National College of Ireland

A New Offloading Technique Based on Deep Learning for Mobile-Edge Computing

MSc Research Project
MSc in Cloud Computing

Prathista Santhosh Kumar Shetty
x21146802

School of Computing
National College of Ireland

Supervisor: Vikas Sahni

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Prathista Santhosh Kumar Shetty
Student ID:	X21146802
Programme:	MSc in Cloud Computing **Year:** 2022
Module:	MSc Research Project
Supervisor:	Vikas Sahni
Submission Due Date:	15th December 2022
Project Title:	A New Offloading Technique Based on Deep Learning for Mobile-Edge Computing
Word Count:	6279 **Page Count:** 20

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Prathista Santhosh Kumar Shetty

Date: 15/12/2022

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

A New Offloading Technique Based on Deep Learning for Mobile-Edge Computing

Prathista Santhosh Kumar Shetty
x21146802

Abstract

Offloading techniques have gained more importance in today's world because of its need for high utilization of computing power, storage capacity and energy. The fundamental principle is to transfer resource-intensive processing operations to a more powerful processor for rapid computation. The development of new mobile technologies is increasing, requiring efficient utilization of resources. Mobile Edge Computing with distributed network has been developed with an aim to bring all the computational servers close to the end-users that meets predefined requirements. The rate at which tasks arrive at the mobile edge computing server for computing fluctuates every minute and it also depends on the user's density. For optimal computational rate in such a dynamic environment, task offloading should be efficient to reduce the uploading and downloading time. The aim of this research is to offload the complexity brought on by the rapid advancements of distributed computing in Mobile Edge technologies. To address this issue, a framework that employs a scalable solution using deep neural network solution is used that trains itself based on the binary offloading predictions by using the K-modes algorithm that chooses the largest reward. Since combinatorial optimization issues are no longer need to be solved, its computational complexity is significantly lesser than 0.1 % on a large user network with 30 systems. The research also shows that the CPU latency is about 67 times lesser when compared to the coordinate descent approach.

1 Introduction

A modern computing technology known as Mobile Edge Computing (MEC) makes online service available at networking edges by using mobile stations. It represents a promising innovation that aids in highlighting the drawbacks of mobile computing. Mobile devices usually have low processing power and this aids in overcoming the offloading in the system. MEC enables the smooth integration of different application services, hence providing available resources wound to the end-location users at the network edges. (Zhang et al. 2022) It is easily adaptable enabling the execution of resource-intensive applications that demand minimal network latency. In order to decrease the charge of battery usage in mobile devices and make high-end complicated processes use storage and processing capabilities to execute on a mobile interface, the data, services, and application processes within mobile devices are downloaded. Due to this, delay in sharing and resource allocation is avoided, regardless of how many users or applications are present. However, in such situations, data is queuing at various intervals that increases the computation request and various user requests can be handled concurrently by the edge storage servers. As a result, the mobile edge computing layer for varying traffic must inherit wireless methods for better performance.

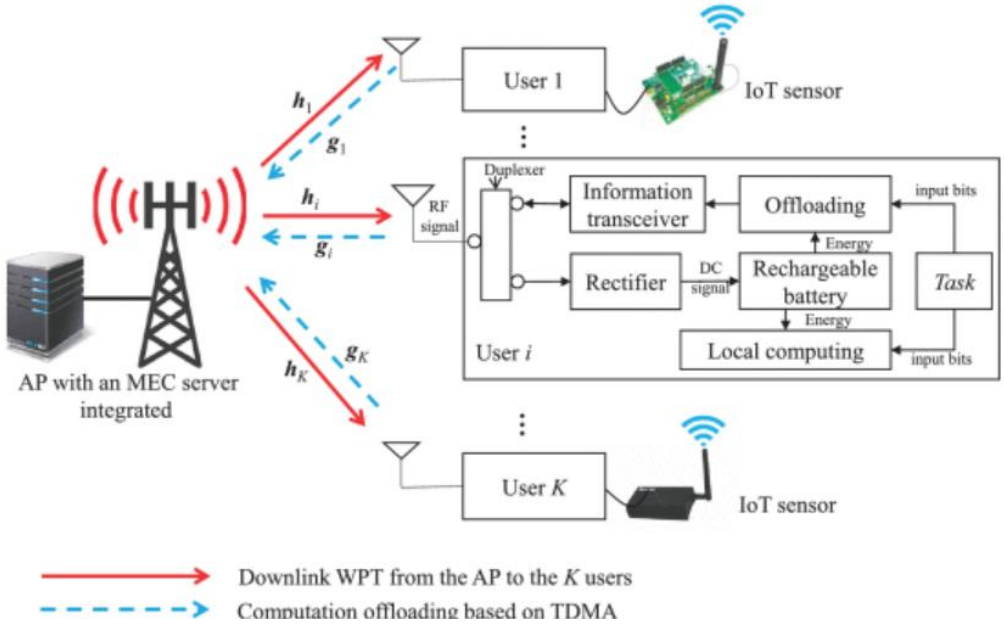


Figure 1: MEC system in a wireless powered network (Wang et al. 2017)

MEC is a subject that is still-evolving in online based computation standard which is being embraced by many commercial entities in an effort to distinguish themselves from competing technologies. Additionally, this technology is set up in the proximity of the mobile devices that are most likely to be nearest, with minimal server capabilities deployed at the network's edge to fulfil user-centric needs. (Huang et al. 2020) A generic MEC system in a wireless power-driven network is displayed in the Figure 1. In order to develop an edge computation with a truly effective technique, a well-organized and practical resource management is required after understanding the limitations such as storage, bandwidth, and CPU.

