JOURNAL OF MARITIME RESEARCH

Vol XXI. No. II (2024) pp 187–195

ISSN: 1697-4840, www.jmr.unican.es

Lightweight Unsupervised Model for Anomaly Detection on Microcontroller Platforms

Le Dang Khanh[1], Nguyen Xuan Long[2,*]

ARTICLE INFO

Article history:
Received 24 Dec 2023;
in revised from 18 Jan 2023;
accepted 31 Mar 2024.

Keywords:
Maritime safety, Equipment maintenance, Anomaly Detection; Unsupervised Learning; TinyML; Autoencoder.

ABSTRACT

Maritime operations are contingent upon the efficiency of the equipment operating on the ship. This bears a considerable impact on not only the overall effectiveness of the vessel, but notably on the safety standards too. An essential cog in this machine is the establishment of effective anomaly detection, which underscores predictive maintenance of the machinery and therein, drastically reduce machine breakdown costs. To this end, this study introduces a groundbreaking approach, utilizing unsupervised learning focused on a low-cost microcontroller unit (MCU) to detect equipment abnormalities via vibration signals. The concentration of the study was towards outliers in vibration signals occurring in multiple machinery, with fans being the key point of our research. We collected an array of accelerometer data using a microcontroller, which was meticulously mounted on a fan for accurate readings. Exploring further, a robust, lightweight unsupervised learning model was developed, trained, and evaluated for precise anomaly detection. The performance of multiple autoencoder architectures with varying complexities was tested and analyzed by measuring the area under the receiver operating curve (AUC), promising active predictive maintenance. The denoising convolutional autoencoder stood out, achieving an impressive AUC of 0.993. Notably, this high-performing model only requires 41 parameters and 3.77 KB of RAM on an Arduino, proving its suitability for deployment on resource-limited edge devices. Compared to previous studies, our models deliver superior anomaly detection while using fewer parameters and computational resources. This research underscores the potential of ultra-lightweight unsupervised learning models to enable accurate and efficient predictive maintenance through on-device anomaly detection with microcontrollers.

1. Introduction.

Spotting anomalies, also known as outliers [1], is a key aspect of data analysis in many fields. The aim is to find rare or unexpected observations that stand out from the normal patterns in a dataset. These anomalies can be caused by various factors, such as errors in data collection, fraudulent activities, system malfunctions, or unforeseen events. Identifying these outliers accurately is highly beneficial, as it allows organizations to take preemptive measures against risks, improve operational efficiency, and make better-informed decisions.

With the rapid expansion of data and the increasing complexity of modern systems, traditional rule-based methods for anomaly detection have become obsolete. As a result, there has been a paradigm shift towards using advanced techniques, especially those based on machine learning, to uncover hidden patterns and detect anomalies. Machine learning algorithms excel at autonomously learning and adapting from historical data, allowing them to identify anomalies that aren't easily defined by fixed rules. This data-driven approach has shown exceptional results across various fields, including finance, cybersecurity,

[1]Faculty of Marine Engineering, Vietnam Maritime University, Haiphong 180000, Vietnam. Tel. (+84) 941982988. E-mail Address: ledangkhanh@vimarul.edu.vn.

[2]Faculty of Navigation, Vietnam Maritime University, Haiphong 180000, Vietnam. Tel. (+84) 941982988. E-mail Address: nguyenxuanlong@vimaru.edu.vn.

[*]Corresponding author: Nguyen Xuan Long. Tel. (+84) 941982988. E-mail Address: nguyenxuanlong@vimaru.edu.vn.

healthcare, and industrial processes [2].

In recent years, a surge of research has been dedicated to employing artificial intelligence for monitoring and controlling manufacturing processes. The goal is to identify measurements or patterns that deviate from the norm, signaling potential defects or anomalies. For example, Jatesiktat et al. [3] introduced an unsupervised method for detecting anomalous movements using autoencoder reconstruction error, achieving over 90% accuracy in identifying anomalies from raw data collected by a wrist-mounted inertial measurement unit (IMU). This technique could be a game-changer for automatic upper limb rehabilitation assessments. Similarly, Sadhu et al. [4] developed innovative deep Convolutional Neural Networks (CNNs) and Long Short-Term Memory Neural Networks (LSTMs) for real-time detection and classification of drone misoperations based on sensor data, demonstrating high accuracy in both simulations and practical experiments. Additionally, Jiang et al. [5] proposed a fault detection strategy for wind turbines using a denoising autoencoder (DAE) alongside multivariate sensor data, including vibration, temperature, pressure, and electrical measurements. By leveraging the DAE model to analyze this data, their approach effectively identifies and diagnoses faults in wind turbines, showcasing the power of AI in enhancing industrial processes.

