applied sciences

MDPI

Article

Comparative Study of Conventional Machine Learning versus Deep Learning-Based Approaches for Tool Condition Assessments in Milling Processes

Agata Przybyś-Małaczek [1], Izabella Antoniuk [1], Karol Szymanowski [2], Michał Kruk [1], Alexander Sieradzki [1], Adam Dohojda [1], Przemysław Szopa [1] and Jarosław Kurek [1,*]

[1] Department of Artificial Intelligence, Institute of Information Technology, Warsaw University of Life Sciences, 02-776 Warsaw, Poland
[2] Department of Mechanical Processing of Wood, Institute of Wood Sciences and Furniture, Warsaw University of Life Sciences, 02-776 Warsaw, Poland
* Correspondence: jaroslaw_kurek@sggw.edu.pl

Abstract: This evaluation of deep learning and traditional machine learning methods for tool state recognition in milling processes aims to automate furniture manufacturing. It compares the performance of long short-term memory (LSTM) networks, support vector machines (SVMs), and boosting ensemble decision trees, utilizing sensor data from a CNC machining center. These methods focus on the challenges and importance of feature selection, data preprocessing, and the application of tailored machine learning models to specific industrial tasks. Results show that SVM, with an accuracy of 96%, excels in handling high-dimensional data and robust feature extraction. In contrast, LSTM, which is appropriate for sequential data, is constrained by limited training data and the absence of pre-trained networks. Boosting ensemble decision trees also demonstrate efficacy in reducing model bias and variance. Conclusively, selecting an optimal machine learning strategy is crucial, depending on task complexity and data characteristics, highlighting the need for further research into domain-specific models to improve performance in industrial settings.

Keywords: boosting ensemble decision trees; long short-term memory (LSTM); support vector machine (SVM); tool condition monitoring (TCM); time series analysis

Citation: Przybyś-Małaczek, A.; Antoniuk, I.; Szymanowski, K.; Kruk, M.; Sieradzki, A.; Dohojda, A.; Szopa, P.; Kurek, J. Comparative Study of Conventional Machine Learning versus Deep Learning-Based Approaches for Tool Condition Assessments in Milling Processes. *Appl. Sci.* **2024**, *14*, 5913. https://doi.org/10.3390/app14135913

Academic Editor: Mark J. Jackson

Received: 2 June 2024
Revised: 28 June 2024
Accepted: 3 July 2024
Published: 6 July 2024

1. Introduction

When addressing problems related to the automation of furniture manufacturing, using sensors is a common practice, especially while evaluating various stages and trends. The problem is complex and involves multiple steps that require a high level of precision, as well as adjustments even after minor changes are introduced. In order to mitigate the amount of work required, incorporating technological advances into the workflow is a logical and necessary step. This is particularly true for issues related to tool condition monitoring. In such applications, incorrect or poorly timed decisions about replacing different elements can lead to poor product quality and financial losses for the manufacturing company. Finding ways to pinpoint the exact moment when the tool needs to be replaced is a vast area of research, with different approaches and available solutions [1–4].

Among different parts of the process, one that is a key focus of this paper is milling. During this stage, any inaccurate decision can be highly influential. To monitor the tool state during work, using a sensor-based approach offers a fresh perspective and significant improvement to the process. While manual tool checks can be done, they are time-consuming and require a pause in production. Automating the evaluation and providing the operator with a clear indication that the tool needs to be checked or exchanged is a significant advancement in the field [5,6].

While the sensor-based approach provides some interesting possibilities, the subject of tool condition monitoring is not new and remains a widely discussed one [7,8]. The edge of the tool gradually deteriorates during the process, which can result in decreased product quality. It is important to note that two situations need to be avoided in this case: unnecessarily stopping the production process and delaying the exchange past the point when the results start to be unacceptable. Bot situations can generate a loss for the company (either due to time or wasted materials), and any automatic solution should consider those. The decision-making process needs to be as precise as possible while providing interactive, insightful feedback for the operator. Incorporating specialized sensors in the process seems to be the best approach in that aspect.

While numerous works focus on similar problems, different signals are used, depending on the chosen tasks. The subsequent evaluation checks and verifies their usefulness in identifying tool conditions at different stages of the machining process. Additionally, even though extensive studies cover the problem, there is still a significant need for an automatic and precise solution that is easy to incorporate into the production process in the actual work environment. One of the significant innovations presented in this paper focuses on a way to solve that problem using sensor-based data.

Since the overall task is complex, applying machine learning (ML) to it seems to be the best approach. The applicability of ML algorithms and the importance of such solutions in manufacturing has grown in recent years. There are numerous examples of different algorithms being applied to similar tasks in image and sensor-based systems [9,10]. The work presented in this paper applies those techniques to tool condition monitoring, extending the overall scope with a novel approach to the task. Depending on the chosen approach, different aspects and applications are considered. At the same time, solutions such as tree species recognition, as shown in [11] show the vast possibilities and adaptability that ML offers. Even the most complicated tasks can be solved, assuming appropriate input data and a well-designed training workflow.

When specifically considering tool condition monitoring, the main distinction concerns the parts used. Some existing solutions use images instead of signals, often pairing such input with convolutional neural networks (CNNs) [1,2,12,13]. The training process itself can be improved by using pre-trained networks (such as AlexNet [14,15] prepared for the ImageNet database [16,17]) or data augmentation. Although preferable for ease of input collection, image-based solutions have some drawbacks. Firstly, to achieve high accuracy, large amounts of uniform training data are required. Secondly, the data acquisition process needs to be closely connected to the manufacturer. The key factors that need to be considered during this process are not always straightforward to derive.

In the aspect of initial problem definition, signals are a better solution, as the measurement process focuses on detecting any changes in them. At the same time, the computation required to process large amounts of data is an immediate issue that needs addressing.

Using sensor-based data as an input for neural networks can pose some problems. To do it efficiently, a new handling method was required, which in this case includes a novel approach to dealing with discrepancies and variations in sensor data. While changes will occur when the tool state deteriorates, not all sensors will produce the same amount of feedback. Because of this, pinpointing the actual moment when the tool needs to be replaced can be challenging. Additionally, the size of data obtained from different sensors differs significantly. The data files will be much larger, especially for sensors requiring higher precision. One of the main problems to consider is optimally using the obtained data while retaining the advantage of precise measurements.

One way to approach this issue is by transferring the signals into an image. For example, the authors of [18] do this with sound signals using short-time Fourier transform. The raw data were denoised and later converted into images, while a pre-trained CNN model performed deep feature extraction. The final step involves using a support vector machine for classification. A different approach uses the scalogram as a target format [19] for induction monitor state diagnosis. In this case, Constant-Q transform with non-stationary

Gabor transform is used. Vibration and acoustic single features are fused with a multi-input CNN solution. In the authors' opinion, the continuous wavelet transform (CWT) was too time-consuming given the initial input structure. Although the overall computation was faster, it also decreased the obtained.

This paper primarily aims to critically compare various machine learning approaches for tool condition monitoring in milling processes, utilizing time series data collected from the machining operations. The specific algorithms include support vector machines (SVMs), boosting ensemble decision trees, and long short-term memory (LSTM) networks. The research seeks to understand the comparative effectiveness of those methods in terms of their ability to accurately classify the tool state based on various performance metrics. This study hypothesizes that each algorithm will exhibit distinct advantages and limitations depending on the complexity and nature of the data involved. This comparative analysis is intended to identify which method provides the most reliable and accurate results under varying operational conditions.

2. Materials and Methods

2.1. Data Acquisition

Experiments were conducted on a Jet 130 CNC machining center manufactured by Busellato, located in Thiene, Italy. The machine was equipped with a 40 mm cutter head featuring a single, replaceable carbide cutting edge sourced from Faba SA in Baboszewo, Poland.

A chipboard panel, measuring 300 × 150 mm, served as the test material. This panel was securely affixed to a measurement platform where a 6 mm groove was milled. The milling was executed at a spindle speed of 18,000 rpm, a cutting speed of 37.68 m/s, with a feed rate of 0.15 mm per tooth. Operational parameters were selected based on a comprehensive review of relevant literature and practical experience in milling chipboard materials.

The condition of the tool was systematically categorized into one of three distinct states: green, yellow, or red. The 'green' state was indicative of a new or well-maintained tool. The 'yellow' category suggested moderate wear, which remained within operational limits. Finally, the 'red' class signaled a critical level of wear, necessitating immediate replacement. The maximum flank wear (VB_{max}) parameter served as the criterion for this classification.

Figure 1 depicts a microscopic examination of tool wear, highlighting the VB_{max} parameter utilized for assessing tool conditions. The wear measurement and subsequent classification into 'green', 'yellow', or 'red' states were performed using a Mitutoyo TM-505 microscope (Mitutoyo, Kawasaki, Japan), renowned for its precision in measuring dimensions and angles.

