big data and cognitive computing

MDPI

Article

Edge vs. Cloud: Empirical Insights into Data-Driven Condition Monitoring

Chikumbutso Christopher Walani and Wesley Doorsamy *

School of Electronic and Electrical Engineering, University of Leeds, Leeds LS2 9JT, UK
* Correspondence: w.d oorsamy@leeds.ac.uk

Abstract: This study evaluates edge and cloud computing paradigms in the context of data-driven condition monitoring of rotating electrical machines. Two well-known platforms, the Raspberry Pi and Amazon Web Services Elastic Compute Cloud, are used to compare and contrast these two computing paradigms in terms of different metrics associated with their application suitability. The tested induction machine fault diagnosis models are developed using popular algorithms, namely support vector machines, k-nearest neighbours, and decision trees. The findings reveal that while the cloud platform offers superior computational and memory resources, making it more suitable for complex machine learning tasks, it also incurs higher costs and latency. On the other hand, the edge platform excels in real-time processing and reduces network data burden, but its computational and memory resources are found to be a limitation with certain tasks. The study provides both quantitative and qualitative insights into the trade-offs involved in selecting the most suitable computing approach for condition monitoring applications. Although the scope of the empirical study is primarily limited to factors such as computational efficiency, scalability, and resource utilisation, particularly in the context of specific machine learning models, this paper offers broader discussion and future research directions of other key issues, including latency, network variability, and energy consumption.

Keywords: condition monitoring; cloud computing; edge computing; induction machine; machine learning

Academic Editors: Muhammad Kazim and Fabrizio Messina

Received: 25 January 2025
Revised: 30 April 2025
Accepted: 6 May 2025
Published: 8 May 2025

Citation: Walani, C.C.; Doorsamy, W. Edge vs. Cloud: Empirical Insights into Data-Driven Condition Monitoring. *Big Data Cogn. Comput.* **2025**, *9*, 121. https://doi.org/10.3390/bdcc9050121

1. Introduction

Electrical machines have been integral to numerous industrial processes for decades. Unexpected failures of these machines can often result in costly downtime, repairs, and even safety hazards [1]. To prevent such unforeseen breakdowns, many industries have adopted condition-based maintenance, which involves continuous monitoring of machines to identify incipient faults and issues [2–4]. Advances in sensing, communications, and computing have significantly transformed condition monitoring, offering a data-driven approach as opposed to traditional model-based methods. Data-driven condition monitoring (DDCM) has thus attracted interest from across different sectors, as they offer more flexibility, among several other benefits, than conventional approaches [5–7]. With data-driven condition monitoring of rotating electrical machines, the data processing typically involves the use of a machine learning model which can be deployed at, or in close proximity to, the machine via edge computing, or remotely via cloud computing. The choice of which computing paradigm is most suitable depends on several factors related to the application. Although there is some literature that compares these two paradigms for machine learning [8–10], there are very few studies that focus on assessing their suitability in the context

of DDCM. This paper addresses this gap by providing an empirical comparison of edge and cloud platforms for condition monitoring. The presented comparisons are based on the practical needs and challenges of modern condition monitoring in industrial settings [8].

Induction motors are widely used across industrial sectors and are crucial for many production processes. Due to their critical role, faults in components such as bearings, rotors, and stators can cause significant operational disruptions if not promptly addressed. Among these faults, broken rotor bars are relatively common and particularly challenging to detect at an early stage. This type of fault accounts for approximately 9% of induction motor failures [11]. Advanced online diagnostic techniques may be employed to detect these faults as early as possible to ensure reliable operation and minimise costly downtime. By analysing sensor data, such as current and vibration signals, machine learning (ML) algorithms can accurately predict motor health and detect these faults at an early stage. Therefore, detection of broken rotor bars using ML serves as an ideal example of modern condition monitoring. Given the widespread use of induction motors and the recent proliferation of DDCM, this application provides a valuable case study for understanding how edge and cloud computing paradigms can meet the demands of modern industrial condition monitoring.

This study conducts a comparative evaluation of edge and cloud computing approaches in DDCM for electrical machines. By investigating the trade-offs between these computing paradigms for a particular case study, the research provides insights into their practical application. The focus is on assessing edge and cloud computing across key metrics such as computational efficiency, scalability, and resource utilisation, when deploying widely used machine learning models—i.e., Support Vector Machines (SVM), Decision Trees (DT), and K-Nearest Neighbours (KNN), in DDCM. An overview of the study focus is depicted in Figure 1. This paper is structured as follows: Section 2 provides a background to condition monitoring, edge computing, and cloud computing, and motivates the study aims. Section 3 details the methodology, including the experimental setup, dataset, and evaluation metrics. Section 4 discusses the results and analysis, focusing on comparative performance and resource utilisation across edge and cloud environments. This section also discusses the practical implications of the findings and offers some recommendations. Finally, Section 5 summarises key insights and opportunities for future research.

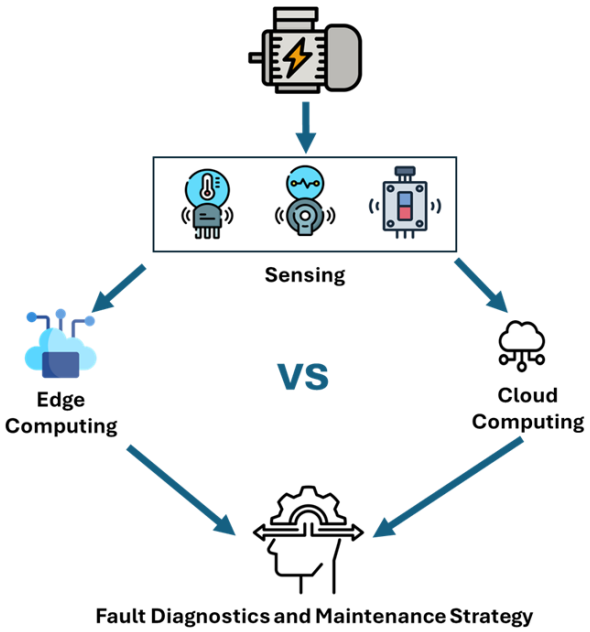

Figure 1. High-level overview of edge and cloud computing paradigms for diagnosing faults in induction motors.

2. Background

2.1. Condition Monitoring of Industrial Motors

Condition monitoring (CM) has evolved significantly, transitioning from manual inspections to advanced, data-driven approaches. Initially, industries relied on manual inspections and reactive maintenance, which were labour-intensive, prone to errors, and only addressed faults after significant damage had occurred [12]. The introduction of motor current signature analysis (MCSA) marked a major milestone in industrial motor fault detection, enabling analysis of motor current signals using techniques such as Fast Fourier Transform (FFT) and Discrete Fourier Transform (DFT) [11,13,14]. MCSA provides a predictive approach, allowing for earlier fault detection compared to manual methods, particularly for detecting common issues like rotor bar damage or bearing faults in motors. However, MCSA faces some notable limitations as it is sensitive to noise, particularly in complex fault scenarios, and relies heavily on expert interpretation, making it less scalable and less adaptable to modern industrial environments [15,16]. These challenges, combined with increasing operational complexity and the rise of industrial automation, have driven the transition towards DDCM. The growth of the Internet of Things (IoT) and the availability of large-scale operational data have further accelerated this shift.

Machine learning (ML) has become a cornerstone of DDCM, offering flexibility, scalability, and superior fault detection capabilities. Unlike traditional techniques, ML can process vast datasets, detect intricate patterns, and adapt to various motor types and operational conditions [17]. Studies have demonstrated the effectiveness of ML techniques such as Support Vector Machines (SVMs), Decision Trees, K-Nearest Neighbours (KNN), and Artificial Neural Networks (ANNs), achieving fault detection accuracies exceeding 95% [17–22]. These advancements position ML as an indispensable tool for predictive maintenance strategies, enabling industries to minimise unplanned downtime and optimise operational efficiency. In the context of DDCM and this study, real-time processing refers to the immediate analysis and response, based on ML models, to data collected from various sensors. This is essential for instant fault detection and diagnosis, continuous online monitoring, and predictive maintenance functions.

2.2. Edge Computing

Edge computing has emerged as a key enabler of the Industrial Internet of Things (IIoT), addressing the need for real-time data processing by bringing computational resources closer to data sources. This proximity reduces latency, conserves network bandwidth, and enhances data privacy, making edge computing particularly well-suited for condition monitoring in industrial motors [23,24]. For example, edge devices like Raspberry Pi have been successfully used to classify motor faults using sensor data such as current and vibration signals [25–27]. Despite its advantages, edge computing does have its limitations, particularly with the compute resources and scalability offered by edge devices, which often constrain the kinds of machine learning tasks that can be accommodated [28]. Researchers have explored techniques to mitigate these limitations, such as reducing the dimension of the feature space and optimising models with techniques such as Principal Component Analysis (PCA) and hyperparameter tuning, in order to improve the efficiency of ML models on edge devices without sacrificing accuracy [29–31]. By adopting such strategies, edge computing can effectively support real-time fault detection while addressing its inherent resource constraints.

