Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH

Open Access

Optimizing energy efficiency in MEC networks: a deep learning approach with Cybertwin-driven resource allocation

Umesh Kumar Lilhore[1], Sarita Simaiya[1,2*], Surjeet Dalal[3], Neetu Faujdar[4], Roobaea Alroobaea[5], Majed Alsafyani[6], Abdullah M. Baqasah[7] and Sultan Algarni[8]

Abstract

Cybertwin (CT) is an innovative network structure that digitally simulates humans and items in a virtual environment, significantly influencing Cybertwin instances more than regular VMs. Cybertwin-driven networks, combined with Mobile Edge Computing (MEC), provide practical options for transmitting IoT-enabled data. This research introduces a hybrid methodology integrating deep learning with Cybertwin-driven resource allocation to enhance energy-efficient workload offloading and resource management in MEC networks. Offloading work is essential in MEC networks since several applications require significant resources. The Cybertwin-driven approach considers user mobility, virtualization, processing power, load migrations, and resource demand as crucial elements in the decision-making process for offloading. The model optimizes job allocation between on-premises and distant execution using a task-offloading strategy to reduce the operating burden on the MEC network. The model uses a hybrid partitioning approach and a cost function to optimize resource allocation efficiently. This cost function accounts for energy consumption and service delays associated with job assignment, execution, and fulfilment. The model calculates the cost of several segmentation and offloading procedures and chooses the lowest cost to enhance energy efficiency and performance. The approach employs a deep learning architecture called "CNN-LSTM-TL" to accomplish energy-efficient task offloading, utilizing pre-trained transfer learning models. Batch normalization is used to speed up model training and improve its robustness. The model is trained and assessed using an extensive mobile edge computing public dataset. The experimental findings confirm the efficacy of the proposed methodology, indicating a 20% decrease in energy usage compared to conventional methods while achieving comparable or superior performance levels. Simulation studies emphasize the advantages of incorporating Cybertwin-driven insights into resource allocation and workload-offloading techniques. This research enhances energy-efficient and resource-aware MEC networks by incorporating Cybertwin-driven techniques.

Keywords Deep learning, Cybertwin, Mobile edge computing, IoT, CNN, LSTM, Transfer learning, Workload offloading

*Correspondence:
Sarita Simaiya
drcse2023@gmail.com
Full list of author information is available at the end of the article

Introduction

The rapid advancement of internet-based technologies has led to the creation of innovative solutions to manage the vast and diverse data generated by ubiquitous intelligent devices, such as mobile edge devices [1]. These connected applications, including virtual reality, augmented reality, vehicle communications, and online media, benefit from the technologies mentioned earlier. Despite being widely distributed across edge networks, cloudlets offer limited resources along with the distant cloud. Real workloads can significantly benefit from faster connections to MEC facilitated by cloud users, reducing the workload on the network's cloud and external cloud servers. Several emerging computational ecosystems have been developed to address fundamental needs like latency, reduced energy consumption, and cost-effectiveness, with MEC being one of the most popular due to the inherent limitations of such technologies [2].

MEC brings computing and storage capabilities to the network's edge, enabling services for information-sharing instances involving multiple wireless endpoints. The number of UEs is rapidly increasing due to the rapid expansion of mobile applications, the IoT, and intelligent systems [3, 4]. Researchers use MEC as a preliminary step for IoT and investigate computing offload techniques for UEs to address the challenge of handling high computational and time-limited workloads. To achieve fine-grained function by offloading work schedules, enabling task parallelization, and improving execution speed, researchers identify opportunities for UEs to offload tasks to edge servers for remote processing. Each piece of equipment in edge communication can independently decide whether to complete an operation on its own or transfer the task to edge servers for remote processing, considering the energy overhead [5].

Figure 1 illustrates the architecture of multi-hop mobile edge communication, consisting of three layers. The first layer, Tier-1, represents the cloud environment, encompassing the cloud server, storage, and processing components. Tier 2 comprises the mobile edge layers containing communication protocols, medium, and connection types. Tier 3, the final layer, mainly includes the connection devices of UEs. MEC provides data storage and computational capabilities

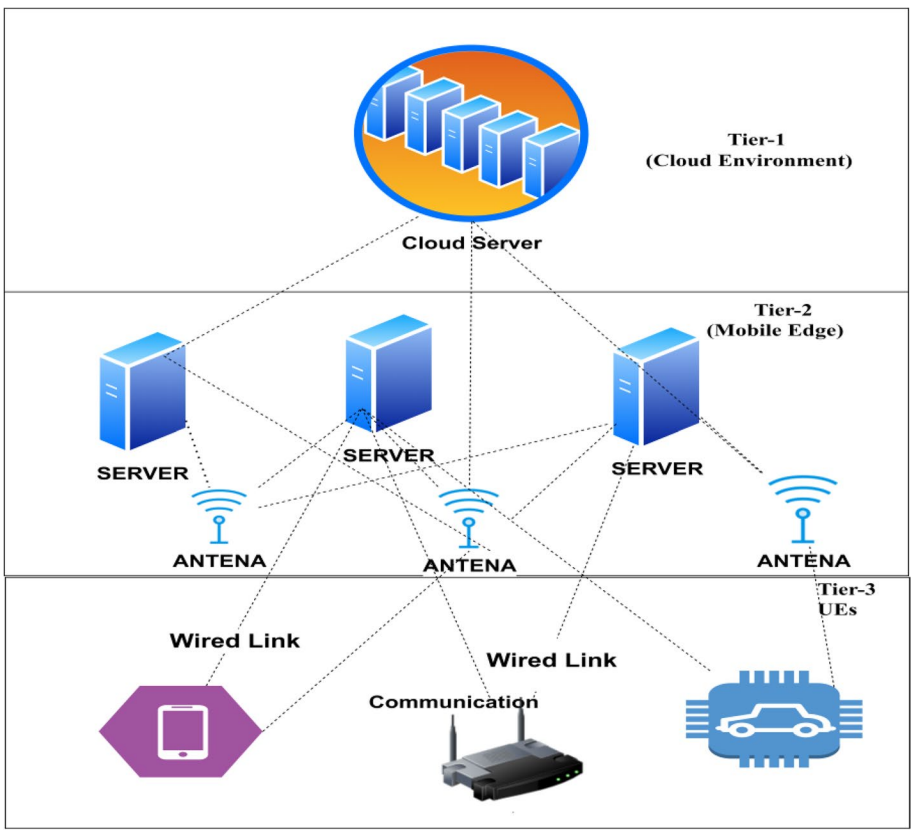

Fig. 1 Multi-hop mobile edge communication

to UEs at the edge of mobile devices. In MEC, UEs offload compute-intensive and time-sensitive optimization techniques to the MES via mobile communications. This offloading is crucial to reduce disruptions in UE serving and energy usage since it is challenging for an UE to meet the demands of high computational systems with limited processing and storage resources [6].

Cybertwin in MEC

Cybertwin technology is a novel approach to network design, especially within the discipline of MEC. Cybertwin in MEC mainly generates a virtual duplicate of entities like persons, equipment, and procedures within a virtual domain. Cybertwins are virtual copies that can imitate the behaviour and traits of real-life entities, allowing them to engage with the MEC network and its services in a way that reflects real-world situations [7]. Cybertwin-powered edge computing is a viable way to meet high user demand, although it brings additional issues. An efficient offloading job while managing computing, communication, and cache resources is challenging for edge networks. Conventional statistical optimization techniques cannot solve the offloading issue in a dynamic edge computing setting. In general, Cybertwin technology can substantially contribute to improving MEC networks' efficacy, performance, and user experience. Cybertwin technology, in the broader context of MEC, provides numerous advantages and functionalities [8].

- **Resource Administration:** Cybertwin can help administer computing resources effectively by offering information about customer and device behaviour and resource requirements. The information gathered can assist in improving the allocation of resources and task offloading choices in MEC settings [9].
- **Workload Offloading:** By incorporating Cybertwin's perspectives, mobile edge computing networks can make intelligent movements related to workload offloading. Cybertwin can assist in determining the best times and places to delegate work to maximize output and efficient use of resources.
- **Experience of Users:** Cybertwin can enhance user experience in MEC settings by providing personalized and context-aware services. Cybertwin can assist in forecasting user behaviour and adjusting programs and services accordingly [1–4].
- **Security and Integrity:** Cybertwin can enhance security in MEC networks by offering a virtualized setting to evaluate security procedures and standards. They tend to help identify and reduce security risks as they occur [5–9].

Deep learning in MEC

Numerous research studies have examined applying deep learning techniques, specifically CNNs and LSTM networks, to enhance workload offloading in MEC settings. The researchers demonstrated how deep learning may improve resource allocation, decrease latency, and boost the overall performance of MEC systems.

One crucial research [10, 11] focuses on utilizing Convolutional Neural Networks for image processing applications in Mobile Edge Computing. CNNs have effectively been used to transfer demanding image identification and processing duties from edge devices to faraway servers. These researches have demonstrated notable enhancements in the efficiency and accuracy of image processing in MEC situations by utilizing the hierarchical feature extraction capabilities of CNNs. Similarly, LSTM networks have been investigated for offloading jobs requiring sequential data, such as speech recognition and natural language processing [12, 13]. LSTM networks excel at capturing distant relationships in sequential data, making them perfect for jobs that involve grasping context over time. Research has shown that LSTM networks help decrease latency and enhance the accuracy of voice recognition and natural language processing tasks in MEC systems.

The present research on the utilization of CNNs and LSTM networks in MEC workload offloading emphasizes the capability of deep learning to improve the effectiveness and productivity of edge computing systems. Deep learning approaches can optimize resource utilization, minimize energy consumption, and enhance user experience in MEC systems by efficiently assigning work between edge devices and distant servers.

Problem statement and research motivation

The decentralization of data processing and storage resources closer to individual users has considerably improved wireless network reliability because of the rapid expansion of MEC. However, there are additional issues with the conservation of energy and management of resources because many MEC applications are resource-intensive. In response to these problems, this paper presents a novel approach to maximize energy efficiency in MEC networks. Our method, which combines deep learning methods with Cybertwin-driven techniques, improves work administration and resource allocation by utilizing Cybertwin technological advances.

The investigation is motivated by the potential advantages of improved energy efficiency in MEC relationships. The suggested technique aims to improve the overall effectiveness and environmental impact of MEC networks by minimizing energy usage and optimizing the allocation of resources. Enhancing MEC networks' intelligence and decision-making skills with the incorporation of Cybertwin technology presents a unique possibility that promises to enhance the handling of workloads and the allocation of resources. This study aims to develop energy-efficient and environmentally friendly MEC networks, enabling carriers of networks, suppliers of services, and consumers.

Key contributions

In our study, we aim to contribute to the field of research by introducing a novel approach for energy-efficient workload offloading and resource allocation in Cybertwin-driven MEC networks. We address the challenge of optimizing resource allocation, workload distribution, and service latency in MEC networks by proposing a deep hybrid transfer learning model coupled with a Cybertwin-driven resource allocation strategy. Our main contributions can be summarized as follows:

- **Cybertwin-driven Resource Allocation Model:** We introduce a novel Cybertwin-driven resource allocation model that leverages digital simulations to enhance resource allocation efficiency and reduce service latency in MEC networks. By employing Cybertwin, which offers real-time adaptability and flexibility, our model can dynamically adjust to user behaviour and environmental changes, leading to improved performance compared to traditional VM-based execution scenarios.
- **Hybrid Deep Transfer Learning for Workload Offloading:** We proposed a hybrid approach for workload offloading in MEC networks. Combining CNN and LSTM models with transfer learning enhances the accuracy and efficiency of task-offloading decisions. This approach allows our model to extract advanced features from MEC trace datasets more effectively, leading to superior performance compared to existing methods.
- **Experimental Validation and Analysis:** We conduct comprehensive simulations to evaluate the performance of our proposed model under different scenarios. Through extensive experimentation and analysis, we demonstrate the effectiveness of our approach in achieving better resource utilization, energy efficiency, and service latency reduction in Cybertwin-driven MEC networks.

By presenting detailed experimental findings and analysis, we provide insights into our proposed methodology's practical implications and benefits. Our results showcase the superiority of the proposed Cybertwin-driven resource allocation model and hybrid deep transfer learning approach over existing methods, highlighting the potential for significant advancements in energy-efficient workload offloading and resource allocation strategies in MEC networks.

Organization of the article

The article is organized to thoroughly discuss the issues and solutions related to resource allocation and task offloading in energy-efficient MEC networks. The article starts with an introduction emphasizing the importance of MEC and the necessity for energy efficiency. The related work section examines current MEC research, specifically focusing on workload offloading and allocation of resources algorithms.

