machines

MDPI

Article

Padé Approximant Neural Networks as Feature Extractors for Unsupervised Domain Adaptation in Bearing Fault Diagnosis

Sertac Kilickaya [1,*], Cansu Celebioglu [2], Murat Askar [1], Turker Ince [3] and Levent Eren [1]

[1] Department of Electrical and Electronics Engineering, Izmir University of Economics, 35330 Izmir, Turkey; murat.askar@ieu.edu.tr (M.A.); levent.eren@ieu.edu.tr (L.E.)
[2] Department of Information Engineering, University of Padova, 35131 Padova, Italy; cansu.celebioglu@studenti.unipd.it
[3] Faculty of Engineering, German International University, 13507 Berlin, Germany; turker.ince@giu-berlin.de
* Correspondence: sertac.kilickaya@ieu.edu.tr

Abstract

Variations in mechanical load constitute a dominant source of domain shift in data-driven fault diagnosis of rotating machinery, causing models trained under one operating condition to degrade sharply when deployed under another. This work addresses the problem through unsupervised domain adaptation (UDA)—which transfers diagnostic knowledge from a labeled source condition to an unlabeled target condition by aligning their feature distributions—and introduces Padé Approximant Neural Networks (PadéNets) as compact yet highly expressive feature extractors. One-dimensional PadéNet encoders are embedded into three established adaptation frameworks—Deep CORAL, Domain-Adversarial Neural Networks (DANNs), and Conditional Domain-Adversarial Networks (CDANs)—to learn load-invariant representations without any labeled target data. On the Case Western Reserve University benchmark, where the models operate directly on raw time-domain vibration signals, replacing conventional convolutional encoders with PadéNets consistently improves cross-load diagnostic accuracy, reaching up to 99.28% average target-domain accuracy at a low parameter count. To assess generalization to a more demanding setting, the CDAN–PadéNet configuration is further evaluated on frequency-domain representations of the Paderborn University dataset, where domain shift arises from simultaneous variation of load torque and radial force on bearings with real accelerated-lifetime damage, attaining 99.84% average accuracy across six cross-condition transfer tasks while requiring fewer parameters than competing methods. These results establish PadéNet-enhanced UDA as an accurate, broadly applicable approach for robust bearing fault diagnosis under varying operating conditions, with a reduced parameter count suited to resource-constrained embedded platforms.

Keywords: condition monitoring; fault diagnosis; unsupervised domain adaptation; Padé approximant neural networks; convolutional neural networks

Academic Editor: Karolina Kudelina

Received: 13 June 2026
Revised: 1 July 2026
Accepted: 3 July 2026
Published: 5 July 2026

1. Introduction

Rotating machinery is a core component of modern industrial systems, where unexpected failures can cause substantial economic losses, safety risks, and prolonged unplanned downtime. Among the components prone to degradation, rolling-element bearings are widely regarded as the most vulnerable, which has motivated extensive research into reliable condition monitoring and fault diagnosis [1]. Large-scale reliability surveys support this assessment. An Electric Power Research Institute (EPRI) study on rotating electrical

machinery identified mechanical and electrical degradation as the dominant failure mechanisms, with bearing-related faults alone accounting for approximately 41% of all reported failures—the highest proportion among all failure modes. This figure reflects the sensitivity of bearings to accumulated mechanical stress, lubricant breakdown, and electrical discharge damage, and underlines the need for diagnostic approaches capable of early and accurate fault detection [2].

Data-driven approaches based on machine learning and deep neural networks have shown strong diagnostic performance in recent years, often surpassing traditional signal-processing methods when sufficient labeled data are available [3,4]. Their practical deployment, however, remains hindered by variations in operating conditions [5]. Changes in mechanical load, load torque, radial force, rotational speed, and the associated acceleration transients alter the vibration signature through several coupled physical mechanisms. Load and radial force modify the contact mechanics, stiffness, and damping of the bearing–housing assembly; speed shifts the characteristic defect frequencies and their harmonics relative to structural resonances; and varying torque changes the energy of both the deterministic gear and shaft components and the fault-excited broadband response. Each mechanism introduces distributional differences between the training and deployment environments, so that models trained under one load condition often degrade considerably when applied to data acquired under another [6]. This phenomenon, known as domain shift, limits the generalization of standard deep learning models in industrial settings. The problem is not unique to bearing diagnosis. For example, one-dimensional CNN-based monitoring pipelines in laser powder bed fusion [7] and electrical discharge machining [8] face the same distribution shift whenever the acquisition context changes, suggesting that adapting 1D signal classifiers without labeled target data is a broader need across modern manufacturing. Among the various sources of shift, load variation is the most frequent and best benchmarked, and therefore forms the focus of our first case study, while the second case study extends the evaluation to simultaneous load-torque and radial-force variation on the Paderborn benchmark.

Unsupervised domain adaptation (UDA) has emerged as an effective strategy against domain shift, aligning feature distributions between a labeled source domain and an unlabeled target domain [9]. Several UDA techniques have been applied to fault diagnosis, including discrepancy-based methods such as correlation alignment [10] and adversarial frameworks that encourage domain-invariant feature extraction [11,12]. The performance of these methods, however, is closely tied to the representational power of the underlying feature extractor [13], and most existing studies rely on conventional convolutional neural networks (CNNs), which may struggle to capture the complex nonlinear patterns of vibration signals under varying operating conditions [14,15].

Padé Approximant Neural Networks (PadéNets) were recently introduced through the Padé neuron (PAON), a neuron model that employs Padé approximation to perform inherently nonlinear feature transformations beyond standard activation functions [16]. Each PAON learns a rational approximation of a nonlinear mapping, providing considerably richer expressiveness at a relatively small parameter cost [16]. In fault diagnosis, one-dimensional PadéNets operating on vibration or acoustic signals have been shown to outperform conventional CNN and Self-Organized Operational Neural Network (Self-ONN) architectures by extracting more discriminative fault-related features under fixed operating conditions [17]. Their use within UDA frameworks, particularly for handling load-induced domain shift in rotating machinery, has not yet been systematically explored.

Motivated by these observations, this paper investigates PadéNet-based feature extractors within well-established UDA frameworks for bearing fault diagnosis under varying operating conditions. PadéNet encoders are incorporated into the Deep CORAL [10],

Domain-Adversarial Neural Network (DANN) [11], and Conditional Domain-Adversarial Network (CDAN) [12] architectures, replacing the conventional convolutional feature extractors while leaving the adaptation mechanisms unchanged. The resulting frameworks are evaluated through systematic cross-load transfer experiments on the Case Western Reserve University (CWRU) bearing dataset [18] and a complementary cross-condition study on the Paderborn (PU) bearing dataset [19] to assess generalization beyond a single benchmark. The main contributions of this study are as follows:

- We introduce PadéNets as feature extractors within three established UDA frameworks—Deep CORAL, DANN, and CDAN—replacing conventional convolutional encoders with higher-order rational operators while leaving each framework's adaptation mechanism unchanged.
- Since the PadéNet formulation subsumes CNN ($P = 1$, $Q = 0$) and Self-ONN ($P > 1$, $Q = 0$) encoders as special cases, we perform a controlled comparison under an identical architecture on the CWRU cross-load benchmark using raw time-domain vibration signals. PadéNet encoders achieve the highest average target-domain accuracy in all three frameworks, with Self-ONN variants generally falling between the CNN and PadéNet.
- We demonstrate that these gains transfer to the more demanding Paderborn benchmark, where frequency-domain inputs are used and domain shift arises from simultaneous load-torque and radial-force variation on bearings with real accelerated-lifetime damage. The best CDAN–PadéNet configuration surpasses established methods, including TLADA [20], with less than half their parameter budget, supporting deployment on resource-constrained platforms.

The rest of this paper is organized as follows. Section 2 reviews related work on UDA methods for fault diagnosis. Section 3 presents the feature extraction architectures, progressing from 1D CNNs through 1D Self-ONNs to 1D PadéNets, and describes their integration into the Deep CORAL, DANN, and CDAN frameworks. Section 4 reports two complementary case studies, the first on the CWRU dataset under cross-load domain shift and the second on the Paderborn dataset under simultaneous variation of load torque and radial force with real accelerated-lifetime defects, covering dataset descriptions, preprocessing, experimental protocols, and analyses of diagnostic performance and computational complexity. Section 5 concludes the paper and outlines directions for future work.

2. Related Works

Fault diagnosis in induction machines has traditionally addressed electrical and mechanical fault categories using a range of sensing modalities, including vibration measurements, acoustic emissions, current signatures, and thermal imaging [17,21–23]. With recent progress in artificial intelligence, the field has shifted from conventional signal processing toward data-driven diagnostic models [24], among which, supervised learning has gained the widest adoption [25]. Deep architectures now dominate this landscape. CNNs learn discriminative spatial and temporal features directly from motor signals [26–29], Self-Organized Operational Neural Networks (Self-ONNs) extend them by optimizing nonlinear operators during training [6,21,23,30,31], Recurrent Neural Networks (RNNs) capture the temporal dependencies of sequential fault patterns [32–34], and hybrid models exploit the complementary strengths of different network types [35–37].

The success of these supervised approaches rests on two assumptions that are frequently violated in practice. The first is that training and test data follow the same distribution, a condition usually met only when data are collected from a single machine under controlled conditions; in deployment, operating conditions, environmental noise, and equipment configurations all vary. The second is the availability of sufficient labeled

fault data, because acquiring fault recordings from operational systems is difficult and risky, leading to scarce and imbalanced datasets for the most critical fault types. The resulting mismatch between source and target distributions is known as the domain shift problem [38].

Domain shift can be mitigated by jointly minimizing the classification error on the source domain and the distributional gap between domains [39], an idea that underlies domain adaptation (DA), a branch of transfer learning applicable to cross-domain fault diagnosis [40]. DA methods seek representations that are invariant to domain-specific characteristics while remaining fault-discriminative, thereby enabling knowledge transfer across operating conditions and equipment. Because labeling target-domain fault data is often impractical, unsupervised domain adaptation (UDA), which combines labeled source data with unlabeled target samples for distribution alignment, has attracted particular attention [41].

UDA methods are commonly categorized by the number of source domains, as single-domain or multi-domain adaptation, and within the single-domain setting by label consistency [9]. This study addresses label-consistent UDA, in which the source and target domains share the same fault categories. Methods in this setting fall into four principal groups, namely network-based, instance-based, mapping-based, and adversarial-based approaches [9].

Network-based methods exploit the hierarchical feature learning of deep networks through a transfer protocol in which a source-pretrained model is adapted to the target domain, keeping the early layers that encode generic low-level features fixed while fine-tuning the deeper task-specific layers [42–44]. Instance-based methods instead reduce domain divergence by adjusting the relative importance of individual training samples, as in TrAdaBoost [45] and Adaptive Batch Normalization (AdaBN) [46], which reweight or normalize source instances to alleviate covariate shift. AdaBN in particular has improved robustness to sensor noise and speed fluctuations in vibration signal analysis [5].

Mapping-based methods project the features of both domains into a shared space, such as a reproducing kernel Hilbert space or a low-dimensional manifold, in which statistical differences are explicitly minimized. Their effectiveness depends largely on the chosen distance metric, with Maximum Mean Discrepancy (MMD) [47] and Correlation Alignment (CORAL) [48] being the most established. MMD has been widely applied to the transfer of diagnostic features [49,50], and alternative metrics such as Maximum Variance Discrepancy (MVD) have likewise proven effective in reducing cross-domain differences [51].

Adversarial-based UDA, inspired by the Generative Adversarial Network (GAN) framework, has become one of the most prominent directions. These methods cast adaptation as a minimax game between a feature extractor, which seeks to produce domain-invariant representations, and a domain discriminator trained to distinguish source from target samples. The Domain-Adversarial Neural Network (DANN) [11] realizes this through a Gradient Reversal Layer (GRL) that permits end-to-end training, and Conditional Domain Adversarial Networks (CDAN) [12] refine the alignment by conditioning the discriminator on the classifier's predictions, which preserves the multimodal structure of the fault classes. Such frameworks have achieved strong results in cross-load and cross-machine diagnosis [52,53]. Wasserstein-distance-based variants such as WDGRL stabilize the adversarial objective by replacing domain classification with a smoother distance metric, and triplet loss guided adversarial domain adaptation (TLADA) further incorporates metric-learning objectives to enforce class-level alignment alongside data-level alignment [20].

Across all of these directions, attention has concentrated on the design of the alignment objective, while the feature extractor itself has remained an essentially standard

convolutional network. The present work addresses this complementary dimension by examining whether a more expressive rational-function encoder can improve established UDA frameworks without modifying their adaptation mechanisms.

3. Methods

This section describes the methodological foundations of the proposed approach for cross-domain fault diagnosis of rotating electrical machinery. We first present the feature extraction architectures, progressing from the conventional convolutional neuron through the generative neuron of Self-ONNs to the Padé neuron (PAON) underlying PadéNets, and highlight how the latter generalizes the former two. We then detail the integration of PadéNet encoders into three established UDA frameworks—Deep CORAL, DANN, and CDAN—in which the proposed extractors replace conventional convolutional backbones while each framework's adaptation mechanism is left unchanged. Although the discussion focuses on one-dimensional (1D) formulations suited to sequential vibration signals, the presented models extend readily to two-dimensional (2D) structures for image-based inputs.

3.1. Padé Approximant Neural Networks (PadéNets)

CNNs are widely used for learning structured representations from data, especially in tasks that require capturing local patterns and translation-invariant features. The CNN architecture can be configured as either one-dimensional (1D) or two-dimensional (2D) depending on the input data format. A 1D CNN applies convolution along a single spatial or temporal axis, enabling the model to capture patterns such as temporal dynamics, repetitive structures, and localized transients in sequential signals.

In a 1D CNN, assuming standard cross-correlation with unit stride and zero padding, the k^{th} pre-activation feature map in the l^{th} layer is given by Equation (1):

$$\mathbf{x}_k^{(l)} = b_k^{(l)} + \sum_{i=1}^{N_{l-1}} \mathbf{x}_{ik}^{(l)}. \tag{1}$$

Here, $\mathbf{x}_{ik}^{(l)} \in \mathbb{R}^M$ is the convolutional response from the i^{th} feature map of the previous layer to the k^{th} map of the current layer, $b_k^{(l)}$ is the learnable bias term (broadcast across the temporal dimension to match $\mathbb{R}^M$), and N_{l-1} is the number of feature maps in the previous layer. Letting $\mathbf{y}_i^{(l-1)} \in \mathbb{R}^M$ denote the i^{th} output activation map from layer $(l-1)$ and $\mathbf{w}_{ik}^{(l)} \in \mathbb{R}^K$ denote the corresponding kernel weights, the m^{th} element of $\mathbf{x}_{ik}^{(l)}$ is computed as

$$x_{ik}^{(l)}[m] = \sum_{r=0}^{K-1} w_{ik}^{(l)}[r] \cdot y_i^{(l-1)}[m+r], \tag{2}$$

where m is the temporal position and K is the kernel size. As defined in Equation (2), the kernel coefficients $w_{ik}^{(l)}[r]$ operate on a sliding window of the previous layer's output $y_i^{(l-1)}$ via cross-correlation. The aggregated result $\mathbf{x}_k^{(l)}$ is then passed through a nonlinear activation function to produce the final output of layer l.

Operational Neural Networks (ONNs) generalize the standard convolution by replacing the fixed linear weighted sum with more flexible operations [15]. To increase nonlinearity beyond pointwise activation functions, ONNs introduce a nodal operator ($\psi_k^{(l)} : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$) and a pool operator ($P_k^{(l)} : \mathbb{R}^K \to \mathbb{R}$) for each connection, resulting in a more expressive and learnable structure. With these two operators, Equation (3) defines the pre-activation contribution from the i^{th} input map:

$$\bar{x}_{ik}^{(l)}[m] = P_k^{(l)}\left(\left\{\psi_k^{(l)}\left(w_{ik}^{(l)}[r],\, y_i^{(l-1)}[m+r]\right)\right\}_{r=0}^{K-1}\right), \tag{3}$$

where m is the temporal index and r is the kernel index. Here, the nodal operator $\psi_k^{(l)}$ is applied element-wise to each weight–input pair, producing K intermediate values. The pool operator $P_k^{(l)}$ then aggregates these values into a single scalar output.

In ONNs, all neurons within a layer share the same nonlinear nodal operator (e.g., sinusoidal, exponential) and pooling strategy (e.g., median), replacing the fixed additive operation of CNNs. The optimal operator configurations are selected using the Greedy Iterative Search (GIS) algorithm, allowing the network to capture complex patterns through greater structural flexibility [15]. However, this design introduces two notable limitations [15]. First, using a single operator type across an entire layer restricts functional diversity, making it harder to model heterogeneous features. Second, the exhaustive offline search for operator selection is computationally expensive and may introduce pre-training bias.

Self-ONNs address these limitations by introducing generative neurons that learn their own nodal operators during training, removing the dependence on fixed, predefined functions [30]. Each neuron approximates its nonlinear function using a Taylor polynomial of order P, and the generative neuron operator in Equation (4) is optimized through backpropagation (BP) during training.

$$\tilde{\psi}_k^{(l)}\left(\{w_{p,ik}^{(l)}[r]\}_{p=1}^{P},\, y_i^{(l-1)}[m+r]\right) = \sum_{p=1}^{P} w_{p,ik}^{(l)}[r] \cdot \left(y_i^{(l-1)}[m+r]\right)^p. \tag{4}$$

The hyperparameter P directly controls the nonlinearity of the nodal operator by setting the polynomial order. Unlike CNNs, where each connection is defined by a single weight, Self-ONNs represent each connection with a P-dimensional coefficient vector, which expands the model's representational capacity and allows each neuron to learn a unique functional transformation. When additive pooling is used, the generative neuron output can also be written as P parallel convolutions:

$$\tilde{\mathbf{x}}_{ik}^{(l)} = \sum_{p=1}^{P} \mathrm{Conv1D}\left(\mathbf{w}_{p,ik}^{(l)},\ (\mathbf{y}_i^{(l-1)})^p\right). \tag{5}$$

Equation (5) shows that Self-ONNs remain architecturally compatible with existing CNN frameworks while offering the ability to capture complex nonlinear dependencies. However, increasing the polynomial order P to improve expressiveness can lead to numerical instability, requiring the use of bounded activation functions to keep the nodal operator's output within a stable range. While activation functions such as tanh provide a stable expansion point for the Taylor approximation, their saturating nature limits the effective gradient region, increasing the risk of vanishing gradients in deeper networks during backpropagation.

