JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Machine Learning based Dynamic Offloading System for Mobile Cloud Computing

Raj Kumari
University Institute of Engineering and Technology

Abstract : In the Mobile Cloud Computing (MCC) environment, to improve the application performance on smart mobile devices computation offloading is preferred. Computation offloading is beneficial only if the contextual information about the wireless environment and application is available. As MCC environment is dynamic in nature, gathering of the contextual information is itself a challenging task. The performance of offloading is influenced by the number of parameters *like* network status, application characteristics, mobile device configuration, cloud configuration, the deadline for task execution etc. The offloading decision is a very time-consuming process if all these parameters are considered manually. This problem can be conquered by performing offloading decision through intelligent computational techniques. Thus, in this paper, an efficient dynamic offloading system is represented which helps to take the offloading decision with high accuracy using machine learning techniques *namely* Logistic Regression (LR) and Support Vector Machine (SVM). The proposed work is divided into two parts. In the first part, the offloading decision is made based on available contextual information. In the second part if offloading has to be executed, then the offloaded task is scheduled on either of the two resources, *namely*, cloudlet or workstation. To evaluate the performance of the proposed algorithm, we developed a dataset based on face recognition application. In the dataset, three different sizes of images are considered i.e., 2MP, 6MP and 8MP. The experimental study shows that LR technique has efficiently taken the offloading decision and has generated the highest accuracy rate with 98.26% in comparison to existing experimental techniques.

***IndexTerms* - machine learning, offloading, mobile cloud computing**

I. Introduction

In recent years, the use of smartphones has been increased rapidly. Mobile users directly or indirectly are extensively using the services provided by cloud service providers. Cloud service providers enable the users to run applications on payment as well as demand basis L.Wang et al. (2010) S.Singh et al. (2016). Users execute any mobile application on their mobile phones irrespective of their computational complexity. Mobile Cloud Computing (MCC) concept makes it possible to execute any mobile application on mobile devices without concerning the internal requirements of the application. MCC uses cloud computing capabilities to execute the application on the mobile device H.T.Dinh et al. (2016). This helps to enhance the battery life of mobile devices and reduces the execution time of the application N.Femando et al. (2013). Connecting mobile devices to the cloud is not a tough job. Availability of 3G, 4G/4G+, Wi-Fi and upcoming 5G networks made it possible to execute any mobile applications through a mobile device from anywhere. As the use of mobile devices has been increased, the battery consumption of the mobile device is also a concerned issue.
Studies are focusing on how to make mobile devices more resourceful and energy efficient S.Guo et al. (2016). The consumption of battery life varies from application to application. If an application is less computation complex then less battery power is consumed, and the usage period of the mobile device is increased. On the other side, if any application is computationally complex then the chances of the battery consumption is very high, and the usage period decreases. The concept of *offloading* is used to enhance the battery life of mobile devices K.Akherfi et al. (2018)I.Technologies et al. (2014).
Many frameworks have been proposed for data offloading to reduce the execution time of the application and to increase the energy level of the mobile device S.Kosta et alE.Cuervo et al (2010). As wireless communication is used for application offloading, the dynamic change in network traffic and bandwidth are the main parameters that directly affect the performance of the offloading process. As a result, offloading with poor network conditions causes degradation of performance and energy consumption. Therefore, to take the offloading decision in a mobile cloud computing environment is a very crucial issue. Many researchers are using the concept of machine learning to enhance the capabilities of mobile devices. Depending on the characteristics of an application, the decision about the task offloading to cloud for execution can be taken using intelligent techniques. After understanding the dynamic network conditions and characteristics of an application, a model is trained to take the offloading decision without human intervention or assistance. The system learns to access the data and the process of learning begins with observations or data in the dataset to make the decisions. This helps to take the offloading decision in MCC under dynamic conditions. The objective is to execute the application with minimum delay and energy consumption on mobile device.

In state-of-the-art, many authors have proposed periodic monitoring based new techniques, algorithms and framework for offloading. In this paper, we accentuate on task offloading on to the cloud based on the mobile device resources (CPU, storage), available network status, deadline of the task execution, available energy level on the mobile device and user inputs. Three resources for task execution, i.e., mobile device, cloudlet and workstation are considered for study. The mobile device is the user's own device for local processing. Cloudlet is the local cloud which is closer to the end users. The data is stored and processed on the edge of the network. Workstation is the remote cloud providing the user with access to computational resources beyond that provided by mobile device and cloudlet. Workstation is more powerful than the cloudlet in terms of processing speed and resources. To offload the task from mobile device to the cloud, two different networks, i.e., Wi-Fi and 4G networks are considered. To show the efficiency of intelligent computational offloading decision system main key points are summarized as follows:

- We have developed an efficient dynamic offloading system which integrates information gathering module, processing module and inference module. Information gathering module collects all contextual information regarding MCC environment and application configuration. Processing module evaluates the information gathered and passes it to the inference module.
- Based on the processing module information, inference module takes the decision for task offloading. This module helps to take the offloading decision based on the proposed intelligent offloading decision algorithm. As the size of dataset increases, there will be too much variation in the execution time of 2MP, 6MP and 8MP images. Therefore, the efficient offloading can be taken with the help of intelligent offloading system.
- Once the offloaded decision is taken, the scheduling of offloaded tasks either on cloudlet or on workstation is done based on the deadline of the task.

The rest of the paper is structured as follows: Section 2 contains a description of cognate work. Section 3 is presented with a dynamic offloading system and its description. Section 4 deals with an intelligent computational offloading decision using machine learning techniques. The performance evaluation, experimental setup and results are discussed in Section 5. Finally, the conclusion and future work are presented in Section 6.

