National College of Ireland

# Reliable Online Offloading Using Deep Reinforcement Learning In Mobile Edge Computing

MSc Research Project
MSc in Cloud Computing


Kamal Nikhar Yadav
Student ID: x20246935

School of Computing
National College of Ireland

Supervisor: Dr. Aqeel Kazmi

**National College of Ireland**
**Project Submission Sheet**
**School of Computing**

| | |
|---|---|
| **Student Name:** | Kamal Nikhar Yadav |
| **Student ID:** | x20246935 |
| **Programme:** | MSc in Cloud Computing |
| **Year:** | 2023 |
| **Module:** | MSc Research Project |
| **Supervisor:** | Dr. Aqeel Kazmi |
| **Submission Due Date:** | 01/02/2023 |
| **Project Title:** | Reliable Online Offloading Using Deep Reinforcement Learning In Mobile Edge Computing |
| **Word Count:** | 6150 |
| **Page Count:** | 18 |

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:** | |
| **Date:** | 1st February 2023 |

**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): | |

# Reliable Online Offloading Using Deep Reinforcement Learning In Mobile Edge Computing

Kamal Nikhar Yadav
x20246935

**Abstract**

IOT Wireless devices(WDs) are resource constraint and the method that is widely opted to overcome this problem is task offloading. In this research, we consider a system in which there are multiple users in MEC network which has wireless channel that vary with time and stochastic data queues. We aim for designing an algorithm that will take care of online offloading in the least amount of time, while increasing the number of bits processed in given time frame. The algorithm is useful because the offloading decisions in online computation are decided without taking into consideration the future channel states and data queues. This is resolved by using Lyapunovs optimization and Deep Reinforcement learning in the framework called Reliable Online Offloading Using Deep Reinforcement Learning(RDRL). RDRL tackles the problem by first applying Lyapunov optimization on the MINLP problem and convert it into sub-problems. Then, RDRL then uses model-free approach to DRL to solve these sub-problems with low time complexity. The evaluation results show that RDRL achieves good computation rates with stable queues. Alongwith this it has very low time complexity which is advantageous for implementations in real-time decisions based on the environment.

# Table Of Contents

# 1 Introduction

The tremendous advancements in wireless communications and networking over the last decade have been propelled by the proliferation of mobile devices and the incredibly rapid growth of mobile Web traffic. The ideas of the the Internet of Things (IoT) and 5G communications have prompted a shift from centralized mobile cloud computing to edge devices in past decades (MEC) Mao, You, Zhang, Huang and Letaief (2017). Mobile Edge Computing (MEC) is a network architecture concept that provides IT and cloud computing capabilities at the mobile network's edge [3]. Due to its proximity to clients, MEC can offer a service strategy with exceptionally low latency, high bandwidth, and direct access to real-time network data. Furthermore, the introduction of new services such as the Internet of Things, smart gadgets, super HD video, and a growing variety of cloud solutions has caused a significant increase in network traffic. Mobile-edge computing (MEC) is a critical solution for enhancing the processing efficiency of wireless devices (WDs).Li et al. (2020) MEC is very helpful for Internet of Things devices with limited size and low battery capacity and processing power. Utilizing MEC servers located at the edge of radio access networks, such as cellular base stations, WDs may offload time- and energy-intensive intensive computation operations to a nearby edge server (ES). Proactive computation offloading refers to tasks that may be calculated locally or at the edge server, and has shown a significant performance boost under time-varying network circumstances, such as harvested energy level, wireless channel gains, and task input-output dependence.Yan et al. (2020)

To increase the efficiency of the MEC (multi-user) network, much research is conducted on opportunistic compute offloading. Binary offloading is a sort of offloading in which a fundamental or deeply connected job is offloaded as a single unit, either at the MEC server or directly on the mobile device. The problem of resolving mixed integer nonlinear programming (MINLP) determines the offloading choices, such as binary offloading, time of task offloading, and edge or local CPU frequency. In complex systems, tackling such challenges often requires prohibitively sophisticated computer systems. According to Dinh et al. (2017), Yan et al. (2020), various studies have focused on developing sub-optimal algorithms with decreased difficulty, such as decomposition-oriented query, relaxations of binary variables, and local-search-based heuristics. These sub-optimal approaches may suffer performance losses and require a vast number of statistical iterations to arrive at an ideal answer.

The MEC servers are widely spread in close proximity to mobile users, and mobile devices may wirelessly offload computational tasks to the MEC servers. Mobile users may leverage online offloading computing to improve Quality of Service through offloading computing (QoS). As a consequence, there is significant interest in the MEC system's key problem of

computation offloading and computational resource allocationLi et al. (2018). It is difficult to determine the optimum strategy in a system that is always changing and evolving. Reinforcement Learning (RL) takes into account not just short-term rewards, but also long-term aims, which is vital for time-variable dynamic systems such as the multi-user wireless MEC system. To handle wireless MEC allocation of resources, an RL-based optimization strategy is proposed. DRL (deep reinforcement learning) is an enhanced variant of RL (reinforcement learning) and an information method. This strategy offers an alternative possible solution to the online offloading issue. The DRL technique employs the Deep neural network (DNN), which is a model-free drive that use DNN to educate itself with the best mapping from the state, including time-varying parameters, to the data offloading choices and allocation of resources action. This method significantly accelerates the pace of information processing by interacting constantly with the surrounding environment, which is rewarded by this algorithm. This method is much more advantageous for online offloading in MEC networks than the MINLP technique, which often requires complex calculation.Liu, Yu, Xie and Zhang (2019)

## 1.1 Research Project Specification

This section defines the research question and the objectives for this paper.

### 1.1.1 Objective

The objective of this research paper is to propose a Lyapunov stability based DRL framework which will reliably offload the task to edge servers with least amount of energy consumption and with stable data queues.
In this paper we aim to propose Reliable Online Offloading Using Deep Reinforcement Learning(RDRL) framework that will leverages the advantage of both Lyapunov optimization to first solve the MINLP equation for a schotastic channel and then use DRL for generating the binary offloading decision alongwith maximizing the bits processed.
Alongwith proposing RDRL framework we will evaluate it when varying the different parameters like number of WDs, system parameters like power constraint and varying the Lyapunov constant parameter.