To address both the numerical instability at higher polynomial degrees in Self-ONNs and the vanishing gradient issues caused by bounded activations such as tanh, Padé Neurons (PAONs) have been proposed [16]. The Padé approximation represents a function as a ratio of two polynomials, offering better accuracy over a wider range of nonlinearities compared to Taylor series expansions [54]. While generative neurons in Self-ONNs rely on Maclaurin series with only positive powers, the Padé approximation extends this idea by introducing learnable terms in both the numerator (degree P) and denominator (degree Q). The resulting ratio of two polynomials of degree P and Q defines a Padé approximant of order $[P/Q]$, expressed as

$$f_{[P/Q]}(y) = \frac{R_P(y)}{S_Q(y)} = \frac{\sum_{m=0}^{P} a_m y^m}{\sum_{n=0}^{Q} b_n y^n}, \tag{6}$$

where a_m and b_n are the learnable parameters of the numerator and denominator polynomials, respectively. This rational structure, which is not present in CNNs or Self-ONNs, addresses the limited convergence radius of Taylor series expansions. By representing functions as ratios of polynomials, Padé approximants can capture pole singularities and remain valid beyond the convergence range of polynomial approximations, providing more accurate modeling of strongly nonlinear behavior. Setting $b_0 = 1$ to normalize the coefficients, Equation (6) simplifies to

$$f_{[P/Q]}(y) = \frac{a_0 + \sum_{m=1}^{P} a_m y^m}{1 + \sum_{n=1}^{Q} b_n y^n}. \tag{7}$$

Therefore, Equation (7) gives the normalized form used hereafter. Building on this rational function framework, Padé Activation Units (PAUs) [55] replace fixed activation functions (e.g., tanh, sigmoid) with learnable rational functions. Each PAU operates element-wise on its input, with coefficients $\{a_m^{(l)}, b_n^{(l)}\}$ shared across all feature maps within layer l and optimized during training. For the i^{th} input element at layer l, the PAU of Equation (8) is defined as

$$\varphi_{\text{PAU}}^{(l)}\left(x_i^{(l)}\right) = \frac{\sum_{m=0}^{P} a_m^{(l)} \left(x_i^{(l)}\right)^m}{1 + \sum_{n=1}^{Q} b_n^{(l)} \left(x_i^{(l)}\right)^n}. \tag{8}$$

Here, P and Q are the polynomial degrees of the numerator and denominator, respectively, and $\{a_m^{(l)}, b_n^{(l)}\}$ are the learnable scalar coefficients at layer l. This formulation can approximate standard activation functions (e.g., tanh or sigmoid) while also allowing the network to discover new, data-driven activation patterns. Extending this idea to the neuron level leads to Padé neurons (PAONs) [16], where the Padé approximation is embedded directly within the neuron operation itself. Unlike standard convolution, which applies activation functions after the weighted sum, PAONs in Equation (9) use separate convolution kernels for the numerator and denominator polynomials combined with powers of the input feature maps. The k^{th} feature map at layer l is computed as

$$\mathbf{x}_k^{(l)} = \frac{w_{p0,k}^{(l)} + \sum_{m=1}^{P} \sum_{i=1}^{N_{l-1}} \mathbf{w}_{pm,ik}^{(l)} * \left(\mathbf{y}_i^{(l-1)}\right)^m}{1 + \sum_{n=1}^{Q} \sum_{i=1}^{N_{l-1}} \mathbf{w}_{qn,ik}^{(l)} * \left(\mathbf{y}_i^{(l-1)}\right)^n}. \tag{9}$$

Here, $\mathbf{w}_{pm,ik}^{(l)}$ and $\mathbf{w}_{qn,ik}^{(l)}$ are the numerator and denominator convolution kernels, $\mathbf{y}_i^{(l-1)}$ is the i^{th} input feature map, and $*$ denotes convolution. Although such rational functions are more expressive than polynomials of comparable order, they inherit a well-known drawback of Padé approximants: the denominator is not guaranteed to be positive and may vanish, producing unbounded activations and unstable gradients during training.

To prevent this, we constrain the denominator to remain strictly positive by applying the absolute value to each term as

$$\mathbf{x}_k^{(l)} = \frac{w_{p0,k}^{(l)} + \sum_{m=1}^{P} \sum_{i=1}^{N_{l-1}} \mathbf{w}_{pm,ik}^{(l)} * \left(\mathbf{y}_i^{(l-1)}\right)^m}{1 + \sum_{n=1}^{Q} \sum_{i=1}^{N_{l-1}} \left| \mathbf{w}_{qn,ik}^{(l)} * \left(\mathbf{y}_i^{(l-1)}\right)^n \right|}. \tag{10}$$

In Equation (10), the denominator is bounded below by one for any input and any kernel values, which keeps both activations and gradients bounded throughout training and guarantees that no division by zero or near-zero can occur at initialization or at any later stage. Initialization also plays a role in training. In all experiments the numerator kernels $\mathbf{w}_{pm,ik}^{(l)}$ and the denominator kernels $\mathbf{w}_{qn,ik}^{(l)}$ are initialized with the Glorot-uniform scheme [56], matching the initialization of the CNN and Self-ONN baselines, while the bias $w_{p0,k}^{(l)}$ is initialized to zero. Each Padé layer therefore operates as a genuine rational function from the start of training, with the rational structure refined as the kernels are updated. The subsequent tanh activation further constrains each layer's output to the interval $(-1, 1)$, reinforcing the stability provided by the bounded denominator.

Figure 1 depicts the operations performed by a Padé neuron with $P = 2$ and $Q = 1$. As shown in Equation (10), the PAON architecture uses $(P + Q)$ convolution branches instead of a single one, where P and Q directly control the model complexity. This formulation is a superset of both CNNs and Self-ONNs: setting $P = 1$ and $Q = 0$ reduces the PAON to a standard CNN layer, while setting $P \geq 2$ and $Q = 0$ recovers the generative neuron used in Self-ONNs. Moderate increases in P and Q allow richer functional representations that can capture higher-degree nonlinearities beyond both architectures. For a kernel size K, a full PAON adds $(P + Q - 1) \times K \times C_{\text{in}} \times C_{\text{out}}$ parameters compared to a standard Conv1D layer, where C_{in} and C_{out} are the number of input and output feature maps, respectively. This overhead comes from the parallel convolution branches corresponding to each polynomial order.

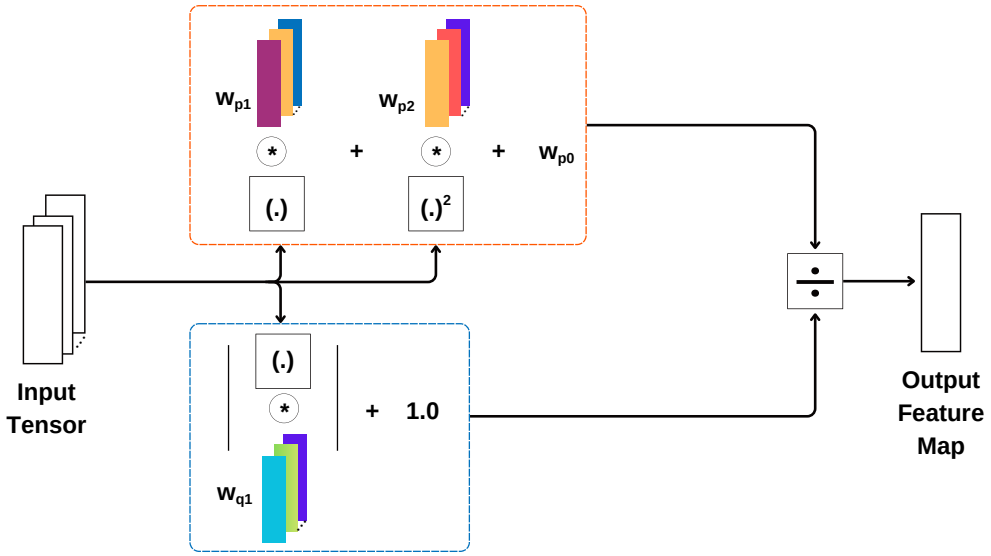

Figure 1. Schematic of the computational operations in a Padé neuron with $P = 2$ and $Q = 1$. Here, w_{p0} is the bias term in the numerator, $(\cdot)^n$ denotes element-wise exponentiation of the input to the n^{th} power, * denotes convolution, and $\div$ implements Equation (10).

When the numerator and denominator orders are comparable, PAONs can be trained stably using standard backpropagation. End-to-end training requires computing gradients

of the loss with respect to the numerator kernels $\mathbf{w}_{pm,ik}^{(l)}$, denominator kernels $\mathbf{w}_{qn,ik}^{(l)}$, and the previous layer's output feature maps $\mathbf{y}_i^{(l-1)}$. The full derivation of backpropagation through a PAON is provided in [17].

3.2. PadéNet Integration into Canonical UDA Frameworks

This subsection describes how PadéNet feature extractors are integrated into three well-established UDA frameworks: Deep CORAL [48], DANN [11], and CDAN [12]. In each framework, the conventional convolutional feature extractor is replaced by the proposed 1D PadéNets. Because the adaptation mechanisms of all three frameworks operate directly on the representations produced by the feature extractor—through statistical alignment in Deep CORAL and through adversarial feedback in DANN and CDAN—the expressiveness of the extractor directly determines the quality of the alignment. The main goal of this integration is therefore to leverage the richer representational capacity of rational Padé neurons to learn domain-invariant features that are robust to the distribution shifts commonly encountered in industrial vibration signals, while leaving the alignment objectives, optimization procedures, and architectural topology of each canonical UDA method unchanged.

3.2.1. Deep Correlation Alignment (Deep CORAL) with PadéNets

The Deep Correlation Alignment (Deep CORAL) framework, shown in Figure 2, reduces domain shift by minimizing the difference between the second-order statistics of the source and target feature distributions [48]. Unlike adversarial methods that rely on minimax optimization, Deep CORAL acts as a statistical regularizer that aligns the covariance matrices of the two domains in the learned feature space. In the proposed configuration, the shared feature extractor $G_\theta(\cdot)$ is realized as a 1D PadéNet: source- and target-domain vibration segments are passed through the same Padé backbone, and the resulting representations are forwarded to the label predictor $C_\psi(\cdot)$. The covariance alignment is performed on the pre-softmax activations of $C_\psi(\cdot)$, which preserves the structural properties of the PadéNet-derived features before they are compressed by the nonlinear softmax transformation. Since the CORAL loss is differentiable with respect to the network parameters, its gradients propagate back through the label predictor into the PadéNet backbone, jointly updating both the numerator and denominator kernels of the rational Padé neurons. In this way, the Padé units themselves adapt their functional shapes to produce feature distributions whose second-order statistics are consistent across operating conditions.

The feature matrices $\mathbf{Z}_s \in \mathbb{R}^{n_s \times d}$ and $\mathbf{Z}_t \in \mathbb{R}^{n_t \times d}$ represent the pre-softmax activations obtained from the PadéNet-based pipeline for the source and target domains, respectively, where n_s and n_t are the number of source and target samples in the mini-batch, and d is the feature dimensionality. The source and target covariance matrices, $\mathbf{C}_s, \mathbf{C}_t \in \mathbb{R}^{d \times d}$, are defined in Equations (11) and (12), and computed as

$$\mathbf{C}_s = \frac{1}{n_s - 1}(\mathbf{Z}_s - \mathbf{1}_{n_s}\boldsymbol{\mu}_s)^\top(\mathbf{Z}_s - \mathbf{1}_{n_s}\boldsymbol{\mu}_s), \tag{11}$$

$$\mathbf{C}_t = \frac{1}{n_t - 1}(\mathbf{Z}_t - \mathbf{1}_{n_t}\boldsymbol{\mu}_t)^\top(\mathbf{Z}_t - \mathbf{1}_{n_t}\boldsymbol{\mu}_t). \tag{12}$$

Here, $\boldsymbol{\mu}_s$ and $\boldsymbol{\mu}_t$ are the sample mean vectors. The domain discrepancy is then measured using the CORAL loss, $\mathcal{L}_{\text{CORAL}}$, defined in Equation (13) as the squared Frobenius norm of the covariance difference:

$$\mathcal{L}_{\text{CORAL}}(\theta, \psi) = \frac{1}{4d^2}\|\mathbf{C}_s - \mathbf{C}_t\|_F^2. \tag{13}$$

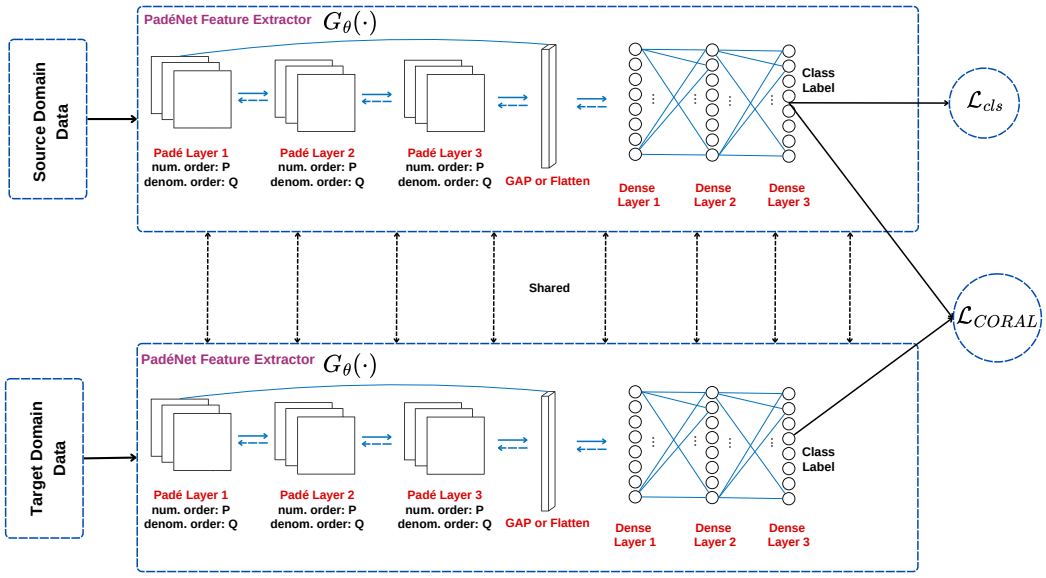

Figure 2. Representative architecture of the Deep CORAL framework with a PadéNet feature extractor.

To ensure that the representations learned by the PadéNet extractor are both domain-invariant and discriminative, the network is trained by minimizing a combined loss function. The classification term, in Equation (14), uses categorical cross-entropy to train the label predictor on the source domain:

$$\mathcal{L}_{cls}(\theta, \psi) = -\frac{1}{n_s} \sum_{i=1}^{n_s} \sum_{c=1}^{L} y_{i,c} \log \operatorname{softmax}\Big(C_\psi^{(c)}(G_\theta(x_{s,i}))\Big), \tag{14}$$

where L is the total number of fault categories. The overall training objective for the Deep CORAL–PadéNet framework is

$$\mathcal{L}_{\text{total}}(\theta, \psi) = \mathcal{L}_{cls}(\theta, \psi) + \lambda \mathcal{L}_{\text{CORAL}}(\theta, \psi) + \mathcal{L}_{reg}. \tag{15}$$

In Equation (15), λ is a trade-off hyperparameter that controls the weight of the alignment loss, and $\mathcal{L}_{reg}$ represents the L2 kernel regularization applied to the learnable parameters $\{\theta, \psi\}$.

3.2.2. Domain-Adversarial Neural Network (DANN) with PadéNets

The DANN framework uses a minimax game to learn features that are both discriminative for fault classification and invariant to load-induced domain shifts [11]. In the proposed DANN–PadéNet configuration, illustrated in Figure 3, the 1D PadéNet serves as the shared feature extractor $G_\theta(\cdot)$ that feeds two downstream branches: the label classifier $C_\psi(\cdot)$, trained on labeled source samples, and the domain discriminator $D_\phi(\cdot)$, which receives PadéNet features from both domains. The adversarial training is enabled by a Gradient Reversal Layer (GRL) placed between the 1D PadéNet extractor and the domain classifier $D_\phi(\cdot)$. As given in Equation (16), the GRL acts as an identity function during the forward pass, but scales the gradients by $-\lambda$ during backpropagation:

$$\mathcal{R}_\lambda(\mathbf{z}) = \mathbf{z}, \quad \frac{d\mathcal{R}_\lambda}{d\mathbf{z}} = -\lambda \mathbf{I}. \tag{16}$$

Through the GRL, the reversed adversarial gradients flow directly into the PadéNet backbone, so that the learnable parameters are updated to confuse the domain discriminator. The flexible functional form of the Padé units allows the extractor to reshape its nonlinearities during this adversarial process, suppressing load-dependent signal charac-

teristics while retaining fault-related spectral patterns. The domain classifier is trained to minimize the binary cross-entropy loss $\mathcal{L}_{dom}$ given in Equation (17), while the PadéNet feature extractor is updated to maximize it, which removes domain-specific information from the learned features:

$$\mathcal{L}_{dom}(\theta, \phi) = -\frac{1}{n_s + n_t} \sum_{i=1}^{n_s+n_t} [d_i \log D_\phi(G_\theta(x_i)) + (1 - d_i) \log(1 - D_\phi(G_\theta(x_i)))]. \quad (17)$$

The global objective is formulated as

$$\min_{\theta,\psi} \max_{\phi} \mathcal{L}_{cls}(\theta, \psi) - \lambda \mathcal{L}_{dom}(\theta, \phi) + \mathcal{L}_{reg}. \quad (18)$$

In Equation (18), λ typically follows a sigmoidal schedule that lets the classifier converge before the adversarial term takes full effect:

$$\lambda(p) = \frac{2}{1 + \exp(-10p)} - 1, \quad (19)$$

where p denotes the normalized training progress [11]. This warm-up schedule is particularly beneficial for the PadéNet extractor, as it allows PAONs to first stabilize around discriminative solutions on the source domain before the adversarial signal begins to reshape them toward domain invariance.

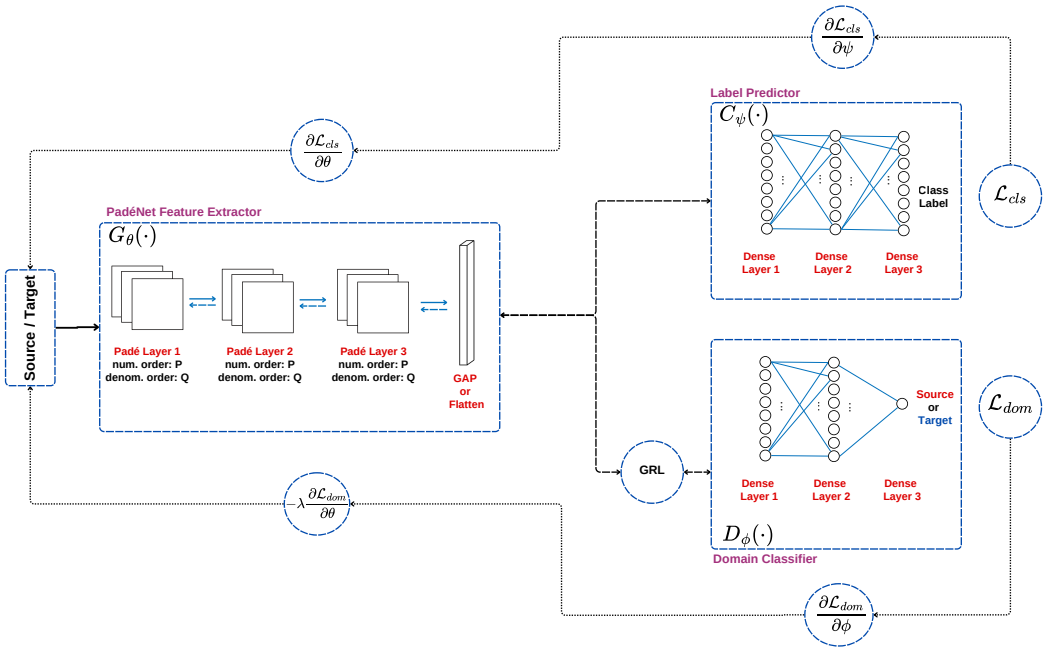

Figure 3. Representative architecture of a Domain-Adversarial Neural Network (DANN) with a PadéNet feature extractor G_θ, label classifier C_ψ, and domain discriminator D_ϕ, linked through a gradient reversal layer (GRL).

3.2.3. Conditional Adversarial Domain Adaptation (CDAN) with PadéNets

While DANN aligns marginal distributions, CDAN handles multimodal distribution shifts by conditioning the domain discriminator to the classifier's predictive output [12]. This prevents distinct fault categories from being incorrectly aligned and overlapping in the latent space. In the CDAN–PadéNet configuration, the integration follows the same principle as in DANN: the 1D PadéNet replaces the conventional feature extractor $G_\theta(\cdot)$, and its output simultaneously serves as the latent representation $\mathbf{f} = G_\theta(\mathbf{x})$ and as the input to the label classifier that produces the probability vector $\mathbf{p} = C_\psi(\mathbf{f})$. The conditioning

mechanism of CDAN therefore operates entirely on quantities derived from the PadéNet backbone, so that the discriminative structure encoded by the Padé layers is explicitly exposed to the domain discriminator.

