This full text paper was peer-reviewed at the direction of IEEE Instrumentation and Measurement Society prior to the acceptance and publication

Automating Predictive Maintenance Using State-Based Transfer Learning and Ensemble Methods

Justin Larocque-Villiers
Department of Mechanical Engineering
University of Ottawa
Ottawa, Canada
jlaro099@uottawa.ca

Patrick Dumond
Department of Mechanical Engineering
University of Ottawa
Ottawa, Canada
pdumond@uottawa.ca

David Knox
Department of Electrical Engineering and Computer Science
University of Ottawa
Ottawa, Canada
dknox@uottawa.ca

2021 IEEE International Symposium on Robotic and Sensors Environments (ROSE) | 978-1-6654-4062-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/ROSE52750.2021.9611768

***Abstract*—The act of leveraging sensor data, such as accelerometers on connected industrial equipment, to diagnose and predict failures in roller element bearings is of growing interest in industrial and commercial applications. To do this, a large effort must be made to gather and process data, then apply and manage algorithms to make predictions and derive value. For these reasons, knowledge from previous models and tools to help in the evaluation and adaptation of new data sources using methods such as transfer learning and meta-learning have received significant attention in recent years. This paper aims to provide a methodology for combining such tools and presents a framework for an automated machine learning (AutoML) process by using a combination of pre-existing AutoML tools and new transfer learning methodologies. The framework uses recent observations in vibration analysis and deep learning architectures to abstract away many of the small, but intricate, decisions required for data preparation and model building. This allows for a unique data science strategy and dramatic reduction in the development time required to achieve competitive baseline models when compared to industry averages, while simultaneously improving the ability to address unseen environments. First, a signal processing engine is used to standardize the data, then fingerprinting and unsupervised learning is used to separate data into groups based on a machine's operating state. Meta-learning is then used to evaluate these states as tasks to be associated with a model architecture from a selection of two pretrained models for transfer learning, as well as one custom built model, based on the tasks themselves. In parallel, baseline models are generated using an existing AutoML library. Finally, the top performers from the existing AutoML models are combined with the transferred or built models in an ensemble method regime that uses a Support Vector Machine (SVM) to yield a final classification output. The framework is tested by performing intelligent fault detection on vibration data from roller element bearings of three separate publicly available datasets. The framework attained prediction results consistent with other studies in the field, while dramatically reducing the net amount of user interaction time and number of required user decisions. The system also presents the opportunity for self-serve data and model management, helping data scientists spend more time on the creation of new and unique machine learning applications.**

Keywords—AutoML, Unsupervised Learning, Signal Processing, Transfer Learning, Meta-Learning, Ensemble Learning

I. Introduction

As the need for data-driven decision making and prediction continues to grow in our society, so does the need to harness and make intelligent use of data. Sensor-driven predictive maintenance presents many challenges to this task because new machines being monitored often present unseen data, requiring fine tuning and adjustments to models and the data strategy used. Lack of labelled data and the rare occurrence of real faults lead to a difficult prototyping environment. Turning raw data into actionable insight is complex, time consuming, and costly. Those in charge of delivering these insights, data scientists, software engineers, or machine learning engineers, require a great deal of education and experience to perform well. The field of predictive maintenance involves leveraging sensor data to predict mechanical machine failures before they happen. Bearings account for approximately 40% of rotating machine failures, and these failures (inner race, outer race, and ball faults) can be predicted by analyzing the vibration signal derived from accelerometers [1]. In applications such as predictive maintenance, new and unseen operation states or environments often present unique challenges, requiring new models or manual data/algorithm level evaluations. These challenges have fueled the need for automation in the machine learning pipeline, and growth in automated machine learning (AutoML) processes, meta-learning, and transfer learning research. The process of automating machine learning covers a wide range of automation topics, including [2]:

- Data preprocessing;
- Feature extraction;
- Feature engineering;
- Combined algorithm selection and hyperparameter optimization (CASH); and
- Model and data deployment, monitoring and management.

Work in this area has already begun to achieve highly desirable results. Tuggener et al. demonstrate how AutoML can be used in predictive maintenance for the transport industry, by automatically generating a model that outperforms a team of three data scientists working for three weeks on the same task [3]. Li et al. combine multiple different datasets with

maintenance history to automatically learn rules and select models that result in railway predictive maintenance capabilities with very low false positive rates [4].