TinyML offers remarkable benefits over traditional machine learning methods. By enabling on-device inference, it reduces latency and lessens reliance on network connectivity, while also addressing privacy and security concerns. TinyML's small model sizes and low computational needs make it ideal for resource-limited microcontrollers and IoT devices, resulting in cost savings and enhanced efficiency in real-time decision-making. For instance, Vitolo, Paola, et al. [6] introduced a low-power anomaly detection and classification system for predictive maintenance based on vibration monitoring. Their approach uses a Convolutional Auto-Encoder (AE) and a hybrid hardware/software Convolutional Neural Network (CNN) to achieve top-tier performance and accuracy. Similarly, Sampaio et al. [7] demonstrated a methodology for predicting industrial equipment failure times using TinyML techniques, specifically an Artificial Neural Network (ANN) based on vibration data, showcasing the potential of efficient predictive maintenance systems in resource-constrained settings. Furthermore, M. Lord and A. Kaplan et al. [8] explored the use of machine learning on an embedded microcontroller to detect mechanical anomalies in a top-load washing machine. They employed autoencoder and variational autoencoder neural network models, implemented with TinyML on an Arduino Nano microcontroller, to accurately detect unbalanced laundry loads in real time.

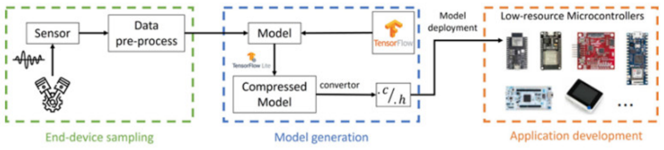
The goal of this research was to design a lightweight unsupervised learning model for real-time anomaly detection in various signal types, capable of running on a low-cost microcontroller. A test setup featured a fan with and without an anomaly component, equipped with an Arduino Nano 33 BLE microcontroller mounted on the motor. This configuration enabled the collection of vibration data under both normal and abnormal load conditions. Different unsupervised neural network models, varying in architecture and size, were trained on this data to detect anomalies. The key objective was to find a model that delivered high accuracy while maintaining a compact memory footprint, making it suitable for microcontroller deployment. Remarkably, the proposed models outperformed prior methods, such as those by Lord and Kaplan et al. [8], in terms of detection performance and resource efficiency. The top-performing model was selected for deployment on the edge device, demonstrating the feasibility of real-time predictive maintenance through on-device machine learning within constrained resources.

2. Proposed Method.

2.1. Experimental Setup.

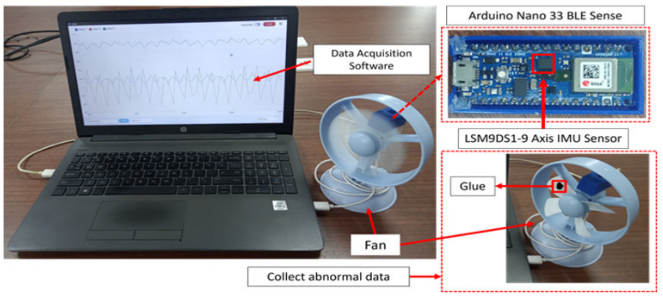
The proposed method consists of three key steps: end-device sampling, model generation, and application development. As illustrated in Figure 1, a sensor captures data from the equipment, which then undergoes preprocessing techniques like data normalization, missing value imputation, and feature selection to ensure quality [9]. To achieve high accuracy in anomaly identification, an unsupervised neural network is constructed using the TensorFlow Framework and trained with the processed data. It's important to note that smaller model sizes result in faster inference times [10].

Figure 1: The deployment of the proposed TinyML-based system.



Source: Authors.

Figure 2: Experimental setup for collecting data.


Data Acquisition Software
Arduino Nano 33 BLE Sense
LSM9DS1-9 Axis IMU Sensor
Glue
Fan
Collect abnormal data

Source: Authors.

Post-training, the model is optimized with TensorFlow Lite, an advanced inference framework designed for executing machine learning algorithms on resource-constrained devices [11]. This optimization involves techniques like parameter quantization and pruning to minimize model size and inference time. The fewer data and training epochs required, the lower the development cost. Finally, the optimized model is converted into

C++ and header files for deployment on a low-cost microcontroller. The comprehensive and open embedded ecosystem allows for easy customization of different applications, showcasing the method's versatility and efficiency.

In this study, data collection was carried out using a setup involving a compact fan with an anomaly component attached to one of its blades. The Arduino Nano 33 BLE microcontroller, mounted on the fan's motor, recorded vibration measurements from the accelerometer. Data was collected over a 10-minute duration for each experiment, repeated four times with different fan directions to create a dataset of normal operation. Anomalous data was generated by attaching a small stick of glue to one blade and recording for another 10 minutes, as illustrated in Figure 2.

We chose the Arduino Nano 33 BLE Sense as our target microcontroller unit (MCU). This board comes equipped with an accelerometer, a gyroscope, and a magnetometer, each offering 3-axis resolution. The accelerometer operates at a sample rate of 119.00 Hz, meaning it collects a sample every 8.4 milliseconds. This high sampling rate ensures precise vibration data capture, crucial for effective anomaly detection.

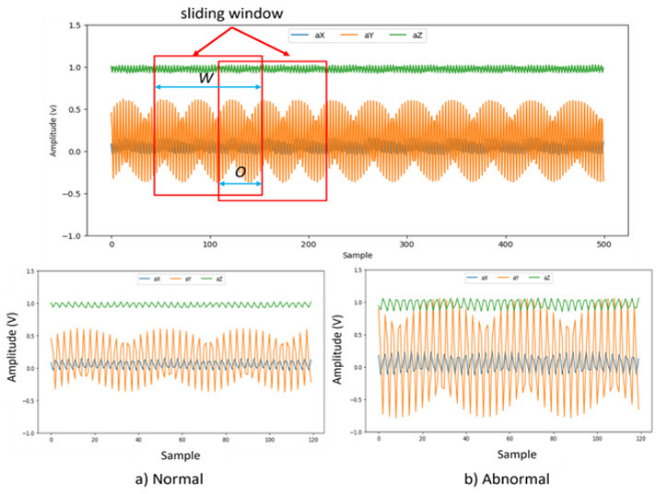
2.2. Dataset preparation.

