International Journal of Electrical and Computer Engineering (IJECE)
Vol. 15, No. 2, April 2025, pp. 1924~1932
ISSN: 2088-8708, DOI: 10.11591/ijece.v15i2.pp1924-1932

Next-generation offloading using hybrid deep learning network for adaptive mobile edge computing

P. Anusha, V. Mary Amala Bai
Department of Information Technology, Noorul Islam Centre for Higher Education, Kanyakumari, India

Article Info

Article history:

Received May 10, 2024
Revised Sep 19, 2024
Accepted Oct 1, 2024

Keywords:

Average energy consumption
Computation offloading
Deep learning
Deep reinforcement learning
Long short-term memory
Mobile devices
Mobile edge computing

ABSTRACT

Deploying mobile application tasks that require a lot of computing and are time-sensitive to distant cloud-based data centers has become a popular method of working around the limitations of mobile devices (MDs). Deep reinforcement learning (DRL) techniques for offloading in mobile edge computing (MEC) environments struggle to adapt to new situations due to low sample efficiency for each new context. To address these issues, a novel computational offloading in mobile edge computing (COOL-MEC) algorithm has been proposed that combines the benefits of attention modules and bi-directional long short-term memory. This algorithm improves server resource utilization by lowering the cost of assimilating processing latency, processing energy consumption, and task throughput of latency-sensitive tasks. The experiment's findings show that, when used as intended, the recommended COOL-MEC algorithm minimizes energy consumption. When compared to the current deep convolutional attention reinforcement learning with adaptive reward policy (DCARL-ARP) and DRL techniques, the energy consumption of the proposed COOL-MEC is decreased by 0.06% and 0.08%, respectively. The average time per channel utilized for the execution of the proposed COOL-MEC also decreased by 0.051% and 0.054% when compared with existing DCARL-ARP and DRL methods respectively.

Corresponding Author:

P. Anusha
Department of Information Technology, Noorul Islam Centre for Higher Education
Kumaracoil, Thucklay, Kanyakumari, Tamil Nadu, 629 180, India
Email: mail2meanu82@gmail.com

1. INTRODUCTION

The internet of things (IoT) smart mobile device (SMD) is a powerful computing device that enables smart networking [1], [2]. An emerging technology called the IoT enables real objects, including cars and home appliances, to interact and even converse with one another [3]–[5]. Simultaneously, SMDs are frequently used to implement applications like virtual reality and interactive online gaming that demand supercomputing capacity, extremely low latency, and perpetual access rights [6], [7]. On the other hand, given that SMDs are portable, their small size results in higher energy consumption, lower processing power, and smaller storage capacity [8]. Many apps are exceedingly difficult to deploy because of this SMD constraint [9]. To overcome these restrictions, SMD uses a wireless network to connect to a remote cloud and moves computational functions to the cloud [10]. However, most sizable data centers housing cloud computing resources remotely are situated at a considerable distance from the bulk of clients [11]. Because of this, SMD will take longer to unload and consume more energy during the process [12].

Mobile edge computing, or MEC, has been developed as a recent solution to the previously described problems. Edge servers, frequently referred to as compute nodes, are dispersed throughout the network when using MEC [13]. Edge servers are thus situated closer to users than independent cloud servers and are capable of processing larger volumes of data than SMD servers [14]. MEC's features enable it to offer consumers services with minimal energy use and quick response times [15]. Edge servers are typically installed at base stations (BS) and access points (AP). Subsequently, SMDs will transfer the computational duties or associated data to the APs. Once at the APs, the edge server installed on the Aps will handle the computational jobs [16].

A wide range of problems have been learned and optimized through the application of deep learning (DL) [17]. Still, the majority of DL techniques need tagged historical data. Meanwhile, a large quantity of human labor is needed to label the training data. Reinforcement learning (RL) occurs by interaction with MEC environments [18]. Computational offloading in mobile edge computing (COOL-MEC) techniques can be used to enable efficient task offloading, unmanned aerial vehicle (UAV) control, and resource allocation hence reducing overall energy usage. The following sums up this paper's main contributions: i) Initially the input task from the user will be given to the parser the parser will parse the task to the Local trainer and offloading; ii) The offloading scheduler uses deep reinforcement learning for providing the schedule for executing the task; iii) The offloading scheduler will provide schedule to the local execution trainer will offload the remote execution service in edge level; and iv) The efficiency of the proposed COOL-MEC is demonstrated through evaluation criteria such as average power consumption, average data queue length, and performance analysis.

The remaining sections of the research is arranged as follows. The section 2 holds the related works. Section 3 holds the effective task offloading strategy. The simulation findings and discussions are detailed in section 4. Section 5 concludes the research with future scope.

2. LITERATURE SURVEY

Numerous studies have delegated the tasks with a range of tactics in recent years. A few of the existing assessment methods are covered in the section that follows, along with some of their drawbacks. These approaches have aimed to improve efficiency, scalability, and accuracy across various domains. A few of the existing assessment methods are covered in the section that follows, along with some of their drawbacks, and highlighting the challenges.

