A LSTM Approach for Secure Energy Efficient Computational Offloading in Mobile Edge Computing

S. Anoop
Research Scholar, Department of CSE, Noorul Islam Centre for Higher Education, Thuckalay, Kumaracoil, Tamil Nadu, India. E-mail: anoopsivasankar@gmail.com

Dr.J. Amar Pratap Singh
Professor, Department of CSE, Noorul Islam Centre for Higher Education, Thuckalay, Kumaracoil, Tamil Nadu, India.

Received April 30, 2021; Accepted August 28, 2021
ISSN: 1735-188X
DOI: 10.14704/WEB/V18I2/WEB18359

Abstract

Mobile technologies is evolving so rapidly in every aspect, utilizing every single resource in the form of applications which creates advancement in day to day life. This technological advancements overcomes the traditional computing methods which increases communication delay, energy consumption for mobile devices. In today's world, Mobile Edge Computing is evolving as a scenario for improving in these limitations so as to provide better output to end users. This paper proposed a secure and energy-efficient computational offloading scheme using LSTM. The prediction of the computational tasks done using the LSTM algorithm. A strategy for computation offloading based on the prediction of tasks, and the migration of tasks for the scheme of edge cloud scheduling based on a reinforcement learning routing algorithm help to optimize the edge computing offloading model. Experimental results show that our proposed algorithm Intelligent Energy Efficient Offloading Algorithm (IEEOA), can efficiently decrease total task delay and energy consumption, and bring much security to the devices due to the firewall nature of LSTM.

Keywords

Deep Learning, Deep Neural Network, Long Term Short Memory, Mobile Edge Computing.

Introduction

Smart mobile systems have become widely utilized in everyday life over recent years which includes smartphones, tablet computers, wearable devices (T.Q. Dinh et al. (2017)), smart cars, etc. The popularity of mobile cellular connectivity and fast 5G technology development has made them a widespread presence. The growing mobile traffic and the

complex computer systems provide tremendous difficulties for networking and computer resources. In recent decades, cloud technology and wireless communication have advanced considerably. Although local computer technology is only able to operate a few simple computing tasks, for reasons such as poor computing and mobile device storage capacity and limited battery capacity in hardware design. The uplines of the Cloud can be used to complete computer-intense and data-intensive tasks (S. Agarwal et al. (2014)). That implies cloud storage, computation, and communication resources can remedy the inadequacies of local devices in these areas. This is the scenario if the volume of users is low or the kind of application is simple.

In addition, a high amount of network infrastructure resources in multi-user mode is required to send computer activities from mobile devices to the cloud, as it deals with a huge amount of data. It readily exceeds the security load threshold of the network, causes network congestion, and causes an unacceptable delay in communication (YC Hu et al. (2015)). Therefore, conventional cloud offloading techniques are not suitable for critical computational tasks in the age of 5G. New computer modes are needed to fulfill the low time, dependability, and large complexity requirements of these computational tasks (Y. Cao et al. (2014)). ETSI Mec ISG (Industry Specification Group) is a revolutionary computing method composed of six members which include Nokia and Huawei (X. Chen et al. (2016)). Today, 5G research became the main theory as well as a conceptual framework. Cloud and cloud storage in areas close to a mobile user are supported by edge computing, providing 5G services to mobile devices using a server at the edge of the internet (which include Wi-Fi access point), routers, base stations, switches, cloud platforms or data centers, as well as any other storage and computational capability enabled devices.

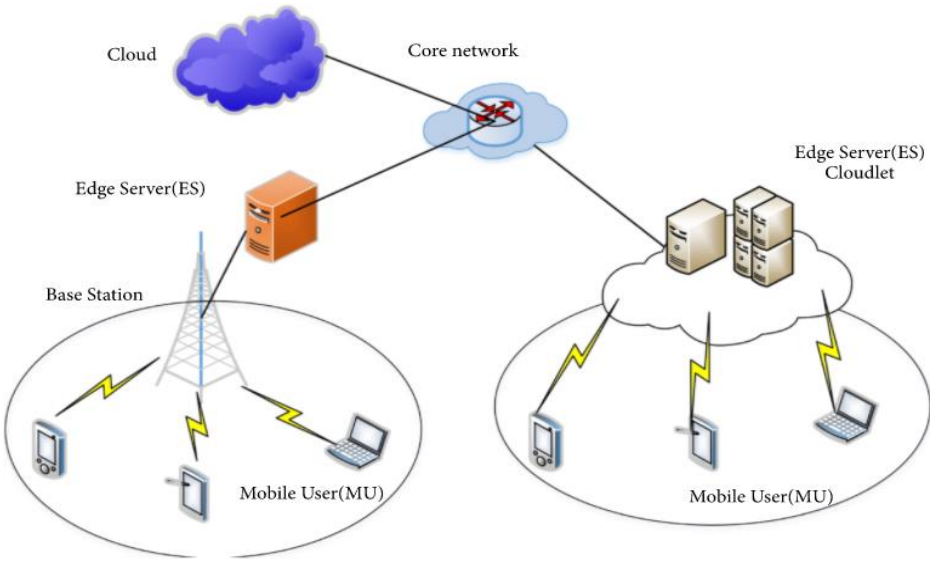

Figure 1 Mobile Edge Computing architecture

Although edge computing is accepted as an additional mode in cloud computing, the simultaneous processing of data requests and calculation tasks still creates a significant demand on intelligent communication systems in the age when 5G mobile communications are being commercialized (M. Chen et al. (2019); M. Chen et al. (2018); J. Feng et al. (2018); G. Orsini et al. (2016)). Intelligent gadgets have varied computing capabilities. There are now numerous kinds of intelligent apps. But the applications include a large number of computing activities and data types, which include enormous unstructured data like text, audio, video, and pictures and structured data such as digital signals. And all these complex calculation jobs typically may be split into several parallel processing sub-tasks. Dynamically altering network resources. In MEC design, computer services are limited by a wide number of unsafe factors, for example, the volume of mobile users (calculation workloads), network security, communications, and resource allocation policies F. Zhang et al. (2019); S. Yu et al. (2017); P. Dai et al. (2019); J.C. Guevara et al. (2020)). The issue in current MEC research is how to constantly deliver services with high dependability and minimal latency for consumers by jointly optimizing the aforementioned factors. The edge cloud's computing capability is diverse and limited. So, when work is transferred onto the edge cloud, the computational complexity is be a problem. At this point, we need to study the different techniques of computational task offloading to the cloud which respond to the dynamic changes of computational resources. In addition, the computer capability of the edge server is not sufficient for all sorts of calculation jobs, in comparison with the conventional cloud server. Therefore, if the traffic statistics of the job vary dynamically, it is important to address completely the issue of computational migration (L. Ale et al. (2019); M. Al-Khafajiy (2018)).

