This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2022.3145056, IEEE Transactions on Network and Service Management

DECENT: Deep Learning Enabled Green Computation for Edge centric 6G Networks

Pankaj Kumar Kashyap, Sushil Kumar, *Senior Member IEEE*, Ankita Jaiswal, Omprakash Kaiwartya, *Senior Member IEEE*, Manoj Kumar, Upasana Dohare, Amir H. Gandomi, *Senior Member IEEE*

***Abstract*— Edge computing has received significant attention from academia and industries and has emerged as a promising solution for enhancing the information processing capability at the edge for next generation 6G networks. The technical design of 6G edge networks in terms of offloading the computationally extensive task is very critical because of the overgrowth in data volume primarily due to the explosion of smart IoT devices, and the ever-reducing size of these energy-constrained devices in IoT systems. Toward harnessing the benefits of deep recurrent neural network based on Long Short Term Memory (LSTM) in the design of next-generation edge networks, this paper presents a framework DECENT- Deep learning Enabled green Computation for Edge centric Next generation 6G neTworks. The data offloading problem is modeled as a Markov decision process considering joint optimization of energy consumption, computation latency, and offloading rate for network utility in 6G environment. The algorithm learns faster from previous long-term offloading experiences and solves the optimization problem with better convergence speed. Simulation results of the proposed framework DECENT shows that it maximizes the network utility by overcoming the challenges as compared to the state-of-the-art techniques.**

***Index Terms*– Edge computing, LSTM, Next generation 6G network.**

I. Introduction

INTERNET of Things (IoT) along with the evolution of 5G supports massive associations among machines, humans, and smart devices [1]. 6G-enabled next generation IoT opens a new paradigm to various computation-intensive and delays constraints applications such as augmented virtual reality, unmanned aerial vehicle, tele-surgery, interactive games and facial recognition systems [2]. These smart things need to be self-sustainable for long-time services in order to effectively support smart industry, smart cities, healthcare devices and environment surveillance. These applications produce exponentially increasing traffic and required strict services such as latency, computation load, sensitivity and wireless communication in the 5G and beyond 5G (B5G) network [3]. There will be 29.3 billion networked devices by 2023, up from 18.4 billion in 2018. The share of Machine-To-Machine (M2M) connections will grow from 33 percent in 2018 to 50 percent by 2023 [4]. Moreover, limited computation capability and limited battery-powered energy are always a bottleneck of these smart things.

P. Kashyap, S Kumar, A. Jaiswal, M. Kumar are with the School of Computer & Systems Sciences, Jawaharlal Nehru University, New Delhi, India. Email: pankaj76_scs@jnu.ac.in, skdohare@mail.jnu.ac.in, ankita79_scs@jnu.ac.in, manoj26_scs@jnu.ac.in.

O. Kaiwartya is with Department of Computer Science, Nottingham Trent University, UK Omprakash.kaiwartya@ntu.ac.uk.

U. Dohare is with Department of Computer Science and Engineering, IIMT College of Engineering, Greater Noida, UP, India Email: Upasana.dohare_gn@iimtindia.net

A. H. Gandomi is with the Faculty of Engineering and Information Technology, University of Technology Sydney, Ultimo, NSW 2007, Australia (e-mail:gandomi@uts.edu.au).

While 5G network maturing towards B5G, the numbers of commercial applications and services are growing. Those have far-reaching impact on our life with extremely diverse set of quality of requirements. Which exhausts the network resources of existing 5G networks and trigger the use cloud–based mobile-edge computing (MEC) and their servers. Therefor in the design of B5G network that is adaptive, intelligent and extremely flexible for heterogeneous services; 6G networks connects massive devices focus on lifetime maximization by reducing the energy consumption and latency for MEC servers. This is because of 6G networks offers fast and ultrareliable communication with higher data rate for cloud servers (data offloading at the edge of network) with low latency [5]. However, there are still problems in data offloading in edge centric 6G networks as follow: (i) 6G offers millions of IoT devices connected simultaneously and it requires dynamics services frequently, then even resource rich MEC server stuck in some cases; (ii) Installation and maintenance of high-computation enabled MEC at the edge of network are costly, so proper number of MEC server in the environment is challenging task; (iii) 6G offers extended spectrum up to 40-50 BSs/Km^2, so that end users may be captured by more than one MEC server. Thus selecting appropriate MEC server for computation offloading to reduce energy consumption and latency is also a complex task. However, joint application of MEC server and 6G for IoT in future communications will be needed for such applications.

Energy harvesting enabled smart things acquires energy from environmental resources: solar, wind, vibration, and thermal. It can also be harvested from electromagnetic fields created by cellular networks such as radio frequency (RF) signals are known as wireless power transfer. However, deployment of a battery charger or release of a periodical energy beacon source at accurate locations in the IoT network is a tedious challenge, and it increases the cost of entire network. In addition, RF causes the emission of greenhouse gases and provides significantly lower amounts of energy as compared to renewable energy sources [6]. Also, it is more preferable due to freely available anywhere (harsh conditions too) and during the day (both solar and wind) and night (wind) and do not harm the global climate. However, unpredictable natural energy source increases complexity for implementation due to (i) its inherent stochastic properties i.e., randomness in energy arrival rate with respect to time i.e., precisely depends upon weather conditions (ii) time-varying channel condition.

The overgrowth of data by computation limited and delay-sensitive IoT devices and their applications represents a new challenge to 6G-IoT networks. MEC network is a promising technique for providing computation capability at the edge of IoT devices with resource-rich high-performance MEC servers [7]. MEC servers are densely deployed in IoT networks, which

1932-4537 (c) 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Nottingham Trent University. Downloaded on January 27,2022 at 11:57:16 UTC from IEEE Xplore. Restrictions apply.

provide low latency computation task, high bandwidth, low energy consumption, and improve the quality of services. In the case of event triggering, IoT nodes generate an enormous amount of data and these data must be transmitted. If the IoT node does not have enough energy, then data should be stored in the buffer and when a sufficient amount of energy becomes available, stored data as computation task are transmitted to MEC servers wirelessly in two ways: full or partial offloading. When the IoT device offloads its full computation task to MEC servers, it is known as full offloading; when a portion of the computation task is offloaded, it is called partial offloading. On the other hand, an MEC server processes the offloaded task and sends back the results to edge devices. One critical problem in MEC network is how much of the computation task should be offloaded to MEC servers, i.e., determination of local processing rate and offloading rate. Therefore, to design an energy-efficient algorithm with delay constraints for EH-MEC server-based IoT networks becomes an interesting research topic. In this paper, we jointly optimize the individual local processing rate and offloading rate and the number of IoT devices allocated to MEC servers in order to maximize the utility of 6G-IoT network.

In recent years, numerous studies have been published that design an optimal computation offloading rate on MEC servers and resource allocation for energy savings in EH-IoT devices under different design objectives [8-12]. In [8], the authors optimized the number of CPU cycles using an online reinforcement learning algorithm (RL) to save energy consumption and to allow multiple users to process the data locally or offload (full) to MEC server. In [9], the authors defined a layer-based software-defined network to minimize the application level delay for edge outlets. In [10-12], the authors focused on a decentralized approach that jointly optimizes the mobile users preceding matrices and saves energy based on game theory. In [13-14], Deep Q-network model used in multi-users environment that simultaneously offload their task to one MEC server and the problem is defined as cost, such as summation of offloading delay of all users plus energy consumed in offloading. However, in the mentioned literatures, computation task offloading problem to MEC server has been taken up, where edge devices execute an entire task either locally or at a server by offloading it. Partial offloading and time-varying channel information is not prioritized for next generation 6G network environment.

In this context, toward harnessing the benefits of deep recurrent neural network (RNN) based LSTM; remembering the long sequence of information over time and predict the output based upon previous computation. We present a framework DECENT to optimize the strategies for offloading the task and minimizing the delay, where both EH-IoT devices and time-varying channel condition is considered. The major contributions are as follows:

1) A system model is presented focusing on the next generation cluster formation with MEC as cluster head, energy consumption, and green computation that support data offloading in 6G edge network.
2) The problem of high-volume data offloading in the next generation 6G edge network is formulated as network utility maximization problem the considering the energy consumption and computation task offloading constraints. Further, maximization problem is converted into Markov Decision Problem to implement LSTM network.
3) DECENT-Deep learning enabled solution is developed for green computation at next generation 6G edge networks focusing on scientific workflow and the algorithm using RNN LSTM network.
4) The convergence property of the presented DECENT algorithm is analyzed; i.e., it avoids the vanishing or exploding gradient value. In addition, computational complexity depends on the weight (ψ) of the LSTM network and it is computed in polynomial time of $O(\psi)$.
5) The performance of DECENT is tested over critical LSTM unit i.e. memory size, mini batch-size, training interval and learning rate. Further, comparative simulations are performed along with state-of-the-art techniques to show the benchmarking results of DECENT.

The remainder of this paper is organized into the following sections. Section II explored literature as related works. Section III and IV present the details of the scientific modeling of the proposed DECENT framework. Section V discusses simulation results and analysis. Finally, the conclusions of the paper are presented in section VI.

II. Related Work

In this section, related pieces of literature regarding offloading rate, application-processing delay, and energy consumption in the MEC network are reviewed. As the information regarding generation of random data by IoT device and time-variant channel condition together makes challenging task to optimize the offloading policy in real-time MEC network, especially when the unknown amount of energy is harvested within given time period. In the 6G edge network, only local performance metrics are known in the current time slot. Thus, researchers have preferred RL technique to solve the problem of statistical uncertainty of environment and channel. In [15], centralized and decentralized Q-learning (QL) algorithms were proposed that minimize energy consumption and maintain the quality of services in random heterogeneous cellular network observed by mobile users. In [16], the authors proposed a policy-gradient based actor-critic algorithm for scheduling EH-IoT nodes for traffic offloading that provides optimal resource (channel) to maximize the network energy efficiency. The authors did not consider the data generation rate by IoT devices, and offloading rate; the algorithms fail to compete in a real-time environment when the state and action space is large. In [17], the authors proposed after-state QL algorithm with a polynomial approximation to diminish the curse of dimensionality to estimate the offloading policy for the EH-IoT devices. As the authors did not take the partial offloading of the computation task, IoT devices experienced a delay in time-responsive applications. In [18], computation task execution latency was considered in addition to wireless channel condition; computation-offloading algorithm was proposed in a multiple computing node environment subject to minimizing the task's execution latency and energy consumption. User fairness was jointly considered in [19] with channel condition and execution latency as compared to [18] and designed an offloading scheme. However, there are limitations to the above works; they are only beneficial for single task application and not suitable for multimedia streaming applications where execution consists of multiple tasks. Above literature uses QL approach to learn the offloading rate to reduce the execution latency and energy consumption. However, the learning rate is very slow because exploration and exploitation strategies suffer from dimensionality of Q-table, which fails in the case of large state space and action space. Thus, long-term stable profit is not gained in the 6G edge network.

Aiming to solve these limitations, neural network based deep reinforcement learning (DRL) framework was introduced into MEC to overcome the curse of dimensionality and experience a fast convergence rate. DRL have learning ability from the past offloading strategy of an IoT device in time-varying energy, computation task, channel condition, and eventually produces mapping from state to action. In [20], the authors proposed a deep Q-network learning (DQL) algorithm to access the network for offloading the computation task subject to minimizing the user's cost. In [21], the authors proposed a DQL method in a multiuser environment to train the network based on past experiences in order to jointly take the offloading decision and resource allocation based upon the overall cost of all users and the capacity of the MEC server. In [22], the authors proposed a deep deterministic policy gradient method, where IoT devices can decide to choose either RF communication or low-power backscatter communication for balancing the energy consumption in computation and data offloading. In [23], the authors proposed a Dyna architecture-based offloading rate decision algorithm that was used to secure both the location privacy and data pattern privacy for healthcare EH-IoT devices. The proposed algorithm failed to compete in a multi-user environment in order to provide interactive performance to multimedia applications. In the Internet of Vehicles, MEC servers are replaced by vehicle edge computing (VEC) servers that provide services to nearby vehicles [24], authors proposed a DQL-based algorithm for optimizing the offloading rate of data, while considering both the delay of computation task and limited computation capabilities of vehicles. In [25][26], the authors took advantage of parallel computing and proposed a distributed DRL offloading algorithm where multiple wireless devices simultaneously generate an offloading decision. However, above studies based on a DQL network suffer from the curse of the dimensionality problem when channel quantization and energy harvesting capability requires higher accuracy.

