Irshad, A, Abbas, ZH, Ali, Z, Abbas, G, Baker, T and Al-Jumeily, D

Wireless Powered Mobile Edge Computing Systems: Simultaneous Time Allocation and Offloading Policies

electronics

Article

Wireless Powered Mobile Edge Computing Systems: Simultaneous Time Allocation and Offloading Policies

Amna Irshad [1], Ziaul Haq Abbas [2], Zaiwar Ali [2], Ghulam Abbas [3], Thar Baker [4,*] and Dhiya Al-Jumeily [5]

1 Telecommunications and Networking Research Center, GIK Institute of Engineering Sciences and Technology, Topi 23640, Pakistan; gee1949@giki.edu.pk
2 Faculty of Electrical Engineering, GIK Institute of Engineering Sciences and Technology, Topi 23640, Pakistan; ziaul.h.abbas@giki.edu.pk (Z.H.A.); zaiwar.ali@giki.edu.pk (Z.A.)
3 Faculty of Computer Science and Engineering, GIK Institute of Engineering Sciences and Technology, Topi 23640, Pakistan; abbasg@giki.edu.pk
4 Department of Computer Science, University of Sharjah, Sharjah 27272, United Arab Emirates
5 School of Computer Science and Mathematics, Liverpool John Moores University, Liverpool L3 3AF, UK; d.aljumeily@ljmu.ac.uk
* Correspondence: tshamsa@sharjah.ac.ae

Citation: Irshad, A.; Abbas, Z.H.; Ali, Z.; Abbas, G.; Baker, T.; Al-Jumeily, D. Wireless Powered Mobile Edge Computing Systems: Simultaneous Time Allocation and Offloading Policies. *Electronics* **2021**, *10*, 965. https://doi.org/10.3390/electronics10080965

Academic Editors: Pietro Manzoni, George Angelos Papadopoulos and Juan M. Corchado

Received: 11 February 2021
Accepted: 15 April 2021
Published: 18 April 2021

Abstract: To improve the computational power and limited battery capacity of mobile devices (MDs), wireless powered mobile edge computing (MEC) systems are gaining much importance. In this paper, we consider a wireless powered MEC system composed of one MD and a hybrid access point (HAP) attached to MEC. Our objective is to achieve a joint time allocation and offloading policy simultaneously. We propose a cost function that considers both the energy consumption and the time delay of an MD. The proposed algorithm, joint time allocation and offload policy (JTAOP), is used to train a neural network for reducing the complexity of our algorithm that depends on the resolution of time and the number of components in a task. The numerical results are compared with three benchmark schemes, namely, total local computation, total offloading and partial offloading. Simulations show that the proposed algorithm performs better in producing the minimum cost and energy consumption as compared to the considered benchmark schemes.

Keywords: mobile edge computing; wireless energy transfer; wireless powered mobile edge computing

1. Introduction

The continuous advancement in mobile technologies has encouraged studies to improve features such as latency, data processing and energy consumption [1]. The response's latency of mobile devices (MDs) and data processing capabilities can be enhanced with the help of cloud computing. The on-demand computing services, namely, storage and processing power, are offered by cloud computing. Among the numerous benefits of this technology, increased productivity, speed and efficiency are the most notable. Since the computation by cloud occurs on remote servers and the distance between the MD and the server is large, there is a high response latency [2]. The response's latency problem is resolved by bringing execution resources at the edge of cellular networks, termed as mobile edge computing (MEC), which results in low network congestion and improved application performance [3]. However, the improved performance of MEC is restricted by the limited battery capacity of MDs. To overcome this hindrance, studies are conducted in a new domain that incorporates the data processing capacity of MEC with energy harvesting ability through wireless energy transfer (WET) [4]. With the incorporation of a hybrid access point (HAP), consisting of an energy node and an information access node, wireless powered-mobile edge computing (WP-MEC) transfers wireless energy to MDs by radio frequency signals. This amalgamation enhances both the battery performance and computing capabilities of cellular devices [5].

The time allocation policy, along with the offloading policy, requires numerous computations because the computational overhead increases exponentially. Therefore, an MD can offload its information to a mobile edge server (MES) as well as harvest energy in the same time slot. Now, the important question is, how much time should be used for harvesting and how much time should be used for offloading the tasks? If we increase the time for harvesting, the time delay for execution of tasks increases and if we increase the time for offloading then energy consumption increases because of low harvesting. In this paper, we propose a deep learning approach to find how much portion (in percent) of a single time slot should be used for harvesting and how much portion for transmitting data to HAP along with partial offloading policy. In partial offloading, we divide a single task into different components and then some of the components are offloaded to MES. The proposed technique simultaneously determines the components to be offloaded to MES as well as the portion of a single time slot required by the MD to harvest energy.