It is therefore important that finite and diverse computing resources are fully utilized on the edge cloud to develop an improved smart offloading approach and decrease processing latency. In order to develop the optimal offloading strategy for the initial time, when mobile consumers request a service, it is important to make a preliminary estimate of the volume of work required to increase the efficiency of the offloading strategy. Deep Learning (DL) is an advanced approach and technique in the field of data analysis and data processing. As an IT industry, deep-learning technological advances enable a range of non-structured data gathered from mobile devices to be processed and extracted in-depth (especially if a significant quantity of historical data is acquired), therefore providing MEC's system with smarter cognitive services (M. Chen et al. (2019)). Advanced computing employs algorithms like RNN (M. Chen et al. (2018)) as the newest optimal prediction technology to deliver cognitive capability for network services, loudening services, traffic, and others to enhance the quality of experience (QoE) and quality of service (QoS). Furthermore, Edge

Node and DL technology are anticipated to foster edge computing growth through the provision of distributed DL services.

Challenges of Existing Systems

With the current system, several obstacles are categorized as 7 major problems that are essentially having natural dynamic behavior and need to be dealt with dynamically. Due to its high data transfer rates the absence of pre-defined information to resolve problems, the improvement of associated offloading metrics, the application of more advanced methods of machine learning such as reinforcement learning derivatives and online learning, or a combination of machine learning approaches. The 7 main challenges are listed below;

a. Scheduling
b. Interoperability
c. Mobility
d. Scalability
e. Security
f. Fault tolerance
g. Partitioning.

Objectives

This paper focuses on a secure and energy-efficient offloading MEC using a deep learning method in which the following keynotes are mentioned below;

a. Two aspects of infrastructure and logic are used for designing new MEC offload-based computing architecture.
b. An LSTM algorithm-based computational task prediction method is being suggested for the MEC framework by integration of Deep Learning, edge computing, and local computing.
c. For mobile devices, the optimum offloading computing approach is given based on workload prediction.
d. A reinforcement learning based routing of these predicted tasks to edge servers.
e. Finally, this is compared with existing systems to analyze how much our proposed system (IEEOA) enhances the flavor of security and energy consumption.

Organization of paper: Section 2 analyses and investigates similar research on the offloading and scheduling of computations and highlights their limitations. Section 3 then

provides a intelligent computer-based offloading MEC architecture which proposes LSTM-based computational prediction method, the computing offloading strategy for a mobile device to migrate computing tasks into edge cloud as well as their routing through reinforcement learning. Section 4 provides a MEC environment for simulation and tests are carried out with time delays to analyze the impact of the computation offloading and intelligent task prediction method.

Related Works

Orsini et al. (G. Orsini et al. (2016)) highlight that partially offloading involves estimates of the cost of computation of each component for the application, therefore placing extra pressure on calculating resources and reserves of energy. Nevertheless, such computations may intelligently select the optimum collection of components to be offloaded so that the volume of data transmission is minimized and latency, as well as overall energy consumption, are reduced. We examine partial offloaded schemes in the proposed work. Hence partial offloading decreases delay energy consumption and needless overhead transmission relative to the complete discharge system.

The collaborative edge offloading technique suggested by Al-Khafajiy et al. enables the fog node collaboration for big data processing using pre-defined fog characteristics. The fact that all essential information about the Fog Node capabilities (i.e. processors) is known in advance makes this technique efficient in processing data at the edge level on a timely basis. However, this technique misses the fog nodes' energy usage, which is not energy efficient.

Li et al. (2018) propose a deep reinforcement learning strategy to strengthening the entire offloading system. Nevertheless, global minima may not be ensured in reinforcement learning techniques because of its unexpected nature of learning. Thus, deep learning techniques observed in recent years have become quite prominent in the computational offloading process in MEC. Fast precise decision-making and greater computing speed with trained models are the significant benefits of deep learning. Using deep learning the learned model can prevent exhaustive computations to find the best solution. Anas et al. (2017) take computational utilization and access probability into consideration and develop a performance model based on queuing theory to address the workload balancing between service providers within a federated cloud environment.

Ma et al. (2020) examine the collaboration between edge nodes and study workload scheduling to reduce the traffic and response time in mobile edge computing. They offer a heuristic algorithm for the scheduling of workload based on water-filling to reduce

complexity in computation. Fuzzy logic is an efficient approach for solving the edge computing workload scheduling problem described in recent years.

In order to tackle the problem of workload orchestration in edge computing systems, Sonmez et al. (2019) adopt a fuzzy logic method. The approach of the offloaded tasks takes into account the characteristics and the present state of computational as well as networking resources and utilizes fuzzy rules to specify networking, computing, and task-specific workload orchestration activities to make the decision on allocating location for the workload execution in the overall edge computing system.

The Foggy Software Platform for the orchestration of loads and resources in the fog computing environment is proposed by Santoro et al. (2017). It plans to do activities on the basis of computing, storage, or network resources.

System Architecture

We are proposing a novel MEC design based on intelligent computing offloading to cope with difficulties of large data exchange, power consumption, and an unacceptable latency in computational offloading in the cloud computation model as illustrated in Figure 2.

