as they are known to be the major reason for machine down time. With the open-source availability of different datasets from various sources and certain data-driven models, the research community has achieved good results for diagnosing faults in bearing fault datasets. However, existing data-driven fault diagnosis methods do not focus on the changing conditions of a machine or assume all conditional data are available all the time. In reality, conditions vary over time. This variability can be based on the measurement noise and operating conditions of the monitored machines such as radial load, axial load, rotation speed, etc. Moreover, the availability of the data measured in varying operating conditions is scarce, as it is not always feasible to collect in-process data in every possible condition or setting. Considering such a scenario, it is necessary to develop methodologies that are robust to conditional variability, i.e., methodologies to transfer the learning from one condition to another without prior knowledge of the variability. This paper proposes the usage of latent values of an auto-encoder as robust features for inter-conditional fault classification. The proposed robust classification method MLCAE-KNN is implemented in three steps. First, the time series data are transformed using Fast Fourier Transform. Using the transformed data of any one condition, a Multi-Layer Convolutional Auto-Encoder (MLCAE) is trained. Next, a K-Nearest Neighbors (KNN) classifier is trained based on the latent features of MLCAE. The so-trained MLCAE-KNN is then used to predict the fault class of any new observation from a new condition. The results of using the latent features of the Auto-Encoder show superior inter-conditional classification robustness and superior accuracies compared to the state-of-the-art.



Citation: Kancharla, C.R.; Vankeirsbilck, J.; Vanoost, D.; Boydens, J.; Hallez, H. Latent Dimensions of Auto-Encoder as Robust Features for Inter-Conditional Bearing Fault Diagnosis. *Appl. Sci.* **2022**, *12*, 965. <https://doi.org/10.3390/app12030965>

Academic Editors: Sofie Van Hoecke and Femke Ongeane

Received: 24 November 2021

Accepted: 13 January 2022

Published: 18 January 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

Keywords: bearing faults; industry 4.0; robust fault diagnosis; transfer learning; k-nearest neighbors; convolutional auto-encoders

1. Introduction

Production lines often encounter downtime, either planned or unplanned. Though planned downtime affects the Overall Equipment Efficiency (OEE), it can be reduced by an optimized usage plan. In contrast, unplanned downtime, as the name suggests, is not planned beforehand and is typically due to the component failure, tool breakage or other technical stoppage. It is difficult to foresee some of these failures and plan the maintenance tasks accordingly. Predictive or condition-based maintenance is a solution that can be employed in manufacturing industries to reduce this unplanned downtime. Predictive maintenance methods, by predicting the condition of a machine or a component, allow dynamic and convenient maintenance scheduling, thus making it an adequate option for modern manufacturing industries [1]. Predictive maintenance, from the authors' perspective, is accomplished in three important steps: (1) monitor a machine, (2) predict

if something is anomalous and the machine is prone to failure and (3) make sure it is maintained without causing any disturbance to the planned production. In situ deployed sensors take care of the monitoring aspect, whereas fault prediction and maintenance are traditionally manual procedures. This intermediate task of predicting the early faults whilst monitoring the machines has been researched for a long time [2]. Importantly, there are two types of fault diagnosis methods to monitor machines or machine components, model-based and data-driven. Though very commonly used, the mathematical model of a fault diagnosis system designed using a model-based method cannot take into account all the noises or subtle variations that a system is exposed to [3]. Hence, depicting an accurate machine model or open-source availability of such models is unfeasible, whereas using an inaccurate model may lead to poor fault diagnostic performance in practice. Unlike the model-based approaches, data-driven methods utilize the collected in-process data. These data are analyzed, and the acquired knowledge is used to build a fault diagnosis model. This allows a user to apply diagnostics even when there are no available mathematical models of a particular machine. It also includes noise and a certain degree of variations into the diagnostic model. This noise inclusion into the diagnostic model is simple, with data-driven modeling compared to the model-based method [3], thus making it a more desirable solution for fault diagnosis.

Recently, data-driven methods have been performing exceptionally well in predictive fault diagnosis [4–7]. One of the main reasons for the increased performance of data-driven models over model-based methods is the availability of open source datasets. Looking into the state of the art for predictive maintenance in the manufacturing industry, most data-driven approaches focus on finding bearing faults, as these are the most predominant source of machine downtime [8]. There is a lot of research that shows very good classification results for bearing fault detection on the standard datasets that exist for vibration-based rolling element faults. Toma et al. show how a 1D Convolutional Neural Network (CNN) can be used on time domain data to accurately predict a bearing fault [9]. Moreover, by reshaping the time domain vibration data into a 2D array, i.e., into an image-style input, the two dimensional convolutional operations were used for feature extraction [10,11]. This study further presented the improved accuracy of the 2D CNNs for both normal and noisy data using the time domain data as input. Unlike the above, other works have shown the effectiveness of time-frequency transformed features for bearing fault detection [12]. In their study, only a few selective features of Empirical Mode Decomposition were used along with certain Machine Learning algorithms for effective classification of bearing faults. In [13], the authors derived an entrogram from the time series data using Frequency Slice Wavelet Transform, which they analyzed to see if the bearing faults can be distinguished from the normal working condition. A state of the art (SOTA) survey on various methodologies used for bearing fault diagnosis is detailed in [14]. As can be observed from this article, there are numerous ways of detecting bearing faults. The results show that bearing fault classification is effective when sufficient data from different conditions is available.

However, variability in the machine operating conditions poses a practical problem for condition-based monitoring. Furthermore, it is impractical to collect the data for all these variabilities (e.g., settings under which a machine operates, such as axial load, radial load, rotation speed, etc.). As we know, general features from the observations X_1 and X_2 of two different machine conditions do not always fall in a similar distribution. For example, data collected in a lab setting can be different to the data acquired from a similar setup in the field. Generic algorithms that are trained with the data from labs are then prone to transfer errors when used to infer the results from data measured in the field [15]. Similarly, the variation in the data can be caused by many parameters such as operating conditions, sensor displacement, variation in the sensor, etc., and it is currently a challenge to develop a model that is robust to these variations.

In an attempt to generalize the classification across conditions, researchers have proposed several algorithms that transfer the learning from one condition to another [15], where the class distribution from one condition is known (source domain) and some

information of another condition (target domain) to which the class information will be fitted and matched. To tackle the problem of needing data from all possible conditions, unlike the existing methods, we propose a new approach based on latent features of Auto-Encoders and nearest neighbor algorithms. This novel methodology considers any one condition's data and performs a robust classification across other conditions. For different open-source datasets, we then compared our method to the best performing source only methods and domain adaptation methods from the SOTA. This work is structured as follows. Section 2 gives the detailed problem description, which is followed up by Section 3, where we discuss some preliminaries necessary for understanding this work. The proposed methodology is then described in Section 4, succeeded by Experimental Setup, Results, and Conclusions in Sections 5–7, respectively.

2. Problem Description and Related Work

Before describing the problem, some notations and definitions regarding the problem are introduced.

Transfer Learning: The process of learning the information from the data of one condition and transferring the knowledge to a new condition.

Source Domain: The condition of the machine from which the data are collected and is used to train the fault diagnosis model.

Target Domain: The condition of the machine from which the data are collected and is used to test the knowledge transfer.