Figure 2 illustrates the schematic layout of the test stand used during the experiments. The setup includes the Busellato JET 130 CNC machine tool (Busellato, Thiene, Italy), various sensors (Kistler 9601A, 8141A, 8152B, B&K 4189 Microphone), amplifiers (Kistler 5125B, 5127B, 5036A, B&K Type 2690-A NEXUS Microphone Conditioner), and data acquisition cards (NI PCI-6034E, NI PCI-6111) connected to a PC for data collection.

The milling process was frequently paused to examine the condition of the blade using a Mitutoyo TM-505 microscope. The wear condition of the blade was quantified through those measurements, allowing classification into one of three distinct wear categories. The numerical ranges for each class were as follows:

1. A wear state classified as 'green' for maximum wear (VBMax) within 0 to 0.15 mm, encompassing five sublevels of wear;
2. A 'yellow' wear state for VBMax values ranging from 0.151 mm to 0.299 mm, incorporating five sublevels of wear;
3. A 'red' wear state for VBMax exceeding 0.299 mm, also with five sublevels of wear.

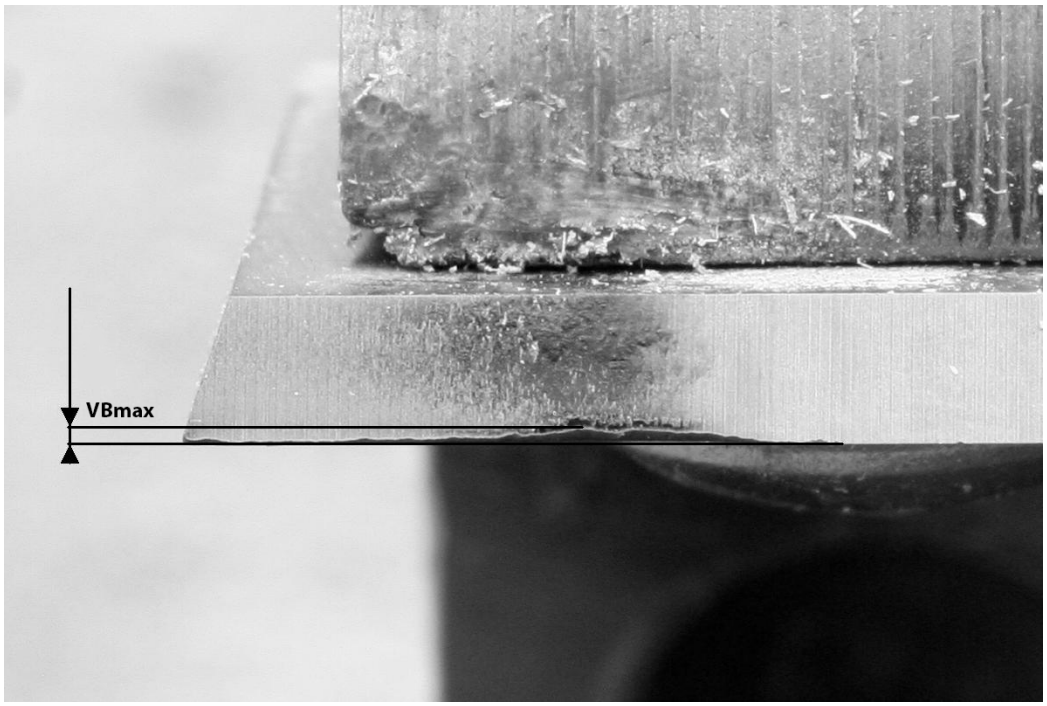


Figure 1. Illustration of tool wear measurement using the VB_{max} parameter.

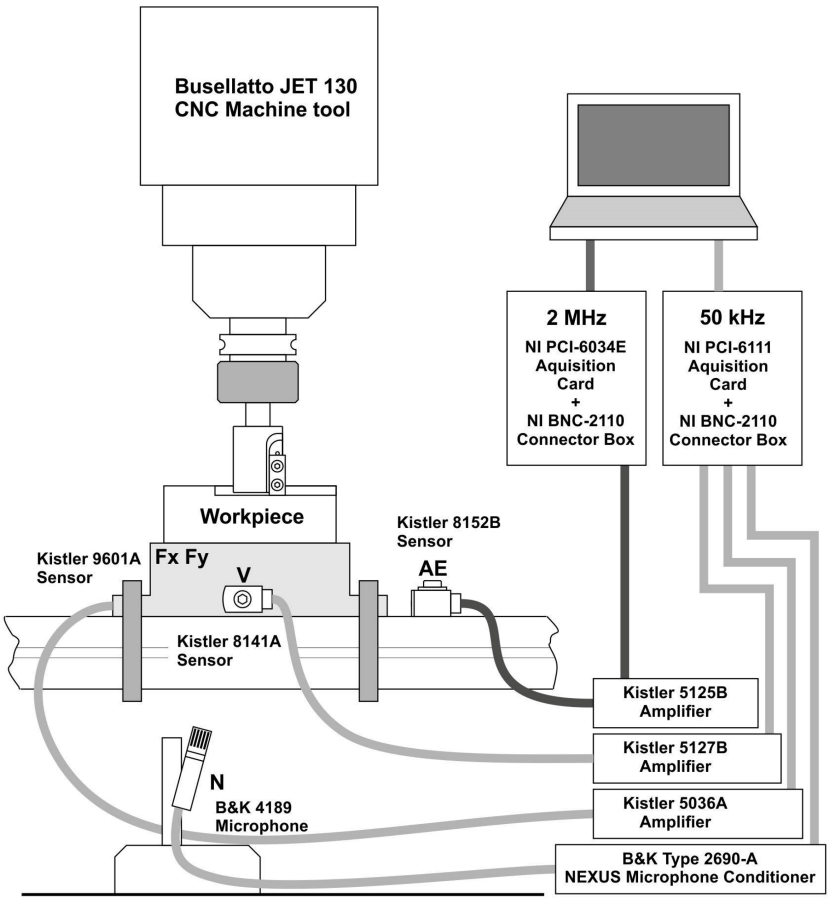


Figure 2. Scheme of the test stand layout used during the experiments.

Furthermore, the experimental setup was equipped with comprehensive sensors capable of recording 11 distinct signals, including force, acoustic emission, noise and

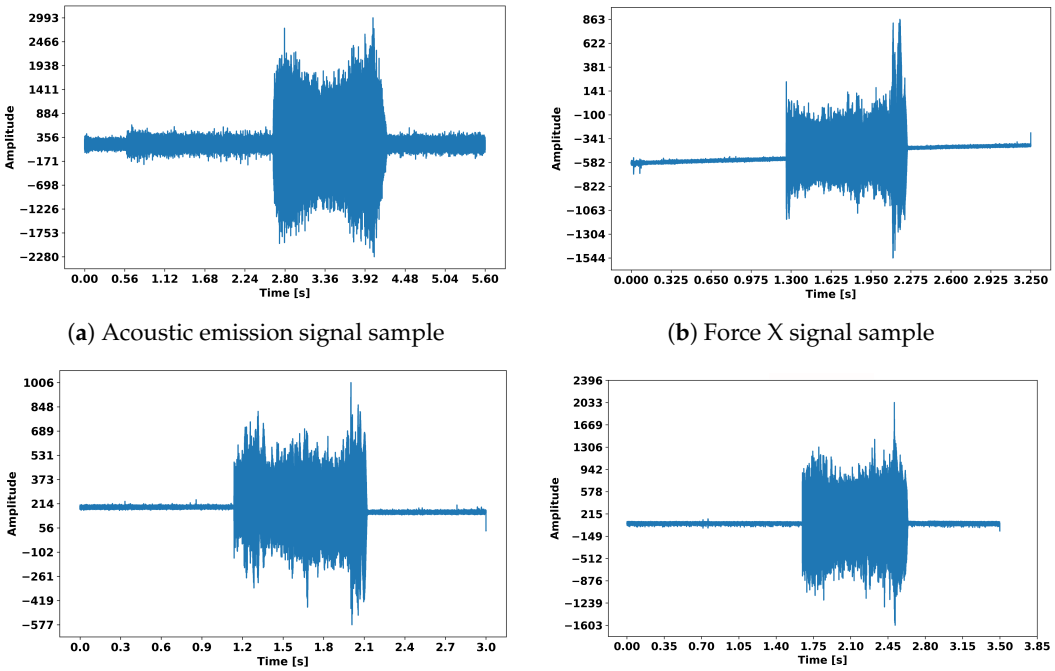
vibration levels, device-related current and voltage, head-rated current and voltage, as well as servo-rated current and voltage. The sensors used during experiments have the same specifications as the ones used in previous research, as outlined in [5].

