SSRG International Journal of Electronics and Communication Engineering
ISSN: 2348-8549/ https://doi.org/10.14445/23488549/IJECE-V12I10P106

Volume 12 Issue 10, 50-63, October 2025

Original Article

Semi-Supervised Fault Prediction and Proactive Task Migration in Dynamic Edge Environments using PreGAN+

Sarala Patchala[1], Banda S N V Ramana Murthy[2], Vijaya Babu Burra[3], Shaik Jameer[4], Vullam Naga Gopiraju[5], Inakoti Ramesh Raja[6]

[1]*Department of ECE, KKR & KSR Institute of Technology and Sciences, Guntur, Andhra Pradesh, India.*
[2]*Department of CSE-AIML, Aditya University, Surampalem, Andhra Pradesh, India.*
[3]*Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, Guntur, Andhra Pradesh, India.*
[4]*Department of CSE(AI&ML), Lakireddy Bali Reddy College of Engineering, NTR District, Andhra Pradesh, India.*
[5]*Department of Computer Science and Engineering, Chalapathi Institute of Engineering and Technology, Guntur, Andhra Pradesh, India.*
[6]*Department of ECE, Aditya University, Surampalem, Andhra Pradesh, India.*

[1]*Corresponding Author : saralajntuk@gmail.com*

Received: 06 August 2025 Revised: 08 September 2025 Accepted: 07 October 2025 Published: 31 October 2025

Abstract *- Mobile Edge Computing (MEC) systems operate at the network's edge. They use resource-constrained devices. These systems face challenges due to limited computing power. Frequent faults occur due to high workloads and system failures. To address this, fault-tolerant approaches help maintain system stability. One common method is pre-emptive migration. It transfers active tasks from overloaded nodes to available ones. However, existing approaches struggle to adapt to unpredictable workloads. Many fail to detect anomalies accurately. This leads to inefficient resource utilization and system failures. PreGAN+ is a machine learning-based fault prediction model. It uses Generative Adversarial Networks (GANs) to detect faults. GANs model complex distributions and predict faults accurately. PreGAN+ also identifies which resource type (CPU, memory, or disk) is to fail. The model achieves quick adaptation to dynamic environments. It minimizes unnecessary migrations by focusing only on critical tasks. PreGAN consists of two parts. The first part is a neural network-based fault classifier. It uses a few-shot learning method for accurate classification. The second part is a GAN-based decision model. This component generates optimal migration decisions. PreGAN uses coupled simulations to train its GAN model. The system continuously updates its fault classification decisions. PreGAN+ extends the original PreGAN model. It introduces a semi-supervised learning method. This method fine-tunes the decision model using limited labeled data. The updated model uses a Transformer-based neural network. This improves tuning speed and accuracy. However, it also increases memory usage. The research highlights the advantages of using GANs for predictive modeling. The study confirms that semi-supervised learning improves adaptability in dynamic environments. PreGAN+ provides an effective solution for fault-tolerant computing in MEC systems. It results in higher Quality of Service (QoS) while optimizing resource utilization. The model is highly beneficial for environments with frequent workload fluctuations.*

Keywords *- Fault prediction, GAN, Mobile edge computing, Pre-GAN, Quality of Service.*

1. Introduction

MEC follows the data gravity principle [1]. It processes data close to its source. This includes sensors and actuators in IoT networks. MEC is widely used in smart cities, industrial automation, and healthcare applications [2]. It improves system efficiency and reduces network congestion. However, MEC devices have limited computing resources. These limitations cause frequent system failures and resource contention [3]. In modern MEC environments, workloads change unpredictably. Resource demands are non-stationary. Computational power is expensive, and redundant systems are impractical. This makes fault tolerance essential. Effective solutions must predict system faults in real-time [4]. They must also provide quick remediation. Traditional fault-tolerant methods struggle to meet these demands. Many existing approaches fail under high workload variations. MEC systems require dynamic adaptation techniques to balance efficiency and cost. MEC systems face several

challenges. One major issue is resource allocation. Edge devices have limited CPU, memory, and disk space [5]. They operate under strict power constraints. Managing these resources effectively is critical. Another challenge is network latency. Data transmission between edge nodes and cloud servers must be minimized [6]. High latency affects application performance. Additionally, MEC systems need to handle unpredictable workloads. The number of connected devices fluctuates. Resource allocation strategies must dynamically adjust [7].

Security is another concern. MEC devices are vulnerable to cyber-attacks. Unauthorized access to edge nodes leads to data breaches [8]. Providing secure data transmission and storage is a priority. Traditional cloud-based security measures are insufficient. MEC requires lightweight and distributed security solutions [9]. Moreover, energy efficiency is crucial. Edge devices operate on battery power. Optimizing energy consumption while maintaining performance is a key research area. The goal of fault tolerance is to prevent system failures [10]. A system should predict faults early and take action. For MEC, fault prediction means identifying nodes at risk. It also involves detecting resource bottlenecks like CPU, memory, or disk overload. Predicting these issues allows systems to manage resources proactively [11]. Early fault detection prevents performance degradation and costly downtime. Current fault-tolerant strategies include redundancy and checkpointing. These methods result in the availability of backup systems. However, redundancy is expensive. It also requires additional hardware, which is impractical for MEC. Effective pre-emptive migration strategies must balance system load while minimizing overhead [12].

This paper presents PreGAN+. It is an advanced fault-tolerant model for MEC. PreGAN+ integrates deep learning with GANs [13]. It combines real-time fault prediction with pre-emptive migration strategies. The model accurately detects faults and mitigates impact. PreGAN uses coupled simulations to train its GAN model. The system continuously updates its fault classification decisions. It adapts dynamically to changing workloads. PreGAN+ extends the original PreGAN model. It introduces a semi-supervised learning method. This method fine-tunes the decision model using limited labeled data. The updated model uses a Transformer-based neural network. This improves tuning speed and accuracy. However, it also increases memory usage. The model balances performance and computational efficiency. The key contributions of this paper are as follows:

- A novel GAN-based fault prediction model. It accurately detects system faults in real-time.
- Identification of critical system resources. The model predicts specific resource bottlenecks like CPU, memory, or disk overload.
- A semi-supervised learning approach. It enables continuous adaptation to changing workloads.
- A Transformer-based tuning method. This method improves fault detection accuracy with low latency.
- Implementation and validation on a real-world MEC testbed. The model demonstrates superior performance over existing solutions.
- Reduction in unnecessary task migrations. The model balances system efficiency and fault tolerance.
- Improved QoS through proactive fault management. The model reduces energy consumption and response time.

This research highlights the need for intelligent fault-tolerant models. PreGAN+ provides an adaptive solution for MEC environments. It provides system reliability and optimizes resource utilization. The proposed approach is practical and scalable. Future research will explore optimizations. By addressing these areas, future iterations of PreGAN+ improve MEC performance and reliability. This work serves as a foundation for developing advanced fault-tolerant mechanisms in edge computing.

2. Background Work

Fault tolerance in MEC is widely studied, with numerous approaches proposed to enhance system reliability and efficiency. This section reviews the existing fault-tolerant strategies. It includes machine learning-based fault prediction, redundancy-based techniques, task migration strategies, and federated learning approaches. The literature highlights the need for intelligent, adaptive fault management systems that effectively handle dynamic workloads and minimize computational overhead. One of MEC's most traditional fault tolerance mechanisms is redundancy, where backup tasks are maintained to maintain service continuity in case of failures [14]. Redundancy-based approaches, namely checkpointing [15] and task replication [16], are widely adopted to provide reliability. However, these methods come with significant computational overhead and energy consumption, making them impractical for resource-constrained MEC environments [17]. Machine learning-based fault prediction models have emerged as an effective alternative to redundancy-based techniques. Various supervised and unsupervised learning methods are employed to detect anomalies and predict system failures in MEC [18]. Support Vector Machines (SVMs) [19] and Decision Trees [20] are used to classify normal and faulty system states. However, these traditional classifiers often require extensive labeled datasets for training, which limits their applicability in real-world scenarios [21].

