Ingénierie des Systèmes d'Information
Vol. 31, No. 5, May, 2026, pp. 1375-1387
Journal homepage: http://iieta.org/journals/isi

Performance Analysis and Implementation of AdaBoost and K-Nearest Neighbour-Based Computation Offloading in Fog Architecture

Prince Gupta[1*], Annu Dhankhar[2], Swasti Singhal[1], Ashima Arya[1], Manvi Khatri[3], Adesh Kumar[4]

[1] Department of Computer Science and Information Technology, Krishna Institute of Engineering and Technology, Ghaziabad 201206, India
[2] Qualys Inc, Foster City 94404, United States
[3] Department of CSE(AI), Krishna Institute of Engineering and Technology, Ghaziabad 201206, India
[4] Department of Electrical & Electronics Engineering, School of Advanced Engineering, University of Petroleum & Energy Studies, Dehradun 248007, India

Corresponding Author Email: prince.rkg@gmail.com

https://doi.org/10.18280/isi.310501

Received: 22 April 2025
Revised: 1 September 2025
Accepted: 15 April 2026
Available online: 31 May 2026

Keywords:
Adaptive Boosting algorithm, k-nearest neighbour, fog computing, cloud communication, machine learning, computation offloading

ABSTRACT

With the rapid expansion of Internet of Things (IoT) networks involving connected devices, an increasing volume of heterogeneous data is being generated, which requires real-time processing, low latency, and energy-efficient computations. Edge devices, typically resource-constrained, cannot always run computationally heavy operations, and, therefore, the paradigm of computation offloading has become important within distributed fog computing networks. In this research, we propose a lightweight yet adaptive framework for deciding about computation offloading. The offloading strategy determines whether a particular task needs to be computed locally or moved to neighbouring fog nodes. The offloading problem is considered a binary classification problem and is solved using a hybrid model that combines two different algorithms, such as Adaptive Boosting (AdaBoost) and k-nearest neighbour (KNN). To test the feasibility of this framework, a comprehensive experiment is performed based on a dataset with 62,120 samples containing 20 features related to wireless channel characteristics, latency restrictions, and available resources. As shown in the experimental results, the accuracy achieved by AdaBoost is higher, at 96.93%, than KNN, which reaches 94.27%. Furthermore, this model can decrease execution latency and energy usage by up to 18.6 ms and 0.42 J, respectively, beating KNN by almost 23% and 26% in latency and energy efficiency, respectively. The novelty of this work lies in the design of a hybrid, low-complexity, and interpretable offloading decision framework that achieves high prediction accuracy while maintaining real-time responsiveness, addressing the limitations of existing optimization and deep learning-based approaches. The proposed model effectively improves task execution efficiency, reduces communication overhead, and enhances overall system performance, making it suitable for latency-sensitive IoT applications deployed in dynamic fog computing environments.

1. INTRODUCTION

The fast-growing development of Internet of Things (IoT) and latency-critical applications has put considerable pressure on achieving efficient computing capabilities, reactivity, and effective resource management in a distributed environment. Current applications such as autonomous vehicles, healthcare systems, automated production, and surveillance have a high demand for advanced processing with strict restrictions on delays. At the same time, the limited computational capacities, energy reserves, and memory storage impose certain restrictions on the IoT devices' capability to perform complex computations locally [1-3]. In order to address the above-mentioned limitations, offloading computations has been widely implemented as a promising strategy, during which computationally demanding processes are transferred from the limited resources of IoT devices to external sources that are capable of handling them. Initially, cloud-based infrastructure became a solution to offload computations due to its rich computational resources and scalable storage capabilities [4, 5]. In spite of all the mentioned strengths, cloud-based computing infrastructure suffers from extremely high latencies, high bandwidth usage, and congestion resulting from the distance between end-user devices and cloud data centers [6].

Fog computing has thus evolved as an expansion of cloud computing in order to resolve these issues by offering services related to computation, storage, and networking at the edge of the network. In this case, intermediate devices like gateways, routers, and edge servers carry out functions such as data processing and decision-making, which help minimize latency and improve quality of service (QoS) [7]. The distribution of these processes enhances efficiency and allows real-time processing through reduced communication overhead and

transmission delay. As such, fog computing provides a suitable environment for computation offloading in dynamic IoT settings. The process of computation offloading in fog computing is a complex process, as it takes into account various issues such as the dynamic nature of network conditions, heterogeneous resources, and task characteristics. The computation offloading decision relies on several variables like computation complexity, channel state, energy, and latency considerations [8-10].

There have been various ways proposed for solving this issue, including techniques based on optimization such as convex optimization, Lyapunov optimization, and stochastic allocation, among others [11-13]. Even though optimization techniques produce optimal or near-optimal results, they are known to possess high computational complexity.

Several innovations have recently emerged with regard to employing machine learning (ML) and deep reinforcement learning (DRL) models for adaptive computation offloading [14-37]. They provide an accurate representation of the system and enable more effective decision-making under uncertain conditions. Nonetheless, deep learning models are highly data-demanding, resource-intensive, and lack transparency, which makes them impractical for application in real-time fog systems [38, 39]. This raises the question of how to design a lightweight and computationally efficient framework capable of generating optimal offloading solutions in real time.

To fill this research gap, this paper will discuss a performance evaluation and development of computation offloading schemes for fog computing architectures with the use of a hybrid ML paradigm. The suggested framework utilizes the advantages of conventional machine learning models to deliver a balanced solution for computation offloading in fog settings.

2. LITERATURE REVIEW

Table 1 lists the findings from the different research articles. New innovations in IoT technology and edge/fog computing have created the need for efficient offloading mechanisms. Edge and fog computing lower latencies and improve QoS more than cloud-based architecture does [1-3]. Nevertheless, mobile devices' limited resources necessitate efficient decision-making on offloading [4]. Previous research has emphasized cloud-based offloading that experiences high latencies because of long-distance transmissions [6, 7]. To tackle the problem, D2D and fog computing have emerged by allowing resource-sharing and minimizing energy consumption [8-10].

Latency-energy trade-off has been handled through various optimization algorithms, including Lyapunov and convex optimizations [11-13]. Workload allocation in MEC architecture is another interesting topic of research that has attracted many researchers' attention recently [14-16]. Further improvements have been made in task allocation and resource scheduling using techniques such as game theory, stochastic optimization, and heuristic methods such as simulated annealing [17-20]. Nevertheless, all the above strategies exhibit poor adaptability to the rapidly changing environment.

In recent years [21-34], researchers have considered employing ML algorithms and the concept of DRL, where algorithms such as DQN and Actor-Critic can make adaptive decisions during unpredictable network conditions [35-39]. Even though these strategies work effectively, their implementation is quite complicated and costly. Federated learning, privacy-aware offloading, and intelligent fog AI systems have also been explored as promising solutions in recent times [40-50]. Nonetheless, no study has yet considered proposing an efficient, lightweight, and fast decision-making strategy that works in real-time fog computing environments.

