Evaluation of Input Presentation in Transfer Learning for Bearing Fault Detection

Amirhossein Berenji, Sławomir Nowaczyk, Zahra Taghiyarrenani, Sepideh Pashami

Center for Applied Intelligence Systems Research, Halmstad University, Sweden

amirhossein.berenji@hh.se

ABSTRACT

Transfer learning is a promising technique to overcome data insufficiency, a noticeable barrier to real-world application of intelligent maintenance solutions. Although the importance of data preparation and preprocessing is widely acknowledged, no study in particular has investigated the effect of vibration data input presentations on transferability capabilities. This study aims to fill in this gap by conducting experiments across three benchmark datasets to evaluate direct transfer, catastrophic forgetting and data efficiency for bearing fault classification. Moreover, we explore the opportunity to employ source model pseudo-labeling to reduce the need for data labeled by human experts. Our findings show that not only does the choice of preprocessing pipeline significantly affect target-set performance, but also that the vulnerability to catastrophic forgetting varies accordingly. Thus, we conclude that finding the right data processing routine is also a key component in achieving supreme transfer learning performance and, indeed, it deserves more attention. The code base of this study is open-sourced and made publicly available to support reproducibility, transparency, and further research.

1. INTRODUCTION

Predictive maintenance is receiving increasing attention in industrial operations, as run-to-failure strategies lead to costly and unpredictable downtime (Carvalho et al., 2019); For illustrative purposes, the authors in (Thomas & Weiss, 2021) report that the adoption of predictive and preventive maintenance strategies can reduce unplanned downtime by up to 52.7%. Recent advances in artificial intelligence have accelerated this shift. In many applications, condition monitoring relies on vibration signals to capture the health state of machinery, particularly for rotating equipment, which is critical in industrial systems (Liu, Yang, Zio, & Chen, 2018). However, despite strong results in controlled settings, real-world deployment remains limited, largely due to scarce and weakly annotated fault data. This challenge is further exacerbated by industrial variability (e.g., changing operating conditions, sensor drift, and external disturbances). Transfer learning (TL) mitigates data scarcity by transferring knowledge from a source domain to a related target domain (Z. Zhao, Alzubaidi, Zhang, Duan, & Gu, 2024), with effectiveness increasing as domain similarity grows.

In this setting, both model design and data are fundamental. The model-centric paradigm has driven significant progress through advances in architectures, optimization, and learning strategies. At the same time, data-centric considerations, such as data quality and preprocessing are equally critical for effective learning; particularly within safety-critical and high-stakes application domains, including condition-based maintenance of industrial machines. These factors not only influence predictive performance but also affect key properties such as explainability (pushing for explainable representation drops classification accuracy up to 13%) (Berenji, Nowaczyk, & Taghiyarrenani, 2023) and diagnostic robustness (through extracting a feature representation that is immune to fault features missing from the signal) (Y. Zhao & Xu, 2024).

Motivated by this perspective, the choice of input representation for vibration signals becomes a key factor in transfer learning for fault diagnosis. A wide range of input representations has been explored, including raw waveforms (B. Zhao, Zhang, Zhan, & Pang, 2020), frequency-domain features (Berenji & Taghiyarrenani, 2022), and time–frequency representations (Shao, McAleer, Yan, & Baldi, 2018). Despite the widespread use of these input preprocessing routines, their impact on transferability remains insufficiently understood. This raises a central question: *how do input representations and their associated preprocessing pipelines influence transfer learning performance in fault diagnosis, under realistic constraints such as limited labeled target data?*

In this study, we investigate the role of vibration signal representations in transfer learning for bearing fault diagnosis. We systematically evaluate a set of commonly used preprocessing routines, including frequency-domain transformations and demodulation techniques, within a unified experimental frame-

work across three benchmark datasets. We consider three transfer scenarios: (i) direct transfer without fine-tuning, (ii) supervised fine-tuning with labeled target data, and (iii) fine-tuning using Dynamic Bootstrapping (DB), where pseudo-labels are generated from the source model for unlabeled target data. Performance is assessed beyond target-domain accuracy, including fine-tuning efficiency under limited data and the extent of catastrophic forgetting in the source domain. All preprocessing pipelines and experimental procedures are implemented using our previously developed library, **Damavand**[1]. To ensure reproducibility, the code and experiments specific to this study are open-sourced[2].

The main contributions of this study are:

- We present a unified and systematic study of vibration signal representations in transfer learning for bearing fault diagnosis, benchmarking a diverse set of commonly used preprocessing routines, including frequency-domain and demodulation-based methods, across multiple datasets under a consistent experimental setup, addressing a gap in understanding their impact on transferability.
- We evaluate transfer learning under practical scenarios, including no target data, limited labeled data, and fully unlabeled target data via pseudo-labeling, reflecting realistic industrial constraints.
- We support reproducible research by building on our existing library, **Damavand**, and releasing all code and experiments specific to this study.

2. RELATED WORKS

TL has been extensively applied to enhance fault detection in rotating machinery. Several studies (T. Lu, Yu, Han, & Wang, 2020; B. Zhao et al., 2020) leverage TL to bridge domain gaps caused by varying operating conditions and fault severities across different bearings. Beyond single-component adaptation, (Berenji & Taghiyarrenani, 2022) shows that classifiers often generalize poorly to unseen conditions or mixed-domain settings, and addresses this through contrastive pre-training to improve robustness to environmental variability. Cross-machine knowledge transfer has also been explored; notably, (Shao et al., 2018) considers a diverse domain space including bearings, induction motors, and gearboxes to handle heterogeneous components and operating regimes.

Different input presentations are employed in TL for rotating machinery, including raw vibration waveforms (B. Zhao et al., 2020), frequency-domain features obtained via FFT (Berenji & Taghiyarrenani, 2022), and time–frequency representations (Shao et al., 2018). Despite their widespread use, systematic comparisons of their effectiveness in transfer learning scenarios remain limited.

While fine-tuning-based TL is effective with limited target-domain data (Misbah, Lee, & Keung, 2024), it still relies on labeled samples, which are costly and time-consuming to collect. To reduce this dependency, pseudo-labeling has been explored as an alternative, where labels are generated from imperfect but accessible sources. This introduces label noise, requiring models that can learn robustly under noisy supervision. Recent studies address this challenge by filtering noisy samples based on loss values (Wang et al., 2026) or improving robustness via label smoothing and decoupled feature extraction (Li, He, Chen, Feng, & Xie, 2025). In contrast, (Zhong et al., 2024) actively refines erroneous labels during training using class prototypes and contrastive regularization within a dual-head architecture.

Despite these advances, the use of noisy or pseudo-labeled data in transfer learning for fault diagnosis remains under-explored. In particular, leveraging source-domain models as supervisory signals for unlabeled target data has not been systematically studied. This motivates our investigation of using a source-domain expert model as an imperfect oracle, combined with Dynamic Bootstrapping, to enable adaptation under fully unlabeled target data.

3. METHODS

This study addresses bearing fault classification using vibration signals. We investigate three TL scenarios as: **i) direct transfer**, **ii) labeled fine-tuning** and **iii) fine-tuning on pseudo-labels by Dynamic Bootstrapping**. The rest of this section discusses each of these scenarios. Additionally, these methods are schematically visualized in Figure 1.

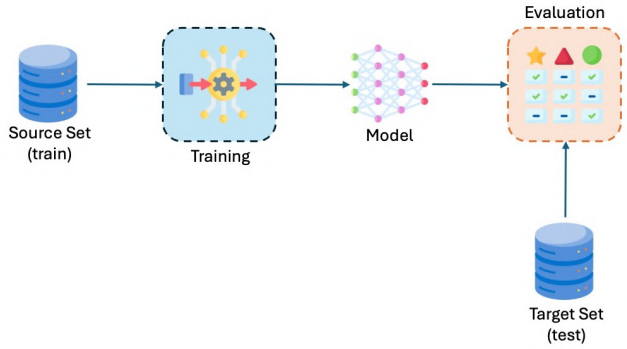
3.1. Direct Transfer

As visualized in Figure 1a, direct transfer does not involve any fine-tuning at all; it only includes training a model on the source set, as a supervised classification problem using cross-entropy loss, and evaluating its performance over the source and the remaining sets to assess both the source-set and the pre-fine-tuning target-sets classification performance.

