Analysis and implementation of computation offloading in fog architecture

Prince Gupta[1,2], Rajeev Sharma[3], Sachi Gupta[4], Adesh Kumar[5]
[1]Computer Science and Information Technology, KIET Group of Institutions, Delhi-NCR, Ghaziabad, UP., India
[2]Department of Computer Science and Engineering, SRM Institute of Science and Technology, Delhi NCR Campus, Delhi-Meerut Road, Modinagar, Ghaziabad, U.P., India
[3]Department of Computer Applications, SRM Institute of Science and Technology, Delhi NCR Campus, Delhi-Meerut Road, Modinagar, Ghaziabad, U.P., India
[4]Department of Artificial Intelligence and Machine Learning, Galgotias College of Engineering and Technology, Knowledge Park-II, Greater Noida, U.P., India
[5]Department of Electrical and Electronics Engineering, School of Advanced Engineering, UPES, Dehradun, India

Article Info

Article history:

Received Apr 17, 2025
Revised Aug 25, 2025
Accepted Nov 11, 2025

Keywords:

Adaptive boosting
Computation offloading
Edge computing
Fog computing
KNN algorithm
Latency optimization
Resource allocation

ABSTRACT

The fast expansion of connected devices has led to an unparalleled increase in data across sectors like industrial automation, social media, environmental monitoring, and life sciences. The processing of this data presents difficulties owing to its magnitude, temporal urgency, and security stipulations. Computation offloading has arisen as a viable alternative, allowing resource-constrained devices to assign demanding work to more robust platforms, thus improving responsiveness and efficiency. This paper examines decision-making strategies for computing offloading by assessing various algorithms, including a deep neural network with deep reinforcement learning (DNN-DRL), coordinate descent (baseline), AdaBoost, and K-nearest neighbor (KNN). The performance evaluation centers on three primary metrics: system accuracy, training duration, and latency. The computation offloading mitigates these issues by transferring intricate workloads from resource-limited devices to more proficient platforms, thus enhancing efficiency and responsiveness. The evaluation examines accuracy, training duration, and latency as key parameters. The results indicate that KNN attains maximum accuracy and minimal latency, AdaBoost provides a robust balance despite increased training costs, and the baseline underperforms in both efficiency and responsiveness. These findings underscore the trade-offs between computational expense, precision, and real-time application, providing insights for forthcoming IoT and edge-computing systems.

Corresponding Author:

Prince Gupta
Computer Science and Information Technology, KIET Group of Institutions
Delhi-NCR, Ghaziabad, India
Email: prince.rkg@gmail.com

1. INTRODUCTION

Computation offloading in fog architecture has become an effective method to address the growing requirements of latency-sensitive and resource-intensive applications. The swift proliferation of Internet of Things (IoT) devices frequently challenges traditional cloud computing in managing real-time processing because of significant transmission delays and capacity constraints. Fog computing addresses this issue by bringing processing and storage resources nearer to the network's edge, hence diminishing dependence on remote cloud servers. Delegating duties from resource-limited end devices to proximate fog nodes not only optimizes reaction time but also augments energy efficiency and service reliability. This paradigm enables applications like smart healthcare, autonomous vehicles, and industrial automation to operate with enhanced precision and efficiency. Effective offloading necessitates meticulous attention to job scheduling, network circumstances, and resource allocation to attain optimal performance while preserving system scalability and user experience quality. The cloud reduces the need to invest in physical storage, offers layered security mechanisms for authorized users, and has made service accessibility relatively affordable. Nevertheless, everything has benefits and drawbacks. Moreover, many new companies are creating online presences without a strong security agreement, which has attracted the attention of security attackers who are developing numerous ways to breach the secured system due to the fast adoption of cloud services. Through internet connectivity, mobile gadgets are growing smarter and are being used extensively over the globe [1]. The IoT has gained increasing relevance over the past several years because of numerous technological advancements. Large-scale wireless sensor networks, computers, and connected cars are among examples [2], [3]. IoT devices will be able to carry out computationally demanding activities like surveillance, interactive online games, and face recognition as they become more advanced. Due to resource constraints, the devices, however, can struggle to run these applications, which might give end users an undesirable level of experience [4]. For more than ten years [5], IoT applications have relied on cloud computing services to free mobile devices from computationally demanding activities. Cloud computing also benefits from virtualization, which provides an isolated environment for IoT facilities to operate on the cloud [6]. The reason compute offloading has gained so much attention is largely because of this support. As a result, as compute duties are offloaded from resource-limited IoT devices to resource-rich cloud structures, the quality of the experience can be improved [7]. As a result, each program is divided into many granular levels, including tasks, classes, methods, and threads.

Computation-intensive partitions appropriate for remote execution are offloaded to cloud servers based on criteria such as mobile devices, their application modules, and network architecture [8]. Numerous offloading frameworks have been proposed, notably Mobile Assistance Using Infrastructure (MAUI) and Clone Cloud [9], [10]. The user experience is markedly enhanced, and the device's energy consumption is reduced when computations are delegated. The Cisco Annual Internet Report [11] evaluates and assesses digital transformation across several corporate sectors. Approximately two-thirds of the global population will gain Internet connectivity during the next decade. This amounts to 5.3 billion users worldwide. By 2023, IP networks will connect to over three times the global population [11]. Everyone will possess 3.6 devices connected to IP networks. By 2023, the average speed of 5G mobile connections is projected to grow thirteenfold. By 2023, 5G connectivity is expected to achieve speeds of 575 Mbps.

2. RELATED WORK

Innovative applications, including surveillance systems, traffic awareness, automatic driving, and industrial automation, are developing with the potential to increase the effectiveness and safety of the systems as IoT and wireless technology advance. These new and high-tech applications' requirements to analyze massive volumes of data to make defensible and informed decisions are one of their most fascinating features [12]. The host devices' limited processing resources are put under a lot of strain because of the high demand for computational power they must meet. Cloud is still the major method for offloading a device's significant computing load [13], but the long communication distance between the device and the cloud center might result in increased latency and inconsistent performance, which negatively impacts service quality. Offloading can even slow down entire networks. The offloading device-to-device (D2D) is covered in certain research [14], [15]. The analysis of communication patterns between mobile devices helped them investigate the mobility-assisted opportunistic computation offloading problem. It was determined how many computational tasks could be distributed across devices using convex optimization. The suggested device-to-device (D2D) fogging paradigm enables users to exchange computer and communication resources in a meaningful and dynamic manner. Users across the network can conserve energy while completing their activities by employing D2D fogging. The multicast-based task implementation architecture was suggested, known as multi-access edge computing (MEC). Multiple devices can collaborate at the network's edge [16] for wireless distributed computing and outcome sharing. The goal of such a system is to establish an energy-efficient task allocation policy for mobile users as well as strong D2D connections. They agreed to use a tree-based Monte-Carlo search technique to achieve their goal [17]. Fog-to-cloud offloading and cloud-to-cloud offloading were discussed in [18], [19], respectively. It has been evaluated the viability of mobile computing offloading has been evaluated using real-world scenarios. The architects thought about synchronizing each actual device with a cloud-based software replica. A technique for dynamic offloading was proposed that is energy-efficient and preserves the latency necessary for face recognition applications [20]. This technique uses Lyapunov optimization. The MEC system that uses energy harvesting technologies [21]. They developed a weighted average energy consumption and computed lag minimization for mobile devices while taking battery life and queue stability into account. This was done using the Lyapunov algorithm. The recent research in mobile edge computing highlights energy efficiency, delay reduction, and collaborative task execution as essential factors for optimizing offloading strategies. A method that concurrently allocates radio and computational resources has been proposed to reduce energy consumption while maintaining reliable service delivery in mobile edge systems [22]. Partial computation offloading through dynamic voltage scaling has shown potential in balancing processing efficiency and power savings, especially for resource-constrained mobile devices [23]. Another area of investigation has concentrated on scheduling techniques aimed at reducing delays. Models for delay-optimal task scheduling have been developed to minimize average response time and enhance service quality in dynamic mobile edge environments [24]. Subsequently, frameworks for multi-user and multi-task offloading have been established to enhance energy efficiency and resource distribution in green mobile edge cloud computing contexts [25]. The collaborative computation offloading across fibre-wireless networks has been investigated beyond individual devices. This method utilizes collaboration among various edge nodes to improve overall system efficiency and mitigate network congestion [26]. These studies emphasize that energy-aware allocation, delay-optimal scheduling, and cooperative offloading are essential components for the advancement of efficient and scalable mobile edge computing frameworks.