2.3. Cloud Computing

Cloud computing offers unparalleled computational resources and storage capabilities, making it a powerful platform for performing machine learning tasks. For example, modern

cloud platforms offered by Amazon and Google can provide scalability, advanced data analytics tools, and resource sharing, enabling the training and deployment of complex ML models [32–34]. These features are particularly advantageous for handling large datasets and performing computationally intensive tasks, such as training deep learning models or executing large-scale simulations. However, cloud computing also presents challenges, including higher operational costs, potential latency issues, and data security concerns. Although these drawbacks limit its effectiveness in applications requiring immediate responses, cloud platforms remain indispensable for off-site processing and large-scale analytics [34].

2.4. Comparative Analysis and Research Gap

In general, a comparison of edge and cloud computing is required to reveal their distinct advantages and trade-offs in the context of their application. Edge computing is known to excel in real-time processing, reducing latency and conserving network bandwidth, making it ideal for immediate fault detection and low-latency applications [9,10]. Conversely, cloud computing is better suited for resource-intensive tasks, such as training ML models and managing large-scale data processing, leveraging its superior computational and storage resources [35]. However, as DDCM becomes more prominent in industrial applications, it is not obvious which of these would be an optimal solution to balance efficiency, scalability, and costs. Despite extensive research into edge and cloud computing individually, a notable gap exists in the literature concerning their comparative evaluation in condition monitoring, specifically for industrial motors. Few studies have explored how different DDCM models perform across these paradigms. This research seeks to address this gap, providing practical insights to guide the selection of computing paradigms for fault detection and predictive maintenance in industrial environments. Table 1 provides a summary of related studies in chronological order, highlighting the methods used, their key strengths, and limitations to emphasise the motivation and novelty of this work.

Table 1. Summary of related work on ML-based edge and/or cloud computing for monitoring.

Study	Focus/Aim	Strengths	Limitations
Ferrari et al., 2019 [9]	Comparison of cloud vs. edge for time-series forecasting using lightweight ML models	Demonstrated lower latency with edge; useful for time-critical systems	Limited analysis on scalability; resource constraints not explored in depth
Paul et al., 2020 [8]	Edge-based deep learning for image-based fault detection	Effective in reducing data transmission; high accuracy in edge models	Focused on vision-based data; limited analysis on scalability and cost
Verma et al., 2021 [10]	Hybrid framework for anomaly detection in IoT environments	Achieved good trade-off between latency and computational load	Generalised IoT environment; lacks evaluation for condition monitoring-specific context
Jagati et al., 2023 [32]	Cloud-centric monitoring using AWS tools	Enabled scalable and complex model training	High cost and network reliance; lacking real-time response capability
Presented study (2025)	Empirical comparison using ML models on AWS EC2 vs. Raspberry Pi for fault diagnosis	Evaluation across performance, cost, data burden, and scalability in DDCM context	Limited to lightweight ML models; energy consumption not evaluated

3. Methodology

3.1. Experimental Setup

3.1.1. Hardware Configuration

This study uses the t3.2xlarge configuration of AWS EC2 as the cloud computing platform. This particular t3 series instance was selected due to its balanced memory and processing capabilities in contrast to other configurations [36]. The chosen configuration includes 8 virtual CPUs and 32GB of RAM, which was deemed adequate for handling intensive machine learning tasks. The popular Raspberry Pi (3B) device is used as the edge computing platform. The choice of platforms for the study was based on the fact that they are widely used and commonly available, and not on any preemptive comparisons of their capabilities or suitability for DDCM applications. As this is an initial study unique to the application area of DDCM, an exploratory approach is taken, intended to serve as a broader basis from which more in-depth, focused comparisons can be articulated.

3.1.2. Dataset

The dataset used in this study is based on data collected for an investigation on a squirrel-cage induction motor, specifically targeting the detection and diagnosis of broken rotor bars [37]. This dataset is selected for this study as MCSA is a typical example of an online application in DDCM. Measurements comprise stator phase currents and voltages sampled at 50 kHz and vibration measurements sampled at 7 kHz. During the original experiment, these signals were measured over an 18-s duration and repeated 10 times, resulting in a total of 1,001,000 samples per feature for each experiment. Different faults, or numbers of broken rotor bars, were collected under different mechanical loads. Figure 2 shows the experimental setup used in [37].

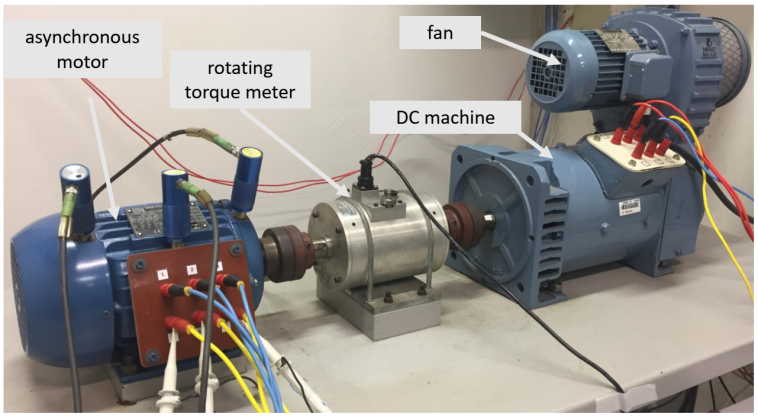

Figure 2. Experimental setup for data collected [37].

For this study, averages were taken across the ten trials to enhance the reliability of the findings. The research primarily focused on the 3-phase current measurements, essentially following the proven spectrum-based MCSA approach for fault diagnosis [38,39].

3.1.3. Software Environment

Data processing and feature engineering were completed using MATLAB R2023b, as it provides a comprehensive set of tools for processing machine signals. Python 3.11 was used for data manipulation and developing the machine learning-based fault diagnosis models.

3.2. Modelling

3.2.1. Machine Learning Models for Condition Monitoring

The models were trained separately on both the edge (Raspberry Pi) and cloud (AWS EC2) platforms to evaluate the impact of computational constraints on training efficiency. The results compare both inference times and training times, as well as CPU/memory usage and scalability for both platforms. Training on the edge device was included to assess its feasibility in resource-constrained environments, while training in the cloud was evaluated to understand its advantages in handling larger datasets and more complex models. Support Vector Machines (SVM), K-Nearest Neighbours (KNN), and Decision Trees (DT) were selected as these are not only popular techniques in condition monitoring, but were also deemed suitable for the purposes of comparative evaluations carried out in this work. Again, this work limits its scope to these models, which are considered to be relatively lightweight when compared to deep learning models. As an initial study intended to fill this gap, an exploratory approach is taken to provide breadth to the scope of the research, and hence, multiple popular learning algorithms are selected as a basis and a framework upon which future research can be built.

The selected algorithms were used in this study for developing the fault diagnosis models due to their proven effectiveness in fault classification tasks, with several studies reporting accuracies exceeding 95% in similar applications [17–21]. Additionally, these algorithms are computationally efficient making them more suitable for application using edge devices like the Raspberry Pi. More resource-intensive models, such as those based on neural networks like Long Short-Term Memory (LSTM), that have demonstrated better performance in condition monitoring [40], were excluded from this study as the variations in computational demands of different model architectures and implementations, particularly on edge devices, require a dedicated separate study [41].

- SVM is suited to handling high-dimensional feature spaces, and it is effective in characterising non-linear relationships using kernel functions. This algorithm is selected here as it has been demonstrated to effectively separate condition monitoring features extracted through Power Spectral Density (PSD) analysis, enabling accurate classification of rotor bar faults [18].
- KNN classifies data instances based on their proximity in the feature space. Its non-parametric nature allows it to adapt well to varying fault patterns, making it suitable for condition monitoring with minimal computational overhead [42,43]. Bayesian optimisation was applied to tune the optimal value of K, improving the classification performance.
- The DT algorithm is a popular choice owing to its interpretability and low computational cost. Its hierarchical structure allows for efficient classification of rotor conditions by following decision paths based on feature thresholds [44].

3.2.2. Data Preprocessing and Feature Engineering

The data were first imported into MATLAB, where the power spectrum density function was used to estimate the PSD of the current signals. In this study, the entire 18-s recordings of the phase-current signals were used. Although still following an MCSA approach, this study uses the PSD instead of the FFT, which has been shown to be more susceptible to false positives [45]. Figure 3 shows the stator current PSD under different conditions. The PSD of these signals reveal distinct characteristics in the frequency domain where harmonics exhibit patterned variations under different conditions, which are learnt by models in order to classify healthy and broken rotor bar fault conditions.