The section on materials and techniques describes a deep hybrid transfer learning model that incorporates Cybertwin-driven allocation of resources and utilizes the MEC trace dataset. The outcomes section displays the outcomes of the experiment, which include energy consumption measurements and enhancements in performance. The discussion section evaluates the findings and explores the implications and constraints of the model. The conclusion provides a comprehensive overview of the main discoveries and proposes potential areas for future research, offering a complete perspective on the research's impact on the field.

Related works

The "Cybertwin" concept is a novel and modern notion that has attracted interest in the possibilities of using it in MEC. Although a recent notion, there is an increasing interest in investigating its relevance to MEC systems. The Cybertwin methodology provides computing flexibility for offloaded jobs, considering energy and time limitations, making it a viable method for enhancing MEC performance.

Multiple research studies have explored using Cybertwin with Mobile Edge Computing to enhance effectiveness. One research [12] suggested an offloading paradigm that uses a cloud-based software component. The model utilized a cross-compute split strategy and an asynchronous optimization mechanism to reduce execution time for all clients. Different research [13] concentrated on an energy-efficient offloading approach that integrates forward and backward methods to optimize energy offloading while constraining transmission latency.

Researchers have investigated many methods to improve the efficiency of MEC. Research [14] examined an offloading approach that utilizes multi-patching procedures to enhance MEC efficiency. This approach utilized a connected feature to improve energy distribution among various customer access points and infrastructure. A different research [15] suggested a Markov decision-making approach to enhance offload frequency in MEC, considering network connection and a variety of MES. The objective was to determine the best offloading window by solving the MDP with the valuation iteration technique.

Machine learning has been used to offload activities in Mobile Edge Computing. The study [16] examined a machine learning-driven offloading model designed to tackle offloading obstacles and improve energy management. Resource scheduling schemes have been suggested to enhance MEC performance. Research suggested a resource scheduling design that utilizes neighbouring static devices to establish a flexible computing network, improving overall performance. Another research [17] investigated the effectiveness and functionality of offloading solutions in mobile cloud-based systems. The study explored advanced technologies utilizing linked devices and IoT across many application areas, highlighting the impact of cloud spatial allocation and energy efficiency.

Several offloading strategies in virtualized channels to decrease energy usage and data transmission emissions are discussed in [18]. Research thoroughly examined energy-efficient computation offloading by addressing the fundamental limitations of offloading value. Workload balancing options utilizing deep learning techniques have also been explored. A study investigated energy-efficient offload strategies for core tasks in a cloud computing platform, introducing an active offloading technique to decrease energy usage.

In [19], the researcher concentrated on developing creative offloading strategies for MEC, utilizing both traditional and multi-objective optimization techniques. One study [20] suggested a paradigm for partial offloading in MEC, while another study [21] presented a practical work offloading approach for mobile devices on mobile network architecture with limited energy resources. MEC has distinctive difficulties and possibilities that demand specialized strategies for peak performance. The study [22] examined a work-offload architecture designed to decrease energy consumption and enhance reaction time by considering the anticipated processing time and energy expended during computational transfer. A different research project [23] created preemptive multitasking offloading and global offloading strategies based on game theory to enhance energy efficiency in MEC.

MEC's unique features, including tiered server hierarchy, dynamic attributes of intelligent devices, and client device mobility, require specialized strategies for best performance. Table 1 presents a comparative analysis of various existing research.

Materials and methods

This section presents the materials and methods related to the present research. It covers the Cybertwin model and the proposed solution for MEC architecture.

Introduction to the Cybertwin model

A new technology for MEC was created called Cybertwin. As a result, a Cybertwin may additionally be considered a type of VM designed to act as the customer's MEC network representation. The Cybertwin replaced virtual machine services. As a result, it functions as a bridge between the MEC network and the customers, collecting user requests and providing them with public resources to do these.

It also suggests that, inside the Cybertwin architecture, cloud services may not connect with consumers and instead exclusively with its Cybertwin. Since cloud servers and clients do not directly communicate, regardless of whether the customer turns and assumes a unique network identifier, the connection can persist as much as Cybertwin maintains the original hostname and address. Also, simply telling the client and the servers the updated Cyber twin location and address in the system makes Cybertwin transfer more straightforward than the VM server's migrations. Besides that, this complete separation between the server and the user implies that Cybertwin mobility is simpler than Virtual machine server mobility by notifying all the individuals and the web server service of the novel Cyber hostname within the cloud infrastructure. Furthermore, including a controlling layer in Cybertwin-based systems simplifies sharing and updating intelligence. Cybertwin is constructed to continuously communicate to the consolidated controller about internet traffic encompassing them, the activity on the servers on which they are located, and details about the jobs that customers receive individually.

As compensation, the central controller can combine these data to obtain comprehensive knowledge about the network to choose the best configuration settings, when to move Cybertwin, and where to complete assignments. These judgments are subsequently

Table 1 Comparison of various exciting research

References	Key method used	Service delay	Multi-user	Energy Consumption	Partitioning	Multi-Server	Deep Learning	Cybertwin and MEC
[12]	Offloading based on MDP	No	No	Yes	No	No	Yes	No
[13]	Energy Harvesting	No	Yes	Yes	No	No	No	No
[14]	Energy Model	Yes	No	Yes	No	Yes	Yes	No
[15]	Conventional offloading Methods	Yes	No	Yes	Yes	No	No	No
[16]	Game theory model	Yes	Yes	No	No	No	No	No
[17]	Energy efficient offloading	Yes	No	Yes	No	Yes	Yes	No
[18]	Offloading using the cost function	Yes	No	Yes	Yes	No	No	No
[19]	Machine learning-based energy model	Yes	Yes	No	No	No	No	No
[24]	Offloading using a Genetic Algorithm	Yes	Yes	No	No	No	No	No
[20]	Reliability-aware energy consumption model	Yes	No	Yes	Yes	No	Yes	No
[21]	Offloading with energy efficient model using OCR case	Yes	Yes	No	No	No	Yes	No
[22]	CNN based offloading	Yes	Yes	Yes	No	No	Yes	No
Proposed Model	Cybertwin-driven resource allocation model and CNN-LSTM with Transfer Learning	Yes	Yes	Yes	Yes	Yes	Yes	Yes

communicated to the Cybertwin. This enables optimal efficiency through the quickest task execution through a stable load among all cloud storage, which is a crucial statistic for the quality of experience at MEC. Cybertwin additionally possesses the capacity to offload duties to any remote server.

Figure 2 represents Cybertwin MEC architecture for communication. This architecture includes physical cloud servers, a Cybertwin layer with a control panel, and Cloud user devices. This architecture concisely describes a Cybertwin-based MEC, focusing on how the central controller receives signals through every Cybertwin to improve task distribution and synchronization. Other significant benefits of Cybertwin include the potential to keep track of critical acts and, if the consumer is potential, the opportunity to negotiate the transfer of intelligence to involved direct or indirect businesses. These details are not particularly relevant to the study discussed in this article; however, readers are curious about how Cybertwin functions and how these additional advantages are directed to secondary sources.

Proposed Cybertwin MEC model

This research presents a deep hybrid transfer learning with a cybertwin-driven resource allocation model for energy-efficient workload offloading and resource allocation for Cybertwin-driven MEC networks. The complete research is divided into two scenarios. Figure 3 presents the critical goals of the proposed Cybertwin MEC model.

The first scenario presents a Cybertwin-driven resource allocation model and performance enhancement for Cybertwin-driven MEC networks. This phase achieves better resource allocation, workload allocation, precise location selection, content caching decisions, and less service delay. The second phase achieves better results for Cybertwin offloading, energy efficient offloading, less delay, high accuracy, and better workload allocation.

Phase 1: Cybertwin-driven resource allocation model

This research aims to assess how the central controller may improve Mobile Edge Computing performance, specifically focusing on the issues encountered by

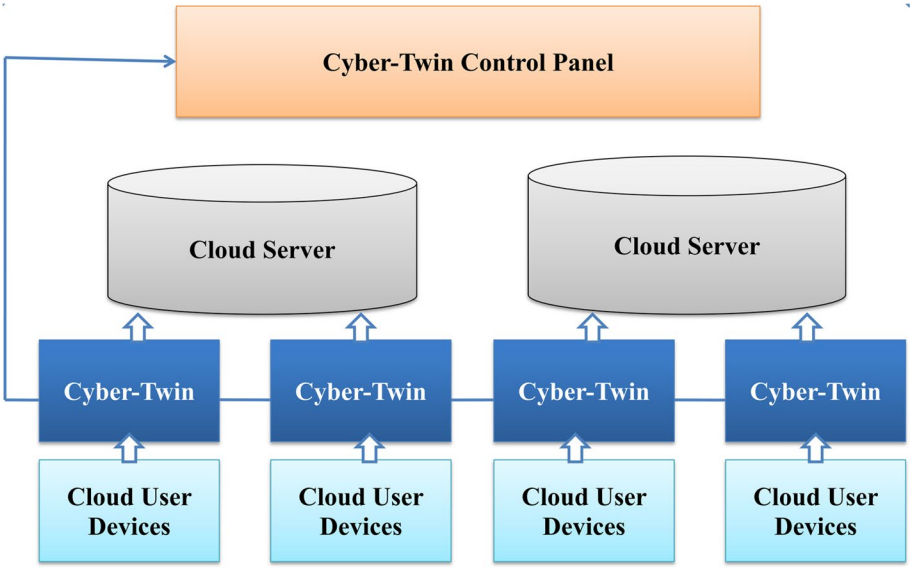

Fig. 2 Cybertwin MEC architecture

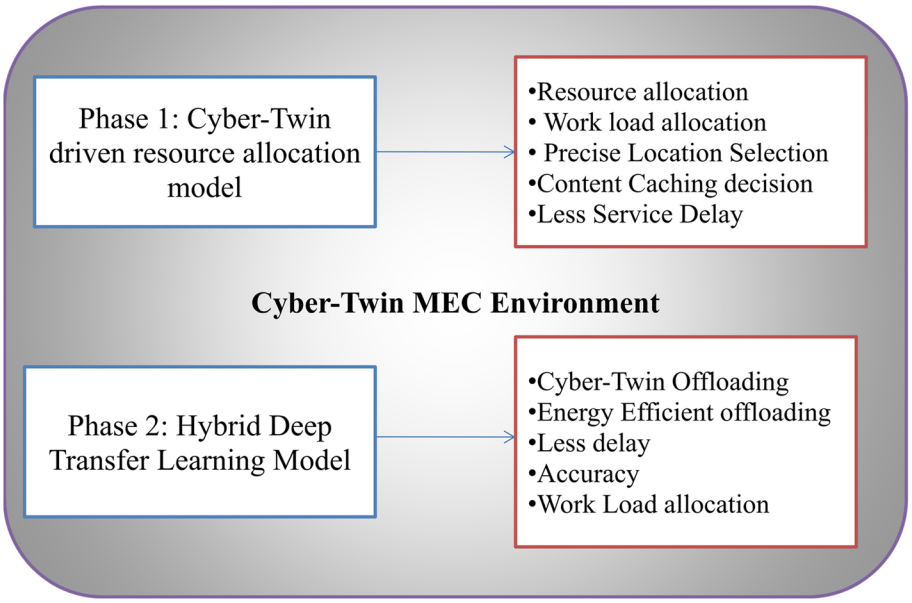

Fig. 3 Key goals of the proposed Cybertwin MEC model

Cybertwin in MEC operations. The obstacles involve delays in communication across both the data and control planes and the Cybertwin, in addition to the current use of VM server-based strategies, which further complicates the shift to Cybertwin and increases costs by requiring updates to all cloud-based technologies to meet the new standard. Cybertwin must substantially improve efficiency to be beneficial.

During this phase, we aim to determine how implementing Cybertwin might decrease service delays. We introduce a resource allocation model driven by Cybertwin to forecast the anticipated service delay in an MEC

framework based on Cybertwin. The approach is created to tackle the difficulties and intricacies of implementing Cybertwin in MEC. Figure 4 displays the interval-based divisions that segment the proposed resource allocation model powered by Cybertwin. The model operates as follows:

- **Feature Extraction:** The framework extracts features from the data collected, including workload intensity, utilization of resources, and delay in the network.
- **Cybertwin Integration:** The framework incorporates Cybertwin towards allocating resources, considering its skills and limitations.
- **Resource Allocation:** The framework uses retrieved characteristics and Cybertwin connectivity to identify the most efficient allocation of resources to reduce service delays.
- **Service Delay Prediction:** The approach utilizes given resources with the Cybertwin-driven allocation of resources framework to forecast the anticipated service latency for each job or request.
- **Feedback Loop:** The framework constantly analyses the MEC ecosystem and adapts the resource distribution approach using the real-time MEC dataset and Cybertwin's input.
- **Performance Evaluation:** The model assesses its success by reducing service delays and enhancing overall MEC efficiency.

