Ingénierie des Systèmes d'Information
Vol. 28, No. 5, October, 2023, pp. 1361-1375
Journal homepage: http://iieta.org/journals/isi

Enhancing Fault Detection in CNC Machinery: A Deep Learning and Genetic Algorithm Approach

Paul Menounga Mbilong[1*], Zineb Aarab[2], Fatima-Zahra Belouadha[1], Mohammed Issam Kabbaj[1]

[1] Computer Science Department, Ecole Mohammadia d'Ingénieurs, Mohammed V University, Rabat 10090, Morocco
[2] LRIT Associated Unit to CNRST (URAC 29), Faculty of Sciences, Mohammed V University, Rabat BP 1014, Morocco

Corresponding Author Email: paulmbilong@research.emi.ac.ma

https://doi.org/10.18280/isi.280525

Received: 15 June 2023
Revised: 30 August 2023
Accepted: 9 October 2023
Available online: 31 October 2023

Keywords:
industrial maintenance, long short-term memory, convolutional neural network, residual network, variational autoencoder, genetic algorithm, predictive maintenance, fault detection

ABSTRACT

This paper introduces a deep learning-based approach (DLBA) tailored for fault detection and condition monitoring in industrial machinery. The presented DLBA architecture is assessed utilizing a dataset derived from a CNC milling machine, as part of the University of Michigan's System-level Manufacturing and Automation Research Testbed (SMART). The results underscore the substantial efficacy of the LSTM model, evidenced by a precision of 94% and an F1 score of 95%. These findings serve as a robust foundation for the identification of CNC machining failures within the manufacturing industry. The DLBA architecture constitutes a comprehensive framework for efficient fault detection, incorporating a variety of network models, including MLP, CNN, CNN auto-encoder, LSTM, and ResNet. Each model leverages its unique strengths in the analysis of complex data. Genetic Algorithm (GA) optimization is employed for parameter tuning across all models, enhancing their performance. The findings from this study contribute significantly to the development of more reliable and cost-effective predictive maintenance systems, enabling manufacturers to detect faults early on, thus preventing expensive downtime and mitigating safety risks.

1. INTRODUCTION

Predictive maintenance is of paramount importance across numerous industrial sectors, providing essential foresight into potentially costly and hazardous equipment malfunctions. In this arena, troubleshooting serves as a proactive measure to forestall technical issues and extend machinery lifespan. Early detection of faults empowers companies to take pre-emptive action, minimising operational downtime and circumventing disruptions to production processes.

The importance of timely fault detection in precluding substantial repair costs and operational interruptions in industrial settings has been emphatically highlighted in prior research. As Melgarejo and Agudelo [1] elucidated, the accurate detection and classification of bearing faults present a considerable challenge due to their nuanced manifestations and intricate identification prerequisites. The deployment of advanced methodologies and efficient signal processing techniques is, therefore, indispensable.

Majidi et al. [2] explore the recognition of patterns in partial discharge, an electrical phenomenon associated with electrical faults. Through the utilization of parsimonious representation and artificial neural networks, the accuracy of fault detection in processing complex signals is enhanced.

Similarly, Azamfar et al. [3] address the diagnosis of gearboxes operating under variable conditions. The complexity of these systems necessitates robust and adaptive measures such as transfer learning and deep convolutional networks for effective fault detection.

The role of Principal Component Analysis (PCA) in early failure detection is emphasised by Sarita et al. [4]. The development of techniques to identify anomalies at their inception using collected data is of utmost importance.

Amy et al. [5] concentrate on the reliability analysis of electronic equipment subjected to shock and vibration. The identification of environmental influences on device performance and the development of relevant reliability measures is crucial.

Moreover, the presence of unstructured data and noise within sensor recordings poses a significant challenge, potentially compromising the accuracy of fault detection models. Cumulatively, it is apparent that maintenance troubleshooting significantly impacts business performance by reducing maintenance costs, decreasing downtime, and augmenting operational safety. Its application is, therefore, vital for companies striving to maintain a competitive edge in a rapidly evolving market.

Notwithstanding the variety of challenges addressed through diverse methodologies, deep learning has demonstrated substantial advancements in prediction, particularly in diagnostics. The progress made in this arena has opened new avenues for identifying faults in industrial equipment. A plethora of recent research articles exhibit notable results, underscoring the effectiveness of deep learning in diagnostic applications.

Yet, amid these promising developments, the challenge of selecting the most suitable deep learning model persists. Given the myriad of articles leveraging deep learning for diagnostics, the ability to discern the optimal model for a given task is paramount. Each piece of research often claims to present the superior model, compounding the complexity of model selection.

In view of the plethora of models and the importance of informed decision-making, an in-depth comparison of existing approaches becomes essential. This study aims to scrutinize five of the most frequently employed deep learning network models from the past decade to uncover their strengths, weaknesses, and to provide recommendations for model selection.

This investigation encompasses various deep network models, including the Multilayer Perceptron (MLP), Convolutional Neural Network (CNN), CNN autoencoder, Long Short-Term Memory (LSTM), and Residual Network (ResNet). Each model boasts unique features beneficial for analysing complex data and detecting fault signals.

A dataset derived from a CNC milling machine, part of the System-level Manufacturing and Automation Research Testbed (SMART) at the University of Michigan, is used for model evaluation. A genetic algorithm (GA) is employed for parameter optimisation to enhance model performance. The primary objective is to present a clear and comprehensive deep learning-based architecture for efficient fault detection in industrial equipment.

The results underscore the remarkable effectiveness of the LSTM model, exhibiting superior precision, recall, and F1 score. These findings provide a robust foundation for identifying machining failures in manufacturing, promoting the adoption of deep learning in predictive maintenance systems.

By combining the advancements in deep learning diagnostics with the necessity of model selection, this research aims to provide valuable insights for industry practitioners and researchers dedicated to improving fault detection in industrial equipment.

By identifying the top-performing models and assessing their efficacy, this study contributes to the evolution of deep learning-driven diagnostic techniques. This research addresses the lack of guidance on model selection and configuration specific to CNC machines, fostering the adoption of more efficient and reliable methods for fault detection.

The subsequent sections delve into the various deep network models employed, elaborate on the evaluation and optimisation processes, and discuss the tangible implications of the approach. The potential for enhancing equipment reliability and reducing operational costs across the manufacturing landscape is underscored.

2. RELATED WORK

The deployment of deep learning models for fault diagnosis is garnering increased attention within industrial research circles. Numerous scholars have delved into a variety of deep learning-based approaches to address this complex issue. The ensuing works offer insightful perspectives on recent progressions in this field.

A comprehensive review by Saufi et al. [6] grapples with the challenges and prospects of employing deep learning models for machine fault detection and diagnosis. The authors expound on the merits and constraints of diverse deep learning models and spotlight their applications across various industrial scenarios.

In an innovative response to the critical challenge of pipeline leakages, Obaid et al. [7] have crafted a deep learning approach that employs image-based edge detection techniques for oil spill identification. Utilizing aerial imagery from drones, the study showcases the efficiency of the DexiNed algorithm at a pivotal 10-meter height, setting a new standard for environmental monitoring in the oil industry.

Taking a more focused approach, Yang et al. [8] concentrate on diagnosing faults in electric motors via deep learning algorithms. Four traditional types of deep learning models are explored, with the authors accentuating their use in electric motor fault detection.

Neupane and Seok [9] scrutinize the application of deep learning methodologies in analysing the Case Western Reserve University (CWRU) bearing fault dataset. The authors elucidate the challenges associated with leveraging deep learning for bearing fault diagnosis and propose a method for selecting the most appropriate learning model for a specific dataset.

Further extending the application of deep learning to fault diagnosis, Jian et al. [10] propose the use of a specialized deep learning algorithm employing stacked autoencoders (SAEs) for diagnosing faults in coal mills. The potential of this approach in this particular domain is demonstrated.

In the context of Industry 4.0's transformative impact, Mohammed et al. [11] propose an IoT and machine learning-based predictive maintenance system tailored for electrical motors. By integrating cutting-edge ML models and MQTT messaging within the Industrial IoT framework, their system skillfully anticipates equipment failures, significantly streamlining maintenance processes.

Glaeser et al. [12] delve into the application of deep learning for fault detection in industrial cold forging processes. Utilizing DLBA for fault detection, the authors employ a convolutional neural network classifier to identify defect conditions, subsequently utilizing a decision tree model for defect classification.

A particular case study conducted by Rao et al. [13] put forth a machine learning approach for diagnosing faults in industrial equipment. In this research, the authors utilized support vector machines (SVMs) and artificial neural networks (ANNs) to classify varying types of faults.

Addressing critical needs in neonatal care, Mahdi et al. [14] introduce a multi-faceted fault detection and monitoring system for infant incubators, powered by machine learning classifiers and the compact Raspberry Pi 4. Their system's adeptness in utilizing Decision Tree, Support Vector Machine, and Neural Network algorithms not only elevates care standards but also ensures the vital comfort of preterm newborns through advanced sensor technology.

A synthesis of the key insights derived from these studies is as follows:

- Deep learning models can be effectively deployed for the detection and diagnosis of faults in industrial machinery.
- Diverse deep learning models possess distinct strengths and weaknesses. The optimal model for a specific application is contingent on the unique characteristics of the data.
- Challenges persist in the deployment of deep learning for fault diagnosis, including the requirement for substantial data quantities and the complexity of interpreting the output of deep learning models.

