ARTICLE

JOURNAL`S UNIVERSITY OF BABYLON FOR ENGINEERING SCIENCES (JUBES)

مـــجلـة جـــامعة بـــابـل للعلـــوم الهندسية

Vol. 33, No. 1. \ 2025 ISSN: 2616 - 9916

Efficient implementation of a tiny deep learning classifier based on vibration-related fault detection using limited resources hardware

Hilal Al-Libawy

Department of Electrical Engineering, College of Engineering, University of Babylon, Babylon, Iraq

Email: Eng.hilal_al-libawy@uobabylon.edu.iq

Received:	21/1/2025	Accepted:	2/2/2025	Published:	28/2/2025

Abstract

The dramatic evolution of limited-resource hardware such as microcontrollers in conjunction with the development of machine learning algorithms has helped open doors to building smart devices in industrial sectors. Predictive maintenance is one of these sectors that uses smart devices efficiently. Vibration data such as data collected from accelerometers can capture precisely any change in moving machine behavior due to mechanical wearing in moving parts such as bearings. In this work, a deep learning one-dimensional convolutional neural network (1DCNN) classifier is built as a fault detector and tuned to enhance detection performance. This classifier is tested using a publicly available vibration dataset for three types of rotating bearing status (healthy, inner race fault, and outer race fault). The classifier is designed using a tiny machine-learning framework, which can be implemented using a microcontroller with limited resources. The accelerometer data is preprocessed using a spectrogram of a vibration signal to extract frequency-time-related features to enhance classifier performance. Moreover, the classifier is quantized using an eight-bit integer to reduce calculation time and required memory. TinyML framework environment handles the building of the ternary classifier and helps to implement this classifier on limited resources hardware. This ternary classifier achieves accurate results with an accuracy of 98.64% and F1 score of 0.99. However, these accurate results were achieved using minimal resources of an Arduino microcontroller with a RAM of 8.3kb and a latency of 20ms.

Keywords: TinyML; Vibration; Limited resources hardware; CNN.

I. Introduction:

Industry 4.0 leverages many important aspects such as predictive maintenance, fault detection, and anomaly detection based on advanced technologies such as machine learning (ML), Internet of Things (IoT), and big data analytics. These new technologies facilitate the way for continuous monitoring and analysis of equipment conditions, enabling the early detection of potential failures before they occur improving the chances for workflow, and reducing downtime of industry processes. Machine learning algorithms, including deep learning, ensemble methods,

and anomaly detection, have been shown to effectively predict maintenance requirements by analyzing sensor and operational data, although challenges such as data quality and model generalization remain [1] [2].

Electric motors are an essential part of each factory and industry sector and this raises the need to guarantee the continuous operation of these parts by using smart and low-cost fault detection techniques. Deep learning algorithms such as Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs) have gained a reputation for being used in prediction maintenance and fault detection solutions, especially when these solutions must be implemented with limited resources and hardware. Machine learning algorithms are known to be data-driven models and their performance is significantly affected by data size and type including data collection tool and noise behind that. So, it is very important to choose training raw data carefully including spatiotemporal patterns generated from faults in electrical motors. Recently, many researchers have developed and optimized deep learning algorithms that can fit microcontroller unit (MCU) resources to cover the large need for fault detection devices. The researchers are trying hard to balance between MCU and RAM constraints from one side and detector performance from the other side [3]. Overall, deep learning contributes to fault analysis and detection across different electrical motors and mechanical systems overcoming traditional techniques that use manual analysis and rely on experts' knowledge [4]. However, while deep learning achieves remarkable progress in the fault detection field, especially with deployments on limited resources hardware, there are still serious challenges that need to be addressed and solved to be widely and confidently used in the industry. One of these challenges is the need for large amounts of data to train deep learning algorithms such as CNNs to achieve the required accuracy. Usually, it is not very affordable to build large datasets that can cover all types of faults including all types of electrical motors which leads to unreliable fault detection outcomes. This untrusted model may raise serious concerns about relying on deep learning approaches in fault detection models which is critical in safety-sensitive applications. Additionally, deploying a deep learning model on resource-constrained hardware like MCUs may not always offer the required performance, so a tradeoff between performance and resource consumption always needs to be handled carefully [5].

In this work, a lite deep learning algorithm is tuned and optimized using a publicly available dataset for ball-bearing fault detection. The model is successfully quantized using an integer 8 quantizer and tuned to reduce its size and computational power needed to fit the low-cost and resources-constrained microcontroller. The proposed model was successfully deployed on the Arduino 33BLE Arduino board and gained an accurate detection rate. Additionally, the time-frequency feature generation stage is used to enhance the classification performance of the three-classes model. The proposed framework optimizes hardware resources and three-classes model performance. The remainder of the research paper is as follows: section II overviews related work, section III explains in detail aspects and background theory behind vibration-related fault detection framework, then Section IV overviews and explains the proposed system. Finally, section V illustrates the main results of this work discusses them, and ends with conclusions of this work.

II. Related work:

Different approaches have been followed to analyze rotating machine faults such as signal processing and machine learning approaches. Vibration-related data of fault diagnosing in electrical motors has complex spatiotemporal patterns, so deep learning models should have the ability to detect these patterns [6] [7], [8]. Andrei S. Maliuk and his colleagues suggested a Gaussian mixture model-based (GMM) using frequency bands for feature extraction followed by a K-nearest neighbor (KNN) classifier [9]. Recent literature has examined thoroughly using of deep learning models in fault analysis and detection areas of electrical motors. Many deep learning algorithms have been used and trained using vibration-related data for electrical motor fault analysis. However, several models have shown substantial accuracy in fault detection such as Convolutional Neural Networks (CNNs) [10] and Domain adaptation network based on Long Short-Term Memory (DA-LSTM) proposed by Kumar and his colleagues [11]. Zhao and his colleagues developed a diagnosis model named deep branch attention network (DBANet), tested it on publicly available dataset, and compared results with several existing deep learning models [12]. A lightweight model has been suggested by Yan and his colleagues for fault detection based on separable multiscale convolution and broadcast self-attention and tested using publicly available dataset [13]. However, real-time implementation is still not addressed thoroughly from the view of inference time and required resources of hardware such as RAM and microcontroller abilities.

