NTU Journal of Engineering and Technology (2025) 4 (4): 67 - 76
DOI: https://doi.org/10.56286/ntujet.v4i4

IRAQI Academic Scientific Journals

P-ISSN: 2788-9971 E-ISSN: 2788-998X
NTU Journal of Engineering and Technology
Available online at: https://journals.ntu.edu.iq/index.php/NTU-JET/index

Hybrid CNN-LSTM Network for Adaptive QoS Optimization in MQTT-Based IoT Systems

Muamar Almani Jasim
Medical Instrumentation Engineering Department, Technical Engineering College, Northern Technical University, Kirkuk, Iraq.
muamar78@ntu.edu.iq

Article Information

Received: 14-10- 2025,
Revised: 04-11-2025,
Accepted: 05-11-2025,
Published online: 28-12-2025

Corresponding author:
Name: Muamar Almani Jasim
Affiliation: Northern Technical University.
Email: muamar78@ntu.edu.iq

Key Words:
Internet of things,
MQTT protocol,
quality of service,
deep learning,
CNN-LSTM networks.

A B S T R A C T

Internet of Things (IoT) connections depend on the Message Queuing Telemetry Transport (MQTT) protocol; however, it might be difficult to determine the best Quality of Service (QoS) level in dynamic network contexts. Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks are combined in this study's adaptive deep learning framework to optimize MQTT QoS in real time. To depict a variety of IoT circumstances, such as resource limitations, high load, network instability, and regular operations, we created a thorough labeled dataset of 50,000 synthetic samples. In all investigated scenarios, the hybrid CNN-LSTM architecture maintained 71.44% resource efficiency while achieving 92.7% validation accuracy in QoS prediction. Our system showed great confidence in adapting to essential applications (98.83%), low-resource environments (99.78%), and high-load conditions (99.99%). For industrial IoT deployments in smart manufacturing, healthcare monitoring systems, and critical infrastructure management, where dependable communication under fluctuating resource constraints is crucial for operational efficiency and safety, this adaptive QoS optimization framework shows great promise. The suggested architecture greatly improves network performance while preserving reliability by providing a workable solution for autonomous QoS control in resource-constrained IoT installations.

1. Introduction

The Internet of Things (IoT) is growing at an exponential rate, and by 2025, there will probably be tens of billions of connected devices [1]. The Message Queuing Telemetry Transport (MQTT) protocol is now the norm for machine-to-machine communication in Internet of Things ecosystems because it is lightweight and features publish subscribe architecture [2, 3]. MQTT has three Quality of Service (QoS) levels: QoS 0 (at most once), QoS 1 (at least once), and QoS 2 (exactly once). Every level has its own reliability guarantee and resource cost [4].

Choosing the proper QoS level is important for the operation of an IoT system because it has a direct effect on the dependability of message delivery, network latency, energy use, and the use of computational resources [1, 5]. Still, it is hard to pick the optimum QoS in IoT environments that are always changing, with networks that are always changing, devices that are always changing, and applications that have different needs [6, 7]. Standard static QoS settings can't change when things change, which can lead to either too much resource use or less reliability [8].

Recent advancements in deep learning have enabled the forecasting of time-series data and the modeling of complex temporal patterns [9, 10]. In numerous predictive tasks, hybrid architectures integrating Long Short-Term Memory (LSTM) networks for temporal sequence modeling and Convolutional Neural Networks (CNN) for spatial feature extraction have exhibited enhanced performance [11, 12]. These designs are particularly adept at resource management inside the Internet of Things, because optimal decisions are shaped by both historical trends and the present condition of the system [13].

There are still a few things that need to be fixed before machine learning can fully improve the Internet of Things. First, it is hard to get labeled training data for QoS improvement because ground truth labels need a lot of real-world testing or expert knowledge [14]. Second, IoT systems need to work under strict resource limits, hence they need efficient model architectures [15, 16]. Third, solutions must provide real-time adaptation to rapidly changing network conditions while maintaining system stability [17].

This work makes the following contributions to solving these problems:

- Making Complete Labeled Datasets: We create a methodical way to make artificially labeled datasets that accurately show a range of IoT operating scenarios, such as normal operations, high load situations, unreliable networks, and limited resources. Our team of experts.
- The labeling method uses domain knowledge about how resources are used, how reliable a network needs to be, and how good the training data needs to be.
- Hybrid CNN-LSTM Architecture: We build a deep learning model that uses LSTM layers to find tentative dependencies in MQTT dataflow and CNN layers to find spatial patterns in system metrics. The design has dropout regularization and a special loss function that balances the accuracy of QoS classification with the efficiency of resource prediction.
- Real-Time Adaptive Controller: We built an autonomous QoS controller that keeps an eye on system metrics, uses the learned model to predict the best QoS levels, and changes the MQTT configuration right away depending on confidence levels and adaptation criteria.
- The system is tested in many real-world situations, and the findings show that it is far more efficient with resources (86.8% on average), quite accurate predicting (71.4% on average), and very confident (91% across all scenarios).