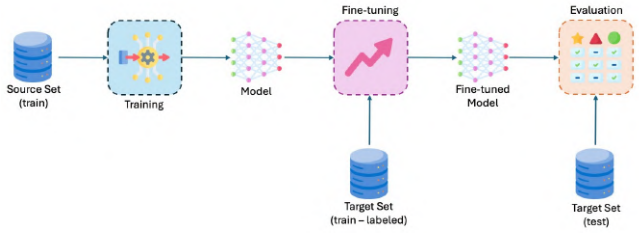
3.2. Conventional Supervised Fine-tuning

As demonstrated in Figure 1b, in this scenario, we treat the fine-tuning problem as a conventional supervised classification problem; the only point to consider is that we use the model pre-trained on a source set, instead of a randomly initialized model. Similar to the case of direct transfer, once the fine-tuning process is finished, we evaluate the classification performance of the model over all the datasets (source, target and the other set). Again, in this set of experiments, we employ cross-entropy loss to fine-tune the model previously trained on the source set on the target domain.

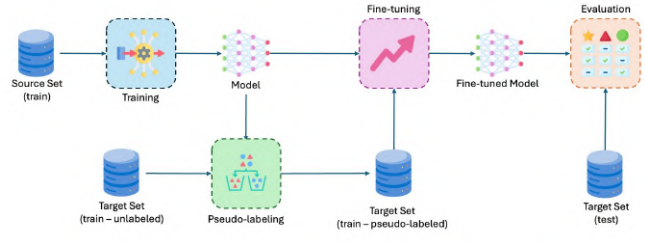
[1] https://pydamavand.github.io/documentation/
[2] http://github.com/caisr-hh/RepresentationCentricTL



(a) Direct transfer



(b) Conventional supervised fine-tuning



(c) Fine-tuning on pseudo-labels

Figure 1. Visualization of the TL scenarios

3.3. Fine-tuning on Pseudo-labels by Dynamic Bootstrapping

Replacing the target data ground-truth labeling (typically by human experts) with pseudo-labels by the source model can reduce the costs of fine-tuning, at the price of inducing label noise; as models start memorizing the label noise, their performance on the clean test set starts degrading, due to overfitting (Y. Lu, Xu, & He, 2023). A wide range of methods are developed to confront label noise; particularly, a category of these methods called label refurbishment, aims to assist neural networks in mitigating the effect of erroneous labels by training on refurbished labels - a combination of the noisy ground-truth labels and the predictions of the model itself (Sukhbaatar & Fergus, 2014).

Dynamic Bootstrapping (DB) (Arazo, Ortego, Albert, O'Connor, & McGuinness, 2019) is one of the promising label refurbishment methods and Equation demonstrates how the classification loss is calculated in DB:

$$\mathcal{L}_{DB} = -\sum_{i=1}^{N}[(1-w_i)y_i + w_i z_i]^T \cdot \log(h_i), \qquad (1)$$

where w_i represents the weight assigned to the model's refurbished prediction (often derived from a mixture model to estimate sample noisiness), y_i is the original noisy ground-truth label, z_i is the refurbished label (typically the model's running prediction), and h_i denotes the current softmax output of the model. Unlike vanilla bootstrapping (Reed et al., 2014) that contribution of ground-truth labels and model's prediction in refurbishing the label is set constantly for all the samples in the training set as a hyperparameter, in DB w_i is sample-specific; inspired by the hypothesis that samples with noisy labels employ higher losses, authors employ Beta mixture models with two components to approximate the probability of each sample being noisy or clean. Then, the probability of one sample being noisy w_i is used to determine the contribution of the model's predicted label and the rest is assigned to the ground-truth label.

We employ DB to fine-tune a source-domain model using only unlabeled samples from the target domain. By treating the source model's initial predictions as noisy labels, DB facilitates effective adaptation to the target domain. In Figure 1c, we have visualized the procedure of this TL scenario.

4. EXPERIMENTS

Throughout this section, we introduce the framework of our experiments; we start by presenting the pipelines included in this study. Next, we briefly review the datasets and overall data preparation; last but not least, we discuss the design of our experiments.

4.1. Data processing pipelines

To evaluate various vibration input presentations, we define several processing routines summarized in Table 1. Here, *Raw*, *Scaling*, *Env*, and *FFT* denote the raw time-domain waveform, sample-specific z-score scaling, Hilbert transform as demodulation, and the Fast-Fourier Transform, respectively. For LSTM-based architectures, signals are partitioned into consecutive subsegments to ensure computational feasibility. Optimal window lengths for each routine-dataset pair were determined by the lowest possible validation-set classification loss value, as detailed in Section A; we use the corresponding window lengths throughout all the experiments. Before frequency transformations (FFT and Zoomed FFT), filtering is applied to eliminate near-zero components and prevent aliasing. All signal processing steps are implemented using the *Damavand* signal processing module [3].

Table 1. Description of the processing routines

No.	Domain	Title	Demodulation	Frequency Transformation
1	Time	Raw → Scaling → CNN	✗	✗
2		Raw → Env → Scaling → CNN	✓	
3		Raw → Scaling → Sequencing → LSTM	✗	
4		Raw → Env → Scaling → Sequencing → LSTM	✓	
5	Frequency	Raw → FFT → Scaling → CNN	✗	FFT
6		Raw → Env → FFT → Scaling → CNN	✓	
7		Raw → FFT → Scaling → Resampling → DNN	✗	
8		Raw → Env → FFT → Scaling → Resampling → DNN	✓	
9		Raw → Zoomed FFT → Scaling → CNN	✗	Zoomed FFT
10		Raw → Env → Zoomed FFT → Scaled → CNN	✓	
11		Raw → Zoomed FFT → Scaling → DNN	✗	
12		Raw → Env → Zoomed FFT → Scaling → DNN	✓	

4.2. Datasets

In this study, we conduct our experiments on three bearing fault diagnosis benchmark datasets: Case-Western Reserve University (CWRU), Machine Failure Prevention Technology (MFPT), and Korea Advanced Institute of Science and Technology (KAIST). To keep the TL problem consistent, we merely consider the three-way (inner- and outer-race faults and the normal) bearing fault detection problem, even though more classes are available in CWRU and KAIST.

Conventionally, fault detection in rotating machines (in general and bearings specifically) is done according to the appearance of specific frequency components, known as fault characteristic frequency components; these components are dependent on **rotational speed** and **bearing geometry**; additionally, the **loading condition** also affects the acceleration signals, significantly. Taking into account these properties, selected datasets offer similarities and differences. According to the rotational speed, while the actual rotational speed differs, its pattern is the same across all of them; in other words, the rotational speed is kept constant across each measurement (segmented signal), while CWRU includes four fairly close rotational speeds.

In Table 2, fault coefficients and their corresponding fault characteristics frequency components, for each combination of dataset-fault, are summarized; according to this table, CWRU and KAIST have very close fault coefficients; however, the significant difference in rotational speed makes this similarity obsolete. On the other hand, CWRU and MFPT are the most similar datasets because of the proximity of their fault-characteristic frequencies. Intuitively, the closer these frequencies are for two datasets, the more similar they are; however, for representations provided by routines not using ZoomedFFT (all except 10 and 12), hugely different sampling frequencies with which these datasets are collected (12 KHz for CWRU, 25.6 KHz for KAIST and $\approx$ 48.8 and 97.6 KHz for MFPT), make them very dissimilar.

Finally, from the loading condition, datasets are boldly different. MFPT has both horizontal and vertical loads, while CWRU has only the conventional horizontal load and KAIST includes load-free operation.

Table 2. Summarization of fault coefficients (COE), rotational speeds and fault characteristics frequencies (FCF) for inner-race (IR) and outer-race (OR) faults and in the datasets.

Dataset	IR COE	OR COE	Rotational Speed (Hz)	IR FCF (Hz)	OR FCF (Hz)
CWRU	5.4152	3.5848	28.83 - 29.95	156.14 - 162.19	103.36 - 107.36
MFPT	4.7552	3.2448	25	118.88	81.12
KAIST	5.4229	3.5764	50.17	272.07	179.43

4.3. Data preparations

Once the datasets are downloaded as raw files, again we used *Damavand* to transform the raw files into structured pairs of signals and corresponding metadata; to do so, we segment raw signals into approximately 200 ms long segments for each dataset; as the three datasets are recorded at different sampling frequencies, the segmented signals have different lengths (2400, 4800 and 9600, respectively) as well. It is important to mention that we use a 25% overlap during the segmentation process. As mentioned earlier, the MFPT dataset is recorded in both 48.8 and 97.6 KHz; in this study, we downsampled the 97.6 KHz signals to match the 48.8 KHz sampling frequency, before segmenting them into 200 ms segments.