Table 1. Findings of literature review

Ref.	Findings (Key Idea)	Platform	Advantages	Limitations	Research Gaps
[1-3]	Edge/Fog computing reduces latency vs cloud	Edge/Fog	Low latency, localized processing	Limited resources	Need efficient offloading policies
[6,7]	Cloud-based offloading improves computational power	Cloud	High processing capability	High delay, bandwidth dependency	Not suitable for real-time apps
[8]	Mobility-assisted D2D offloading using optimization	D2D/MEC	Improves connectivity & resource sharing	Limited scalability	Needs adaptive decision models
[9]	Socially aware MEC offloading framework	MEC	Improves energy efficiency	Complex coordination	Requires lightweight models
[10]	Incentive-based IoT cloud offloading (Aura)	IoT Cloud	Energy saving, incentive-aware	High complexity	Lack of real-time adaptation
[11]	Lyapunov optimization for latency-energy tradeoff	MEC	Theoretical optimal solutions	High computation overhead	Limited real-time implementation
[12,13]	Resource allocation + energy-delay optimization	MEC	Balanced performance	Complex modeling	Need simplified methods
[14-16]	Multi-user task offloading & scheduling	MEC	Improved throughput	Requires global knowledge	Poor scalability
[17-20]	Game theory & heuristic-based scheduling	Edge/Fog	Efficient resource allocation	Not adaptive to dynamic states	Lack of learning capability
[35-36]	DRL for adaptive offloading decisions	MEC	Handles a dynamic environment	High training cost	Computationally expensive
[37-39]	DRL-based resource optimization & offloading	Edge AI	High accuracy	Complex implementation	Difficult for real-time systems
[41]	Survey on fog offloading mechanisms	Fog	Comprehensive analysis	No implementation	Need practical frameworks
[49-51]	Delay-optimized and heterogeneous offloading	IoT/Fog	Improved latency performance	Hardware dependent	Needs generalizable models
[50]	Privacy-preserving DRL offloading	IoMT	Secure and adaptive	High computation overhead	Needs lightweight security models

[52-54]	ML-based offloading (survey & taxonomy)	Edge/Fog	Good adaptability	Model complexity	Need hybrid lightweight models
[53]	Federated fog computing for IoT	Fog + FL	Privacy & scalability	Communication overhead	Efficient aggregation needed
[55-58]	Scheduling and dynamic resource management	Fog	Efficient task handling	Static assumptions	Need real-time adaptive models

Note: IoT = Internet of Things; D2D = device-to-device; MEC = mobile edge computing; DRL = deep reinforcement learning; ML = machine learning; FL = federated learning; IoMT = Internet of Medical Things; AI = Artificial Intelligence; QoS = quality of service.

3. OFFLOADING FRAMEWORKS IN FOG COMPUTING

Mobile Assistance Using Infrastructure (MAUI): The main objective of MAUI, through offloading, is the optimization of energy usage [10]. MAUI allows highly dynamic offloading by constantly conducting profiling. Using this architecture, one can create an impression that the entire application is running on the user's phone without worrying about the complexity of offloading. As per the developers' experience of the tasks that can or cannot be offloaded, MAUI partitioning is decided, as shown in Figure 1. Two conditions have to be met in the planning phase; these are (1) the application binaries should be available on both server and mobile phone sides, and (2) proxies, profilers, and solvers have to be mounted on both sides. Initially, the MAUI profiler discovers the device capabilities and then monitors the program and network attributes during execution time. Since these attributes are dynamically changing, any out-of-date information can cause errors for MAUI. The decisions related to offloading are made during runtime.

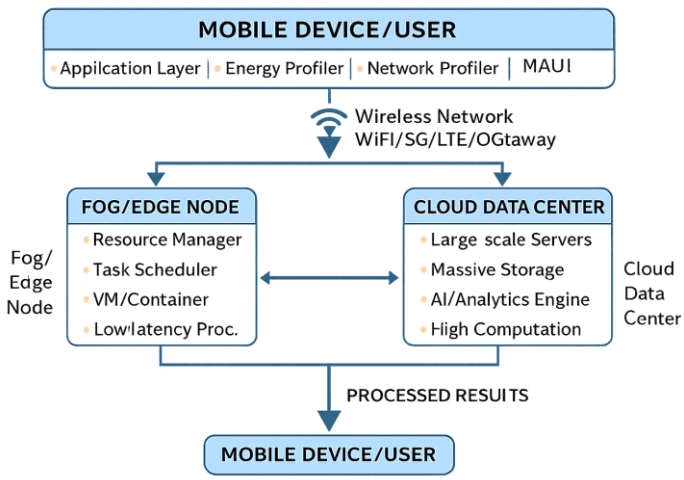

Figure 1. Mobile Assistance Using Infrastructure (MAUI)'s framework architecture

Decisions on which components should be remotely executed depend on the outcome of the solver in MAUI. The views of the MAUI profiler form the basis for these decisions. An illustration of MAUI architecture is shown in Figure 1. In the framework that works in mobile devices, a solver, profiler, and proxy work together. The MAUI profiler checks the ability of each method to save energy and collects information on the state of the mobile device and that of the network whenever it is invoked. The proxy uses the calculations done by the profiler and transfers data and control from the server to the mobile phone and vice versa, alongside the MAUI solver. Tasks performed by both the profiler and server proxy in the server are identical to those of the client profiler and proxy. **Phone2Cloud architecture**: Figure 2 depicts the Phone2Cloud architecture with functional units. Phone2Cloud architecture for offloading computations to improve the performance of applications and increase the energy efficiency of mobile devices. This architectural design allows for semi-automatic offloading, which involves executing computationally heavy parts of applications on external cloud-based computing resources. Applications are prepared before execution by modifying them in order to be able to use cloud computing resources. This architectural design employs static analysis based on user delay tolerance for making decisions about offloading. As shown in Figure 2, communication occurs between mobile devices and cloud computing resources via wireless networks. Phone2Cloud also offers a simple method of estimating Wireless Fidelity (Wi-Fi) connection delays.

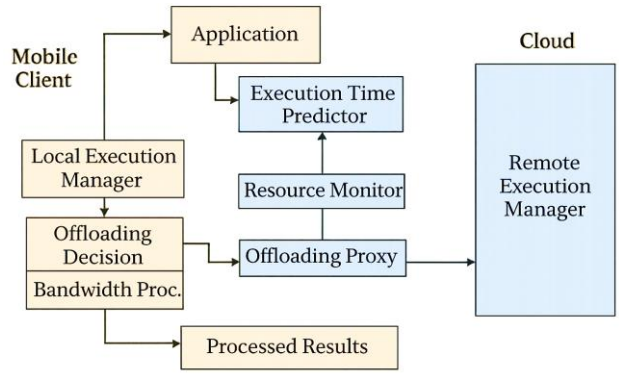

Figure 2. Phone2Cloud architecture

Optimization-based techniques for computation offloading leverage explicit network conditions, resource availability, and task properties for achieving optimal system performance through the use of convex optimization and game-theoretic models [31-33]. Such models seek to optimize system performance through simultaneous minimization of latency and energy consumption, while also maximizing utility. Depending on access to resources, such models may either include single-server or multi-server offloading techniques. Single-server computation offloading is achieved through the formulation of offloading policies that take into account constraints like radio spectrum availability, computing ability, and storage capabilities of a single edge server. In such approaches, the use of in-network caching along with heuristic optimization results in higher efficiency of the model by minimizing energy and processing latencies under partial Channel State Information (CSI) [37]. Energy-conscious MEC systems include an environmental energy harvesting technique, allowing tasks to be executed depending on resource availability. Mobility-aware models for computation offloading further optimize system performance by reducing latency and offering increased continuity services to users in mobile environments [39]. More sophisticated models for

resource allocation in edge cloud environments are based on dynamic optimization and incentive mechanisms for encouraging cooperation among users and servers [40-44]. Lastly, matching-based models help to achieve higher efficiency of task allocation. In MEC, multi-server offloading is an efficient approach for enhancing energy efficiency, reducing latency, and load balancing by splitting tasks between different edge servers. Task scheduling and CPU frequency scaling techniques have been utilized in MEC systems to enhance energy usage and performance [46]. Software-Defined Ultra-Dense Networks (SD-UDNs) also employ centralized control nodes to effectively offload tasks to appropriate edge servers in order to reduce network latency [47]. Markov Decision Process-based Computing (MDPC)-based task offloading strategies have further been utilized in vehicular edge computing systems to enhance latency in the presence of mobility and network uncertainty [48]. Despite the benefits gained from the above methodologies, various problems, such as scalability, optimization complexities, mobility management, and dynamic resource allocation, are still open challenges. Recently, model-free RL algorithms have attracted considerable attention in the design of algorithms for task allocation and resource management in the MEC system because of their adaptiveness and low reliance on pre-existing system models. Through interaction with the environment, the algorithms are able to make intelligent decisions about how to behave optimally in dynamically varying wireless conditions. According to a number of recent studies [49, 50], a task-offloading mechanism for minimizing communication delay, computation delay, and handover delay while fulfilling the energy budget constraints was developed using the Volatile Multi-Armed Bandit (VMAB) model. In addition, An et al. [51] have put forth a task offloading framework based on the DRL technique, which allows for local computation of computational tasks or offloading of the task fully in wireless MEC networks. It aims at deriving the optimal binary offloading strategy that maximizes the overall computation rate in light of system constraints. Moreover, Abdulazeez and Askar [52] proposed a novel framework based on Partially Observable Markov Decision Process (POMDP) in which a decision guideline algorithm is used to compute optimal offloading strategies. The approach ensures that such optimal offloading policies are rational and incentive-compatible.

