THE ASIAN BULLETIN OF BIG DATA MANAGMENT
Vol. 5. Issue 1 (2025)
https://doi.org/10.62019/aqhpxx47

ASIAN BULLETIN OF BIG DATA MANAGEMENT
http://abbdm.com/
ISSN (Print): 2959-0795
ISSN (online): 2959-0809

Cyber-Resilient Mobile Edge Computing: A Deep Neural Approach for Secure and Efficient Task Offloading

Syed Mehtab Hussain Shah, Fahad Amin, Ajab Khan*

Chronicle

Article history
Received: Nov 24, 2024
Received in the revised format: Dec 10, 2024
Accepted: Jan 10 2025
Available online: Jan 15 2025

Syed Mehtab Hussain Shah currently affiliated with the Department of Computer Science, Abbottabad University of Science and Technology, Pakistan.
Email: mehtabshah916@gmail.com,

Fahad Amin affiliated with Department Computer Science, Shaheed Zulfikar Ali Bhutto Institute of Science and Technology University, Karachi, Pakistan.
Email: cs1912185@szabist.pk
***Dr. Ajab Khan** is currently working as Director of the Office Research, innovation and Commercialization at Abbottabad University of Science and Technology.

Email: ajabk66@yahoo.com

Corresponding Author *Dr. Ajab Khan

Abstract

Mobile edge computing (MEC) pushes cloud resources including computation and storage in vicinity of end devices. This effectively reduces communication latency. However, the minimum battery power of end devices limits the MEC performance. Integration of Wireless power transfer (WPT) with MEC enhances the performance by charging the end devices simultaneously while computation offloading. The problem lies in the variation of network state and channels while this offloading, which reduces the overall computation rate. To this end, in this paper, we aim to maximize the computation rate of whole MEC network using deep learning. We use a Deep Neural Network which learns from multiple episodes and decides whether to offload the task or not based on network state. This binary decision leads to remote execution of the task in case of optimal network state, and if not optimal, then leads to local execution. The experiments and results validate the effectiveness of our proposed framework.

Keywords: Computation Offloading, Deep Learning, Wireless Power Transfer, Mobile Edge Computing.

INTRODUCTION

Smart devices and sensors can only store limited energy, so they need to be charged repeatedly, which isn't always possible. Thanks to computational offloading and WPT, smart gadgets can control how much energy they use. By moving computation to cloud servers, smart devices can use less energy. Efficient computational offloading in MEC [1, 2] is a powerful way to lower the latency limits that are usually associated with geo-distributed cloud computing. Cloud servers handle programs and parts of programs that require a lot of processing power. This lets smart devices do more while keeping their batteries alive. When deciding whether to offload work or do it locally, it is important to look at how much energy is used and how much it costs to run mobile devices [3]. On the other hand, sending computing tasks to cloud servers may slow down transmission [6, 4] and cause security and privacy problems [4, 5]. A MEC server is put close to end devices to get around latency limits and shorten the amount of time it takes for data to be sent [7, 8]. Edge computing optimizes and controls energy cloud systems in a smart way, which makes them safer and more reliable [9, 10]. On the other hand, the performance of smart devices for end users is still limited by how much energy they use and how quickly their batteries die. WPT has been proposed as a feasible solution to the energy consumption and battery depletion issues. WPT charges neighboring devices without a cable power connection by utilizingtechniques such as inductive coupling and MRC. WPT can also persuade intelligent devices to offload computation to neighboring edge servers. In a wireless context, researchers presented

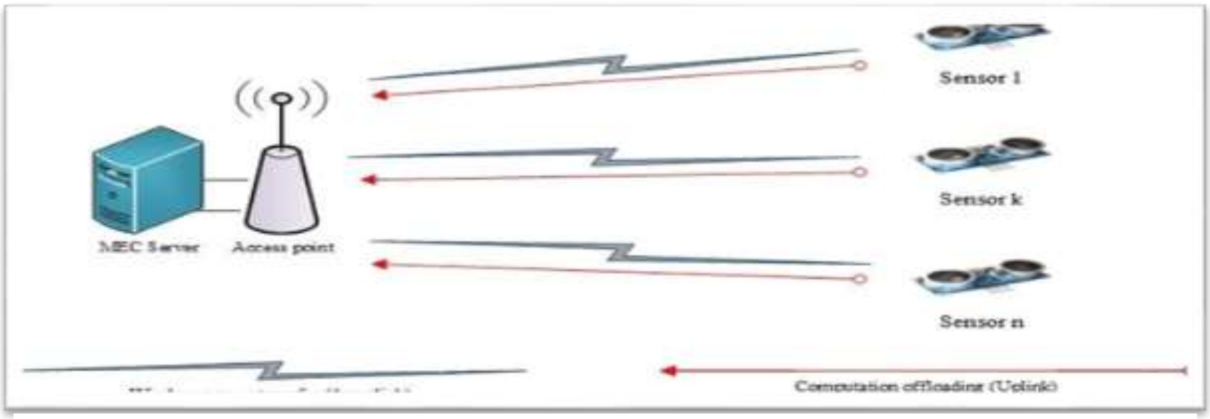

Figure 1WPT-MEC Mechanism

an approach for combining WPT with MEC computational offloading to MEC servers [11]. The WPT idea is depicted in Figure 1.1 [9], in which end devices get wireless power from a BS linked with a MEC server and are instructed to offload work to the edge server. Endpoints connected to the MEC server benefit from the WPT. This method may be used to charge smart devices from access points that support WPT (AP).

Several studies have found that smart devices use less energy when they use MEC computational offloading [12, 13]. MEC offloading can help smart devices finish jobs faster, use less energy, and keep their batteries charged longer. In the same way, WPT helps smart devices for end users keep their functionality while extending their battery life [14]. [15] talked about how charging smart devices could give them a reason to send their computing tasks to a nearby MEC server. For these gadgets to work, they need electricity from BS. [16] Considered a multi-antenna access point (AP) that sends wireless power to smart devices and lets them use that power to send their work to the MEC server or run locally.

We aim to maximize the computation rate of overall MEC system using deep learning approach. Further, We aim to increase the battery life of end devices that perform repetitive operations by combining an access point with a MEC server that broadcasts WPT to these end devices. We coordinate periodic tasks on the end device using deep learning approaches to boost computation performance and decrease task execution time.