The latency-aware workload offloading (LEAD) problem was used [27] to create a task offloading problem and shorten the average response time for wireless users. By offloading assignments to the appropriate cloudlets, their LEAD technique reduced response times. It was recommended to use an ordered edge cloud architecture to improve the deployment of mobile computing, suggesting using cloud computing [28] and offloading mobile workloads for distant execution in the cloud. They utilized simulated annealing (SA) [29] to determine which applications should be executed on which edge cloud servers to optimize the placement of workloads and the deployment of resources. With a more complex topology, a simulation study was conducted, and a small-scale evaluation of a proposed workload placement algorithm was conducted. The experimentation with computer vision algorithms in the context of wearable computing is one of its objectives. Unfortunately, many of the needed algorithms cannot be executed by the wearable computers of the current generation due to their lack of central processing unit (CPU) power. Rather, these algorithms are created and evaluated using strong workstations, such as Silicon Graphics models [30]. Computation offloading in fog and edge environments demonstrates potential; however, current methods face challenges in achieving low latency, energy efficiency, and scalability within dynamic IoT ecosystems. The absence of a cohesive framework that accommodates resource diversity and variable workloads presents a significant obstacle to the advancement of next-generation applications, including autonomous driving, smart healthcare, and industrial automation.

3. METHOD

3.1. Offloading frameworks in Fog computing

3.1.1. MAUI

Through offloading, MAUI's primary goal is to maximize energy efficiency on smartphones. MAUI enables very dynamic offloading through a continual profiling process. With the help of this framework, it is possible to provide the impression to a mobile user that the complete application is operating on their device while hiding the complexity of a remote execution. According to developer observations showing which tasks may be completed remotely and which cannot, MAUI partitions are based, as illustrated in Figure 1. During the planning stage, two requirements must be met: i) the application binaries must be present on both the server and the mobile device, and ii) proxies, profilers, and solvers must be mounted on both platforms. The MAUI profiler first establishes the device's features before continuously monitoring the program and network attributes during the execution phase. Because these attributes are dynamic, any outdated or wrong measurement could cause MAUI to produce errors. Decisions regarding offloading are made at runtime. The framework decides which parts must be remotely executed based on the MAUI solver's conclusion. The MAUI profiler's opinions constitute the foundation for these choices. The framework for smartphones includes a solver, a profiler, and a proxy. Every time it is called, the MAUI profiler evaluates each method for its capacity to conserve energy and gathers data regarding the device and the network's status. The proxy, which relies on the calculations provided by the profiler, is responsible for transferring control and data between the server and the smartphone, in addition to the MAUI solver. The profiler and the server proxy carry out similar tasks on the server, just as their counterparts on the client-side. The MAUI controller verifies incoming calls and allocates resources.

3.1.2. Phone2Cloud

A framework for offloading computing is introduced. The project's objectives were to increase the application's performance and increase the energy efficiency of smartphones. The reduction in the system's energy consumption may be precisely measured by the authors by conducting application trials as well as scenario experiments. As depicted in Figure 2, a framework for semi-automatic cloud offloading is Phone2Cloud [30]. Before running apps in the cloud and obtaining the results, it is required to modify them during the preparation stage to make them compatible for execution on cloud servers. A static analysis is performed while taking user delay tolerance into account to determine whether to offload. This framework offers a straightforward model for forecasting possible Wi-Fi connectivity delays.

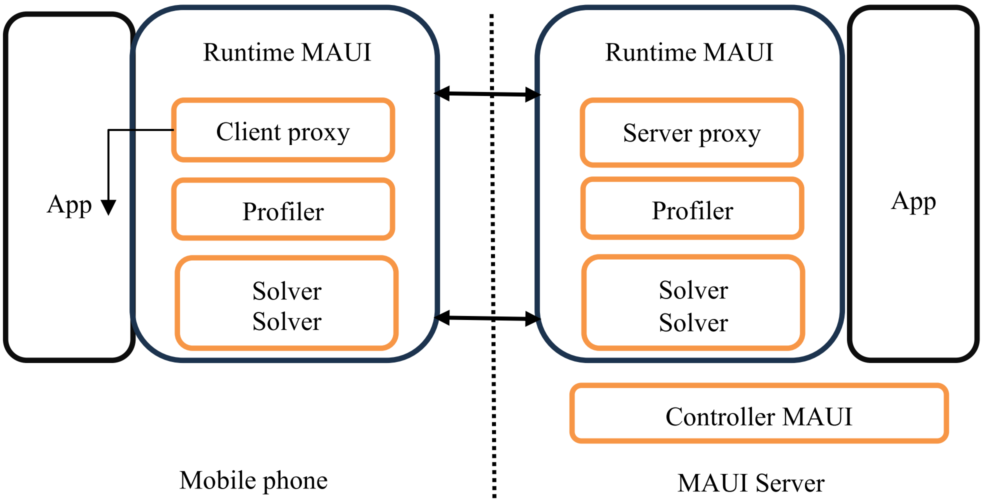

Figure 1. MAUI's architecture [10]

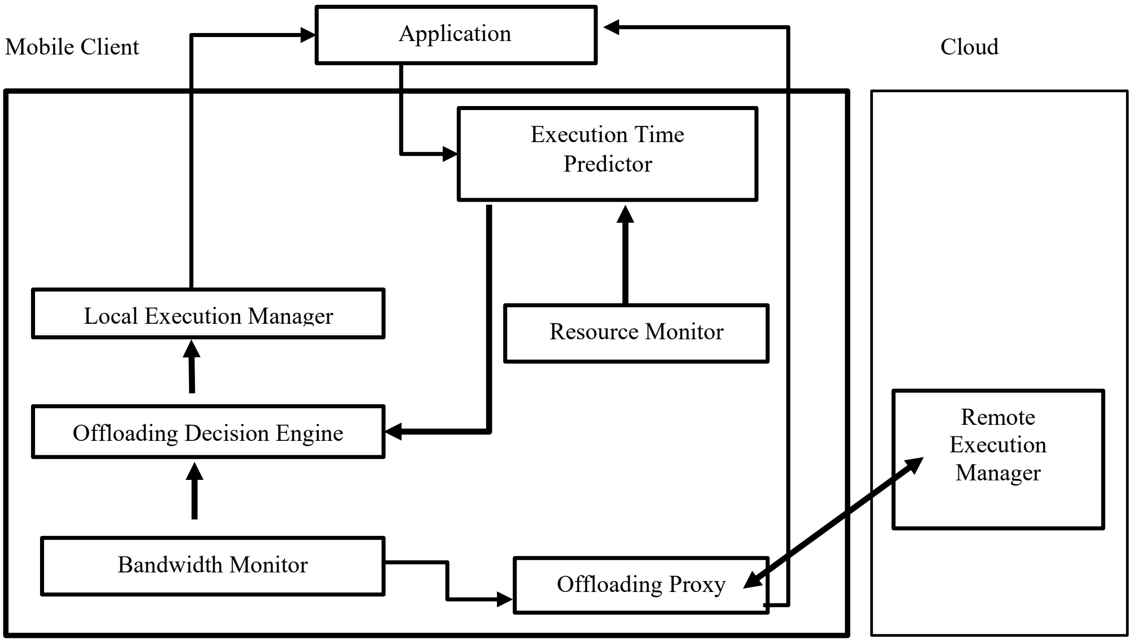

Figure 2. Architecture of Phone2Cloud [30]

3.2. Categories of offloading computation

In a fog environment, there are two types of computation offloading algorithms: model-based and model-free. The former uses a model to foretell how environmental situations will change. Additionally, model-free algorithms employ learning-by-trial methods to direct users to act by studying the surrounding settings.