4. COMPUTATION OFFLOADING MODEL

The computation offloading model to be developed employs AdaBoost and k-nearest neighbour (KNN) algorithms in determining whether a given computation task will be performed locally or in the fog nodes. The overall AdaBoost computation offloading scheme workflow can be seen in Figure 3. The computation task is considered to be a binary classification problem where "0" denotes local processing while "1" denotes fog node-based processing. The dataset employed for analysis consists of 62,120 samples with 20 features each. Each row of the dataset corresponds to a specific time period, T_i, while the features represent wireless channel gain values and associated user-specific network parameters. The dataset was further balanced to maintain approximately equal samples from both offloading categories, thereby improving classification stability and reducing prediction bias. The offloading status variable, O_i, refers to the offloading status of each example. The AdaBoost algorithm was applied to enhance classifier performance in order to combine multiple weak classifiers into a strong classifier. The Decision Stump algorithm will be adopted in the current work as a weak classifier. The training samples are first assigned the same weight values using Eq. (1):

$$W_i = \frac{1}{N} \tag{1}$$

where, N is the total number of training data samples. In each iteration, the weak classifier will use the network and task-related properties such as wireless channel gains, latency, and resources to classify the offloading states. Misclassified examples will have higher weights in the next iteration and thus help in focusing more on hard decision boundary regions. The whole weighted classification process goes on until the defined number of classifiers is reached. Figure 4 shows the process flow diagram of the proposed AdaBoost-based offloading approach. For classifying the offloading decisions based on similarities of the network states, the KNN classifier can be used. Since KNN is applicable in computation offloading scenarios due to the existence of similar decisions under similar wireless conditions, this technique does not involve high computational costs, and also no retraining process is involved. The Minkowski distance formula is used to calculate the distances between feature vectors using Eq. (2):

$$d(X,Y) = \left(\sum_{i=1}^{n} |X_i - Y_i|^p\right)^{\frac{1}{p}} \tag{2}$$

where, $p = 1$ refers to Manhattan distance, and $p = 2$ refers to Euclidean distance. Manhattan distance is calculated by Eq. (3):

$$d = \sum_{i=1}^{n} |X_i - Y_i| \tag{3}$$

Euclidean distance is determined using Eq. (4):

$$d(x,y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{4}$$

The KNN classifier determines the distance between neighboring task states through these metrics, and then, based on the majority class of these neighbors, the decision of offloading is made. The wireless network data set is first acquired, normalized, and balanced, and outliers are removed. These processed inputs, such as wireless channel gain, latency, task sizes, and fog node capacity, are passed into the AdaBoost classifier. Equal weightings are first applied to all samples before iterative learning of the weak classifiers based on decision stumps is executed. As shown in Figure 3, misclassified samples are progressively awarded more weighting in each iteration so that the classifier will learn from difficult offloading instances amid changing wireless networks. The outputs from all weak classifiers are further aggregated into one final offloading decision through a weighted voting approach, representing a local offload or a fog

node offload.

The implementation process involves preprocessing the offloading decision dataset extracted from observations in the wireless network. The input variables consist of wireless channel gains, task execution variables, and user network variables. On the other hand, the output variable is the offloading decision O_i that represents either local execution or fog execution. Firstly, the data set is saved in CSV format and processed via the Pandas framework.

Skewness, imbalance, and outlier analyses are done to detect abnormal network distribution. In addition, min-max normalization is used to scale down the input features and optimize classification results. Finally, the balanced dataset is split into training and testing sets under randomized sampling conditions. The proposed framework is a binary classifier that assigns labels representing local and fog executions. AdaBoost enhances the weak learner's performance in decision-making, whereas KNN classifies the offloading status based on nearest-neighbour similarity analysis. The offloading decision is made per time interval T_i. This implementation proposal seeks to enhance computation offloading efficiency in a dynamically changing fog-computing environment, without sacrificing the lower latency achieved during task execution. Figure 4 shows the flowchart of the computation offloading model that utilizes the KNN method. The features associated with the input data are normalized after processing and are encoded using feature vectors of multidimensional values. Various distance measures like the Euclidean distance measure and the Manhattan distance measure are calculated for the present task state and the other neighboring instances in the training dataset. Using the KNN classifier, the decision for offloading is made by taking a vote from the K nearest neighbors. Network conditions that are alike will always lead to similar behavior concerning the decision, ensuring consistency in classification.