Many well documented open source AutoML frameworks exist and are constantly being improved. Auto-sklearn is a python based framework that primarily relies on Bayesian optimization [5]. Auto-sklearn also handles sparse and missing data, and automatically constructs ensembles. Auto-WEKA also relies on Bayesian optimization, but is built on Java [6]. This automated approach combines multiple search methods and two ensemble methods and was shown to provide competitive performance with standard optimization methods on the CIFAR-10 and MNIST datasets. TPOT is another python alternative that relies on evolutionary algorithms, essentially mimicking biology in order to breed the best algorithm for the given task [7]. There are numerous other frameworks, including: Devol (Python), MLBox ML-Plan, SmartML, Autostacker, AlphaD3M, OBOE, PMF, and VDS, but none of these alternatives offer specialization in predictive maintenance [8]. The majority of AutoML libraries do not support neural network architectures, but instead use feature extraction and select from a wide range of traditional machine learning models. Auto-Net is an extension that was added to sklearn for the purpose of expanding its functionality to neural networks. However, dramatic improvements and customizations can be made for applications in predictive maintenance. Most notably, no framework has addressed spectrograms as inputs. Commercial frameworks have also been introduced, including: DataRobot, H20.ai, Google AutoML, SAS Factory Miner, and IBM SPSS Modeler [8]. Although, the performance benefits of these commercial alternatives are often disputed when compared to open-source libraries. The value of these commercial products tends to revolve around friendly user interfaces and configurable dashboards.

This work focuses on developing and testing an AutoML framework for intelligent fault detection, meant to take raw vibration input data, and output a final prediction on the presence of a bearing fault. The data preprocessing and feature engineering is automated using an ingestion engine as described in *Section II. B*. The use of transfer learning techniques and automated model building are explained in *Section II. A*. The combination of these auto-generated models is presented as an ensemble learning output in *Section II. D*. These three sections represent the sub-modules of the proposed AutoML framework as shown in Figure 1. This system results in the ability to ingest vibration data (and associated fault labels) and automatically build and train models to output a fault prediction. It is hypothesized that strong performance can be achieved on a wider range of data using simple model architectures without human intervention or manual hyperparameter tuning. The framework also minimizes the need to consider a large number of parameters shown by previous studies as having a low impact on performance and focus attention, rather, on a few parameters that have a major impact: the size of the data segments, window size, data augmentation techniques, and fineness of the data distribution samples [9]. This may result in a process that provides reasonable performance and a significantly reduced development time, as the standard approach of persistently tweaking numerous hyperparameters often yields diminishing returns over time.

II. Methodology

The methodology used for this study uses a sequential progression of operations applied to the system input data. An overview of the framework is provided in Figure 2.

A. Datasets Used

The Case Western Reserve University (CWRU) dataset is widely used with studies on unsupervised learning, supervised learning, as well as ensemble and transfer learning tasks [10]. One aspect to note is that questions have been raised in recent studies regarding similarity between the train/test distributions [9]. As a result, different ways of splitting up the data have been proposed: a train/test split by horsepower of the machine, or a train/test split by fault severity. This work follows both these data split approaches. The Paderborn University (PbU) dataset is also a popular dataset used in many studies [11]. The unique aspect of this dataset is that it features both real and artificial faults and provides much needed diversity to the research community in deploying algorithms on naturally occurring faults. The train/test split in this work follows the same regime as that of the benchmark sources discussed in Section III. The MFPT dataset was collected on behalf of the Society for Machinery Failure Prevention Technology and features similar faults and operating speeds as the CWRU and PbU datasets. The MFPT dataset also features real world examples and features publications on both signal processing-based methods, as well as supervised and unsupervised methods [12]. Just like the PbU