Yang [19] suggested an optimization problem for the work with the least average completion time while resources are scarce. Simulation studies are employed to evaluate and compare the effects of different parameters on task execution efficiency. Simulation findings show that compared to existing approaches, the recommended technique can effectively minimize system overhead and shorten job execution time.

Zhang *et al.* [20] suggested the computing power and battery life of SMDs limit the computationally demanding and delay-sensitive applications that are constantly evolving, despite being suggested by new computer technologies. The experimental results show that the deep deterministic policy gradient (DDPG)-based offloading technique achieves a long-term improvement of at least 19% over previous systems, while still achieving ultra-low latency, frequent SMD server movement, and efficient server utilization. Zhang *et al.* [21] suggested a mobile application that presents in edge cloud networks during task offloading and resource allocation because of fluctuating wireless channels, network connection workloads, and to manage unpredictable resource availability. The outcomes of the simulation confirm that the suggested resource scheduling and task offloading strategies outperform the baseline systems.

Ebrahim *et al.* [22] suggested a deep reinforcement learning (DRL) based approach to enhance the MEC parameter offloading procedure for the IoT. The best offloading choice can be found using this technique. The simulation's findings indicate that the proposed model outperforms both actor-critic (AC) and deep Q-networks (DQNs), indicating that it might be a useful tool for lowering latency and energy usage in the MEC system. Nguyen *et al.* [23] suggested a cooperative computing architecture to shift online computational activities to parked vehicles (PVs) in an efficient manner during business hours. We suggest using advanced features of Kubernetes-based container orchestration to ensure service continuity. Our proposed computing architecture improves the average work offloading cost by at least 40% and boosts the potential of accepting online assignments significantly, according to extensive simulation results.

Anusha and Bai [24] suggested a unique method known as deep convolutional attention reinforcement learning with adaptive reward policy (DCARL-ARP), which combines the feature map attention mechanism with deep convolution and Lyapunov optimization. The findings of the experimental evaluation show that an effective reduction of 50% can be achieved in the average data queue length and an effective reduction of 0.02% in the average execution energy consumption. Gao *et al.* [25] suggested a novel technique that integrates the Markov decision problem, DDPG, and DRL for work offloading in MEC. Experiments were conducted and the outcomes show that, in comparison to the other three baseline techniques, the recommended strategy can improve performance. Because of its great scalability, ability to manage expansive and complicated environments, and suitability for practical implementation, it may be applied to a broad variety of task offloading and MEC applications.

Several disadvantages to the job offloading strategies mentioned above include high latency, low resource utilization, energy use, and network congestion. An innovative COOl-MEC approach has been proposed and described to address these obstacles. The comparison of proposed and existing methods with demerits and merits are shown in Table 1.

Table 1. Comparison of existing approach with advantages and disadvantages

Author	Method	Advantage	Disadvantage
Yang [19]	Optimization problem for least average completion time	This approach can efficiently allocate computing tasks	Increased overhead in task allocation for vehicles
Zhang *et al.* [20]	Computing power and battery for smart mobile devices	Faster processing of multiple tasks	Increased complexity
Zhang *et al.* [21]	Job offloading in hybrid edge-cloud networks	Efficient task offloading and resource scheduling in hybrid edge-cloud networks can improve performance	Increased network latency
Ebrahim *et al.* [22]	Offloading procedure in MEC based on DLR	Adapt to complex, dynamic environments for efficient resource utilization	It may require large amounts of training data and computational resources for UAV-assisted MEC
Nguyen *et al.* [23]	Cooperative computing architecture	Reducing the workload on the network and improving efficiency	Own computational power can disrupt task offloading
Anusha and Bai [24]	Deep convolution and Lyapunov optimization	This approach combines deep learning for feature extraction, reinforcement learning for decision-making	The complex nature of the approach might lead to high computational overhead
Gao *et al.* [25]	DRL for work offloading in MEC	It can dynamically adapt to changing network conditions	computationally expensive

3. PROPOSED COMPUTATIONAL OFFLOADING IN MOBILE EDGE COMPUTING

The system receives tasks from users, which can be trained globally on a distant server or locally on the user's device. Whether a task is executed locally on the device or routed to a cloud server for processing is determined by the offload scheduler. This choice takes into account variables such as network latency, task difficulty, and resource availability. Virtual machines are used by the cloud server to provide separate environments for the execution of tasks that have been offloaded. In the real world, offloading complex and time-sensitive mobile app tasks to distant cloud data centers is a common practice, but it can be inefficient due to network latency. A new algorithm called COOL-MEC has been proposed to address this by using MEC with better learning techniques. This could lead to significant improvements in battery life and responsiveness for mobile apps that rely on a lot of processing power. The MEC host's network, storage, and processing resources are provided by the virtualization infrastructure and MEC platform combined. What is provided by the MEC platform is edge services in addition to traffic management, which includes domain name processing and traffic rule control. The global training service, local trainer, offload scheduler, analyzer, and remote execution service are the five primary COOL-MEC modules. The MEC system allows for the autonomous deployment of each of these modules at the user and device levels. The architecture of the remote cloud server for mobile edge computing is depicted in Figure 1.