RELATED WORK

To save energy, dual band simultaneous wireless information and power transmission (SWIPT) was created. It uses a base station (BS) and an edge server to send power to end devices [11]. In dual band mode, the properties of both the low frequency (LF) band and the high frequency (HF) band were studied. The LF band is used to send data over long distances and send energy over short distances. The HF frequency, on the other hand, is used to send data over short distances. In terms of channel resource allocation and energy harvesting, the suggested dual-band system was both efficient at energy harvesting and fair to all users. Wang et al. [15] showed a multiuser MEC system in which a BS is connected to a mobile edge server that gives wireless power to end mobile devices to encourage them to share tasks. The authors came up with an optimization problem that maximizes the usefulness of data while minimizing the amount of energy used by the operator. The authors (16) made a TDMA-based access point with multiple antennas that charges end devices and uses the collected energy for each end device's local processing or job offloading at the MEC server (TDMA). Researchers worked together to improve CPU frequencies, time distribution between users, offloading bits at the end device, and energy beam-forming at the access point (AP). Article 22 describes a way to charge mobile devices that improves performance by using energy harvesting technology and low latency. This method cut the cost of execution by a lot, which was missed in earlier studies, but it made latency and energy use go up because the end device had to wait longer. System performance was made sure by taking into account the costs of running the system and using less energy. [23] talked about research on UAV systems that use binary and partial offloading modes. By making communication and processing resources for UAV-enabled devices that run on wireless power, the rate of calculations was actually sped up. To get a high processing rate, the researcher used both partial and binary offloading. By assigning CPU cycles and frequencies together, the suggested study made users' experiences fairer. Also, these CPU cycles made it easier for more users to get to the top of the list.

[24] looks into a cooperative MEC system that can be controlled by UAVs wirelessly. An energy transmitter (ET) and a MEC server are put on a UAV to power and process sensor equipment (SDs). The SDs who are good at their jobs want to use the UAV and the SDs who aren't doing anything nearby to code. Combining the CPU frequencies, transmission power, number of offloading, and flight path to lower the UAV's overall energy consumption leads to an optimization problem. A sequential convex approximation-based strategy is used to solve a problem that is not convex. Given how hard it is to do the math, a deconstruction and iteration-based method is also shown to be a good choice. [25] tries to get the UAV to send as many completed task input bits and their weighted sum to end devices as possible. These unmanned aerial vehicles (UAVs) get their power from the AP and then send it to the end devices wirelessly. This lets the end devices do their jobs or send their work to the MEC server. The authors suggested using a block-coordinated descending method to solve the non-convex weighted sum job input bits maximization problem in a continuous way.

Using binary offloading mode, coupled optimization of mode selection (i.e., local execution or offloading), transmission time during WPT, and task offloading, [26] and [27] increased the rate at which wireless devices could do computations. Researchers looked into decoupled optimization and assumed that computing mode selection is already known. They then came up with a bi-section strategy for getting the best time allocation to solve the problem of transmission time allocation and multi-user mode selection being tied together. The authors of [28] used binary offloading and NOMA for offloading users to speed up computations in the WPT-MEC system. Researchers looked into the fact that the highest rate that can be reached with TDMA and NOMA is the same, but NOMA is better at making sure that all users get the same rate. Wang et al. [29] used a hybrid WPT-MEC system with TDMA for work offloading and multi-antenna AP broadcast power to end devices to get the most out of the computation rate. The author made sure that energy beam-forming, task offloading, and time allocation were all done in the best way possible.

Proposed System Model

We established a framework that covers all of the challenges described after doing a full literature research and determining the problem statement. The proposed solution provides the best response to the problem description and answers all questions adequately. We explained our proposed system and the methods we used to simulate it in this part. We first used the near field WPT technology to transmit energy from the AP to end devices. Second, we used Python- based tensor flow and machine learning methodologies, as well as other libraries. Following these simulations and code execution, we compare our findings to those of currently used approaches. Based on this comparison, we discovered that our proposed technique produces better results. After the detailed literature review and identification of problem statement we proposed our framework that answers all the question raised in the problem statement. Proposed solution effectively answers all the questions and provides optimal solution to the problem statement. Magnetic resonant coupling was used to simulate energy transfer from an AP (active circuit capable of broadcasting energy) to end devices using passive circuits in order to address the problem of battery depletion (inductor and capacitors to store energy). To experiment with network protocols, we employed the embedded WPT technique with the ns-3 simulator.

The access point (AP) contained several transmission coils (TX) in our simulation, and each wireless device had a single receiving coil (RX). A test message is sent when the power of a device falls below a certain threshold. Energy is transmitted to the end WDs on the response, which store these watts at random intervals by moving the transmitter coils and matching the impedance between RX and TX. To meet the battery drainage issue, we chose magnetic resonant coupling for simulation of energy from AP (Active circuit that has the power of broadcasting energy) to end devices having passive circuit (inductor and capacitors to store energy). We used ns-3 simulator and embedded WPT approach with it to work with network protocols. In our simulation, we used number of transmitting (TX) coils at the AP, and a single receiving (RX) coil with each wireless device. Whenever, a device power becomes less than a threshold, it broadcast a test message. On the response, transmitter coils moves around and matches impedance between RX and TX, energy is transferred to end WDs in and these devices store these watts periodically at a random time interval.

Wireless Power Transfer (WPT) is a technique that enables the wireless movement of electrical energy from one location to another without the need of wires or cables. The idea of "broadcasting" or "wireless power transmission," which refers to the technique of distributing electrical energy over a long distance using electromagnetic waves, is a major component of WPT. In WPT broadcasting, a power source, often a power amplifier or oscillator, is used to create an alternating current (AC) signal at a certain frequency. This alternating current signal is then sent through an antenna or transmitter coil, which turns the electrical energy into an electromagnetic field that radiates out into the surroundings. One or more receiving coils, normally positioned some distance distant from the transmitter, can receive the electromagnetic field created by the transmitter coil.

These receiving coils are tuned to the same frequency as the transmitter coil, allowing them to take up the electromagnetic field and transform it back into electrical energy. WPT employs two forms of broadcasting: near-field and far-field broadcasting. Magnetic induction is used in near-field broadcasting to transmit energy across small distances. This technology is frequently employed in situations where the transmitter and receiver are near to each other, such as wireless charging of electrical gadgets.