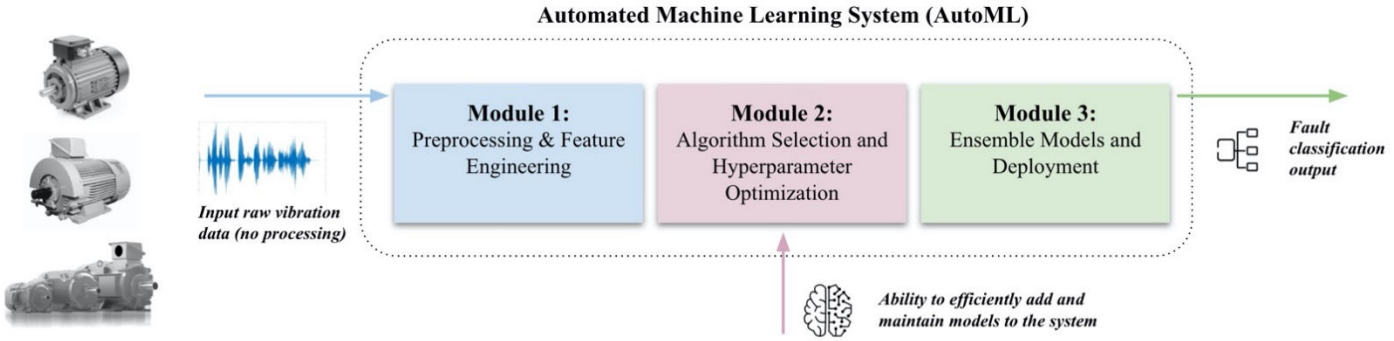

Figure 1: Overview of the proposed AutoML framework.

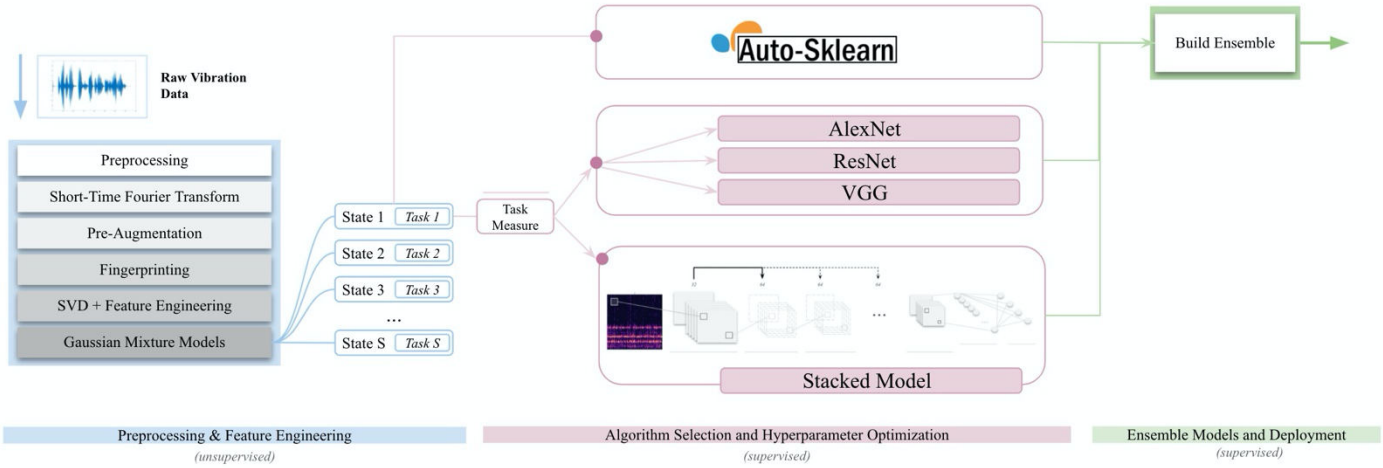

Figure 2: Proposed framework overview

dataset, the train/test split used in this study follows the benchmark sources.

B. Module 1: Preprocessing and Feature Engineering

The first part of the framework seeks to process the data and apply all transformations required to feed it to the subsequent algorithm selection and optimization processes. This section also serves to standardize the data and provide the user adjustable options, such as window size, overlap, specific filters being applied, etc.

Preprocessing, STFT, and Pre-Augmentation

The data is resampled at 10kHz, with outlier detection and zero mean then being applied. The short-time Fourier transform (STFT) is then computed on a full-length segment:

$$s_i(k,m) = \sum_{n=0}^{N-1} x_i(n) \cdot \omega(m-n) \cdot e^{-\frac{j2\pi kn}{K}} \qquad (1)$$

where the signal is represented as $x_i(n)$ and the window function, $\omega(m-n)$, can be either a rectangular window, Hanning window, or other. The window size of the STFT is adjustable by the user, but the program defaults to 256 and 50% overlap, resulting in an output of *(129,p),* where the length of the original sample dictates p. There will be a cutoff in which a segment could be abnormally long. In this case, the program will start a new process for it. Otherwise, the program will compute the STFT over the full sample, and then segment the spectrogram into square sizes of *(step size + 1)* with a total number of segments given by:

$$\left(\frac{p}{step\ size + 1} \times 2\right) + 1 \qquad (2)$$

Using floor rounding with the last end discarded, the spectrogram segments are then normalized and labelled before being stored.