Cybertwin service model The Cybertwin service framework outlines the interactions between Cybertwin components and the MEC system for the provision of services. According to this approach, every user has a Cybertwin that acts as a virtualized version of them in the MEC context. Working in conjunction with the central controller and additional MEC system components, Cybertwin optimizes resource allocation and carries out various functions. We have analyzed a situation that includes several cloud users, various cloudlets, several cloud base stations, and a substantial external server. Every user has a Cybertwin, controlled by a unique cloud system. Activities are sometimes best completed regionally based on the status of cloud servers as well as networks of communication. Users of clouds create activities and decide whether to process them on their local device or in the cloud system.

There are two primary sorts of tasks: a) requests for resources and b) the processing of tasks. Particular data will be retrieved and returned to the user by cloud services upon completion of the needed tasks. The microprocessors in the ecological system need to perform processing operations in a background environment where the work is completed. Resource requests should be resolved in the cloud but can be handled locally if it is more efficient. Queries can be processed locally and sent to a base station wirelessly, while jobs

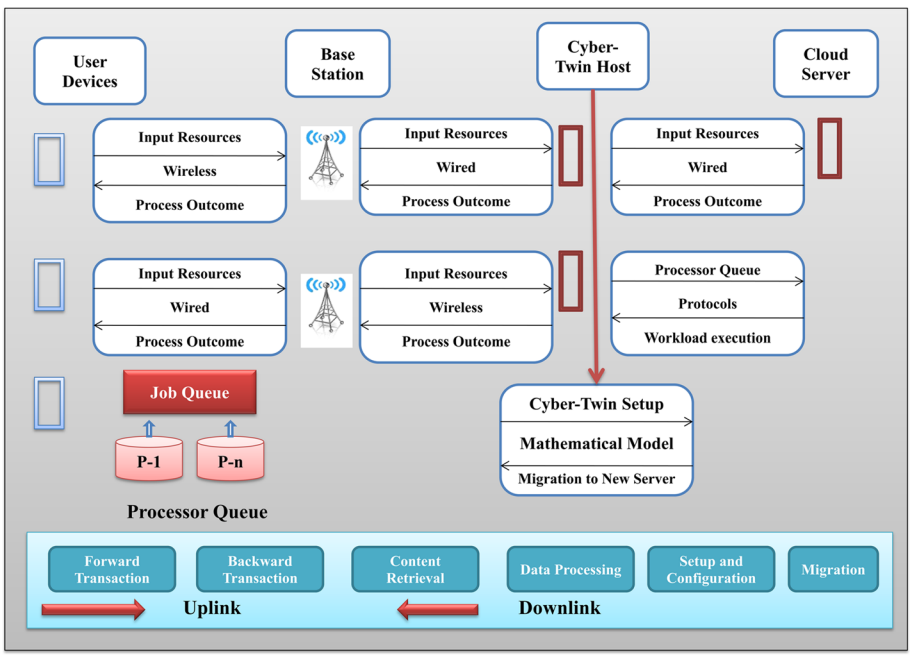

Fig. 4 Phase 1 service model for Cybertwin MEC

can be handled on a virtual machine in a cloud-based setting.

We are examining a scenario that includes numerous cloud users, various cloudlets, and several cloud base stations, including a substantial external server. Every user utilizes Cybertwin, which is controlled by a unique cloud-based platform. Various factors, including the state of the servers in the cloud and their communication networks, make it more advantageous to finish tasks in the particular area where they are located. Cloud users create activities and decide whether to process them locally or within the ecosystem.

Tasks are classified into two distinct categories: demands for resources and processing of tasks. Cloud-based services retrieve and transmit back the specific data as prompted by activities. Task completion ecosystem microprocessors are required to execute processing duties in the system's background. Resource queries must be resolved in the cloud, although they can be performed locally if more practical. The requests can be performed locally and sent to a base station wirelessly, while activities can be executed in a VM through a cloud-based computing system (Algorithm 1).

Where:

Symbol	Meaning
T_r	Task requests from the user
Cl_i	Cloudlet
P_i	Processor set
O_m	Optimum model with a higher capacity
∑**cloudlet Capacity**	Total capacity of all cloudlets
∑ **Full capacity**	Full capacity of the system
Cl_i **for local-level processing**	Assign Cloudlet for processing tasks locally.
Cl_i **to** O_m	Assign Cloudlet to the optimum model.
Modified the capacity of Cl_i	Update the capacity of the Cloudlet after the assignment.

Cloud user mobility Participants can keep moving in the Cybertwin MEC environment. If consumers relocate, they might situate themselves far from the cloud environment currently hosting they employ through Cybertwin, making accessing it more difficult. It ought to be taken into account by the network when choosing where or how to maintain the Cybertwin when one must relocate it and from where to transmit the activities for processing. Because customer locations, as well as network accessibility mechanisms, can transform,

Algorithm 1. Cybertwin Region Selection Method

Input: Cloud resources, Server, user request, Processor and Job Set
Output: Region selected accurately for each Cybertwin-driven model

1. Initialize: For each user task request, T_r
2. For each Cloudlet Cl_i and processor set P_i:
3. Find Optimum Model: Search for an optimum model O_m near Cl_i and P_i with higher capacity.
4. Capacity Check:
   4.1 If ∑Cloudlet Capacity == ∑Full Capacity
   4.2 Assign Cl_i for local-level processing.
   4.3 Else:
   Assign Cl_i to O_m.
5. Modify Capacity:
   5.1 Update Cli's capacity for all remaining assignments.
6. End For

limiting movements could result in inaccurate modelling and ineffective solutions. We shall partition time further into time frames and characterize motion across individual time slots. Suppose many CU cloud users are defined (c_{u0},..., c_{ui}).

Also, their t_s time-spaces are expressed by (t_{s0}... t_{sj}). The customer cu_i parameters for timeframe t_{sj} can then be (x (cu_i, ts_j), y (cu_i, ts_j)). The position shifts each time the individual moves around. In the case of a stable customer, this position is constant across all time frames. Given that too many users adhere to set patterns with predicted courses and places, it is reasonable to presume that the system can retrieve this data (Algorithm 2).

Algorithm 2. Cybertwin Content Caching Modelling and User Mobility

Input: *Cloud resources, Server, user request, Processor and Job Set*
Output: *Region selected accurately for each Cybertwin-driven model*

1. *Initialization: All processors are initialized to a processor set P_i.*
2. *Content Caching:*
   2.1 For each content and resource:
   2.2 Verify all cloudlet details starting from the initial cloud.
   2.3 Set the cache at the initial cloud and let it wait in the queue.
3. *Predict the required memory:*
   3.1 If the sum of demanded memory exceeds the sum of available memory, skip caching for this content.
   3.2 Set the processor set Sp_i accordingly.
   3.3 If the number of processors exceeds a certain limit, reset it.
4. *Cloud User Mobility:*
   4.1 Define a set of cloud users.
   4.2 For each user at each time:
   4.3 Predict the movement.
   4.4 Update the position based on the prediction.
   4.5 Consider user mobility in caching and processing decisions.
5. *End*

Task mapping Each cloud user uses a Poisson production process to build activities, including an optimum frequency of (γ) one work per instant. Every action has a (ϕ) probability of getting a resource processor work and a (1-ϕ) probability of getting a resource request activity. The customers may request (ζ) particular items, which are indicated by (σ_0...σ_i..., σ_ζ) for index i. The chance of contents becoming ordered follows directly from the concept of Zf variances if we believe that the elements are contained in decreasing rank, so it is the Zf distribution factor.

$$\beta * (\sigma_k) = \frac{k^{-\varphi}}{\sum_{i=1}^{\zeta} * q^{-\varphi}} \tag{1}$$

In Eq. (1), q represents summation, ζ shows the number of cloudlets, p_i is the set of processors, and β is the Z_f distribution factor. The task can be executed locally in the cloud environment. The task allocation depends on the mapping factor Δ, which can be defined by Eq. 2.

$$f(\Delta) = cl^n + \sum_{n=1}^{\infty} \left(p_n * \frac{c_u}{l} + \frac{\sigma 0(\zeta)}{q} \right) \tag{2}$$

Here C_l is the number of cloudlets, P_n is the number of cloud processors, and c_u is the number of cloud users.

Here:

Symbol	Description	Symbol	Description
γ	Optimum frequency of one work per instant for each cloud user	φ	The probability of a cloud user's activity being a resource processor work
ζ	Number of particular items that customers may request	σ	Index indicating particular items requested by customers
β	Zf distribution factor	k	Index for a specific item requested by a cloud user
q	Summation	ζ	Number of cloudlets
pi	Set of processors	Δ	Mapping factor determining task allocation.
f(Δ)	Task allocation function based on the mapping factor, number of cloudlets, processors, and users	Cl	Number of cloudlets
Pn	Number of cloud processors	cu	Number of cloud users

Phase 2: hybrid deep transfer learning model

The second scenario is based on fractional offloading processes, which estimate the cost for all potential segmentation and perhaps even offload strategies and then select the fractional and offload system having the lowest cost. As a result, despite the substantial computational complexity, the energy usage and implementation time are minimal. The proposed model divides a task into small sub-tasks. User direction forecasting is determined using an LSTM. Figure 5 shows the working model of the proposed model. The proposed model divides tasks into partitions using a task partitioning method. The proposed model utilizes two types of task processing strategy: one is local execution at the UE site, and another is remote execution using an offloading process [25].

A time series forecast problem emerges from the mobility data's non-linear characteristics. To train the training algorithm and forecast the future position, the LSTM uses primary motion variables, including work, speed, and directions, as inputs to a feed technique. We utilize the CNN deep learning method to accelerate and optimize the strategic decision procedure while preventing a high computational overhead and complexity. The proposed model divides a task into small sub-tasks. User direction forecasting is determined using LSTM. Transfer learning is utilized to transfer the learning knowledge to the CNN-LSTM model to enhance earning accuracy. The proposed model is trained on a comprehensive mobile edge computing dataset. The proposed model is based on a partial offloading strategy [26].

In which a cumulative work is divided into n partitions, $p = (p1, p2 \ldots pn)$, before partitioning, researchers presume the degree of divisions per activity, p_n, is normally known. As indicated in Fig. 5, all divisions, $pi\ \varepsilon\ p$, for (i = 1, 2, p_n) are sometimes sequentially performed natively or

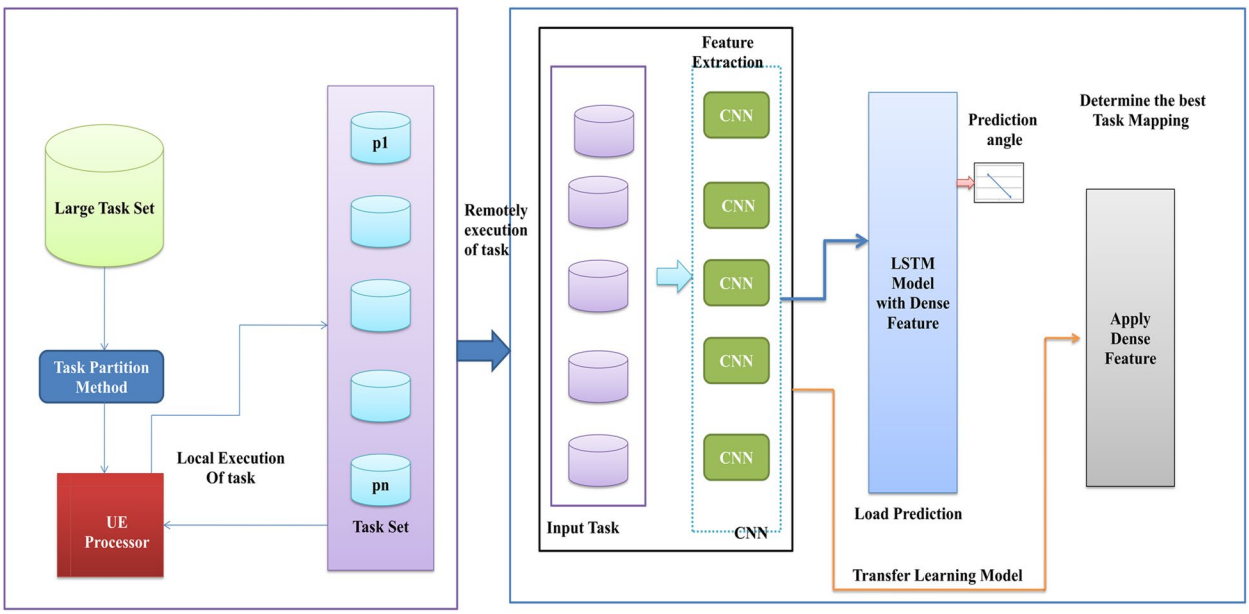

Fig. 5 Working of the proposed model

offloaded onto MES. We add a Boolean value, Bi ε[0,1] to express it numerically, its range is [0, 1]. Pi runs immediately as UE when Bi = 0; otherwise, it runs globally via MES. As a result, we make predictions, including local and distant deployments. The final acquired data for pi is denoted simply as i, and the required pi information is recorded as i. The number of CPU cycles needed to execute assignments pi is βi, which directly relies on the cost of γ_i, as $\beta i = (\gamma i * ni)$, where n_i shows the number of CPU cycles for i iteration per bit [27].