### 1.1.2 Research Question

How efficiently the energy consumption can be reduced to increase the reliable task offloading using Lyapunov stability and Deep Reinforcement Learning in multi-access edge computing for Online devices?

## 1.2 Project Structure

Section 2 presents the related work for MEC, IOT, DRL, and Lyapunov optimization. It takes into account the methods they proposed and also their limitations. Section 3 talks about the system model that we propose for RDRL framework and the architecture of it. It discusses the various modules in the framework and also the resource allocation algorithm being used in RDRL.
Section 4 discusses the tools that are used for developing and evaluating RDRL and also the code flow of it. Section 5 discusses the evaluation of RDRL based on various

parameters like Lyapunov optimization constant, power constraint, number of WDs and discussing its findings.

# 2 Related Work

Mao, You, Zhang, Huang and Letaief (2017) talks about the positive attributes of Mobile edge computing over Mobile Cloud environment. MEC provides IoT with mobile energy saving by utilizing computation offloading, maintaining lower latency by propagating data, and enhancing security and privacy for mobile applications. The paper also discusses the offloading models which are commonly seen such as partial and binary offloading computing models. In this paper, the binary offloading strategy is discussed which is defined as a particular task being executed or offloaded to MEC or processed at the mobile device itself. In this paper, the binary offloading strategy is discussed which is defined as a particular task being executed or offloaded to the MEC server or processed at the local mobile device itself.
The binary model is a non-partitionable design for processing simple computation tasks. A multi-user MEC network that adopts a binary offloading model struggles with the intractable combinatorial computational offloading complexity. Lee et al. (2019) has mentioned using a technique in which the Wireless devices will select neighbouring fog nodes to shorten the latency in fog computing by attaining a low competitive ratio. The competitive ratio is determined in the paper by a threshold-based algorithm this is the proportion of the internet delay method to the offline optimum latency. The main aim of the technique is to reduce transmission latency. However, in this paper, the optimization of a long-term objective is aimed which is not practical for this technique.

Yan et al. (2020) demonstrates in their study the Gibbs sampling algorithm that obtains a satisfactory offloading decision. This study is based on a various clients MEC system utilizing a method in which a task input on a single Wireless device is needed for the final output from multiple other wireless devices. A threshold-level optimal offloading decision is achieved by the Gibbs sampling method. Bi and Zhang (2018) puts forward an multiply rotating orientation technique also called an ADMM which tackle com- plexity issue in vast networks. ADMM is a decomposition Decays the initial optimization problem into multiple simultaneous comment thread. The paper also states about the integration of edge computing and RF- based on power transmission without wired setup which can probably tackle the performance struggles and limitations in the IoT networks.

As per Du et al. (2019), presents a positive strategies to tackle the issue of offloading smart Vehicular terminal, VT applications to their proximal MEC server. Smart Internet of Vehicles supports programs that includes real-time navigation systems, and gaming. The above devices demand heavy bandwidth and are often delay-sensitive which in turn pressures the Vehicular terminal and its radio access network. To tackle the VT's excessive energy consumption and weak terminal processing capability, MEC enabled-Vehicular networking is proposed in which the MEC computing platform and MEC roadside units are constituted. The roadside units are attached to the Mec servers and deployed along the wireless access to VT. The major problem is tackled by offloading the respective computations to the MEC servers. This optimization such as the VT side and MEC-enabled roadside units sides optimization issues are decoupled into inde- pendent frame optimiz-

ation without needing any knowledge of network state info or any future upcoming task arrival. Continuous relaxation is developed and a low-complexity algorithm is obtained. Although beneficial, These methods of optimization generate the problem of performance complexity tradeoff upon operating the integer variables.

Dinh et al. (2017) suggests a platform that offloads from a single Mobile units to many edge systems. The paper studies the impact of a individual unit to allocate tasks to multiple Access points. The framework which is proposed would minimize the lag in task completion and consumption of resources. Based on the study, their SDR approach (semidefinite relaxation) could reach optimal performance. It was assumed that Access points process all of their tasks offload by the same particular Mo- bile device which will not be possible if there would be multiple Mobile devices offloading to the same access point. Liu, Yu, Xie and Zhang (2019) ) focuses on the DRL method (deep reinforcement learning), to achieve the desired resource allocation . In this paper, a VEC called a Vehicle edge computing net- work is designed where the automobiles offer compute capabilities as well as typical edge services. The suggested framework would increase the flexibility and service range of MEC. The offloading is acquired to increase the functionality of the VEC network. The paper is concluded by saying DRL-based technique can obtain a improved output as compared to the pure FES, fixed edge server method, or Vehicle edge server method.

Li et al. (2018) propose RL-based learning (Reinforcement learning) that is as flexible as the multi-user Wireless Mobile edge computing device. The RL-based optimization paradigm addresses allocation of resources in the wireless MEC. Methods based on Q-learning and DRL are presented to overcome the offloading computing issue. The DRL-based method is proposed in a multiuser system. Resource allocation and reduction in energy consumption are studied in this paper and an RL and DRL-based solution is derived which tackles the offloading computation decision and resource allocation issues in this framework. Chen et al. (2019) brings us the comparison between two computational algorithms which are double DQN-based (Deep Q -network) reinforcement learning (DARLING) and Deep-SARL (deep state-action-reward-state-action-based reinforcement learning algorithm). It also discusses the MEC being a mobile user in a sliced RAN (radio access network) where for computation offloading, multiple base stations (BSs) are available. The main objective is to maximize the utility performance through which an offloading conclusion is made which in turn is based on the energy state, task queue state, and the channel qualities among the Base stations and multiple users. Markov decision process model is proposed for problem-solving of an optimal offloading computation policy. Using the utility function structure and based on tests undertaken to solve randomized computation offloading, the Deep-SARL method outperforms the DQN-based (Deep Q -network) reinforcement learning.