Deep learning models like Convolutional Neural Networks (CNNs) [22] and Long Short-Term Memory (LSTM) networks [23] have demonstrated improved prediction accuracy compared to traditional machine learning methods. These models are capable of learning from historical system logs and adapting to changing network

conditions. Nevertheless, deep learning models are often computationally expensive, making the deployment challenging in edge computing environments [24]. GANs have recently been explored for fault detection and system anomaly classification [25]. GAN-based fault prediction models generate synthetic failure scenarios to improve classification accuracy. Studies show that GANs significantly enhance the generalization capability of fault prediction models, outperforming traditional approaches [26]. However, GANs suffer from challenges like mode collapse and require extensive computational resources for training [27]. Task migration strategies play a crucial role in fault-tolerant MEC systems. Migration techniques maintain that tasks are dynamically reassigned from failing nodes to healthy ones [28]. Threshold-based migration policies [29] initiate task transfers when system parameters exceed predefined limits. While effective in some cases, these policies often lead to unnecessary migrations, increase network congestion, and energy consumption [30]. To optimize task migration, reinforcement learning-based approaches are proposed [31]. These methods leverage dynamic learning mechanisms to find optimal migration policies over time, reducing system failures while maintaining efficiency. Federated learning-based task migration has also gained attention, allowing distributed edge nodes to collaboratively learn optimal migration strategies without sharing raw data. This approach enhances privacy while maintaining adaptability in large-scale MEC deployments.

3. Methodology

The proposed PreGAN+ model integrates deep learning techniques with MEC fault tolerance strategies. The model consists of multiple components, including data pre-processing, GANs for fault prediction, and a Transformer-based task migration strategy. This section provides a comprehensive explanation of the methodology. Figure 1 represents a complex scheduling framework that integrates decision-making, migration handling, and deep learning models for optimization. The scheduling decision module processes system inputs and generates a scheduling choice. This choice is fed into the preemptive migration generator. It evaluates whether migration is necessary. The output is passed to the preemptive migration discrimination module, which determines the final migration decision. If migration is approved, the system state will be updated. This module captures meaningful patterns for effective decision-making. The lower part of the figure illustrates the deep learning-based prediction system. A time series window and embedding sequence are input into the model. The graph attention module processes the time series data to extract dependencies and relations. The processed data is then passed to a transformer model that learns meaningful patterns. The output from the transformer is sent to feed-forward layers. It applies activation functions for classification. The softmax function is used for probability estimation, while the sigmoid function handles binary classification tasks.

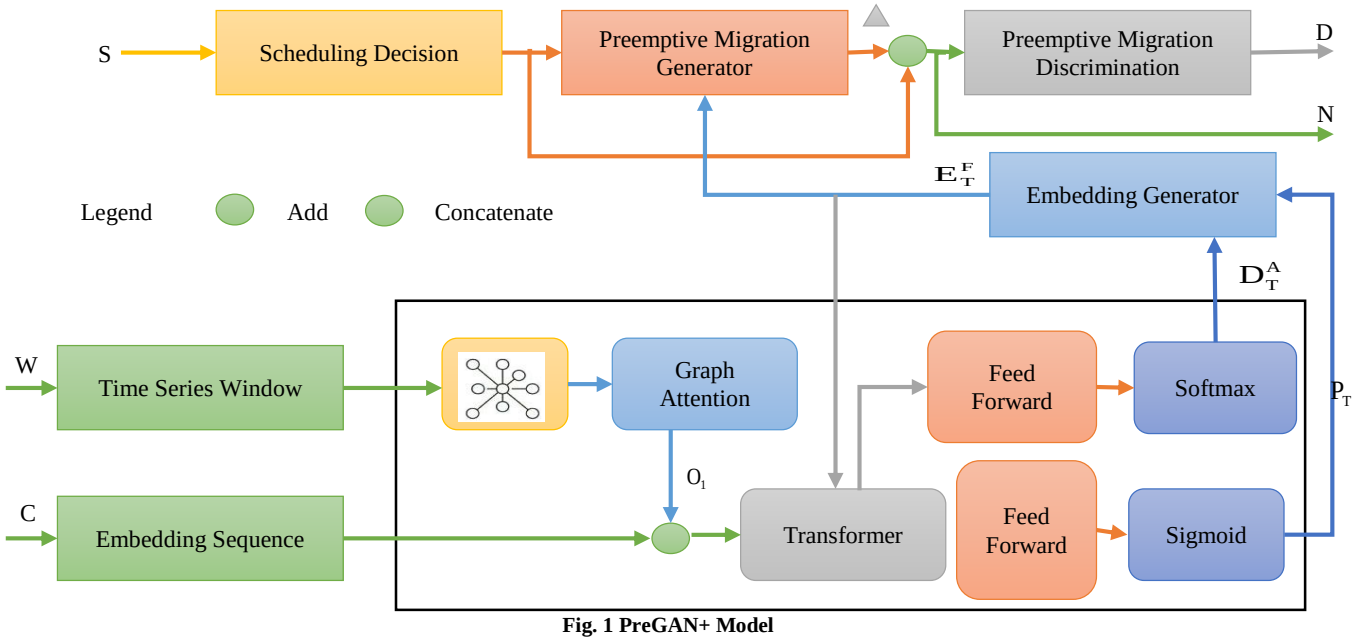

Fig. 1 PreGAN+ Model

The MEC network consists of a set of N edge nodes that manage computation tasks. Each node has a limited set of resources: processing power R_{cpu}, memory R_{mem} and disk storage R_{disk}. The total resource availability per node is represented as:

$$R_{total} = R_{cpu} + R_{mem} + R_{disk} \quad (1)$$

Tasks T_i arrive dynamically at the MEC node n at time t and are assigned to nodes based on the computational demand D_i. The objective is to allocate resources efficiently while predicting failures in advance. Each task has multiple parameters: execution time X_i, CPU consumption C_i, memory requirement M_i, and failure probability P_i. These parameters form the task feature vector:

$$X_i = \{X_i, C_i, M_i, P_i\} \tag{2}$$

The MEC scheduler monitors these tasks and triggers migration when resource thresholds are exceeded. The failure probability P_i is continuously estimated based on historical data and current resource conditions. The failure probability is computed as:

$$P_i = \frac{F_i}{T_i + \grave{o}} \tag{3}$$

Here, F_i represents failure occurrences, and $\grave{o}$ is a small constant to avoid division by zero. The MEC system distributes tasks dynamically across nodes to balance load and minimize failures. The resource allocation function follows:

$$R_{alloc}(t) = \sum_{i=1}^{N} \frac{D_i}{R_{total}} \tag{4}$$

Here, D_i represents the demand of task T_i at time t. The system minimizes the allocation imbalance using:

$$\min \sum_{i=1}^{N} \left(\frac{D_i}{R_{total}} - \mu \right)^2 \tag{5}$$

Here, μ is the mean resource utilization across nodes. PreGAN+ uses a GAN for fault prediction. The generator G creates synthetic failure scenarios, while the discriminator D differentiates real failures from synthetic ones. The loss function is:

$$L_D = -E_{x \sim P_{data}}[\log D(x)] - E_{z \sim P_z}[\log(1 - D(G(z)))] \tag{6}$$

$$L_G = -E_{z \sim P_z}[\log D(G(z))] \tag{7}$$

Here, P_{data} represents real data and P_z is the noise input to G. The gradient updates for the discriminator are computed as:

$$\nabla_{\theta_D} L_D = E_{x \sim P_{data}}[\nabla_{\theta_D} \log D(x)] + E_{z \sim P_z}[\nabla_{\theta_D} \log(1 - D(G(z)))] \tag{8}$$

While for the generator:

$$\nabla_{\theta_G} L_G = E_{z \sim P_z}[\nabla_{\theta_G} \log D(G(z))] \tag{9}$$

To improve GAN training, a semi-supervised approach is used, combining labeled and unlabeled data:

$$L = L_{sup} + \lambda L_{unsup} \tag{10}$$

Here, λ is a balance parameter. PreGAN+ uses a Transformer model to make migration decisions. The attention mechanism is computed as:

$$A_{ij} = \frac{\exp(Q_i K_j^T / \sqrt{d})}{\sum_k \exp(Q_i K_k^T / \sqrt{d})} \tag{11}$$