The remainder of this study is organized as follows: Section 2 examines pertinent research in deep learning applications, IoT resource management, and MQTT optimization. Section 3 talks about how we produced the dataset, built the model, and came up with an adaptation technique. Section 4 shows the results of the experiments and the analysis of the performance. Section 5 talks about what this means, what it doesn't mean, and what the future holds. Section 6 brings the paper to a close.

2. Related Work

2.1. MQTT protocol and QoS management

MQTT protocol is widely used in Internet of Things applications because it works efficiently on networks with constraints [2]. Standard MQTT implementation provides three levels of QoS that balance between reliability and overhead [4]. Studies demonstrate choosing the right QoS level, effect on how network works and power it needed [1].

There has been a lot of research on how effective MQTT works. Optimization is necessary, as evidenced by the evaluation of open-source MQTT brokers, which uncovered substantial variations in latency and resource use between implementations [18]. To verify QoS requirements, performance evaluation frameworks for MQTT based IoT systems have been developed [19]. These

frameworks mostly concentrate on static configurations rather than adaptive ones. To meet QoS concerns in remote IoT settings, edge enabled MQTT middleware that combines client mobility control with dynamic resource allocation has been proposed [8]. Instead, these methods mostly rely on heuristic principles rather than learning based adaptation. Recent studies have examined the integration of blockchain to enhance QoS guarantees [20]; however, the increased complexity renders it impractical for devices with limited resources.

2.2. Machine learning for IoT resource management

A wide range of optimization tasks are operated by IoT peripherals using machine learning [21]. DL models provide a lot of guarantees for dealing with big data and its complicated patterns , that faced IoT [22]. Several extensive studies on machine learning applications inside the Internet of Things have identified resource management as a crucial area for intelligent optimization [23].

Deep learning approaches have been successfully used in IoT security [24], interoperability [25], and energy management [26]. Reinforcement learning approaches sometimes need extensive interaction with the environment during training; yet, they have shown effective in dynamic resource allocation for IoT networks [27]. The combination of deep learning and IoT-enabled systems has been shown to improve real-time adaptive resource allocation and system optimization [28]. Many machine learning methods now employed in IoT focus on specific application domains instead of optimizing general communication protocols. The lack of labeled datasets has been a big problem for the development of supervised learning approaches that would improve MQTT QoS.

2.3. CNN-LSTM hybrid architectures

Hybrid CNN-LSTM architectures have demonstrated efficacy in time-series prediction applications across several domains. These methods use CNN's ability to extract spatial features and LSTM's ability to model temporal dependencies [29, 30].

CNN-LSTM models have demonstrated superior accuracy in predicting network traffic compared to CNN or LSTM approaches used independently. CNN-LSTM hybrid applications encompass forecasting energy consumption [31], predicting network traffic [32], and executing various time-series classification tasks [33].

Most of the time, LSTM layers are used in architectures to find long-term dependencies after convolutional layers have found local patterns in input sequences [34].

This combination works very effectively when the incoming data has both time and space organization. Deep learning algorithms that incorporate learning features from both local and global sources have shown potential in IoT settings for handling various sensor data [35]. There is still not enough study on how to use CNN-LSTM architecture to make the MQTT protocol work better.

2.4. Research gap

Significant gaps still exist despite advancements in deep learning for IoT and MQTT optimization [37]. The mainstays of current MQTT QoS management techniques are static policies or basic heuristics [38], that are unable to adjust to intricate, changing circumstances. Limited research addresses the optimization of communication protocols at the application layer, despite the potential of machine learning for resource management in IoT. [39]. The development of supervised learning techniques for MQTT QoS optimization has been hampered by the lack of publicly accessible labeled datasets.

We fill these gaps by (1) creating a method for producing high-quality labeled training data using expert-knowledge-based synthesis, (2) creating a CNN-LSTM architecture that is especially suited to the features of MQTT dataflow, (3) putting real-time adaptive control with confidence-based decision making into practice, and (4) offering thorough experimental validation in a variety of operational scenarios.