On the other hand, the implementation feasibility of these deep learning models on limited-resource hardware such as microcontrollers (MCUs), many studies have explored that by optimizing the deep learning model to tradeoff between fault detection performance and memory usage[14] [15]. To overcome the need for a large amount of data to train deep learning models, Transfer learning techniques have also been employed, as seen in the application of models like ResNet152 for induction motor fault detection [16]. The outstanding success of using deep learning algorithms in fault analysis and detection in electrical motors and mechanical systems offers reliable solutions. These solutions can be implemented trustily on hardware-constrained resources, thereby supporting predictive maintenance and reducing operational disruptions [17]. Although some researchers have been working on fault detection models and implementing them in real-time, there is still an area of performance enhancement that needs to be covered with new research work.

III. Vibration-Based Fault Detection Techniques:

Generated vibration from rotating parts of electrical rotating machines can supply information about the health status of these machines including but not limited to bearing faults and unbalanced faults [18]. Frequency domain analysis is an essential tool in vibration-related fault analysis solutions which normally collect data as time-series signals using inertial sensors such as accelerometer and gyroscope or vibration sensors [19]. Different types of vibration-based techniques have been used in fault detection frameworks such as Fourier transform, power spectral density wavelet, spectrogram, and others [20], [21]. The spectrogram technique gained a good reputation for fault detection of electrical motors especially when it relates to ball bearing faults and this reputation comes from accurate results of detection when using spectrogram as a feature extractor for machine learning algorithms [22].

A. Spectrogram

The spectrogram is a visual representation of a time-series signal such as an audio or accelerometer signal based on time-frequency analysis. It depends primarily on Short-Time Fourier Transform (STFT) and computing the magnitude of STFT as explained in the following equations [23]:

$$STFT(t,f) = \int_{\infty}^{\infty} x(\tau)\omega(\tau - t)e^{-i2\pi f\tau}d\tau \qquad (1)$$

Where:

$x(\tau)$: is the time domain input signal (such as the accelerometer signal)

$\omega(\tau - t)$: is the window function which isolates time segments and is centered around *t*

f: is the required frequency to be calculated from the time segment.

After the calculation of STFT, the spectrogram can be derived from the power spectral density (PSD) of STFT or the squared magnitude as follows:

$$\text{Spectrogram}(t,f) = |STFT(t,f)|^2 \qquad (2)$$

The spectrogram is calculated using the following steps:

1. The input signal $x(t)$ is divided into overlapping segments using the window function $\omega(t)$
2. Applying Fourier transform for each segment using STFT.
3. Calculate the power spectral density of each time-frequency paired segment.
4. Visualization of the result PSD over a two-dimensional plot which represents time over the X-axis and frequency over the Y-axis and the power is represented by intensity or color.

The spectrogram representation is affected widely by window size and overlap time and these factors should be selected carefully based on signal behavior and signal complexity. Moreover, a tradeoff between time resolution which is improved by a shorter window, and frequency resolution which is improved by a wider window.

B. Deep learning classifier and Tiny Machine Learning

Deep learning (DL) offers significant progress in fault detection and diagnosis in electrical machines compared with traditional methods. The algorithms of deep learning such as convolutional neural network (CNN) and recurrent neural network (RNN) have been used widely in fault detection and classify types of faults efficiently based on vibration signals such as signals generated from accelerometers [10]. DL classifiers are data-driven models so they perform accurately if the training data is collected properly and represent all types of faults. However, many performance metrics have been used to test the ability and performance of the classifier and some of the main metrics are detailed as follows [24]:

Accuracy is mainly used to evaluate classifier performance, especially with balanced data and it can be calculated as in equation 3:

$$Accuracy\% = \frac{T_P + T_N}{T_P + T_N + F_P + F_N} \times 100\% \quad (3)$$

Where T_P: is the positive instants that are classified correctly

T_N: is the negative instants that are classified correctly

F_P: is the positive instants that misclassified

F_N: is the negative instants that misclassified

Another metric is precision which is usually used when a false alarm is costly and calculated as in equation 4:

$$Precision = \frac{T_P}{T_P + F_P} \quad (4)$$

While recall metrics focus on negative false rather than positive false and are calculated as in equation 5

$$Recall = \frac{T_P}{T_P + F_N} \quad (5)$$

However, both metrics precision and recall can be gathered in one metric called F1 score and calculated as in equation 6

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (6)$$

One of the classifier performance metrics is named Area Under the Receiver Operating Characteristic Curve (AUC-ROC) which measures the ability of the classifier to separate between fault and non-fault classes with different values of threshold. It is important to mention that a multi-class classifier weighted metric is used to count the number of instants related to each class. Recently, electrical machine fault detection and classification have been widely used in factories and industrial plants, especially with concepts of Industry 4.0. One reason behind this spread is the ability to implement deep learning algorithms on low-cost and limited resources hardware such as microcontrollers and single-chip computers. DL algorithm needs high computational power resources hardware which is not the case when using limited resources hardware. So, the lite version of the DL algorithm is built to be implemented on limited resources hardware such as a microcontroller, and this version of DL is called a tiny machine learning algorithm (TinyML) [25]. Moreover, to reduce the required size and the need for high computational power, a quantized model is adopted as well as transfer learning approaches. The quantization stage is fed with floating point variables and weights of the network to produce are quantized to integer-8 fixed point variables. The quantization procedure can be described as in equation 7 [26]:

$$\hat{Q} = round\left(\frac{Q - \min(Q)}{\max(Q) - \min(Q)} \times (2^8 - 1)\right) \quad (7)$$

where $\hat{Q}$ is the quantized 8-bit integer, and Q is the floating-point variable.