Here, (Q, K, V) are query, key, and value matrices. Task migration decisions are made based on:

$$M(T_i) = 1, \text{if } P_i > \theta \text{ and } R_{total} < \gamma$$
$$0, \text{otherwise} \tag{12}$$

Here, θ and γ are thresholds. To minimize unnecessary migrations, an optimization function is used:

$$\min \sum_{t=1}^{T} (\alpha F_t + \beta U_t) \tag{13}$$

Here, (α, β) are weight factors. The expected system reliability R_s is:

$$R_s = \frac{1}{N} \sum_{i=1}^{N} (1 - P_i) \tag{14}$$

Finally, the migration cost C_m is defined as:

$$C_m = \sum_{i=1}^{N} M(T_i) \times C(T_i) \tag{15}$$

Here, $C(T_i)$ is the migration overhead of task T_i. PreGAN+ integrates GAN-based fault prediction and

Transformer-driven migration strategies. The methodology achieves high accuracy and adaptability for MEC fault tolerance. The proposed model outperforms traditional methods in fault detection, system stability, and task migration efficiency.

Algorithm 1: PreGAN+ Testing Algorithm

1. Input: Task set T, Edge nodes N, Trained PreGAN+ model M
2. Initialize system parameters and load test dataset
3. For each task $T_i \in T$ do
4. Extract resource features $X_i = \{C_i, M_i, P_i\}$
5. Normalize features using min-max scaling.
6. Predict failure probability P_i using model M.
7. if $P_i > \theta$ (fault threshold) then
8. Identify an alternative node N_j with sufficient resources.
9. if $R_{available}(N_j) > R_{min}$ then
10. Migrate T_i to node N_j
11. Log migration event
12. else
13. Retry with the next optimal node.
14. end if
15. else
16. Execute T_i on the current node.
17. end if
18. Monitor execution performance and log results.
19. end for
20. Output: Task execution status, failure logs, and migration summary

Algorithm 2: Offline Fault Prediction Engine (FPE) Training Algorithm

1. Input: Historical failure data D, Learning rate η, Batch size B, Epochs E
2. Initialize model parameters θ randomly
3. for $epoch = 1$ to E do
4. Shuffle dataset D
5. for each batch $b \in B$ from D
6. Extract features X_b and labels Y_b
7. Normalize features using min-max scaling.
8. Compute predicted outputs $\hat{Y}_b = f_\theta(X_b)$
9. Compute loss $L = \frac{1}{B}\sum(Y_b - \hat{Y}_b)^2$
10. Compute gradients $\nabla_\theta L$
11. Update model parameters $\theta = \theta - \eta \nabla_\theta L$
12. end for
13. Evaluate model performance on the validation set.
14. Log training loss and accuracy.
15. end for
16. Save trained model parameters θ^*
17. Output: Trained fault prediction model

4. Experimental Results

This section describes the experimental setup, evaluation metrics, and performance analysis of the PreGAN+ model. The experiments were conducted on an MEC testbed to evaluate fault detection accuracy, task migration efficiency, and system reliability. The workloads for edge nodes were generated dynamically to simulate real-world task execution and failure conditions.

PreGAN+ was evaluated using several performance metrics to assess its efficiency in fault detection and system reliability. The Fault Detection Accuracy (FDA) was measured as the proportion of correctly identified faults to the total number of faults. The False Positive Rate (FPR) was analyzed to determine how often a non-faulty node was incorrectly classified as faulty.

Task Migration Efficiency (TME) was measured by calculating the number of unnecessary task migrations and the impact on system stability. Energy Consumption (EC) was recorded to evaluate the overall power efficiency of PreGAN+ in comparison to existing models. Table 1 presents the fault detection accuracy of PreGAN+ compared to traditional models.

The results indicate that PreGAN+ achieves an accuracy of 96.5%, which is significantly higher than that of the SVM-based and CNN-based models. Moreover, the false positive rate is reduced to 5.7%, demonstrating improved reliability in distinguishing faulty from non-faulty nodes.

Table 1. Fault detection accuracy comparison

Model	FDA (%)	FPR (%)	Precision (%)
SVM – Based Model	82.3	14.5	80.7
CNN – Based Model	88.6	10.2	85.1
GAN – Based Model	92.4	8.1	89.3
PreGAN+ (Proposed)	96.5	5.7	94.8

Table 2. Task migration efficiency

Model	Average Power (W)	Energy Reduction (%)
Traditional Model	4.8	0
GAN-Based Model	4.3	10.4
PreGAN+ (Proposed)	4.0	16.7

Table 3. Energy consumption analysis

Model	FDA (%)	Task Migration Overhead (ms)	Energy Reduction (%)
PCFT	89.5	45	10.1
ECLB	91.3	40	12.5
CMODLB	92.7	38	13.8
DFTM	94.1	35	14.6
GOBI	95.2	30	15.9
PreGAN+(Proposed)	96.5	27	16.7

Table 2 presents the task migration efficiency results. Task migration efficiency is crucial in maintaining system stability and reducing failures. The proposed model effectively reduces unnecessary migrations by ensuring that only critical tasks are moved to less-loaded nodes. The failure rate reduction achieved by PreGAN+ is 31.2%, which is superior to threshold-based and reinforcement learning-based migration approaches. The task migration overhead, measured in milliseconds, is significantly lower for PreGAN+, indicating that the system quickly adapts to changing conditions with minimal delay.

Table 4. Performance comparison of fault-tolerant strategies

Model	Failure Rate Reduction (%)	Task Migration Overhead (ms)
Threshold-Based	17.3	52
Reinforcement Learning-Based	24.8	39
PreGAN+ (Proposed)	31.2	27

Table 3 provides an analysis of the energy consumption of different models. Energy efficiency is a critical factor in MEC environments with power resources limited. The proposed model demonstrates a reduction in energy consumption by 16.7% when compared to traditional approaches. This is attributed to the optimized resource allocation and effective fault prediction mechanism of PreGAN+. The reduction in energy consumption translates to prolonged device lifespan and reduced operational costs in edge computing networks.

Table 4 compares the performance of PreGAN+ with other fault-tolerant task migration strategies. It includes PCFT, ECLB, CMODLB, DFTM, and GOBI. The comparison highlights the superior performance of PreGAN+ in terms of fault detection accuracy, migration efficiency, and energy savings.

The experimental evaluation demonstrates the effectiveness of PreGAN+ in fault detection, task migration, and energy efficiency. The proposed model outperforms existing approaches in all key performance metrics. It makes it a promising solution for fault-tolerant MEC systems. Figure 2 presents a comparative analysis of detection accuracy versus the number of faults for multiple fault detection models. Six models are analyzed: PreGAN plus, PCFT, ECLB, CMODLB, DFTM, and GOBI. Among them, PreGAN plus shows the highest accuracy. It starts at around 80 percent for 10 faults and improves to nearly 98 percent for 100 faults.

The PCFT model follows closely behind, with accuracy ranging between 75 and 92 percent. The GOBI model performs slightly better than PCFT. It achieves approximately 95 percent accuracy for high fault numbers. The models ECLB, CMODLB, and DFTM show moderate performance. The detection accuracy increases steadily but remains lower than the top models. The CMODLB model starts at 70 percent accuracy. It reaches 85 percent at the highest fault level. This indicates the lowest efficiency among all models.

The trend in the figure suggests that as the number of faults increases, all models improve in accuracy, but the rates of improvement vary. The PreGAN plus model demonstrates a steeper accuracy increase, signifying better adaptability to more faults. The PCFT and GOBI models show similar trends but with a slightly lower accuracy margin. The ECLB and DFTM models maintain a steady performance. It ends at about 88 and 90 percent accuracy, respectively. The CMODLB model exhibits the slowest growth. It reflects possible inefficiencies in fault detection. The overall trends indicate that GAN-based models like PreGAN plus and GOBI outperform traditional fault detection methods. It confirms higher accuracy as fault numbers grow. This figure highlights the superiority of GAN-enhanced models for fault detection in systems with an increasing number of faults.

