Hindawi
Security and Communication Networks
Volume 2024, Article ID 9762430, 1 page
https://doi.org/10.1155/2024/9762430

Retraction

Retracted: Secure and Energy-Efficient Computational Offloading Using LSTM in Mobile Edge Computing

Security and Communication Networks

Received 8 January 2024; Accepted 8 January 2024; Published 9 January 2024

This article has been retracted by Hindawi following an investigation undertaken by the publisher [1]. This investigation has uncovered evidence of one or more of the following indicators of systematic manipulation of the publication process:

(1) Discrepancies in scope

(2) Discrepancies in the description of the research reported

(3) Discrepancies between the availability of data and the research described

(4) Inappropriate citations

(5) Incoherent, meaningless and/or irrelevant content included in the article

(6) Manipulated or compromised peer review

The presence of these indicators undermines our confidence in the integrity of the article's content and we cannot, therefore, vouch for its reliability. Please note that this notice is intended solely to alert readers that the content of this article is unreliable. We have not investigated whether authors were aware of or involved in the systematic manipulation of the publication process.

Wiley and Hindawi regrets that the usual quality checks did not identify these issues before publication and have since put additional measures in place to safeguard research integrity.

We wish to credit our own Research Integrity and Research Publishing teams and anonymous and named external researchers and research integrity experts for contributing to this investigation.

The corresponding author, as the representative of all authors, has been given the opportunity to register their agreement or disagreement to this retraction. We have kept a record of any response received.

References

[1] M. Arif, F. Ajesh, S. Shamsudheen, and M. Shahzad, "Secure and Energy-Efficient Computational Offloading Using LSTM in Mobile Edge Computing," *Security and Communication Networks*, vol. 2022, Article ID 4937588, 13 pages, 2022.

Hindawi
Security and Communication Networks
Volume 2022, Article ID 4937588, 13 pages
https://doi.org/10.1155/2022/4937588

Research Article

Secure and Energy-Efficient Computational Offloading Using LSTM in Mobile Edge Computing

Muhammad Arif,[1] **F. Ajesh**,[2] **Shermin Shamsudheen**,[3] **and Muhammad Shahzad**[1]

[1]*Department of Computer Science and Information Technology, University of Lahore, Lahore, Pakistan*
[2]*Department of Computer Science and Engineering, Sree Buddha College of Engineering, Alappuzha, Kerala, India*
[3]*Department of Computer Science, College of Computer Science and Information Technology, Jazan University, Jizan, Saudi Arabia*

Correspondence should be addressed to Muhammad Arif; arifmuhammad36@hotmail.com

Received 9 November 2021; Revised 24 November 2021; Accepted 26 November 2021; Published 7 January 2022

Academic Editor: Mamoun Alazab

The use of application media, gamming, entertainment, and healthcare engineering has expanded as a result of the rapid growth of mobile technologies. This technology overcomes the traditional computing methods in terms of communication delay and energy consumption, thereby providing high reliability and bandwidth for devices. In today's world, mobile edge computing is improving in various forms so as to provide better output and there is no room for simple computing architecture for MEC. So, this paper proposed a secure and energy-efficient computational offloading scheme using LSTM. The prediction of the computational tasks is done using the LSTM algorithm, the strategy for computation offloading of mobile devices is based on the prediction of tasks, and the migration of tasks for the scheme of edge cloud scheduling helps to optimize the edge computing offloading model. Experiments show that our proposed architecture, which consists of an LSTM-based offloading technique and routing (LSTMOTR) algorithm, can efficiently decrease total task delay with growing data and subtasks, reduce energy consumption, and bring much security to the devices due to the firewall nature of LSTM.

1. Introduction

Smart mobile systems have become widely utilized in everyday life over recent years which include smartphones, tablet computers, wearable devices [1], 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 [2] have advanced considerably. However, the local computer technologies are only able to operate in few simple computing tasks such as, poor computing, device storage, and limited battery storage in hardware design. The uplines of the cloud can be used to complete computer-intense and data-intensive tasks [3]. This implies that 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. Figure 1 shows traditional MEC architecture [4–7].

In addition, a high amount of network infrastructure resources in multiuser 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 [3]. 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 [8, 9]. ETSI MEC ISG (Industry Specification Group) is a revolutionary computing method composed of six

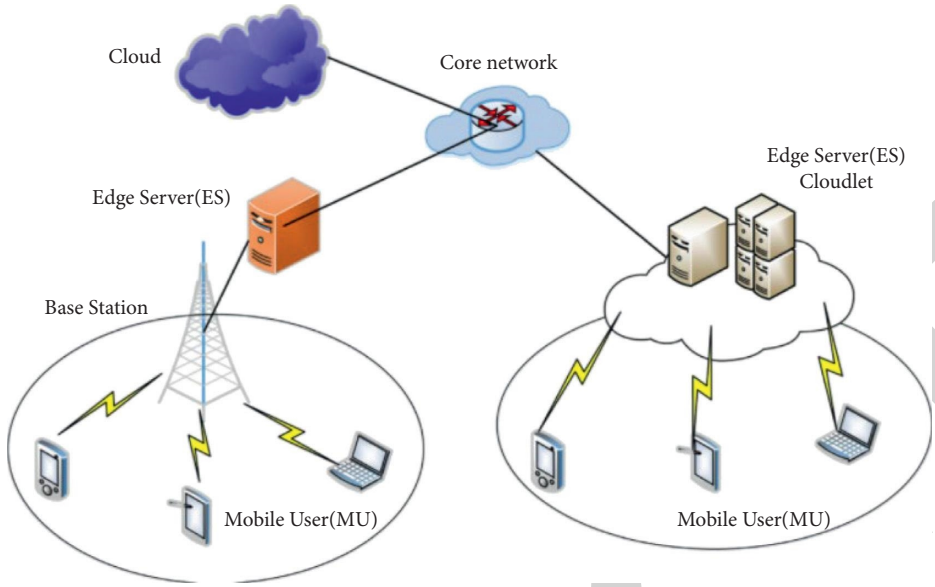

Figure 1: Mobile edge computing architecture.

members which include Nokia and Huawei [10]. 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, and any other storage and computational capability-enabled devices.