C. Limited resources hardware for TinyML algorithms

Different versions and vendors of low-cost hardware have been developed in the last years to be a container for the lite version of deep learning algorithm (TinyML) such as microcontrollers, embedded systems, and Internet of Things devices [27]. Table I depends on machine learning task and data relation complexity and this reflects selecting the proper hardware for the specified task. It can be noticed that the microcontroller Arduino Nano 33 BLE Sense has the minimum resources among other boards which will be used in our work to perform the classification task of the ternary classifier. Figure 1 shows the development board from Arduino company which is used in this work to implement a fault detection classifier. This board has its accelerometer hardware which can capture vibration data and in turn, it can implement a deep learning classifier. Table I lists several types of hardware boards that are used to implement the TinyML algorithm, especially classifiers. Hardware resources of each TinyML algorithm.

Specification of the processor constrains the ability of the board to deal with complex machine learning tasks such as object detection algorithms and segmentation algorithms since they require much higher resources than less demand task such as binary classifier. Moreover, the processing time (inference time) of handling specific tasks in real-time applications depends mainly on processor computational power. Also, a complex model requires a deeper network with a large number of layers and this in turn needs a more powerful processor. On the other hand, deep learning models required RAM size to fit the trained model and the more sophisticated model reflected in more RAM size and higher data rate.

Table I: example of limited resources hardware for TinyML algorithms

Hardware	Processor	Memory	Features	Use Cases
Arduino Nano 33 BLE Sense	**ARM Cortex-M4 (64 MHz)**	**256 KB SRAM, 1 MB Flash**	**Integrated sensors (IMU, microphone, environmental sensors), BLE**	**Gesture recognition, audio classification, environmental monitoring**
Raspberry Pi Pico	ARM Cortex-M0+ (133 MHz)	264 KB SRAM, up to 16 MB Flash	Dual-core, low power consumption	Anomaly detection, simple classification
STM32 Microcontrollers	ARM Cortex-M series (varied)	Up to 320 KB SRAM, 1 MB Flash	STM32Cube.AI support	Predictive maintenance, motor control
ESP32	Dual-core Xtensa LX6 (240 MHz)	520 KB SRAM, 4 MB Flash	Wi-Fi, Bluetooth	IoT applications, lightweight ML models
NVIDIA Jetson Nano	ARM Cortex-A57 (1.43 GHz)	4 GB LPDDR4 RAM	128-core Maxwell GPU	Vision-based applications, robotics

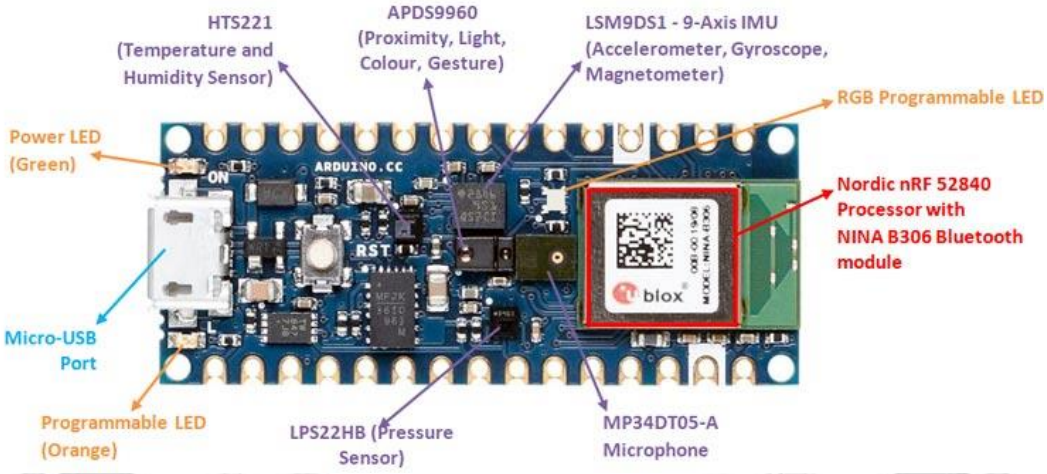


Figure 1: Arduino Nano 33 BLE sense (microcontroller)[28]

IV. Proposed system.

Vibration signals can reflect the health status of motors and especially rotated bearings. This work is designed and implemented to capture the variation of vibration patterns and classify the signal into three classes: healthy class, inner race fault, and outer race fault. Figure 2 shows the five stages of the proposed framework followed in this work.

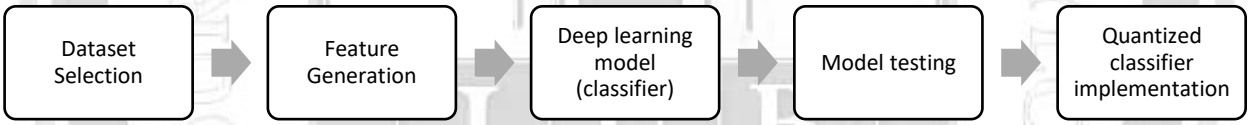


Figure 2: proposed framework

A. Dataset

A publicly available dataset is used in this work to test and evaluate the proposed system[29]. Figure 3 shows the experiment setup and data collection devices used to collect the dataset. The experiment was conducted to test three statuses of bearing (healthy, inner race face fault, and outer race face fault) under different speeds and different speed variations using a motor and electronic drive. An accelerometer with a 200000Hz sampling rate was used in the experiment to collect vibration data.