The experimental system has multiple sensors with the ability to collect a total of 11 parameters, as follows:

- The orce value in the (1) *X* and (2) *Y*-axes (Kistler 9601A sensor; Impexron GmbH, Pfullingen, Germany);
- (3) acoustic emission (Kistler 8152B sensor; Kistler Group, Winterthur, Switzerland);
- (4) noise level (Brüel & Kjær 4189 sensor; Brüel and Kjær, Nærum, Denmark);
- (5) vibration level (Kistler 5127B sensor; Kistler Group, Winterthur, Switzerland);
- (6) device-rated current (Finest HR 30 sensor; Micom Elektronika, Zagreb, Croatia);
- (7) device-rated voltage (Testec TT-Si9001 sensor; Testec, Dreieich, Germany);
- (8) head-rated current (Finest HR 30 sensor; Micom Elektronika, Zagreb, Croatia);
- (9) head-rated voltage (Testec TT-Si9001 sensor; Testec, Dreieich, Germany);
- (10) servo-rated current (Finest HR 30 sensor; Micom Elektronika, Zagreb, Croatia);
- (11) servo-rated voltage (Testec TT-Si9001 sensor; Testec, Dreieich, Germany).

For the data acquisition part, two measurement cards were used. National Instruments PCI-6111 was used for acoustic emission, while the National Instruments PCI-6034E was utilized for other parameters. This setup ensures comprehensive and consistent analysis, and its efficiency was also confirmed during previous experiments [5]. For the data collection part, a PC with Lab ViewTM software was used. The software version from the National Instruments Corporation (Austin, TX, USA) was 2015 SP1.

The composition of the dataset assembled during this phase is detailed in Table 1. A total of 75 samples were collected for the experiments, 25 for each of the represented classes. The number of samples was set at this level to make sure that each class would contain highly representative examples, outlining the signal characteristics for this specific label. Figure 3 illustrates sample raw signal plots for acoustic emission, *X*-axis force, *Y*-axis force, and noise levels.



(**a**) Acoustic emission signal sample

(**b**) Force X signal sample

(**c**) Force Y signal sample

(**d**) Noise level signal sample

Figure 3. Illustrative examples of raw signal data for (**a**) acoustic emission, (**b**) *X*-axis force, (**c**) *Y*-axis force, and (**d**) noise levels.

Table 1. Overview of variable configuration within datasets.

Dataset Name	Measured Parameter	Trial Duration	Sampling Rate (Hz)	Observation Time (s)
HighData	Acoustic Emission	27,999,960	5,000,000	5.59
LowData	*X*-axis Force	700,000	200,000	3.50
LowData	*Y*-axis Force	700,000	200,000	3.50
LowData	Noise	700,000	200,000	3.50
LowData	Vibration	700,000	200,000	3.50
CurrentData	Device Current	30,000	50,000	0.60
CurrentData	Device Voltage	30,000	50,000	0.60
CurrentData	Head Current	30,000	50,000	0.60
CurrentData	Head Voltage	30,000	50,000	0.60
CurrentData	Servo Current	30,000	50,000	0.60
CurrentData	Servo Voltage	30,000	50,000	0.60

2.2. Data Preprocessing

Data preprocessing is a critical step in preparing the collected sensor data for analysis. The min–max normalization was applied to scale the features to a fixed range—[0, 1]. This technique is essential to ensure that all features contribute equally to the machine learning models, especially those sensitive to the input data scale.

The formula for min–max normalization is as follows:

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}$$

where x is the original value, x' is the normalized value, x_{min} is the minimum value of the feature, and x_{max} is the maximum value of the feature. By applying this normalization, all features are scaled proportionally, which helps mitigate the bias toward features with larger ranges and improves the overall performance of the machine learning models.

2.3. Features Generation in the Time Domain

Feature extraction in the time domain is a crucial aspect of signal analysis, particularly for applications such as tool condition monitoring in milling processes. In this study, we extracted 11 distinct time-domain features from the signal data for every 11 signals. These features include the following: mean (μ), RMS (root mean square), standard deviation (σ), shape factor, SNR (signal-to-noise) ratio, THD (total harmonic distortion), SINAD (signal-to-noise and distortion) ratio, peak value (x_{peak}), crest factor, clearance factor, and impulse factor [20].

These features provide a comprehensive insight into the characteristics of the signal, facilitating an effective analysis of tool conditions. The feature extraction process involved computing these mathematical formulas for each signal captured, thereby enabling a detailed examination of the signal's properties.

In this study, a total of 11 features were generated for each of the 11 signals, resulting in 121 features from the time domain.

Below, we detail these features along with their mathematical formulations [20–22]:

1. Mean (μ): The average value of the signal; providing a measure of the signal's central tendency.

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i \quad (1)$$

2. RMS (Root Mean Square): Represents the square root of the mean of the squares of all values in the signal, indicating its overall energy.

$$RMS = \sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2} \quad (2)$$

3. Standard Deviation (σ): Measures the amount of variation or dispersion from the average signal value.

$$\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu)^2} \tag{3}$$

4. Shape Factor: Describes the shape of the signal's waveform by comparing it with an ideal shape.

$$Shape\ Factor = \frac{RMS}{\frac{1}{N}\sum_{i=1}^{N}|x_i|} \tag{4}$$

5. SNR (Signal-to-Noise Ratio): Compares the level of the desired signal to the level of background noise.

$$SNR = 10\log_{10}\left(\frac{P_{signal}}{P_{noise}}\right) \tag{5}$$

6. THD (Total Harmonic Distortion): Indicates the distortion of the signal by measuring the harmonics.

$$THD = \frac{\sqrt{\sum_{n=2}^{N} P_n}}{P_1} \tag{6}$$

7. SINAD (Signal-to-Noise and Distortion ratio): Represents the ratio of the total power of the signal to the undesired signal's power (noise plus distortion).

$$SINAD = 10\log_{10}\left(\frac{P_{signal}}{P_{noise} + P_{distortion}}\right) \tag{7}$$

8. Peak Value (x_{peak}): The maximum absolute value of the signal, showing its amplitude.

$$x_{peak} = \max(|x_i|) \tag{8}$$

9. Crest Factor: The ratio of the peak value of the signal to its RMS value; used to assess signal peaks.

$$Crest\ Factor = \frac{x_{peak}}{RMS} \tag{9}$$

10. Clearance Factor: A measure that evaluates the signal's peaks in relation to the RMS value; similar to the crest factor but emphasizes the extreme peaks more.

$$Clearance\ Factor = \frac{x_{peak}^{3/2}}{\sqrt{\frac{1}{N}\sum_{i=1}^{N}|x_i|^{3/2}}} \tag{10}$$

11. Impulse Factor: The ratio of the peak value to the mean value of the signal; indicates the impulsiveness of the signal.

$$Impulse\ Factor = \frac{x_{peak}}{\mu} \tag{11}$$

2.4. Features Generation in Frequency Domain

In the frequency domain, the following features are generated: mean frequency, median frequency, band power, occupied bandwidth, power bandwidth, peak amplitude, and peak Location [23–25].

The extraction of these features from the frequency domain is pivotal for a comprehensive analysis, offering nuanced insights into the tool's condition and operational efficiency.

For the frequency domain, a total of seven distinctive features were generated in order to enhance the performed analysis. The definitions and formulas for these features are as follows [23–26]:

1. Mean Frequency: The average of all frequency components, weighted by their amplitudes.

$$\text{Mean Frequency} = \frac{\sum f \cdot P(f)}{\sum P(f)} \tag{12}$$

where f represents the frequency components, and $P(f)$ denotes the power spectral density at frequency f.

2. Median Frequency: The frequency at which the power spectrum is divided into two equal halves.

$$\int_0^{\text{Median Frequency}} P(f)\, df = \frac{1}{2} \int_0^{\infty} P(f)\, df \tag{13}$$

3. Band Power: The sum of spectral power within a specified frequency band.

$$\text{Band Power} = \int_{f_{\text{low}}}^{f_{\text{high}}} P(f)\, df \tag{14}$$

where f_{low} and f_{high} are the lower and upper bounds of the frequency band.

4. Occupied Bandwidth: The frequency bandwidth contains a specified percentage (e.g., 90%) of the total power.

$$\int_{f_{\text{low}}}^{f_{\text{high}}} P(f)\, df = 0.9 \cdot \int_0^{\infty} P(f)\, df \tag{15}$$

5. Power Bandwidth: The width of the frequency band within which a significant portion of the signal power is concentrated.

$$\text{Power Bandwidth} = f_{\text{high}} - f_{\text{low}} \tag{16}$$

where f_{low} and f_{high} are defined such that they encompass a specified percentage of the total power.