Despite the significant progress demonstrated by the aforementioned studies in the field of fault detection, they do not necessarily offer definitive guidance for researchers navigating a particular data set. Consider, for instance, data derived from numerically controlled (CNC) machines. The objective in this scenario is to halt or schedule maintenance programs when specific faults are detected in CNC-controlled equipment. The questions that researchers might pose in this context could include:

• Which models are most suitable for this task?

• How many layers should be incorporated into these models?

• What is the ideal number of neurons, units, or filters required per layer?

• Which models are best equipped to handle multiple outputs?

Table 1. Papers on fault detection on CNC machines

Papers	Model Used	Accuracy (%)	Outputs
[15]	SVM, XGBoost, RF	62, 99, 99	1
[16]	DT, ANN, KNN, SVM, LR, MNB	99, 94, 90, 87, 60, 57	1
[17]	LSTM-SVM	99	1
[18]	Autoencoder	100	1
[19]	CNN	95	1

A critical examination of the literature reveals a dearth of useful guidance in addressing the posed research questions, as the studies cited previously tend to be overly specific or excessively broad. What is indeed required is a reproducible framework and comparative studies that specifically deal with the classification of machining faults in CNC machines. It is within this context that the current study is positioned.

Table 1 presents a compilation of studies that have sought to address the problem of identifying machining faults in CNC machines.

Park et al. [15] applied a variety of machine learning algorithms to classify machining faults in CNC machines, with accuracies of 62%, 99%, and 99% being accomplished respectively. However, a reproducible framework for comparing different deep learning models was not provided in their study.

Similarly, Shurrab et al. [16] also employed machine learning algorithms for fault classification in CNC machines, achieving accuracies of 99%, 94%, 90%, 87%, 60%, and 57%, respectively. However, their study did not specifically concentrate on deep learning models.

On the other hand, Polat [17] utilized a combination of an LSTM network and an SVM to classify data from an EMCO Concept Mill CNC, obtaining exceptional results with an accuracy of 99%.

Zhang et al. [18] proposed an autoencoder model to tackle the problem of permanent magnet synchronous machine (PMSM) fault detection in CNC machines, achieving an accuracy of 100%. However, their study did not focus on real-time fault detection.

Lastly, Chung et al. [19] proposed a real-time fault detection solution for CNC machines using a convolutional neural network with binary weights, achieving an accuracy of 95.07%. However, their work did not include a comparative study of different deep learning models.

The present study casts its focus on fault diagnosis in CNC machines [15-19], an area that is attracting burgeoning interest in the sphere of industrial research. Despite significant strides in the realm of fault detection, conspicuous gaps remain evident. The deficiencies that this paper intends to address can be summarized as follows:

• The absence of specific guidance pertaining to model selection and configuration in the context of CNC machine fault diagnosis.

• The lack of concrete methodologies for handling multi-output classification issues in fault diagnosis for CNC machines.

• The dearth of comprehensive, reproducible frameworks for tackling the challenges associated with the classification of machining faults in CNC machines.

The current paper endeavors to propose a reproducible deep learning approach for fault diagnosis in CNC machines that sufficiently addresses these identified gaps. A comparison of five state-of-the-art deep learning models is conducted, and practical recommendations for model configuration are provided. Furthermore, a novel method for resolving multi-output classification problems in fault diagnosis for CNC machines is proposed. The ensuing section of this paper will detail the methodology employed.

3. METHODOLOGY

The Before presenting the overall architecture, it is important to outline its main components. The overall architecture we have used is shown in Figure 1.

After first partitioning our dataset into a training set and a test set, we feed this data into a model to train it. The model can be of different types such as MLP, CNN, Autoencoder CNN, LSTM and ResNet. The model is then optimized using a genetic algorithm. Optimizable parameters include the number of units, filter size, kernel size, choice of activation function and number of layers. Once the optimization is complete, a prediction vector is generated for each training input. This vector contains three elements: tool condition, machine completion and successful visual inspection. The following sections provide a more detailed description of each of these elements.

3.1 Dataset used

The dataset used in our article is called the "CNC Milling Dataset - University of Michigan SMART Lab". The data was collected from machining experiments conducted on wax blocks using a CNC milling machine at the University of Michigan SMART Lab. This dataset offers opportunities for studying areas like tool wear detection and inadequate clamping detection.

The dataset comprises multiple files, with "train.csv" being the primary one. This file contains general data from 18 different experiments (Table 2).

In addition to the "train.csv" file, the dataset also includes files named "experiment_01.csv" to "experiment_18.csv". Each file contains time series data collected from the 18 experiments, with a sampling rate of 100ms. Time series measurements are available for the CNC motors (X, Y, Z axes, and spindle), along with other variables related to the CNC machine (Table 2, Table 3). The dataset contains 25,286 entries, as depicted in Figure 2. Specifically, for the output "tool condition" we have 63.1% data in class 1 and 36.9% in class 0. Regarding the output "machining finalized" 74.4% of the data belongs to class 1, while 25.6% is in class 0. Lastly, for the output "passed visual inspection" we have 52.4% data in class 0 and 47.6% in class 1.

These characteristics are repeated for the Y and Z axes as well as the spindle (S). Additionally, the characteristics of the execution program (refer to Table 4) can be included, resulting in a total of 48 characteristics.

These features provide detailed information on machining operations and CNC parameters, making the dataset a valuable resource for research and analysis of machining processes.

In our study, we used a train-test split approach to partition the dataset into training and testing subsets. Specifically, we divided the dataset into training data (X_train, y_train) and testing data (X_test, y_test) using a ratio of 80:20. We ensured that the data shuffling and stratification were applied during this process to maintain a balanced distribution of classes. While we did not explicitly employ any specific regularization techniques, such as dropout or L1, L2 regularization, our emphasis was on optimizing hyperparameters using a genetic algorithm to mitigate overfitting. The genetic algorithm's iterative optimization process, combined with the appropriate choice of hyperparameters, serves as a means of promoting model generalization and preventing overfitting.

Having laid the groundwork with the data used, we now move on to the essential stage of data preprocessing to guarantee the quality and readiness of our dataset for further analysis.

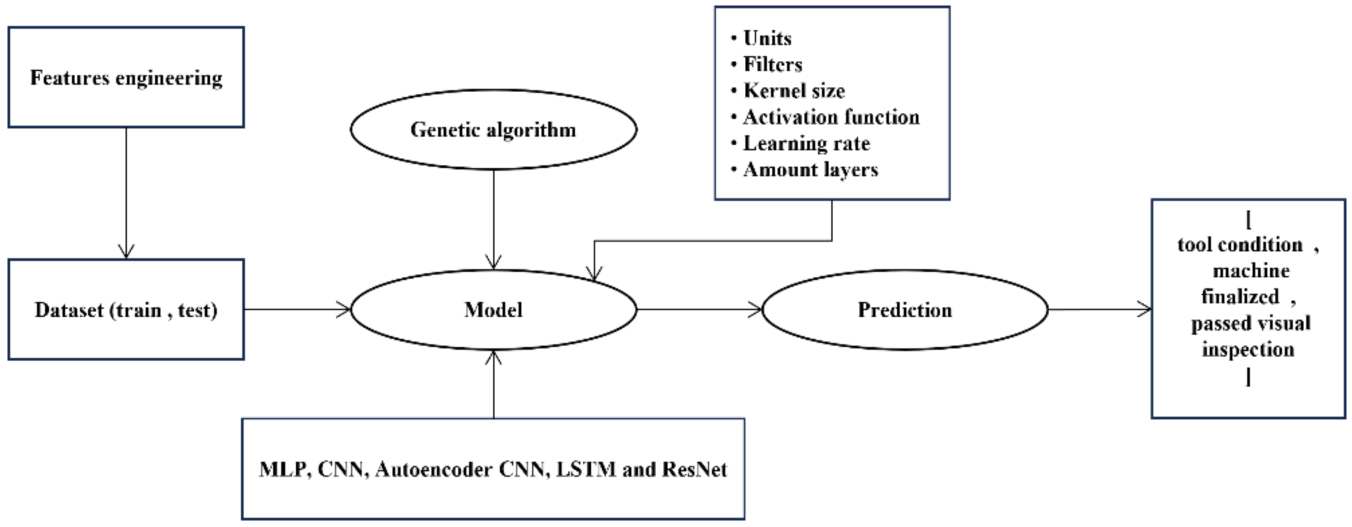

Figure 1. Framework proposed

Table 2. The train.csv file

Features	Description
No	Experiment number
material	Material (wax)
feed_rate	Relative feed speed of the cutting tool along the workpiece (mm/s)
clamp_pressure	Pressure used to hold the workpiece in the vice (bar)
tool_condition	Tool condition (unworn or worn)
machining_finalized	Indicator if machining has been completed without moving the part
passed_visual_inspection	Indicator of whether the part has passed visual inspection

Table 3. Characteristics of sub-data sets experiment_01.csv to experiment_18.csv, X axis

