Optimizing Data Training Quantity for Bearing Condition Monitoring

Ethan Wescoat
Department of Mechanical Engineering
Clemson University
Greenville, USA
0000-0003-2476-1368

Vinita Jansari
Department of Automotive Engineering
Clemson University
Greenville, USA
0000-0002-7175-2758

Laine Mears
Department of Automotive Engineering
Clemson University
Greenville, USA
0000-0002-4793-1406

2023 IEEE International Conference on Prognostics and Health Management (ICPHM) | 979-8-3503-4625-1/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICPHM57936.2023.10193864

***Abstract*—Prognostics Health Management (PHM) in manufacturing seeks to reduce the amount of unexpected downtime that inhibits manufacturing competitiveness. However, a common challenge for the manufacturing industry is the lack of known failure data to train a predictive classifier. This work optimizes the necessary quantity of required failure training data and healthy data for three different exemplar datasets by assessing classifier performance. Particle swarm optimization with penalty factors associated with the training data amount were used to identify the required training data amount for fault classification. Two separate analysis cases are considered: a binary classification and multi-class classification case termed the progressive case. In both analysis cases, the optimal training data depended on how separable the bearing data were between the different baseline and defect stages. In those instances where the differences in the data classes were apparent, the bearing data optimal training data amount was lower than in those instances where the data class differences were not present. Future work focuses on the investigation of these overlap cases to determine the best means for classifying progressive damage for remaining useful life calculations.**

I. Introduction

Prognostics Health Management (PHM) leverages condition monitoring (CM) systems to predict failures prior to disruptions along the production line. These systems generally comprise data acquisition, data processing, data analysis, and decision-making systems to determine when maintenance repair should be undertaken [1]. Hence, maintenance teams are able to make information-driven decisions related to their equipment to save time and money relative to the breakdown cost. However, some challenges with implementing CM systems stem from the imbalance or lack of known failure data available to train the data analysis and decision-making systems for PHM [2]. The lack of experimental or physical failure data is further exacerbated by the low quantity of labeled data generated from the manufacturing environment. One method to circumvent this problem is to employ purposefully generated datasets by creating failure data to represent critical components from the production line. The generated datasets can then train these CM systems before deployment to the production line without relying on data from the production environment. Currently, the quantity of data needed for effective training is subjective based on the collector and the available components.

For condition monitoring applications, there is generally not enough failure training data in condition monitoring systems [3]. The lack of training data can cause data imbalance and potential skewness in the analysis. In training predictive algorithms for condition monitoring, the amount of failure training data is increased through the use of resampling techniques [4], data augmentation [5], and transfer learning [6]. However, it is never really explored what the minimal required amount of training data for ensuring optimal classifier performance. This paper investigates algorithm performance while optimizing the training process to use a minimum amount of failure training data to achieve a high performing classifier.

In this work, particle swarm optimization (PSO) was used to optimize a cost function related to the algorithm performance based on the amount of failure data passed to the system. PSO is an optimization algorithm that uses a swarm behavior to find the optimal solutions to a problem [7]. In CM and PHM applications, PSO optimizes algorithms and models to detect potential equipment failure. Yang *et al.* [8] used the PSO technique to initialize and optimize the weights for neural networks [10] used to calculate remaining useful life for an ultrasonic motor. Lu *et al.* [9] used the PSO combined with chaos strategy and quantum behavior to optimize a deep belief network for detecting equipment faults.

PSO is selected as the optimization algorithm because of its computational efficiency to search the design space better than a stochastic gradient descent optimization. However, the PSO algorithm is susceptible to particles terminating at local minima and not reaching the global optimal point [11]. In an effort to decrease the possibility of particles stopping at local minima, successive iterations of the optimization analysis were implemented to ensure that the results obtained are repeatable with different particle initialization schemes. To determine the minimum training point, this paper uses six different classifiers (Naive Bayes [NB], Support Vector Machine [SVM], *k*-Nearest Neighbors [KNN], Multilayer Perceptron [MLP], Random Forest [RF], and Decision Tree [DT]) to determine classifier performance as the training data amount changes. These classifiers have been used in previous condition monitoring literature to determine component health [12]. By determining the minimum training data quantity, further analysis could inform what data quantity is needed to transfer from similar applications in transfer learning. The primary contributions of this paper investigate:

- The framework to determine the minimum training data quantity over changing operating conditions and bearing health rates.
- A comparative analysis of different classifiers and how the minimum training data quantity affects the classifier performance.