Not much work has been performed that was focused on joint optimization of the task offloading strategy and resource allocation in a multiuser access control MEC server environment. In addition, time complexity and learning rate are usually large and slow, and above mentioned algorithms are not feasible in a real-time MEC network. Fortunately, LSTM has two changes in existing DQL: first, the network has cell memory that is used to remember past offloading experiences and able to execute long sequence of samples; second, the network is trained with mini-batch samples to minimize the correlation between samples and then the target Q network is provided to regularly update the network weights [28]. Recently in 2017, Transformers is first kind of transduction neural network architecture introduced and shows better improvement still in development phase. Transformers use behavior self-attention to convert input sample to output without relying on sequence-aligned RNN [27]. However, Transformers solves the major problem of parallelization that inhibits in sequential LSTM and probability of forgetting the information exponentially from a sample that is far away from the current sample by encoding each word of sample into hidden state then passed to decoding stage and paying more attention to current sample respectively. But, the major challenge is to build Transformers requires sheer size, cost and processing requirement such as for parallelization requires Pre-Trained unsupervised model- Google's BERT or GPT , which faces customization of hyper parameters (encoders and decoders) for the specific applications to get proper results and these model are limited to transduction fixed-length samples (BERT= 512 characters at a time). Further, the parallelization causes fragmentation of context without worrying about semantic and respect of sample then it could suffers from loss of significant information. In this paper, we use LSTM-based deep RNN to train and learn the network, which overcomes the traditional DQL based RNN issue and of long–term dependencies of input-output networks pair and is able to predict the optimal offloading rate for next time slot in order to minimize energy consumption, computation delay and task drop ratio in economical and resolve the context fragmentation issue by using gates and sequential processing.

III. System Model

A. *Next Generation Edge Network*

We consider an uplink MEC network with N number of EH-IoT nodes (e.g. smart sensors, smartphones, or smart watches), and multiple MEC servers; i.e., $\mathfrak{M} = \{1,2 \dots m, .. M\}$ for complete coverage of the network, as shown in Fig. 1. The MEC servers have a fixed location and IoT nodes are randomly scattered over the network. The IoT node has the capability to do less extensive computation tasks locally or can offload all or partial computation tasks to its nearby MEC server and store the remaining task in its buffer. The IoT node is battery-powered renewable ambient energy sources.

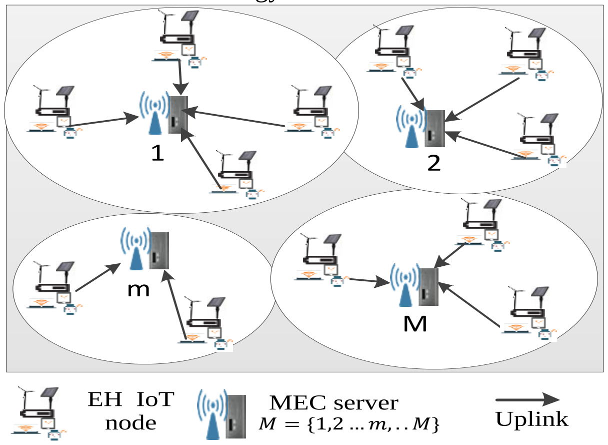

Fig.1: Illustration of a MEC server cluster network

The MEC network operates based-on time slots $T = \{1,2, \dots .\}$ which have an equal time span. Each time slot is divided into two parts ($t_1 and\ t_2$, $t_1 \ll t_2$). In t_1 time, MEC server makes a cluster by selecting its K number of nearby IoT

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2022.3145056, IEEE Transactions on Network and Service Management

nodes and divides the available channels into K number of orthogonal channels with equal bandwidth B. In t_2 time, MEC server chooses an optimal offloading rate (action) for each IoT node in its cluster to maximize the expected discounted reward or utility of the network. The action of the MEC server depends upon channel state information, estimated harvested energy, battery level, and the size of the data offloaded to the MEC server by each IoT node. Note that each IoT node is assigned to only one MEC server. The m^{th} MEC server chooses randomly '$K_m(T(t_1)) = \{1,2, \ldots k \ldots K\}$' number of IoT nodes out of N IoT nodes to form cluster at time slot $T(t_1)$, such that $\sum_{m=1}^{M} K_m(T(t_1)) = N$.

Where 'K' represents the number of orthogonal channels available in each cluster and each channel has an equal frequency bandwidth B. In $T(t_2)$, selected k_m IoT nodes send its status (battery level, estimated harvested energy, channel gain, generated data, and buffered data) to its MEC server. Further, MEC server decides their offloading rate based on the received information. Next, the member IoT nodes of a cluster compute some of the tasks locally and offload some of the computation tasks to their MEC server. Thereafter, the MEC server computes their task and returns the result back to the respective IoT nodes.

B. *Edge Energy Consumption And Channel*

For the simplicity, we consider a single cluster operation for task offloading to the MEC server for both maximization of the information rate and minimization of energy consumption within the cluster, as shown in Fig. 2. At the beginning of time slot T , the k^{th} IoT node has new sensing data of size $c_{m,k}^{g}(T)$ and buffered data of previous time slot $c_{m,k}^{b}(T-1)$ size in the cluster, with MEC server referred to as cluster head. The k^{th} IoT node has a total amount of data $C_{m,k}(T)$ for computation i.e., $c_{m,k}^{g}(T) + c_{m,k}^{b}(T-1)$ and it is partitioned into β equivalent parts, similar to the scheme referred in [29]. For delay-sensitive applications, we consider that the execution time for the computation task on an MEC server is no longer than time slot T. The reason behind this is if the data offloading at the MEC server does not occur within time slot T, then computation latency arises during offloading of the task, and the energy consumed during task computation is difficult to calculate, as the channel coefficient varies in each time slot. The IoT node partially offloads the data to the MEC server with an offloading rate of $D_{m,k}^{o}\big(T(t_2)\big)$ over the wireless uplink of the radio channel. Next, locally compute the data with a local execution rate of $D_{m,k}^{l}\big(T(t_2)\big)$. The remaining $\big[1 - D_{m,k}^{o}(T(t_2)) - D_{m,k}^{l}(T(t_2))\big]$ computation task is stored in the buffer for the processing that occurs in the next upcoming time slots, with $\{D_{m,k}^{o}(T(t_2)), D_{m,k}^{l}(T(t_2))\} \epsilon \left\{\frac{\beta_0}{\beta}, \frac{\beta_l}{\beta}\right\} 0 \le \beta_0, \beta_l \le \beta$. The IoT node is powered with a rechargeable battery of maximum battery capacity b_{max} . The IoT node harvests $e(T)$ amount of energy during time slot T and this harvested energy is used for the next time slot $(T+1)$ for local processing or offloading of the computation task. The function of the battery is considered to be ideal; i.e., the IoT node does not lose energy in storing or retrieving energy. We assume that energy consumption in the node is only due to local computation and data transmission. Once the battery reaches its maximum capacity b_{max}, the additional harvested energy is abandoned. The energy $e_{m,k}(T)$ harvested by the IoT node is dynamic in nature and is modelled as a Markov chain model with E number of quantized level energy state sets. The transition probability of $e_{m,k}$ from e^{x} to e^{y} during time slot T is given as $T_{m,k}^{x,y} = prob\left(e_{m,k}(T+1) = y \middle| e_{m,k}(T) = x\right) = e^{x.y}$,$\forall x, y \in E$. The amount of energy harvested by the IoT node during time slot T is calculated according to the profile energy prediction (Pro-Energy) model referred to in [32]. The transmission power $p_{m,k}(T(t_2))$ of IoT node holds the inequality, i.e., $p_{m,k}(T(t_2)) \le b_{m,k}(T)$, $p_{m,k}\big(T(t_2)\big) \ge 0$ with binary indicator $I_{m.k}(T(t_2)) = 1$, which means that offloading of the computation task was successfully performed and $I_{m.k}(T(t_2)) = 0$ shows that the IoT device failed to offload the computation task in current time slot T, as the IoT node does not have enough power for data transmission.

$$I_{m.k}(T(t_2))p_{m,k}(T(t_2)) \le b_{m,k}(T), p_{m,k} \ge 0, \forall m, k = 1,2 \ldots . \mathrm{k} \tag{1}$$

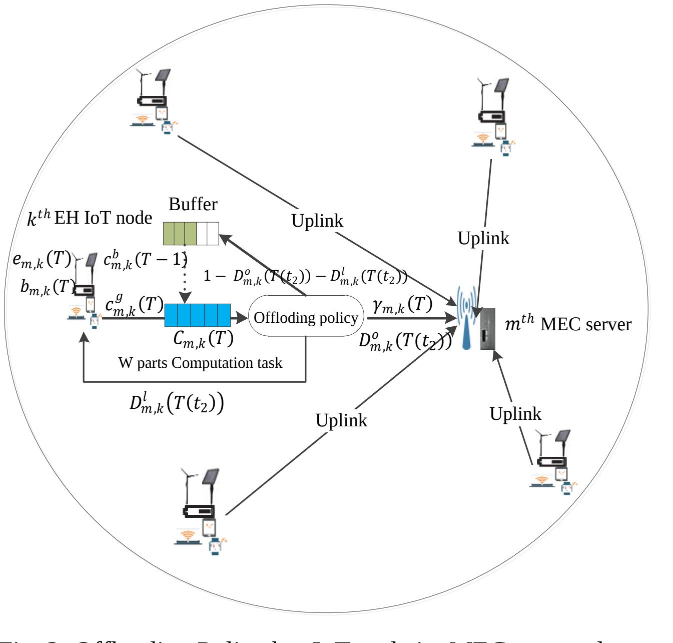

Fig. 2: Offloading Policy by IoT node in MEC server cluster

The channel gain $h_{m,k}(T)$ between m^{th} MEC server and k^{th} EH-IoT node is assumed to be stochastic in nature and is modelled as a Markov chain model, with transition probability $T_{m,k}^{u,v} = prob\left(h_{m,k}(T+1) = u \middle| h_{m,k}(T) = v\right) = h^{u,v}$,$\forall u, v \in H$, where H represents the number of quantized radio channel state sets. We assume that at the beginning of each time slot, the instantaneous channel power gain is obtained at the MEC server by feedback from IoT node. The signal-to-interference-plus-noise ratio (SINR) of k^{th} IoT node during time slot T is given as

$$\gamma_{m,k}(T) = \frac{h_{m,k}(T)\, I_{m.k}(T(t_2)) p_{m,k}(T(t_2))}{\sum_{i \epsilon k \setminus \{k\}} h_{m,i}(T) p_{m,i}(T) + \delta^2} \tag{2}$$

where $p_{m,k}(T(t_2))$ is the transmission power of k^{th} IoT node for offloading the computation task and δ^2 is the noise power

1932-4537 (c) 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Nottingham Trent University. Downloaded on January 27,2022 at 11:57:16 UTC from IEEE Xplore. Restrictions apply.

gain variance of additive white Gaussian channel with zero mean. The $h_{m,i}(T)$ *and* $p_{m,i}(T(t_2))$ represent the channel gain and transmission power of other nodes in the same cluster, respectively.