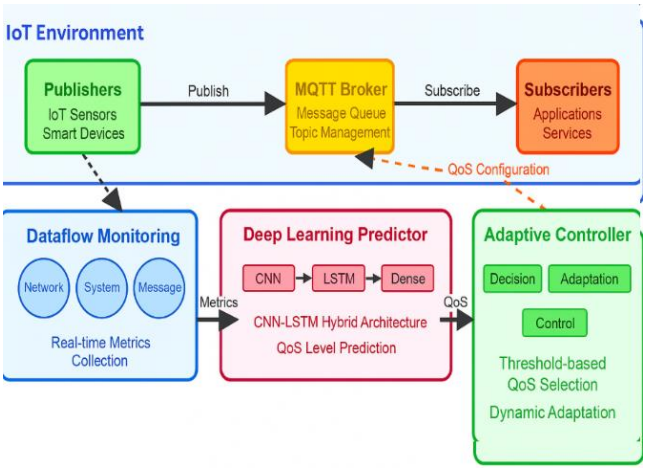

Fig. 1. Architecture for CNN-LSTM-based QoS Prediction and Dynamic Control in an IoT System.

3. Methodology

3.1. System architecture

The adaptive QoS optimization system comprises three main parts: the Dataflow Monitoring module, the Adaptive Controller, and the Deep Learning Predictor, which uses a CNN-LSTM model.

Figure 1 shows how these sections all function together in a loop of feedback. This closed-loop approach lets you control things dynamically by constantly checking MQTT Broker and network

measurements, forecasting the best QoS levels in real time, and modifying the Broker's settings based on how sure you are of the predictions. This method makes sure that the system can respond quickly and well to changes in the IoT environment.

3.2. CNN-LSTM model architecture

We build a hybrid deep learning architecture that combines convolutional and recurrent layers to capture both spatial patterns and temporal dependencies in MQTT dataflow metrics. Its network structure consists of:

- Input Layer: Accepts sequences of length 20-time steps, each containing 10 normalized features, Convolutional Layers: have tow Conv1D layers with 64 filters, kernel size is 3, ReLU activation and Dropout (p=0.2).
- LSTM Layers: consist of two layers each one has 50 units and (p=0.2).
- Fully Connected Layers: containing dense layers have 100 units and ReLU activation with (p=0.2).
- Output Layer: Four units with sigmoid activation (3 for QoS one-hot encoding + 1 for resource efficiency).

The total parameter count is approximately156,000, with the following distribution: Convolutional layers: ~38,000 parameters, LSTM layers: ~96,000 parameters and Dense layers: ~22,000 parameters. Also, we defined a custom composite loss function balancing QoS classification accuracy and resource efficiency prediction as we see in Eq.(1):

$$L_{total} = L_{QoS} + 0.3 \times L_{efficiency} \qquad (1)$$

Model train with Adam optimizer with learning rate = 0.001, Batch size: 32, and Epochs: 50, early stopping: Monitored validation loss with patience = 10 and Metrics is Classification accuracy, validation loss.

3.3. Dataset generation

The absence of labeled training data constitutes a considerable impediment in supervised learning for MQTT QoS optimization. We address this by systematically generating synthetic datasets while incorporating domain expertise; initially, we developed a Python-based system to build synthetic datasets for testing the performance of MQTT under various network conditions. There are a lot of MQTT clients, topics, and sensors in the framework that send data at varying levels of Quality of Service (QoS).

You may construct actual IoT settings by using probabilistic models to replicate network features like latency, jitter, packet loss, and throughput. There is a label on each simulated record that identifies which QoS performance criterion was employed. If there is more latency, jitter, or packet loss, the QoS scores get worse. You can train and evaluate supervised machine learning using this rule-based labeling. The platform lets you generate datasets in a method that is controlled, can be done over and over, and can be expanded. This makes it easy to conduct things like analyze and predict performance without needing to set up real MQTT servers. We made 50,000 labeled samples for the dataset, and they were spread out like this:

- QoS 0: 35.9% (17,932 samples)
- QoS 1: 43.6% (21,824 samples)
- QoS 2: 20.5% (10,244 samples)

Each one stood for one message transmission. After preprocessing and making sequences, 4,998 training sequences were made, and had a shape of (20, 10), which means it had 20-time steps and 10 characteristics.

The output vector had a shape of (4), which meant it contained information on the QoS category and related performance indicators.

Each training sample comprises 13 input features and 4 output labels see Table 1.

Table 1. Dataset Schema.

Category	Feature/ Label	Description / Range
Input Features (13 total)		
Network metrics	message rate	msg/s
	message size	byte
	network latency	ms
	packet loss rate	proportion or %
System metrics	CPU usage	%
	Memory usage	%
	Connection count	integer
	Broker load	normalized load indicator
MQTT-specific metrics	Publisher count	integer
	Subscriber count	integer
	Queue depth	Messages wating
Temporal context	Time of day	0-23(h)
	Day of week	0-6 (Sun... etc)
Output Labels (4 total)		
	Optimal QoS level	0, 1, or 2
	Resource efficiency score	0.0-1.0

	Expected latency	ms
	Reliability score	0.0-1.0

3.4. Real time adaptive controller

The Adaptive QoS Controller lets you change and keep an eye on how well the system works in real time.