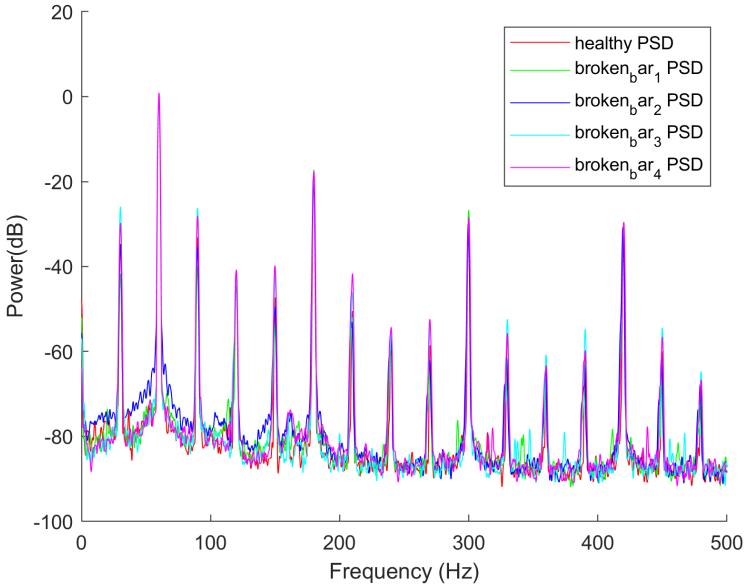

Figure 3. PSD of current signals for healthy and fault conditions.

The harmonics of interest for each of the signals between 60 Hz to 500 Hz were extracted from the PSD estimates, producing a total of 8 features. Table 2 presents a sample of instances extracted from the PSDs to demonstrate how some of the key features, or harmonics, vary under different conditions.

Table 2. Example of instances showing variation in some key features, or harmonics, from PSD analysis for different healthy and fault conditions.

Health	Instance	60 Hz	120 Hz	180 Hz
healthy	7	1.483471	0.010803	0.011020
healthy	10	2.703580	0.019458	0.005308
healthy	11	2.724468	0.019831	0.005443
broken_bar_1	7	1.585917	0.011808	0.012588
broken_bar_1	10	2.838809	0.020390	0.008129
broken_bar_1	11	2.819858	0.020250	0.008034
broken_bar_2	7	1.431021	0.010651	0.011763
broken_bar_2	10	2.712764	0.019249	0.010197
broken_bar_2	11	2.692282	0.019351	0.011599
broken_bar_3	7	1.438370	0.010723	0.013016
broken_bar_3	10	2.865887	0.020938	0.015434
broken_bar_3	11	2.850097	0.020939	0.014791
broken_bar_4	7	1.629478	0.012014	0.016912
broken_bar_4	10	2.961412	0.021510	0.019296
broken_bar_4	11	2.942672	0.021436	0.017215

1. PCA was used to reduce the feature space from 8 to 2 dimensions, yielding a variance retention of 99.99%, which ensures minimal loss of information. The primary motivation for using PCA was to evaluate the impact of dimensionality reduction on both training efficiency and data transmission, particularly in edge computing environments. PCA has been widely recognised for reducing data transmission in IoT and Industry 4.0 applications, enhancing security by avoiding raw data transmission, and improving energy efficiency [46,47]. These benefits align with the needs of real-time, resource-constrained condition monitoring systems. Although PCA effectively reduces the complexity of the input data and improves training efficiency,

the results indicate increased CPU and memory usage during inference. This is due to the computational cost of applying the transformation matrix to incoming data before classification, a known challenge in embedded machine learning implementations [46,48]. While previous studies suggested PCA could improve performance for DDCM applications [49,50], these results highlight a trade-off where PCA reduces training time and data transmission load, but this may come at the cost of latency in inference.

2. The evaluation metrics used in this study were the F1-score, training and inference times, resource utilisation (CPU and memory), and costs. With regards to resource utilisation, CPU usage is normalised across platforms to afford comparison between CPUs with different number of cores. Data burden and scalability were also evaluated by varying the size of the test data. The metrics were also evaluated with and without dimensionality reduction to estimate the impacts between platforms.
3. Experimental tests were conducted by firstly dividing the dataset, where 60% of the set was used for model development and testing, and 40% of the set was set aside for scalability testing. An 80:20 split was used for training and testing of the models. Scalability tests used data at different increments—i.e., 10%, 25%, 50%, and 100%. Models were developed using SVM, DT, and KNN, tuned using Bayesian optimisation, and tested on both edge and cloud platforms.

4. Results and Analysis

4.1. Tables of Results

Tables 3–6 summarise the various metrics recorded with and without dimensionality reduction and model optimisation.

Table 3. Performance summary without PCA and model optimisation.

Metric	Edge (Raspberry Pi)	Cloud (AWS EC2)
DT		
F1-Score	1.00	1.00
Training Time (s)	0.0347	0.0015
Inference Time (s)	0.0014	0.0008
Train CPU Usage (%)	3.73	0.52
Train Memory (MB)	43.20	11.50
SVM		
F1-Score	0.38	0.38
Training Time (s)	0.0124	0.0040
Inference Time (s)	0.0023	0.0004
Train CPU Usage (%)	3.79	0.93
Train Memory (MB)	43.96	11.62
KNN		
F1-Score	0.75	0.75
Training Time (s)	0.0069	0.0009
Inference Time (s)	0.0196	0.0011
Train CPU Usage (%)	4.30	0.10
Train Memory (MB)	42.67	11.60

Table 4. Performance summary with PCA and model optimisation.

Metric	Edge (Raspberry Pi)	Cloud (AWS EC2)
DT		
F1-Score	0.96	0.96
Training Time (s)	0.0067	0.0012
Inference Time (s)	0.0013	0.0006
Train CPU Usage (%)	8.07	0.41
Train Memory (MB)	71.61	13.30
SVM		
F1-Score	0.93	0.93
Training Time (s)	0.0094	0.0004
Inference Time (s)	0.0031	0.0011
Train CPU Usage (%)	8.96	0.17
Train Memory (MB)	75.70	13.30
KNN		
F1-Score	0.95	0.95
Training Time (s)	0.0043	0.0011
Inference Time (s)	0.0207	0.0023
Train CPU Usage (%)	7.22	0.28
Train Memory (MB)	76.60	13.40

Table 5. Comparison of inference time and accuracy at varying sample sizes.

Model	Sample Size (%)	Edge Inference Time (s)	Edge Accuracy (%)	Cloud Inference Time (s)	Cloud Accuracy (%)
DT	10%	0.0020	100.00	0.0000	100.00
	25%	0.0010	88.89	0.0000	88.89
	50%	0.0010	94.44	0.0000	94.44
	75%	0.0011	96.30	0.0000	96.30
	100%	0.0012	97.26	0.0000	97.26
SVM	10%	0.0023	100.00	0.0000	100.00
	25%	0.0020	100.00	0.0000	100.00
	50%	0.0028	100.00	0.0000	100.00
	75%	0.0037	100.00	0.0000	100.00
	100%	0.0043	98.63	0.0000	98.63
KNN	10%	0.0265	100.00	0.0000	100.00
	25%	0.0125	100.00	0.0081	100.00
	50%	0.0214	97.22	0.0000	97.22
	75%	0.0243	98.15	0.0086	98.15
	100%	0.0315	97.26	0.0000	97.26

4.2. Discussion of Results

4.2.1. F1-Score

The results presented in Table 4 demonstrate that the F1 score remained consistent across both the edge (Raspberry Pi) and cloud (AWS EC2) platforms for all the machine learning models tested. This consistency indicates that the choice of deployment platform has no impact on the accuracy of the models. It should be highlighted that model accuracies are 100% in some cases. This is related to overfitting and the scalability testing, which is discussed later in Section 4.2.4. Regardless of which platform the model is deployed on, the predictive performance remains unchanged. This finding suggests that when selecting a deployment platform for machine learning-based condition monitoring systems, accuracy

is not a determining factor. Instead, the decision should focus on other performance metrics such as computational speed, scalability, and resource efficiency.

Table 6. Analysis of data burden results.

Data Type	Samples (18 s)	Samples (24 h)	Size per Sample/Segment	Total Size (24 h)	Impact on Network Bandwidth
Original Data	1,001,000	4,318,320,000	8 Bytes	34.55 GB	Extremely high, impractical for real-time cloud upload.
Feature Engineered Data	245	52,920	15,680 Bytes/segment	830.78 MB	Very low, feasible for real-time transmission.

4.2.2. Training and Inference Times

A significant difference between the platforms was observed in the training and inference times, as illustrated in Figures 4 and 5. The results revealed that the cloud platform consistently outperformed the edge platform in both metrics across all models. Specifically, the cloud achieved approximately 844% faster training times and 1013% faster inference times, on average, when compared to the edge. These performance gains can be attributed to the cloud's superior computational resources and scalability, which allow for faster data processing and model training. Although edge training times are significantly reduced by more than 50% on average with the use of PCA, they are still not comparable to that of times achieved using the cloud.

These findings have important implications for industrial condition monitoring applications. Systems that demand real-time fault detection and continuous model updates would benefit significantly from cloud deployment due to its faster processing capabilities. In contrast, the edge platform may still be suitable for applications where low-latency responses and offline operation are critical. However, for scenarios requiring scalable, high-performance computing, the cloud platform offers a clear advantage. Further analysis reveals that the DT model exhibited the longest training time among the models, while the KNN model had the longest inference time. Despite these model-specific differences, the cloud platform consistently outperformed the edge platform across all tested scenarios. This performance gap highlights the cloud's advantage in handling complex models and workloads that require frequent retraining or rapid data processing.