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 [11]. (1) 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. Also, all these complex calculation jobs typically may be split into several parallel processing subtasks. (2) 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 [12–17, 26]. 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. (3) The edge cloud's computing capability is diverse and limited. So, when work is transferred onto the edge cloud, the computational complexity may 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 [18], it is important to address completely the issue of computational migration.

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 nonstructured data gathered from mobile devices to be processed and extracted in depth (especially if a significant quantity of historical data is acquired), thus providing MEC's system with smarter cognitive services [19]. Advanced computing employs algorithms like RNN [20], GRU, and GRU deep learning algorithms 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.

1.1. 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. To resolve the problem of the high data transfer rates and the absence of the predefined information, associated offloading metrics and advanced

machine learning methods, such as reinforcement learning, are used. The 7 main challenges are listed below:

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

1.2. Objectives of LSTMOTR. This paper focuses on secure and energy-efficient offloading MEC using a deep learning method in which the following keynotes are mentioned:

(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 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) These predicted tasks are routed using reinforcement learning.
(e) Finally, this is compared with existing systems to analyze how much our proposed system (LSTMOTR) enhances the flavor of security and energy consumption.

This paper is organized as follows. Section 2 analyzes and investigates similar research on the offloading and scheduling of computations and highlights their limitations. Section 3 then provides a clever computer-based offloading MEC architecture. Section 4 proposes the LSTM-based computational prediction method and the computing offloading strategy for a mobile device to migrate computing tasks into edge cloud as well as their scheduling. Section 5 shows task routing through reinforcement learning. Section 6 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. Finally, conclusions are drawn in Section 6. Abbreviations are given in Table 1.

2. Related Works

Orsini et al. [21] highlighted that partial offloading involves estimates of the cost of computation of each component for the application, thus 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, is reduced. We examine partially 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. [22] enables the fog node collaboration for big data processing using predefined 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. [23] proposed a deep reinforcement learning strategy to strengthen the entire offloading system. Nevertheless, global minima may not be ensured in reinforcement learning techniques because of their 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. [24] took computational utilization and access probability into consideration and developed a performance model based on queuing theory to address the workload balancing between service providers within a federated cloud environment.

Ma et al. [25] examined the collaboration between edge nodes and studied workload scheduling to reduce the traffic and response time in mobile edge computing. They offered 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. [26] adopted 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 was proposed by Santoro et al. [27]. It plans to do activities on the basis of computing, storage, or network resources [28].

Previous research has highlighted a number of research gaps that can be addressed. Several articles proposed new offloading [32] frameworks between user terminals and the cloud, laying the groundwork for future MEC architectural research. Their study, however, has a restriction in that their primary focus was on the design of the system's functional aspects, and they did not offer techniques to optimize offloading under varied operating situations. There is a complex relationship between computation task offloading and caching in actual MEC architecture, which leads to caching issues. The transfer of offloaded applications to the cloud and back, as well as the time wasted computing at the cloud, adds

Table 1: Abbreviations.

MEC	Mobile edge computing
UE	User equipment
EU	End user
DL	Deep learning
AI	Artificial intelligence
RL	Reinforcement learning
LSTM	Long short-term memory
RNN	Recurrent neural network
SL	Supervised learning
MES	Mobile edge server
FC	Fog computing
EC	Edge computing
MCC	Mobile cloud computing
ETSI	European Telecommunications Standards Institute
UAV	Unmanned aerial vehicle
ITS	Intelligent transportation system
VANETs	Vehicular ad hoc networks
DRL	Deep reinforcement learning
DNN	Deep neural network
QoS	Quality of service
MDP	Markov decision process
TOT	Total offloading technique
ROT	Random offloading technique
EEDOT	Energy-efficient deep learning-based offloading technique
CEDOT	Comprehensive and energy effective deep learning-based offloading technique
LSTMOT	Long short-term memory offloading technique
WT	Waiting time

up to a considerable execution delay with MEC. Offloading is inconvenient and unsuitable for real-time applications because of this latency. A new evolving concept known as LSTM [33] has been developed to deal with the delay problem. To deal with the challenges of huge data exchange, power consumption, and unacceptable latency in computational offloading in the cloud computation paradigm, LSTM on intelligent computing offloading was developed. Summary of the related work is presented in Table 2.

3. 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, where the infrastructure is on the left and the logic is on the right.

It is possible to divide the infrastructure into three. (1) This includes mobiles, smartphones, and tablet computers in daily life, as well as self-driving cars, wearables, and robotic devices [4, 34–36]. These gadgets may provide many different uses and services, not just using more sophisticated hardware but also using the background system enabled with DL algorithms. Local 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. However, it will cause the access network to be overloaded by channel, through massive data interactions which are not effective in delivering intelligent, latency-sensitive services. The intermediary edge cloud layer is thus created in the cloud architecture between MEC architecture and local devices used for processing and communications. (2) This section comprises edge servers, which are also known as edge nodes, like the base station, wireless access point, and routers. These nodes can connect with local mobile devices through wireless media and share some tasks with limited computing resources of users and send difficult computing tasks to faraway clouds for additional computing via the pull links. (3) Cloud servers are able to deliver DL services featuring powerful processing as well as storage resources. The integration of cloud technology and DL is considered a key component of cognitive computing. This can compensate for the poor intelligence of edge cloud computing, take harder computational works, send back results to edge nodes, and finally provide them to mobile users through a wireless network.