The traditional optimization techniques have high computational overhead and high algorithm complexity. We propose an algorithm which generates a training dataset to train a deep neural network (DNN). The algorithm complexity for generating a training dataset is also high; however, it is a one-time computation during the training phase only. After the training phase, the trained DNN has constant complexity $O(1)$ [6]. In our dataset we consider all the possible offloading policies with all possible time allocation policies simultaneously. For example, if we consider three components of a task then there will be 2^3 possible partial offloading policies as shown in Table 1.

Table 1. All offloading policies for three components.

Offloading Policy	**1st Component**	**2nd Component**	**3rd Component**
Policy-1	0	0	0
Policy-2	0	0	1
Policy-3	0	1	0
Policy-4	0	1	1
Policy-5	1	0	0
Policy-6	1	0	1
Policy-7	1	1	0
Policy-8	1	1	1

Now, one of these policies will have a minimum cost. Similarly, the number of possible time allocation policies depends on time resolution, r_s. For example, if we have $r_s = 0.1$ then there will be $\frac{1}{r_s} + 1$ possible time allocation policies, as shown in Table 2. Now, one of these policies will have a minimum cost. Actually, the proposed algorithm checks all the possible offloading policies with all possible time allocation policies and selects the policy with the minimum cost as the label for the training dataset. Similarly, the size of each component, the frequency of the user equipment for execution, the transmit power of MD, and the distance between MD and MES are stored as input data for this label. For different inputs we calculate the minimum costs in the same manner and get a training dataset. The training dataset has inputs and outputs (labels) with minimum costs in all possible options of offloading and time allocation policies.

Table 2. Possible time allocation policies, for resolution $r_s = 0.1$.

Time Allocation Policy	ρ_0 (Harvesting Time Percent)	ρ_1 (Offloading Time Percent)
1	0	1
2	0.1	0.9
3	0.2	0.8
4	0.3	0.7
5	0.4	0.6
6	0.5	0.5
7	0.6	0.4
8	0.7	0.3
9	0.8	0.2
10	0.9	0.1
11	1	0

The DNN is used as a tool to find the offloading policy and time allocation policy with minimum cost in terms of energy consumption and time delay.

Novelty and Contribution

The contributions of this paper are listed below:

- For the first time in MEC, we use a deep learning approach for both optimal offloading policy and optimal time fraction for harvesting in partial offloading schemes and propose a deep learning-based algorithm which gives minimum cost, in terms of delay and energy consumption, for computational offloading in MEC.
- Our approach is scalable in terms of time resolution, number of components per task, and distance between HAP and MD. The proposed algorithm generates training datasets for different desired values of time resolution, number of components per task, and distance between HAP and MD.
- The proposed algorithm, when compared with three benchmark cases—namely, total local computation, total offloading, and partial offloading—demonstrates better performance in terms of minimum cost, delay, and energy consumption.

The remainder of the paper is structured as follows. Section 2 describes the related work. Section 3 discusses the system model and problem formulation. Section 4 presents numerical results. Finally, Section 5 discusses conclusion.

2. Related Work

The maximization problem, considering computation rate, was studied in [7] to reduce the severe propagation loss by considering unmanned aerial vehicle (UAV)-enabled MEC wireless powered systems. The authors in [8] considered a multi-user wireless powered MEC system and jointly optimized the energy transmit beamforming at the access point (AP), the central processing unit (CPU) frequencies, the number of offloaded bits at the user, and the time allocation among the users. A binary computation offloading scheme for all energy harvesting MDs in the WP-MEC system was considered in [9]. The sum computation rate of MD was maximized by jointly considering the offloading policy and time allocation. An online service rate maximization algorithm (OSRM) was proposed in [10] to give the optimal time allocation policy considering a single MD in the WP-MEC system. The authors in [11] presented a user cooperation approach to improve the computation performance of active devices, where the surrounding devices act as helpers to use their harvested energy to help other active devices. In their work, the computation rate was maximized by considering the frequency division multiple access (FDMA) for computation offloading policy. The energy optimization problem, considering the transmit power at HAP was investigated in [12]. All the above-mentioned works focus on maximizing the computation rate and do not consider minimizing energy consumption and delay simultaneously.

The authors in [5] discussed the minimization of energy consumption and limited battery capacity of MD separately via optimization algorithms; however, the combined effect of energy consumption and time delay was not considered. The authors in [12] investigated the trade-off between energy efficiency and delay by transforming the offloading policy problem into a series of deterministic algorithms via Lyapunov optimization. However, since optimization schemes do not guarantee that the result will give global minima, a deep learning approach appears to be in dire need. In [13], the authors considered a deep learning-based online offloading framework that optimally decides task offloading decisions and gives a wireless offloading policy. WP-MEC with binary offloading policy (each task is either executed locally or fully offloaded to an MEC server) was considered. Their proposed scheme significantly improved the computational delay.