II. Methodology

A. Condition Based Monitoring Data

Three CM data sets were used to investigate the minimum failure training quantity needed to obtain the best algorithm performance for determining bearing health. Two of the datasets used are publicly available: the Case Western Reserve University (CWRU) Bearing Dataset ($https://engineering.case.edu/bearingdatacenter$) [13] and the NASA bearing dataset ($https://www.nasa.gov/content/prognostics-center-of-excellence-data-set-repository$) [14]. These offline datasets provide a method for verifying and validating CM models before deploying those models to an online application. Offline refers to equipment not on the production line, but rather in a testing environment, and online applications are production equipment examples. The CWRU dataset contains purposefully damaged bearing data where faults are seeded at specific sizes and placements along the bearing elements [15]. The NASA bearing data contains run-to-failure data from bearings under an overload test [16]. Both datasets, CWRU [17] and NASA [18], are used as methods for model verification and validation. The listed techniques utilized for these datasets (CWRU and NASA) involve, but are not limited to, Support Vector Machines, Neural Networks, K-Nearest Neighbors, and Deep Belief Networks.

The third dataset is collected from Clemson University (CU). The data contained in this dataset reflect different bearing failure modes, such as Fatigue, Contamination, and Brinelling across different bearing sizes. Purposeful fatigue data are created by engraving a small defect on the bearing raceway or rolling element. The engraving defect represents a spall that might occur due to bearing fatigue. This term ”spalling” refers to the chips or parts of the bearing surface removed from the bearing raceway during failure. For Brinelling (*i.e.*, indentation) defects, the data are generated by dropping the bearing from predetermined heights. This defect simulates improper handling or installation that might occur in practice. The contamination mode simulates the progressive accumulation of contamination elements that occurs during operation in unclean environments. The contamination particles eventually lead to spalling and speed up the bearing fatigue rate, causing early failure and degradation. For this analysis, the contamination data are used from the CU dataset to test a different failure mode than the fatigue case. Three different bearing sizes are collected with the dataset: 6205, 6206, and 6207 deep groove ball bearings. Collecting data for different sizes allows for investigating the variation in signal due to the bearing size. The CWRU dataset has different bearing sizes, but only tests the fatigue case. The collected data are primarily vibration data, placed in the vertical and horizontal configurations. The data are collected at a sample rate of 50 kS/s and account for a range of vibration up to 25g. The collection length is 3 minutes and tests last for at least an hour. There is a period of 7 minutes in between tests, allowing for up to 5 collections per hour. The test and collection time periods are determined to see if there are any changes to the bearing state after the damage is applied and running for a set period. The collection period of three minutes may seem excessive in relation to the collection periods for the NASA (1 second) and CWRU (20 seconds) data sets. However, this difference allows for monitoring and capturing potential operating changes introduced to simulate dynamically changing systems.

Each dataset splits the data and subsequently stores the data in different formats. The CWRU dataset structure is organized based on the sampling rate, the defect location, the damage size, and operating conditions. The NASA dataset structures are based on the test number and bearing number. The CU data structure is organized based on the operating parameters (speed and load), bearing size, failure mode, and damage quantity applied to represent a progressive failure. The different organization structures affect the labels attached to each dataset and the data used in this analysis. Table I contains the data used and the corresponding characteristics from each dataset that would influence data labelling. The data selected for each dataset are chosen based on the most complete data for all tests. Completeness refers to all the data being present for their respective columns. As an example, the CWRU dataset discusses collecting data from three locations during testing; yet, in some instances data are missing from the second and third locations. The 12 kHz Fan End Bearing Data had a complete set of data for the inner and ball defect and at least one of the outer race cases. This set also provided a different bearing size for comparison with the CU dataset. The NASA dataset considered a two-sensor configuration for test one, but not for test two or three. Hence, test two and three have been removed from this analysis. The CU data had additional data for some of their tests due to improper collection times. The extra data were therefore removed from the analysis. The data selection for each dataset ensured that different bearing sizes were tested in each case and at least two vibration sensors were available for each bearing tested.

B. Particle Swarm Optimization Structure

PSO was implemented to search the design space for determining minimum failure data quantity to train an effective

Bearing Dataset	Bearing Data Used	Bearing Operating Conditions	Bearing Sizes	Bearing Failure Modes	Possible Labels	# of Feature sets (Failure/Baseline)
CWRU	12kHz Bearing Data Fan End (Ball, Inner, Outer Race), only used the fan end and drive end vibration data	Load: 0 hp - 3 hp, Speed: 1730 - 1797 RPM	6203	Fatigue	Yes: operating condition, and damage level	3341/1696
NASA	Test 1 x and y data for all bearings	Load: 6000 lbf, Speed: 2000 RPM	2115	Fatigue	Yes: damage state	1087/7537
CU	Two stages of contamination data using the horizontal and vertical data	Load: 1049 N, Speed: 1690 RPM	6205, 6206, 6207	Contamination	Yes: damage level and bearing size	5608/12355

TABLE I
BEARING DATA BREAKDOWN PER DATASET

classifier based on the provided datasets. The optimization input parameters were the train-test split fraction for the failure data from each dataset. Equation 1 represents the cost function the optimization tries to maximize during each particle iteration. F_{alg} represents the classifier optimized by the particle. A set of different classifiers are trained to assess the variation in the optimization. T_{fn} represents the failure training data quantity used to train the classifier algorithm. The n refers to the number of types of failure data added to the optimization and the number of damage levels. CP refers to the classifier performance based on the cost function.