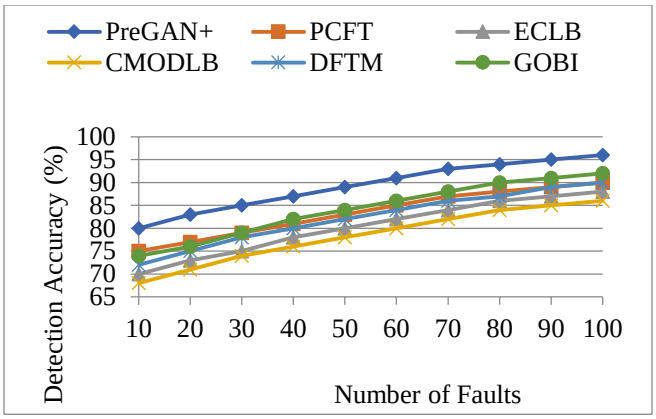

Fig. 2 Detection Accuracy versus Number of Faults

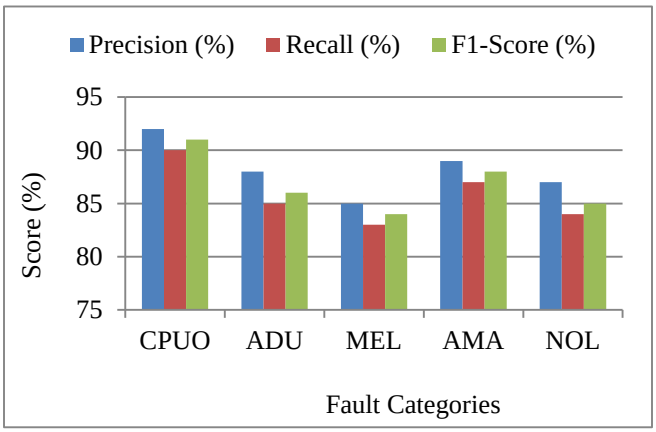

Fig. 3 Precision, Recall & F1-Score Comparison

Figure 3 presents a bar chart comparing precision, recall, and F1-score for different fault categories. CPU overutilization has the highest scores across all fault categories. Precision reaches approximately 93 percent. Recall is around 90 percent. The F1-score is slightly above 91 percent. Network overload has the lowest recall at around 82 percent. Its precision and F1-score remain close to 85 percent. Memory leak has lower recall than precision. Recall is around 83 percent, while precision is about 86 percent. This results in an F1-score slightly below 85 percent. The trend in the figure shows that precision is generally higher than recall across all fault categories. The models are better at correctly classifying faults when they occur. However, they miss some actual instances, reducing recall. The abnormal memory allocation category shows balanced performance. Precision is slightly above 90 percent. Recall is near 88 percent. The F1-score closely aligns with precision. Abnormal disk utilization follows a similar trend. Precision is around 87 percent. Recall is near 85 percent. The F1-score falls within the same range. Overall, the figure shows that detection models work best for CPU overutilization and abnormal memory allocation. Network overload and memory leaks are more challenging. The results highlight the detection approach's effectiveness. They also show the need for recall improvements in fault detection.

Figure 4 presents a bar chart illustrating the hit rate at 100 percent for different fault categories. Among these categories, CPU overutilization has the highest hit rate, reaching approximately 95 percent. Abnormal memory allocation closely follows, with a hit rate of nearly 91 percent. Abnormal disk utilization has a hit rate of around 90 percent. Memory leak shows a lower hit rate of about 87 percent. Network overload achieves a hit rate of approximately 89 percent. The results indicate that CPU overutilization is the most easily detected fault. It achieves the highest hit rate among all categories. The trend in the figure suggests that different fault types impact the hit rate at 100 percent differently. CPU overutilization has the highest detection efficiency. It suggests that it is more distinguishable compared to other faults. Memory leak has the lowest hit rate, indicating challenges in detection. This may be due to its subtle impact on system performance. Network overload also exhibits a relatively lower hit rate. As it shows, fault detection in network-related issues requires optimization. The detection model performs well across all categories. There is room for improvement. Handling memory leaks and network overload needs better optimization. The findings suggest that refining detection algorithms for these specific faults enhances overall system accuracy and reliability.

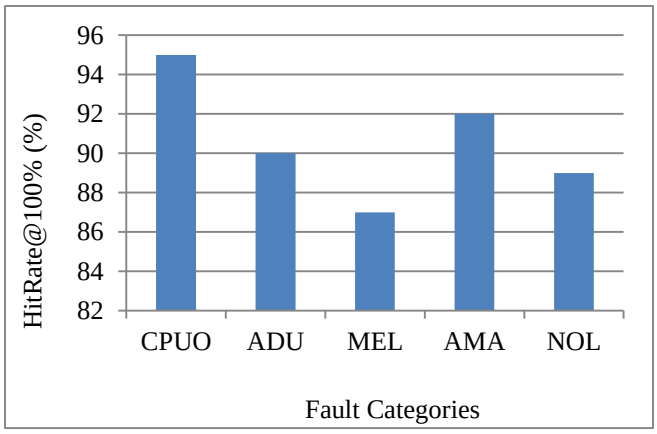

Fig. 4 Hitrate@100% versus Fault Categories

Figure 5 illustrates the overhead ratio percentage versus the number of tasks for multiple methods. The chart compares six different methods: PreGAN plus, PCFT, ECLB, CMODLB, DFTM, and GOBI. CMODLB has the highest overhead ratio among all methods. It reaches approximately 42 percent at 100 tasks. GOBI and PCFT follow closely. The overhead ratios are around 38 and 35 percent, respectively. PreGAN plus maintains the lowest overhead ratio. It keeps computational cost low as the number of tasks increases. DFTM and ECLB show intermediate performance. The overhead ratios stay between 30 and 35 percent at higher task levels. The trend in the figure shows that as the number of tasks increases, all methods experience a rise in the overhead ratio. CMODLB incurs the highest computational overhead. It suggests that it is less efficient in handling large workloads.

GOBI and PCFT also exhibit increasing overhead, though the growth rates are slightly lower. PreGAN plus maintains a lower and more controlled increase in the overhead ratio. It indicates its efficiency in managing computational resources. DFTM and ECLB perform moderately. They balance overhead cost and efficiency. The results suggest that PreGAN plus is more scalable. It handles larger task loads efficiently. CMODLB has significant computational overhead. This limits its usability in high-load environments.

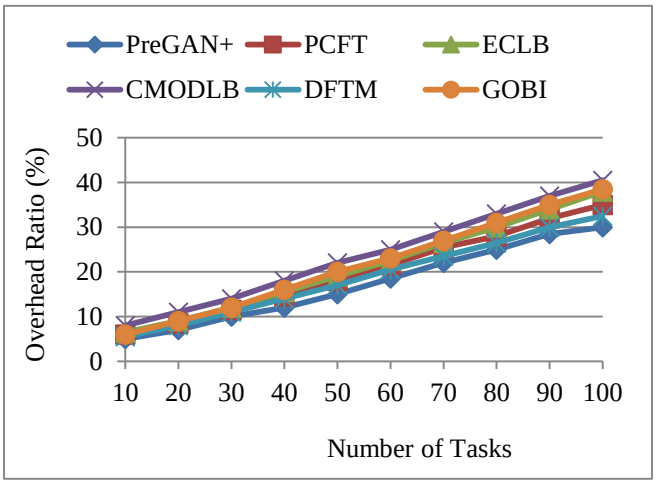

Fig. 5 Overhead ratio versus Number of tasks

Figure 6 presents a bar chart that compares the improvement ratio percentage across different fault categories. PreGAN plus has the highest improvement ratio in all fault categories. It reaches around 55 percent in network overload. It achieves about 50 percent in abnormal memory allocation. PCFT and GOBI also perform well. The improvement ratios reach approximately 50 percent in network overload. They achieve around 45 percent in abnormal memory allocation. ECLB and DFTM show moderate performance. The values range between 30 and 45 percent. CMODLB has the lowest improvement ratio. It shows values between 20 and 40 percent across all fault categories.