Tang and Wong (2022) shows the LSTM (long short-term memory), double-DQN technique, and DQN (dueling deep Q-network) algorithm so that offloading decision can be determined for each device without knowing the offloading decision of other devices and the task models. Mainly, the problems with delay-sensitive and non-divisible tasks in the MEC system are studied and the offloading algorithm is designed which enables the wireless mobile devices to make their offloading decisions in a decentralized manner. The problem of tasks being dropped due to increased traffic and delay is reduced to a

certain amount. Similarly in Min et al. (2019), through simulation situations, the author has shown that latency, energy consumption, and the task drop rate is reduced by the proposed RL-based offloading scheme and thereby increasing the IoT device's utility. The RL-based offloading scheme also permits the IoT device to choose the offloading rate and the edge device as per the battery level and previous transmission rate. However, in the case of Wireless devices, the number of offloading actions is exponential and a DQN-based method would be very costly. To settle this issue, actor-critic Deep Reinforcement Learning is proposed in Wei et al. (2019) which is an algorithm to learn the stochastic policy for various things such as computing offloading, caching of the content, and radio resource allocation. The actor part of the algorithm utilizes another DNN (deep neural network) to represent a stochastic policy based on parameters. The DNN which is employed here evaluates the huge number of actions and system states and estimates a value function for the actor-critic. This is the policy-based approach studied in this paper that tackles the problem of computing, joint caching, and resource allocation in the IoT that are Fog enabled. The overall framework helps in reducing service latency. Similarly, Xiao et al. (2020) has proposed a similar model to improve the computational performance that utilizes the critic network to update actor network weights and the actor-network to select the offloading policy. For a MEC to select the edge device and the offloading rate, the author has also designed the RL-based mobile offloading. This allows the device to address the problem of heavy interference and smart jamming. To further enhance the computing performance of mobile devices that enable deep learning, a deep RL version of the model that combines the actor-critic technique mentioned above and DQN is also proposed. According to game theory, we have presented the computational complexity of the suggested scheme as well as its performance bound, which takes into account the processing delay of the tasks, energy consumption, and the usefulness of the mobile device.

In Du et al. (2020), the author talks about the wireless equipment that has the features of Virtual reality mode which needs very large bandwidth and good processing capability. To reduce energy consumption and task latency for the WD such as Head-mounted displays, Multi-access edge computing is proposed. Ultrahigh-speed wireless data transmission is expected by the THz (bandwidth-rich terahertz) communication. So, to minimize the energy consumption of a THz wireless-based MEC system is proposed, that will provide high-quality VR video support. The DDPG (deep deterministic policy gradient) is another policy-based method talked about in Zhang et al. (2020) and Xiao et al. (2020) that uses DNN to directly build the best mapping strategy from the input state to the output action. To determine the best way to transfer a continuous input state to discrete output actions, Xiao et al. (2020) evaluates a Wireless device that only performs discrete offloading operations, such as integer offloading decisions, offloading rates, and discredited transmit power. Whereas Zhang et al. (2020) trains two individual modules to obtain continuous resource allocation successively and discrete offloading decisions. For picking a discrete offloading action, Zhang et al. (2020) has combined a DNQ-based critic network with an actor DNN for developing the resource allocation solution. To enhance the computing experience for IoT devices, MEC with EH (energy harvesting) is an emerging paradigm. To tackle the problem of continuous-discrete hybrid action spaces and coordination of devices, the author has proposed two DRL- based algorithms, that are Hybrid-AC ( hybrid-decision-based actor-critic learning) and Multi device-Hybrid-AC. The problem of hybrid action space is solved by the Hybrid-AC with the advancement of actor-critic architecture. In this situation, the critic

assesses the continuous actions and yields the distinct server selection action and the actor outputs continuous actions (local computation capacity and offloading ratio). The framework of centralized training is adopted by MDHybrid-AC with decentralized execution. By building a centralized critique to output server selections that take into account all device's continuous action policies, it learns how to make coordinated decisions. The results of the simulations demonstrate that the suggested algorithms successfully strike a balance between the amount of time and energy they require and that they perform significantly better than the default offloading rules.
The previously discussed DRL- based methods do not address the problems such as the stability of the queue under random environments which are long-term per long-term performance requirements. Recent studies in Mao et al. (2016), have utilized Lyapunov optimization to plan an online offloading strategy. The Lyapunov optimization-based computation offloading algorithm splits the stochastic problem into deterministic subproblems in each timeslot at the time of implementation. The Lyapunov computation offloading algorithm decides the transmit power and CPU-cycle frequencies for computation offloading and mobile execution respectively. These decisions depend on the current state of the system, without requiring information on the EH request, task request, and wireless channel. Sun et al. (2017) investigated the Mobility systemic issue for a MEC-enabled Ultra dense network (UDN). By merging the ideas of Lyapunov enhancement and MAB (multi-armed bandit), a novel user-specific energy-aware mobility management (also known as the EMM) is created that really can meet 7 the restrictions of power consumption simultaneously obtain the optimal latency. Mao, Zhang, Song and Letaief (2017) and Liu, Bennis, Debbah and Poor (2019) examine the multiple user's joint offloading decisions which are different from the binary offloading policy regarded in this paper. The optimization theory allows Wireless devices to tackle the resource allocation problem and continuous joint offloading. As the binary offloading policy is utilized in some of the before-mentioned papers and also in Du et al. (2019) and Liu, Bennis, Debbah and Poor (2019), the number of potential offloading options increases exponentially with the user count. An algorithm is designed in Du et al. (2019) which is based on the Lyapunov optimization called DDROV to first receive server provisioning independently and then come up with the continuous relaxation plan to tackle the combinatorial problem of joint offloading. In Liu, Bennis, Debbah and Poor (2019), the author proposes a two-term mechanism, the first term is to obtain a user-server decision and then the second term is to execute task offloading and resource allocation policy. But if we look into the consistent long-term result that may degrade the long-term performance, a high-quality solution cannot be promised.