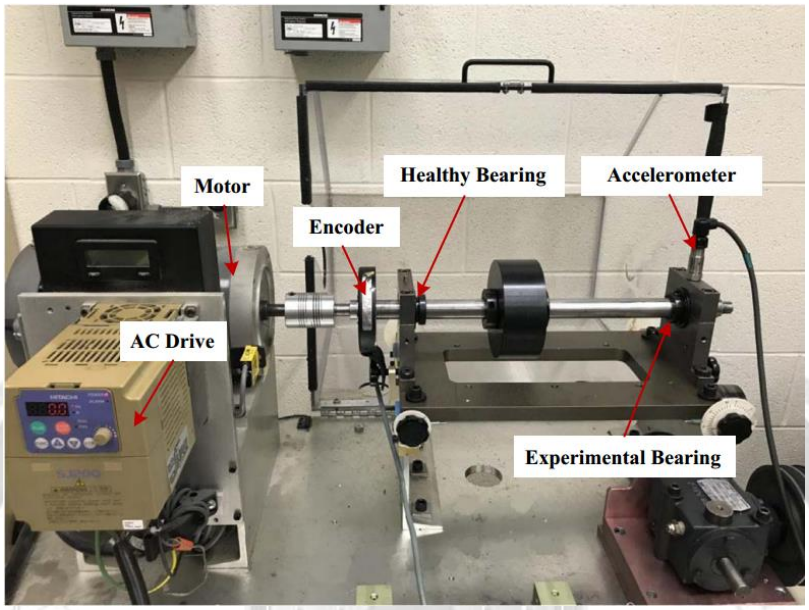
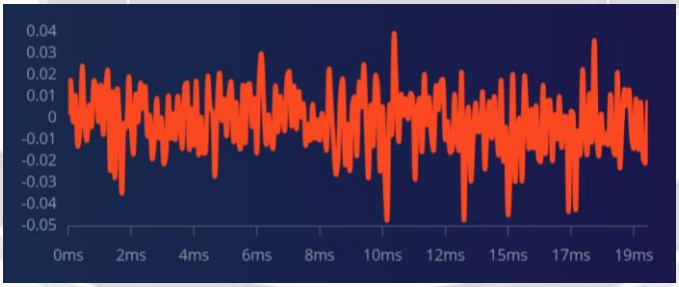
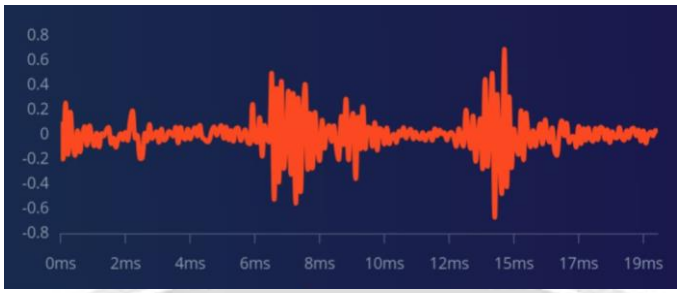


Figure 3: Dataset generation experiment [29]

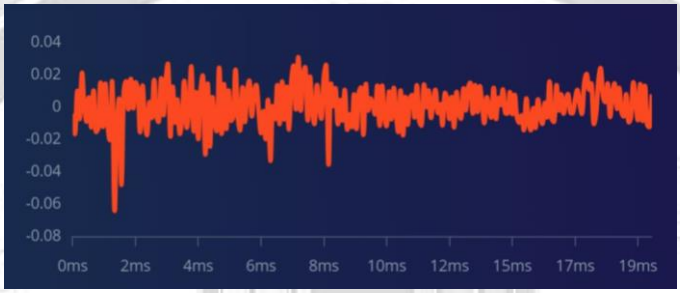
In this work, three subsets of the mentioned dataset which belong to increasing speed for the three bearing statuses are selected. Each subset includes 16 seconds of data (around 3200000 samples) and the total selected data gathers around 10 million samples for three bearing statuses. The reason behind selecting part and not the whole dataset is to reduce the machine learning network so it can fit with limited resources and hardware. Moreover, each 20 ms of data is grouped in one sample for feature generation which produces a dataset with 3000 samples. Figure 4 shows three samples of the updated dataset and each sample belongs to one of the three bearing statuses



a. Healthy



b. Inner fault



c. outer fault

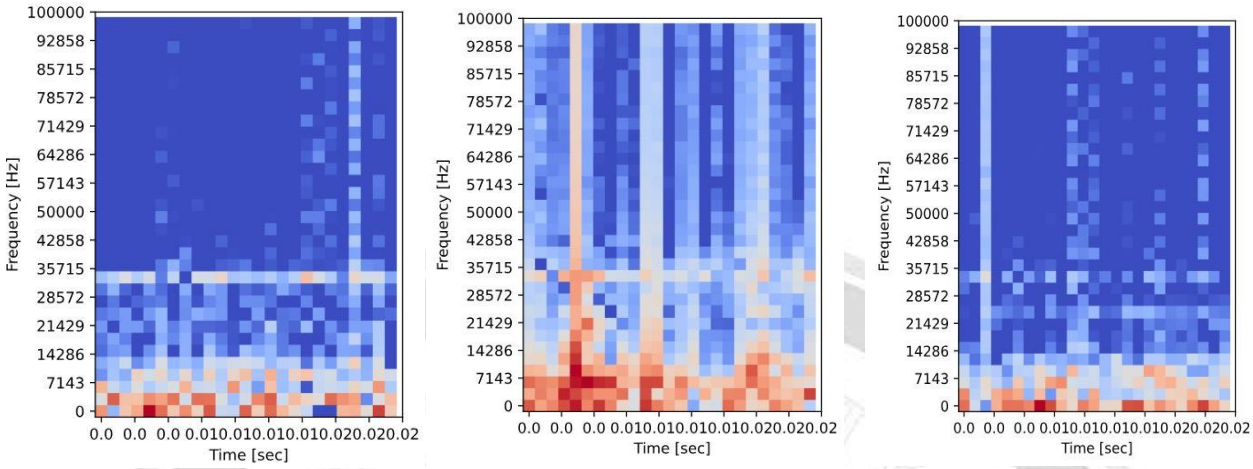
Figure 4: examples of 20ms-grouped samples