The combined optimization of wireless resource distribution and single offloading presents a significant issue in a multi-user scenario. (He et al. 2021) Because there are binary offloading parameters involved, these issues are typically described as multi objective programming issues. In order to improve this, different deep reinforcement-based learning algorithm have been applied over network edges to provide efficient communication and task balancing. This research was implemented considering a wirelessly–powered edge computing network with a single access point and several wireless devices with binary offloading strategy for each wireless device. The primary objective of this research was to optimize the offloading decisions made by each wireless device and the transfer time frame between offloading and wireless transmission. To achieve this, a deep reinforcement-based learning algorithm offloading system was used to increase all wireless devices' computation rates.

1.2 Research project specifications

This section describes the motivation and objectives for this research project.

1.2.1 Research Question

How effectively can deep reinforcement learning based algorithm be leveraged to reduce offloading computational complexity in Mobile Edge Computing?

1.2.2 Objectives and Contributions

- Design and implement deep learning-based algorithms utilizing binary offloading strategy.
- Evaluate the computation rate with different number of hidden layers.
- Visualize the computation rate, time consumed and average time per unit when a task is performed with the proposed algorithm.

The contribution will consist of the following:

- Critically analysing and reviewing the existing research performed on task offloading on mobile edge computing.
- Created a configuration manual for others to replicate the work.

1.3 Document Structure

- The related research is a discussion of mobile edge computing in Section 2.
- The research approach utilized to implement the newly suggested deep reinforcement learning-based algorithm is covered in Section 3.
- Section 4 discusses the architecture and algorithm proposed for the MEC network.
- Section 5 contains the system configuration, data used, followed by workflow of the implemented algorithm.
- Section 6 shows the outcomes of the proposed approach and the graphs generated for the same.
- Section 7 comprises of the conclusion and recommended future work.

2 Related Work

This section discusses existing research in the domain of compute offloading in MEC, in addition to the effectiveness of machine learning approaches. Additionally, many approaches for improving MEC have been reviewed in this section.

MEC research is popular research area due to large number of research opportunities at the conjunction of wireless infrastructures and mobile computing. Many researchers have looked at a variety of MEC-related challenges, such as allocating multiuser resources, system modelling and network modelling, etc. The two main MEC challenges are resource scheduling, resource allocation and compute offloading. These approaches are categorized into enhancing the profit made by the operators, to reduce the cost incurred by the consumers, optimizing the overall cost of building the MEC infrastructure and maintaining it, to reduce the measure of time needed to conclude a task and to focus on minimizing energy use when doing computationally intensive workloads. (Liu et al. 2017) The majority of MEC research has not addressed user mobility or random variations in task generation. As a result, there is a drastic decrease in QoS and a large decline in network performance when the platform deviates from an ideal configuration discovered through static allocation of resources. The major purpose using machine learning approaches is to manage system dynamics and the unpredictable pattern of tasks generated.

(Zang et al. 2021) suggested a computational resource allocation approach using deep reinforcement learning networks. In this approach, only the edge server could be used to offload tasks in the absence of a central cloud. Therefore, selecting the correct edge server to execute tasks is the main challenge in the offloading scheme in this case. In contrast, (Guo et al. 2016) included both the power consumption cost as well as the latency cost of performing the task, with the optimization aim being the reduction of the overall offloading cost. Non-linear processing with mixed integers is the type of optimization issue. The author suggests a combined deep reinforcement learning-based optimization strategy for bandwidth allocation and task offloading. The initial non-convex optimisation technique is identical to a reinforcement learning-based problem, and the Deep Q Network technique is employed to determine the best solution. (Shi et al. 2022) All alternative strategies are patterned as state spaces. Additionally, the multi-task multi-user offloading technique along with the other systems are outperformed by the DQN-based method.

(Huang et al. 2019) anticipates the existence of a popular area that is covered by many user devices but also that each of those devices can connect to a number of edge servers through the closest base stations. This research proposes a Markov decision framework-based task scheduling method. This approach will, however, result in significant computational complexity and communication overhead when the edge servers expand. In order to lessen complexity of the algorithm and overall communication overhead, the author suggests an index-based resource allocation technique. As per the state of its remaining computing resources, per edge server creates its own index, which it then broadcasts throughout the

network so that network device can choose the most effective edge server to reduce latency and power consumption.

The scenario in which computing workloads are offloaded locally to the edge server is taken into consideration in publications (Guo et al. 2016) and (Huang et al. 2019). In reality, edge nodes computation and storage capabilities are confined. The central cloud should continue to receive tasks when edge nodes resources are unable to handle them. (Guo et al. 2016) and (Huang et al. 2019), however, do not take into account the interaction in between central and edge cloud.

A wireless device within a local computing environment can simultaneously gather energy and do its task easily. (Wang et al. 2017) Let's consider e_i as the computing speed for the processor where $0 \leq T_i \leq$ T that represents the time taken for the computation. Furthermore, energy consumption of the wireless device will be limited by $k_i T_i e_i^3 \leq E_i$. This can be demonstrated that a wireless device must use all of the harvested energy while continuing to compute during the time period in order to analyse the utmost amount of data inside T considering the energy limitation. (Guo et al. 2016) Therefore, the rate of local computation can be defined as below:

$$\frac{e_i * T_i}{\phi T} = \eta 1 \left(\frac{c_i}{k_i}\right)^{\frac{1}{3}} a^{\frac{1}{3}} \quad (1)$$

If the edge node in the state is including the involvement of the principal cloud lacks sufficient computational resources, then the edge device can work directly with the centralized node. According to (Zhao et al. 2015), if the edge server satisfies the application interval criteria, a target level supportive scheduling method is employed to enhance the total of applications installed and deploying on the edge server. The significance of the program in addition to the allocation of computational resources in the edge server decides wherever the decommissioned application must be stored when the network device decides to uninstall the computing. Connect the virtual device executing the program towards the edge node if it has sufficient resources. However, if the edge server's computational capacity is not adequate, then the scheduler assigns the operation to the secluded central cloud. Hence, the author suggests a prioritized collaboration technique to configure distinct buffer spaces with various priorities in order to optimize the numerous of tasks handled throughout the network edge and satisfy their latency necessities. The processing of the task will only be forwarded to the cloud layer when the buffer is overflowing.