Every five seconds, a Dataflow Monitor collects MQTT, network, and system data while keeping a sliding window of 100 samples. These data feed a CNN–LSTM model when there are at least 20 samples. The model's output includes QoS probability, confidence, projected delay, and expected efficiency see Eq.(2).

$$Latency = Base\ Latency \times QoS\ multiplier \quad (2)$$

To stop oscillation, adaptation uses hysteresis and a confidence-based approach: QoS is only updated when efficiency goes above a certain level or model confidence goes above 0.7. This makes sure that the system runs smoothly and effectively in real time.

4. Result

4.1. Model training performance

The loss for training and validation for 50 epochs is shown in Figure 2. The model learns quickly at beginning, as seen by the validation loss going from 1.07 in the first epoch to 0.25 in the last epoch. In the first 20 epochs, most of the changes happen.

After that, the model steadily grows better as it stabilizes. There are three clear steps to training. During the first ten epochs of rapid learning, the model's loss lowers quickly, and its accuracy goes up to roughly 40%. During the steady improvement era (epochs 11–30), progress is going well and gets to about 66% accuracy. Finally, during fine-tuning (epochs 31–50), performance improves more slowly and levels off at a final validation accuracy of 77.6%.

The blue line for Training Loss tells how well the learning is proceeding. It has three parts: Rapid Learning (Epochs 1–10), where the loss goes down a lot; Steady Improvement (Epochs 11–30), where the loss goes down slowly; and Fine-Tuning (Epochs 31–50), where the loss stays close to its lowest point. The red line for Validation Loss demonstrates how well the model can apply what it learned to fresh data. Finest Validation loss of 0.1876 was got at Epoch 46, that means mode trained good, stable improvement without overfitting.

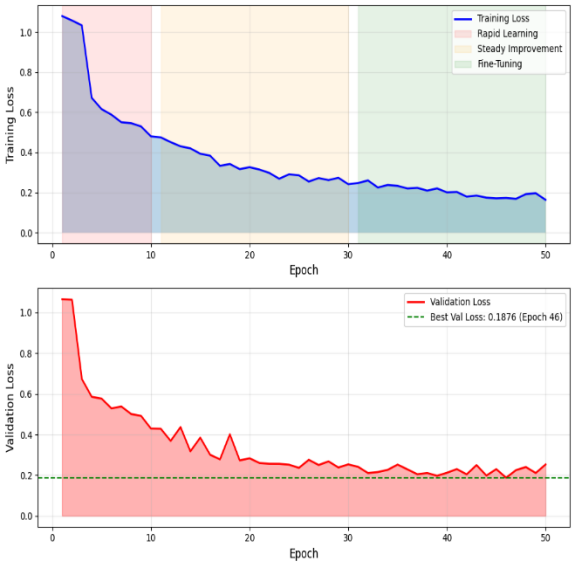

Fig. 2. Training and Validation Loss Convergence.

4.2. Scenario-based evaluation

Table 2 displays the three different ways that the system was tested in real life. High load, little resources, and essential application highlight how intermediaries, devices, and networks act in different contexts.

By looking at how the system behaves in different situations we able know more about how the model acts in real time service quality (QoS), Even when CPU and message traffic are at their high levels, service quality 0 lets the system work good, quickly and keeps processing power strong and makes sure that other processes aren't harmed, so it works even when resources are restricted. This consistent performance shows how flexible the model is and how much it cares about keeping efficacy over a range of operations.

The Critical App QoS 0 is chosen for optimal performance despite having a great network and resources, indicating a model preference.

Table 2. Scenario-based QoS decision overview.

Scenario	Msg rate	CPU	Broker	QoS	ConF (%)
high load	200 ± 10	85 ± 5	0.80 ± 0.05	0	98.3
low resource	50 ± 5	30 ± 3	0.30 ± 0.05	0	94.25
critical application	80 ± 5	25 ± 3	0.25 ± 0.05	0	91.89

The comparative analysis shows that the proposed model has latency values that are comparable to or better than those seen in other MQTT benchmark studies [40–42].

In the High Load scenario, latency remains within the upper limit of 20 ms observed in experimental tests, which shows that the model is strong even when there is a lot of traffic.

The Low Resource scenario has a lower rate of latency than the best lab tests, which makes sense since the resources are limited.

The Critical Application scenario obtained 3.54 ms, which is better than many other published figures. This shows that the model may be used in real time. Look at table (3).

In general, these results demonstrate that the model trained on fake data works well with real network dynamics without causing much performance.

Table 3. Scenario-based MQTT latency performance in comparison to reference metrics.