B. Features generation (Spectrogram)

The selection of the proper tool for feature generation is vital for classifier performance enhancement and this selection should target the type and complexity of the dataset. Time and frequency are crucial factors in the selected dataset and several tools for feature generation specialized in accelerometer and time-series data are tested such as wavelet, short-time Fourier transform, and spectrogram. In this work, the best results of classifier performance come up with a spectrogram tool with the best combination of its parameters as listed in Table II. Figure 5 illustrates the generated spectrogram image for the three statuses. Different sets of features are generated such as statistical, time domain, frequency domain, and texture features from spectrogram images to produce 825 features in total.

Table II Spectrogram parameters

Parameter	Value
Frame length	**8 ms**
Frame stride	**0.5 ms**
FFT length	**64**
Noise floor	**-52 dB**



a. Healthy **b. Inner fault** **c. outer fault**

Figure 5: examples of generated spectrogram

C. Deep learning network

The deep learning classifier follows the feature generation stage and uses 825 time-frequency features generated which are fed as input to the classifier stage. With the aid of an online development platform for machine learning algorithm building tool called Edge Impulse, a dimensional convolutional neural network (1DCNN) ternary classifier is building [30]. The architecture of the CNN model is illustrated in Figure 6 where features are reshaped to fit 1DCNN then followed by two layers of CNN with a proper number of filters and kernel size. Extra layers of dropout with a rate of 0.25 are used to speed up the learning process and enhance performance efficiency by reducing overfitting problems. The deep learning model ends with a flattened layer and a fully connected dense layer with SoftMax activation function as the final stage classifier. The deep learning network is trained using the well-known neural network optimizer ADAM with 100 cycles (Epochs) based on the training set (80% of the selected dataset) and in turn, the training set is also split into training or validation set (20% of the training set) [31]. The learning rate is chosen to be 0.005 and the batch size is 32. As known, CNN is time and resources consuming deep learning algorithm, so to reduce the requirements for model implementation of microcontroller hardware (Nano33 BLE Sens), the model variables are quantized using integer 8 variables. The quantized profile helps to make the deep learning model fit the hardware resources and also reduces latency to acceptable ranges.

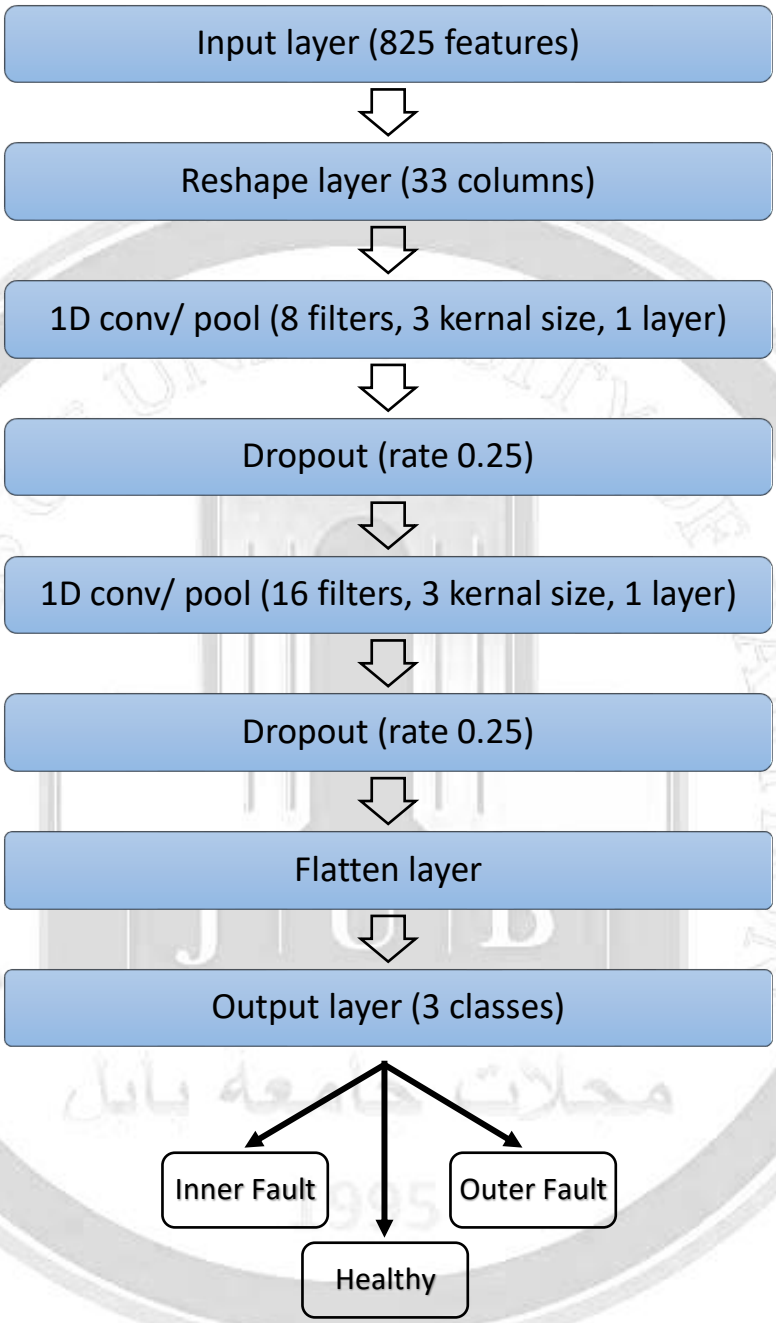


Figure 6: Deep learning network architecture

V. Results and discussion

The proposed deep learning model is tested and evaluated using the available dataset of the accelerometer data and this model has two stages: training phase and testing phase. The training phase uses 80% of the dataset which is in turn divided into two subsets (80% for training

and 20% for validation). After 100 Epochs of training, the model is settled with minimum changes in error, and the validation set is used to calculate the performance metrics including confusion matrix and accuracy and other metrics as shown in Table III and Table IV.