Features	Description
X1_ActualPosition	Actual position of workpiece X axis (mm)
X1_ActualVelocity	Actual workpiece X-axis speed (mm/s)
X1_ActualAcceleration	Actual X-axis acceleration (mm/s²)
X1_CommandPosition	X axis reference position (mm)
X1_CommandVelocity	X axis reference speed (mm/s)
X1_CommandAcceleration	X-axis reference acceleration (mm/s²)
X1_CurrentFeedback	Current (A)
X1_DCBusVoltage	Voltage (V)
X1_OutputCurrent	Output current (A)
X1_OutputVoltage	Output voltage (V)
X1_OutputPower	Output power (kW)

Table 4. Characteristics of the CNC execution program

Features	Description
M1_CURRENT_PROGRAM_NUMBER	Program number under which it is listed on the CNC
M1_sequence_number	Line of G-code being executed
M1_CURRENT_FEEDRATE	Instantaneous spindle feed speed
Machining_Process	Machining step in progress

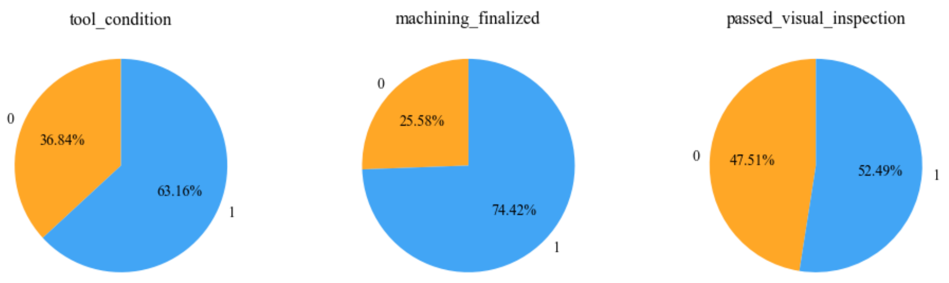

Figure 2. Distribution of outputs criteria

3.2 Data pre-processing

To ensure the robustness and effectiveness of our models, a crucial preprocessing step involved the utilization of the standard scaler technique on our input data. This approach normalized the data by transforming each feature to have a mean of zero and a standard deviation of one. By employing standard scaler, we not only mitigated the impact of varying scales among different features but also facilitated the convergence and performance of our models during training. This preprocessing step played a pivotal role in ensuring that each feature's contribution to the model's performance was balanced, and thus, led to more reliable and accurate outcomes. The standardized data was then effectively utilized for training and evaluating our deep learning models, contributing to the overall success of our fault diagnosis approach for CNC machines. The formula for standard scaler transformation is given by Eq. (1):

$$z = \frac{x-\mu}{\sigma} \tag{1}$$

where:

- z represents the standardized value of the feature.
- x is the original value of the feature.
- μ is the mean of the feature.
- σ is the standard deviation of the feature.

This transformation ensures that the feature's distribution has a mean of zero and a standard deviation of one, making it suitable for various machine learning algorithms, particularly those that are sensitive to the scale of features. Having prepared and standardized the data using the standard scaler transformation, we can now delve into the optimization process with GA that lies at the core of our methodology.

The application of GA provides us with a systematic approach to fine-tuning the hyperparameters of our deep learning models. By employing the principles of natural selection and evolution, GAs enables us to navigate the vast search space of possible configurations efficiently. In the subsequent sections, we will delve into the details of our approach, starting with an exploration of the Root Mean Square Error (RMSE), a pivotal performance metric. Understanding RMSE and its role in evaluating model accuracy is essential before we proceed to outline how we leverage genetic algorithms to discover optimal hyperparameters for our multi-layer perceptron (MLP), convolutional neural network (CNN), and variational autoencoder with convolutional neural network (VAE CNN) models. This dual focus on RMSE and genetic algorithms empowers us to enhance the models' performance on the CNC Milling Dataset and extract meaningful insights from the data.

3.3 Root Mean Square Error (RMSE)

The Root Mean Square Error (RMSE) is a widely used performance metric in regression analysis and machine learning to quantify the accuracy of a predictive model's predictions. It measures the average squared differences between the predicted values and the actual values. RMSE provides an intuitive understanding of how well the model's predictions align with the true values, considering both the magnitude and direction of the errors. The formula for calculating RMSE is as follows Eq. (2):

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2}{n}} \tag{2}$$

where:

- n is the number of data points.
- y_i represents the actual (true) value of the i-th data point.
- $\hat{y}_i$ represents the predicted value of the i-th data point.

In our study, we employ RMSE as the fitness evaluation metric within the Genetic Algorithm (GA) optimization process. The utilization of RMSE ensures that we prioritize model architectures that exhibit accurate predictions on the test dataset. The lower the RMSE value, the closer the predicted values are to the actual values, indicating better model performance. With a comprehensive understanding of RMSE and its significance, we proceed to describe the GA methodology that we have implemented to optimize our model architectures and hyperparameters, aiming to enhance predictive accuracy further.

3.4 Genetic Algorithm (GA)

In this paper, we have implemented a specific algorithm for optimizing the parameters of our models, as presented in Table 5. In our methodology, we employed a genetic algorithm to optimize the architecture of each model, considering its specific type.

The genetic algorithm employs an iterative approach to identify the optimal hyperparameter values [20-22]. The hyperparameters considered are as follows:

- Number of filters per CONV1D layer (for models with convolutional layers).
- Kernel size for CONV1D layers.
- Number of units per LSTM layer.
- Activation functions: we considered the Relu, sigmoid, and tanh activation functions.

- Optimizer: We used the Adam, Sgd, and RMSprop optimizers.
- Learning rate: We defined a set of possible learning rates, including 0.001, 0.002, 0.005, 0.01, and 0.1.

Table 5. Pseudo-code genetic algorithm used

```
Algorithm 1: Genetic_Algorithm
     Output: Best_Individual
     Begin:
1     Generate_Initial_Population ()
2     Set best_individuall_overall = None
3     For each generation G from 1 to N:
4       Begin
5       Evaluate_Fitness(Population[G],
        best_individuall_overal)
6       Select_Parents(Population[G])
7       New_Population = []
8       While size of New_Population < Population_Size:
9         Begin
10          Parent1 = Select_Parent()
11          Parent2 = Select_Parent()
12          Child1, Child2 = Crossover (Parent1, Parent2)
13          Mutate (Child1)
14          Mutate (Child2)
15          Add Child1 to New_Population
16          Add Child2 to New_Population
17        End
18      Population[G+1] = New_Population
19    Best_Individual= Select_Best_Individual(Population[N],
     best_individuall_overall)
10   Return Best_Individual
     End
```

The rationale behind the chosen hyperparameter ranges in our study is a result of a combination of both literature insights and experimental tuning. We drew from existing research and best practices in the field of deep learning to establish initial ranges that are commonly considered effective for each hyperparameter. These ranges were aligned with recommendations from literature and experienced practitioners. However, to tailor these ranges specifically to our problem and dataset, we performed extensive experimental tuning. Our iterative process involved testing various combinations of hyperparameters within different ranges to understand their effects on the model's performance. This empirical approach allowed us to identify ranges that led to optimal results on the CNC Milling Dataset. By considering both established guidelines and empirical results, we aimed to strike a balance between relying on existing knowledge and adapting to the unique characteristics of our problem domain.

The genetic algorithm evaluated each combination of hyperparameters using a predefined evaluation function called Fitness, that provides a measure of error on the test data. Fitness evaluation is an important step in the genetic algorithm. It allows us to evaluate the quality of the individuals and select those that will be used for reproduction and mutation. In our case, we used the RMSE on the test set as the fitness to evaluate the individuals.

By using the RMSE on the test set as the fitness, we were able to select the individuals that achieved the best results on the test data. These individuals were then used to create the final model.

We also used other performance metrics to evaluate the individuals, such as accuracy and precision. However, we found that the RMSE was the most effective metric for selecting the best-performing individuals.

Additionally, we incorporated a mutation function to facilitate a broader exploration of the hyperparameter space. The mutation function randomly selected a target attribute from the defined hyperparameters and introduced a mutation by randomly selecting a new value within the corresponding value range.

The choice of specific population size and generations in our genetic algorithm was made to strike a balance between computational efficiency and thorough exploration of the hyperparameter space. A population size of 50 and 50 generations were selected based on empirical experimentation to ensure an adequate diversity of solutions while keeping the computational complexity manageable. This approach allowed us to explore a range of potential solutions and evolutionary paths within a reasonable computational timeframe. The parameters were fine-tuned through iterative testing, taking into consideration the trade-off between exploration and exploitation of the search space. Our objective was to ensure that the genetic algorithm has sufficient iterations to converge towards an optimal solution while avoiding excessive computational demands.

Finally, we selected the top individuals from the final population to compose our ultimate model. We employed an external variable to store the best individual overall for each population. Finally, we compared this overall best individual with the best individual from the last population to accurately identify the ultimate best individual. This selected model was trained using the training data and evaluated on the test set. We recorded the highest accuracy achieved, as well as the model's architecture represented by these best individuals.

In summary, our methodology utilized a genetic algorithm.

to optimize the overall architecture of each network by considering hyperparameters such as the number of filters, activation functions, optimizers, and learning rates. This approach enabled us to determine an optimal architecture for our model, resulting in commendable performance on the test data.

