IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 15 May 2024, accepted 23 May 2024, date of publication 27 May 2024, date of current version 8 August 2024.

*Digital Object Identifier 10.1109/ACCESS.2024.3406262*

RESEARCH ARTICLE

# Unlocking Dual Utility: 1D-CNN for Milling Tool Health Assessment and Experimental Optimization

**GHAZAL FARHANI, SRIHARI KURUKURI, RYAN MYERS, NELSON SANTOS, AND MOHAMMED TAUHIDUZZAMAN**
National Research Council Canada, London, ON N6G 4X8, Canada

Corresponding author: Ghazal Farhani (Ghazal.Farhani@nrc-cnrc.gc.ca)

This work was supported by the National Research Council Canada.

**ABSTRACT** In a novel application of 1D Convolutional Neural Networks (1D-CNN), this study pioneers a tri-class classification framework for accurately forecasting the Remaining Useful Life (RUL) of milling tools. By harnessing the 1D-CNN's innate capability to analyze raw time-series data, we eliminate the traditional bottleneck of extensive feature engineering. Our model undergoes rigorous validation using a leave-one-out cross-validation method, catering to the constraints of a limited dataset. When optimized, the model delivers compelling performance metrics: average accuracy, precision, and recall scores stand at $0.90 \pm 0.02$, $0.85 \pm 0.12$, and $0.87 \pm 0.08$, respectively. What sets this work apart is its dual utility: not only does it excel in tool health assessment, but its output also serves as a diagnostic tool for experimental setups. For instance, anomalies detected in the model's predictions can act as early warnings for potential sensor malfunctions. Additionally, the model's performance metrics offer invaluable guidance in optimizing experimental parameters, such as choosing the most efficient sampling rate. In summary, this study not only establishes the robustness of 1D-CNNs in assessing milling tool health but also unveils their untapped potential as diagnostic aids for fine-tuning experimental setups.

**INDEX TERMS** 1D-CNN, RUL prediction, CNC machining.

## I. INTRODUCTION

Machining tool failures contribute to the surface roughness of a workpiece and might also inflict damage on the computer numerical control (CNC) machine. Given these risks, there is a serious need to introduce prognostic technology capable of estimating the remaining useful life (RUL) of the machining tool, both for economic and safety considerations. Generally, RUL prediction methods fall into three categories: experience-based, physics-based, and data-driven models [1], [2], [3].

Experience-based models leverage expert knowledge to establish rule-based prognostic systems. With access to historical data or events, observed scenarios can be correlated to RUL. While the rules in expert systems are usually straightforward and results are interpretable, these models may struggle when faced with multifaceted failure characteristics. As a result, when inputs and outputs grow in number, performance can degrade. Additionally, potential failure modes might go unnoticed in experience-based systems [1].

Physics-based models pivot around understanding system failure mechanisms, using this knowledge to mathematically describe the degradation and failure processes. These models, rooted in physical mechanics and fundamental principles, and are useful in simple systems. However, for more intricate systems where failure mechanisms remain unknown, they might not be the best fit [2], [3]. Every physics-based model is tailored to a specific system based on its unique characteristics, making it less versatile. Furthermore, model parameters often rely on empirical estimates from designed experiments, introducing a degree of uncertainty. While recent studies have sought to quantify and understand these uncertainties [2], [4], addressing them remains a challenge.

When the complexity of failure mechanics complicates RUL prediction in both experience-based and physics-based methods, data-driven models emerge as viable solutions.

The associate editor coordinating the review of this manuscript and approving it for publication was Jiajie Fan.

These models exclusively are built on data observations, aiming to discern statistical relationships between historically observed data and their corresponding RUL [5]. Machine learning (ML) techniques due to their data-driven nature, have recently seen a surge in their application for RUL prediction. Both conventional ML and advanced deep-learning approaches have demonstrated remarkable efficacy in this realm where these methods mostly rely on regression analysis for early failure detection [6], [7], [8], [9], [10]. Nevertheless, regression models frequently encounter fluctuating accuracy during the tool's operational lifespan, which can result in inaccurate predictions of its remaining useful life, and they cannot provide a direct measure of the health state of the milling tool [9], [11]. Therefore, this paper concentrates on a classification-based approach for its distinct advantages in early failure detection and in estimating the remaining life of the tool.

The main contribution of this study is implementing a lightweight 1D-CNN (one-dimensional convolutional neural network) model for multi-label classification, particularly for RUL prediction. We utilized time series data related to force from a CNC machine, which underwent initial preprocessing steps such as denoising and scaling before being inputted into our proposed model. Additionally, we created labels representing various states - healthy, transition, and failure - derived from tool wear assessments using laser beam measurements. As our model is lightweight, it is computationally fast and has fast inference time which makes it suitable for potential real-time implementation. From an experimental perspective, this represents a notable advancement as these labels (ground truth) are derived independently from the time series data. Another key contribution of our study is establishing a link between the outcomes of the trained model and practical experimental settings. For instance, anomalies identified in the model's predictions could serve as indicators of potential sensor issues. The paper is structured as follows:

Section II delves deeper into previous studies focused on RUL prediction using data-driven techniques, particularly highlighting the role of neural networks. Section III outlines the specifics of the 1D-CNN approach, describes our experimental setup, the nature of the raw data, and the preprocessing steps involved. Section V details the model training procedures and the approach to hyperparameter optimization. Section VI presents the outcomes of our training, discusses our findings, and assesses the model's accuracy. Finally, Section VII summarizes the entire research, offering insights into its importance and suggesting directions for future investigations.

## II. RELATED WORK

In recent years, there has been a marked increase in the use of ML models for predicting RUL of machinery. For instance, He et al. [6] combined empirical mode decomposition with a support vector machine to estimate RUL in aerial hydraulic systems. Similarly, Cheng et al. [7] employed multidimensional feature extraction from multi-sensor data, utilizing domains such as time, frequency, and time-frequency to train a multi-class support vector machine. Liu et al. [8] adopted multidimensional feature extraction along with a light gradient-boosting machine to evaluate tool wear values. Moreover, studies by [12] and [13] examined the degradation sensitivity of various features in RUL prediction over time, while [14] implemented a random forest model to better quantify the impact of thermal error in RUL prediction.

Additionally, Gebraeel et al. [15] developed an RUL prediction model using a feed-forward neural network for bearings, and Niu et al. [16] integrated a 1D-CNN with a long short-term memory (LSTM) algorithm for similar purposes. Notably, although deep learning models like CNN and LSTM are inherently adept at processing time series data without manual feature engineering, these studies primarily utilized conventional feature extraction methods. This underutilization highlights a missed opportunity to harness CNN and LSTM models to fully capture temporal dependencies in time series data. Hence, [17] implemented a BiLSTM model that directly processed time series data from a CNC machine, achieving a root mean squared error of less than 10%.

An et al. [10] introduced a novel hybrid model composed of CNN, bi-directional, and uni-directional LSTM layers (SUBLSTM). This model began with a CNN layer for local feature extraction, followed by BiLSTM for encoding temporal information, and concluded with multiple fully connected layers to enhance the output's complexity. Designed for regression tasks, this model attained up to 90% accuracy, significantly improving early-failure prediction compared to traditional ML models. However, the definition of 'failure mode' was solely based on the data collected, without independent validation measures. Wang et al. [11] utilized a recurrent neural network, instead of an LSTM, to estimate RUL of tools, emphasizing the integration of uncertainty in predictions. Zhang et al. [18] also employed a CNN model, using a Wiener process to detect tool wear and establish different degradation stages corresponding to the CNN output.

In another approach, several studies incorporated an attention layer within their network architecture to boost efficiency [19], [20], [21], [22]. The attention mechanism helps allocate computational resources more effectively, addressing information overload challenges in LSTM architectures. These models showed promising performance in predicting the RUL of milling tools, although their regression results demonstrated oscillations around true values.

Sun et al. [9] employed autoencoders, which excel in reducing data dimensionality. After initially training a neural network model with data from one tool, they applied transfer learning to adapt another tool using the optimized weights from the first. Despite its satisfactory performance, this method was restricted to just one additional tool. A significant benefit of this approach was the elimination of the need

for target labels. Concurrently, Li et al. [13] retrained a model named WearNet to estimate the wear states of workpiece surfaces and correlate them to the RUL of the milling tool. This method showed minimal uncertainty of 5%, but the correlation of workpiece wear to RUL was determined heuristically. Another innovative study by [13] applied a transfer learning approach to detect failure modes, recommending the model for very limited data sets.