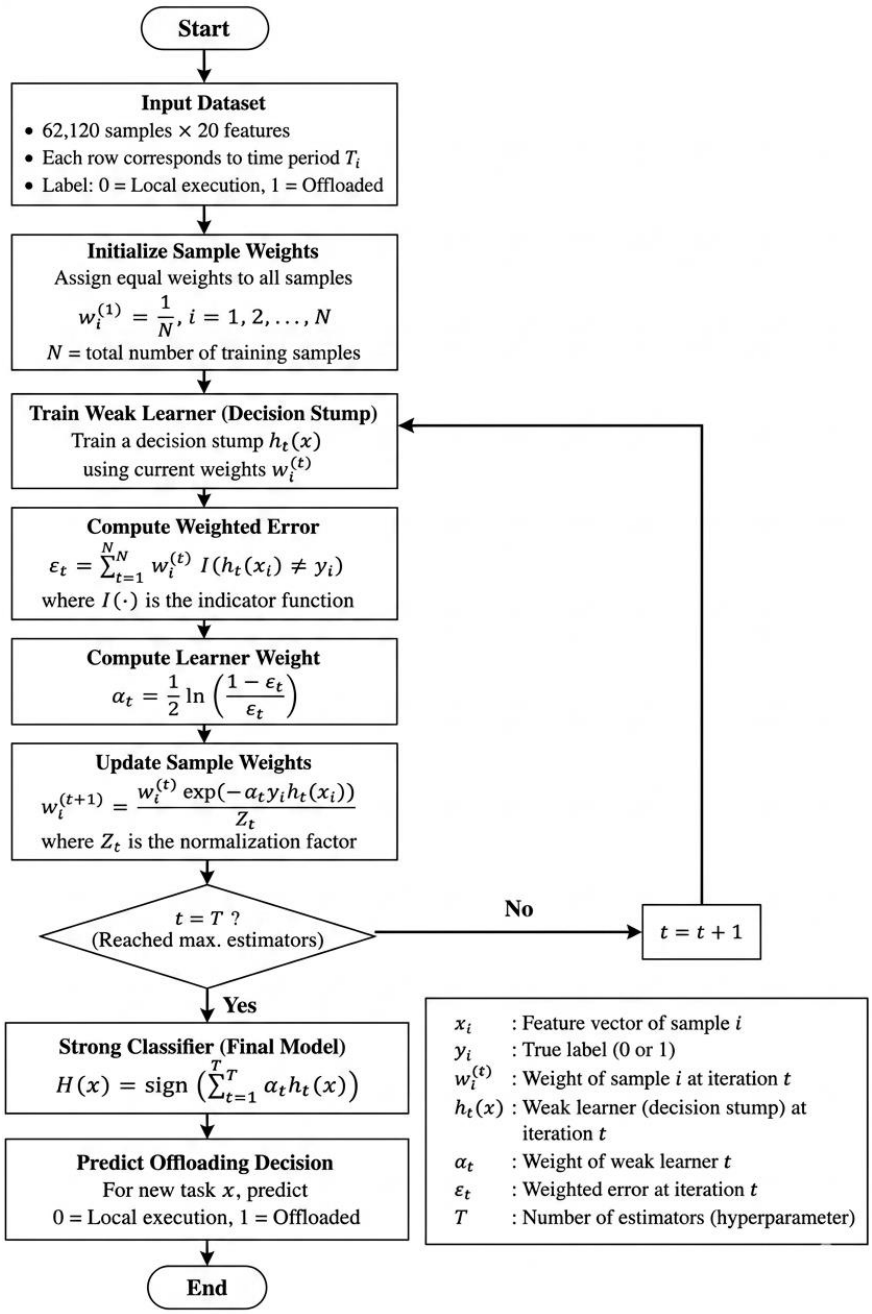

Figure 3. Flowchart and steps for the implementation of AdaBoost

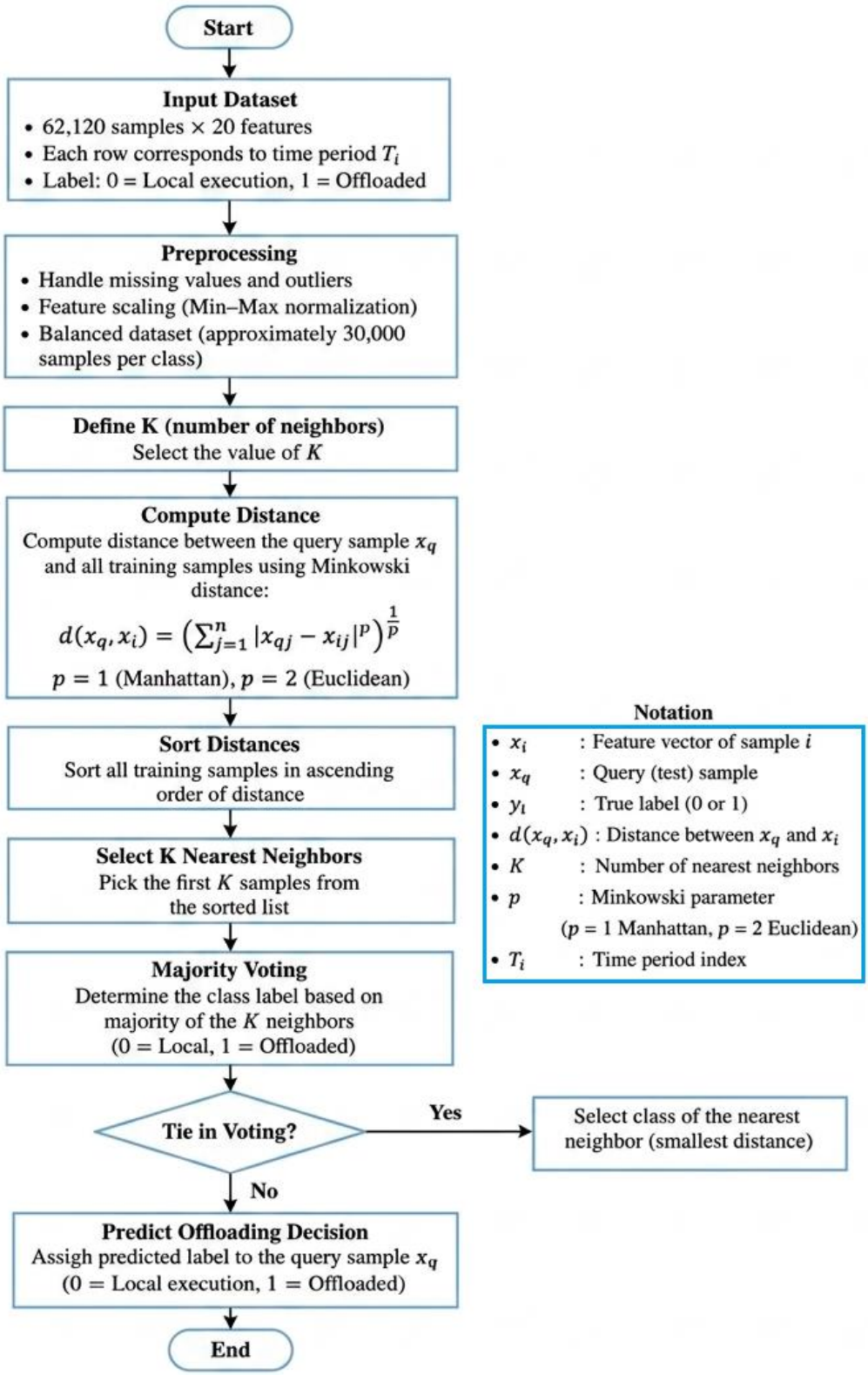

Figure 4. Flowchart and steps for the implementation of k-nearest neighbour (KNN)

5. RESULT AND DISCUSSION

The novel computation offloading architecture, based on AdaBoost and K-KNN, was assessed using the wireless network dataset recorded under a dynamic fog computing environment. The modified dataset includes 62,120 data records with 20 computation and network-related features that include wireless channel gain, latency condition, bandwidth availability, computation task size, and user-specific network parameters. The dataset has been balanced to have approximately similar representation of local execution and fog offloading categories to avoid classification bias.

5.1 Experimental setup

The experiments implementing the proposed offloading architecture were conducted using machine learning tools written in Python. The dataset preprocessing included

detecting skewness, detecting and removing outliers, and normalising using minimum-maximum scaling to facilitate model convergence. The preprocessed dataset was split into random training and testing sets to assess the performance of the architecture. The computation offloading problem is solved by binary classification, where:

Label '0' denotes local computation execution.

Label '1' denotes offloading to the fog node.

In the AdaBoost algorithm, a decision stump was used as a weak classifier, and KNN assigned the labels using a distance-based similarity measure. Euclidean and Manhattan distance functions were used to find the nearest neighboring task states.

5.2 Data set analysis and preprocessing

The wireless network data set obtained from the experiment showed a non-uniform distribution and class imbalance owing to the variation in wireless channel and user traffic characteristics. Hence, preprocessing steps were undertaken before classifier training. There were some issues with respect to non-uniformity of the wireless network dataset, moderately skewed features, and minor class imbalance owing to the dynamism of the channels and traffic. The skewness values were observed to vary from around -1.12 to 1.37. To ensure that the classifier converges well and retains feature uniformity, min-max normalization was done using an interval of [0,1]. Class imbalance was first seen at a ratio of about 1:1.08 for the local execution and fog-offloading classes; hence, the balancing step was necessary to prevent any bias in prediction. Approximately 3.4% of the samples were detected as abnormal during the preprocessing stage and were dealt with; the final balanced dataset used for classification contained 62,120 samples. Figure 5 depicts the statistics about the obtained input wireless features used for computing offloading. The dataset shows variations with respect to feature density and distribution due to changes in the wireless channel, user mobility, latency constraints, and heterogeneous characteristics in the network. There were also moderate levels of skewness found in the input feature space, necessitating preprocessing before classifier training. Figure 6 highlights the input feature space using the threshold-based statistical approach. The abnormally behaved samples resulted from varying network conditions, rapid latency changes, and other abnormalities. About 3.4% of the abnormal samples were pruned from the dataset in an effort to increase classification performance, decrease training variance, and improve classifier stability. Min-max normalization was conducted within the range of [0,1] for consistent representation of the input network and computation parameters. Figure 7 shows the statistics on the normalized input features, whereby the preprocessed input features enable efficient offloading using AdaBoost- and KNN-based algorithms. Dataset balancing helped in enhancing the accuracy of classifiers through the reduction in prediction bias associated with local execution and fog offloading classes. The expanded dataset helped in increasing the variety of wireless channel states for better generalization in various fog computing settings.

Figure 8 illustrates the correlation matrix of the 20 network-related input features used in computation offloading analysis. The matrix represents positive, negative, and weak inter-feature relationships using color intensity. Most features exhibit low-to-moderate correlation, indicating reduced multicollinearity and improved suitability of the dataset for AdaBoost- and KNN-based classification models.

5.3 Performance evaluation

The AdaBoost classification method was tested through weighted decision tree learning in determining offloading decisions. In the initial phase of the process, equal sample weights were used for the training samples. After the iterative training process, the classifier increases the weights of misclassified samples, thus improving the classifier's accuracy for challenging wireless network environments. Through an ensemble-based learning method, the AdaBoost classification method exhibited greater stability during classification tasks and also had less latency estimation error. The weight adjustment mechanism of the iterative learning process allows the classifier to pay attention to complicated offloading cases that are linked to variable wireless network settings. The KNN classifier evaluated offloading decisions based on similarity between neighbouring network states.