3.5 Multi-Layers Perceptron (MLP)

The Multilayer Perceptron (MLP) is a widely utilized artificial neural network in deep learning [23-25]. It comprises multiple layers of neurons, including an input layer, one or more hidden layers, and an output layer. Each neuron in the hidden and output layers is connected to all neurons in the preceding and succeeding layers, forming a dense network structure. The calculation for the activation of a neuron in a hidden layer is defined by Eq. (3).

$$a_j^{(l)} = \sigma\left(\sum_{i=1}^{n^{(l-1)}} w_{ij}^{(l)} a_i^{(l-1)} + b_j^{(l)}\right) \tag{3}$$

where, $a_j^{(l)}$ is the activation of the neuron j in the layer $l, w_{ij}^{(l)}$ is the weight between the neuron i in the layer l-1 and the neuron j in the layer $l, a_i^{(l-1)}$ is the activation of the neuron i in the layer $l-1, b_j^{(l)}$ is the bias of the neuron j in the layer l and σ is the activation function.

The calculation of the loss function is given by Eq. (4).

$$J(\mathbf{w},\mathbf{b}) = \frac{1}{m}\sum_{i=1}^{m} L(y_i, \hat{y}_i) \tag{4}$$

where, J is the loss function, w and b are the MLP weights and biases, m is the number of training examples, y_i is the true label of the example $i, \hat{y}_i$ is the MLP prediction for the example i is L is the specific loss function.

The table summarizes the hyperparameter bounds for the MLP model:

- Hidden layer sizes define the number of neurons per hidden layer in the model. The permitted values are: (50),(100),(50,50),(100,100),(50,100) and (50,100,200).
- Activation: specifies the activation function for the model layers. The permitted values are Relu and sigmoid.
- optimizer: indicates the optimizer to be used to drive the model. The permitted values are Adam and Sgd.
- Learning rate sets the learning rate for the optimizer. The permitted values are 0.001,0.01 and 0.1.

The MLP we have implemented consists of 1, 2 or three layers depending on the feedback from our genetic algorithm. Its configuration in terms of neurons per layer and activation functions is given in Table 6.

The MLP model is a versatile and powerful tool for machine learning. It can be used for a variety of tasks, including classification, regression, and clustering. The MLP model we have implemented is configurable with a variety of hyperparameters, which allows us to optimize the model for a specific task.

Table 6. AG configuration for MLP

Hyperparameters	Values
hidden_layer_sizes	[(50), (100), (50, 50), (100, 100), (50, 100), (50, 100, 200)]
activation	Relu, sigmoid, tanh
optimizer	Adam, Sgd
learning_rate	[0.001, 0.01, 0.1]

3.6 Convolutional neural networks (CNN)

The CNN model is designed to automatically extract relevant features from input data [26]. In our case, the input data is 48-column data resized in sequence of 1. These variables include measurements of CNC motor position, speed, acceleration, current and voltage.

Our CNN model consists of three Conv1D layers that perform convolution operations on the input sequences. Each Conv1D layer is followed by an activation function that introduces non-linearity into the model. This allows the model to capture complex, non-linear patterns in the input data.

After the convolution layers, the extracted features are flattened and fed into a dense layer with three outputs corresponding to the classification classes. The dense layer uses a sigmoid activation function that assigns probabilities to each class. The essential equations for CNN networks are summarized in equations Eqs. (5)-(7).

For each convolution layer:

$$\begin{aligned} Z^{[l]} &= \text{Conv1D}\left(A^{[l-1]}, W^{[l]}, b^{[l]}\right) \\ A^{[l]} &= \text{ReLU}\left(Z^{[l]}\right) \end{aligned} \tag{5}$$

For the last convolution layer:

$$\begin{aligned} Z^{[L-1]} &= \text{Conv1D}\left(A^{[L-2]}, W^{[L-1]}, b^{[L-1]}\right) \\ A^{[L-1]} &= \text{ReLU}\left(Z^{[L-1]}\right) \end{aligned} \tag{6}$$

For the Fully Connected Layer:

$$\begin{aligned} Z^{[L]} &= A^{[L-1]} \cdot W^{[L]} + b^{[L]} \\ A^{[L]} &= \text{Sigmoid}\left(Z^{[L]}\right) \end{aligned} \tag{7}$$

where:

- l is the layer index (from 1 to L-1 for convolution layers, and L for the dense layer).
- $A^{[l-1]}$ is the input activation of the l-1.
- $W^{[l]}$ is the matrix of layer weights l.
- $b^{[l]}$ is the bias vector of the layer l.
- $Z^{[l]}$ is the weighted activation of the l.
- Conv1D represents the convolution operation in one dimension.
- ReLU is the ReLU (Rectified Linear Unit) activation function.
- Sigmoid is the activation function used for independent multi-output classification.

Table 7. AG configuration for the CNN

Hyperparameters	Values
filter	[16, 20, 64, 128, 200, 250, 300]
activation	Relu, sigmoid, tanh
optimizer	Adam, Sgd
kernel_size	[2, 3, 4, 6, 8, 10]
learning_rate	[0.001, 0.002, 0.005]

Table 7 summarizes the hyperparameter bounds for the CNN model:

- Filter: specifies the number of filters for each convolution layer. The acceptable values are 16, 20, 64, 128, 200, 250 and 300.
- Activation: determines the activation function to be used. Permitted values are Relu, sigmoid and tanh.
- Optimizer indicates the optimizer to be used for training the model. The permitted values are Adam, Sgd and RMSprop.
- Kernel size: determines the size of the kernel for each convolution layer. The permitted values are 2, 3, 4, 6, 8 and 10.
- Learning rate sets the learning rate for the optimizer. The permitted values are 0.001, 0.002 and 0.005.

In this article, the CNN model we have implemented is made up of three CONV1D layers depending on the feedback from our genetic algorithm. Its configuration in terms of filters per layer and activation functions is given in Table 7.

The Convolutional Neural Network (CNN) stands as a robust asset in the realm of machine learning tasks that encompass the handling of sequential data or images. Its applicability spans a wide range of objectives, such as classification, regression, and the identification of anomalies. The CNN model we've constructed comes with a flexible array of hyperparameters, affording us the opportunity to tailor the model's configuration precisely to a given task.

3.7 Variational Autoencoder with Convolutional Neural Network (VAE CNN)

The method used in this section for modelling and analyzing the data is the Variational Autoencoder with Convolutional Neural Network (VAE CNN). The VAE CNN is a powerful model that combines the advantages of variational autoencoders and convolutional neural networks to learn a latent representation of the data and generate accurate reconstructions [27, 28].

The architecture of the CNN VAE consists of two main parts: the encoder and the decoder. The encoder takes the input data and transforms it into a reduced-dimension latent representation. The decoder, on the other hand, takes the latent representation as input and attempts to reconstruct the original data. This approach makes it possible to learn a meaningful representation of the data while retaining the essential information.

In our implementation, we use convolution layers (Conv1D) to encode and decode the data. Convolution layers are used to extract important features from the data by considering the spatial relationships between different points. Using convolution filters and pooling operations, the model can capture complex patterns and structures in the data.

The CNN VAE encoder comprises several convolution layers that progressively reduce the dimensionality of the data. Next, the means and logarithms of the latent space variances are calculated from the outputs of the convolution layers. Using a reparameterization function, samples are generated in the latent space, allowing exploration and interpolation of different representations of the data.

The CNN VAE decoder takes input latent space samples and reconstructs them using Conv1DTranspose layers. These transposed layers allow the dimensionality of the data to be increased until the initial input shape is reached.

During CNN VAE training, the loss function is defined as the sum of two terms: the reconstruction loss and the KL divergence. The reconstruction loss measures the difference between the input data and the reconstructions generated by the model. The KL divergence is used to regularize the distribution of the latent space and promote a structured latent representation.

The VAE CNN model is optimized using a specified AG optimizer, and hyperparameters such as the number of filters, kernel size, latent dimension, and learning rate are tuned to obtain the best performance on the data.

Once the latent representation has been learned, we use a dense 5-layer MLP model for classification.

Using this VAE CNN approach, we can learn a meaningful latent representation of the CNC Milling Dataset, capturing the complex patterns and structures inherent in the data. In addition, the model can generate accurate reconstructions, allowing the quality of the learned representation to be validated.

In this article, the configuration of the VAE CNN model is given in Table 8.

Table 8. GA configuration for CNN VAE

Hyperparameters	Values
filter	[16 ,20 ,64 ,128 ,200, 250 ,300]
activation	[Relu, sigmoid, tanh]
optimizer	[Adam, Sgd, RMSprop]
kernel_size	[2 ,3 ,4 ,6,8 ,10]
learning_rate	[0.001, 0.002, 0.005]
latent_dim	[2 ,3 ,4 ,6,8 ,10, 12]

- Filter: This is the number of filters per convolution layer in the VAE model encoder. In this configuration, the GA will randomly extract an array of size 2 corresponding to our CNN layers.
- Activation: This is the activation function used in the convolution and dense layers of the model. In this case, activation is defined as Relu, which corresponds to the ReLU (Rectified Linear Unit) activation function.
- Optimizer: This is the optimizer used to update the model weights during training. Three options are provided: Adam, Sgd (Stochastic Gradient Descent) and RMSprop (Root Mean Square Propagation). In this configuration, the Adam optimizer is used.
- Kernel size: This is the size of the kernel (or filter) used in convolution layers.
- Latent dim: This is the dimension of the latent space, i.e., the reduced data representation space.
- Learning rate: This is the learning rate of the optimizer. It determines the size of the model weight update steps during training.