Normal data was collected by observing a balanced fan operating at low and medium speeds. Abnormal data was gathered by adding weight to the fan blades. After the collected data had been carefully cleaned and prepared, the normal data was divided into three sets: 80% for training, 10% for validation, and 10% for testing. The testing data set consisted of a combination of normal and abnormal data points that had been selected. Figure 3a displays 3-axis accelerometer data measured from the fan running normally. Figure 3b shows 3-axis accelerometer data that was recorded for an attached hot stick glue, indicating abnormal fan operation.

In the data preprocessing phase, we employed a sliding window approach to segment the accelerometer readings into fixed-size frames. The window length (W) was set to 119 samples, aligning with the accelerometer's sampling frequency. This configuration ensures each frame encapsulates one complete cycle of readings. As depicted in Figure 3, the sliding window method unveils temporal patterns in the data that might not be apparent from individual samples alone. By dividing the sensor outputs into overlapping segments (i.e., O = 0%, 25%, 50%, 75%), with O is overlap area scale, we can examine short sequences of adjacent readings and their dynamics over time. This approach provides crucial context for distinguishing between normal and anomalous operations based on subtle deviations in vibration profiles. Moreover, using overlapping windows effectively increases the training dataset size by incorporating all segments of the original readings, enhancing the model's learning capability.

More importantly, the frame-based representation enhances our ability to extract features pertinent to anomaly detection. Neural networks can learn characteristic patterns within each window directly from the data, eliminating the need for manual feature engineering. This data-driven approach allows for the

Figure 3: The example for normal and abnormal samples.

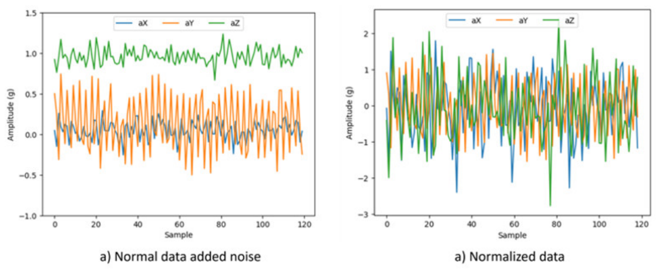


Source: Authors.

discovery of potentially significant but previously unknown indicators of anomalies. Compared to analyzing individual samples, applying sliding windows to the accelerometer outputs enables better capture of timeseries behavior and uncovers subtle differences that may indicate underlying mechanical faults or load imbalances. Overall, this technique amplifies the predictive capabilities of our models, making them more effective in identifying anomalies.

To increase the robustness of our models, we augmented the training data by introducing random noise. This simulates real-world sensor noise that may be present during deployment. Gaussian white noise scaled between 0 and 0.1 was added to each sample to represent disturbances. The Gaussian noise was generated.

Figure 4: The example for normal and abnormal samples after adding noise.



Source: Authors.

using the equation:

$$N(0, \sigma) = \sigma \times randn(0.1) \quad (1)$$

where σ is the standard deviation set to 0.1, and produces a random normal value. By incorporating noise in this controlled manner, the models were exposed to a diverse set of patterns during training to improve flexibility while maintaining the underlying data structure. This also helped address overfitting and enforced a degree of noise resilience to more closely mimic practical conditions. Overall, this noise augmentation enhanced the robustness and generalization of the anomaly detection models.

To ensure consistency across the different features in our dataset, we applied min-max scaling to normalize the data in each dimension. Min-max scaling rescales the data values to fall within a specific range, which is often between 0 and 1. This process facilitates modeling and training by bringing all features onto a comparable numeric scale.

$$Z = \frac{X - \mu}{\sigma} \tag{2}$$

where Z is normalized data, $X \in R^{m \times n}$ is input, in this work m is 119 and n is 3, μ is mean value and σ is the standard deviation. By doing so, we create a standardized representation of the data as depicted in Figure 4 that facilitates further analysis and modeling.

2.3. Proposed Methods.

2.3.1. Autoencoder.

An Autoencoder (AE) is a type of neural network that is specifically designed to learn representations of input data with a lower dimensionality. Figure 5 illustrates the structure of an autoencoder, which comprises two components: an encoder and a decoder. The encoder takes the input data and transforms it into a compressed, low-dimensional internal representation (also known as a bottleneck). Conversely, the decoder takes this low-dimensional representation and reconstructs it back into output data. It is important to note that the number of neurons in the output layer of the autoencoder must match the number in the input layer. In most cases, the output is referred to as the reconstruction because the autoencoder aims to reconstruct its input as the output.

In particular, autoencoders learn a map from the input to itself through a pair of encoding and decoding phases as shown in Equation 3

$$\bar{X} = D(E(X)) \tag{3}$$

where X ∈ Rm×n is the input data, E is an encoding map from the input data to the hidden layer, D is a decoding map from the hidden layer to the output layer, and X¯ is the recovered version of the input data. The idea is to train E and D to minimize the difference between X and X¯.