2 Related Work

Offloading in MCC has been widely studied in the literature. Most of the previous works had focused on the time, cost or energy efficient offloading. The basic architecture of MCC, challenges and issues are studied by many researchers B.Zhou et al (2015)M.Shiraz et al (2013) H.Allam et al (2017). They all discussed the influence of computational capacity, connectivity, security, latency, and heterogeneity in MCC. The computational offloading framework shows that the partitioning of the application can be done statically or dynamically on a single site or multi-sites. In static partitioning, the affordable components are decided before the execution of the application. In dynamic partitioning approach, the offloadable components are decided during the execution of the application. These frameworks have also discussed the different granularity levels of the application partitioning like class level, method level, component level, module level, etc. M.Shiraz et al (2014). In general, mobile applications are broadly divided into two categories, i.e., computation-intensive and communication intensive. Offloading of tasks requires the estimation of the computational complexity of the mobile application. Depending on the computational complexity D.Meil et al (2014), offloading of the tasks is performed in MCC. Offloading efficiency depends on many parameters like dynamic network conditions, characteristics of the mobile application, processing efficiency of mobile devices and cloud server R.Kumari et al (2017). A computation offloading-based system is devised in F.Xia et al (2014) for energy saving on smartphones. It offloads the computation of an application running on smartphones to the cloud. The objective was to improve the energy efficiency of smartphones and at the same time, enhance the application's performance through reducing its execution time. An online code offloading and scheduling algorithm is proposed in B.Zhou et al (2018) to minimize the completion time of the task. There has been a lot of research on how to reduce mobile energy usage. The process of offloading also consists of the partitioning of tasks, profiling, decision-making, and offloading of components. The characteristics of local devices, cloudlets, and the remote cloud are required to be explored well before to take the offloading decision. The connectivity between the mobile device and cloud is the determinant factor for offloading performance evaluation. Therefore, all the perspective of the mobile device and the Cloud are required to be examined thoroughly for performance evaluation in terms of the energy-awareA.Boukerche et al (2019). Others have suggested that energy consumption can be optimized by offloading computation to the cloud S.Kosta et alA.Beloglazov et al (2012). The feasibility of applying machine learning techniques for the mobile offloading framework is discussed in H.Eom et al (2012). Authors showed the comparison of different machine learning techniques for the offline offloading scheduler. In article P.A.L Rego et al (2019), used a machine learning technique to take the decision of when and where to offload based on system metrics and user's mobility in the hybrid environment. A context-sensitive offloading system is proposed in the paper W.Junior et al (2019). It used JRIP and J48 classifiers on the dataset to take the offloading decision. An agent-based architecture with supervised and reinforcement learning strategies to manage the execution of tasks within the framework of MCC P.Nawrocki et al (2019). In paper O.B Yetim et al (2015), the authors have focused on the delay-tolerant for data offloading. They discussed the adaptive estimators that learn delay tolerance during execution. A method-based offloading framework for Android and Windows Phone platforms that dynamically decides when a method must be offloaded has been discussed in P.B Costa et al (2015) P.A.L Rego et al (2017).
The above-mentioned approaches considered different parameters for offloading and scheduling the application for execution. In comparison, this paper improves the decision of offloading by considering real-time environment parameters such as available energy on the mobile device, network status and deadline of task execution that directly enhance the offloading performance. The focus of the paper is to take the offloading decision efficiently and with less effort. The proposed intelligent computational offloading algorithm in this work helps to take the efficient offloading decision and after making the decision, it generates the result for where to schedule the offloaded task, i.e., on cloudlet or on the workstation for execution. The simulation results show the performance of the proposed algorithm with various machine-learning techniques. In the comparison of LR, SVM, JRIP and J48 machine learning techniques, LR has the highest accuracy rate with 98.26% on the proposed dataset. It shows that the execution of the task is performed under the take care of deadline, the available energy of the mobile device and network status.

3 Dynamic Offloading System

In this section, the proposed dynamic offloading system, architecture, its components and their associations are discussed.

3.1 Architecture

Several challenges and issues are investigated in literature related to offloading in mobile cloud computing. This section, the proposed dynamic offloading system is presented that helps in efficient execution of the user's application. The architectural overview of the proposed system is shown in Fig.1. This system will help the user to take the expeditious decision of data offloading. The description of the system, components and their associations is as follows:

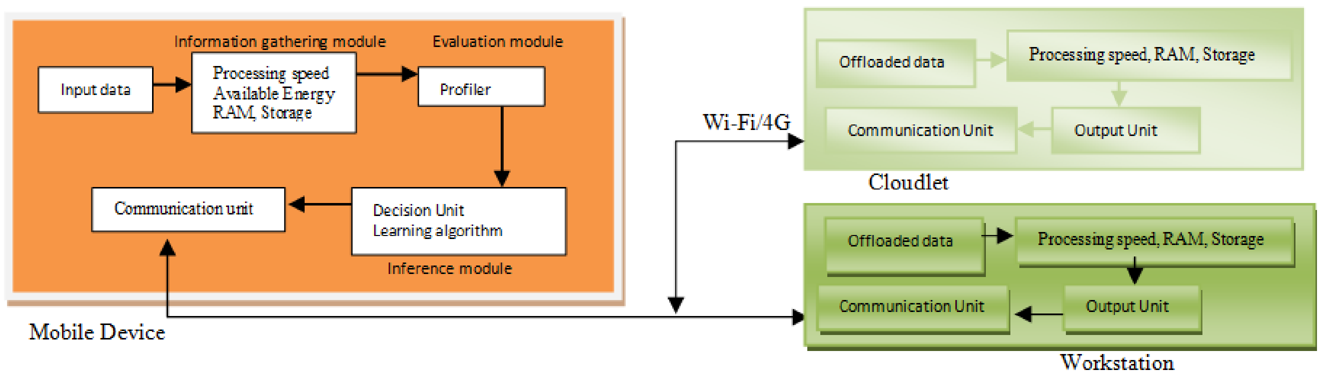


Fig. 1 Proposed DOS system

In this study, the user's task can be executed on a mobile device or on the cloud. The functioning of various sub-components of mobile device are presented as follows:

A. Input data: User task for execution can be in different formats like ASCII characters, images, videos, etc. To test proposed work, a sample application for face recognition that takes images as input has been considered. The input images can have any of the supported formats (.jpg, .jpeg, .png etc.), and they can have any size or dimension.

B. *Information gathering module*

- Processing time: In this paper, mainly three resources are considered for task execution, i.e., mobile device, cloudlet and workstation. Among these, the workstation is the fastest than cloudlet and mobile device is the slowest device. When the task is executed on the mobile device, its execution time depends on the CPU capacity of the mobile device. If the mobile device has high CPU processing speed, then there is no need for offloading otherwise offloading of data is an option.
- Available energy on a mobile device: The available energy on the mobile device is related to the battery life of the mobile device. To execute or offload the application, minimum amount of battery life of the mobile device is required. If the sufficient battery life is available to support the application execution, then there is no need for offloading.

C. *Evaluation module*

- Profiler: The responsibility of profiler is to pass the complete status of the mobile device to the decision unit. It passes the information regarding the size of the application, available bandwidth rate in the network, CPU processing power, available battery life on the mobile device, and other information about the status of the mobile device like available RAM, storage etc.

D. *Inference module*

- Decision unit: It receives input from the profiler and communication unit. Using this information, it takes the decision about the data offloading. It checks all parameters and according to the delay sensitivity and computation intensity of the application, learning algorithms take the decision for offloading. The computational offloading is not always favorable. The offloading decision depends on multiple parameters like the size of data to be offloaded, the available bandwidth for data transmission, processing cycles required, and processing time on mobile device and the cloud. If the computational requirements can be fulfilled on the cloudlet, this will reduce the transmission time and shorten the execution time of the application.
- Communication Unit: This specifies the time taken to offload the task from mobile device to cloud. Depending on the bandwidth rate (good, average or poor), offloading of the data from the mobile device to the cloud takes place.

The functionality of components of cloudlet and workstation are presents as follows:

A. Offloaded data: If the mobile device is not able to execute the data, then it can be offloaded on the cloud. The cloud receives the data offloaded by the mobile device.

B. Execution unit: The processing of offloaded data is done by this unit. On the cloud side, the CPU processing time is very less as compared to the mobile device, because the processor of the cloud is more powerful than the mobile device.

C. Output unit: After processing the offloaded data, to send the results back to the user is the responsibility of the output unit.

D. Communication unit: It provides the communication between the mobile device and the cloud to pass the output of offloaded data.