As shown in Figure 4, the conditioning is implemented through a multilinear map that captures the cross-covariance between the latent PadéNet features $\mathbf{f}$ and the probability vector $\mathbf{p} = C_\psi(\mathbf{f})$. The joint representation $\mathbf{h}$ in Equation (20) is formed by vectorizing the outer product:

$$\mathbf{h} = \mathrm{vec}(\mathbf{f} \otimes \mathbf{p}) = \mathrm{vec}(\mathbf{f}\mathbf{p}^\top), \tag{20}$$

where $\mathbf{h} \in \mathbb{R}^{d \times L}$. The domain discriminator $D_\phi(\cdot)$ then uses $\mathbf{h}$ to minimize the conditional domain loss given in Equation (21) as

$$\mathcal{L}_{dom}(\theta, \phi) = -\mathbb{E}_{\mathbf{x}_s \sim \mathcal{D}_s}[\log(1 - D_\phi(\mathbf{h}_s))] - \mathbb{E}_{\mathbf{x}_t \sim \mathcal{D}_t}[\log D_\phi(\mathbf{h}_t)]. \tag{21}$$

The CDAN–PadéNet framework uses the same minimax optimization structure, GRL-based gradient reversal, and adaptive λ schedule as the DANN–PadéNet configuration, with the reversed gradients again propagating through both branches into the shared PadéNet extractor. By explicitly modeling the interaction between the PadéNet-derived latent features and the class probabilities, the CDAN–PadéNet configuration ensures semantically consistent, class-aware alignment across different operating conditions.

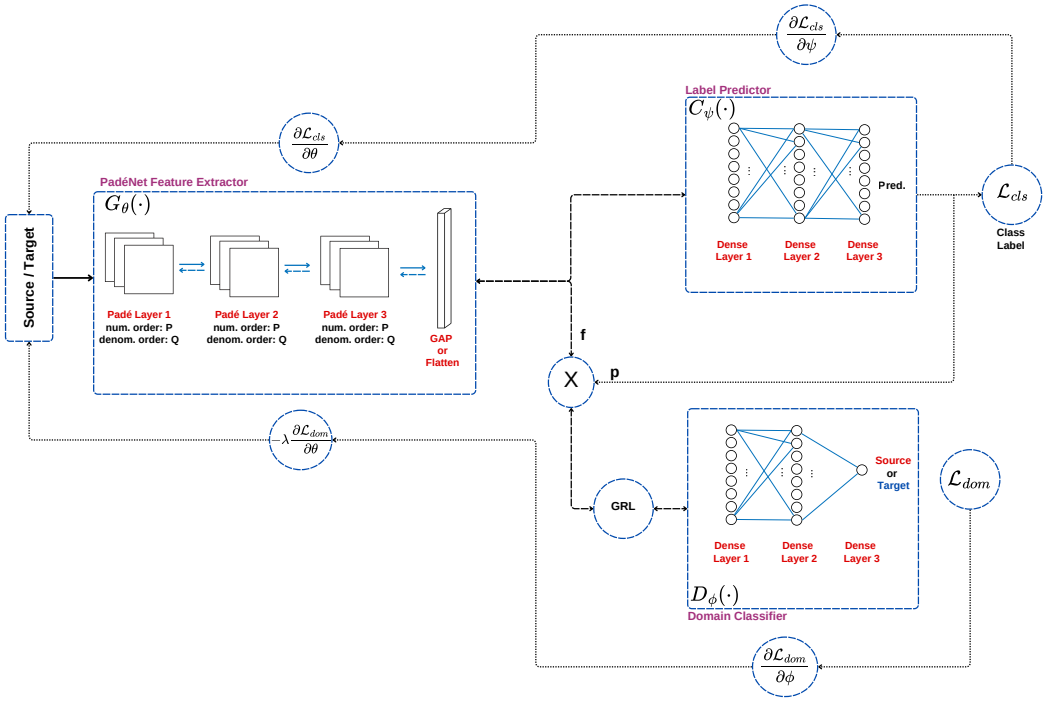

Figure 4. Representative architecture of a Conditional Domain Adversarial Network (CDAN) with a PadéNet feature extractor G_θ, label classifier C_ψ, and predictor-conditioned domain discriminator D_ϕ.

4. Experiments and Results

In this section, we present a comprehensive empirical evaluation of the proposed PadéNet-based UDA frameworks. Two complementary case studies are conducted to assess both diagnostic accuracy and cross-domain generalization. The first case study uses the Case Western Reserve University (CWRU) bearing dataset [18], where load-induced domain shift is examined across all 12 cross-load transfer scenarios spanning the four operating loads (0, 1, 2, and 3 hp). The second case study employs the Paderborn University (PU) bearing dataset [19], which introduces a more demanding setting by combining simultaneous variations in load torque and radial force on naturally developed, accelerated-

lifetime defects across six cross-condition transfer tasks. Throughout these studies, PadéNet feature extractors of varying numerator and denominator orders (P, Q) are integrated into the Deep CORAL, DANN, and CDAN frameworks and benchmarked against conventional 1D CNN ($P = 1$, $Q = 0$) and 1D Self-ONN ($P > 1$, $Q = 0$) backbones using an identical architecture, isolating the contribution of the rational-function parameterization. All models are trained with labeled source data and unlabeled target data only, with evaluation performed exclusively on held-out target test sets. Results are reported as mean $\pm$ standard deviation over five independent runs, and are further supported by t-SNE visualizations, confusion matrices, and analyses of model complexity and inference cost.

4.1. Case Study I: Unsupervised Domain Adaptation on CWRU Bearing Dataset

We first introduce the CWRU bearing dataset, the preprocessing of the raw vibration signals, the cross-load adaptation protocol, and the network and training setup used to evaluate the proposed PadéNet-based UDA frameworks.

4.1.1. CWRU Bearing Dataset

The Case Western Reserve University (CWRU) bearing dataset [18] is used as the benchmark for evaluating the proposed PadéNet-based domain adaptation frameworks. This dataset is widely adopted in the bearing fault diagnosis community, allowing direct comparison with existing methods while covering realistic industrial bearing failure modes. As shown in Figure 5, the experimental setup consists of a 2 hp three-phase induction motor coupled to a torque transducer/encoder and dynamometer, enabling data collection under controlled loading conditions.

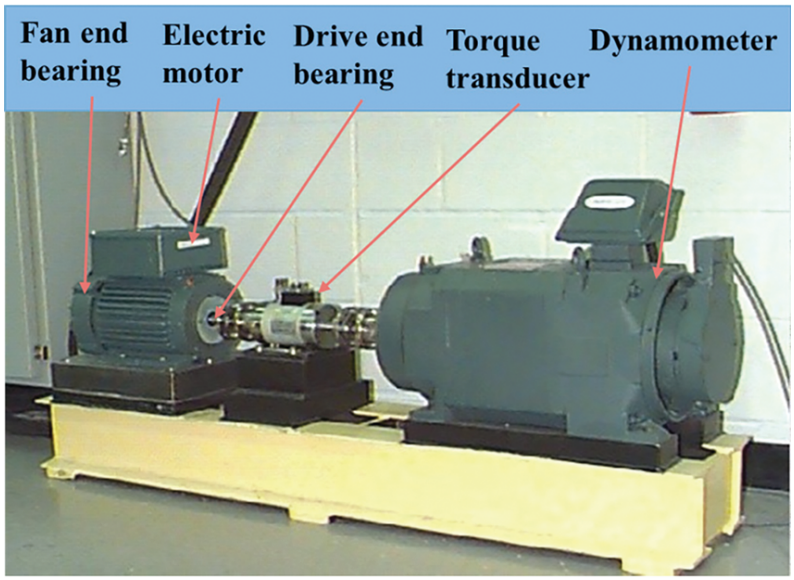

Figure 5. Experimental configuration of the CWRU bearing test rig [57].

Localized bearing defects were artificially introduced into ball bearings using electro-discharge machining (EDM), a commonly used method for simulating real-world pitting and spalling failures. Faults were seeded at three locations: the inner raceway (IR), rolling elements (ball, B), and outer raceway (OR)—the most common failure sites in rotating machinery. Three fault severities were considered, with defect diameters of 0.007 inches (0.178 mm), 0.014 inches (0.356 mm), and 0.021 inches (0.533 mm), representing early, moderate, and severe degradation levels. Together with the healthy baseline condition, this results in a ten-class fault classification covering the full range of bearing health states (Table 1).

Table 1. Bearing fault classification taxonomy for the CWRU dataset.

Class Label	Fault Location	Fault Diameter	Class Identifier
0	Healthy (baseline)	—	Normal
1	Inner raceway	0.007 in (0.178 mm)	IR007
2	Inner raceway	0.014 in (0.356 mm)	IR014
3	Inner raceway	0.021 in (0.533 mm)	IR021
4	Ball element	0.007 in (0.178 mm)	B007
5	Ball element	0.014 in (0.356 mm)	B014
6	Ball element	0.021 in (0.533 mm)	B021
7	Outer raceway	0.007 in (0.178 mm)	OR007
8	Outer raceway	0.014 in (0.356 mm)	OR014
9	Outer raceway	0.021 in (0.533 mm)	OR021

Vibration signals were recorded using piezoelectric accelerometers mounted on the motor housing at both the drive-end (DE) and fan-end (FE) bearing locations. Following common practice in the literature, only the DE measurements are used in this study, as they offer higher sensitivity to bearing-induced vibrations. Data were collected under four motor loading conditions—0, 1, 2, and 3 hp—applied through the dynamometer to simulate different operational demands. Each load level is associated with a slightly different shaft speed, since the induction motor slip increases with load, giving approximately 1797 rpm at 0 hp, 1772 rpm at 1 hp, 1750 rpm at 2 hp, and 1730 rpm at 3 hp. The dominant source of domain shift between the four conditions is therefore the mechanical load itself, accompanied by a small speed variation that proportionally shifts the characteristic defect frequencies. The faulty bearing signals were sampled at 12 kHz, providing sufficient bandwidth to capture characteristic fault frequencies and their harmonics. Normal condition data, originally sampled at 48 kHz, were downsampled by a factor of four to ensure consistency across all conditions.

To show the domain shift in the CWRU dataset, we compare the three 0.021″ drive-end faults—inner race, outer race, and ball—under two loads, 0 hp and 3 hp, which correspond to shaft speeds of about 1797 and 1730 rpm. Because bearing defects produce modulated impulses rather than clean tones, we use envelope analysis, where each signal is band-pass filtered around the bearing resonance band (2–5 kHz) and the Fourier transform of its Hilbert envelope is computed.

As shown in Figure 6, the envelope spectra exhibit clear peaks at the expected fault frequencies of the SKF 6205 bearing (BPFI, BPFO, and BSF), which confirms each fault type. As the load increases and the shaft slows down, each fault frequency shifts to a lower value—BPFI from 162 to 156 Hz, BPFO from 107 to 103 Hz, and BSF from 71 to 68 Hz—together with clear changes in peak height and harmonic content. This shift in the features that a classifier depends on is the domain shift that reduces cross-load generalization, and it is the main motivation for the domain adaptation method proposed in this work.

The ball-fault signatures are weaker and less clear than those of the inner- and outer-race faults. This is expected because the ball spins and changes orientation as it moves through the load zone, so its impacts are irregular and strongly modulated by the cage motion, which spreads the energy and broadens the BSF peak. As a result, ball faults are the hardest type to identify and pose a particular problem for domain adaptation, since their features are already faint and easily hidden. An adapted model must align an already weak and noisy signature across loads, where even a small shift can make it hard to separate from the surrounding spectrum.

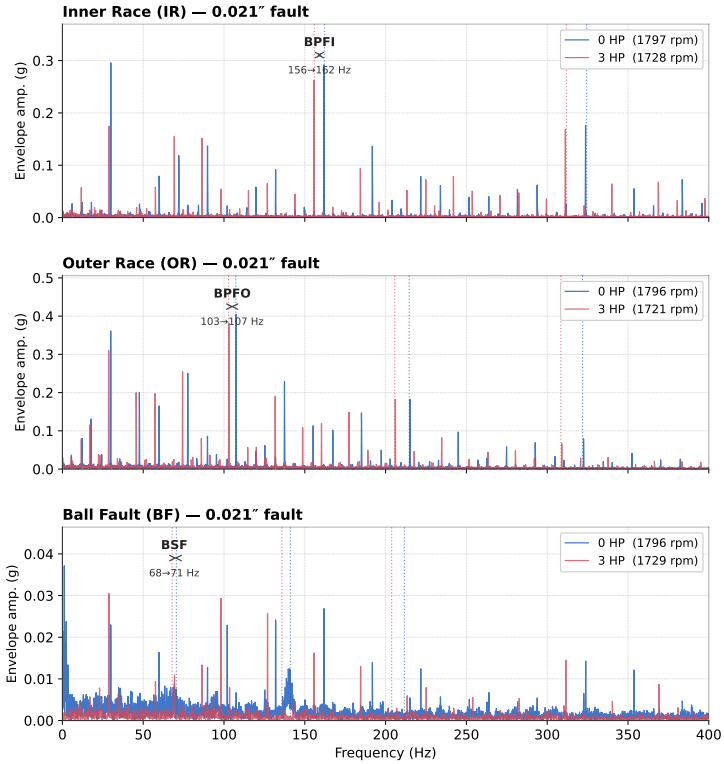

Figure 6. Envelope spectra of the three 0.021″ drive-end faults (inner race, outer race, and ball) under 0 hp and 3 hp loads. Each fault frequency (BPFI, BPFO, and BSF) shifts to a lower value as the load increases and the shaft slows, illustrating the load-induced domain shift.

4.1.2. Preprocessing Pipeline for CWRU Bearing Dataset

As shown in Figure 7, the raw vibration signals were segmented into fixed-length samples using a sliding window approach. Each window contains 1024 data points (approximately 85.3 ms at 12 kHz), chosen to cover multiple shaft rotations while keeping computational cost manageable. Windows were extracted with a stride of 512 samples (50% overlap) to increase the number of training samples while preserving diversity and reducing boundary effects. To account for amplitude variations caused by different loading conditions, and to improve training stability, min–max normalization was applied independently to each segment, scaling all values to the $[-1, 1]$ range.

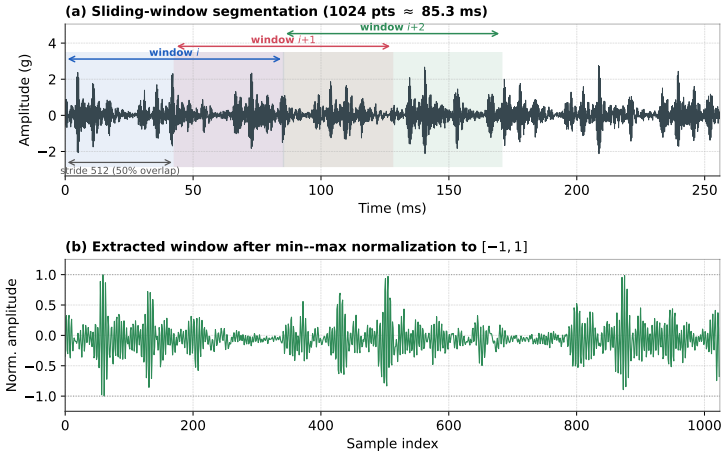

Figure 7. Preprocessing applied to a sample drive-end vibration signal. (**a**) Sliding-window segmentation with 1024-point windows (≈85.3 ms at 12 kHz) and a stride of 512 (50% overlap). (**b**) One extracted window after per-segment min–max normalization to $[-1, 1]$.

The four loading conditions (0, 1, 2, and 3 hp) introduce systematic changes in vibration signal characteristics due to differences in contact mechanics, damping properties, and resonance behavior. These load-dependent variations form natural domain boundaries, making the CWRU dataset well suited for evaluating cross-domain generalization. The resulting sample distribution after segmentation is given in Table 2.

Table 2. Dataset composition showing sample counts per fault class across four operating loads.

Fault Class	0 hp	1 hp	2 hp	3 hp	Total
Normal	118	235	235	236	824
IR007	235	237	237	239	948
IR014	236	236	236	236	944
IR021	237	236	236	237	946
B007	238	236	236	236	946
B014	236	237	237	237	947
B021	237	236	237	237	947
OR007	237	238	236	238	949
OR014	236	237	236	237	946
OR021	238	237	237	237	949
Total per Load	2248	2365	2363	2370	9346

4.1.3. Domain Adaptation Experimental Protocol

This study addresses the UDA setting, where diagnostic models must generalize to new loading conditions without access to labeled target data during training. Cross-domain transfer tasks are denoted as $\mathcal{S} \rightarrow \mathcal{T}$, where $\mathcal{S}$ is the labeled source domain and $\mathcal{T}$ is the unlabeled target domain.

During training, the model has access to a labeled source dataset $\mathcal{D}_s = \{(\mathbf{x}_i^s, y_i^s)\}_{i=1}^{n_s}$ and an unlabeled target dataset $\mathcal{D}_t = \{\mathbf{x}_i^t\}_{i=1}^{n_t}$, where $\mathbf{x} \in \mathbb{R}^{1024}$ is the input vibration segment and $y \in \{0, 1, \ldots, 9\}$ is the corresponding fault class. The goal is to minimize classification error on the target domain despite the distribution shift, i.e., $P_s(\mathbf{x}, y) \neq P_t(\mathbf{x}, y)$. Generalization is evaluated on a held-out labeled target test set that is strictly isolated from the adaptation process. For example, the task $0 \rightarrow 1$ means training on labeled 0 hp data while adapting to unlabeled 1 hp samples, with evaluation performed on reserved 1 hp test data. Excluding same-domain pairs (e.g., $0 \rightarrow 0$), this design yields 12 unique cross-load transfer scenarios, covering all possible load transitions.

To ensure statistical reliability and reproducibility, a stratified 5-fold cross-validation protocol is applied to each of the 12 transfer tasks, as shown in Figure 8. Both the source and target datasets are split into five stratified folds that preserve the original class distribution. In each fold iteration, four folds (80%) from both domains are used for training, where source folds provide labeled supervision and target folds contribute only unlabeled samples for adaptation. The remaining target fold (20%) is used exclusively for evaluation. This is repeated across all five rotations, yielding five independent accuracy measurements per transfer task, reported as mean $\pm$ standard deviation. Importantly, the test fold is entirely excluded from the training and adaptation process, preventing data leakage and ensuring that reported accuracies reflect true generalization to unseen target samples.

No separate validation fold was used for hyperparameter tuning or early stopping. All hyperparameters were fixed a priori and kept identical across all transfer tasks, feature extractors, and both case studies wherever applicable. The optimizer settings, learning rate, batch size, and number of epochs follow common practice for the respective frameworks, the CORAL trade-off $\lambda = 1.0$ follows [48], and the sigmoidal λ schedule for DANN/CDAN follows [11]. Training always runs for the full 100 epochs and the final-epoch model is evaluated, so no labeled target data influences any modeling decision. This avoids the

optimistic bias that arises when hyperparameters are tuned, even indirectly, on target-domain labels, which would violate the unsupervised assumption of the UDA setting. The reported standard deviations across the five folds therefore reflect the variability of the method under a fully fixed configuration.