Local task execution strategy

We take into account UEs' capacity for heterogeneous processing. As a result, the overall time required to execute pi natively, T_{delayi}, can only be expressed as [28].

$$T_{delayi} = \frac{\beta i}{Fre_{UE}} \tag{3}$$

Here Fre_{UE} shows the CPU frequency to execute a process for partition pi under UES. In a comparable pattern, total energy utilization resultant from the localized processing of pi, $Energy_{Locali}$ can only be described as (4). Here, ε denotes a constant that depends on the typical UE activation component and switches sensitivity. Similar ζ is a constant energy variable $\zeta > 2$.

$$Energy_{Locali} = T_{delayi} \varepsilon Fre_{UE}^{\zeta} \tag{4}$$

Remote task execution strategy

A factor p_i may be uploaded by a UE toward the MES for processing. Unless the channel employs multi-flexing, we may consider its broadcast frequency band. $Freq_{Band}$ is partitioned across Sub_{part} Sub-channels [29]. The possible carrier frequencies are enabling. T_{rai} as well as Rec_i Rec_i transmissions and receptions, correspondingly, are denoted as Sub_i =(1, 2,3,..., Sub), at which Sub is the highest allowable sub-carriers.

Identical to Rec_i, *here* Rec indicates the highest range of CPU processors at MES and Rec_i indicates the number of CPU processors that are used to handle p_i. Rec_i= 0 suggests that now the systems are occupied, and therefore, there is no separate CPU reserved for the element p_i. We assume additive white Gaussian distortion for both downlink and uplink connection speeds [30], which can be represented as follows.

$$Up_{link} = \frac{Sub_i}{Sub} C * log_2(1 + \frac{Transmission_{Power|Channel_{Fa\_uplink\_Cofficient}|^2}}{\varphi(Signal_{Uplink\_noise})\gamma_i^0 * Noise_{Power}}) \tag{5}$$

$$Dn_{link} = \frac{Sub_i}{Sub} C * log_2(1 + \frac{Transmission_{Power|Channel_{Fa\_down\_link\_Cofficient}|^2}}{\varphi(Signal_{downlink\_noise})\gamma_i^0 * Noise_{Power}}) \tag{6}$$

Cost function estimation

The complexity of the algorithms and high computational costs of traditional optimization algorithms with cost functions including limitations. As a result, we must create a particular cost function that considers all relevant variables to generate a trained model exclusively [31]. Even though only one computation is made throughout the training process, the method performance for creating a trained model using such a particular cost function is substantial. After the initial training, the developed CNN has continuous complexity O (1).

The total functional cost primarily depends on the implementation, communication, receiving, task-division-related latencies, and specific energy utilization. Moreover, the cost model considers the duty cycle, radio capabilities, and computational power. The task-division action causes the division procedure to take longer and use more energy as the number of elements climbs [32]. As a result, the assignment latency per element, $Delay\_Task_{division}$, can be expressed as follows:

$$Delay\_Task_{division} = fun1(m) = \frac{(m-1)}{m} \varphi_{delayed\_task\_division} \tag{7}$$

Proposed algorithm for workload offloading

Towards the proposed method, we partition a workload across n portions, and afterwards, using the fractional offloading method, the UE offloads parts of the parts to MES while others are processed on UE. Nevertheless, there have been n_{Z_m} entail partition alternatives, as well as 2^m Potential offloading possibilities for a work of length m, therefore finding the alternative only with the lowest cost requires a solution of performance $O(n_{Z_m} * 2^m)$.

We build a training sample utilizing the fully comprehensive computational formula to calculate the expenses, including all mZn2n alternatives, to prevent this calculation overhead with computational complexities. Algorithm 3 considers all feasible partitioning (division matrix, (P, Par)) with partially offloading strategies (offloading strategies matrix, (P, OP)) for quite a workload of length m with just a fixed variable Y. where (P, Par) shows the predicted partitions, and (P, OP) shows indicated offloading policies [33].

Algorithm 3. Fractional offloading with task partitioning

Input n: number of portions, γ: fixed variable, o: offloading parameter, s_i: input data size, l_i: computation load, β_i: bandwidth parameter **Output** Better policies (BP*), less energy consumption (LEC*)
Step 1: Initialization: Initialize all the initial parameters, i.e., n, γ, s_i, l_i, and β_i Step 2: Set the division matrix (P, Par) Step 3: Set the offloading policies matrix (P, OP) Step 3: Initialize other required variables Step 4: Procedure: For each feasible partition (division matrix P, Par): 4.1 For l = 2 to n_Z_m: 4.2 Assign P(l) from P(l, :) 4.3 For each offloading strategy (offloading strategy matrix P, OP): 4.4 For q = 1 to 2^j: 4.5 Assign e(q) from OP(l, :) Step 5. Calculate Costs for Each Strategy: 5.1 For k = 1 to p: 5.2 If e(k) == 0 (no offloading): 5.3 Calculate cost(k) using cost function for local processing f(md) 5.4 Else (offloading): 5.5 Calculate cost(k) using cost function for offloading f(nd) Step 6. Compute Total Cost: 6.1 Sum up costs: $cost_j(m) = \sum cost(i)$ Step 7: Determine Minimum Cost: 7.1 Find the index with the minimum cost: [index, cost_minimum] = min(cost_minimum) 7.2 Assign optimal offloading policies: OP2(m) = OP2(index, :) 7.3 Assign optimal partitions: BP* = OP2(index, :) 7.4 Assign lowest energy consumption: LEC* = Par(index, :) Step 8: Output Results: Bp*, LEC*

Where:

Symbol	Description	Symbol	Description
n	Number of portions	βi	Bandwidth parameter
γ	Fixed variable	P	Division matrix
o	Offloading parameter	Par	Predicted partitions
si	Input data size	OP	Offloading policies matrix
li	Computation load	e	Offloading strategy
md	Cost function for local processing	nd	The cost function for offloading
BP*	Optimal offloading policies	LEC*	Lowest energy consumption
p	Number of strategies	l	Loop variable for partitions
q	Loop variable for offloading strategies	k	Loop variable for calculating costs
j	Exponent for offloading options	index	Index of the minimum cost
cost	Cost array for different strategies	cost_minimum	Minimum cost value

Dataset

The MEC (trace files) dataset comprises traces and logs documenting different operations and occurrences inside a MEC environment [34]. This dataset is commonly utilized for research to analyze and assess algorithms, models, and systems associated with MEC. The dataset mainly contains information such as:

- **User Activities:** Details of user interactions with edge services, including task requests, task completions, and user mobility patterns.
- **Resource Usage:** Information about utilizing computational resources, such as CPU, memory, and storage, at the edge servers and cloud servers.
- **Network Conditions:** The information on network efficiency parameters, including latency bandwidth and packet loss, among edge servers, cloud servers, and consumer devices.
- **Energy Consumption:** Monitoring energy usage of edge servers and cloud servers, including devices used by users across various tasks.

- **Task Offloading:** Information on task offloading selections specifying the tasks offloaded their destination and the outcomes of the offloading procedure.
- **Environment Characteristics:** Details on the MEC environment, including the quantity and placement of edge servers, the specific type of network infrastructure, and the resource availability.

Data preprocessing

Data preparation for the MEC trace dataset entails many essential stages to prepare the data for usage in the CNN-LSTM approach with transfer learning. The dataset, which includes user communications through edge services, activity requests, completed tasks, and mobility patterns, is initially gathered and processed to exclude unnecessary or incorrect data. The data is then converted into an appropriate format, which may involve adjusting timestamps and encoding category variables. The dataset is divided into training, validation, and test sets. Normalization is used to standardize the scale of all characteristics. Data augmentation strategies can be optionally employed to expand the dataset size and enhance the model's generalization. The data is sequenced into fixed-length sequences to consider its sequential character before input into the LSTM component. The preprocessed data is inputted into the model for training, validation, and testing.

Feature extraction and hyperparameter tuning

The CNN-LSTM model for the MEC trace dataset has two primary components in the feature extraction process. The CNN first extracts spatial features. The CNN module comprises convolutional layers, ReLU activation functions, and pooling layers. The layers collaborate to extract spatial information from the incoming data, such as pictures or sensor data. Transfer learning utilizes pre-trained CNN models like VGG-16, trained on extensive datasets like ImageNet. The pre-trained models function as feature extractors, extracting high-level information from the MEC trace dataset. The CNN produces feature maps that capture the spatial characteristics of the input data, which are subsequently forwarded to the LSTM for additional analysis.

Tuning of hyperparameters is an essential step in improving the CNN-LSTM model's performance on the MEC trace dataset. This method entails adjusting the hyperparameters, which govern the learning process and model structure. Important hyperparameters adjusted during tuning include the learning rate, batch size, number of epochs, and dropout rate.

The learning rate influences the speed at which the model learns and reaches the best answer. A greater learning rate can accelerate convergence but increase the likelihood of overshooting the ideal answer. Adjusting the learning rate optimizes the trade-off between velocity and precision. The batch size dictates the quantity of samples processed before adjusting the model's weights. Increasing the batch size can accelerate the training process but necessitate a higher memory capacity. Adjusting the batch size improves the training process to enhance efficiency and efficacy. Epochs determine the number of times the model goes through the entire dataset during training. Insufficient epochs can cause underfitting, while excessive epochs can result in overfitting.

Adjusting the number of epochs helps determine the most effective training period for the model. Dropout rate is a regularization method employed to mitigate overfitting. It randomly removes some neurons throughout training and compels the model to acquire more resilient characteristics. Adjusting the dropout rate enhances the model's capacity to generalize. In this research, we utilized the grid search method for Hyperparameter tuning. Optimizing the hyperparameters of the CNN-LSTM model can enhance its performance and accuracy on the MEC trace dataset, resulting in more dependable predictions and insights for Mobile Edge Computing applications. Table 2 presents the CNN LSTM parameters used for simulation.

Performance measuring parameters

The following performance assessment metrics are applied to evaluate the proposed hybrid model's effectiveness [35, 36].

Table 2 CNN, LSTM parameters

Parameter	Value
Input Shape	(128, 128, 3)
CNN Architecture	VGG16
CNN Layers	Conv2D(64, (3, 3)), ReLU,MaxPooling2D,Conv2D(128, (3, 3)), ReLU,MaxPooling2D
Transfer Learning	Yes
LSTM Layers	LSTM(128), Dropout(0.2), LSTM(64), Dropout(0.2)
Output Units	1
Loss Function	Binary Crossentropy
Optimizer	Adam
Learning Rate	0.001
Batch Size	32
Number of Epochs	50
Dropout Rate	0.2
Metrics	Accuracy, Precision, Recall

- **Delay**: The latency, communication delay, processing delay, and queue delay are added together to determine the transmission time delay.
- **Transmission Delay (TD):** The duration needed for a datagram to travel from the server toward the communication channel is referred to as the "transmission delay (TD)".

$$TD = \frac{Data\ Size}{Bandwidth} \tag{8}$$

- **Propagation delay (PD):** The packet must pass across the channel once transferred to the communication system to achieve its target. So, PD is just the duration for the final bit of a message to reach its intended location.

$$PD = \frac{Distance}{Velocity} \tag{9}$$

- **Queuing Delay (QD):** The receiver will not instantly begin processing the packets after delivery. It must remain idle in a buffer, which is a row. Hence, waiting in the queue is delayed, which means the time something takes to get in line before getting analyzed.
- **Processing delay (PrD):** The packets will subsequently be transferred for a computation procedure known as PrD. The processors need some time to execute the incoming packets, which is the same amount of time intermediary gateways need to select where to transmit it, update TTL, and calculate header checksums.
- **Energy Consumption (EC):** EC represents the energy the infrastructure uses for data gathering, transportation, and receiving. The evaluations between the various methods are primarily based on how much energy the multicast group and cluster formation edge devices consume.