3.2 Methodology

This section gives an overview about the step by step processing of information in proposed dynamic offloading system.
The flow of data processing is shown in Fig. 2. From the database, the execution time of the task on mobile device, user defined task's deadline and mobile device configuration are evaluated. Firstly, on the mobile device, if the task execution time is within the deadline and it has sufficient energy then there is no need of offloading. On the other side, if the previous condition is false then network condition is checked. If network conditions are favourable and task can be executed within the deadline then task is offloaded to cloud for execution. If none of the condition is satisfied, then task execution is deferred until the favourable conditions are met.
The proposed work is divided into two main parts. The first one focuses on the computation offloading decision making. The second part highlights about where to execute offloaded data, i.e., on the workstation (remote cloud) or on cloudlet (local cloud). The details are presented below:

(i) Data Offloading

The decision of data offloading is based on various parameters like characteristics of an application, wireless channel, computation capacity of cloudlet and workstation. Out of these parameters, to make simplified offloading decision process, the following parameters are considered.

- available energy on mobile device
- computation capability of mobile device
- deadline for task execution
- execution time on mobile device

The calculations of execution time on mobile device, execution time on cloud and transmission time are presented in Section 3.3.

(ii) Where to execute the offloaded data

The decision for the scheduling of offloaded data on cloudlet or workstation is covered in this part. The mobile device is considered for local processing, cloudlet is the nearest cloud to the mobile device and workstation is the remote cloud. Cloudlet has slow processing speed as compared to the workstation system. Therefore, the execution time of the task on cloudlet is more as compared to the workstation. But, the data transmission time on cloudlet is less compared to the workstation. Hence, to make an efficient decision, both the parameters, i.e., executing time and transmission time are significant.

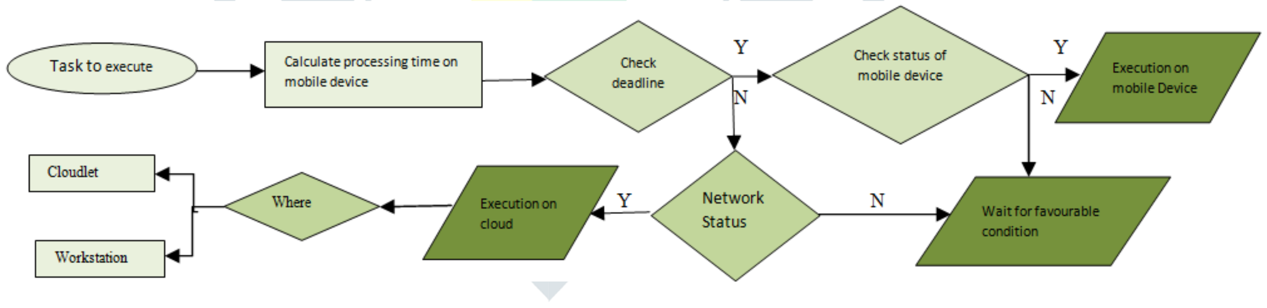

Task to execute
Calculate processing time on mobile device
Check deadline
Y
N
Check status of mobile device
Y
N
Execution on mobile Device
Network Status
Y
N
Wait for favourable condition
Execution on cloud
Where
Cloudlet
Workstation

Fig. 2 Flowchart for data offloading

3.3 Decision Metrics for offloading

The metrics used for taking the dynamic decision for offloading are execution time on the mobile device, execution time on the workstation, execution time on cloudlet and transmission time. The explanation of metrics B.Zhou et al (2018) is as follows:

i. Execution time on a mobile device (ET_m): It is defined as the time taken by the mobile device to execute the instructions of the methods (millions of instruction per second, MIPS) selected for local execution.

$$ET_m = \frac{\mu_m}{\alpha} \qquad (1)$$

where μ_m denotes the number of instructions being executed on mobile device and α denotes the mobile device processor's CPU cycles needed to complete the μ_m instructions.

ii. Execution time on the cloud (ET_c) : It is defined as the time taken to execute the instructions of the task (millions of instruction per second, MIPS) selected for remote execution by the cloudlet or workstation.

$$ET_c = \frac{\mu_c}{\beta} \qquad (2)$$

where μ_c denotes the number of instructions being executed on cloud (cloudlet/workstation) and β denotes the cloud processor's CPU cycles needed to complete the μ_c instructions.

iii. Transmission time (TR): it is the time taken to send the data from the local device to the remote cloud.

$$TR = \frac{f}{BW_{ch}} \qquad (3)$$

where f is the data size of the offloaded task and BW_{ch} is the network speed of the wireless channel used to offload data.

iv. Total execution time (TET): It is the summation of execution time and transmission time. If the execution of the task is selected to be on cloud, then only the transmission time is added, otherwise, its value is 0.

$$TET = ET_{m/c} + TR \qquad (4)$$

where $ET_{m/c}$ is the execution time on cloud/mobile device.

4. Intelligent techniques based Offloading

Intelligent technique is a process of making the system to learn automatically and improve from experience without being explicitly programmed. The proposed system accesses the data in the dataset and applies the learning algorithms to take the offloading decision. The primary aim is to allow system to learn automatically without human intervention or assistance and adjust actions accordingly. With this, the system is able to perform tasks that are highly complex in nature.

Fig. 3 shows the flowchart describing the offloading process for dynamic offloading system used in inference module. Machine-learning algorithms are applied on the dataset. All classifiers train the model and test the dataset with 10-fold cross-validation condition. In this work, four classifiers Logistic Regression (LR), Support Vector Machine (SVM), JRIP and J48 are used. JRIP is rule based classifier W.Junior et al (2019) and J48 is decision tree based classifier.

J48 classification algorithm is easy to understand as the derived rules have a very straightforward interpretation. JRIP Decision tree classifiers are quite popular techniques because the construction of tree does not require any domain expert knowledge or parameter setting, and is proper for exploratory knowledge discovery.

SVM algorithm, each data item as a point in n-dimensional space (where n is number of features) with the value of each feature being the value of a particular coordinate. Then, classification by finding the hyper-plane that differentiate the two classes.

Logistic regression measures the relationship between the categorical dependent variable and one or more independent variables by estimating probabilities using a logistic function, which is the cumulative logistic distribution. The model of logistic regression, however, is based on quite different assumptions (about the relationship between dependent and independent variables) from those of linear regression

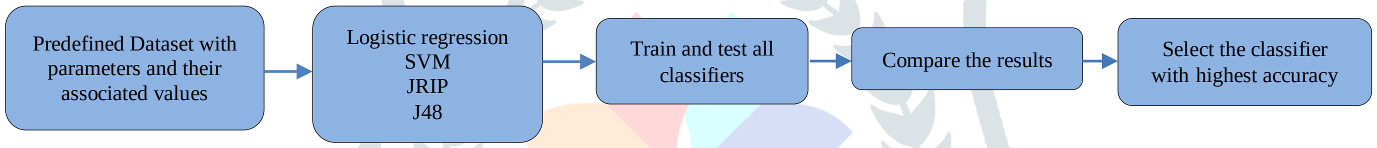

Predefined Dataset with parameters and their associated values
Logistic regression
SVM
JRIP
J48
Train and test all classifiers
Compare the results
Select the classifier with highest accuracy

Fig. 3 Flowchart for offloading process

When the decision unit takes the decision for application offloading, the setting of necessary API and dependencies between the user system and called system (cloudlet /workstation) is established. Once the connection is established successfully, the process of offloading commences. The Android-based mobile device is considered for local processing. The configuration of the mobile device is mentioned in Table 4. The decision unit interacts with the learning algorithms and takes the decision for offloading using eq. (5) and eq. (6) which are explained as follows:

For SVM Y.Yang et al (2015), *fitcsvm* function is made in MATLAB for classification, which trains a support vector machine model for binary classification. Results are obtained using 10 fold cross-validation Radial Basis Function (RBF) kernel function. The function is shown in Eq. (5) where variable A contains training instances and B contains training labels.