The logic is also separated into three components, which are equivalent to infrastructure. (1) Mobile users only can do a few basic computer activities locally, owing to restricted computing and storage capacity of local devices. Edge cloud would offload more difficult tasks over the wireless channel. (2) 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, and other variables. (3) 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.

These three parts deal with the infrastructure terminal layer, network layer, and access layer.

The purpose and benefits of developing a smart MEC architecture computational offloading are as follows. (1) It is appropriate for diverse computing jobs and for heterogeneous data applications. Prediction is a vital stage in the offloading of computers for various work. If computational activities can be forecast in advance for the type, size, and computing resources, a crucial benchmark to optimize the offloading can be provided. (2) It can be adapted to network communications and computer resources which are changing dynamically. The optimal offloading approach can help enhance QoE from a variety of aspects, including computation latency and complexity because of dynamically changing network resources and computing tasks. Optimized transfer of tasks can help minimize network access congestion. (3) Increases in processing and storage capacity may be increased by local devices as well as edge computing nodes. 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. (4) As LSTM is utilized as a DL technique, it basically works as a firewall to protect the security of every device connected. (5) We can easily carry out multiple activities with the lowest energy consumption with appropriate computational offloading.

3.1. 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 is not smart enough; therefore, 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 device 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 introduced to support planning across edge clouds with a view to further reduce computer delay based on an improved computation offloading technique. The process flow is as illustrated in Figure 3 for the whole method.

3.2. Computation Task 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 [37]. Set $V_k \in \{V_1, V_2, V_K\}$, the data size. W_f, W_C, b_f, and b_C, are utilized to describe the biases and weights of forget and input gates, and σ and tanh are employed as activation functions in multilevel LSTM architecture. Forget gate can be specified as

$$fk = \sigma(Wf.[hk-1, Vk] + bf). \tag{1}$$

The input gate is defined as

$$C_k = f_k * C_{k-1} + \sigma(W_i \cdot [h_{k-1}, V_k] + bi) * \tanh(W_C \cdot [h_{k-1}, V_k] + b_C). \tag{2}$$

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

3.3. 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 one of three ways: locally, partially locally while the remaining is performed at the cloud edge, or offloading to the cloud edge. As a result, the computation delay is T local k when a mobile user chooses to run a task locally, such as task k. The number of bits of the computation job k is being offloaded is represented by the task offloading variable k [0, 1]. When k is 0, the job should be handled locally while k is 1, and "as per traffic data V for computer jobs anticipated in Section 4.1" the job has to be handled on the edge cloud.

If $\alpha k \in (0, 1)$, $\alpha_k \widetilde{V}_k$ should be sent to the edge cloud for processing, whereas $(1 - \alpha_k \widetilde{V}_k)$ 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 result, the time $t_{i,k}^{\text{proc}}$ that node i needs to process k may be calculated as follows:

$$t_{i,k}^{proc} = \frac{\alpha_k \widetilde{V}_k C_{i,k}}{F_{i,k}}. \tag{3}$$

The uplink wireless channel is used for mobile device offloading. As a result, the maximum uplink transmission rate $r_{i,k}$ [20] for task offloading is expressed using Shannon's theorem:

$$r_{i,k} = B \log_2\left(1 + \frac{p_k h^2}{\sigma^2 + w_{i,k}}\right), \tag{4}$$

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.

Table 2: A survey of the related methods with remarks.

SLNo.	Author	Title	Method	Remarks
1	Orsini et al. [21]	CloudAware: a context-adaptive middleware for mobile edge and cloud computing applications	CloudAware as a holistic approach	Computation offloading
2	Al-Khafajiy et al. [22]	IoT-fog optimal workload via fog offloading	Collaborative edge offloading technique	Misses the fog nodes Not energy efficient
3	Li et al. [23]	Deep reinforcement learning-based computation offloading and resource allocation for MEC	Deep reinforcement learning strategy	Achieves significant reduction on the sum cost
4	Anas et al. [24]	Autonomous workload balancing in cloud federation environments with different access restrictions	Performance model based on queuing theory	Provides a solution for access probability and resource utilization at a given time
5	Ma et al. [25]	Cooperative service caching and workload scheduling in mobile edge computing	Fuzzy logic method: developed an iterative algorithm named ICE	Solves the edge computing workload scheduling problem
6	Sonmez et al. [26]	Fuzzy workload orchestration for edge computing	Fuzzy logic method	Low complexity and efficiency in handling uncertain nonlinear systems
7	Santoro et al. [27]	Foggy: a platform for workload orchestration in a fog computing environment	Foggy, an architectural framework and software platform based on open source technologies	Supports IoT operations for multitier distributed, heterogeneous, and decentralized cloud computing systems
8	Prabadevi et al. [29]	Toward blockchain for edge-of-things: a new paradigm, opportunities, and future directions	Blockchain-enabled EoT (BEoT)	Enables future low-latency and high-security services and applications
9	Feng et al. [30]	Attribute-based encryption with parallel outsourced decryption for edge intelligent IoV	Attribute-based encryption model with parallel outsourced decryption (ABEM-POD)	Improves the speed of outsourced decryption in edge intelligent IoV
10	Nguyen et al. [31]	Integration of blockchain and cloud of things: architecture, applications and challenges	Blockchain and cloud of things integration, called BCoT	BCoT increases the efficiency of blockchain technologies