$$CP = F_{clf}(T_{fn}) \tag{1}$$

The possible failure training data fraction used in the cost function is constrained between 0.01 and 0.8. The constraints are set to ensure that some failure data are used in each training iteration and that some failure data are left out for the classifier test case. The classifier and training data are trying to maximize the classifier performance (CP), while minimizing the training data quantity. The CP is measured by the accuracy and the F1 score through a 5 fold cross-validation. The training accuracy was also calculated using a 5 fold cross validation to assess the training data. The minimum training data quantity, the Training Data Score (TDS), is measured based on the amount of possible available training data. Both the CP and the TDS are ranked in a hierarchy as shown in Table II. A secondary analysis only considered the algorithm metrics and did not include the TDS score. Table II contains a summary of the different parameters for the optimization structure.

Algorithm Metrics (Weighting)	Classifier	Inputs (constraints)
Train Accuracy (0.4), F1 (0.5), Accuracy (0.5), Training Data Score (0.6)	Decision Trees, Naive Bayes, Support Vector Machine, *K*-Nearest Neighbors, Multilayer Perceptron, Random Forest	T_{fn}: 0.01 - 0.8

TABLE II
OPTIMIZATION PARAMETERS

Two different cases were considered for labeling and organizing the data for classification: healthy versus damaged state and healthy versus progressive damage. Healthy data are when the equipment is operating in a normal or baseline state. The ”damaged” state refers to the data that are damaged through either fatigue or contamination. These data all have the same label, regardless of the damage level. The progressive damage case considers the damage level in the analysis. Damage level for the CWRU case refers to the change in size for the damage applied. The damage level change for CU data refers to the change in the amount of contamination added to the bearing. The NASA dataset was not considered for the progressive damage case, since there was not a clear point in the documentation for when the damage began.

Figure 1 shows the flowchart for the process optimization to determine the minimum training data quantity. The particles are initialized randomly based on the constraints listed in Table II, along with an initial velocity value. In the first case of healthy versus damaged, the training data either have a label of 0, which represents the baseline data, or 1, which represents the damaged data. For each classifier training, 80% of the healthy data are used, leaving the remaining 20% for the classifier test case. The healthy data are not optimized as these data are the majority class in online CM systems. The classifier is tested based on the remaining data and assessed using the algorithm metrics mentioned in Table II. The TDS is added to the score from the algorithm metrics to create the particle score. The ”best” particle for each iteration is determined based on the maximum cost value. The stopping criterion is assessed in each iteration. Two of the three criteria must be met: the number of iterations for the optimization, a certain number of particles have reached the threshold of the maximum score, or the maximum score has not changed significantly between different iterations. At least half of the particles need to reach within 95% of the maximum score to meet the stopping criterion. The tolerance change between the maximum score between iterations is $1e-5$. If these stopping criterion are not met, the particles are updated based on the local and swarm best position and then reevaluated using the cost function.

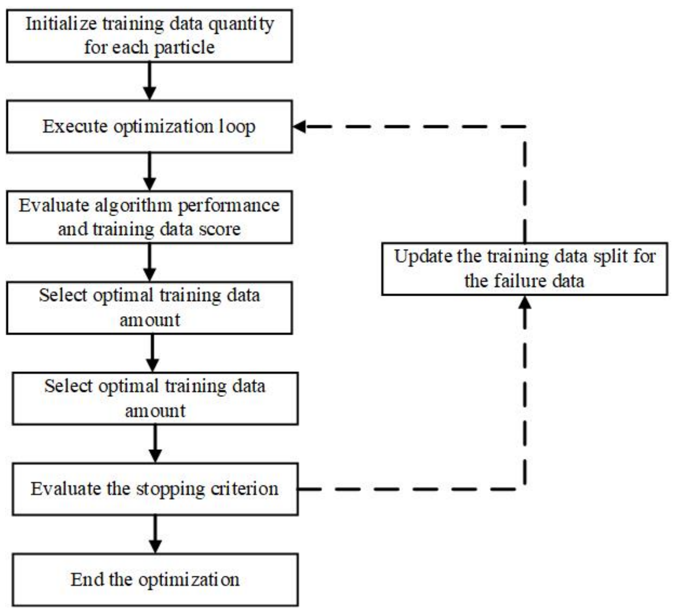

Initialize training data quantity for each particle
Execute optimization loop
Evaluate algorithm performance and training data score
Select optimal training data amount
Select optimal training data amount
Evaluate the stopping criterion
End the optimization
Update the training data split for the failure data

Fig. 1. Optimization process to determine minimum training data quantity

C. Classifiers

The classifiers used to optimize the training data are mentioned in Table II. Each of these algorithms have been used in prior condition monitoring literature: Decision Trees (DT) [4], Random Forest (RF) [19], *k*-Nearest Neighbors (KNN) [20], Support Vector Machine (SVM) [21], Naive Bayses (NB) [22], and Multilayer Perceptron (MLP) [23]. Each algorithm used comes from the Python scikit-learn library [24]. The training pipeline used the same configuration for each algorithm to try and minimize the variation between each algorithm. The features generated for the analysis were time-based based on the signal sampling rate and are: Root Mean Square, Kurtosis, Average, Variance, Skewness, Standard Deviation, Shape factor, Crest Factor, and Absolute Average. Based on the dataset quantity of baseline and failure data depicted in Table I, the datasets are split into considerations of small, medium, and large datasets.