SVMModel = fitcsvm(A, B, 'Standardize',true,'KernelFunction','RBF', 'KernelScale','auto') (5)

Similarly for implementing LRC.Y.J.Peng et al (2002), the fundamental equation of the generalized linear model is as $g(E(y)) = \alpha + \beta x1 + \gamma x2$. Here, g() is the link function, E(y) is the expectation of target variable and $\alpha + \beta x1 + \gamma x2$ is the linear predictor (α, β, γ to be predicted). The role of link function is to 'link' the expectation of y to linear predictor. The function *fitglm* is made in MATLAB to implement logistic regression classifier which is shown in Eq. (6) where variable A contains training instances and B contains training labels.

GeneralizedLinearModel = fitglm(A, B, 'Distribution', 'binomial', 'link', 'logit') (6)

4.1 **Intelligent computational offloading**

The proposed intelligent computational offloading algorithm for task offloading decision and selection of appropriate location for offloaded task execution is presented in algorithm 1. This algorithm has two parts. In first part, task offloading decision is determined. Second part deals with the scheduling of offloaded task either on cloudlet or on workstation. The dataset with parameters used to take the offloading decision (energy level of mobile device, network status and deadline) and their associated values are initialized. The procedure *Isoffload* is called with the dataset and context information gathered by inference module. The decision unit of the DOS system uses this algorithm to take the decision. In result, it checks the minimum value for execution time with respect to cloud and mobile device. Whichever the value is minimum accordingly decision for offloading is taken. During classification, the execution time of the task is checked against the mobile device execution time. If the decision is favourable for the mobile device, then there is no offloading. This implies that the execution time of the task is within the deadline of the mobile device otherwise offloading is the option. If the decision is in favour of offloading, then the result of *Isoffload* procedure is passed to the *wheretoexecute* procedure for the selection of location for offloaded task execution. In this procedure, the decision about where to execute the offloaded task either on the workstation or on cloudlet is taken. This decision is based on the network status, available energy of the mobile device and result from *Isoffload* procedure. If network status and available energy of the mobile device are in the favourable side for offloading, then the decision for task execution on cloudlet or workstation is taken. If the total execution time of the task is within the deadline of the cloudlet, then the task is executed on the cloudlet. Otherwise, if the total execution time of the task is within the deadline of the workstation then the task is executed on the workstation. But, if both the cases are not satisfied then the task is deferred till suitable conditions.

Algorithm 1: Procedure for intelligent computational offloading

```
Input: dataset with defined parameters P=1..n
        values corresponds to P: V=1..m
    1.  procedure *Isoffload* (dataset, context)
            a)  for P=1: n //all parameter
            b)  for V=1:m // all values
            c)  $ET_m$=cal($ET_m$)  //using eq. 1
            d)  $ET_c$ = cal($ET_{c)}$)   //using eq. 2
            e)  M=Min($ET_m$, $ET_c$)
            f)  classifier (P, M); // apply all classifier
            g)  result=calculate(precision, specificity, sensitivity, accuracy);// using formulas (7-10)
            h)  end for
            i)  end for
            j)  return(result)
    2.  end procedure
    1. procedure *wheretoexecute* (result)
            if M ← Min($ET_{m)}$ then
                a) loc = mobile_device
            elseif result ^ network_status ^ available_energy = true
                a)  if total_time ≤ deadline_cloudlet then
                        loc = cloudlet
                b)  else if total_time ≤ deadline_workstation then
                        loc = workstation
        end if
    2. end procedure
```

4.2 The real-time application execution using learning algorithms

In this section, the proposed system is implemented using real-time face recognition applications. This application is based on the Scale Invariant Feature Transform (SIFT) algorithm D.G.Lowe et al (2004). SIFT is a powerful technique for general object recognition. It transforms image data into scale-invariant coordinates relative to local features. An important aspect of this approach is that it generates large numbers of features that densely cover the image over the full range of scales and locations. For image matching and recognition, SIFT features are first extracted from a set of reference images and stored in a database. A new image is matched by individually comparing each feature from the new image to the database and finding candidate matching features based on Euclidean distance of their feature vectors. The computation stages that are utilized to produce the image feature points are: building Gaussian Scale-space using Difference-of-Gaussian (DoG) function, keypoints detection and localisation; orientation assignment; and descriptor generation. In this regard, a face recognition application based dataset is developed and implemented in Matlab17b. The application is implemented with three different size images 2MP, 6MP and 8MP, respectively. On the basis of empirical study, parameters categorization and values are shown in Table 1. The instances of 2MP images from the dataset used for the experimental study are presented in Table 2. The description of Table 2 is as follows:

- In instance I1, the available energy is positive, the deadline is within the mobile device execution time and network status is good. The offloading decision is no ('N'). Even though network status is good but offloading decision is no 'N'. This is because the deadline for the task is within the mobile device execution time.
- In instance I2, values of the parameters - available energy on the mobile device and network status are both negative. The value of the deadline is within the total execution time of cloudlet. But, the offloading decision is no. It is due to the non-availability of available energy on the mobile device and network status. In this case, instance I2 has to wait for favourable condition.
- In instance I3, the values of the parameters - available energy on the mobile device is positive and network status is poor. The value of the deadline is within the total execution time of cloudlet. But, the offloading decision is no 'N'. It is due to the non-availability of good network conditions for the support of offloading.
- In instance I4, values of the parameters - available energy on the mobile device is negative and network status is average. The value of the deadline is within the total execution time of cloudlet. But, the offloading decision is no 'N'. It is due to the non-availability of available energy on the mobile device.
- In instance I5, values of the parameters - available energy on the mobile device is positive and network status is good. The value of the deadline is within the total execution time of cloudlet. The offloading decision is yes 'Y'. It is due to the influence of available energy of mobile device, the deadline for task execution and network status. Similarly, the offloading decision for 6MP and 8MP images is analysed.

The complexity of the dataset increases with the increase in the instances in the dataset. Similarly, instances are generated for the 6MP and 8MP images. Due to the uncertainty in the MCC environment, the need of machine learning plays a vital role to overcome the issue of taking offloading decision efficiently.