3.2.1. Algorithms of model-based

In recent years, a variety of offloading strategies have been put forth to maximize task duration, energy consumption, and system utility alone or in combination [31]–[33], stochastic geometry, game theory [34]–[36], and matching theory. The existing algorithms can also be divided into two groups based on how many edge servers a single mobile user can offload tasks to. Offloading from a single server: The MEC systems with in-network caching functions were explored with a job offloading mechanism to maximize system usefulness under constraints such as radio spectrum, processing resources, and storage capacity. The analysis clarified a scenario for an MEC that uses environmental energy to carry out activities. The authors developed a heuristic approach by applying an algorithm for lowering energy usage without full knowledge of the Channel state information (CSI). It has been examined the task offloading conundrum by looking at the effect of user mobility [37], [38]. It has been proven that spatial modelling in wireless networks significantly decreases latency in MEC situations [39]. The concept was introduced with dynamic and game-theoretic offloading mechanisms that markedly enhance resource allocation and reliability [40], [41]. It has been demonstrated that contract-based and joint optimization methodologies improve task offloading efficiency, scalability, and performance in dense IoT and cloud-RAN networks [42], [43]. They improved the social welfare of mobile users and edge servers by encouraging them to collaborate with one another through the implementation of an incentive mechanism [44]. In [45], the matching theory that was applied was investigated to assess the probability of task offloading in MEC topologies to lower energy consumption while satisfying incentive-friendly circumstances.

3.2.2. Multi-server offloading

To balance energy consumption and resource consumption, the author of [46] proposed a method for allocating tasks and scaling CPU resources in MEC. In this method, work assignments and task durations for a single mobile user might be assigned to different edge servers. In a software-defined ultra-dense network (UDN) environment, task offloading has been used to solve the overall delay minimization challenge [47]. Mobile devices are intelligently directed to offload processing duties to edge servers based on global data gathered by a centralized software defined ultra-dense network (SD-UDN) controller. An MDP-based work offloading strategy has been implemented to mitigate delays in vehicle edge computing [48]. This algorithm targets both imperfect and complete information of state transition probabilities.

3.2.3. Algorithms of model-free

For the past few years, model-free reinforcement learning techniques have also been used to examine task allocation and resource management for MEC systems [49], [50]. The authors developed a method for mobile task-offloading based on a volatile multi-armed bandit (VMAB), which, in accordance with, decreases communication delay, calculation delay, and handover delay while adhering to an energy budget restriction. A wireless MEC network can be set up so that each task is carried out locally or as a fully offloaded network, according to Huang *et al.* [51]. The development of an ideal binary offloading policy that would increase the overall computation rate within the constraints of its settings was then described using a task offloading framework based on deep reinforcement learning (DRL). A partially observable MDP (POMDP) is proposed by [52] and is solved by a decision-guideline-based algorithm that creates individual rationality and incentives and determines the best solution. It successfully optimizes computational offloading and energy consumption while accounting for the potential that a task may not be completed within the maximum allowable delay [53].

3.3. Research methodology of computation offloading

Task offloading in mobile edge computing entails six steps: awareness of the MEC environment [54], [55], task allocation, choice to offload, transmission of offloading tasks, and task return. Offloading activities is crucial during the entire process [56]. In other words, based on the characteristics of the task, the user selects the appropriate decision-making mode, which is a crucial aspect of the user service experience. Figure 3 depicts what takes place during the off-loading procedure. The system framework for MEC is an infrastructure architecture that runs as a virtual computer. When a user offloads a task to the MEC within the range of communication when it is offloaded, the MEC receives the task. Users are subsequently provided with the relevant service by edge servers, which first generate VM environments in accordance with the tasks they have submitted [57]. The MEC server delivers the task's outcome through the backhaul link as soon as the computing task is finished. The infrastructure of the MEC system framework is characterized as a virtual computer [58]. When a task is offloaded, it may be done anywhere within the MEC's communication range.

In response to tasks that users submit, edge servers build virtual machine environments and subsequently offer related services to consumers. When the task has been completed, the MEC server transfers the output of processing the task that the user offloaded across its backhaul link. The MEC server reclaims certain resources after the virtual machine is destroyed. The MEC paradigm divides offloading into two main categories: partial offloading and binary offloading [59]. When binary offloading is employed, the computing jobs at the user cannot be partitioned. They must be carried out at the user or at the edge cloud instead. Offloading procedures can be carried out locally by partially offloading chores at the user's workplace.

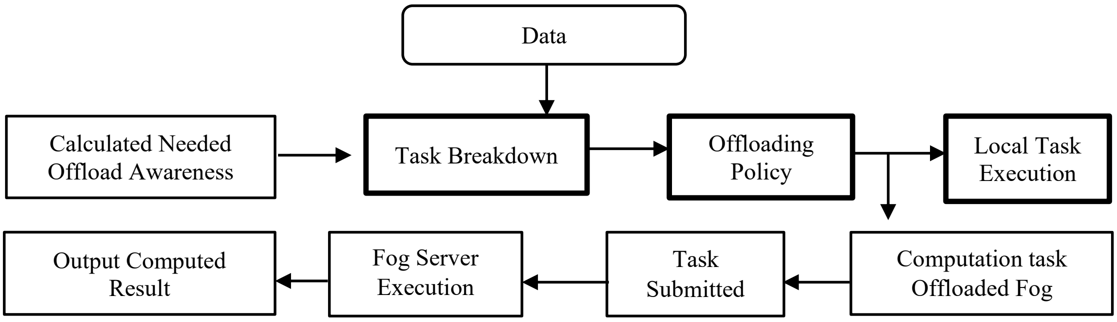

Figure 3. Computation offloading methodology

3.4. Computation offloading using adaptive boosting and K-nearest neighbour (KNN) algorithm

Some trees are taller than others, and each tree has a unique number of leaves and branches. But the AdaBoost platform only permits a stump, which is a single node with two leaves. A dataset was reported on a dataset of 31.060 rows and 20 columns. Periods are represented by rows, while the number of users for each era is displayed in columns. Each record is a wireless channel gain representation. With N being the number of records and W being the sample weight, we discharge all these channels with W=1/N. We create a base learner from the dataset and give each user the same weight. The number of different features in each stump is used to group this data set. The suggested adaptive boosting and training optimization technique entails the stages below, according to the flowchart in Figure 4. The sampled weights are assigned to each record, create a decision tree for each label, classify the record, and then evaluate the results to analyze each tree's relevance. Considering the errors in earlier decision trees, the records are updated with the new weights. Once the number of outcomes matches the number of hyperparameters (estimators), the dataset is revised, the process and then used to create a forest of decision trees to forecast the test data.

3.4.1. KNN algorithm

The data is categorized using the K nearest neighbors based on the K hyperparameters and the distances between the labels. The gap between each record and the newly established training record is then determined. Euclidean and Manhattan [60] methods are used to compute distance metrics. [61] Asserts that several distance measures can be manipulated using the Minkowski distance. Equation 1 can be solved to determine how far a variable X is from a variable Y. Figure 3 depicts the flowchart of KNN implementation.

$$\left(\sum_{i=1}^{n} |\boldsymbol{X_i} - \boldsymbol{Y_i}|^{\boldsymbol{p}}\right)^{\boldsymbol{1/p}} \tag{1}$$

The p-value can be used to determine the separation between two places.
p = 1, Manhattan distance selected
p = 2, Euclidean distance selected
In equation (2), the Manhattan distance between two places is estimated by adding up the variations in their Cartesian coordinates.

$$\boldsymbol{d} = \sum_{i=1}^{n} |\boldsymbol{X_i} - \boldsymbol{Y_i}| \tag{2}$$

A Euclidean distance, as defined by (3), is the separation between two points in Euclidean space.

$$\boldsymbol{d(x, y)} = \sqrt{\sum_{i=1}^{n} (\boldsymbol{x_i} - \boldsymbol{y_i})^2} \tag{3}$$

A dataset of 31,060 rows × 20 columns is used, **and later** 30,000 samples per class (0 and 1).
Sample size: 31,060 instances used for initial training; later balanced to 30,000 per class.
Selection criteria: Rows represent time periods; columns represent user channel gain values.

Task categorization: Binary classification (0 = local execution, 1 = offloaded).
Offloading indicator: O_i represents the offloading decision per time frame T_i.