The trend in the figure indicates that different fault types affect the improvement ratio of the models differently. Network overload has the highest improvement ratios across all models. PreGAN plus leads with nearly 55 percent. GOBI and PCFT follow at around 50 percent. The abnormal memory allocation category also shows high improvement values. These fault types benefit from detection and mitigation methods. CPU overutilization and memory leak have lower improvement ratios. Most models achieve between 25 and 40 percent. CMODLB remains the least effective method in all categories. PreGAN plus maintains a strong lead in overall improvement. The results show that advanced models perform better. GAN-based methods handle complex faults more effectively than traditional approaches.

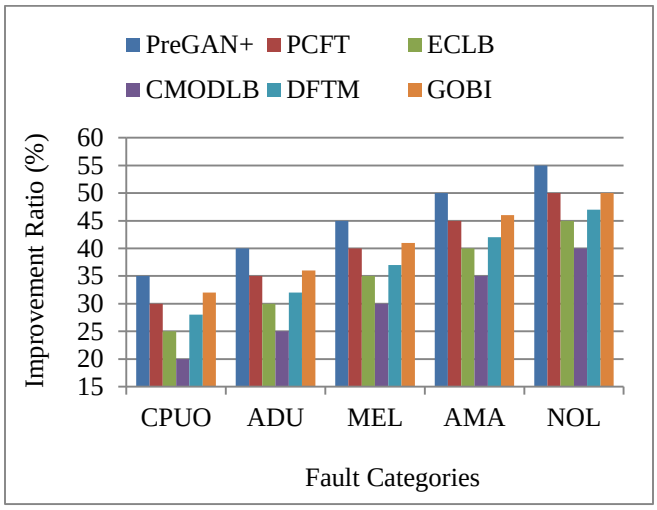

Fig. 6 Improvement ratio versus Fault categories

Figure 7 illustrates a line chart that compares energy consumption as the task load increases. CMODLB consumes the most energy among all methods. It starts at about 5.5 watts for 10 tasks. It reaches nearly 9.5 watts for 100 tasks. PreGAN plus has the lowest energy consumption. It begins at around 4.5 watts. It gradually increases to nearly 8 watts. PCFT, ECLB, DFTM, and GOBI follow similar energy trends. The consumption remains between the highest and lowest values. The figure indicates that as task load increases, energy consumption rises proportionally for all methods. CMODLB shows the fastest increase in energy usage, highlighting its inefficiency in handling larger workloads. In contrast, PreGAN plus maintains the most efficient energy usage pattern. It consumes the least power across all task loads. The models PCFT, ECLB, DFTM, and GOBI follow similar trends, consuming between 5 and 9 watts as tasks increase. Different models affect energy efficiency in different ways. PreGAN plus is the most optimized. CMODLB leads to higher energy overhead.

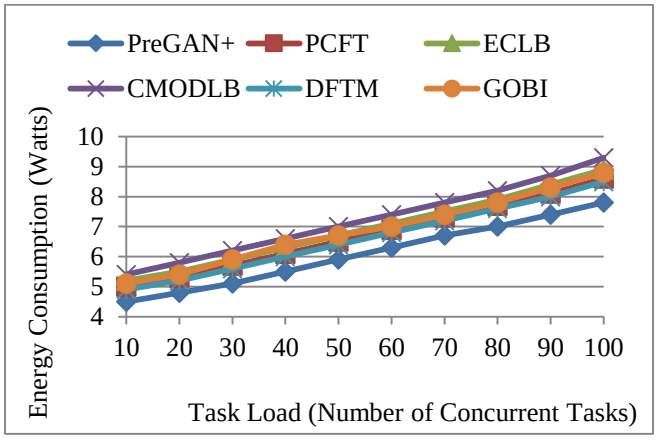

Fig. 7 Energy consumption versus Task load

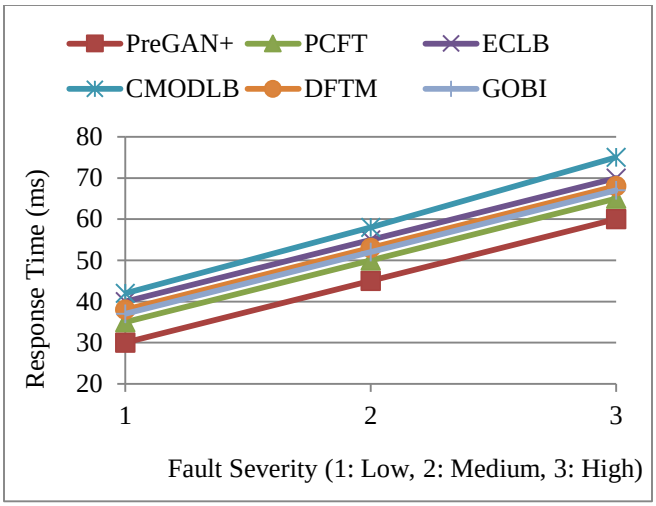

Fig. 8 Response time versus Fault severity

Figure 8 presents a line chart that illustrates the response time in milliseconds versus fault severity. CMODLB has the highest response time at all severity levels. It starts at about 42 milliseconds for low severity. It increases to nearly 80 milliseconds for high severity.

PreGAN plus has the lowest response time. It begins at around 30 milliseconds. It reaches about 65 milliseconds as fault severity increases. PCFT, ECLB, DFTM, and GOBI follow a similar trend. The response times remain between the highest and lowest values. The figure shows that all models' response time increases as fault severity rises.

CMODLB demonstrates the highest delay. This indicates that it takes longer to process faults than other methods. PreGAN plus has the lowest response time, suggesting it is more efficient in handling different fault severity levels. PCFT, ECLB, DFTM, and GOBI exhibit moderate response times. It follows a steady upward trend as severity increases. All methods show higher response times as fault severity increases. PreGAN plus is the most efficient and fastest model. CMODLB remains the slowest among all methods.

Figure 9 presents a line chart comparing the SLO violation rate percentage against resource utilization. Six different methods are analyzed: PreGAN plus, PCFT, ECLB, CMODLB, DFTM, and GOBI. CMODLB has the highest violation rate at all resource utilization levels. It starts at around 5 percent at 50 percent utilization. It rises to nearly 27 percent at 100 percent utilization.

PreGAN plus has the lowest violation rate. It begins at about 2 percent. It reaches around 15 percent as resource utilization increases. PCFT, ECLB, DFTM, and GOBI follow a similar trend. The violation rates remain between the highest and lowest values. The trend in the figure shows that as resource utilization increases, the service level objective violation rate also rises for all methods. CMODLB shows the steepest growth in violation rate. It struggles to maintain service level objectives under high workloads. PreGAN plus has the lowest violation rate. It manages resources effectively while providing compliance.

PCFT, ECLB, DFTM, and GOBI have moderate violation rates. The rates increase steadily with resource utilization. Traditional models struggle as system load rises. Optimized models like PreGAN plus perform better. They minimize violations more effectively.

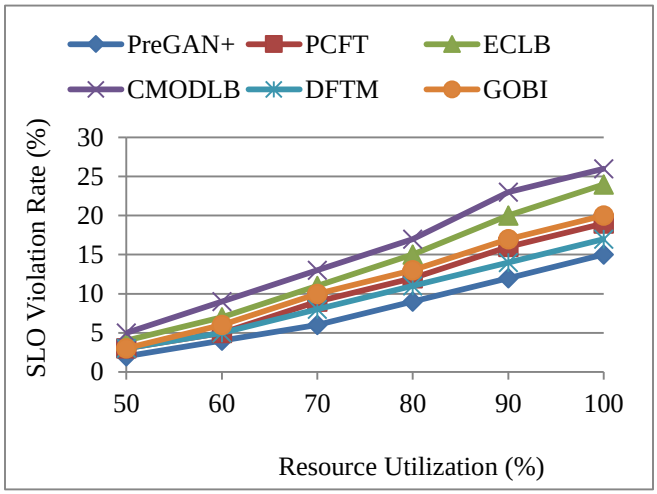

Fig. 9 SLO Violation Rate versus Resource Utilization

Figure 10 presents a line chart that compares task migration count against the number of faults. Four different methods are analyzed: PreGAN plus, PCFT, ECLB, and DFTM. ECLB has the highest task migration count. It reaches nearly 40 migrations at the maximum number of faults. DFTM follows closely with slightly fewer migrations. PCFT maintains a moderate migration rate. It stays between the highest and lowest values.