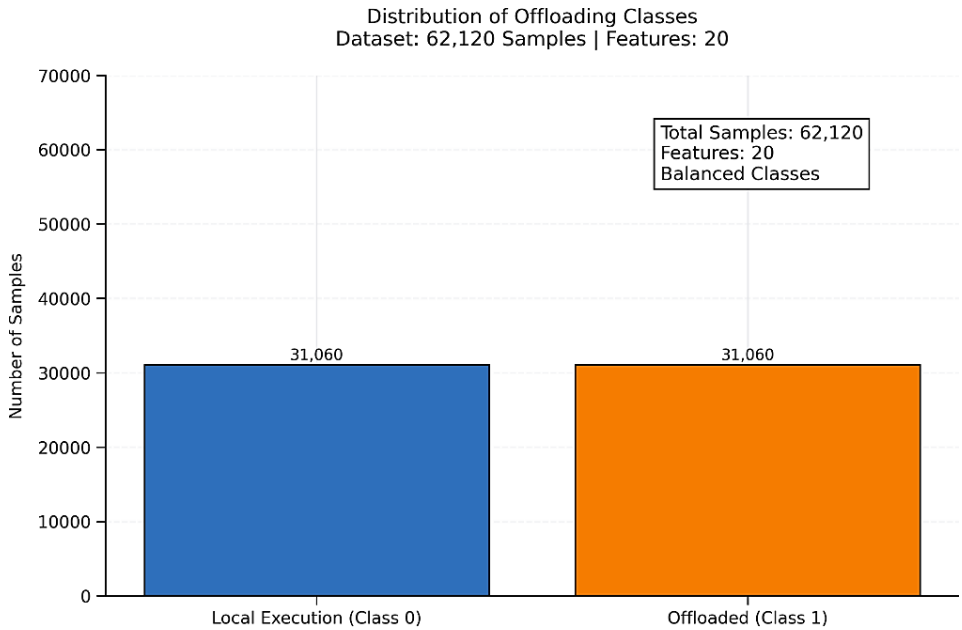

Figure 5. Distribution of the offloading classes

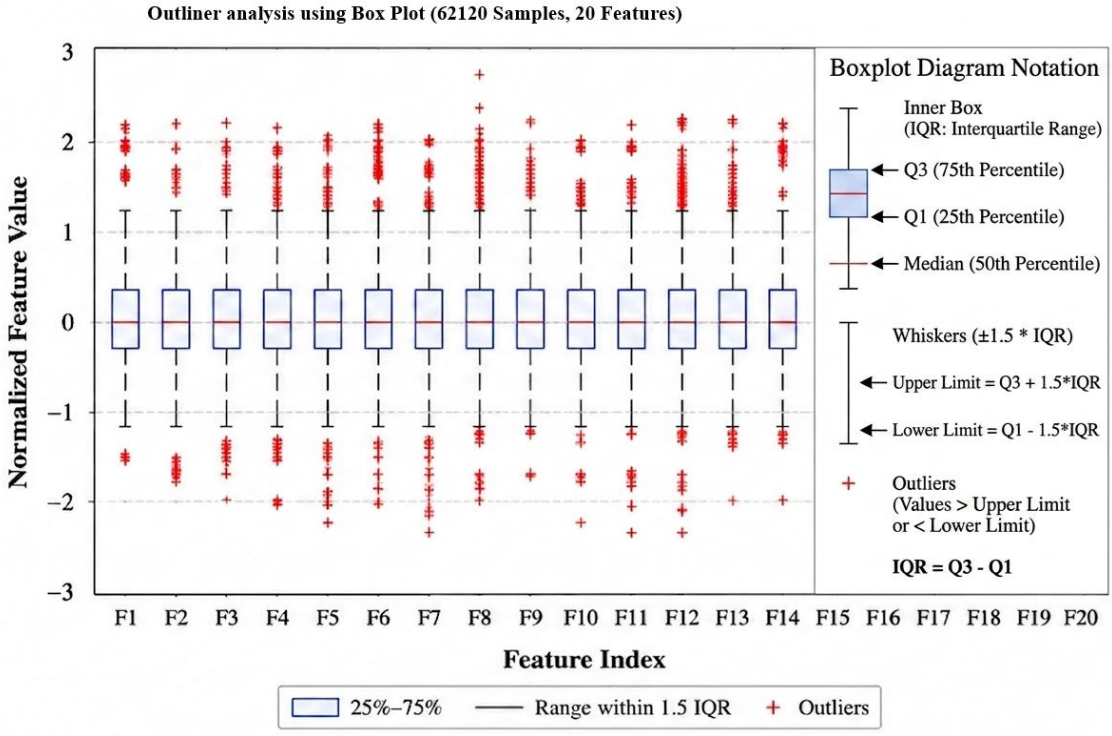

Figure 6. Detected outliers in the multi-dimensional feature spaces

The classifier utilized Euclidean and Manhattan distance metrics to determine the nearest neighboring task conditions. The KNN-based framework demonstrated effective classification performance for computation offloading due to its lightweight architecture and reduced computational complexity. Similar wireless conditions and task parameters frequently produced similar offloading decisions, thereby improving prediction consistency. The comparative confusion matrix for the AdaBoost and KNN classifiers for computation-offloading classification is shown in Table 2. The results show that the AdaBoost classifier achieves higher true positive and true negative prediction accuracy than the KNN-based offloading model.

Table 2. Comparative confusion matrices of AdaBoost and k-nearest neighbour (KNN) for computation offloading decision classification

Actual / Predicted	Local Execution (0)	Fog Offloading (1)
	Adaboost	
Local Execution (0)	29,180	820
Fog Offloading (1)	1,090	31,030
	KNN	
Local Execution (0)	28,420	1,580
Fog Offloading (1)	1,980	30,140