# 3 Methodology

### 3.0.1 System Model

Considering the system depicted in the Figure 1, we consider N WDs with ES assisting them in computation for T same duration sequential time frames. For a $i$ th time frame, $A_n^i$ represents the rate that which the tasks arrive in the data queue for $n$ th WD. Assuming $A_n^i$ has i.i.d. distribution, i.e. $E[(A_n^t)^2] = \eta_n < \infty$, for $n = 1, ..., N$. $h_n^i$ represents the channel gain between ES and the $n$ th WD. According to the block fading hypothesis, $h_n^i$ is constant inside a given time frame and fluctuates separately across frames.

Assume that a tagged $n$ WD computes $D_i^n$ bits of input data in $i$ th time frame and

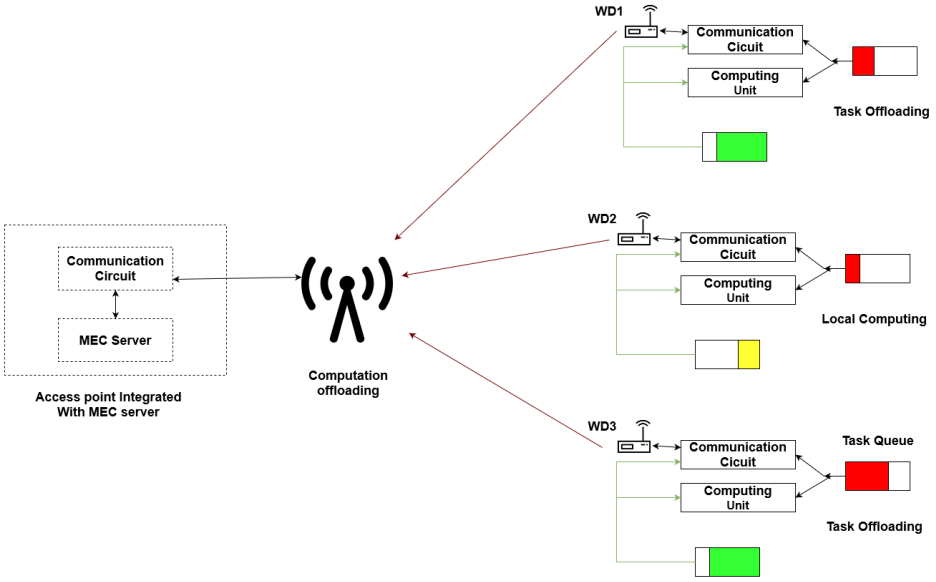



Figure 1: Multi-user Network In MECLiu, Bennis, Debbah and Poor (2019)

outputs a result of the computation at the termination of the frame. We suppose that WDs follow a binary rule for task offloading in particular. Consequently, the raw data for each time frame should either be handled locally at WD or remotely at ES. For example WD2 processes the data locally and WD1 and WD3 offload task to ES in figure. WDs offload their task to the ES using shared bandwith W by TDMA process. We are considering a binary property $x_n^i$ to represent the offloading decision. $x_n^i = 0$ and 1 representing that $n$ WD computes the data locally or remotely respectively.
Data(in bits) processed locally by WD is given by:

$$D_{n,L}^i = f_n^i T/\phi,\ E_{n,L}^i = \kappa \left(f_n^i\right)^3 T,\ \forall x_n^i = 0, \tag{1}$$

In equation1 $\phi > 0$ represents the processing cycles required to compute 1 bit and $\kappa > 0$ represents efficiency of the required energy.
When $x_n^i = 1$, the task if offloaded to the ES. $P_n^i$ represents the transmit power limited by $P_n^i \leq P_n^{max}$ and $\tau_n^t T$ is the allocated time to $n$ th WD for offloading. Here, $\tau_n^i \in [0, 1]$ and $\sum_{n=1}^{N} \tau_n^i \leq 1$. Offloading data consumes $E_{n,O}^i = P_n^i \tau_n^i T$ energy. As given in You et al. (2016) and Bi and Zhang (2018) Since we don't account for the edge computing latency, the volume of data analyzed at the edge in the allotted time is:

$$\begin{aligned} D_{n,O}^i =& \frac{W\tau_n^i T}{v_u} \log_2\left(1 + \frac{P_n^i h_n^i}{N_0}\right) \\ =& \frac{W\tau_n^i T}{v_u} \log_2\left(1 + \frac{E_{n,O}^t h_n^i}{\tau_n^i T N_0}\right), \quad \forall x_n^i = 1, \end{aligned} \tag{2}$$

here $v_u \geq 1$ represents the overhead required for communication and noise power is represented by $N_0$.

In this research, we build an online approach to increase the long-term total mean sum calculation speed of all the WDs while observing average power and data queue stability restrictions. We specifically make judgments online in the sense that we maximize number of bits processed and resource allocation for each time duration without making the assumption that we know how random channel and data arrivals will manifest in the future. Each brief time frame, such as the channel coherence time, demands real-time decision-making due to the fast-varying channel situation. We propose a Reliable Online Offloading Using Deep Reinforcement Learning framework to solve this with robustness and efficiency.