6. Peak Amplitude: The maximum amplitude within the specified frequency band.

$$\text{Peak Amplitude} = \max(P(f)) \tag{17}$$

where the maximum is taken over the specified frequency band.

7. Peak Location: The frequency at which the peak amplitude occurs.

$$\text{Peak Location} = \text{argmax}(P(f)) \tag{18}$$

where argmax returns the frequency at which $P(f)$ is maximized.

In this study, a total of 7 features were generated for each of the 11 signals, resulting in 77 features from the frequency domain.

2.5. Features Generation in Time–Frequency Domain

For the time–frequency domain, the *MeanEnvelopeEnergy* feature is used to analyze tool wear, capturing the mean energy of the upper and lower envelopes for each intrinsic mode function (IMF) [27–30]. The generation involves computing the mean envelope energy of the IMFs derived from the input time-domain signals, for which the empirical mode decomposition (EMD) [31] method was applied. The final result was calculated by averaging the energy of the upper and lower envelopes for each IMF. By doing this, the signal's energy distribution is encapsulated in it. The resulting feature provides a compact yet informative representation of the signal's energy characteristics, pivotal for assessing tool wear in the milling processes. Each of the 11 original signals produced a 10-element vector, and a total of 110 potential features (*MeanEnvelopeEnergy*).

Mathematically, the *MeanEnvelopeEnergy* feature can be defined as follows:

$$MeanEnvelopeEnergy = \frac{1}{N}\sum_{n=1}^{N}\left(\frac{1}{M_n}\sum_{m=1}^{M_n}\sqrt{U_m^n(t)^2 + L_m^n(t)^2}\right) \quad (19)$$

where:

- N is the total number of IMFs considered for each signal.
- M_n represents the number of discrete time points in the n-th IMF.
- $U_m^n(t)$ and $L_m^n(t)$ denote the upper and lower envelope values at time t for the m-th time point in the n-th IMF, respectively.

2.6. Summary of Feature Generation Techniques

We consolidate three distinct approaches to feature generation: time domain, frequency domain, and time–frequency domain analyses, which are applied to monitoring the milling process. The cumulative exploration led to the identification of 308 potential features, offering a comprehensive toolkit for enhancing recognition accuracy.

In traditional machine learning approaches such as support vector machines (SVMs) or boosting ensemble decision trees, the feature generation process typically involves extracting relevant information from the dataset without considering sequential dependencies. Each training sample is represented as a single point in a high-dimensional feature space, with a size of 1×308. This method is effective for models that do not take into account the temporal sequence of the data.

However, when dealing with long short-term memory (LSTM) networks, the scenario changes significantly. LSTMs are designed to recognize patterns in sequences of data, making them particularly suitable for time-series analysis. In those applications, the order and context of the data points are crucial. For LSTMs, the feature generation process must incorporate sequential information, leading to a transformation in the data representation.

In the context of the presented application, while traditional machine learning models treat the entire dataset as a single block (frame size equal to the length of the signal, resulting in a sample size of 1×308), LSTM requires the data to be divided into smaller sequences. This division results in the creation of 100 frames from the original signal, with each frame being a sequence of data points. The frame size for LSTM is determined by dividing the length of the signal by 100 and rounding down to the nearest integer (floor function). This ensures that each sequence captures a portion of the temporal pattern present in the data. Consequently, for LSTM, the dimensionality of each training sample transforms to 100×308, with each sequence representing a step in the temporal pattern.

This fundamental difference in feature generation highlights the versatility of LSTM in handling time-series data, enabling the model to capture the dynamic changes over time. This factor is particularly advantageous in applications such as tool state recognition. The ability of LSTM to process data in sequences allows it to learn from the temporal dependencies and variations in the signal, providing a more nuanced understanding of the changes in the tool's condition over time.

2.7. Support Vector Machines (SVMs)

A support vector machine (SVM) [32] is a supervised learning method that creates a hyperplane or set of hyperplanes in a high-dimensional space. The algorithm itself has a wide variety of applications, including classification. In this case, good separation (called a functional margin) is achieved by finding a hyperplane with the highest distance to the nearest data points of any given class. A larger margin in this case indicates that the algorithm can generalize better, improving the classification quality.

SVMs are effective tools for solving problems with high-dimensional feature spaces and non-linear data structures. They are particularly useful when the number of dimensions exceeds the number of samples. SVMs are well-suited for complex machine learning

problems, such as the ones presented in this paper, where traditional algorithms may struggle to find an accurate solution.

The parameters used for training the SVM are as follows:

- method: onevsone;
- kernel: radial basis function (RBF);
- C (regularization parameter): 100;
- gamma (kernel coefficient): 0.001.

Feature Selection for SVM Using Sequential Feature Selection

Sequential feature selection (SFS) [33] is a method used in machine learning to choose a subset of important features for building a model. SFS aims to reduce the number of input features to enhance the model's performance. This method is especially helpful when dealing with datasets that have a large number of parameters, some of which may be unnecessary or duplicative.

Sequential feature selection for SVM operates as follows:

1. Define a random, non-stratified partition for 10-fold cross-validation on n observations, where n is the number of observations in the dataset.
2. Initialize the selected feature set S as an empty set.
3. For each feature x_i not in S, compute the cross-validated criterion value using the SVM function. This involves training the SVM model on the dataset including x_i and evaluating its performance.
4. Add the feature x_i with the smallest criterion value (or the largest if the criterion is accuracy or another measure where higher values are better) to S.
5. For each feature x_i not in S, define a candidate feature set C_i as $S \cup \{x_i\}$. Compute the cross-validated criterion value using the SVM function for C_i.
6. Among the candidate sets C_i, select the set that reduces (or increases) the criterion value the most when compared to the criterion value for S. Add the feature corresponding to the selected candidate set to S.
7. Repeat steps 5 and 6 until adding a feature does not decrease (or increase) the criterion value by more than the termination tolerance value.

This process results in a subset of features that are deemed most relevant for the SVM model, thereby potentially improving the model's performance by reducing the overfitting and improving generalization.

2.8. Boosting Ensemble Decision Trees

In the presented approach, an ensemble classifier that combines the decisions of 100 individual classification trees is constructed. Each tree contributes to the final decision. Subsequent trees focus on the errors made by their predecessors. This process not only amplifies the ensemble's ability to learn from the training data but also enhances its generalization to new, unseen data [34].

For the boosting ensemble decision trees training phase, the following hyperparameters were used:

- NLearn: 100;
- Method: Bag;
- MinLeaf: 2;
- Prune: off;
- MergeLeaves: off;
- LearnRate: 0.13717.

This setup, particularly the method 'Bag', aims to create an ensemble of trees where each tree is trained on a random subset of the data. This randomization helps in reducing overfitting and improving the model's robustness. The learning rate of 0.13717 controls the contribution of each tree to the final model, ensuring that the ensemble learns gradually and avoids overfitting.

The use of Bayesian optimization for hyperparameter tuning led to a more efficient and accurate boosting ensemble decision trees model. This improvement minimized model overfitting and maximized its predictive accuracy.

2.9. Long Short-Term Memory (LSTM)

The LSTM model is designed to process varying lengths of input sequences initiated by a sequence input layer. The architecture includes two LSTM layers: the first has 750 units with the output mode set to 'sequence', enabling it to return sequences for the subsequent layers, and the second has 500 units with the output mode set to 'last', which helps in reducing the output to the final state only. Between and after these LSTM layers, dropout layers with a rate of 0.2 are applied to mitigate overfitting by randomly omitting a portion of the features during training. The model concludes with a fully connected layer that maps to the number of classes, followed by a softmax layer for classification probabilities and a final classification layer. The structure is summarized as follows [35]:

```
layers = [ ...
    sequenceInputLayer(numFeatures)
    lstmLayer(750, 'OutputMode', 'sequence')
    dropoutLayer(0.2)
    lstmLayer(500, 'OutputMode', 'last')
    dropoutLayer(0.2)
    fullyConnectedLayer(numClasses)
    softmaxLayer
    classificationLayer];
```

The training utilized the Adam optimizer, with the following hyperparameters set:

- learning rate: 0.001;
- maxEpochs: 1000;
- batchSize: 128;
- L2 Regularization: 0.0001.

The training utilized the Adam optimizer, with settings adjusted to accommodate GPU execution for enhanced performance. The options included shuffling data every epoch, displaying training progress plots, and setting a maximum of 1000 epochs.

The model's performance was evaluated using k-fold cross-validation, with k set to 10. This approach partitioned the data into 10 sets, using 9 for training and 1 for testing in each fold. Predictions were made on the test sets, and the results were aggregated to compute the overall performance metrics, such as accuracy and the confusion matrix.