In most studies on WP-MEC systems, resource allocation and time allocation policies were investigated using numerical optimization methods, specifically, Lyapunov and Lagrange multipliers. However, the hard-combinatorial optimization problems of resource allocation in WP-MEC do not guarantee global minima/maxima, which directly affects the performance of these algorithms. An alternate approach is required which not only gives global minima/maxima but can also effectively reach results in a minimum amount of time. This leads us to deep learning, which reduces computational complexity by learning offloading decisions from experience. Little work is done using deep learning for making optimal allocation decisions in WP-MEC systems. In [13], the authors considered deep learning; however, they ignored the cost function. Their major focus was on the adaptive task offloading decisions of the online algorithm. In [3], the authors gave an optimal offloading policy for all components of a task by considering remaining energy of user equipment, energy consumption of application components, network conditions, and computational load, based on a deep learning approach. However, their work did not incorporate the energy harvesting capability of hybrid access point (HAP) attached with MEC.

By studying the related work on computational offloading in WP-MEC, we conclude that, if an MD executes all the components locally, its energy consumption will be low. However, it brings a huge delay due to limited local computational resources. Similarly, if an MD executes all the components on MES, it can reduce the time delay; however, the energy consumption is high due to offloading to MES. Therefore, partial offloading is a median technique, in which some of the components are executed locally and some of the components are offloaded to MES. In this paper, we propose an effective cost function by considering both energy consumption and time delay of an MD. The simulation results show that the proposed technique performs better than the considered benchmark techniques in terms of delay and energy consumption.

3. System Model and Problem Formulation

We consider a single MD in a WP-MEC system, in which the MEC server is connected with HAP with double antenna, one for energy harvesting and the other for data processing. HAP works on a time division duplex mode [14]. We assume that an HAP has the ability to transmit energy through RF signals to MDs and WET technology is fully incorporated in MEC as in [10]. MDs follow harvest-then-transmit protocol in which a device first harvests energy from HAP and then offloads components of the task onto HAP. In our paper, HAP is assumed to be equipped with adequate energy and processing power. In the case of partial offloading, $\rho \in \{\rho_0, \rho_1\}$ represents the time allocation policy. ρ_0 and ρ_1 are parts in percentage of one time slot. ρ_0 is the percent portion of one time slot in which an MD harvests energy; if ρ_0 is calculated as 0.58 then it means 58 percent of one time slot will be used for energy harvesting and in the rest of time slot (42% of time slot) MD will transmit the data to HAP. Therefore, ρ_o and ρ_1 are continuous variables taking values from $[0, 1]$. A variable x for the ith component of a task, denoted as x_i, is a binary variable, i.e., $x_i \in \{0, 1\}$, and is defined to determine the offloading policy. In the case of local execution, the value of x_i is 0 and in the case of remote execution, the value of x_i is 1. All the notations used in this paper are given in Table 3.

Table 3. List of notations used.

Notation	Meaning
B	Bandwidth
B_t	Battery of MD
c_i	*i*th component of a task in current time slot
c_{i+1}	*i*th component of a task in next time slot
d	Distance between MD and HAP
d_0	Reference distance
E_c	Total energy consumption of MD
E_{ci}	Energy consumption by MD for *i*th component
E_{hi}	Harvested energy by the MD for *i*th component
E_{max}	Maximum energy consumption by MD
E_{oi}	Offloading energy consumption in remote model
f_{di}	Frequency of MD for *i*th component
f_s	CPU frequency at MES
g_0	Path loss
H_i	Channel power gain of *i*th component
h_{di}	Downlink channel gain of *i*th component
L	Number of CPU cycles to process 1 bit of data
N_0	Noise spectral density
P_{di}	Transmit power of *i*th component of MD
P_{si}	Transmit power of MEC
P_t	Transmit power of HAP
R	Execution rate
r_i^{dl}	Maximum data rate on the down-link channel of *i*th Component
r_i^{ul}	Maximum data rate on the up-link channel of *i*th Component
T_c	Total time delay for the whole task
T_{ci}	Time delay of *i*th component in remote model
T_{di}	Down-link delay time
T_{ei}	Processing time at MEC
T_{li}	Time delay of *i*th component in local execution model
T_{max}	Maximum time delay of MD
T_{oi}	Total time delay of *i*th component
T_{ui}	Up-link delay time from MD to MEC
w_1, w_2	Weighing constants
x_i	Variable indicating the offloading policy for *i*th component
α, β	Scaling factors
γ	Cost of offloading policy
η	Energy harvesting constant

Table 3. *Cont.*

Notation	Meaning
ρ_0	Time policy for harvesting energy from HAP
ρ_1	Time policy for offloading task to MES
ε	Effective switching capacitance
θ	Path loss exponent

Figure 1 depicts the system model containing one MD, either computing a task locally or transmitting data to MEC and harvesting energy from HAP attached to MEC, along with downloading the computational data from MES.