4.4. Experimental design

Our experimental framework follows a classical TL paradigm across three datasets. Generally, our experiments involve training over one of the sets (source), followed by fine-tuning on the two remaining target datasets. To take into account the three scenarios of direct transfer, target set data limitation and fine-tuning on pseudo-labels, we designed three sets of experiments:

- Comparing the direct classification performance of each of the routines on target datasets.
- Benchmarking the data efficiency of the routines by changing the amount of target-set data used for supervised fine-tuning.

[3] ://pydamavand.github.io/documentation/signal_processing/

- Investigating the feasibility of using the source-set model as an erroneous oracle to pseudo-label the unlabeled data from the target set and fine-tuning on it using DB.

The first set of experiments explores the most basic question behind this paper, namely whether some preprocessing routines are inherently better than others in terms of knowledge transferability. In the second set, we try to compare and rank the routines according to their data efficiency; results from this set of experiments are important to address the few-shot regimes. In this study, we go over 8 levels of limited data, including 10%, 5%, and 1% of the original training data, in addition to 25, 20, 15, 10, and 5 shots per class. Finally, we regard the third set of experiments as a feasibility study to fine-tune on merely unlabeled data from the target set.

To ensure rigorous evaluation, we first conduct a hyperparameter optimization phase for every routine-dataset combination. Once the optimal parameters are identified for both initial training and fine-tuning, models are trained end-to-end. To mitigate the stochastic effects of network initialization and data splitting, all experiments are repeated five times.

Data is partitioned using a 60-40 split for training and testing, respectively, with 25% of the training set reserved for validation. To ensure comparability across all experiments, the held-out test set for each dataset remains fixed. All performance metrics reported in the subsequent sections correspond to the classification accuracy achieved on these consistent test sets.

5. RESULTS AND DISCUSSION

Table 3 summarizes the source-domain classification accuracy across all routines. Most configurations achieve near-perfect performance, with the lowest accuracy (0.9629) observed for R6 and R10 on the CWRU dataset. The slight performance dip can be attributed to the inherent class imbalance within CWRU, whereas the perfectly balanced KAIST dataset yields 1.00 accuracy for nearly all routines.

Figure 2 visualizes the pre-fine-tuning "base" performance (yellow), the post-fine-tuning target accuracy (blue), and the relative performance degradation on the source set — commonly referred to as catastrophic forgetting (red). These results indicate that the choice of routine significantly influences the generalizability of learned features. The "fine-tuning-free" performance serves as a critical performance intercept; a higher starting point typically facilitates superior final target accuracy. For instance, the initial accuracy for the $M \rightarrow C$ varies by as much as 0.40 comparing R3 vs. R6; similarly, $K \rightarrow M$ differs again by around 0.40, comparing R11 and R12. It is worth mentioning that the quantified version of all the radar charts discussed during this section can be found in Appendix B; moreover, detailed results are also available in the official repository of this study.

Notably, while nearly all routines converge toward high target-set accuracy after fine-tuning (indicated by the expansive blue areas in Figure 2), the magnitude of source-domain degradation or catastrophic forgetting varies dramatically. From a plasticity-stability dilemma perspective, some routines are capable of not impeding the model's *plasticity* (its capacity to adapt to new target data) and simultaneously, learn preservative knowledge through the *stability* of the learned features. An example of such phenomena is the comparison of R9 and R10; convolution filters in R9 adapt themselves to the patterns from the target domain; therefore, the source drop is quite significant when the source and target domains are too dissimilar (in some cases as high as the base). On the other hand, for the case of R11, the DNN model identifies the frequency ranges associated with different faults, leading to considerably smaller source drop. This suggests that certain processing pipelines, while high-performing for the task at hand, may create **brittle** feature representations that collapse when the network weights shift to accommodate new domain statistics.

Detailed observations from Figure 2 include:

- **LSTM vs. CNN:** R3 and R4 are less vulnerable to catastrophic forgetting, in comparison with R1 and R2.
- **Env promotes (not always) catastrophic forgetting:** For both base pre-fine-tuning and source drop, the application of Env yields inconsistent results (e.g., comparing R1 vs. R2, Env enhances the base in $K \rightarrow C$ and $M \rightarrow C$ and degrades base as well in $M \rightarrow K$ and $C \rightarrow M$; similarly, comparing R9 and R10, application of Env decreases the source drop for $K \rightarrow C$ and $K \rightarrow M$ but increases it for $M \rightarrow K$ and $C \rightarrow K$). In general, though, the application of Env is likely to contribute to higher rates of catastrophic forgetting; examples are R7 vs. R8 and R11 vs. R12.
- **Resampling Robustness:** Contrary to the assumption that resampling might induce information loss, R7 (with resampling) remains competitive with R5. Even with the addition of Env (R8), performance remains robust despite some domain-specific variability.
- **R11 offers the lowest catastrophic forgetting**: With a maximum catastrophic forgetting of below 0.4 ($K \rightarrow C$), R11 looks like the best choice for domain-preservative TL scenarios.

Following the results from the first set of experiments, we now move into the second set of experiments: robustness towards few-shot fine-tuning. In Figure 3 and 4, results from different volumes of the training set are visualized. The constant shrinkage of the blue hexagon through these figures is the most dominant feature of these two figures; quite intuitively, the performance is decreasing towards the decrease of the available data for fine-tuning; additionally, this shrinkage is not uniform for different combinations of source-target

Table 3. Classification accuracy over source domain, for different methods and datasets (mean over 5 iterations)

Dataset	1	2	3	4	5	6	7	8	9	10	11	12
M	1.0000	1.0000	0.9758	0.9523	1.0000	0.9957	1.0000	1.0000	1.0000	0.9957	0.9815	1.0000
C	0.9998	0.9897	0.9813	0.9904	0.9991	0.9629	1.0000	1.0000	0.9991	0.9629	0.9993	0.9955
K	1.0000	1.0000	0.9993	0.9990	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000