Experimental results, analysis, and discussion

This section presents the implementation, results, and discussion. This research shows a deep hybrid transfer learning with a Cybertwin-driven resource allocation model for energy-efficient workload offloading and resource allocation for Cybertwin-driven MEC networks. The complete analysis is divided into two scenarios. The implementation of the proposed model and the existing model was performed based on the two scenarios. The first scenario is for the Cybertwin-driven resource allocation model, and the second phase uses hybrid deep transfer learning workload offloading and resource allocation for Cybertwin-driven MEC networks.

The modelling tool is MATLAB R2020a, which executes at 3.7 GHz over an Intel Core i-7, the company's 11th-gen microprocessor. Aside from the existing method, these techniques divide a workload of length n into eight modelling factors, subsequently performed sequentially on UE and via MES. For specific parts, every randomized parameter was autonomous [37]. The trained model is generated from 45,000 original data, which implies that we locally operated our proposed methodology on 45,000 assignments of various sizes, dispersed randomly throughout [0.0, 2.5] GB, storing the results at the most affordable price as labelling for the relevant raw data. Table 3 shows the Simulation parameters and respective values for each parameter used for scenarios one and two.

Simulation results for phase 1

The first scenario presents a Cybertwin-driven resource allocation model and performance enhancement for Cybertwin-driven MEC networks. This phase achieves better resource allocation, workload allocation, precise location selection, content caching decisions, and less service delay.

In the scenario of VM-based execution, tasks are transferred to VMs in the cloud. Although this method offers scalability and flexibility, it may lead to higher service latency because of the virtualization overhead and the requirement to communicate with cloud servers. On the other hand, the execution scenario using Cybertwin allows tasks to be assigned to digital simulations, providing more control and flexibility compared to traditional VMs. Our Cybertwin's can quickly

Table 3 Simulation parameters and details

Simulation parameter	Value
Time slots	8
Cloud users	5000
Cloud lets	20
The base station (for uplink)	20
A base station (for downlink)	10
Terrain	1000*1000 m
Memory (Cloudlet level)	15 GB
Delay for Cybertwin	1000 ms
Antennas (for uplink BS)	30
Antennas (for Downlink BS)	100
Capacity (Cloudlet)	1000 cloud users
Processor (Local)	5
ZF Distribution	0.75
Processor (Cloudlet)	15

adjust to user behaviour and adapt to changes in the environment.

Figure 6 illustrates how service delay times vary from 10,000 to 50,000 users, with service delay times expressed in seconds for various user counts across four levels: WiFi Meet, Cyber Twin, Remote, and Local. With 10,000 users, the Local Level delay is 1000 s, and for 50,000 users, it increases linearly to 5000 s, showing a consistent decline in performance as the number of users increases. Due to remote processing overhead, the Remote Level exhibits slightly higher delays, starting at 1100 s and ending at 5100 s. WiFi Meet performs worse than Local Level but better than Remote Level, ranging from 1050 to 5050 s. Cyber Twin Level performs the best, with a start time of 950 s and an end time of 4550 s for users in the same range. Cyber Twin Level is the best option for environments with high user demands because of its superior performance, attributed to its advanced load balancing, efficient resource allocation, enhanced scalability, and decreased latency.

Figure 7 showcases the service delay time for four different methods: Local Level, Remote Level, WiFi Meet, and Cyber Twin Level. The graph demonstrates how the delay time changes as cloudlets increase from 3 to 30. The Cyber Twin Level consistently demonstrates the lowest delay times, beginning at 4800 s for three cloudlets and gradually decreasing to 320 s for 30. On the other hand, the Local Level begins at 5000 s and drops to 520 s, the Remote Level starts at 5200 s and reduces to 720 s, and the WiFi Meet begins at 5100 s and decreases to 620 s for the same range of cloudlets. Based on the data, it is clear that adding more cloudlets generally leads to a decrease in service delay time for all methods. Notably, the Cyber Twin Level method shows the most substantial improvement. This emphasizes the effectiveness of the Cyber Twin Level in optimizing service delay, making it the most efficient method among those compared.

Figure 8 illustrates the correlation between service delay and execution time across four different levels: Local Level, Remote Level, WiFi Meet, and Cyber Twin Level. The service delay ranges from 50 to 500 s, while the execution time ranges from 1000 to 10,000 s. As the service delay increases, execution times at all levels decrease. At the Local Level, there is a reduction from 10,000 s to 1100 s, while at the Remote Level,

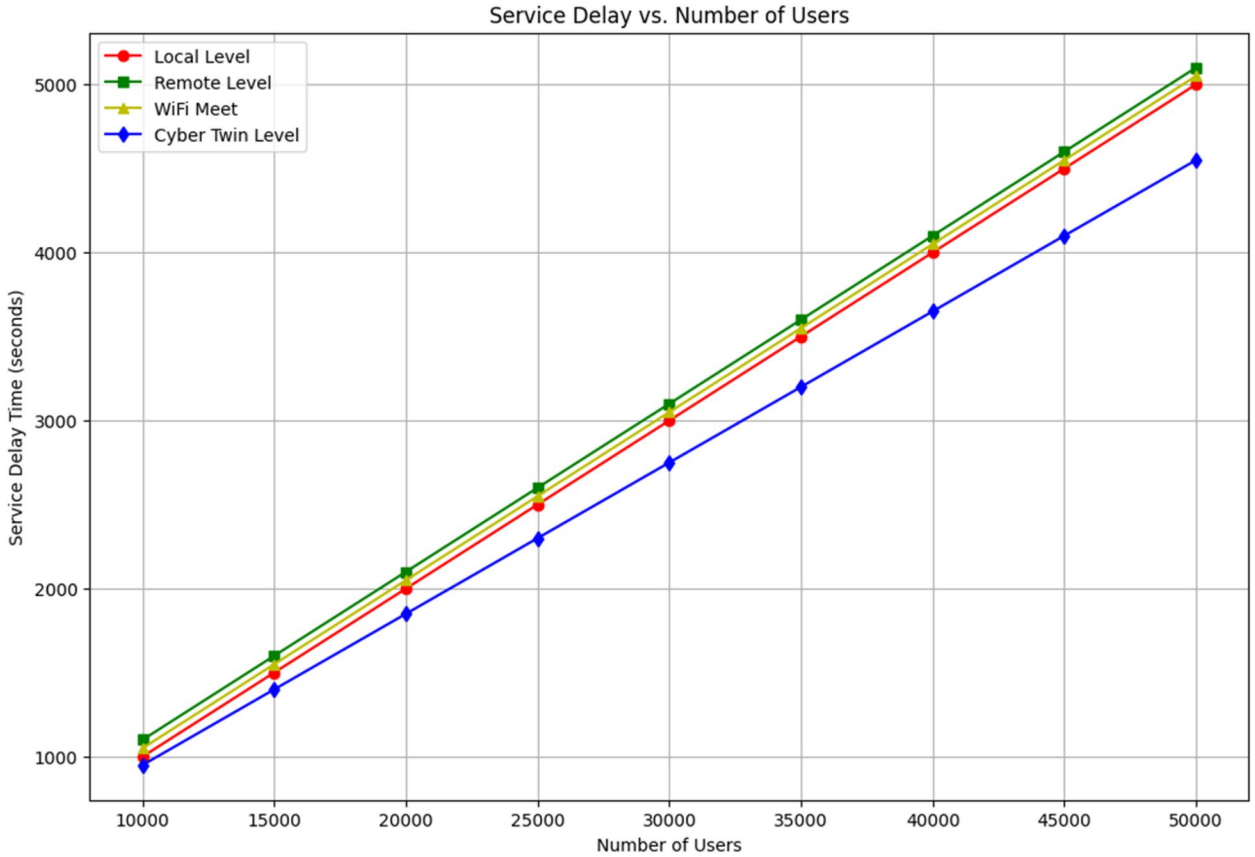

Fig. 6 Service delay vs. no. of users

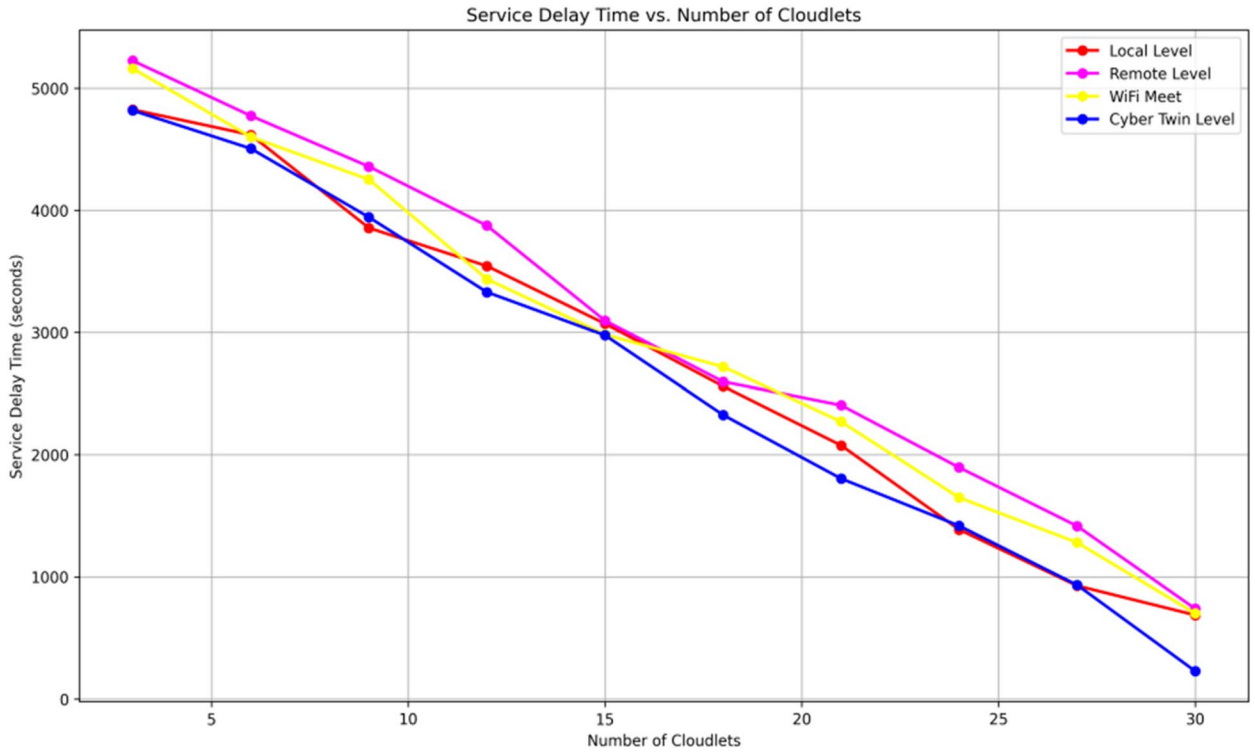

Fig. 7 Service delay vs. no. of cloudlets

the drop is from 9500 to 600 s. Like a pro, the execution time of the WiFi Meet level drops from 9000 to 500 s, while the Cyber Twin Level shows the biggest improvement, reducing execution time from 8500 to 350 s. The data fluctuations highlight the varying efficiency levels when increasing service delay. The Cyber Twin Level consistently achieves the lowest execution times, showcasing its outstanding achievement in handling higher service delays.

For the four scenarios, Local Level, Remote Level, WiFi Meet, and Cyber Twin Level, Fig. 9 depict the relationship between Service Delay and Remote Execution Time. In every scenario, Remote Execution Time rises tandem with the Service Delay, which goes from 500 to 5000 s. However, the rate of increase differs for every scenario. The Remote Execution Time in the Local Level scenario gradually increases since it is 80% of the Service Delay. A steeper increase results from setting the Remote Execution Time at 120% of the Service Delay in the remote-level scenario. The WiFi Meet scenario is in the middle, where the Remote Execution Time is higher for the Service Delay. The Cyber Twin Level scenario exhibits superior performance than the other scenarios. Its Remote Execution Time is only 50% of the Service Delay, indicating a significant reduction in execution times. Based on this fictitious data, for a given Service Delay, the Cyber Twin Level provides the best performance regarding Remote Execution Time.