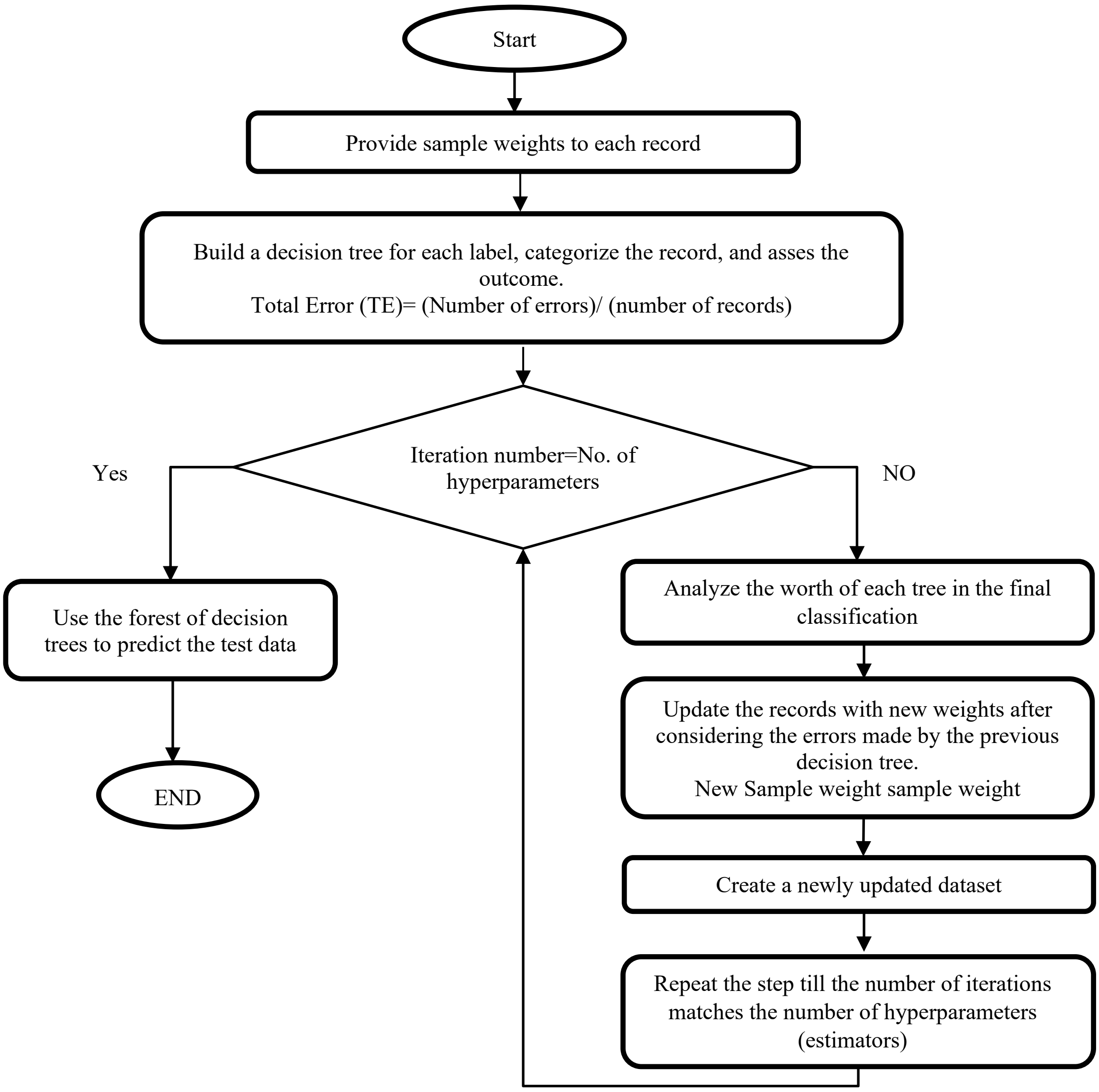
Start
Provide sample weights to each record
Build a decision tree for each label, categorize the record, and asses the outcome.
Total Error (TE)= (Number of errors)/ (number of records)
Iteration number=No. of hyperparameters
Yes
NO
Use the forest of decision trees to predict the test data
END
Analyze the worth of each tree in the final classification
Update the records with new weights after considering the errors made by the previous decision tree.
New Sample weight sample weight
Create a newly updated dataset
Repeat the step till the number of iterations matches the number of hyperparameters (estimators)

Figure 4. Flowchart implementing Adaboost

3.5. Implementation using adaptive boosting and KNN optimization

The following steps are involved in offloading computations using adaptive boosting and KNN optimization. Before determining the wireless signal gain h_r for each time frame Ti and the label classifier K (i.e., based on the error rate or accuracy score, input and output must first be defined. The result is a description of each offloading operation Oi and the optimum resource allocation that goes along with it for each time frame Ti. The model receives data input and generates comma-delimited values (CSV) to show the dataset's form. Figure 6 displays the identification of skewness and outliers, whereas Figure 7 displays a histogram. The flattened array is of the same type as the input array and retains the dataset structure's chosen order. Panda then converts the dataset into a binary data frame and gives each variable a binary value of 0 or 1. The data frame's form is then determined. The slicing operator in Panda can balance unbalanced data frames. Later, data frames are transformed into CSV. When used, the transformation feature minmax scaling condenses the input dataset to a scalable range. The datasets used for testing and training are divided into categories, while, where needed, maintaining randomization values.

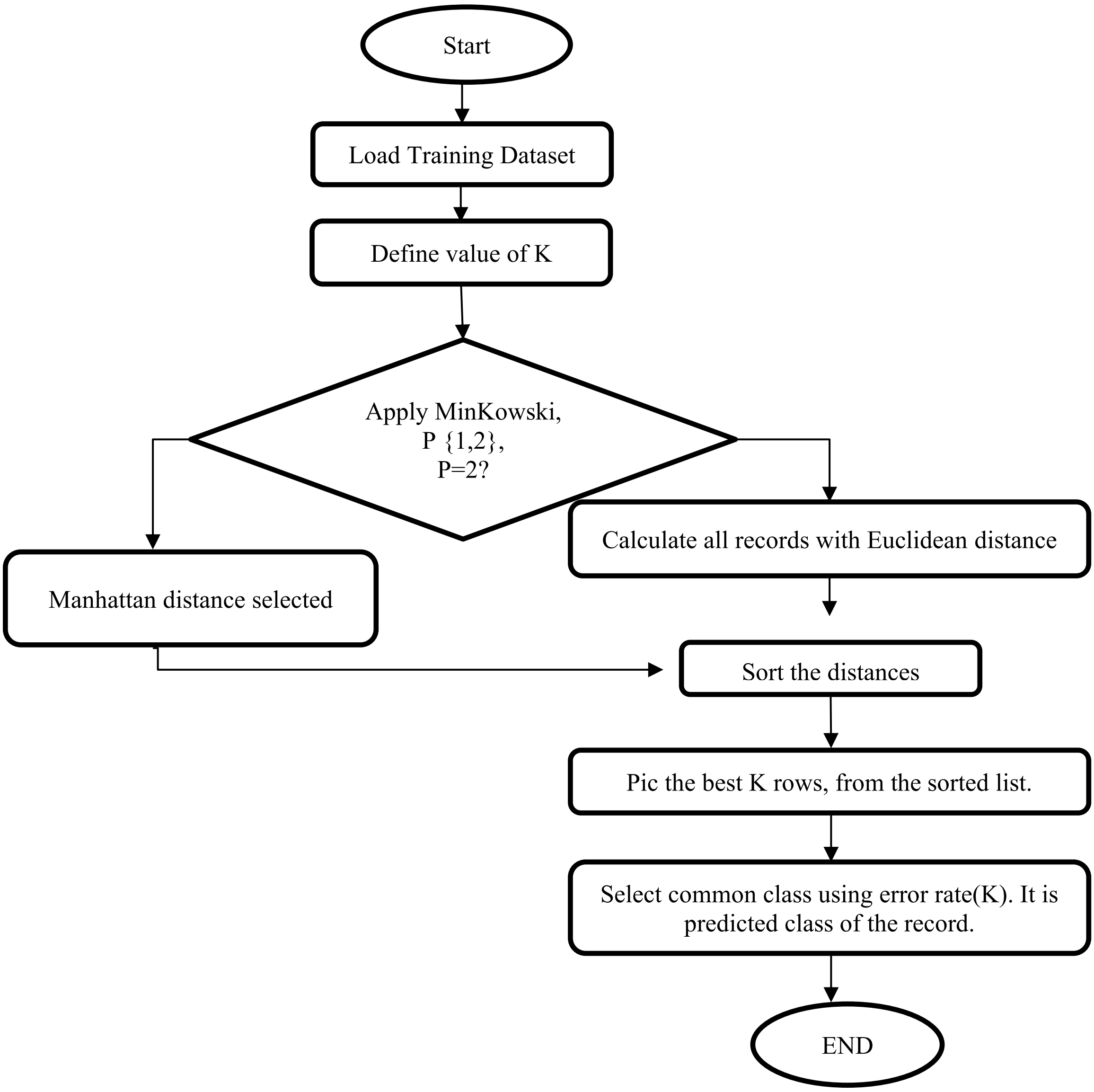
Start
Load Training Dataset
Define value of K
Apply MinKowski,
P {1,2},
P=2?
Manhattan distance selected
Calculate all records with Euclidean distance
Sort the distances
Pic the best K rows, from the sorted list.
Select common class using error rate(K). It is predicted class of the record.
END