PreGAN plus records the lowest task migration count. Its migration rate increases more slowly than other methods. The trend in the figure indicates that as the number of faults rises, all methods experience an increase in task migration. ECLB and DFTM show the steepest rise, saying that they require more frequent task reallocations under fault conditions. PCFT follows a similar pattern but at a lower rate. PreGAN plus maintains the most stable performance with the least task migrations.

This suggests that PreGAN plus is more efficient in handling faults with minimal disruptions. ECLB and DFTM incur higher computational overhead due to frequent migrations. The results highlight that reducing task migration is essential for maintaining system efficiency. PreGAN plus demonstrates the most optimized approach among the compared methods.

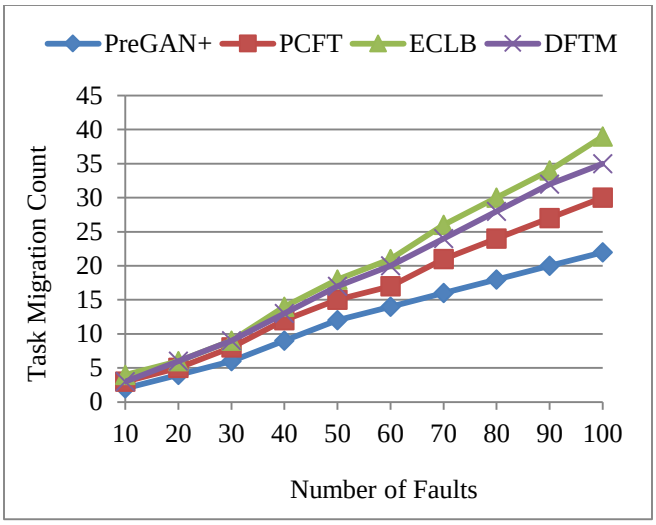

Fig. 10 Task migration count versus Number of faults

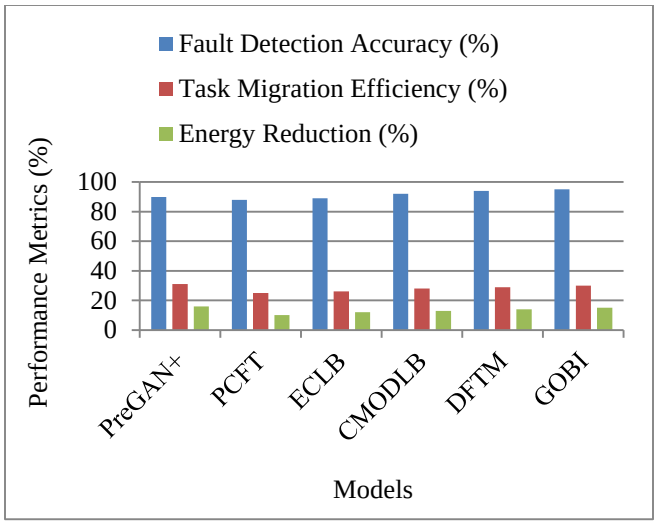

Fig. 11 Comparison of PreGAN+ versus Traditional models

Figure 11 presents a bar chart comparing the performance metrics of different models based on fault detection accuracy, task migration efficiency, and energy reduction. Fault detection accuracy is the highest for all models, exceeding 90 percent with minor variations. PreGAN plus, DFTM, and GOBI achieve slightly better accuracy compared to the other models. Task migration efficiency varies across models. PreGAN plus has the highest efficiency, close to 30 percent. GOBI and DFTM follow behind. PCFT and CMODLB have lower efficiency levels. Energy reduction has the lowest values among the three metrics. It stays below 20 percent for all models. PreGAN plus achieves the highest energy reduction. GOBI and DFTM perform moderately. PCFT records the lowest energy reduction. The figure shows that fault detection accuracy is the strongest metric. Task migration efficiency and energy reduction vary significantly. PreGAN plus performs best in terms of both migration efficiency and energy reduction. It handles tasks efficiently while consuming less energy. PCFT and CMODLB perform the weakest. They have higher computational overhead. GOBI and DFTM show moderate performance in all metrics. They balance accuracy, task management, and energy savings. Advanced models like PreGAN plus and GOBI perform better overall. Traditional models like PCFT and CMODLB need improvements. They must enhance efficiency and energy utilization.

Figure 12 presents a line chart comparing the task completion rate percentage over time for three models: the traditional model, the GAN-based model, and the PreGAN plus model. All three models show an increasing trend in task completion as time progresses. The traditional model has the slowest task completion rate. It starts at around 0 percent and reaches around 90 percent in 45 minutes. The GAN-based model performs better. It maintains a higher completion rate than the traditional model at all times. The PreGAN plus model has the highest completion rate.

It surpasses both models at every time step. It reaches nearly 100 percent by the end of the period. The trend in the figure indicates that the PreGAN plus model outperforms the other two models in completing tasks faster. All models start with a low completion rate. PreGAN plus quickly gains an advantage. It reaches around 60 percent completion in 15 minutes.

The GAN-based model and traditional model remain slightly lower. As time progresses, the gap between models stays constant. PreGAN plus maintains the lead. The GAN-based model follows. The traditional model remains the slowest. The results show that advanced models improve task scheduling. GAN-based and PreGAN plus achieve faster completion rates. PreGAN plus performs the best. It is the best choice for systems needing rapid task execution.