Fingerprinting and Feature Engineering

With preprocessing, STFT, and pre-augmentation complete, the next step is to separate the data into related operating modes of the machine being observed. These individual groups of data are defined as tasks that the selection algorithm associates most closely with an existing model. The process of doing so involves constructing a fingerprint based on the spectrogram segment from the prior process and is shown in Figure 3. The fingerprint is constructed by extracting time-frequency peaks within the spectrogram. This strategy was chosen due to its robustness and consistency in dealing with noise and other problematic signals within the spectrogram [13]. To find the peaks in the spectrogram, a binary map of a local spectrogram neighbourhood is first derived, and then a combination of high pass filtering and image maxima is used to isolate and identify these peaks. Traditional audio-based methods would then use these peaks to form a constellation map used to match other incoming audio segments. However, for rotating machines, the nature of their operation results in the program sorting fingerprints by frequency-amplitude, rather than by frequency-time. This is due to its particular interest in monitoring the amplitude of peaks of an operating machine, whereas in most audio cases, this information is not as pertinent.

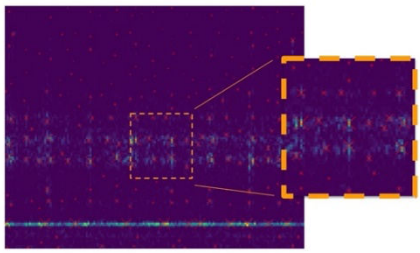

Figure 3: Fingerprints (red) identified on a spectrogram.

These peaks are valuable time-frequency data features that describe the operation of the machine. In traditional machine learning literature these descriptions are usually provided by statistical features such as root mean square, skewness, kurtosis, entropy and others, as covered by Pinedo-Sánchez et al [18]. For

the purpose of this study, time and frequency domain features are combined and leveraged to illustrate variation across principal components within classes as well as across datasets.

Gaussian Mixture Models

With a matrix of combined frequency peaks, as well as time and frequency features, Gaussian mixture models (GMMs) can then be used as a classification method to group similar task vectors together. They are governed by overlapping clusters with multivariate distributions given by [15]:

$$p(x) = \sum_{c} \pi_c \, \aleph(x; \mu_c, \sigma_c) \tag{3}$$

where μ_c is the mean and σ_c is the variance of a given cluster, denoted by the cluster number *c.* A multivariate Gaussian distribution describes each of these, and is given by:

$$\aleph(x; \mu, \beta) = \frac{1}{(2\pi)^{d/2}} |\Sigma|^{-1/2} \, exp\left\{-\frac{1}{2}(x-\mu)^T \cdot \Sigma^{-1} \cdot (x-\mu)\right\} \tag{4}$$

where Σ is a $n \times n$ covariance matrix. Each of these multivariate clusters represents the individual groupings of tasks. The question remains on how to apply this to the right number of clusters and segregate the right total number of tasks. A larger number of unique tasks would result in a finer division of data distribution. However, this would also lead to overfitting and bias when fed to a supervised learning algorithm, as less data volume and variation will hinder generalization. A common strategy is to leverage Expectation Maximization (EM) to initialize the number of clusters [16]. This process involves two steps that first computes the probabilities of belonging to a given cluster under fixed parameters, then involves computing the log probability of the data captured by a mixture model and increasing its fit on each iteration. A Bayesian Information Criterion (BIC) is introduced which penalizes the complexity as a negative log likelihood of the data. This results in the optimization of the number of starting clusters. The tasks are formed based on these Gaussian clusters, where each task represents a given state of data similarity in the system. The task data will eventually be sent to specific algorithms for training, as this granular understanding of the relative data distribution will improve the process of meta-learning. The system outputs a probability score for the data being classified. If scores for incoming data are found to be substantially low when compared to existing task groups, the system creates a new task for that data. Tasks with minimal amounts of data captures at the end of a system run will be grouped together, effectively acting as a "catch-all" in order to avoid over-complexity.

C. Module 2: Warm Start, Metalearning and Transfer Learning

Now that data is separated into task groups with similar features organized for each, the process can begin to allocate this data task group to a learning architecture and compute an output. Domain knowledge and previously trained tasks can be leveraged to influence the training of new tasks, which is the focus of meta-learning [17]. In the previous module, tasks were separated from within datasets based on their feature-vector similarity. In this module, the framework will automatically generate traditional machine learning models, as well as associate separated tasks to a set of models for transfer learning based on the data distribution in the corresponding datasets. First, the data confined to a specific state *S* is fed to auto-sklearn's library, which uses Bayesian Optimization to build a probability-based model that correlates hyperparameter options and resulting performance and uses aid models to guide the search space for future model parameters based on certainty of performance in given space areas. The resulting output is a range of up to 14 traditional machine learning models. Note that this method often fails to attain significant performance on complex tasks, but can often achieve desirable baselines and robust approximations due to the fact that it leverages multiple models. However, to address this issue, Kullback-Leibler (KL) divergence is used in parallel to measure the data similarity between two datasets and associates similar tasks to either an existing deep learning architecture to transfer, or to an iterative algorithm to develop a new architecture. The equation for KL divergence is given as:

$$D_{KL}(p(x)||q(x)) = \sum_{x \in X} p(x) ln \frac{p(x)}{q(x)} \tag{5}$$

where *p(x)* and *q(x)* are two distributions of the variable *x*.

Palanisamy et al. discuss transfer learning on large models such as ResNet and ImageNet, pretrained on the CIFAR dataset, with applications in audio [18]. Hendricks et al. show that hyperparameter selection tends to not result in large changes in performance when testing convolutional networks on spectrograms for predictive maintenance applications [9]. These two studies motivate the approach of selecting an existing architecture for transfer learning, or simplifying the building of a custom architecture. To assess whether a task is fit for transfer learning on a given model, tasks are mathematically summarized by the feature vector of *k* features defined in *Section II. B*. Previous tasks given by t_j are assessed for transfer to a new task t_{new}, based on the similarity and distance calculated between the two vectors using KL divergence. The task vectors are compared with three existing architectures: ResNet, ImageNet, and VGG. The model with the closest match is used with the data. The feature detection layers (convolutional layers) are frozen on all models, and the fully connected layer is retrained with a small and exponentially decaying learning rate. If the task vectors measure a large discrepancy between previous learning tasks, or the user otherwise errs on the side of caution and opts for more redundancy, an iterative stacked learner can be used to build a machine learning model automatically. This process relies on previous studies that have correlated the window size and CNN architecture with various outcomes [9][17][19]. These studies have influenced the development of this algorithm, which iteratively stacks small filters on top of each other, covered by a final max pool layer, to build an algorithm with strong precision and generalizability in an automated manner. The algorithm, shown in Figure 4, starts with the input of the task vector and compares itself to similar tasks from previous runs and fetches the number of layers that were successful in that task. Then, the algorithm adds or removes a small filter layer gradually and iteratively, with and without a bottleneck layer. This process allows for an extremely simple form of random search, in which many of the hyperparameter options are fixed,

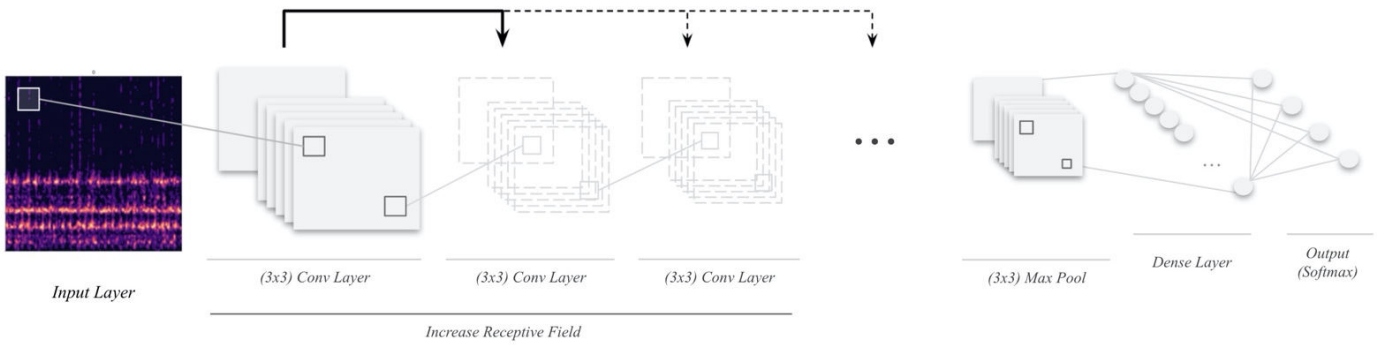

Figure 4: Visualization of the stacked algorithm process.

and the width and depth of the network are changed relative to the task vectors and previous runs. The net outcome of this process is an algorithm that can solve for model architecture based both on previous learning tasks, and on domain intuition by knowing which hyperparameters should be changed.