3.1. User level

The parser's job is to translate user tasks into user-level directed acyclic graphs (DAGs). Using a local transport device, the trainer uses the analyzed DAG as training data, downloading and loading policy network parameters to and from the MEC server. This kind of training is called “inner loop” training. The offloading scheduler will use the trained policy network to infer whether to offload a task based on policy network inference. The local executor will finish the locally scheduled tasks once the DAG has determined on all of its actions, and the MEC host will get the offloaded jobs.

3.2. Offloading scheduler

In the proposed study, the model is utilized to decide when to offload jobs in conjunction with the DRL model. The model is intended to simulate a multi-terminal, multi-edge network. $M = \{1, 2 \ldots M\}$ indicates that there are a number of terminal layer devices. Task and computation queues are present on every terminal device, also known as mobile devices (MD). The computation queue handles activities that are performed locally, whereas the task queue holds jobs that need to be offloaded. An additional set of edge layer servers is denoted by $N = \{1, 2 \ldots N\}$. Each edge server has multiple computation queues that can be used to parallelize the computation of tasks that are offloaded from MDs. As depicted in Figure 2, the edge system is capable of processing data from MDs and storing the processed records.

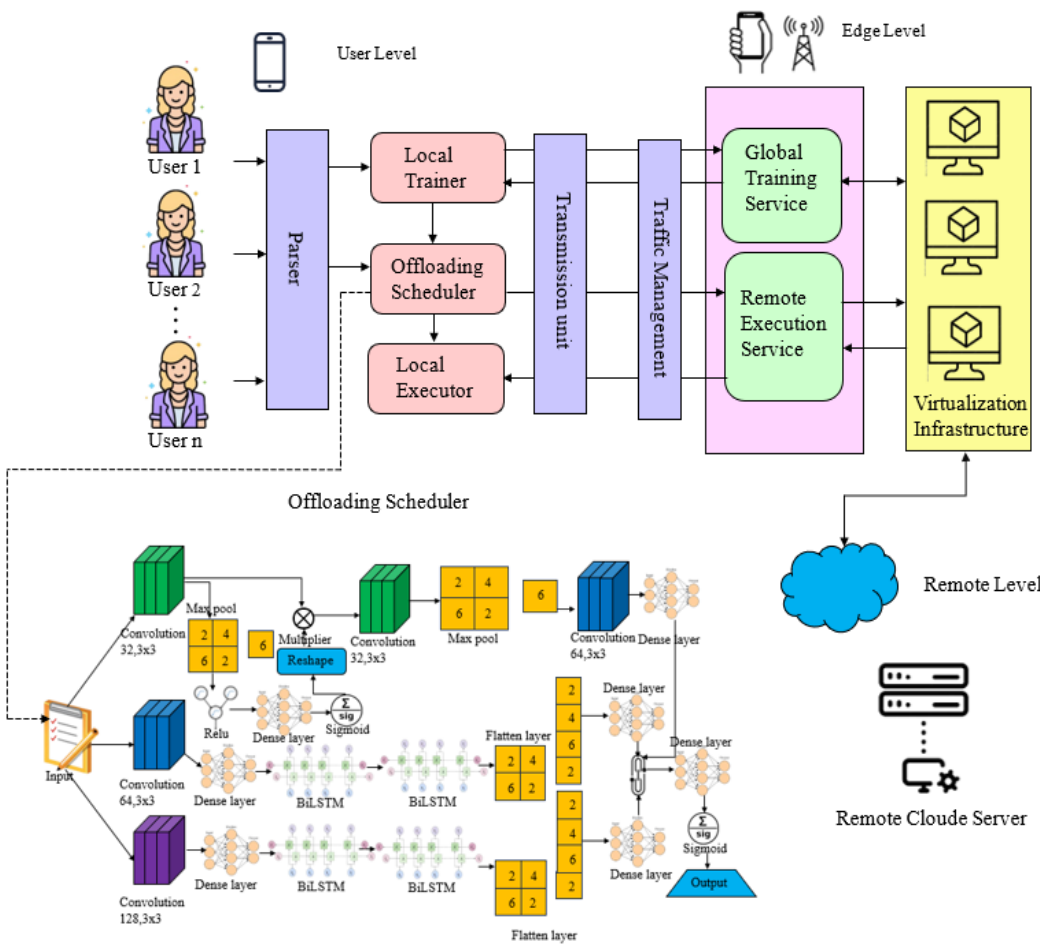

Figure 1. Architecture for the proposed COOL-MEC

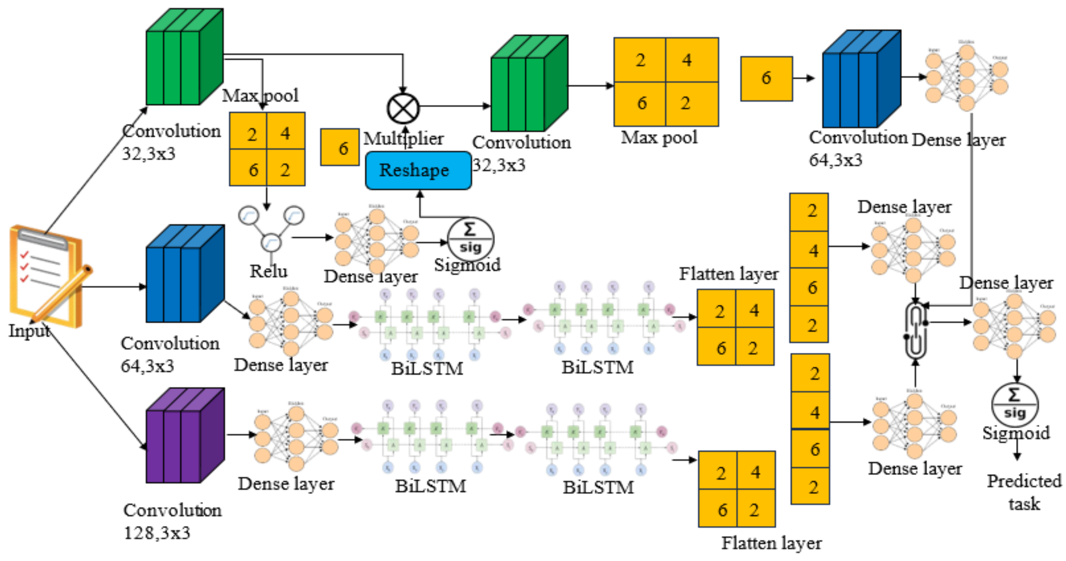

Figure 2. Architecture for task offloading

3.2.1. Deep convolution LSTM attention reinforcement learning

Deep convolution long short-term memory (LSTM) attention reinforcement learning reward-based learning for complex tasks. Deep convolution LSTM attention reinforcement learning is an approach that tackles sequential decision-making by combining feature extraction, long-term memory, and selective focus for better performance. Using LSTM, the feature information learned belongs to upcoming task can be forecasted. The reinforcement learning decision model, which is intended to create an offloading strategy for the anticipated task, is then fed this projected data. The decision to offload the job is made immediately when the actual task is delivered, provided that the difference between the actual and anticipated tasks is within a reasonable range. When an error beyond the permitted range, the actual work employs the decision model to determine the outcome.

By using task information prediction, it is possible to predict the computation node that the task is aiming at, which helps to minimize the response and waiting delays that the task encounters in the system.