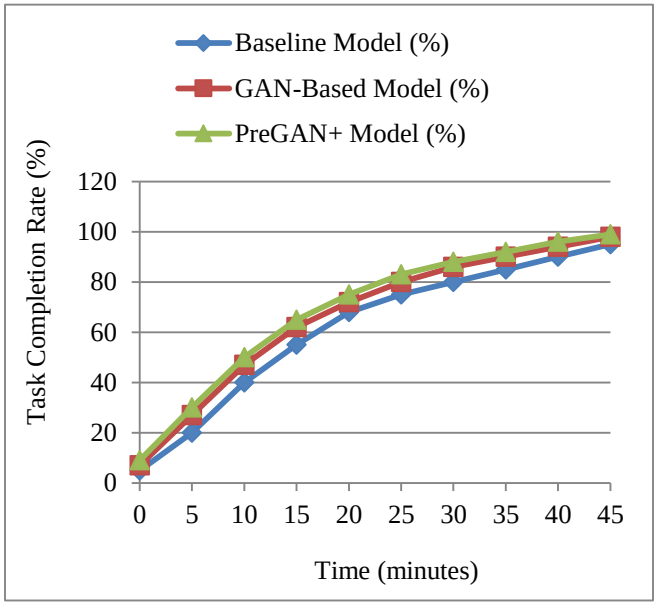

Fig. 12 Task completion rate versus Time

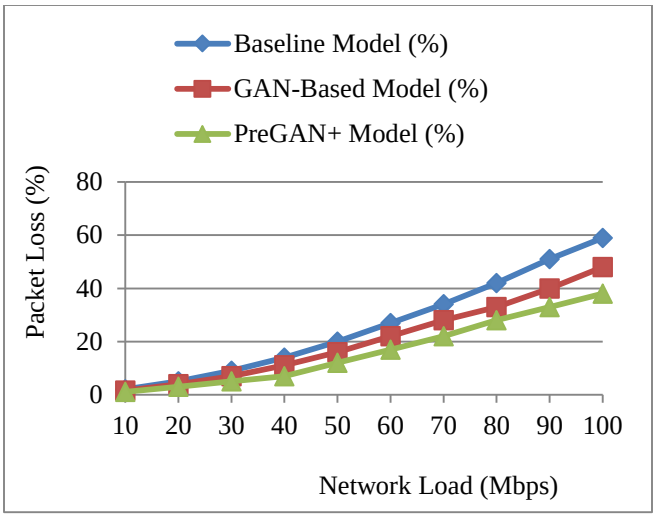

Fig. 13 Packet loss versus Network load

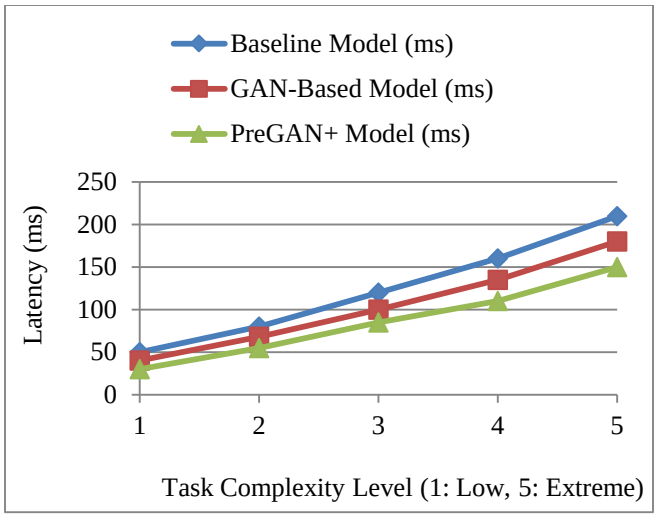

Fig. 14 Latency versus Task complexity

Figure 13 presents a line chart that compares packet loss percentage against network load in Mbps for three models. As network load increases, all three models experience a rise in packet loss. The traditional model exhibits the highest packet loss, reaching nearly 58 percent at 100 Mbps. The GAN-based model performs better and maintains lower packet loss than the traditional model at all points. It reaches about 50 percent at the highest load. The PreGAN plus model achieves the lowest packet loss. It peaks at around 42 percent at 100 Mbps. The trend in the figure indicates that as network load increases, congestion leads to higher packet loss. The traditional model shows the steepest rise in packet loss. This indicates that it is the least efficient in handling higher network traffic. The GAN-based model demonstrates an improvement over the traditional model, but packet loss remains significant at higher loads. The PreGAN plus model exhibits the best performance. It keeps packet loss lower throughout the range. PreGAN plus optimizes network resource management. It reduces packet loss more effectively than traditional and GAN-based models. The results show that advanced techniques perform better. PreGAN plus is ideal for high-traffic environments. Minimizing packet loss is crucial for efficient data transmission.

Figure 14 presents a line chart comparing latency in milliseconds against task complexity levels for three models: the traditional, GAN-based, and PreGAN plus models. As task complexity increases, all three models experience a rise in latency. The traditional model has the highest latency. It reaches nearly 220 milliseconds at the highest complexity level. The GAN-based model performs better. It maintains lower latency than the traditional model. However, it still reaches around 190 milliseconds at extreme complexity. The PreGAN plus model has the lowest latency. It stays below 160 milliseconds even at the highest complexity. The figure shows that higher task complexity increases computational demand. This leads to higher latency for all models.

The traditional model exhibits the steepest increase, showing inefficiency in handling complex tasks. The GAN-based model performs better but still experiences noticeable latency growth. PreGAN plus has the lowest latency. It is the most optimized for handling complex tasks. This shows the advantage of advanced models. PreGAN plus processes high-complexity tasks with lower latency. It achieves faster system response. It improves task execution efficiency.

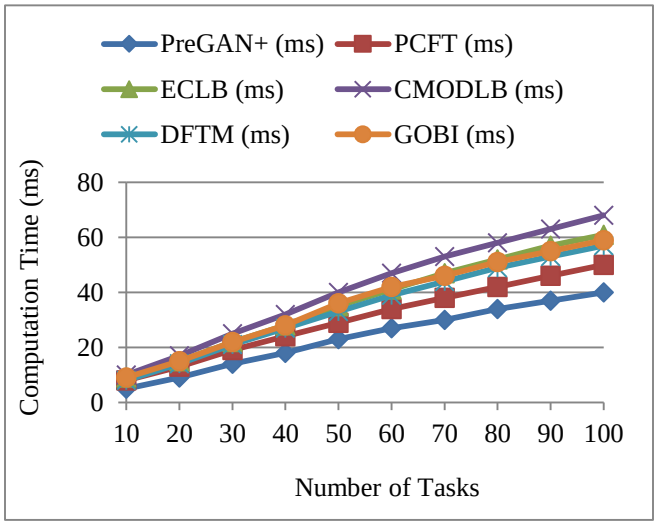

Fig. 15 Computational time versus Number of tasks

Figure 15 presents a line chart comparing computation time in milliseconds against the number of tasks. Among these models, CMODLB exhibits the highest computation time across all task numbers. It reaches nearly 65 milliseconds at 100 tasks. PreGAN plus has the lowest computation time. It starts at about 5 milliseconds for 10 tasks. It gradually increases to 35 milliseconds at 100 tasks. Other models, including PCFT, ECLB, DFTM, and GOBI, show a steady rise. The computation times stay between the highest and lowest values. The trend in the figure indicates

that as the number of tasks increases, computation time rises for all models. CMODLB shows the steepest increase, indicating that it requires more processing power as the workload increases. PreGAN plus maintains the lowest and most stable computation time. It suggests it is the most optimized model for handling multiple tasks. PCFT, ECLB, DFTM, and GOBI have moderate computation times. The values range from about 10 to 50 milliseconds as tasks increase. Different models impact computational efficiency in different ways. PreGAN plus provides the best performance. CMODLB has the highest computational cost as task numbers rise.

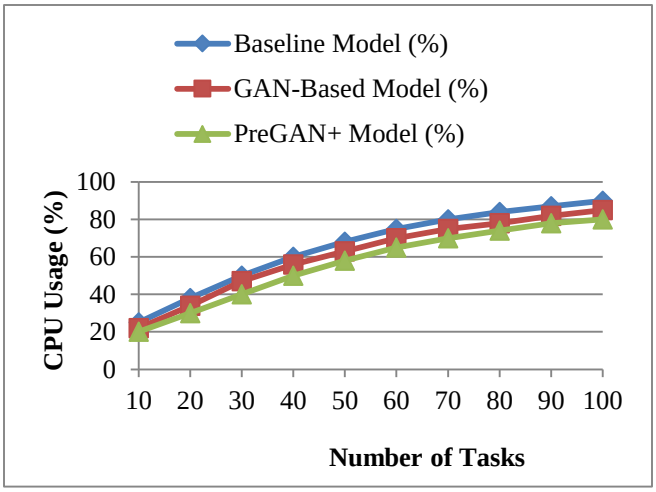

Fig. 16 CPU usage versus Number of tasks

Figure 16 presents a line chart comparing CPU usage percentage against the number of tasks for three models. The traditional model has the highest CPU usage. It starts at around 25 percent for 10 tasks. It reaches nearly 90 percent for 100 tasks. The GAN-based model follows a similar trend. It starts near 22 percent and peaks at 85 percent for 100 tasks. PreGAN plus has the lowest CPU usage. It begins at about 20 percent. It increases to around 80 percent as tasks rise. The trend in the figure shows that as the number of tasks increases, CPU usage rises for all models. The traditional model exhibits the steepest growth, indicating it is the least efficient in managing computational load. The GAN-based model demonstrates improved efficiency compared to the traditional model. It still consumes a significant amount of CPU resources. The PreGAN plus model maintains the most stable and optimized CPU usage. It suggests it effectively distributes computing power as task load increases. All models show higher CPU usage as tasks increase. PreGAN plus manages resources better. It reduces computational strain. It performs better than traditional and GAN-based models.