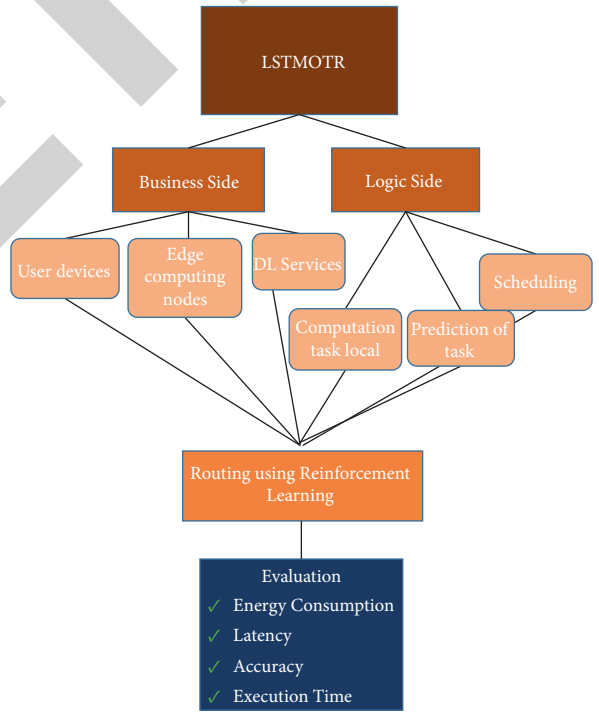

Figure 2: LSTMOTR workflow.

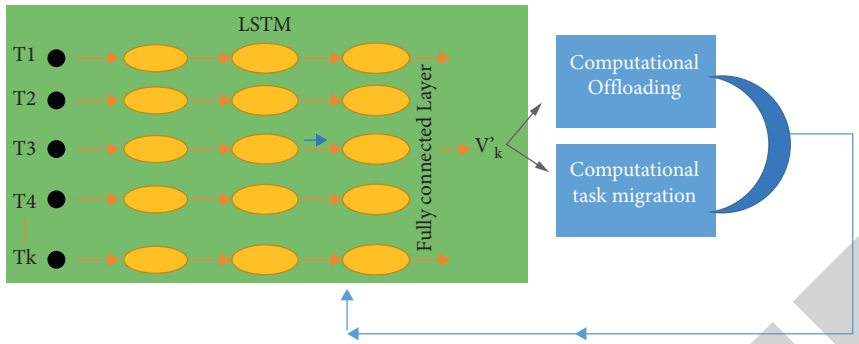

Figure 3: Process flow of computational offloading using LSTM.

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

$$t_{i,k}^{up} = \frac{\alpha_k \widetilde{V}_k}{r_{i,k}}. \tag{5}$$

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

$$T_k = t_{i,k}^{up} + t_{i,k}^{proc} + t_{i,k}, \tag{6}$$

where ti, k means computing delay. The resulting 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 can so be simplified as

$$T_k = t_{i,k}^{up} + t_{i,k}^{proc} t_{i,k} = \frac{\alpha_k \widetilde{V}_k}{r_{i,k}} + \frac{\alpha_k \widetilde{V}_k C_{i,k}}{F_{i,k}}. \tag{7}$$

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

$$T_k = T_k^{e\ dg\ e} + (1 - \alpha_k) T_{local} = \frac{\alpha_k \widetilde{V}_k}{r_{i,k}} + \frac{\alpha_k \widetilde{V}_k C_{i,k}}{F_{i,k}} + (1 - \alpha_k) T_k. \tag{8}$$

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

$$\begin{aligned} &\left(T_k | \widetilde{V}_k, G_k, Q_{i,k}\right), \\ &\alpha k, \\ &Tk < Tk. \end{aligned} \tag{9}$$

3.4. 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 calculate the migration task across clouds is necessary at this moment. Task K to N has subtaks $k = \{k1, k2, \ldots, Kn\}$. The data size may then be stated for all the subtasks k as follows: $\{\phi k1, \phi k2, \ldots, \phi kN\}$.

Subtasks 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, and the migration delay for $n + 1$ to N to j node may be stated as follows:

$$ti, j = \sum_{n}^{N} \Psi ke. \tag{10}$$

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

$$t j, k^{proc} = \sum_{n}^{N} \phi keC j, k. \tag{11}$$

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

$$\pi^* (s) = \arg\max_a V^* (s'). \tag{12}$$

The aforementioned derivative approach may also be simplified with a target of lower latency and is represented as follows: minimize $(T_k | V_k, G_k, Q_{i,j})$ subject to $\alpha k, \psi kn$ $Tk < Tk$.

4. Routing Using Reinforcement Learning

Once the task is predicted and offloaded using LSTM, these resources or tasks should be routed using a routing mechanism in which we use reinforcement learning method 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. The agent must overcome the issue of the temporary credit assignment to

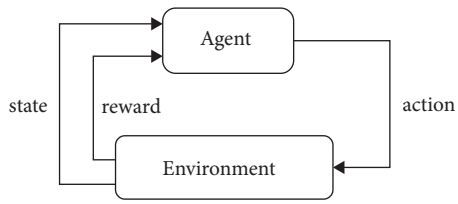

Figure 4: Agent-environment interactions.

learn such a sequence, i.e., an agent must decide which states are accountable for the reward obtained in the action sequence.

To determine the optimum sequence of activities, the trial and error method in a setting is used to maximize 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 direct them to the objective. This means that the reward obtained following each step sets out the problem to be resolved completely. A further distinction in supervised learning is that typically a task does not have discrete phases of training and testing. On the contrary, certain tasks need continuous lifelong learning.

4.1. Value Functions. A Markov decision process (MDP) [38, 39] may be used to model the reinforcement learning problem that an agent encounters. A finite Markov decision process is defined as follows:

(a) A finite set of tasks *S*.