The magnetic field generated by the transmitter coil is strong enough to induce a current in the receiving coil, allowing electrical energy to flow between the two coils. Far-field broadcasting, on the other hand, uses electromagnetic radiation to convey energy across greater distances. This approach is frequently used in situations where the transmitter and receiver are many meters distant, such as wireless power delivery for electric cars. The electromagnetic field generated by the transmitter coil radiates in all directions, and the receiving coil must be built to capture as much of this energy as feasible. One of the most difficult aspects of WPT broadcasting is ensuring that the transmitted energy is efficiently received by the receiving coils. This necessitates careful design of both the transmitter and receiver coils, as well as the employment of sophisticated control systems to guarantee that the broadcast signal's frequency and amplitude are adjusted for maximum power transmission. Notwithstanding these obstacles, broadcasting remains an important component of WPT and a necessary technology for the development of new applications such as wireless charging, electric cars, and renewable energy. As research in this subject continues, we may expect to see additional breakthroughs in the design and execution of WPT broadcasting systems, leading to more efficient and effective means of wirelessly delivering electrical energy over larger distances.

Coordination of periodic tasks

The techniques for conducting and coordinating recurrent tasks on wireless devices are as follows.

Local Computation (Without Coordination)

Deep Learning Coordination (MB)

Algorithmic Coordination (DLPTO)

Local Computation (No Coordination)

Each WD in local computing completes its duty at a rate of 105 bits per second. Periodic tasks are not synchronized in this situation, and wireless channels that change over time are ignored. In local computation each WD computes its task locally with a computation rate of 105 bits/s. In this case there is no coordination of periodic task and time varying wireless channels are ignored. Local computing is a computational paradigm that focuses on distributed and decentralized data processing. Computations are done on local data in this paradigm, where local refers to a portion of data stored on a single computer or node in a distributed system. Since calculations are conducted on data that is instantly accessible, rather than waiting for all data to be aggregated in a single place, local computing provides for efficient processing of big datasets while minimizing communication overhead.

Local computing is particularly beneficial when the data is too vast to be stored and processed centrally. In such instances, local computing allows data to be processed on dispersed nodes, with each node processing a piece of the data stored locally. Because the work is split among numerous nodes and the results are pooled afterwards, this strategy can yield in a considerable decrease in overall processing time. One of the primary benefits of local computation is its capacity to handle data in a distributed and decentralized fashion. This allows for parallel processing since numerous nodes may conduct computations concurrently, boosting total processing speed. Furthermore, because just the relevant data is transferred rather than the complete dataset, local computing can decrease the quantity of data that must be transmitted between nodes.

This has the potential to minimize communication overhead, which is a major bottleneck in many distributed systems. Several practical uses of local computation exist, including data analysis, machine learning, and optimization. For example, in machine learning, local computing is used to train models on distributed data, where each node analyses a piece of the data and changes the model parameters locally. The model is thus trained in a distributed and parallel fashion, which can greatly reduce total training time. To summarize, local computing is a strong computational paradigm that permits distributed and decentralized processing of big datasets. Local computing has several practical applications in machine learning, data analysis, and optimization, and it is a necessary tool for data processing in distributed systems.

Problem Formulation

We proposed a deep learning based algorithm (DLPTO) that is based on deep neural network (DNN) approach. As we have considered time varying channels, our model takes input channel gain at each tagged time frame, selects the best action among multiple offloading actions and updates the DNN memory with best offloading actions. In more detail, DLPTO consist of two phases. In first phase data samples are collected including channel gain and DNN first outputs a normal offloading actions in which one devices executes task locally and other one offloads. For this purpose, each device is assigned a weight below 0.5 or above 0.5. A deep learning algorithm (DLPTO) based on a deep neural network (DNN) technique was developed. Because time-varying channels were considered, our model takes the input channel gain for each tagged time frame, chooses the best offloading action from a set of options, and updates the DNN memory with the best offloading options.

Deep Learning Periodic Task Offloading (DLPTO) is a strategy for increasing deep learning task performance and energy efficiency on mobile and embedded devices. DLPTO entails offloading compute-intensive processes to remote servers or cloud-based platforms while retaining local control over task execution on mobile and embedded devices. Speech recognition, natural language processing, and picture recognition are examples of tasks where DLPTO can be extremely effective. Mobile and embedded devices can save resources by outsourcing certain activities to more capable servers while still benefiting from the accuracy and speed of deep learning algorithms. DLPTO operates by a mix of prediction and preemption.

Initially, the DLPTO system predicts the future execution of a periodic job using machine learning methods. The job is then offloaded to a remote server before it is scheduled to run. This permits the distant server to finish the work, relieving the local device's resources. After the distant server finishes the job, the result is returned to the local device for further processing. DLPTO has the potential to greatly increase deep learning task performance and energy economy on mobile and embedded systems. Nevertheless, a stable network connection and a high level of coordination between the local device and the distant server are required. DLPTO is a thriving research field, with continuing efforts to build more efficient and effective offloading mechanisms for deep learning applications.

The DLPTO is split into two stages. DNN takes data samples, including channel gain, in the first phase, and then provides a typical offloading action in which one device performs the operation locally and another offloads. As a result, each gadget has a weight greater than or equal to 0.5. The technique for creating a common offloading action is as follows.

$$a1,i = \begin{cases} 0 & a_{t,i} < 0.5 \\ 1 & a_{t,i} >= 0.5 \\ 0 & \end{cases}$$

Where i= 1, 2, 3…… N, and ‘a’ denotes the offloading action. Binary value 1 suggests offloading, while binary value 0 indicates local execution. After generating a normal offloading action, it is quantized into K binary offloading actions, where K = 2N. Computation rates for each action are then determined based on the channel state, and the best one with the highest computation rate is picked. In the second phase, the best offloading actions update DNN's memory. To conform to the memory restriction, older best actions are replaced with new ones. DNN uses the optimum offloading action to achieve maximum processing speed and maintain its memory updated during these periods. The DLPTO pseudopod is depicted below. To meet the memory constraint, new best actions are replaced with older ones. Through these phases DNN keeps it memory updated with best offloading action and achieves maximum computation rate. Following is the pseudopod of DLPTO.

```
DLPTO Algorithm
Provides optimal offloading action between local and remote Computation
Input: Channel Gain ht at each time frame T
Output: Best Offloading Action at the Tth time frame with   optimal resource allocation
Initializing DNN parameters (Random) and with empty memory
Set the Iteration number N 3. For t=1,2,3…          N do
Generate Number of Normal Offloading Action an
Evaluate Computation rates C for all actions ( ht, at)
Select the best action at = max (ht, at)
Return (ht, at) to memory for updating
Train the DNN with (ht, at) and update Log loss l
End
```