D. Module 3: Ensemble Methods

The module described in *Section II. C.,* results in two main outputs. The first is a series of traditional machine learning outputs coming from the auto-sklearn library, and second, outputs coming from the transfer learning or stacked based model. Together, these outputs form a committee of models. Combinations from within these can improve the performance when compared to any of the individual models on their own. Performance improvements are not necessarily limited to accuracy; ensemble methods have been shown to improve variance (usually using bagging), reduce bias (usually using boosting), and result in higher overall accuracy and more stable models (using stacking) [20]. A Support Vector Machine (SVM) is introduced, which learns to optimize the prediction combinations from each model in the committee in order to achieve an improved output prediction. This linear classifier improves the final output decision of the framework by taking, as input, the predictions of the traditional machine learning algorithms, as well as the predictions at each epoch of the transfer learning model and makes a combined decision. The final output of the SVM represents the system prediction. The SVM can also be replaced by taking the average of the algorithm prediction if the user prefers.

III. Results and Discussion

Testing was conducted using TensorBoard and Google Colaboratory, with K80 GPUs running on off-peak hours. Task specific performances were tracked using a naming convention (taksID-number of samples-…) appended with the final architecture (e.g. 3-layer3x3-bottleneck-16-FClayer-…) facilitating the tracking of results. In Figure 5, a visualization of the task separation (below) can be seen along with the BIC criterion for the optimization of number of clusters in the GMM (above). The CWRU dataset was only grouped into two main tasks based on fault severity, as opposed to machine operation. The PbU dataset was divided into more tasks based on operation mode and real versus artificial failures. This lends more weight to the theory that the CWRU dataset presents far less data variance than the other datasets. The MFPT dataset also features tasks with various operation modes. The algorithm favored grouping similar tasks by operating frequency of the machine, which may suggest that this parameter is of higher importance when categorizing machine vibration data; more important than other parameters such as load, torque, or other operating metrics. This has been previously addressed in the literature with solutions specializing in transient environment and non-stationary operating frequencies [21].

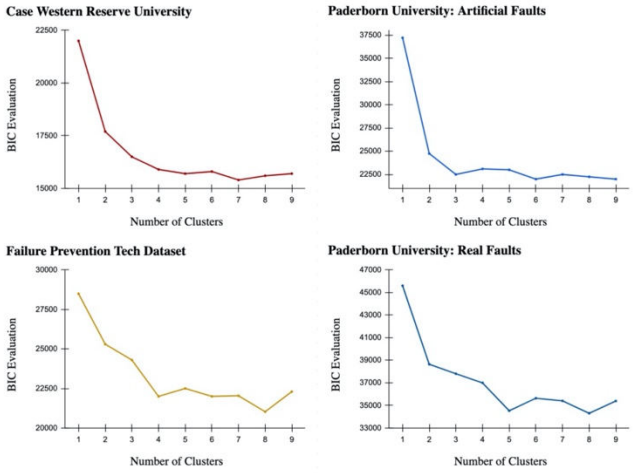

Figure 5: Number of clusters -task- optimization (above), and task association breakdown (below) from a single run.

The performances of subsequent associated algorithms when compared to the benchmarks are provided in Table 1. Two separate cases were used with the CWRU dataset. First, the train and test data was separated by horsepower, then by fault severity. The results were compared to a method developed by Zhang et al. using wide first layer kernels (WDCNN) in conjunction with a CNN architecture [22]. The PbU data results were compared to Pandhare et al. who used spectrograms as inputs for their proposed CNN architecture [23]. Similarly, the MFPT data results were compared to the work of Sun et al. using a Hilbert-Huang Transform (HTT) with a CNN [24]. In all cases, the automated method achieves performances within 10% of the target benchmarks. Note that large gains were made in development effort, as all datasets except for the MFPT dataset were conducted in less than 2 iterations (this dataset was ran a second time with larger spectrogram segments and overlap increased to 75%). This means that the user would run the

Table 1: Automated results compared to existing benchmarks.

Dataset	Target Method	Target Benchmark	Automated Result
Case Western Reserve University (Train/test split by horsepower)	WDCNN	90.9%	**84.3%**
Case Western Reserve University (Train/test split by fault severity)	WDCNN	53.1%	**62.5%**
Paderborn University (Artifical)	CNN	95.0%	**87.1%**
Paderborn University (Real)	CNN	61.9%	**71.0%**
MFPT	HHT + CNN	75.9%	**74.5%**

framework and return to view the results of the program. Required changes would be made before running the program again.