The provision of computational resources in the case of just one edge server is taken into account in the research implemented in (Zhao et al. 2015). In reality, there are frequently several diverse edge servers with each one containing a wide range of computational resources. A scenario with different range of IOT terminal regions, numerous edge node, and a centralized cloud node was also taken into consideration in the research paper (Guo et al. 2019). In the constraint that the job end time does not go beyond the interruption limit, the optimization purpose of this research is to reduce the energy usage. Local and nearby edge node are distinguished in this paper. A local node in the network is equivalent to every IoT terminal

region. IoT terminal will send the task to the suitable edge node. Here, the local nodes will send the task to the centralized cloud server for processing or to its nearby edge node. By leveraging Lyapunov's drift plus penalty model, the authors provide a delayed workload allocation technique that enables local edge node, nearby edge node, and centralized cloud node to perform computational workloads fairly in order to diminish overall energy utilisation.

The authors from (Guo et al. 2019) does not take into account the terminal area's computing power, namely, the terminal region in this article is only responsible to create tasks and offloading these towards the local network edge but not for computing. The terminal device can actually process tasks in real life. Traditional approaches can be applied in contexts that are static but they need further knowledge of people and network factors. However, there is little to no previous knowledge to aid in decision-making in a constantly changing network, necessitating the use of a machine learning-based approach.

There is very less research done on deep reinforcement learning for resource allocation and task offloading. The paper (He et al. 2015) conducted research for smart capitals, software-defined, virtualized systems integrating caching and mobile edge computing. The study provided an integrated approach that included networking, resource allocation to improve the application's performance and caching. Additionally, the model has allocation of resources strategy considering it as a joint optimization issue that is resolved in this work with the aid of a novel deep reinforcement learning method. The evaluation is shown through deep Q-network model's visualization. The evaluation demonstrates how well the algorithm performed in terms of convergence for several cases under the proposed method. To determine the efficiency of the suggested scheme, simulation results with various system parameters were also conducted. However, in this paper, the authors did not consider energy consumption efficiency issues. MEC networks research in (Huang et al. 2019) examines the decision of many wireless devices to delegate computation-related duties to a network edge. The improvement of bandwidth allocation and offloading decision is described as a mixed integer coding problem in order to save energy for wireless devices. However, the complexity in dimensionality places a computational constraint on the problem, making it impossible for ordinary optimization techniques to effectively and efficiently solve it, across several wireless devices.

In the above mentioned Deep learning techniques, there are drawbacks such as poor learning rates and poor environmental adaptation. The paper (Qu et al. 2021) suggests a deep meta reinforcement learning-based algorithm for offloading to discourse these concerns, which amalgamates several concurrent deep neural networks with Q learning to produce accurate offloading judgments. The finest offloading approach within a dynamic setting can be rapidly and adaptably obtained by a combination of the perceptual capability for deep learning along with the decision-making influence for reinforcement learning, and meta-learning capability in a fast-moving environment. Through many numbers of numerical simulations, DMRO is associated with conventional Deep Reinforcement Learning systems and the result shows an increase by 17.6 percent for offloading decision.

To address the challenging combinatorial computing mode selection, (Bi et al. 2018) proposed two effective solution algorithms. The algorithms used are coordinate descent method and the alternating direction method of multipliers-based method, optimizes both simultaneously. The coordinate descent algorithm uses bisection examination to obtain an optimal time allocation. The proposed coordinate descent based and alternating direction method of multipliers-based methods both perform better than other traditional benchmark methods. However, the algorithm does not work efficiently when the computation modes have alternating weights. The newly proposed algorithm utilizes the bisection search proposed in (Bi et al. 2018) to implement offloading decision which then chooses the best offloading action using K modes algorithm.

Main Content	Research Result	Limitation	Reference
Deep Reinforcement learning-based optimization strategy	Achieved better offloading performance with reduced power consumption cost	Overall communication overhead was more when edge servers are increased	(Guo et al. 2016)
Deep reinforcement learning	Resource allocation was improved	Energy consumption efficiency	(He et al. 2015)
Collaboration technique for edge and cloud	Tasks offloading performance was improved and latency	Only 1 edge server was taken into account for task offloading	(Zhao et al. 2015)
Lyapunov's drift plus penalty model	Delayed workload allocation	Computing power for terminal area was not considered	(Guo et al. 2019)
Markov decision framework and Index-based resource allocation technique	Reduced latency and power consumption	Communication between edge and cloud was not taken into account	(Huang et al. 2019)
DMRO algorithm	Increased 17.6 percent offloading decision performance when compared to DRL algorithm	Limitation to implement the algorithm with different learning structure techniques	(Qu et al. 2021)
Coordinate descent and Alternating direction method of multipliers	Offloading action performance was improved when compared to the DRL algorithm with bi section search method	These algorithms do not work efficiently when there are multiple tasks assigned to the wireless devices	(Bi et al. 2018) and (Bi et al. 2021)
A newly proposed deep reinforcement learning-based algorithm	The offloading decision was improved even with the varying computational tasks and the algorithm showed 67 times more improvement in	Algorithm can be improved by considering larger user network with more than 30 users.	

	execution latency when compared to CD method.		