III. Results

The healthy versus damaged case is referred to in the explanation of results as the binary case, and the healthy versus progressive damage case is referred to in the explanation as the progressive case. The healthy versus damaged case was assessed based on the analysis mentioned in Section II. Figures 2 through 3 show the comparison of the training features versus the cost function value based on the algorithm used. The algorithms generally performed along two trend lines. The 95% threshold represents the point where the cost function score has moved below the optimal point. From the CWRU dataset, each algorithm appeared to perform at a similar level, regardless of the bearing data quantity used for the failure data training split. The NB and SVM classifiers had the highest performance, followed by the KNN, MLP, RFC, and DT classifiers. Each algorithm was able to perform at the 95% threshold for the algorithm metrics at the low data quantity, where approximately 1% of the available failure data is used. For the first algorithm set, the classifiers cross the 95% threshold at 0.14 for the training data split. For the second algorithm set, the classifiers cross the 95% threshold at the 0.09 training data split.

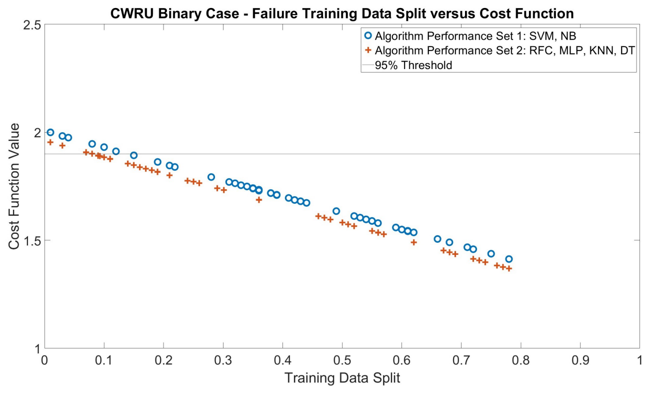


Fig. 2. CWRU binary case cost scores for all classifiers

The NASA dataset cost function values were more separable but generally followed one of two trend lines for Figure 3. Only the RFC, DT, and NB for the algorithm performance set 1 scoring performed close to the 95% threshold but did not necessarily cross the line and perform at an acceptable level. As the training data split decreased, the cost function value decreased linearly, implying that the classifier performance did not change as the training data increased. The other classifiers tested in Algorithm set 2 did not perform at an acceptable level close to the 95% threshold. The slight increase and maintaining cost function value until the 0.3 training data split implies that the classifier performance does increase as the training data increases.

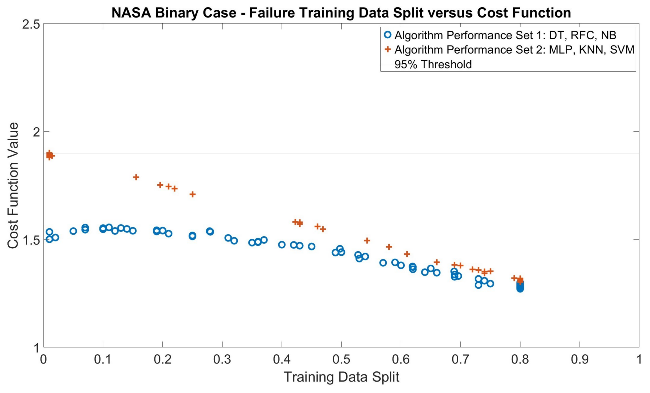


Fig. 3. NASA binary case cost scores for all classifiers

The CU dataset had a similar algorithm performance across the different training data cases shown in Figure 4. The algorithm performance was split into two sets of similar scores. Set 1 contained the KNN, NB, SVM, and DT classifiers, whereas set 2 contained the RF and MLP classifiers. Each algorithm performs above the 95% threshold in the minimum training data case. In each case, the algorithms perform above the 95% threshold and begin to decrease linearly, implying that the change in training data did not influence the algorithm performance. Interestingly, though, the performance is similar to the CWRU dataset, depsite the difference in failure modes.

It was thought that the minimum training data quantity analysis might skew the optimization performance. A secondary analysis was conducted with no minimum training data quantity assessed. The NASA dataset was the only data set that changed with respect to the amount of minimum training data needed in the algorithm, as shown in Figure 5. When assessing the CWRU and CU dataset, only a flat line was present which held true with respect to when the minimum training data quantity was used as part of the cost function analysis. In the NASA bearing dataset, algorithm set 1 did not cross the 95% threshold and remained steady as the amount of training data increased. However, algorithm set 2 did change as the training data set increased and eventually reached close to the algorithm set 1 performance.