Most of the development time involved in the process was in downloading the datasets and providing them as inputs to the framework. Therefore, future efficiencies can be obtained by ensuring a standardized data format and process. One particular observation is that the framework was shown to provide competitive performance on real faults, as well as when data distributions are more challenging (this is not typically true of existing benchmark methods). This may be a result of the framework's ability to custom-configure an architecture that is more suitable to a given noisy environment without overfitting due to the constraint on hyperparameter selection. One of the challenges in the system is that the task separation in Module 1 may add unnecessary complexity if there is very large diversity in the data. It could result in frustrating troubleshooting and could invoke the reverse effect to what was intended. In this case, the user may now have to make decisions on task separation, which could lead to many unwanted train/test cycles. This problem subsides when there is a large amount of data and a large amount of variance, as more of the variance can be captured by the program with fewer clustered groups. Another option (which hasn't been attempted here) would be to use a hybrid system where the task separation is a mix of machine learning and manual driven solutions, where dimensional reduction and visualization tools are used to guide the user in decision making. In this work, the task separation was useful due to differences in the data, in which different model architectures were able to be treated differently. The task separation was useful in distinguishing between operation modes, which led to more customized model building, without the need for human intervention. For example, in the PbU dataset the operation modes were automatically distinguished, and the associated training model was customized accordingly. This, without the need for manual adjustments by a data scientist, allowed for the mitigation of validation error between training and testing datasets where there is potential for memorization and, therefore, lack of generalization.

A major problem in industry often occurs when lab data is used to train an algorithm with the expectation that performance will remain consistent on real faults in noisy environments. Adaptability is a rare trait when considering solution architecture. The proposed framework has proven its ability to be flexible in the customization of architectures based on new tasks, with the additional bonus of not having to redesign and reconfigure the parameters. The program performs well in separating tasks for the purpose of associating unique architectures to them, leading to stronger performance, and facilitating algorithm reuse. This reduces the friction in rolling out the system on unseen environments and avoids scenarios in which solutions must be redesigned. That being said, one of the potential drawbacks of the system is the event in which transfer learning results in worse performance than simple models, or negative transfer. This could potentially occur if the unseen data differs greatly from the established theory in both the transfer learning and stacked model making regime. This could also lead to wasted time and resources as models would be trained with suboptimal results. However, since the task association is conducted based on data similarity, with the machine operating frequency being a primary driver for this, there is opportunity for the user to gain deeper insight on the data in an organized and meaningful fashion. There is also an opportunity to implement a preprocess to conduct visual analysis of the data to guide a human in a judgement call to apply the system to a new application. Data and model management is also conveniently streamlined by the framework. This also leads to the opportunity for the framework to gradually train larger and more complex models by introducing them to their next nearest task neighbors.

IV. CONCLUSION

In conclusion, this framework dramatically simplifies the model building process by reducing the solution space and minimizing the need for many hyperparameters that would otherwise lead to multiple iterations of parameter tuning or model architecture redesign. Prediction accuracy results that are competitive with current benchmark methods are achieved by the system, with all but one dataset achieving these results within two iterations. By having the system automate redundancy for the user, it provides the user with the opportunity to prioritize their time for deciding on features such as window size, data segment size, filters, and data augmentation techniques, which have more impact on the outcome. Future work would involve leveraging this framework for generalized artificial intelligence. Using task association in the framework, nearest tasks can be introduced to models to gradually build larger more generalized architectures.

REFERENCES

[1] T. P. Carvalho, Fabrízzio A. A. M. N. Soares, R. Vita, R. D. P. Francisco, J. P. Basto, and S. G. S. Alcalá, "A systematic literature review of machine learning methods applied to predictive maintenance," Computers & Industrial Engineering, vol. 137, p. 106024, 2019.

[2] F. Hutter, L. Kotthoff, and J. Vanschoren, "Automated machine learning: methods, systems, challenges". Cham, Switzerland: Springer, 2019.

[3] L. Tuggener, M. Amirian, K. Rombach, S. Lorwald, A. Varlet, C. Westermann, and T. Stadelmann, "Automated Machine Learning in Practice: State of the Art and Recent Results," 2019 6th Swiss Conference on Data Science (SDS), 2019.

[4] H. Li, D. Parikh, Q. He, B. Qian, Z. Li, D. Fang, and A. Hampapur, "Improving rail network velocity: A machine learning approach to predictive maintenance," Transportation Research Part C: Emerging Technologies, vol. 45, pp. 17–26, 2014.

[5] M. Feurer, A. Klein, K. Eggensperger, J. T. Springenberg, M. Blum, and F. Hutter, "Auto-sklearn: Efficient and Robust Automated Machine Learning," Automated Machine Learning The Springer Series on Challenges in Machine Learning, pp. 113–134, 2019.