3. Results

3.1. Numerical Experiments

In this study, numerical experiments were conducted to evaluate the performance of the three machine learning approaches: support vector machines (SVMs), boosting ensemble decision trees, and long short-term memory (LSTM) networks. The experiments were designed to classify the tool condition into three distinct classes: green, yellow, and red. A total of 75 samples were used, with each class containing 25 samples. All experiments were performed in Matlab 2024a computational environment on a PC with Ubuntu operating system. The machine specification is the same as in previous experiments, and is presented in [5].

To ensure a robust evaluation of the models, k-fold cross-validation was applied, with $k = 10$ representing the number of folds. This method involves partitioning the data into k subsets (folds). Each model is trained on $k - 1$ folds and tested on the remaining fold, repeating this process k times, such that each fold is used exactly once as a test set. The results are then averaged to provide an overall performance metric.

The dataset for each class was balanced, with 25 samples per class, ensuring that each model had an equal representation of each tool condition during training and testing. This

balance is crucial to avoid biased results and ensure that the models can generalize well to new, unseen data.

For each approach, the following steps were taken:

1. Data partitioning using k-fold cross-validation.
2. Training the model on $k-1$ folds and testing on the remaining fold.
3. Repeating the process k times to ensure each fold is used as a test set.
4. Averaging the performance metrics to obtain an overall evaluation of the model.

The k-fold cross-validation method provided a comprehensive evaluation of each model, ensuring that the results were reliable and could be generalized to new data. This approach also helped in identifying the strengths and weaknesses of each model in classifying the tool condition based on the given sensor data.

3.2. Results for SVM

The parameters used for training the SVM are as follows:

- Standardize the predictors: Standardizing the predictors ensures that each feature contributes equally to the model by scaling them to have a mean of zero and a standard deviation of one.
- RBF kernel: The radial basis function (RBF) kernel is used as the kernel function, which helps in handling non-linear data by mapping it into a higher-dimensional space.
- FitPosterior: This option is enabled to estimate posterior probabilities for classification, providing probabilistic outputs for each class.
- onevsone: For each binary learner, one class is positive, another is negative, and the software ignores the rest. This design exhausts all combinations of class pair assignments, ensuring comprehensive coverage of all class pairs for multi-class classification.
- Hyperparameter optimization: Bayesian optimization (BoxConstraint = 212.13, KernelScale = 3.5563, coding = onevsone, standardize = true)

The confusion matrix for SVM classification (Figure 4) presents strong performance in differentiating between the three classes: green, yellow, and red. For the green class, the model has a perfect classification with 25 true positives and no false positives or false negatives. This indicates that every instance that was actually green was predicted as green, and no instances of other classes were incorrectly predicted in this case.

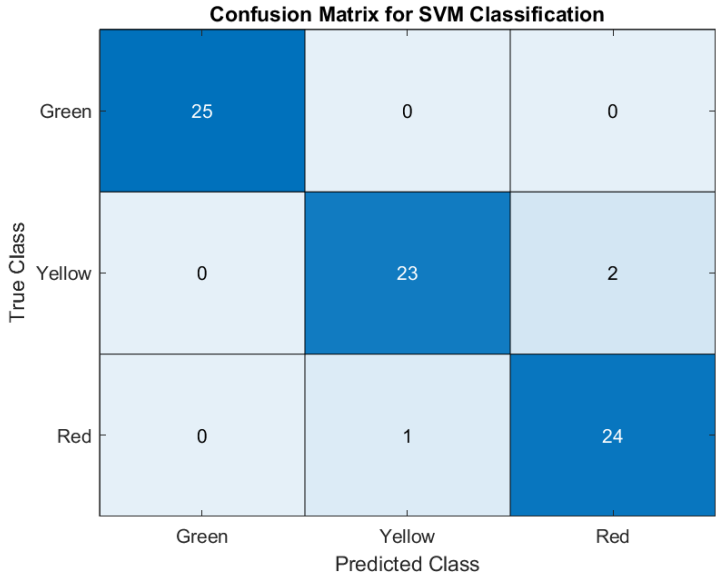


Figure 4. Confusion matrix for SVM. Numbers on the diagonal denote the correctly classified cases. Numbers off the diagonal denote cases that were not correctly classified. The colors used represent the proportions of the examples, and further visualize the number of cases for each case, with lighter values denoting lower counts.

The yellow class also shows a strong classification performance, with 23 true positives. However, there are two instances where the yellow class was incorrectly predicted as red. This represents a small classification error for yellow, showing that the SVM model occasionally confuses yellow with red.

The red class has 24 true positives, demonstrating that the SVM model is generally effective in identifying this class as well. There is only a single instance where red was misclassified as yellow, indicating a very high accuracy in classifying red instances as well.

The SVM model demonstrates excellent performance with a total of 3 misclassifications out of 75 instances, resulting in a high overall accuracy. The model is particularly precise in classifying green, with slightly lower precision for yellow and red due to minimal confusion between these two classes.

The support vector machine (SVM) classifier demonstrated outstanding performance across various metrics for each class, as summarized in Table 2. For the 'green' class, the SVM achieved perfect scores in all metrics with precision, sensitivity (recall), specificity, F1 score, and accuracy of 100.00%. This indicates the SVM model's exemplary capability to classify the 'green' class without any errors.

Table 2. Performance metrics of the SVM model, denoting the key parameters used for model evaluation for each of the recognized classes.

Class	Precision	Sensitivity	Specificity	F1Score	Accuracy
Green	100.00%	100.00%	100.00%	100.00%	100.00%
Yellow	95.83%	92.00%	98.00%	93.87%	96.00%
Red	92.30%	96.00%	96.00%	94.11%	96.00%
Overall					96.00%

For the 'yellow' class, the classifier also showed high effectiveness with a precision of 95.83%, sensitivity of 92.00%, and specificity of 98.00%. The F1 score for the 'yellow' class was 93.87%, and the accuracy was reported at 96.00%. These results signify a high degree of reliability in classifying the 'yellow' class, with a very small margin of error.

For the 'red' class, the SVM achieved a precision of 92.30%, sensitivity of 96.00%, and specificity of 96.00%. The F1 score was slightly lower at 94.11%, with the accuracy for the 'red' class also at 96.00%. This denotes that the classifier was slightly less precise with the 'red' class compared to the 'green' class but still maintained high accuracy.

Overall, the SVM classifier's performance was robust, with an overall accuracy of 96.00% across all classes, showcasing its capability as a reliable model for classification tasks in the given context.

3.3. Results for Boosting Ensemble Decision Trees

The parameters used for training boosting ensemble decision trees are as follows:

- Method: AdaBoostM2—boosting method for multi-class classification.
- NumLearningCycles: 96—the number of learning cycles in the ensemble.
- LearnRate: 0.58881—the learning rate for the ensemble.
- MinLeafSize: 5—the minimum number of observations per leaf in the decision tree.
- Hyperparameter optimization: Bayesian optimization.

The confusion matrix depicted in Figure 5 shows the performance of the optimized boosting ensemble decision trees classification model, applying a boosting approach with 100 trees for the three wear classes used for classification.

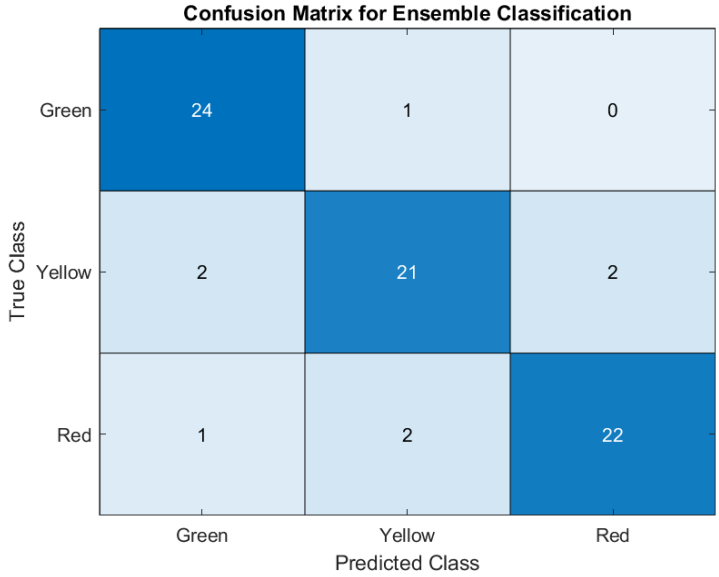


Figure 5. The confusion matrix for optimized boosting ensemble decision trees. Numbers off the diagonal denote cases that were not correctly classified. The colors represent the proportions of the examples, further visualizing the number of cases for each category, with lighter values denoting lower counts.