These studies collectively highlight the effectiveness of neural networks in predicting the future state of milling tools, though they fail to directly identify the health state of the tools and exhibit oscillations around the ground truth, complicating the identification of tool health. Additionally, they typically rely on training deep networks with large time series data, making them computationally expensive.

In our current study, with access to independent measurements of the milling tool after each pass, we explore the potential of developing a 1D-CNN for classification tasks, as well as the feasibility of creating lightweight neural networks for faster, more stable training. In the following section, we explain the rationale behind our choice of the proposed neural network.

## III. METHODS AND MATERIAL

### A. 1D-CNN

Convolutional Neural Networks (CNNs) have emerged as a pivotal tool in pattern recognition [23]. Originally designed for 2-dimensional data and widely used in image classification and object detection, 1D-CNNs are adaptations tailored for 1-dimensional data such as time series, offering notable computational benefits [24]. A distinct attribute of 1D-CNNs is their adeptness at extracting salient features, enabling accurate predictions across a spectrum of applications, from industrial equipment malfunction detection to human motion analysis [25], [26], [27], [28]. Their streamlined parameter configuration not only facilitates real-time model training but also minimizes computational overhead. In the context of 1D-CNNs, when an input data vector $x$ of length $N$ is convolved with a kernel vector $k$ of length $L$, the convolution for layer $l$ can be articulated as [29]:

$$c_j^l = f\left(\sum_{i=1}^{L-1} x_i^{l-1} * k_{ij}^l + b_j^l\right) \tag{1}$$

In this context, $x_i^{l-1}$ represents the input, $c_j^l$ denotes the output, $k_{ij}^l$ signifies an element of the kernel, and $b_j^l$ is the bias value. The nonlinear function $f$ serves as the activation function, which acts upon the accumulated input values multiplied by the kernel weights (see Fig1). In the 1D-CNN framework, the convolution layer's output undergoes a max pooling process, with a kernel window $u$ of length $m$ being stridden through.

In the suggested architecture, each convolution layer is succeeded by a pooling process. This process serves as a deterministic function aimed at downsizing the convolution process's output dimension. This could manifest as average pooling, where the average value on the feature map is computed, or maximum pooling, which identifies the feature map's peak values. The optimization procedure, rooted in back-propagation, aligns with other DL network practices.

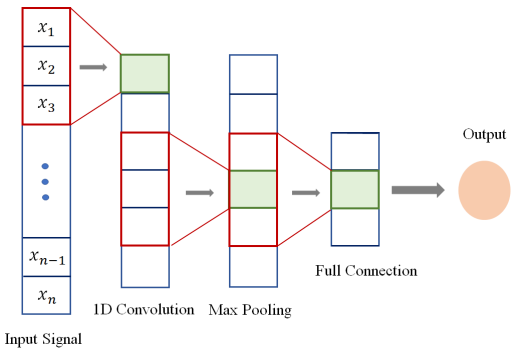



**FIGURE 1.** Schematics of a simple 1D-CNN network.

### B. EVALUATION OF MODELS

To gauge the efficacy of the classifiers, we employed the confusion matrix. For a binary classifier, the confusion matrix provides insights into [30]:

**True positives (TP)**: Instances correctly identified as positive.
**False positives (FP)**: Instances wrongly identified as positive.
**True negatives (TN)**: Instances accurately identified as negative.
**False negatives (FN)**: Instances incorrectly identified as negative.

This framework can be extended to accommodate multi-label classifiers. To delve deeper into model assessment, we determined accuracy, precision, recall, and F1 score. These metrics are defined as [30]:

$$\text{Accuracy} = \frac{\text{TP+TN}}{\text{TP+TN+FP+FN}}$$
$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}}, \text{Recall} = \frac{\text{TP}}{\text{TP+FN}}$$
$$\text{F1 score} = \frac{\text{2*Precision*Recall}}{\text{Precision+Recall}}$$

We applied these metrics to appraise the models corresponding to each tool.

## IV. EXPERIMENTAL STUDY

### A. EXPERIMENTAL SETUP

The data for this investigation was sourced from a 5 axis HURON CNC milling located at the National Research Council's Automotive and Surface Transportation laboratory at London Ontario. Fig 2 illustrates the dry milling procedure. As depicted in the figure, the milling machine sweeps the surface in three dimensions (Axes $x, y$, and $z$). Air jets serve to regulate the temperature during milling. In this

**TABLE 1. The key parameters for the dataset is summarized.**

| | |
|---|---|
| CNC machine | HURON K3X8Five milling machine (Siemens 840D controller) |
| Tool diameter, mm | 20 |
| No of inserts | 2 |
| Insert type | Kennametal ADKT103504PDERLCKC725M |
| Material | Stainless Steel 4130 |
| Coolant | Chilled air jet |
| Cutting speed, m/min | 250 |
| Feed rate, mm/min | 915 |
| Axial depth of cut, mm | 1 |
| Radial depth of cut | 50% of tool diameter |
| Tool number (Experiment no) | 1, 2, 3, 4, and 5 |
| No of layers of each experiment | 14, 14, 08, 13, and 14 respectively |

operation, the tool advances along the y-axis, completing one pass, and then progresses along the x-axis over the surface with 51% radial depth until one layer is fully milled (11 passes). Following this, the tool's wear is assessed using a in-machine laser probe (Renishaw Bloom sensor) at a 0.8 mm distance from the tip of the tool. The milling process is then repeated until there are visible indicators, such as burning chips, suggesting tool wear. In Figure 3, images of the milling tool in both its pristine and deteriorated conditions are presented. This cycle persists for each tool until clear signs of tool degradation are evident. For this study, the milling machine operated at a cutting speed of 3979 rotations per minute (RPM) and a cut depth of 1 mm, while maintaining a cutting velocity of 250 m/min, using a two-flute cutter insert of type Kennametal ADKT103504PDERLCKC725M on a 20mm diameter shank. Feed rate was kept constant at 0.115mm/tooth. The workpieces used was 4030 stainless steel, sized 150 mm length and 110 mm wide. Of note, Kistler Type 9255C three component dynamometer was used to collect the cutting forces. The charges from the dynamometer was converted in to voltage using Kistler LabAmp Type 5167A charge amplifier. Voltage signal from charge amplifier was captured using a NI9215 card on a NI cDAQ-9174 chassis and was recorded in to a computer at 2000Hz sampling rate using NI-LabVIEW. For our analysis, we selected five sets of unique tool inserts and monitored them until the point of complete wear when burning chips were observed. A summery related to key parameters of the data is provided in Table 1

### B. DATA DESCRIPTION

The dataset was captured at a sampling rate of 2000 Hz for each layer and was recorded using LabView Software, saved in the `.lvm` file format. Using the Python `pandas` library, we aggregated all the layers into a single data frame for each tool. Each data frame is structured with five columns: time step, force measurements $F_x$, $F_y$, and $F_z$, along with the corresponding layer number. Of note, each layer encompasses 11 passes. The transitions between passes are identified by the milling tool's absence from the surface, which is indicated by zero force readings, thereby distinguishing the end of one pass and the start of another. For our analysis, we selected five unique tools and monitored them until the point of complete wear. Figure 4 showcases the raw $F_x$ data for a singular layer, specifically associated with