$$Forget\ gate\ (f_o)\ f_o = \sigma\ (w_f * \left[h_{(t-1),x_t}\right] + b_f) \tag{1}$$

$$Input\ gate\ (i_n)\ i_n = \sigma\left(W_i * \left[h_{(t-1),X_t}\right] + b_i\right) \tag{2}$$

$$Candidate\ cell\ (c_a)\ c_a = tanh(w_c * \left[h_{(t-1),X_t}\right] + b_c \tag{3}$$

$$Cell\ state\ (c_e)\ c_e = f_0 *\ c_{(t-1)} + i_n * c_e \tag{4}$$

$$output\ Gate\ (o_u)o_u = \sigma\ (w_o\ *)\left[h_{(t-1),X_t}\right] + b_o \tag{5}$$

$$Hidden\ state\ (h_i)h_i = o_u * tanh(c_e) \tag{6}$$

The significant of this proposed model is as follows, in channel attention mechanism incorporating bidirectional LSTM which is the recurrent neural network. When using a bidirectional LSTM, input is provided in both left-to-right and right-to-left directions. Here, every component of an input sequence has information from both the past and present. Incorporating distributed concept in objective values of binary offloading modes (multiple subset parallel mode which resulted in fast and optimal resource allocation). Based on threshold values the objective values get partition and thereby optimal resource allocation is achieved. The amount of time allocated for offloading the task and to utilize the resources is effectively determined by the optimal parameter. The optimal parameter selection is based on dual binary search algorithm. Some time it may take lot of iterations depends on the size of the active queue length. Hence in this research instead of applying binary search concept, a queue length based either search length is divided by either two or three. Also, upper bound (UB) considered to be sufficiently large value and lower bound (LB) to be 0. Here the mid value is determined by the following concept:

```
$if\ (UB - LB)/LB > 0.65\ then$
$mid - value = (upper\ bound\ (UB) - lower\ bound\ (LB))/3$
$else$
$mid - value = (upper\ bound\ (UB) - lower\ bound\ (LB))/2$
```

Normally all the task from the queue is transferred for the further process if it is not empty. In this module, each available task in the queue is divided by its maximum value to attain as the contribution weight. That is multiplied by the actual data of the queue and transformed to find the optimal solution.

3.3. Edge level

The MEC platform gains new capabilities with the addition of global training services and remote execution services modules. The entire infrastructure training procedure is carried out digitally on the MEC server, and the global training service manages the transmission and receiving of policy network parameters to and from the end user (EU). This type of training is referred to as “outside the box” training. Tasks that are offloaded from the user environment are managed by the remote execution service, which also assigns them to relevant virtual machines and returns the results to the user environment.