For the green class, the model exhibits commendable accuracy, with the majority of instances correctly classified, reflecting a high sensitivity rate of 96%. The precision for this class stands at 88.88%, indicating a relatively lower, but still substantial, likelihood of instances being accurately identified as green out of all instances predicted as green.

The yellow class presents a challenge with a sensitivity of 84%, showing some instances of misclassification. The precision rate for yellow, at 87.50%, indicates that while the model can identify yellow instances with reasonable accuracy, there is room for improvement in reducing false positives.

The red class showcases a sensitivity of 88% and a precision of 91.66%, indicating a strong ability of the model to identify red instances accurately, albeit with a small margin of error.

Table 3 further details the performance metrics of the boosting ensemble decision trees model. It highlights the model's strengths in specific areas, such as high precision and sensitivity for the red class. It identifies areas for improvement, such as the slightly lower sensitivity for the yellow class.

Table 3. Performance metrics of the optimized boosting ensemble decision trees, denoting the key parameters used for model evaluation for each of the recognized classes.

Class	Precision	Sensitivity	Specificity	F1Score	Accuracy
Green	88.88%	96.00%	94.00%	92.30%	94.66%
Yellow	87.50%	84.00%	94.00%	85.71%	90.66%
Red	91.66%	88.00%	96.00%	89.79%	93.33%
Overall					89.33%

Overall, the optimized boosting ensemble decision trees model demonstrates robust performance with an overall accuracy of 89.33%. The model effectively balances precision and sensitivity across classes, making it a reliable tool for classification tasks in this context. Future work may focus on enhancing the model's sensitivity for the yellow class to improve its applicability and effectiveness further.

3.4. Results for LSTM

The parameters used for training LSTM are as follows:

- Optimizer: adam—optimizer for training.
- ExecutionEnvironment: GPU—utilizes GPU for training.
- Shuffle: every-epoch—shuffles the data every epoch.
- MaxEpochs: 1000—sets the maximum number of training epochs to 1000.
- No of lstmLayer: 750

The performance evaluation of the long short-term memory (LSTM) model is illustrated through the confusion matrix in Figure 6 and the detailed performance metrics in Table 4. The LSTM model was similarly assessed across three distinct classes: green, yellow, and red, representing different states of the tool used in the milling process.

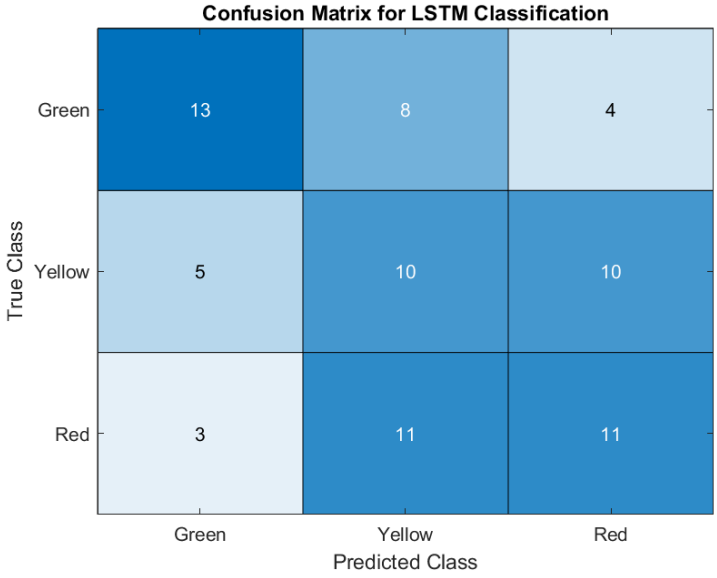


Figure 6. Confusion matrix for LSTM. Numbers off the diagonal denote cases that were not correctly classified. The colors used represent the proportions of the examples and visualize the number of cases for each case, with lighter values denoting lower counts.

Table 4. Performance metrics of LSTM denoting the key parameters used for model evaluation for each of the recognized classes.

Class	Precision	Sensitivity	Specificity	F1Score	Accuracy
Green	61.90%	52.00%	84.00%	56.52%	73.33%
Yellow	34.48%	40%	62.00%	37.03%	54.66%
Red	44.00%	44.00%	72.00%	44.00%	62.66%
Overall					45.33%

The confusion matrix reveals the model's performance and challenges in classifying the tool states. The green class shows a precision of 61.90%, with the model correctly identifying 52% of green instances but also exhibiting a relatively high misclassification rate, leading to a specificity of 84%. This indicates a difficulty in distinguishing green states from others, potentially due to overlapping characteristics with yellow and red states.

For the yellow class, the LSTM model achieves a precision of 34.48% and a sensitivity of 40%, suggesting a significant challenge in accurately identifying this intermediate tool state. The model's specificity of 62% for the yellow class further underscores the difficulty in classifying instances that may not exhibit clear-cut features of wear or optimal condition.

The red class results indicate a precision and sensitivity of 44%, with a specificity of 72%. These figures suggest that while the LSTM can recognize severely worn tools at

a moderate rate, there is still substantial room for improvement in distinguishing these instances from less worn states.

Table 4 provides a comprehensive breakdown of the LSTM's performance, quantifying its accuracy, precision, sensitivity, and F1 scores across all classes. Notably, the overall accuracy of the LSTM model stands at 45.33%, highlighting the challenges faced by the model in this multi-class classification task.

3.5. Comparison

The performance comparison of the three models, support vector machine (SVMs), boosting ensemble decision trees, and long short-term memory (LSTM) neural network, is summarized in Table 5. The comparison focuses on two main aspects: the number of critical errors and the average accuracy achieved by each model.

Table 5. Comparative analysis of the performance metrics across three different classifiers. Two main parameters are shown: accuracy, which measures the overall amount of correctly classified cases, and the number of critical errors (mistakes between green and red classes) that each classifier performed.

Classifier	Critical Errors	Training Time	Prediction Time	Accuracy
SVM	0	2 h 34 m 12 s	0.12 s	96.00%
Boosting Ensemble DT	1	18 m 29 s	0.05 s	89.33%
LSTM	7	13 m 17 s	0.69 s	45.33 %

The SVM model outperforms the others with the highest average accuracy of 96.00% and no critical errors. This demonstrates its robustness and reliability for the tool state recognition task in milling processes. This high performance can be attributed to SVM's effectiveness in handling high-dimensional data and its capability of finding the optimal hyperplane that separates the different tool states.

The boosting ensemble decision trees, applying a boosting approach with 100 trees, show decent performance with an average accuracy of 89.33%. They encounter only one critical error, indicating reasonable reliability. The ensemble's strength lies in its ability to reduce bias and variance, combining multiple weak learners to improve overall prediction accuracy.

In contrast, the LSTM model exhibits a significant drop in performance with the lowest average accuracy of 45.33% and the highest number of critical errors (7). This may be due to the LSTM's sensitivity to parameter settings and the complexity of time-series data in tool state recognition, which poses challenges in capturing temporal dependencies effectively.

The comparative analysis of the LSTM, SVM, and boosting ensemble decision trees reveals significant insights into the challenges and advantages of each model in the context of tool state recognition in milling processes. The low effectiveness of the LSTM model is particularly noteworthy, with its challenges attributed to the limited training data, the absence of feature selection for sequential data, and the lack of domain-specific pre-trained models for time series analysis. These factors collectively hinder the LSTM's ability to learn and generalize effectively, leading to its diminished performance compared to SVM and boosting ensemble decision trees.

In contrast, the SVM model's superior performance underscores the importance of appropriate feature selection and the advantage of methods well-suited to high-dimensional data. The boosting ensemble decision trees, with their inherent feature selection capabilities and ensemble learning approach, also demonstrate commendable performance, though not matching the SVM's effectiveness.

This comparison not only highlights the strengths and weaknesses of each model but also points to broader challenges in applying deep learning models like LSTM to time series data in industrial applications. The absence of pre-trained networks and the critical need for extensive training data and effective feature selection mechanisms are evident challenges that need to be addressed in future research.

Efforts to develop domain-specific pre-trained models for time series data, advanced feature selection techniques, and strategies to augment training data could significantly enhance the applicability and performance of deep learning models in this domain. Such advancements would not only improve model accuracy and reliability but also expand the potential of machine learning in industrial and manufacturing settings, leading to more efficient and automated processes.

The choice of a traditional machine learning or deep learning algorithm critically depends on the nature of the data, the specific requirements of the application, and the desired outcome of the analysis. Table 6 provides a detailed comparative analysis of three widely used algorithms: support vector machines (SVMs), boosting ensemble decision trees, and long short-term memory (LSTM) networks. Each of these classifiers has distinct characteristics that make them suitable for different types of machine learning or deep learning tasks.

Table 6. Comparative analysis of algorithms, denoting requirements for sequential data, key challenges that each of them faces, and the main advantages.