Transfer of class knowledge in Transfer Learning (TL) tasks can be classified into two methods. This is based on what is available during the training process:

Robust Learning: Refers to how to appropriately predict class label Y_j^t of the j th observation X_j^t of the target domain when there is an available set of observations from the source domain $D_s = (X_i^s, Y_i^s)_{i=0}^{n_s-1}$

Domain adaptation: In some cases, a few observations from the target domain $D_t = (X_i^t, Y_i^t)_{i=0}^{n_t-1}$ are also available, and the model is adapted to the target domain. This is called domain adaptation.

Here, $X_i^s = (x_0^i, x_1^i, \dots, x_m^i)$ is the sample set of the i th observation from the source domain, and Y_i^s is the corresponding label of that observation. In addition, X_i^t and Y_i^t stand for the sample set and corresponding label in the target domain, respectively.

While applying this to an industrial machine, the domains represent different conditions. There are various TL tasks from the literature that learn the distribution of the D_s and tries to adapt to the distribution of D_t . As represented in Figure 1, some methods train using information from the target domain, which can be considered as Domain Adaptation methods, and some do not; these can be considered as Robust Learners. The problem considered in this work is robust learning as, most of the time, the target domain data are scarce and infeasible to collect.

From the investigated literature, there are many transfer learning methodologies between machine conditions which have achieved high accuracies. The majority of these methods are domain-adaptation-based transfers of learning. The following are the state-of-the-art articles in the domain adaptation for bearing fault diagnosis. Wang et al. [8] proposed one such domain adaptation method named MDIAN. In [8], the authors use a modified ResNet-50 structure to extract multiple scale features and to try to reduce the conditional maximum mean discrepancy using the so extracted features of source and target domain observations. As reported in their work, MDIAN outperforms several SOTA approaches, such as Convolutional Neural Network (CNN) and CNN + Maximum Mean Discrepancy, for various transfer learning tasks of fault diagnosis. In another domain adaptation work, Li et al. [16] proposed a methodology where features from vibration data are extracted using a Convolutional Neural Network. Along with the general prediction loss of the source domain data, this CNN parallelly trains to minimize the central moment discrepancy between the source and target distributions. They have achieved great transfer learning accuracies across conditions for various datasets. To the knowledge of the authors,

it has been the best performing algorithm yet for the task of transfer learning given some target domain data. Furthermore, Li et al. [16] also described their implementation of several algorithms such as CNN-NAP (Nuisance Attribute Projection) [17] and MCNN (Multitask Convolutional Neural Network) [18] from the literature for transfer learning of machine faults. The uniqueness of these implementations (CNN-NAP, MCNN) is that they use a source-only dataset to train the classifier, similar to the consideration in this work. Though they are the only other robust learning methodologies that are comparable with this work, they are not much better than a simple Convolutional Neural Network when considered for inter-conditional classification [18]. Thus, these above-mentioned methodologies from the literature, indiscriminate of domain adaptation or robust learner, will be further used to compare our proposal.

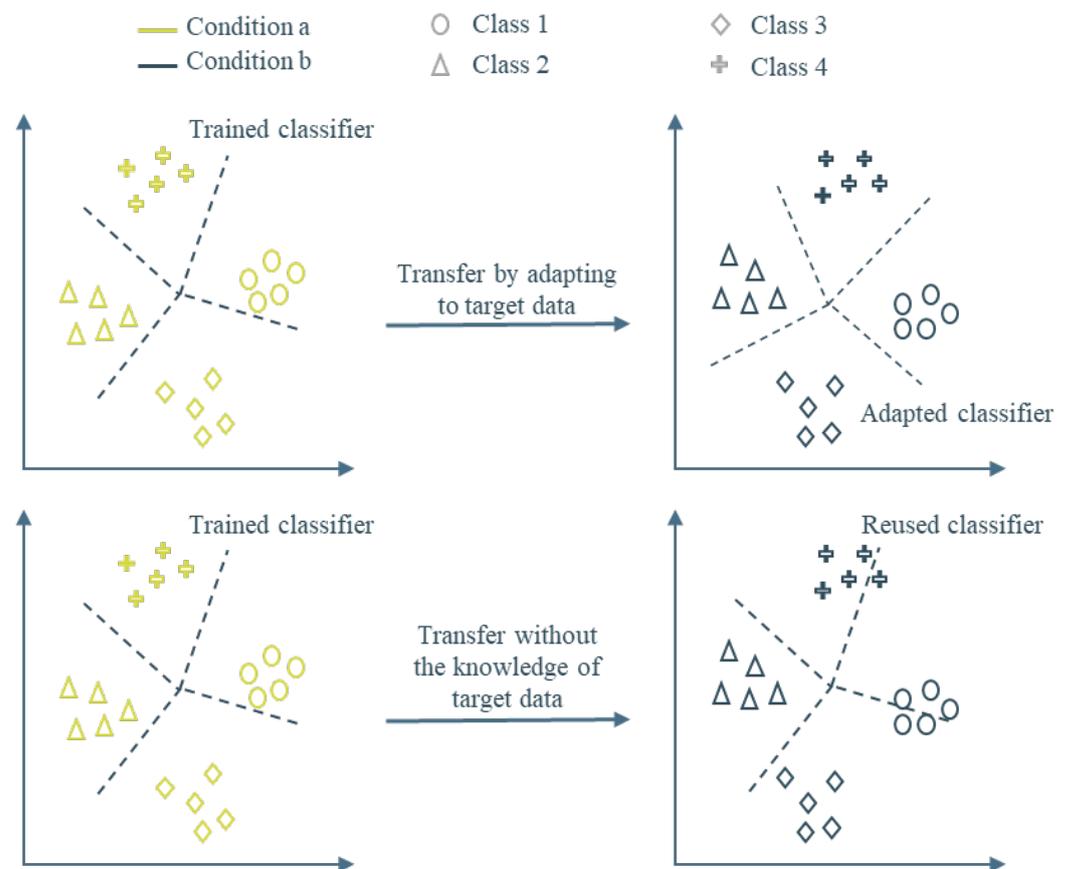


Figure 1. Different considerations for transfer learning. The top figure represents a domain adaptation approach and the bottom figure represents a robust learning approach across different operational conditions. The latter is the focus of this paper.

Contributions of this work are:

- The empirical evaluation of Auto-Encoder latent features as robust features across conditions was performed. It was systematically performed using various transfer tasks of two widely used open source bearing fault datasets.
- A simple methodology combining latent values of the Auto-Encoder and their proximity in the latent space is proposed for robust inter-conditional bearing fault diagnosis, given only the source domain data for training.
- The results of the proposed method are presented and compared with the results of the other state-of-the-art methods.

Furthermore, in this work, the transfer learning experiments are considered to be homogeneous and use source-only datasets for training (robust learner). Homogeneous here refers to the classes being consistent across both source and target domain, whereas a source-only dataset means the trained algorithm does not consider any information of the target domain data. These constraints on the experiments are based on the practical requirement for industries, where gathering data is a difficult task. Thus, to tackle the challenge efficiently, a robust learner is proposed in this work.

3. Materials and Methods

First, we describe the used datasets for the experiments. Next, we describe the Transfer Learning tasks from those datasets.

3.1. Data-Sets

Two open-source bearing fault diagnosis datasets were considered for experiments in this work. One is from Case Western Reserve University (CWRU) and the other is from Paderborn University (PU). Though the datasets are from 2015 and 2016, respectively, a Scopus search based on terms (('Case western' OR 'Paderborn') AND 'bearing fault') produced 205 research articles between 2020 and 2022. This sheer number of high quality research articles in the recent past signify the relevance and contribution of these two open source datasets in evaluating state-of-the-art bearing fault diagnostic models.