Figure 5. Flowchart of KNN implementation

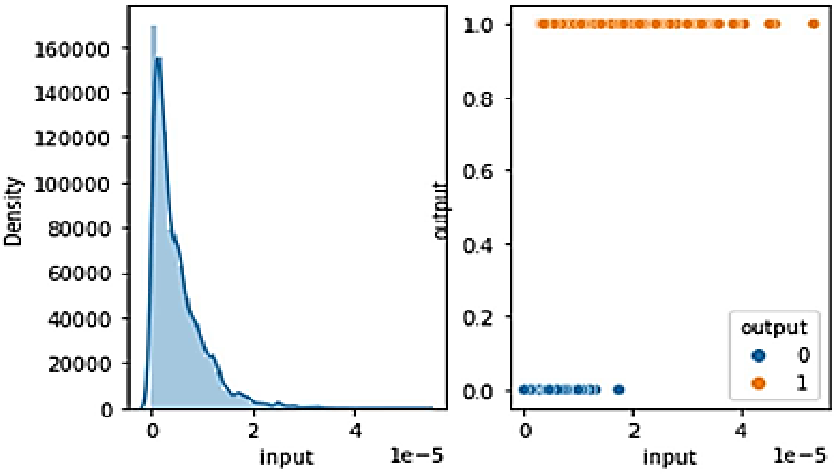

Figure 6. Non-uniform data distribution

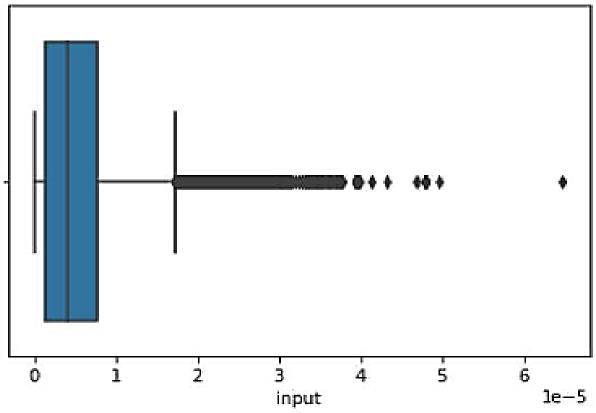

Figure 7. Outliers of input data

4. RESULTS AND DISCUSSION

The coordinate descent method, which is described in [62], is used to create data samples with an input shape and an output shape. Diagrams called histograms are used to display data distribution in two dimensions. Figure 8 illustrates the histogram's skewness, overlapping distributions, and several outliers. This flowchart shows how adaptability boosting and KNNs are utilized for the training model's best computation offloading. Machine learning models are vulnerable to outliers. The wireless channel level affects the computing mode. Data frames are produced by flattening input and output data with the ravel function. The binary-value = class (0,1) is used to further categorize it. We evaluate the labels for class_1 shape (36666, 2) and class_0 shape (118634, 2), which display significant form differences, to distinguish the dataset shape. 30000 samples from each class, with labels 0 and 1, are randomly chosen to construct a sample dataset. The input data are displayed using a distribution plot. Each feature is scaled to a range using the min-max scaler function, and the modified feature is then used, as illustrated in Figure 9. Data offloading options are available in fog situations, where n is the number of users. Several methods, including coordinate Descent and deep reinforcement learning, are used to train the model. We contrast the two most effective algorithms, AdaBoost and KNN, based on accuracy, training time, and network latency. The model is initially trained using KNN optimization and adaptive boosting strategies. The model of random forests enhances accuracy, serves as a base estimator, manages variation, and guards against overfitting.

Furthermore, because there was previously such a high degree of overlap in the data distribution, the number of false positives and false negatives has been significantly decreased. The wireless channel gain, $h_{r,}$ is the only factor dependent on the system configuration, assuming other components remain static when working with a dataset of 20 users. The offloading indicator is O_i. Figure 9 illustrates how to identify the K value after plotting the graph of error rate following model training with KNN optimization.

The Euclidean algorithm is employed to calculate distance. Based on a cumulative decision-making algorithmic study of the various algorithms, as shown in Table 1, the KNN algorithm is thought to be the most effective one. This is how the KNN algorithm is described in terms of metrics [63].

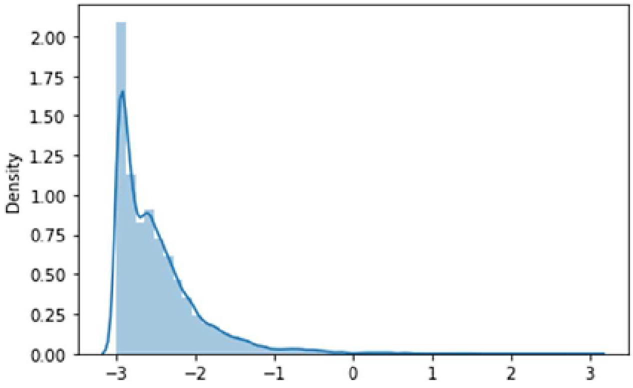

Figure 8. Feature of scaling data

The performance metrics [64], [65] used to evaluate computation offloading strategies include

a. System Accuracy – measures how effectively the model performs tasks after offloading.
b. Training Time – the time required to train the model, indicating computational efficiency.
c. Latency – the delay incurred during task execution or response after offloading.

The accuracy, latency, and training time are the three most essential measures used to compare the performance of AdaBoost, KNN, and the baseline coordinate descent technique. Figure 10. presents the comparative performance graph to support the analysis. The findings reveal that KNN had the highest accuracy (0.99) and the lowest latency (4.7×10^{-7} seconds). It also needed less training time (164.11 seconds) than AdaBoost. AdaBoost had the biggest training overhead (625s), but it did well in terms of accuracy (0.94) and latency (0.017s). The baseline coordinate descent approach, on the other hand, had the lowest accuracy (0.85) and the worst response time (0.150 s). These results show a clear trade-off: KNN is more accurate and responds in real time, while AdaBoost balances accuracy with moderate latency, although it is harder to train. The baseline is less suitable for applications that need great reliability and speed, even though it is easier to compute in some ways.

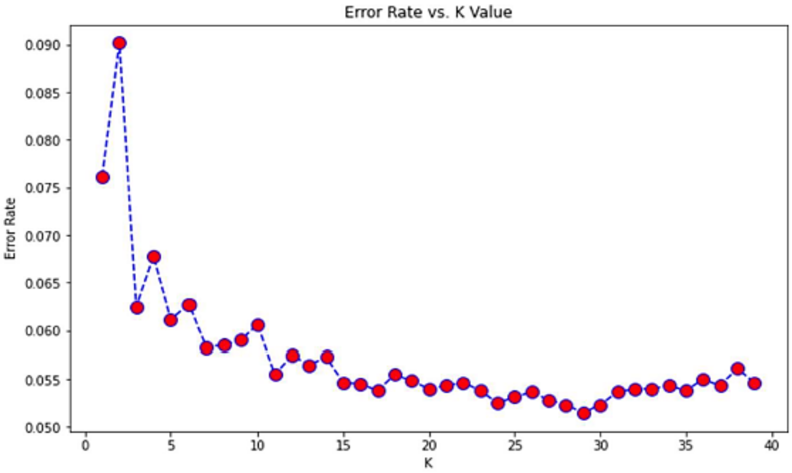

Figure 9. Error rate vs. K. value

Table 1. Cumulative decision-making algorithmic analysis

Metric	Accuracy	Training Time (s)	Latency (s)
DNN-DRL	0.99	279.57	0.032
Baseline (Coordinate Descent)	0.85	328.43	0.150
AdaBoost	0.94	625.00	0.017
KNN	0.99	164.11	4.7E-07