(b) A finite set of actions *A*.

(c) A reward function: *R*: S *x* A *R*.

(d) A task transition function: *T*: *S x A x S R*, where *T* (*s*, *a*, *S*′) is the probability of advancing from task *s* to *s*′ when taking action a.

When the probability of transition *T* is independent of prior states, the model is called Markov. As a result, the transition function *T*, as well as the current state and action, is sufficient to probabilistically describe the future job. The model is a nondeterministic MDP since the actions are chosen probabilistically. At each time step *t*, an agent observes the state St and takes action. The returning reward $r_{t+1} = R\ (st, at)$, and next task S_{t+1} with probability $T\ (s_t$, at, $S_{t+1})'$ is a response to the environment. This procedure is continuously performed until the agent reaches its goal or for nonepisodic activities indefinitely.

The policy $\pi\ (s, a)$ of an agent is to map every task S and take action of every task *s*. An agent's objective is to enhance its policy, increasing the compounded reward that the agent receives over time. This is also termed the anticipated return.

Depending on the specific job that the agent must do, the anticipated return R_t can be computed in a variety of ways. Some tasks may be broken down into episodes or trials, each of which has a different outcome. At the end of each episode, the agent is reset to its initial state. We calculate the anticipated return in these episodic activities by accumulating total incentives received over a certain time horizon *h*:

$$Rt = \sum_{k=0}^{h} rt + k + 1. \tag{13}$$

Certain tasks never finish; therefore, the aforementioned total might be indefinite. This issue can be resolved by reducing future rewards:

$$Rt = \sum_{k=0}^{\infty} \gamma rt + k + 1, \tag{14}$$

where γ is the discount rate and is 0; $S \leq \mu < 1$ In this analysis, we will concentrate only on this situation, called the discounted infinite horizon case. With the addition of an absorbing state entered shortly after the terminal state, this definition of the expected return may be used for episodic activities as well. The null reward is the only reward for the transition from the absorbing state to itself.

To assess the task's effectiveness, most reinforcement learning approaches rely on estimating value functions. The task's value or utility is the potential benefit, or yield, that an agent can get in the future. Because an agent's behaviors affect future rewards, the value function is defined by the policy the agent follows. The value $V^{\pi}\ (s)$ of a task *s* under policy 7^{π} is the potential return from state's resources if policy 7 is followed:

$$V^{\pi}(s) = E_{\pi}\{R_t \mid s_t = s\}. \tag{15}$$

When policy 7^{π} is followed, where E_{π} {} indicates the expected reward, we have the following for the discounted infinite horizon case:

$$V^{*}(s) = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^{k} rt + k + 1 | st\right\} = s. \tag{16}$$

By maximizing V^{π} for every task, the optimum value function V^{*} is achieved:

$$V^{*}(s) = \max_{\pi} V^{\pi}(\forall s). \tag{17}$$

The best policy is the one that corresponds to the best value function in the maximization as shown in the above expression:

$$\pi^{*} = \arg\max_{\pi} V^{\pi}(\forall s). \tag{18}$$

We may utilize the dynamic programming method known as value iteration to identify the optimum value function because we have the environment dynamics model *T* and the reward function *R* in an MDP. We can use value iteration in the MDP. We achieve the optimum policy π^{*} by picking the action that results in the maximum value function of all immediate succeeding tasks in each state after we know the ideal value function:

$$\pi^{*}(s) = \arg\max_{a} V^{*}(s'), \tag{19}$$

where s' is the next task of *s*. In the context of learning issues with reinforcement, the agent does not typically have access to environmental dynamics in the form of transitional

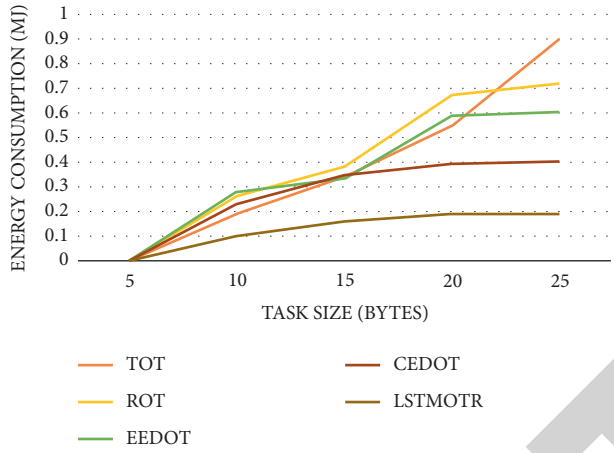

Figure 5: Energy consumption of various methods vs. LSTMOTR.

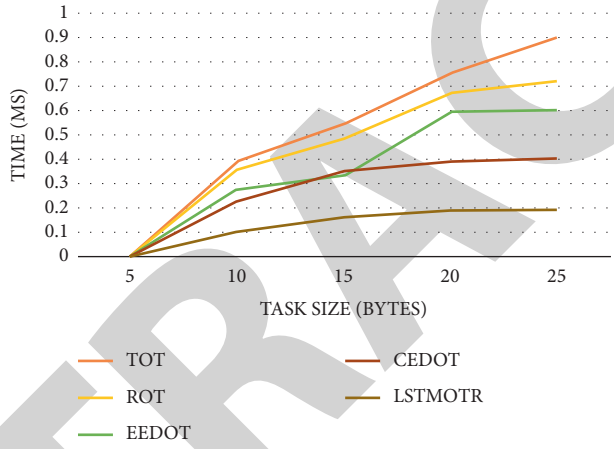

Figure 6: Latency of various methods vs. LSTMOTR.

probabilities (T). In the following sections, we look into reinforcement learning methods based on dynamic programming in situations when we do not have access to a dynamic environment. Instead, an agent must learn from the environment via the rewards that various actions provide.