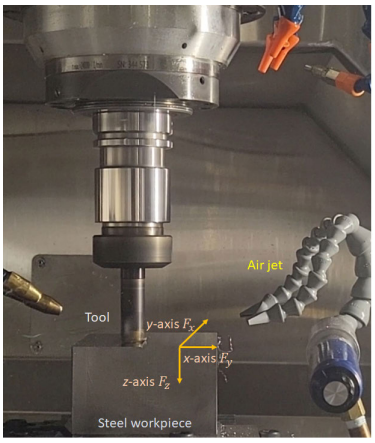



**FIGURE 2. Image of tool and work piece.**

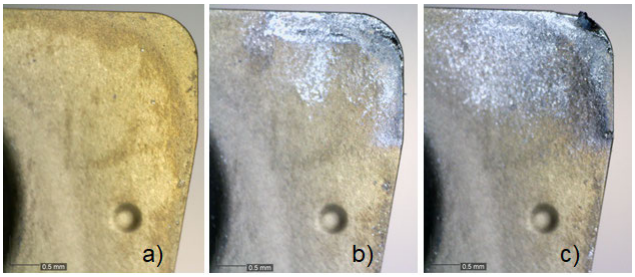



**FIGURE 3. The visualization of the milling tool is presented as follows: Panel a) displays the new tool, Panel b) illustrates the tool after it has worn out (following six layers of tooling), and Panel c) shows a tool that is completely worn out. Notably, the top right portion of the tool in Panel c) has worn away entirely.**

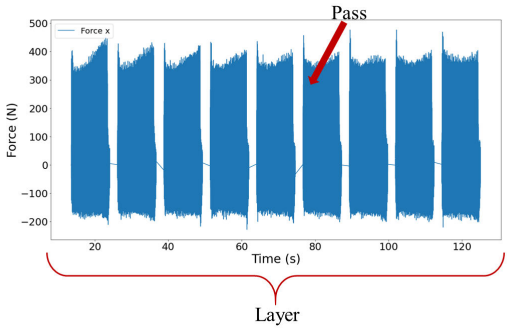



**FIGURE 4. Raw data corresponding to tool number 4.**

Tool number 4. This dataset encapsulates the tool's motion across one layer, featuring bi-directional sweeps in $x$ and $y$ directions. The smaller data segments within the file indicate each individual pass, which involves a full-length movement in the $x$-direction.

After the completion of each layer for a specific tool, we took measurements to assess the degree of tool wear. These measurements were subsequently employed as labels in the multi-label classification task. Tools with wear values below 30 $\mu m$ were designated as healthy, while those with wear values ranging from 30 $\mu m$ to 70 $\mu m$ were

identified as being in the transition phase. Tools with wear values exceeding 70 $\mu m$ were classified as being in failure mode. Notably, the occurrence of visible sparks during the experiment typically signaled that the tool had reached its failure mode. Figure 5 depicts the wear of Tool number 4 across different layers, with each layer's data labeled based on the respective tool wear measurement.

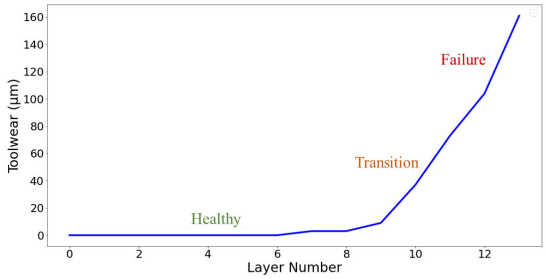



**FIGURE 5.** The tool wear measurements for tool number 4 for each layer.

### C. DATA PRE-PROCESSING

Each data layer is comprised of multiple passes, as depicted in Fig 4. Each pass's time series functions as a distinct input for the DNN algorithm. As the tool moves from one pass to the next during data collection, certain segments of the data may become superfluous. This redundancy can be observed in Fig 6, highlighted in red, for Tool number 4, layer two.

To segregate genuine data associated with each pass, we implemented a straightforward rule-based technique. Here, successive sets of 50 data points in the time series are assessed based on their peak value. If this value surpasses 50, these data points are added to an array. This array continues to accumulate points as long as the 50-point sets fulfill this criterion. However, once a deviation from this pattern arises, the existing array is completed, and a fresh array is initiated. Given that the primary data chunks, which represent the actual pass, are substantially larger, it becomes simple to identify and omit the smaller, redundant segments. This ensures that each pass's data can be efficiently input into the proposed model. It's important to note that all passes within a particular layer share the same label.

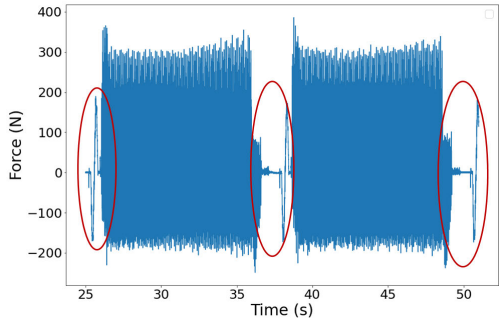



**FIGURE 6.** The redundant part of data that occurred between passes are within red ellipses.

Additionally, given that the tool is equipped with two cutters, it interacts with the workpiece twice for each rotation, resulting in a tooth passing frequency of 133 Hz ($3979 Rev/60 sec \times 2$ flute). Before processing the data with the DNN algorithm, we employed a 4th-order Butterworth bandpass filter with a low-frequency threshold set to 130 Hz and a high-frequency threshold at 135 Hz. After filtering, the data was then adjusted by removing its mean and standardizing to unit variance. Fig 7 showcases the filtered and standardized data for Tool number 4, layer two, associated with $F_x$. The DNN algorithm was fed with filtered and standardized data across all layers and passes.

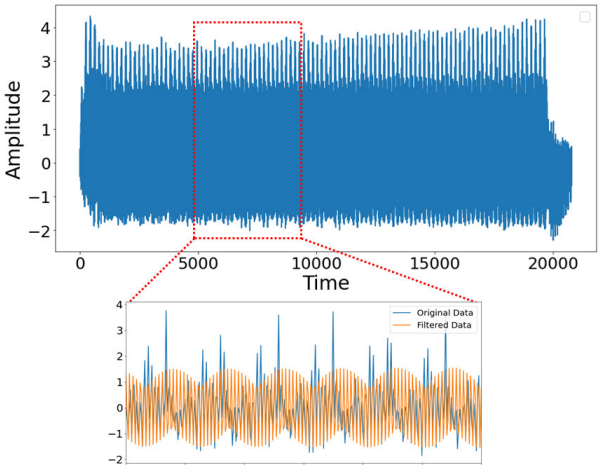



**FIGURE 7.** The original data along with the standardized, filtered data have been graphically represented. For better visualization, the zoomed section is also under-sampled (every 10th point has been sampled).

## V. MODEL TRAINING

The model was built using Python and leveraged Keras, an open-source library that provides a Python-based interface for NN applications. To optimize the model's efficacy, we implemented several enhancements, including the implementation of batch normalization. Batch normalization is designed to address internal covariate shift, a phenomenon where the input distribution for each layer alters as the network trains. By performing normalization on each layer's inputs, batch normalization not only speeds up the training phase but also stabilizes it [31].

Model training had two steps. In the first step, via trial and error we found a base model determining the structure of NN model like number of layers and type of layers used. The decision to keep the number of layers fixed stemmed from our experimentation, where we observed that increasing the number of layers did not yield performance improvements. Instead, it tended to noticeably prolong the training duration. Further elaboration on this can be found in Section VI-B. This base configuration was the same for all the tools within the leave- one-out validation process. After establishing the base model, in the second step we tuned the model. Hence, in our pursuit of the optimal NN

design, we employed a random search for hyperparameter tuning. This method allowed us to effectively explore the vast and intricate hyperparameter landscape without the extensive computational demands of a complete grid search. Our experiments spanned various activation functions, notably ReLU and tanh [32]. Also diverse optimization strategies that included stochastic gradient descent (sgd), Adam, Adaptive Gradient algorithm (Adagrad), and rmsprop [33], [34], [35], [36] were tested. Moreover, a variety of kernel and filter dimensions as shown in Table 2. For each tool, we probed 50 distinct hyperparameter combinations. These combinations were assessed using a validation subset, which constituted 20% of our training data. The combinations that yielded the highest accuracy were then chosen. To discern the differences in chosen hyperparameters across various tools, we reviewed the top three hyperparameter sets for each tool and drew comparisons among them. Additionally, to hasten the training process and mitigate overfitting, we implemented Early Stopping with a patience setting of five and capped the epochs at 500. The Categorical Cross-Entropy served as our chosen loss function.