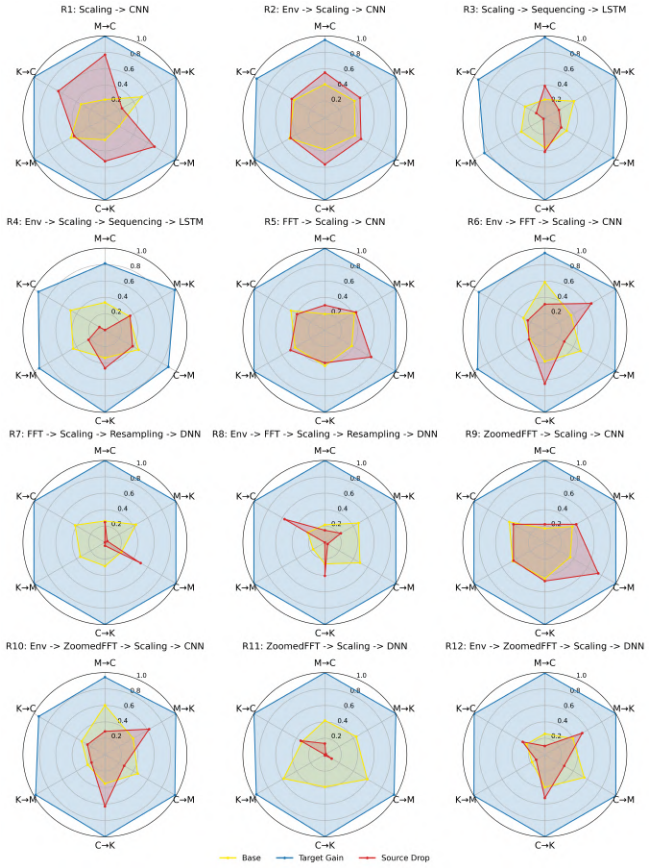
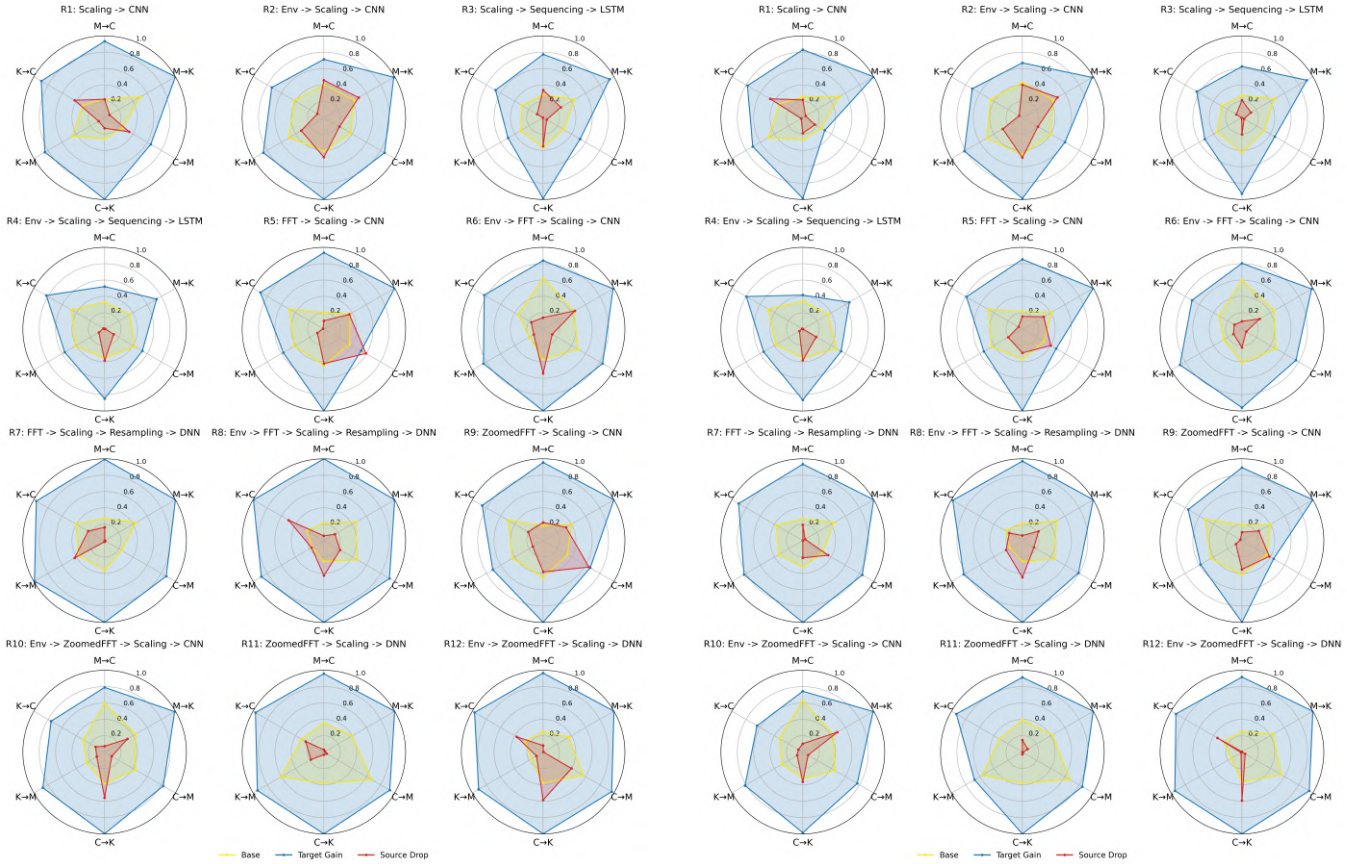


Figure 2. Fine-tuning-free, target gain and source drop of different routines for different source-target pairs

in routines. This means that different combinations require different volumes of data to maintain efficient fine-tuning; take R5 at Figure 3a as an example, where $M \rightarrow K$ and $C \rightarrow K$ are still almost 1.00, but $C \rightarrow M$ has reached below 0.6. Moreover, different routines offer notably different levels of data efficiency; taking classifier architecture into account, LSTM-based routines show the highest dependency on the amount of data, and this pattern is fairly consistent throughout all fine-tuning data limits. Next, for the case of CNN-based routines, routines extracting frequency-domain presentations (using either FFT or ZoomedFFT) offer higher robustness compared with routines that classify time-domain signals directly (R1 and R2); this aligns with the generally accepted fact that rotating machinery faults - bearings in particular - are easier to distinguish in the frequency domain. Routines with DNN classifiers (all utilizing frequency-domain features) offer the highest robustness towards the limitation of the fine-tuning data; as mentioned earlier, the superiority of the DNN-based routines over their CNN counterparts can be attributed to the fact that DNNs learn the frequency bands associated with the faults - that although are different for different datasets but can be learned persistently; CNNs on the other hand, recognize the patterns within their kernel ranges and such patterns vary from dataset to dataset. Amongst the DNN-based routines, R12 is the most data-efficient routine; according to the Figure 4d, with only 5 shots from each class of the target set, it manages to reach almost 0.8 for two of the transferring scenarios ($M \rightarrow C$ and $K \rightarrow C$) and 0.9 or higher for the rest of them; moreover, the blue hexagon offered by this routine stays arguably the most uniform one through the downsampling of the fine-tuning set.

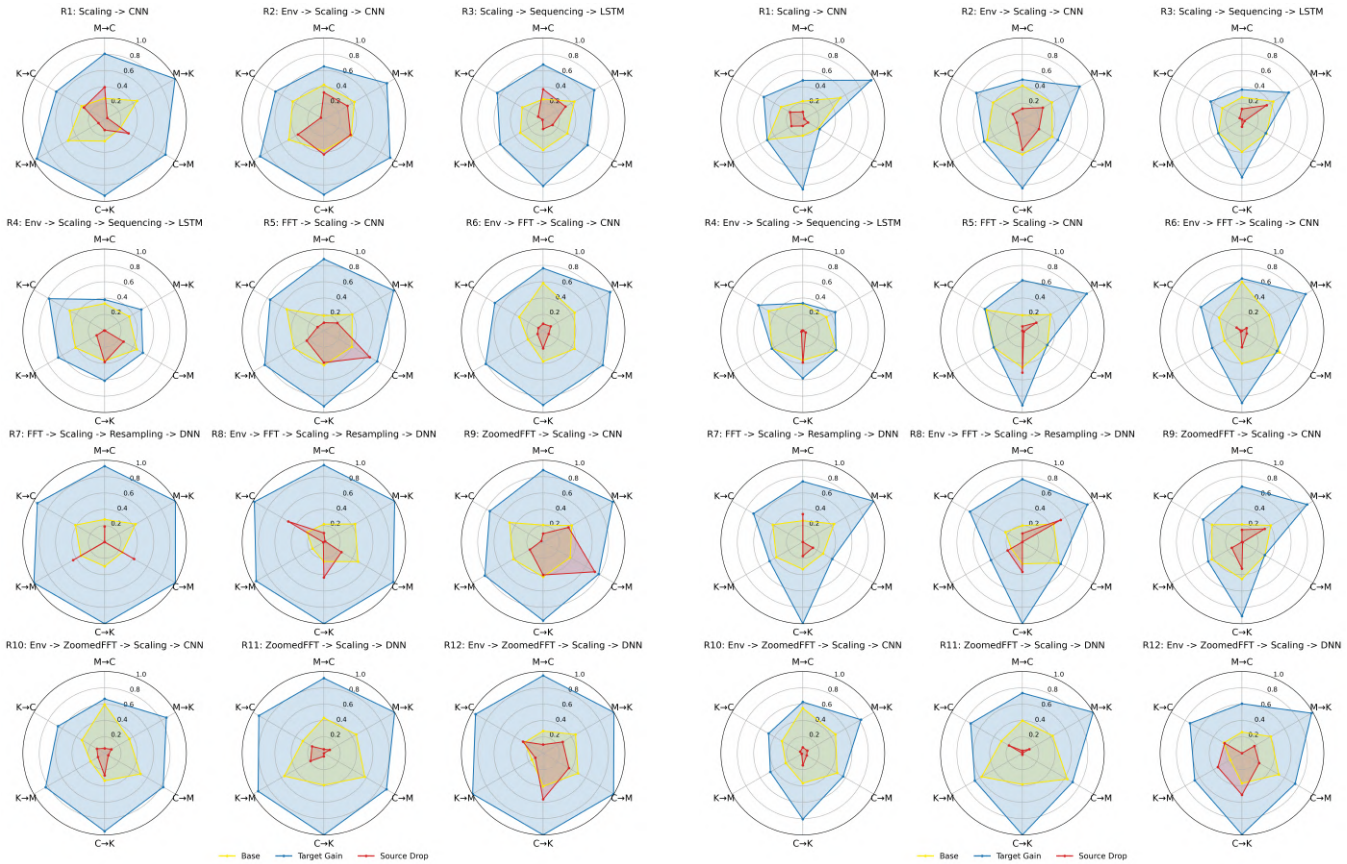
Last but not least, we reach results from the third set of experiments, assessing the feasibility of pseudo-labeling the unlabeled target set data and employing DB to fine-tune merely using the unlabeled data and pseudo-labels. In Figure 5a, results from fine-tuning on pure pseudo-labels are visualized; the almost perfect overlay of the blue and yellow hexagons for all routines shows that it is almost impossible to train using the pure pseudo-labels from the source model. In fact, the only exception is R7 on the $M \rightarrow C$ scenario, where the fine-tuning improves the target domain accuracy by around 0.2. It is worth mentioning that R7 and R8 (both using resampling) have considerably higher vulnerability to catastrophic forgetting; considering that the CNN counterparts of these two routines offer almost no source drop, this looks mostly attributable to resampling.