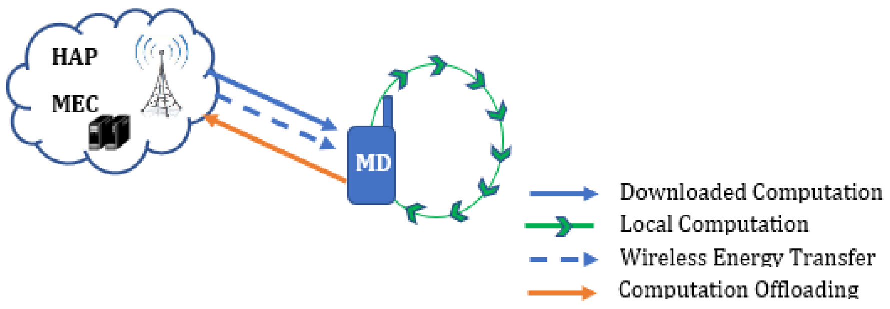

Figure 1. Wireless energy powered MEC system with one MD.

3.1. Local Execution Model

The time delay in the local execution model is represented by T_{li}, and is obtained by the relation

$$T_{li} = \frac{c_i L}{f_{di}}, \tag{1}$$

where c_i is the input data size of a component (measured in bytes) and L is measured in cycles/byte (cpb). This L indicates the number of clock cycles a microprocessor will use per byte of data processed in an algorithm. The parameter depends on the nature of the component, e.g., the complexity of the algorithm. The estimation of this value was studied in [9,15] and is beyond the scope of our work. f_{di} represents the frequency of MD. The energy consumption in this model [5] is calculated as

$$E_{li} = \varepsilon f_{di}^3 T_{li}, \tag{2}$$

where ε is defined as the effective switching capacitance factor which depends on chip architecture. εf_{di} represents the computing power of the MD [7].

3.2. Remote Execution Model

In the case that $x_i = 1$ for the ith component of a task, the executed data offloads the components of the task to HAP for high processing power. For simplicity, it is assumed that MEC has strong computing capabilities and HAP has a high transmission capacity. This renders the downloading delay from HAP to MD negligible, implying $\rho_0 + \rho_1 \approx 1$ [5].

The noise spectral density for the up-link is considered to be N_0, and the available bandwidth is represented by B. When considering small scale fading channel power gain h_d, independent and identically distributed (i.i.d.) channel fading is assumed between an MD and an HAP [16]. The channel power gain is $H_i = h_d g_0 (\frac{d_0}{d})^{\theta}$, where g_0 represents path loss, d_0 is the reference distance, and d is the distance between the HAP and the MD [8]. Using the Shannon–Hartley theorem [17], the data rate can be calculated as

$$r_i^{ul} = B \log_2(1 + \frac{H_i P_{di}}{BN_0}), \tag{3}$$

where r_i^{ul} represents the maximum data rate on the up-link channel from MD to HAP, and P_{di} denotes the transmit power of the MD. Similarly, the maximum amount of executed data on the down-link, transmitted by HAP to MD, is represented by r_i^{dl}, and is given as

$$r_i^{dl} = B \log_2(1 + \frac{H_i P_{si}}{BN_0}). \tag{4}$$

In (4), P_{si} represents the transmit power of MEC.

In order to obtain the time delay in the remote execution model, we consider the up-link time (time for data transmission), the down-link time (time for data reception), and the processing time in MEC [18]. The time delay in transferring data from MD to MEC is represented by T_{ui}, and is calculated as

$$T_{ui} = \frac{c_i}{r_i^{ul}}. \tag{5}$$

Similar to up-link, the down-link delay time, T_{di}, is calculated as

$$T_{di} = \frac{c_{i+1}}{r_i^{dl}}. \tag{6}$$

In (6), c_{i+1} represents that the executed data at MEC arrives on the MD in the next time slot. The time for the processing of data in MEC is represented as T_{ei}, and is calculated as

$$T_{ei} = \frac{c_i}{f_s}, \tag{7}$$

where f_s is the CPU frequency of MES. The total time delay, T_{oi}, for component i, is calculated as

$$T_{oi} = \rho_1(T_{ui} + T_{di} + T_{ei}). \tag{8}$$

It is worth mentioning that, as the frequency at MES is high, the time delay for the execution of data at HAP can be neglected [19]. Moreover, c_{i+1}, which is the output data at MEC, is small as compared to input data c_i , which renders the down-link time delay also negligible. Therefore, we can write

$$T_{oi} \approx T_{ui}\rho_1. \tag{9}$$

Now, the time delay in the remote model for component i can be defined as

$$T_{ci} = \begin{cases} T_{oi}, x_i = 1, \\ T_{li}, x_i = 0. \end{cases} \tag{10}$$