Both the datasets provide vibration data for various faults of a bearing unit under different machine operating conditions,; please refer to these articles for more details regarding the datasets [19,20].

3.1.1. CWRU Dataset

This is an open source dataset provided by Case Western Reserve University bearing data center. The motor in the setup is equipped with normal and faulty bearings Figure 2. Different single point faults ranging from 0.007 to 0.14 to 0.021 inches are induced onto the bearing's rolling element, inner race and outer race. Along with the normal bearings, the total number of grouped classes is thus 4 (normal, inner race, outer race and bearing fault). These 4 classes of data were collected from 4 different loads of the motor, 0, 1, 2 and 3 hp. These multi-dimensional qualities of this dataset allow it to be used for different application evaluations, particularly for inter-conditional bearing fault diagnosis.

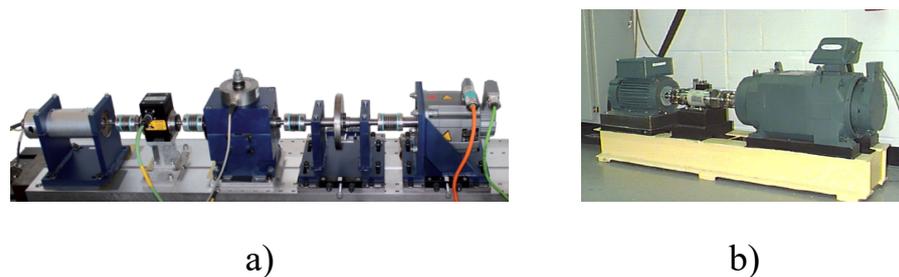


Figure 2. Setup of the open-source data-sets considered for validating the proposed method: (a) Case Western Reserve University setup [19]; (b) Paderborn University setup [20].

3.1.2. PU Data-Set

The Paderborn University dataset is provided by the bearing data center of Paderborn University. It comprises of vibration and electrical data collected for various artificial and accelerated bearing faults. A total of 32 bearings were used with their motor setup, among those 6 are normal condition bearings, 12 artificial faults and 14 natural faults (Figure 2). Please note that, in this work, only the artificial faults of the PU dataset were considered for analysis, similar to the work of [16]. The conditions in which these fault class data were collected vary in rotation speed, load torque and radial force of the spindle. This

dataset is thus very relevant and more complicated than the CWRU for inter-conditional bearing fault diagnosis evaluation.

3.2. Transfer Tasks

The datasets mentioned above have different working conditions, as mentioned in Table 1. The CWRU dataset has 4 conditions that are due to the variation in the load and rotational speed of the shaft as a result of this changing load. The transfer of learning between each of these conditions to another condition can be considered as one task. With 4 different working conditions based on the load applied ($L = 0, 1, 2, 3$ HP) and its respective motor speeds ($MS = 1797, 1772, 1750, 1730$), 12 transfer learning tasks can be formulated for the CWRU dataset. Accordingly, we can also formulate 12 transfer tasks for the PU dataset as there are 4 different conditions in which this dataset exists based on varying load torques ($M = 0.7, 0.1$ Newton meter), rotational speeds ($N = 1500, 900$ RPM) and radial forces ($F = 1000, 400$ Newtons). The transfer tasks considered for the experiments are across the conditions mentioned in Table 1.

3.3. Observation Definition

The data are processed and supplied to the algorithm by means of a moving window. The length of this moving window is chosen such that it fits approximately two rotations' information (CWRU dataset) and one rotations' information (PU dataset) of the motor spindle in any given condition. Furthermore, this window size is rounded to a power of two to make it practically feasible for frequency domain transformation by Fast Fourier Transform algorithm. Based on the different settings of the datasets, the window size of each observation for the CWRU and PU datasets is chosen to be 1024 and 4096, respectively. Another point to note is that there is no overlap between the windowed observations as we can produce a large enough dataset as is. Some of the so-created observations both in raw and frequency domains are shown in Figure 3.

Table 1. Different datasets and the settings in which they exist.

CWRU Dataset			
Name	Motor Speed (RPM)	Load (HP)	Sampling Frequency
C0	1797	0	12 KHz
C1	1772	1	12 KHz
C2	1750	2	12 KHz
C3	1730	3	12 KHz
Paderborn Dataset			
Name	Condition (N_M_F)	Sampling Frequency	
P0	N09_M07_F10	64 KHz	
P1	N15_M01_F10	64 KHz	
P2	N15_M07_F04	64 KHz	
P3	N15_M07_F10	64 KHz	

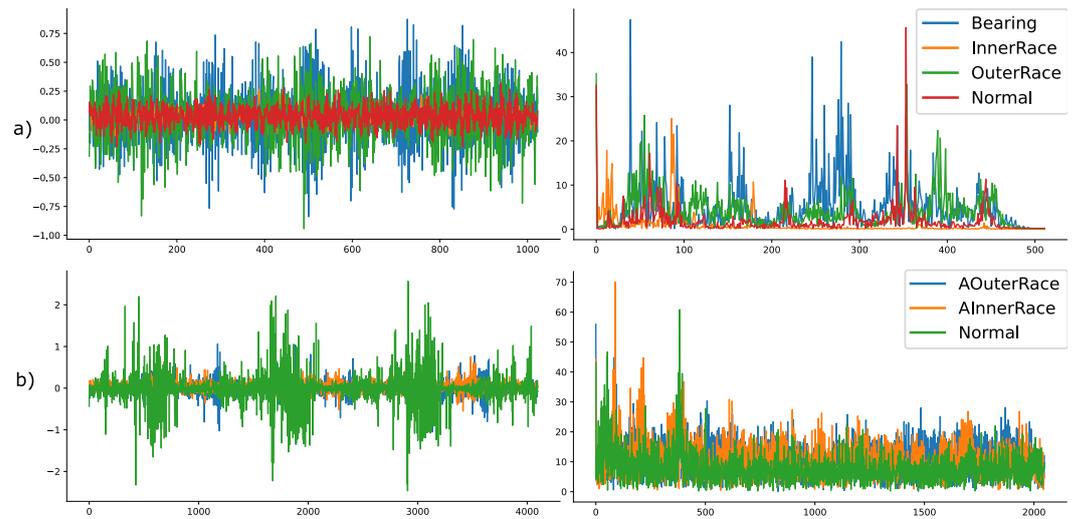


Figure 3. Random observations of various faults from raw data (X-axis: sample number, Y-axis: g) and frequency transformed data (X-axis: Harmonic order (HZ), Y-axis: g) of (a) CWRU and (b) Paderborn dataset.

4. Proposed Method

The proposed method contains three key stages, (i) extracting and analyzing the features using an encoder–decoder architecture; (ii) these latent values are used to train a classifier; and (iii) the trained encoder and classifier will then be used to infer observations from other conditions. The overview of training and testing is as shown in Figure 4.