To further explore the feasibility of pseudo-labels, we repeat the same experiments with different percentages of pseudo-labels to be used during fine-tuning, replaced by the ground-truth labels to simulate the limited labeling budget (assuming by a human oracle) setup. In Figures 5b, 5c and 5d we visualize the results with 25%, 50% and 75% random ground-truth label recovery - where we randomly choose a percentage of the pseudo-labels to be replaced with the ground-truth labels. Comparing these figures with Figure 2, it is not lower than 75% labeling budget that some of the routines start achieving competitive classification accuracy; in particular, R7, R8, R11, and R12 manage to get near-perfect fine-tuning classification accuracy at 75% labeling budget. Hence, to achieve competitive performance with pseudo-labels at lower labeling budgets, we either need to improve the quality of the pseudo-labels (e.e, through physics-informed methods or an



(a) 10%

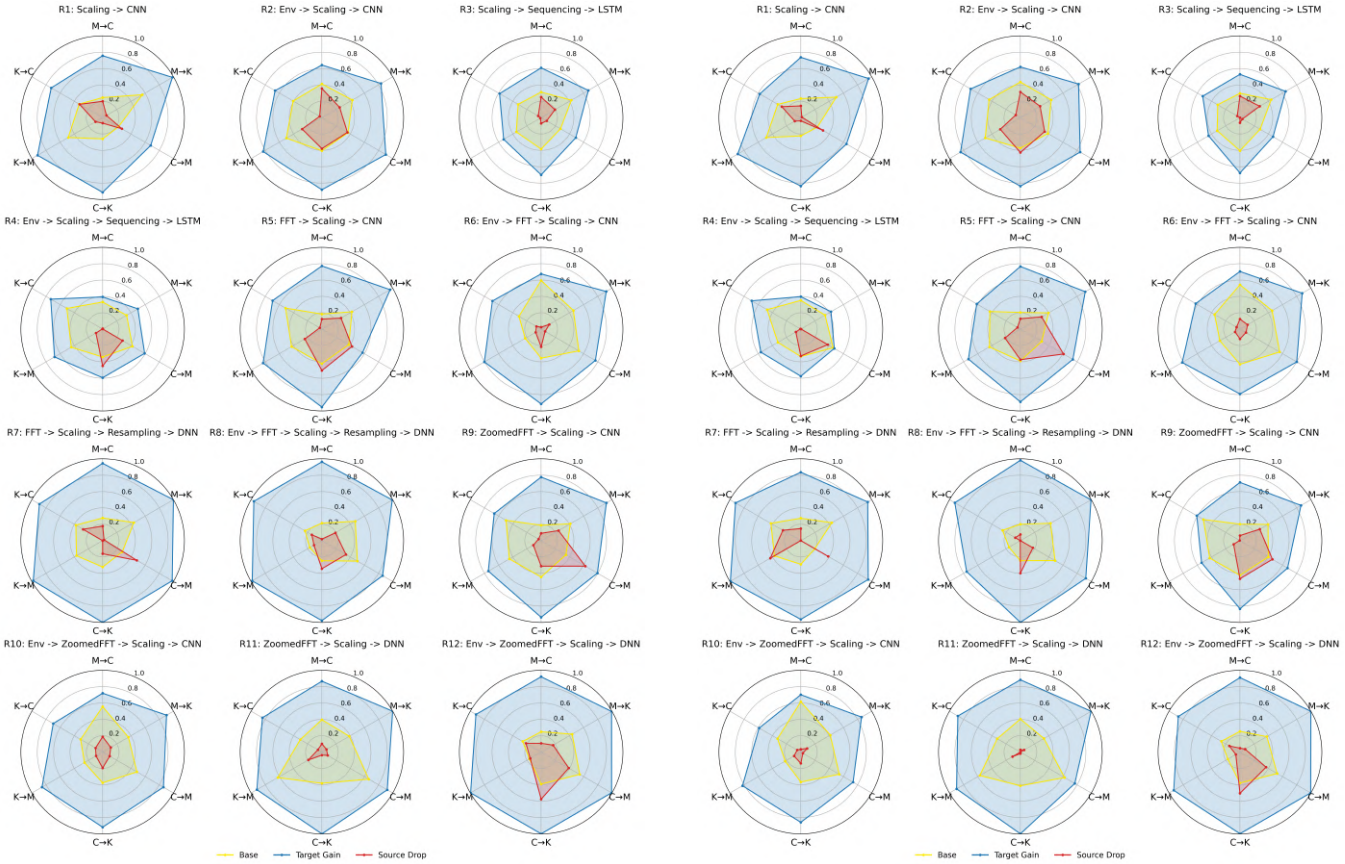
(b) 5%



(c) 25 shots per class

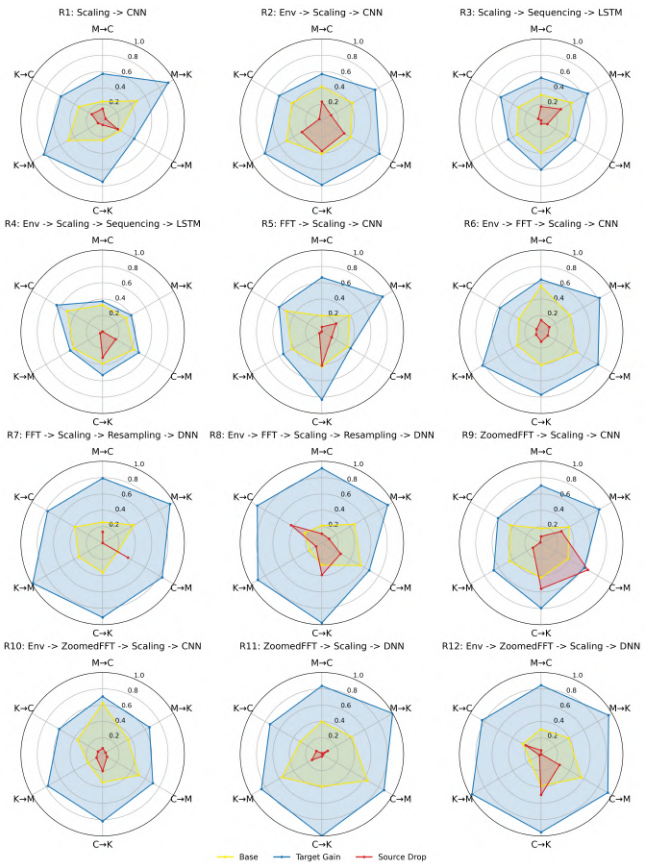
(d) 1%

Figure 3. Analysis of data efficiency across different subsampling strategies and percentages.