The total time delay, T_c, can thus be calculated as

$$T_c = \sum_{i=1}^{n} T_{oi} x_i + \sum_{i=1}^{n} T_{li}(1 - x_i). \tag{11}$$

The energy consumption of MD for c_i in this model, denoted as E_{oi}, is calculated as

$$E_{oi} = T_{oi} P_{di}. \tag{12}$$

We assume the energy from HAP via radio frequency signals can recharge the battery of MD, and that the computational capabilities of HAP are strong [10,20]. The power of HAP to transmit energy to MD on the down-link is denoted by P_t. To reduce complexity, it is assumed that the down-link channel gain for a component i, represented as h_{di}, is equal to the up-link channel gain $H = h_d$. Thus, the energy harvested by MD, E_{hi}, is defined as

$$E_{hi} = \rho_0 T_{oi} \eta P_t h_{di}, \tag{13}$$

where ρ_0 is the percent portion of offloading time (T_{oi}), η denotes the energy harvesting constant and represents how effectively the harvested energy of HAP can be received by the MD [10,20]. The main focus of our work is to find the offloading policy and time allocation policy with minimum cost in terms of energy consumption and time delay.

The battery of an MD is also considered in this paper to demonstrate the significance of the harvested energy, and is denoted by B_t. The energy consumed by the MD for c_i is the sum of consumed energies in the local and remote models, and is given as

$$E_{ci} = E_{oi} + E_{li}, \tag{14}$$

where E_{ci} can be denoted as

$$E_{ci} = \begin{cases} E_{oi}, x_i = 1, \\ E_{li}, x_i = 0. \end{cases} \tag{15}$$

The total energy consumption of the MD, E_c, is thus calculated as

$$E_c = \sum_{i=1}^{n} E_{oi} x_i + \sum_{i=1}^{n} E_{li}(1 - x_i). \tag{16}$$

An MD should satisfy the energy harvesting constraint for self-sustainable operation so that the total consumed energy at an MD does not exceed the harvested energy [7]

$$E_{ci} \leq B_t + E_{hi}. \tag{17}$$

After the execution of all the components of a task, we can calculate the execution rate as

$$R = \frac{\sum_{i=1}^{n} c_i}{\sum_{i=1}^{n} (1 - x_i) T_{li} + T_{oi} x_i}. \tag{18}$$

In (18), n represents the number of iterations required by the device to process specific task size bits. If a component is executed locally, that is $x_i = 0$, then T_{oi} must be zero, and for $x_i = 1$, T_{oi} must be one. For a meaningful comparison of our proposed technique with the benchmark techniques, we model a cost that considers energy consumption and time delay simultaneously, represented as γ, given as

$$\gamma = \alpha E_c + \beta T_c, \tag{19}$$

where

$$\alpha = \frac{w_1}{E_{max}}, \tag{20}$$

and

$$\beta = \frac{w_2}{T_{max}}. \tag{21}$$

In (20) and (21), E_{max} and T_{max}, i.e., the maximum energy consumed and time delay for the whole task, respectively, are taken as constants for all components of a task. w_1 and w_2 represent the weighting constants. This cost function, γ, is for optimal time allocation and offloading policy.

3.3. The Proposed Algorithm

The proposed algorithm, JTAOP, is presented as Algorithm 1. The algorithm first takes the inputs from the frequency range, transmit power range, and distance range between MD and HAP for n number of components of a task randomly. Thus, the number of possible offloading policies for n components per task can be calculated as 2^n. Then for all components of a task, time delay and energy consumption are calculated for local or remote models. After the measurement of time delay and energy consumption, the cost is determined by the function described in our mathematical model. For the remote model, the JTAOP algorithm then calculates the cost by varying all possible values of ρ_o. At the end of the algorithm, the minimum cost gives the time allocation and offloading policies with minimum cost. The algorithm calculates the cost values for all 2^n offloading policies, considering all possible values of ρ_o, represented by n_{ρ_o}; the complexity of the algorithm becomes $O(2^n n_{\rho_o})$. If the resolution of ρ_o increases, the complexity of the algorithm increases; however, the results tend towards high accuracy. Since the complexity increases with the number of components per task and the resolution of ρ_o, to reduce this extensive computational burden we train a DNN. The intensive calculations can be solved with a trained DNN, with complexity $O(1)$, on the training dataset to achieve high accuracy results. To generate the training dataset, we calculate all possible values of costs for all possible offloading and time allocation policies and then select the minimum cost, γ^*; the corresponding input data and labels are stored as training dataset. We also generate a dataset for testing the trained DNN and calculate its accuracy as $\frac{\text{number of correct decisions}}{\text{total number of tests}}$. The proposed algorithm is then compared with three existing benchmark schemes, namely, local computing only, total offloading only, and partial offloading. In the case of local computing only, all the components undergo local execution for the whole time slot. In the case of total offloading only, all the components are offloaded to MES for fast computation. In the case of partial offloading, some components of the task are locally executed while the remaining are offloaded to MES.