Based on the learning algorithms precision, sensitivity and specificity metrics D.G.Lowe et al (2004) are used to assess the accuracy as shown in Eq. (7) to Eq. (10). Table 4 shows the accuracy of SVM, LR, JRIP and J48 techniques on the proposed dataset.

$$Accuracy = \frac{TP+TN}{n} \qquad (7)$$

where TP is the number of True Positives, TN is True Negatives, FP is False Positives, and FN is False Negatives, n is TP + TN + FP + FN (the total number of observations).

$$Precision = \frac{TP}{TP+FP} \quad (8)$$

$$Sensitivity = \frac{TP}{TP+FN} \quad (9)$$

$$Specificity = \frac{TN}{TN+FP} \quad (10)$$

Table 1
Attributes and contextual values

Parameters	Values
Energy level of mobile device: ≤ 20% > 20%	0: no 1: yes
Network status: Bandwidth ≤ 10Mbps ≤ 20Mbps > 20Mbps	1: poor 2: average 3: good
Processing speed on mobile device (CPU Speed) Processing speed on cloudlet (CPU Speed) Processing speed on workstation(CPU Speed)	10 Mhz 30 Mhz 60 Mhz

Table 2
Dataset with values

Instance	Image size	Available energy on mobile device	Deadline	Network status	Execution time on mobile device (s)	Execution time on cloudlet (s)	Execution time on workstation (s)	Offloading decision
I1	2MP	1	4	3	4	5	8	N
I2	2MP	0	5	1	4	5	8	N
I3	2MP	1	5	1	4	5	8	N
I4	2MP	0	5	2	4	5	8	N
I5	2MP	1	5	3	4	5	8	Y

Table 3
Confusion Matrix

Actual	Offloading (Positive)	No-Offloading (Negative)
Positive	TP	FN
Negative	FP	TN

Table 4
Average value of metric (%)

Machine Learning technique	Specificity	Sensitivity	Precision	Accuracy
Logistic regression (LR)	95.37	98.93	98.93	98.26
SVM	95.35	97.86	98.92	97.39
J48	93.13	96.52	93.32	95.57
JRIP	91.21	98.08	93.21	95.67

Table 5
Experimental environment summary

Equipment	Configuration
Android device	Android 7.1.2,Octacore,1.40 GHz, 3GB RAM
Desktop cloudlet	Windows 7, 64 bit,8 GB RAM,
Work Station	Windows 10 Pro, 64-bit OS, intel(R), 2 cores, 2.10 GHz and 64 GB RAM

4.3 Classifiers performance

By using the above-mentioned formulas and confusion matrix (Table 3), the values for the aforementioned metrics for four algorithms with their average values are specified in Table 4. For JRIP and J48 results are discussed in W.Junior et al (2019). For SVM and LR, results are better than JRIP and J48. For specificity SVM and LR are similar i.e. 95.35% and 95.37% respectively. The sensitivity is maximum for LR with a rate of 98.93%. The precision rate is similar for SVM and LR with 98.92% and 98.93% respectively. In summary, LR is the algorithm that has most correctly predicted positive records and has the highest accuracy rate with 98.26%. LR outperforms than the other classifiers as the data in dataset is linearly separable in the input space.

5 Performance Evaluation

In this section, the performance of the proposed intelligent computation offloading algorithm is evaluated with three different size images (i.e., 2MP, 6MP and 8MP) and two different wireless channels (i.e., 4G and Wi-Fi). The proposed algorithm is evaluated in two aspects, namely the offloading decision and the scheduling of the offloaded task. The effect of task size on offloading and scheduling with different network conditions are also compared with various machine learning techniques.

5.1 Experimental setup

The experimental environment is composed of one android mobile phone, one windows based system as a cloudlet and one windows based high computing power system as workstation system and one D-Link (802.11n) access point. Table 5 shows the configuration of all equipment used in the experiment.

During the experiment, the applications are executed both locally on the mobile device and on remote servers. The communication between the mobile device and cloudlet is performed using the wireless network. Moreover, the mobile device connectivity between the mobile device and workstation is available with Wi-Fi and D link (4G). Table 6 shows the average download and upload speed of 4G and Wi-Fi network used for the implementation.

Table 6
Network configuration

Network	Download speed (Mbps)	Upload speed (Mbps)
4G	21.6	4.3
Wi-Fi	1.5	0.5

For the implementation of the face recognition system, different sizes of the images are considered. 24 images with 2 MP size, next 24 images with 6 MP size and next 24 images with 8 MP size are considered respectively.

5.2 Results

The total execution time of the images 2MP, 6MP and 8MP on the mobile device, cloudlet and workstation with 4G and Wi-Fi is presented in Table 7 and Table 8. In the dataset, the offloading decision is depending on the deadline, the available energy of the mobile device and network status. If the task can be completed within the deadline of the mobile device, then there is no offloading. Offloading is performed only if the mobile device has available energy to offload the task and favourable network condition otherwise task has to abort. Using this methodology, various machine-learning techniques are applied. The accuracy of the different machine learning techniques is presented in Table 4. Logistic regression has given the highest accuracy as compared to the other machine learning techniques. Based on discussion in section 4.2 and section 4.3, the logistic regression technique is the best classifier with the highest accuracy. The decision of selection of task to offload or not is a binary decision which is highly efficiently classified by logistic regression as compared to other classifiers.

After taking the offloading decision, scheduling of offloadable tasks for execution is required to be done in an efficient manner. As two locations namely cloudlet and workstation are considered for offloaded task execution. Based on the deadline, the most appropriate location for offloaded task execution is selected. If the task can be completed on a workstation within the deadline then it is offloaded to the workstation, if not then compared with the cloudlet. If all parameters are satisfied and the overall task can be completed within the deadline of the cloudlet than successful execution of the task is done on cloudlet. Otherwise, the task can be aborted.

Table 7
Total execution time with 4G

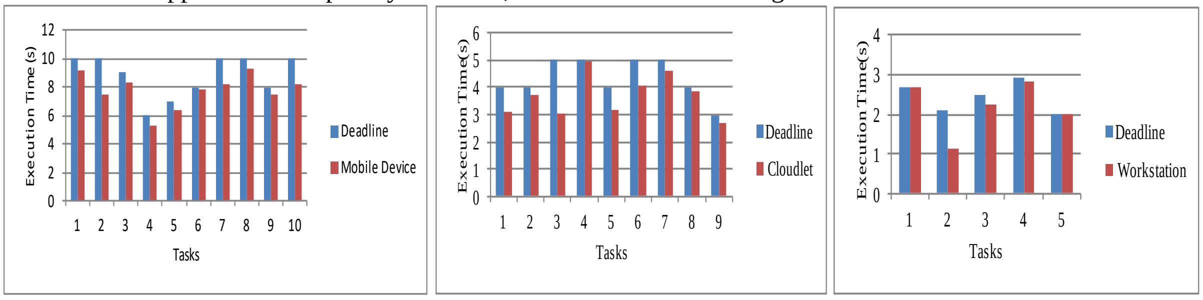
Execution environment	2 MP	6 MP	8 MP
Android device	5.01-8.3	23.2-25.12	34.31-36.12
Cloudlet	2.7-5.02	15.22-22.21	15.32-21.22
Work station	1.98-2.32	8.97-14.12	33.34-38.21