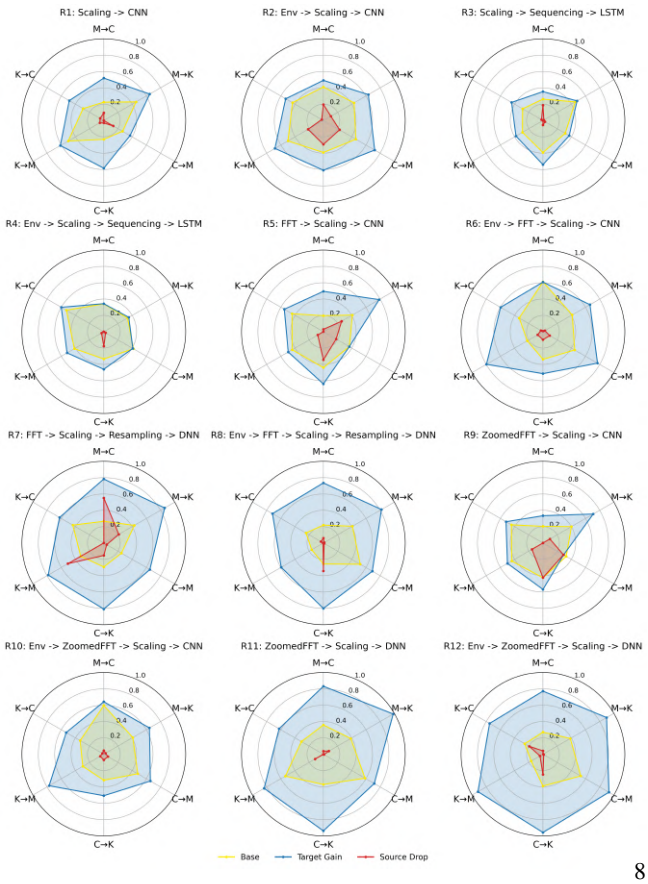


(a) 20 shots from class

(b) 15 shots from class



(c) 10 shots from class



(d) 5 shots from class

Figure 4. Robustness towards fine-tuning data scarcity (II)

additional domain adaptation step before pseudo-labeling) or replace the DB with a more sophisticated method that is capable of unlocking higher classification accuracies with stronger label noise presence.

Another interesting observation to make from the comparison of Figure 2 vs. Figure 5 is that although the fine-tuning classification is suboptimal, the source drop is noticeably lower. This observation suggests that fine-tuning using DB may be beneficial, even with ground-truth labels, by increasing the persistence of source set knowledge. To investigate this hypothesis, in Figure 5a, we summarize similar results for the case of fine-tuning over the whole training set and using the ground-truth results (similar to the first set of experiments whose results are summarized in Figure 2) but with DB as the fine-tuning method. In comparison with Figure 5a, fine-tuning performance (from the target-gain perspective) is fairly similar, except for $K \rightarrow C$ in R8 that fails to achieve perfect performance. On the other hand, from the source-drop perspective, generally, DB fine-tuning tends to reduce the vulnerability to catastrophic forgetting. To name a few examples, $M \rightarrow K$ in R4, $K \rightarrow M$ in R5, $C \rightarrow M$ in R12, all tend to have significantly lower source-drop in Figure 6 compared with Figure 2. This suggests that DB fine-tuning favors adaptation to the target domain without strongly learning target-specific features, thereby preserving source-domain knowledge.

6. CONCLUSION

Data insufficiency remains a primary obstacle to the real-world deployment of intelligent predictive maintenance solutions. While TL has emerged as a promising strategy to mitigate this scarcity, the critical role of input data presentation within the TL framework remains underexplored. The current study addresses this gap through a comparative analysis of different data processing pipelines. Our findings demonstrate that the choice of data representation significantly influences the efficacy of knowledge transfer, and the insights provided herein offer a roadmap for selecting the most promising routines to enhance the efficiency of TL applications in bearing diagnostics.

Furthermore, we showcase that fine-tuning over unlabeled data from the target set and pseudo-labels from the source model – as an imperfect oracle – can noticeably reduce the amount of ground-truth data required. By treating initial source predictions as noisy labels, this approach effectively eliminates the prohibitive costs associated with manual data labeling. Consequently, this methodology makes TL more accessible and affordable for industrial environments where unlabeled data is affordably collectible but expert annotation can be prohibitively expensive.

Different directions exist for future research; to start with, we encourage the application of similar comparative studies in other fields involving unstructured data, such as speech recognition and computer vision, where the role of the presentation is often overlooked. Additionally, while this study focused on the most prevalent signal processing techniques for bearing fault classification, future work should expand this diversity by incorporating time-frequency methods, such as Short-Time Fourier and Wavelet Transforms, or signal decomposition techniques like Empirical Mode Decomposition. Finally, research towards achieving competitive TL performance through pseudo-labels and learning from noisy labels at lower labeling budgets requires either improved pseudo-labeling approaches or more effective fine-tuning over erroneous labels approaches.

7. ACKNOWLEDGEMENTS

The work was carried out with support from The Knowledge Foundation and from Vinnova (Sweden's innovation agency) through the Vehicle Strategic Research and Innovation Programme, FFI.

REFERENCES

Arazo, E., Ortego, D., Albert, P., O'Connor, N., & McGuinness, K. (2019). Unsupervised label noise modeling and loss correction. In *International conference on machine learning* (pp. 312–321).

Berenji, A., Nowaczyk, S., & Taghiyarrenani, Z. (2023). Data-centric perspective on explainability versus performance trade-off. In *International symposium on intelligent data analysis* (pp. 42–54).

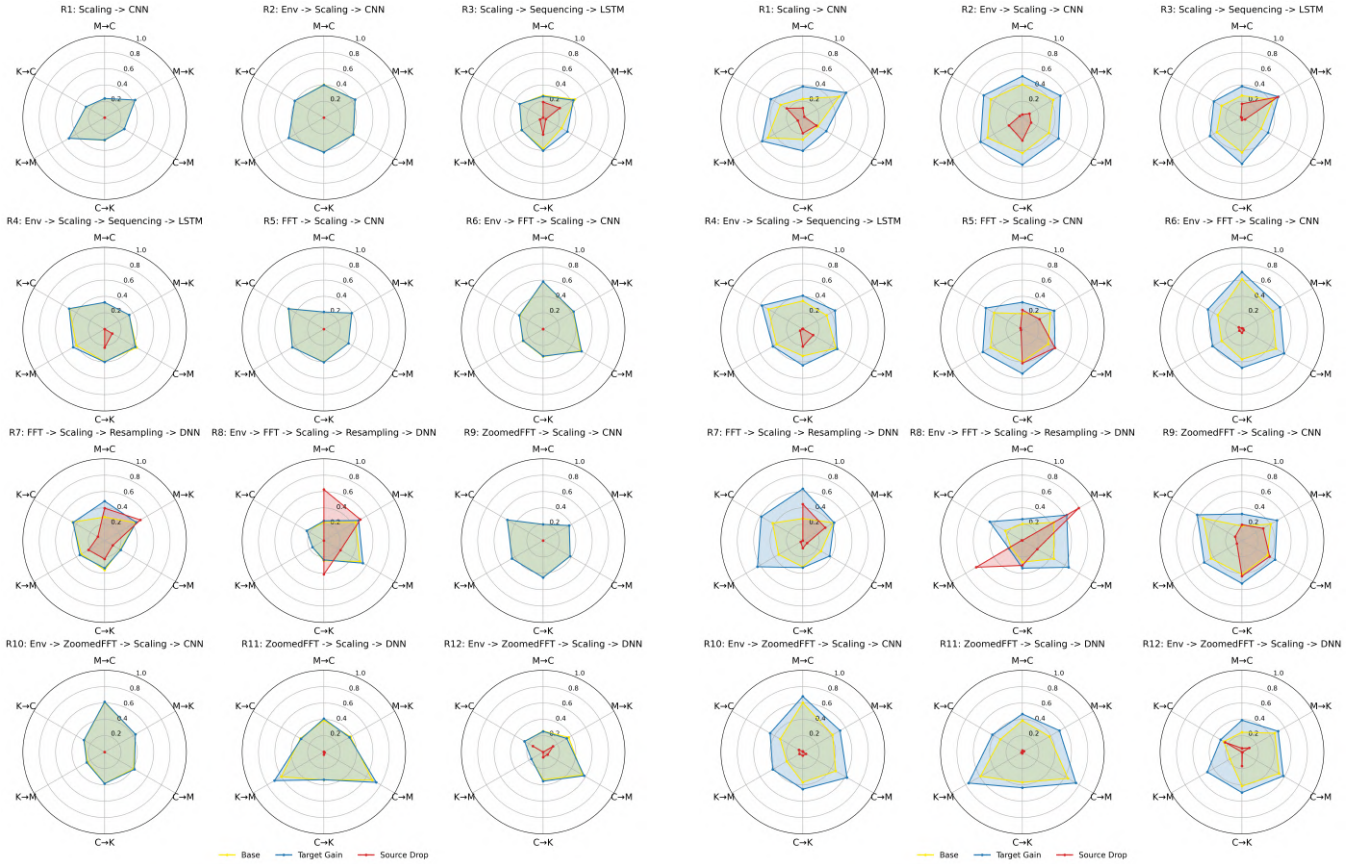
Berenji, A., & Taghiyarrenani, Z. (2022). An analysis of vibrations and currents for broken rotor bar detection in three-phase induction motors. In *Phm society european conference* (Vol. 7, pp. 43–48).