C. Edge Green Computation

1) Local Computing

We assume that ζ is the number of CPU cycles required to compute one bit. Therefore, the total number of cycles required to compute $C_{m,k}(T)D^l_{m,k}(T(t_2))$ bits is $C_{m,k}(T)D^l_{m,k}(T(t_2))\zeta$. The CPU can control the number of cycles required for local execution in each cycle l by adjusting the frequency f^ldefined as dynamic frequency and voltage policy [31]. The local execution latency defined by $L_l(T(t_2))$ during time slot T is given as

$$L_l(T(t_2)) = \sum_{l=1}^{C_{m,k}(T)D^l_{m,k}(T(t_2))\zeta} \frac{1}{f^l} \tag{3}$$

The energy consumed in local execution of a task by IoT node with ϖ as the coefficient of CPU effective capacitance at time slot T is calculated as

$$E_l(T(t_2)) = \sum_{l=1}^{C_{m,k}(T)D^l_{m,k}(T(t_2))\zeta} \varpi(f^l)^2 \tag{4}$$

2) Computation Offloading

The k^{th} IoT node offloads its computation task to the m^{th} MEC server. The uplink transmission rate $r_{m,k}(T)$ for task offloading can be calculated by considering the mutual interference due to simultaneous transmission of other IoT nodes as

$$r_{m,k}(T) = B\log_2(1+\gamma_{m,k}(T)) \tag{5}$$

where B is the uplink channel bandwidth at time slot T. The transmission delay for offloading $C_{m,k}(T)D^o_{m,k}(T(t_2))$ bits of data to the MEC server is given as

$$L_o(T(t_2)) = \frac{C_{m,k}(T)D^O_{m,k}(T(t_2))}{r_{m,k}(T)} \tag{6}$$

The energy consumption of the IoT node in offloading the computation task to the m^{th} MEC server with transmission power $p_{m,k}(T)$ depends upon computation delay, calculated as

$$E_o(T(t_2)) = L_o(T(t_2))p_{m,k}(T(t_2)) \tag{7}$$

The computation latency $L_{m,k}(T)$ of the IoT node depends upon local execution latency $L_l(T(t_2))$ and offloading delay to the MEC server $L_o(T(t_2))$ represented as

$$L_{m,k}(T) = \max\{L_l(T(t_2)),\ L_o(T(t_2))\} \tag{8}$$

Note that energy consumption in other operations, regardless of local computing and offloading, are assumed to be negligible. Then, the total energy consumption in the IoT node is the sum of energy consumed for local computation and offloading the computation task, calculated as

$$E_{m,k}(T) = E_l(T(t_2)) + E_o(T(t_2)) \tag{9}$$

At the beginning of time slot T , the battery level of the IoT node, which depends upon energy consumption, energy harvested, and previous battery level, is calculated as

$$b_{m,k}(T+1) = \min\{b_{max},\ b_{m,k}(T) + e_{m,k}(T) - E_{m,k}(T)\} \tag{10}$$

IV. DECENT- Deep Learning Enabled Green Computation For Edge Network

In this section, we formulated optimal data offloading problem as a Markov decision process (MDP) to represent the network utility maximization that jointly optimizes energy consumption, computation latency, and offloading rate. An LSTM based deep learning DECENT algorithm is presented to solve the MDP problem.

A. Optimization Problem Formulation

The performance of each cluster is measured in terms of utility $(U_m(T))$ function. Whereas, utility $U_{m,k}(T)$ of k^{th} IoT node (member) refer to m^{th}CH depends upon the offloading task, task drop ratio, energy consumption, computation latency and waiting cost $(Z(T))$ at buffer given as,

$$U_{m,k}(T) = C_{m,k}(T)D^o_{m,k}(T(t_2)) - \varphi I\big(b_{m,k}(T+1) < 0\big) - \partial E_{m,k}(T) - \nu L_{m,k}(T) - \vartheta Z(T) \tag{11}$$

where φ is the weight parameter for task drop rate and I is the binary indicator. If $I=0$, then the battery level of the IoT node in the current time slot is not enough for executing the computation task and therefore the data would be dropped. Let ∂, ν *and* ϑ denote the weight parameter for energy saving, computation latency, and waiting cost respectively. Finally, we formulate the utility at the MEC server as the sum of utility of each member node, given as

$$U_m(T) = \sum_{k=1}^{K} U_{m,k}(T) \tag{12}$$

The main objective of the proposed algorithm is to maximize the utility at each MEC server by deciding the number of member nodes, the offloading rate, and the local processing rate during time slot T. Accordingly, we formulate the optimization problem as

(P1) $\max \lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} U_m(t), \quad \forall, m = 1,2,..m\ldots M$ (13a)

Subject to $I_{m.k}(T(t_2))p_{m,k}(T(t_2)) \le b_{m,k}(T)$,

$\forall m, k = 1,2,..k\ldots K$ (13b)

$b_{m,k}(T+1) \le \min\{b_{max}, b_{m,k}(T) + e_{m,k}(T) - E_{m,k}(T)\}$,

$\forall m, k = 1,2,..k\ldots K$ (13c)

$\{D^o_{m,k}(T(t_2)), D^l_{m,k}(T(t_2))\}\epsilon\left\{\frac{w_0}{w}, \frac{w_l}{w}\right\} 0 \le w_0, w_l \le w$,

$\forall m, k = 1,2,..k\ldots K$ (13d)

$\sum_{m=1}^{M} K_m(T) = N$ (13e)

B. Optimization Problem Modelled as MDP

Here briefly define the three key elements of the RL includes state, action, and reward. Then, we formulate the utility maximization problem as a classical Q-learning RL for finding the solution. Note that in this paper, multiple IoT node

1932-4537 (c) 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Nottingham Trent University. Downloaded on January 27,2022 at 11:57:16 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2022.3145056, IEEE Transactions on Network and Service Management

offloading scenarios are considered. Thus, the number of states and actions are large for the RL agent on each MEC server. To avoid the curse of dimensionality, we further use a DQL-based LSTM network layer to estimate the long-range action-value function of correlated patterns of input and output for Q-learning. The RL agent on each MEC server jointly selects the number of IoT nodes to form clusters and also chooses both offloading rate and local processing rate to maximize the expected discounted long-term utility in current time slot T. The system state $S(T)$ is the controlled stochastic process of the network across time slot $T = 1,2 ....$ Generally, $S(T)$ can be extracted from M number of MEC server placed in the network at different locations and defined as follows

$$S(T) = \left(s_1(T),\ s_2(T),\ s_3(T), \dots\dots, s_m(T) \dots\dots, s_M(T)\right) \quad (14)$$

At the beginning of time slot $T(t_2)$, each IoT node evaluates it's $\mathrm{b}_{\mathrm{m,k}}$, $e_{m,k}$ and $\gamma_{m,k}$ then sends these information as causal knowledge to their MEC server. Now, MEC server uses this received information along with previous SINR value, and decides the offloading rate and local processing rate of IoT nodes. In (14), the $s_m(T)$ describes the configuration at m^{th} MEC server with its total k number of member nodes, with five elements: the SINR $\gamma_m(T)$ information, the current battery level $b_m(T)$, the new generated data $c^g_{m,k}(T)$, the buffered data $c^b_{m,k}(T-1)$ and energy harvested $e_m(T)$ of each member IoT node in current time slot T, as given by

$$s_m(T) = \left(\gamma_m(T), b_m(T), c^g_m(T), c^b_m(T), e_m(T)\right) \quad (15)$$
$$\gamma_m(T) = (\gamma_{m,1}(T), \gamma_{m,2}(T), \gamma_{m,3}(T) \dots\dots \gamma_{m,k}(T)) \quad (16)$$
$$b_m(T) = (b_{m,1}(T), b_{m,2}(T), b_{m,3}(T) \dots\dots b_{m,k}(T)) \quad (17)$$
$$c^g_m(T) = \left(c^g_{m,1}(T), c^g_{m,2}(T), D^g_{m,3}(T) \dots\dots D^g_{m,k}(T)\right) \quad (18)$$
$$c^b_m(T) = (c(T), c^b_{m,2}(T), c^b_{m,3}(T) \dots\dots c^b_{m,k}(T)) \quad (19)$$
$$e_m(T) = (e_{m,1}(T), e_{m,2}(T), e_{m,3}(T) \dots\dots e_{m,k}(T)) \quad (20)$$

By knowing the system state $S(T)$, the RL agent on each MEC server takes an action space $A(T)$, which ensures that the formatted cluster evaluates the maximum expected discounted reward. Action includes the selection of local processing rate and offloading rate for each member IoT nodes in different clusters during each time slot T, defined as

$$A(T) = (a_1(T), a_2(T), a_3(T) \dots\dots a_m(T) \dots\dots, a_M(T)) \quad (21)$$
$$a_m(T) = (D^l_m(T(t_2)), D^o_m(T(t_2))) \quad (22)$$

The primary goal of the RL agent on each MEC server is to maximize the reward $R(T)$ that is interpreted as the maximization of utility at each MEC server, which denotes the objective of problem (P1). The profit of the reward $R(T) = \left(S(T), A(T)\right)$ depends upon taking all the action $a_m(T) \epsilon A(T)$ in a certain state $s_m(T) \epsilon S(T)$ to maximize the expected discounted reward, given as

$$R(T) = (r_1(T), r_2(T), r_3(T) \dots\dots r_m(T) \dots\dots r_M(T)) \quad (23)$$
$$r_m(T) = \max \mathbb{E}\left[\textstyle\sum_{T=1}^{T} U_m(T)\right] \quad (24)$$

The RL agent on each MEC server aims to maximize the reward function in the long run by optimizing the policy. The long-term discounted expected reward during time slot T can be written as

$$R^{\gamma}(T) = \textstyle\sum_{j=T}^{\infty} \gamma^{k-T} R(T+1) \quad (25)$$

where γ represent the discount factor; if $\gamma = 0$, then a myopic situation occurs and the reward depends upon only on the transition from the current state to the next state. As γ approaches unity, the reward value depends upon the future value. For the classical RL network, the state action-value function $Q_\pi(s(T), a(T))$ under the policy π for solving the MDP is defined as follow

$$Q_\pi\left(s(T), a(T)\right) = \mathbb{E}_\pi[R^{\gamma}(T) |\ S(T) = s(T), A(T) = a(T)]$$
$$= \mathbb{E}_\pi\left[\textstyle\sum_{j=T}^{\infty} \gamma^{k-T} R(T+1)\, | S(T) = s(T), A(T) = a(T)\right] \quad (26)$$

The RL agent takes an action on each step and stores $Q_\pi\big(s(T), a(T)\big)$ in a Q-table. The updated state-action value $Q_\pi\big(s(t), a(t)\big)$ in one-step learning rate $(0 < \eta < 1)$ given as

$$Q_\pi\big(s(T), a(T)\big) = Q_\pi\big(s(T), a(T)\big) + \eta\Big[R(T+1) +$$
$$\gamma \max_{a(T)\epsilon A(T)} Q_\pi\big(s(T+1), a(T)\big) - Q_\pi\big(s(T), a(T)\big)\Big] \quad (27)$$

where $R(T+1)$ represents the immediate reward gain during time slot T. The goal of the RL agent on each MEC server is to maximize the expected discounted reward under optimal policy π^* at any state, so we can formulate equation (27) as an optimality equation in a recursive manner by using the Bellman equations, as follows

$$Q_{\pi^*}(s(T), a(T)) = \mathbb{E}\Big[R(T+1) + \gamma \max_{a(T)\epsilon A(T)} Q_{\pi^*}(s(T+$$
$$1), a(T) |\ s(T+1) = s, A(T+1) = a)\Big] \quad (28)$$

The objective of this learning is to find an optimal policy π^* to maximizes the long-term expected discounted reward, given as

$$\pi^*\big(s(T)\big) = \arg \max_{a(T)\epsilon A(T)} Q_{\pi^*}\big(s(T), a(T)\big) \quad (29)$$

Where, $Q_{\pi^*}(s(T), a(T))$ denotes the optimal state-action pair value. This inherently calls utility maximization problem (P1) at each MEC server.