DNNs are emerging as an efficient solution for extensive computations due to their high accuracy performance. By increasing the size of trained dataset, their accuracy can be increased. Due to an increase in the number of components and resolution, our algorithm becomes more complex. To reduce this complexity, we use DNN. The data is divided randomly into training, validation, and test sets of ratios 75, 15, and 15 percent, respectively. To solve the combinatorial nature of the offloading problem discussed, DNNs are used. Using DNN, complexity is reduced as the network gets trained and outputs the result based on its gained experience.

Figure 2 shows the flowchart of DNN [21]. As depicted in the figure, for a specific task size, our proposed algorithm selects either the local model or the remote model of execution, based on the offloading policy. In the case of the remote model, cost is calculated for all possible values of ρ_0 and then total cost for all components is calculated, incorporating both local and remote models. Then, for 2^n offloading policies and n number of components, cost is calculated, and for the minimum cost, time allocation and offloading policy are stored as datasets for further training of DNN.

Algorithm 1: Joint Time Allocation and Offload Policy

```
1  Input: $c\epsilon[c_1,c_2...c_n]$, $f_d\epsilon[f_{dmin},f_{dmax}]$,$P_d\epsilon[P_{dmin},P_{dmax}]$,$d\epsilon[d_{min},d_{max}]$
2  Output: $\gamma^* = \arg\min_{\{b\epsilon\ 1,2,...2^n\}} \gamma_b$
3  for $b = 1 : 2^n$ do
4      for $j = 1 : n$ do
5          if $(x_i = 0)$
6          while ($c_j$ completes execution) do
7              $T_{li} \leftarrow c_i L / f_{di}$
8              $E_{li} \leftarrow \varepsilon f_{di}^3 T_{li}$
9          end while
10         else
11         for $k = 0 : r_s : 1$ do
12             while ($c_j$ completes execution) do
13                 $T_{ok} \leftarrow T_{ui}\rho_1$
14                 $E_{ok} \leftarrow \varepsilon T_{oi} P_{di}$
15             end while
16             $\gamma_k \leftarrow \alpha E_{ok} + \beta T_{ok}$
17         end for
18         $[index, \gamma_j] = min(\gamma_k)$
19         $T_{oj} \leftarrow T_{ok}(index)$
20         $E_{oj} \leftarrow E_{ok}(index)$
21         end if
22     end for
23     $\gamma_b \leftarrow \alpha \sum_{i=1}^{n} E_{li}(1 - x_i) + x_i E_{oj} + \beta \sum_{i=1}^{n} T_{li}(1 - x_i) + x_i T_{oj}$
24 end for
25 $\gamma^* = \min_{\{b=1,2,..,2^n\}}(\gamma_b)$
26 Save corresponding input data $\leftarrow [c_i, f_{di}, P_{di}, d_i]$
27 Save corresponding labels $\leftarrow [b^*, \rho_{o1}^*, \rho_{o2}^*]$
28 Repeat for different task size
29 Train DNN $\leftarrow$ train(input data,labels)
30 Test train DNN
31 Accuracy $\leftarrow$ number of correct decisions/number of total decisions
```

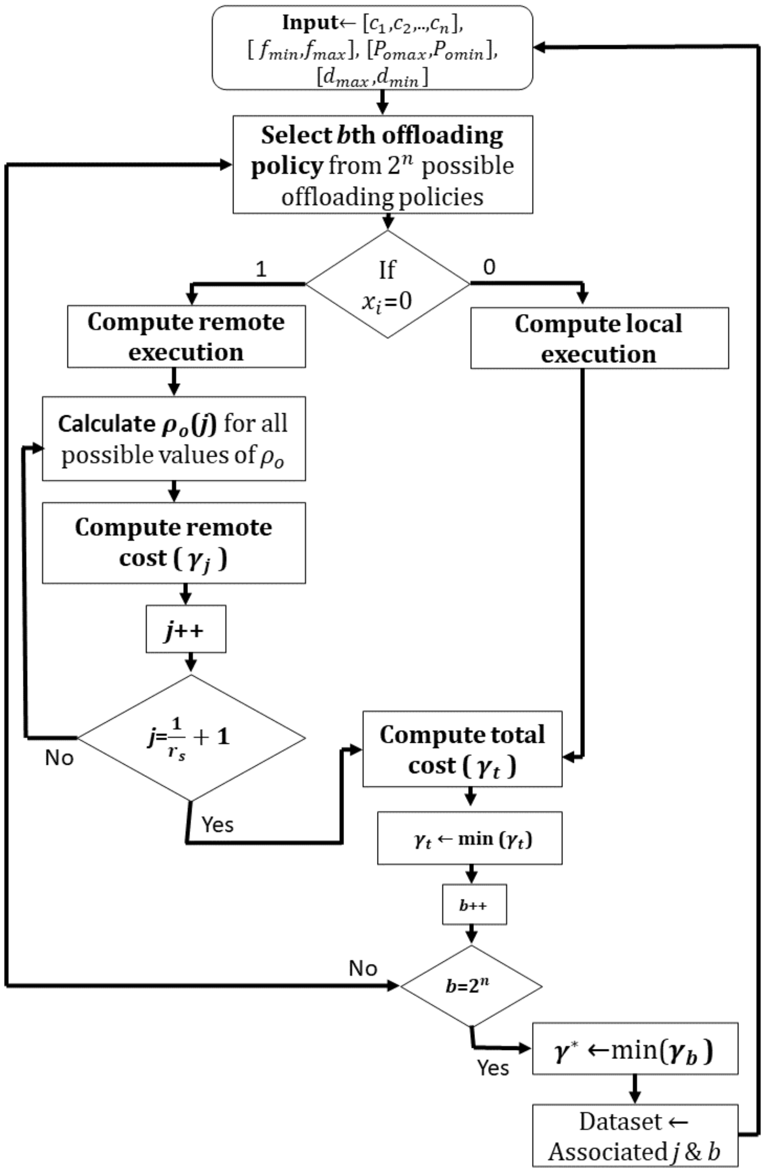
Input← [c_1,c_2,..,c_n], [f_{min},f_{max}], [P_{omax},P_{omin}], [d_{max},d_{min}]
Select bth offloading policy from 2^n possible offloading policies
If x_i=0
1
0
Compute remote execution
Compute local execution
Calculate $\rho_o(j)$ for all possible values of ρ_o
Compute remote cost (γ_j)
j++
$j=\frac{1}{r_s}+1$
No
Yes
Compute total cost (γ_t)
$\gamma_t \leftarrow \min(\gamma_t)$
b++
No
$b=2^n$
Yes
$\gamma^* \leftarrow \min(\gamma_b)$
Dataset ← Associated j & b