Simulation results for phase 2

In the simulation, two results were calculated on the MEC trace dataset for existing deep learning models CNN, CNN-LSTM, and proposed CNN-LSTM with Transfer Learning, and they were compared based on various performance measuring parameters. The dataset is divided into 80% training and 20% testing ratio.

The second phase achieves better results for Cybertwin offloading, energy efficient offloading, less delay, high accuracy, and better workload allocation. The proposed model CNN + LSTM with transfer learning is compared with two existing models: the CNN model and CNN with LSTM. Following performance measuring, parameters were calculated for all these three moles.

An analysis of service delay times depending on different task sizes is shown in Fig. 10. The plot contrasts the delay times of a suggested hybrid model with those of existing models (CPNs, GCNs, GNNs, RNNs, VAEs, GANs, and RL). There are 10 to 100 Mega Bytes

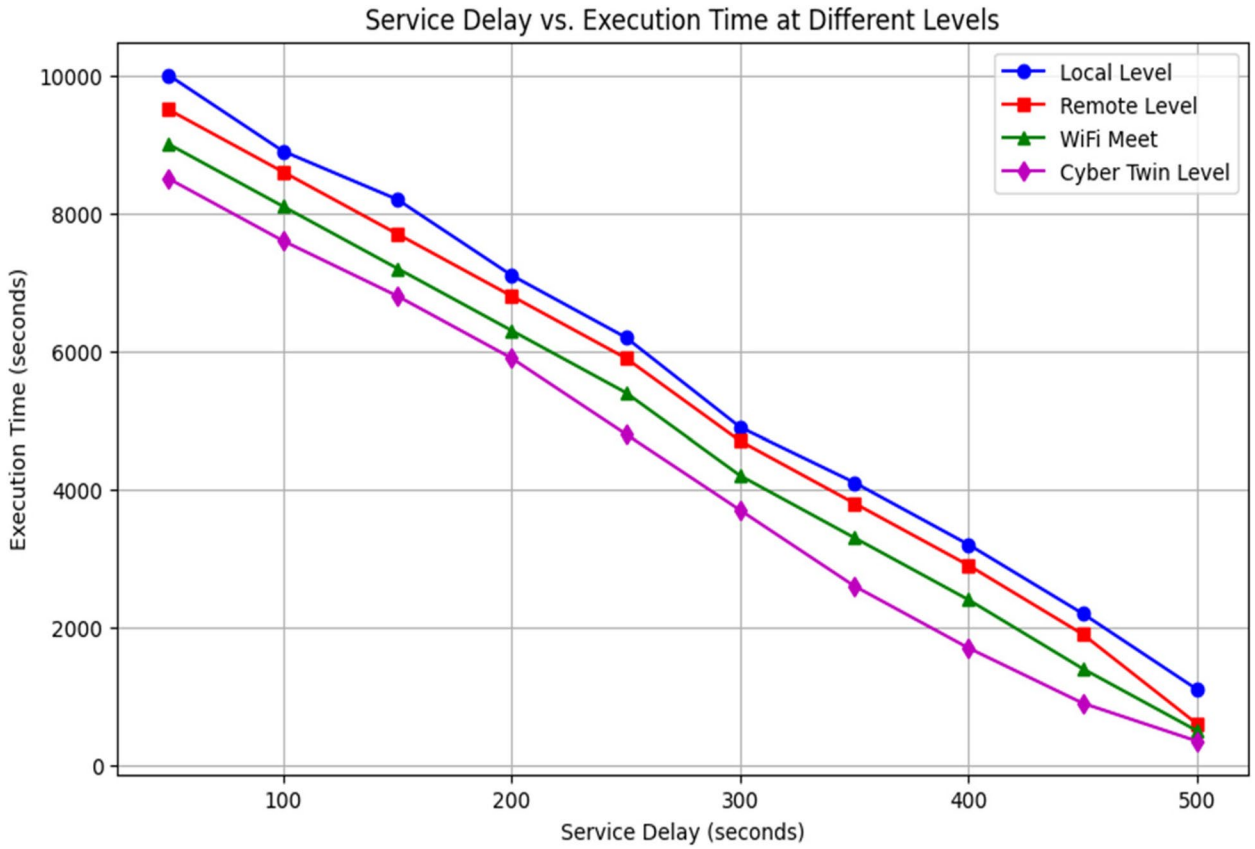

Fig. 8 Service delay vs. local execution

(MB) in a task. Every model currently in use shows increasing delay times as task sizes increase, exhibiting unique patterns influenced by their computational complexity. Across a range of task sizes, the suggested hybrid model typically provides competitive delay times, indicating possible efficiency gains. This analysis offers insights into optimizing service performance based on task characteristics, which is important for allocating resources and designing systems in computing environments.

The service delay times for a range of task sizes (10 MP to 100 MP) are shown in Fig. 11 for the following models: RL, CPNs, GCNs, GNNs, RNNs, VAEs, GANs, and the suggested hybrid model. The data shows a range of performance, representing actual variations in service delays. The suggested hybrid model performs marginally better than the others. The efficiency of the hybrid model, which combines the best features of BLSTM and CNN to process data more effectively, is demonstrated by this steady but marginal improvement. The outcomes highlight the hybrid model's potential for faster service delivery, especially for larger task sizes, which makes it a better option than the current models.

Figure 12 displays the accuracy analysis for various models, including RL, CPNs, GCNs, GNNs, RNNs, VAEs, GANs, and the suggested hybrid model, for varying numbers of components per task (from 100 to 5000). The suggested model routinely outperforms the top-performing current models by 4.8%. This steady improvement shows the hybrid model's superior efficiency, which makes it more appropriate for challenging tasks requiring greater accuracy. The suggested model's ability to consistently improve performance demonstrates how well it works to improve prediction accuracy.

The Offloading Decision-Making loss results for the suggested and current methods are shown in Fig. 13, along with the number of epochs. The procedure entails creating epochs ranging from 0 to 100 and allocating fictitious loss values to every model that exhibits notable fluctuations throughout these epochs. At this stage, we have extracted the loss results for each model by concentrating on the last epoch (epoch 100). Because of its advanced workload offloading and resource allocation techniques, along with its optimized architecture, the proposed model performs better than existing methods.

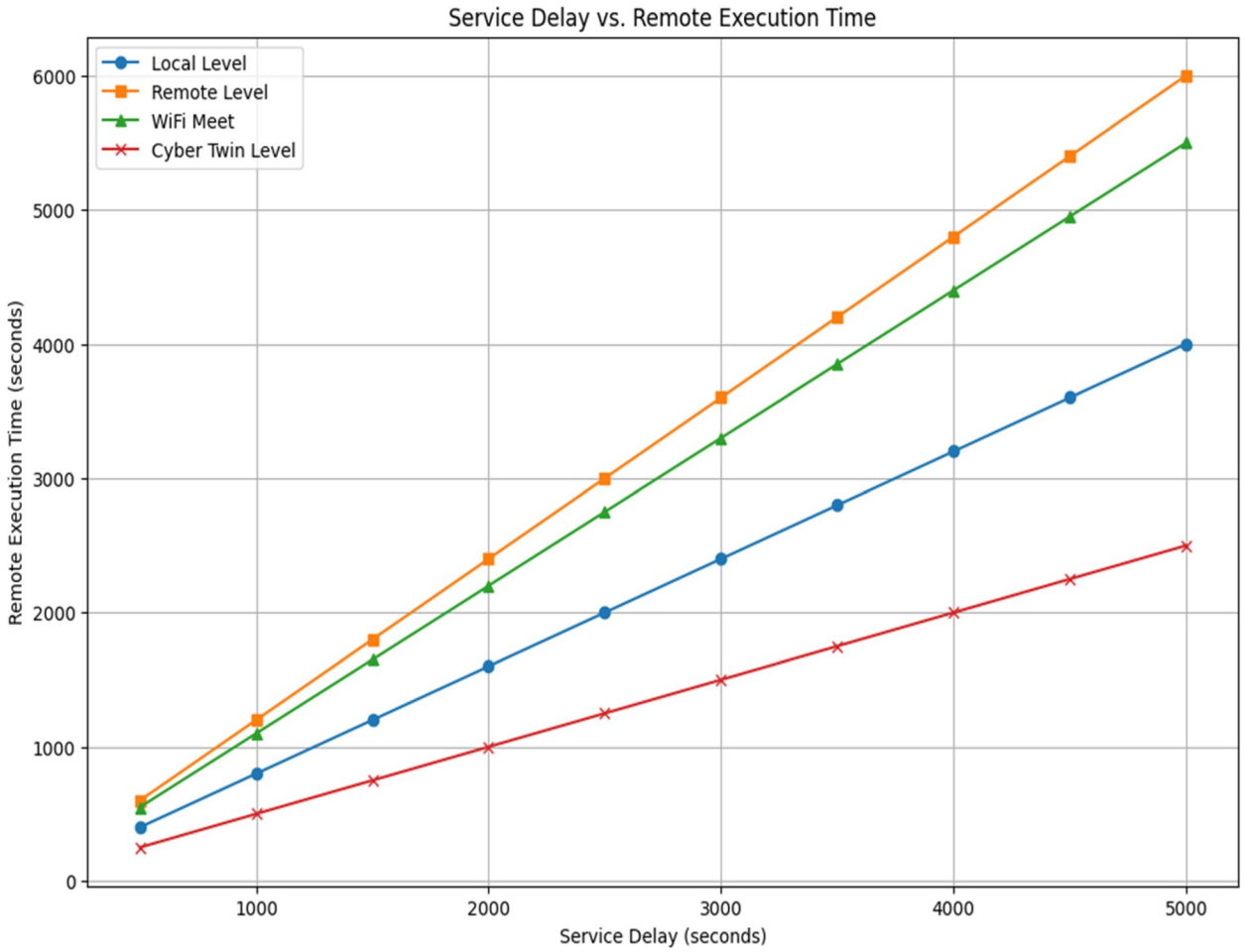

Fig. 9 Service delay vs. remote execution

The suggested model's capacity to effectively manage resources and cut energy consumption is improved by integrating deep learning with a Cybertwin-driven methodology, which lowers loss results. This enhancement demonstrates how well the suggested approach works in Mobile Edge Computing networks to optimize offloading choices.

Tradeoff analysis

To measure the performance of the proposed model, we have conducted a trade-off analysis between delay and total energy consumption concerning various task allocation strategies [38–41]. Table 4 presents the experimental results for tradeoff analysis for the proposed model for various task allocation strategies.

Our suggested approach has a definite trade-off between time and energy usage, as illustrated in Table 4. Increased energy use tends to be necessary to reduce the delay and vice versa. Table 4 presents the experimental findings from using several task allocation techniques as a component of the trade-off evaluation of the Proposed Model. Table 4 displays the differences in the percentage of jobs completed locally vs. offloaded jobs and how these decisions affect average delay and overall energy use. The average latency decreases as activities are offloaded, although energy consumption increases, indicating the trade-off involved in network optimization for Mobile Edge Computing. On the other hand, prioritizing local execution lowers energy usage but can result in higher delays. Considering their individual needs and goals allows participants to make well-informed decisions.