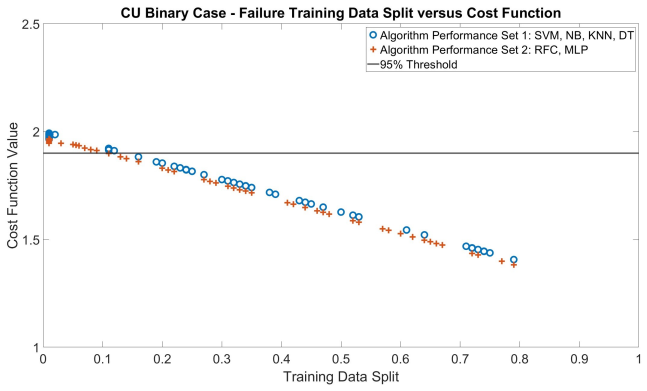


Fig. 4. CU binary case cost scores for all classifiers

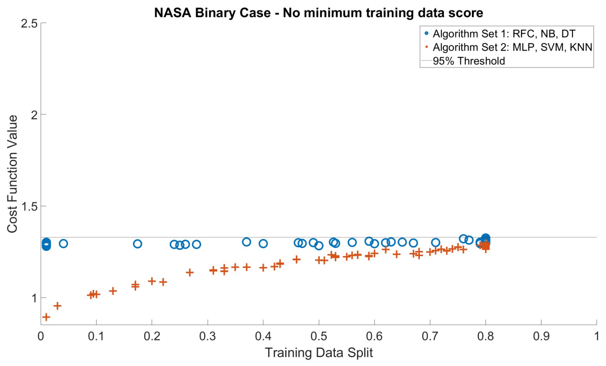


Fig. 5. NASA binary case cost scores for all classifiers without the minimum training case score

For the progressive analysis, the bearing data in the CWRU dataset was labeled based on increasing bearing damage size. The CU dataset was labeled the data based on the increasing contamination quantity in each bearing. The NASA data were not considered in the progressive damage case as progressive damage states were not determinable based on the documentation. For the CWRU case, there were three failure cases with respect to the corresponding damage size. Figure 6 represents the different training cases with respect to the different cost function scores. No discernible trend was determined except that the cost function value scores decreased in an expected trend that follows from Figure 2. The cost function value also decreased from the binary case implying that the algorithm performance also decreased with respect to the defect case, where the maximum threshold is 3.2. The increase in the threshold value stems from the additional failure cases. For the CWRU cases, three different training failure sets are optimized, corresponding to a different defect size. It is noted that the cost function value appears to vary even when the training data split remains steady. It is possible that the skewness could come from when other training data quantities are high, when the assessed point may be lower. As the training data quantity reaches the higher training data split, the variance in the cost function value is reduced.

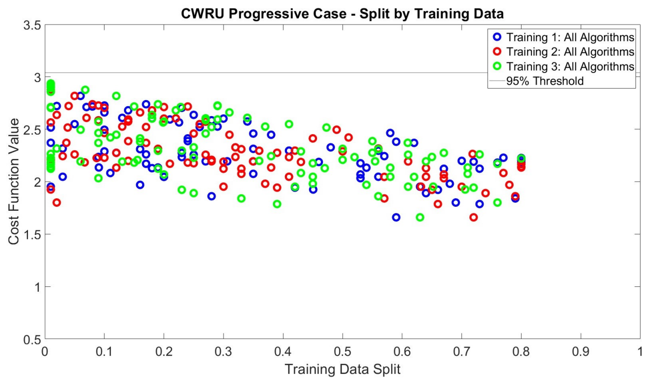


Fig. 6. CWRU progressive case cost scores for all classifiers with respect to the different training cases

Figure 7 represents the cost functions from each classifier for each contamination training case. The trend follows the expected response from Figure 4 and only crosses the 95% threshold when the algorithm performance is set at a lower interval. Training 1 refers to the initial contamination case, and Training 2 considers a later training case. Similarly to the CWRU dataset case, there is no discernible trend in whether the training data quantity used influences the cost function value. However, as the training data split increases, the dispersion of cost function values remains the same at each training split. It is possible from that observation that the training data quantity does not influence the progressive performance metrics analysis, much like the binary case.