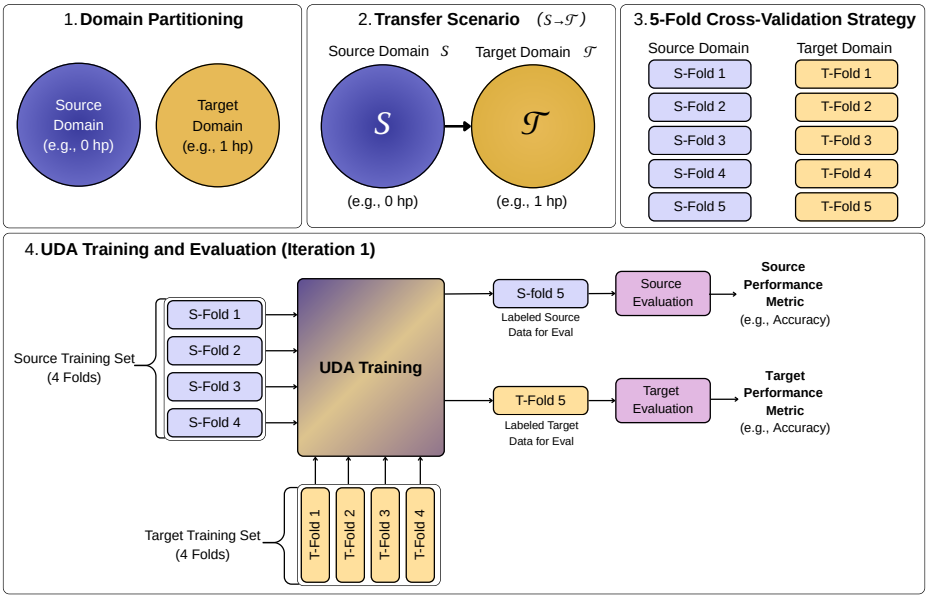

Figure 8. Overview of the proposed stratified 5-fold cross-validation protocol for unsupervised domain adaptation. In each fold iteration, the model is trained using labeled source-domain samples and unlabeled target-domain samples, while evaluation is performed exclusively on the held-out labeled target fold to ensure unbiased assessment of cross-domain generalization.

All frameworks are evaluated on the held-out labeled target test set, which remains unseen during adaptation, using classification accuracy as the primary metric together with precision, recall, and F1-score weighted by class support to account for class imbalance. For every transfer task, all metrics are computed independently on each of the five folds and reported as mean $\pm$ standard deviation, and per-fold confusion matrices are aggregated to identify the fault types that remain difficult under domain shift.

4.1.4. Proposed Network and Training Configurations for the CWRU Bearing Dataset

To isolate the contribution of each adaptation mechanism, a unified 1D PadéNet architecture is employed as the shared feature extractor $G_\theta(\cdot)$ across all three frameworks. In contrast to conventional CNNs, which rely on linear convolutional filters, the PadéNet backbone applies rational operators of orders (P, Q) within four sequential Padé blocks. Since the PadéNet formulation subsumes CNN ($P = 1$, $Q = 0$) and Self-ONN ($P \geq 2$, $Q = 0$) encoders as special cases (Section 3.1), all three neuron types are compared under identical architectures.

Each of the four blocks comprises 64 filters with a kernel size of 3, as illustrated in Figures 9–11. Instance normalization is applied after every Padé layer with the learnable affine parameters (γ, β) disabled. Changes in mechanical load act on the vibration signal mainly as an amplitude scaling and a mean shift of each feature map, whereas the diagnostic structure—the spacing of fault-related transients, the harmonic ratios, and the waveform impulsiveness—is far less affected. Standardizing each feature map to zero mean and unit variance over the temporal axis removes these per-domain offsets and scalings before the alignment losses, while preserving the temporal morphology that carries the fault signatures. The affine pair (γ, β) is disabled because it provides a per-channel rescaling capacity immediately after normalization that the network can exploit to partially undo

this domain-symmetrizing effect. Although the adversarial objective propagates target gradients through these layers, the classification signal—available only on the source domain—dominates the optimization of the affine parameters, so retaining them would re-introduce a source-biased rescaling. Disabling them keeps the embedding scale-free and domain-symmetric. The normalized feature maps are then passed through hyperbolic tangent (tanh) activation and downsampled via max pooling. Finally, a Global Average Pooling (GAP) layer followed by a dropout layer (rate $= 0.25$) produces a 64-dimensional latent embedding $\mathbf{f} \in \mathbb{R}^d$, which serves as the common input to the framework-specific alignment modules described below.

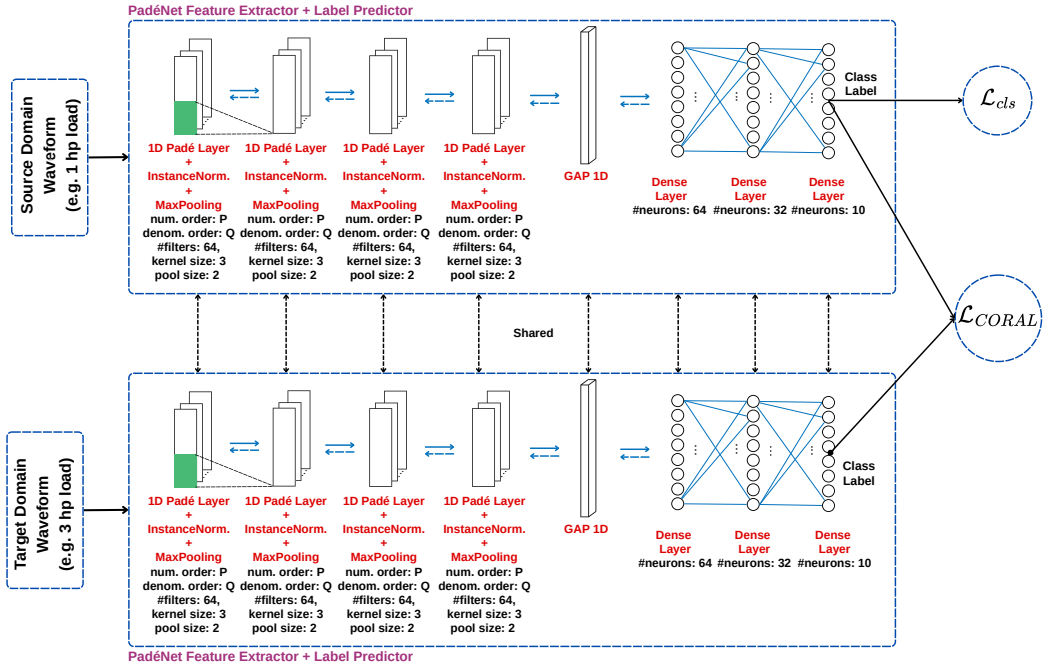

Figure 9. Deep CORAL with the proposed 1D PadéNet backbone for bearing fault diagnosis on CWRU dataset.

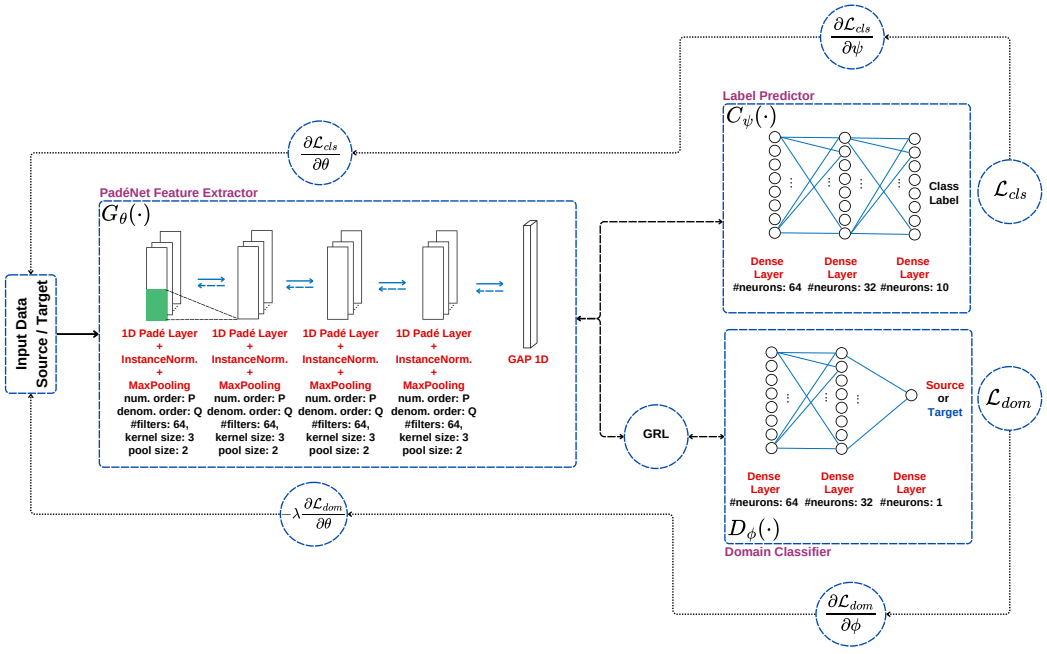

Figure 10. Proposed DANN architecture utilizing a GRL for adversarial distribution alignment on CWRU dataset.

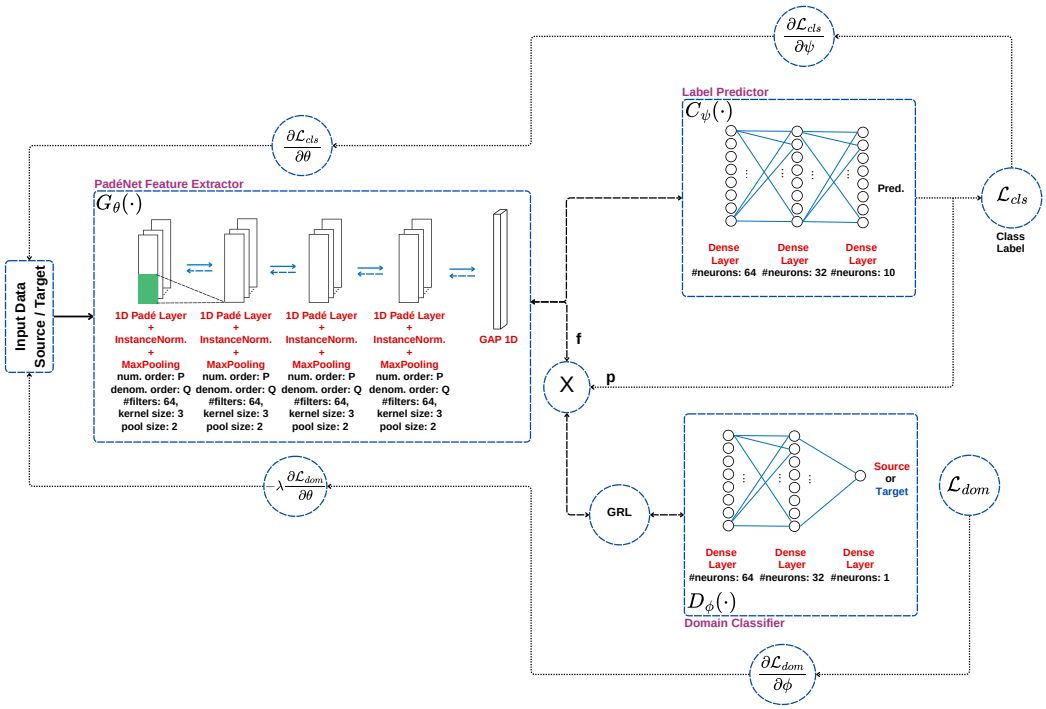

Figure 11. Proposed CDAN architecture with multilinear conditioning of features and class probabilities on CWRU dataset.

The resulting embedding **f** is consumed by the three framework-specific alignment modules, whose complete architectures are depicted in Figure 9 for Deep CORAL, Figure 10 for DANN, and Figure 11 for CDAN. All three frameworks share a common optimization configuration: each model is trained end-to-end using the Adam optimizer with a learning rate of 2×10^{-4} for 100 epochs and a mini-batch size of 64. The frameworks differ only in their respective alignment objectives and the scheduling of the trade-off hyperparameter λ. The trade-off parameter follows $\lambda = 1.0$ for Deep CORAL [48] and the sigmoidal schedule of Equation (19) for DANN and CDAN [11].

4.1.5. Results on the CWRU Bearing Dataset

This section presents the experimental evaluation of the three UDA frameworks—Deep CORAL, DANN, and CDAN—combined with 1D CNN, 1D Self-ONN, and the proposed 1D PadéNet feature extractors on all 12 cross-load transfer tasks ($S \rightarrow T$) defined by the four operating loads (0, 1, 2, and 3 hp). Following the protocol of Section 4.1.3, each model is trained with labeled source and unlabeled target data, and evaluation is performed exclusively on held-out target test sets, with results reported as mean $\pm$ standard deviation over five folds (Tables 3–8). Because the weighted F1-scores closely follow the corresponding accuracies in all configurations, the discussion below focuses on accuracy, while the F1 tables are provided for completeness.

To quantify the severity of the domain shift, a source-only model—a standard 1D CNN ($P = 1, Q = 0$) trained on labeled source data with hyperparameters identical to the adaptive experiments—serves as the common reference. Its average target-domain accuracy of 90.47% conceals a strongly asymmetric error distribution: transfers between adjacent loads remain above 95%, whereas models trained at 3 hp degrade sharply under light-load conditions, falling to 74.69% on $3 \rightarrow 0$ and 77.34% on $3 \rightarrow 1$. The cross-load generalization problem on this benchmark is therefore concentrated in a small number of difficult transfers, which are examined repeatedly in the analysis that follows.

Deep CORAL (Tables 3 and 4) recovers a substantial portion of this performance loss through covariance alignment alone. With a CNN backbone, the average accuracy rises to

95.95%, and the $3 \rightarrow 0$ task improves from 74.69% to 89.24%. Replacing the CNN with a PadéNet $(P = 1, Q = 1)$ encoder increases the average to 97.77% and the $3 \rightarrow 0$ accuracy to 94.70%. The Self-ONN variants also improve on the CNN but consistently remain below the rational configurations, an ordering that recurs in every framework considered.

Table 3. Target-domain accuracy (%) for Deep CORAL with PadéNet feature extractors across all cross-load transfer tasks. Best results per task shown in bold.

	Source-Only	**Deep CORAL with PadéNet (P, Q)**					
Task	**(1, 0)**	**(1, 0)**	**(1, 1)**	**(1, 2)**	**(2, 0)**	**(2, 1)**	**(3, 0)**
$0 \rightarrow 1$	95.43 ± 3.39	97.80 ± 1.35	**98.99 ± 1.07**	98.86 ± 1.00	97.63 ± 0.84	97.38 ± 1.08	97.04 ± 0.93
$0 \rightarrow 2$	93.78 ± 4.13	96.91 ± 2.64	**99.66 ± 0.44**	99.58 ± 0.39	99.11 ± 0.28	98.90 ± 1.04	98.73 ± 0.72
$0 \rightarrow 3$	80.84 ± 4.57	93.80 ± 2.22	97.76 ± 0.84	**99.16 ± 0.62**	96.24 ± 0.81	96.79 ± 2.23	96.96 ± 1.73
$1 \rightarrow 0$	98.40 ± 0.55	**98.93 ± 0.46**	98.58 ± 1.17	98.67 ± 1.20	97.20 ± 1.10	97.82 ± 0.95	97.11 ± 1.37
$1 \rightarrow 2$	99.87 ± 0.19	**100.00 ± 0.00**	**100.00 ± 0.00**	**100.00 ± 0.10**	99.70 ± 0.35	99.87 ± 0.19	99.79 ± 0.26
$1 \rightarrow 3$	92.66 ± 3.89	99.28 ± 0.35	99.32 ± 0.46	**99.49 ± 0.41**	98.14 ± 0.97	98.48 ± 0.78	98.35 ± 0.86
$2 \rightarrow 0$	95.55 ± 1.35	95.95 ± 1.61	97.24 ± 1.01	**97.73 ± 1.10**	95.82 ± 1.66	96.44 ± 0.93	95.73 ± 1.79
$2 \rightarrow 1$	95.94 ± 1.84	97.67 ± 1.05	98.52 ± 0.30	**98.56 ± 0.28**	97.67 ± 0.75	98.22 ± 0.61	98.05 ± 0.28
$2 \rightarrow 3$	95.36 ± 4.82	99.07 ± 0.49	98.86 ± 1.00	**99.32 ± 0.35**	98.61 ± 1.06	99.16 ± 0.78	98.57 ± 1.19
$3 \rightarrow 0$	74.69 ± 3.44	89.24 ± 4.10	94.70 ± 1.75	94.53 ± 1.80	92.35 ± 2.24	**94.84 ± 1.02**	92.75 ± 1.75
$3 \rightarrow 1$	77.34 ± 2.01	86.05 ± 4.74	90.78 ± 6.03	84.82 ± 6.94	93.91 ± 1.99	**95.48 ± 2.51**	94.25 ± 2.49
$3 \rightarrow 2$	85.82 ± 3.53	96.74 ± 2.14	**98.77 ± 0.62**	98.69 ± 0.23	98.01 ± 0.71	98.14 ± 1.46	97.33 ± 1.86
Average	90.47	95.95	**97.77**	97.45	97.03	97.58	97.06

Table 4. Target-domain F1-score (%) for Deep CORAL with PadéNet feature extractors across all cross-load transfer tasks. Best results per task shown in bold.

	Source-Only	**Deep CORAL with PadéNet (P, Q)**					
Task	**(1, 0)**	**(1, 0)**	**(1, 1)**	**(1, 2)**	**(2, 0)**	**(2, 1)**	**(3, 0)**
$0 \rightarrow 1$	95.02 ± 3.77	97.74 ± 1.42	**98.98 ± 1.08**	98.83 ± 1.03	97.59 ± 0.85	97.31 ± 1.13	96.96 ± 0.97
$0 \rightarrow 2$	92.21 ± 5.50	96.58 ± 3.12	**99.66 ± 0.44**	99.58 ± 0.30	99.11 ± 0.29	98.90 ± 1.05	98.72 ± 0.73
$0 \rightarrow 3$	77.02 ± 7.42	93.08 ± 2.98	97.75 ± 0.82	**99.16 ± 0.61**	96.18 ± 0.84	96.81 ± 2.20	96.96 ± 1.73
$1 \rightarrow 0$	98.40 ± 0.56	**98.93 ± 0.46**	98.58 ± 1.18	98.67 ± 1.18	97.21 ± 1.09	97.82 ± 0.94	97.11 ± 1.35
$1 \rightarrow 2$	99.87 ± 0.19	**100.00 ± 0.00**	**100.00 ± 0.00**	**100.00 ± 0.00**	99.70 ± 0.35	99.87 ± 0.19	99.79 ± 0.26
$1 \rightarrow 3$	92.17 ± 4.44	99.28 ± 0.35	99.32 ± 0.46	**99.49 ± 0.41**	98.14 ± 0.97	98.48 ± 0.79	98.35 ± 0.87
$2 \rightarrow 0$	95.52 ± 1.36	95.93 ± 1.66	97.24 ± 1.00	**97.71 ± 1.11**	95.80 ± 1.66	96.44 ± 0.92	95.73 ± 1.77
$2 \rightarrow 1$	95.69 ± 1.99	97.57 ± 1.13	98.48 ± 0.31	**98.53 ± 0.29**	97.58 ± 0.82	98.22 ± 0.68	98.00 ± 0.30
$2 \rightarrow 3$	95.22 ± 5.08	99.07 ± 0.49	98.85 ± 1.01	**99.33 ± 0.35**	98.61 ± 1.06	98.61 ± 1.20	98.56 ± 1.19
$3 \rightarrow 0$	67.96 ± 3.96	88.14 ± 4.80	94.43 ± 2.12	94.34 ± 1.93	91.76 ± 2.74	**94.65 ± 1.10**	92.37 ± 1.95
$3 \rightarrow 1$	70.35 ± 2.52	81.98 ± 7.45	88.87 ± 7.71	80.04 ± 9.90	92.80 ± 3.27	**94.92 ± 3.29**	93.61 ± 3.08
$3 \rightarrow 2$	83.49 ± 4.39	96.68 ± 2.20	**98.77 ± 0.63**	98.68 ± 0.24	98.00 ± 0.72	98.11 ± 1.51	97.31 ± 1.88
Average	88.58	95.41	**97.58**	97.03	96.87	97.51	96.96

Adversarial alignment yields further gains. DANN (Tables 5 and 6) reaches an average of 97.32% with the CNN backbone and 98.42% with PadéNet $(P = 1, Q = 1)$, while the $(P = 1, Q = 2)$ configuration performs comparably at 98.36%. On the $3 \rightarrow 0$ task, accuracy improves from 92.13% (CNN) to 95.77% (PadéNet), with the Self-ONN variant $(P = 3, Q = 0)$ at 94.13% again occupying an intermediate position. The persistence of the performance ordering, in which the CNN is outperformed by the Self-ONN, which is in turn outperformed by PadéNet, under an alignment objective fundamentally different from that of Deep CORAL provides an early indication that the encoder's contribution is largely independent of the adaptation mechanism.