5. Simulation Results

For this model to be implemented, the hardware specification is as follows: 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 Colab Platform. The proposed model (LSTMOTR) is compared with other existing offloading techniques such as the (i) total offloading technique (TOT), (ii) random offloading technique (ROT), (iii) energy-efficient deep learning-based offloading technique (EEDOT), and (iv) comprehensive and energy effective deep learning-based offloading technique (CEDOT).

The algorithm's inputs are as follows: the LSTM module's training dataset comprises 1,500 computational offloading logs for edge cloud nodes, while the test dataset has 250 computation offloading logs. There are four hidden layers which are available, with 1000 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, queuing delay, and other factors.

Figure 5 depicts an examination of energy utilization of UE with a fluctuating errand size. The energy utilization of LSTMOT is the least since it considers the appropriate part size alongside the offloading strategy.

Figure 6 depicts the progression of total task latency as data volume grows in three distinct offloading techniques. The data volume of job V is divided into 40, 50, ..., 120. It can be

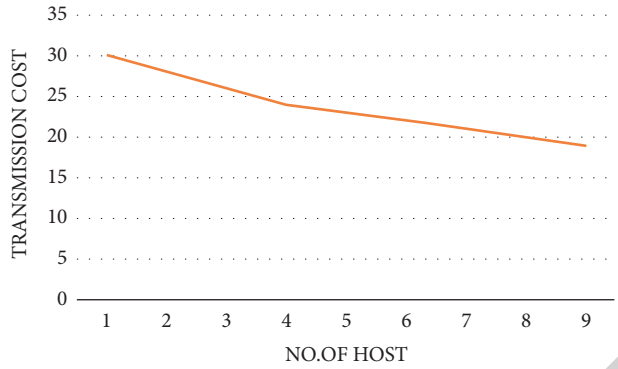

Figure 7: No. of hosts vs. total transmission cost of LSTMOTR.

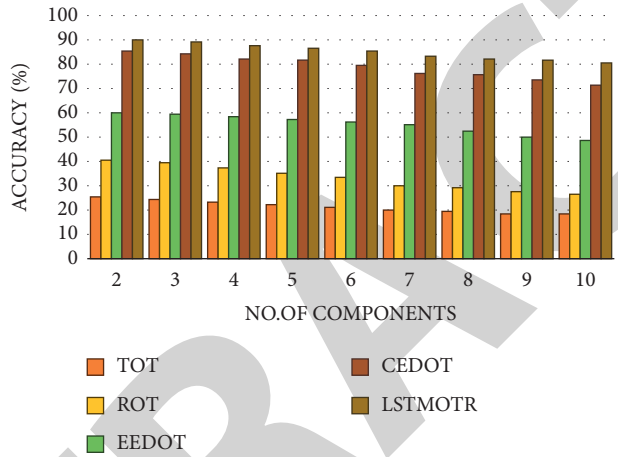

Figure 8: Accuracy of various methods vs. LSTMOTR.

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 7 presents the size of the training dataset alongside an alternate number of parts per task. As the quantity of segments per task builds, the intricacy of the choice limits increments. So, the appropriate size of datasets for training needs to be used for achieving more than 80%.

Figure 8 also shows the effect of the number of components per task on exactness. As the number of segments per job grows, so does the variety of offloading arrangements that may be used. As a result, the likelihood of selecting an offloading approach decreases, as the presentation of ROT and TOT diminishes. While the complexity of the relationship between consistent data and yield data grows for other DL-based methods, the precision of CEDOT, EEDOT, and DOT decreases as the number of segments per task grows. In any event, the LSTMOT exhibits better than a wide range of techniques and is almost comparable to CEDOT, with the added bonus of low energy consumption and computational offloading time delay.

Figure 9 depicts a graphical representation of how much these methods are secure when it comes to MEC in which LSTMOT is much secure due to each component per task achieving security throughout the process and also due to it acting as a firewall between the networks.

Figure 10 depicts a graphical representation of the execution time of various offloading techniques with respect to LSTMOTR in which our model took comparatively less time.

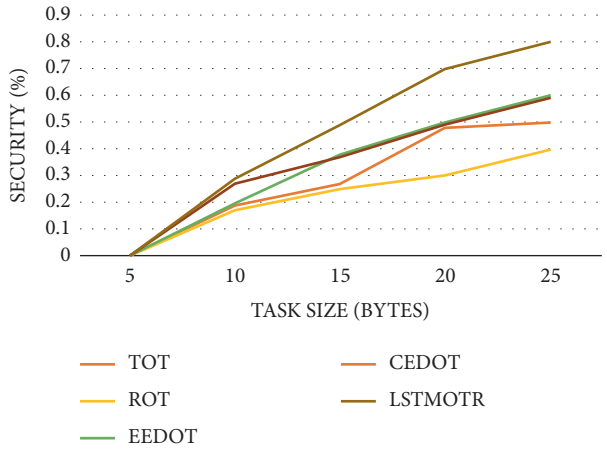

Figure 9: Various offloading schemes vs. security.

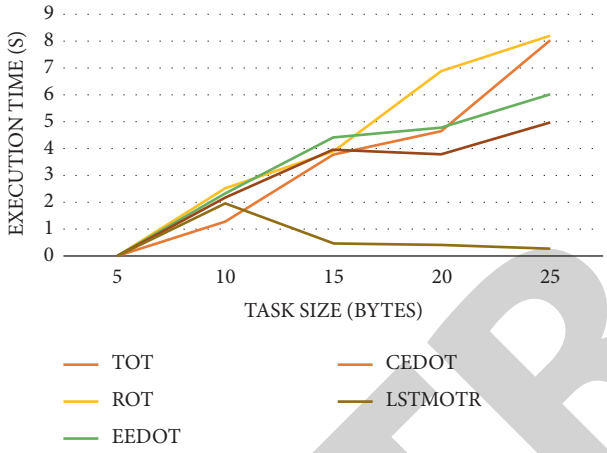

Figure 10: Execution time vs. task size.