**TABLE 2. Hyperparameter space of the random search.**

| Kernel size | Filter size | Learning Rate | Optimizer |
|---|---|---|---|
| 3, 5, 7 | 32, 64, 128 | 0.01, 0.001, 0.0001, 0.00001 | Adam, SGD, RMSProp, AdaGrad |

## VI. RESULTS AND ANALYSIS

### A. BUILDING AND TRAINING

For each tool, the time series data corresponding to each pass of every layer served as an independent input to the model. It is noteworthy that the data for each pass is three-dimensional, encompassing force measurements for $F_x, F_y, F_z$. The model employs a leave-one-out analysis methodology for training. In this scheme, a single tool is reserved for testing while the model is trained on data from the remaining tools. For each tool selected for testing, Table 3 provides the details on the number of training, validation, and test data entries. This process is iteratively conducted, ensuring each tool is isolated for testing at least once. We present the training results for individual tools, evaluating the model's performance using metrics such as accuracy, precision, and recall. An average score for these metrics is also reported across all tools. Importantly, the dataset is imbalanced: 69% of the data falls under the healthy category, 14% under the transition phase, and 17% under the failure mode. Consequently, precision and recall, along with the analysis of confusion matrices, serve as more reliable evaluation metrics compared to mere accuracy. Below, the result of our hyperparameter tuning is discussed in detail.

#### 1) HYPERPARAMETERS TUNING

Our assessment of hyperparameters was tailored to each tool, where we examined the three most promising architectures for every tool. This approach provided a better perspective on the variability across different tools. In all evaluations, across all top-performing architectures, the ReLU activation function consistently emerged as the best choice. Each model was structured with five layers. There were observed variations in filter sizes, kernel sizes, learning rates, and optimization algorithms for different tools. The three top architectures for each tool are documented in Table 11. Notably, among optimization algorithms, RMSProp, and AdaGrad frequently emerged as top choices. The flow chart of the optimal network for tool number 1 is shown in Fig 8.

**TABLE 3. For each tool selected for testing, the numbers of entries in the test, train, and validation sets are provided.**

| Testing Tool | | Training Set | Validation Set |
|---|---|---|---|
| 1 | 126 | 425 | 106 |
| 2 | 126 | 424 | 108 |
| 3 | 63 | 475 | 118 |
| 4 | 108 | 425 | 108 |
| 5 | 107 | 432 | 126 |

#### 2) A GENERAL ARCHITECTURE

In contrast, a general model was trained for all tools, characterized by a filter size of 32, kernel size of 3, and a batch size of 32. This model was optimized using the 'RMSProp' algorithm with a learning rate of $10^{-3}$. The rationale for this general model was its relatively swift training time, thanks to the smallest filter and kernel sizes and an efficient optimization algorithm. This made it independent of any tool-specific nuances. It is worth mentioning that both the AdaGrad and RMSProp algorithms have their foundations in the stochastic gradient approach. While AdaGrad approximates the loss function's Hessian, RMSProp incorporates a momentum term, resulting in quicker optimization. Several studies have emphasized the superior efficiency and accuracy of momentum-based algorithms, especially for high-frequency target functions [37], [38], [39], [40]. Consequently, RMSProp was deemed appropriate for our general model.

This general model allowed us to compare a rapid algorithm with the most optimal one. Comparing the top-performing architectures with our standard model revealed that, although the general model's performance was marginally lower, its architecture possessed commendable generalization capabilities, making it apt for training across all tools. A detailed comparison of accuracy, precision, and recall between the optimal models and the baseline model is presented in Table 4.

**TABLE 4. Comparison of the performance of the optimal architecture with a proposed general architecture.**

| Tool Number | Optimal Model Accuracy | General Model Accuracy | Optimal Model Precision | General Model Precision | Optimal Model Recall | General Model Recall |
|---|---|---|---|---|---|---|
| 1 | 0.97 | 0.95 | 0.90 | 0.85 | 0.94 | 0.90 |
| 2 | 0.93 | 0.87 | 0.71 | 0.65 | 0.78 | 0.78 |
| 3 | 0.87 | 0.84 | 0.87 | 0.83 | 0.87 | 0.83 |
| 4 | 0.93 | 0.92 | 0.91 | 0.90 | 0.87 | 0.85 |
| 5 | 0.97 | 0.96 | 0.93 | 0.92 | 0.94 | 0.94 |

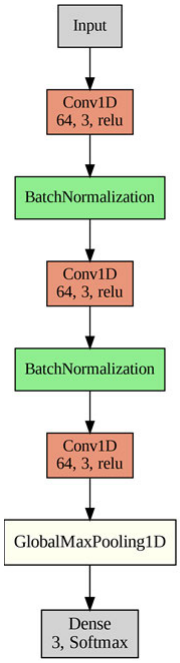



**FIGURE 8.** The flow chart of the neural network's architecture used in this study.

### B. EXPLORING THE IMPACT OF NETWORK DEPTH ON DNN PERFORMANCE

As one of the ongoing discussions in the field revolves around the impact of network depth on the performance of DNNs. To explore the influence of depth on our model's efficacy, we trained variants of our architecture with three, five, ten, and twenty hidden layers. Importantly, these experiments utilized an already-optimized neural network configuration. The averaged metrics of accuracy, precision, and recall across all tools for different network depths are presented in Table 5. Contrary to expectations, increasing the number of hidden layers did not lead to a discernible improvement in the model's performance. In fact, networks with three and five hidden layers demonstrated superior effectiveness. The lack of improvement in model performance despite increased network depth can be explained by multiple considerations. Firstly, DNNs typically demand large datasets for effective generalization, and a dataset of limited size could result in overfitting in deeper architectures. Secondly, the additional complexity brought by extra layers might be superfluous for the given problem, especially if the problem itself is not overly complicated. Thirdly, deeper networks are more prone to challenges such as vanishing or exploding gradients, which can hinder the optimization process. In light of these factors, we found that architectures with three and five hidden layers were best suited for our specific use case.

### C. OPTIMIZING INPUT DATA SAMPLING FOR IMPROVED MODEL TRAINING

A noteworthy observation pertained to the manipulation of the input dataset. The original time series for each tool featured dimensions of $3 \times 20800$. To expedite the training process, we explored the feasibility of under-sampling the data. We adopted a straightforward method of selecting every $n^{th}$ data point. Models were trained with $n$ set to 0, 10, 20, 40, and 60, where $n = 0$ signifies no under-sampling. As displayed in Table 6, the metrics of average accuracy, precision, and recall across all tools for each $n$ value are presented. Notably, an $n$ value of 60 led to a decline in precision and recall, and also resulted in a substantial increase in the standard deviation of these metrics. The data implies that an under-sampling rate of $n = 40$ offers a balanced trade-off, achieving high levels of accuracy, precision, and recall.

**TABLE 5.** The average accuracy, precision, recall, and F1 score among all tools for different numbers of hidden layers.

| Layer | Accuracy | Precision | Recall | F1 score |
|---|---|---|---|---|
| 3 | $0.90 \pm 0.04$ | $0.81 \pm 0.12$ | $0.82 \pm 0.05$ | $0.82 \pm 0.08$ |
| 5 | $0.88 \pm 0.03$ | $0.82 \pm 0.10$ | $0.81 \pm 0.04$ | $0.82 \pm 0.08$ |
| 10 | $0.85 \pm 0.08$ | $0.72 \pm 0.22$ | $0.72 \pm 0.22$ | $0.72 \pm 0.20$ |
| 20 | $0.82 \pm 0.11$ | $0.74 \pm 0.12$ | $0.76 \pm 0.11$ | $0.75 \pm 0.11$ |

**TABLE 6.** The average accuracy, precision, recall, and F1 score among all tools for different choices of under-sampling.

| $n$ | Accuracy | Precision | Recall | F1 score |
|---|---|---|---|---|
| 0 | $0.91 \pm 0.04$ | $0.81 \pm 0.12$ | $0.82 \pm 0.08$ | $0.82 \pm 0.07$ |
| 10 | $0.90 \pm 0.02$ | $0.83 \pm 0.12$ | $0.84 \pm 0.05$ | $0.83 \pm 0.08$ |
| 20 | $0.90 \pm 0.04$ | $0.80 \pm 0.13$ | $0.83 \pm 0.09$ | $0.82 \pm 0.07$ |
| 40 | $0.90 \pm 0.02$ | $0.85 \pm 0.12$ | $0.87 \pm 0.07$ | $0.86 \pm 0.04$ |
| 60 | $0.90 \pm 0.04$ | $0.77 \pm 0.24$ | $0.71 \pm 0.20$ | $0.74 \pm 0.18$ |

### D. FEATURE IMPORTANCE ANALYSIS FOR TOOL PERFORMANCE PREDICTION

The input dataset includes three features: $F_x, F_y, F_z$. To assess the relative importance of each feature in the learning process, separate models were trained using all features, as well as individual ones—namely $F_x$, $F_y$, and $F_z$. The average accuracy, precision, and recall across all tools for each feature configuration are tabulated in Table 7. This exercise yielded intriguing insights; notably, $F_x$ emerged as the most impactful feature, leading in accuracy, precision, and recall metrics. Conversely, the low scores for $F_z$ suggest it may not offer significant informational value.