An autoencoder can be viewed as a solution to the following optimization problem:

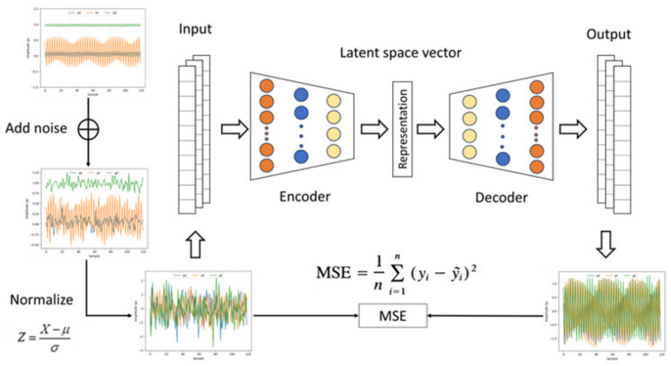
$$\min_{E,D} \sum L_{rec}(X, X) \tag{4}$$

A common choice for the reconstruction error is the mean-squared error (MSE):

$$L_{rec}(X, \bar{X}) = L_{MSE}(X, \bar{X}) = \left\|X - \bar{X}\right\|^2 \tag{5}$$

An autoencoder can consist of multiple hidden layers, referred to as a stacked autoencoder or deep autoencoder. Increasing the number of hidden layers can enhance its learning capabilities. However, there is a drawback to making the autoencoder overly powerful. A highly complex autoencoder may excellently reconstruct the exact training data but struggle to reconstruct data that it has not encountered before. In such cases, the model is essentially memorizing rather than truly learning. To address this issue, a simple technique is to restrict the size of the internal representation layers, resulting in an incomplete network. This approach compels the model to focus on learning important features from the input data, enabling it to effectively reconstruct unseen data.

Figure 5: Architecture of Autoencoder Model.


Input
Latent space vector
Output
Add noise
Representation
Encoder
Decoder
Normalize
$z = \frac{X-\mu}{\sigma}$
$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2$
MSE

Source: Authors.

2.3.2. Denoising Autoencoder.

A Denosing Autoencoder (DAE) is trained to reconstruct the unperturbed data sample from an input sample that has been subjected to noise [12, 13]. This results in more robust and perturbation-invariant representations. The most commonly used noise is additive Gaussian noise, i.e.: $\tilde{X} = X + \varepsilon$, with $\varepsilon \sim N(0, \sigma^2)$, for a small value. Thus X^{-} in Equation 3 becomes $\bar{X} = D(E(\tilde{X}))$.

Our first Denoising Autoencoder model, which uses only a Dense layer, consists of six layers. The encoder part of the model comprises Dense layers with 8, and 4 neurons, while the decoder part mirrors this architecture in reverse. The latent space, which serves as an intermediate representation of the data, has a number of units is 2. Meanwhile, the second model uses only one Conv1D layer in both the encoder and decoder for Convolutional AE, which made the number of parameters decrease significantly, as described in Table 1.

The input to our autoencoder model is the noisy accelerometer data (x, y, z). The encoder component learns to compress the three-dimensional input data into the lower-dimensional latent space, while the decoder component learns to reconstruct the original input (x, y, z) from the latent space. To facilitate efficient computation, we chose the Rectified Linear Unit (ReLU) activation function for every layer, except the last layer of the decoder. ReLU is known for its computational speed compared to other activation functions. We opted for the Adam optimizer and used the mean square error (MSE) as the loss function.

Table 1: Detailed setting and parameters of Conv1D Autoencoder.

Layer	Output shape	Layer setting	Params
Input	119 × 3		0
Conv1D 1	119 × 3	Conv1D (3×3, filters=2, stride=1, padding = 'same', ReLU)	20
Conv1D 2	119 × 3	Conv1D (3×3, filters=3, stride=1, padding = 'same', ReLU)	21
Total params: 41			
Trainable params: 41			
Non-trainable params: 0			

Source: Author.

2.3.3. *Variational Autoencoder.*

A Variational Autoencoder [14, 15] assumes a latent variable model where a latent variable z causes the observation x, facilitating a lower bound of the probability of a data sample with

$$\log p(x) \geq -D_{KL}(q(z|x)\|p(z)) + \mathrm{E}_{q(z|x)}[\log p(x|z)] \tag{6}$$

which is often termed the Evidence Lower Bound (ELBO). Here p(z) is the prior distribution of the latent variable, q(z|x) is the approximate inference model, and p(x|z) is the generative model. By maximizing the ELBO, the probability distribution approximates the true data distribution and enables a probability estimate for a data sample. VAEs parameterize q(z|x) and p(x|z) by neural networks and for p(z) and p(x|z) assume diagonal Gaussian distributions:

$$\begin{aligned} q(z|x) &= N(z; f_{\mu,\theta_1}(x), f_{\sigma,\theta_2}(x)^2), \\ p(x|z) &= N(x; g_{\mu,\gamma}(z), c(z)^2) \end{aligned} \tag{7}$$

where fμ, fσ, and gμ are neural networks with parameters θ1, θ2 and γ respectively and c is often chosen as constant. In analogy to AEs f is called the encoder and g is called the decoder. The often-used formulation for VAE training is:

$$\min_{\theta_1,\theta_2,\gamma} \sum_x D_{KL}(N(f_{\mu,\theta_1}(x), f_{\sigma,\theta_2}(x)^2)\|N(0,1)) + L_{rec}(x, g_{\mu,\gamma}(\tilde{z})) \tag{8}$$

with $\tilde{z}$ being sampled from $N(f_{\mu,\theta_1}(x), f_{\sigma,\theta_2}(x)^2)$ using the reparametrization trick [15, 16], and MSE is chosen for L_{rec}.