Conclusion and future work

The research paper presents a new advanced method that combines deep hybrid transfer learning with a Cybertwin-driven resource allocation model to achieve energy-efficient workload offloading in Mobile Edge Computing networks. The research is split into two

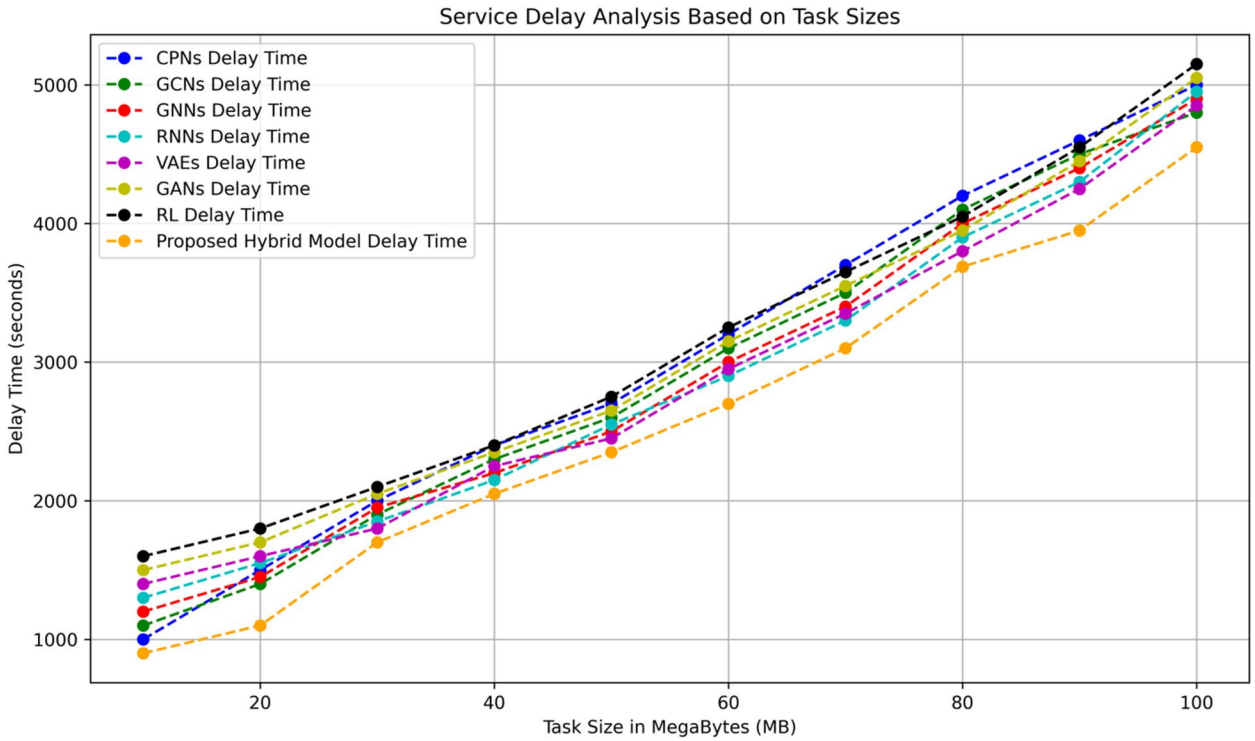

Fig. 10 Energy consumption analysis

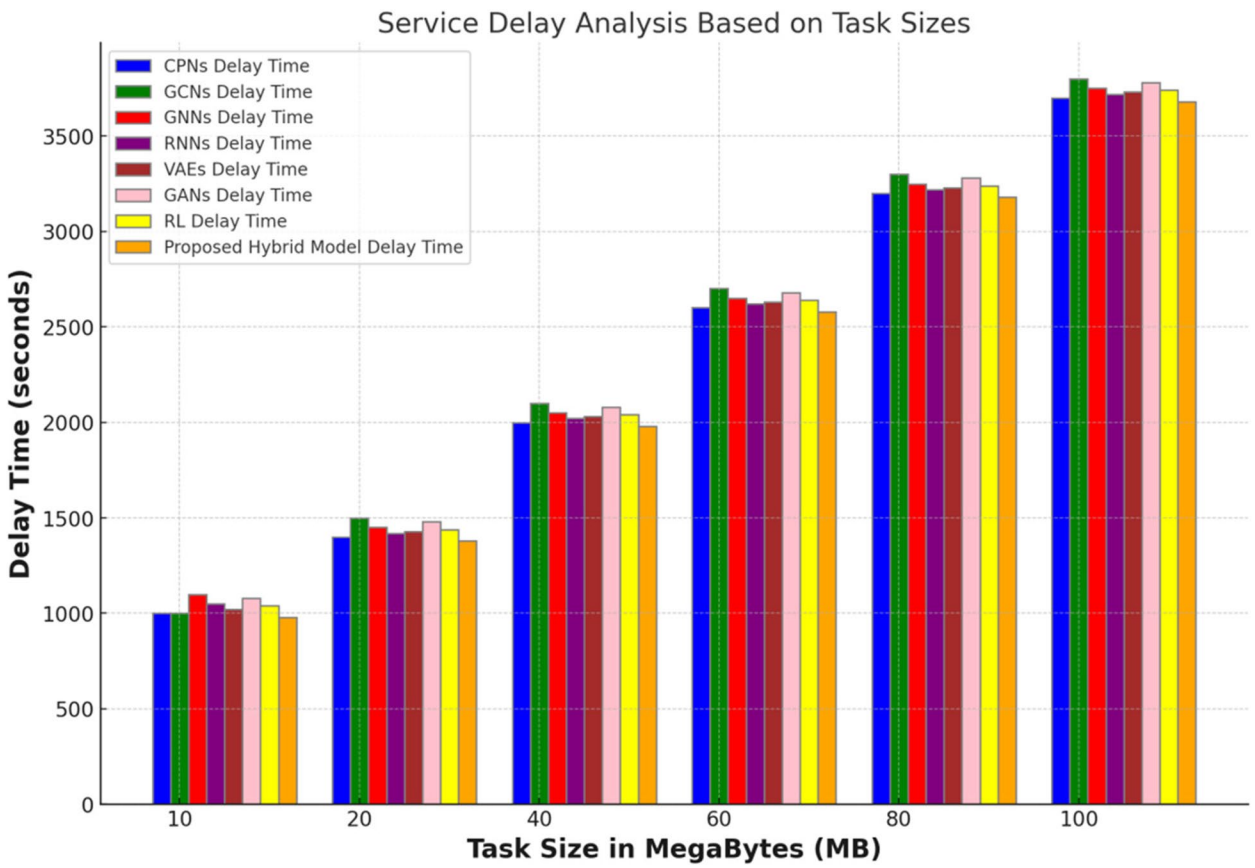

Fig. 11 Service delay analysis based on task size

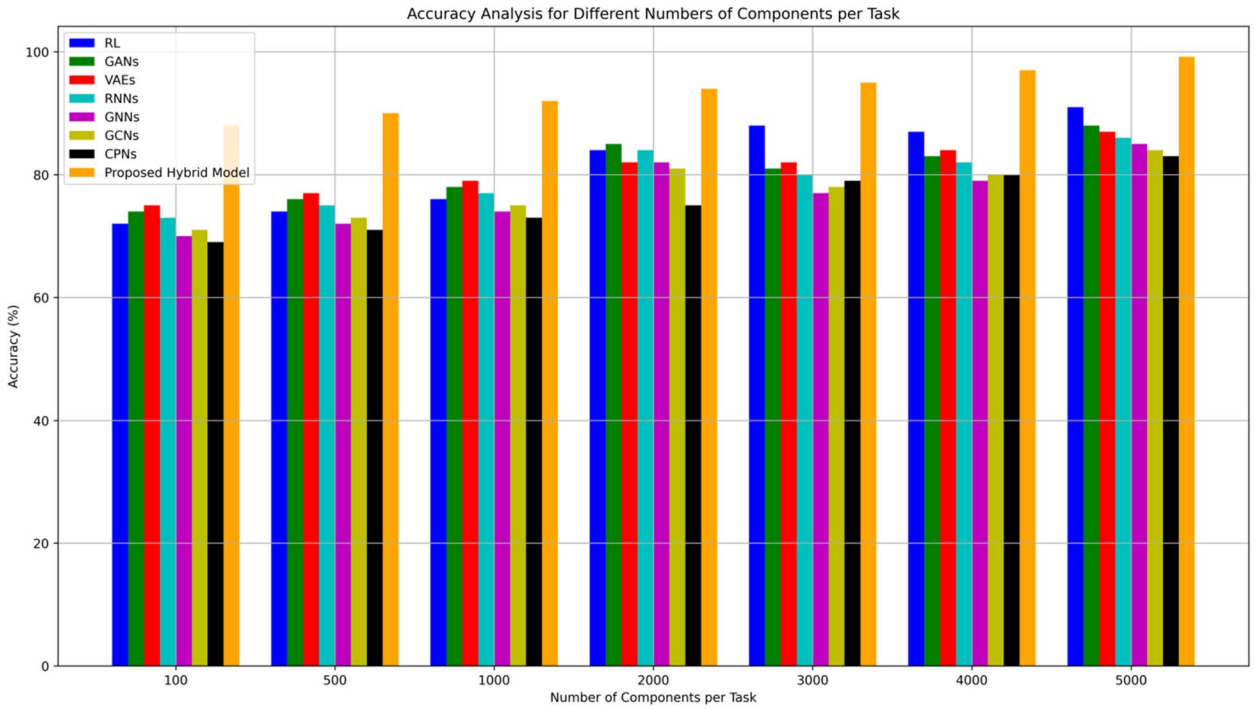

Fig. 12 Accuracy analysis for different numbers of components per task

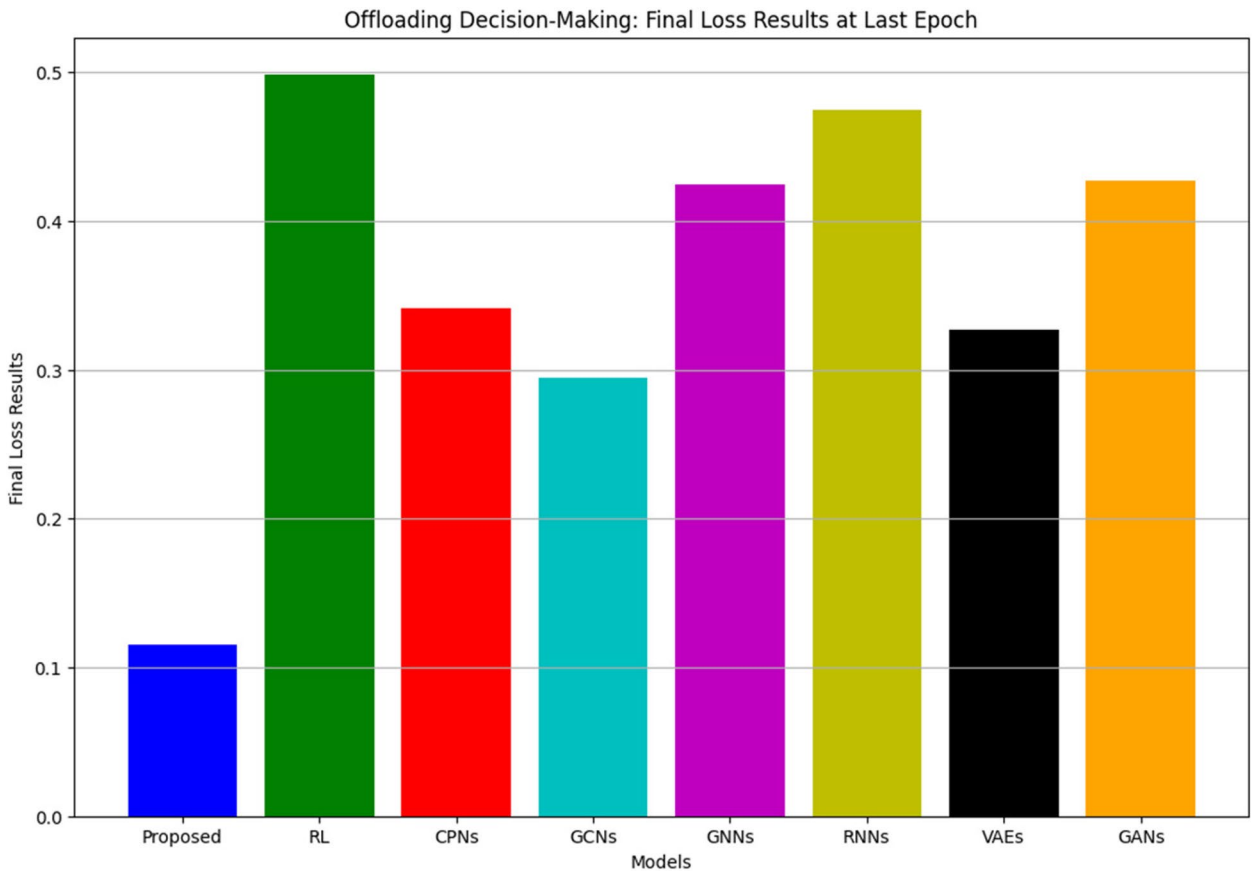

Fig. 13 Offloading decision-making (loss %) vs no of epoch results for existing and proposed

Table 4 Experimental results for tradeoff analysis for the proposed model

Task Allocation Strategy	Average delay (ms)	Total Energy Consumption (joules)
100% Local Execution	50	1200
80% Local, 20% Offloaded	35	1400
50% Local, 50% Offloaded	25	1600
20% Local, 80% Offloaded	15	1800
100% Offloaded	10	2000

scenarios, each focusing on distinct characteristics of Cybertwin-driven MEC networks. The first scenario aims to improve Cybertwin-driven MEC performance by using a Cybertwin control panel to handle communication delays. The second scenario utilizes a hybrid transfer learning approach to tackle energy-efficient work-offloading challenges in the Cybertwin MEC environment. We utilize supervised computational intelligence to distribute work across several parts and incorporate an offload mechanism with low power consumption and notable delay overhead.