## 3.1 Algorithm For DRL and Resource Allocation

To solve the MINLP problem we consider a parameter $\xi^i$ which depends on the gains of the channel represented by $\{h_n^i\}_{n=1}^N$ and status of the queues in the system $\{Q_n(i), Y_n(i)\}_{n=1}^N$ where $Q_n(i)$ denotes the queue length and $Y_n(i)$ denotes the queue of virtual energy. $\xi^i$ parameter based on energy queue and queue length and channel gains define the control actions $\{\mathbf{x^i}, \mathbf{y^i}\}$, where $\mathbf{x^i}$ is the binary decision and $\mathbf{y^i}$ is the parameter defining the resource allocation. In the MINLP problem the resource allocation will be a easy convex problem when the offloading decision fixed. Hence the solution of this problem will be a function of $x^i$ and $\xi^i$ and the optimal decision for offloading, $(x^i)^*$ will be the maximum value of $x^i$ and $\xi^i$.
To find $(x^i)^*$ we need to enumerate a total of $2^N$ offloading decision, hence it is a really high time complexity even for normal sizes of $N$. Other search algorithms like branch and bound, block coordinate descent, will also have high execution time when N has a large value. So, practically none of these methods are applicable to decision making for online offloading in channels with fast varying dynamics. To overcome this we propose Reliable Online Offloading In Mobile Edge Computing Using DRL(RDRL) framework which uses DRL which creates an offloading policy $\pi$ which maps the input to it to the optimal offloading action with low time complexity.
Figure show that RDRL comprises of four modules which are:

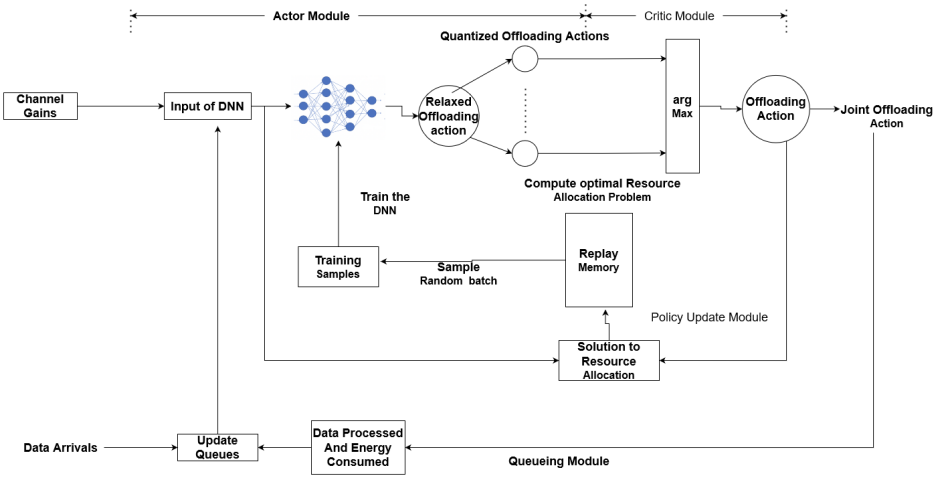



Figure 2: Schematic Of RDRL

1. Actor Module: This module takes an input and generates a list of offloading actions. It comprises of DNN and an quantizer for actions. $\theta^i$ is the parameter for DNN which for the $i$ th frame of time and is initialized randomly using standard normal distribution for $i = 1$. Based on the input parameter $\xi^i$, DNN generates a relaxed decision for offloading which is then quantized into binary offloading actions.
   The universal approximation theorem states that we can approximate accurately continuous mappings of any kind with right activation function acting(eg Sigmoid, ReLu, etc.) on neurons and with sufficient number of neurons in its multi-layer perceptron. For our framework we will be using sigmoid activation function for the output layer. After applying the activation functions we quantize the relaxed decision for offloading into $M_i$ probable candidates for offloading action. Here $M_i$ is a parameter that is time dependent. An ideal quantization function should take care of the exploration-exploitation trade-off while formulating the binary offloading actions to ensure optimal training convergence. For effective usage of DNN output, and to also separate premature convergence to sub-par values in the process of training, $x_n^i$ should be near to the feasible offloading actions(the Euclidean distance).
   For this reason we use the method of noisy order-preserving(NOP) quantization which suggests $M_i \leq 2N$ offloading actions. In NOP the first $M_i/2$ actions are generated by using order-preserving qunatizer(OPQ) on the feasible offloading actions. The next $M_i/2 - 1$ actions are generated by using the order of entries of feasible offloading actions based on relative distance to 0.5. To generate the next $M_i/2$ actions, a noisy feasible offloading action is generated using Sigmoid function and random Gaussian noise. The last $M_i/2$ actions are produced by again using OPQ on the elements generated by Sigmoid function.

2. Critic Module: The following module will be the critic module which processes prospective offloading actions generated by the action module and picks the best offloading boolean i.e. $x^i$. As compared to the orthodox actor-critic paradigm who use a model independent DNN as the critic to generate the offloading action, RDRL uses the model to evaluate and select the binary offloading action by finding the solution to the resource allocation equation optimally. This makes the offloading action generated by the critic module and its evaluation more accurate and thus making it robust and the process of DRL training converge faster.
   The calculation to find the best offloading action is performed $M_i$ times. This means that a larger value of probable candidates to offload will mean that a better candidate will be selected but it will also means that the time complexity of the algorithm also increases with it. To overcome this we suggest an adaptive procedure that takes care of the performance-complexity trade off based on time-varying value of $M_i$.
   The main logic behind this is that as the DNN approaches the right policy iterating $M_i$ over time, even a small value of $M_i$ can find the correct offloading action within nearest distance in the feasible offloading actions. We can represent the index of the best offloading action as $m_i$ and it can be found by finding the modulus of the index and $M_i/2$ which can be either the noiseless or the noise candidate action.
   We initially set the probable candidates for offloading as $2N$ and update $M_i$ after small interval of time frame($\delta_M$). We need to take care of time frames for updating because too frequent updates can degrade training of DNN and too big $\delta_M$ will

increase the time complexity of the algorithm.

3. Policy Update: RDRL updates the policy of DNN using the output from the critic module which acts as the sample for it. RDRL maintains a replay memory which has the capacity to store only $q$ number of data samples which are most recent. Initially the replay memory is empty and the DNN starts training periodically at intervals of $\delta_I$ to overcome model over-fitting. A random batch of selected samples($S_i$) is picked up when the modulus of $i$ th time frame and $\delta_T$ is 0, and using this we minimize the function relating to loss of average-cross-entropy using Adam Algorithm discussed in Zhang (2018). Once the training is complete for the current time frame we then update the actor module with the parameter for the next time frame.