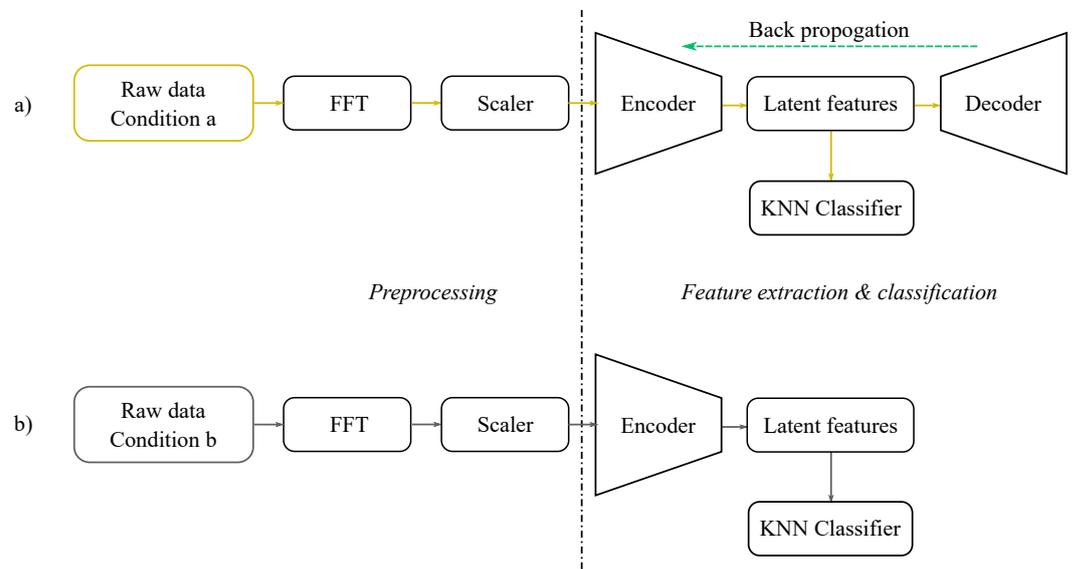


Figure 4. Structure of the proposed method train (a) and test (b). Note that every module that can be trained in the proposed methodology will be trained using only one condition’s data.

4.1. Auto-Encoder for Feature Extractor

From the studied literature, it is evident that the generic features extracted based on statistical methods do not provide much robustness across the conditions. Unlike the statistical methods from the literature, here we investigated the usage of latent features of an Auto-Encoder (AE) for designing a classifier that can be robust across conditions.

Auto-Encoder Architecture

Convolution layers have empowered many deep learning algorithms in computer vision tasks and have followed the trend with one-dimensional data alike. In our approach,

we use convolutional layers as the feature extractors followed by dense network layers for further learning. Contrary to the typical implementation of CNN for bearing fault feature extraction, we employ a Multi-Layer Convolutional AE network (MLCAE). This makes network training feasible without the class labels.

The architecture of MLCAE has multiple things to consider. As the original application of this work is to perform fault detection on resource-constrained devices, our main criteria will be to keep the network simple while not losing the prediction accuracies. By simple we mean as few parameters as possible to compute the necessary features for appropriate cross-domain fault detection. After parameter tuning, a 7-layered convolutional AE structure was found to be producing good results in terms of fault detection accuracy. Thus, a similar convolutional AE architecture was used for all the experiments going forward. The structure of the AE is detailed below.

The Encoder (E_i) and Decoder (DE_i) networks of an AE are generally understood as,

$$E_i : X_i \mapsto F_i$$

$$DE_i : F_i \mapsto X'_i$$

where i is the unique condition of the considered dataset, X is the input data, F is the compressed representation or latent features of X_i and X'_i is the reconstructed output from latent features. The optimization of networks E_i and DE_i is performed through backpropagation while minimizing the objective function 'Mean Squared Error loss':

$$E_i, DE_i = \operatorname{argmin} \left(\frac{1}{N} \sum_{i=1}^N (X_i - X'_i)^2 \right) \quad (1)$$

where N represents the batch size.

The first part of the AE, an encoder E_i , compresses input data X_i to F_i using several convolution, pooling, flatten and dense layers. Output of each layer differs based on the type of the layer. In this paper, the first layer is a 1D convolution layer followed by a pooling layer. Here, the convolution layer extracts features by performing convolution operations, and a pooling operation reduces the number of parameters from convolution layer and avoids over-fitting. Outputs of a l th combination of convolution and pooling layer can be defined as

$$X_i^l = \operatorname{pool}(\sigma(X_i^{l-1} * k^l + b^l))$$

Here, pool represents an average pooling layer, σ represents an activation function relu , $*$ represents a convolution operation, k is the kernel and b is the biases of the layer l . In the Encoder, there are two such combinational layers which are followed by a flatten layer and two dense layers. The last of the dense layers in the encoder network provides the latent representation F_i of the input X_i .

The encoder E_i from input X_i to F_i thus looks like:

$$X_i^1 = \operatorname{pool}(\sigma(X_i * k^1 + b^1))$$

$$X_i^2 = \operatorname{pool}(\sigma(X_i^1 * k^2 + b^2))$$

$$X_i^3 = \operatorname{flatten}(X_i^2)$$

$$X_i^4 = (\sigma(W_1 X_i^3 + b_1))$$

$$F_i = (\sigma(W_2 X_i^4 + b_2)) \quad (2)$$

W_1, W_2, b_1, b_2 in the above equations represent the weights and biases of the two dense layers, respectively.

The second part of the AE, the decoder DE_i is formed by a network which is similar but mirrored in representation to that of encoder E_i . Two major differences are the transposed alternatives to the pooling layer and convolution layer. Contrary to the pooling layer, using the up-sampling layer, we repeat the features to increase the output sizes. Similarly,

contrary to the convolution layer, using the transposed 1D convolution layer we achieve a feature projection from lower dimensions to higher dimensions. The decoder network DE_i from latent features F_i to reconstructed output X'_i looks like:

$$\begin{aligned} X_i^{3'} &= (\sigma(W_3 F_i + b_3)) \\ X_i^{2'} &= \sigma(\text{Upsample}(X_i^{3'}) * k^3 + b^3) \\ X_i^{1'} &= \sigma(\text{Upsample}(X_i^{2'}) * k^4 + b^4) \\ X_i' &= \sigma(X_i^{1'} * k^5 + b^5) \end{aligned}$$

where k and b in the above equations represent *Kernels* and *biases*, respectively.

The exact composition of the AE layers and their parameters are detailed in Section 5.4.

4.2. Data Preparation for Auto-Encoders

The CWRU and PU datasets discussed above are originally in the time domain, which can be used as inputs to the AE, whereas transformed data can also provide a different perspective of the data. In this investigation, along with the raw data, we have used Fourier-transformed frequency domain data. Furthermore, only the absolute value (magnitude) of the positive spectrum was considered from the FFT-transformed data. Figure 3 shows data of few observations of the CWRU and PU datasets in both time and frequency domains. In addition, these time and frequency data are normalized within the range of [0,1] using a Min–Max scaler according to Equation (3). The normalization step reduces the bias of features due to very high or very low scales. These transformed and normalized data will further become the inputs for our next step where features are extracted.

$$X_{scaled} = \left(\frac{X - \min(X)}{\max(X) - \min(X)} \right) \quad (3)$$

4.3. Feature Extraction and Analysis

To understand the essence of these latent features in segregating similar classes of different conditions, we considered random transfer tasks C0–C3 and P0–P3 from the CWRU and PU datasets, respectively, where C0 data will be used for training the MLCAE, and C0 and C3 data will be used for representing the data segregation results. The same is implemented using the PU data. Four different MLCAEs were trained using raw and frequency-transformed data of the conditions mentioned above. Latent features of those 4 MLCAEs using Equation (2) were then extracted to analyze and plot the data clusters. For ease of representation of the data, we further reduced the latent features to 2 dimensions using t-distributed Stochastic Neighbor Embedding (tSNE) [21].