C. *Proposed DECENT Algorithm Based On LSTM*

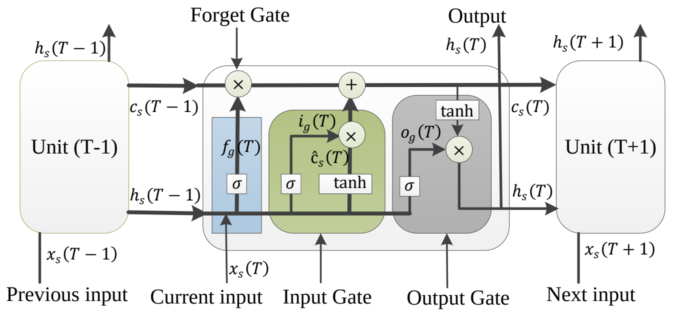

Fig.3: LSTM unit

As in the 6G network, there are a large number of states and actions; it is difficult to frequently store all Q-values in a table. With the introduction of DQL, the neural network absorbs the large number of states and actions of classical Q-learning and produces an approximate Q-value [32]. In the proposed scheme, LSTM network layer is used rather than traditional RNN to

1932-4537 (c) 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Nottingham Trent University. Downloaded on January 27,2022 at 11:57:16 UTC from IEEE Xplore. Restrictions apply.

build the DQL network [28], which is used to approximate action value pairs for all state-action pairs.

The LSTM network layer consists of more than one LSTM unit. This LSTM unit are able to process very long series of information using cell memory to memorize previous computation and improves convergence speed over traditional RNN. The operation of a single LSTM unit is shown in Fig. 3, where it passes a message from its predecessor unit to a successor unit as intermediate output over a large number of correlated input-output pairs. The key feature of the LSTM unit is the cell state $c_s(T-1)$ memory vector, which flows through all units. Each LSTM has three gates: forget gate, input gate, and output gate. The forget gate is to determine the degree information needed to be forgotten of the previous cell state. The output of forget gate $(f_g(T))$ in time slot T depends upon the previous time slot hidden state $h_s(T-1)$ and current input $x_s(T)$. Forget gate outputs a number between 0 (forget all previous information) and 1(keep all the previous state information) by sigmoid (σ) neural net layer operation with weight factor w_{f_g} and balance factor b_{f_g}, given as

$$f_g(T) = \sigma(w_{f_g}[h_s(T-1) * x_s(T)] + b_{f_g}) \tag{30}$$

The next step is divided into two operations: their primary goal is which kind of new information is added to the cell state. In the first operation, the output of input gate $i_g(T) \in \{0,1\})$ with weight factor w_{i_g} and balance factor b_{i_g} generates

$$i_g(T) = \sigma(w_{i_g}[h_s(T-1) * x_s(T)] + b_{i_g}) \tag{31}$$

Whereas the second operation resembles to update the cell state memory vector known as cell gate output alias candidate values $\hat{c}_s(t)$, given as

$$\hat{c}_s(T) = \tanh(w_{c_{th}}[h_s(T-1) * x_s(T) + b_{c_{th}}]) \tag{32}$$

After this, the old cell state $(c_s(T-1))$ value is updated with the new state $c_s(T)$ by summation of two terms (1) product of forget gate output with previous cell state and (2) product of input gate output to new candidate values, given as

$$c_s(T) = f_g(T) * c_s(T-1) + i_g(T) * \hat{c}_s(T) \tag{33}$$

The final step decides what information of the new state is shown as output $o_g(T)$ for the next cell state as hidden state output $h_s(T)$. Output gate also works in two steps: first output is generated on a sigmoid neural network, then $tanh$ layer is used to push the value between -1 to 1 of the new state. The output of the hidden cell state is given as

$$o_g(T) = \sigma(w_{o_g}[h_s(T-1) * x_s(T)] + b_{o_g}) \tag{34}$$

$$h_s(T) = o_g(T) * \tanh(c_s(T)) \tag{35}$$

The complete workflow and algorithm of the proposed DECENT framework shown in the fig.4 and algorithm 1 respectively with 'n' LSTM units, where output of the LSTM layer is fed to a fully connected network layer with weight factor $w_{m,fc}$ and balance factor $b_{m,fc}$ to get the final output.

Initially, the RL agent at m^{th}MEC server receives the information of current state $s_m(T) = (\gamma_m(T), b_m(T), D_m^l(T(t_2)), D_m^o(T(t_2)), e_m(T))$, then the LSTM network layer produced the approximated Q-value as mini-batch $Q_{minibatch}(s_m(T), a_m(T)) \in \mathbb{R}^{batchsize \times Z_m}$, where Z_m denotes the size of action space $a_m(T) = (D_m^l(T(t_2)), D_m^o(T(t_2))) \in A(T)$. To fit in the action space of Q-value, a fully connected network layer uses its filter to adjust the space for Q-value. The action generator $\phi_{m,A}$ for the LSTM network takes the input state $s_m(T)$ and produces output as $(s_m(T), a_m(T); \theta_{m,a}(T))$. The action generator $\phi_{m,A}(T)$ is composed of weights of both LSTM units (n) and fully connected network layer, as follows:

$$\phi_{m,A}\left(\theta_{m,a}(T)\right) = \phi_{m,A}\{w_{m,l1}, w_{m,l2}, w_{m,lt} \ldots . w_{m,ln}, w_{m,fc}\} \tag{36}$$

For updating the $Q(s_m(T), a_m(T))$ value, an action generator is used to evaluate the approximate value. If the approximated value is accurate, then the policy is greedy. The accurate estimation of Q-value for the given current state $s_m(T)$ by taking an action $a_m(T) \in A(T)$ is not always found. Thus, the RL agent uses the greedy policy with probability $\epsilon\ (0 < \epsilon < 1)$ to select an action that provides maximum reward (exploitation of knowledge); otherwise it selects any random action from the action space with probability $1-\epsilon$ (as exploration of action space). After taking an action $a_m(T)$, the RL agent receives the reward $r_m(T)$ and the network switches to new state $s_m(T+1)$. Experience replay $e_{m,r}(T)$ memory is used to store the received reward and new state in each time slot that consists of four tuple: $e_{m,r,T}(T) = (s_m(T), a_m(T), r_m(T), s_m(T+1))$ in memory buffer data set $\mathfrak{D}_m = \{e_{m,r,1}(T), e_{m,r,2}(T), \ldots e_{m,r,T}(T)\}$.

The size of experience replay memory is limited to the size of action space Z, which stores up to maximum Z_m experiences. Furthermore, mini-batch is formed by choosing some random experiences from $\mathfrak{D}_m$. Thereafter, we randomly select any tuple $(\tilde{e}_{m,r}(T) = (\tilde{s}_m(T), \tilde{a}_m(T), \tilde{r}_m(T), \hat{s}_m(T+1))$ from the mini-batch and an approximated Q-value is estimated. In DECENT, the parameterized Q-value is used for updating the network weight $\theta_{m,a}(T)$, denoted as $Q(s_m(T), a_m(T); \theta_{m,a}(T))$. In addition, update only that network parameter that minimizes the loss function $L_m(s_m(T), a_m(T); \theta_{m,a}(T))$. The Q-value updated in this way avoids the issue of correlation between large input-output data during the transition in the same episode [38]. Where, $y_m(T)$ represents the state update target Q-value with previous state weight parameter $\theta_{m,a}(T)$, given as follows

$$y_m(T) = r_m(T) + \gamma \max_{a_m(t) \in A(t)} Q\left(s_m(T+1), a_m(T); \theta_{m,a}(T)\right) \tag{37}$$

The loss function of the network is evaluated as follows

$$\begin{aligned} L\left(\theta_{m,a}(T)\right) = \mathbb{E}(\tilde{s}_m(T), \tilde{a}_m(T), \tilde{r}_m(T), \hat{s}_m(T+1) \\ \sim\mathfrak{D}_m\left[\left(y_m(T) - Q\left(s_m(T+1), a_m(T); \theta_{m,a}(T)\right)\right)^2\right] \\ = \left[\left(y_m(T) - Q(\tilde{s}_m(T), \tilde{a}_m(T); \theta_{m,a}(T))\right)^2\right] \end{aligned} \tag{38}$$

The gradient vector ∇_θ of the loss function is obtained by differentiating Eq. (39) with respect to weight, given as

$$\nabla \mathrm{m},_{\theta} L\left(\theta_{m,a}(T)\right) = (\tilde{r}_m(T) + \gamma \max_{a(T) \epsilon A(T)} Q\left(s_m(T+1), a_m(T); \theta_{m,a}(T-1) - Q\left(s_m(T), a_m(T); \theta_{m,a}(T)\right)\right) \nabla_{\mathrm{m},\theta} Q(\tilde{s}_m(T), \tilde{a}_m(T); \theta_{m,a}(T)) \tag{39}$$

The network weight $\theta_{m,a+1}(T)$ is updated according to a stochastic Gradient descent method with learning rate $\alpha \epsilon \{0,1\}$, as follows:

1932-4537 (c) 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Nottingham Trent University. Downloaded on January 27,2022 at 11:57:16 UTC from IEEE Xplore. Restrictions apply.

$$\theta_{m,a+1}(T) = \theta_{m,a}(T) + \alpha\nabla_{m,\theta}L(\theta_{m,a}) \quad (40)$$

Algorithm 1 *DRN- LSTM* based offloading rate control Algorithm

```
1.  Initialize $\mathfrak{D}_m$ and $\phi_{m,A}$ with random weights $\theta_{m,a}$.
2.  Assign maximum number of training episode $E_{ps} = E_{ps}^{max}$ .
3.  initialize and observe the environment to get initial state $S_m(T)$= $(\gamma_m(T), b_m(T), c_m^g(T), c_m^b(T), e_m(T))$
4.  For $T = 1,2 \ldots\ldots$ do
5.      If $random\ () \le \epsilon$
            Randomly select an action
            $a_m(T) = \left(D_m^l(T), D_m^o(T)\right) \in A(T)$.
6.      Else
        a.  Form the experience $e_{m,r,T}(T) = (s_m(T-1) \ldots s_m(T))$
        b.  Set $S_m(T)$ as input to LSTM network
        c.  For each action $a_m(T) \in A(T)$ evaluate the output of LSTM to obtain $Q(s_m(T), a_m(T); \theta_{m,a}(T))$ using $\phi_{m,A}$ with random weights $\theta_{m,a}$.
        d.  Select an action such that $a_{max}(T) = \arg \max_{a_m(T)\in A(T)} (Q(s_m(T), a_m(T))$
7.      End if
8.  Take an action $a_{max}(T)$ to switch new state $(s_m(T+1))$
9.  Evaluate $e_m(T), L_m(T)$ and task drop loss
10. calculate $U_m(T)$ using Eq.(12)
11. Evaluate $r_m(T)$ using Eq.(24)
12. Formulate the experience tuple from transition $(s_m(T), a_m(T), r_m(T), s_m(T+1))$ and store into memory data set $\mathfrak{D}_m$,
13.     For $E_{ps} = 1,2,3 \ldots\ldots E_{ps}^{max}$ do
        a.  Select randomly from mini-batch $\tilde{e}_{m,r}(T) = (\tilde{s}_m(T), \tilde{a}_m(T), \tilde{r}_m(T), \hat{s}_m(T+1))$
        b.  Calculate target Q-value $y_m(T)$ using Eq. (37)
14.     End For
15. Evaluate Loss function by using Eq.(38)
16. Update the network parameter $\theta_{m,a}(T)$ by performing stochastic Gradient descent optimization using Eq. (39) & Eq. (40).
17. End For
```