Mobile, smartphones, and tablet computers (M. Chen et al. (2017); Y. He et al. (2018); J. Liu et al. (2019); L.T. Tan et al. (2018); L. Huang et al. (2018)) devices interact with the cloud server directly in conventional cloud computing mode, collect local user data and immediately pass computation workloads onto the remote cloud. To improve data transfer and to avoid latency an intermediary edge cloud layer is created in the cloud architecture between MEC Architecture and local devices used for processing and communications. Multi user Edge servers, which are also known as edge nodes, like the base station, wireless access point, and routers can connect with local mobile devices through wireless media and share tasks via the pull links. Cloud servers are able to deliver deep learning services featuring powerful processing as well as storage resources. The integration of cloud technology and deep learning is considered a key component to increase intelligence of edge cloud computing by making them improving energy efficiency.

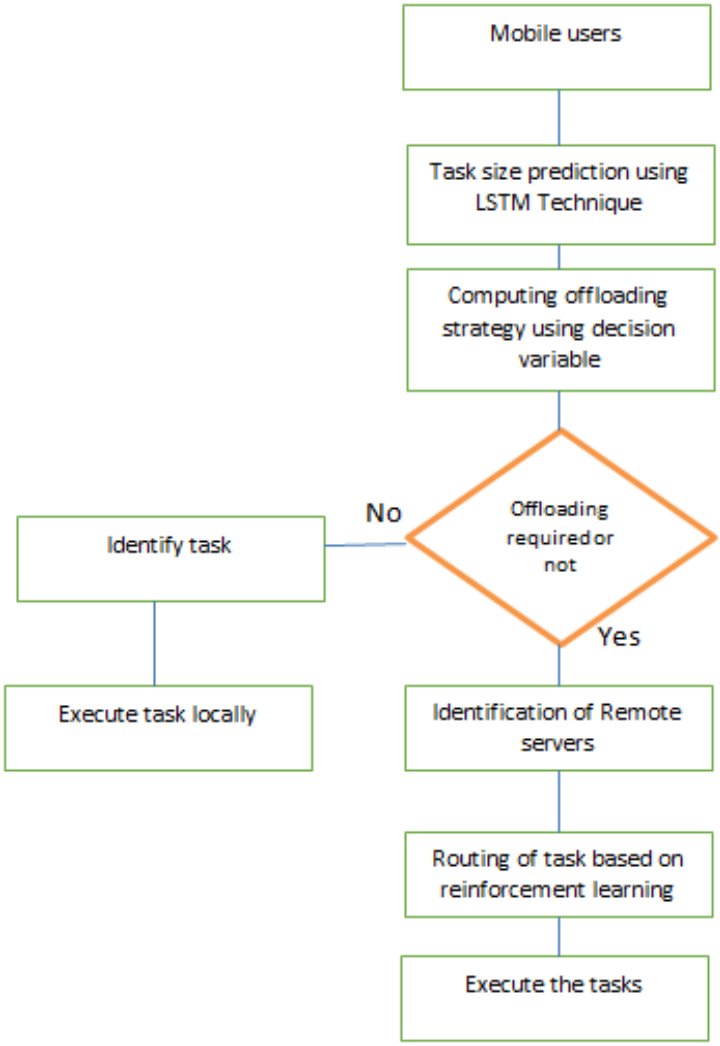

Figure 2 Proposed Architecture

Mobile users only can do a few basic computer activities locally, owing to restricted computing and storage capacity of local devices. Edge cloud would be offloaded more difficult tasks over the wireless channel. Edge computer nodes will decide if this job is to be handled locally or moved to other nodes, taking into account the expected task complexity, node computing capacity, power reserve nodes, as well as other variables. For more complex applications, service data are typically transmitted directly over the distant cloud. Some computer nodes also schedule the cloud for computational activities. This ensures intelligent services at the cost of communication delays.

The purpose and benefits of developing a smart MEC architecture computational offloading is to diverse computing jobs and for heterogeneous data applications. If computational activities can be forecast in advance for the type, size, and computing resources based on prediction, one cn determine whether to offload or not. Since network

communications and computer resources which are changing dynamically an optimal offloading approach can help enhance QoE from a variety of aspects, including computation latency and complexity. Optimized transfer or routing of tasks can help minimize network access congestion. The MEC architecture, which is based on intelligent task predictions and computational offloading, can enable local devices to conduct more sophisticated processing while reducing the load on the distant cloud. Thus a deep learning approach using LSTM technique to compute task prediction together with a deep reinforcement learning based routing for channel allocation can easily carry out multiple task transmission from local devices to edge node with the lowest energy consumption with appropriate computational offloading.

Algorithm of LSTM

The offloading approach cannot ensure minimum latency since edge computing and conventional cloud computing offloading modes only consider direct offloading of computational workloads. It's not smart enough, therefore. Thus, in this work, three elements optimized and enhanced the loading method of edge computing: (1) Algorithm based on LSTM computing task prediction. In order to forecast functionalities and to help judge computer delays in the offload approach, the in-depth learning approach is applied. (2) Mobile devices computer offload technique based on job forecasting. Once the LSTM algorithm has been utilized for precise task traffic data, an in-depth assessment is carried out to offload performance based on various aspects of edge cloud computation nodes, with the aim of achieving the optimal offload strategy. (3) Migration of edge cloud scheduling scheme of computing tasks. A new task migration system is being introduced to improve computation offloading technique.

The process flow is as illustrated in Figure 3 for the task prediction.