Our VAE model has two hidden layers in both the encoder and decoder with a number of units equal to 8 and 4, the mean vector, standard deviation vector, and sampled la-tent space were created from a Dense layer having 2 neurons.

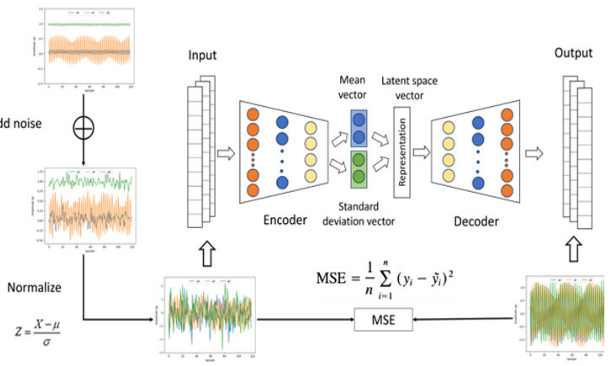
2.3.4. *Attention Autoencoder.*

The Attention Autoencoder (AE) is a variant of the traditional autoencoder architecture that incorporates attention mechanisms [16]. Attention mechanisms have been widely successful in natural language processing tasks and computer vision, and their integration into autoencoders aims to enhance the model's ability to capture and focus on important features in the input data.

The attention mechanism, when introduced into the autoencoder, allows the model to assign different importance weights to different parts of the input sequence during the encoding and decoding processes.

The input sequence is transformed into three sets of vectors: query, key, and value. These transformations allow the model to acquire meaningful representations tailored to the specific task being addressed. Subsequently, attention scores are computed by measuring the likeness between each query and key pair, typically through dot product operations. This computation captures the nuanced relationships between elements within the sequence.

Figure 6: Architecture of Variational Autoencoder.


Input
Output
Add noise
Mean vector
Latent space vector
Representation
Standard deviation vector
Encoder
Decoder
Normalize
$Z = \frac{X-\mu}{\sigma}$
$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2$
MSE

Source: Authors.

Following this, attention scores are normalized across the sequence dimension using a softmax function, resulting in attention weights. These weights indicate the relative importance of each value vector in relation to its corresponding query, enabling the model to focus on pertinent segments of the input sequence. Finally, the attention weights are employed to weigh the value vectors, resulting in a weighted summation that aggregates information across the input sequence. The output of the attention layer was computed as in Equation 9.

$$O = softmax\left(Q \cdot K^T\right) \cdot V \tag{9}$$

where O is the output, Q is the query matrix, K is the key matrix and V is the value matrix.

Our architecture incorporates a single Conv1D layer in both the encoder and decoder components. The attention mechanism is utilized to form the latent space in the model. The model is compiled with the Adam optimizer and mean squared error (MSE) loss, which is common for autoencoder-based reconstruction tasks.

Table 2: Detailed setting and parameters of Variational Autoencoder.

Layer	Output shape	Layer setting	Params
Input	119 × 3		0
Dense	119 × 8	ReLU	32
Dense	119 × 4	ReLU	36
Dense	119 × 2	ReLU	10
Lambda	119 × 2	ReLU	0
Dense	119 × 4	ReLU	12
Dense	119 × 8	ReLU	40
Dense	119 × 3	None	27
Total params: 157			
Trainable params: 157			
Non-trainable params: 0			

Source: Authors.

2.3.5. Transformer Autoencoder.

In a traditional autoencoder, the encoder compresses the input data into a fixed-size representation, and the decoder reconstructs the input from this representation. Transformers, on the other hand, use self-attention mechanisms to weigh different parts of the input sequence dynamically, allowing the model to focus on relevant information at each step of processing.

The encoder comprises either a single or multiple attention - encoder blocks. Each attention encoder block applies self-attention to the input sequence, followed by residual connections. After processing through the attention blocks, the output is passed through a series of dense layers (multi-layer perceptron or MLP) to further refine the representations. The final output is generated by a dense layer with the same dimensionality as the input sequence. Our encoder architecture comprises three attention-encoder blocks, while the decoder consists of three Dense layers with ReLU activation, followed by a final layer that is a Dense layer without activation.

2.4. Evaluation.

For the evaluation of our anomaly detection model, prediction accuracy is not enough, because a model that was trained only with normal data and has high accuracy may not perform well when classifying anomalous data. Such a model may misclassify normal data as anomalous (false positive - FP) or anomalous data as normal (false negative - FN). Therefore, Area Under Curve (AUC) provides better metrics than accuracy for anomaly detection applications. AUC is computed from Sensitivity which has another name for True Positive Rate (TPR), which is computed by taking the sum of True Positive (TP) and True Negative (TN), and 1-Specificity which has another name for False Positive Rate (FPR).

$$Accuracy = \frac{(TP + TN)}{(TP + FP + TN + FN)} \qquad (10)$$

Sensitivity tells us what proportion of the positive class got correctly classified FPR tells us what proportion of the negative class got incorrectly classified by the classifier.