Table 5. Target-domain accuracy (%) for DANN with PadéNet feature extractors across all cross-load transfer tasks. Best results per task shown in bold.

	Source-Only	**DANN with PadéNet (P, Q)**					
Task	**(1, 0)**	**(1, 0)**	**(1, 1)**	**(1, 2)**	**(2, 0)**	**(2, 1)**	**(3, 0)**
$0 \to 1$	95.43 ± 3.39	96.91 ± 2.42	98.27 ± 1.30	**99.53 ± 0.46**	96.11 ± 1.02	98.94 ± 0.62	97.38 ± 1.61
$0 \to 2$	93.78 ± 4.13	99.49 ± 0.53	**99.92 ± 0.12**	99.66 ± 0.12	99.15 ± 0.40	99.79 ± 0.26	99.24 ± 0.32
$0 \to 3$	80.84 ± 4.57	93.67 ± 5.49	95.49 ± 8.81	**99.54 ± 0.46**	97.13 ± 0.85	99.28 ± 0.44	99.24 ± 0.51
$1 \to 0$	98.40 ± 0.55	98.71 ± 0.51	98.53 ± 1.40	**98.75 ± 0.64**	96.31 ± 1.58	97.73 ± 1.40	96.93 ± 0.81
$1 \to 2$	99.87 ± 0.19	**100.00 ± 0.00**	**100.00 ± 0.00**	99.92 ± 0.12	99.87 ± 0.12	99.83 ± 0.09	99.87 ± 0.12
$1 \to 3$	92.66 ± 3.89	99.28 ± 0.83	99.62 ± 0.41	**99.70 ± 0.24**	98.31 ± 0.91	98.78 ± 0.84	98.78 ± 0.64
$2 \to 0$	95.55 ± 1.35	95.82 ± 1.76	97.46 ± 0.87	97.55 ± 1.04	95.15 ± 1.14	**97.60 ± 0.62**	95.46 ± 1.38
$2 \to 1$	95.94 ± 1.84	97.93 ± 0.94	98.56 ± 0.61	**99.70 ± 0.35**	97.55 ± 1.10	98.22 ± 0.61	98.05 ± 0.23
$2 \to 3$	95.36 ± 4.82	99.45 ± 0.41	99.66 ± 0.46	**99.85 ± 0.19**	98.99 ± 0.35	99.16 ± 0.78	99.16 ± 0.62
$3 \to 0$	74.69 ± 3.44	92.13 ± 5.43	95.77 ± 2.25	93.24 ± 7.93	93.60 ± 1.78	**96.84 ± 1.26**	94.13 ± 2.15
$3 \to 1$	77.34 ± 2.01	95.56 ± 2.82	**98.01 ± 1.02**	94.63 ± 7.52	95.90 ± 0.24	94.04 ± 6.98	95.77 ± 1.09
$3 \to 2$	85.82 ± 3.53	98.94 ± 1.21	**99.79 ± 0.15**	99.66 ± 0.24	98.69 ± 0.57	99.53 ± 0.41	99.11 ± 0.80
Average	90.47	97.32	**98.42**	98.36	97.23	98.31	97.76

Table 6. Target-domain F1-score (%) for DANN with PadéNet feature extractors across all cross-load transfer tasks. Best results per task shown in bold.

	Source-Only	**DANN with PadéNet (P, Q)**					
Task	**(1, 0)**	**(1, 0)**	**(1, 1)**	**(1, 2)**	**(2, 0)**	**(2, 1)**	**(3, 0)**
$0 \to 1$	95.02 ± 3.77	96.77 ± 2.57	98.23 ± 1.36	**99.53 ± 0.47**	95.94 ± 1.09	98.93 ± 0.63	97.28 ± 1.71
$0 \to 2$	92.21 ± 5.50	99.49 ± 0.53	**99.92 ± 0.12**	99.66 ± 0.12	99.15 ± 0.40	99.79 ± 0.26	99.24 ± 0.32
$0 \to 3$	77.02 ± 7.42	92.69 ± 6.93	95.49 ± 8.81	**99.54 ± 0.46**	97.09 ± 0.89	99.28 ± 0.44	99.24 ± 0.51
$1 \to 0$	98.40 ± 0.56	98.71 ± 0.50	98.52 ± 1.43	**98.75 ± 0.65**	96.25 ± 1.63	97.72 ± 1.42	96.95 ± 0.79
$1 \to 2$	99.87 ± 0.19	**100.00 ± 0.00**	**100.00 ± 0.00**	99.92 ± 0.12	99.87 ± 0.12	99.83 ± 0.09	99.87 ± 0.12
$1 \to 3$	92.17 ± 4.44	99.28 ± 0.83	99.62 ± 0.41	**99.70 ± 0.24**	98.31 ± 0.91	98.78 ± 0.84	98.78 ± 0.64
$2 \to 0$	95.52 ± 1.36	95.76 ± 1.85	97.44 ± 0.89	97.53 ± 1.04	95.08 ± 1.17	**97.58 ± 0.62**	95.42 ± 1.38
$2 \to 1$	95.69 ± 1.99	97.87 ± 0.98	**98.53 ± 0.64**	98.44 ± 0.10	97.46 ± 1.17	98.18 ± 0.63	98.00 ± 0.26
$2 \to 3$	95.22 ± 5.08	99.45 ± 0.41	99.62 ± 0.41	**99.70 ± 0.35**	98.99 ± 0.35	99.16 ± 0.78	99.16 ± 0.62
$3 \to 0$	67.96 ± 3.96	91.42 ± 6.33	95.71 ± 2.28	92.72 ± 9.01	93.41 ± 1.82	**96.78 ± 1.32**	93.80 ± 2.58
$3 \to 1$	70.35 ± 2.52	95.22 ± 3.19	**97.94 ± 1.10**	94.03 ± 8.74	95.68 ± 0.26	93.80 ± 7.29	95.47 ± 1.26
$3 \to 2$	83.49 ± 4.39	98.94 ± 1.21	**99.79 ± 0.15**	99.66 ± 0.24	98.68 ± 0.57	99.53 ± 0.41	99.11 ± 0.80
Average	88.58	97.13	**98.40**	98.27	97.16	98.28	97.69

CDAN achieves the best overall performance (Tables 7 and 8). With a CNN backbone it already reaches 98.96%, indicating that conditioning the domain discriminator on the classifier output is a strong adaptation strategy in itself. The highest accuracies in this study are nevertheless obtained with rational encoders: 99.28% for $(P = 1, Q = 2)$ and 99.26% for $(P = 1, Q = 1)$, with all tasks targeting load 2 exceeding 99.9%. The encoder remains relevant on the difficult transfers, where $3 \to 0$ improves from 96.58% to 98.18% with $(P = 1, Q = 1)$ and $3 \to 1$ from 98.22% to 98.82% with $(P = 1, Q = 2)$. These gains are attributed to an interaction between the two components rather than to two additive effects: the multilinear conditioning in CDAN can prevent distinct fault classes from collapsing onto one another only if the features it receives are sufficiently separable, and the rational encoder provides precisely such representations for the class-aware alignment to preserve. Considering that the source-only model attains only 74.69% on $3 \to 0$, the combination of conditional alignment and rational feature extraction effectively resolves the load-shift problem on this benchmark.

Table 7. Target-domain accuracy (%) for CDAN with PadéNet feature extractors across all cross-load transfer tasks. Best results per task shown in bold.

	Source-Only	CDAN with PadéNet (P, Q)					
Task	**(1, 0)**	**(1, 0)**	**(1, 1)**	**(1, 2)**	**(2, 0)**	**(2, 1)**	**(3, 0)**
$0 \to 1$	95.43 ± 3.39	98.77 ± 1.39	99.32 ± 0.55	**99.66 ± 0.28**	97.84 ± 0.80	99.28 ± 0.68	98.18 ± 1.33
$0 \to 2$	93.78 ± 4.13	99.66 ± 0.12	99.87 ± 0.19	**99.92 ± 0.12**	99.53 ± 0.28	99.75 ± 0.23	99.49 ± 0.24
$0 \to 3$	80.84 ± 4.57	**99.79 ± 0.15**	99.75 ± 0.18	99.75 ± 0.23	98.48 ± 1.01	99.54 ± 0.41	99.45 ± 0.38
$1 \to 0$	98.40 ± 0.55	98.80 ± 0.64	**99.07 ± 1.04**	98.84 ± 0.73	97.15 ± 1.49	97.47 ± 1.43	97.91 ± 0.90
$1 \to 2$	99.87 ± 0.19	99.96 ± 0.09	**100.00 ± 0.00**	99.96 ± 0.09	99.83 ± 0.09	99.92 ± 0.12	99.87 ± 0.19
$1 \to 3$	92.66 ± 3.89	99.75 ± 0.09	**99.87 ± 0.12**	99.79 ± 0.21	99.07 ± 0.80	99.32 ± 0.46	99.41 ± 0.23
$2 \to 0$	95.55 ± 1.35	97.78 ± 1.19	98.26 ± 0.95	98.44 ± 0.52	**98.52 ± 0.40**	97.73 ± 0.82	96.57 ± 0.68
$2 \to 1$	95.94 ± 1.84	98.52 ± 0.52	98.69 ± 0.46	98.39 ± 0.46	**99.07 ± 0.74**	98.56 ± 0.35	98.27 ± 0.18
$2 \to 3$	95.36 ± 4.82	99.75 ± 0.09	99.66 ± 0.46	**99.79 ± 0.21**	98.82 ± 0.72	99.37 ± 0.52	99.20 ± 0.71
$3 \to 0$	74.69 ± 3.44	96.58 ± 2.26	**98.18 ± 0.66**	98.09 ± 0.95	96.57 ± 0.60	97.51 ± 0.85	96.89 ± 1.10
$3 \to 1$	77.34 ± 2.01	98.22 ± 0.63	98.52 ± 0.47	**98.82 ± 0.28**	98.01 ± 0.35	98.10 ± 0.50	97.76 ± 0.98
$3 \to 2$	85.82 ± 3.53	99.92 ± 0.12	99.92 ± 0.19	**99.96 ± 0.09**	99.75 ± 0.23	99.87 ± 0.12	99.45 ± 0.61
Average	90.47	98.96	99.26	**99.28**	98.24	98.87	98.54

Table 8. Target-domain F1-score (%) for CDAN with PadéNet feature extractors across all cross-load transfer tasks. Best results per task shown in bold.

	Source-Only	CDAN with PadéNet (P, Q)					
Task	**(1, 0)**	**(1, 0)**	**(1, 1)**	**(1, 2)**	**(2, 0)**	**(2, 1)**	**(3, 0)**
$0 \to 1$	95.02 ± 3.77	98.76 ± 1.41	99.32 ± 0.56	**99.66 ± 0.29**	97.80 ± 0.84	99.27 ± 0.69	98.14 ± 1.38
$0 \to 2$	92.21 ± 5.50	99.66 ± 0.12	99.87 ± 0.19	**99.92 ± 0.12**	99.53 ± 0.28	99.75 ± 0.23	99.49 ± 0.24
$0 \to 3$	77.02 ± 7.42	**99.79 ± 0.15**	99.75 ± 0.18	99.75 ± 0.23	98.47 ± 1.04	99.54 ± 0.41	99.45 ± 0.38
$1 \to 0$	98.40 ± 0.56	98.81 ± 0.64	**99.06 ± 1.04**	98.84 ± 0.73	97.12 ± 1.52	97.44 ± 1.46	97.92 ± 0.88
$1 \to 2$	99.87 ± 0.19	99.96 ± 0.09	**100.00 ± 0.00**	99.96 ± 0.09	99.83 ± 0.09	99.92 ± 0.12	99.87 ± 0.19
$1 \to 3$	92.17 ± 4.44	99.75 ± 0.09	**99.87 ± 0.12**	99.79 ± 0.21	99.07 ± 0.80	99.33 ± 0.46	99.41 ± 0.23
$2 \to 0$	95.52 ± 1.36	97.76 ± 1.20	98.25 ± 0.97	**98.43 ± 0.52**	95.79 ± 0.41	97.72 ± 0.82	96.56 ± 0.69
$2 \to 1$	95.69 ± 1.99	98.49 ± 0.53	**98.66 ± 0.48**	98.35 ± 0.49	97.92 ± 0.77	98.54 ± 0.35	98.23 ± 0.18
$2 \to 3$	95.22 ± 5.08	99.75 ± 0.09	99.66 ± 0.46	**99.79 ± 0.21**	98.82 ± 0.72	99.37 ± 0.52	99.20 ± 0.70
$3 \to 0$	67.96 ± 3.96	96.46 ± 2.39	**98.17 ± 0.67**	98.07 ± 0.97	96.52 ± 0.62	97.49 ± 0.88	96.83 ± 1.17
$3 \to 1$	70.35 ± 2.52	98.18 ± 0.65	98.49 ± 0.50	**98.79 ± 0.30**	97.95 ± 0.38	98.05 ± 0.52	97.67 ± 1.08
$3 \to 2$	83.49 ± 4.39	99.92 ± 0.12	99.92 ± 0.19	**99.96 ± 0.09**	99.75 ± 0.23	99.87 ± 0.12	99.45 ± 0.61
Average	88.58	98.94	99.25	**99.28**	98.21	98.86	98.52

Viewed across Tables 3–8, the results exhibit a consistent structure. The frameworks rank in the order Deep CORAL, DANN, and CDAN, in agreement with the increasing sophistication of their alignment objectives—from matching second-order statistics, to confusing a marginal domain discriminator, to confusing a class-conditioned one. Averaged across all transfer tasks, the PadéNet encoders consistently rank first within each adaptation framework. Although the Self-ONN variants improve upon the CNN baseline under Deep CORAL and on the most difficult transfer scenarios under DANN, they do not consistently outperform the CNN under CDAN, whose class-conditional alignment already alleviates part of the encoder's representational deficiency. By contrast, the rational PadéNet configurations yield superior performance across all three frameworks at the same parameter count as the Self-ONNs. This controlled comparison indicates that the performance gains stem primarily from the rational formulation enabled by the denominator polynomial, rather than from increased polynomial modeling capacity alone.

The t-SNE projections in Figure 12 illustrate both the failure mode and its correction for the $3 \to 0$ task. In the source-only embedding, the fault classes form compact, well-

separated clusters, yet each cluster is divided by domain. The classifier has therefore learned the fault categories, but in a load-specific representation, confirming that distribution shift rather than class confusability is the obstacle to generalization. Deep CORAL narrows this division but leaves residual misalignment in the load-sensitive ball-defect categories. DANN merges the domains more thoroughly at the cost of mild cluster stretching, as the feature extractor sacrifices some class compactness to confuse the discriminator. CDAN avoids this trade-off entirely. Its clusters remain compact while the source and target distributions overlap almost completely, which is consistent with the intended effect of conditioning the discriminator on the classifier output.

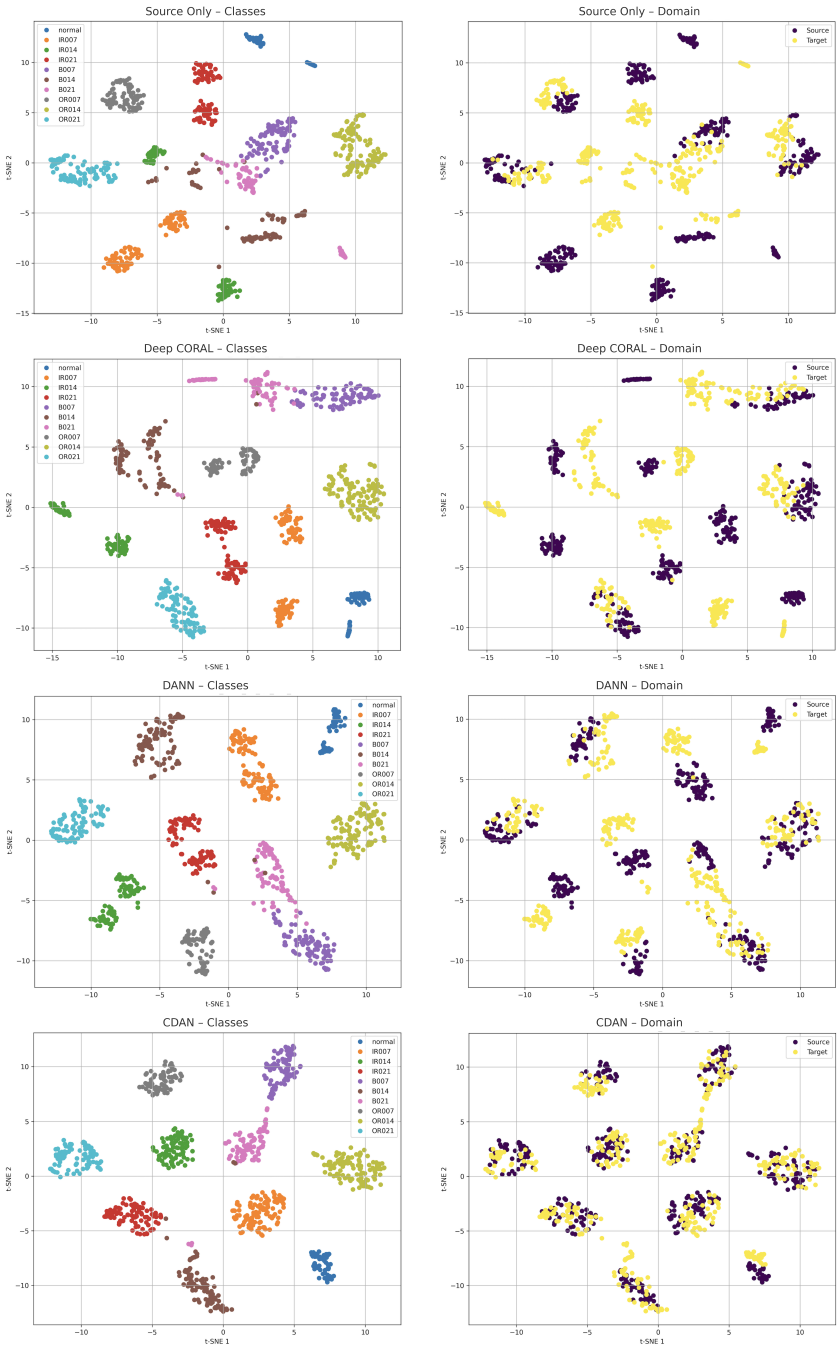

Figure 12. t-SNE visualization of learned feature embeddings for the $3 \rightarrow 0$ transfer task. The left column shows class-wise distributions, while the right column illustrates source–target domain alignment. From top to bottom, the rows correspond to the source-only baseline with 1D CNN, Deep CORAL, DANN, and CDAN, where all domain adaptation methods employ PadéNet ($P = 1, Q = 1$) as the feature extractor.

The confusion matrices provide the corresponding class-level evidence. Without adaptation (Figure 13), the errors are systematic rather than diffuse. IR014 samples are predominantly misclassified as OR021, and the ball-defect severities are confused with one another, indicating that the load shift primarily disrupts the discrimination of defect location and severity. Deep CORAL corrects the most severe of these errors, DANN further strengthens the diagonal dominance, and under CDAN the residual ball-severity confusion is reduced to a negligible level. Figure 14 addresses the complementary question of what the encoder alone contributes within each fixed framework. In all three cases, replacing the CNN with the PadéNet $(P = 1, Q = 1)$ encoder visibly reduces the off-diagonal entries, most markedly under Deep CORAL, where the CNN's confusion among IR014, B014, and B021 is largely eliminated. The benefit of the rational encoder is thus independent of the alignment mechanism it serves.