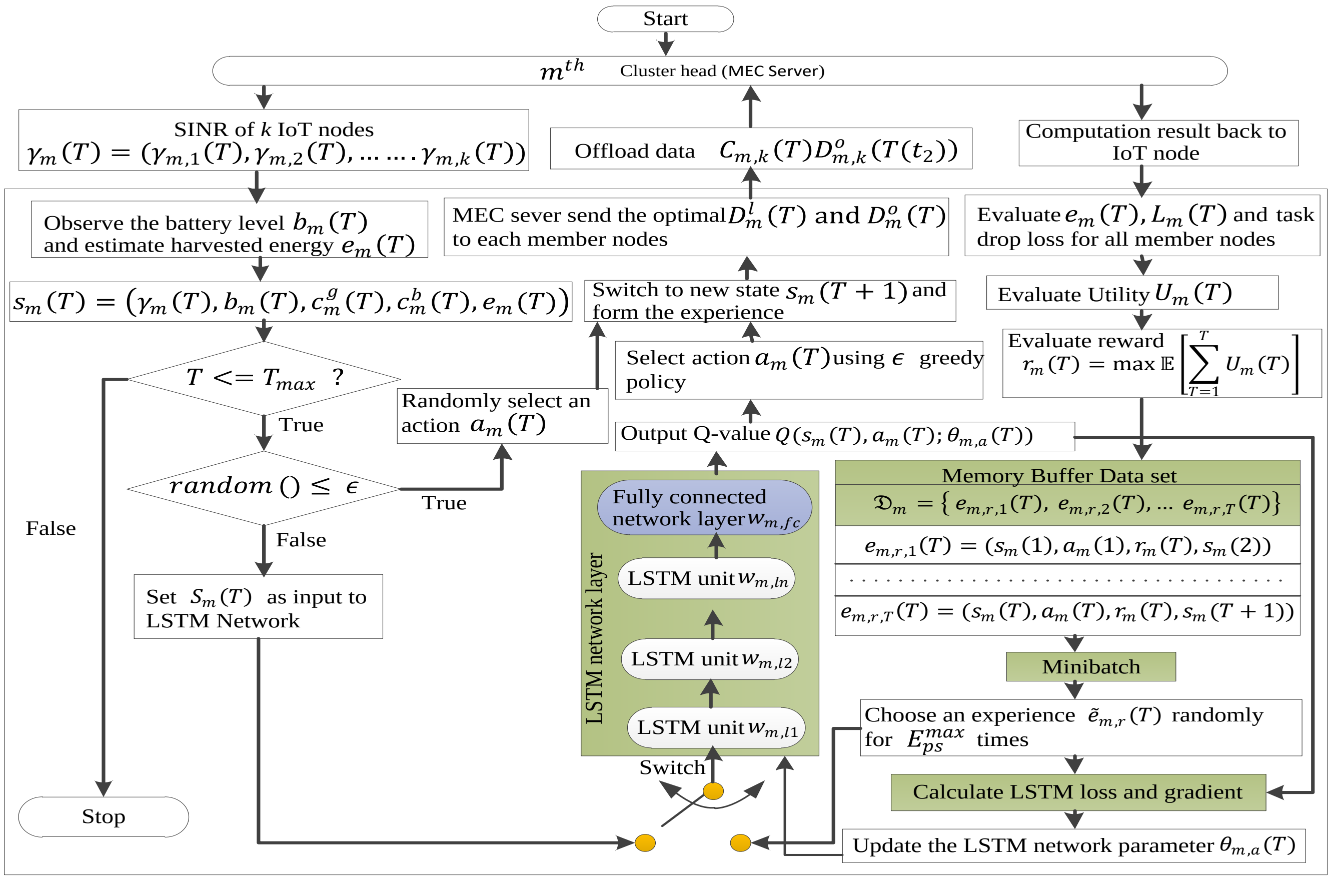
Start
m^{th} Cluster head (MEC Server)
SINR of k IoT nodes $\gamma_m(T) = (\gamma_{m,1}(T), \gamma_{m,2}(T), \ldots\ldots \gamma_{m,k}(T))$
Offload data $C_{m,k}(T)D_{m,k}^o(T(t_2))$
Computation result back to IoT node
Observe the battery level $b_m(T)$ and estimate harvested energy $e_m(T)$
MEC sever send the optimal $D_m^l(T)$ and $D_m^o(T)$ to each member nodes
Evaluate $e_m(T), L_m(T)$ and task drop loss for all member nodes
$s_m(T) = (\gamma_m(T), b_m(T), c_m^g(T), c_m^b(T), e_m(T))$
Switch to new state $s_m(T+1)$ and form the experience
Evaluate Utility $U_m(T)$
$T <= T_{max}$?
Select action $a_m(T)$ using ϵ greedy policy
Evaluate reward $r_m(T) = \max \mathbb{E}\left[\sum_{T=1}^{T} U_m(T)\right]$
True
Randomly select an action $a_m(T)$
Output Q-value $Q(s_m(T), a_m(T); \theta_{m,a}(T))$
$random\ () \le \epsilon$
True
False
False
Memory Buffer Data set
$\mathfrak{D}_m = \{e_{m,r,1}(T), e_{m,r,2}(T), \ldots e_{m,r,T}(T)\}$
$e_{m,r,1}(T) = (s_m(1), a_m(1), r_m(T), s_m(2))$
$e_{m,r,T}(T) = (s_m(T), a_m(T), r_m(T), s_m(T+1))$
Fully connected network layer $w_{m,fc}$
LSTM network layer
LSTM unit $w_{m,ln}$
LSTM unit $w_{m,l2}$
LSTM unit $w_{m,l1}$
Set $S_m(T)$ as input to LSTM Network
Minibatch
Choose an experience $\tilde{e}_{m,r}(T)$ randomly for E_{ps}^{max} times
Switch
Calculate LSTM loss and gradient
Stop
Update the LSTM network parameter $\theta_{m,a}(T)$

Fig.4: Scientific workflow of the DECENT algorithm

V. Performance Evaluation and Result Analysis

A. *Analysis of DECENT*

In this section, two major analyses are used to validate the effectiveness of the proposed algorithm: (1) convergence property, and (2) computational complexity.

1. *Convergence Property*

The convergence of the DECENT algorithm depends upon two conditions: (1) the DECENT network overcomes the vanishing or exploding gradient; i.e., it controls the variation of gradient values too large or small exponentially in each time slot; (2) it stabilizes the training process with close to zero training error using (mini-batch) stochastic gradient descent. Both conditions hold in polynomial time under mild assumption and the convergence holds the condition as follows

$$f(\theta(t)) \le \varepsilon \text{ for all } t \in [1,2,3 \ldots\ldots T] \quad (41)$$

1932-4537 (c) 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Nottingham Trent University. Downloaded on January 27,2022 at 11:57:16 UTC from IEEE Xplore. Restrictions apply.

Above the inequality state that, when training, loss of the DECENT drops to ε during time instant t with linear convergence speed, the algorithm converges to optimal policy. The proof of the convergence property for the proposed algorithm is similar to the convergence of training RNN [33].

2. *Computational Complexity*

Time complexity of the DECENT mainly depends upon the weight ψ of each LSTM unit cell state. Each LSTM unit (cell) has four connections used to evaluate output: forget gate, input gate, cell gate and output gate that control the flow of information shown in fig 3. Each component is associated with the previous cell state input plus the current time slot input. Therefore, if there are N_T number of units in the LSTM layer in the current time slot and N_{T-1}number of units in the previous layer, then the total number of inputs to each component is $N_T + N_{T-1}$. As, there are four connections in each unit, then there is total $4(N_T + N_{T-1})$ weights associated with each unit. Since there are N_T units in one LSTM layer, so there is, $4N_T(N_T + N_{T-1})$ weight associated with LSTM layer. The weights associated with output component is $N_T * N_o$, where N_o is the total number of output. The weights associated with forget gate, input gate and cell weight components of each layer is $N_T * 3$. If there are H numbers of hidden layers, so total weights are given as

$$\psi = H * (4N_T(N_T + N_{T-1}) + N_T * 3) + N_T * N_o \quad (42)$$

The computational complexity of DECENT learning model per weight and time slot using stochastic gradient optimization technique is $O(1)$. Thus, the overall complexity of the proposed DECENT algorithm is $O(\psi)$.

B. *Simulation And Discussion*

In this section, we evaluate the performance of the proposed LSTM based DECENT algorithm for the optimal selection of a data offloading rate in EH MEC-based IoT network with respect to different LSTM parameters and state-of-art algorithms.

1. *Simulation Environment*

The simulation of proposed DECENT algorithm to optimize the policy for data offloading is implemented using Python 3.7 and Tensorflow 1.2.1 framework. For data preprocessing and management Numpy, pandas and scikit-learn libraries of Python is used such as sklearn.preprocessing import LabelEncoder, sklearn.utils import shuffle and sklearn.model_selection import train_test_split (for training and testing of samples). Whereas tensforflow.contrib import RNN is being used for creating deep RNN LSTM network along with primary functions: 1) RNN (input, weights, biases) - responsible for creating and training of the LSTM network, 2) RNN.BasicLSTMCell(n_hidden)- for creating single layer LSTM with n_hidden cells 3) squaredelta()- to evaluate the loss 4) GradientDescentOptimizer(learning_rate)- to optimize and update the loss and network weight parameter respectively 5) tf.global_variables_initializer()- to initialize and process all the variables. We trained the proposed algorithm with size of experience replay memory buffer, training mini-batch 1024, 128 samples. And the training interval (number of iterations at which LSTM network is refreshed) be 10. The LSTM network consists of $n = 128$ hidden cells and $tanh$ activation function for the output.

Table 1. Simulation Parameters

Parameter	Value	Parameter	Value
$p_{m,k}(T)$	$(0.5, 15)\ dBm$	δ^2	$-30\ dBm$
$c^g_{m,k}(T)$	$150\ kb/s$	∂	0.3
T	$1000\ slot$	ν	$0.1\ sec/bit$
ϖ	0.75	ϑ	$0.2sec$
f^l	2 GHz/sec	φ	0.1
B	10 MHz	b_{max}	$8\ J$
$\hat{C}$	0.06	ζ	$1000cycle/bit$
γ	0.92	$E^{m,k}$	$(0.5, 6)J$
ϵ	0.25	η	0.4

We consider a network area of $100 \times 100\ m^2$, where 60 EH-IoT nodes are randomly deployed. For distribution of IoT nodes and creating the network MATLAB R2017a function NetArch(length, width, MEC-location, initial energy) and NodeArch(NetArch, number of nodes) are used respectively. For energy consumption of IoT nodes first order radio [2] is used with initial energy of $2\,J$ for all IoT nodes. The MEC server has computation capacity of 10GHz/sec. The transmission power $[p_{m,k}(T) \in (0.5, 15)\ dBm\ ]$of an IoT node crucial parameter and depends upon amount of data being offloaded to MEC server and quantity of energy harvested in the previous time slot. The bandwidth B of the channel is 10 MHZ for smooth transmission for data. The parameters related to CPU of an IoT node such as frequency $f^l = 2\ \text{GHz/sec}$, computation latency $\nu = 0.1\ sec/bit$ and number of cycle required to compute one bit $\zeta = 1000 cycle/bit$ is fixed. And finally the critical parameters of LSTM RNN network such as discount factor $\gamma = 0.92$ for getting higher long-term discounted reward corresponds to future value, learning parameter $\eta = 0.4$ to achieve optimal policy quickly. These simulation parameters along with others are listed in the Table 1. The time-varying channel gain $h_{m,k}(T)$of an IoT device follow the Rayleigh fading model, $h_{m,k}(T) = h_{m,k}\, \Gamma_{m,k}(t)$ where, $\Gamma_{m,k}$ denotes the independent random channel fading with unit mean [35]. The MATLAB Simulink Super-resolution Model is used for the implementation of EH technique [36] and for generating dynamic channel gain values while satisfying channel statistical properties, MATLAB function ANDIFFSR(s, delta_est, phi_est, factor) is used [37].

2. *Result Analysis*

This section analyzes the convergence rate of DECENT algorithm in terms of utility over critical LSTM parameters. Further, performance test is performed between DECENT and over state-of-art algorithms for the comparative analysis of energy consumption, computation time, task drop rate and utility of EH MEC-based IoT network.