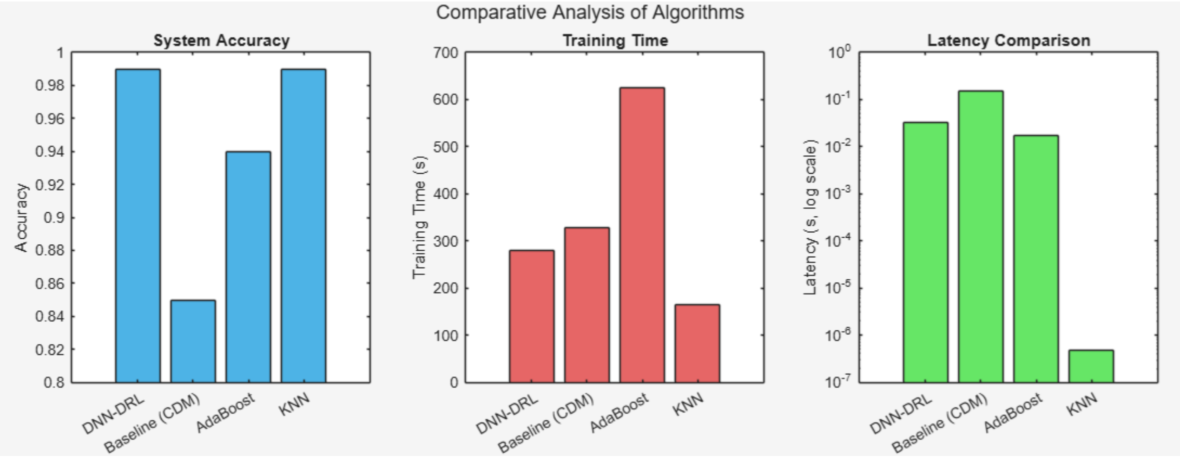

Figure 10. Comparative performance graph

This work is limited by the utilization of simulated environments instead of extensive real-world implementations, potentially impacting generalizability. The assessment concentrated on certain metrics, possibly neglecting additional contributing elements like security vulnerabilities or diverse device habits. The dataset size and model assumptions may limit applicability across various network and application scenarios.

5. CONCLUSION

To increase network computation rates, the research provides an optimized computation offloading approach that makes use of K nearest neighbors. We investigate the requirement for computation offloading in Internet of Things applications as we introduce the optimization for computation offloading. The KNN algorithm learns the offloading value faster than more traditional optimization techniques like coordinate descent, DNN-DRL, and adaptive boosting. This enables the creation of judgements that are close to ideal, resulting in system optimization. Deep learning models are particularly expensive to train because of the complex data patterns involved. Furthermore, fog computing and offloading are not yet standardized, providing researchers with lots of room to experiment and explore without being constrained by established guidelines. In the end, we aim to optimize wireless communication by using numerous criteria for resource allocation and offloading choices, which will increase its effectiveness. As wearable computing starts to realize the promise of a fully personal digital assistant, it offers an interesting avenue to experiment with augmented realities. The analysis shows DNN-DRL and KNN achieving 99% accuracy, with KNN offering the lowest latency at 4.7×10^{-7} s. AdaBoost attains 94% accuracy and 0.017 s latency, while the baseline lags at 85% accuracy and 0.150 s latency. Overall, advanced models outperform the baseline across accuracy, training time, and latency metrics. The field's potential is only now starting to be appreciated, even though numerous applications and issue domains were covered here. In the upcoming years, as technology becomes more widely used and embraced, the apps will upend preconceived notions about computers. Future research may concentrate on adaptive offloading strategies that address network dynamics, device mobility, and variations in workload. Energy efficiency must be prioritized in conjunction with accuracy and latency to enhance sustainable mobile and edge systems. Furthermore, investigating federated learning, edge intelligence, and real-world implementation in sectors such as healthcare and IoT may improve scalability, security, and practical applicability.

ACKNOWLEDGEMENTS

The authors express their sincere gratitude to SRMIST, NCR, for providing the platform and facilities to carry out this work.

FUNDING INFORMATION

No funding was involved.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Prince Gupta	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓			
Rajeev Sharma	✓	✓							✓	✓		✓		
Sachi Gupta	✓								✓	✓		✓		
Adesh Kumar	✓						✓		✓	✓	✓		✓	

C	: **C**onceptualization	I	: **I**nvestigation	Vi	: **Vi**sualization
M	: **M**ethodology	R	: **R**esources	Su	: **Su**pervision
So	: **So**ftware	D	: **D**ata Curation	P	: **P**roject administration
Va	: **Va**lidation	O	: Writing - **O**riginal Draft	Fu	: **Fu**nding acquisition
Fo	: **Fo**rmal analysis	E	: Writing - Review & **E**diting		

CONFLICT OF INTEREST STATEMENT

The authors state no conflict of interest.

DATA AVAILABILITY

The authors confirm that the data supporting the findings of this study are available within the article.

REFERENCES

[1] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, Oct. 2016, doi: 10.1109/JIOT.2016.2579198.

[2] E. Ahmed, A. Gani, M. Sookhak, S. H. A. Hamid, and F. Xia, "Application optimization in mobile cloud computing: Motivation, taxonomies, and open challenges," *Journal of Network and Computer Applications*, vol. 52, pp. 52–68, Jun. 2015, doi: 10.1016/j.jnca.2015.02.003.

[3] B. Varghese, N. Wang, S. Barbhuiya, P. Kilpatrick, and D. S. Nikolopoulos, "Challenges and opportunities in edge computing," in *Proc. 2016 IEEE Int. Conf. Smart Cloud (SmartCloud)*, Nov. 2016, pp. 20–26, doi: 10.1109/SmartCloud.2016.18.

[4] P. Hu, S. Dhelim, H. Ning, and T. Qiu, "Survey on fog computing: Architecture, key technologies, applications and open issues," *Journal of Network and Computer Applications*, vol. 98, pp. 27–42, Nov. 2017, doi: 10.1016/j.jnca.2017.09.002.

[5] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, "A survey of computation offloading for mobile systems," *Mobile Networks and Applications*, vol. 18, no. 1, pp. 129–140, Feb. 2013, doi: 10.1007/s11036-012-0368-0.

[6] R. W. Ahmad, A. Gani, S. H. A. Hamid, M. Shiraz, A. Yousafzai, and F. Xia, "A survey on virtual machine migration and server consolidation frameworks for cloud data centers," *Journal of Network and Computer Applications*, vol. 52, pp. 11–25, Jun. 2015, doi: 10.1016/j.jnca.2015.02.002.

[7] R. Kemp, N. Palmer, T. Kielmann, and H. Bal, "Cuckoo: A computation offloading framework for smartphones," in *Int. Conf. Mobile Computing, Applications, and Services*, 2012, vol. 76, pp. 59–79, doi: 10.1007/978-3-642-29336-8_4.

[8] C. Wang and Z. Li, "Parametric analysis for adaptive computation offloading," *SIGPLAN Notices*, vol. 39, no. 6, pp. 119–130, Jun. 2004, doi: 10.1145/996893.996857.

[9] B. G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "CloneCloud: Elastic execution between mobile device and cloud," in *Proc. 6th Conf. Computer Systems*, Apr. 2011, pp. 301–314, doi: 10.1145/1966445.1966473.

[10] E. Cuervo *et al.*, "MAUI: Making smartphones last longer with code offload," in *Proc. 8th Int. Conf. Mobile Systems, Applications, and Services*, Jun. 2010, pp. 49–62, doi: 10.1145/1814433.1814441.

[11] "Cisco annual internet report (2018–2023) white paper." Accessed: Mar. 27, 2022. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/whitepaper-c11-741490.html.

[12] M. Hadji, H. Sedjelmaci, I. B. Jemaa, and A. Kaiser, "Security framework for vehicular edge computing network based on behavioral game," in *Proc. IEEE GLOBECOM*, 2018, pp. 1–6, doi: 10.1109/GLOCOM.2018.8647348.

[13] L. Li, H. Zhou, S. Xiong, J. Yang, and Y. Mao, "Compound model of task arrivals and load-aware offloading for vehicular mobile edge computing networks," *IEEE Access*, vol. 7, p. 1, Feb. 2019, doi: 10.1109/ACCESS.2019.2901280.