3.4. Remote level

Remote cloud servers provide scalable, on-demand resources that are accessible over the internet, remote cloud servers are revolutionizing computing. By using them, businesses can save a considerable amount of money and improve operational efficiency by not having to manage and invest in physical infrastructure. These servers enable companies to process massive amounts of data, quickly roll out apps, and enable smooth teamwork between geographically scattered workers. All things considered, distant cloud servers are a vital component of contemporary information technology (IT) infrastructure, allowing companies to compete and develop in a constantly changing digital market.

4. RESULT AND DISCUSSION

The proposed COOL-MEC method's experimental results are analyzed in this section. The simulation results in this section show how successful the recommended COOL-MEC approach is. The proposed system is implemented using the NS2 simulator. The experimental analysis utilized to evaluate the proposed COOL-MEC based method is presented in the accompanying Table 2. The suggested COOL-MEC method performance is compared with existing DRL and DCARL-ARP regarding average power consumption, average data queue length, and performance analysis.

Table 2. Environmental setup

Parameter	Setting
Optimization algorithm	Adam
Batch size	256 Samples
Activation function	Rectified linear unit (ReLU)
Smoothing coefficient	0.005
Update interval	1
Gradient update step	1
Learning rate	3.10^{-4}
Discount factor	0.999
Replay memory buffer size	106

4.1. Discussion

The work of the proposed COOL-MEC framework is discussed in this section. The COOL-MEC offloading scheduler is proposed to be sent remotely to a remote cloud server or at the edge level using a DL framework. In comparison to existing approaches such as DRL and DCARL-ARP, the performance of the proposed COOL-MEC framework is assessed regarding average energy consumption, average data queue length, average time per channel, average accuracy, and DL-based performance analysis. The DRL approach yields average data queue lengths of 60.69, 32.32, and 20.78 for DCARL-ARP, COOL-MEC respectively. In this comparison, the proposed COOL-MEC framework achieves a shorter average mean data queue length of 20.78 than other existing approaches which is depicted in Figure 3. In Figure 4, the proposed COOL-MEC framework obtains less energy consumption up to 0.013, compared to the existing approaches such as DRL of 0.08, and DCARL-ARP of 0.06 respectively. Figures 5 and 6 depict the training accuracy and loss of the proposed COOL-MEC framework. The training loss of the COOL-MEC framework is 0.13, which is less than that of the current DRL and DCARL-ARP models, which reach training losses of 0.31 and 0.2, respectively. In Figure 7, the average time per channel for the proposed COOL-MEC framework is compared. where the offloading process with the DRL method is added to the proposed multi-objective optimal scheduling algorithm is also evaluated with the help of a workflow simulator with different kinds of objectives as well as different offloading types of algorithms. The DL-based performance analysis of the existing methods such as convolutional neural network (CNN), probabilistic recurrent neural network (PRNN), bidirectional gated recurrent unit with CNN (Bi-GRU-CNN) and the proposed COOL-MEC framework is presented in Figure 8. The maximum accuracy attained by the proposed methods is 99.27%, while the existing CNN, simple logistics, and deep recurrent neural network (DRNN) models obtain 98.4%, 97.2%, and 96.5%. The effectiveness of memorizing using the proposed approach is 51.5% and 52.7% higher than that of existing approaches. Finally, the experimental findings show that, when used as intended, the proposed COOL-MEC algorithm minimizes energy consumption, contains short data queue length, obtains less time per channel, and achieves better accuracy in adaptive mobile edge computing.

4.2. Performance comparison

The efficacy of the suggested framework is contrasted with DRL and DCARL-ARP by analyzing the average data queue length, average energy consumption, average accuracy, and average time per channel. Figure 3 illustrates the computation of task performance at various time intervals for every task in the task queue. The proposed COOL-MEC framework, however, are 20.78 times shorter than the DCARL-ARP and DRL, average data queue length of 32.32 and 60.69. Assume an edge node has N task data queues. Every queue is assigned to a task store according to the particular types of applications it receives from various mobile nodes. Python is used to construct COOL-MEC with Keras; it has a memory capacity of 1024 and a training batch size of 128. One is selected as the interval frame. Its learning rate is 0.01 for the Adam optimizer.

Comparing the suggested COOL-MEC in Figure 4 to the current DRL and DCARL-ARP, our method reduces COOL-MEC by up to 0.013, DCARL-ARP by up to 0.06, and DRL by up to 0.08, as can be seen. It also works effectively at any time of day. The suggested method's training loss analysis is shown in Figure 5. The DCARL-ARP approach has a training loss of 0.2 in comparison to the DRL training loss (0.31), while our proposed COOL-MEC has a lower training loss of 0.13. In addition, the suggested the COOL-MEC technique reduces the entropy loss while the network is trained using data samples. Duration of each channel on average.

In comparison to DCARL-ARP (0.051) and DRL (0.054), the amount of Figure 6 utilized for execution is likewise lower for COOL-MEC (0.043). The suggested model's training time and accuracy are improved by using the predictive capabilities of LSTM. The suggested algorithm minimizes the offloading decision latency for jobs that preemptively indicate the best offloading choices during actual inference. It is important to keep in mind that simulation approaches form the basis of the experimental results of this strategy in this study.