4.2.3. CPU and Memory Usage

The findings of this study reveal that the edge platform consistently exhibited higher CPU and memory usage compared to the cloud platform across all tested models, as illustrated in Figures 6 and 7. Specifically, the edge platform consumed more than three times the CPU and memory resources than the cloud. This significant disparity highlights the superior resource efficiency of the cloud platform, enabling it to handle complex computational tasks and data-intensive operations without overloading system resources.

The data further show that while the cloud platform maintained exceptionally low CPU usage during both training and inference phases—reporting usage as low as 0.3% in Python-based metrics—the AWS CloudWatch dashboard reflected a slightly higher average CPU utilisation of approximately 4%, as depicted in Figure 8. This discrepancy can be attributed to the background processes of the cloud platform, including the management

of the operating system, Docker containers, NAT gateways, and other system services that contribute to the baseline resource consumption.

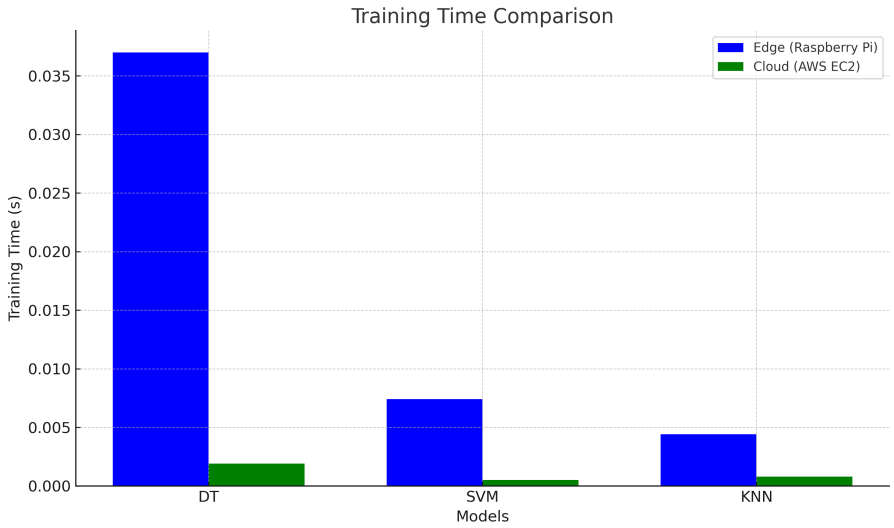

Figure 4. Comparison of training times on different platforms across models.

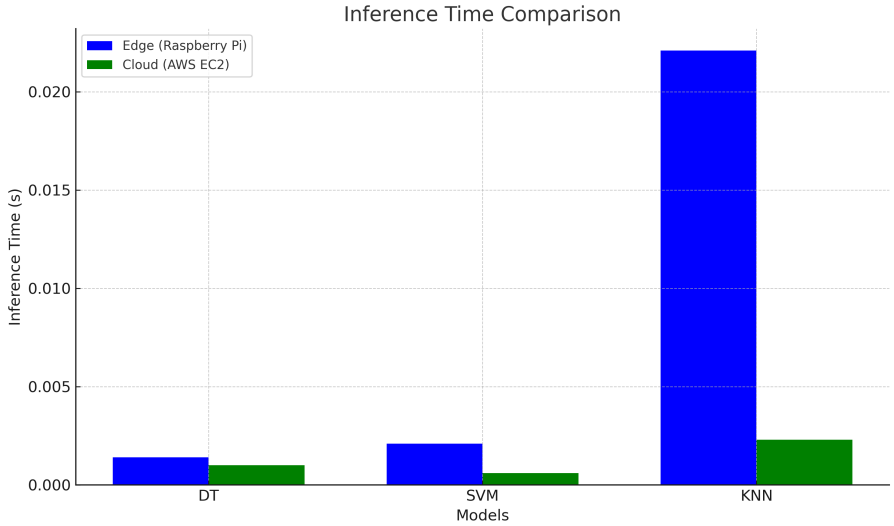

Figure 5. Comparison of inference times on different platforms across models.

These results emphasise that the cloud platform is better suited for executing memory- and CPU-intensive tasks due to its scalable infrastructure and optimised resource utilisation. The flexible and centralised nature of the cloud paradigm may also be particular useful in cases where models are frequently updated. However, it is important to recognise that as task complexity increases, the cloud may engage additional background processes, potentially leading to increased latency and higher operational costs. Therefore, while the cloud is advantageous for handling complex workloads, careful management of computational tasks is essential to mitigate hidden overhead and ensure cost-efficiency.

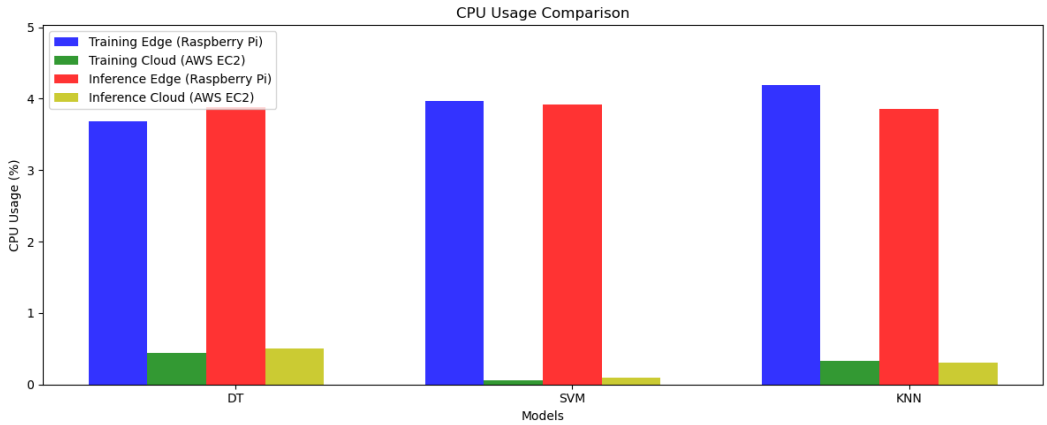

Figure 6. Comparison of CPU usage on different platforms across models.

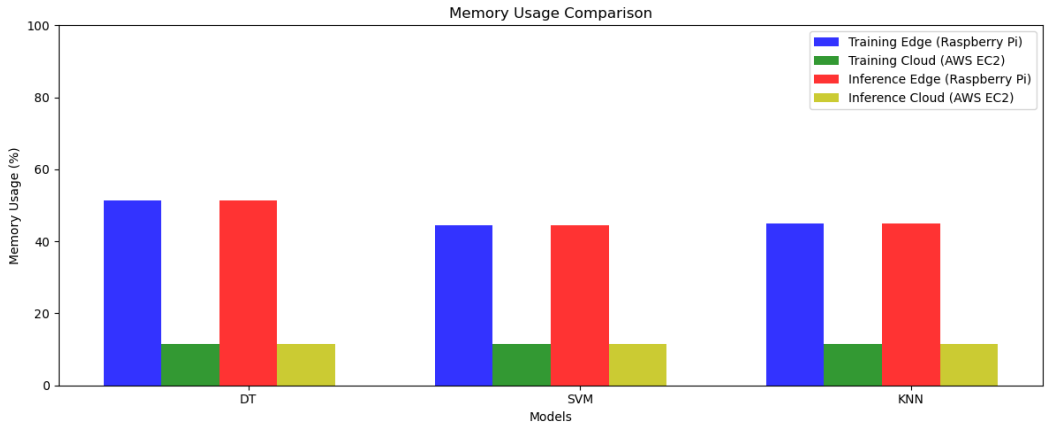

Figure 7. Comparison of Memory usage on different platforms across models.

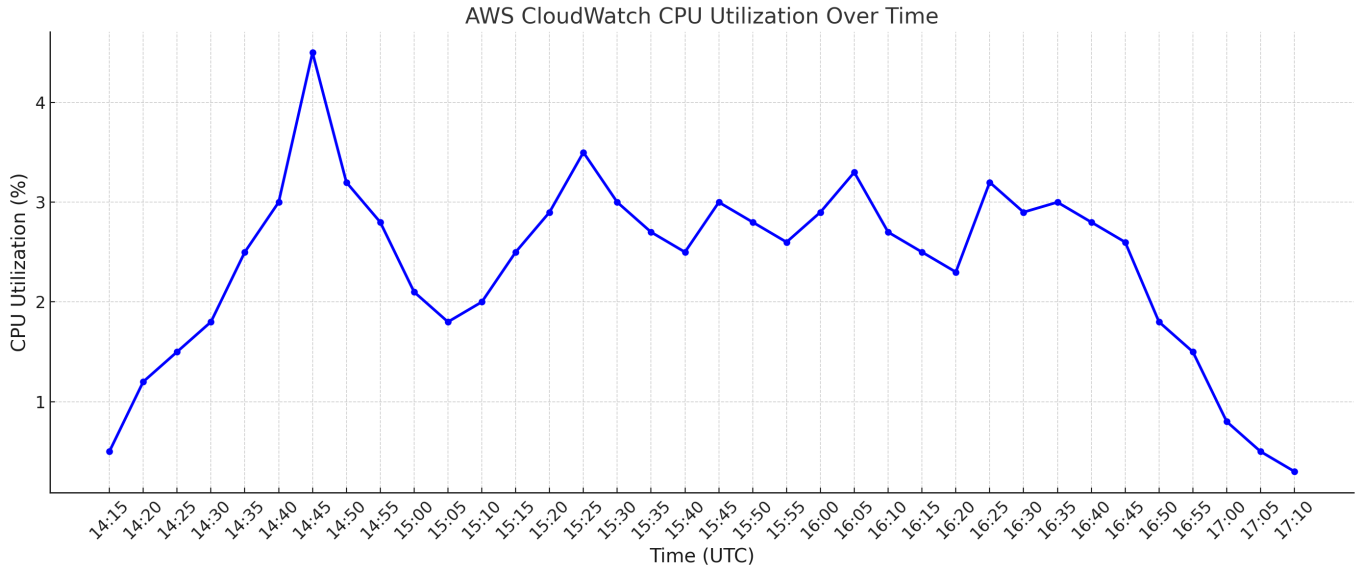

Figure 8. AWS dashboard showing resource utilisation.