[6] L. Kotthoff, C. Thornton, H. H. Hoos, F. Hutter, and K. Leyton-Brown, "Auto-WEKA: Automatic Model Selection and Hyperparameter Optimization in WEKA," Automated Machine Learning The Springer Series on Challenges in Machine Learning, pp. 81–95, 2019.

[7] R. S. Olson and J. H. Moore, "TPOT: A Tree-Based Pipeline Optimization Tool for Automating Machine Learning," Automated Machine Learning The Springer Series on Challenges in Machine Learning, pp. 151–160, 2019.

[8] T. Nagarajah and G. Poravi, "A Review on Automated Machine Learning (AutoML) Systems," 2019 IEEE 5th International Conference for Convergence in Technology (I2CT), 2019.

[9] J. Hendriks, P. Dumond, D. Knox, "Towards More Realistic Domain Shift Benchmarking Using the CWRU Bearing Fault Dataset", 2021.

[10] "Bearing Data Center," Welcome to the Case Western Reserve University Bearing Data Center Website | Bearing Data Center. [Online]. Available: https://csegroups.case.edu/bearingdatacenter/pages/welcom e-case-western-reserve-university-bearing-data-center- website.

[11] "Konstruktions- und Antriebstechnik (KAt) - Data Sets and Download (Universität Paderborn)," Konstruktions- und Antriebstechnik (KAt) - Data Sets and Download (Universität Paderborn). [Online]. Available: https://mb.uni-paderborn.de/en/kat/main- research/datacenter/bearing-datacenter/data-sets-and- download.

[12] "Fault Data Sets," Society For Machinery Failure Prevention Technology, 27-Feb-2020. [Online]. Available: https://www.mfpt.org/fault-data-sets/#:~:text=A bearing fault dataset has,and three real-world faults.

[13] J. Larocque-Villiers, P. Dumond, "A New Vibration-Based Algorithm for Operational Machine State Identification", 27th International Congress on Sound and Vibration, Prague, 2021.

[14] L. A. Pinedo-Sánchez, D. A. Mercado-Ravell, and C. A. Carballo-Monsivais, "Vibration analysis in bearings for failure prevention using CNN," Journal of the Brazilian Society of Mechanical Sciences and Engineering, vol. 42, no. 12, 2020.

[15] X. Z. N. Z. a. K. G. J. Chen, "Fault detection for turbine engine disk using adaptive Gaussian mixture model," Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering, vol. 231, no. 10, p. 827–835, 2017.

[16] B. H. a. X. Y. Q. Jiang, "GMM and optimal principal components-based Bayesian method for multimode fault diagnosis," Computers & Chemical Engineering, vol. 84, p. 338–349, 2016.

[17] M. Huisman, J. N. V. Rijn, and A. Plaat, "A survey of deep meta-learning," Artificial Intelligence Review, 2021.

[18] K. Palanisamy, D. Singhania, A. Yao, "Rethinking CNN Models for Audio Classification", Department of Instrumentation and Control Engineering, National University of Singapore, Nov. 2020.

[19] J. Pons, T. Lidy, and X. Serra, "Experimenting with musically motivated convolutional neural networks," 2016 14th International Workshop on Content-Based Multimedia Indexing (CBMI), 2016.

[20] C. Zhang and Y. Ma, "Ensemble Machine Learning", Computational Intelligence and Complexity, Springer New York, 2012.

[21] Lughofer and James, Predictive Maintenance in Dynamic Systems. Springer International Publishing, 2019.

[22] W. Zhang, G. Peng, C. Li, Y. Chen, and Z. Zhang, "A New Deep Learning Model for Fault Diagnosis with Good Anti- Noise and Domain Adaptation Ability on Raw Vibration Signals," Sensors, vol. 17, no. 2, p. 425, Feb. 2017, doi: 10.3390/s17020425.

[23] V. Pandhare, J. Singh, and J. Lee, "Convolutional Neural Network Based Rolling-Element Bearing Fault Diagnosis for Naturally Occurring and Progressing Defects Using Time-Frequency Domain Features," 2019 Prognostics and System Health Management Conference (PHM-Paris), 2019.

[24] G. Sun, Y. Gao, K. Lin, and Y. Hu, "Fine-Grained Fault Diagnosis Method of Rolling Bearing Combining Multisynchrosqueezing Transform and Sparse Feature Coding Based on Dictionary Learning," Shock and Vibration, vol. 2019, pp. 1–13, 2019.