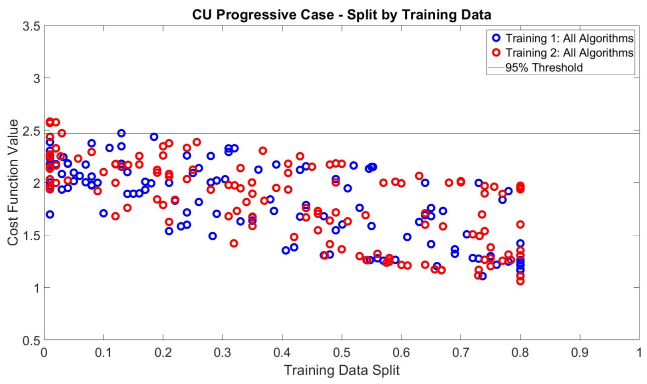


Fig. 7. CU progressive case cost scores for all classifiers with respect to the different training cases

IV. Discussion

Each dataset yielded different results for the minimum training data amount required to return an algorithm with a positive performance. For the CWRU and CU data in the binary case, the algorithm performance did not appear to change with the minimum training data quantity. In each algorithm case, the classifiers performed above the 95% threshold for all metrics at any training data fraction between 0.01 and 0.8 when not considering the minimum training data quantity in the algorithm performance. In the NASA dataset case, the DT, RF, and NB classifiers were the only classifiers that reached close to the 95% threshold. The imbalance ratio could have affected the results for the NASA classifiers, but Figure 2 and 4 imply and confirm that the minimum amount of training data did not appear to affect the algorithm performance even in instances when the failure data were heavily imbalanced. A possible reason could stem from the dataset collection practice. The CWRU and CU datasets are collected using purposeful damage, whereas the NASA dataset uses run-to-failure data. It is possible that the lead-up to the failure in the NASA dataset provides more overlap between the healthy and damage states. The purposeful damage nature of the other datasets may provide increased separability to the data between classes. Another issue could stem from improper feature labeling or documentation. For the NASA dataset, the fourth bearing has a failure that occurs, and a spike in the Kurtosis notes the damage. However, after some time, the data returns to a ”normal” Kurtosis level, as seen at the beginning of the operation. This phenomenon is noted by Qiu *et al.* [16] as the bearing ”re-healing factor” and is shown in Figure 8. This generated feature may cause class overlap since the feature after progressing damage is similar. However, this bearing data cannot be reclassified as healthy data due to the clear damage present at the end of the test. It does provide one instance where the failure data can overlap between damaged and healthy cases, causing additional misclassification and reduced performance. It should be noted that there was a decrease in algorithm performance in the CWRU case of progressive damage case. The possible decrease could come from lower separability between the different failure cases as damage increases. The separability is lower between the different failure modes, which may have skewed the classification results. Figure 9 demonstrates the overlap in the baseline data and the defect cases from the CWRU and the NASA datasets.

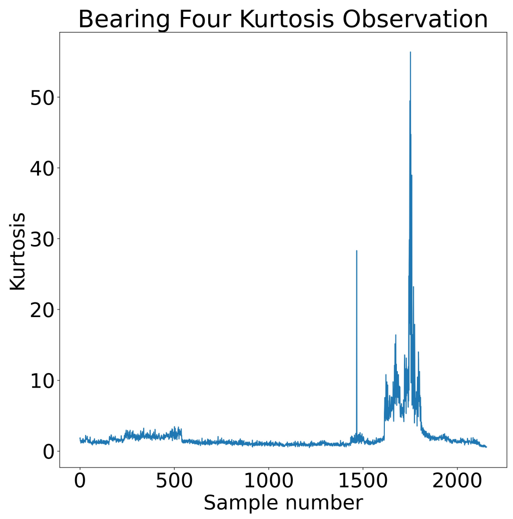
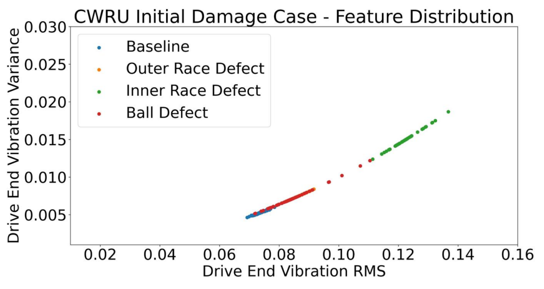


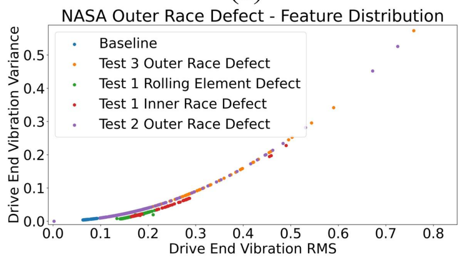
Fig. 8. Kurtosis observation for the Test 1 Bearing 4 exhibiting the rehealing factor

Incidentally, each dataset had a different order of performance for the classifiers tested in the binary case. Table III demonstrates the different classifier groupings ordered based on decreasing classifier performance for the binary case. Each set of algorithms is grouped into either set 1 or 2 for each case, as the algorithms appeared to follow one of two trend lines in each instance. Table III details the groupings of classifiers into either algorithm set 1 or algorithm set 2. Algorithm set 1 contains the classifiers that performed better overall. The reason the SVM and NB classifiers performed well could have been due to how well the data was separable and distributed within each class for the CWRU and CU datasets. A potential reason for the better performance in the NASA dataset in the RF and DT classifier comes from the growth of the decision tree to account for the possible variance in the bearing results. Oddly, the NB classifier was grouped into algorithm set 1 due to the variance in the feature distributions. Note that while the RF and DT are not in the top tier for the CWRU and CU dataset, these algorithms did perform above the 95% threshold based on the algorithm metrics of F1 and accuracy.