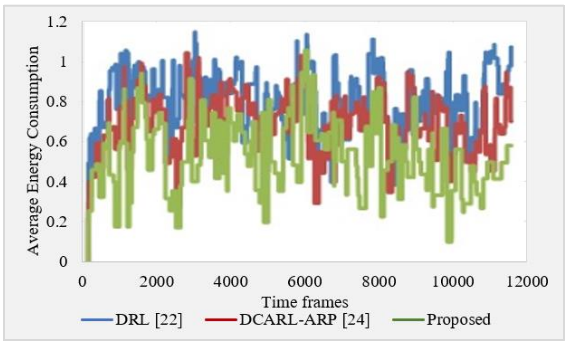

Figure 3. Average energy consumption

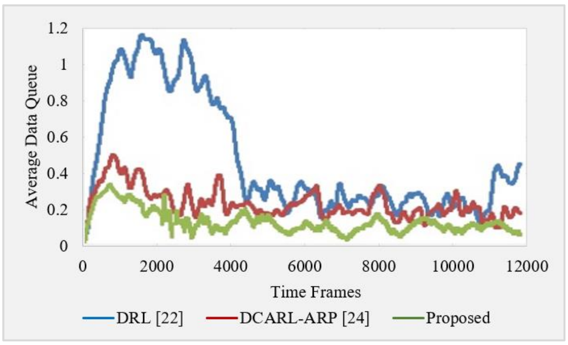

Figure 4. Average data queue length

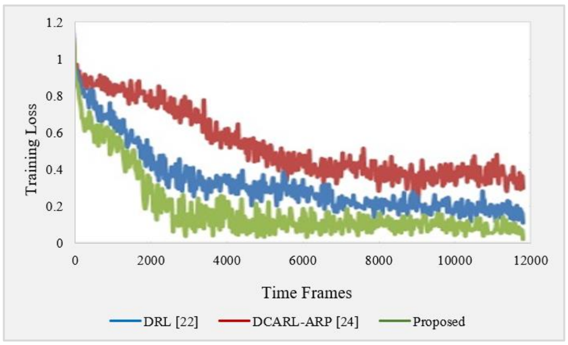

Figure 5. Training loss

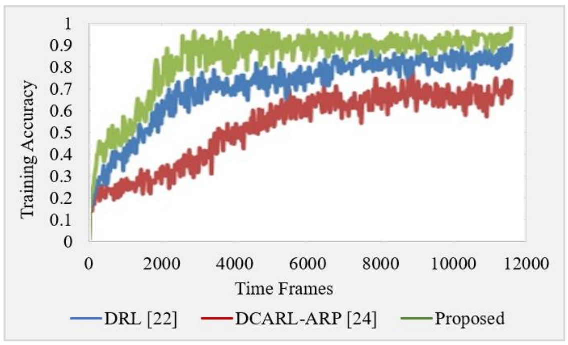

Figure 6. Accuracy training

The average energy consumption, average training loss, average time per channel, and average data queue length all show how successful and efficient the suggested COOL-MEC method is. This offloading process with DRL method is added with the proposed multi objective optimal scheduling algorithm is also evaluated with the help of workflow simulator with different kinds of objectives as well as different kinds of offloading algorithms. Average time per channel is shown in Figure 7.

An analytical comparison of existing CNN, PRNN, and Bi-GRU-CNN techniques with the proposed COOL-MEC method is shown in Figure 8. The maximum accuracy attained by the proposed methods is 99.27%, while the existing CNN, simple logistics, and DRNN models obtain 98.4%, 97.2%, and 96.5% respectively. In terms of recollection, the suggested method performs 9.48%, 5.15%, and 52.7% better than the current approaches.

The proposed methodology has a maximum F-measure of 97.4%, which is greater than the maximum F-measure of the existing methods. The proposed method outperforms existing methods, according to this comparison. The limitation of the COOL-MEC approach require high power. This limitation will be considered as a future work for extending our COOL-MEC system.