[14] W. Shi and S. Dustdar, "The promise of edge computing," *Computer*, vol. 49, no. 5, pp. 78–81, May 2016, doi: 10.1109/MC.2016.145.

[15] C. Wang, Y. Li, and D. Jin, "Mobility-assisted opportunistic computation offloading," *IEEE Communications Letters*, vol. 18, no. 10, pp. 1779–1782, Oct. 2014, doi: 10.1109/LCOMM.2014.2347272.

[16] S. Yu, B. Dab, Z. Movahedi, R. Langar, and L. Wang, "A socially-aware hybrid computation offloading framework for multi-access edge computing," *IEEE Transactions on Mobile Computing*, vol. 19, no. 6, pp. 1247–1259, Jun. 2020, doi: 10.1109/TMC.2019.2908154.

[17] Q. Liu, Z. Su, and Y. Hui, "Computation offloading scheme to improve QoE in vehicular networks with mobile edge computing," in *Proc. WCSP*, 2018, pp. 1–5, doi: 10.1109/WCSP.2018.8555879.

[18] M. V Barbera, S. Kosta, A. Mei, and J. Stefa, "To offload or not to offload? The bandwidth and energy costs of mobile cloud computing," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 1285–1293, doi: 10.1109/INFCOM.2013.6566921.

[19] R. Hasan, M. Hossain, and R. Khan, "Aura: An incentive-driven ad-hoc IoT cloud framework for proximal mobile computation offloading," *Future Generation Computer Systems*, vol. 86, pp. 821–835, Sep. 2018, doi: 10.1016/j.future.2017.11.024.

[20] D. Huang, P. Wang, and D. Niyato, "A dynamic offloading algorithm for mobile computing," *IEEE Transactions on Wireless Communications*, vol. 11, no. 6, pp. 1991–1995, Jun. 2012, doi: 10.1109/TWC.2012.041912.110912.

[21] G. Zhang, W. Zhang, Y. Cao, D. Li, and L. Wang, "Energy-delay tradeoff for dynamic offloading in mobile-edge computing system with energy harvesting devices," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4642–4655, Oct. 2018, doi: 10.1109/TII.2018.2843365.

[22] P. Zhao, H. Tian, C. Qin, and G. Nie, "Energy-saving offloading by jointly allocating radio and computational resources for mobile edge computing," *IEEE Access*, vol. 5, pp. 11255–11268, 2017, doi: 10.1109/ACCESS.2017.2710056.

[23] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Transactions on Communications*, vol. 64, no. 10, pp. 4268–4282, Oct. 2016, doi: 10.1109/TCOMM.2016.2599530.

[24] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," in *Proc. IEEE Int. Symp. Information Theory (ISIT)*, Jul. 2016, pp. 1451–1455, doi: 10.1109/ISIT.2016.7541539.

[25] W. Chen, D. Wang, and K. Li, "Multi-user multi-task computation offloading in green mobile edge cloud computing," *IEEE Transactions on Services Computing*, vol. 12, no. 5, pp. 726–738, Sep. 2019, doi: 10.1109/TSC.2018.2826544.

[26] H. Guo and J. Liu, "Collaborative computation offloading for multiaccess edge computing over fiber–wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 5, pp. 4514–4526, May 2018, doi: 10.1109/TVT.2018.2790421.

[27] X. Sun and N. Ansari, "Latency aware workload offloading in the cloudlet network," *IEEE Communications Letters*, vol. 21, no. 7, pp. 1481–1484, Jul. 2017, doi: 10.1109/LCOMM.2017.2690678.

[28] L. Tong, Y. Li, and W. Gao, "A hierarchical edge cloud architecture for mobile computing," in *Proc. IEEE INFOCOM - The 35th Annual IEEE Int. Conf. Computer Communications*, Apr. 2016, pp. 1–9, doi: 10.1109/INFOCOM.2016.7524340.

[29] P. J. M. van Laarhoven and E. H. L. Aarts, *Simulated Annealing: Theory and Applications*. Dordrecht, The Netherlands: Springer, 1987.

[30] F. Xia, F. Ding, J. Li, X. Kong, L. T. Yang, and J. Ma, "Phone2Cloud: Exploiting computation offloading for energy saving on smartphones in mobile cloud computing," *Information Systems Frontiers*, vol. 16, no. 1, pp. 95–111, Mar. 2014, doi: 10.1007/s10796-013-9458-1.

[31] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 1171–1181, Dec. 2016, doi: 10.1109/JIOT.2016.2565516.

[32] Y. Mao, J. Zhang, and K. B. Letaief, "Joint task offloading scheduling and transmit power allocation for mobile-edge computing systems," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Mar. 2017, pp. 1–6, doi: 10.1109/WCNC.2017.7925615.

[33] Y. Mao, J. Zhang, S. H. Song, and K. B. Letaief, "Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems," *IEEE Transactions on Wireless Communications*, vol. 16, no. 9, pp. 5994–6009, Sep. 2017, doi: 10.1109/TWC.2017.2717986.

[34] Y. Wang, J. Li, M. Sheng, and others, "A game-based computation offloading method in vehicular multiaccess edge computing networks," *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 4987–4996, Jun. 2020, doi: 10.1109/JIOT.2020.2972061.

[35] Q. Xu, Z. Su, Q. Zheng, M. Luo, and B. Dong, "Secure content delivery with edge nodes to save caching resources for mobile users in green cities," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 6, pp. 2550–2559, Jun. 2018, doi: 10.1109/TII.2017.2787201.

[36] C. Yi, J. Cai, and Z. Su, "A multi-user mobile computation offloading and transmission scheduling mechanism for delay-sensitive applications," *IEEE Transactions on Mobile Computing*, vol. 19, no. 1, pp. 29–43, Jan. 2020, doi: 10.1109/TMC.2019.2891736.

[37] W. Zhan, C. Luo, G. Min, C. Wang, Q. Zhu, and H. Duan, "Mobility-aware multi-user offloading optimization for mobile edge computing," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 3, pp. 3341–3356, Mar. 2020, doi: 10.1109/TVT.2020.2966500.

[38] S.-W. Ko, K. Han, and K. Huang, "Wireless networks for mobile edge computing: Spatial modeling and latency analysis," *IEEE Transactions on Wireless Communications*, vol. 17, no. 8, pp. 5225–5240, Aug. 2018, doi: 10.1109/TWC.2018.2840120.

[39] J. Zheng, Y. Cai, Y. Wu, and X. Shen, "Dynamic computation offloading for mobile cloud computing: A stochastic game-theoretic approach," *IEEE Transactions on Mobile Computing*, vol. 18, no. 4, pp. 771–786, Apr. 2019, doi: 10.1109/TMC.2018.2847337.

[40] C.-F. Liu, M. Bennis, M. Debbah, and H. V Poor, "Dynamic task offloading and resource allocation for ultra-reliable low-latency edge computing," *IEEE Transactions on Communications*, vol. 67, no. 6, pp. 4132–4150, Jun. 2019, doi: 10.1109/TCOMM.2019.2898573.

[41] B. Zhang, L. Wang, and Z. Han, "Contracts for joint downlink and uplink traffic offloading with asymmetric information," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 4, pp. 723–735, Apr. 2020, doi: 10.1109/JSAC.2020.2971807.

[42] H. Zhang, Y. Xiao, S. Bu, D. Niyato, F. R. Yu, and Z. Han, "Computing resource allocation in three-tier IoT fog networks: A joint optimization approach combining Stackelberg game and matching," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1204–1215, Oct. 2017, doi: 10.1109/JIOT.2017.2688925.

[43] Q. Zhang, L. Gui, F. Hou, J. Chen, S. Zhu, and F. Tian, "Dynamic task offloading and resource allocation for mobile-edge computing in dense cloud RAN," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 3282–3299, Apr. 2020, doi: 10.1109/JIOT.2020.2967502.

[44] G. Li and J. Cai, "An online incentive mechanism for collaborative task offloading in mobile edge computing," *IEEE Transactions on Wireless Communications*, vol. 19, no. 1, pp. 624–636, Jan. 2020, doi: 10.1109/TWC.2019.2947046.