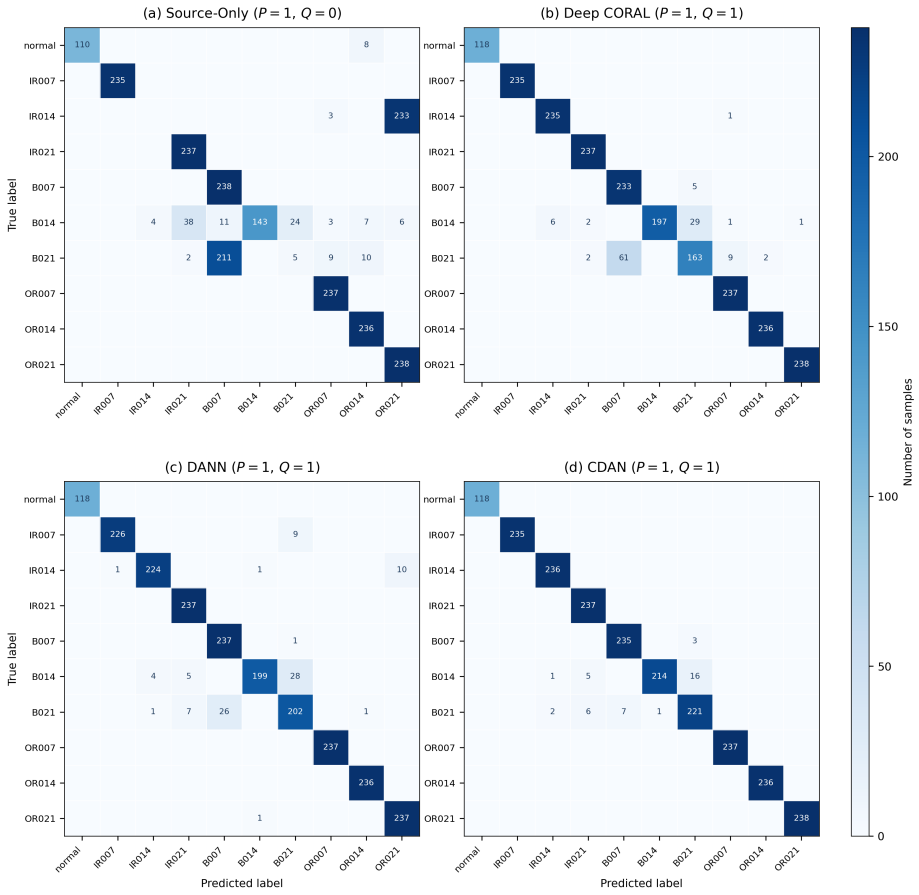

Figure 13. Aggregated confusion matrices for the $3 \rightarrow 0$ transfer task, comparing the source-only CNN baseline, Deep CORAL, DANN, and CDAN, where all domain adaptation methods employ PadéNet $(P = 1, Q = 1)$ as the feature extractor.

These accuracy gains are achieved at a modest computational cost. As reported in Table 9, the parameter count scales linearly with the total Padé order, because each additional polynomial term introduces a single convolutional branch while the classifier and discriminator heads remain unchanged. Under Deep CORAL, the model size grows from 43.63 K parameters for the baseline CNN to 117.74 K at $(P = 1, Q = 2)$, whereas CDAN, whose multilinear conditioning layer raises its baseline to 86.76 K, reaches 160.88 K in the largest configuration. Training times remain below 1.32 s per epoch on an NVIDIA RTX 3070 across all settings. Deep CORAL is marginally the slowest owing to its batch-level covariance computation, while the adversarial methods incur only the overhead of a lightweight binary domain classifier.

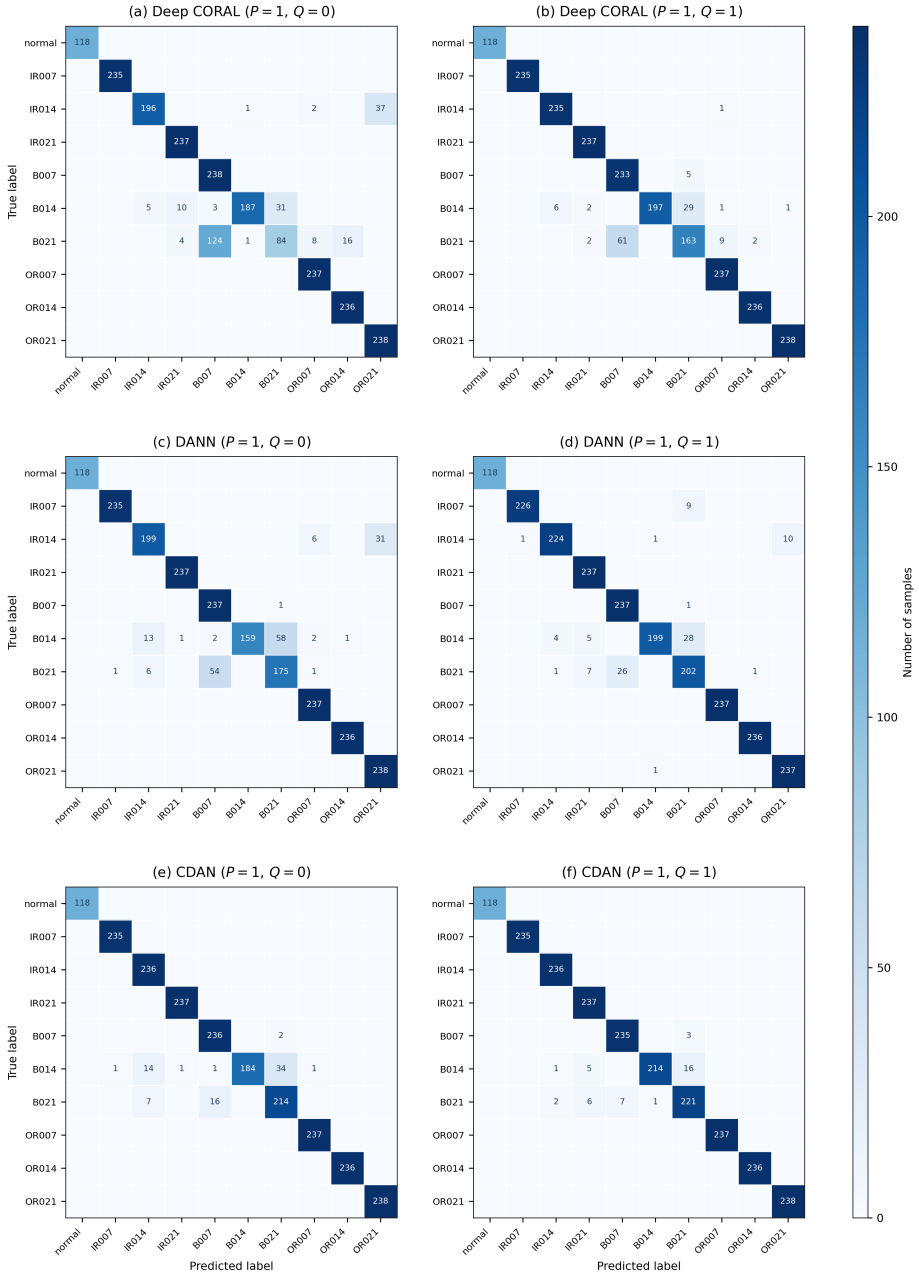

Figure 14. Impact of feature extractor architecture on target-domain classification for the $3 \rightarrow 0$ task. Confusion matrices compare CNN $(P = 1, Q = 0)$ versus PadéNet $(P = 1, Q = 1)$ within each UDA framework: Deep CORAL, DANN, and CDAN.

Table 9. Trainable parameter counts and per-epoch training duration for Deep CORAL, DANN, and CDAN with shared PadéNet encoders. All experiments were conducted on an NVIDIA RTX 3070 GPU.

Padé (P, Q)	Deep CORAL		DANN		CDAN	
	Params (K)	Time (s)	Params (K)	Time (s)	Params (K)	Time (s)
(1, 0)	43.63	0.69	49.90	0.58	86.76	0.58
(1, 1)	80.68	1.05	86.95	0.86	123.82	0.87
(1, 2)	117.74	1.32	124.01	1.08	160.88	1.08
(2, 0)	80.68	0.90	86.95	0.73	123.82	0.73
(2, 1)	117.74	1.25	124.01	1.02	160.88	1.02
(3, 0)	117.74	1.08	124.01	0.88	160.88	0.88

At deployment, only the feature extractor and label predictor are retained after adaptation (Table 10). The deployed models occupy between 170.4 and 459.9 KB of memory and comprise between 43.63 and 117.74 K parameters across the Padé orders. Inference latency on a Raspberry Pi 4, measured with the corresponding TFLite models as the average over 100 runs following warmup, ranges from 8.11 to 19.67 ms. Latency scales with the Padé order because the parallel convolutional branches, evaluated concurrently on a GPU, are processed sequentially on the CPU. Nevertheless, since condition-monitoring decisions are typically made on the order of seconds, even the largest configuration leaves a wide operational margin, confirming the suitability of PadéNet-based encoders for deployment on resource-constrained edge platforms.

Table 10. Inference-time computational profile of deployed models following domain-adaptive training. Measurements reflect feature extractor and label predictor only (domain classifiers excluded).

Padé (P, Q)	Params (K)	Memory (KB)	Latency (ms)
(1, 0)	43.63	170.4	8.11
(1, 1)	80.68	315.2	13.46
(1, 2)	117.74	459.9	19.67
(2, 0)	80.68	315.2	12.03
(2, 1)	117.74	459.9	18.08
(3, 0)	117.74	459.9	15.58

Table 11 situates the proposed framework relative to established domain adaptation methods on the CWRU benchmark. A direct comparison nevertheless requires caution, as the experimental configurations differ across studies. The DANN, MMD, and AdaBN results of Wang et al. [58] were obtained with a three-layer CNN backbone operating on 512-dimensional FFT spectra, WDCNN [5] is a non-adaptive supervised model trained on 2048-sample raw windows, and the proposed framework processes raw 1024-sample time-domain windows directly. Consequently, the reported values should be interpreted as indicative of overall trends and efficiency–accuracy trade-offs rather than as the outcome of a controlled head-to-head evaluation.

Within these constraints, several trends can be observed. In terms of classification accuracy, CDAN–PadéNet $(P = 1, Q = 2)$ achieves 99.28%, which is within a sub-percent margin of MMD (99.42%) and higher than DANN (99.07%). The benefit of explicit adaptation is most evident on the more difficult transfer tasks. For the $3 \rightarrow 1$ transfer, the non-adaptive WDCNN reaches only 78.10%, whereas the proposed method retains 98.82%; likewise, for $3 \rightarrow 0$, AdaBN achieves 89.27% compared with 98.09% for the proposed method. The proposed framework also shows good stability across tasks, with fewer than two percentage points separating its best and worst results (98.09–99.96%), in contrast to AdaBN, which spans more than ten percentage points (89.27–100.00%). The most important difference, however, concerns model size. The complete CDAN–PadéNet model contains 160.88 K parameters, approximately 8.6× fewer than MMD (1379.99 K) and 16.8× fewer than DANN (2694.82 K). AdaBN (1380.57 K) performs worse than all adversarial methods in accuracy despite its comparable size. In addition, the proposed framework avoids the quadratic kernel estimation cost associated with MMD training reported in [58]. This parameter efficiency involves a corresponding trade-off in inference latency. Since the proposed model processes raw signals of twice the FFT dimension together with the rational function evaluations, its inference time increases to 19.67 ms, approximately an order of magnitude higher than the FFT-based methods (∼4.0 ms) and WDCNN (0.57 ms). This comparison is, however, partly favourable to the reference methods, whose FFT preprocessing is not included in their reported timings. For condition-monitoring deployment,

model storage on embedded hardware is often a more limiting constraint than throughput, and in such cases the reduction in parameter count may be the more relevant factor.

Table 11. Comparative evaluation of the proposed CDAN with PadéNet ($P = 1$, $Q = 2$) framework against established domain adaptation methods. Results for DANN, MMD, and AdaBN are as reported in Wang et al. [58]; WDCNN results from Zhang et al. [5]. Dash (–) indicates unreported values.

Task	CDAN + PadéNet	DANN [58]	MMD [58]	AdaBN [58]	WDCNN [5]
$0 \rightarrow 1$	99.66	98.76	99.38	98.87	–
$0 \rightarrow 2$	99.92	99.96	99.98	99.30	–
$0 \rightarrow 3$	99.75	99.81	100.00	99.75	–
$1 \rightarrow 0$	98.84	98.73	99.31	98.83	–
$1 \rightarrow 2$	99.96	99.96	99.98	99.95	99.20
$1 \rightarrow 3$	99.79	99.65	99.97	99.82	91.00
$2 \rightarrow 0$	98.44	97.70	98.61	95.89	–
$2 \rightarrow 1$	98.39	98.40	98.52	97.83	95.10
$2 \rightarrow 3$	99.79	99.82	100.00	100.00	91.50
$3 \rightarrow 0$	98.09	97.62	98.72	89.27	–
$3 \rightarrow 1$	98.82	98.41	98.53	94.42	78.10
$3 \rightarrow 2$	99.96	99.98	100.00	99.95	85.10
Average Acc. (%)	99.28	99.07	99.42	97.82	90.00
Total Params. (K)	160.88	2694.82	1379.99	1380.57	53.83
Inference Speed (ms)	19.67	3.99	3.99	4.14	0.57

In summary, the CWRU experiments demonstrate that class-conditional adversarial alignment is the most effective of the three adaptation strategies, that rational PadéNet encoders improve all frameworks over CNN and Self-ONN baselines, and that the resulting CDAN–PadéNet models combine accuracy on par with substantially larger published methods with a parameter budget nearly an order of magnitude smaller. The following section examines whether these advantages extend to a more demanding and realistic benchmark.

4.2. Case Study II: Unsupervised Domain Adaptation on Paderborn Bearing Dataset

To assess whether the advantages of PadéNet-based feature extraction extend beyond a single benchmark, a second case study is conducted on the Paderborn University (PU) bearing dataset [19], a widely used benchmark for condition monitoring of rolling-element bearings.

4.2.1. Paderborn Bearing Dataset

The Paderborn bearing dataset was acquired on an electromechanical drive-train test rig in which a 6203-type ball bearing is subjected to controlled operating conditions, with vibration of the bearing housing recorded at a sampling rate of 64 kHz over measurement intervals of approximately four seconds [19]. In contrast to the CWRU dataset, where domain shift is introduced primarily through load variation, the Paderborn benchmark is designed to capture changes arising from multiple operating conditions. Following [20], this study considers three domains recorded at a fixed rotational speed of 1500 rpm, thereby excluding the additional 900 rpm condition available in the full dataset. As a result, the induced domain shift is driven solely by variations in load torque and radial force, without the confounding influence of speed changes. Compared with CWRU, these simultaneous variations create a more heterogeneous transfer scenario, providing a complementary and more challenging benchmark for evaluating cross-domain generalization.

Following common practice for label-consistent domain adaptation on this dataset, the bearings are grouped into three health states: healthy, inner-race fault, and outer-race

fault. To reflect a realistic diagnostic scenario, following [20], the fault classes are drawn from bearings with damage produced by accelerated lifetime tests rather than artificial machining, so that the learned representations must capture naturally developed defect signatures. Figure 15 shows examples of accelerated-lifetime damage from the Paderborn University dataset, including an indentation on the outer-ring raceway and minor pitting on the inner-ring raceway [19]. The bearing codes used for each class are summarized in Table 12.

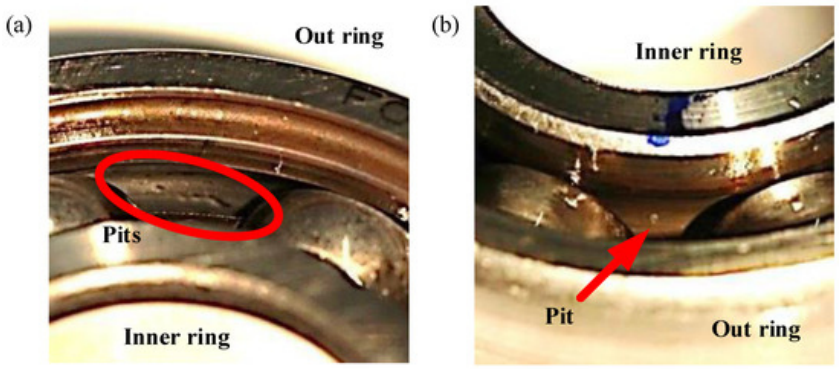

Figure 15. Real damage induced by accelerated-lifetime tests: (**a**) indentation on the outer-ring raceway; and (**b**) minor pitting on the inner-ring raceway.

Table 12. Bearing health-state taxonomy and bearing codes used for the Paderborn case study (real, accelerated-lifetime damage).

Health State	Class Label	Bearing Codes
Healthy	0	K001, K002, K003, K004, K005
Inner-race fault	1	KI04, KI14, KI16, KI18, KI21
Outer-race fault	2	KA04, KA15, KA16, KA22, KA30

Three operating conditions are used to define the domains, each holding the rotational speed fixed at 1500 rpm while varying the load torque and radial force, as listed in Table 13. These conditions isolate the effect of mechanical loading on the vibration signature and yield six directional cross-condition transfer tasks among the three domains.

Table 13. Sample distribution and operating conditions for the Paderborn domains.

Domain	Split	Samples per Class			Working Conditions		
		Normal	Inner Race	Outer Race	Speed (rpm)	Torque (Nm)	Force (N)
PA	Train	4000	4000	4000	1500	0.1	1000
	Test	1000	1000	1000			
PB	Train	4000	4000	4000	1500	0.7	400
	Test	1000	1000	1000			
PC	Train	4000	4000	4000	1500	0.7	1000
	Test	1000	1000	1000			

4.2.2. Preprocessing Pipeline and Data Split for Paderborn Bearing Dataset

Because Paderborn signals are acquired at a substantially higher sampling rate than CWRU and contain broadband structure, a frequency-domain representation is adopted for this case study. We follow a protocol similar to that of [20]. Given the 64 kHz sampling rate and the 1500 rpm rotational speed, each vibration recording is divided via a sliding-window mechanism into windows of 5120 samples, a length chosen to cover two full cycles

of rotation. For every window, the single-sided magnitude spectrum is obtained via the real-valued fast Fourier transform, retaining the first half of the spectrum; the DC component is discarded, yielding a 2560-dimensional spectral vector. Logarithmic compression is then applied to reduce the dynamic range of dominant spectral peaks, and each spectrum is standardized to zero mean and unit variance.

The source and target domains were each split into 80% training and 20% test sets using class-stratified sampling. The source training set and unlabeled target training set were used for adaptation, while the test sets were reserved for evaluation. Each (P, Q) configuration was repeated five times with different random initializations, and the target-domain accuracy is reported as mean $\pm$ std. For each operating condition, 4000 training samples and 1000 test samples were selected per class, resulting in a balanced class and domain distribution, following [20]. The resulting spectra are used directly as one-dimensional inputs to the proposed PadéNet-CDAN.

4.2.3. Proposed Network and Training Configuration for the Paderborn Bearing Dataset

The PadéNet feature extractor comprises four sequential Padé layers of orders (P, Q), each with 32 filters, a kernel size of 5, a stride of 4, and a hyperbolic-tangent activation. The Padé stack is followed by flattening, layer normalization, and dropout (rate 0.5), which together yield the latent embedding shared by the label predictor and the conditional domain discriminator. The label predictor consists of two hidden dense layers (64 and 32 units, tanh activation) and a softmax output over the three bearing-health classes (healthy, inner-race fault, and outer-race fault). The domain discriminator mirrors this structure with a sigmoid output but, following the CDAN formulation, operates on the conditioned representation rather than on the raw features, obtained by combining the latent features and the class-probability predictions through their multilinear map (outer product). The complete architecture is illustrated in Figure 16.