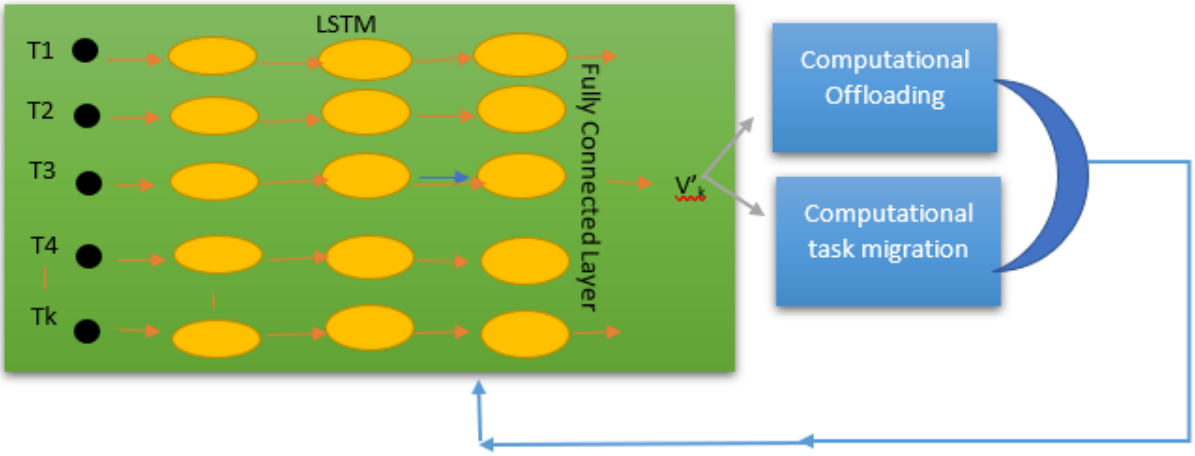

Figure 3 Process flow of computational offloading task size prediction using LSTM

Computation Task Size Prediction using LSTM

As shown in Figure 3, K-mobile users are expected to offload computing workloads to edge cloud computing nodes connected to their mobile networks for processing. To develop a better offloading technique, we must first determine the traffic data for each computing activity, also known as the computation offloading data volume. Unlike previous techniques for the description of computer functionality, a profound LSTM-based learning algorithm is used to anticipate computational tasks (Yiming Miao et al. (2020)). Set $V_k \in \{V_1, V_2, V_K\}$, the data size. W_f, W_C, bf, b_C, are utilized to describe the biases and weights of forgetting and input gates, σ and tanh are employed as activation functions in multi-level LSTM architecture. Forget gate can be specified as

$$f_k = \sigma(W_f \cdot [h_{k-1}, V_k] + b_f)$$

The input gate is defined as

$$C_k = f_k * C_{k-1} + \sigma(W_i \cdot [h_{k-1}, V_k] + bi) * \tanh(W_C \cdot [h_{k-1}, V_k] + b_C)$$

The hidden layer output may be specified as $h_k = \sigma(Wo \cdot [h_{k-1}, V_k] + b_o) * \tanh(Ck)$. Finally, a complete connection layer combines the previously extracted characteristics to produce the $\tilde{V}_k \in \{\tilde{V}_1, \tilde{V}_2, \ldots, \tilde{V}_K\}$ output sequence. In this case, $\tilde{V}_k$ denotes the expected data amount for computation task k. This anticipated data will be used in a subsequent compute offloading technique. As a result, the algorithm's optimization aim is to increase task data size prediction accuracy ($|\tilde{V}_k - V_k| \propto 0$) as much as feasible.

Computational Offloading Strategy

A mobile device can specify an offloading mechanism for a computing task based on its processing capabilities. Tasks are often carried out in either locally or remotely performed at the cloud edge platform depending on the decision variable. The number of bits that the computation job k is being offloaded is represented by the task offloading variable β_k [0, 1]. When k is 0, the job should be handled locally while k is 1, job has to be handled on the edge cloud.

If $\beta_k \in (0, 1)$, $\beta_k \tilde{V}_k$ should be sent to the edge cloud for processing, whereas $(1 - \beta_k \tilde{V}_k)$ should be handled locally. To execute an offloading operation, we must first determine the quantity of data that needs to be offloaded as well as the essential features of edge cloud computing nodes that are linked to a mobile user. Consider the total frequency of CPU cycles required by edge computing node i to perform job k, which is $C_{i,k}$, and the computing

frequency of task k, which is $F_{i,k}$. As a consequence, the time $t_{i,k}^{proc}$ that node i needs to process k may be calculated as follows.

$$t_{i,k}^{proc} = \frac{\beta_k \tilde{V}_k C_{i,k}}{F_{i,k}}$$

The uplink wireless channel is used for mobile device offloading. As a result, the maximum uplink transmission rate $UL_{i,k}$ (M. Chen et al. (2018)) for task offloading is expressed using Shannon's theorem:

$$UL_{i,k} = Blog_2\left(1 + \frac{p_k h^2}{\sigma^2 + w_{i,k}}\right)$$

where B denotes channel bandwidth, σ^2 denotes noise power, p_k denotes mobile device transmitting power, h^2 denotes wireless channel gain, and $w_{i,k}$ denotes the power of interference during offloading.

The transmission latency $t_{i,k}^{up}$ of task k may be calculated as follows:

$$t_{i,k}^{up} = \frac{\beta_k \tilde{V}_k}{UL_{i,k}}$$

If a mobile device sends a computing job k to the edge cloud, the total delay T_k^{edge} maybe defined as:

$$T_k^{edge} = t_{i,k}^{up} + t_{i,k}^{proc} + t_{i,k}^{down}$$

Where, $t_{i,k}^{down}$ means computing delay. The results data package is often small, and the downlink between a mobile user and an edge node has enough bandwidth. This means that the downlink transmission delay may be ignored. T_k^{edge} can so be simplified as:

$$T_k^{edge} = t_{i,k}^{up} + t_{i,k}^{proc} = \frac{\beta_k \widetilde{V_k}}{UL_{i,k}} + \frac{\beta_k \widetilde{V_k} C_{i,k}}{F_{i,k}}$$

At the present, the following is the general equation for the overall delay in the processing of computation task k:

$$T_k = T_k^{edge} + (1 - \beta_k)T_{local} = \frac{\beta_k \widetilde{V_k}}{UL_{i,k}} + \frac{\beta_k \widetilde{V_k} C_{i,k}}{F_{i,k}} + (1 - \beta_k)T_k^{local}$$

The total delay in the offloading of the computation is related to the task data size V, the computational resource R on the mobile device, and Q is computing resource on the edge of the cloud, according to the above-mentioned formula. As a minimum delay, the above derivative procedure may be simplified:

$$minimize\ \left(T_k \mid \tilde{V}_k, R_k, Q_{i,k}\right)$$
$$\beta_k$$
$$subject\ to\ T_k \leq T_k^{local}$$

Total Energy consumption ECi, due to delay can be written as:

$$EC_i = Elocal_i + Edec_i,\ \beta_k = 0,$$
$$EC_i = ERemote_i + Edec_i,\ \beta_k = 1.$$

Total cost can be computed by combining the cost incurred for time delay in decision making and energy consumed for task scheduling.

$$Total_{cost} = \alpha(Elocal_i + Edec_i) + T_k$$

(ci) represents the local cost when ci executes locally on UE and (ci) is the remote cost when ci executes remotely on MES.