1) *Convergence Performance Over Different Parameter*

In the fig. 5(a-d), impact of different LSTM unit parameters for different values on the convergence of the proposed DECENT algorithm in terms of average utility are shown with LR (η)=0.4. Fig. 5(a) shows that small size of memory size-experience replay buffer (MS =128) causes larger variation in the convergence of average utility, while after increasing the MS to

1932-4537 (c) 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Nottingham Trent University. Downloaded on January 27,2022 at 11:57:16 UTC from IEEE Xplore. Restrictions apply.

2048, it requires both more training data and time to converge to an optimal value close to unity. Thus, for further simulation MS=1024 is selected that converges to an optimal value in less time. Furthermore, random data from the MS of 1024 is selected and formed mini-batch samples for the training procedure.

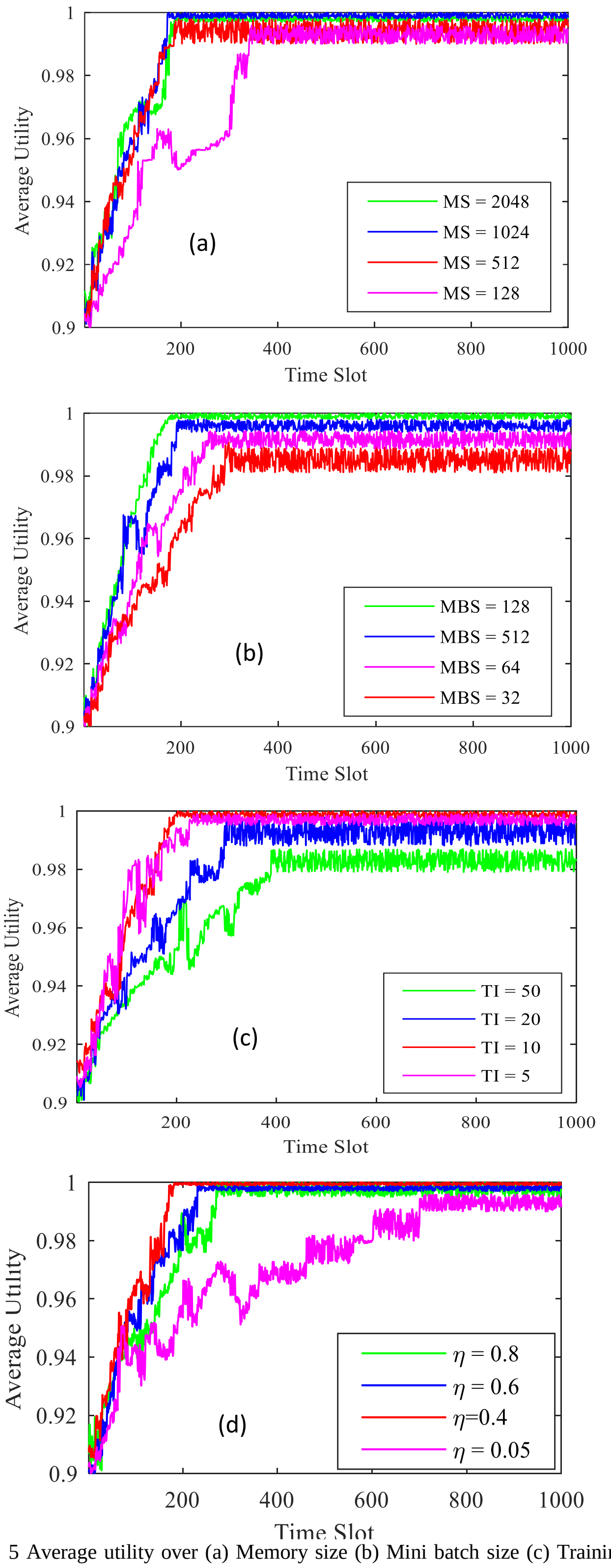

Fig. 5 Average utility over (a) Memory size (b) Mini batch size (c) Training interval (d) learning rate

Fig. 5(b) shows that a small mini-batch size (MBS=32) is not suitable for storing all the training data, so the convergence time of the proposed algorithm is fast but failed to reach the optimal value. When using a larger MBS (=512), frequently consider old data for the training procedure, which forces the algorithm to take a long time to learn the optimal policy for convergence. Thus, utility of the system degrades. Whereas on setting MBS=128, the utility of the system reaches optimal value quickly, because of DECENT consider proper mix of old and recent data for training procedure. We set MBS size to 128 for the further simulations.

Fig. 5(c) shows that for a small value of training interval (TI=5), the proposed algorithm converges very fast, which means it learns the optimal policy by updating frequently, while a larger TI (>25) showed poor performance regarding convergence property because the policy is not updated properly. Thus, the results indicate unnecessary updation of the policy for lower values of TI while higher values of TI update the policy rarely. Therefore, we set the TI =10 for the upcoming simulation results. In the Fig. 5(d), we investigated the impact of learning rate (LR) in the weight updation of Gradient descent optimizer using Eq. (40). The LR (η) is responsible for mapping inputs to outputs in the training dataset of the model or simply controls the rate of loss during updation of weight for each batch training sample. It is observed from the result that either too small of a value of $\eta \leq 0.05$ or a higher value $\eta \geq 0.8$ results in the proposed algorithm being stuck into local optimal (stuck with high training error) and never reaches the global optimal value; consequently, network performance degrades. While setting up $\eta = 0.4$, the proposed algorithm learns the optimal policy faster and converges in less time. This is because DECENT uses a memory cell to map the input to output values, and weights are updated using a stochastic gradient method, which avoids the numerical overflow (explode) of weight. Thus, for the next experiment set, learning rate $\eta = 0.4$ is set that helps in producing the set of global optimal weights.

2) Convergence Performance Over Number of Time Slots

It is observed from Fig. 6(a-c) that the proposed algorithm DECENT achieves the optimal offloading rate pertaining to maximize the average utility of the network after convergence with setting up parameters $\gamma = 0.92$ and $\eta = 0.4$ within 1000 time slots. Particularly, fig. 6(a) shows that energy consumption of DECENT decreases as the time slot increases till $T = 175$ and become stable with consumption of 3J energy that validates the convergence property. However, DQL and QL converge the energy consumption after time slot T= 420 and 495, respectively. In addition to these, Fig. 6(b-c) show that the computation latency and task drop rate for DECENT, DQL, and QL also decreases until T = 175, 420, 495, respectively, and converges after further increase in the time slot. The proposed algorithm uses the EH technique and computation task partition, which enables IoT nodes to use the energy in operation and the offloading task to MEC in an efficient manner.

The network utility of the EH MEC based IoT network depends upon energy consumption, computation latency, and task drop rate, as given in Eq. (11). Combining all the results obtained from fig. 6 (a-c), the average network utility is evaluated and is

1932-4537 (c) 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Nottingham Trent University. Downloaded on January 27,2022 at 11:57:16 UTC from IEEE Xplore. Restrictions apply.

shown in fig. 6(d) with respect to the time slot. As the time slot increases, the utility of the proposed algorithm increases rapidly and saturates within time slot 175. The utility of DQL and QL takes 245 and 320 more timeslots to converge as compared to DECENT. The proposed algorithm outperformed the DQL and QL offloading algorithms in terms of average utility after $T = 175$ by 19.25% and 32%, respectively.

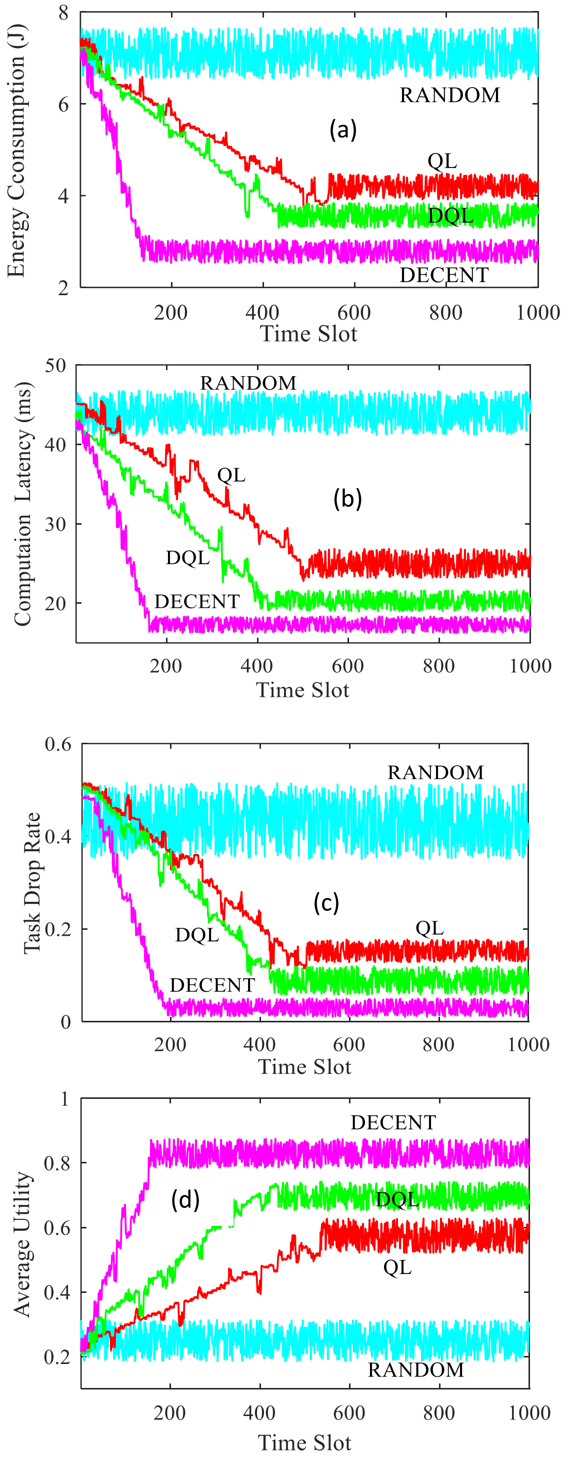

Fig. 6 Convergence performance (a) Energy consumption (a) Computation latency (c) Task drop rate (d) Average Utility

Fig. 6(a-c) reveals that the improved utility performance of DECENT is achieved by reducing the energy consumption by 22.34%, 36.23%, shortening the computation latency by 19.18%, 33.45%, and lowering the task drop rate by 52%, 72 % with respect to DQL and QL, respectively. The reason behind is twofold: (i) the DECENT uses LSTM network wherein each memory cell stores only relevant previous state information such as energy consumption, task drop rate, computation latency during and local processing to estimate the next state offloading rate in an efficient manner; (ii) LSTM unit compresses the large state space using fully connected layer and $tanh$ activation function improves the learning performance, as a result proposed algorithm convergences faster than state-of-art techniques. Whereas, DQL stores all the previous state information and takes much time to evaluate the next state output, as searching for all possible neural network combination is required. Although, QL based algorithm suffers from the curse of dimensionality between large input-output data during the transition of state. This can be attribute to slow learning rate which causes convergence of QL based algorithm took many time slots. In addition, fig. 5(a-c) also reveals that the random algorithm lags significantly behind DECENT by 68%, 76%, and 83% in energy consumption, computation latency, and task drop ratio, respectively. This is due to the fact that random algorithm randomly selects any offloading rate for data offloading.

3) *System Performance Over Size of Computation Task*

Fig. 7(a-c) shows that energy consumption, computation latency, and task drop for the state-of-the-art algorithms is a monotonically increasing function of computation task offloaded to the MEC server by EH-IoT nodes. This is because of more computation task offloaded on the MEC server consumes more energy and time. Higher amount of data requires more energy to transfer the bits of information to MEC server according to first order radio model energy consumption formula [2]. In addition, it also increases the task drop ratio. Fig. 7(a) shows that, as the computation task size varies from 80 kb to 130 kb, the energy consumption of the presented algorithm increased by 62 %. With further increases in the computation task, variation in energy consumption is not visible at all i.e., converged. However, in DQL and QL, energy consumption increases at a much higher rate by 71 % and 77%, respectively, up to 140 kb. This is because the presented algorithm selects an optimal offloading rate and harvested energy efficiently. Thus, the proposed algorithm outperformed the DQL and QL by reducing energy consumption by 13 % and 15 %, respectively.