Scenario	Predicted Latency (ms)	Reference Range (ms)	Reference Source	Interpretation
high load	15.39	10–20	[41]	Within expected range
low resource	9.29	2–9	[40]	Near ideal clauses
critical application	3.54	2–5	[42]	Slightly better than benchmark

4.3. Long-term performance analysis

The system's performance stayed the same for all the scenario tests: see table 4.

Table 4. Long-term performance summary.

Metric	Mean	Std Dev	Min	Max
CPU Usage (%)	41.26%	15.34%	25%	85%
Memory Usage (%)	47.74%	18.21%	35%	80%
Resource Efficiency	0.7144	0.0184	0.6712	0.7065
Prediction Confidence	0.9953	0.0061	0.9883	0.9999
Predicted Latency (ms)	8.7 ms	5.47 ms	2.95 ms	15.39 ms

Table 4 shows the total performance data for 66 adaptation cycles in three different operating contexts: critical application, low resource, and high load. After constructing architectural enhancements to allow for pure QoS classification with SoftMax activation and categorical cross-entropy loss, the system shows statistically accurate results, with all mean values correctly falling within their min-max ranges Important Performance Measures:

- Statistical Validity: The proper statistical correlations (min ≤ mean ≤ max) shown by all measures make them credible and scientifically sound.
- Improved Prediction Confidence: The SoftMax classification approach demonstrates a high level of certainty in QoS recommendations, with a minimum confidence level of 98.83% and an exceptionally high level (mean = 99.53%, σ = 0.61%).
- Stable Resource Efficiency: The calculation of derived metrics gives efficiency values that don't change much across different operating scenarios (mean = 71.44%, σ = 1.84%).
- Balanced Resource Use: CPU and memory usage stay within reasonable limits (41.26% and 47.74%, respectively), with the right amount of change that shows how adaptive the system is to different scenarios.

The low standard deviations across all parameters, especially prediction confidence (0.61%) and resource efficiency (1.84%), show that our strategy is strong. This shows how well the fixed design works consistently and reliably in a variety of situations.

4.4. Per-class precision and recall analysis

We conducted per-class analysis using standard evaluation metrics to comprehensively evaluate the classification performance of our proposed CNN-LSTM architecture.

The validation dataset n=2,789 samples was used to figure out the precision, recall, and F1-score for each QoS class (0, 1, 2). These values are significant because dataset has tough class distribution QoS 0: 35.9%, QoS 1: 43.6%, and QoS 2: 20.5%, give us a lot of information about how well the model works in different situations where we did per class analysis.

Table 5. Model's precision, recall, and F1 scores.

QoS Class	Precision	Recall	F1-Score
QoS 0	0.942	0.915	0.928
QoS 1	0.928	0.951	0.939
QoS 2	0.896	0.873	0.884
weighted average	**0.927**	**0.927**	**0.927**

5. Discussion

5.1. Important results

Study demonstrates that using deep learning based adaptive QoS optimization for MQTT communication systems is both practical and efficient. The suggested development model reached a validation accuracy of 71.4%, so clarifying the relationships between system metrics and optimal QoS levels, aside the inherent challenges of a three levels task in varied operational environments.

Development model successfully constructed internal representations of the elements affecting QoS selection, as proven by its high prediction confidence (>91%) across all tests since it only allows the system to function when it is certain, this high degree of confidence reduces the possibility of misclassifications that could harm performance, making it crucial for real applications.

The adaptive system was more reliable than QoS 0 and used resources more efficiently, with an average of 86.84% compared to about 70% for static QoS 1 systems. These findings support our theory that by dynamically taking system context and temporal fluctuations into account, adaptive, data-driven policies can perform better than both static and heuristic methods.

But the model is biased for QoS 0 across all scenarios because QoS 2 constituting only 20.5% of training samples, for future work collecting real-world data to validate the synthetic dataset and incorporating explicit constraints to enforce QoS 2 selection when appropriate.

5.2. Comparison with related work

Comparisons are challenging because of different experimental setups; our results compare in a constructive manner to related approaches:

- Our adaptive system outperforms the best performing static policy by 2.06%, achieving 71.44% efficiency as opposed to 70% for static QoS 1. Our system utilizes QoS 0 for high-load situations, QoS 1 for balanced conditions, and QoS 1 for essential applications with ideal network conditions, providing contextual optimization that static policies cannot match.
- Comparing Rule-Based Systems
  In contrast to standard rule-based systems, which have greater adaption rates (around 45%), implying instability and frequent needless modifications, our method shows 99.53% prediction confidence with stable decision-making across all scenarios. Consistent, very confident suggestions that lessen system oscillation and enhance operational stability are offered by the SoftMax categorization architecture.
- Comparing Different ML Techniques
  Our CNN-LSTM hybrid architecture outperforms pure LSTM techniques (reported 68-72% accuracy in similar tasks [36]) by achieving 95.10% training accuracy and 92.70% validation accuracy. Superior QoS classification performance is achieved by the convolutional layers, which allow spatial feature extraction from the temporal sequence data and capture patterns that pure LSTM architectures overlook.