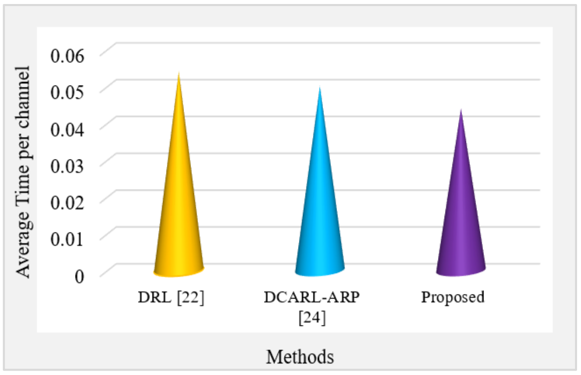

Figure 7. Average time per channel

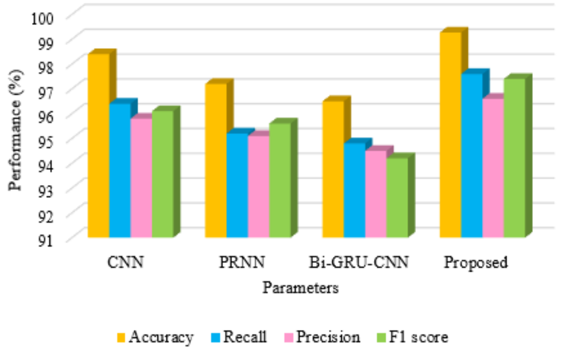

Figure 8. Comparison graph for suggested and existing technique

5. CONCLUSION

In this paper, a novel COOL-MEC technique has been proposed to optimize the distribution of tasks. There are three levels in the proposed COOL-MEC which are the device, remote, and user. Initially, the input task from the user will be given to the parser the parser will parse the task to the Local trainer and offloading. The offloading scheduler uses deep reinforcement learning to provide the schedule for executing the task. The offloading scheduler will provide a schedule to the local execution trainer will offload the remote execution service at the edge level. Evaluation criteria like average data queue length, average energy usage, performance analysis, and performance analysis have been used to show how successful the proposed COOL-MEC. The energy consumption of the COOL-MEC approach is 0.06% and 0.08% less than that of the current DCARL-ARP and DRL systems, respectively. When comparing the proposed COOL-MEC to the current DCARL-ARP and DRL techniques, the average time per channel utilized for execution dropped by 0.051% and 0.054%, respectively. Future research endeavors will entail transitioning the proposed method to real-world experimentation. Our future work is in progress to identify and optimize parameters for expediting control actions in scenarios involving partial offloading, paving the way for more efficient and responsive systems. COOL-MEC achieves lower energy consumption, explore if further reductions are possible. This could involve investigating dynamic voltage and frequency scaling techniques for the MEC server.

ACKNOWLEDGEMENTS

The author would like to express his heartfelt gratitude to the supervisor for his guidance and unwavering support during this research for his guidance and support.

REFERENCES

[1] E. Mustafa *et al.*, "Joint wireless power transfer and task offloading in mobile edge computing: a survey," *Cluster Computing*, vol. 25, no. 4, pp. 2429–2448, Aug. 2022, doi: 10.1007/s10586-021-03376-3.
[2] J. Bharathi Madavarapu, H. Islam, A. Appathurai, G. A. Safdar, N. Muthukumaran, and J. Gnanamalar, "Heterogeneous energy harvesting techniques for smart home IoT acceleration," *IEEE Access*, vol. 12, pp. 73667–73675, 2024, doi: 10.1109/ACCESS.2024.3397664.
[3] M. Amanullakhan, M. Usha, and S. Ramesh, "Intrusion detection architecture (IDA) in IoT-based security system," *International Journal of Computer and Engineering Optimization (IJCEO)*, vol. 1, no. 1.
[4] S. Zafar, N. Iftekhar, A. Yadav, A. Ahilan, S. N. Kumar, and A. Jeyam, "An IoT method for telemedicine: Lossless medical image compression using local adaptive blocks," *IEEE Sensors Journal*, vol. 22, no. 15, pp. 15345–15352, Aug. 2022, doi: 10.1109/JSEN.2022.3184423.
[5] R. R. Sathiya, S. Rajakumar, and J. Sathiamoorthy, "Secure blockchain-based deep learning approach for data transmission in IoT-enabled healthcare system," *International Journal of Computer and Engineering Optimization (IJCEO)*, vol. 1, no. 1, pp. 15–23, 2023.
[6] Z. Wang, P. Li, S. Shen, and K. Yang, "Task offloading scheduling in mobile edge computing networks," *Procedia Computer Science*, vol. 184, pp. 322–329, 2021, doi: 10.1016/j.procs.2021.03.041.
[7] R. K. Walters, E. M. Gale, J. Barnoud, D. R. Glowacki, and A. J. Mulholland, "The emerging potential of interactive virtual reality in drug discovery," *Expert Opinion on Drug Discovery*, vol. 17, no. 7, pp. 685–698, Jun. 2022, doi: 10.1080/17460441.2022.2079632.
[8] M. Zhao *et al.*, "Energy-aware task offloading and resource allocation for time-sensitive services in mobile edge computing systems," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 10, pp. 10925–10940, Oct. 2021, doi: 10.1109/TVT.2021.3108508.
[9] S. Yang, G. Lee, and L. Huang, "Deep learning-based dynamic computation task offloading for mobile edge computing networks," *Sensors*, vol. 22, no. 11, May 2022, doi: 10.3390/s22114088.

[10] M. Tang and V. W. S. Wong, "Deep reinforcement learning for task offloading in mobile edge computing systems," *IEEE Transactions on Mobile Computing*, vol. 21, no. 6, pp. 1985–1997, Jun. 2022, doi: 10.1109/TMC.2020.3036871.

[11] Y. Qian, J. Xu, S. Zhu, W. Xu, L. Fan, and G. K. Karagiannidis, "Learning to optimize resource assignment for task offloading in mobile edge computing," *IEEE Communications Letters*, vol. 26, no. 6, pp. 1303–1307, Jun. 2022, doi: 10.1109/LCOMM.2022.3159742.