In addition, Fig. 7(b) shows that computation latency of all algorithms increases rapidly; only DECENT and DQL managed to converge over 140 kb of computation size, whereas other algorithms failed to control the increasing rate of computation latency. The reason for this is that DECENT uses cell memory to learn optimal policy (offloading rate) from long-term dependencies provided by offloading experiences faster than other algorithms. Moreover, Fig. 7(c) shows that the task drop rate of the proposed algorithm also increases as the computation task increases up to 130 kb; furthermore, in the computation task (130-160kb), only 9 % of total computation task dropped with a constant rate. However, in the DQL task, the drop rate is

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2022.3145056, IEEE Transactions on Network and Service Management

higher than DECENT and it is about 13 % in the range of 130-160 kb size of the computation task.

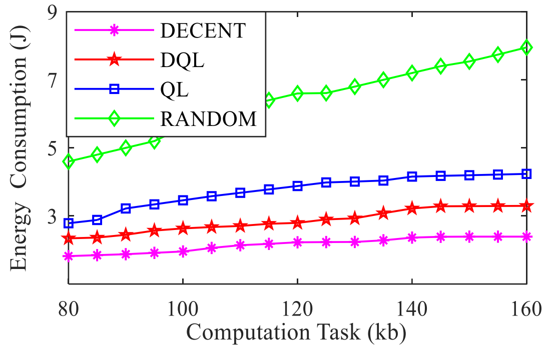

Fig. 7 (a). Energy consumption over computation task

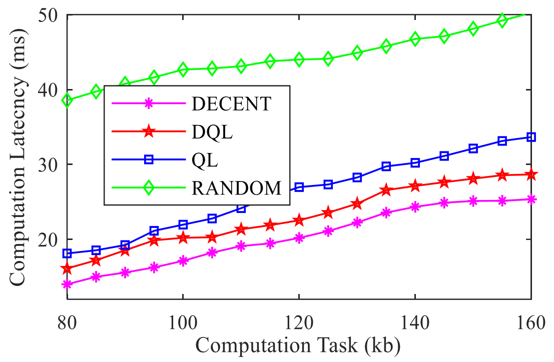

Fig. 7 (b). Computation latency over computation task

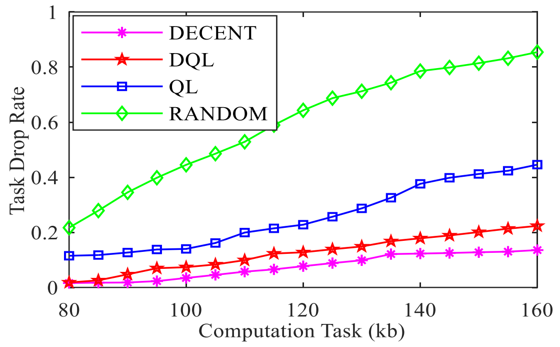

Fig. 7 (c). Task drop rate over computation task

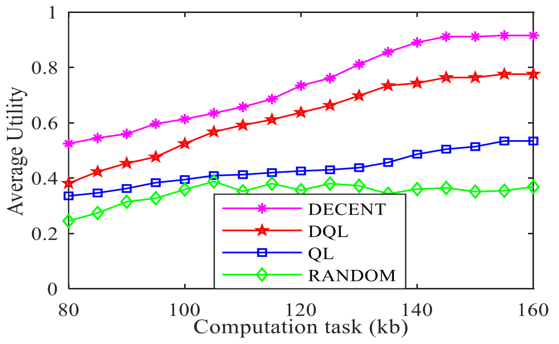

Fig. 7 (d). Average Utility over computation task

Overall, the proposed algorithm reduced the task drop rate as compared to DQL and QL by 37 % and 68%, respectively. The random algorithm showed the worst performance among others and it could not converge as the computation size increased. Overall, DECENT beats all the considered state-of-the-art algorithms, such as DQL and QL, by the lower energy consumption of 22.23%, 43.3%, shorter computation latency of 17.25%, 35.4%, and significantly lower task drop ratio of 87.2%, with a computation size of 130kb. The reason for this is that the proposed algorithm efficiently divides the computation task and learns the optimal policy faster to decide the data offloading rate to the MEC server and local computation rate for local execution. Also, DECENT does not account for offloading all the generated data instantly. It stores some of the generated data in a buffer and a fraction of data is offloaded to the MEC server; by doing so, latency and the task drop ratio decrease.

The results in the fig.7(d) shows that with the increase in computation task, average utility of the network of the proposed algorithm increases rapidly and stable at 140kb other than state-of-art-algorithms. This observation affirms that the energy consumption, computation latency and task drop rate of DECENT is less than other state-of-algorithms and validates the equation (11). It can be also observed that average utility of the proposed algorithm reached about 0.91. This is because the proposed algorithm uses $tanh$ activation function and experiences replay memory (reduces the state space) to improve the learning rate rather than $\epsilon-$greedy policy used in QL to select an action. The network utility of random selection schemes shows much lower increment in utility, as the computation task increases because of random selection policy; i.e., take any random offloading rate for data offloading, although it is not possible by the random algorithm to control the variation of the energy consumption, computation latency, and task drop rate.

4) *Normalized Computation Rate and Training Loss vs. Time Slots*

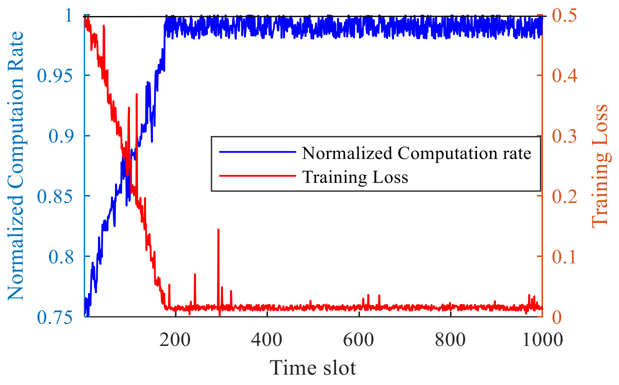

Fig.8. Normalized computation rate and Training loss over time slot

The MEC server forms a cluster with randomly selected EH-IoT node and itself as cluster head. Here, we define normalized (average) computation rate $\bar{Q}$ (state-action $\in [0,1]$) or utility for proposed DECENT algorithm as follows:

$$\bar{Q}\big(s_m(T), a_m(T)\big) = \frac{Q_{\pi^*}(s(T),a(T))}{\max_{a(T)\epsilon A(T)} Q_{\pi^*}(s(T),a(T))} \quad (41)$$

It is observed from the fig 8, for the first 175 time slots, the $\bar{Q}$ increases rapidly because of the LSTM network in the training phase shown in left side of vertical y-axis. After that, with an increase in times slot, the variation in the curve reduces and converges to 0.98 and the variance is close to zero. Whereas, right vertical y-axis shows that the training loss $L\left(\theta_{m,a}(T)\right)$ is much higher for the first time slots. Further,

1932-4537 (c) 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Nottingham Trent University. Downloaded on January 27,2022 at 11:57:16 UTC from IEEE Xplore. Restrictions apply.

with an increase in the time slot, LSTM is able to train its network (update the experience replay memory) and training loss $L\left(\theta_{m,a}(T)\right)$ that gradually decreases and stabilizes at approximately 0.02 in time slot 320. This shows that DECENT quickly minimizes its loss and converges to the optimal computation rate. This is because of mini-batch memory that consists of only relevant experiences from the input gate; irrelevant experiences are ignored by forget gate. Finally, the output gate only stores the experience based upon the $tanh$ function gate. Therefore, the size of mini-batch is also efficiently utilized and searching of previous computation rate is accelerated, which minimizes the training loss and helps to converge the algorithm faster.

VI. Conclusions and Future Scope

This paper presented deep learning LSTM-RNN based framework for EH IoT nodes to choose the optimal offloading rate and the local processing rate in MEC networks with only local causal knowledge of time-varying EH and channel states. The main objective of the work is to learn the optimal policy in order to maximize the long-term expected overall network utility by decreasing energy consumption, computation latency, and task drop rate in each time slot. An efficient algorithm, DECENT, is proposed that uses the past offloading experiences to improve the selection of the next action using a memory cell of LSTM units. The memory cells are able to compress the large state and action space in the learning process, which ultimately avoids the curse of dimensionality problem and makes the proposed algorithm to be computable in polynomial time. Furthermore, the proposed algorithm uses a stochastic gradient descent method as a parameterized policy to update the network parameter and generate optimal actions. It avoids the vanishing or exploding gradient value and minimizes the loss subject to the convergence of the presented algorithm. Simulation results shows that DECENT generates the optimal policy and improves the network utility by 23% and 43% as compared to benchmark DQL and QL algorithms, respectively. In the future, we will extend the DECENT offloading framework to MEC-cloud's server architecture. In addition, cooperation between the MEC server and a cloud server with transmission power selection to EH-IoT node will also be considered.

References

[1] S. Kumar, O. Kaiwartya, M. Rathee, N. Kumar and J. Lloret, "Toward Energy-Oriented Optimization for Green Communication in Sensor Enabled IoT Environments," in IEEE Systems Journal, doi: 10.1109/JSYST.2020.2975823, 2020

[2] Kashyap, P. K., Kumar, S., Dohare, U., Kumar, V., &Kharel, R.: Green Computing in Sensors-Enabled Internet of Things: Neuro Fuzzy Logic-Based Load Balancing. MDPI Electronics , 8(4), pp. 384-405, 2019.

[3] https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html.

[4] Z. Liao et al., "Distributed Probabilistic Offloading in Edge Computing for 6G-Enabled Massive Internet of Things," in IEEE Internet of Things Journal, vol. 8, no. 7, pp. 5298-5308, 1 April1, 2021.

[5] T. Koketsu Rodrigues, J. Liu and N. Kato, "Offloading Decision for Mobile Multi-Access Edge Computing in a Multi-Tiered 6G Network," in IEEE Transactions on Emerging Topics in Computing, doi: 10.1109/TETC.2021.3090061.

[6] E. Fitzgerald, M. Pióro and A. Tomaszwski, "Energy-Optimal Data Aggregation and Dissemination for the Internet of Things," in IEEE Internet of Things Journal, vol. 5, no. 2, pp. 955-969, April 2018.

[7] P. K. Kashyap, S. Kumar, A. Jaiswal, M. Prasad and A. H. Gandomi, "Towards Precision Agriculture: IoT-Enabled Intelligent Irrigation Systems Using Deep Learning Neural Network," in IEEE Sensors Journal, vol. 21, no. 16, pp. 17479-17491, 15 Aug.15, 2021.

[8] P. K. Kashyap, S. Kumar and A. Jaiswal, "Deep Learning Based Offloading Scheme for IoT Networks Towards Green Computing," IEEE International Conference on Industrial Internet (ICII), Orlando, USA, 2019, pp. 22-27

[9] Dinh, T. Q., La, Q. D., Quek, T. Q. S., & Shin, H. Distributed Learning for Computation Offloading in Mobile Edge Computing. IEEE Transactions on Communications, pp. 1–1, 2018.

[10] N. Kiran, C. Pan, S. Wang and C. Yin, "Joint resource allocation and computation offloading in mobile edge computing for SDN based wireless networks," in Journal of Communications and Networks, vol. 22, no. 1, pp. 1-11, Feb. 2020.

[11] E. Meskar, T. D. Todd, D. Zhao, and G. Karakostas, "Energy aware offloading for competing users on a shared communication channel, "IEEE Trans. Mobile Comput., vol. 16, no. 1, pp. 87–96, Jan. 2017.