4. Queuing Module: The output of the critic module gives us the best resource allocation $y^i$ for given $x^i$. Based on this the system consumes energy $\{e_n^i\}_{n=1}^N$ and process data $\{D_n^i\}_{n=1}^N$ for finding the appropriate offloading and resource allocation using the algorithm in Bi and Zhang (2018). Depending on the consumed energy and the data processed the queuing module updates the energy queue and data queues when the new time frame starts($i+1$). The system then inputs the new channel gains, and also the input parameter for the next time frame to DNN to start the iteration in actor module.

# 4 Implementation

## 4.1 Deep-Reinforcement Learning(DRL)

Over through the previous decade, an enormous quantity of data has been created and saved and analyzed via the Cloud. Cloud computing refers to the leasing of data storage and processing. Cloud computing has spread to numerous businesses, and the paradigm allows them to retrieve stored information remotely from any point on the network or the web. Its goal is to consolidate processing, storage, and infrastructure management in the Cloud for data centers, IP networks, and cellular core networks. Clouds' massive resources may then be utilized to provide elastic computational storage and processing power to end systems with restricted resource availability. Numerous Internet businesses have been expanding quickly thanks to cloud computing. Mao, You, Zhang, Huang and Letaief (2017) A network architectural idea called Mobile Edge Computing (MEC) provides IT and cloud computational resources at the edge of a cell network. A nearby edge server (ES) can be used by Wireless devices to offload energy and time-consuming computations by the online mobile-edge computing offloading (MEC) which is a major advantage to IoT devices. Reinforcement Learning (RL) is utilized to address the issue of an ever-changing dynamic system. Deep reinforcement learning also known as DRL, is an improved version of RL which is combined with the Deep neural network (DNN) to form a data-driven approach that tackles the online offloading problem and gives out the long-term objectives such as maximizing the data processing rate. This approach is considered for the multi-user wireless MEC system, which involves utilizing the DNN that can educate itself with the appropriate mapping from the status, including time-varying characteristics, to information offloading selections and resource provisioning action. It thus increases the reward by repeated interactions with the environment and is advantageous over the MINLP approach and delivers good Quality of Service. Liu, Yu, Xie and Zhang (2019)

**input :** Parameters $V, \{\gamma_n, c_n\}_{n=1}^{N}, K$ training interval $\delta_I, M_i$ update interval $\delta_M$;
**output:** Control actions $\{x^i, y^i\}_{i=1}^{K}$;

1. Initialize the DNN with random parameter $\theta^1$ and empty replay memomry, $M_1 \leftarrow 2N$
2. Empty initial data queue $Q_n(1) = 0$ and energy queue $Y_n(1) = 0$ for $i = 1, \ldots, N$;
3. **for** $t = 1, 2, \ldots, K$ **do**
4. Observe the input $\xi^i = \{h^i, Q_n(i), Y_n(i)\}_{n=1}^{N}$ and update $M_i$ if $\text{mod}(t, \delta_M) = 0$;
5. Quantize the relaxed offloading action with the DNN;
6. Quantize the offloading action into $M_i$ binary actions using NOP method;
7. Compute the solution to the problem by optimizing the resource allocation $y_n^i$ for each $x_n^i$;
8. Select the best solution for maximum value of the problem function and execute the joint offloading action;
9. Update the replay memory by adding the of relaxed offloading action
10. **if** $mod(i, \delta_I) = 0$ **then**
11. Unifromly sample a batch of data set from the memory
12. Train the DNN with the data set and update $\theta^i$ using Adam algorithm;
13. **end**
14. $t \leftarrow t + 1$;
15. Update the queue based on the previous time frame offloading action and resource allocation.
16. **end**

Figure 3: Algorithm For RDRL

RDRL has 4 layers comprising of single input layer, There are two hidden tiers and one output unit. The input and output layers each contain 120 neurons, while the hidden layers each have 80 neurons.
In RDRL the DRL works on the basis of reward driven behaviour in which based on the reward of the actions performed by it the algorithm decides the optimal task offloading decision. In DRL a machine learning algorithm observes a state in a given time frame. When the DRL agent performs an action the state changes and this makes the decision to change for the algorithm.

## 4.2 Lyapunov optimization

Lyapunov optimization is a framework that is well-known for studying the Online computation offloading technique. Recent studies have attempted to use Lyapunov optimization to develop an online compute offloading approach in MEC networks which long-term performance assurance. The Lyapunov optimization is aimed to increase the capability by

constructing the online offloading technique, data transmission processing can be governed by long-term data queue stability and overall average power limits. The MINLP algorithm, is the continuous approach and pragmatic in that selections for every timespan are performed without assuming the foreseeable insights of stochastic network conditions and data deliveries. The problem is formulated as (MINLP) issue which makes resource provisioning choices in the system in consecutive timescales, and binary offloading, in which the device computes the job possibly natively or at the Edge server. I t allows the WD to process data tasks both at the Edge server and locally parallelly and applies optimization technique for allocation of resources and continuous joint offloading problem. These heuristic techniques, however, cannot ensure good solution quality on a constant basis, which might potentially lead to lead to a decline in long-term performance. Mao, You, Zhang, Huang and Letaief (2017)

### 4.3 PyCharm

For development we used Pycharm IDE as it provides user friendly environment to code and debug the code. It also provides an easy to access terminal via which you can run the Python scripts written. To install Pycharm on your system you can download the Pycharm installer from their official website and then run it.

### 4.4 Code flow

The initial parameters with which the code runs is number of users, number of time frames, $\kappa$, decoder mode i.e. OP, KNN or OPN, memory capacity, interval for adaptive $\kappa$. Then we generate the channel and call the Queueing module of the framework for initial generation of queue.
After this the actor module is called which creates a batch of actions based on which the framework trains and records the largest reward value. The critic modules is called within a list where it records all the resource allocated for the generated offloading modes. The policy update module encodes the mode with largest value and stores the max result.
We plot the average data queue and average energy consumption using matplot library. Matplot library is a that is used for visualization in python.
We then run the above code with varying parameters like the value of $V$, the varying data arrival rates, the number of devices.