Carvalho, T. P., Soares, F. A., Vita, R., Francisco, R. d. P., Basto, J. P., & Alcalá, S. G. (2019). A systematic literature review of machine learning methods applied to predictive maintenance. *Computers & industrial engineering*, *137*, 106024.

Li, M., He, S., Chen, J., Feng, Y., & Xie, J. (2025). Label-smoothing dynamic decoupling augmented network for intelligent fault diagnosis under imbalanced data distribution with noisy labels. *Measurement*, 118664.

Liu, R., Yang, B., Zio, E., & Chen, X. (2018). Artificial intelligence for fault diagnosis of rotating machinery: A review. *Mechanical Systems and Signal Processing*, *108*, 33–47.

Lu, T., Yu, F., Han, B., & Wang, J. (2020). A generic intelli-



(a) Pure pseudo-labels

(b) 25% ground-truth label recovery

(c) 50% ground-truth label recovery

(d) 75% ground-truth label recovery

Figure 5. Fine-tuning on pseudo-labels from the source model

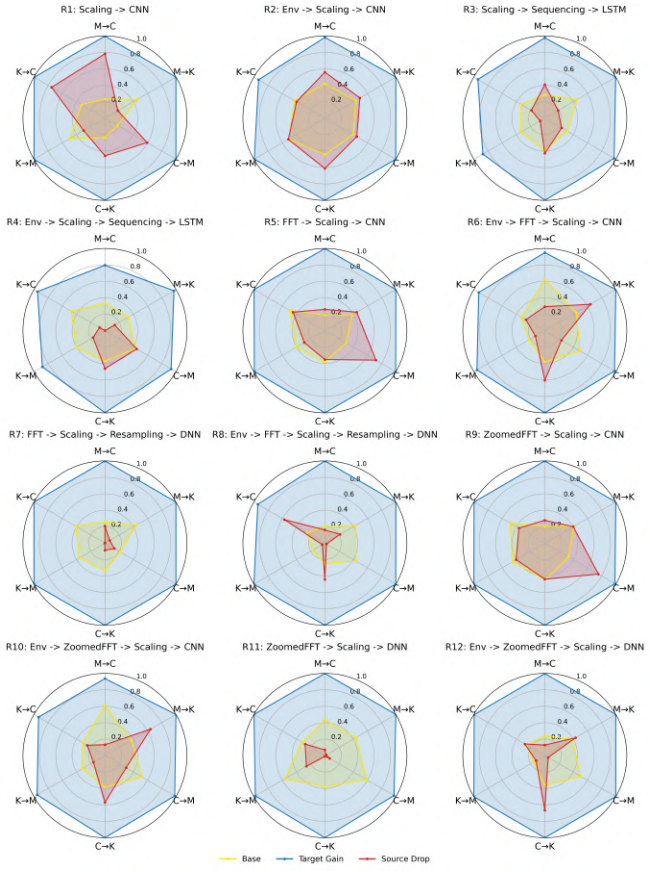


Figure 6. Fine-tuning results on ground-truth labels using DB

gent bearing fault diagnosis system using convolutional neural networks with transfer learning. *IEEE Access*, *8*, 164807-164814.

Lu, Y., Xu, Z., & He, W. (2023). Rethinking label refurbishment: model robustness under label noise. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 37, pp. 15000–15008).

Misbah, I., Lee, C. K., & Keung, K. L. (2024). Fault diagnosis in rotating machines based on transfer learning: Literature review. *Knowledge-Based Systems*, *283*, 111158.

Reed, S., Lee, H., Anguelov, D., Szegedy, C., Erhan, D., & Rabinovich, A. (2014). Training deep neural networks on noisy labels with bootstrapping. *arXiv preprint arXiv:1412.6596*.

Shao, S., McAleer, S., Yan, R., & Baldi, P. (2018). Highly accurate machine fault diagnosis using deep transfer learning. *IEEE transactions on industrial informatics*, *15*(4), 2446–2455.

Sukhbaatar, S., & Fergus, R. (2014). Learning from noisy labels with deep neural networks. *arXiv preprint arXiv:1406.2080*, *2*(3), 4.

Thomas, D., & Weiss, B. (2021). Maintenance costs and advanced maintenance techniques in manufacturing machinery: Survey and analysis. *International journal of prognostics and health management*, *12*(1), 10–36001.

Wang, S., Tian, J., Zhang, J., Yang, H., Yuan, X., & Liang, P. (2026). A dual model joint learning framework for intelligent fault diagnosis of rotating machinery under noisy labels. *Advanced Engineering Informatics*, *69*, 104024.

Zhao, B., Zhang, X., Zhan, Z., & Pang, S. (2020). Deep multi-scale convolutional transfer learning network: A novel method for intelligent fault diagnosis of rolling bearings under variable working conditions and domains. *Neurocomputing*, *407*, 24-38.

Zhao, Y., & Xu, J. (2024). Bearing fault diagnosis based on data missing and feature shift suppression strategy. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, *238*(7), 1193–1205.

Zhao, Z., Alzubaidi, L., Zhang, J., Duan, Y., & Gu, Y. (2024). A comparison review of transfer learning and self-supervised learning: Definitions, applications, advantages and limitations. *Expert Systems with Applications*, *242*, 122807.

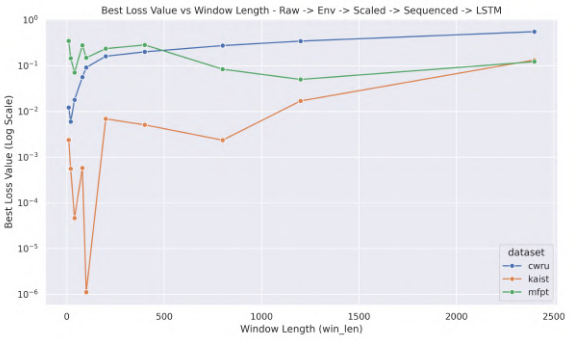
Zhong, J., Yang, Y., Mao, H., Qin, A., Li, X., & Tang, W. (2024). Contrastive regularization guided label refurbishment for fault diagnosis under label noise. *Advanced Engineering Informatics*, *61*, 102478.

A. THE EFFECT OF WINDOW LENGTH

As mentioned in section 4.1, to keep the LSTM models computationally affordable, we sequence the original signals before feeding them into the models; therefore, the choice of the right window length becomes an important hyperparameter. In Figure 7, the best (in combination with learning rate and batch size) validation-set loss function for various values of the window length is visualized; generally, both figures follow the same pattern, experiencing their lowest loss values at window lengths below or equal to 100, except the MFPT in R4, where the lowest loss value happens at the window length of 1200.

In Table 4, the optimum window length for combinations of routines and datasets is summarized.

(a) Best validation loss of R3 for different window lengths



(b) Best validation loss of R4 for different window lengths

Figure 7. Illustration of the best validation-set loss for different window lengths.

Table 4. Optimum window length for different combinations of datasets and routines

Dataset	Optimum window length (by routine)	
	R3	R4
MFPT	40	1200
CWRU	40	20
KAIST	80	100

B. NUMERICAL SUMMARY OF RADAR CHARTS

To quantify the performance comparisons we made between different routines under various circumstances, in this section, we summarize each figure as a table. We use the normalized area within each curve of those figures as the metric.