Our model has one layer that goes in, two layers that stay hidden, and one layer that goes out. There are 80 and 20 neurons in each of the hidden layers. We used the ReLU activation function for the first hidden layer. For the second hidden layer, we used tanh, and for the output layer, we used sigmoid. But the system can be easily made bigger by adding more neurons to help train better. We used the Keras and Tensorflow tools to train our DNN model. Our model is made up of one input layer, two hidden layers, and one output layer. There are 80 and 20, respectively, neurons in each hidden layer. The ReLU activation function was used for the first hidden layer. The Tanh activation function was used for the second hidden layer, and the sigmoid activation function was used for the output layer. Just add more neurons to the system to make training go faster. We used preprocessed CD technique data to synchronize periodic actions [26]. 30,000 samples are included in the preprocessed data. Each data sample contains Wireless channel gain, Computing mode, Transmit time of wireless device and Sum of computation rates

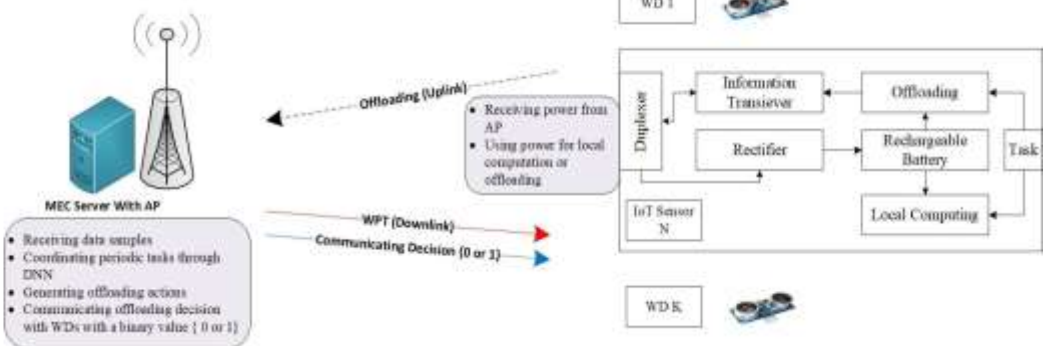
MEC Server With AP
Receiving data samples
Coordinating periodic tasks through DNN
Generating offloading actions
Communicating offloading decision with WDs with a binary value (0 or 1)
Offloading (Uplink)
WPT (Downlink)
Communicating Decision (0 or 1)
Receiving power from AP
Using power for local computation or offloading
WD 1
Duplexer
Information Transiever
Offloading
Rectifier
Rechargeable Battery
Task
IoT Sensor N
Local Computing
WD K

Figure 2Proposed system model

RESULTS AND DISCUSSION

We examined our findings and evaluated our system in this chapter. We demonstrated the effectiveness of our suggested method using simulated logs. We also included graphics to compare our suggested system against alternative options.

Experimental Setup

We used the open-source ns-3 simulator to simulate network protocols. In order to allow numerous offloading mechanisms, we integrated WPT simulation with ns-3. In order to assess performance, we estimated the WDs' power consumption per hour. This consumption ranges from 5 mW to 150 mW depending on the design of the final WDs. Tensor-Flow was used in conjunction with the Python programming language to simulate, code, and evaluate computation rate and training log loss performance. We used ns-3 simulator which is an open source tool for the simulations of network protocols. We embedded WPT simulation with ns-3 to work with different offloading protocols. To measure the performance we estimated the power consumption of WDs in watts/hour. Normally, this consumption relies between 5 mw-150 mw depend upon the architecture of end WDs.

On the other side, we used python language with tensor-flow for simulation, coding and performance measurement of computation rate and training log loss. In order to allow numerous offloading mechanisms, we integrated WPT simulation with ns-3. In order to assess performance, we estimated the WDs' power consumption per hour. This consumption ranges from 5 mW to 150 mW depending on the design of the final WDs. Tensor- Flow was used in conjunction with the Python programming language to simulate, code, and evaluate computation rate and training log loss performance. We evaluated the performance of our DLPTO algorithm to that of numerous deep learning benchmarking systems. We compared the performance of our DLPTO algorithm by using deep learning technique with other benchmarks schemes. We evaluated the performance of our DLPTO algorithm to that of numerous deep learning benchmarking systems.

Performance Evaluation Parameters

We divide this section into following three subsections to categorize our performance evaluation setup.

- Battery Performance
- Computation Rate Performance
- DNN Performance
- Battery Performance

We use WPT to efficiently tackle the WDs' battery draining issue. We investigated and showed the power consumption of two WDs with and without WPT. Figure 6 shows a comparison of WD battery power and WPT efficacy across different time periods. We compared the battery power of two WDs in the following graph by reporting the power consumption in watts/hour. The graph indicates that WDs without WPT may run out of battery power during computation if they do not receive wireless power from the AP. WPT, on the other hand, ensures that WDs are constantly charged so that they can perform their duties without interruption. We plotted the power consumption of two

WDs with and without WPT. Following Figure 6 shows the comparison between battery power of WDs and effectiveness of WPT in different time durations. In the following figure, we compared the battery power of two WDs in which we plotted the power consumed in watts/hour. Graph shows that without WPT, a WD can be dead after a time span depending upon battery power of this WD during computation if these devices do not receive wireless power from the AP. However, WPT ensures the continuous charging of WDs so that these devices could process their task without any power interruption.

We've also included the simulation and data log in the diagram below. This graph depicts the constant, uniform, and adequate charging of WDs. Figure 6 shows the comparison between battery power of WDs and effectiveness of WPT in different time durations. In the following figure, we compared the battery power of two WDs in which we plotted the power consumed in watts/hour. Graph shows that without WPT, a WD can be dead after a time span depending upon battery power of this WD during computation if these devices do not receive wireless power from the AP. However, WPT ensures the continuous charging of WDs so that these devices could process their task without any power interruption.