3 Research Methodology

The approaches and algorithms that are applied to implement the resource allocation in addition to the offloading mechanism for the MEC network are described in this section.

The newly proposed framework automatically enhances its state of action being able to generate policy based on the experiences from past offloading decisions under diverse wireless fading environments. As a result, it fully eliminates the requirement to solve challenging multiple integer programming problems, which prevents the complexity of the algorithm from increasing as the network size increases.

This algorithm subdivides the fundamental optimization model to a sub-problem of offloading decisions and resource allocation, ensuring that the physical limitations are addressed. This is in contrast to many current deep learning approaches that optimizes the system parameters simultaneously and produce impractical solutions. It overcomes the problem of dimensionality by working for continuous actions without needing the requirement to normalize the wireless channel quality.

Offloading decision is the most challenging issue to resolve in the MEC network system. The optimization in real-time system under rapid fading channels is not feasible with traditional optimization techniques iteratively modifying the offloading routing decisions to an idealistic environment. An offloading wireless device uses up all of its gathered energy towards task offloading to increase computation speed. (Bi et al. 2021) Therefore, the computing rate is proportional to its capacity for data offloading. To identify an ideal or a suitable offloading decision, it must search among the 2N potential offloading decisions. To enhance the offloading decisions, for example, metaheuristic search techniques are suggested in (Bi et al. 2018) and (Tran et al. 2018). This search space is exponentially huge, though, thus it requires more time to converge using the existing algorithms. For the resource allocation, the bisection search is employed with a single dimension for dual parameter linked to constraint of time allocation in O(N) complexity that can be used to effectively solve the convex problem's maximum time allocation. (Bi et al. 2018) In order to address the offloading issue within the order of the computation time being milliseconds, a unique deep reinforcement learning-based offloading technique is proposed to address the complexity of the issue.

In the newly proposed deep reinforcement learning-based algorithm, an interconnected deep neural network with a single input layer consisting of 2 hidden layers and a single output layer is considered and the first 2 hidden layers has 80 and 120 hidden neurons. For this learning problem, different structures with varying number of neurons and hidden layers, or a different type of neural network can be used as a replacement to the deep neural network to match a particular learning problem, for example a recurrent neural network (Goodfellow et al. 2016)

can be employed. In this research, a straightforward 2-layer deep learning model is sufficient to obtain acceptable convergence performance, while further refining the deep neural network input variables is likely to produce higher convergence performance.

The algorithm is divided into two alternative processes that make up to this process. A deep neural network, whose embedded parameters include things like the weights connecting the masked neurons used to generate the first offloading technique in the algorithm. The deep neural network receives the channel gain as an input for the time frame and, also by using its existing offloading policy that is defined by a variable and outputs an offloading action. The action with best complexity is chosen from among K binary actions from that same relaxed action depending on the feasible computation rate. The newly acquired state-action pair is added towards the replay memory by the network after it performs the offload action, receives a reward, and then performs the offload action. (Li et al. 2021) The deep neural network is then trained using a group of training data that are pulled from memory during the update of policy stage of the time frame. The deep neural network will then update the variable from time frame to time frame + 1 and correspondingly increases with every iteration. The following time frame generates an offloading decision based on the newly observed channel using the updated offloading policy. As subsequent channel iterations are observed, these iterations continue, and the deep neural network policy is progressively enhanced with its computational complexity.

4 Design Specification

The section below gives the details of the new algorithm applied and its architecture.

4.1 Architecture

In this design specification, a new order-preserving action generating technique to effectively produce offloading actions was developed. In particular, the technique was computationally more effective and feasible in a large-scale system with a complex space since it only requires to choose from a small number of alternative actions every time. In addition, it generates actions with a high degree of diversity and improves convergence performance over traditional action generating techniques. The algorithm has been further improved by creating an adaptive process that instantly modifies the algorithm's settings with varying weights. To be more precise, it steadily reduces the quantity for sub problems of resource allocation that must be resolved in a given amount of time.

By doing this, overall computational complexity is effectively reduced without lowering the quality of the solutions.