Table 8
Total execution time using with Wi-Fi

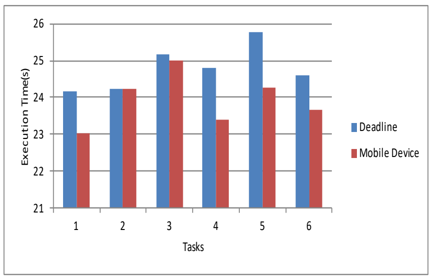
Execution environment	2 MP	6 MP	8 MP
Android device	12.01-14.3	25.2-30.42	45.11-46.32
Cloudlet	9.02-12.32	21.08-27.21	41.22-44.12
Work station	5.32-2.91	17.59-19.41	34.21-39.32

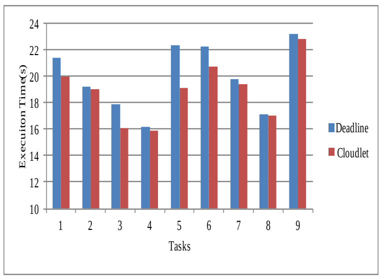
5.2.1 Total Execution time

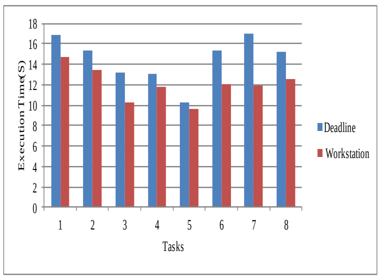
In Fig. 4, it shows the result for Table 7. The results indicate that out of 24 images for 2 MP size, 10 tasks are executing on the mobile device, 9 tasks are executing on cloudlet and 5 tasks are executing on the workstation with 4G network. All these offloading are done under the take care of the deadline of the tasks. It shows that for small size images more tasks can be executed on the mobile device. For 6 MP size images, 6 tasks are executing on mobile device 9 tasks are executing on cloudlet and 8 tasks are executing on the workstation. And for 8 MP size images, 5 tasks are executing on mobile device 7 tasks are executing on cloudlet and 12 tasks are executing on workstation respectively. As the size of the image increases, the complexity of the task execution increases which results in more execution time on the mobile device. This ultimately increases the utilization of the energy of the mobile device. Therefore, to save the energy of the mobile device offloading is beneficial. From the results, it has been proved that as the application complexity increases, the number of offloading of tasks also increases.



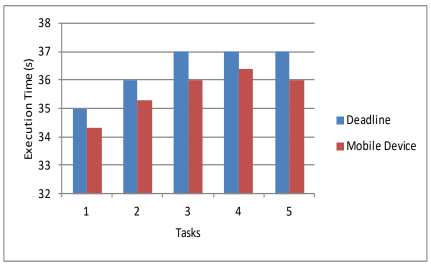
2 MP size 24 images

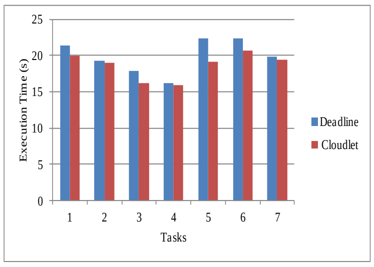


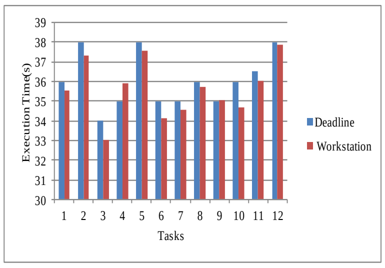




6 MP size 24 images







8 MP size 24 images

Fig. 4 Total execution time for SIFT operation on different size images with 4G

It has also been analysed that although the overall execution time on workstation and cloudlet is very low as compared to the mobile device but some tasks are executed on the mobile device. It is due to mainly two reasons. First, if the tasks can be executed within the deadline and with suitable energy level on the mobile device then there is no need for offloading. Second, it can save energy consumption during offloading as well as the cost for task execution on the cloud. Although in this paper, we have not showed the effect of cost on offloading. In future work, we will consider.

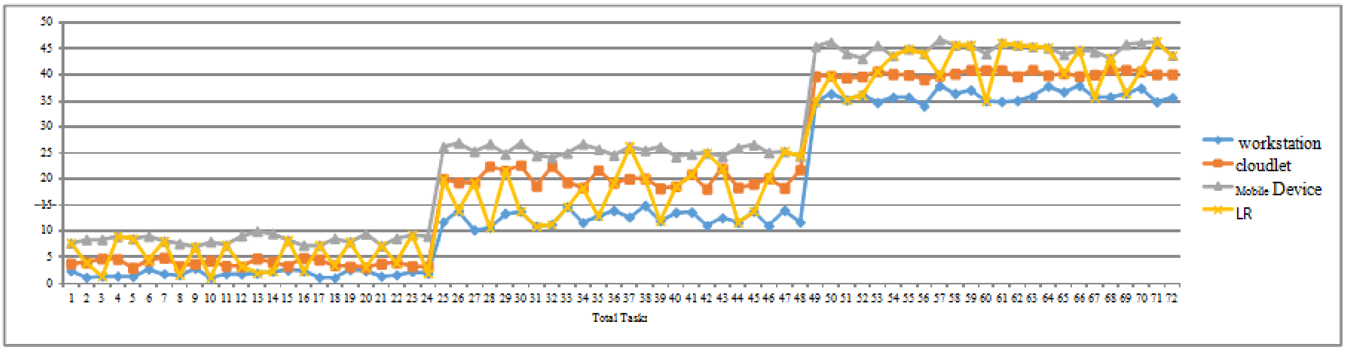
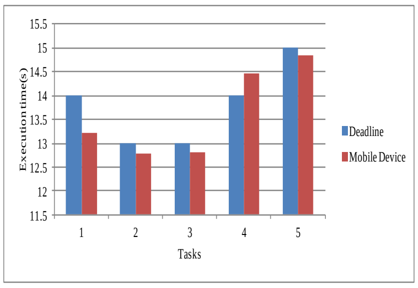
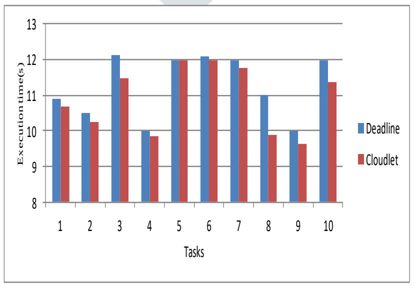
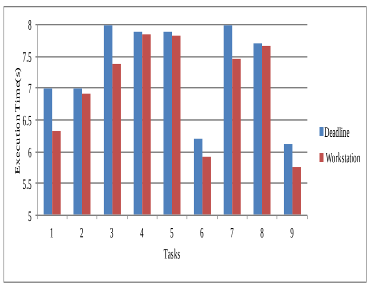


Fig. 5 Task execution using Machine Learning technique

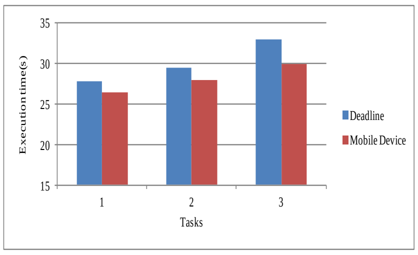
Fig. 5 presented the selection of appropriate resource among mobile device, workstation and cloudlet for task execution with the use of machine learning technique LR correspond to Table 7. The result shows the completion time of all 2 MP 24 images is within the 10s. For 6 MP next 24 images are in-between 10s to 27s and for next 24 images of size 8 MP is in-between 34s to 37s respectively. This decision is based on the dataset parameters and proposed intelligent offloading decision algorithm. The variation in time is due to the difference in processing capability of the mobile device, cloudlet and workstation.