As can be seen from Figure 5, the MLCAE features of the frequency transformed data are better at clustering appropriate classes than the features of raw data, at least based on the two components of the tSNE algorithm. We can also infer that the clustering results of the PU dataset are suboptimal, suggesting the complex nature of the PU dataset. To further classify observations, we train a K-Nearest Neighbors classifier.

4.4. K-Nearest Neighbors Classifier

The main assumption of this work is that the latent features inferred using an MLCAE trained with one condition share an approximately similar distribution to other conditions' data. Thus, proximity-based classifiers are preferred to boundary- or threshold-based classifiers. To test our hypothesis of latent features across conditions sharing similar distribution, we trained K-Nearest Neighbors (KNN). KNN is a well-known algorithm that predicts class label based on the proximity of the input features. It measures the distance to the K number of known observations in a multi-dimensional feature space and votes for its class accordingly. Using above inferred latent features from the MLCAE as

inputs for a KNN classifier, we tested its performance for various parameters of K and distance function.

To benchmark the performance of KNN with different hyper-parameters, we investigated the effect of varying K and distance functions on data from the two bearing fault datasets. After splitting the data into 80% and 20% (training and validation, respectively) of one randomly chosen condition from each dataset (C0 and P0), we first trained an MLCAE and then used its latent features to train and validate various KNNs for classification accuracy.

For the K value, the range investigated was between 1 and 30 in steps of 3. For the distance metric, which plays a major role in the performance of the KNN, we chose Chi-square, Euclidean, Manhattan and Chebyshev distances. Chi-squared and Euclidean were the best performing distance functions for numerical datasets such as ours according to [22], while Euclidean, Manhattan and Chebyshev distances were among the top 10 performing distance metrics for 28 datasets in a previous investigation [23]. It is to be noted that the Hassanat distance performed well when the data were not normalized. If the inputs are normalized, this distance metric may not perform at a similar level to the other known metrics. Thus, in our benchmark study to find optimal hyper-parameters for KNN, we omitted Hassanat.

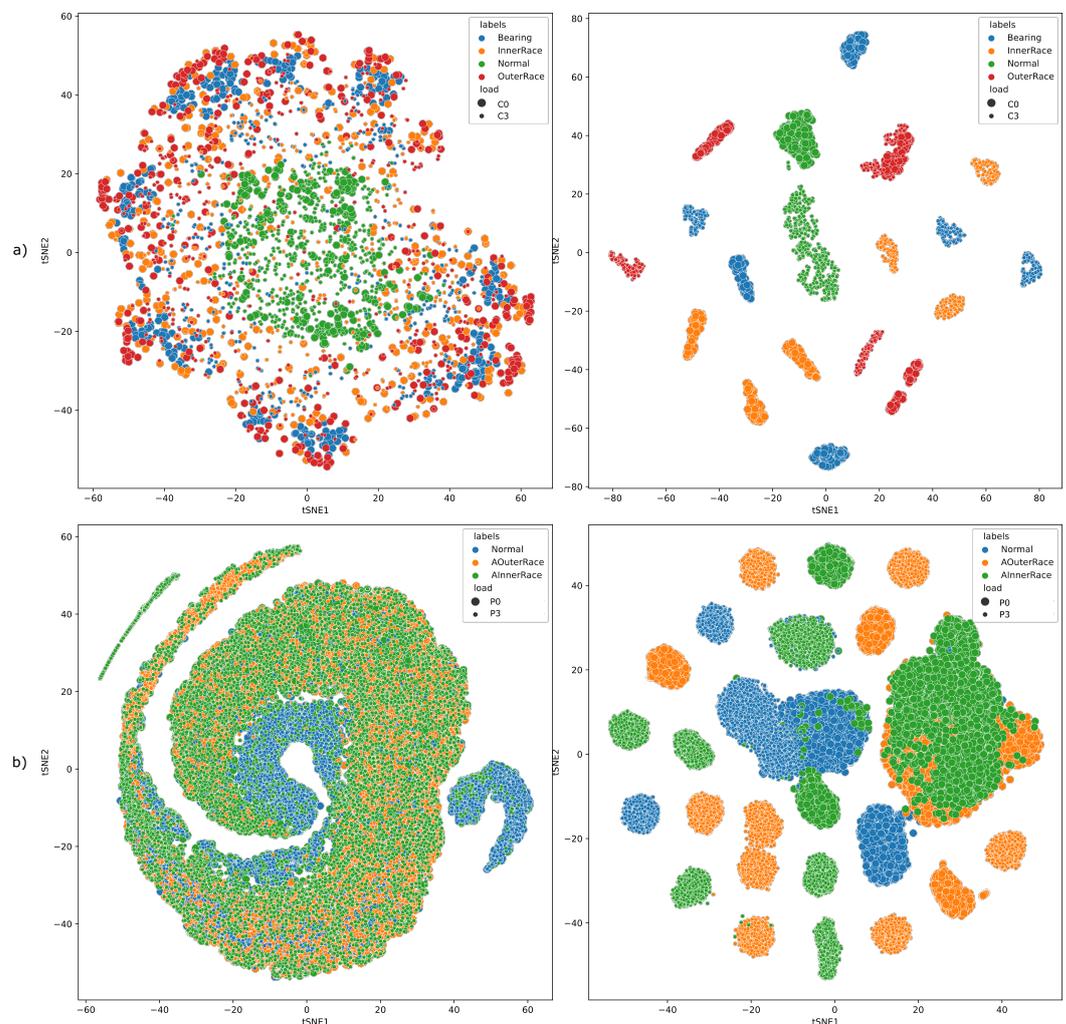


Figure 5. tSNE-based clustering results of features from MLCAE for Raw and FFT data of both the datasets. CWRU, raw and FFT features (a) and Paderborn, Raw and FFT features (b). Prefix A in Inner Race and Outer Race fault labels of PU data refer to artificial faults.

As can be seen from Figure 6, the results of one of the iterations show that Chi-square distance performs the worst for both bearing fault datasets, whereas Euclidean distance

works well in both experiment cases (C0 and P0) and shows better performance over the other considered distance functions. Euclidean distance is measured using the equation mentioned in (4). In addition, multiple values of K have produced good results on the validation data, making it a free choice within the range of 1 to 15. An absolute value of K chosen for the robust learning experiments will be discussed in later sections.