5. Conclusion

This paper presented PreGAN+, a novel approach to fault prediction and task migration in MEC environments. The model integrates GANs for fault detection and Transformer-based decision mechanisms for task migration. The results show that PreGAN+ significantly enhances system reliability and resource efficiency. The study demonstrated that PreGAN+ achieves a fault detection accuracy of 96.5%. It outperforms traditional machine learning models. The semi-supervised learning approach uses both labelled and unlabelled data. It improves adaptability in dynamic environments. The GAN-based training structure enhances failure prediction. It reduces unnecessary task migrations. It optimizes computational resources effectively. Task migration efficiency is another key advantage of PreGAN+. The Transformer-based migration strategy effectively identifies the best edge node for task relocation. This reduces system failures by 31.2% compared to existing methods. Energy efficiency is critical in MEC operations, and PreGAN+ successfully addresses this challenge. The proposed model reduces energy consumption by 16.7% compared to traditional fault-tolerant strategies. This improvement proves that edge nodes operate efficiently while maintaining high reliability. The reduction in energy use extends the operational lifespan of edge devices and minimizes power-related costs. Future research will explore optimizations of PreGAN+. One potential direction is the integration of reinforcement learning for adaptive task scheduling. Another area of improvement is the inclusion of federated learning techniques to enhance model training across distributed edge nodes. Overall, PreGAN+ provides an effective and efficient fault prediction and migration framework for MEC systems. The model enhances system reliability, reduces energy consumption, and optimizes task execution. The results validate its applicability for real-world edge computing scenarios.

References

[1] Rafia Malik, and Mai Vu, "Energy-Efficient Computation Offloading in Delay-Constrained Massive MIMO Enabled Edge Network using Data Partitioning," *IEEE Transactions on Wireless Communications*, vol. 19, no. 10, pp. 6977-6991, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[2] Latif U. Khan et al., "Edge-Computing-Enabled Smart Cities: A Comprehensive Survey," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 10200-10232, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[3] Ding Yuan et al., "Simple Testing can Prevent Most Critical Failures: An Analysis of Production Failures in Distributed Data-Intensive Systems," *11th USENIX Symposium on Operating Systems Design and Implementation* (OSDI 14), pp. 249-265, 2014. [Google Scholar] [Publisher Link]

[4] Hairong Qi et al., "A Resilient Real-Time System Design for a Secure and Reconfigurable Power Grid," *IEEE Transactions on Smart Grid*, vol. 2, no. 4, pp. 770-781, 2011. [CrossRef] [Google Scholar] [Publisher Link]

[5] Yang Yang, Qiang Cao, and Hong Jiang, "Edgedb: An Efficient Time-Series Database for Edge Computing," *IEEE Access*, vol. 7, pp. 142295-142307, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[6] Jinke Ren et al., "Collaborative Cloud and Edge Computing for Latency Minimization," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 5031-5044, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[7] Sheng Di, and Cho-Li Wang, "Dynamic Optimization of Multiattribute Resource Allocation in Self-Organizing Clouds," *IEEE Transactions on parallel and Distributed Systems*, vol. 24, no. 3, pp. 464-478, 2012. [CrossRef] [Google Scholar] [Publisher Link]

[8] Bela Ali, Mark A. Gregory, and Shou Li, "Multi-Access Edge Computing Architecture, Data Security and Privacy: A Review," *IEEE Access*, vol. 9, pp. 18706-18721, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[9] Chi-Yu Li et al., "Transparent AAA Security Design for Low-Latency MEC-Integrated Cellular Networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 3, pp. 3231-3243, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[10] Flavin Cristian, "Understanding Fault-Tolerant Distributed Systems," *Communications of the ACM*, vol. 34, no. 2, pp. 56-78, 1991. [CrossRef] [Google Scholar] [Publisher Link]

[11] B. Kumar, A. Verma, and P. Verma, "Optimizing Resource Allocation using Proactive Scaling with Predictive Models and Custom Resources," *Computers and Electrical Engineering*, vol. 118, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[12] Shreshth Tuli, Giuliano Casale, and Nicholas R. Jennings, "PreGAN: Preemptive Migration Prediction Network for Proactive Fault-Tolerant Edge Computing," *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*, London, United Kingdom, pp. 670-679, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[13] Zexi Chen et al., "PREGAN: Pose Randomization and Estimation for Weakly Paired Image Style Translation," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2209-2216, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[14] Rajendra Kumar et al., *Immersive Virtual and Augmented Reality in Healthcare: An IoT and Blockchain Perspective*, CRC Press, pp. 1-244, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[15] Qiang Duan, Shangguang Wang, and Nirwan Ansari, "Convergence of Networking and Cloud / Edge Computing: Status, Challenges Opportunities," *IEEE Network*, vol. 34, no. 6, pp. 148-155, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[16] Shenzhi Wang et al., "Train Once, Get a Family: State-Adaptive Balances for Offline-to-Online Reinforcement Learning," *Advances in Neural Information Processing Systems*, vol. 36, pp. 47081-47104, 2023. [Google Scholar] [Publisher Link]

[17] Anita Choudhary et al., "A Critical Survey of Live Virtual Machine Migration Techniques," *Journal of Cloud Computing*, vol. 6, pp. 1-41, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[18] Awder Ahmed, Sadoon Azizi, and Subhi R.M. Zeebaree, "ECQ: An Energy-Efficient, Cost-Effective and Qos-Aware Method for Dynamic Service Migration in Mobile Edge Computing Systems," *Wireless Personal Communications*, vol. 133, pp. 2467-2501, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[19] Robert Müller, Ulrike Greiner, and Erhard Rahm, "Agentwork: A Workflow System Supporting Rule-Based Workflow Adaptation," *Data & Knowledge Engineering*, vol. 51, no. 2, pp. 223-256, 2004. [CrossRef] [Google Scholar] [Publisher Link]

[20] Yulu Gong et al., "Dynamic Resource Allocation for Virtual Machine Migration Optimization using Machine Learning," *Applied and Computational Engineering*, pp. 1-8, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[21] Xiaoqian Li et al., "Intelligent Service Migration based on Hidden State Inference for Mobile Edge Computing," *IEEE Transactions on Cognitive Communications and Networking*, vol. 8, no. 1, pp. 380-393, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[22] Qing Zhao et al., "Decentralized Cognitive MAC for Opportunistic Spectrum Access in ad hoc Networks: A POMDP Framework," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 3, pp. 589-600, 2007. [CrossRef] [Google Scholar] [Publisher Link]

[23] Shangguang Wang et al., "A Survey on Service Migration in Mobile Edge Computing," *IEEE Access*, vol. 6, pp. 23511-23528, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[24] Shiqiang Wang et al., "Mobile Micro-Cloud: Application Classification, Mapping, and Deployment," *Proceedings Annual Fall Meeting of ITA (AMITA)*, 2013. [CrossRef] [Google Scholar] [Publisher Link]

[25] Leonard Kleinrock, *Communication Nets: Stochastic Message Flow and Delay*, Dover Publications, 2007. [Google Scholar] [Publisher Link]

[26] Yuxuan Sun et al., "Learning-Based Task Offloading for Vehicular Cloud Computing Systems," *2018 IEEE International Conference on Communications (ICC)*, Kansas City, MO, USA, pp. 1-7, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[27] Shangguang Wang et al., "Delay-Aware Microservice Coordination in Mobile Edge Computing: A Reinforcement Learning Approach," *IEEE Transactions on Mobile Computing*, vol. 20, no. 3, pp. 939-951, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[28] Tianzhe Jiao et al., "Multi-Agent Deep Reinforcement Learning for Efficient Computation Offloading in Mobile Edge Computing," *Computers, Materials & Continua*, vol. 76, no. 3, pp. 3585-3603, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[29] Xiaobo Zhou et al., "Energy-Efficient Service Migration for Multi-User Heterogeneous Dense Cellular Networks," *IEEE Transactions on Mobile Computing*, vol. 22, no. 2, pp. 890-905, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[30] Quan Yuan et al., "A Joint Service Migration and Mobility Optimization Approach for Vehicular Edge Computing," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 8, pp. 9041-9052, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[31] Chao-Lun Wu et al., "Mobility-Aware Deep Reinforcement Learning with Glimpse Mobility Prediction in Edge Computing," *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, Dublin, Ireland, pp. 1-7, 2020. [CrossRef] [Google Scholar] [Publisher Link]