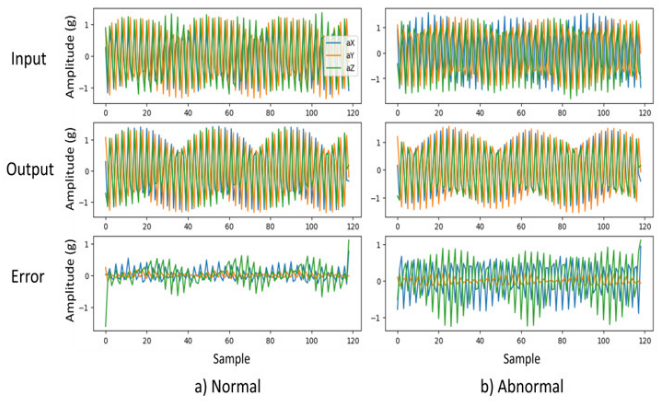
$$Sensitivity(TPR) = \frac{TP}{(TP + FN)} \qquad (11)$$

$$FPR = \frac{FP}{(TN + FP)} \qquad (12)$$

3. Results and Discussion

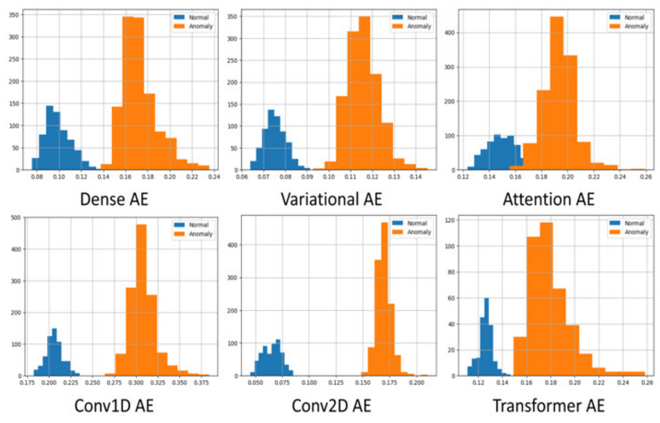
The results of the above-discussed model are presented in this section. Every model was trained with 100 epochs with a batch size equal to 16, the optimizer was set to Adam, and the learning rate was 0.01. The model was trained on a PC Intel(R) Core(TM) i3-1005G1 CPU @ 1.20GHz 1.19 GHz, RAM 20 GB, 64-bit Win 11. Figure 7 shows the reconstruction and reconstruction error of one example. As expected, the reconstruction error of abnormal data is higher than the normal one. The reconstruction of abnormal data wasn't reconstructed back to the same as input, as shown in Figure 7b.

Figure 7: Reconstruction data and reconstruction error.



Source: Authors.

Figure 8: Histogram of reconstruction error.

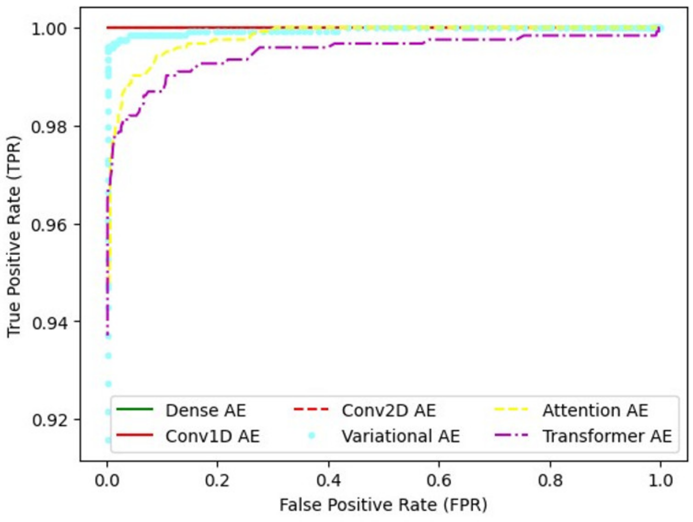


Source: Authors.

In Figure 8, the histogram of reconstruction errors for the six models is illustrated. These models were trained on a dataset containing 75% overlap between normal and abnormal samples. Notably, the autoencoder models employing Conv1D and

Conv2D architectures exhibit superior performance, showcasing a clear distinction between the distribution of normal and abnormal data. On the other hand, the remaining models, including the Attention model, display a closer proximity between the distributions of normal and abnormal data. In fact, in the Attention model, there is an observable overlap between the two distributions.