Table III Confusion matrix of the training phase

	HEALTHY	INNER	OUTER
HEALTHY	98.2%	0%	1.8%
INNER	0%	99.4%	0.6%
OUTER	3.4%	0%	96.6%
F1 SCORE	0.98	1.00	0.97

Table IV Training set performance metrics

Metric	Value
Accuracy	**98.1%**
Weighted average F1 score	**0.98**
Loss	**0.05**
AU-ROC	**1.00**
Weighted average of Three- classes Precision	**0.98**
Weighted average of Three- classes Recall	**0.98**

The trained CNN model is tested using the unseen 20% of the dataset based on the quantized profile model. All samples of the test set (589 samples) are classified by the three classes classifier to evaluate and test the model's performance. Table V lists the main performance metrics and shows a high accuracy of 98.64% and a weighted average F1 score of 0.99. The two evaluation metrics values are very close due to the balance of the number of samples in each class (around 1/3 of the test dataset in each class). This balance is also reflected in weighted average Recall and precision with the value of 0.99 for each of them.

Table V Test set performance metrics

Metrics	Value
Accuracy	**98.64%**
Weighted average F1 score	**0.99**
AU-ROC	**1.00**
Weighted average of Three- classes Precision	**0.99**
Weighted average of Three- classes Recall	**0.99**

The classification results that are depicted in the confusion matrix shown in Table VI, overview the high performance of the implemented deep learning model. It can be noticed easily

that the inner fault class is perfectly predicted with a 1.00 F1 score while the other two classes (healthy and outer fault) have slight errors with an F1 score of 0.99 for both classes.

Table VI Confusion matrix of the test phase

	HEALTHY	INNER	OUTER	UNCERTAIN
HEALTHY	97.7%	0%	0.6%	1.7%
INNER	0%	100%	0%	0%
OUTER	0.5%	1.0%	98.1%	0.5%
F1 SCORE	0.99	1.00	0.99	

The one-dimensional convolutional neural network and feature extraction stage representing the spectrogram creation technique has been optimized and deployed using limited resources microcontroller nano 33 BLE sense. After model deployment on the Arduino board and tested with the test dataset, it consumes the following resources as shown in Table VII

Table VII fault detection model hardware consumption quantized (integer 8)

Resources	Spectrogram	Classifier	Total
Latency	16ms	4ms	20ms
RAM	8.3KB	4.4KB	8.3KB
FLASH RAM	-	32.0KB	32.0KB
ACCURACY			98.64%

Notably, the fault detector didn't consume a large part of hardware resources (less than 5% of board resources) and this reflects the success of the implemented detector with the proposed setting of the tiny machine learning model (1DCNN). Moreover, the optimization process helps using of RAM efficiently by relocating resources between the detector components as shown in Table V. Hence, the classifier used the same resources of RAM after the Spectrogram finished its role. This relocation appears in total usage of RAM where it's not the sum of individual resources for each stage.

Deep learning algorithms have been used recently for electrical motor fault detection based on vibration-relate data, yet detection performance still needs more work to enhance it. Moreover, real-time implementation is still not addressed thoroughly from the view of inference time and required resources of hardware such as RAM and microcontroller abilities. Table VIII shows a comparison of proposed work with related work especially research papers using the same dataset.

The proposed model outperforms the existing fault detection models that using vibration-related datasets. The results listed in Table VIII show that the proposed model gains higher classification performance metrics (accuracy of 98.64 and F1 scores of 0.99). Moreover, the proposed model can be implemented practically in real time using limited resources hardware (microcontroller)

Table VIII a comparison of the proposed model with the most recent existing models

Method	Precision	Recall	F1-score	Accuracy %	Hardware Implementation
GMM [9]	0.96	0.96	0.96	95.93	No
DA_LSTM [11]	-	-	0.78	75.33	No
DBANet [12]	-	-	-	97.28	No
LiConvFormer [13]	-	-	-	97.31	No
ResNet18 [13]	-	-	-	97.56	No
TinyML DLN [14]	-	-	-	95.6	yes
Proposed	**0.99**	**0.99**	**0.99**	**98.64**	**Yes**

VI. Conclusions

Early signs of electrical faults can be captured and detected by collecting vibration data and analyzed with one of the deep learning algorithms. It is known that deep learning algorithms require very powerful hardware with high computational power because these algorithms are very demandable for computation and hardware resources. However, lite versions of deep learning algorithms have been developed recently to reduce the model requirement for resources by using quantization techniques and developing less sophisticated models with fewer layers. In this work, a one-dimensional convolutional neural network is adopted and a quantized model is implemented to train the publicly available accelerometer dataset. The dataset is generated from text experiments to simulate three types of bearing status (healthy, inner race fault, and outer race fault). The fault classifier has been implemented successfully and deployed on Arduino 33 BLE sense microcontroller using around 5% of hardware available resources with a RAM size of 14.0 kb and Flash RAM size of 38 kb. The implemented classifier works accurately with an accuracy of 98.64% and a latency of 4ms.

References

[1] Dr. Akula. V. S. S. R. Rao, Dr. S. Kulkarni, D. S. K. Bhatia, L. V Sambasivarao, and K. S. Singh, “Advanced Machine Learning Algorithms for Predictive Maintenance in Industrial Manufacturing Systems,” *South East Eur J Public Health*, pp. 716–723, Nov. 2024, doi: 10.70135/seejph.vi.2057.