4.2.4. Scalability Analysis

The study results demonstrate that the edge platform (Raspberry Pi) consistently exhibited higher CPU and memory usage compared to the cloud platform (AWS EC2) across all machine learning models tested. Specifically, the edge platform consumed more than three times the CPU and memory resources of the cloud. This significant disparity

highlights the cloud platform's superior resource efficiency, enabling it to manage complex computational tasks and data-intensive operations without straining its system resources. For instance, during both training and inference phases, the DT model recorded a CPU usage of approximately 4.2% and memory usage of 52% on the edge platform, whereas the cloud platform maintained CPU usage below 1% and memory usage around 13%. Similar trends were observed for the SVM and KNN models. This consistent performance on the cloud is attributed to its ability to dynamically allocate computational resources as needed. Additionally, while the cloud platform's CPU usage during training and inference was as low as 0.3%, data collected from the AWS CloudWatch dashboard revealed a slightly higher average CPU utilisation of approximately 4%, as depicted in Figure 8. This discrepancy arises because the cloud allocates CPU resources not only to the primary computational tasks but also to various background operations, including the management of the operating system, Docker containers, NAT gateways, and other essential services. With respect to inference times and accuracies at various scales of sample data (Table 5), the recorded inference times for the edge device were impacted the most with increasing data sizes. The accuracies are not impacted across computing paradigms as expected. Some models are found to achieve 100% accuracy, which does indicate overfitting, especially in the case of the SVM models. This is expected where reduced sample sizes are intentionally used for the scalability tests, and therefore, cross-validation is not directly employed to overcome this issue. However, improvements with models in relation to this aspect are then observed as the sampled sizes are scaled up.

These findings emphasise the cloud platform's advantage in handling memory- and CPU-intensive workloads due to its scalable and efficient infrastructure. However, it is important to recognise that as computational complexity increases, the cloud may engage additional background processes, potentially leading to increased latency and higher operational costs. Therefore, while the cloud is well-suited for large-scale, complex computations, careful resource management is essential to optimise performance and maintain cost-efficiency. In contrast, the edge platform, with its higher resource consumption, may be more suitable for lightweight, latency-sensitive applications but is less capable of scaling effectively for data-intensive tasks in industrial environments. This limitation could affect critical processes such as real-time fault detection and predictive maintenance, where rapid data processing is essential for timely decision-making.

4.2.5. Data Burden

The results of the data burden analysis, as summarised in Table 6, revealed that the cloud platform incurs a significantly higher data transmission burden compared to the edge platform. For example, collecting raw data over an 18-s period at a 50 kHz sampling rate produced approximately 8 MB of data. In contrast, when processed into feature-engineered data, the same dataset was reduced to around 10 KB. However, when scaled to a 24-h monitoring period, the raw data would accumulate to approximately 34 GB, whereas the feature-engineered data would only consume about 870 MB of storage. This substantial difference underscores the inefficiency of transmitting raw data to the cloud, as it would place a considerable burden on network bandwidth, leading to increased latency and higher operational costs. Such an overhead makes the cloud platform less suitable for real-time condition monitoring, where rapid data processing and cost-efficiency are critical. In contrast, the edge platform processes data locally, significantly reducing the need for large data transmissions and mitigating network strain. This local processing capability makes the edge platform more appropriate for time-sensitive industrial applications that demand immediate fault detection and minimal network dependency.

4.2.6. Cost Analysis

The cost analysis compared the expenses associated with running machine learning models on the cloud platform (AWS EC2) and the edge platform (Raspberry Pi 3B). The total cost for using the AWS EC2 instance was $8.14, based on a rate of approximately $0.80 per hour over a 10-h period. In contrast, the edge platform incurred a one-time purchase cost of $35 for the Raspberry Pi 3B. At first glance, the edge platform appears more expensive due to its higher initial investment. However, the cloud platform's operational costs increase proportionally with usage duration, which could make it significantly more expensive over time than the fixed-cost edge device.

Additionally, the AWS billing dashboard (Figure 9) reveals that the cloud charges approximately $0.40 for compute resources (CPU and memory) and an additional $0.40 for background services. These include resources such as NAT gateways, EBS storage volumes, Docker containers, and idle Elastic IP addresses, which continue to incur charges even when not actively in use. This continuous billing for idle services can contribute to unexpected costs, especially for applications that require intermittent or lightweight processing.

While this study focused on short-term cost analysis, real-world condition monitoring applications typically require continuous or periodic operation over months or years. The long-term cost implications of cloud vs. edge computing will depend on factors such as:

- Frequency of model retraining, which increases cloud usage costs.
- Power consumption of edge devices over extended periods.
- Data transmission costs, particularly when large amounts of sensor data are sent to the cloud.
- Maintenance and hardware replacement for edge devices.

A more comprehensive long-term cost analysis would require extended testing under real industrial conditions. Future studies should consider evaluating these costs over a prolonged duration, incorporating factors such as total cost of ownership (TCO) and operational efficiency in real-world deployments.

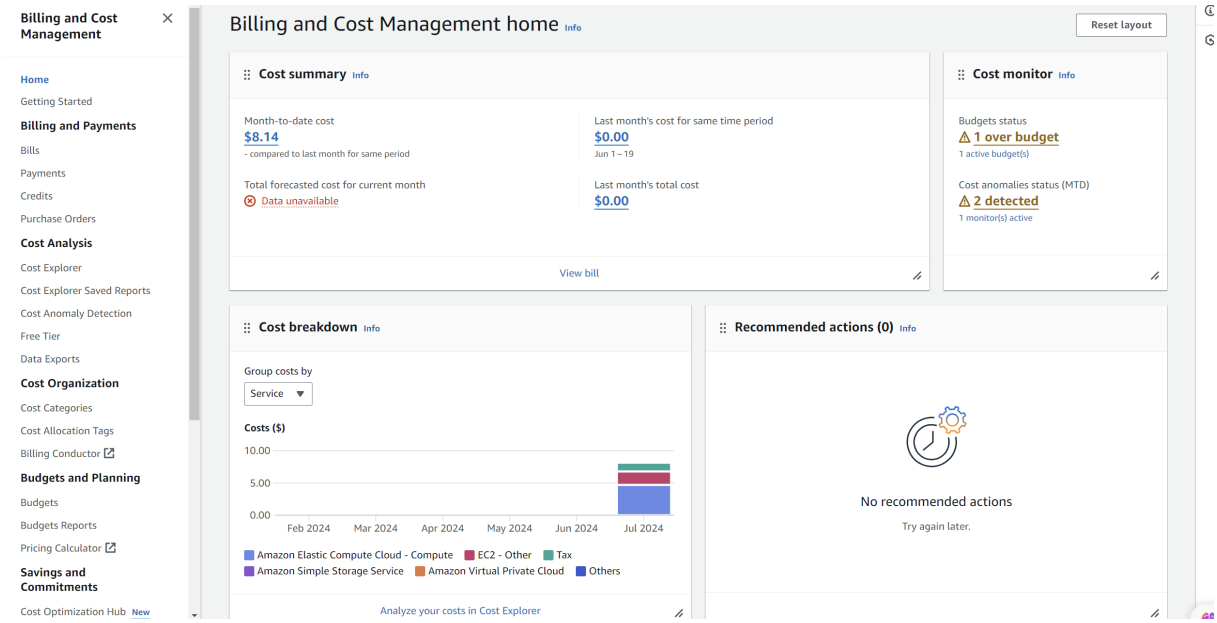

Figure 9. AWS dashboard showing costs incurred during testing.