6. Conclusion and Future Scope

Edge computing and deep learning have seen tremendous growth and great success in their respective fields in recent years. The massive amount of valuable data generated and collected at the edge, on the other hand, necessitates more intelligent and powerful processing capabilities on the local level in order to fully unleash the underlying potentials of big data and meet the ever-increasing expectations of different applications. Mobile edge computing (MEC) networks have two major challenges: energy efficiency and security. Offloading computing workloads securely and efficiently is difficult due to unpredictable task arrivals, a time-varying dynamic environment, and passive existing adversaries. By 2025, there would be 18 billion IoT devices, each requiring network access. Small-scale personal IoT devices to large-scale design settings, such as smart cities and new industrial applications, can all benefit from mobile edge computing. Mobile edge computing can be used by small devices, such as in-home IoT equipment, to offload computational activities that are too sophisticated for their limited memory capacity. Users streaming videos from their mobile devices can take advantage of cached versions of their specific content from mobile edge computing base stations or videos that are automatically supplied in a quality/bandwidth that their network can handle based on local network conditions.

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 three modes is suggested in this paper. We also go through the newest MEC generation's research aims and advantages. The recommended architecture is used to build the compute offloading and task migration technique based on task prediction. The LSTM-based algorithm, and the prediction-based computational offloading strategy, along with the computational job migration for the edge cloud scheme is well explained. The optimization approach is thoroughly explained. Performance tests are done using the algorithm and architecture that we recommend. Unlike local computing and a single edge offloading approach, our methodology successfully decreases overall task delay by increasing the quantity of calculating data and subtasking, allowing time-delay sensitive jobs to be performed quickly. Once the job had been offloaded, reinforcement learning was used to route it. Finally, LSTM serves as a firewall that protects such user devices.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] M. Chen, J. Zhou, G. Tao, J. Yang, and L. Hu, "Wearable affective robot," *IEEE Access*, vol. 6, pp. 64766–64776, 2018.

[2] M. Arif, G. Wang, M. Zakirul Alam Bhuiyan, T. Wang, and J. Chen, "A survey on security attacks in VANETs: communication, applications and challenges," *Vehicular Communications*, vol. 19, p. 100179, 2019.

[3] S. Agarwal, M. Philipose, and P. Bahl, "Vision: the case for cellular small cells for cloudlets," in *Proceedings of the Fifth International Workshop on Mobile Cloud Computing & Services*, ACM Bretton Woods, USA, June2014.

[4] Y. He, C. Liang, F. R. Yu, and Z. Han, "Trust-based social networks with computing, caching and communications: a deep reinforcement learning approach," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 1, pp. 66–79, 2020.

[5] J. Liu and Q. Zhang, "Code-partitioning offloading schemes in mobile edge computing for augmented reality," *IEEE Access*, vol. 7, pp. 11222–11236, 2019.

[6] L. T. Tan and R. Q. Hu, "Mobility-aware edge caching and computing in vehicle networks: a deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 11, pp. 10190–10203, 2018.

[7] L. Huang, X. Feng, L. Qian, and Y. Wu, "Deep reinforcement learning-based task offloading and resource allocation for mobile edge computing," in *Proceedings of the International*

Conference on Machine Learning and Intelligent Communications, pp. 33–42, Springer, Cham, July 2018.

[8] J. Sachs, *5G Ultra-reliable and Low Latency Communication*, URL:, 2017.

[9] M. Arif, W. Balzano, A. Fontanella, S. Stranieri, G. Wang, and X. Xing, "Integration of 5G, VANETs and blockchain technology," in *Proceedings of the IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pp. 2007–2013, IEEE, Guangzhou, China, January 2021.

[10] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, 2016.

[11] ETSI, *Mobile Edge Computing: A Key Technology towards 5G*, White Paper, ETSI, Sophia Antipolis, France, 2015.

[12] F. Zhang, J. Ge, C. Wong et al., "Online learning offloading framework for heterogeneous mobile edge computing system," *Journal of Parallel and Distributed Computing*, vol. 128, pp. 167–183, 2019.

[13] S. Yu, X. Wang, and R. Langar, "Computation offloading for mobile edge computing: a deep learning approach," in *Proceedings of the 2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pp. 1–6, IEEE, Montreal, QC, Canada, 2017, October.

[14] P. Dai, K. Liu, X. Wu, H. Xing, Z. Yu, and V. C. Lee, "A learning algorithm for real-time service in vehicular networks with mobile-edge computing," in *Proceedings of the ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, Shanghai, China, May 2019.

[15] J. C. Guevara, R. d. S. Torres, and N. L. S. da Fonseca, "On the classification of fog computing applications: a machine learning perspective," *Journal of Network and Computer Applications*, vol. 159, p. 102596, 2020.

[16] L. Ale, N. Zhang, H. Wu, D. Chen, and T. Han, "Online proactive caching in mobile edge computing using bidirectional deep recurrent neural network," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 5520–5530, 2019.

[17] Z. Ning, P. Dong, X. Wang et al., "Deep reinforcement learning for intelligent internet of vehicles: an energy-efficient computational offloading scheme," *IEEE Transactions on Cognitive Communications and Networking*, vol. 5, no. 4, pp. 1060–1072, 2019.