Computation Rate Performance

To enhance the computation rate per channel, DNN is used. The suggested Python-coded and Tensor-flow-imported approach (DLPTO) takes the channel gain at each time frame as input and uses experience to train a DNN. The DLPTO generates offloading actions for each input while concurrently updating the DNN memory and chooses the best one to offload. Using this training method, DLPTO effectively enhances the calculation rate for each input. Our proposed system (DLPTO) is coded in python and imported tensor-flow that takes channel gain at each time frame as input and trains DNN through experience. For every input, DLPTO generates offloading actions and selects the best one for offloading and also updates the DNN memory with this action. Through this training approach, DLPTO effective increases the computation rate for each input. Following Figure shows the performance gain with DLPTO. Figure 2 demonstrates the improved performance brought about by DLPTO.

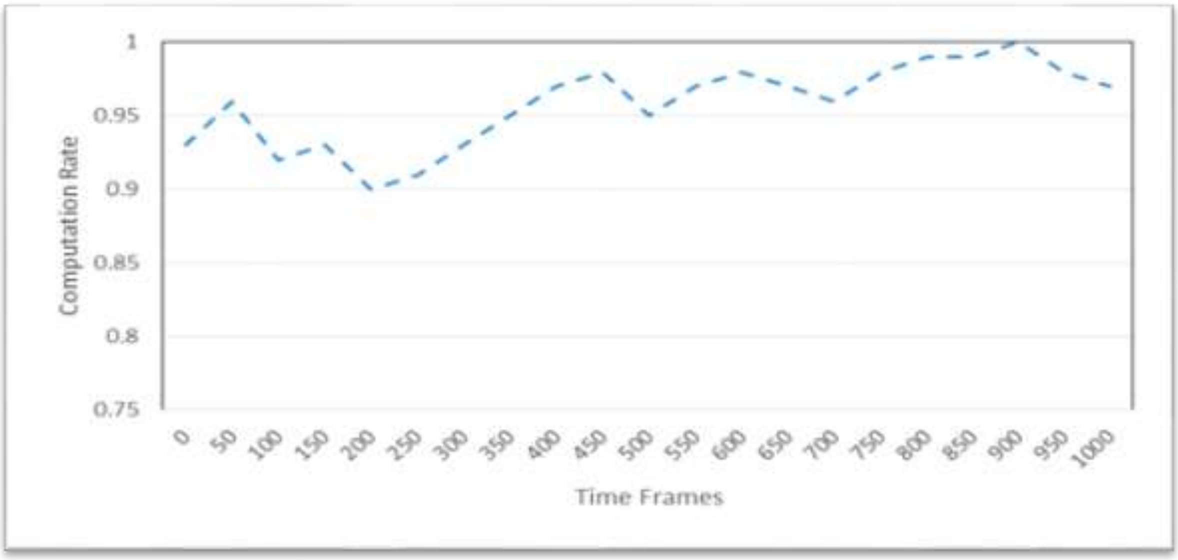

Figure 3Average Computation Rate

A blue graph represents the moving average of calculation rate as the number of time frames increases.

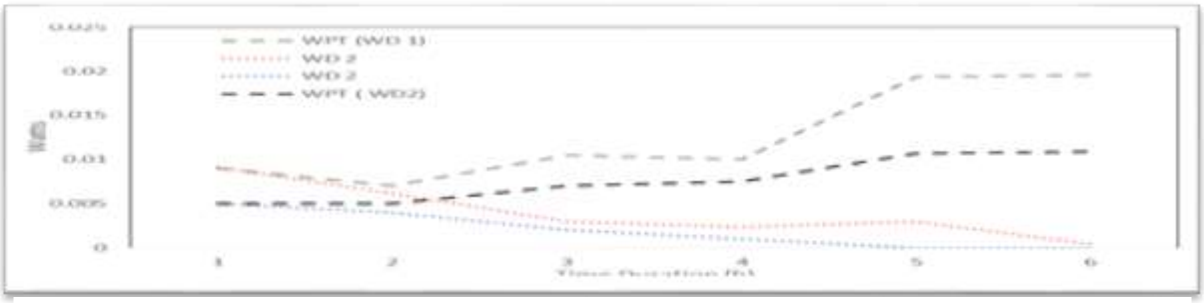

Figure 4Effectiveness of WPT

The simulation produces a per-channel average calculation rate of 0.98, which is close to real-time response.

Keep in mind that we evaluated 1000 time intervals; if the time intervals are changed, the computation rate will change accordingly.

The maximal calculation rate (bits/s) of DLPTO coordination was compared to that of the MB method and local computing. The MB method efficiently maximizes the computation rate with a high number of devices. However, DLPTO performs best when there are between 10 and 30 devices. We compared the maximum computation rate (bits/s) achieved by DLPTO coordination with MB algorithm and local computing. MB algorithm effectively maximizes the computation rate with a large number of devices. However, when devices are in between10-30, DLPTO achieves best performance. Figure 9 shows the effectiveness of our algorithm compared to MB and local computing. Figure 9 compares our algorithm's performance to that of MB and local computing.