[2] A. Shaala, D. Baglee, and D. Dixon, "Machine learning model for predictive maintenance of modern manufacturing assets," in *2024 29th International Conference on Automation and Computing (ICAC)*, IEEE, Aug. 2024, pp. 1–6. doi: 10.1109/ICAC61394.2024.10718768.

[3] S. Brockmann and T. Schlippe, "Optimizing Convolutional Neural Networks for Image Classification on Resource-Constrained Microcontroller Units," *Computers*, vol. 13, no. 7, p. 173, Jul. 2024, doi: 10.3390/computers13070173.

[4] Z. Guo, "Fault Diagnosis Technology for Complex Mechanical Systems Based on Deep Learning," *Transactions on Computer Science and Intelligent Systems Research*, vol. 8, pp. 15–19, Oct. 2024, doi: 10.62051/jn6ctt96.

[5] J. Liu *et al.*, "Enabling Efficient Deep Learning on MCU With Transient Redundancy Elimination," *IEEE Transactions on Computers*, vol. 73, no. 12, pp. 2649–2663, Dec. 2024, doi: 10.1109/TC.2024.3449102.

[6] P. Kumar, P. Prince, A. K. Sinha, and H. S. Kim, "Electric Vehicle Motor Fault Detection with Improved Recurrent 1D Convolutional Neural Network," *Mathematics*, vol. 12, no. 19, p. 3012, Sep. 2024, doi: 10.3390/math12193012.

[7] G. Demidova, D. Privalov, D. Semenov, D. Lukichev, Z. Liu, and A. Anuchin, "Fault Detection in Electric Drives Based on LSTM Autoencoder Model Machine Learning Approach," in *2024 IEEE 25th International Conference of Young Professionals in Electron Devices and Materials (EDM)*, IEEE, Jun. 2024, pp. 1670–1675. doi: 10.1109/EDM61683.2024.10615179.

[8] A. Li, J. Xu, and W. Lu, "Fault Detection of Electrical Equipment Using Attention Based Hybrid Deep Learning Approach," in *2024 International Conference on Data Science and Network Security (ICDSNS)*, IEEE, Jul. 2024, pp. 1–4. doi: 10.1109/ICDSNS62112.2024.10690935.

[9] A. S. Maliuk, A. E. Prosvirin, Z. Ahmad, C. H. Kim, and J.-M. Kim, "Novel Bearing Fault Diagnosis Using Gaussian Mixture Model-Based Fault Band Selection," *Sensors*, vol. 21, no. 19, p. 6579, Oct. 2021, doi: 10.3390/s21196579.

[10] Ranjit M. Gawande, "Machine Learning Approaches for Fault Detection and Diagnosis in Electrical Machines: A Comparative Study of Deep Learning and Classical Methods," *Panamerican Mathematical Journal*, vol. 34, no. 2, pp. 121–137, Jul. 2024, doi: 10.52783/pmj.v34.i2.930.

[11] D. Kumar, S. M. Ujjan, K. Dev, S. A. Khowaja, N. A. Bhatti, and T. Hussain, "Towards soft real-time fault diagnosis for edge devices in industrial IoT using deep domain adaptation

training strategy," *J Parallel Distrib Comput*, vol. 160, pp. 90–99, Feb. 2022, doi: 10.1016/J.JPDC.2021.10.005.

[12] D. Zhao, S. Liu, H. Du, L. Wang, and Z. Miao, "Deep branch attention network and extreme multi-scale entropy based single vibration signal-driven variable speed fault diagnosis scheme for rolling bearing," *Advanced Engineering Informatics*, vol. 55, p. 101844, Jan. 2023, doi: 10.1016/j.aei.2022.101844.

[13] S. Yan, H. Shao, J. Wang, X. Zheng, and B. Liu, "LiConvFormer: A lightweight fault diagnosis framework using separable multiscale convolution and broadcast self-attention," *Expert Syst Appl*, vol. 237, p. 121338, Mar. 2024, doi: 10.1016/J.ESWA.2023.121338.

[14] V.-K. Nguyen, V.-K. Tran, H. Pham, V.-M. Nguyen, H.-D. Nguyen, and C.-N. Nguyen, "A multi-microcontroller-based hardware for deploying Tiny machine learning model," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 13, no. 5, p. 5727, Oct. 2023, doi: 10.11591/ijece.v13i5.pp5727-5736.

[15] S.-M. Song, J.-H. Han, E.-J. Choi, J.-H. Park, and S.-K. Hong, "A Study on the Practical Operation of DSP for Motor Control Embedded with Fault Diagnosis and Predictive Algorithm," *The transactions of The Korean Institute of Electrical Engineers*, vol. 73, no. 1, pp. 97–105, Jan. 2024, doi: 10.5370/KIEE.2024.73.1.97.

[16] H. Issah, A. Prince Kwabena, B. Kelvin Osei, E. Afful, N. Awuah, and A. Osumanu, "Optimizing Induction Motor Fault Detection with Transfer Learning: A Comparative Analysis of Deep Learning Models," *International Journal of Innovative Science and Research Technology (IJISRT)*, pp. 398–408, Nov. 2024, doi: 10.38124/ijisrt/IJISRT24NOV003.

[17] M. Rizvi, "Leveraging Deep Learning Algorithms for Predicting Power Outages and Detecting Faults: A Review," *Adv Res*, vol. 24, no. 5, pp. 80–88, Jun. 2023, doi: 10.9734/air/2023/v24i5961.

[18] R. F. Ribeiro Junior, I. A. dos S. Areias, and G. F. Gomes, "Fault detection and diagnosis using vibration signal analysis in frequency domain for electric motors considering different real fault types," *Sensor Review*, vol. 41, no. 3, pp. 311–319, Aug. 2021, doi: 10.1108/SR-02-2021-0052.

[19] A. Brandt, *Noise and Vibration Analysis*. Wiley, 2023. doi: 10.1002/9781118962176.