These parameters (Table 8) are used to define the architecture of the VAE model and to perform optimization and evaluation on the input data. Each parameter controls a specific aspect of the model, such as the number of filters, the activation function, the optimizer, the kernel size, the latent space dimension and the learning rate, allowing different parameters to be explored during the genetic optimization process.

In summary, the Variational Autoencoder with Convolutional Neural Network (VAE CNN) merges the strengths of variational autoencoders and convolutional neural networks, providing a robust means to learn a condensed representation of data and generate precise reconstructions. By employing convolution layers to extract intricate patterns and spatial relationships, the encoder transforms input data into a meaningful latent representation. The decoder then reconstructs the original data from this latent space. With the integration of convolutional filters and pooling operations, the model adeptly captures complex structures within the data. Regularized through the combination of reconstruction loss and KL divergence, the VAE CNN achieves a balanced latent distribution. The optimized VAE CNN, defined by hyperparameters specified in Table 8, demonstrates its effectiveness in capturing intricate patterns within the CNC Milling Dataset while enabling reliable classification through its learned representation.

3.8 Long Short-Term Memory (LSTM)

The LSTM model is a type of recurrent neural network that can capture long-term dependencies in data sequences [29, 30]. LSTM can be described by Eqs. (8)-(13):

- Forget gate equation:

$$f_t = \sigma\left(W_f \cdot \left[h_{t-1}, x_t\right] + b_f\right) \tag{8}$$

- Input gate equation:

$$i_t = \sigma\left(W_i \cdot \left[h_{t-1}, x_t\right] + b_i\right) \tag{9}$$

- Output gate equation:

$$o_t = \sigma\left(W_o \cdot \left[h_{t-1}, x_t\right] + b_o\right) \tag{10}$$

- Equation for the candidate cell gate:

$$\tilde{C}_t = \tanh\left(W_c \cdot \left[h_{t-1}, x_t\right] + b_c\right) \tag{11}$$

- Cell state equation :

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \quad (12)$$

- LSTM output equation :

$$h_t = o_t \cdot \tanh(C_t) \quad (13)$$

In these equations h_t is the output at time t, x_t is the input at time t, W_f, W_i, W_o, W_c are the weight matrices, b_f, b_i, b_o, b_c are the bias vectors, σ is the sigmoid function and tanh is the hyperbolic tangent function.

In this article, the configuration of the LSTM model is given in Table 9.

Table 9. AG configuration for LSTM

Hyperparameters	Values
units	[2 to 500]
activation	[Relu, sigmoid, tanh]
optimizer	[Adam, Sgd, RMSprop]
learning_rate	[0.001, 0.002, 0.005]

We have different options for each parameter. For example, for the units parameter, the GA will try several configurations from among the values 2, 16, 20, 64, 128, 200, 250 and 500. The same applies to the other parameters.

By considering our data as sequences, LSTM can play an important role. The Long Short-Term Memory (LSTM) model emerges as a powerful recurrent neural network capable of capturing intricate temporal dependencies. Utilizing a series of equations as described Eq. (8) through Eq. (13) the LSTM framework introduces the forget, input, output, and candidate cell gates to manage information flow over sequential time steps. The architecture's adaptability is reflected in Table 9, where hyperparameters such as units, activation functions, optimizers, and learning rates are meticulously tuned. The LSTM model, by its nature, is suited to handling data sequences, making it an ideal candidate for applications demanding the recognition of long-term relationships and patterns.

3.9 Residual Network (ResNet)

The ResNet model is a deep neural network architecture that introduces residual connections to facilitate learning and optimization. These residual connections allow the model to better capture relevant features by avoiding gradient vanishing problems and facilitating the flow of information across layers [31, 32].

To optimize the number of Conv1D filters and layers in each ResNet block, we used a genetic algorithm. This genetic algorithm searches the space of hyperparameters by evaluating different combinations of Conv1D filters and layers, then selecting the best performing combinations.

The genetic algorithm evaluates each combination using a performance metric, such as accuracy or precision, and selects the combinations that produce the best results. It then performs mutation and crossover operations to generate new combinations, while preserving the characteristics of the best-performing combinations.

This iterative process of selection, mutation and crossover is repeated over several generations until the best performing combinations converge on an optimal solution. Ultimately, the genetic algorithm allows us to find the optimal configuration of Conv1D filters and layers for each block of the ResNet model.

Using this approach, we can build a ResNet model of 4 CNN blocks with filter and Conv1D layer configurations specifically adapted to our problem. This allows us to make the most of the characteristics of the data and improve the performance of our model.

In summary, our methodology involves using a ResNet model of 4 CNN blocks with residual connections and optimizing the Conv1D filter and layer configurations using a genetic algorithm. This approach enables us to obtain a model adapted to our specific problem and to improve the performance of our classification system. The configuration of the ResNet model is given in Table 10.

Table 10. AG configuration for ResNet

Hyperparameters	Values
filter	[16, 20, 64, 128, 200, 250 ,300]
activation	[Relu, sigmoid, tanh]
optimizer	[Adam, Sgd, RMSprop]
kernel_size	2
learning_rate	[0.001, 0.01, 0.1]

4. RESULTS AND ANALYSIS

In this section, we present the results obtained by the different deep network models that we have studied for the detection of faults in industrial equipment. We compared the performance of the Multilayer Perceptron (MLP), the Convolutional Neural Network (CNN), the CNN auto-encoder, the Long Short-Term Memory (LSTM) and the Residual Network (ResNet) on a dataset from a CNC milling machine in the System-level Manufacturing and Automation Research Testbed (SMART) at the University of Michigan.

First, we applied a special parameter optimization process to all these models using a special genetic algorithm (GA). This allowed us to find the best combinations of parameters for each model, thereby maximizing their performance. The optimized parameters included the number of units in the hidden layers, the activation functions, the optimizers chosen and the learning rates. This approach enabled us to obtain models specifically adapted to our problem of detecting faults in industrial equipment.

To assess the performance of our models, it is important to present several metrics:

- Precision: This metric measures the accuracy of positive predictions made by the model. It is calculated as the number of true positive predictions divided by the total number of predictions made for the positive class. A high precision indicates that the model is good at avoiding false positives.
- Recall: This metric measures the ability of the model to correctly identify positive instances from the actual positive cases in the dataset. It is calculated as the number of true positive predictions divided by the total number of actual positive cases. A high recall indicates that the model is good at avoiding false negatives.
- F1-score: This metric provides a balanced view of both precision and recall. It is calculated as the harmonic mean of precision and recall. A high F1-score indicates that the model is good at both avoiding false positives and false negatives.

- Support: This metric represents the number of actual occurrences of each class in the test dataset. It is important to consider support when interpreting the evaluation metrics, as it provides context for understanding the scale of each class's presence.

By collectively considering precision, recall, F1-score, and support, we gain a comprehensive understanding of our models' strengths and limitations in classifying the different fault categories in the CNC Milling Dataset. These metrics allow us to make informed decisions about model selection and fine-tuning, ultimately contributing to the overall success of our fault diagnosis framework.

4.1 MLP model

In the MLP section of our study, we evaluated the performance of the MLP (Multilayer Perceptron) model for fault detection on three different criteria: tool condition, machine completion and visual inspection. The results obtained are presented in Tables 11-13.

Table 11. Performance MLP on output tool condition

	Precision	Recall	F1-Score	Support
0	0.81	0.70	0.75	2106
1	0.81	0.88	0.84	2952
Accuracy			0.81	5058
Macro avg	0.81	0.79	0.80	5058
Weighted avg	0.81	0.81	0.80	5058

Table 11 shows the results of fault detection on tool status. We obtain a precision of 0.81 for class 0 and 0.81 for class 1. Recall is 0.70 for class 0 and 0.88 for class 1. The F1 score is 0.75 for class 0 and 0.84 for class 1. These results demonstrate a satisfactory overall performance of the MLP model in detecting tool state failures.

Table 12. Performance MLP on output machine finalized

	Precision	Recall	F1-Score	Support
0	0.77	0.66	0.71	1165
1	0.90	0.94	0.92	3893
Accuracy			0.88	5058
Macro avg	0.84	0.80	0.82	5058
Weighted avg	0.87	0.88	0.87	5058

Table 13. Performance MLP on output passed visual inspection

	Precision	Recall	F1-Score	Support
0	0.75	0.76	0.76	2430
1	0.78	0.76	0.77	2628
Accuracy			0.76	5058
Macro avg	0.76	0.76	0.76	5058
Weighted avg	0.76	0.76	0.76	5058

Table 12 shows the results of fault detection on machine finalization. We obtain a precision of 0.77 for class 0 and 0.90 for class 1. Recall is 0.66 for class 0 and 0.94 for class 1. The F1 score is 0.71 for class 0 and 0.92 for class 1. These results demonstrate the high performance of the MLP model in detecting faults in the finalization of the machine.

Table 13 shows the results of fault detection on successful visual inspection. We obtain a precision of 0.75 for class 0 and 0.78 for class 1. Recall is 0.76 for class 0 and 0.76 for class 1. The F1 score is 0.76 for class 0 and 0.77 for class 1.