[18] M. Arif, G. Wang, and S. Chen, "Deep learning with non-parametric regression model for traffic flow prediction," in *Proceedings of the 2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech)*, pp. 681–688, IEEE, Athens, Greece, August 2018.

[19] M. Chen, Y. Hao, H. Gharavi, and V. C. M. Leung, "Cognitive information measurements: a new perspective," *Information Sciences*, vol. 505, pp. 487–497, 2019.

[20] M. Chen, Y. Hao, L. Hu, M. S. Hossain, and A. Ghoneim, "Edge-CoCaCo: toward joint optimization of computation, caching, and communication on edge cloud," *IEEE Wireless Communications*, vol. 25, no. 3, pp. 21–27, 2018.

[21] G. Orsini, D. Bade, and W. Lamersdorf, "CloudAware: a context-adaptive middleware for mobile edge and cloud computing applications," in *Proceedings of the IEEE 1st Int. Workshops Found. Appl. Self Syst. (FAS W)*, pp. 216–221, IEEE, Augsburg, Germany, Sep. 2016.

[22] M. Al-Khafajiy, T. Baker, A. Waraich, D. Al-Jumeily, and A. Hussain, "IoT-Fog optimal workload via fog offloading," in *Proceedings of the IEEE/ACM Int. Conf. Utility Cloud Comput. Companion (UCC Companion)*, pp. 359–436, IEEE, Zurich, Switzerl, Dec. 2018.

[23] J. Li, H. Gao, T. Lv, and Y. Lu, "Deep reinforcement learning based computation offloading and resource allocation for mec," in *Proceedings of the IEEEWireless Commun. Netw. Conf. (WCNC)*, pp. 1–6, IEEE, Barcelona, Spain, April 2018.

[24] A. Anas, M. Sharma, R. Abozariba, M. Asaduzzaman, E. Benkhelifa, and M. N. Patwary, "Autonomous workload balancing in cloud federation environments with different access restrictions," in *Proceedings of the 2017 IEEE 14th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, pp. 636–641, IEEE, Orlando, FL, October 2017.

[25] X. Ma, A. Zhou, S. Zhang, and S. Wang, "Cooperative service caching and workload scheduling in mobile edge computing," in *Proceedings of the IEEE INFOCOM 2020 IEEE Conference on Computer Communications*, pp. 2076–2085, IEEE, Toronto, ON, Canada, July 2020.

[26] C. Sonmez, A. Ozgovde, and C. Ersoy, "Fuzzy workload orchestration for edge computing," in *Proceedings of the IEEE Transactions on Network & Service Management*, p. 1, IEEE, Belgrade, Serbia, November 2020.

[27] D. Santoro, D. Zozin, D. Pizzolli, F. De Pellegrini, and S. Cretti, "Foggy: a platform for workload orchestration in a fog computing environment," in *Proceedings of the 2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pp. 231–234, IEEE, Hong Kong, China, December 2017.

[28] M. Chen, J. Yang, X. Zhu et al., "Smart home 2.0: innovative smart home system powered by botanical IoT and emotion detection," *Mobile Networks and Applications*, vol. 22, no. 6, pp. 1159–1169, 2017.

[29] P. Prabadevi, N. Deepa, Q.-V. Pham et al., "Toward blockchain for edge-of-things: a new paradigm, opportunities, and future directions," *IEEE Internet of Things Magazine*, vol. 4, no. 2, pp. 102–108, 2021.

[30] C. Feng, K. Yu, M. Aloqaily, M. Alazab, Z. Lv, and S. Mumtaz, "Attribute-based encryption with parallel outsourced decryption for edge intelligent IoV," *IEEE Transactions on Vehicular Technology*, p. 1, 2020.

[31] D. C. Nguyen, P. N. Pathirana, M. Ding, and A. Seneviratne, "Integration of blockchain and cloud of things: architecture, applications and challenges," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 4, pp. 2521–2549, 2020.

[32] T. Wang, Y. Liang, Y. Zhang et al., "An intelligent dynamic offloading from cloud to edge for smart IoT systems with big data," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 4, pp. 2598–2607, 2020.

[33] H. T. Rauf, J. Gao, A. Almadhor, M. Arif, and M. T. Nafis, "Enhanced bat algorithm for COVID-19 short-term forecasting using optimized LSTM," *Soft Computing*, vol. 25, no. 20, pp. 12989–12999, 2021.

[34] J. Li, X. Zhang, J. Zhang, J. Wu, Q. Sun, and Y. Xie, "Deep reinforcement learning-based mobility-aware robust proactive resource allocation in heterogeneous networks," *IEEE Transactions on Cognitive Communications and Networking*, vol. 6, no. 1, pp. 408–421, 2020.

[35] S. Wang, H. Liu, P. H. Gomes, and B. Krishnamachari, "Deep reinforcement learning for dynamic multichannel access in wireless networks," *IEEE Transactions on Cognitive Communications and Networking*, vol. 4, no. 2, pp. 257–265, 2018.

[36] J. Chen, S. Chen, Q. Wang, B. Cao, G. Feng, and J. Huiraf, "iRAF: a deep reinforcement learning approach for collaborative mobile edge computing IoT networks," *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 7011–7024, Aug 2019.
[37] J. Feng, X. Chen, R. Gao, M. Zeng, and Y. Li, "DeepTP: an end-to-end neural network for mobile cellular traffic prediction," *IEEE Network*, vol. 32, no. 6, pp. 108–115, 2018.
[38] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: a survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.
[39] W. H. Andrag, *Reinforcement Learning for Routing in Communication Networks (Doctoral Dissertation*, Stellenbosch University, Stellenbosch, 2003.