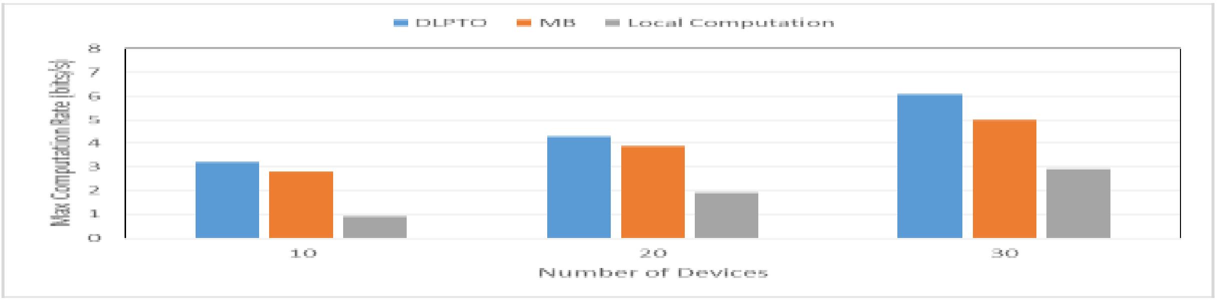

Figure 5Maximum Computation Rate

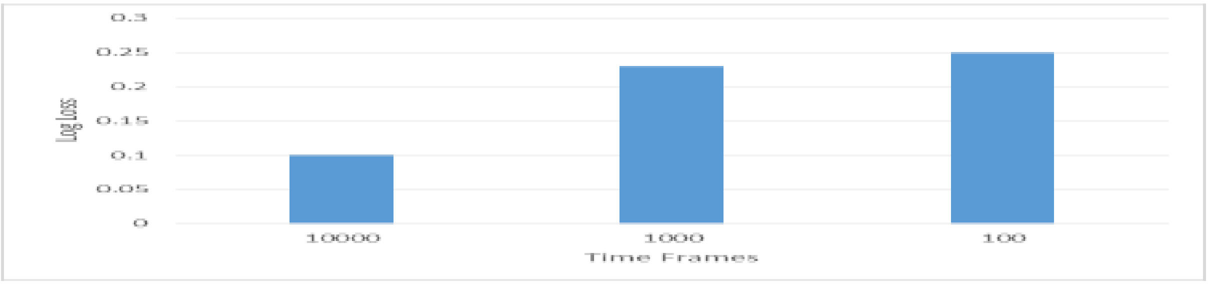

Figure 6Impact of time frames Computation Rate

The amount of time frames impacts the DLPTO's computation speed. Because this method learns from experience, increasing the number of time frames boosts calculation performance while reducing log loss. Figure 10 demonstrates the effects of time frames, with the first bar reflecting an increase in calculation rate as the number of frames increases and the second bar representing a decrease in calculation rate as the number of frames decreases. Number of time frames affects the computation rate of the DLPTO. As this algorithm learns from experience, increasing the number of time frames improves the computation rate with minimizing log loss and vice versa. Figure 10 shows the impact of time frames where first bar denotes the improvement of computation rate with maximum number of time frames and second bar denotes the computation rate affected by less number of frames. Last bar denotes the computation rate influenced by minimum number of frames. The last bar reflects the computation rate as influenced by the fewest amount of frames.

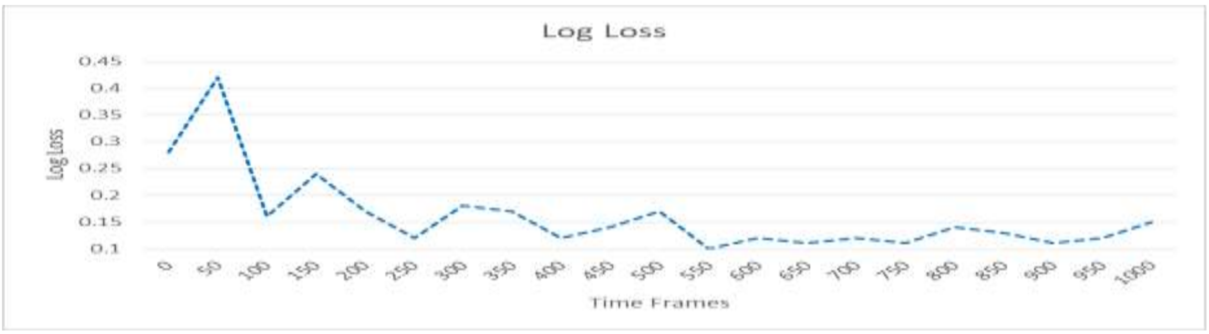

Figure 7Log Los

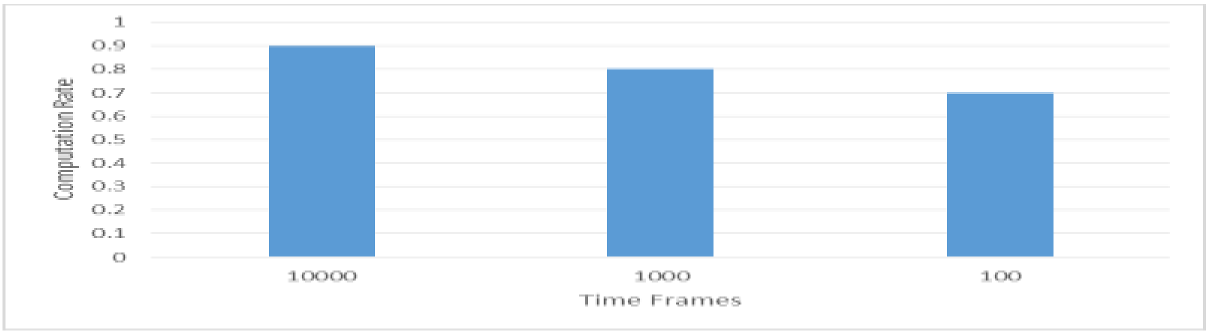

Figure 8Impact of Time Frames on Log Loss

DNN Performance

As the suggested system learns via experience, we estimated the cross entropy loss for our DNN model. This loss is also known as the training or log loss. Cross entropy is a type of loss function in which the loss is determined by how far each predicted value probability deviates from the intended output value, which can be 0 or 1. Over time, our model learns new skills and develops its own training schedule. Since the proposed system learns from experience, we calculated the log or training loss, also called cross entropy loss, for our DNN model. Cross entropy is the loss function where each predicted value probability is compared to the actual value desired output 0 or 1, and loss is calculated based on how far away it is from the actual value. Our model slowly learns and gets better as it uses itself. The log loss for our proposed model is shown in the next figure. The graph below depicts the log loss for our suggested model. We provided a log loss comparison in proportion to the amount of time frames in our DNN model. Due to DNN's limited or maximum experience gain, the number of time frames has a significant impact on this training loss. Because of the large number of time frames, the first bar in Figure 12 shows a rapid decrease in log loss, which becomes practically constant at 0.1. We plotted the comparison of log loss depending upon the number of time frames in our DNN model. Number of time frames highly influences this training loss because of minimum or maximum experience gin by DNN. In Figure 12, first bar denotes quickly decrease of log loss and becomes almost stable at a certain point 0.1 due to large number of time frames. However, second and third bar presents the gradual increase in log loss and also unstable due to minimum number of time frames. However, because to the limited number of time frames, the second and third bars show a continuous increase in log loss and are similarly unstable.