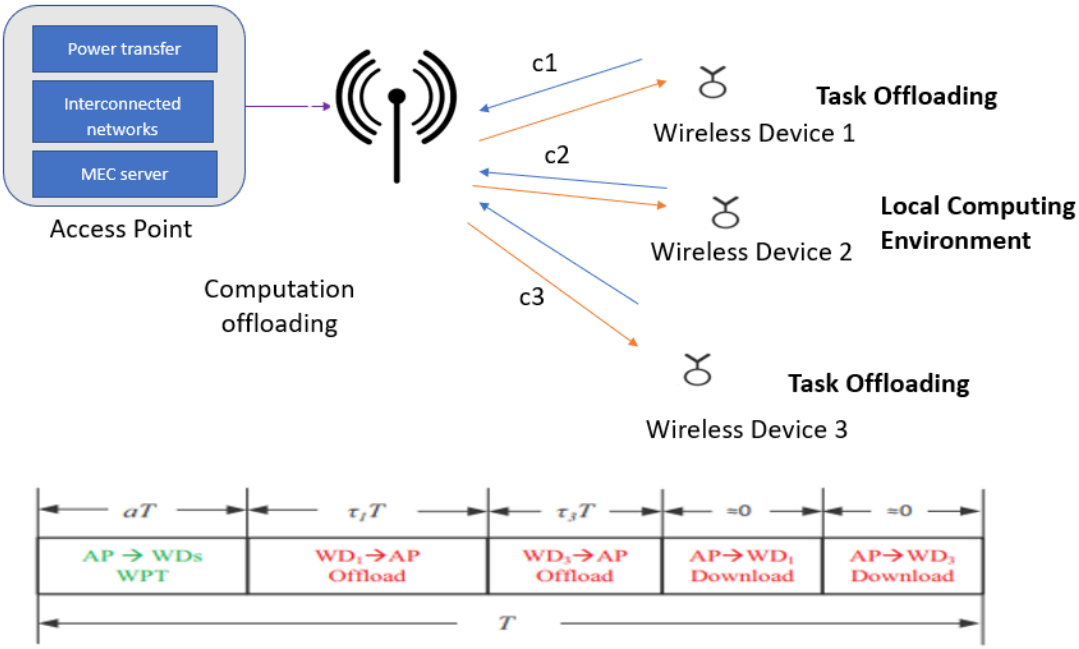


Figure 2: Architecture of the Mobile Edge Computing network

The design considers a wireless power-driven mobile edge computing network with N fixed wireless devices and an access point, indicated as N = {1, 2..., N}, individually with one antenna. In reality, this can be equivalent to a reduced IoT system or a sensor network with static configuration. The access point can transmit radio frequency energy to the wireless devices and has a reliable power supply. Every wireless device contains a battery which is rechargeable that can be used to store the energy that has been collected and used to operate the device. Assume that the access point has greater processing power than the wireless devices, allowing the wireless devices to delegate some of their computing workload to the access point. (Huang et al. 2019) The design assumes that communication and wireless power transfer both occur within the identical frequency range. Furthermore, offloading decision and resource allocation are the two subparts that are addressed in this research.

4.2 Algorithm workflow

An offloading wireless device uses up all of its gathered energy towards task offloading to increase computation speed. (Bi et al. 2021) Therefore, the computing rate is proportional to its capacity for data offloading. Considering this explanation, a new Deep reinforcement learning algorithm mentioned below is applied to address the offloading decision strategy using the above equation that has achieved a computational time within a millisecond order of magnitude to explain the offloading decision drawback. The deep neural network continuously improves its output from offloading decisions by learning out from best state of action pairings. The deep neural network can only learn using the most recently generated data samples by the most refined and latest offloading strategies due to the constrained memory space limitation.

Until convergence, the closed-loop learning process continuously enhances the offloading policy. The newly proposed pseudo code of the algorithm is given below:

Algorithm 1 Newly proposed deep reinforcement learning based pseudo code

Input: Overall gain from wireless transmission channel c_t at t time considering the offloading actions as k
Output: Computational complexity and time taken for Offloading decision for the t time frame
Configure the deep neural network using parameter θ_1 and zero memory.
Input iteration for N with a training interval α
for each t from 1, 2, 3, 4……, N **do**
    Generate offloading decision $a_t = k_{\theta_t}(c_t)$
    Transform a_t to binary offloading state of action operation
    Compute best action for a_t
    Update the weights of the memory by calculating the sum of c_t and a_t
    if t mod α = 0 **then**
        Train the deep neural network with the offloading decision and the binary action
        Using Adam optimizer algorithm update θ_t
    end
end

5 Implementation

This section includes the environment set up and implementation of the algorithm used to execute the proposed algorithm.

5.1 Device set up

The algorithm is implemented using Python programming language. Additionally, TensorFlow 2.0 is leveraged to significantly improve the algorithm's performance. TensorFlow 2.0 programs need an Nvidia GPU to perform, hence the Google Colab cloud environment was selected as the IDE for implementing this work. The simulations in this research are also implemented on the Google Colab cloud environment.

5.2 Device Configuration

TensorFlow 2.0, numpy, and scipy are necessary packages to run this program. If the desktop computer's GPU is Nvidia CUDA enabled, the program can be executed without any issue; alternatively, Google Colab's GPU-enabled runtime can be used.

5.3 Data

All the data is stored in the sub directory which includes the training and testing datasets. The dataset is generated with the coordinate descent mentioned in (Bi et al. 2018). The dataset consists of columns that has the input for gain from wireless channel, binary offloading state of action, access points for different radio frequency energy for all the wireless devices, time allocation for the offloading and the sum of weights for computational rate. The folder contains different samples that is classifies by different number of wireless devices.

5.4 Algorithm implementation

The algorithm used can be categorised into a reinforcement learning-based algorithm as it continuously learns from the previous feedback to optimize the final reward by adjusting the offloading actions.