The overall accuracy of 82% is encouraging, and the model can identify faults with a high degree of confidence. However, there are some areas where the model could be improved. For example, the precision for the "passed visual inspection" class is relatively low, indicating that the model may be making a high number of false positives in this case.

4.2 CNN model

In the CNN section of our study, we used the CNN (Convolutional Neural Network) model for fault detection on three different criteria: tool condition, machine completion and visual inspection. The results obtained are presented in Tables 14-16.

Table 14. Performance CNN on output tool condition

	Precision	Recall	F1-Score	Support
0	0.85	0.76	0.80	2370
1	0.80	0.88	0.84	2688
Accuracy			0.82	5058
Macro avg	0.83	0.82	0.82	5058
Weighted avg	0.82	0.82	0.82	5058

Table 15. Performance CNN on output machine finalized

	Precision	Recall	F1-Score	Support
0	0.79	0.79	0.79	1257
1	0.93	0.93	0.93	3801
Accuracy			0.89	5058
Macro avg	0.86	0.86	0.86	5058
Weighted avg	0.89	0.89	0.89	5058

Table 16. Performance CNN on output passed visual inspection

	Precision	Recall	F1-Score	Support
0	0.80	0.84	0.82	2467
1	0.84	0.80	0.82	2591
Accuracy			0.82	5058
Macro avg	0.82	0.82	0.82	5058
Weighted avg	0.82	0.82	0.82	5058

While the CNN model demonstrates solid performance, there are nuances to consider. Despite high precision and recall scores in some cases, there remains room for improvement in achieving balanced performance across all metrics. A potential implication is the model's vulnerability to misclassifications in specific fault categories. This can lead to costly errors if not mitigated. Further investigation is required to determine the underlying reasons behind the model's varying performance and to fine-tune its hyperparameters.

In summary, the CNN model exhibits robust performance in various fault detection criteria, achieving an overall accuracy of 85%. However, an in-depth analysis reveals potential challenges and the need for further refinement.

4.3 VAE CNN model

The results obtained for VAE are presented in Tables 17-19.

It is noteworthy that the VAE CNN model achieves lower results compared to the previous models. This difference could be attributed to the model's emphasis on generalization. The VAE is known for its ability to generate data, underscoring the importance of robust generalizability in various applications. Despite these relatively lower scores, the VAE CNN model remains an intriguing option for fault detection, considering its potential for generating valuable insights and data. The VAE CNN model's relatively lower performance could indicate challenges in effectively capturing complex patterns and representations inherent in the dataset. The focus on data generation might have led to a trade-off between predictive performance and generalization. This suggests the need for more advanced architectures and careful hyperparameter tuning to enhance the model's performance. While the model's accuracy of 76% is lower compared to other models, its potential to contribute to data generation and exploration remains intriguing, making it an avenue worthy of further exploration and refinement.

In conclusion, the VAE CNN model exhibits promising potential despite its relatively lower performance on various fault detection criteria. The model's emphasis on data generation and generalization underscores its unique capabilities, opening the door for future advancements and improvements.

Table 17. Performance VAE on output tool condition

	Precision	Recall	F1-Score	Support
0	0.69	0.69	0.69	2370
1	0.73	0.72	0.72	2688
Accuracy			0.71	5058
Macro avg	0.71	0.71	0.71	5058
Weighted avg	0.71	0.71	0.71	5058

Table 18. Performance VAE on output machine finalized

	Precision	Recall	F1-Score	Support
0	0.75	0.52	0.61	1257
1	0.86	0.94	0.90	3801
Accuracy			0.84	5058
Macro avg	0.80	0.73	0.75	5058
Weighted avg	0.83	0.84	0.83	5058

Table 19. Performance VAE on output passed visual inspection

	Precision	Recall	F1-Score	Support
0	0.79	0.63	0.70	2467
1	0.71	0.84	0.77	2591
Accuracy			0.74	5058
Macro avg	0.75	0.74	0.74	5058
Weighted avg	0.75	0.74	0.74	5058

4.4 ResNet model

The results presented in this section concern the Resnet model, and Tables 20-23 summarize the results in terms of performance on all the criteria.

The ResNet model showcases robust performance across various fault detection criteria, with an overall accuracy of 84%. Its balanced precision, recall, and F1-scores indicate its proficiency in classifying diverse fault categories. The model's relatively high accuracy is a testament to its effectiveness in generalizing its learning across different fault types, making it a promising option for robust fault detection.

While the ResNet model demonstrates commendable performance, interpretability of the model's decisions could be a challenge, as deep learning architectures often lack transparency. Additionally, fine-tuning the hyperparameters could potentially enhance the model's performance. Ensuring a comprehensive understanding of the model's strengths and limitations can guide further development and improvements.

In conclusion, the ResNet model emerges as a reliable and robust contender for fault detection across various criteria. Its strong overall accuracy and balanced performance in precision, recall, and F1-scores underscore its potential to contribute effectively to fault diagnosis tasks. As we progress, the ResNet model provides a solid foundation for advancing the field of CNC milling fault detection and potentially inspiring further research in deep learning-based fault detection methodologies.

Table 20. Performance ResNet on output tool condition

	Precision	Recall	F1-Score	Support
0	0.78	0.80	0.79	2106
1	0.86	0.84	0.85	2952
Accuracy			0.82	5058
Macro avg	0.82	0.82	0.82	5058
Weighted avg	0.82	0.82	0.82	5058

Table 21. Performance ResNet on output machine finalized

	Precision	Recall	F1-Score	Support
0	0.81	0.71	0.76	1165
1	0.92	0.95	0.93	3893
Accuracy			0.89	5058
Macro avg	0.86	0.83	0.84	5058
Weighted avg	0.89	0.89	0.89	5058

Table 22. Performance ResNet on output passed visual inspection

	Precision	Recall	F1-Score	Support
0	0.81	0.74	0.77	2430
1	0.78	0.84	0.81	2628
Accuracy			0.79	5058
Macro avg	0.79	0.79	0.79	5058
Weighted avg	0.79	0.79	0.79	5058

4.5 LSTM model

The results obtained for LSTM are presented in Tables 23-25.

The outcomes of this section, as encapsulated in Table 23-25, showcase the LSTM model's prowess in consistently delivering top-tier performance.

Tool Condition Detection (Table 23): The LSTM model shines with precision scores of 0.91 for class 0 (healthy tool) and a remarkable 0.94 for class 1 (faulty tool). These figures highlight the model's exceptional precision in distinguishing between tool conditions. Recall rates of 0.93 for both classes affirm the model's robustness in correctly identifying instances from both categories. The corresponding F1-scores of 0.92 and 0.94 underscore the model's balanced precision and recall performance. With an overall accuracy of 93%, the LSTM model showcases its prowess in accurately classifying tool conditions.

Machine Finalization Detection (Table 24): For machine finalization detection, the LSTM model demonstrates a precision of 0.93 for class 0 and an impressive 0.96 for class 1. These precision scores underscore the model's ability to identify both successful and problematic machine finalizations with high precision. Recall rates of 0.90 for class 0 and an outstanding 0.97 for class 1 highlight the model's robustness in accurately identifying instances from both categories. The resulting F1-scores of 0.91 and 0.97 further emphasize the model's balanced performance. An accuracy of 95% accentuates the LSTM model's exceptional generalization.

Table 23. Performance LSTM on output tool condition

	Precision	Recall	F1-Score	Support
0	0.93	0.90	0.91	1322
1	0.96	0.97	0.97	3736
Accuracy			0.95	5058
Macro avg	0.95	0.94	0.94	5058
Weighted avg	0.95	0.95	0.95	5058

Table 24. Performance LSTM on output machine finalized

	Precision	Recall	F1-Score	Support
0	0.94	0.93	0.94	2658
1	0.92	0.94	0.93	2400
Accuracy			0.93	5058
Macro avg	0.93	0.93	0.93	5058
Weighted avg	0.93	0.93	0.93	5058

Table 25. Performance LSTM on output passed visual inspection

Model	Hidden Layers	Activation	Optimizer	Learning Rate
MLP	(50, 100)	sigmoid, tanh	Adam	0.001
CNN	(250, 200, 128)	tanh	RMSprop	0.005
VAE	(64, 300)	tanh	Sgd	0.002
LSTM	(351, 374, 129)	tanh	RMSprop	0.005
ResNet	(16, 128)	Relu, tanh	Sgd	0.1

Visual Inspection Detection (Table 25): During visual inspection detection, the LSTM model exhibits precision scores of 0.94 for class 0 and 0.92 for class 1. These figures exemplify the model's proficiency in accurately differentiating between instances of successful and unsuccessful visual inspections. Recall rates of 0.93 for both classes underline the model's ability to correctly identify instances from both categories. The F1-scores of 0.94 and 0.93 highlight the model's balanced performance. An accuracy of 93% further cements the LSTM model's robust generalization.

The LSTM model's standout performance lies in its ability to consistently deliver high precision and recall across all fault detection criteria. Its balanced F1-scores indicate that it effectively maintains a harmony between precision and recall. With an impressive overall accuracy of 94%, the LSTM model demonstrates exceptional capabilities in classifying various fault categories accurately.

The LSTM model's remarkable performance holds significant implications for industrial equipment fault detection and predictive maintenance systems. Its ability to excel across different criteria underscores its potential to enhance industrial operations by facilitating early fault detection and minimizing downtime. The model's suitability for real-world applications positions it as a key player in the evolution of efficient and reliable fault detection mechanisms.