(ci) can be calculated as:

$$fcl(ci) = \delta 1\left(\frac{Lli + Ldi}{Lmax}\right) + \delta 2\left(\frac{Elocali + Edeci}{Emax}\right),$$

where δ1 and δ2 are the weighting coefficients and Lmax is the total delay time.

Cost for remote execution can be calculated as

$$(ci) = \delta 1\left(\frac{Lli+Ldi}{Lmax}\right) + \delta 2\left(\frac{Elocali+Edeci}{Emax}\right) + T_{sch}$$

where T_{sch} is the scheduling time delay.

Computational Task Migration

Figure 2 shows that at the edge cloud several computer nodes serving a mobile network are typically present. This is because the coverage of each node is varied and there are various objects to be served.

If a system problem, hardware damage, or excessive load happens on a node while a computation job is running, the computation offloading or continuing work will be disturbed. A new approach to help calculates the migration task across clouds is necessary at this moment. Task k to N subtasks, that is to say, k = (k1, k2, . . . , kN). The data size may then be stated for all task subtasks k as follows

$$\{\phi k1 , \phi k2 , . . . , \phi kN \}$$

Sub-tasks are assumed to be no longer divisible and a particular task has to be completed fully on a computer node. If subtasks 1 to n are performed on node i, subtasks n + 1 to N are migrated to j node for execution, the migration delay for n + 1 to N to j node may be stated as follows:

$$t_{i,j}^{mig} = \frac{\sum_n^N \varphi_{k_e}}{r_{i,j}}$$

The delay in the migration of subtasks n+1 to N of node j is:

$$t_{j,k}^{proc} = \frac{\sum_n^N \varphi_{k_e} C_{j,k}}{F_{j,k}}$$

The standard expression for the overall delay of a computation migration task may also be derived:

$$T_k = (1 - \alpha_k) T_{local} + T_{i,k,n}^{edge} + t_{i,j,k,n}^{mig} + t_{j,k,n}^{proc}$$

Once the task is predicted using LSTM, these resources or can say tasks should be routed using a routing mechanism in which here we use reinforcement learning method (J. Li et al. (2020); S. Wang et al. (2018); J. Chen et al. (2019); X. Fu et al. (2019)) for allocation. In general terms, reinforcement learning is the challenge of learning, in a dynamic environment, to attain an objective through interaction. The learning entity that takes measures is termed an agent. As demonstrated in Figure 4, the agent continuously interacts with the environment through actions and rewards. The objective of the agent is to test alternative sequences of action so that the reward earned is maximized over time. A key part of reinforcement learning algorithms is the ability to learn from delayed rewards. An agent must carry out a certain set of activities in certain situations before receiving a reward. Agent must overcome the issue of the temporary credit assignment to learn such a sequence, i.e. an agent must decide which states are accountable for the reward obtained in the action sequence.

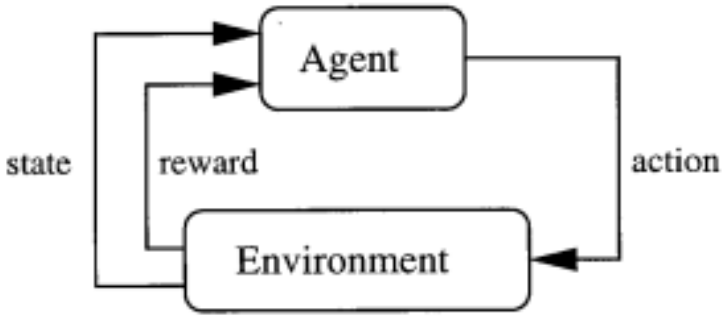

Figure 4 Agent-Environment Interactions

To determine the optimum sequence of activities by combining trial and error methods in a setting that maximizes the reward gained over time. Because they are not developed on input and output pairs to define the best action at each stage, reinforcement learning algorithms vary from supervised learning algorithms. Instead, the benefits obtained directly them to the objective. This means that the reward obtained following each step sets out the problem to be resolved completely. Q learning method is used to solve the scenario. Once Q table is initialized actions are selected and performed and is updated based on rewards to determine and learn best route towards the destination.

IEEOA Algorithm

Input: $Node_{min}$, $Node_{max}$, Data size, Decision variable.
Output: Routing Decision and routing of tasks.
1: Initialize the Q-table, state space, action space and Qvalues.
2: for Mobile devices ranging from n = $Node_{min}$ ... $Node_{max}$ do
3: Observe current Network utilization, Bandwidth, Tasksize
4: for each edge sever n = 1...q+1 do
5: find Minimum loaded edge server, $Edge_{min}$ and create an authentication mechanism.
6: end for
7: Identify the key indicators to build current state St
8: Store all running states to fill state space S where St ∈ S
9: if offloading required then agent takes new route that has not been selected before to obtain higher reward;
10: else agent takes the best route that has already observed so far.
11: After take action at, receive the reward R and update Q-values:
13: if destination is not available, then create acknowledgment and update the information.
14: end if
15: if destination is available then transfer the task to edge server and execute task.