4.3. Impact of PCA

PCA led to a significant rise in resource consumption, with memory usage increasing by 72.46% and CPU usage by 105.16% on both platforms. This suggests that while PCA

reduces complexity during training, it introduces additional processing demands during inference, thereby impacting overall operational efficiency. Importantly, the analysis shows that PCA did not significantly impact the F1 scores across the tested models, indicating that model accuracy remained stable despite the reduction in feature space. This stability suggests that PCA successfully preserved the most informative features for model performance. In summary, while PCA effectively enhanced training efficiency by reducing training time, it introduced trade-offs in the form of increased inference time and greater CPU and memory utilisation. These findings suggest that PCA should be applied with caution, particularly in real-time industrial applications where inference speed and resource usage are critical performance metrics.

4.4. Qualitative Considerations: Ease of Deployment, User Experience, and Network Variability

Although this study has focused primarily on quantitative performance metrics, qualitative factors also play a significant role in determining the practical feasibility of edge and cloud computing in industrial condition monitoring. Based on widely acknowledged advantages and limitations of these paradigms in the literature, there are some key impacts of real-world deployment that are not necessarily quantifiable with the presented experimental setup but still require careful consideration. One of these key considerations is the ease of deployment. Cloud platforms provide a streamlined deployment pipeline where machine learning models can be integrated with scalable infrastructure and managed remotely. This reduces the burden of on-site hardware setup and maintenance but introduces additional concerns regarding API configurations, cloud security policies, and compliance with regulatory frameworks [51]. In contrast, edge computing requires local hardware deployment and software management, which may involve additional setup complexity. However, it offers greater control over the data pipeline, minimising dependence on external service providers. Another critical aspect is user experience. Cloud-based solutions benefit from centralised monitoring dashboards and managed services, allowing seamless integration with industrial IoT frameworks [52]. However, they require stable internet connectivity to ensure uninterrupted operation. Edge devices, on the other hand, process data locally, reducing reliance on network connectivity and enabling real-time decision-making. This makes edge computing particularly advantageous in remote locations or industrial environments where consistent internet access is not guaranteed.

Network variability can also significantly impact the efficiency of cloud computing. Variability in latency, bandwidth availability, and network outages may introduce delays in data transmission, affecting real-time inference. The cloud platform's performance is highly dependent on internet stability, which can fluctuate due to factors such as network congestion, infrastructure limitations, or geographic constraints. In contrast, edge computing ensures that critical computations remain independent of external network conditions, thereby offering greater reliability in latency-sensitive applications. These qualitative factors, alongside the quantitative performance metrics of this study, should be taken into account when selecting a computing paradigm for condition monitoring applications. Although cloud computing offers a highly scalable and centralised approach, edge computing provides localised decision-making and operational independence. The choice between these two paradigms depends on the specific requirements of the industrial application, including deployment complexity, data sensitivity, and real-time processing needs.

4.5. Practical Implications and Recommendations

Selecting the most suitable computing paradigm for DDCM of industrial induction motors requires a careful evaluation of multiple factors, including real-time processing needs, data complexity, cost, model retraining frequency, network stability, and data

security. Developers must carefully weigh these factors to determine whether to implement a solution using edge computing, cloud computing, or a hybrid approach. This discussion explores key questions that industrial developers must consider and offers evidence-based insights derived from the findings of this study.

4.5.1. Is Real-Time Fault Detection and Low Latency Critical?

The edge platform proves to be the most effective solution for condition monitoring systems where real-time fault detection and immediate response are critical. This study demonstrates that edge computing offers significantly lower latency compared to the cloud due to its localised data processing, eliminating delays caused by network data transfer. In industrial settings, where machine faults can lead to costly downtimes or safety hazards, the ability to detect anomalies and trigger immediate responses is essential. The cloud, despite its superior computational resources, introduces inherent latency through data transmission, which can hinder timely interventions. Therefore, in environments where instantaneous decision-making is required, deploying machine learning models on the edge is recommended. The cloud can still be utilised effectively for more complex analyses in applications where minor delays are acceptable. However, latency should be assessed in these cases where the risk of delay is more substantial. Since latency is highly system-dependent, these end-to-end delays should be carefully evaluated based on the features of the application scenario, such as the network topology, sensor and data centre networks, etc.

4.5.2. How Large and Complex Are the Data That Are Being Processed?

The nature and scale of data being processed play a crucial role in determining the suitable computing paradigm. This study revealed that the edge platform struggles with large datasets and complex machine learning models due to its limited computational power and memory capacity. In contrast, the cloud platform efficiently manages large-scale data processing and supports complex models through scalable computational resources. When condition monitoring involves high-frequency sensor data or requires advanced analytics using deep learning models, the cloud's ability to dynamically allocate resources would make it the more suitable option. Conversely, for lightweight models and moderate data volumes, the edge platform offers a more practical and efficient solution. There is certainly scope for further comparisons in this regard, with a focus on more computationally intensive models, where the scope of this study is limited to lightweight models. Additionally, the case study used in this work employed MCSA, where current is the key measurement modality. Although this is one of the most popular online measurement modalities for DDCM in rotating electrical machines, it should be noted that there is scope for further comparison of different measurement modalities, inter alia, vibration- and thermal-based modalities, that may introduce different context and complexities to the data handling requirements.

4.5.3. What Are the Cost Constraints of Deployment and Operation?

The study's findings indicated that the edge platform offers a cost-effective solution due to its one-time hardware investment (e.g., the Raspberry Pi at $35), with minimal ongoing expenses. In contrast, the cloud platform operates on a pay-as-you-go model, incurring continuous costs for compute power, data storage, and background services, which can accumulate over extended use. These findings suggest that for industries with tight budget constraints or long-term monitoring needs, the edge platform is more financially viable. However, for applications requiring advanced computational power, large-scale data processing, and frequent model retraining, the cloud's higher operational costs are justified by its ability to handle complex workloads efficiently.

4.5.4. How Frequently Does the Model Need to Be Retrained and Updated?

In dynamic industrial environments, where equipment conditions and operational demands are constantly evolving, the frequency of machine learning model retraining becomes a critical factor for maintaining high predictive accuracy. This study demonstrated that the cloud platform significantly outperforms the edge platform in managing frequent model updates due to its scalable computational resources and ability to handle large-scale retraining tasks efficiently. Conversely, the edge platform's limited processing power and memory capacity make it unsuitable for regularly updating complex models, restricting its use in adaptive learning environments. These findings suggest that industries with operations that require continuous model adaptation, such as those affected by changing load conditions or environmental factors, would greatly benefit from deploying models on the cloud platform. In contrast, the edge platform is more effective for applications involving static models that rarely need updates, offering a stable and cost-efficient solution to monitor consistent and predictable processes.

4.5.5. Is Network Reliability and Bandwidth Availability a Concern?

Network reliability and bandwidth availability are critical factors in determining the suitability of edge or cloud computing for industrial condition monitoring. This study demonstrated that the edge platform is inherently more resilient to network instability because it performs data processing locally, eliminating reliance on continuous internet connectivity. This makes edge computing particularly effective for deployment in remote industrial environments or facilities with limited or unreliable network infrastructure, where transmitting large datasets to the cloud could introduce significant latency and increase the risk of data loss. In contrast, the cloud platform requires a stable, high-speed internet connection for efficient data transfer and real-time processing. Although the cloud is ideal for operations in environments with robust and reliable network infrastructure, it may be impractical in settings where network interruptions are frequent, or bandwidth is constrained. Therefore, industries operating in network-constrained or geographically isolated locations would benefit from the edge platform, ensuring consistent and uninterrupted condition monitoring. Conversely, industries with access to reliable high-bandwidth networks can leverage the cloud platform for scalable, high-performance data analysis.

4.5.6. Large-Scale Industrial Condition Monitoring

Considering the trade-offs between edge and cloud computing, this study recommends adopting a hybrid edge-cloud architecture for large-scale industrial condition monitoring. The hybrid solution combines the real-time processing capabilities of the edge with the scalability and computational power of the cloud, offering an optimised and balanced framework. In this configuration, edge devices could handle real-time data acquisition, preprocessing, and fault detection, reducing network traffic and latency. Processed data, particularly key features, anomalies, and even just assessments, could be transmitted to the cloud for either front-end dashboards, complex analysis, model retraining, and/or long-term data storage. This division of labour could potentially ensure fast and localised responses to critical issues while leveraging the computational resources of the cloud for advanced analytics. The hybrid model offers several advantages. It significantly reduces network burden by transmitting only essential data to the cloud, thereby lowering bandwidth costs and minimising latency. It also optimises costs by limiting reliance on expensive cloud services for routine data processing. Additionally, the cloud provides scalability for handling large datasets and complex models, while edge devices enable real-time responsiveness. Despite implementation challenges, such as data synchronisation and system integration, the benefits of the hybrid model make it a superior solution for in-

dustrial condition monitoring. Although this study focuses on performance metrics such as computational efficiency, scalability, and cost, power consumption remains a critical factor for industrial condition monitoring applications. Edge computing platforms like Raspberry Pi typically consume significantly less power (2–5 W) compared to cloud instances (over 100 W under load) [53,54]. However, power demand on edge devices increases by up to 40% during machine learning operations, depending on model complexity [53]. Emerging energy management strategies, such as dynamic power scaling and integration with renewable energy sources, could potentially offer optimised edge computing for industrial deployments [55]. Thus, the hybrid architecture requires careful design and implementation, where the impacts of power consumption should be considered in conjunction with computational efficiency, scalability, and costs for large-scale DDCM.