In summary, Table 26 provides a comprehensive overview of the different models employed in this study, highlighting key aspects of their architectures and hyperparameter configurations. The Multi-Layer Perceptron (MLP) model incorporates two hidden layers with neuron sizes of 50 and 100, utilizing the Adam optimizer with a learning rate of 0.001. On the other hand, the Convolutional Neural Network (CNN) exhibits a more complex structure with hidden layers of sizes 250, 200, and 128, employing the tanh activation function along with the RMSprop optimizer and a learning rate of 0.005. The Variational Autoencoder (VAE) model follows a similar path with hidden layers of sizes 64 and 300, using the tanh activation function in combination with the SGD optimizer and a learning rate of 0.002. The Long Short-Term Memory (LSTM) model showcases its architecture with hidden layers of sizes 351, 374, and 129, integrating the tanh activation function and RMSprop optimizer with a learning rate of 0.005. Lastly, the ResNet model, distinguished by hidden layers of sizes 16 and 128, aligns with the SGD optimizer and a higher learning rate of 0.1. This comprehensive comparison underscores the diversity of the models employed in this study, paving the way for a thorough evaluation of their performance on the CNC Milling Dataset.

As we reflect upon the outcomes of our comprehensive study, it becomes evident that our findings resonate deeply with the overarching research goals we initially outlined. The intent of our investigation was to meticulously compare and evaluate the performance of five prominent deep learning network models in the domain of fault diagnosis for CNC machines. The conclusions drawn from our analysis substantiate the significance of such an endeavor. By empirically assessing the strengths and limitations of the Multilayer Perceptron (MLP), Convolutional Neural Network (CNN), CNN autoencoder, Long Short-Term Memory (LSTM), and Residual Network (ResNet), we have contributed a holistic understanding of their applicability in the context of complex industrial data analysis.

Furthermore, our findings coalesce harmoniously with existing literature in the field of fault diagnosis for CNC machines. While previous research efforts have made strides in the domain, our study addresses certain crucial gaps that have been identified in the current state of research. By providing specific guidance on model selection, configuration, and multi-output classification challenges, our research advances the literature's understanding of tackling intricate fault diagnosis scenarios. The alignment of our findings with previous work underscores the validity and relevance of our study's objectives and outcomes.

Our research goes beyond theoretical exploration, offering practical implications that resonate with manufacturers seeking to enhance their fault diagnosis and predictive maintenance practices. The elucidation of the strengths and weaknesses of each deep learning model equips manufacturers with actionable insights for informed decision-making. Manufacturers can now tailor their model selection based on their specific industrial requirements and data characteristics. By implementing the most suitable model, manufacturers can proactively identify and address faults, thereby minimizing operational disruptions, reducing downtime, and optimizing overall production efficiency.

In conclusion, our study's profound alignment with original research objectives and existing literature reaffirms its significance. The practical implications extend well beyond research realms, offering manufacturers a roadmap to bolster their operational efficiency and reliability through informed model selection and proactive fault detection strategies. By merging theory with practicality, our research seeks to catalyze the evolution of industrial fault diagnosis in the context of CNC machines.

Table 26. Comparison of all models

Model	Hidden Layers	Activation	Optimizer	Learning Rate
MLP	(50, 100)	sigmoid, tanh	Adam	0.001
CNN	(250, 200, 128)	tanh	RMSprop	0.005
VAE	(64, 300)	tanh	Sgd	0.002
LSTM	(351, 374, 129)	tanh	RMSprop	0.005
ResNet	(16, 128)	Relu, tanh	Sgd	0.1

5. DISCUSSIONS

Our investigation has culminated in a comprehensive analysis of various deep network models applied to the realm of industrial equipment fault detection, as outlined in Table 27. This section affords us the opportunity to connect our findings back to the core research aims that propelled this study into motion. The central objective was to scrutinize the efficacy of five prevalent deep learning models in the domain of fault detection, culminating in the attainment of predictive maintenance excellence. Our exploration has not only fulfilled this intent but also delivered a myriad of key insights and practical implications that are poised to revolutionize the landscape of industrial fault diagnosis.

Our study ventures into novel terrain, shedding light on the performance dynamics of the Multilayer Perceptron (MLP), Convolutional Neural Network (CNN), Variational Autoencoder (VAE), Long Short-Term Memory (LSTM), and Residual Network (ResNet) models. These models, each bearing unique attributes tailored for intricate data analysis and fault signal detection, uncover the intricate relationships between machine conditions and signals of impending faults.

The VAE model, though yielding relatively lower performance than its counterparts, unveils a significant facet - its potential as a tool for data generation. This unique ability to synthesize data makes the VAE a remarkable avenue for further exploration, potentially transcending its initial performance drawbacks.

The MLP model showcases commendable performance across diverse criteria, highlighting its potential as a reliable tool for fault detection across multiple dimensions. Achieving accuracies of 82% in tool condition detection, 88% in machine completion, and 76% in visual inspection, the MLP model emerges as a versatile contender for real-world industrial applications.

The ResNet model further solidifies the landscape of fault detection, attaining an overall accuracy of 84%. Its capabilities shine particularly bright in tool condition detection (79%), machine completion (92%), and visual inspection (78%), positioning the ResNet as a robust solution with potential implications for predictive maintenance frameworks.

The CNN model, showcasing its prowess, achieves an impressive 85% overall accuracy in fault detection across various criteria. Its aptitude for capturing pertinent features bolsters its reputation as a viable approach for intricate fault signal analysis.

The LSTM model, incontestably the frontrunner, demonstrates the pinnacle of performance with a remarkable 94% overall accuracy. Its skill in capturing temporal dependencies stands as a testament to its potential in unravelling the complexities of CNC machine data, propelling it to the forefront of predictive maintenance systems development.

Our study does not exist in isolation; it intersects with previous research, particularly [15, 16]. While our LSTM model's accuracy may not reach the heights of decision tree-based models, the true value lies in its capability to generalize. Our 95% LSTM accuracy, albeit not the highest, excels in capturing temporal dependencies, outperforming decision trees and confirming the significance of our approach.

In conclusion, our study emerges as a pivotal milestone in the quest for superior fault diagnosis and predictive maintenance in industrial equipment. The LSTM model, with its unmatched performance, serves as a beacon for future research endeavours. The implications are clear: harnessing deep learning models for early fault detection, manufacturers can champion operational efficiency and reliability. The synthesis of theory and practice laid forth in our study sets the stage for a new era of data-driven decision-making, poised to transform industrial maintenance landscapes.

Table 27. Global accuracy of all models

Models	Precision
VAE CNN	76%
MLP	82%
ResNet	84%
CNN	85%
LSTM	94%

6. CONCLUSION

Our research delved into the extensive realm of industrial equipment fault detection using deep learning models. Throughout this study, we scrutinized and compared the performance of five prominent deep network architectures: Multilayer Perceptron (MLP), Convolutional Neural Network (CNN), Variational Autoencoder with CNN (VAE CNN), Long Short-Term Memory (LSTM), and Residual Network (ResNet). Our investigation was driven by the imperative of informed decision-making in model selection and configuration for fault diagnosis in CNC machines.

The results obtained through our rigorous evaluation shed light on the strengths and weaknesses of each model. The MLP model showcased its efficacy in detecting faults across diverse criteria, achieving notable accuracies in tool condition detection, machine completion, and successful visual inspection. Meanwhile, the CNN model excelled in capturing relevant features for fault detection, achieving commendable performance across the evaluated aspects. The VAE CNN, despite yielding lower results, remains an intriguing option due to its data generation capability. The ResNet model exhibited substantial effectiveness, particularly in fault detection and the development of predictive maintenance systems. However, the pinnacle of our study was the LSTM model, which triumphed in capturing temporal dependencies and achieved a remarkable overall accuracy of 94%.

In relation to the original research aims, our study successfully unravelled the strengths, weaknesses, and capabilities of each model, providing valuable insights for future decision-making. Furthermore, we addressed existing gaps in the field by providing specific guidance on model selection and configuration, and by offering a comprehensive framework for solving multi-output classification problems in fault diagnosis on CNC machines.

Our findings contribute to the existing literature by confirming the potential of deep learning-based approaches in fault detection, even amidst the complexities of industrial equipment. While our LSTM accuracy may not rival decision tree-based models in certain aspects, its prowess in avoiding overfitting and capturing temporal dependencies adds a unique layer of significance to our research.

From a practical standpoint, our results hold implications for manufacturers seeking robust fault diagnosis solutions. The success of our models in analysing complex data and detecting fault signals opens doors to the development of predictive maintenance systems. With the ability to detect anomalies and predict potential malfunctions, manufacturers can proactively address issues, minimize downtime, and optimize equipment performance.

Nonetheless, our study is not without limitations. We acknowledge the need for further exploration in hyperparameter optimization and the extension of our approach to diverse industrial equipment types and larger datasets. As the field of deep learning continues to evolve, we anticipate that the insights gained from our research will contribute to the ongoing enhancement of predictive maintenance practices in the manufacturing industry.