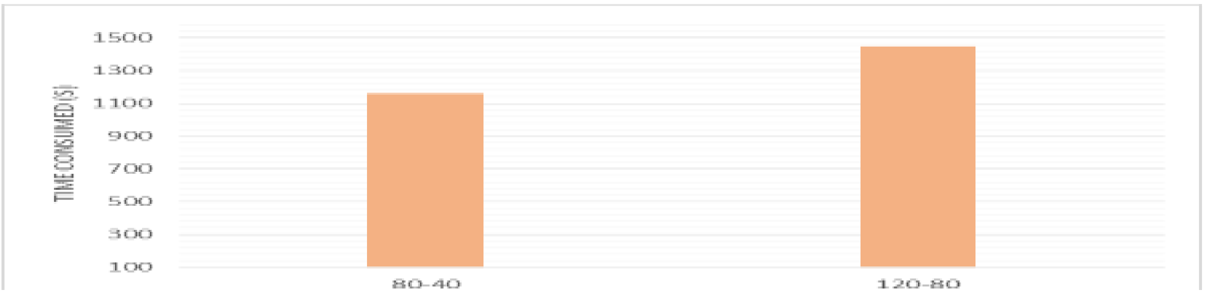

Figure 9Impact of Time Frames on time consumption

The duration of DNN processing is influenced by time frames. The smallest number of time frames has little influence on processing performance but has a significant effect on time consumption. A large number of time frames improves calculation performance but dramatically increases the time necessary to finish the task as DNN learns from deeper layers and experience. The graph below shows how time frames affect the overall amount of time DNN takes. Time frames also impacts the time consumption for DNN processing. Minimum number of time frames slightly impacts the computation rate, however greatly influences the time consumption. Large number of time frames in improves the computation rate as DNN learns from inner layers and experience, however consumes much more time to complete. Following graph shows the impact of time frames on total time consumption of DNN.

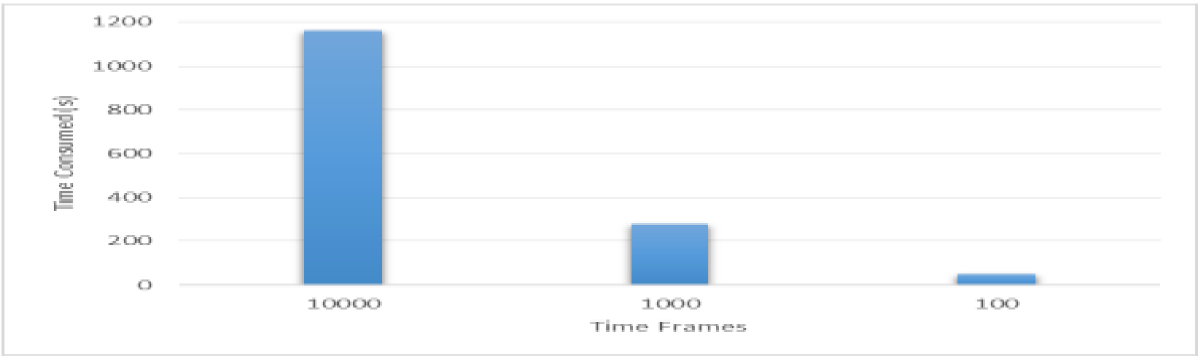

Figure 10Impact of number of neurons on computation rate

In order to evaluate DNN's performance, we compared its essential parameters. We used 120 and 80 neurons in the DNN's two hidden layers, respectively. Because our DNN model is totally coupled, the number of neurons has a significant impact on the time of the learning operation. Figure 14a shows how having more neurons has minimal effect on calculation speed, which may be ignored. Figure 14-b shows that using fewer neurons speeds up the execution of a DNN. To measure the performance of DNN, we also compared the inner parameters of DNN. We have used 120 and 80 neurons in two hidden layers of DNN respectively. As our DNN model is fully connected, more number of neurons slightly improves the computation rate, however it greatly influences the time consumption in learning process. Figure 14-a shows the impact of more number of neurons on computation rate which is minor, hence can be ignored. However, figure 14-b shows that using fewer number of neurons reduces the time consumption of running DNN.

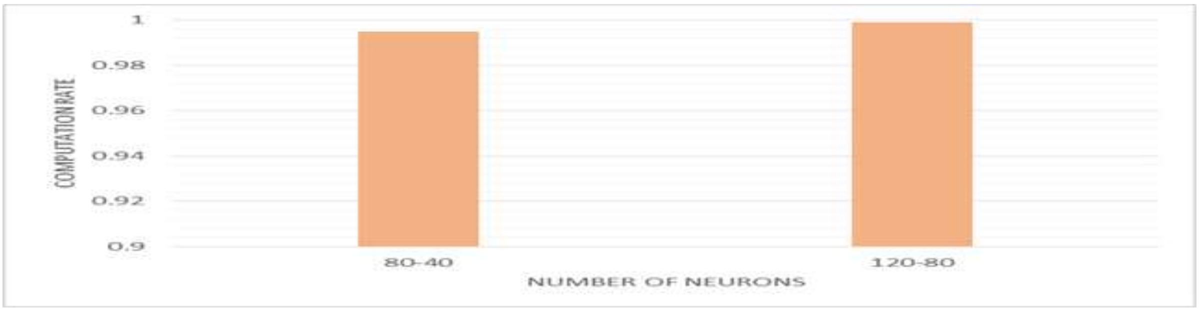

Figure 11Impact of number of neurons on time consumption

CONCLUSION

The ability to charge smart devices through Wireless power transfer (WPT) from a base station in combination with a Mobile Edge Computing (MEC) server broadens MEC's use. Furthermore, this thesis provided a thorough examination of intelligent online compute offloading in mobile edge networks. The research addressed the difficulties in effectively managing computation offloading in mobile edge networks and suggested a unique intelligent framework for efficient computation offloading. To optimize the decision-making process for computing offloading, the framework employs machine learning techniques. The experimental findings suggest that the proposed framework is successful at optimizing computation offloading and enhancing mobile edge network performance. The framework can adapt to changing network circumstances and workload demands, guaranteeing effective resource use and enhanced QoS for end users. This thesis' contributions go beyond the creation of the intelligent framework.

The paper also contributes to the larger field of mobile edge computing research by offering a better understanding of the compute offloading process in mobile edge networks. The suggested framework may be used in a variety of application situations, including real-time data processing and Internet of Things (IoT) applications. In conclusion, this thesis shows the promise of intelligent compute offloading in mobile edge networks and lays the groundwork for future study in this field. The proposed architecture provides an effective solution to the issues associated with compute offloading in mobile edge networks, and it has the potential to greatly increase network performance and efficiency.