$$d_{euclidean}(q, p) = \sqrt{\sum_{i=1}^m (q_i - p_i)^2} \quad (4)$$

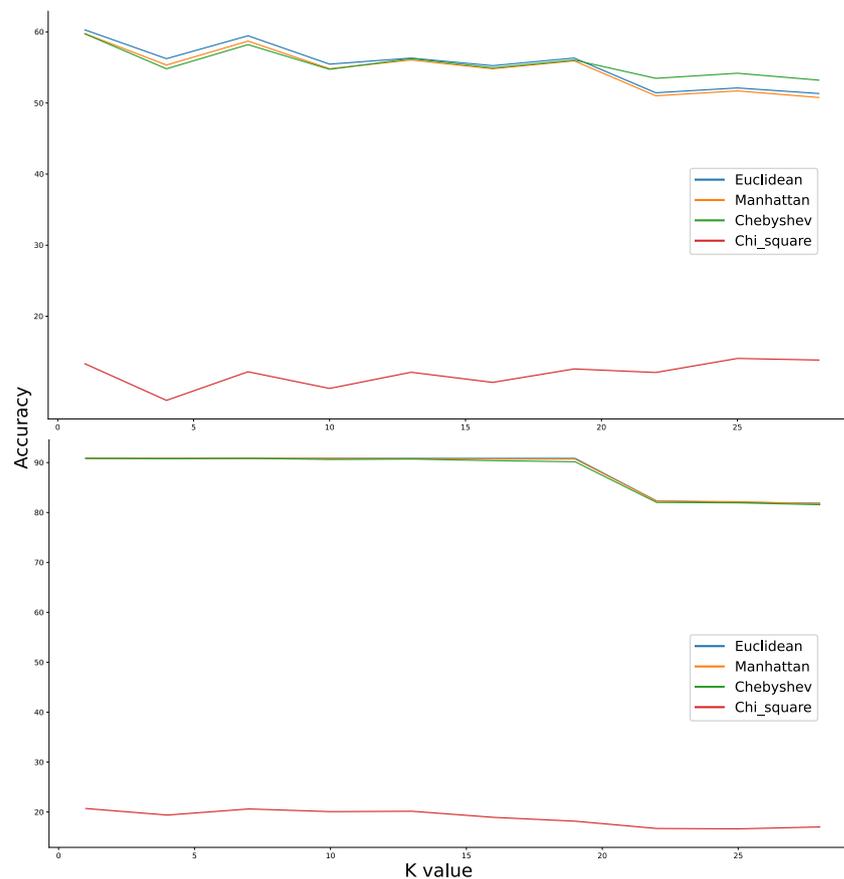


Figure 6. Comparison of different K values and distance function of K Nearest Neighbors algorithm for condition 0. Euclidean distance with K values around 1–15 seem to be producing good classification results for both the datasets. The top and bottom comparisons are for CWRU and PU datasets, respectively.

5. Experimental Validation

To prove the effectiveness of the proposed methodology for robust transfer learning across conditions, we developed an experimental setup. There are important aspects of this setup, considered datasets, transfer learning tasks and chosen methods from the state-of-the-art to compare the results against.

5.1. Computational Unit and Training Time

The experiments have been performed on a DELL precision 5530. The configuration of the computational unit used for further experiments is as follows: 32GB RAM, 6 2.6GHz processors and an NVIDIA Quadro P2000 Graphical Programming Unit. The experimental scripts are written in Python 3.7.8.

The loss function used for training the MLCAE is the Mean Squared Error loss, and the stochastic optimizer used was Adam. The training times differ from task to task. For each transfer task, it takes approximately about 30 s to train with CWRU data and 400 s with PU data. Inference times are quite fast and were thus not measured.

5.2. Compared Methods

As discussed in the related work section, state-of-the-art methods mentioned in Table 2 were chosen to compare the results. Two of the compared methodologies, Support Vector Machine (SVM) and Convolutional Neural Network (CNN), are source-only methods similar to our proposal, whereas CNN-MMD (CNN- Maximum Mean Discrepancy), MDDAN, MDIAN and CMD (Central Mean Discrepancy) use some information from the target domain for the transfer tasks. Nevertheless, we made the comparisons to show the effectiveness of the proposed method against state-of-the-art domain adaptation methods.

Table 2. Robust learning and domain adaptation methods from the literature are used to compare with the proposed method.

Method	Reference	Source Only
SVM	[8]	Yes
CNN	[8]	Yes
CNN-MMD	[8]	No
MDDAN	[8]	No
MDIAN	[8]	No
CMD	[16]	No
MLCAE-KNN	Proposed	Yes

5.3. Evaluation Metric, Training and Testing Process

The evaluation of the proposed method is performed using Accuracy as a metric. Accuracy here is sufficient, as the considered cases are balanced across different labels.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{Total number of samples}}$$

True Positives (TP) and True Negatives (TN) together form the total number of samples that are predicted properly.

5.4. Architecture of the Auto-Encoder

Upon investigating the classification results of our methodology on the validation data (a part of the training dataset), we understood that the frequency features as inputs to the MLCAE are better at segregating appropriate classes than the raw data. Thus, the inputs to the MLCAE in further experiments will be the magnitude of the positive spectrum of the FFT data. In the case of the CWRU dataset, each observation consisted of 1024 sample points so that it approximately fits information of two rotations of the motor spindle. This means that considering only the positive spectrum of the FFT data will lead to an input size of (512,1) for the CWRU dataset. Similarly, to approximately fit the information of two spindle rotations, the samples in each observation for the Paderborn data were chosen as 4096. Thus, the input size of the architecture used for Paderborn will be of size (2048,1). Some of the parameters used in architectures for CWRU and Paderborn are similar, whereas some are not. Details of these parameters are also mentioned in Table 3.

Such AE-based extractors are trained with the FFT data of each condition. During the process of training, it considers the loss of reconstruction and tries to minimize this loss while optimizing the whole network. Once the stop criteria (non-reducing loss or suggested epochs) are met, the training algorithm stops, and the encoder part of the so-trained AE is considered for further usage. When an observation from a different condition is inferred using this encoder, it produces a reduced number of features (values from the latent dimension), which will then be used as inputs to train the classifier. Numerically, the MLCAE (Multi Layer Convolutional AE) will train and bring sizes of (512,1) of CWRU and (2048,1) of Paderborn to (20,1) latent dimensions, which will then be used by the KNN classifier for further classification.

Table 3. Used Multi-Layer Convolutional Auto-Encoder architecture for the experiments.

Layers	Parameters
Conv1	Kernel size: (5,1), number of Kernels: 30, Stride: 1, activation: relu, Padding: Same
Pool1	Average Pooling, Size: 4, Stride: 1
Conv2	Kernel size: (3,1), number of Kernels: 15, Stride: 1, activation: relu, Padding: Same
Pool2	Average Pooling, Size: 2, Stride: 1
Flat1	converts 2d output from previous layer to 1d
Dense1	Dense, Size: (50,1), activation: relu
Latent	Dense, Size: (20,1), activation: relu
Dense2	Dense, Size: (CWRU: (100,1), PU: (256,1)), activation: relu
UpSamp1	Upsampling1d, Size: 2
ConvT1	Conv1dTranspose, Kernel size: (3,1), number of Kernels: 15, Stride: 1, activation: relu, Padding: Same
UpSamp2	Upsampling1d, Size: 4
ConvT2	Conv1dTranspose, Kernel size: (5,1), number of Kernels: 30 Stride: 1, activation: relu, Padding: Same
ConvT3	Conv1dTranspose, Kernel size: (3,1), number of Kernels: 1 Stride: 1, activation: sigmoid, Padding: Same

We also discussed in the previous section that the choice of K value for the KNN may fall anywhere between 1 and 15. Upon further investigation for random transfer tasks, the classification results across conditions were inconsistent when the K values are small and was showing degraded performance across conditions when the value is high. The exact nearest number of observations to consider for predicting a new sample in case of both the datasets was thus found to be optimal at $k = 5$.

The trained classifier from one condition was then tested against other conditions, and the experimental results for various transfer tasks were discussed accordingly. The training and testing process adopted in this article is as described in Algorithm 1.