Upon deeper analysis of $F_y$, it was observed that for Tools 1, 4, and 5, models trained with this feature yielded performance metrics comparable to those achieved with $F_x$. However, for Tools 2 and 3, the performance significantly declined when trained solely on $F_y$ (refer to Table 8 for details).

It is worth mentioning that for models utilizing $F_x$, Tool 2 displayed a marked decline in both precision and recall compared to other tools. A detailed examination of its confusion matrix reveals this discrepancy (refer to Fig 9, left panel). The matrix shows that seven time series were misclassified as failure mode (highlighted in yellow)

**TABLE 7.** The average accuracy, precision, recall, and F1 score among all tools for different choices of feature.

| Feature | Accuracy | Precision | Recall | F1 socre |
|---|---|---|---|---|
| $F_x, F_y, F_z$ | $0.84 \pm 0.07$ | $0.81 \pm 0.12$ | $0.82 \pm 0.08$ | $0.82 \pm 0.10$ |
| $F_x$ | $0.90 \pm 0.02$ | $0.85 \pm 0.12$ | $0.87 \pm 0.07$ | $0.86 \pm 0.08$ |
| $F_y$ | $0.81 \pm 0.15$ | $0.73 \pm 0.20$ | $0.74 \pm 0.17$ | $0.73 \pm 0.15$ |
| $F_z$ | $0.60 \pm 0.18$ | $0.50 \pm 0.07$ | $0.50 \pm 0.07$ | $0.50 \pm 0.07$ |

**TABLE 8.** The accuracy, precision, and recall among all tools for $F_x$ and $F_y$.

| Tool Number | Accuracy $F_x$ | Accuracy $F_y$ | Precision $F_x$ | Precision $F_y$ | Recall $F_x$ | Recall $F_y$ |
|---|---|---|---|---|---|---|
| 1 | 0.97 | 0.94 | 0.90 | 0.85 | 0.94 | 0.93 |
| 2 | 0.93 | 0.60 | 0.71 | 0.53 | 0.78 | 0.54 |
| 3 | 0.87 | 0.72 | 0.87 | 0.45 | 0.87 | 0.59 |
| 4 | 0.93 | 0.87 | 0.91 | 0.91 | 0.87 | 0.72 |
| 5 | 0.97 | 0.97 | 0.93 | 0.96 | 0.94 | 0.88 |

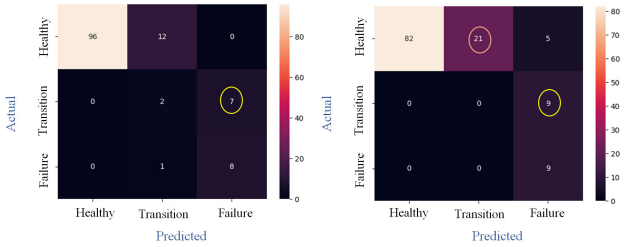



**FIGURE 9.** Left panel: Confusion matrix for tool number 2, that was trained using $F_x$. Right panel: Confusion matrix for tool number 2, that was trained using $F_y$.

when their true label was transition mode. Interestingly, all seven instances belonged to layer 13, which had actual tool wear of $69\mu m$, just below the $70\mu m$ threshold set for the failure mode. Given the closeness of this value to the threshold, the misclassification is less concerning. In fact, the model's prediction suggests that these time series were more characteristic of failure mode than of transition mode.

Additionally, 12 time series (circled in orange) were incorrectly labeled as transition mode when they were, in fact, in the healthy mode. Ten of these instances belonged to layer 12, where the tool wear was $23\mu m$, not too distant from the $30\mu m$ threshold for transition. This is particularly noteworthy given that the tool wear for previous layers was below $11\mu m$.

For Tool 2, as illustrated in Table 8, $F_y$ yielded unsatisfactory results. Further investigation into its confusion matrix (see Fig 10, right panel) revealed frequent misclassifications of instances as healthy, underscoring the feature's ineffectiveness for this particular tool.

Tool 3 also exhibited inferior performance when trained using $F_y$. Upon examining the confusion matrices for models trained on $F_x$ (Fig. 10, left panel) and $F_y$ (Fig. 10, right panel), it is evident that the $F_y$-based model frequently misclassified instances in the transition mode as healthy, and those in the failure mode as transition. This pattern of misclassification was also observed in Tool 2, suggesting that $F_y$ may not effectively distinguish between healthy and transition states, as well as between transition and failure states.

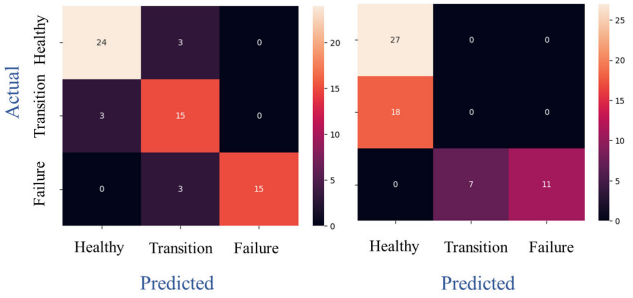



**FIGURE 10.** Left panel: Confusion matrix for tool number 3, that was trained using $F_x$. Right panel: Confusion matrix for tool number 3, that was trained using $F_y$.

Notably, the performance metrics for models trained using $F_y$ for Tools 1, 4, and 5 were comparable to those for $F_x$. The subpar performance for Tools 2 and 3 may be attributable to faulty sensor readings for $F_y$. This can also explain why models trained using all three features $(F_x, F_y, F_z)$ yielded lower performance; the measurements in $F_z$ and, for some tools, in $F_y$ act more as noise than as informative signals. It is worth emphasizing that none of the models trained solely on $F_z$ showed high performance, further solidifying the notion that $F_z$ should be excluded from the model inputs as it likely contributes more noise than useful information.

Tool 3 also exhibited inferior performance when trained using $F_y$. Upon examining the confusion matrices for models trained on $F_x$ (Fig. 9, left panel) and $F_y$ (Fig. 9, right panel), it is evident that the $F_y$-based model frequently misclassified instances in the transition mode as healthy, and those in the failure mode as transition. This pattern of misclassification was also observed in Tool 2, suggesting that $F_y$ may not effectively distinguish between healthy and transition states, as well as between transition and failure states. Summery of F1 score values for $F_y$ and $F_y$ is available in Table 12.