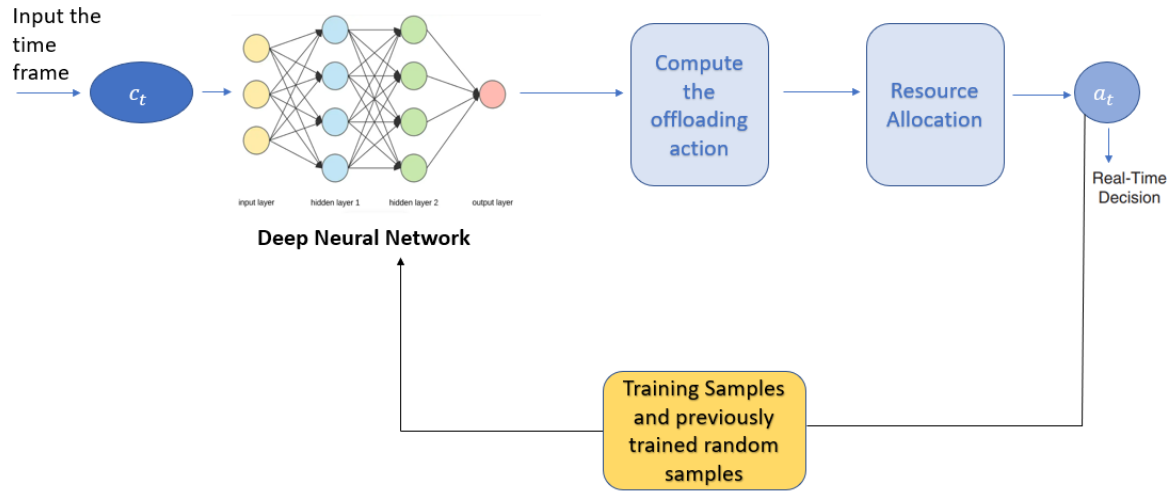


Figure 3: Sequence diagram of the algorithm implemented

As seen in Figure 3, the newly proposed algorithm divides the initial optimization problem into two subproblems: 0 to 1 binary action offloading decision along with the resource allocation problem. The two subproblems are then addressed independently with a dynamically learning module that is model free and an optimization module that is model based. This learning-based algorithm emphasis on optimizing resource allocation with every offloading action then chooses the best decision for every time slot after the offloading action module maps the input parameters to the binary offloading action parameters using an interconnected deep neural network. Following that, the relaxed action is classified as binary offloading state of action, from which the best action has been chosen after considering the maximum computation rate. The output ensures that all physical constraints have been met.

The newly acquired state of action pair is added to a replay memory once the network performs the offloading operation, earns the reward. The mapping from the input state vector to the action state vector is eventually produced by using deep neural network to train from the data samples that will be used for continuous learning. (Ke et al. 2021) The data training samples are taken from the previous behaviour that will be used to train the deep neural network. This training sample will continuously iterate to increase the time frame with every batch being trained and will be used to generate the next offloading decision.

6 Evaluation

As per the observation, the conventional two-layer deep learning model is sufficient to produce acceptable convergence performance, though further refining the deep neural network parameters is predicted to result in even higher convergence performance. TensorFlow 2.0 is used to implement this method in Python whereby default the interval for training is set as 10, followed by the batch size for training is fixed to 128 and the memory dimension is 1024. (Wang et al. 2022) The learning rate is set as 0.01 for the Adam optimizer.

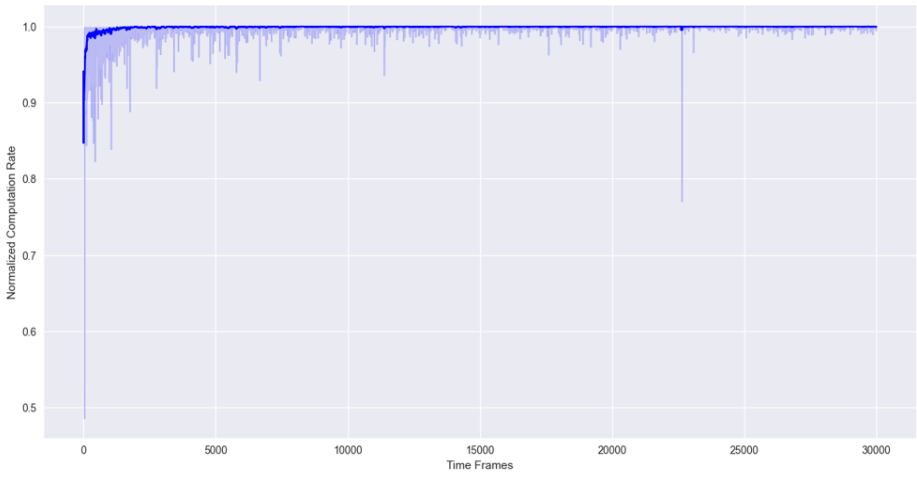
6.1 Experiment 1:

The standard normalized computation rate is determined as 0.9996, total time consumed to perform this task is 1635.33 seconds time frame and average per channel is computed as 0.054 seconds when K=10 and N = 10 as observed in the Figure: 4.

```
1/1 [==============================] - 0s 14ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 14ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 15ms/step
Averaged normalized computation rate: 0.9996670236095926
Total time consumed:1635.3216528892517
Average time per channel:0.054510721762975056
```

Figure 4: Output generated when K=10 and N=10

The parameters are fixed as K = N in this scenario. The sky-blue shadows in Figure 5 represents the minimum and maximum values over the previous 50 frames, while the sky-blue curve represents the trend line over those 50 frames. When time frame is large, the algorithm steadily approaches the ideal outcome when in moving average.



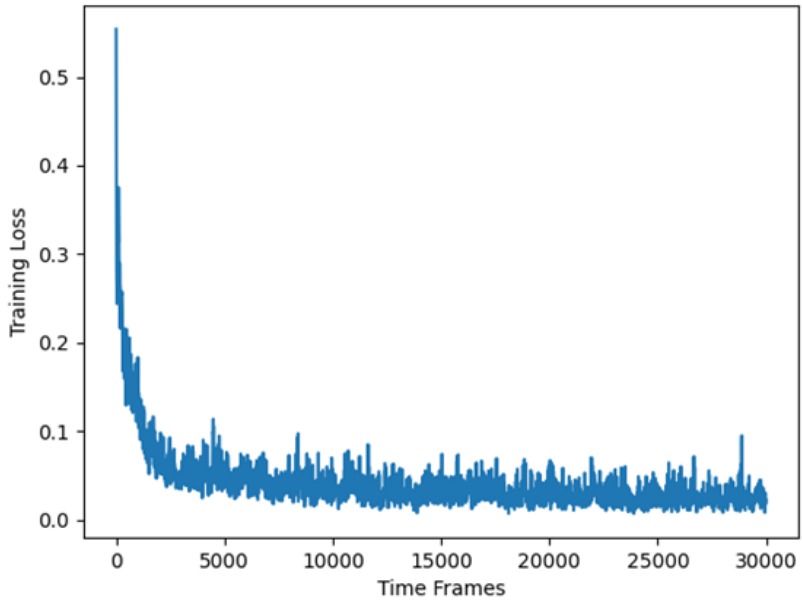


Figure 5: Graphs generated for computation rate when K=10 and N=10

6.2 Experiment 2:

The average normalized computation rate is determined as 0.9987, total time consumed to perform this task is 571.921 seconds time frame and average per channel is computed as 0.057 seconds as observed in the Figure: 6 when the weights are altered dynamically.

```
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 14ms/step
1/1 [==============================] - 0s 14ms/step
1/1 [==============================] - 0s 14ms/step
1/1 [==============================] - 0s 14ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 14ms/step
Averaged normalized computation rate: 0.9987449242648451
Total time consumed:571.921135187149
Average time per channel:0.05719211351871491
```