Algorithm 1 Pseudo algorithm used for training and testing various transfer tasks of CWRU and PU datasets.

- 1: **Training Input:** Labeled source domain data $D_s = (X_i^s, Y_i^s)_{i=0}^{n_s-1}$
- 2: **Training Output:** Min-Max Scaler (MMS_i), Auto-Encoder Network (encoder E_i and decoder DE_i) and K-Nearest Neighbours classifier (KNN_i)
- 3: **Testing Input:** Labeled target domain data $D_t = (X_i^t, Y_i^t)_{i=0}^{n_t-1}$ and MMS_i , E_i and KNN_i
- 4: **Testing Output:** Predicted labels of target domain data Y_i^t
- 5: **Begin:**
- 6: **for** each value of i in range of n :
- 7: **Train and transform** X_i with MMS_i as shown by Equation (3)
- 8: **for** (m epochs or stopping criteria met):
- 9: **Optimize** (E_i, DE_i) with **MSE** as shown in (1)
- 10: **end for**
- 11: Compute **latent features** F_i of X_i using Equation (2)
- 12: **Train** KNN_i with F_i and Y_i
- 13: **for** each value of k in range of n and $k \neq i$:
- 14: **Transform** X_k with MMS_i
- 15: Compute **latent features** F_k of X_k with E_i
- 16: **Predict** Y'_k with F_k and KNN_i
- 17: Compute **accuracy** of prediction with (Y_k) and (Y'_k)
- 18: **end for**
- 19: **end for**

As an important note, the normalisation part of our pre-processing step has two separate methods for the train and test datasets. Considering the goal of using a source-only dataset for algorithm training, the test dataset (target domain data) will not be used during the training process. The same scaler that is used to fit against and transform the training data will be used to transform the test data. In this way, we make sure that no information of the target domain is used during training.

6. Results

In this section, the results of the experiments are discussed. Since the two used datasets are significantly different, they are handled separately. First, the results on the CWRU dataset are discussed, and next the PU dataset results are detailed.

6.1. CWRU Data-Set

Considering the different conditions of the CWRU dataset as discussed in Section 3, we have implemented 12 different transfer learning tasks. The results of our methodology on the CWRU dataset are very good, as shown in Table 4. The best runs of our method have a 100% accurate class transfer across different conditions. The exceptions are cases such as C2–C0, C2–C1, etc., where the accuracy drops to 99.7%. Comparing the results with other source-only methods or the domain adaptation methods, the MLCAE-KNN outperforms other methods. One particular note about the presented table is that the method from Reference [16], CMD, did not consider one condition from the dataset. Thus, we have left the unconsidered tasks null in the comparisons.

Table 4. Comparison of accuracies of various state-of-the-art methodologies on CWRU dataset. Except for SVM, CNN and MLCAE-KNN, the other methods are for domain adaptation. Also mentioned along with MLCAE-KNN is the accuracy variance across multiple iterations.

Transfer Task	SVM	CNN	CNN-MMD	MDDAN	MDIAN	CMD	MLCAE-KNN
C0 → C1	70.70	72.25	81.00	87.15	99.60	-	100
C0 → C2	66.45	70.55	79.90	90.60	99.30	95.54	100
C0 → C3	63.40	62.45	55.85	91.65	99.10	99.54	100 (3%)
C1 → C0	71.30	87.30	88.95	84.00	99.70	-	100
C1 → C2	70.00	89.80	88.70	92.40	99.65	-	100
C1 → C3	74.00	74.70	80.50	94.20	99.80	-	100 (5%)
C2 → C0	62.85	60.35	64.65	87.40	97.60	100	99.8
C2 → C1	61.60	75.50	79.80	91.95	99.45	-	99.7
C2 → C3	67.65	84.30	79.95	91.50	99.45	96.9	100 (2%)
C3 → C0	65.30	66.90	75.25	84.25	97.45	100	99.9 (2%)
C3 → C1	65.70	81.15	71.15	87.35	98.60	-	99.9 (3%)
C3 → C2	63.25	74.95	74.85	92.15	99.50	100	100

As the AE structure trains without the necessity of class labels, the learned latent features can be quite different from one iteration of the complete experiment to another. As expected, few features make the distinction much better than the others. Thus, one iteration produces exceptional results and some may not. The criteria of selecting the best model for further practical usage is out of the scope of this work. The results presented in this article are from the best performing models from the experiments, along with the variability shown by our methodology across various experimental iterations. Though deviation of accuracy using our method is about 0–5%, it is low compared to the accuracy variability mentioned for some of the other methods according to [16].

6.2. Paderborn Dataset

As the CMD was performing great along with our method during the previous comparison, and as the CNN was the best of source-only method, we will use these two for further comparisons on the Paderborn dataset. This dataset is known to be complicated because of the multitude of variations considered in their setup. That said, our method

shows promising results for various tasks, even on this dataset. The MLCAE-KNN clearly outperforms the CNN for every transfer task. From the results, it is also evident that our classifier is robust for some tasks such as P1–P3, P2–P3 and P3–P1 where even the CMD, the state-of-the-art domain adaptation method, was underperforming compared to ours.

Further analyzing the results from this dataset, we notice that collecting data at higher scales and using our methodology for prediction provides better class transfer across conditions. For example, training on condition P0 and testing on conditions P1 and P3 have always provided better results than the other way around (From P1, P2 and P3 to P0). Here, P0's conditions are on a higher scale compared to the other three, as seen in Table 1, whereas, between other conditions such as P1, P2 and P3, class information transfer has been acceptable both ways. For this particular dataset, the CMD was better or as good as our method in the majority of the transfer tasks, as shown in the comparisons in Table 5. Overall, considering just the source domain data, the MLCAE-KNN has been robust in transferring the class information from one condition to another, proving its usability in an industrial context.

Table 5. Comparison of accuracies of CNN (Robust learner) and CMD (Domain adaptation) against our methodology on the Paderborn University dataset. MLCAE-KNN outperforms CNN in all the transfer tasks and performs better than CMD for a few tasks. Also mentioned along with MLCAE-KNN accuracies is the variance across multiple iterations of the experiments.

Transfer Task	CNN	CMD	MLCAE-KNN
P0 → P1	39.65	70.44	59.1 (6%)
P0 → P2	51.33	75.30	62.7 (6%)
P0 → P3	40.04	69.73	58.8 (6%)
P1 → P0	44.43	82.62	45.3 (5%)
P1 → P2	82.32	94.01	83.8 (3%)
P1 → P3	89.39	91.63	94.9 (1%)
P2 → P0	39.23	78.03	72.5 (5%)
P2 → P1	57.10	89.97	88.7 (6%)
P2 → P3	50.94	80.34	89.6 (6%)
P3 → P0	43.52	70.93	51.0 (4%)
P3 → P1	94.11	94.99	95.2 (1%)
P3 → P2	47.43	88.87	85.6 (5%)

7. Conclusions

In the context of condition-based monitoring, the collection of data from different conditions/different machine settings is a difficult task and is sometimes even impossible. Ideally, it is efficient to perform transfer of learning from one condition to another. We proposed a source-only methodology to effectively transfer class learning between conditions of a machine for bearing fault detection. We implemented and compared the performance with various state-of-the-art transfer learning methods, both source-only methods (SVM and CNN) and domain adaptation methods (CNN-MMD, MDDAN, MDIAN and CMD) for a set of tasks. From the results, the following can be concluded:

- (1) The proposed method performs more robust classification compared to other transfer learning methods. For many inter-conditional transfer tasks, the MLCAE-KNN source-only method performs as good or better than the other domain adaptation methods that consider certain information from the target domain.
- (2) Though the proposed methodology is robust, for some transfer tasks, it has a certain deviation in the accuracies across different runs (up to 6% for certain tasks of the Paderborn dataset, as presented in Table 5).
- (3) In our observation with the experiments of MLCAE-KNN, training a classifier using data from higher parameter settings (rotational speed, radial load, etc.) and trans-

ferring the learning onto lower settings provides better a transfer of class learning compared to the other way around.