### E. COMPARISON WITH OTHER MODELS

In this section, we assess the three-layer 1D-CNN model we have developed, which features a straightforward structure facilitating fast training. We benchmark our model against state-of-the-art models detailed in the literature. Notably, given the frequent application of BiLSTM in processing time series data, we have implemented the algorithm as suggested by [17]. Additionally, we compare our model with the SUBLSTM model, as detailed in [10], which employs bi-directional, uni-directional, and 1D-CNN layers. This comparison aims to determine if integrating LSTM layers might improve performance. We were also interested in exploring if attention-based models could offer better outcomes. To this end, we implemented the attention model described in [19], which incorporates an attention layer into a deep LSTM framework. Moreover, we explored the efficacy of the hybrid attention-based CNN-LSTM model proposed in [21]. For each algorithm, we present accuracy,

**TABLE 9.** Averaged accuracy, precision, recall, and F1 score among all tools for different models is provided. The highest values are shown in bold.

| Model | Accuracy | Precision | Recall | F1 score |
|---|---|---|---|---|
| BiLSTM [17] | $0.85 \pm 0.07$ | $0.72 \pm 0.14$ | $0.75 \pm 0.07$ | $0.70 \pm 0.05$ |
| SUBLSTM [10] | $0.81 \pm 0.08$ | $\mathbf{0.85 \pm 0.07}$ | $0.73 \pm 0.09$ | $0.75 \pm 0.07$ |
| Attention 1 [19] | $0.86 \pm 0.05$ | $0.73 \pm 0.10$ | $0.70 \pm 0.05$ | $0.70 \pm 0.08$ |
| Attention 2 [21] | $0.86 \pm 0.07$ | $0.75 \pm 0.12$ | $0.74 \pm 0.10$ | $0.75 \pm 0.10$ |
| Proposed | $\mathbf{0.90 \pm 0.04}$ | $0.81 \pm 0.12$ | $\mathbf{0.82 \pm 0.05}$ | $\mathbf{0.82 \pm 0.08}$ |

**TABLE 10.** The accuracy, precision, recall, and F1 score for Tool 5 based on implementing different models is provided. Moreover, precision and recall for each sub-class are shown. A: accuracy, P: precision, R: recall, P1: precision for class 1, P2: precision for class 2, P3: precision for class 3, R1: recall for class 1, R2: recall for class 2, R3: recall for class 3 The highest values are shown in bold.

| Model | A | P | R | F1 | P1 | P2 | P3 | R1 | R2 | R3 |
|---|---|---|---|---|---|---|---|---|---|---|
| BiLSTM | 0.85 | 0.67 | 0.69 | 0.62 | 0.99 | 0.50 | 0.53 | 0.96 | 0.11 | **1.00** |
| SUBLSTM | 0.90 | 0.81 | 0.86 | 0.83 | 0.99 | 0.67 | 0.78 | 0.94 | 0.67 | **1.00** |
| Attention 1 | 0.77 | 0.61 | 0.5 | 0.52 | 0.80 | 0.00 | **1.00** | **1.00** | 0.00 | 0.50 |
| Attention 2 | 0.82 | 0.75 | 0.61 | 0.64 | 0.86 | 0.38 | **1.00** | **1.00** | 0.28 | 0.56 |
| Proposed | **0.96** | **0.93** | **0.95** | **0.93** | **1.00** | **0.78** | **1.00** | 0.96 | **1.00** | 0.89 |

precision, recall, and F1 score, averaged across our dataset (Table 9). The models evaluated demonstrated comparable performance, with very similar accuracy, precision, recall, and F1 score values. Despite these similarities, none of the models exceeded the overall performance of our proposed 1D-CNN model.

Upon detailed examination of how the models perform on individual tools, we observe that the more complex models exhibit lower per-class precision and recall. While these models effectively identify class 1 (healthy state), they struggle to differentiate between transition and failure modes. Specifically, for Tool Number 5, we have provided the accuracy, precision, recall, and F1 score for each class across all models. It is evident from our analysis that our proposed model excels in distinguishing between failure and transition modes, as shown in (Table 10).

## VII. SUMMARY AND CONCLUSION

In the current study, we employed a three-layer 1D-CNN to develop a tri-class classification model for predicting the RUL of milling tools. The advantage of using 1D-CNN lies in its ability to effectively learn from raw time-series data, thereby obviating the need for extensive feature engineering and permitting minimal pre-processing. The experimental data was gathered from a milling machine housed at the Automotive and Surface Transportation Laboratory of the National Research Council. During the milling operation, the tool traverses in the $x$, $y$, and $z$ directions. Specifically, the tool advances along the $x$-axis to complete one pass, during which the forces $F_x$, $F_y$, and $F_z$ are recorded as time-series data and stored. The milling operation then progresses along the $y$-axis until a single layer is entirely milled. This process is repeated until visible indications of tool wear become apparent. After each layer is completed, the degree of tool wear is assessed using a bloom laser sensor.

It is crucial to emphasize that the bloom laser sensor enabled us to generate target labels that are independent of the input features and the measurement device itself. For the scope of this study, we utilized five distinct milling tools. Given the limited size of the dataset, we employed a leave-one-out cross-validation strategy for model evaluation. In this approach, the model is trained on data from all tools except one, which is subsequently used as the test set. This procedure is iteratively repeated, each time designating a different tool's data as the test set. Through this methodology, we computed the accuracy, precision, and recall metrics for each individual tool. For the leave-one-out cross-validation task, data for each tool was stored in distinct data frames. Consequently, for each test iteration, one data frame was designated as the test set, while the remaining data frames were concatenated to form the training set.

Our meticulous scrutiny of the input data yielded several instructive findings with implications for future research. For instance, we demonstrated that the model retains its performance efficacy even when trained on under-sampled data, specifically by selecting every 40th data point in a time series. This finding suggests that it may be unnecessary to collect data at an exceedingly high sampling rate of 2000 Hz. Moreover, an evaluation of models trained on different input features ($F_x$, $F_y$, $F_z$, and $F_x, F_y, F_z$) revealed that $F_z$ contributes no meaningful information towards predicting RUL. Interestingly, models trained on $F_y$ exhibited varying performance: while three tools showed robust results, the remaining two exhibited subpar performance, hinting at potential measurement inconsistencies during experimentation. On the other hand, models trained solely on $F_x$ consistently demonstrated high performance, suggesting its sufficiency as a standalone feature. These insights not only underscore the utility of machine learning models in identifying data anomalies but also guide us in optimizing experimental setups, such as determining the most effective sampling rate. It should also be noted that the performance of the models trained with $F_x$ for Tool 2 and Tool 3 was comparable to that of other tools. This similarity suggests that there may not be a significant difference in physical tool wear, but rather, it is likely due to erroneous data acquisition in the y-direction. Interestingly, when $F_y$ from these tools was used in the training set (e.g., for Tool 1), the model's performance for Tool 1 was satisfactory. This indicates a high tolerance of the models to noisy or faulty ground truth data.

In our study, we explored various network architectures and determined the most effective configuration via random search. We also compared the optimal architecture of each tool with a general architecture and we showed a general architecture for all tools has the capability of providing high accuracy, precision, and recall among all the tools. More precisely, as detailed in Table 4, the general architecture demonstrated satisfactory performance and was thus chosen for the results presented throughout the paper. This includes comparisons with recent state-of-the-art models discussed in Section VI-E. The average metrics of accuracy, precision,

and recall for models trained on $F_x$ were $0.90 \pm 0.02$, $0.85 \pm 0.12$, and $0.87 \pm 0.08$, respectively. We also conducted a comparison of our proposed model with cutting-edge models from recent research. Our findings indicate that our lightweight 1D-CNN surpasses more complex algorithms in performance. One possible explanation is that, given the small size of our datasets, more complex algorithms tend to overfit. This observation is noteworthy as it highlights that while a model like the CNN, which capitalizes on time series data, achieves high performance, more complex and deeper networks may actually diminish performance.

Notably, we performed an in-depth analysis of the confusion matrix for Tool 2, as it exhibited the lowest precision and recall among all tools. Our examination revealed that the mislabeled samples predominantly belonged to data points where the actual tool-wear values were at the boundary of transitioning between labels. For instance, seven passes were predicted to be in the failure mode, whereas the ground truth categorized them as being in transition. Further scrutiny showed that the tool-wear for these passes was $69\mu m$, just shy of the $70\mu m$ threshold we set for the failure mode. It's important to highlight that our dataset is imbalanced, with approximately 70% of the data labeled as 'healthy', making the correct classification of the remaining 'transition' and 'failure' modes more challenging. Despite this imbalance, the high precision and recall values, along with individual confusion matrices, confirm the capability of our trained models to accurately classify all three states.

In summary, this study demonstrates the efficacy of 1D-CNN algorithms for assessing the health status of milling tools. While many existing RUL prediction studies employ NNs, they often rely on manual feature engineering to combat the curse of dimensionality. This approach, however, leads to static features that fail to capture the inherent dynamic behavior of time series, thereby underutilizing the full potential of neural network models. In contrast, our methodology necessitated minimal pre-processing and allowed us to directly input time series data.