5. Conclusions

This study compared the Raspberry Pi (edge) and AWS EC2 (cloud) platforms for the DDCM application of induction motor fault diagnosis based on three widely used machine learning algorithms: SVM, KNN, and DT. Key differences were highlighted between the two platforms in terms of accuracy, training time, inference time, CPU and memory usage, data burden, cost, and scalability. The findings indicate that the edge platform excels in applications requiring low latency and real-time processing due to its localised data handling, making it ideal for time-sensitive industrial operations. However, its limited computational power restricts its ability to process large datasets and complex machine learning models. In contrast, the cloud platform offers superior computational capacity and scalability, making it more suitable for data-intensive tasks and models requiring frequent retraining, albeit with higher costs and greater reliance on stable network infrastructure. These findings suggest that neither paradigm alone can comprehensively meet all of the various requirements of modern industrial condition monitoring in a cost-sensitive manner.

Valuable insights were also provided to support practical and balanced assessment of computing frameworks to be deployed in DDCM applications. The insights gained from this initial study provide a basis for future work, and certain limitations are acknowledged where there are several issues that warrant further investigation. The analysis did not include comparisons of energy consumption between edge and cloud systems, which is a critical factor for sustainable deployment in industrial environments. The study did not fully explore the performance of more advanced deep learning models, which are becoming increasingly important in condition monitoring. Instead, we decided to focus on relatively lightweight models as an initial scope of exploratory work. Further investigation dedicated to more complex deep learning models is required in this context and will, therefore, form part of future work. The experimental evaluation offered here focused solely on induction motor data related to broken rotor bar faults. Although the results provide valuable insights for this specific fault type, more research is needed to validate the generalisability of these findings across other condition monitoring tasks, including different measurement modalities and fault types, in industrial machines. Future work will also include direct power measurements for both edge and cloud computing platforms to provide a comprehensive comparison of energy efficiencies. These investigations will be presented in future work to provide further insight into the trade-offs between performance, cost, and sustainability of computing paradigms in DDCM systems.

Author Contributions: Conceptualization, W.D.; methodology, C.C.W. and W.D.; software, C.C.W.; validation, C.C.W.; formal analysis, C.C.W. and W.D.; investigation, C.C.W.; resources, C.C.W. and W.D.; writing—original draft preparation, C.C.W. and W.D.; writing—review and editing, W.D.; visualization, C.C.W. and W.D.; supervision, W.D.; project administration, C.C.W. and W.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data used in this study is publicly available [37].

Acknowledgments: The authors acknowledge the University of Leeds for providing the necessary resources and facilities that supported this research project.

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

References

1. Tavner, P.; Ran, L.; Penman, J.; Sedding, H. *Condition Monitoring of rOtating Electrical Machines*; The Institution of Engineering and Technology: London, UK, 2008.
2. Ahmad, R.; Kamaruddin, S. A review of condition-based maintenance decision-making. *Eur. J. Ind. Eng.* **2012**, *6*, 519–541. [CrossRef]
3. Ali, A.; Abdelhadi, A. Condition-based monitoring and maintenance: State of the art review. *Appl. Sci.* **2022**, *12*, 688. [CrossRef]
4. Haq, S.U.; Trivedi, A.; Rochon, S.; Moorthy, M.T. Alternative Methods of Machine Online Condition Monitoring: Recommendations for Rotating Machines in the Petroleum and Chemical Industry. *IEEE Ind. Appl. Mag.* **2024**, *30*, 19–31. [CrossRef]
5. Das, O.; Das, D.B.; Birant, D. Machine learning for fault analysis in rotating machinery: A comprehensive review. *Eng. Appl. Artif. Intell.* **2023**, *9*, e17584. [CrossRef]
6. Latil, D.; Ngouna, R.H.; Medjaher, K.; Lhuisset, S. Enhancing Data-driven Vibration-based Machinery Fault Diagnosis Generalization Under Varied Conditions by Removing Domain-Specific Information Utilizing Sparse Representation. In Proceedings of the PHM Society European Conference, Prague, Czech Republic, 3–5 July 2024; Volume 8.
7. Zhao, C.; Chen, J.; Jing, H. Condition-driven data analytics and monitoring for wide-range nonstationary and transient continuous processes. *IEEE Trans. Autom. Sci. Eng.* **2020**, *18*, 1563–1574. [CrossRef]
8. Paul, A.K. Edge or Cloud: What to Choose? In *Cloud Network Management*; Chapman and Hall/CRC: New York, NY, USA, 2020; pp. 15–25.
9. Ferrari, P.; Rinaldi, S.; Sisinni, E.; Colombo, F.; Ghelfi, F.; Maffei, D.; Malara, M. Performance evaluation of full-cloud and edge-cloud architectures for Industrial IoT anomaly detection based on deep learning. In Proceedings of the 2019 II Workshop on Metrology for Industry 4.0 and IoT (MetroInd4.0 & IoT), Naples, Italy, 4–6 June 2019; IEEE: New York, NJ, USA, 2019; pp. 420–425.
10. Verma, A.; Goyal, A.; Kumara, S.; Kurfess, T. Edge-cloud computing performance benchmarking for IoT based machinery vibration monitoring. *Manuf. Lett.* **2021**, *27*, 39–41. [CrossRef]
11. Edomwandekhoe, K.I. Modeling and Fault Diagnosis of Broken Rotor Bar Faults in Induction Motors. Ph.D. Thesis, Memorial University of Newfoundland, St. John's, NL, Canada, 2018.
12. Lee, S.B.; Stone, G.C.; Antonino-Daviu, J.; Gyftakis, K.N.; Strangas, E.G.; Maussion, P.; Platero, C.A. Condition monitoring of industrial electric machines: State of the art and future challenges. *IEEE Ind. Electron. Mag.* **2020**, *14*, 158–167. [CrossRef]
13. El Hachemi Benbouzid, M. A review of induction motors signature analysis as a medium for faults detection. *IEEE Trans. Ind. Electron.* **2000**, *47*, 984–993. [CrossRef]
14. Raman, R.; Naikade, K. Smart Industrial Motor Monitoring with IoT-Enabled Photovoltaic System. In Proceedings of the 2023 7th International Conference on IoT in Social, Mobile, Analytics and Cloud (I-SMAC), Kirtipur, Nepal, 11–13 October 2023; IEEE: New York, NJ, USA, 2023; pp. 53–57.
15. Thorsen, O.; Dalva, M. Condition monitoring methods, failure identification and analysis for high voltage motors in petrochemical industry. In Proceedings of the 8th International Conference on Electrical Machines and Drives, Cambridge, UK, 1–3 September 1997; IET: Stevenage, UK, 1997.
16. Oñate, W.; Perez, R.; Caiza, G. Diagnosis of incipient faults in induction motors using mcsa and thermal analysis. In *Advances in Emerging Trends and Technologies: Volume 2*; Springer: Cham, Switzerland, 2020; pp. 74–84.
17. Manikandan, S.; Duraivelu, K. Fault diagnosis of various rotating equipment using machine learning approaches–A review. *Proc. Inst. Mech. Eng. Part E J. Process Mech. Eng.* **2021**, *235*, 629–642. [CrossRef]
18. Bensaoucha, S.; Moreau, S.; Bessedik, S.A.; Ameur, A. Broken Rotor Bars Fault Detection in Induction Machine Using Machine Learning Algorithms. In Proceedings of the 2022 19th International Multi-Conference on Systems, Signals & Devices (SSD), Sétif, Algeria, 6–10 May 2022; IEEE: New York, NJ, USA, 2022; pp. 851–856.
19. Ali, M.Z.; Shabbir, M.N.S.K.; Liang, X.; Zhang, Y.; Hu, T. Machine learning-based fault diagnosis for single-and multi-faults in induction motors using measured stator currents and vibration signals. *IEEE Trans. Ind. Appl.* **2019**, *55*, 2378–2391. [CrossRef]
20. Khalil, A.F.; Rostam, S. Machine Learning-based Predictive Maintenance for Fault Detection in Rotating Machinery: A Case Study. *Eng. Technol. Appl. Sci. Res.* **2024**, *14*, 13181–13189. [CrossRef]
21. Ferraz Júnior, F.; Romero, R.A.F.; Hsieh, S.J. Machine Learning for the Detection and Diagnosis of Anomalies in Applications Driven by Electric Motors. *Sensors* **2023**, *23*, 9725. [CrossRef]