As for how and what effects a transfer learning task has across conditions is still a preliminary hypothesis, further investigation is needed. To further validate our proposal and the preliminary hypothesis, as a future work we will investigate transfer of class learning tasks by considering other datasets for a similar application. In addition, as discussed in the proposed methodology, one of the criteria is to keep the AE structure simple. This is due to its intended use on resource-constrained embedded devices. Thus, our future work also extends to investigating the practicalities of inferring MLCAE-KNN on an embedded device.

Author Contributions: Conceptualization, C.R.K.; software, C.R.K.; formal analysis, C.R.K.; investigation, C.R.K.; data curation, C.R.K. writing—original draft preparation, C.R.K.; writing—review and editing, J.V., D.V., J.B. and H.H.; visualization, C.R.K.; supervision, J.V., D.V., J.B. and H.H.; funding acquisition, J.B. and H.H. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by a research grant from the Flemish Agency for Innovation and Entrepreneurship (VLAIO) within the HBC.2017.0650 RESSIAR-MID/TransSIMS project and the HBC.2019.2644 WEAR-AI project. It is also partially funded by project number 180493 AIO-PROEFTUIN-Industry4.0 and by project number 1301 Co-Creatie binnen het Competentiecentrum Machinebouw en Mechatronica W-VL.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Publicly available datasets were analyzed in this study. This data can be found here: <https://engineering.case.edu/bearingdatacenter/download-data-file> and <https://mb.uni-paderborn.de/en/kat/main-research/datacenter/bearing-datacenter/data-sets-and-download>.

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

References

1. Li, Z.; Wang, K.; He, Y. Industry 4.0—Potentials for Predictive Maintenance. In *6th International Workshop of Advanced Manufacturing and Automation*; Atlantis Press: Paris, France, 2016. [CrossRef]
2. Teixeira, H.N.; Lopes, I.; Braga, A.C. Condition-based maintenance implementation: A literature review. *Procedia Manuf.* **2020**, *51*, 228–235. [CrossRef]
3. Jung, D.; Sundstrom, C. A Combined Data-Driven and Model-Based Residual Selection Algorithm for Fault Detection and Isolation. *IEEE Trans. Control Syst. Technol.* **2019**, *27*, 616–630. [CrossRef]
4. Gonzalez-Jimenez, D.; Del-Olmo, J.; Poza, J.; Garramiola, F.; Madina, P. Data-driven fault diagnosis for electric drives: A review. *Sensors* **2021**, *21*, 4024. [CrossRef]
5. Aherwar, A. An investigation on gearbox fault detection using vibration analysis techniques: A review. *Aust. J. Mech. Eng.* **2012**, *10*, 169–183. [CrossRef]
6. Wei, Y.; Li, Y.; Xu, M.; Huang, W. A review of early fault diagnosis approaches and their applications in rotating machinery. *Entropy* **2019**, *21*, 409. [CrossRef]
7. 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]
8. Wang, X.; Shen, C.; Xia, M.; Wang, D.; Zhu, J.; Zhu, Z. Multi-scale deep intra-class transfer learning for bearing fault diagnosis. *Reliab. Eng. Syst. Saf.* **2020**, *202*, 107050. [CrossRef]
9. Toma, R.N.; Prosvirin, A.E.; Kim, J.M. Bearing fault diagnosis of induction motors using a genetic algorithm and machine learning classifiers. *Sensors* **2020**, *20*, 1884. [CrossRef] [PubMed]
10. Zhang, Y.; Xing, K.; Bai, R.; Sun, D.; Meng, Z. An enhanced convolutional neural network for bearing fault diagnosis based on time–frequency image. *Meas. J. Int. Meas. Confed.* **2020**, *157*, 107667. [CrossRef]
11. Zhang, J.; Sun, Y.; Guo, L.; Gao, H.; Hong, X.; Song, H. A new bearing fault diagnosis method based on modified convolutional neural networks. *Chin. J. Aeronaut.* **2020**, *33*, 439–447. [CrossRef]
12. Patel, S.P.; Upadhyay, S.H. Euclidean distance based feature ranking and subset selection for bearing fault diagnosis. *Expert Syst. Appl.* **2020**, *154*, 113400. [CrossRef]
13. Zhang, K.; Xu, Y.; Liao, Z.; Song, L.; Chen, P. A novel Fast Entrogram and its applications in rolling bearing fault diagnosis. *Mech. Syst. Signal Process.* **2021**, *154*, 107582. [CrossRef]

14. Hoang, D.T.; Kang, H.J. A survey on Deep Learning based bearing fault diagnosis. *Neurocomputing* **2019**, *335*, 327–335. [[CrossRef](#)]
15. Li, C.; Zhang, S.; Qin, Y.; Estupinan, E. A systematic review of deep transfer learning for machinery fault diagnosis. *Neurocomputing* **2020**, *407*, 121–135. [[CrossRef](#)]
16. Li, X.; Hu, Y.; Zheng, J.; Li, M.; Ma, W. Central moment discrepancy based domain adaptation for intelligent bearing fault diagnosis. *Neurocomputing* **2021**, *429*, 12–24. [[CrossRef](#)]
17. Ma, H.; Li, S.; An, Z. A fault diagnosis approach for rolling bearing based on convolutional neural network and nuisance attribute projection under various speed conditions. *Appl. Sci.* **2019**, *9*, 1603. [[CrossRef](#)]
18. Guo, S.; Zhang, B.; Yang, T.; Lyu, D.; Gao, W. Multitask Convolutional Neural Network with Information Fusion for Bearing Fault Diagnosis and Localization. *IEEE Trans. Ind. Electron.* **2020**, *67*, 8005–8015. [[CrossRef](#)]
19. 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](#)]
20. 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 Third European Conference of the Prognostics and Health Management Society 2016, Bilbao, Spain, 5–8 July 2016.
21. van der Maaten, L.; Hinton, G. Visualizing data using t-SNE. *J. Mach. Learn. Res.* **2008**, *9*, 2579–2605.
22. Hu, L.Y.; Huang, M.W.; Ke, S.W.; Tsai, C.F. The distance function effect on k-nearest neighbor classification for medical datasets. *SpringerPlus* **2016**, *5*, 1304. [[CrossRef](#)]
23. Abu Alfeilat, H.A.; Hassanat, A.B.; Lasassmeh, O.; Tarawneh, A.S.; Alhasanat, M.B.; Eyal Salman, H.S.; Prasath, V.S. Effects of Distance Measure Choice on K-Nearest Neighbor Classifier Performance: A Review. *Big Data* **2019**, *7*, 221–248. [[CrossRef](#)] [[PubMed](#)]