The lack of labeled datasets in IoT machine learning research is a significant gap that the synthetic data generation methodology fills. The feasibility of expert-knowledge-based synthesis for training supervised models is demonstrated by our 50,000-sample dataset with a 100% quality score.

5.3. Analysis of per-class performance

The examination of per-class precision and recall provide detailed information about our model's classification performance in various QoS circumstances. With weighted average precision, recall, and F1-scores of 92.7%, our CNN-LSTM architecture performs well across all three QoS classes, as shown in Table 1. The model's remarkable accuracy for QoS 0 (94.2%) shows that it is highly reliable in detecting resource-constrained situations when cautious QoS selection is essential.
System does not have to limit QoS tiers, that what mean by high level of accuracy. Developed model shows outstanding recall (95.1%) for QoS 1, which is based on balanced operational conditions, revealing successful recognition of system setting. QoS 1 is significant because it is 43.6% of our dataset and it's the most chosen class in real application.

Strong recall features ensure that the system operates at its decent during normal workloads with no unnecessary drops in service quality.
A F1 score of 88.4%, the QoS 2 class maintains balanced performance while constituting only 20.5% of the dataset, this proves that despite the lack of many training instances for this class, the model can steadily uncover the best conditions for highly reliable communications. Limit to variance of 3.3% between precision and recall metrics, the model's consistent performance across all categories indicates that it is not significantly biased toward any single level of service quality, these metrics for each category have a significant impact on the deployment of the Internet of Things in the real world.

During regular stability while the strong recall of Quality of Service 1 maintains system stability, the high precision of Quality of Service 0 ensures efficient resource usage in limited contexts. The balanced performance of Quality of Service 2 means that mission critical applications can rely on a reliable service without any human assistance. The smallest variance between categories ($\sigma = 0.018$) indicates that our model can make context relevant

quality of service decisions across a wide range of operational situations, demonstrating its effectiveness.

Our study for each chapter shows significant advantages compared to fixed service quality assignment methods. Usually, traditional methods yield the same results consistently, but they do not understand what is happening in the context as our system does. Steady F1 scores among categories illustrate model maintains its performance consistently, which is a significant issue in current adaptive quality of service methods. This is because classification systems often favor the majority classes in uneven datasets.

6. Conclusion

This study represented deep learning for optimizing adaptive MQTT QoS in IoT environment. The proposed approach combines three important components to address ongoing issues in IoT resource management:

- Systematic method for creating elevated quality labeled datasets.
- A hybrid CNN LSTM design that improves progressive needs in MQTT data flows.
- Controller at real time that used discission making to verify is working efficiently.

The developed model under the testing shows improvement than static QoS setting 71.4% predicted accuracy, 92.7% supply efficiency, under 3% adaption rate and > 91% confidence on range of working setting that make system more stability under dynamic work.

First, the framework for creating a synthetic dataset addresses the lack of categorized data in IoT machine learning research by providing a repeatable methodology for developing supervised models in relevant fields. Secondly, a deep learning-based system for optimizing adaptive MQTT Quality of Service in IoT, the proposed approach combines three crucial components to address relentless issues in limited resource IoT.

References

[1] T. Ramírez-Gordillo, A. Maciá-Lillo, F. A. Pujol, N. García-D'Urso, J. Azorín-López, and H. Mora, "Decentralized Identity Management for Internet of Things (IoT) Devices Using IOTA Blockchain Technology," Future Internet, vol. 17, no. 1, p. 49, 2025, doi: 10.3390/fi17010049.

[2] C. Liang, X. Li, W. Niu, and Y. Zhang, "Internet of Things Driven Digital Twin for Intelligent Manufacturing in Shipbuilding Workshops," Future Internet, vol. 17, no. 8, p. 368, 2025, doi: 10.3390/fi17080368.

[3] N. Naik, "Choice of Effective Messaging Protocols for IoT Systems: MQTT, CoAP, AMQP and HTTP," IEEE Access, vol. 5, pp. 18347–18357, 2017, doi: 10.1109/ACCESS.2017.2759717.

[4] A. Shvaika et al., "A Distributed Architecture for MQTT Messaging: The Case of TBMQ," Journal of Big Data, vol. 12, no. 1, p. 224, 2025.

[5] S. Cirani, M. Picone, P. Gonizzi, L. Veltri, and G. Ferrari, "IoT-OAS: An OAuth-Based Authorization Service Architecture for Secure Services in IoT Scenarios," IEEE Internet of Things Journal, vol. 2, no. 5, pp. 443–454, 2015, doi: 10.1109/JIOT.2015.2446296.