The increasing popularity and energy and power consumption of smart devices has increased the need for MEC. Nonetheless, the demand for recurring end-device tasks such as traffic monitoring and surveillance is growing, necessitating longer battery life and more computational power from these devices. End devices, such as sensors, require more energy during periodic offloading than during non-periodic activities. In order to enhance the energy restrictions of end devices, these periodic efforts must be coordinated via joint WPT. By coordinating periodic task offloading, we investigated the combined technique of periodic task offloading with WPT from base station to end devices in order to extend the battery life of these end nodes and enhance the computation rate for decreasing job execution time.

REFERENCES

Ahmed, E. and M.H. Rehmani, Mobile edge computing: opportunities, solutions, and challenges. 2017, Elsevier.

Al Ridhawi, I., et al., Enabling Intelligent IoCV Services at the Edge for 5G Networks and Beyond.

Al-Shuwaili, A. and O. Simeone, Energy-efficient resource allocation for mobile edge computing- based augmented reality applications. IEEE Wireless Communications Letters, 2017. 6(3): p. 398- 401.

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

Bi, S., Y. Zeng, and R. Zhang, Wireless powered communication networks: An overview. IEEE Wireless Communications, 2016. 23(2): p. 10-18.

Chaudhry, S.A., et al., An improved anonymous authentication scheme for distributed mobile cloud computing services. Cluster Computing, 2019. 22(1): p. 1595-1609.

Choi, K.W., et al., Distributed wireless power transfer system for Internet of Things devices. IEEE Internet of Things Journal, 2018. 5(4): p. 2657-2671.

Feng, J., et al., Computation offloading and resource allocation for wireless powered mobile edge computing with latency constraint. IEEE Wireless Communications Letters, 2019. 8(5): p. 1320- 1323.

Hu, X., K.-K. Wong, and Z. Zheng. Wireless-powered mobile edge computing with cooperated UAV. in 2019 IEEE 20th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC). 2019. IEEE.

Huang, L., S. Bi, and Y.-J.A. Zhang, Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks. IEEE Transactions on Mobile Computing, 2019. 19(11): p. 2581-2593.

IEEE Transactions on Intelligent Transportation Systems, 2021.

Jararweh, Y., Enabling efficient and secure energy cloud using edge computing and 5G. Journal of Parallel and Distributed Computing, 2020. 145: p. 42-49.

Jia, M. and W. Liang. Delay-sensitive multiplayer augmented reality game planning in mobile edge computing. in Proceedings of the 21st ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems. 2018.

Jošilo, S. and G. Dán, Computation offloading scheduling for periodic tasks in mobile edge computing. IEEE/ACM Transactions on Networking, 2020. **28**(2):

Lhazmir, S., et al., A decision-making analysis in UAV-enabled wireless power transfer for IoT networks. Simulation Modelling Practice and Theory, 2020. 103: p. 102102.

Li, L., et al., Jointly Optimize the Residual Energy of Multiple Mobile Devices in the MEC–WPT System. Future Internet, 2020. 12(12): p. 233.

Liu, Y., et al., UAV-assisted wireless powered cooperative mobile edge computing: Joint offloading, CPU control, and trajectory optimization. IEEE Internet of Things Journal, 2019. 7(4): p. 2777-2790.

Mach, P. and Z. Becvar, Mobile edge computing: A survey on architecture and computation offloading. IEEE Communications Surveys & Tutorials, 2017. 19(3): p. 1628-1656.

Mao, Y., et al., Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems. IEEE Transactions on Wireless Communications, 2017. 16(9): p. 5994-6009.

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

Mao, Y., J. Zhang, and K.B. Letaief. Joint task offloading scheduling and transmit power allocation for mobile-edge computing systems. in 2017 IEEE wireless communications and networking conference (WCNC). 2017. IEEE.

Mazouzi, H., N. Achir, and K. Boussetta, Dm2-ecop: An efficient computation offloading policy for multi-user multi-cloudlet mobile edge computing environment. ACM Transactions on Internet Technology (TOIT), 2019. 19(2): p. 1-24.

Patel, Y.S., M. Reddy, and R. Misra, Energy and cost trade-off for computational tasks offloading in mobile multi-tenant clouds. Cluster Computing, 2021: p. 1-32.

Peng, K., et al., A survey on mobile edge computing: Focusing on service adoption and provision.

Posner, J., et al., Federated Learning in Vehicular Networks: Opportunities and Solutions. IEEE Network, 2021.

Rana, M.M. and W. Xiang, IoT communications network for wireless power transfer system state estimation and stabilization. IEEE Internet of Things Journal, 2018. 5(5): p. 4142-4150.

Rana, M.M., et al., Internet of Things infrastructure for wireless power transfer systems. IEEE Access, 2018. 6: p. 19295-19303.

Samanta, A. and Y. Li, Poster: Latency-oblivious incentive service offloading in mobile edge computing. ACM/IEEE SEC, 2018.

Taleb, T., et al., On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration. IEEE Communications Surveys & Tutorials, 2017. 19(3): p. 1657-1681.

Varga, D. and S. Laki. Scalable Surface Reconstruction in the Mobile Edge. in Proceedings of the ACM SIGCOMM 2018 Conference on Posters and Demos. 2018.

Varghese, B. and R. Buyya, Next generation cloud computing: New trends and research directions. Future Generation Computer Systems, 2018. 79: p. 849-861.

Wang, F. Computation rate maximization for wireless powered mobile edge computing. in 2017 23rd Asia-Pacific Conference on Communications (APCC). 2017. IEEE.

Wang, F., et al., Joint offloading and computing optimization in wireless powered mobile-edge computing systems. IEEE Transactions on Wireless Communications, 2017. 17(3): p. 1784-1797.

Wireless Communications and Mobile Computing, 2018. 2018.

Yu, Y., J. Zhang, and K.B. Letaief. Joint subcarrier and CPU time allocation for mobile edge computing. in 2016 IEEE Global Communications Conference (GLOBECOM). 2016. IEEE.

Zeng, M., et al. Computation rate maximization for wireless powered mobile edge computing with NOMA. in 2019 IEEE 20th International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM). 2019. IEEE.

Zhai, D., et al., Simultaneous wireless information and power transfer at 5G new frequencies: Channel measurement and network design. IEEE Journal on Selected Areas in Communications, 2018. 37(1): p. 171-186.

Zhou, F., et al., Computation rate maximization in UAV-enabled wireless-powered mobile-edge computing systems. IEEE Journal on Selected Areas in Communications, 2018. 36(9): p. 1927- 1941.