Interestingly, the anomalies detected in the model's output can serve as diagnostic indicators for potential sensor malfunctions. This offers invaluable feedback to experimenters about the quality of their collected data. Furthermore, insights gleaned from our model can guide optimal experimental setups, including the selection of the most effective sampling rate.

Looking ahead, we aim to broaden the scope of our research by incorporating larger datasets and exploring the integration of additional sensor modalities such as thermal and visual imaging, with the objective of further enhancing model performance.

## APPENDIX
## TABLE FOR COMPARISON

The architecture of the three models with the highest accuracy for each Tool is presented in Table 11.

**TABLE 11. The architecture of the three models with highest accuracy for each Tool is shown.**

| Tool Number | Kernel size | Filter size | Optimizer | Learning rate | Batch size | Accuracy |
|---|---|---|---|---|---|---|
| 1 | 32, 64, 64, 32, 64<br>32, 64, 32, 32, 128<br>32, 32, 32, 32,128 | 3, 7, 3, 3, 5<br>3, 3, 5,3, 7<br>5,5,5,5,3 | RMSProp<br>AdaGrad<br>RMSProp | 0.001<br>0.0001<br>0.000001 | 32<br>32<br>32 | 0.97<br>0.96<br>0.96 |
| 2 | 64, 32, 128,32, 128<br>128,32, 128, 32, 128<br>128, 128, 128, 128,64 | 3,5,3,5,7<br>3,5,3,5,7<br>3,5,7,3,3 | AdaGrad<br>AdaGrad<br>Adam | 0.001<br>0.01<br>0.01 | 32<br>32<br>128 | 0.93<br>0.91<br>0.90 |
| 3 | 32, 128, 32, 128, 32<br>64, 128, 64, 128, 64<br>64, 32, 64, 128, 128 | 3, 5, 5, 7, 7<br>7, 7, 5, 5, 5<br>7, 3, 5, 3,3 | AdaGrad<br>SGD<br>Adam | 0.00001<br>0.0001<br>0.0001 | 128<br>32<br>64 | 0.87<br>0.86<br>0.86 |
| 4 | 64, 32, 128, 32, 128<br>128, 128, 32, 128, 64<br>32, 64, 128, 128, 128 | 5, 5, 5, 3, 5<br>5, 3, 3, 7, 7<br>7, 3, 7, 7, 3 | RMSProp<br>RMSProp<br>SGD | 0.00001<br>0.001<br>0.001 | 32<br>32<br>32 | 0.93<br>0.92<br>0.91 |
| 5 | 64, 64, 64, 128, 32<br>64, 64, 128, 32, 128<br>32, 64, 128, 32, 32 | 3, 5, 5, 7, 3<br>7, 5,7, 5, 7<br>7, 5, 5, 5, 5 | RMSProp<br>AdaGrad<br>AdaGrad | 0.01<br>0.001<br>0.001 | 32<br>32<br>32 | 0.96<br>0.95<br>0.95 |

**TABLE 12. F1 scores for $F_x$ and $F_y$ forces in each tool.**

| Tool Number | F1 score, $F_x$ | F1 scorem $F_y$ |
|---|---|---|
| 1 | 0.92 | 0.88 |
| 2 | 0.74 | 0.53 |
| 3 | 0.87 | 0.51 |
| 4 | 0.89 | 0.80 |
| 5 | 0.93 | 0.92 |

The F1 score for $F_x$ and $F_y$ components of the force for each tool is shown in Table12.

### A. CODE AND DATA AVAILABILITY

Upon request, the ML algorithms used for the training will be shared.

## ACKNOWLEDGMENT

The authors would like to thank Dr. Mihnea Lonescu for his valuable insights.

## REFERENCES

[1] A. Majidian and M. Saidi, "Comparison of fuzzy logic and neural network in life prediction of boiler tubes," *Int. J. Fatigue*, vol. 29, no. 3, pp. 489–498, Mar. 2007.

[2] J. Z. Sikorska, M. Hodkiewicz, and L. Ma, "Prognostic modelling options for remaining useful life estimation by industry," *Mech. Syst. Signal Process.*, vol. 25, no. 5, pp. 1803–1836, Jul. 2011.

[3] L. Liao and F. Köttig, "Review of hybrid prognostics approaches for remaining useful life prediction of engineered systems, and an application to battery life prediction," *IEEE Trans. Rel.*, vol. 63, no. 1, pp. 191–207, Mar. 2014.

[4] D. C. Swanson, J. Michael Spencer, and S. H. Arzoumanian, "Prognostic modelling of crack growth in a tensioned steel band," *Mech. Syst. Signal Process.*, vol. 14, no. 5, pp. 789–803, Sep. 2000.

[5] X.-S. Si, W. Wang, C.-H. Hu, and D.-H. Zhou, "Remaining useful life estimation—A review on the statistical data driven approaches," *Eur. J. Oper. Res.*, vol. 213, no. 1, pp. 1–14, 2011.

[6] Z. He, S. Wang, K. Wang, and K. Li, "Prognostic analysis based on hybrid prediction method for axial piston pump," in *Proc. IEEE 10th Int. Conf. Ind. Informat.*, Jul. 2012, pp. 688–692.

[7] Y. Cheng, H. Zhu, K. Hu, J. Wu, X. Shao, and Y. Wang, "Multisensory data-driven health degradation monitoring of machining tools by generalized multiclass support vector machine," *IEEE Access*, vol. 7, pp. 47102–47113, 2019.

[8] M. Liu, X. Yao, J. Zhang, W. Chen, X. Jing, and K. Wang, "Multi-sensor data fusion for remaining useful life prediction of machining tools by IABC-BPNN in dry milling operations," *Sensors*, vol. 20, no. 17, p. 4657, Aug. 2020.

[9] C. Sun, M. Ma, Z. Zhao, S. Tian, R. Yan, and X. Chen, "Deep transfer learning based on sparse autoencoder for remaining useful life prediction of tool in manufacturing," *IEEE Trans. Ind. Informat.*, vol. 15, no. 4, pp. 2416–2425, Apr. 2019.

[10] Q. An, Z. Tao, X. Xu, M. El Mansori, and M. Chen, "A data-driven model for milling tool remaining useful life prediction with convolutional and stacked LSTM network," *Measurement*, vol. 154, Mar. 2020, Art. no. 107461.

[11] B. Wang, Y. Lei, T. Yan, N. Li, and L. Guo, "Recurrent convolutional neural network: A new framework for remaining useful life prediction of machinery," *Neurocomputing*, vol. 379, pp. 117–129, Feb. 2020.

[12] F. C. Zegarra, J. Vargas-Machuca, and A. M. Coronado, "Tool wear and remaining useful life (RUL) prediction based on reduced feature set and Bayesian hyperparameter optimization," *Prod. Eng.*, vol. 16, no. 4, pp. 465–480, Aug. 2022.

[13] W. Li, L.-C. Zhang, C.-H. Wu, Y. Wang, Z.-X. Cui, and C. Niu, "A data-driven approach to RUL prediction of tools," *Adv. Manuf.*, vol. 12, no. 1, pp. 6–18, Mar. 2024.

[14] M. Zhu, Y. Yang, X. Feng, Z. Du, and J. Yang, "Robust modeling method for thermal error of CNC machine tools based on random forest algorithm," *J. Intell. Manuf.*, vol. 34, no. 4, pp. 2013–2026, Apr. 2023.

[15] N. Gebraeel, M. Lawley, R. Liu, and V. Parmeshwaran, "Residual life predictions from vibration-based degradation signals: A neural network approach," *IEEE Trans. Ind. Electron.*, vol. 51, no. 3, pp. 694–700, Jun. 2004.

[16] J. Niu, C. Liu, L. Zhang, and Y. Liao, "Remaining useful life prediction of machining tools by 1D-CNN LSTM network," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, Dec. 2019, pp. 1056–1063.

[17] T. F. De Barrena, J. L. Ferrando, A. García, X. Badiola, M. S. de Buruaga, and J. Vicente, "Tool remaining useful life prediction using bidirectional recurrent neural networks (BRNN)," *Int. J. Adv. Manuf. Technol.*, vol. 125, nos. 9–10, pp. 4027–4045, Apr. 2023.