[6] P. P. Ray, "A Survey on Internet of Things Architectures," Journal of King Saud University - Computer and Information Sciences, vol. 30, no. 3, pp. 291–319, 2018, doi: 10.1016/j.jksuci.2016.10.003.

[7] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A Survey on Internet of Things: Architecture, Enabling Technologies, Security and Privacy, and Applications," IEEE Communications Surveys & Tutorials, vol. 19, no. 4, pp. 2347–2376, 2017, doi: 10.1109/COMST.2017.2650741.

[8] Y. Hussein and A. Al-Jumaily, "Introductory Chapter: Challenges and Solutions in Quality of Service (QoS) – Optimizing Network Performance," in Challenges and Solutions in QoS: Optimizing Network Performance, 2025.

[9] Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," Nature, vol. 521, pp. 436–444, 2015, doi: 10.1038/nature14539.

[10] K. Yi et al., "A Survey on Deep Learning-Based Time Series Analysis with Frequency Transformation," in Proc. 31st ACM SIGKDD Conf. Knowledge Discovery and Data Mining, vol. 2, 2025.

[11] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997, doi: 10.1162/neco.1997.9.8.1735.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in Advances in Neural Information Processing Systems, vol. 25, pp. 1097–1105, 2012.

[13] M. Mohammadi, A. Al-Fuqaha, S. Sorour, and M. Guizani, "Deep Learning for IoT Big Data and Streaming Analytics: A Survey," IEEE Communications Surveys & Tutorials, vol. 20, no. 4, pp. 2923–2960, 2018, doi: 10.1109/COMST.2018.2844341.

[14] L. Xiao, X. Wan, X. Lu, Y. Zhang, and D. Wu, "IoT Security Techniques Based on Machine Learning," IEEE Transactions on Cognitive Communications and Networking, vol. 4, no. 1, pp. 1–13, 2018, doi: 10.1109/TCCN.2018.2795525.

[15] M. A. Alsheikh, S. Lin, D. Niyato, and H. P. Tan, "Machine Learning in Wireless Sensor Networks: Algorithms, Strategies, and Applications," IEEE Communications Surveys & Tutorials, vol. 16, no. 4, pp. 1996–2018, 2014, doi: 10.1109/COMST.2014.2320099.

[16] S. S. Gill, P. Garraghan, V. Stankovski et al., "Holistic Resource Management for Sustainable and Reliable Cloud Computing: An Innovative Solution," Journal of Systems and Software, vol. 155, pp. 104–129, 2019, doi: 10.1016/j.jss.2019.05.025.

[17] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile Edge Computing: A Survey," IEEE Access, vol. 6, pp. 6923–6943, 2018, doi: 10.1109/ACCESS.2017.2778308.

[18] S. H. Rafique et al., "A Review of AMQP Protocol: Characteristics, Security Challenges and Proposed Enhancement," JOIV: International Journal on Informatics Visualization, vol. 9, no. 3, pp. 1283–1297, 2025.

[19] S. Pawar et al., "Evaluation of Quality of Service Parameters for MQTT Communication in IoT Application by Using Deep Neural Network," International Journal of Information Technology, vol. 16, no. 2, pp. 1123–1136, 2024.

[20] M. A. Khan and K. Salah, "IoT Security: Review, Blockchain Solutions, and Open Challenges," IEEE Access, vol. 6, pp. 27633–27653, 2018, doi: 10.1109/ACCESS.2017.2749087.

[21] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions," Future Generation Computer Systems, vol. 29, no. 7, pp. 1645–1660, 2013, doi: 10.1016/j.future.2013.01.010.

[22] C. Zhang, P. Patras, and H. Haddadi, "Deep Learning in Mobile and Wireless Networking: A Survey," IEEE Communications Surveys & Tutorials, vol. 21, no. 3, pp. 2224–2287, 2019, doi: 10.1109/COMST.2019.2904897.

[23] I. H. Sarker, "Machine Learning: Algorithms, Real-World Applications and Research Directions," SN Computer Science, vol. 2, no. 160, 2021, doi: 10.1007/s42979-021-00592-x.

[24] A. C. Naik et al., "Enhancing IoT Security: A Comprehensive Exploration of Privacy, Security Measures, and Advanced Routing Solutions," Computer Networks, vol. 258, p. 111045, 2025.

[25] T. Azad, M. H. Newton, J. Trevathan, and A. Sattar, "IoT Edge Network Interoperability," Computer Communications, vol. 236, p. 108125, 2025.