22. Zhong, X.; Ban, H. Crack fault diagnosis of rotating machine in nuclear power plant based on ensemble learning. *Ann. Nucl. Energy* **2022**, *168*, 108909. [CrossRef]
23. Qiu, T.; Chi, J.; Zhou, X.; Ning, Z.; Atiquzzaman, M.; Wu, D.O. Edge computing in industrial internet of things: Architecture, advances and challenges. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 2462–2488. [CrossRef]
24. Holmes, T.; McLarty, C.; Shi, Y.; Bobbie, P.; Suo, K. Energy Efficiency on Edge Computing: Challenges and Vision. In Proceedings of the 2022 IEEE International Performance, Computing, and Communications Conference (IPCCC), Austin, TX, USA, 11–13 November 2022; IEEE: New York, NJ, USA, 2022; pp. 1–6.
25. de Las Morenas, J.; Moya-Fernández, F.; López-Gómez, J.A. The edge application of machine learning techniques for fault diagnosis in electrical machines. *Sensors* **2023**, *23*, 2649. [CrossRef] [PubMed]
26. Mostafavi, A.; Sadighi, A. A novel online machine learning approach for real-time condition monitoring of rotating machines. In Proceedings of the 2021 9th RSI International Conference on Robotics and Mechatronics (ICRoM), Tehran, Iran, 17–19 November 2021; IEEE: New York, NJ, USA, 2021; pp. 267–273.
27. Shubita, R.R.; Alsadeh, A.S.; Khater, I.M. Fault detection in rotating machinery based on sound signal using edge machine learning. *IEEE Access* **2023**, *11*, 6665–6672. [CrossRef]
28. Mirani, A.A.; Velasco-Hernandez, G.; Awasthi, A.; Walsh, J. Key challenges and emerging technologies in industrial IoT architectures: A review. *Sensors* **2022**, *22*, 5836. [CrossRef]
29. Joshi, R.; Somesula, R.S.; Katkoori, S. Empowering Resource-Constrained IoT Edge Devices: A Hybrid Approach for Edge Data Analysis. In Proceedings of the IFIP International Internet of Things Conference, Denton, TX, USA, 2–3 November 2023; Springer: Cham, Switzerland, 2023; pp. 168–181.
30. Filho, C.P.; Marques, E., Jr.; Chang, V.; Dos Santos, L.; Bernardini, F.; Pires, P.F.; Ochi, L.; Delicato, F.C. A systematic literature review on distributed machine learning in edge computing. *Sensors* **2022**, *22*, 2665. [CrossRef]
31. Phan, T.L.J.; Gehrhardt, I.; Heik, D.; Bahrpeyma, F.; Reichelt, D. A systematic mapping study on machine learning techniques applied for condition monitoring and predictive maintenance in the manufacturing sector. *Logistics* **2022**, *6*, 35. [CrossRef]
32. Jagati, A.; Subbulakshmi, T. Building ML workflow for walware images classification using machine learning services in leading cloud platforms. In Proceedings of the 2023 International Conference on Computational Intelligence and Sustainable Engineering Solutions (CISES), Greater Noida, India, 28–30 April 2023; IEEE: New York, NJ, USA, 2023; pp. 233–239.
33. Gautam, A.; Jindal, S.; Baitha, P.; Arora, A.; Gautam, A. The role of cloud computing in machine learning approaches. *Int. J. Eng. Appl. Sci. Technol.* **2023**, *8*, 73–79. [CrossRef]
34. Pourmajidi, W.; Steinbacher, J.; Erwin, T.; Miranskyy, A. On challenges of cloud monitoring. *arXiv* **2018**, arXiv:1806.05914.
35. Bajic, B.; Cosic, I.; Katalinic, B.; Moraca, S.; Lazarevic, M.; Rikalovic, A. Edge vs cloud computing: Challenges and opportunities in Industry 4.0. *Ann. DAAAM Proc.* **2019**, *30*, 864–871.
36. Amazon Web Services. Amazon EC2 Instance Types. Available online: https://aws.amazon.com/ec2/instance-types (accessed on 20 January 2025).
37. Treml, A.E.; Flauzino, R.A.; Suetake, M.; Maciejewski, N.R.; Afonso, N. Experimental database for detecting and diagnosing rotor broken bar in a three-phase induction motor. *IEEE DataPort* **2020** . [CrossRef]
38. Valles-Novo, R.; de Jesus Rangel-Magdaleno, J.; Ramirez-Cortes, J.M.; Peregrina-Barreto, H.; Morales-Caporal, R. Empirical mode decomposition analysis for broken-bar detection on squirrel cage induction motors. *IEEE Trans. Instrum. Meas.* **2014**, *64*, 1118–1128. [CrossRef]
39. Qiao, W.; Qu, L. Prognostic condition monitoring for wind turbine drivetrains via generator current analysis. *Chin. J. Electr. Eng.* **2018**, *4*, 80–89.
40. Li, Z.; Fei, F.; Zhang, G. Edge-to-cloud IIoT for condition monitoring in manufacturing systems with ubiquitous smart sensors. *Sensors* **2022**, *22*, 5901. [CrossRef]
41. Ameen, S.; Siriwardana, K.; Theodoridis, T. Optimizing Deep Learning Models For Raspberry Pi. *arXiv* **2023**, arXiv:2304.13039.
42. Zhang, Z. Introduction to machine learning: K-nearest neighbors. *Ann. Transl. Med.* **2016**, *4*, 11. [CrossRef]
43. Taunk, K.; De, S.; Verma, S.; Swetapadma, A. A brief review of nearest neighbor algorithm for learning and classification. In Proceedings of the 2019 International Conference on Intelligent Computing and Control Systems (ICCS), Madurai, India, 15–17 May 2019; IEEE: New York, NJ, USA, 2019; pp. 1255–1260.
44. Khan, M.A.; Bilal, A.; Vaimann, T.; Kallaste, A. An Advanced Diagnostic Approach for Broken Rotor Bar Detection and Classification in DTC Controlled Induction Motors by Leveraging Dynamic SHAP Interaction Feature Selection (DSHAP-IFS) GBDT Methodology. *Machines* **2024**, *12*, 495. [CrossRef]
45. Edomwandekhoe, K.; Liang, X. Current spectral analysis of broken rotor bar faults for induction motors. In Proceedings of the 2018 IEEE Canadian Conference on Electrical & Computer Engineering (CCECE), Quebec, QC, Canada, 13–16 May 2018; IEEE: New York, NJ, USA, 2018; pp. 1–5.

46. Marino, R.; Lanza-Gutierrez, J.M.; Riesgo, T.M. Embedding principal component analysis inference in expert sensors for big data applications. In *Big Data Recommender Systems—Volume 2: Application Paradigms*; IET: Stevenage, UK, 2019; Chapter 6, pp. 83–105. [CrossRef]
47. Rooshenas, P.; Rabiee, H.R.; Movaghar, A.; Naderi, M.Y. Reducing the data transmission in Wireless Sensor Networks using the Principal Component Analysis. In Proceedings of the IEEE Sixth International Conference on Intelligent Sensors, Sensor Networks and Information Processing, Brisbane, Australia, 7–10 December 2010; pp. 133–138.
48. Burrello, A.; Marchioni, A.; Brunelli, D.; Benatti, S.; Mangia, M.; Benini, L. Embedded Streaming Principal Components Analysis for Network Load Reduction in Structural Health Monitoring. *IEEE Internet Things J.* **2021**, *8*, 4433–4447. [CrossRef]
49. Chippalakatti, S.; Renumadhavi, C.; Pallavi, A. Comparison of unsupervised machine learning Algorithm for dimensionality reduction. In Proceedings of the 2022 International Conference on Knowledge Engineering and Communication Systems (ICKES), Chickballapur, India, 28–29 December 2022; IEEE: New York, NJ, USA, 2022; pp. 1–7.
50. Reddy, G.T.; Reddy, M.P.K.; Lakshmanna, K.; Kaluri, R.; Rajput, D.S.; Srivastava, G.; Baker, T. Analysis of dimensionality reduction techniques on big data. *IEEE Access* **2020**, *8*, 54776–54788. [CrossRef]
51. Ross, P.; Luckow, A. EdgeInsight: Characterizing and Modeling the Performance of Machine Learning Inference on the Edge and Cloud. In Proceedings of the IEEE International Conference on Big Data, Los Angeles, CA, USA, 9–12 December 2019; pp. 1897–1906.
52. Raileanu, S.; Borangiu, T.; Morariu, O.; Iacob, I. Edge Computing in Industrial IoT Framework for Cloud-based Manufacturing Control. In Proceedings of the IEEE 22nd International Conference on System Theory, Control and Computing (ICSTCC), Sinaia, Romania, 10–12 October 2018; pp. 261–266.
53. Sebbio, S.; Morabito, G.; Catalfamo, A.; Carnevale, L.; Fazio, M. Federated Learning on Raspberry Pi 4: A Comprehensive Power Consumption Analysis. In Proceedings of the IEEE/ACM 16th International Conference on Utility and Cloud Computing, Taormina, Italy, 4–7 December 2023; pp. 1–6.
54. Anand, A.; Goel, S.; Panesar, G.S. Energy-Efficient Edge Computing Architectures for AI Workloads: A Comparative Analysis in Cloud-Driven Environments. In Proceedings of the 2024 International Conference on Emerging Innovations and Advanced Computing (INNOCOMP), Sonipat, India, 25–26 May 2024; pp. 614–620.
55. Abdoulabbas, T.E.; Mahmoud, S.M. Power consumption and energy management for edge computing: State of the art. *Telkomnika Telecommun. Comput. Electron. Control* **2023**, *21*, 836–845. [CrossRef]