## 5 Evaluation

The simulations were run on a machine with Intel Core i5 9300H 2.4GHz and 8GB of memory and 4GB of NVIDIA GeForce GTX 1650 graphic memory. For the computation we are using TensorFlow 2.0.
The simulation was carried out with default values as given in the table below:

### 5.1 Computational Complexity

RDRL execution mainly involves of two parts: generation of actions for offloading and updation of policy. The offloading action needs to iterated in every action but updation of policy occurs infrequently. Hence we focus on finding the complexity of offloading

Table 1: Initial Values For System Parameter

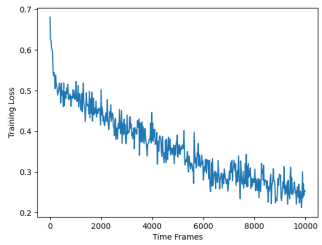
| | |
|---|---|
| Memory Size | 1024 |
| Lyapunovs Parameter | 20 |
| Data Arrival Rate | 3 Mbps |
| Power constraint | 0.08 W |
| Number Of Users | 10 |

action for every time frame.

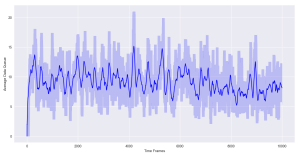
The offloading action complexity will be $O(Nlog_2(\frac{\Delta}{\sigma_0}) + N^3L)$ where the first parameter is the parameter for bisectional search and the second parameter is the time for solving the LP where $L$ is the input length of the binary representation. In comparison to solving a complex problem with 4N variables, RDRL solve the LP in N variables. Since RDRL executes $M_i$ times in a given time frame the complexity of executing the algorithm will be $O([Nlog_2(\frac{\Delta}{\sigma_0}) + N^3L]M_i)$. Since the $M_i$ reduces with the learning process of RDRL, hence the number of iterations required to generate optimal offloading action reduces.

## 5.2 Experiment 1

For the first experiment we take into account values of $\lambda_n$ rates that is 2.5 Mbps and 3 Mbps. We then plot the various graphs for rate of weighted sum computation, average size of the data queue, average performance for power consumption over time. We have an i.i.d. distribution of $10,000$ frames, where every part of graph is a partition of 200 frames. From the graphs we can analyze that the data queues are stable for computation rate for both the data arrival rates conditions. The average power limitation of 0.08 W is also satisfied by RDRL. For higher data rate RDRL also the same results are shown. For both data rates it take more time initially to to learn the correct offloading policy because of abrupt increase in the queue length of data i.e. for $i \leq 3,000$. But, as the DNN iterates through the offloading policies the queue length drop showing faster convergence.



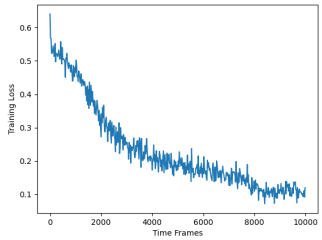

(a) Training Loss



(b) Average Queue Length

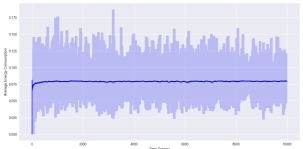
(c) Average Energy Cost

Figure 4: Graphs for $\lambda = 2.5$




(a) Training Loss

(b) Average Queue Length

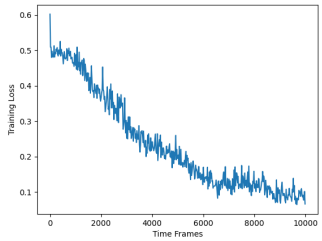


(c) Average Energy Cost

Figure 5: Graphs for $\lambda = 3$

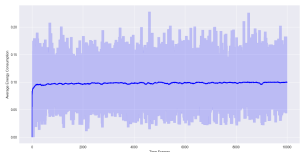
## 5.3 Experiment 2

In this experiment we will check the affect of system parameters of power constraint on RDRL. For this we fix the data arrival rate to 3 Mbps and change the power constraint between 0.06 and 0.1. We see that in the figure the data queue is stable for RDRL for all the values of power constraint and the length of the decreases with less stringent constraints of power. IT also shows that RDRL has great computational rates for all the cases considered.




(a) Training Loss

(b) Average Queue Length



(c) Average Energy Cost

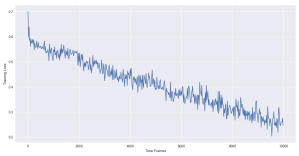
Figure 6: Graphs for power constraint= 0.1

## 5.4 Experiment 3

In this experiment we vary the control parameter for Lyapunov i.e. $V$. We observe that when Lyapunov parameter is small i.e. $V \leq 40$ the length of the queue and power constraint are inversely proportional to the value of Lyapunov parameter. The reason behind this is that the number of offloading candidates increase for WD as the value of Lyapunov parameter increases. For $V \geq 40$ the data queue and power consumed all increase with Lyapunov parameter because the increase in the number of offloading candidates means for one WD means decrease in the number of offloading candidates for another WD. Hence we should set a moderate value of Lyapunov parameter to reduce the buffer for task data.

## 5.5 Experiment 4

For this experiment we vary the number of WDs between the value of $10, 20\&30$. The WDs are placed within $[120, 255]$ meters distance from ES. Since the value of $M_i$ is time varying the performance for all these scenarios is satisfactory.