Figure 2. Flowchart of DNN.

4. Results

This section presents the results obtained by MATLAB (R2018a) simulations for the optimal time allocation and offloading policies. Each task is divided into three components. The task size is considered as a uniform distribution in $[0.1, 1]$ gigabit. The value of CPU cycles to process one bit of data is taken as 737.5 cycles per bit. The frequency of MD is considered as a uniform distribution in $[0.1, 1]$ GHz. For simulation purpose only and to check the effect of the mobility in the training dataset, the distance between HAP and MD is taken as a uniform distribution in $[3, 200]$ m. However, the training dataset can be generated for any suitable range of distances between HAP and MDs. Effective switching capacitance is taken as 10^{-25}. Efficiency η of MD is taken as 0.8. g_0 is taken as -30 dB, $\theta \geq 2$, and $d_0 = 1$. The available bandwidth is taken as 0.5 MHz. The noise spectral density N_0 is -174 dBm/Hz. The weighing constants are taken as 1. The offloading transmission power of MD is taken as a uniform distribution in $[0.8, 1.5]$ W.

The validation of the dataset is determined by the proposed mathematical model because three datasets using the proposed mathematical model, namely, the training dataset, validation dataset, and test dataset, are generated. In each dataset, the input data contains the "size of each component, frequency of the user equipment for execution, transmit power of MD, distance between MD and MES"; therefore, the input layer of the DNN has 12 neurons. Similarly, using this input data information, our proposed mathematical model calculates the output data (labels for DNN).

For different possible offloading policies with different data sizes, the minimum and maximum value for harvesting percentage time of a single time slot are observed to be 26.1 and 64 percent, respectively. For simulation, r_s is selected as 0.001. There are 380 points between 0.261 and 0.64. Therefore, the output layer contains 380 neurons for ρ_0 and 3 neurons for offloading policy (total 383 neurons in the output layer). There are two hidden layers, each containing 100 neurons (through cross validation) and the Softmax activation function [22] is used to train the DNN. The accuracy of the network is calculated using the test dataset. The training function used is trainscg (scaled conjugate gradient backpropagation). The higher the number of neurons, the greater the complexity of the network. Similar is the case with the number of hidden layers, i.e., the higher the number of hidden layers, the greater the network complexity, but using two hidden layers in our dataset provided the maximum accuracy. In considering DNN, we have 10 datasets of samples from 1000 to 10,000 by calculating data via state variables. We measured the cost for all possible combinations of data to obtain the minimal cost. The trained network was then tested for the discrete 15 percent data. The MATLAB toolbox "nprtool" was used to solve the classification problem. The accuracy of the DNN was plotted by increasing the size of the training dataset, as shown in Figure 3. The simulation parameters are given in Table 4.