Simulation Results

For this model to be implemented, the hardware specification such as Windows 10 OS, NVIDIA GeForce GTX 1650 graphic processor, 9th generation i5 Core and 512 SSD. Also, the programming language used for building this model is python under the Google collab platform. The proposed model (IEEOA) is compared with other existing offloading techniques such as (i) With out Offloading (ii) Energy Efficient Deep Learning-based Offloading Technique (EEDOT), and (iii) Comprehensive and Energy effective Deep-learning-based Offloading Technique (CEDOT).

The algorithm's inputs are as follows: the LSTM module's training data set comprises 1000 computational offloading logs for edge cloud nodes, while the test data set has 150 computation offloading logs. There are four hidden layers are available, with 500 iterations. It has a batch size of 50 and a convergence loss of 0.025. To evaluate the complete process delay task after the deployment of various algorithms, the data size V and the data size ϕ for the subtasks of a computer task must be varied on a linear basis as experimental variables. Two techniques are chosen for comparative studies in order to assess the computation offloading methodology based on task prediction: (1) On mobile devices, the computing work must be performed directly. There is no transmission delay in this mode, and the task's overall duration is mostly due to computing delays. (2) Mobile devices must offload all computing activities to linked edge computing nodes for execution. The overall delay for the job in this manner comprises not only communication delays, but also computation delay, queueing delay, and other factors.

Figure 5 depicts an examination of energy utilization of a UE with a fluctuating data size. The energy utilization of IEEOA is the least since it considers the appropriate best routes alongside the offloading strategy.

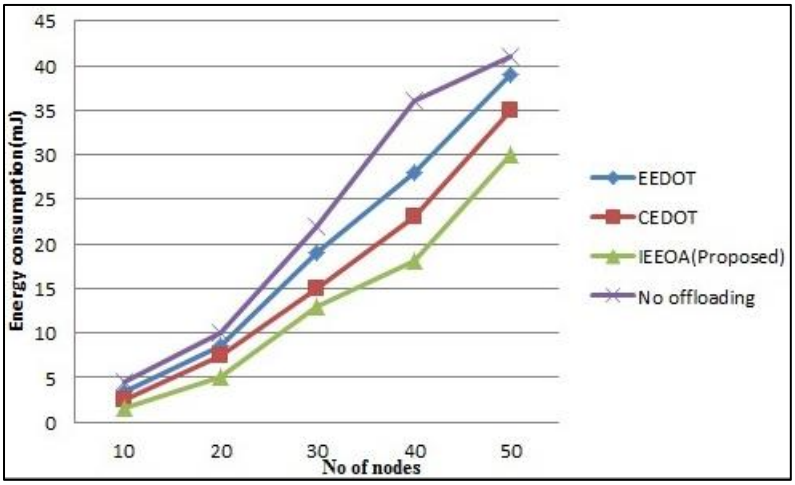

Figure 5 Energy consumption vs. No of nodes

Figure 6 depicts the progression of total task latency as data volume grows in three distinct offloading techniques. It can be observed that, given the current state of our local hardware, the computational capacity of mobile devices is insufficient to perform tasks involving huge amounts of data. As a result, local computing is faster when the data amount is less. Local computation time will rise in a nonlinear fashion as data size grows, which is inconvenient for services that are sensitive to delay. Small data size works against the total delay optimization because of the network transmission latency in edge computing offloading mode. The benefit in the computational capability of edge computing nodes, on the other hand, might be reflected as data size grows. Local computing, edge computing, and subtask migration may all be integrated into our approach when considering the subtask forms of computation tasks. In certain ways, an effective computation offloading technique can be developed for jobs with various data sizes in order to reduce the overall task latency.

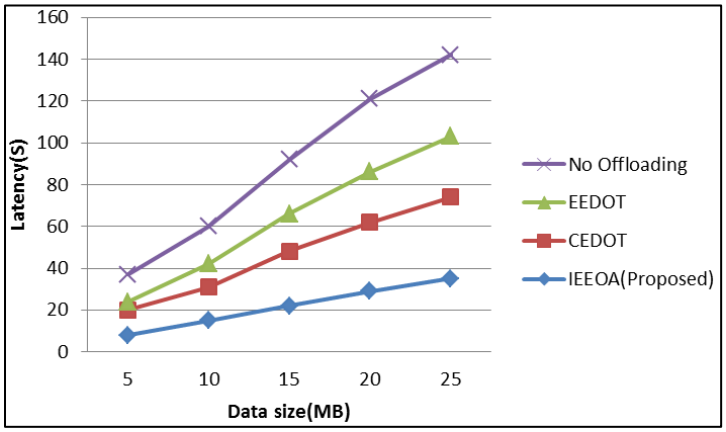

Figure 6 Latency of various methods vs. Data size

Figure 7 presents total cost depending on various data sizes. As our system uses an optimal cost effective strategy which depending on time delay. However, the proposed technique selects the offloading policy with minimum cost for offloading learning optimal routes.

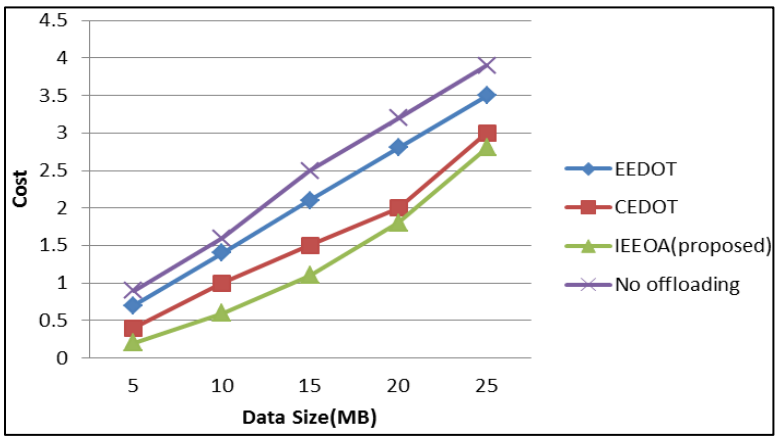

Figure 7 Data size vs total cost

Figure 8 depicts a graphical representation of the delay of various offloading techniques with respect to IEEOA on depending on efficiency as the number of nodes increases, our model took comparatively less time to execute.