[12] X. Chen, "Decentralized computation offloading game for mobile cloud computing," IEEE Trans. Parallel Distrib. Syst., vol. 26, no. 4, pp. 974–983, Apr. 2015.

[13] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," IEEE Trans. Netw., vol. 24, no. 5, pp. 2795–2808, Oct. 2016.

[14] J. Li, H. Gao, T. Lv and Y. Lu, "Deep reinforcement learning based computation offloading and resource allocation for MEC," 2018 IEEE Wireless Communications and Networking Conference (WCNC), Barcelona, pp. 1-6, 2018,.

[15] L. Huang, S. Bi and Y. J. Zhang, "Deep Reinforcement Learning for Online Computation Offloading in Wireless Powered Mobile-Edge Computing Networks," in IEEE Trans. on Mobile Computing, 2019.

[16] X. Chen, J. Wu, Y. Cai, H. Zhang and T. Chen, "Energy-Efficiency Oriented Traffic Offloading in Wireless Networks: A Brief Survey and a Learning Approach for Heterogeneous Cellular Networks," in IEEE Journal on Selected Areas in Commu., vol. 33, no. 4, pp. 627-640, 2015.

[17] Y. Wei, F. R. Yu, M. Song and Z. Han, "User Scheduling and Resource Allocation in HetNets With Hybrid Energy Supply: An Actor-Critic Reinforcement Learning Approach," in IEEE Transactions on Wireless Communications, vol. 17, no. 1, pp. 680-692, Jan. 2018.

[18] Z. Wei, B. Zhao, J. Su and X. Lu, "Dynamic Edge Computation Offloading for Internet of Things With Energy Harvesting: A Learning Method," in IEEE IoT Journal, vol. 6, no. 3, pp. 4436-4447, June 2019.

[19] T. Q. Dinh, J. Tang, Q. D. La and T. Q. Quek, "Offloading in mobile edge computing: Task allocation and computational frequency scaling,"IEEE Trans. Commun., vol. 65, no. 8, pp. 3571–3584, 2017.

[20] J. Du, L. Zhao, J. Feng and X. Chu, "Computation offloading and resource allocation in mixed fog/cloud computing systems with minmax fairness guarantee," IEEE Trans. Commun., vol. 66, no. 4, pp. 1594–1608, Apr. 2018.

[21] C. Zhang, Z. Liu, B. Gu, K. Yamori, and Y. Tanaka, "A deep reinforcement learning based approach for cost-and energy-aware multi-flow mobile data offloading," IEICE Trans. Commun., vol. E101-B, no. 7, pp.2017–2025, 2018.

[22] L. Ji, G. Hui, L. Tiejun, and L. Yueming, "Deep reinforcement learning based computation offloading and resource allocation for mec," in proc. IEEE WCNC, pp. 1–5, 2018.

[23] Y. Xie, Z. Xu, Y. Zhong, J. Xu, S. Gong and Y. Wang, "Backscatter-Assisted Computation Offloading for Energy Harvesting IoT Devices via Policy-based Deep Reinforcement Learning," 2019 IEEE/CIC International Conference on Communications Workshops in China (ICCC Workshops), Changchun, China, 2019, pp. 65-70.

[24] M. Min et al., "Learning-Based Privacy-Aware Offloading for Healthcare IoT With Energy Harvesting," in IEEE Internet of Things Journal, vol. 6, no. 3, pp. 4307-4316, June 2019.

[25] Y. Liu, H. Yu, S. Xie and Y. Zhang, "Deep Reinforcement Learning for Offloading and Resource Allocation in Vehicle Edge Computing and Networks," in IEEE Trans. on Vehicular Technology, vol. 68, no. 11, pp. 11158-11168, Nov. 2019.

[26] L. Huang, X. Feng, A. Feng, Y. Huang, and P. Qian, "Distributed Deep Learning-based Offloading for Mobile Edge Computing Networks," Mobile Netw. Appl., 2018, doi: 10:1007/s11036-018-1177-x

[27] L. Huang, X. Feng, C. Zhang, L. Qian, Y. Wu, "Deep reinforcement learning-based joint task offloading and bandwidth allocation for multi-user edge computing", Digital Communication and Networks, vol. 5, no. 1, pp. 10-17, 2019.

1932-4537 (c) 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Nottingham Trent University. Downloaded on January 27,2022 at 11:57:16 UTC from IEEE Xplore. Restrictions apply.

[28] Wolf, Thomas; Debut, Lysandre; Sanh, Victor; Chaumond, et al., “Transformers: State-of-the-Art Natural Language processing”. Proceedings on the Conference on Empirical Methods in Natural Language Processing: System Demonstrations. Pp. 38-45, 2020.
[29] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, “Performance optimization in mobile-edge computing via deep reinforcement learning,” IEEE Internet of Things Journal, Oct. 2018.
[30] P. K. Kashyap, S. Kumar, A. Jaiswal, M. Prasad and A. H. Gandomi, "Towards Precision Agriculture: IoT-enabled Intelligent Irrigation Systems Using Deep Learning Neural Network," in IEEE Sensors Journal, doi: 10.1109/JSEN.2021.3069266.
[31] W. Liu, J. Cao, L. Yang, et al., “Appbooster: Boosting the performance of interactive mobile applications with computation offloading and parameter tuning,” IEEE Trans. Parallel. Distrib. Syst., vol. 28, no. 6, pp. 1593–1606, 2017.
[32] A. Cammarano, C. Petrioli and D. Spenza, "Online Energy Harvesting Prediction in Environmentally Powered Wireless Sensor Networks," in IEEE Sensors Journal, vol. 16, no. 17, pp. 6793-6804, Sept.1, 2016.
[33] Y. Wang, M. Sheng, X. Wang, et al., “Mobile-edge computing: Partial computation offloading using dynamic voltage scaling,” IEEE Trans.Commun., vol. 64, no. 10, pp. 4268–4282, Aug. 2016.
[34] A. A. Nasir, X. Zhou, S. Durrani, and R. A. Kennedy, “Relaying protocols for wireless energy harvesting and information processing,” IEEE Trans. Wireless Commun., vol. 12, no. 7, pp. 3622–3636, 2013.
[35] Allen-Zhu, Zeyuan & Li, Yuanzhi & Song, Zhao On the Convergence Rate of Training Recurrent Neural Networks” NeurIPS (2019).
[36] K. E. Baddour and N. C. Beaulieu, "Autoregressive modeling for fading channel simulation," in IEEE Transactions on Wireless Communications, vol. 4, no. 4, pp. 1650-1662, July 2005.
[37] http://venividiwiki.ee.virginia.edu/mediawiki/images/4/4f/EHarv_Documentation_v1.1.pdf.
[38] https://venividiwiki.ee.virginia.edu/mediawiki/index.php/SUPR_Models

PANKAJ KUMAR KASHYAP is working as District Informatics Officer in National Informatics Centre, Jammu & Kashmir (Union Territory) under Ministry of Electronics and Information Technology, Government of India, New Delhi. He received his Ph.D. degree and M.Tech degree in Computer Science and technology from Jawaharlal Nehru University, New Delhi in 2020 and 2014 respectively. His research area interest is load balancing and energy optimization in wireless sensor networks, Internet of Things or vehicles using machine learning approaches. Dr. Kashyap has published more than 5 papers in international journals and over 10 papers in international conferences. Dr. Kashyap is currently working on quantum learning, deep learning based energy optimization, energy trading, security analysis through blockchain and load balancing in IoT networks.

SUSHIL KUMAR ((M’11 SM’18) is currently working as Assistant Professor at the School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi, India. Prior to that, he spent three years as Lecturer in Jamia Millia Islamia (Central University), Delhi, India and as Lecturer (Computer Science) in Shri Lal Bahadur Shastri R. S. Vidyapeeth (Deemed University), New Delhi. He received his Ph. D. degree in Computer Science from the School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi, India. His research interest includes the area of vehicular cyber-physical systems, Internet of things, Internet of unmanned aerial vehicles, cybersecurity and wireless sensor networks. He has supervised seventeen doctoral theses and currently supervising eight doctoral theses in vehicular communication, the energy efficiency of terrestrial sensor networks, blockchain, quantum computing, machine learning/deep learning and green and secure computing in next generation Internet of Things. Dr. Kumar has authored and coauthored over 100 technical papers in international journals and conferences. Dr. Kumar served as session chair in many international conferences and workshops. He is a reviewer in many IEEE/IET and other international journals. He is a reviewer for projects for various research funding organizations. He is served as guest editor in the Journal of Computer Networks and Communications, and Transaction on emerging technologies.

ANKITA JAISWAL is currently a Ph.D. senior research scholar at School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi, India. She received her M. Tech. degree in Computer Science and Technology from School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi, India in 2016 and B. Tech. degree in Computer Science and Engineering from Uttar Pradesh Technical University, India in 2013. Her research interests include 5G centric Wireless Sensor Networks, Internet of Things, Next Generation Wireless Systems, Machine Learning, and Quantum Computing. She has published three papers in international journals and over three papers in international conferences. She is currently working on techniques such as reinforcement learning, quantum learning, deep learning, blockchain, energy trading and trust in next generation IoT networks.

OMPRAKASH KAIWARTYA (Senior Member, IEEE) received the Ph.D. degree in computer science from Jawaharlal Nehru University, New Delhi, India, in 2015. He is currently a Senior Lecturer with the Department of Computer Science, Nottingham Trent University, Nottingham, U.K. He was previously a Research Associate with Northumbria University, Newcastle upon Tyne, U.K., in 2017 and a Postdoctoral Research Fellow with the University of Technology Malaysia, Johor Bahru, Malaysia, in 2016. His research interests include drone-enabled networking, E-mobility-centric electric vehicles, Internet-of-Things-enabled smart services, connected vehicles, and next-generation wireless systems. He is a Fellow of Higher Education Academy, U.K. He is also a Professional Member of British Computer Society, U.K. He is an Associate Editor and/or a Guest Editor of the IEEE Internet of Things Journal, IEEE Access, IET Intelligent Transport Systems, EURASIP Journal on Wireless Communication and Networking, Sensors (MDPI), and Electronics.

Manoj Kumar has received his M.tech degree from Jawaharlal Nehru University, New Delhi, India. He is currently pursuing Ph.D. from Jawaharlal Nehru University, New Delhi. His research area of interest is energy optimization, routing, localization in Internet of Things. He has published three papers in international conferences and journals.

UPASANA DOHARE is currently working as Assistant Professor at the Department of Computer Science & Engineering, IIMT College of Engineering, Greater Noida, India. She received her Ph.D. at School of Computer and Systems Sciences, Jawaharlal Nehru University, India in 2018. She received her M.Tech. degree in Computer Science & Technology from School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi, India in 2011. Her research interest includes Green Computing in Wireless Sensor Networks, Game Theoretic modeling of Ad Hoc and Internet of Things based Networks.

AMIR H. GANDOMI is a Professor of Data Science and an ARC DECRA Fellow at the Faculty of Engineering & Information Technology, University of Technology Sydney. Prior to joining UTS, Prof. Gandomi was an Assistant Professor at Stevens Institute of Technology, USA and a distinguished research fellow in BEACON center, Michigan State University, USA. Prof. Gandomi has published over two hundred journal papers and seven books which collectively have been cited 20k+ times (H-index = 66). He has been named as one of the most influential scientific mind and Highly Cited Researcher (top 1% publications and 0.1% researchers) for four consecutive years, 2017 to 2020. He also ranked 18th in GP bibliography among more than 12,000 researchers. He has served as associate editor, editor and guest editor in several prestigious journals such as AE of IEEE TBD and IEEE IoTJ. Prof Gandomi is active in delivering keynotes and invited talks. His research interests are global optimisation and (big) data analytics using machine learning and evolutionary computations in particular.

1932-4537 (c) 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: Nottingham Trent University. Downloaded on January 27,2022 at 11:57:16 UTC from IEEE Xplore. Restrictions apply.