[18] X. Zhang, B. Shi, B. Feng, L. Liu, and Z. Gao, "A hybrid method for cutting tool RUL prediction based on CNN and multistage Wiener process using small sample data," *Measurement*, vol. 213, May 2023, Art. no. 112739.

[19] Z. Chen, M. Wu, R. Zhao, F. Guretno, R. Yan, and X. Li, "Machine remaining useful life prediction via an attention-based deep learning approach," *IEEE Trans. Ind. Electron.*, vol. 68, no. 3, pp. 2521–2531, Mar. 2021.

[20] H. Zhang, Q. Zhang, S. Shao, T. Niu, and X. Yang, "Attention-based LSTM network for rotatory machine remaining useful life prediction," *IEEE Access*, vol. 8, pp. 132188–132199, 2020.

[21] T. Zuo, K. Zhang, Q. Zheng, X. Li, Z. Li, G. Ding, and M. Zhao, "A hybrid attention-based multi-wavelet coefficient fusion method in RUL prognosis of rolling bearings," *Rel. Eng. Syst. Saf.*, vol. 237, Sep. 2023, Art. no. 109337.

[22] Z. Qu, L. Zhu, S. Ma, and B. Zhang, "Remaining useful life prediction of high-speed railroad contact network based on stacking integrated attention-LSTM-CNN deep learning," *Arabian J. Sci. Eng.*, pp. 1–18, Mar. 2024.

[23] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *Proc. Int. Conf. Eng. Technol. (ICET)*, Aug. 2017, pp. 1–6.

[24] O. Abdeljaber, O. Avci, M. S. Kiranyaz, B. Boashash, H. Sodano, and D. J. Inman, "1-D CNNs for structural damage detection: Verification on a structural health monitoring benchmark data," *Neurocomputing*, vol. 275, pp. 1308–1317, Jan. 2018.

[25] S. Ma, W. Cai, W. Liu, Z. Shang, and G. Liu, "A lighted deep convolutional neural network based fault diagnosis of rotating machinery," *Sensors*, vol. 19, no. 10, p. 2381, May 2019.

[26] I. Mitiche, A. Nesbitt, S. Conner, P. Boreham, and G. Morison, "1D-CNN based real-time fault detection system for power asset diagnostics," *IET Gener., Transmiss. Distrib.*, vol. 14, no. 24, pp. 5766–5773, Dec. 2020.

[27] G. Farhani, Y. Zhou, M. E. Jenkins, M. D. Naish, and A. L. Trejos, "Using deep learning for task and tremor type classification in people with Parkinson's disease," *Sensors*, vol. 22, no. 19, p. 7322, Sep. 2022.

[28] R. Ileri, F. Latifoglu, and E. Demirci, "New method to diagnosis of dyslexia using 1D-CNN," in *Proc. Med. Technol. Congr. (TIPTEKNO)*, Nov. 2020, pp. 1–4.

[29] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj, and D. J. Inman, "1D convolutional neural networks and applications: A survey," *Mech. Syst. Signal Process.*, vol. 151, Apr. 2021, Art. no. 107398.

[30] D. M. W. Powers, "Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation," 2020, *arXiv:2010.16061*.

[31] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 448–456.

[32] S. Sharma, S. Sharma, and A. Athaiya, "Activation functions in neural networks," *Towards Data Sci.*, vol. 6, no. 12, pp. 310–316, 2017.

[33] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural Netw. Mach. Learn.*, vol. 4, no. 2, pp. 26–31, 2012.

[34] L. Bottou, "Stochastic gradient descent tricks," in *Neural Networks: Tricks Trade*, 2nd ed. Cham, Switzerland: Springer, 2012, pp. 421–436.

[35] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

[36] A. Lydia and S. Francis, "Adagrad—An optimizer for stochastic gradient descent," *Int. J. Inf. Comput. Sci.*, vol. 6, no. 5, pp. 566–568, 2019.

[37] K. K. Chandriah and R. V. Naraganahalli, "RNN/LSTM with modified Adam optimizer in deep learning approach for automobile spare parts demand forecasting," *Multimedia Tools Appl.*, vol. 80, pp. 26145–26159, Apr. 2021.

[38] G. Farhani, A. Kazachek, and B. Wang, "Momentum diminishes the effect of spectral bias in physics-informed neural networks," 2022, *arXiv:2206.14862*.

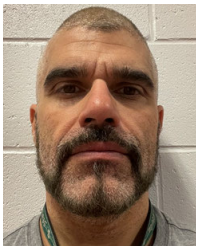
[39] R. O. Ogundokun, R. Maskeliunas, S. Misra, and R. Damaševičius, "Improved CNN based on batch normalization and Adam optimizer," in *Proc. Int. Conf. Comput. Sci. Its Appl.* Cham, Switzerland: Springer, 2022, pp. 593–604.

[40] J. Zhang, S. P. Karimireddy, A. Veit, S. Kim, S. Reddi, S. Kumar, and S. Sra, "Why are adaptive methods good for attention models?" in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 15383–15393.

**GHAZAL FARHANI** received the Ph.D. degree in physics from the University of Western Ontario, in 2019. Currently, she is a Research Officer with the Automotive and Surface Transportation Research Center, National Research Council Canada. Her research interests include machine learning, physical modeling, optimization, and data analytics. She boasts extensive academic and industrial experience in bridging the gap between machine learning algorithms and scientific applications, having worked in multidisciplinary fields, such as algorithm design for wearable devices, remote sensing data retrieval, manufacturing, and life cycle analysis. Her notable contributions include providing data-driven solutions for environmental challenges, particularly in manufacturing and related to life cycle analysis and contributing to climate change adaptation policies.

**SRIHARI KURUKURI** received the Ph.D. degree in computational mechanics from the University of Twente, The Netherlands. He is currently a Research Council Officer with the Automotive and Surface Transportation Research Center, National Research Council Canada. He joined NRC, in 2018. His current research interests include numerical modeling of manufacturing processes, including metal forming, friction modeling, and the application of AI techniques in industrial contexts. Prior to joining NRC, he was a Research Associate with the University of Waterloo, Canada. He has authored or coauthored more than 70 peer-reviewed articles, conference papers, and proprietary research and development reports.

**RYAN MYERS** has been a Research Officer with the National Research Council Canada, London, ON, Canada, since 2018, with a focus and expertise in the domains of automation, controls, robotics, and data-science. By leveraging data-driven approaches, he has been a Lead Expert for many successful projects leading to process improvements and the development of quality control tools for a wide range of manufacturing processes. He is also the Thrust Leader of Cyber-Physical Manufacturing Systems within the Advanced Manufacturing Program and is responsible for providing technical expertise and steering with in the domain of data-driven manufacturing.



**NELSON SANTOS** has been a Senior Technical Officer with the National Research Council of Canada, since 2001 (22 years of service). Started his working career in the steelmaking industry and then transitioned to the Automotive Industry, where he worked in an engineering laboratory for a tier-one automotive parts supplier. He joined the Additive Manufacturing Team as a Technical Officer with the London's site for the National Research Council of Canada in 2001. He has had the pleasure of working on various projects that involve additive manufacturing, machining disciplines, and project management activities. In his 22 years with NRC, he has had many experiences that have allowed him to apply his civil and mechanical engineering education.

**MOHAMMED TAUHIDUZZAMAN** received the B.Sc. degree in mechanical engineering from Bangladesh University of Engineering and Technology, the M.Eng. degree from the National University of Singapore, and the Ph.D. degree from McMaster University. He is currently an Associate Research Officer with the Automotive and Surface Transportation Research Center, National Research Council (NRC-AST) Canada. He joined with NRC, in 2016. He is also a Technical Leader of Advanced Manufacturing Program (AMP) and the Team Leader of Advanced Manufacturing Team. Previously, he worked for a Defense Supplier as a Manufacturing Engineering Specialist (optics) and successfully applied his academic research contributions that are still in practice. He has published more than 30 peer-reviewed articles, conference papers, and proprietary research and development reports. One of his articles in micro-milling has more than 140 citations to date. His research interests include laser-based SurfLock joining and digital process control. He received NRC-AST Technical Excellence Award, in 2017 and 2019.