In the progressive case, the CWRU dataset and CU dataset both experienced an increased amount of variation with respect to their classifier performance. In either case, the classifier performance and cost function varied based on the amount of training failure data for the fatigue and contamination cases. This increased variation could stem from the decreased separability between the damage cases. In the binary case,



(a)



(b)

Fig. 9. (a) CWRU feature distribution for the initial defect case relative to the baseline data (b) NASA bearing dataset initial defect case

Algorithm Set	CWRU	NASA	CU
1	SVM, NB	DT, RFC, NB	SVM, NB, KNN, DT
2	RFC, MLP, KNN, DT	MLP, KNN, SVM	RFC, MLP

TABLE III
CLASSIFIER PERFORMANCE FOR THE BINARY CASE

these classifiers had a higher accuracy; however, with the progressive damage, the classes could experience more variation in terms of noise in the multi-class classification problem. In earlier work with the CU dataset, Manjunath *et al.* [25] found a significant overlap between the engraving and contamination cases for specific speeds and loads of bearing data. Those overlaps could be present in the progressive damage case, increasing the possibility of misclassification. For the CWRU dataset progressive case, the RF, DT, SVM, NB, and KNN classifiers performed with a similar distribution in the range of minimum and maximum cost functions from Figure 6. The decrease in classifier accuracy stems from the similarity in failure modes observed in Figure 9. For the CU classifier, the algorithm performance did not appear to vary with respect to the training data quantity change.

Incidentally, the minimum training data quantity for an effective classifier could depend more on the separability of bearing data features than the actual amount of bearing data available in particular classes. For example, it was initially thought that the CWRU data would perform poorer in the binary case due to the lower amount of failure training data per failure mode versus the NASA and CU datasets. However, the classifiers performed on par with the CU dataset, and the NASA dataset performed poorer. In the progressive analysis, the CWRU dataset performed poorer due to the overlap in the feature distribution. Hence, instead of framing the minimum training data quantity as a number, it should be considered more as a function related to the separability between the different health classes of the equipment.

V. CONCLUSION

This paper's novelty comes from the investigation into the minimum failure data fraction needed for effective training and testing of a condition monitoring algorithm based on the different sizes of bearing datasets. A particle swarm optimization algorithm was implemented to search the design space for the optimal split between train and test data. The swarm optimization was considered in both a binary (*i.e.* good/bad) case for all three datasets (CWRU, NASA, and CU) and a progressive damage case for the CWRU and CU datasets. A set of classifiers, listed in Table II, were used as the cost function to optimize the minimum failure data quantity for classifier training. During the optimization, the binary case for the CWRU and CU case found a minimum training data quantity of approximately 1% of the total failure data when using the full baseline data set. The low amount of training data needed is due to the separability of the features. For the NASA dataset, the algorithm performance did not meet the 95% case for any of the classifiers performed but came closest for the DT, RF, and NB classifiers. There was a decrease in classifier separability in the CWRU and CU progressive cases, indicating that there may be an overlap between the damage cases. As a result of the work, instead of an actual number related to the minimum training data quantity, a function is needed to describe the minimum training data quantity needed to train these classifiers. With the CWRU and CU datasets, the different training splits did not have an impact on the classifier performance in the binary case or progressive damage case. Instead, the minimum training data quantity needed should be described based on the separability between different classes. Future work should focus on how to accurately capture the separability between different bearing classes for classifier diagnosis. This separability quantification could then inspire the correct data and model transfer between different bearing applications. In addition, future work will consider additional data types (*e.g.,* temperature, ultrasound, or acoustic detection) and models, such as Long Short Term Memory networks for raw data, to supplement the learning provided by vibration analysis.

REFERENCES

[1] A. Jardine, D. Lin, and D. Banjevic, ”A review on machinery diagnostics and prognostics implementing condition-based maintenance,” *Mechanical Systems and Signal Processing*, vol. 20, issue 7, pp. 1483-1510.
[2] J. Leukel, J González, and M. Riekert, ”Adoption of machine learning technology for failure prediction in industrial maintenance: A systematic review,” *Journal of Manufacturing Systems*, vol. 61, pp. 87-96, 2021.
[3] G. Vogl, B. Weiss, and M. Helu, ”A review of diagnostics and prognostic capabilities and best practices for manufacturing,” *Journal of Intelligent Manufacturing*, vol. 30, pp. 79-95, 2019.

[4] Y. Zhang, X. Li, L. Gao, and L. Wang, L. Wen, "Imbalanced data fault diagnosis of rotating machinery using synthetic oversampling and feature learning," *Journal of Manufacturing Systems*, vol. 48, pp. 34 - 50, 2018

[5] X. Li, W. Zhang, Q. Ding, and J. Sun, "Intelligent rotating machinery fault diagnosis based on deep learning using data augmentation", *Journal of Intelligent Manufacturing*, vol. 31, pp. 433-452, 2020.