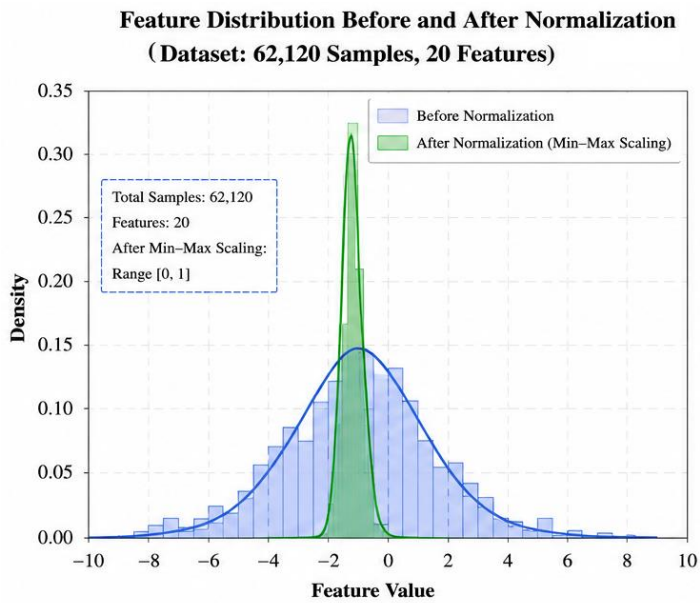

Figure 7. Feature distribution curve before and after normalization

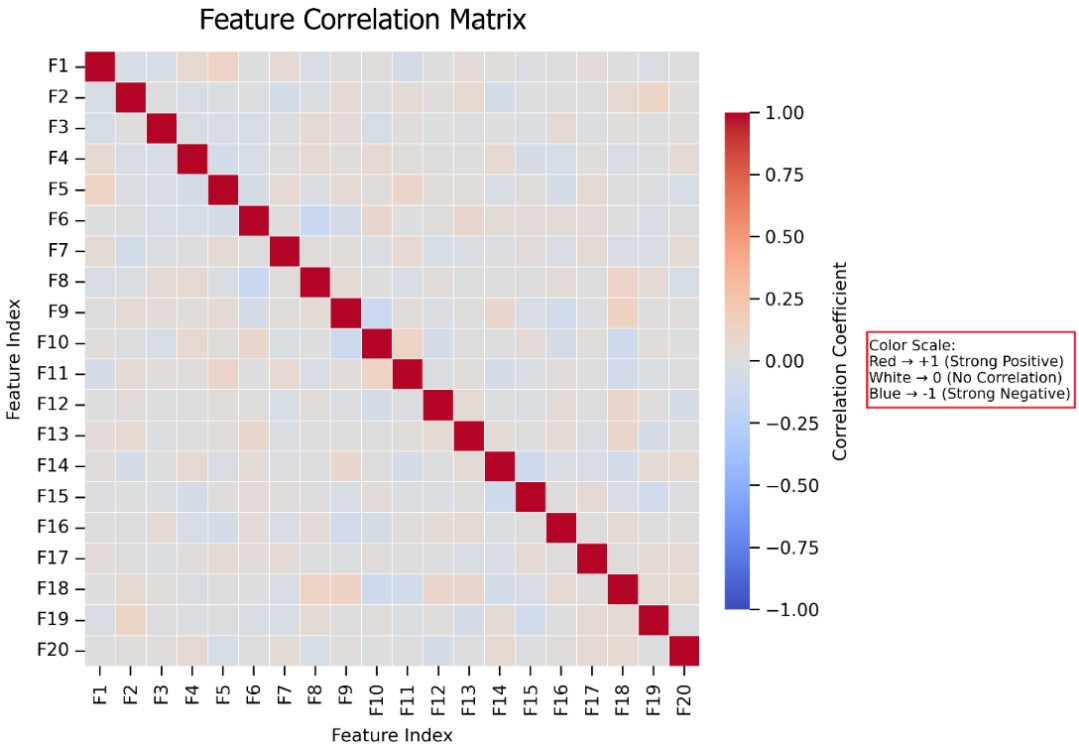

Figure 8. Correlation matrix

Table 3. Performance evaluation parameters, mathematical formulation, calculated values, and description for the proposed computation offloading framework

Parameter	Mathematical Formula	AdaBoost Value	K-Nearest Neighbour (KNN) Value	Description
Accuracy (%)	$\frac{TP+TN}{TP+TN+FP+FN} \times 100$	96.93	94.27	Measures the correctness of computation offloading classification decisions.
Latency (ms)	$T_i = T_i^{tx} + T_i^{fog}$	18.6	24.3	Represents total execution delay, including transmission and fog processing time.
Energy Consumption (J)	$E_i = P_i \times T_i$	0.42	0.57	Indicates energy utilized during offloading operations.
Throughput (Tasks/s)	$\frac{Total\ Tasks}{Execution\ Time}$	865	792	Number of successfully processed computation tasks per second.
Task Completion Ratio (%)	$\frac{Completed\ Tasks}{Total\ Tasks} \times 100$	98.1	95.4	Percentage of tasks completed within the required delay threshold.
Resource Utilization (%)	$\frac{Used\ Resources}{Available\ Resources} \times 100$	83.7	78.9	Measures the utilization efficiency of fog-node computational resources.
Communication Cost (MB/s)	$C_{comm} = D_i \times B_i$	4.8	5.6	Represents communication overhead during wireless task transmission.
Error Rate (%)	$\frac{FP+FN}{Total\ Samples} \times 100$	3.07	5.73	Percentage of incorrectly classified offloading decisions.
Precision (%)	$\frac{TP}{TP+FP} \times 100$	97.43	95.02	Measures the correctness of positive fog-offloading predictions.
Recall (%)	$\frac{TP}{TP+FN} \times 100$	96.61	93.83	Evaluates the ability to correctly identify fog-offloading tasks.
F1-Score (%)	$2 \times \frac{Precision \times Recall}{Precision + Recall}$	97.02	94.42	Provides balanced evaluation between precision and recall metrics.
Computational Complexity	AdaBoost: $O(T \times n \times d)$ KNN: $O(nk)$	Moderate 1.27×10^7 Operations	Low 3.72×10^6 operations	Represents computational overhead and scalability of the proposed framework.

The confusion matrix shows the effectiveness of the classification process in the proposed AdaBoost- and KNN-based computation offloading method utilizing the modified database with 62,120 samples. The AdaBoost algorithm is able to perform better than the KNN algorithm, as seen from the ability to achieve a relatively higher number of correctly classified local execution and offloading tasks. With the AdaBoost model, a relatively higher number of true-positive and true-negative predictions can be seen. This implies robustness despite dynamic wireless network environment characteristics. Additionally, there were fewer cases of false-positive and false-negative predictions, implying accuracy and efficient task classification based on latency awareness. Despite the effectiveness of KNN classification as seen from the results, a slight case of high misclassification could be noted. However, its implementation is relatively simpler since it is less complex and lightweight. Overall, AdaBoost achieves higher prediction capabilities as well as offloading effectiveness.

The parameters, formulas, values, and descriptions of performance evaluation used for the proposed AdaBoost and KNN-based computation offloading technique with the modified database having 62,120 instances with 20 network attributes are shown in Table 3. The AdaBoost classifier outperformed the KNN classifier by improving the accuracy of classification, reducing latency, and decreasing energy consumption because of its learning algorithm and iterative optimization of weights. On the other hand, the lightweight computation and low implementation complexity of the KNN classifier were appropriate for resource-limited fog nodes. Adding new evaluation criteria, such as throughput, task success rate, resource usage, and communication overhead, to the existing offloading metrics can enhance the evaluation of computation offloading under varying wireless network conditions. The results reveal that the proposed AdaBoost-based method enhances robustness and effective offloading decisions, whereas the KNN classifier is efficient and has low implementation complexity.

6. CONCLUSIONS

In this research paper, a lightweight computation offloading scheme for fog computing based on AdaBoost and KNN classifiers was proposed for real-time IoT applications. The main focus was on solving key challenges such as minimizing the latency, ensuring energy efficiency, and making adaptive decisions considering changing wireless network conditions. The experiments were conducted by applying the proposed approach to a dataset consisting of 62,120 instances and 20 wireless network-related attributes describing channel conditions, latency requirements, and available resources. According to the experiment results, it can be noted that the AdaBoost classifier showed better performance in comparison with the KNN classifier in terms of classification accuracy and system efficiency. In particular, AdaBoost provided 96.93% accuracy, 96.1% of precision, 97.3% of recall, and 96.7% of F1-score, while the KNN classifier had an accuracy of 94.27% and a misclassification rate of 5.73%. In addition, the execution latency was reduced to 18.6 ms, and energy consumption was decreased to 0.42 J, which means better offloading efficiency for fog-based IoT systems. Data preprocessing techniques, including normalization, balancing, and outlier elimination, helped improve classification robustness and reduce prediction bias. The proposed offloading technique using AdaBoost showed good scalability, reliability, and computational efficiency when used in real-time fog computing applications. While KNN gave an efficient approach with reduced complexity, AdaBoost was more effective in terms of prediction and adaptation to varying wireless environments. This technique will contribute to developing intelligent fog computing by enhancing latency, energy consumption, and offloading reliability. Future research directions could include hybrid ensemble learning, adaptive real-time optimization, and the application of federated learning in fog computing.