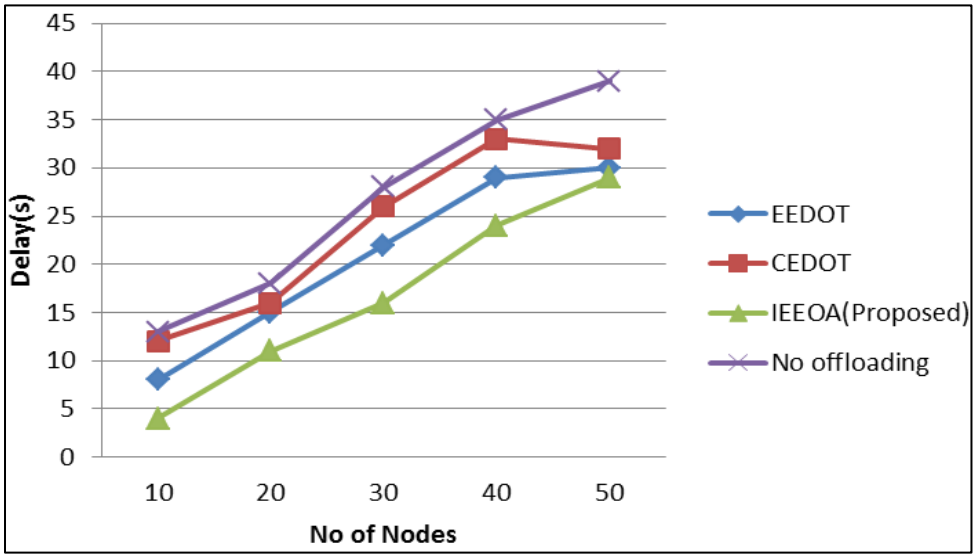

Figure 8 Delay vs. No of nodes

Conclusion

In response to the shortcomings of traditional local computing, cloud computing, and edge computing modes, a novel intelligent computation offloading-based MEC architecture with a combination of deep learning and reinforcement learning approaches is suggested in this paper. The recommended architecture is used to build the compute offloading and task migration technique based on task prediction together with routing methods. From the perspective of the LSTM-based algorithm, the prediction-based computational offloading strategy, and a computational job migration for the edge cloud scheme, the optimal offloading technique for computation is explained. Unlike local computing and a single edge offloading approach, our methodology successfully decreases overall task delay and increases energy efficiency by selecting optimal offloading strategy and allowing high data size offloaded jobs to be routed based on learning aspects which improves allocation of tasks to the edge server in a quick manner. Performance tests are done using the algorithm improves overall energy efficiency and reduces latency of tasks.

Table 3 Abbreviations

MEC	Mobile Edge Computing
UE	User Equipment's
EU	End Users
DL	Deep Learning
AI	Artificial Intelligence
RL	Reinforcement Learning
LSTM	Long Short Term Memory
RNN	Recurrent Neural Network
MES	Mobile Edge Server
EC	Edge Computing
MCC	Mobile Cloud Computing
DRL	Deep Reinforcement Learning
DNN	Deep Neural Network
QoS	Quality of Service
TOT	Total Offloading Technique
EEDOT	Energy Efficient Deep Learning-based Offloading Technique
CEDOT	Comprehensive and Energy effective Deep-learning-based Offloading Technique

References

Dinh, T.Q., Tang, J., La, Q.D., & Quek, T.Q. (2017). Offloading in mobile edge computing: Task allocation and computational frequency scaling. *IEEE Transactions on Communications, 65*(8), 3571-3584.

Agarwal, S., Philipose, M., & Bahl, P. (2014). Vision: The case for cellular small cells for cloudlets. *In Proceedings of the fifth international workshop on Mobile cloud computing & services*, 1-5.

Cao, Y., Jiang, T., & Wang, C. (2014). Optimal radio resource allocation for mobile task offloading in cellular networks. *IEEE Network, 28*(5), 68-73.

Chen, X., Jiao, L., Li, W., & Fu, X. (2015). Efficient multi-user computation offloading for mobile-edge cloud computing. *IEEE/ACM Transactions on Networking, 24*(5), 2795-2808.

Hu, Y.C., Patel, M., Sabella, D., Sprecher, N., & Young, V. (2015). Mobile edge computing—A key technology towards 5G. *ETSI white paper, 11*(11), 1-16.

Chen, M., Hao, Y., Gharavi, H., & Leung, V.C. (2019). Cognitive information measurements: A new perspective. *Information sciences, 505*, 487-497.

Chen, M., Hao, Y., Hu, L., Hossain, M.S., & Ghoneim, A. (2018). Edge-CoCaCo: Toward joint optimization of computation, caching, and communication on edge cloud. *IEEE Wireless Communications, 25*(3), 21-27.

Feng, J., Chen, X., Gao, R., Zeng, M., & Li, Y. (2018). Deeptp: An end-to-end neural network for mobile cellular traffic prediction. *IEEE Network, 32*(6), 108-115.

Orsini, G., Bade, D., & Lamersdorf, W. (2016). Cloudaware: A context-adaptive middleware for mobile edge and cloud computing applications. *In 2016 IEEE 1st International Workshops on Foundations and Applications of Self* Systems (FAS* W)*, 216-221.

Al-Khafajiy, M., Baker, T., Waraich, A., Al-Jumeily, D., & Hussain, A. (2018). IoT-fog optimal workload via fog offloading. *In IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)*, 359-364.

Li, J., Gao, H., Lv, T., & Lu, Y. (2018). Deep reinforcement learning based computation offloading and resource allocation for MEC. *In IEEE Wireless Communications and Networking Conference (WCNC)*, 1-6.