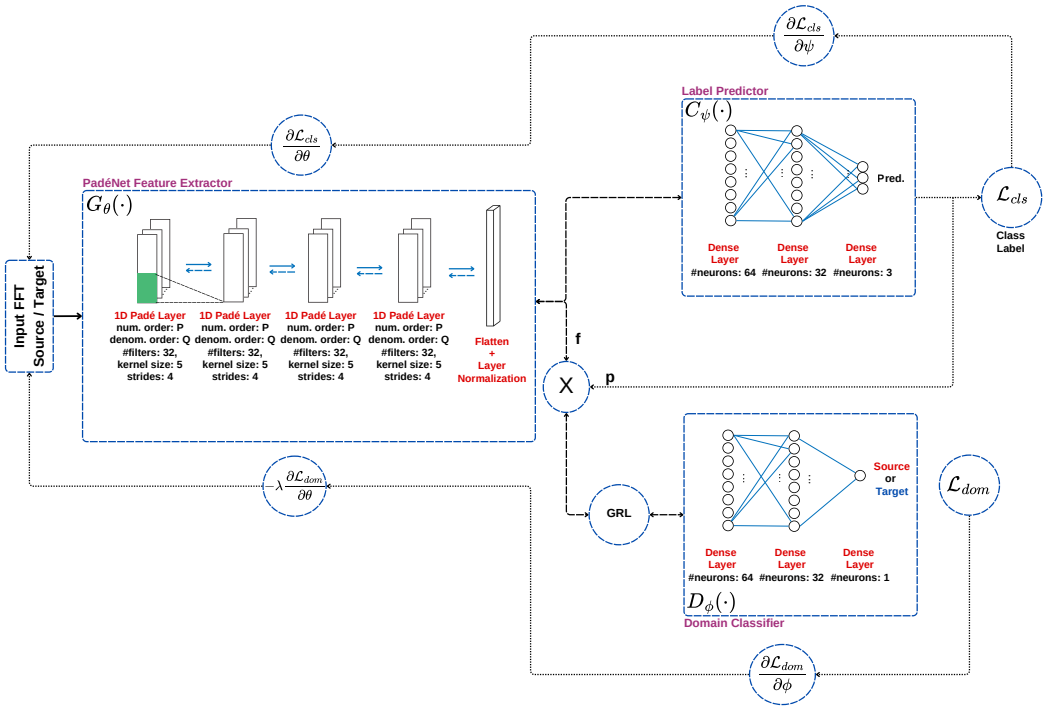

Figure 16. Proposed CDAN architecture for the Paderborn bearing dataset.

Conditional adversarial alignment is realized through a gradient reversal layer whose coefficient λ follows the schedule in Equation (19), allowing the discriminator's influence to grow progressively as training advances. Each model is trained for 100 epochs with a batch

size of 256 using the AdamW optimizer, an inverse-decay learning-rate schedule (initial rate 0.001, decay parameters $\alpha = 10$, $\beta = 0.75$), and a weight-decay coefficient of 0.004. The feature extractor, label predictor, and domain discriminator are updated by separate optimizers. To stabilize the adversarial dynamics, a reduced first-moment momentum ($\beta_1 = 0.5$) is applied to the feature extractor and the discriminator, while the label predictor retains the standard value ($\beta_1 = 0.9$).

Each (P, Q) configuration of the feature extractor is assessed over five independent runs differing only in random weight initialization, with target-domain performance reported as mean $\pm$ standard deviation. As in the CWRU study, the held-out target test set is excluded from adaptation, and the accuracy, precision, recall, and weighted F1-score described in Section 4.1.3 are used unchanged.

4.2.4. Results on the Paderborn Bearing Dataset

This section evaluates the proposed CDAN with 1D PadéNet framework on the six cross-condition transfer tasks defined over the three Paderborn operating domains PA, PB, and PC. Following the protocol of [20], the held-out target test set is excluded from adaptation, and each configuration is evaluated over five independent runs, with results reported as mean $\pm$ standard deviation. The encoder family in Table 14 again spans all three neuron types, where $(1, 0)$ corresponds to a standard CNN, $(2, 0)$ and $(3, 0)$ to Self-ONNs, and the configurations with $Q \geq 1$ are genuinely rational and unique to PadéNet (Section 3.1).

Table 14. Target-domain accuracy (%) for CDAN with PadéNet feature extractors on the Paderborn dataset across all six cross-condition transfer tasks. Best results per task shown in bold.

	Source-Only	**CDAN with PadéNet (P, Q)**					
Task	**(1, 0)**	**(1, 0)**	**(1, 1)**	**(1, 2)**	**(2, 0)**	**(2, 1)**	**(3, 0)**
PA $\rightarrow$ PB	93.02 $\pm$ 1.40	98.57 $\pm$ 0.32	98.66 $\pm$ 0.46	98.70 $\pm$ 0.29	99.06 $\pm$ 0.54	**99.72 $\pm$ 0.21**	99.21 $\pm$ 0.90
PA $\rightarrow$ PC	99.66 $\pm$ 0.16	99.75 $\pm$ 0.10	99.91 $\pm$ 0.05	99.92 $\pm$ 0.06	**100.00 $\pm$ 0.00**	99.97 $\pm$ 0.00	99.96 $\pm$ 0.04
PB $\rightarrow$ PA	94.08 $\pm$ 0.58	98.88 $\pm$ 0.22	99.82 $\pm$ 0.09	99.76 $\pm$ 0.08	99.83 $\pm$ 0.14	**99.92 $\pm$ 0.04**	99.90 $\pm$ 0.11
PB $\rightarrow$ PC	94.98 $\pm$ 0.51	99.07 $\pm$ 0.23	99.61 $\pm$ 0.08	99.64 $\pm$ 0.10	99.88 $\pm$ 0.07	99.87 $\pm$ 0.08	**99.89 $\pm$ 0.05**
PC $\rightarrow$ PA	99.86 $\pm$ 0.06	99.85 $\pm$ 0.03	99.93 $\pm$ 0.01	99.94 $\pm$ 0.04	**99.99 $\pm$ 0.02**	99.91 $\pm$ 0.09	**99.99 $\pm$ 0.01**
PC $\rightarrow$ PB	94.54 $\pm$ 0.92	98.72 $\pm$ 0.23	98.63 $\pm$ 0.11	98.84 $\pm$ 0.15	99.47 $\pm$ 0.34	**99.67 $\pm$ 0.24**	99.24 $\pm$ 0.76
Average Acc. (%)	96.02	99.14	99.43	99.47	99.70	**99.84**	99.70
Total Params. (K)	36.90	94.37	109.89	125.41	110.02	125.54	125.67

The source-only baseline indicates that the difficulty of this benchmark is concentrated in specific transfer directions. Transfers between the high-load domains are nearly saturated, with PA $\rightarrow$ PC and PC $\rightarrow$ PA both exceeding 99.6%, whereas every task directed into the reduced-force domain PB falls in the 93–95% range. This asymmetric, rather than uniform, degradation is consistent with the structured nature of the shift induced by the simultaneous change in load torque and radial force, which alters the vibration distribution beyond what a model trained on a single condition can absorb.

Conditional adversarial alignment improves every task, and the pattern across the encoder family is unambiguous. The CNN configuration $(1, 0)$ raises the average from 96.02% to 99.14%, recovering most of the source-only gap. The Self-ONN configurations improve on this further, with $(2, 0)$ reaching 99.70%, which indicates that the additional polynomial flexibility aids the representation of the broadband, multi-resonant Paderborn spectra. The best result, however, is obtained by the rational model $(2, 1)$, which attains 99.84% on average and ranks first on three of the six tasks, including the difficult PB-directed transfers; on PA $\rightarrow$ PB, the largest single improvement, accuracy rises from 93.02% to 99.72%. Since the rational configurations operate at a parameter budget comparable to

the Self-ONNs, this advantage is attributable to the denominator polynomial itself, which captures spectral structure that neither the linear nor the purely polynomial operators represent adequately and thereby provides the representational margin required on the hardest domain shifts.

The confusion matrices in Figure 17 provide the corresponding class-level evidence for the PA→PB task. Without adaptation, the errors arise predominantly from confusion between the two fault types and, to a lesser extent, between the outer-race fault and the healthy state. CDAN with the 1D CNN removes much of this off-diagonal mass, while the $(2, 1)$ model yields a nearly diagonal structure in which the residual inner- versus outer-race and outer-race versus healthy confusions are reduced to a negligible level, even under the largest distribution shift in the study. In condition monitoring, the most critical errors are those in which a faulty bearing is classified as healthy, since such missed detections allow an incipient fault to develop undetected, whereas a healthy bearing classified as faulty only leads to an unnecessary inspection. As shown in Figure 17, the proposed encoder markedly reduces this type of error on the most challenging transfer task (PA → PB). The number of faulty samples misclassified as healthy decreases from 288 out of 10,000 for the source-only model (88 inner-race and 200 outer-race) to 26 for CDAN with the $(P = 2, Q = 1)$ encoder (0 inner-race and 26 outer-race), corresponding to a missed detection rate of 0.26%. In addition, since the label predictor provides class-probability outputs, this rate can be further reduced at deployment without retraining by adopting a fault-oriented decision rule, in which a higher confidence is required to classify a bearing as healthy and low-confidence samples are flagged for inspection. Such a strategy is well suited to practical condition monitoring, where the cost of an undetected failure is considerably higher than that of an occasional false alarm.

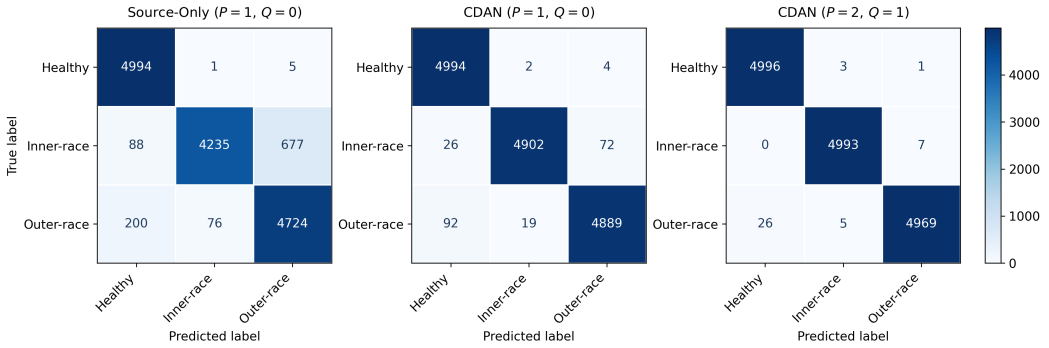

Figure 17. Aggregated target-domain confusion matrices over five runs on the Paderborn benchmark for the PA → PB transfer task, comparing source-only, CDAN $(P = 1, Q = 0)$, and CDAN $(P = 2, Q = 1)$.

Vibration signals acquired in industrial environments are inevitably contaminated by sensor and electrical noise, and a diagnostic model intended for field deployment may therefore degrade gracefully as signal quality deteriorates. To characterize this behavior, the trained CDAN models were evaluated on the held-out target test set of the PA → PB task, which constitutes the most severe distribution shift in the present study, under additive white Gaussian noise (AWGN) applied at signal-to-noise ratios (SNRs) ranging from −4 to 10 dB. The noise was injected into the raw time-domain window prior to the fast Fourier transform, since SNR is defined in the time domain. Noise was added exclusively at evaluation time; no corrupted samples were presented during adaptation, so that the analysis reflects the intrinsic robustness of the learned representation rather than a denoising effect arising from noise augmentation. The subsampling and train–test partitioning seeds were fixed and held independent of the noise level, ensuring that the held-out test set comprised identical samples across all SNRs and that only the additive

noise varied between conditions. As in the preceding experiments, each configuration was evaluated over five independent runs, and the target-domain accuracy is reported as the mean across runs.

The results are presented in Figure 18, which compares the standard CNN ($P = 1$, $Q = 0$), the two Self-ONN encoders ($P = 2$, $Q = 0$ and $P = 3$, $Q = 0$), and the three rational PadéNet encoders across the full range of noise levels. The PadéNet ($P = 2$, $Q = 1$) encoder, previously identified as the optimal configuration on clean signals in Table 14, retains the highest accuracy at every SNR, and its margin over the convolutional baseline widens as the noise intensifies. At 0 dB, where signal and noise powers are equal, this encoder obtains 75.97% accuracy against 62.21% for the CNN, a difference of nearly fourteen percentage points, and it maintains its lead even under the most severe -4 dB condition, reaching 44.60% compared with 37.97% for the CNN. Although the ($P = 1$, $Q = 1$) and ($P = 1$, $Q = 2$) encoders surpass the CNN on clean signals and at high SNRs, their first-order numerators leave them vulnerable under severe noise, where they fall to or below the convolutional baseline in the low-SNR regime. The ($P = 2$, $Q = 1$) configuration, by contrast, exceeds the CNN at every noise level, indicating that realizing the robustness benefit of the rational formulation requires careful joint selection of the numerator and denominator orders. Beyond approximately 6 dB, all encoders converge toward saturation as the fault signatures come to dominate the noise floor, and at the highest SNRs the accuracies approach the clean-signal values reported in Table 14. These findings confirm that the accuracy advantage of the proposed PadéNet ($P = 2$, $Q = 1$) encoder is not restricted to clean laboratory recordings but is preserved, and in relative terms amplified, under substantial measurement noise. This property is particularly relevant for condition monitoring in noisy industrial settings, where the robustness of the learned representation directly determines the reliability of the diagnostic decision.

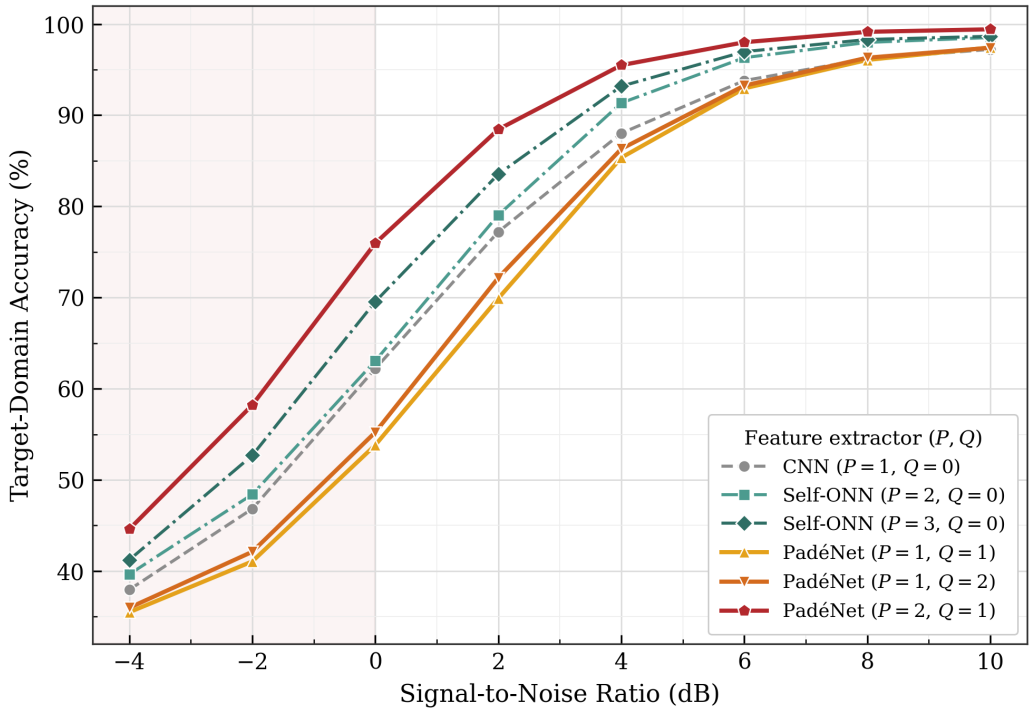

Figure 18. Noise robustness comparison for the PA → PB transfer task on the Paderborn benchmark. Target-domain accuracy is shown as a function of the additive white Gaussian noise signal-to-noise ratio for the CNN, Self-ONN, and PadéNet feature extractors within the CDAN framework.

Table 15 positions CDAN with PadéNet $(2, 1)$ against established domain adaptation methods, with all baseline figures as reported by Wang and Liu [20]. The baselines cover the principal families of the domain adaptation literature, including classical subspace alignment (TCA), a non-adapted source-trained CNN, moment-matching approaches (Deep CORAL, DDC, and DAN), and adversarial frameworks (DCTLN and WDGRL). The strongest among them is TLADA [20], which augments Wasserstein-based data-level alignment with a triplet loss enforcing class-level alignment. The comparison is well controlled on the baseline side, since all deep methods in [20] share an identical four-block 1D convolutional feature extractor with matching classifier and domain-critic heads, and operate on the same input representation used here, namely a 2560-dimensional frequency-domain vector obtained by applying an FFT to a 5120-point vibration frame and retaining the first half of the spectrum. Full architectural details are given in [20].

With this common input, the proposed method achieves the highest average accuracy at 99.84%, ahead of TLADA at 99.46% and clearly above the remaining methods, which trail by roughly two to seven percentage points. The improvement is concentrated on the transfers that were hardest for the source-only model. On PA $\rightarrow$ PB and PB $\rightarrow$ PA, the proposed method is the best among all compared approaches, at 99.72% and 99.92%, respectively, while on the easier transfers it remains within a fraction of a percentage point of the best baseline. The gains on the difficult directions are therefore not obtained at the expense of the easy ones.

Table 15. Comparative evaluation of the proposed CDAN with PadéNet (P = 2, Q = 1) framework against established domain adaptation methods on the three-class Paderborn benchmark. Results for all other methods are as reported in Wang and Liu [20]. Best result per task in bold.

Task	CDAN + PadéNet (P = 2, Q = 1)	TCA	CNN	Deep CORAL	DAN	DDC	DCTLN	WDGRL	TLADA
PA $\rightarrow$ PB	**99.72**	87.27	90.93	92.23	91.70	90.83	96.17	96.33	99.00
PA $\rightarrow$ PC	99.97	99.87	99.73	99.54	99.33	99.97	99.84	99.97	**100.00**
PB $\rightarrow$ PA	**99.92**	92.99	88.73	92.20	93.03	98.13	99.87	98.80	99.17
PB $\rightarrow$ PC	99.87	92.53	84.20	95.10	91.03	97.23	99.75	97.90	**99.97**
PC $\rightarrow$ PA	99.91	99.80	99.36	99.60	97.37	99.83	99.91	99.80	**99.93**
PC $\rightarrow$ PB	**99.67**	89.71	92.80	92.93	95.00	93.37	91.32	93.23	98.67
Average Acc. (%)	**99.84**	93.70	92.63	95.27	94.58	96.56	97.81	97.67	99.46
Total Params. (K)	125.54 K	155.56	155.56	155.56	155.56	155.56	300.56	300.56	300.56
Inference Speed (ms)	3.11	1.27	1.27	1.27	1.27	1.27	1.27	1.27	1.27

These results are obtained with a substantially leaner model. As reported in the lower rows of Table 15, the proposed network contains 125.54 K trainable parameters, compared with 155.56 K for the CNN-based baselines (TCA, CNN, Deep CORAL, DAN, and DDC) and 300.56 K for DCTLN, WDGRL, and TLADA. The highest overall accuracy is therefore achieved with roughly 19% fewer parameters than the most compact competing methods and less than half the parameter budget of the strongest baselines. This constitutes the central observation of the Paderborn study, namely that the accuracy improvement is driven by the expressiveness of the rational Padé formulation rather than by additional model capacity.

The increased expressiveness incurs only a modest computational cost. Measured on a Raspberry Pi 4 over 100 inference runs following warm-up, the proposed model averages 3.11 ms per inference, against 1.27 ms for the competing methods. Because all deep-learning baselines in [20] share the same four-block 1D CNN encoder and classifier architecture, their deployed inference graphs are identical. The reported 1.27 ms latency therefore reflects the performance of this common backbone, measured on the same Raspberry Pi 4 under an

identical evaluation protocol. Since bearing condition monitoring requires decisions on the scale of seconds rather than milliseconds, this difference imposes no practical constraint on real-time deployment, and the favourable accuracy–efficiency trade-off established on CWRU is preserved on resource-limited edge hardware.