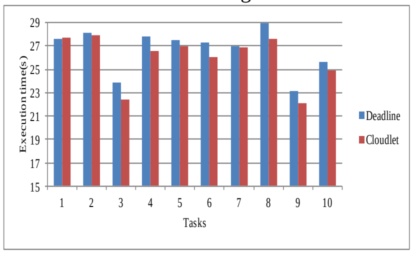


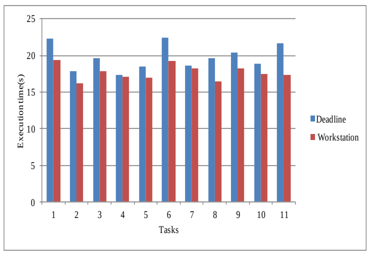




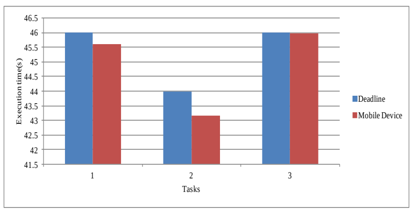
2 MP size 24 images

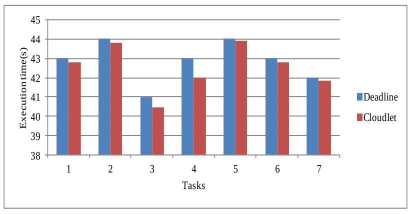


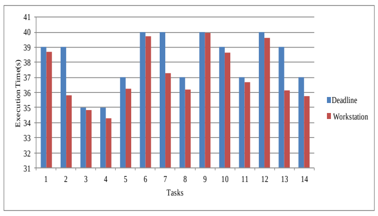




6 MP size 24 images







8 MP size 24 images

Fig. 6 Total execution time for SIFT operation on differnet size images with WiFi link

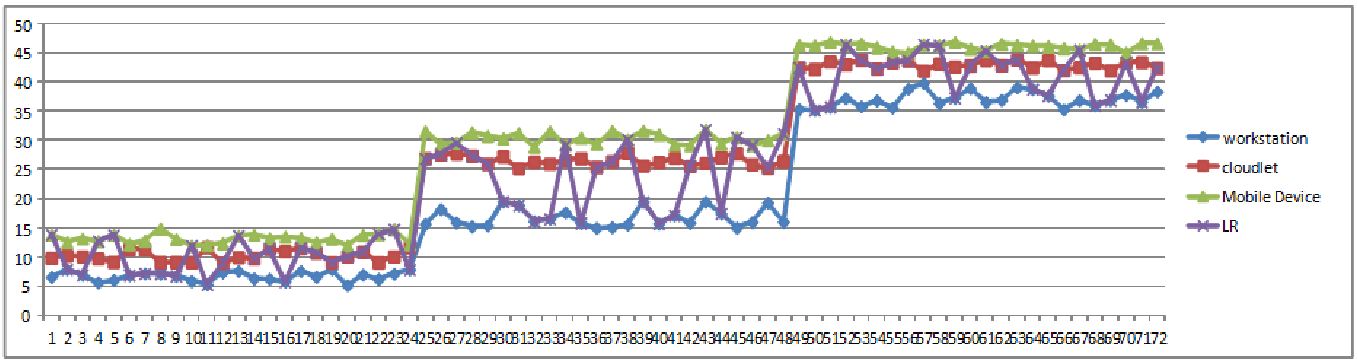


Fig. 7 Task execution using Machine Learning technique

The total execution time for 2 PM, 6 MP and 8 MP size images are shown in Fig. 6 using Wi-Fi. The results indicate that out of 24 images for 2 MP size, 5 tasks are executing on the mobile device, 10 tasks are executing on cloudlet and 9 tasks are executing on the workstation. For 6 MP size images, 3 tasks are executing on mobile device 10 tasks are executing on cloudlet and 11 tasks are executing on the workstation. And for 8 MP size images, 3 tasks are executing on mobile device 7 tasks are executing on cloudlet and 14 tasks are executing on workstation respectively.

Fig. 7 shows the overall result corresponds to Table 8 using machine-learning technique LR for task execution. This result shows the selection of an appropriate location for task execution within the deadline. The result shows the completion time of all 2 MP 24 images is within the 15s. For 6 MP next 24 images are in-between 16s to 33s and for next 24 images of size 8 MP is in-between 35s to 47s respectively. Hence, it proves that for small size images, if the configuration and energy level of the mobile device is sufficient then offloading can be avoided. As the complexity of the tasks increases, the level of offloading also increases. When the machine-learning technique is applied to the dataset, there is a difference of offloading. It is due to the change in available energy on the mobile device and network status parameters. Among the parameters energy on the mobile device and network status, if anyone parameter is negative than it directly shows the effect on offloading. Therefore, there is an offloading difference in Fig.5 and Fig.7 when performed with Wi-Fi and 4G. There is also a difference in total execution time on cloudlet and workstation. The reason is transmission time with Wi-Fi and 4G. 4G is providing 21.6 Mbps download rate and 4.3 Mbps uplink rate whereas 1.5 Mbps download rate and 0.5 Mbps uplink rate with Wi-Fi. The tasks offloading with Wi-Fi consumes more time as compared to tasks offloading with 4G.

6 Conclusion

In this article, we proposed the intelligent computational offloading algorithm in the mobile cloud computing environment. The algorithm is defined in two parts that operate on the proposed dataset. The parameters of the dataset are considered according to the real-time application environment. The dataset also considered the dynamic nature of the wireless network to take the offloading decision. In order to take the offloading decision intelligently, machine-learning techniques such as LR, SVM, JRIP and J48 are used. Depending on the performance of the machine learning techniques, the technique with the highest accuracy is selected to take the offloading decision. After taking the offloading decision, the scheduling of offloaded task either on cloudlet or on the workstation is done. To make the work comparable with the real-time application, the deadline of the task is considered to perform the scheduling. If the task can be completed within the deadline of the cloudlet then the task is offloaded on cloudlet otherwise offloaded on the workstation. For the experimental study, four machine learning techniques: SVM, LR, JRIP and J48 are considered. The results show the performance of the machine learning techniques for metrics: sensitivity, precision, specificity and accuracy. The results show the accuracy of JRIP, J48, LR, and SVM. The overall performance of LR is the highest among the other machine learning techniques. Therefore, it is concluded that the proposed intelligent computational offloading algorithm can efficiently offload and schedule the task. It also shows that the level of offloading increases with the increase in the computational complexity of the task. For future work, machine learning techniques can be used for the scheduling of tasks with respect to mobile edge computing. To take the scheduling decision, parameters of machine learning techniques can be tuned and comparison can be made.