Figure 6: Output generated when weights are continuously changed

The algorithm is evaluated for MEC system network with changing weights for wireless devices. By simultaneously varying all the wireless devices weights from 1 and 1.5, specifically at the time frame 6000 and 8000 to understand the worst possible outcome. When the time frame is greater than 6000, as shown in Figure 7, the trend line of the computational rate is always more than 0.99 and the minimum computational rate is more than 0.95.

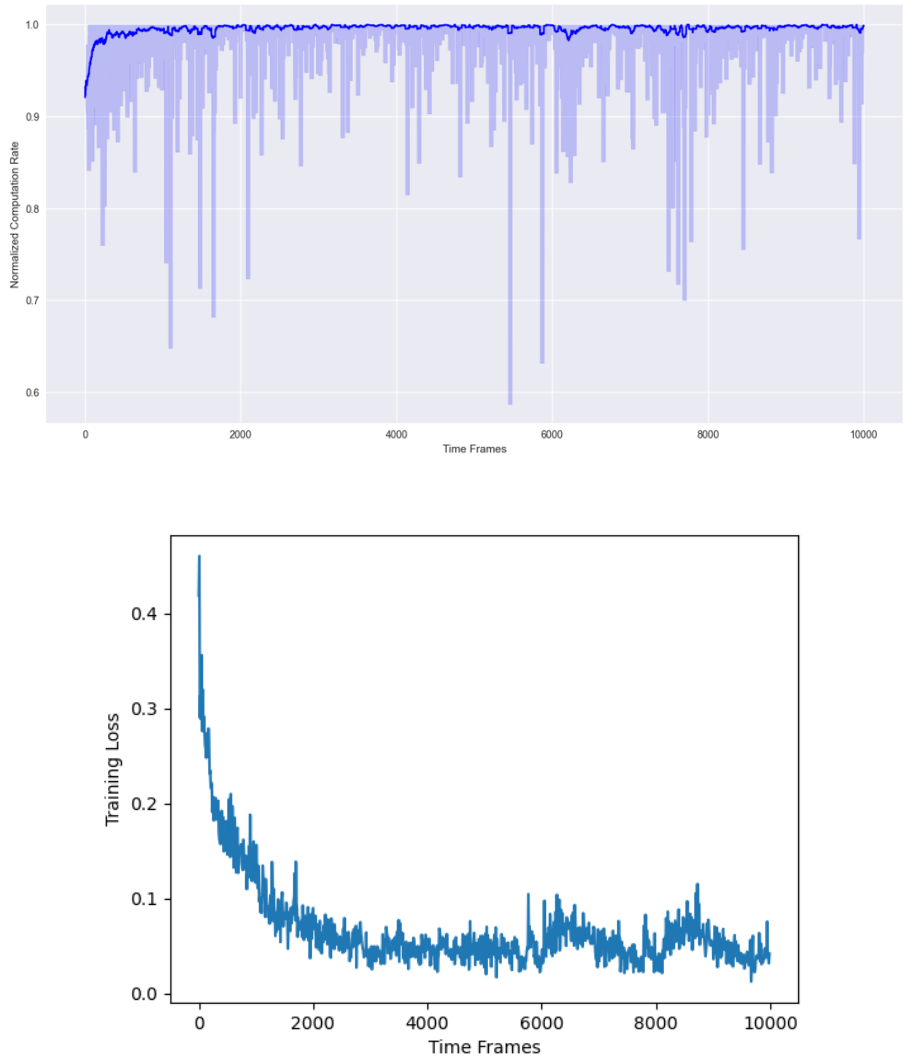


Figure 7: Graph generated when the weights are altered dynamically

The computing rate of the wireless devices used by the coordinate descent algorithm is iteratively switched every round to improve the computation rate. (Bi et al. 2018) The offloading action varies from 0 to 1 in this algorithm. It is proven that the coordinate descent approach performs nearly optimally for varying N. The iteration, however, reaches its end when the performance of the computation is not further enhanced by switching computing modes. With the newly proposed algorithm, the convergence is improved even with the alternating weights. This proves that the offloading decision performance is better with the newly proposed deep learning-based algorithm when compared to coordinate descent algorithm.

The algorithm is less complicated when compared to the Lyapunov optimization approach (Bi et al. 2021) as the algorithm is capable to using different learning tasks like RNN and CNN algorithms.

Additionally, the time taken to complete the offloading action is 0.054 seconds. Coordinate descent takes about 67 times more when compared to the newly proposed algorithm. The latency for the execution has also been improved when compared to Coordinate decent algorithm.

7 Conclusion and Future Work

For wireless power-driven mobile edge computing networks with k binary search offloading technique, this paper has presented the deep learning algorithms for online offloading to improve the computation rate for multiple wireless devices and varying wireless devices. The approach utilizes deep reinforcement learning to improve the offloading action produced by a deep neural network that learns from the past offloading behaviour. The suggested algorithm eliminates the requirement to solve challenging MIP problems, in contrast to traditional optimization techniques. According to simulation results, the algorithm achieves better computational rate with a CPU latency that is much lower when compared to the surveyed research articles.