Classifier	Sequential Data Required	Challenges	Advantages
SVM	No	Sensitive to feature scaling, kernel choice critical	Effective in high dimensional spaces
Boosting Ensemble DT	No	Prone to overfitting; requires careful tuning	Good performance with non-linear data
LSTM	Yes	Requires large dataset, complex model tuning	Excellent with sequential data

The suitability of each classifier varies depending on the specific challenges and data characteristics of the task at hand. SVMs are preferred in high-dimensional environments where the main goal is to create a clear margin of separation between classes. Boosting decision trees are often chosen for their flexibility and effectiveness in solving regression and classification problems involving complex datasets with non-linear relationships. LSTMs are useful in scenarios where understanding the dynamics of data over time is crucial for prediction.

4. Discussion

The research presented in this paper evaluates the efficacy of different machine learning models—SVM, LSTM, and boosting ensemble decision trees—in the context of tool condition monitoring in the milling processes.

Support vector machines (SVMs) are particularly renowned for their ability to handle high-dimensional data efficiently. They work by finding a hyperplane that best divides a dataset into classes, which is particularly useful in complex classification problems where the decision boundaries are not readily apparent. The major challenges with SVM involve its sensitivity to the choice of kernel and the scaling of features; the right choice of kernel and proper feature scaling can significantly influence the model's performance. Despite these challenges, SVM's effectiveness in high-dimensional spaces makes it an excellent choice for classification tasks where the number of features exceeds the number of samples.

The superiority of SVM in our experiments can be theoretically attributed to its structural risk minimization principle, which effectively handles the high-dimensional space of the features extracted from the milling data. The SVM's ability to construct a hyperplane in a high-dimensional space provides an optimal separation boundary between the classes, which is critical given the subtle variations in the tool's wear condition. On the other hand, the LSTM model underperformed, which could be due to its reliance on large datasets to capture complex dependencies in sequence data observed, while in our study it had limited data size and lacked a pre-trained network for times series data.

Boosting ensemble decision trees, such as those used in gradient-boosting machines (GBMs), harness the power of many weak decision trees to create a robust classifier. By focusing on correcting the errors of previous trees in a sequence, boosting methods can dramatically improve model accuracy. However, these models are susceptible to overfitting, especially when the data are noisy or when the trees are too complex. They also require careful tuning of parameters such as the number of trees and learning rate to balance bias and variance effectively. Nevertheless, their strength in handling non-linear data makes them suitable for complex data sets where the relationships between variables are nonlinear.

Boosting ensemble decision trees showed a balanced performance, leveraging the strengths of multiple weak learners to improve classification accuracy. The ensemble approach is particularly effective for non-linear and complex decision boundaries that characterize tool wear data.

A long short-term memory (LSTM) network, a type of recurrent neural network, is uniquely capable of analyzing sequential data. This makes this network ideal for applications such as time-series analysis, natural language processing, and other tasks where the order of data points is crucial. LSTMs can capture long-term dependencies in data sequences, a feature that traditional methods may fail to encapsulate. However, LSTMs come with their own set of challenges, including the need for large amounts of data to train effectively and the complexity involved in tuning multiple hyperparameters such as the number of layers and the size of the LSTM units. Despite these challenges, the ability of LSTMs to process sequences and their robustness to "input shifts" (changes in the input data distribution over time) make them invaluable for dynamic systems analysis.

The results contribute to the ongoing debate regarding the selection of appropriate machine learning techniques for predictive maintenance in manufacturing. While SVM and boosting algorithms are well-established in the literature, their comparative analysis with deep learning techniques like LSTM in the specific context of milling operations offers new insights. This study highlights the need for further investigation into hybrid models that combine the memory capability of LSTM with the robust classification features of SVM or decision trees, potentially leading to models that better capture the temporal and spatial characteristics of tool wear.

Furthermore, our findings underscore the importance of developing domain-specific pre-trained models for time-series analysis in industrial applications. This could significantly reduce the data requirements and computational costs associated with training models from scratch.

This study is not without limitations. The relatively small dataset and the specific focus on milling operations limit the generalizability of the findings. Future research could explore the application of these models across different types of machining processes and with more diverse datasets to validate and extend our results. Additionally, integrating sensor fusion techniques to combine multiple types of data, such as vibration, acoustic, and force measurements, may enhance the accuracy and robustness of the predictive models. Finally, exploring feature engineering and selection methods tailored to sequential data can further optimize LSTM performance.

Improvements Achieved

During this study, the following key improvements were realized:

- Feature selection for SVM: Implementing sequential feature selection (SFS) significantly enhanced the performance of the SVM model by reducing the feature space and improving the model's ability to generalize from the training data.
- Optimizing boosting ensemble decision trees: Utilizing Bayesian optimization for hyperparameter tuning led to a more efficient and accurate boosting ensemble decision trees model. This improvement minimized model overfitting and maximized its predictive accuracy.

- Enhanced data preprocessing: Improved data preprocessing techniques, including standardization and normalization, ensured that the models received high-quality inputs, which in turn improved their performance.
- Comprehensive evaluation framework: The implementation of k-fold cross-validation provided a robust framework for evaluating the models, ensuring that the performance metrics were reliable and representative of the models' capabilities.
- Integration of domain knowledge: Incorporating domain-specific knowledge in the feature engineering process helped in selecting the most relevant features, improving the model's ability to discern between different tool states.
- Real-time monitoring capabilities: Developing a framework for real-time data acquisition and processing enabled immediate feedback on tool conditions, which is crucial for industrial applications.
- Feature engineering: Developing novel feature engineering techniques that transform raw sensor data into more meaningful representations, enhancing the models' ability to learn from the data.
- Cross-validation techniques: Applying advanced cross-validation techniques, including a stratified k-fold approach, to ensure that each fold was representative of the overall dataset, thereby improving the robustness of the evaluation.
- Automated hyperparameter tuning: Implementing automated hyperparameter tuning methods, such as grid search and random search, ensured that the models were optimized for performance without manual intervention.
- Balanced dataset: Ensuring that the dataset was balanced across the three classes (green, yellow, red) prevented biased training and testing, which contributed to more accurate and generalizable models.

These improvements have collectively contributed to the advancement of machine learning techniques for tool condition monitoring in milling processes, demonstrating the potential for more effective and reliable predictive maintenance in industrial settings.

The key improvements achieved in this study underline the importance of meticulous feature selection, optimized hyperparameter tuning, and robust data preprocessing. These enhancements not only improved the individual performance of the SVM and boosting ensemble decision tree models but also set a foundation for future research that could explore the integration of these methodologies with more complex models like LSTM. The balanced dataset and comprehensive evaluation framework further ensured the reliability and applicability of the findings, paving the way for the development of more sophisticated, domain-specific predictive maintenance solutions.

5. Conclusions

This study assessed the effectiveness of support vector machines (SVM), long short-term memory (LSTM) networks, and boosting ensemble decision trees in tool condition monitoring during the milling processes. Each method demonstrated unique strengths and limitations when applied to the high-dimensional and sequential data typical for industrial applications. The findings illustrate that SVM provided the highest accuracy, while LSTM lagged in performance due to its sensitivity to data size and feature selection challenges.

Several implications arise for future research. The potential for hybrid models integrating the temporal learning capabilities of LSTM with the robust classification abilities of SVM and decision trees should be explored. Such models may better capture both the temporal dynamics and complex feature interactions of the tool wear data.

The development of domain-specific models tailored for industrial applications could address the requirement for extensive labeled training data for models such as LSTM. Investigating techniques for effective transfer learning and unsupervised pre-training in this domain might reduce this dependency and enable the implementation of advanced machine learning models in industrial settings.

Moreover, future work should consider expanding the dataset and including diverse machining operations to validate and generalize the current findings. Extending the

feature set through advanced signal processing techniques and exploring the integration of multimodal data sources, such as temperature and material properties, could also provide deeper insights into the condition monitoring processes.

Lastly, addressing the interpretability of machine learning models in manufacturing contexts remains a crucial research direction. Developing methods that provide clearer insights into the decision-making processes of complex models will enhance their practical applications and trustworthiness in real-world industrial solutions.

Author Contributions: Conceptualization, A.P.-M. and J.K.; data acquisition, K.S.; formal analysis, investigation, A.P.-M. and J.K.; methodology, J.K., A.P.-M., A.S., A.D., P.S. and M.K.; project administration J.K.; supervision, J.K.; visualization, A.P.-M., A.S., A.D., P.S. and J.K.; writing—original draft, I.A. and J.K.; writing—review and editing, all the authors (A.P.-M., I.A., K.S., M.K., A.S., A.D., P.S. and J.K.). All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The raw data supporting the conclusions of this article will be made available by the authors on request.