[45] P. Agarwal, T. K. Garg, and A. Kumar, "Analysis of 3D NoC router chip on different FPGA for minimum hardware and fast switching," *National Academy Science Letters*, pp. 1–5, 2023, doi: 10.1007/s40009-023-01295-y.

[46] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 587–597, Mar. 2018, doi: 10.1109/JSAC.2018.2815360.

[47] M. Yadav, K. Singh, A. S. Pandey, A. Kumar, and R. Kumar, "Smart communication and security by key distribution in multicast environment," *Wireless Communications and Mobile Computing*, 2022, doi: 10.1155/2022/1011407.

[48] X. Zhang, J. Zhang, Z. Liu, Q. Cui, X. Tao, and S. Wang, "MDP-based task offloading for vehicular edge computing under certain and uncertain transition probabilities," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 3, pp. 3296–3309, Mar. 2020, doi: 10.1109/TVT.2020.2965159.

[49] X. Chen, Z. Zhao, C. Wu, M. Bennis, H. Liu, and Y. Ji, "Multi-tenant cross-slice resource orchestration: A deep reinforcement learning approach," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 10, pp. 2377–2392, 2019, doi: 10.1109/JSAC.2019.2933893.

[50] N. C. Luong *et al.*, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Communications Surveys \& Tutorials*, vol. 21, no. 4, pp. 3133–3174, 2019, doi: 10.1109/COMST.2019.2916583.

[51] L. Huang, S. Bi, and Y.-J. A. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Transactions on Mobile Computing*, vol. 19, no. 11, pp. 2581–2593, Nov. 2020, doi: 10.1109/TMC.2019.2928811.

[52] Y. Zhan, S. Guo, P. Li, and J. Zhang, "A deep reinforcement learning based offloading game in edge computing," *IEEE Transactions on Computers*, vol. 69, no. 6, pp. 883–893, Jun. 2020, doi: 10.1109/TC.2020.2969148.

[53] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4005–4018, Jun. 2019, doi: 10.1109/JIOT.2018.2876279.

[54] M. Mukherjee, L. Shu, N. Kumar, A. Zomaya, and others, "Computation offloading strategy in heterogeneous fog computing with energy and delay constraints," in *Proc. IEEE International Conference on Communications (ICC)*, Jun. 2020, pp. 1–5, doi: 10.1109/ICC40277.2020.9148852.

[55] M. S. Sofla, M. H. Kashani, E. Mahdipour, and R. F. Mirzaee, "Towards effective offloading mechanisms in fog computing," *Multimedia Tools and Applications*, vol. 81, no. 2, pp. 1997–2042, Jan. 2022, doi: 10.1007/s11042-021-11423-9.

[56] J. Sheng, J. Hu, X. Teng, B. Wang, and X. Pan, "Computation offloading strategy in mobile edge computing," *Information*, vol. 10, p. 191, Jun. 2019, doi: 10.3390/info10060191.

[57] A. Takeda, T. Kimura, and K. Hirata, "Joint optimization of edge server and virtual machine placement in edge computing environments," in *Proc. APSIPA Annual Summit and Conference (APSIPA ASC)*, Dec. 2020, pp. 1545–1548.

[58] Z. Tao, Q. Li, W. Shi, and others, "A survey of virtual machine management in edge computing," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1482–1499, Aug. 2019, doi: 10.1109/JPROC.2019.2927919.

[59] A. Mahmood, A. Ahmed, M. Naeem, and Y. Hong, "Partial offloading in energy harvested mobile edge computing: A direct search approach," *IEEE Access*, vol. 8, pp. 36757–36763, 2020, doi: 10.1109/ACCESS.2020.2974809.

[60] R. Suwanda, Z. Syahputra, and E. Zamzami, "Analysis of Euclidean distance and Manhattan distance in the K-means algorithm for variations number of centroid K," in *Journal of Physics: Conference Series*, Jun. 2020, vol. 1566, p. 12058, doi: 10.1088/1742-6596/1566/1/012058.

[61] H. A. A. Alfeilat *et al.*, "Effects of distance measure choice on K-nearest neighbor classifier performance: A review," *Big Data*,

vol. 7, no. 4, pp. 221–248, Dec. 2019, doi: 10.1089/big.2018.0175.

[62] S. Bi and Y.-J. A. Zhang, "Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading," *IEEE Transactions on Wireless Communications*, vol. 17, no. 6, pp. 4177–4190, Jun. 2018, doi: 10.1109/TWC.2018.2821664.

[63] J. J. Ferreira, C. Fernandes, and others, "Wearable technology and consumer interaction: A systematic review and research agenda," *Computers in Human Behavior*, vol. 118, p. 106710, 2021, doi: 10.1016/j.chb.2020.106710.

[64] A. Goel, A. Katiyar, A. K. Goel, and A. Kumar, "LSTM neural networks for brain signals and neuromorphic chip," in *Proc. 2024 2nd International Conference on Advances in Computation, Communication and Information Technology (ICAICCIT)*, 2024, pp. 1033–1039, doi: 10.1109/ICAICCIT64383.2024.10912358.

[65] A. Goel, A. Katiyar, A. K. Goel, and A. Kumar, "Comparative study of ANN, CNN, and RNN hardware chips," *National Academy Science Letters*, pp. 1–7, 2025.

BIOGRAPHIES OF AUTHORS

Prince Gupta has earned his Ph.D from the Department of Computer Science and Engineering at SRM Institute of Science and Technology, Delhi NCR Campus, Delhi-Meerut Road, Modinagar, Ghaziabad, U.P., India. He is an M.Tech. in computer science and engineering and a B.Tech. in computer science and engineering in 2015 and 2006, respectively. He has published three research papers. He is working as an Assistant Professor at KIET Group of Institutions, Delhi-NCR, Ghaziabad-Meerut Road, Ghaziabad-201206, U.P. He has more than 15 years' experience. He can be contacted at prince.rkg@gmail.com or pg8179@srmist.edu.in.

Rajeev Sharma is currently working as an associate professor in the Department of Computer Science and Engineering at SRM Institute of Science and Technology (Deemed to be University u/s 3 of the UGC Act,1956), Delhi, NCR, Campus Ghaziabad, India. He received a doctorate in computer science and engineering from Singhania University, Rajasthan. He is the author of more than 30 research publications (patents, Journals (SCI/SCIE/SCOPUS), and National/International conferences). He is also supervising many students to pursue their research work. He has published a book on the Wireless Application Protocol. He has more than 18 years' experience. He is also a member of IEEE and, Computer Society of India. He can be contacted at rajeevks@srmist.edu.in.

Sachi Gupta is currently working as a professor in the Department of Artificial Intelligence and Machine Learning at Galgotias College of Engineering and Technology, Greater Noida, India. She has more than 20 years of teaching and research experience. She completed her Ph.D. and M.Tech. (gold medalist) degrees from Banasthali Vidyapith, Rajasthan, in Computer Science. She completed her B.Tech. (CS and IT) from UPTU, Lucknow. She has filed six patents, out of which three have been granted, and published more than thirty papers in national/international level conferences/ journals of repute. She is an active member of CSI, Vibha, IACSIT, and IAENG. Her areas of interest include task scheduling, genetic algorithms, machine learning, and fuzzy logic. She can be contacted at sachi.gupta@galgotiacollege.edu.

Adesh Kumar is working as Professor in the Department of Electrical & Electronics Engineering, School of Advanced Engineering, The University of Petroleum and Energy Studies (UPES), Dehradun, India since 2010. He has B. Tech in Electronics & Communication Engineering from UPTU, Lucknow, India, in 2006. M.Tech (Hons) in Embedded Systems Technology from SRM University, Chennai in 2008. Ph.D (Electronics Engineering) from UPES, Dehradun, India, in 2014. He has also worked as a Senior Engineer in TATA ELXSI LIMITED, Bangalore, and as a faculty member in ICFAI University, Dehradun. His areas of interest are VLSI Design, Embedded Systems Design, Telecommunications, and Signal Processing. He has supervised 10 Ph.D. scholars, and 5 candidates are doing research under his supervision. He has worked on more than 10 editorial assignments for edited books, Conference proceedings, and journals, having citations of more than 3000, H-index >30. He has 17 years of experience in teaching, research, and industry, and published more than **150**+ research papers in international peer-reviewed journals (SCI/Scopus) and conferences. He can be contacted at adeshmanav@gmail.com; adeshkumar@ddn.upes.ac.in.