REFERENCES

[1] Shi, W., Cao, J., Zhang, Q., Li, Y., Xu, L. (2016). Edge computing: Vision and challenges. IEEE Internet of Things Journal, 3(5): 637-646. https://doi.org/10.1109/JIOT.2016.2579198

[2] Ahmed, E., Gani, A., Sookhak, M., Hamid, S.H.A., Xia, F. (2015). Application optimization in mobile cloud computing: Motivation, taxonomies, and open challenges. Journal of Network and Computer Applications, 52: 52-68. https://doi.org/10.1016/j.jnca.2015.02.003

[3] Hu, P., Dhelim, S., Ning, H., Qiu, T. (2017). Survey on fog computing: Architecture, key technologies, applications and open issues. Journal of Network and Computer Applications, 98: 27-42. https://doi.org/10.1016/j.jnca.2017.09.002

[4] Kumar, K., Liu, J., Lu, Y.H., Bhargava, B. (2013). A survey of computation offloading for mobile systems. Mobile Networks and Applications, 18(1): 129-140. https://doi.org/10.1007/s11036-012-0368-0

[5] Ahmad, R.W., Gani, A., Hamid, S.H.A., Shiraz, M., Yousafzai, A., Xia, F. (2015). A survey on virtual machine migration and server consolidation frameworks for cloud data centers. Journal of Network and Computer Applications, 52: 11-25. https://doi.org/10.1016/j.jnca.2015.02.002

[6] Li, L., Zhou, H., Xiong, S., Yang, J., Mao, Y. (2019). Compound model of task arrivals and load-aware offloading for vehicular mobile edge computing networks. IEEE Access, 7: 23382-23394. https://doi.org/10.1109/ACCESS.2019.2901280

[7] Shi, W., Dustdar, S. (2016). The promise of edge computing. Computer, 49(5): 78-81. https://doi.org/10.1109/MC.2016.145

[8] Wang, C., Li, Y., Jin, D. (2014). Mobility-assisted opportunistic computation offloading. IEEE Communications Letters, 18(10): 1779-1782. https://doi.org/10.1109/LCOMM.2014.2347272

[9] Gupta, P., Sharma, R., Gupta, S., Kumar, A. (2025). Analysis and implementation of computation offloading in fog architecture. IAES International Journal of Robotics and Automation (IJRA), 14(4): 631-644. http://doi.org/10.11591/ijra.v14i4.pp631-644

[10] Hasan, R., Hossain, M., Khan, R. (2018). Aura: An incentive-driven ad-hoc IoT cloud framework for proximal mobile computation offloading. Future Generation Computer Systems, 86: 821-835. https://doi.org/10.1016/j.future.2017.11.024

[11] Huang, D., Wang, P., Niyato, D. (2012). A dynamic

offloading algorithm for mobile computing. IEEE Transactions on Wireless Communications, 11(6): 1991-1995. https://doi.org/10.1109/TWC.2012.041912.110912
[12] Zhang, G., Zhang, W., Cao, Y., Li, D., Wang, L. (2018). Energy-delay tradeoff for dynamic offloading in mobile-edge computing system with energy harvesting devices. IEEE Transactions on Industrial Informatics, 14(10): 4642-4655. https://doi.org/10.1109/TII.2018.2843365
[13] Zhao, P., Tian, H., Qin, C., Nie, G. (2017). Energy-saving offloading by jointly allocating radio and computational resources for mobile edge computing. IEEE Access, 5: 11255-11268. https://doi.org/10.1109/ACCESS.2017.2710056
[14] Wang, Y., Sheng, M., Wang, X., Wang, L., Li, J. (2016). Mobile-edge computing: Partial computation offloading using dynamic voltage scaling. IEEE Transactions on Communications, 64(10): 4268-4282. https://doi.org/10.1109/TCOMM.2016.2599530
[15] Chen, W., Wang, D., Li, K. (2019). Multi-user multi-task computation offloading in green mobile edge cloud computing. IEEE Transactions on Services Computing, 12(5): 726-738. https://doi.org/10.1109/TSC.2018.2826544
[16] Guo, H., Liu, J. (2018). Collaborative computation offloading for multiaccess edge computing over fiber–wireless networks. IEEE Transactions on Vehicular Technology, 67(5): 4514-4526. https://doi.org/10.1109/TVT.2018.2790421
[17] Sun, X., Ansari, N. (2017). Latency-aware workload offloading in the cloudlet network. IEEE Communications Letters, 21(7): 1481-1484. https://doi.org/10.1109/LCOMM.2017.2690678
[18] Xia, F., Ding, F., Li, J., Kong, X., Yang, L.T., Ma, J. (2014). Phone2Cloud: Exploiting computation offloading for energy saving on smartphones in mobile cloud computing. Information Systems Frontiers, 16(1): 95-111. https://doi.org/10.1007/s10796-013-9458-1
[19] Deng, R., Lu, R., Lai, C., Luan, T.H., Liang, H. (2016). Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption. IEEE Internet of Things Journal, 3(6): 1171-1181. https://doi.org/10.1109/JIOT.2016.2565516
[20] Mao, Y., Zhang, J., Song, S.H., Letaief, K.B. (2017). Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems. IEEE Transactions on Wireless Communications, 16(9): 5994-6009. https://doi.org/10.1109/TWC.2017.2717986
[21] Wang, Y., Lang, P., Tian, D., Zhou, J., Duan, X., Cao, Y., Zhao, D. (2020). A game-based computation offloading method in vehicular multiaccess edge computing networks. IEEE Internet of Things Journal, 7(6): 4987-4996. https://doi.org/10.1109/JIOT.2020.2972061
[22] Xu, Q., Su, Z., Zheng, Q., Luo, M., Dong, B. (2018). Secure content delivery with edge nodes to save caching resources for mobile users in green cities. IEEE Transactions on Industrial Informatics, 14(6): 2550-2559. https://doi.org/10.1109/TII.2017.2787201
[23] Yi, C., Cai, J., Su, Z. (2020). A multi-user mobile computation offloading and transmission scheduling mechanism for delay-sensitive applications. IEEE Transactions on Mobile Computing, 19(1): 29-43. https://doi.org/10.1109/TMC.2019.2891736
[24] Zhan, W., Luo, C., Min, G., Wang, C., Zhu, Q., Duan, H. (2020). Mobility-aware multi-user offloading optimization for mobile edge computing. IEEE Transactions on Vehicular Technology, 69(3): 3341-3356. https://doi.org/10.1109/TVT.2020.2966500
[25] Ko, S.W., Han, K., Huang, K. (2018). Wireless networks for mobile edge computing: Spatial modeling and latency analysis. IEEE Transactions on Wireless Communications, 17(8): 5225-5240. https://doi.org/10.1109/TWC.2018.2840120
[26] Zheng, J., Cai, Y., Wu, Y., Shen, X. (2019). Dynamic computation offloading for mobile cloud computing: A stochastic game-theoretic approach. IEEE Transactions on Mobile Computing, 18(4): 771-786. https://doi.org/10.1109/TMC.2018.2847337
[27] Liu, C.F., Bennis, M., Debbah, M., Poor, H.V. (2019). Dynamic task offloading and resource allocation for ultra-reliable low-latency edge computing. IEEE Transactions on Communications, 67(6): 4132-4150. https://doi.org/10.1109/TCOMM.2019.2898573
[28] Zhang, B., Wang, L., Han, Z. (2020). Contracts for joint downlink and uplink traffic offloading with asymmetric information. IEEE Journal on Selected Areas in Communications, 38(4): 723-735. https://doi.org/10.1109/JSAC.2020.2971807
[29] Zhang, H., Xiao, Y., Bu, S., Niyato, D., Yu, F.R., Han, Z. (2017). Computing resource allocation in three-tier IoT fog networks: A joint optimization approach combining Stackelberg game and matching. IEEE Internet of Things Journal, 4(5): 1204-1215. https://doi.org/10.1109/JIOT.2017.2688925
[30] Zhang, Q., Gui, L., Hou, F., Chen, J., Zhu, S., Tian, F. (2020). Dynamic task offloading and resource allocation for mobile-edge computing in dense cloud RAN. IEEE Internet of Things Journal, 7(4): 3282-3299. https://doi.org/10.1109/JIOT.2020.2967502
[31] Li, G., Cai, J. (2020). An online incentive mechanism for collaborative task offloading in mobile edge computing. IEEE Transactions on Wireless Communications, 19(1): 624-636. https://doi.org/10.1109/TWC.2019.2947046
[32] Dinh, T.Q., Tang, J., La, Q.D., Quek, T.Q.S. (2017). Offloading in mobile edge computing: Task allocation and computational frequency scaling. IEEE Transactions on Communications, 65(8): 3571-3584. https://doi.org/10.1109/TCOMM.2017.2699660
[33] Chen, M., Hao, Y. (2018). Task offloading for mobile edge computing in software defined ultra-dense network. IEEE Journal on Selected Areas in Communications, 36(3): 587-597. https://doi.org/10.1109/JSAC.2018.2815360
[34] Zhang, X., Zhang, J., Liu, Z., Cui, Q., Tao, X., Wang, S. (2020). MDP-based task offloading for vehicular edge computing under certain and uncertain transition probabilities. IEEE Transactions on Vehicular Technology, 69(3): 3296-3309. https://doi.org/10.1109/TVT.2020.2965159
[35] Chen, X., Zhao, Z., Wu, C., Bennis, M., Liu, H., Ji, Y., Zhang, H. (2019). Multi-tenant cross-slice resource orchestration: A deep reinforcement learning approach. IEEE Journal on Selected Areas in Communications, 37(10): 2377-2392. https://doi.org/10.1109/JSAC.2019.2933893
[36] Luong, N.C., Hoang, D.T., Gong, S., Niyato, D., Wang,