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

References

1. Hu, J.; Song, W.; Zhang, W.; Zhao, Y.; Yilmaz, A. Deep learning for use in lumber classification tasks. *Wood Sci. Technol.* **2019**, *53*, 505–517. [CrossRef]
2. Kurek, J.; Antoniuk, I.; Górski, J.; Jegorowa, A.; Świderski, B.; Kruk, M.; Wieczorek, G.; Pach, J.; Orłowski, A.; Aleksiejuk-Gawron, J. Classifiers ensemble of transfer learning for improved drill wear classification using convolutional neural network. *Mach. Graph. Vis.* **2019**, *28*, 13–23. [CrossRef]
3. Kurek, J.; Antoniuk, I.; Świderski, B.; Jegorowa, A.; Bukowski, M. Application of Siamese Networks to the Recognition of the Drill Wear State Based on Images of Drilled Holes. *Sensors* **2020**, *20*, 6978. [CrossRef]
4. Kurek, J.; Antoniuk, I.; Górski, J.; Jegorowa, A.; Świderski, B.; Kruk, M.; Wieczorek, G.; Pach, J.; Orłowski, A.; Aleksiejuk-Gawron, J. Data augmentation techniques for transfer learning improvement in drill wear classification using convolutional neural network. *Mach. Graph. Vis.* **2019**, *28*, 3–12. [CrossRef]
5. Kurek, J.; Krupa, A.; Antoniuk, I.; Akhmet, A.; Abdiomar, U.; Bukowski, M.; Szymanowski, K. Improved Drill State Recognition during Milling Process Using Artificial Intelligence. *Sensors* **2023**, *23*, 448. [CrossRef]
6. Przybyś-Małaczek, A.; Antoniuk, I.; Szymanowski, K.; Kruk, M.; Kurek, J. Application of Machine Learning Algorithms for Tool Condition Monitoring in Milling Chipboard Process. *Sensors* **2023**, *23*, 5850. [CrossRef] [PubMed]
7. Szwajka, K.; Trzepieciński, T. Effect of tool material on tool wear and delamination during machining of particleboard. *J. Wood Sci.* **2016**, *62*, 305–315. [CrossRef]
8. Wei, W.; Li, Y.; Xue, T.; Tao, S.; Mei, C.; Zhou, W.; Wang, J.; Wang, T. The research progress of machining mechanisms in milling wood-based materials. *BioResources* **2018**, *13*, 2139–2149. [CrossRef]
9. Schmidhuber, J. Deep learning in neural networks: An overview. *Neural Netw.* **2015**, *61*, 85–117. [CrossRef] [PubMed]
10. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
11. Ibrahim, I.; Khairuddin, A.S.M.; Abu Talip, M.S.; Arof, H.; Yusof, R. Tree species recognition system based on macroscopic image analysis. *Wood Sci. Technol.* **2017**, *51*, 431–444. [CrossRef]
12. Li, R.; Wei, P.; Liu, X.; Li, C.; Ni, J.; Zhao, W.; Zhao, L.; Hou, K. Cutting tool wear state recognition based on a channel-space attention mechanism. *J. Manuf. Syst.* **2023**, *69*, 135–149. [CrossRef]
13. Li, Z.; Liu, X.; Incecik, A.; Gupta, M.K.; Królczyk, G.M.; Gardoni, P. A novel ensemble deep learning model for cutting tool wear monitoring using audio sensors. *J. Manuf. Process.* **2022**, *79*, 233–249. [CrossRef]
14. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. *Commun. ACM* **2017**, *60*, 84–90. [CrossRef]
15. Lin, K.K.Y. GitHub Repository for AlexNet Model. 2020. Available online: https://gist.github.com/kevinlin311tw/a0a36e2b4d6ab9b09201 (accessed on 24 April 2023).
16. Stanford Vision Lab, Stanford University; Princeton University. ImageNet Web Page. 2020. Available online: https://image-net.org/ (accessed on 24 April 2023).
17. Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. Imagenet large scale visual recognition challenge. *Int. J. Comput. Vis.* **2015**, *115*, 211–252. [CrossRef]

18. Demir, F.; Turkoglu, M.; Aslan, M.; Sengur, A. A new pyramidal concatenated CNN approach for environmental sound classification. *Appl. Acoust.* **2020**, *170*, 107520. [CrossRef]
19. Choudhary, A.; Mishra, R.K.; Fatima, S.; Panigrahi, B. Multi-input CNN based vibro-acoustic fusion for accurate fault diagnosis of induction motor. *Eng. Appl. Artif. Intell.* **2023**, *120*, 105872. [CrossRef]
20. Tsotsopoulou, E.; Karagiannis, X.; Papadopoulos, P.; Dyśko, A.; Yazdani-Asrami, M.; Booth, C.; Tzelepis, D. Time-domain protection of superconducting cables based on artificial intelligence classifiers. *IEEE Access* **2022**, *10*, 10124–10138. [CrossRef]
21. Chan, A.D.; Green, G.C. Myoelectric control development toolbox. *CMBES Proc.* **2007**, *30*, 1–4.
22. Ye, J.; Janardan, R.; Li, Q.; Park, H. Feature extraction via generalized uncorrelated linear discriminant analysis. In Proceedings of the Twenty-First International Conference on Machine Learning, Banff, AB, Canada, 4–8 July 2004; p. 113. [CrossRef]
23. Li, D.; Cai, Z.; Qin, B.; Deng, L. Signal frequency domain analysis and sensor fault diagnosis based on artificial intelligence. *Comput. Commun.* **2020**, *160*, 71–80. [CrossRef]
24. Bergmann, S.; Moussa, D.; Brand, F.; Kaup, A.; Riess, C. Frequency-Domain Analysis of Traces for the Detection of AI-based Compression. In Proceedings of the 2023 11th International Workshop on Biometrics and Forensics (IWBF), Barcelona, Spain, 19–20 April 2023; IEEE: Piscataway, NJ, USA, 2023; pp. 1–6. [CrossRef]
25. Wang, M.; Fan, P.; Yang, T. Fake face detection based on deep learning and frequency domain processing. In Proceedings of the 2023 IEEE 5th International Conference on Civil Aviation Safety and Information Technology (ICCASIT), Dali, China, 11–13 October 2023; IEEE: Piscataway, NJ, USA, 2023; pp. 77–82. [CrossRef]
26. Lin, C.H. Frequency-domain features for ECG beat discrimination using grey relational analysis-based classifier. *Comput. Math. Appl.* **2008**, *55*, 680–690. [CrossRef]
27. Prince, A.A.; Ganesh, S.; Verma, P.K.; George, P.; Raju, D. Efficient implementation of empirical mode decomposition in FPGA Using Xilinx System Generator. In Proceedings of the IECON 2016—42nd Annual Conference of the IEEE Industrial Electronics Society, Florence, Italy, 23–26 October 2016; pp. 895–900. [CrossRef]
28. Wen, W.; Gao, R.X.; Cheng, W. Planetary Gearbox Fault Diagnosis Using Envelope Manifold Demodulation. *Shock Vib.* **2015**, *2016*, 3952325. [CrossRef]
29. Palani, P.; Sompur, V.; Thondiyath, A. Characterisation of Physiological Tremor using Multivariate Empirical Mode Decomposition and Hilbert Transform. In Proceedings of the 2023 45th Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC), Sydney, Australia, 24–27 July 2023; IEEE: Piscataway, NJ, USA, 2023; pp. 1–4. [CrossRef]
30. Hussein, H.M.; Abdalla, K.K. Seizure prediction algorithm based on simulated annealing and machine learning. *Int. J. Nonlinear Anal. Appl.* **2023**, *14*, 1499–1508. [CrossRef]
31. Zhang, C.; Wei, H.; Zhao, J.; Liu, T.; Zhu, T.; Zhang, K. Short-term wind speed forecasting using empirical mode decomposition and feature selection. *Renew. Energy* **2016**, *96*, 727–737. [CrossRef]
32. 1.4. Support Vector Machines—Scikit-Learn.org. 2024. Available online: https://scikit-learn.org/stable/modules/svm.html (accessed on 29 April 2024).
33. Aggrawal, R.; Pal, S. Sequential feature selection and machine learning algorithm-based patient's death events prediction and diagnosis in heart disease. *SN Comput. Sci.* **2020**, *1*, 344. [CrossRef]
34. 1.11. Ensembles: Gradient Boosting, Random Forests, Bagging, Voting, Stacking—Scikit-Learn.org. 2024. Available online: https://scikit-learn.org/stable/modules/ensemble.html (accessed on 29 April 2024).
35. Sherstinsky, A. Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. *Phys. D Nonlinear Phenom.* **2020**, *404*, 132306. [CrossRef]