The reduced parameter count reported above translates directly into deployability on embedded hardware, and we summarize the evidence separately for each benchmark. On the CWRU benchmark, the proposed CDAN–PadéNet model requires 160.88 K total trainable parameters, roughly 8.6× and 16.8× fewer than MMD and DANN, respectively (Table 11). On the Paderborn benchmark, it requires 125.54 K total trainable parameters, less than half the budget of the strongest baselines and about 19% fewer than the most compact competing methods (Table 15). At deployment, the domain discriminator is discarded and only the feature extractor and label predictor are retained. For the CWRU benchmark, the best-performing Padé configuration ($P = 1$, $Q = 2$) occupies 117.74 K parameters and 459.9 KB of memory at an inference latency of 19.67 ms on a Raspberry Pi 4 (Table 10); for the Paderborn model, inference on the same device takes 3.11 ms. Both benchmarks thus combine the accuracy gains of the rational PadéNet formulation with a footprint well within the limits of resource-constrained embedded platforms.

Considering that the Paderborn faults stem from real accelerated-lifetime damage under simultaneous variation of load torque and radial force, these findings, together with the CWRU results, confirm that the benefits of PadéNet-based feature extraction extend beyond a single benchmark to a more demanding and realistic operating setting.

5. Conclusions

This study introduced PadéNet feature extractors into three established UDA frameworks—Deep CORAL, DANN, and CDAN—for bearing fault diagnosis under varying operating conditions, requiring no labeled target-domain data. On the CWRU benchmark, CDAN with PadéNet ($P = 1$, $Q = 2$) achieves 99.28% average target-domain accuracy across all 12 cross-load transfer tasks with only 160.88 K trainable parameters. On the more demanding Paderborn benchmark, where domain shift arises from simultaneous variation of load torque and radial force on bearings with real accelerated-lifetime damage, CDAN with PadéNet ($P = 2$, $Q = 1$) attains 99.84% across six cross-condition tasks, surpassing the strongest reported baseline (TLADA) with approximately 2.4× fewer parameters (125.54 K).

Across both case studies, PadéNet encoders consistently outperformed CNN and Self-ONN backbones within every adaptation framework; since Self-ONN variants of identical parameter count did not reach comparable accuracy, these gains are attributable to the rational-function formulation rather than to model capacity. The t-SNE projections and confusion matrices further showed that the rational representations yield tighter class clusters with stronger cross-domain overlap, benefiting fault discrimination and domain alignment simultaneously. The associated computational overhead remains modest in both training and inference, supporting deployment on resource-constrained embedded platforms. Finally, the consistency of the improvements across artificially seeded and naturally developed defects, under two distinct types of domain shift, indicates that the benefits of rational-function feature extraction are not tied to a particular benchmark or fault-generation mechanism.

Both benchmarks, however, originate from controlled laboratory test rigs under a closed-set assumption and do not capture the full complexity of industrial operation. Future work will therefore extend the framework to industrial-scale data with non-stationary noise and variable rotational speed, and to open-set adaptation scenarios in which the fault categories differ between source and target domains.

Author Contributions: Conceptualization, S.K. and L.E.; methodology, S.K., L.E., T.I. and M.A.; software, S.K.; validation, S.K.; writing—original draft preparation, S.K., L.E. and C.C.; writing—review and editing, S.K., L.E., T.I., M.A. and C.C.; supervision, L.E., T.I. and M.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The raw data supporting this study are openly available in the Case Western Reserve University Bearing Data Center at https://engineering.case.edu/bearingdatacenter/download-data-file (accessed on 5 January 2026) and in the Paderborn University Bearing Data Center at https://mb.uni-paderborn.de/kat/forschung/bearing-datacenter/data-sets-and-download (accessed on 5 January 2026).

Acknowledgments: Sertac Kilickaya acknowledges the financial support provided by The Scientific and Technological Research Council of Turkey (TÜBİTAK) through the 2211-E National PhD Scholarship Program.

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

Abbreviations

The following abbreviations are used in this manuscript:

UDA	Unsupervised Domain Adaptation
CNN	Convolutional Neural Network
Self-ONN	Self-Organized Operational Neural Network
PadéNet	Padé Approximant Neural Network
DANN	Domain-Adversarial Neural Network
CDAN	Conditional Domain-Adversarial Network
Deep CORAL	Deep Correlation Alignment
PAON	Padé Neuron
PAU	Padé Activation Unit
ONN	Operational Neural Network
DA	Domain Adaptation
WDCNN	Wide-Kernel Deep Convolutional Neural Network
GAN	Generative Adversarial Network
AdaBN	Adaptive Batch Normalization
MMD	Maximum Mean Discrepancy
MVD	Maximum Variance Discrepancy
CORAL	Correlation Alignment
GRL	Gradient Reversal Layer
GAP	Global Average Pooling
BP	Backpropagation
GIS	Greedy Iterative Search
RNN	Recurrent Neural Network
CWRU	Case Western Reserve University
EDM	Electro-Discharge Machining
EPRI	Electric Power Research Institute
DE	Drive-End
FE	Fan-End
IR	Inner Raceway
OR	Outer Raceway
B	Ball Element
1D	One-dimensional
2D	Two-dimensional
t-SNE	t-Distributed Stochastic Neighbor Embedding

References

1. Leffler, J.; Trnka, P. Failures of electrical machines-review. In *Proceedings of the 2022 8th International Youth Conference on Energy (IYCE)*; IEEE: New York, NY, USA, 2022; pp. 1–4.
2. Albrecht, P.; Appiarius, J.; McCoy, R.; Owen, E.; Sharma, D. Assessment of the Reliability of Motors in Utility Applications-Updated. *IEEE Trans. Energy Convers.* **1986**, *EC-1*, 39–46.
3. Wen, Y.; Rahman, M.F.; Xu, H.; Tseng, T.L.B. Recent advances and trends of predictive maintenance from data-driven machine prognostics perspective. *Measurement* **2022**, *187*, 110276. [CrossRef]
4. Fink, O.; Wang, Q.; Svensen, M.; Dersin, P.; Lee, W.J.; Ducoffe, M. Potential, challenges and future directions for deep learning in prognostics and health management applications. *Eng. Appl. Artif. Intell.* **2020**, *92*, 103678. [CrossRef]
5. 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] [PubMed]
6. Ince, T.; Kilickaya, S.; Eren, L.; Devecioglu, O.C.; Kiranyaz, S.; Gabbouj, M. Improved domain adaptation approach for bearing fault diagnosis. In *Proceedings of the IECON 2022—48th Annual Conference of the IEEE Industrial Electronics Society*; IEEE: New York, NY, USA, 2022; pp. 1–6.
7. Wu, M.; Shukla, S.; Vrancken, B.; Verbeke, M.; Karsmakers, P. Data-driven approach to identify acoustic emission source motion and positioning effects in laser powder bed fusion with frequency analysis. *Procedia CIRP* **2025**, *133*, 531–536. [CrossRef]
8. Wu, M.; Yao, Z.; Abts, R.; Karsmakers, P.; Verbeke, M.; Reynaerts, D. A threshold-free and label-free pipeline for adaptive pulse classification in electrical discharge machining. *Procedia CIRP* **2025**, *133*, 692–697.
9. Zhao, Z.; Zhang, Q.; Yu, X.; Sun, C.; Wang, S.; Yan, R.; Chen, X. Applications of unsupervised deep transfer learning to intelligent fault diagnosis: A survey and comparative study. *IEEE Trans. Instrum. Meas.* **2021**, *70*, 3525828. [CrossRef]
10. Sun, B.; Feng, J.; Saenko, K. Correlation alignment for unsupervised domain adaptation. In *Domain Adaptation in Computer Vision Applications*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 153–171.
11. Ganin, Y.; Ustinova, E.; Ajakan, H.; Germain, P.; Larochelle, H.; Laviolette, F.; March, M.; Lempitsky, V. Domain-adversarial training of neural networks. *J. Mach. Learn. Res.* **2016**, *17*, 1–35.
12. Long , M.; Cao, Z.; Wang, J.; Jordan, M.I. Conditional adversarial domain adaptation. In *Advances in Neural Information Processing Systems 31 (NeurIPS 2018)*; Curran Associates, Inc.: Red Hook, NY, USA, 2018.
13. Yan, J.; Ye, Z.S.; He, S.; He, Z. A feature disentanglement and unsupervised domain adaptation of remaining useful life prediction for sensor-equipped machines. *Reliab. Eng. Syst. Saf.* **2024**, *242*, 109736.
14. Xu, M.; Guan, P.; Shi, X.; Jiang, R.; Tian, J.; Geng, J.; Xiong, G. Research on Bearing Fault Diagnosis Methods Based on Various Convolutional Neural Network Architectures. *IEEE Access* **2025**, *13*, 44445–44465. [CrossRef]
15. Kiranyaz, S.; Ince, T.; Iosifidis, A.; Gabbouj, M. Operational neural networks. *Neural Comput. Appl.* **2020**, *32*, 6645–6668. [CrossRef]
16. Keleş, O.; Tekalp, A.M. Paon: A new neuron model using padé approximants. In *Proceedings of the 2024 IEEE International Conference on Image Processing (ICIP)*; IEEE: New York, NY, USA, 2024; pp. 207–213.
17. Kilickaya, S.; Eren, L. Padé Approximant Neural Networks for Enhanced Electric Motor Fault Diagnosis Using Vibration and Acoustic Data. *J. Vib. Eng. Technol.* **2025**, *13*, 539. [CrossRef]
18. Case Western Reserve University. Bearing Data Center. Available online: https://engineering.case.edu/bearingdatacenter/download-data-file (accessed on 2 July 2025).
19. Paderborn University Bearing Data Center. Bearing Data Center. Available online: https://mb.uni-paderborn.de/kat/forschung/bearing-datacenter/data-sets-and-download (accessed on 16 May 2026).
20. Wang, X.; Liu, F. Triplet loss guided adversarial domain adaptation for bearing fault diagnosis. *Sensors* **2020**, *20*, 320. [CrossRef] [PubMed]
21. Kilickaya, S.; Celebioglu, C.; Eren, L.; Askar, M. Thermal image-based fault diagnosis in induction machines via self-organized operational neural networks. In *Proceedings of the 2025 IEEE Symposium on Computational Intelligence on Engineering/Cyber Physical Systems (CIES)*; IEEE: New York, NY, USA, 2025; pp. 1–7.
22. Kilickaya, S.; Ahishali, M.; Celebioglu, C.; Sohrab, F.; Eren, L.; Ince, T.; Askar, M.; Gabbouj, M. Audio-based anomaly detection in industrial machines using deep one-class support vector data description. In *Proceedings of the 2025 IEEE Symposium on Computational Intelligence on Engineering/Cyber Physical Systems Companion (CIES Companion)*; IEEE: New York, NY, USA, 2025; pp. 1–5.
23. Kilickaya, S.; Eren, L. Bearing fault detection in adjustable speed drives via self-organized operational neural networks. *Electr. Eng.* **2025**, *107*, 4503–4515.
24. Qian, Q.; Luo, J.; Qin, Y. Adaptive intermediate class-wise distribution alignment: A universal domain adaptation and generalization method for machine fault diagnosis. *IEEE Trans. Neural Netw. Learn. Syst.* **2024**, *36*, 4296–4310. [CrossRef]
25. Zhang, W.; Li, X.; Ma, H.; Luo, Z.; Li, X. Open-set domain adaptation in machinery fault diagnostics using instance-level weighted adversarial learning. *IEEE Trans. Ind. Inform.* **2021**, *17*, 7445–7455. [CrossRef]

26. Wang, X.; Mao, D.; Li, X. Bearing fault diagnosis based on vibro-acoustic data fusion and 1D-CNN network. *Measurement* **2021**, *173*, 108518. [CrossRef]
27. Youcef Khodja, A.; Guersi, N.; Saadi, M.N.; Boutasseta, N. Rolling element bearing fault diagnosis for rotating machinery using vibration spectrum imaging and convolutional neural networks. *Int. J. Adv. Manuf. Technol.* **2020**, *106*, 1737–1751.
28. Celebioglu, C.; Kilickaya, S.; Eren, L. Smartphone-based bearing fault diagnosis in rotating machinery using audio data and 1d convolutional neural networks. In *Proceedings of the International Conference on Computer Systems and Technologies 2024*; ACM: New York, NY, USA, 2024; pp. 149–154.
29. 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]
30. Kiranyaz, S.; Malik, J.; Abdallah, H.B.; Ince, T.; Iosifidis, A.; Gabbouj, M. Self-organized operational neural networks with generative neurons. *Neural Netw.* **2021**, *140*, 294–308. [CrossRef] [PubMed]
31. Ince, T.; Malik, J.; Devecioglu, O.C.; Kiranyaz, S.; Avci, O.; Eren, L.; Gabbouj, M. Early bearing fault diagnosis of rotating machinery by 1D self-organized operational neural networks. *IEEE Access* **2021**, *9*, 139260–139270. [CrossRef]
32. Zhang, Y.; Zhou, T.; Huang, X.; Cao, L.; Zhou, Q. Fault diagnosis of rotating machinery based on recurrent neural networks. *Measurement* **2021**, *171*, 108774. [CrossRef]
33. Zhu, J.; Jiang, Q.; Shen, Y.; Qian, C.; Xu, F.; Zhu, Q. Application of recurrent neural network to mechanical fault diagnosis: A review. *J. Mech. Sci. Technol.* **2022**, *36*, 527–542. [CrossRef]
34. Qiu, S.; Cui, X.; Ping, Z.; Shan, N.; Li, Z.; Bao, X.; Xu, X. Deep learning techniques in intelligent fault diagnosis and prognosis for industrial systems: A review. *Sensors* **2023**, *23*, 1305. [CrossRef] [PubMed]
35. Moradzadeh, A.; Teimourzadeh, H.; Mohammadi-Ivatloo, B.; Pourhossein, K. Hybrid CNN-LSTM approaches for identification of type and locations of transmission line faults. *Int. J. Electr. Power Energy Syst.* **2022**, *135*, 107563. [CrossRef]
36. Fernandes, M.; Corchado, J.M.; Marreiros, G. Machine learning techniques applied to mechanical fault diagnosis and fault prognosis in the context of real industrial manufacturing use-cases: A systematic literature review. *Appl. Intell.* **2022**, *52*, 14246–14280. [CrossRef]
37. Xu, Y.; Li, Z.; Wang, S.; Li, W.; Sarkodie-Gyan, T.; Feng, S. A hybrid deep-learning model for fault diagnosis of rolling bearings. *Measurement* **2021**, *169*, 108502. [CrossRef]
38. Quiñonero-Candela, J.; Sugiyama, M.; Schwaighofer, A.; Lawrence, N.D. *Dataset Shift in Machine Learning*; MIT Press: Cambridge, MA, USA, 2022.
39. Li, X.; Zhang, W.; Ding, Q.; Sun, J.Q. Multi-layer domain adaptation method for rolling bearing fault diagnosis. *Signal Process.* **2019**, *157*, 180–197. [CrossRef]
40. Pan, S.J.; Yang, Q. A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.* **2009**, *22*, 1345–1359. [CrossRef]
41. Liu, X.; Liu, X.; Hu, B.; Ji, W.; Xing, F.; Lu, J.; You, J.; Kuo, C.C.J.; El Fakhri, G.; Woo, J. Subtype-aware unsupervised domain adaptation for medical diagnosis. *Proc. Proc. AAAI Conf. Artif. Intell.* **2021**, *35*, 2189–2197. [CrossRef]
42. Zhang, R.; Tao, H.; Wu, L.; Guan, Y. Transfer learning with neural networks for bearing fault diagnosis in changing working conditions. *IEEE Access* **2017**, *5*, 14347–14357. [CrossRef]
43. Zhang, C.; Xu, L.; Li, X.; Wang, H. A method of fault diagnosis for rotary equipment based on deep learning. In *Proceedings of the 2018 Prognostics and System Health Management Conference (PHM-Chongqing)*; IEEE: New York, NY, USA, 2018; pp. 958–962.
44. Chen, D.; Yang, S.; Zhou, F. Incipient fault diagnosis based on DNN with transfer learning. In *Proceedings of the 2018 International Conference on Control, Automation and Information Sciences (ICCAIS)*; IEEE: New York, NY, USA, 2018; pp. 303–308.
45. Dai, W.; Yang, Q.; Xue, G.R.; Yu, Y. Boosting for transfer learning. In *Proceedings of the 24th International Conference on Machine Learning*; ICML '07; ACM: New York, NY, USA, 2007, pp. 193–200. [CrossRef]
46. Li, Y.; Wang, N.; Shi, J.; Liu, J.; Hou, X. Revisiting batch normalization for practical domain adaptation. *arXiv* **2016**, arXiv:1603.04779.
47. Borgwardt, K.M.; Gretton, A.; Rasch, M.J.; Kriegel, H.P.; Schölkopf, B.; Smola, A.J. Integrating structured biological data by kernel maximum mean discrepancy. *Bioinformatics* **2006**, *22*, e49–e57. [CrossRef] [PubMed]
48. Sun, B.; Saenko, K. Deep coral: Correlation alignment for deep domain adaptation. In Proceedings of the European Conference on Computer Vision; Springer: Berlin/Heidelberg, Germany, 2016; pp. 443–450.
49. Lu, W.; Liang, B.; Cheng, Y.; Meng, D.; Yang, J.; Zhang, T. Deep model based domain adaptation for fault diagnosis. *IEEE Trans. Ind. Electron.* **2016**, *64*, 2296–2305. [CrossRef]
50. Zhang, B.; Li, W.; Li, X.L.; Ng, S.K. Intelligent fault diagnosis under varying working conditions based on domain adaptive convolutional neural networks. *IEEE Access* **2018**, *6*, 66367–66384. [CrossRef]
51. Zhang, Z.; Chen, H.; Li, S.; An, Z. Unsupervised domain adaptation via enhanced transfer joint matching for bearing fault diagnosis. *Measurement* **2020**, *165*, 108071. [CrossRef]
52. Liu, Z.H.; Lu, B.L.; Wei, H.L.; Chen, L.; Li, X.H.; Rätsch, M. Deep adversarial domain adaptation model for bearing fault diagnosis. *IEEE Trans. Syst. Man. Cybern. Syst.* **2019**, *51*, 4217–4226. [CrossRef]

53. Yu, X.; Zhao, Z.; Zhang, X.; Sun, C.; Gong, B.; Yan, R.; Chen, X. Conditional adversarial domain adaptation with discrimination embedding for locomotive fault diagnosis. *IEEE Trans. Instrum. Meas.* **2020**, *70*, 3503812 . [CrossRef]
54. George, A., Jr. *Essentials of Padé Approximants*; Elsevier: Amsterdam, The Netherlands, 1975.
55. Molina, A.; Schramowski, P.; Kersting, K. Padé activation units: End-to-end learning of flexible activation functions in deep networks. *arXiv* **2019**, arXiv:1907.06732.
56. Glorot, X.; Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*; JMLR Workshop and Conference Proceedings; PMLR: Cambridge, MA, USA, 2010; pp. 249–256.
57. Guo, J.; Chen, K.; Liu, J.; Ma, Y.; Wu, J.; Wu, Y.; Xue, X.; Li, J. Bearing Fault Diagnosis Based on Deep Discriminative Adversarial Domain Adaptation Neural Networks. *CMES-Comput. Model. Eng. Sci.* **2024**, *138*, 2619. [CrossRef]
58. Wang, Q.; Michau, G.; Fink, O. Domain adaptive transfer learning for fault diagnosis. In *Proceedings of the 2019 Prognostics and System Health Management Conference (PHM-Paris)*; IEEE: New York, NY, USA, 2019; pp. 279–285.