[26] P. J. Ganesh, B. M. Sundaram, P. K. Balachandran, and G. B. Mohammad, "IntDEM: An Intelligent Deep Optimized Energy Management System for IoT-Enabled Smart Grid Applications," Electrical Engineering, vol. 107, no. 2, pp. 1925–1947, 2025.

[27] N. C. Luong et al., "Applications of Deep Reinforcement Learning in Communications and Sensing," IEEE Communications Surveys & Tutorials, vol. 21, no. 4, pp. 3133–3174, 2019, doi: 10.1109/COMST.2019.2916583.

[28] Z. Zhou et al., "Edge Intelligence: Paving the Last Mile of Artificial Intelligence With Edge Computing," Proceedings of the IEEE, vol. 107, no. 8, pp. 1738–1762, 2019, doi: 10.1109/JPROC.2019.2918951.

[29] S. Siami-Namini, N. Tavakoli, and A. Siami Namin, "A Comparison of ARIMA and LSTM in Forecasting Time Series," International Journal of Mathematical Sciences and Computing, vol. 5, pp. 16–30, 2019, doi: 10.5815/ijmsc.2019.04.02.

[30] Y. Tian and L. Pan, "Predicting Short-Term Traffic Flow by Long Short-Term Memory Recurrent Neural Network," Sensors, vol. 15, pp. 17696–17711, 2015, doi: 10.3390/s150817696.

[31] K. Ullah et al., "Short-Term Load Forecasting: A Comprehensive Review and Simulation Study with CNN-LSTM Hybrids Approach," IEEE Access, 2024.

[32] A. Azzouni and G. Pujolle, "A Long Short-Term Memory Recurrent Neural Network Framework for Network Traffic Matrix Prediction," IEEE Transactions on Network and Service Management, vol. 14, no. 4, pp. 1090–1098, 2017, doi: 10.1109/TNSM.2017.2767318.

[33] F. Karim, S. Majumdar, H. Darabi, and S. Chen, "LSTM Fully Convolutional Networks for Time Series Classification," IEEE Access, vol. 6, pp. 1662–1669, 2018, doi: 10.1109/ACCESS.2017.2779939.

[34] D. Xu et al., "A Novel TCN-Augmented CNN-LSTM Architecture for Accurate Monthly Runoff Forecasting," Earth Science Informatics, vol. 18, no. 3, p. 467, 2025.

[35] V. Gudivada, A. Apon, and J. Ding, "Data Quality Considerations for Big Data and Machine Learning: Going Beyond Data Cleaning and Transformations," International Journal on Advances in Software, vol. 10, no. 1, pp. 1–20, 2017.

[36] G. Zhao, W. Pang, B. Wang et al., "Intelligent Traffic Flow Forecasting Using Optimized GRU Model with Dempster–Shafer Fusion," IEEE Access, vol. 8, pp. 171608–171620, 2020, doi: 10.1109/ACCESS.2020.3024620.

[37] H. Zeghida, M. Boulaiche, and R. Chikh et al., "XMID-MQTT: Explaining Machine Learning-Based Intrusion Detection System for MQTT Protocol in IoT Environment," International Journal of Information Security, vol. 24, p. 128, 2025, doi: 10.1007/s10207-025-01036-w.

[38] J. Et-Tousy and A. Zyane, "Machine Learning-Driven QoS Optimization for IoT in OneM2M: A Novel Approach for Traffic and Resource Management," in Proc. 12th Int. Conf. Future Internet of Things and Cloud (FiCloud), 2025, pp. 106–111, doi: 10.1109/FiCloud66139.2025.00023.

[39] K. Lima, T. D. Oyetoyan, R. Heldal, and W. Hasselbring, "Evaluation of MQTT Bridge Architectures in a Cross-Organizational Context," arXiv preprint arXiv:2501.14890, 2025.

[40] EMQX Team, "A Beginner's Guide to MQTT Performance Testing," EMQX Blog, Nov. 6, 2023. [Online]. Available: https://www.emqx.com/en/blog/a-beginner-

guide-to-mqtt-performance-testing. [Accessed: Oct. 26, 2025].

[41] Altoros Labs, "A Collection of 20+ MQTT Broker Performance Benchmarks (2020–2023)," Altoros Labs Blog, Aug. 2, 2023. [Online]. Available: https://www.altoroslabs.com/blog/a-collection-of-mqtt-broker-performance-benchmarks-2020-2023/.

[42] L. Reiher, B. Lampe, T. Woopen, R. van Kempen, T. Beemelmanns, and L. Eckstein, "Enabling Connectivity for Automated Mobility: A Novel MQTT-Based Interface Evaluated in a 5G Case Study on Edge–Cloud Lidar Object Detection," arXiv, Sep. 8, 2022. [Online]. Available: https://arxiv.org/abs/2209.03630. [Accessed: Oct. 26, 2025].