REFERENCES

[1] Melgarejo Agudelo, C.F., Blanco Rodriguez, J.J., Maradey Lázaro, J.G. (2020). Bearing fault detection and classification: A framework approach. Volume 7A: Dynamics, Vibration, and Control. American Society of Mechanical Engineers. https://doi.org/10.1115/IMECE2020-24124

[2] Majidi, M., Fadali, M.S., Etezadi-Amoli, M., Oskuoee, M. (2015). Partial discharge pattern recognition via sparse representation and ANN. IEEE Transactions on Dielectrics and Electrical Insulation, 22(2): 1061-1070. https://doi.org/10.1109/TDEI.2015.7076807

[3] Azamfar, M., Singh, J., Li, X., Lee, J. (2021). Cross-domain gearbox diagnostics under variable working conditions with deep convolutional transfer learning. Journal of Vibration and Control, 27(7-8): 854-864. https://doi.org/10.1177/1077546320933793

[4] Sarita, K., Devarapalli, R., Kumar, S., Malik, H., García Márquez, F.P., Rai, P. (2022). Principal component analysis technique for early fault detection. Journal of Intelligent & Fuzzy Systems, 42(2): 861-872. https://doi.org/10.3233/JIFS-189755

[5] Amy, R.A., Aglietti, G.S., Richardson, G. (2009). Reliability analysis of electronic equipment subjected to shock and vibration - A review. Shock and Vibration, 16: 45-59. https://doi.org/10.1155/2009/546053

[6] Saufi, S.R., Ahmad, Z.A.B., Leong, M.S., Lim, M.H. (2019). Challenges and opportunities of deep learning models for machinery fault detection and diagnosis: A review. IEEE Access, 7: 122644-122662. https://doi.org/10.1109/ACCESS.2019.2938227

[7] Obaid, M.H., Hamad, A.H. (2023). Deep learning approach for oil pipeline leakage detection using image-based edge detection techniques. Journal Européen des Systèmes Automatisés, 56(4): 663-673. https://doi.org/10.18280/jesa.560416

[8] Yang, Y., Haque, M.M.M., Bai, D., Tang, W. (2021). Fault diagnosis of electric motors using deep learning algorithms and its application: A review. Energies (Basel), 14(21): 7017. https://doi.org/10.3390/en14217017

[9] Neupane, D., Seok, J. (2020). Bearing fault detection and diagnosis using case western reserve university dataset with deep learning approaches: A review. IEEE Access, 8: 93155-93178. https://doi.org/10.1109/ACCESS.2020.2990528

[10] Jian, Y., Qing, X., Zhao, Y., He, L., Qi, X. (2020). Application of model-based deep learning algorithm in fault diagnosis of coal mills. Mathematical Problems in Engineering, 2020: 1-14. https://doi.org/10.1155/2020/3753274

[11] Mohammed, N.A., Abdulateef, O.F., Hamad, A.H. (2023). An IoT and machine learning-based predictive maintenance system for electrical motors. Journal Européen des Systèmes Automatisés, 56(4): 651-656. https://doi.org/10.18280/jesa.560414

[12] Glaeser, A., Selvaraj, V., Lee, S., Hwang, Y., Lee, K., Lee, N., Lee, S., Min, S. (2021). Applications of deep learning for fault detection in industrial cold forging. International Journal of Production Research, 59(16): 4826-4835. https://doi.org/10.1080/00207543.2021.1891318

[13] Rao, G.S.V., Diwanji, V., Parthasarathi, J. (2012). Application case study of machine learning techniques towards a fault diagnosis system for a manufacturing plant environment. In Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication, New York, NY, USA: ACM, pp. 1-4. https://doi.org/10.1145/2184751.2184798

[14] Mahdi, M.A., Gittaffa, S.A., Issa, A.H. (2022). Multiple fault detection and smart monitoring system based on machine learning classifiers for infant incubators using raspberry Pi 4. Journal Européen des Systèmes Automatisés, 55(6): 771-778. https://doi.org/10.18280/jesa.550609

[15] Park, S., Lee, K., Sung, S., Park, D. (2019). Prediction of the CNC tool wear using the machine learning technique. 2019 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, pp. 296-299. https://doi.org/10.1109/CSCI49370.2019.00059

[16] Shurrab, S., Almshnanah, A., Duwairi, R. (2021). Tool wear prediction in computer numerical control milling operations via machine learning. In 12th International Conference on Information and Communication Systems (ICICS), Valencia, Spain, pp. 220-227. https://doi.org/10.1109/ICICS52457.2021.9464580

[17] Polat, K. (2020). The fault diagnosis based on deep long-term memory model from the vibration signals in the computer numerical control machines. Journal of the Institute of Electronics and Computer, 2: 72-92. https://doi.org/10.33969/JIEC.2020.21006

[18] Zhang, Z., Cao, S., Cao, J. (2018). fault diagnosis of servo drive system of CNC machine based on deep learning. In Chinese Automation Congress (CAC), Xi'an, China, pp. 1873-1877. https://doi.org/10.1109/CAC.2018.8623472

[19] Chung, C.C., Liang, Y.P., Chang, Y.C., Chang, C.M. (2023). A binary weight convolutional neural network hardware accelerator for analysis faults of the CNC

machinery on FPGA. In 2023 International VLSI Symposium on Technology, Systems and Applications (VLSI-TSA/VLSI-DAT), HsinChu, pp. 1-4. https://doi.org/10.1109/VLSI-TSA/VLSI-DAT57221.2023.10134316

[20] Bouktif, S., Fiaz, A., Ouni, A., Serhani, M. (2018). Optimal deep learning LSTM model for electric load forecasting using feature selection and genetic algorithm: Comparison with machine learning approaches. Energies (Basel), 11(7): 1636. https://doi.org/10.3390/en11071636

[21] Pan, Y., Yang, Y., Li, W. (2021). A deep learning trained by genetic algorithm to improve the efficiency of path planning for data collection with multi-UAV. IEEE Access, 9: 7994-8005. https://doi.org/10.1109/ACCESS.2021.3049892

[22] Levy, E., David, O.E., Netanyahu, N.S. (2014). Genetic algorithms and deep learning for automatic painter classification. In Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation, New York, NY, USA: ACM, pp. 1143-1150. https://doi.org/10.1145/2576768.2598287

[23] Naskath, J., Sivakamasundari, G., Begum, A.A.S. (2023). A Study on different deep learning algorithms used in deep neural nets: MLP SOM and DBN. Wireless Personal Communications, 128: 2913-2936. https://doi.org/10.1007/s11277-022-10079-4

[24] Widiasari, I.R., Nugroho, L.E., Widyawan. (2017). Deep learning multilayer perceptron (MLP) for flood prediction model using wireless sensor network based hydrology time series data mining. International Conference on Innovative and Creative Information Technology (ICITech), Salatiga, Indonesia, pp. 1-5. https://doi.org/10.1109/INNOCIT.2017.8319150

[25] Teoh, T.T., Chiew, G., Franco, E.J., Ng, P.C., Benjamin, M.P., Goh, Y.J. (2018). Anomaly detection in cyber security attacks on networks using MLP deep learning. In 2018 International Conference on Smart Computing and Electronic Enterprise (ICSCEE), Shah Alam, Malaysia, pp. 1-5. https://doi.org/10.1109/ICSCEE.2018.8538395

[26] Kiangala, K.S., Wang, Z. (2020). An effective predictive maintenance framework for conveyor motors using dual time-series imaging and convolutional neural network in an industry 4.0 environment. IEEE Access, 8: 121033-121049. https://doi.org/10.1109/ACCESS.2020.3006788

[27] Zhang, S., Ye, F., Wang, B., Habetler, T.G. (2021). Semi-supervised bearing fault diagnosis and classification using variational autoencoder-based deep generative models. IEEE Sensors Journal, 21(5): 6476-6486. https://doi.org/10.1109/JSEN.2020.3040696

[28] Remadna, I., Terrissa, L.S., Al Masry, Z., Zerhouni, N. (2023). RUL prediction using a fusion of attention-based convolutional variational AutoEncoder and ensemble learning classifier. IEEE Transactions on Reliability, 72(1): 106-124. https://doi.org/10.1109/TR.2022.3190639

[29] Bruneo, D., De Vita, F. (2019). On the use of LSTM networks for predictive maintenance in smart industries. In IEEE International Conference on Smart Computing (SMARTCOMP), Washington, DC, USA, pp. 241-248. https://doi.org/10.1109/SMARTCOMP.2019.00059

[30] Rahhal, J.S., Abualnadi, D. (2020). IOT based predictive maintenance using LSTM RNN estimator. In International Conference on Electrical, Communication, and Computer Engineering (ICECCE), Istanbul, Turkey, pp. 1-5. https://doi.org/10.1109/ICECCE49384.2020.9179459

[31] Wei, H., Zhang, Q., Shang, M., Gu, Y. (2021). Extreme learning Machine-based classifier for fault diagnosis of rotating Machinery using a residual network and continuous wavelet transform. Measurement, 183: 109864. https://doi.org/10.1016/j.measurement.2021.109864

[32] Ahmed, M., Kamal, K., Ratlamwala, T.A.H., Hussain, G., Alqahtani, M., Alkahtani, M., Alzabidi, A. (2023). Tool health monitoring of a milling process using acoustic emissions and a ResNet deep learning model. Sensors, 23(6): 3084. https://doi.org/10.3390/s23063084