[20] T. Chu, T. Nguyen, H. Yoo, and J. Wang, "A review of vibration analysis and its applications," *Heliyon*, vol. 10, no. 5, p. e26282, Mar. 2024, doi: 10.1016/j.heliyon.2024.e26282.

[21] W. Zhang *et al.*, "Enabling Accurate Detection and Localization of Bearing Faults Under Noise and Vibration," 2024. doi: 10.2139/ssrn.4710354.

[22] G. Seong and D. Kim, "An Intelligent Ball Bearing Fault Diagnosis System Using Enhanced Rotational Characteristics on Spectrogram," *Sensors*, vol. 24, no. 3, p. 776, Jan. 2024, doi: 10.3390/s24030776.

[23] S. Ghosh, M. Lin, and D. Sun, "Signal Analysis via the Stochastic Geometry of Spectrogram Level Sets," *IEEE Transactions on Signal Processing*, vol. 70, pp. 1104–1117, 2022, doi: 10.1109/TSP.2022.3153596.

[24] G. Naidu, T. Zuva, and E. M. Sibanda, "A Review of Evaluation Metrics in Machine Learning Algorithms," 2023, pp. 15–25. doi: 10.1007/978-3-031-35314-7_2.

[25] D. S. Pete Warden, *TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers*. Sebastopol, CA: O'Reilly Media, 2019.

[26] L. Zhao, Z. Dong, and K. Keutzer, "Analysis of Quantization on MLP-based Vision Models," Sep. 2022.

[27] R. Adlakha and E. Kabbar, "The Challenges of TinyML Implementation: A Literature Review," in *Proceedings: CITRENZ 2023 Conference, Auckland, 27-29 September*, Unitec ePress, Jul. 2024, pp. 160–169. doi: 10.34074/proc.240120.

[28] Aswinth Raj, "Arduino Nano 33 BLE Sense." Accessed: Feb. 01, 2025. [Online]. Available: https://circuitdigest.com/microcontroller-projects/arduino-nano-33-ble-sense-board-review-and-getting-started-guide

[29] H. Huang and N. Baddour, "Bearing vibration data collected under time-varying rotational speed conditions," *Data Brief*, vol. 21, pp. 1745–1749, Dec. 2018, doi: 10.1016/j.dib.2018.11.019.

[30] S. Hymel, "Edge Impulse: An MLOps Platform for Tiny Machine Learning," Nov. 2022.

[31] R. O. Ogundokun, R. Maskeliunas, S. Misra, and R. Damaševičius, "Improved CNN Based on Batch Normalization and Adam Optimizer," 2022, pp. 593–604. doi: 10.1007/978-3-031-10548-7_43.

ARTICLE
JOURNAL`S UNIVERSITY OF BABYLON FOR ENGINEERING SCIENCES (JUBES)
مـــجلـة جـــامعة بـــابـل للعلـــوم الهندسية
Vol. 33, No. 1. \ 2025 ISSN: 2616 - 9916

تنفيذ كفوء لمصنف بخوارزمية التعليم العميق المصغر المعتمد على اكتشاف الأخطاء المتعلقة بالاهتزاز باستخدام أجهزة ذات موارد محدودة

هلال عبد الحسين عبود الليباوي

قسم الهندسة الكهربائية، كلية الهندسة، جامعة بابل، بابل، العراق

البريد الالكتروني : Eng.hilal_al-libawy@uobabylon.edu.iq

الخلاصة

لقد ساعد التطور الكبير للأجهزة ذات الموارد المحدودة مثل المتحكمات الدقيقة بالتزامن مع تطوير خوارزميات التعلم الآلي في فتح الأبواب لبناء أجهزة ذكية في القطاعات الصناعية. الصيانة التنبؤية هي أحد هذه القطاعات التي تستخدم الأجهزة الذكية بكفاءة. يمكن لبيانات الاهتزاز مثل البيانات التي تم جمعها من مقاييس التسارع أن تلتقط بدقة أي تغيير في سلوك الآلة المتحركة بسبب التآكل الميكانيكي في الأجزاء المتحركة مثل المحامل. في هذا العمل، تم بناء مصنف شبكة عصبية ملتوية أحادية البعد للتعلم العميق (DCNN1) ككاشف للأخطاء وضبطه لتحسين أداء الكشف. يتم اختبار هذا المصنف باستخدام مجموعة بيانات اهتزاز متاحة للجمهور لثلاثة أنواع من حالة المحمل الدوار (سليم، خطأ الاطار الداخلي، وخطأ الاطار الخارجي). تم تصميم المصنف باستخدام إطار عمل صغير للتعلم الآلي، والذي يمكن تنفيذه باستخدام متحكم دقيق بموارد محدودة. تتم معالجة بيانات مقياس التسارع مسبقًا باستخدام مخطط طيف لإشارة اهتزاز لاستخراج ميزات مرتبطة بالتردد والوقت لتحسين أداء المصنف. علاوة على ذلك، يتم تحديد كمية المصنف باستخدام عدد صحيح مكون من ثماني بتات لتقليل وقت الحساب والذاكرة المطلوبة. تتولى بيئة إطار عمل TinyML بناء المصنف الثلاثي وتساعد في تنفيذ هذا المصنف على أجهزة ذات موارد محدودة. يحقق هذا المصنف الثلاثي نتائج دقيقة بدقة 98.64٪ ودرجة F1 0.99. ومع ذلك، تم تحقيق هذه النتائج الدقيقة باستخدام الحد الأدنى من الموارد لوحدة تحكم Arduino مع ذاكرة وصول عشوائي تبلغ 8.3 كيلو بايت وزمن انتقال 20 مللي ثانية.

الكلمات الدالة : TinyML؛ الاهتزاز ؛ أجهزة ذات موارد محدودة؛ CNN.