Acknowledgment

I would like to express my heartfelt gratitude to all the researchers who have contributed in this field of study.

References

[1] L. Wang *et al.*, “Cloud computing: A perspective study,” *New Gener. Comput.*, vol. 28, no. 2, pp. 137–146, 2010.
[2] S. Singh, Y. S. Jeong, and J. H. Park, “A survey on cloud computing security: Issues, threats, and solutions,” *J. Netw. Comput. Appl.*, vol. 75, pp. 200–222, 2016.
[3] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, “A Survey of Mobile Cloud Computing : Architecture , Applications , and

Approaches," no. Cc, pp. 1–38.

[4] N. Fernando, S. W. Loke, and W. Rahayu, "Mobile cloud computing: A survey," *Futur. Gener. Comput. Syst.*, vol. 29, no. 1, pp. 84–106, 2013.

[5] S. Guo, B. Xiao, Y. Yang, and Y. Yang, "Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing," *Proc. - IEEE INFOCOM*, vol. 2016–July, 2016.

[6] K. Akherfi, M. Gerndt, and H. Harroud, "Mobile cloud computing for computation offloading: Issues and challenges," *Appl. Comput. Informatics*, vol. 14, no. 1, pp. 1–16, 2018.

[7] I. Technologies, "Overview of Offloading in Smart Mobile Devices for Mobile Cloud Computing," vol. 5, no. 6, pp. 7855–7860, 2014.

[8] S. Kosta and R. Mortier, "ThinkAir : Dynamic resource allocation and parallel execution in cloud for mobile code offloading."

[9] E. Cuervo, A. Balasubramanian, and D. Cho, "MAUI : Making Smartphones Last Longer with Code Offload."

[10] B. Zhou, A. V. Dastjerdi, R. N. Calheiros, S. N. Srirama, and R. Buyya, "A Context Sensitive Offloading Scheme for Mobile Cloud Computing Service," *Proc. - 2015 IEEE 8th Int. Conf. Cloud Comput. CLOUD 2015*, pp. 869–876, 2015.

[11] M. Alizadeh and W. H. Hassan, "Challenges and opportunities of mobile cloud computing," *2013 9th Int. Wirel. Commun. Mob. Comput. Conf. IWCMC 2013*, pp. 660–666, 2013.

[12] H. Allam, N. Nassiri, A. Rajan, and J. Ahmad, "A critical overview of latest challenges and solutions of Mobile Cloud Computing," *2017 2nd Int. Conf. Fog Mob. Edge Comput. FMEC 2017*, pp. 225–229, 2017.

[13] M. Shiraz, E. Ahmed, A. Gani, and Q. Han, "Investigation on runtime partitioning of elastic mobile applications for mobile cloud computing," *J. Supercomput.*, vol. 67, no. 1, pp. 84–103, 2014.

[14] D. Meil??nder, F. Glinka, S. Gorlatch, L. Lin, W. Zhang, and X. Liao, "Using mobile cloud computing for real-time online applications," *Proc. - 2nd IEEE Int. Conf. Mob. Cloud Comput. Serv. Eng. MobileCloud 2014*, pp. 48–56, 2014.

[15] R. Kumari and S. Kaushal, "Energy efficient approach for application execution in mobile cloud IoT environment," *Proc. Second Int. Conf. Internet things, Data Cloud Comput. - ICC '17*, pp. 1–8, 2017.

[16] 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," *Inf. Syst. Front.*, vol. 16, no. 1, pp. 95–111, 2014.

[17] B. Zhou, A. V. Dastjerdi, R. N. Calheiros, and R. Buyya, "An online algorithm for task offloading in heterogeneous mobile clouds," *ACM Trans. Internet Technol.*, vol. 18, no. 2, 2018.

[18] A. Boukerche, S. Guan, and R. E. De Grande, "Sustainable offloading in mobile cloud computing: Algorithmic design and implementation," *ACM Comput. Surv.*, vol. 52, no. 1, 2019.

[19] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing," *Futur. Gener. Comput. Syst.*, vol. 28, no. 5, pp. 755–768, 2012.

[20] H. Eom, P. S. Juste, R. Figueiredo, O. Tickoo, R. Illikkal, and R. Iyer, "Machine learning-based runtime scheduler for mobile offloading framework," *Proc. - 2013 IEEE/ACM 6th Int. Conf. Util. Cloud Comput. UCC 2013*, pp. 17–25, 2013.

[21] P. A. L. Rego, F. A. M. Trinta, M. Z. Hasan, and J. N. De Souza, "Enhancing Offloading Systems with Smart Decisions, Adaptive Monitoring, and Mobility Support," *Wirel. Commun. Mob. Comput.*, vol. 2019, 2019.

[22] W. Junior, E. Oliveira, A. Santos, and K. Dias, "A context-sensitive offloading system using machine-learning classification algorithms for mobile cloud environment," *Futur. Gener. Comput. Syst.*, vol. 90, pp. 503–520, 2019.

[23] P. Nawrocki and B. Sniezynski, "Adaptive Service Management in Mobile Cloud Computing by Means of Supervised and Reinforcement Learning," *J. Netw. Syst. Manag.*, vol. 26, no. 1, pp. 1–22, 2018.

[24] O. B. Yetim and M. Martonosi, "Dynamic Adaptive Techniques for Learning Application Delay Tolerance for Mobile Data Offloading," *2015 IEEE Conf. Comput. Commun.*, pp. 1885–1893, 2015.

[25] P. B. Costa, P. A. L. Rego, L. S. Rocha, F. A. M. Trinta, and J. N. De Souza, "MpOS: A Multiplatform offloading system," *Proc. ACM Symp. Appl. Comput.*, vol. 13–17–Apri, pp. 577–584, 2015.

[26] P. A. L. Rego, P. B. Costa, E. F. Coutinho, L. S. Rocha, F. A. M. Trinta, and J. N. De Souza, "Performing computation offloading on multiple platforms," *Comput. Commun.*, vol. 105, pp. 1–13, 2017.

Y.Yang et al (2015) Y. Yang, J. Li, and Y. Yang, "The research of the fast SVM classifier method," *2015 12th Int. Comput. Conf. Wavelet Act. Media Technol. Inf. Process. ICCWAMTIP 2015*, no. 1, pp. 121–124, 2015.

[28] C. Y. J. Peng, K. L. Lee, and G. M. Ingersoll, "An introduction to logistic regression analysis and reporting," *J. Educ. Res.*, vol. 96, no. 1, pp. 3–14, 2002.

[29] D. G. Lowe, "[10](IJCV2004)SIFT.pdf," pp. 1–28, 2004.