The algorithm can be enhanced in the future by taking into account a user network with more than 30 network systems and concentrating on resource allocation that has continuous allocations on various types of networks.

References

Bi, S. and Zhang, Y.J., 2018. Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading. IEEE Transactions on Wireless Communications, 17(6), pp.4177-4190.

Bi, S., Huang, L., Wang, H. and Zhang, Y.J.A., 2021. Lyapunov-guided deep reinforcement learning for stable online computation offloading in mobile-edge computing networks. IEEE Transactions on Wireless Communications, 20(11), pp.7519-7537.

Goodfellow, I., Bengio, Y. and Courville, A., 2016. Deep learning. MIT press.

Guo, M., Li, L. and Guan, Q., 2019. Energy-efficient and delay-guaranteed workload allocation in IoT-edge-cloud computing systems. IEEE Access, 7, pp.78685-78697.

Guo, S., Xiao, B., Yang, Y. and Yang, Y., 2016, April. Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing. In IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications (pp. 1-9). IEEE.

Guo, X., Singh, R., Zhao, T. and Niu, Z., 2016, May. An index based task assignment policy for achieving optimal power-delay tradeoff in edge cloud systems. In 2016 IEEE International Conference on Communications (ICC) (pp. 1-7). IEEE.

He, Y. and Tang, Z., 2021. Strategy for task offloading of multi-user and multi-server based on cost optimization in mobile edge computing environment. Journal of Information Processing Systems, 17(3), pp.615-629.

He, Y., Yu, F.R., Zhao, N., Leung, V.C. and Yin, H., 2017. Software-defined networks with mobile edge computing and caching for smart cities: A big data deep reinforcement learning approach. IEEE Communications Magazine, 55(12), pp.31-37.

Huang, H., Ye, Q. and Du, H., 2020, June. Reinforcement learning based offloading for realtime applications in mobile edge computing. In ICC 2020-2020 IEEE International Conference on Communications (ICC) (pp. 1-6). IEEE.

Huang, L., Feng, X., Zhang, C., Qian, L. and Wu, Y., 2019. Deep reinforcement learning-based joint task offloading and bandwidth allocation for multi-user mobile edge computing. Digital Communications and Networks, 5(1), pp.10-17.

Ke, H., Wang, H., Sun, W. and Sun, H., 2021. Adaptive computation offloading policy for multi-access edge computing in heterogeneous wireless networks. IEEE Transactions on Network and Service Management, 19(1), pp.289-305.

Li, C., Zhang, Y. and Luo, Y., 2021. Deep reinforcement learning-based resource allocation and seamless handover in multi-access edge computing based on SDN. Knowledge and Information Systems, 63(9), pp.2479-2511.

Liu, H., Eldarrat, F., Alqahtani, H., Reznik, A., De Foy, X. and Zhang, Y., 2017. Mobile edge cloud system: Architectures, challenges, and approaches. IEEE Systems Journal, 12(3), pp.2495-2508.

Li, H., Shou, G., Hu, Y. and Guo, Z., 2016, March. Mobile edge computing: Progress and challenges. In 2016 4th IEEE international conference on mobile cloud computing, services, and engineering (MobileCloud) (pp. 83-84). IEEE.

Mao, Y., Zhang, J. and Letaief, K.B., 2016. Dynamic computation offloading for mobile-edge computing with energy harvesting devices. IEEE Journal on Selected Areas in Communications, 34(12), pp.3590-3605.

Qu, G., Wu, H., Li, R. and Jiao, P., 2021. DMRO: A deep meta reinforcement learning-based task offloading framework for edge-cloud computing. IEEE Transactions on Network and Service Management, 18(3), pp.3448-3459.

Shi, X., 2022. A task segmentation and computing offload algorithm for mobile edge computing. The Journal of Engineering.

Tran, T.X., Hajisami, A., Pandey, P. and Pompili, D., 2017. Collaborative mobile edge computing in 5G networks: New paradigms, scenarios, and challenges. IEEE Communications Magazine, 55(4), pp.54-61.

Tran, T.X. and Pompili, D., 2018. Joint task offloading and resource allocation for multi-server mobile-edge computing networks. IEEE Transactions on Vehicular Technology, 68(1), pp.856-868.

Wang, C., Lu, W., Peng, S., Qu, Y., Wang, G. and Yu, S., 2022. Modeling on Energy Efficiency Computation Offloading Using Probabilistic Action Generating. IEEE Internet of Things Journal.

Wang, F., Xu, J., Wang, X. and Cui, S., 2017. Joint offloading and computing optimization in wireless powered mobile-edge computing systems. IEEE Transactions on Wireless Communications, 17(3), pp.1784-1797.

You, C., Huang, K., Chae, H. and Kim, B.H., 2016. Energy-efficient resource allocation for mobile-edge computation offloading. IEEE Transactions on Wireless Communications, 16(3), pp.1397-1411.

Zhang, H., Wu, W., Wang, C., Li, M. and Yang, R., 2019, April. Deep reinforcement learning-based offloading decision optimization in mobile edge computing. In 2019 IEEE Wireless Communications and Networking Conference (WCNC) (pp. 1-7). IEEE.

Zhang, Y., Zhang, M., Fan, C., Li, F. and Li, B., 2021. Computing resource allocation scheme of IOV using deep reinforcement learning in edge computing environment. EURASIP Journal on Advances in Signal Processing, 2021(1), pp.1-19.

Zhao, T., Zhou, S., Guo, X., Zhao, Y. and Niu, Z., 2015, December. A cooperative scheduling scheme of local cloud and internet cloud for delay-aware mobile cloud computing. In 2015 IEEE Globecom Workshops (GC Wkshps) (pp. 1-6). IEEE.