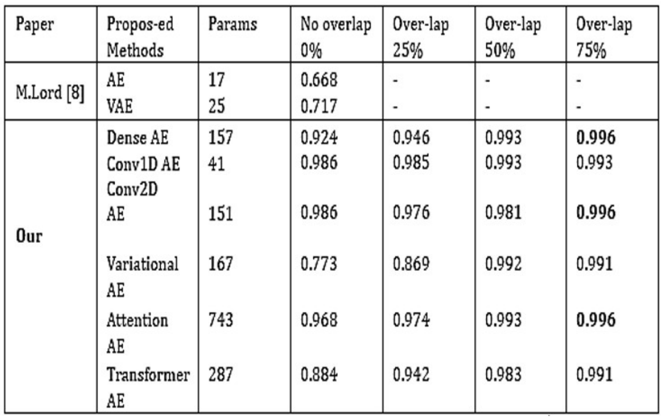
Figure 9: Comparing the Receiver Operating Characteristic Curve (ROC) of 6 methods: Dense AE, Variational Autoencoder (VAE), Conv1D AE, Conv2D AE, Attention AE and Transformer AE.



Source: Authors.

Table 3 presents a comparison of AUC scores among various anomaly detection models applied to fan vibration data This comparison includes the method introduced by Lord and Kaplan et al. [8] and our proposed models, evaluated across different degrees of overlap between normal and abnormal data. Despite

Table 3: Comparing Area Under Curve (AUC) of each method.



Paper	Propos-ed Methods	Params	No overlap 0%	Over-lap 25%	Over-lap 50%	Over-lap 75%
M.Lord [8]	AE	17	0.668	-	-	-
	VAE	25	0.717	-	-	-
Our	Dense AE	157	0.924	0.946	0.993	**0.996**
	Conv1D AE	41	0.986	0.985	0.993	0.993
	Conv2D AE	151	0.986	0.976	0.981	**0.996**
	Variational AE	167	0.773	0.869	0.992	0.991
	Attention AE	743	0.968	0.974	0.993	**0.996**
	Transformer AE	287	0.884	0.942	0.983	0.991

Source: Authors.

the smaller parameter count in the previous method, which considers each sample point as input, it did not yield satisfactory results in our study. The most compact model, Conv1D AE, containing 41 parameters, achieved the best AUC score across all models when trained on data with a 75% overlap. However, some models demonstrated suitable AUC scores even with a reduced overlap space.

Conclusions.

In this study, we developed and evaluated several lightweight unsupervised learning models for vibration-based anomaly detection on a microcontroller. A key goal was to create models that achieved high accuracy while maintaining a compact size suitable for resource-constrained edge devices.

Our results demonstrate that convolutional autoencoders are well-suited for this application, outperforming previous architectures in terms of anomaly detection performance. Specifically, the Conv1D autoencoder achieved the best AUC score of 0.993 when trained on data with a 75% overlap between normal and anomalous classes. Crucially, it accomplished this using only 41 parameters – quite suitable for low-memory edge devices.

By framing the accelerometer data with overlapping sliding windows, our models were able to better learn characteristic patterns over time compared to considering each sample independently. This enhanced their ability to distinguish between normal and anomalous vibration profiles. Noise augmentation during training also improved the robustness and generalization of the models.

Real-time testing validated the suitability of the Dense autoencoder for on-device predictive maintenance applications. It detected anomalies within about 21 ms using just 3.77KB of RAM on the Arduino microcontroller. Overall memory usage of our models was consistent with the low-resource constraints of edge devices.

While this initial study focused on a single fan motor, the approach shows promise for wider industrial machinery. Future work involves collecting data from diverse equipment, optimizing model compression further, and evaluating performance under variable operating conditions over longer periods. Nonetheless, these findings indicate convolutional autoencoders enable accurate and efficient anomaly detection for predictive maintenance through tiny machine learning at the network edge.