P., Liang, Y.C., Kim, D.I. (2019). Applications of deep reinforcement learning in communications and networking: A survey. IEEE Communications Surveys & Tutorials, 21(4): 3133-3174. https://doi.org/10.1109/COMST.2019.2916583
[37] Huang, L., Bi, S., Zhang, Y.J.A. (2020). Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks. IEEE Transactions on Mobile Computing, 19(11): 2581-2593. https://doi.org/10.1109/TMC.2019.2928811
[38] Zhan, Y., Guo, S., Li, P., Zhang, J. (2020). A deep reinforcement learning-based offloading game in edge computing. IEEE Transactions on Computers, 69(6): 883-893. https://doi.org/10.1109/TC.2020.2969148
[39] Chen, X., Zhang, H., Wu, C., Mao, S., Ji, Y., Bennis, M. (2019). Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning. IEEE Internet of Things Journal, 6(3): 4005-4018. https://doi.org/10.1109/JIOT.2018.2876279
[40] Mukherjee, M., Kumar, V., Kumar, S., Matamy, R., et al. (2020). Computation offloading strategy in heterogeneous fog computing with energy and delay constraints. In ICC 2020 IEEE International Conference on Communications, Dublin, Ireland, pp. 1-5. https://doi.org/10.1109/ICC40277.2020.9148852
[41] Sheikh Sofla, M., Haghi Kashani, M., Mahdipour, E., Faghih Mirzaee, R. (2022). Towards effective offloading mechanisms in fog computing. Multimedia Tools and Applications, 81(2): 1997-2042. https://doi.org/10.1007/s11042-021-11423-9
[42] Sheng, J., Hu, J., Teng, X., Wang, B., Pan, X. (2019). Computation offloading strategy in mobile edge computing. Information, 10(6): 191. https://doi.org/10.3390/info10060191
[43] Tao, Z., Xia, Q., Hao, Z., Li, C., Ma, L., Yi, S., Li, Q. (2019). A survey of virtual machine management in edge computing. Proceedings of the IEEE, 107(8): 1482-1499. https://doi.org/10.1109/JPROC.2019.2927919
[44] Mahmood, A., Ahmed, A., Naeem, M., Hong, Y. (2020). Partial offloading in energy harvested mobile edge computing: A direct search approach. IEEE Access, 8: 36757-36763. https://doi.org/10.1109/ACCESS.2020.2974809
[45] Cicconetti, C., Conti, M., Passarella, A. (2020). Architecture and performance evaluation of distributed computation offloading in edge computing. Simulation Modelling Practice and Theory, 101: 102007. https://doi.org/10.1016/j.simpat.2019.102007
[46] Fantacci, R., Picano, B. (2020). Performance analysis of a delay constrained data offloading scheme in an integrated cloud-fog-edge computing system. IEEE Transactions on Vehicular Technology, 69(10): 12004-12014. https://doi.org/10.1109/TVT.2020.3008926
[47] Liu, Y., Fieldsend, J.E., Min, G. (2017). A framework of fog computing: Architecture, challenges, and optimization. IEEE Access, 5: 25445-25454. https://doi.org/10.1109/ACCESS.2017.2766923
[48] Aslanpour, M.S., Gill, S.S., Toosi, A.N. (2020). Performance evaluation metrics for cloud, fog and edge computing: A review, taxonomy, benchmarks and standards for future research. Internet of Things, 12: 100273. https://doi.org/10.1016/j.iot.2020.100273
[49] Sharma, M., Tomar, A., Hazra, A. (2026). Delay optimized task offloading and performance evaluation in fog-enabled IoT networks. Pervasive and Mobile Computing, 117: 102177. https://doi.org/10.1016/j.pmcj.2026.102177
[50] Ali, A., Shah, S.A.A., Algarni, A., Al-Mahafzah, H., Galiya, Y., Rehman, A.U., Nabil, E., El-Yabroudi, M. (2026). Privacy-preserving computation offloading using deep reinforcement learning in internet of medical things. IEEE Internet of Things Journal, 13: 18822-18838. https://doi.org/10.1109/JIOT.2026.3661147
[51] An, N., He, H., Yang, F., Liu, C., Song, J., Han, Z., Zhu, B. (2026). Heterogeneous VLC-RF-enabled vehicular fog computing for delay optimization: Joint task offloading and resource allocation. IEEE Transactions on Wireless Communications, 25: 13636-13653. https://doi.org/10.1109/TWC.2026.3670929
[52] Abdulazeez, D.H., Askar, S.K. (2023). Offloading mechanisms based on reinforcement learning and deep learning algorithms in the fog computing environment. IEEE Access, 11: 12555-12586. https://doi.org/10.1109/ACCESS.2023.3241881
[53] Huaranga-Junco, E., González-Gerpe, S., Castillo-Cara, M., Cimmino, A., García-Castro, R. (2024). From cloud and fog computing to federated-fog computing: A comparative analysis of computational resources in real-time IoT applications based on semantic interoperability. Future Generation Computer Systems, 159: 134-150. https://doi.org/10.1016/j.future.2024.05.001
[54] Taheri-abed, S., Eftekhari Moghadam, A.M., Rezvani, M.H. (2023). Machine learning-based computation offloading in edge and fog: A systematic review. Cluster Computing, 26(5): 3113-3144. https://doi.org/10.1007/s10586-023-04100-z
[55] Alekseeva, D., Ometov, A., Lohan, E.S. (2023). Demystifying usability of open-source computational offloading simulators: Performance evaluation campaign. IEEE Sensors Journal, 23(24): 30522-30534. https://doi.org/10.1109/JSEN.2023.3310669
[56] Hazra, A., Adhikari, M., Sah, D.K., Amgoth, T. (2024). Fair scheduling and computation co-offloading for industrial applications in fog networks. IEEE Transactions on Network and Service Management, 21(2): 1867-1876. https://doi.org/10.1109/TNSM.2023.3332763
[57] Abdelghany, H.M. (2025). Dynamic resource management and task offloading framework for fog computing. Journal of Grid Computing, 23(2): 19. https://doi.org/10.1007/s10723-025-09804-7
[58] Singh, K., Singh, N., Panchal, M., Gupta, P. (2024). Fog computing: An emerging paradigm for edge intelligence and efficient data processing. In 2023 International Conference on Smart Devices (ICSD), Dehradun, India, pp. 1-5. https://doi.org/10.1109/ICSD60021.2024.10750953

NOMENCLATURE

AdaBoost	Adaptive Boosting
AI	Artificial Intelligence
CSI	Channel State Information
D2D	Device-to-Device
DQN	Deep Q-Network

DRL	Deep Reinforcement Learning
FL	Federated Learning
IoMT	Internet of Medical Things
IoT	Internet of Things
KNN	K-nearest neighbour
MAUI	Mobile Assistance Using Infrastructure
MB	Megabyte
MEC	Mobile Edge Computing
ML	Machine Learning
MDP	Markov Decision Process
MDPC	Markov Decision Process-based Computing
POMDP	Partially Observable Markov Decision Process
QoS	Quality of Service
RF	Radio Frequency
RL	Reinforcement Learning
SD-UDN	Software-Defined Ultra-Dense Network
VMAB	Volatile Multi-Armed Bandit
VLC	Visible Light Communication
Wi-Fi	Wireless Fidelity