[6] L. Guo, Y. Lei, S. Xing, T. Yan, and N. Li, "Deep Convolutional Transfer Learning Network: A New Method for Intelligent Fault Diagnosis of Machines With Unlabeled Data," *IEEE Transactions on Industrial Electronics*, vol. 66, issue 9, pp. 7316-7325, 2019

[7] D. Wang, D. Tan, and L. Liu, "Particle swarm optimization algorithm: an overview," *Soft Computing*, vol. 22, issue 2, pp. 387-408, 2018.

[8] L. Yang, F. Wang, J. Zhang, and W. Ren, "Remaining useful life prediction of ultrasonic otor based on Elman neural network with improved particle swarm optimization," *Measurement*, vol. 143, issue 2019, pp. 27-38, 2019.

[9] L. Lu, Y. He, Y. Ruan, and W. Yuan, "Wind Turbine Planetary Gearbox Condition Monitoring Method Based on Wireless Sensor and Deep Learning Approach," *IEEE Transactions on Instrumentation and Measurement*, vol. 70, 2021.

[10] J. Elman, "Finding Structure in Time" *Cognitive Science*, vol. 14, pp. 179-121, 1990.

[11] Y. Shi, and R. C. Eberhart, "Empirical Study of particle swarm optimization,"*Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, Washington, DC, USA, 1999, pp. 1845-1950, Vol.3, doi: 10.1109/CEC.1999.785511.

[12] S. Lu, H. Chai, A. Sahoo, and B. T. Phung, "Partial Discharge Diagnostics Using Machine Learning Methods: A Comprehensive State-of-the-Art Review," *IEEE Transactions on Dielectrics and Electrical Insultation*, vol. 27, issue 6, pp. 1861-1888, 2020.

[13] Bearing Data Center. Case Western Reserve University. "Seeded Fault Test Data", Case Western Reserve University, Cleveland, Ohio.

[14] J. Lee, H. Qiu, G. Yu, J. Lin, and Rexnord Technical Services (2007). IMS, University of Cincinnat. "Bearing Data Set", NASA Prognostics Data Repository, NASA Ames Research Center, Moffett Field, CA.

[15] W. Smith and R. Randall, "Rolling element bearing diagnostics using the Case Western Reserve University data: A benchmark study," *Mechanicl Systems and Signal Processing*, vol. 64 issue 2015, 100-131, 2015.

[16] H. Qiu, J. Lee, J. Lin, and G. Yu, "Wavelet filter-based weak signature detection method and its application on rolling element bearing prognostics," *Journal of Sound and Vibration*, vol. 289, issue 4, pp. 1066-1090, 2006.

[17] D. Neupane and J. Seok,"Bearing Fault Detection and Diagnosis Using Case Western Reserve University Dataset with Deep Learning Approaches," *IEEE Access*, vol. 8, pp. 93155-93178, 2020.

[18] S. Zhang, S. Zhang, B. Wnag, and T. Habelter, "Deep Learning Algorithms for Bearing Fault Diagnostics - A comprehensive review," *IEEE Access*, vol. 8, pp. 29857-29881, 2020.

[19] S. Roy, S. Dey, and S. Chatterjee, "Autocorrelation Aided Random Forest Classifier-Based Bearing Fault Detection Framework," *IEEE Sensors Journal*, vol. 20, issue 18, pp. 10792 - 10800, 2020.

[20] M. S. Rathore, and S. P. Harsha, "Prognostic Analysis of High-Speed Cylindrical Roller Bearing Using Weibull Distribution and *k*-Nearest Neighbor," *Journal of Nondestructive Evaluation, Diagnostics and Prognostics of Engineering Systems*, vol. 5, issue 1, pp. 1-12, 2022.

[21] J. Saari, D. Strömbergsson, and A. Thomson, "Detection and identification of windmill bearing faults using a one-class support vector machine (SVM), *Measurement*, vol. 137, pp. 2817-301, 2019.

[22] N. Zhang, L. Wu, J. Yang, and Y. Guan, "Naive bayes bearing fault diagnosis based on enhanced indpendence of data," *Sensors*, vol. 18, issue 2, pp. 1-17, 2018.

[23] D. Kateris, D. Moshou, X. Pantazi, I. Gravalos, N. Sawalhi, and S. Loutridis, "A machine learning approach for the condition monitoring of rotating machinery," *Journal of Mechanical Science and Technology*, vol. 28, pp. 61-71, 2014.

[24] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825-2830, 2011.

[25] S. Manjunath, E. Wescoat, V. Jansar, M. Krugh, L. Mears, "Classification Analysis of Bearing Contrived Dataset under Different Levels of Contamination," *2022 IEEE Symposium on Software Reliability Engineering Worshop (ISSREW)*, Charlotte, NC, USA, 2022, pp. 387-393, doi: 10.1109/ISSREW55968.2022.00097.