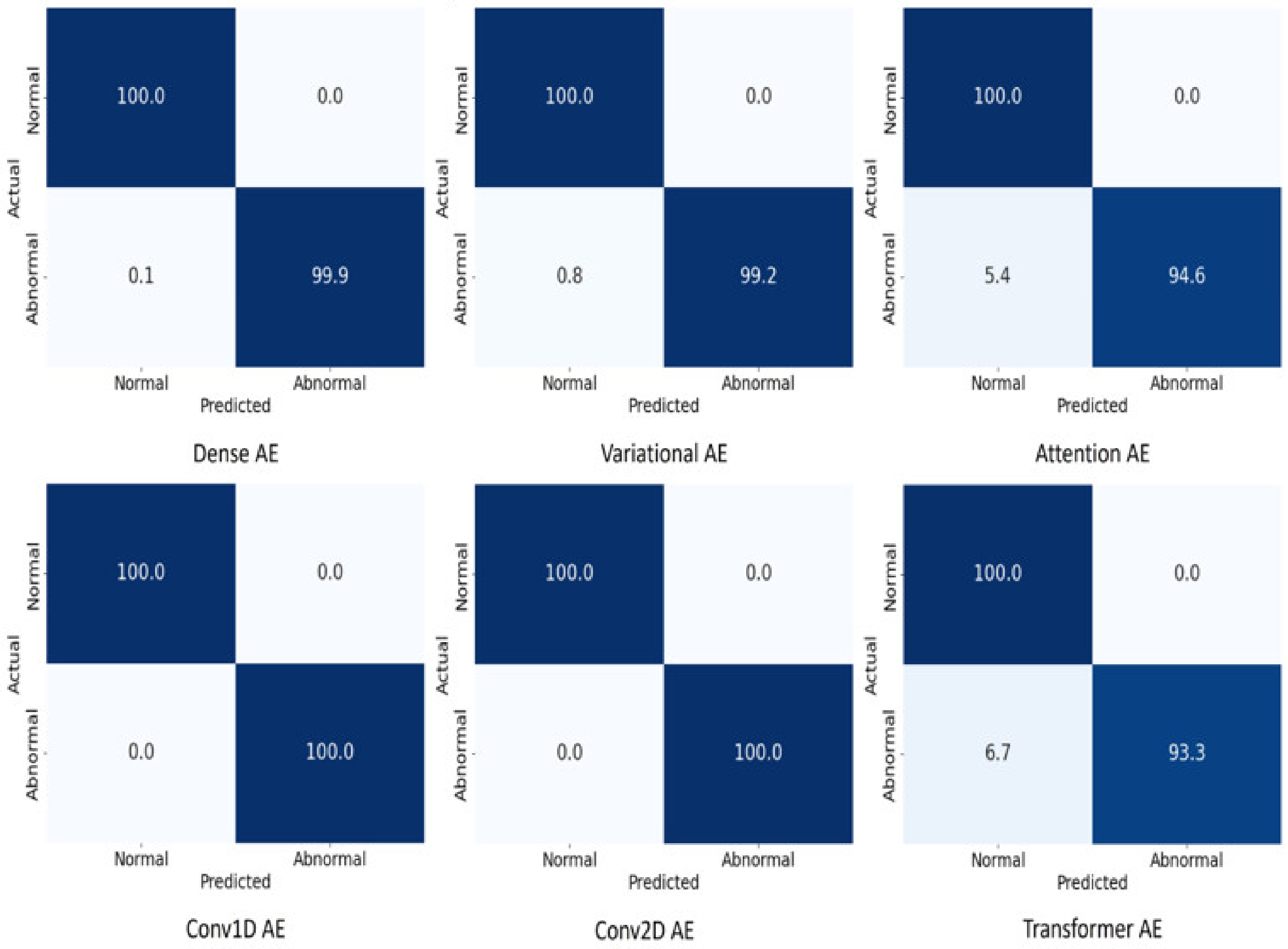
References.

[1] Sonali B Wankhede. “Anomaly detection using machine learning techniques”. In: 2019 IEEE 5th International Conference for Convergence in Technology (I2CT). IEEE. 2019, pp. 1–3.

[2] Ramon Sanchez-Iborra and Antonio F Skarmeta. “Tinyml-enabled frugal smart objects: Challenges and opportunities”. In: IEEE Circuits and Systems Magazine 20.3 (2020), pp. 4–18.

[3] Prayook Jatesiktat and Wei Tech Ang. “Unsupervised anomalous movement detection using autoencoder reconstruction error”. In: Proceedings of AUN/SEED-NET Regional Conference on Computer and Information Engineering. 2016.

Figure 10: Confussion charts for six proposed models.



Source: Authors.

[4] Vidyasagar Sadhu, Saman Zonouz, and Dario Pompili. "On-board deep-learning-based unmanned aerial vehicle fault cause detection and identification". In: 2020 ieee international conference on robotics and automation (icra). IEEE. 2020, pp. 5255–5261.

[5] Guoqian Jiang et al. "Wind turbine fault detection using a denoising autoencoder with temporal information". In: IEEE/Asme transactions on mechatronics 23.1 (2017), pp. 89–100.

[6] Paola Vitolo et al. "Low-power detection and classification for in-sensor predictive maintenance based on vibration monitoring". In: IEEE Sensors Journal 22.7 (2022), pp. 6942–6951.

[7] Gustavo Scalabrini Sampaio et al. "Prediction of motor failure time using an artificial neural network". In: Sensors 19.19 (2019), p. 4342.

[8] Mansoureh Lord and Adam Kaplan. "Mechanical Anomaly Detection on an Embedded Microcontroller". In: 2021 International Conference on Computational Science and Computational Intelligence (CSCI). IEEE. 2021, pp. 562–568.

[9] Salvador Garc´ıa et al. "Big data preprocessing: methods and prospects". In: Big Data Analytics 1.1 (2016), pp. 1–22.

[10] Alessandro Montanari et al. "ePerceptive: energy reactive embedded intelligence for batteryless sensors". In: Proceedings of the 18th Conference on Embedded Networked Sensor Systems. 2020, pp. 382– 394.

[11] Robert David et al. "Tensorflow lite micro: Embedded machine learning for tinyml systems". In: Proceedings of Machine Learning and Systems 3 (2021), pp. 800–811.

[12] Pascal Vincent et al. "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion." In: Journal of machine learning research 11.12 (2010).

[13] David Zimmerer et al. "Context-encoding Variational Autoencoder for Unsupervised Anomaly Detection– Short Paper". In: arXiv preprint arXiv:1907.12258 (2019).

[14] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes". In: arXiv preprint arXiv:1312.6114 (2013).

[15] Danilo Jimenez Rezende, Shakir Mohamed, and Daan

Wierstra. “Stochastic backpropagation and approximate inference in deep generative models”. In: International conference on machine learning. PMLR. 2014, pp. 1278–1286.

[16] Ariyo Oluwasanmi et al. “Attention autoencoder for generative latent representational learning in anomaly detection”. In: Sensors 22.1 (2021), p. 123.