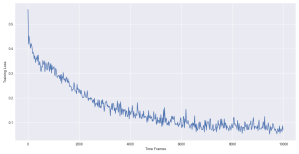


(a) Training Loss

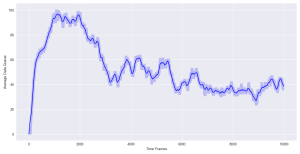
(b) Average Queue Length

(c) Average Energy Cost

Figure 7: Graphs for $V = 40$



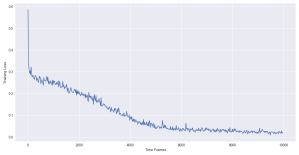
(a) Training Loss



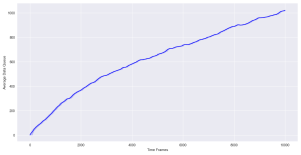
(b) Average Queue Length

(c) Average Energy Cost

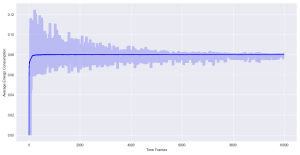
Figure 8: Graphs for $N = 20$



(a) Training Loss



(b) Average Queue Length



(c) Average Energy Cost

Figure 9: Graphs for $N = 30$

# 6 Conclusion and Future Work

In this paper we have discussed an reliable computation offloading framework in MEC network with multiple users and stochastic channel with task arrivals in the form of data. We propose an RDRL that leverages Lyapunov optimization and DRL to solve the MINLP problem which maximizes the number of bits processed by the WDs with stable queues and power constraints. It was a challenge because the WD has to make the decision of binary offloading and also the allocation of resource in short frames of time without the knowledge of future channel conditions and the data queues.

We have proven that RDRL achieves the computation rates and stable data queues with in the required constrained average power. Also RDRL converges to the offloading action with less number of iterations thus reducing the offloading time.

The proposed RDRL can also be used for partial offloading where in the task has multiple sub-tasks. This can be achieved by meticulously choosing the binary variable to point to the subtasks of the task that needs to be offlaoded to the ES.

We have discussed RDRl being implemented with TDMA access in this paper, but RDRL can be used with any access method like CDMA, OFDMA, FDMA, etc. The only thing that we need to take care of is the critic module whether it can predict the optimal resource allocation.

# 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): 4177–4190.

Chen, X., Zhang, H., Wu, C., Mao, S., Ji, Y. and Bennis, M. (2019). Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning, *IEEE Internet of Things Journal* **6**(3): 4005–4018.

Dinh, T. Q., Tang, J., La, Q. D. and Quek, T. Q. S. (2017). Offloading in mobile edge computing: Task allocation and computational frequency scaling, *IEEE Transactions on Communications* **65**(8): 3571–3584.

Du, J., Yu, F. R., Chu, X., Feng, J. and Lu, G. (2019). Computation offloading and resource allocation in vehicular networks based on dual-side cost minimization, *IEEE Transactions on Vehicular Technology* **68**(2): 1079–1092.

Du, J., Yu, F. R., Lu, G., Wang, J., Jiang, J. and Chu, X. (2020). Mec-assisted immersive vr video streaming over terahertz wireless networks: A deep reinforcement learning approach, *IEEE Internet of Things Journal* **7**(10): 9517–9529.

Lee, G., Saad, W. and Bennis, M. (2019). An online optimization framework for distributed fog network formation with minimal latency, *IEEE Transactions on Wireless Communications* **18**(4): 2244–2258.

Li, J., Gao, H., Lv, T. and Lu, Y. (2018). Deep reinforcement learning based computation offloading and resource allocation for mec, *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–6.

Li, K., Tao, M. and Chen, Z. (2020). Exploiting computation replication for mobile edge computing: A fundamental computation-communication tradeoff study, *IEEE Transactions on Wireless Communications* **19**(7): 4563–4578.

Liu, C.-F., Bennis, M., Debbah, M. and Poor, H. V. (2019). Dynamic task offloading and resource allocation for ultra-reliable low-latency edge computing, *IEEE Transactions on Communications* **67**(6): 4132–4150.

Liu, Y., Yu, H., Xie, S. and Zhang, Y. (2019). Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks, *IEEE Transactions on Vehicular Technology* **68**(11): 11158–11168.

Mao, Y., You, C., Zhang, J., Huang, K. and Letaief, K. B. (2017). A survey on mobile edge computing: The communication perspective, *IEEE Communications Surveys Tutorials* **19**(4): 2322–2358.

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): 3590–3605.

Mao, Y., Zhang, J., Song, S. H. and Letaief, K. B. (2017). Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems, *IEEE Transactions on Wireless Communications* **16**(9): 5994–6009.

Min, M., Xiao, L., Chen, Y., Cheng, P., Wu, D. and Zhuang, W. (2019). Learning-based computation offloading for iot devices with energy harvesting, *IEEE Transactions on Vehicular Technology* **68**(2): 1930–1941.

Sun, Y., Zhou, S. and Xu, J. (2017). Emm: Energy-aware mobility management for mobile edge computing in ultra dense networks, *IEEE Journal on Selected Areas in Communications* **35**(11): 2637–2646.

Tang, M. and Wong, V. W. (2022). Deep reinforcement learning for task offloading in mobile edge computing systems, *IEEE Transactions on Mobile Computing* **21**(6): 1985–1997.

Wei, Y., Yu, F. R., Song, M. and Han, Z. (2019). Joint optimization of caching, computing, and radio resources for fog-enabled iot using natural actor–critic deep reinforcement learning, *IEEE Internet of Things Journal* **6**(2): 2061–2073.

Xiao, L., Lu, X., Xu, T., Wan, X., Ji, W. and Zhang, Y. (2020). Reinforcement learning-based mobile offloading for edge computing against jamming and interference, *IEEE Transactions on Communications* **68**(10): 6114–6126.

Yan, J., Bi, S., Zhang, Y. J. and Tao, M. (2020). Optimal task offloading and resource allocation in mobile-edge computing with inter-user task dependency, *IEEE Transactions on Wireless Communications* **19**(1): 235–250.

You, C., Huang, K. and Chae, H. (2016). Energy efficient mobile cloud computing powered by wireless energy transfer, *IEEE Journal on Selected Areas in Communications* **34**(5): 1757–1771.

Zhang, J., Du, J., Shen, Y. and Wang, J. (2020). Dynamic computation offloading with energy harvesting devices: A hybrid-decision-based deep reinforcement learning approach, *IEEE Internet of Things Journal* **7**(10): 9303–9317.

Zhang, Z. (2018). Improved adam optimizer for deep neural networks, *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*, pp. 1–2.