Anas, A., Sharma, M., Abozariba, R., Asaduzzaman, M., Benkhelifa, E., & Patwary, M.N. (2017). Autonomous Workload Balancing in Cloud Federation Environments with Different Access Restrictions. *In IEEE 14th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, 636-641.

Ma, X., Zhou, A., Zhang, S., & Wang, S. (2020). Cooperative service caching and workload scheduling in mobile edge computing. *In IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, 2076-2085.

Sonmez, C., Ozgovde, A., & Ersoy, C. (2019). Fuzzy workload orchestration for edge computing. *IEEE Transactions on Network and Service Management, 16*(2), 769-782.

Santoro, D., Zozin, D., Pizzolli, D., De Pellegrini, F., & Cretti, S. (2017). Foggy: A platform for workload orchestration in a fog computing environment. *In IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, 231-234.

Chen, M., Yang, J., Zhu, X., Wang, X., Liu, M., & Song, J. (2017). Smart home 2.0: Innovative smart home system powered by botanical IoT and emotion detection. *Mobile Networks and Applications*, *22*(6), 1159-1169.

He, Y., Liang, C., Yu, F.R., & Han, Z. (2018). Trust-based social networks with computing, caching and communications: A deep reinforcement learning approach. *IEEE Transactions on Network Science and Engineering, 7*(1), 66-79.

Liu, J., & Zhang, Q. (2019). Code-partitioning offloading schemes in mobile edge computing for augmented reality. *IEEE Access*, *7*, 11222-11236.

Hu, R.Q. (2018). Mobility-aware edge caching and computing in vehicle networks: A deep reinforcement learning. *IEEE Transactions on Vehicular Technology, 67*(11), 10190-10203.

Huang, L., Feng, X., Qian, L., & Wu, Y. (2018). Deep reinforcement learning-based task offloading and resource allocation for mobile edge computing. *In International Conference on Machine Learning and Intelligent Communications*, 33-42.

Zhang, F., Ge, J., Wong, C., Li, C., Chen, X., Zhang, S., & Chang, V. (2019). Online learning offloading framework for heterogeneous mobile edge computing system. *Journal of Parallel and Distributed Computing, 128*, 167-183.

Yu, S., Wang, X., & Langar, R. (2017). Computation offloading for mobile edge computing: A deep learning approach. *In IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, 1-6.

Dai, P., Liu, K., Wu, X., Xing, H., Yu, Z., & Lee, V.C. (2019). A learning algorithm for real-time service in vehicular networks with mobile-edge computing. *In ICC 2019-2019 IEEE International Conference on Communications (ICC)*, 1-6.

Guevara, J.C., Torres, R.D.S., & da Fonseca, N.L. (2020). On the classification of fog computing applications: A machine learning perspective. *Journal of Network and Computer Applications, 159*.

Ale, L., Zhang, N., Wu, H., Chen, D., & Han, T. (2019). Online proactive caching in mobile edge computing using bidirectional deep recurrent neural network. *IEEE Internet of Things Journal, 6*(3), 5520-5530.

Al-Khafajiy, M., Baker, T., Waraich, A., Al-Jumeily, D., & Hussain, A. (2018). IoT-fog optimal workload via fog offloading. *In IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)*, 359-364.

Ning, Z., Dong, P., Wang, X., Guo, L., Rodrigues, J.J., Kong, X., & Kwok, R.Y. (2019). Deep reinforcement learning for intelligent internet of vehicles: An energy-efficient computational

offloading scheme. *IEEE Transactions on Cognitive Communications and Networking*, *5*(4), 1060-1072.

Li, J., Zhang, X., Zhang, J., Wu, J., Sun, Q., & Xie, Y. (2019). Deep reinforcement learning-based mobility-aware robust proactive resource allocation in heterogeneous networks. *IEEE Transactions on Cognitive Communications and Networking*, *6*(1), 408-421.

Wang, S., Liu, H., Gomes, P. H., & Krishnamachari, B. (2018). Deep reinforcement learning for dynamic multichannel access in wireless networks. *IEEE Transactions on Cognitive Communications and Networking*, *4*(2), 257-265.

Chen, J., Chen, S., Wang, Q., Cao, B., Feng, G., & Hu, J. (2019). IRAF: A deep reinforcement learning approach for collaborative mobile edge computing IoT networks. *IEEE Internet of Things Journal*, *6*(4), 7011-7024.

Ali, Z., Jiao, L., Baker, T., Abbas, G., Abbas, Z. H., & Khaf, S. (2019). A deep learning approach for energy efficient computational offloading in mobile edge computing. *IEEE Access*, *7*, 149623-149633. http://doi.org/10.1109/ACCESS.2019.2947053

Fu, X., Fortino, G., Li, W., Pace, P., & Yang, Y. (2019). WSNs-assisted opportunistic network for low-latency message forwarding in sparse settings. Future generation computer systems, 91, 223-237.

Miao, Y., Wu, G., Li, M., Ghoneim, A., Al-Rakhami, M., & Hossain, M.S. (2020). Intelligent task prediction and computation offloading based on mobile-edge cloud computing. *Future Generation Computer Systems*, *102*, 925-931.

Peng, K., Leung, V., Xu, X., Zheng, L., Wang, J., & Huang, Q. (2018). A survey on mobile edge computing: Focusing on service adoption and provision. *Wireless Communications and Mobile Computing*, 2018. https://doi.org/10.1155/2018/8267838

Meteab, A.A., Sadiq, A.S., & Hadrawi, H.K. (2020). Effect of Continuous Improvement of Information Technology Applications on E-Costumer Behavior in Social Media. *Webology*, *17*(1), 19-29.

Authors Profile

S. Anoop is currently doing his research in Department of Computer Science and Engineering at Noorul Islam Centre for Higher Education in the area of Mobile cloud computing and network security.

Dr.J. Amar Pratap Singh is Professor in Department of Computer Science and Engineering at Noorul Islam Centre for Higher Education. He received his Ph.D. from Anna University in 2013. His main research interests include Cloud computing, Data Mining, Mobile Computing, Wireless Networks, Image Processing and Software Engineering.