Table 5. Quantification of Figure 2

Routine	Base	Gain	Drop
1	0.1064	0.9987	0.2900
2	0.1826	0.9697	0.2622
3	0.1009	0.9132	0.0458
4	0.1615	0.8547	0.0743
5	0.1470	0.9968	0.1969
6	0.1524	0.9375	0.1403
7	0.1064	1.0000	0.0073
8	0.1002	1.0000	0.0238
9	0.1458	0.9991	0.2142
10	0.1579	0.9464	0.1254
11	0.2049	0.9789	0.0084
12	0.1207	0.9964	0.0801

Table 6. Quantification of Figure 3a

Routine	Base	Gain	Drop
1	0.1052	0.7789	0.0379
2	0.1786	0.7360	0.1112
3	0.1088	0.5134	0.0241
4	0.1525	0.4335	0.0138
5	0.1430	0.6528	0.0903
6	0.1716	0.7902	0.0511
7	0.1176	0.9401	0.0237
8	0.0950	0.9348	0.0555
9	0.1443	0.7387	0.1105
10	0.1623	0.7593	0.0329
11	0.1996	0.9319	0.0105
12	0.1132	0.9385	0.0593

Table 7. Quantification of Figure 3b

Routine	Base	Gain	Drop
1	0.1186	0.5571	0.0270
2	0.1978	0.6172	0.0967
3	0.1215	0.4419	0.0092
4	0.1581	0.3916	0.0151
5	0.1326	0.5751	0.0601
6	0.1775	0.7155	0.0169
7	0.1115	0.8402	0.0156
8	0.0956	0.8580	0.0471
9	0.1467	0.5812	0.0476
10	0.1689	0.6775	0.0257
11	0.2044	0.7940	0.0019
12	0.1196	0.9164	0.0066

Table 8. Quantification of Figure 3c

Routine	Base	Gain	Drop
1	0.1138	0.7590	0.0374
2	0.1820	0.6934	0.0971
3	0.1129	0.4621	0.0338
4	0.1604	0.3369	0.0246
5	0.1500	0.7293	0.0865
6	0.1624	0.6830	0.0105
7	0.1098	0.9598	0.0004
8	0.0922	0.9537	0.0293
9	0.1468	0.7448	0.1123
10	0.1463	0.6462	0.0112
11	0.2003	0.8856	0.0084
12	0.1212	0.9658	0.0766

Table 9. Quantification of Figure 3d

Routine	Base	Gain	Drop
1	0.0998	0.3103	0.0112
2	0.1870	0.3895	0.0417
3	0.1177	0.2079	0.0108
4	0.1604	0.2299	0.0040
5	0.1430	0.3517	0.0042
6	0.1691	0.4065	0.0038
7	0.1214	0.4802	0.0040
8	0.0967	0.5128	0.0221
9	0.1384	0.3793	0.0167
10	0.1627	0.3886	0.0036
11	0.1983	0.6417	0.0015
12	0.1128	0.6118	0.0704

Table 10. Quantification of Figure 4a

Routine	Base	Gain	Drop
1	0.1149	0.6825	0.0247
2	0.1766	0.6320	0.0736
3	0.1174	0.3522	0.0136
4	0.1568	0.3390	0.0283
5	0.1493	0.6256	0.0828
6	0.1732	0.6294	0.0075
7	0.1161	0.9348	0.0233
8	0.0992	0.9138	0.0404
9	0.1487	0.6505	0.0687
10	0.1515	0.6832	0.0164
11	0.2017	0.8552	0.0064
12	0.1224	0.9437	0.0756

Table 11. Quantification of Figure 4b

Routine	Base	Gain	Drop
1	0.1046	0.5912	0.0138
2	0.1830	0.6075	0.0814
3	0.1159	0.2944	0.1581
4	0.1430	0.2721	0.0249
5	0.1272	0.6018	0.0904
6	0.1929	0.5922	0.0083
7	0.1073	0.8737	0.0243
8	0.0906	0.8645	0.0127
9	0.1522	0.4942	0.0668
10	0.1724	0.5778	0.0049
11	0.1929	0.8143	0.0006
12	0.1117	0.9067	0.0435

Table 12. Quantification of Figure 4c

Routine	Base	Gain	Drop
1	0.1087	0.4528	0.0103
2	0.1854	0.5243	0.0522
3	0.1315	0.2975	0.1373
4	0.1590	0.2354	0.0116
5	0.1452	0.4067	0.0185
6	0.0176	0.5501	0.0101
7	0.1062	0.7676	< 0.0001
8	0.1071	0.7789	0.0433
9	0.1404	0.4896	0.1082
10	0.1583	0.5108	0.0078
11	0.2022	0.7809	0.0030
12	0.1243	0.8418	0.0263

Table 13. Quantification of Figure 4d

Routine	Base	Gain	Drop
1	0.1020	0.2835	0.0023
2	0.1912	0.3752	0.0307
3	0.1094	0.1798	0.0014
4	0.1550	0.2014	0.0017
5	0.1442	0.2954	0.2378
6	0.1568	0.4258	0.0036
7	0.1081	0.5612	0.0348
8	0.0955	0.5254	0.0015
9	0.1380	0.2212	0.0361
10	0.1640	0.3852	0.0020
11	0.1762	0.6706	0.0006
12	0.1157	0.7617	0.0049

Table 14. Quantification of Figure 5a

Routine	Base	Gain	Drop
1	0.1052	0.1050	0.0
2	0.1869	0.1859	< 0.0001
3	0.1076	0.1176	0.0118
4	0.1615	0.1673	0.0041
5	0.1449	0.1445	0.0
6	0.1721	0.1745	< 0.0001
7	0.1165	0.1471	0.0659
8	0.0976	0.1094	0.0900
9	0.1515	0.1514	0.0
10	0.1564	0.1605	< 0.0001
11	0.1952	0.2099	< 0.0001
12	0.1082	0.1089	0.0021

Table 15. Quantification of Figure 5b

Routine	Base	Gain	Drop
1	0.1070	0.2052	0.0166
2	0.1859	0.2960	0.0187
3	0.1141	0.1964	0.0185
4	0.1493	0.2147	0.0066
5	0.1330	0.2277	0.0625
6	0.1773	0.2843	0.0009
7	0.1135	0.2545	0.0281
8	0.0895	0.1777	0.0726
9	0.1472	0.2429	0.0669
10	0.1615	0.2784	0.0012
11	0.1918	0.3030	0.0001
12	0.1246	0.2154	0.0030

Table 16. Quantification of Figure 5c

Routine	Base	Gain	Drop
1	0.1052	0.4423	0.0259
2	0.2007	0.5085	0.0462
3	0.1116	0.4259	0.0283
4	0.1447	0.3905	0.0140
5	0.1323	0.4849	0.0670
6	0.1807	0.5256	0.0047
7	0.1153	0.6603	0.0003
8	0.0979	0.6000	0.0059
9	0.1466	0.4936	0.0909
10	0.1674	0.5217	0.0074
11	0.1901	0.6523	0.0014
12	0.1118	0.6324	0.0039

Table 17. Quantification of Figure 5d

Routine	Base	Gain	Drop
1	0.1195	0.7082	0.0362
2	0.1882	0.7361	0.0832
3	0.1095	0.7334	0.0330
4	0.1516	0.6267	0.0343
5	0.1492	0.7808	0.0805
6	0.1824	0.7673	0.0120
7	0.1098	0.9579	0.0061
8	0.0952	0.9219	0.0407
9	0.1483	0.7570	0.0980
10	0.1792	0.7550	0.0153
11	0.1842	0.9243	0.0048
12	0.1300	0.9541	0.0167

Table 18. Quantification of Figure 6

Routine	Base	Gain	Drop
1	0.0978	0.9981	0.2455
2	0.1907	0.9702	0.2526
3	0.1224	0.9246	0.0566
4	0.1523	0.8372	0.0597
5	0.1257	0.9962	0.1732
6	0.1750	0.9405	0.1120
7	0.1091	1.0000	0.0057
8	0.0870	0.9809	0.0296
9	0.1487	0.9949	0.1949
10	0.1720	0.9379	0.1019
11	0.1998	0.9835	0.0162
12	0.1063	0.9971	0.0433