[12] C. Sun *et al.*, "Task offloading for end-edge-cloud orchestrated computing in mobile networks," In *2020 IEEE Wireless Communications and Networking Conference (WCNC)*, 2020, pp. 1-6, doi: 10.1109/WCNC45663.2020.9120496.

[13] A. Naouri, H. Wu, N. A. Nouri, S. Dhelim, and H. Ning, "A novel framework for mobile-edge computing by optimizing task offloading," *IEEE Internet of Things Journal*, vol. 8, no. 16, pp. 13065–13076, Aug. 2021, doi: 10.1109/JIOT.2021.3064225.

[14] G. Qu, H. Wu, R. Li, and P. Jiao, "DMRO: A deep meta reinforcement learning-based task offloading framework for edge-cloud computing," *IEEE Transactions on Network and Service Management*, vol. 18, no. 3, pp. 3448–3459, Sep. 2021, doi: 10.1109/TNSM.2021.3087258.

[15] Z. Yu, X. Xu, and W. Zhou, "Task offloading and resource allocation strategy based on deep learning for mobile edge computing," *Computational Intelligence and Neuroscience*, vol. 2022, pp. 1–11, Aug. 2022, doi: 10.1155/2022/1427219.

[16] Z. H. Abbas *et al.*, "Computational offloading in mobile edge with comprehensive and energy efficient cost function: A deep learning approach," *Sensors*, vol. 21, no. 10, May 2021, doi: 10.3390/s21103523.

[17] L. Ale, N. Zhang, X. Fang, X. Chen, S. Wu, and L. Li, "Delay-aware and energy-efficient computation offloading in mobile-edge computing using deep reinforcement learning," *IEEE Transactions on Cognitive Communications and Networking*, vol. 7, no. 3, pp. 881–892, Sep. 2021, doi: 10.1109/TCCN.2021.3066619.

[18] X. Cheng, J. Liu, and Z. Jin, "Efficient deep learning approach for computational offloading in mobile edge computing networks," *Wireless Communications and Mobile Computing*, vol. 2022, pp. 1–12, Feb. 2022, doi: 10.1155/2022/2976141.

[19] S. Yang, "A task offloading solution for internet of vehicles using combination auction matching model based on mobile edge computing," *IEEE Access*, vol. 8, pp. 53261–53273, 2020, doi: 10.1109/ACCESS.2020.2980567.

[20] H. Zhang, Y. Yang, X. Huang, C. Fang, and P. Zhang, "Ultra-low latency multi-task offloading in mobile edge computing," *IEEE Access*, vol. 9, pp. 32569–32581, 2021, doi: 10.1109/ACCESS.2021.3061105.

[21] Q. Zhang, L. Gui, S. Zhu, and X. Lang, "Task offloading and resource scheduling in hybrid edge-cloud networks," *IEEE Access*, vol. 9, pp. 85350–85366, 2021, doi: 10.1109/ACCESS.2021.3088124.

[22] M. A. Ebrahim, G. A. Ebrahim, H. K. Mohamed, and S. O. Abdellatif, "A deep learning approach for task offloading in multi-UAV aided mobile edge computing," *IEEE Access*, vol. 10, pp. 101716–101731, 2022, doi: 10.1109/ACCESS.2022.3208584.

[23] K. Nguyen, S. Drew, C. Huang, and J. Zhou, "Parked vehicles task offloading in edge computing," *IEEE Access*, vol. 10, pp. 41592–41606, 2022, doi: 10.1109/ACCESS.2022.3167641.

[24] P. Anusha and V. M. A. Bai, "Online computation offloading via deep convolutional feature map attention reinforcement learning and adaptive rewarding policy," *Wireless Networks*, vol. 29, no. 8, pp. 3769–3779, Jul. 2023, doi: 10.1007/s11276-023-03437-y.

[25] X. Gao, M. C. Ang, and S. A. Althubiti, "Deep reinforcement learning and Markov decision problem for task offloading in mobile edge computing," *Journal of Grid Computing*, vol. 21, no. 4, Dec. 2023, doi: 10.1007/s10723-023-09708-4.

BIOGRAPHIES OF AUTHORS

P. Anusha is currently working towards her Ph.D. degree at the Noorul Islam Centre for Higher Education. She obtained her BCA and MCA from Manonmaniam Sundaranar University with a focus on computer networks. Her current research interests include performance optimization in wireless and mobile networks, applied machine learning. She can be contacted at email: mail2meanu82@gmail.com.

V. Mary Amala Bai had her B.E. from Institute of Road and Transport Technology, Erode and M.E from Noorul Islam College of Engineering, Kumaracoil and PhD from Anna University of Technology, Tirunelveli. She has experiences of working in various Engineering colleges since 1999. She has several research publications in international journals and conferences. She had attended number of seminars, workshops, conferences and faculty development programs. She has also organized workshops and symposiums. Currently she is working as an associate professor in Noorul Islam Centre for Higher Education. She is a recognized supervisor for guiding Ph.D. students. Currently 6 scholars are pursuing research under her supervision. She can be contacted at email: maryamalabai@niuniv.com.