Our method notably involves CNN-LSTM architecture with transfer learning. The cost function considers different energy needs and time delays related to communication, execution, and job completion. This approach accurately replicates the virtual world, making it ideal for realistic scenarios. The suggested method employs a trained CNN-LSTM model to identify the most cost-effective offload mechanism and work division compared to existing methods, i.e., RL, CPNs, GCNs, GNNs, RNNs, VAEs, and GANs. This leads to decreased energy usage and operational interruptions, with a model precision exceeding 10%.

Nonetheless, our methodology is subject to certain constraints, including the inability to identify the optimal number of components for every assignment and the limitation to programs that necessitate component operations in succession. Future research might create a robust Cybertwin model for Mobile Edge Computing and cloud settings to enhance resource utilization and task offloading mechanisms. Future research might investigate how to calculate the optimal number of characteristics per assignment and create ways to offload tasks in different applications to improve the effectiveness of Mobile Edge Computing networks.

Abbreviations

CT	Cybertwin
MEC	Mobile Edge Computing
IoT	Internet of Things
VM	Virtual Machine
CNN-LSTM-TL	Convolutional Neural Network-Long Short-Term Memory-Transfer Learning
PPA	Partial Partitioning Approach
BN	Batch Normalization
UEs	User equipment
MES	Mobile Edge Server
MDP	Markov Decision Process
TL	Transfer Learning
MEC	Mobile Edge Computing
CC	Cloud Computing
UEs	User equipments
VMs	Virtual Machines
TD	Transmission Delay
PD	Propagation delay
QD	Queuing Delay
PrD	Processing delay
EC	Energy Consumption

Authors' contributions

The authors confirm their contribution to the paper as follows: study conception and design: UKL, SD, NF, and SS; data collection: RA and MA; analysis and interpretation of results: UKL and AMB; draft manuscript preparation: UKL, SA and SD. All authors reviewed the results and approved the final version of the manuscript. Author names abbreviation's are as follows: Umesh Kumar Lilhore (UKL), Sarita Simaiya (SS), Surjeet Dalal (SD), Neetu Faujdar (NF), Roobaea Alroobaea (RA), Majed Alsafyani (MA), Abdullah M. Baqasah (AMB), Sultan Algarni (SA).

Funding

The author extends their appreciation to Taif University, Saudi Arabia, for supporting this work through project number (TU-DSPP-2024-17).

Availability of data and materials

Available on personal request.

Declarations

Competing interests

The authors declare no competing interests.

Author details

[1]School of Computing Science and Engineering, Galgotias University, Greater Noida, UP, India. [2]Arba Minch University, Arba Minch, Ethiopia. [3]Amity School of Engineering and Technology, Amity University Haryana, Gurgaon, Haryana, India. [4]Department of Computer Engineering and Applications, GLA University, Mathura 281406, India. [5]Department of Computer Science, College of Computers and Information Technology, Taif University, P. O. Box 11099, Taif 21944, Saudi Arabia. [6]Department of Computer Science, College of Computers and Information Technology, Taif University, P. O. Box 11099, Taif 21944, Saudi Arabia. [7]Department of Information Technology, College of Computers and Information Technology, Taif University, Taif 21974, Saudi Arabia. [8]Department of Information Systems, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah 21589, Saudi Arabia.

Received: 2 March 2024 Accepted: 21 July 2024
Published online: 03 August 2024

References

1. Yu C, Quan W, Gao D, Zhang Y, Liu K, Wu W, Zhang H, Shen X (2021) Reliable cyber twin-driven concurrent multipath transfer with deep reinforcement learning. IEEE Internet Things J 8(22):16207–16218
2. Juneja S, Gahlan M, Dhiman G, Kautish S (2021) Futuristic Cybertwin architecture for 6G technology to support internet of everything. Sci Program 2021:1–7
3. Rodrigues TK, Liu J, Kato N (2021) Application of cyber twin for offloading in mobile multi-access edge computing for 6G networks. IEEE Internet Things J 8(22):16231–16242
4. Violos J, Pagoulatou T, Tsanakas S, Tserpes K, Varvarigou T (2021) Predicting Resource Usage in Edge Computing Infrastructures with CNN and a Hybrid Bayesian Particle Swarm Hyper-parameter Optimization Model. https://doi.org/10.1007/978-3-030-80126-7_40.

5. Du R, Liu C, Gao Y, Hao P, Wang Z (2022) Collaborative cloud-edge-end task offloading in NOMA-enabled mobile edge computing using deep learning. J Grid Comput 20(2):14
6. Song S, Ma S, Zhao J, Yang F, Zhai L (2022) Cost-efficient multi-service task offloading scheduling for mobile edge computing. Applied Intelligence 7(3):1–13
7. Qian Y, Xu J, Zhu S, Xu W, Fan L, Karagiannidis GK (2022) Learning to optimize resource assignment for task offloading in mobile edge computing. IEEE Commun Lett 26(6):1303–1307
8. Li Z, Qian Y, Tang F, Zhao M, Zhu Y (2022) H-BILSTM: a novel bidirectional long short term memory network based intelligent early warning scheme in mobile edge computing (MEC). IEEE Trans Emerg Top Comput 11(1):253–64
9. Zeng X, Zhang X, Yang S, Shi Z, Chi C (2021) Gait-based implicit authentication using edge computing and deep learning for mobile devices. Sensors 21(13):4592
10. Liu L, Zhao M, Yu M, Jan MA, Lan D, Taherkordi A (2022) Mobility-aware multi-hop task offloading for autonomous driving in vehicular edge computing and networks. IEEE Trans Intell Transp Syst 24(2):2169–82
11. Xu H, Wu J, Li J, Lin X (2021) Deep-reinforcement-learning-based cybertwin architecture for 6G IIoT: an integrated design of control, communication, and computing. IEEE Internet Things J 8(22):16337–16348
12. Coffen B, Mahmud MS (2021) Tinydl: Edge computing and deep learning based real-time hand gesture recognition using wearable sensor. In 2020 IEEE International Conference on E-health Networking, Application & Services (HEALTHCOM). IEEE, p 1–6
13. Singh S, Sulthana R, Shewale T, Chamola V, Benslimane A, Sikdar B (2021) Machine-learning-assisted security and privacy provisioning for edge computing: a survey. IEEE Internet Things J 9(1):236–260
14. Hou W, Wen H, Song H, Lei W, Zhang W (2021) Multiagent deep reinforcement learning for task offloading and resource allocation in cyber twin-based networks. IEEE Internet Things J 8(22):16256–16268
15. Guan Y, Lu R, Zheng Y, Zhang S, Shao J, Wei G (2021) Toward privacy-preserving cybertwin-based spatiotemporal keyword query for ITS in 6G era. IEEE Internet Things J 8(22):16243–16255
16. Chen M, Hao Y (2018) Task offloading for mobile edge computing in software-defined ultra-dense network. IEEE J Sel Areas Commun 36(3):587–597
17. Chen Y, Gu W, Li K (2022) Dynamic task offloading for internet of things in mobile edge computing via deep reinforcement learning. International Journal of Communication Systems 13(4):e5154
18. Adhikari M, Munusamy A, Kumar N, Srirama SN (2021) Cybertwin-driven resource provisioning for IoE applications at 6G-enabled edge networks. IEEE Trans Industr Inform 18(7):4850–4858
19. Zhang Y, Chen C, Liu L, Lan D, Jiang H, Wan S (2022) Aerial edge computing on orbit: a task offloading and allocation scheme. IEEE Trans Netw Sci Eng 10(1):275–85
20. Xu X, Jiang Q, Zhang P, Cao X, Khosravi MR, Alex LT, Qi L, Dou W (2022) Game theory for distributed iov task offloading with fuzzy neural network in edge computing. IEEE Trans Fuzzy Syst 30(11):4593–4604
21. Naouri A, Wu H, Nouri NA, Dhelim S, Ning H (2021) A novel framework for mobile-edge computing by optimizing task offloading. IEEE Internet Things J 8(16):13065–13076
22. Yang L, Zhang H, Li M, Guo J, Ji H (2018) Mobile edge computing empowered energy-efficient Task offloading in 5G. IEEE Trans Veh Technol 67(7):6398–6409
23. Sun Y, Guo X, Song J, Zhou S, Jiang Z, Liu X, Niu Z (2019) Adaptive learning-based task offloading for vehicular edge computing systems. IEEE Trans Veh Technol 68(4):3061–3074
24. Alfakih T, Hassan MM, Gumaei A, Savaglio C, Fortino G (2020) Task offloading and resource allocation for mobile edge computing by deep reinforcement learning based on SARSA. IEEE Access 8:54074–54084
25. Khanna A, Sah A, Choudhury T (2020) "Intelligent mobile edge computing: A deep learning based approach." In Advances in Computing and Data Sciences: 4th International Conference, ICACDS 2020, Valletta, Malta, April 24–25, 2020, Revised Selected Papers 4. Springer, Singapore, p 107–116
26. Xue X, Shanmugam R, Palanisamy S, Khalaf OI, Selvaraj D, Abdulsahib GM (2023) A hybrid cross layer with Harris-hawk-optimization-based efficient routing for wireless sensor networks. Symmetry 15:438. https://doi.org/10.3390/sym15020438
27. Zhao Z, Zhou W, Deng D, Xia J, Fan L (2020) Intelligent mobile edge computing with pricing in internet of things. IEEE Access 8:37727–37735
28. Zhang J, Hu X, Ning Z, Ngai EC-H, Zhou L, Wei J, Cheng J, Hu B (2017) Energy-latency trade-off for energy-aware offloading in mobile edge computing networks. IEEE Internet Things J 5(4):2633–2645
29. Khalaf OI, Abdulsahib GM, Sabbar BM (2020) Optimization of wireless sensor network coverage using the Bee Algorithm. J Inf Sci Eng 36(2):377–386
30. Akhila SR, Alotaibi Y, Khalaf OI, Alghamdi S (2022) Authentication and resource allocation strategies during handoff for 5G IoVs using deep learning. Energies 15(6):2006–2018
31. Trivedi NK, Anand A, Lilhore UK, Guleria K. (2022) "Deep learning applications on edge computing." In Machine Learning for Edge Computing. CRC Press, p 143–168
32. Mustafa E, Shuja J, uz Zaman SK, Jehangiri AI, Din S, Rehman F, Mustafa S, Maqsood T, Khan AN (2022) Joint wireless power transfer and task offloading in mobile edge computing: a survey. Clust Comput 25(4):2429–2448
33. Chen Y, Zhao F, Lu Y, Chen X (2022) Dynamic task offloading for mobile edge computing with hybrid energy supply. Tsinghua Sci Technol 28(3):421–432
34. MEC (trace) dataset. Available at https://github.com/hetianzhang/Edge-DataSet. Accessed 20 June 2023
35. Hamdi AM, Hussain FK, Hussain OK (2022) Task offloading in vehicular fog computing: state-of-the-art and open issues. Future Gener Comput Syst 133:201–12
36. Lilhore UK, Imoize AL, Li CT, Simaiya S, Pani SK, Goyal N, Kumar A, Lee CC (2022) Design and implementation of an ML and IoT-based adaptive traffic-management system for smart cities. Sensors 22(8):2908
37. Dalal S, Manoharan P, Lilhore UK, Seth B, Simaiya S, Hamdi M, Raahemifar K (2023) Extremely boosted neural network for more accurate multi-stage Cyber attack prediction in cloud computing environment. J Cloud Comput 12(1):1–22
38. Sufyan F, Banerjee A (2023) Computation offloading for smart devices in fog-cloud queuing system. IETE J Res 69(3):1509–1521
39. Sufyan F, Banerjee A (2020) Computation offloading for distributed mobile edge computing network: a multiobjective approach. IEEE Access 8:149915–149930
40. Sufyan F, Banerjee A (2019) Comparative analysis of network libraries for offloading efficiency in mobile cloud environment. Int J Adv Comput Sci Appl 10(2):574–584
41. Banerjee A, Sufyan F, Nayel MS, Sagar S (2018) Centralized framework for controlling heterogeneous appliances in a smart home environment. In: International conference on information and computer technologies (ICICT), Dekalb, IL, USA. pp 78–82

Publisher's Note