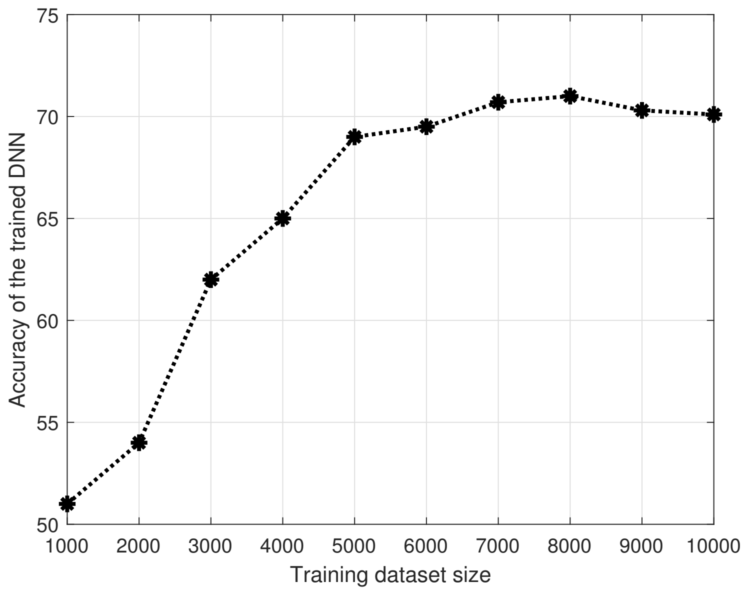

Figure 3. Accuracy vs. training dataset size.

Table 4. Simulation Parameters.

Parameters	Values
B	0.5 MHz
d	[3–200] m
d_0	1 m
f_{di}	[0.1–1] GHz
g_0	−30 dB
L	737.5 cycles/bit
N_0	−174 dBm/Hz
ε	10^{-25}
η	0.8
θ	≥2

In Figure 4, energy consumed by an MD is plotted verses varying task size. With the increasing task size, the energy consumption of all the methods increases. This is because a task size of 1 gigabit requires more energy for its completion than that of 0.1 gigabit. However, as is evident from the figure, in case of our JTAOP algorithm, the minimum amount of energy is consumed by the MD, making it the most efficient method. This is because JTAOP considers the optimal offloading policy for energy harvesting and offloading, emphasizing minimum energy expenditure. The maximum energy is consumed in the case of total offloading as all the components of the task are offloaded to MES, which leads to significant energy expenditure, followed by partial offloading and then total local execution.

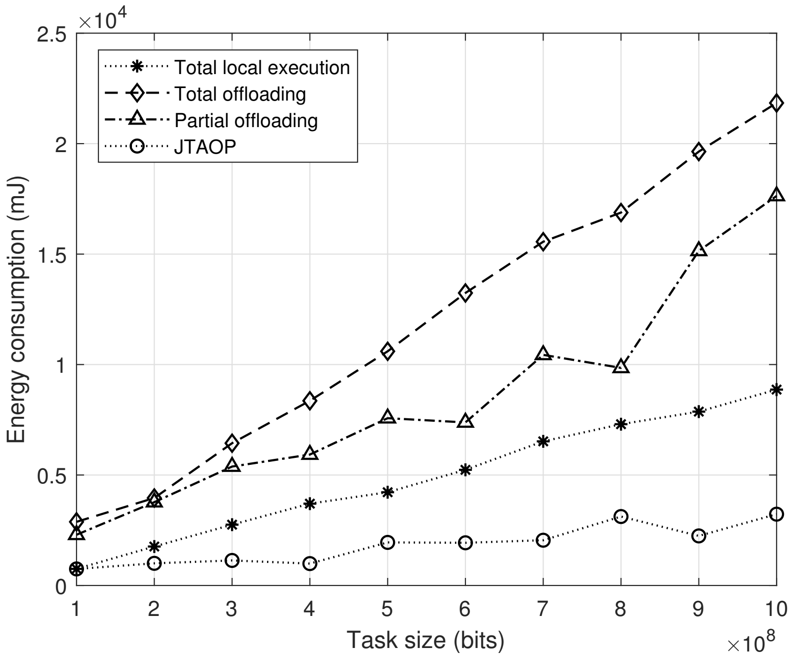

Figure 4. Consumed energy of an MD vs. task size.

Figure 5 explains the relation of cost with the increasing task size. An increasing task size leads to an increasing trend of cost as more energy and time is required for the completion of a large task size. As depicted in the figure, our JTAOP algorithm results in minimum cost, which shows its effectiveness. This minimum cost is due to the joint consideration of time delay and energy consumption of MD in the cost function. While energy conservation is second best in the case of total local execution, nonetheless, maximum cost is observed in that case, as MD takes a lot of time, leading to high computational delay in local execution, rendering it inefficient.

Figure 6 gives the relationship of execution rate with increasing task size. While the maximum execution rate can be observed in the case of total offloading because of the powerful execution power available at MES, it is at the expense of energy consumption and cost. Increasing the execution rate directly affects the energy consumption and cost of a scheme, which shows the trade-off between the energy consumed by an MD and the execution rate. This trade-off is observed in JTAOP where the optimal policy is selected for the maximum execution rate with the minimum energy consumed by the MD. The JTAOP algorithm not only demonstrates minimum energy consumption performance and minimum cost, but also shows a satisfactory performance in terms of execution rate.

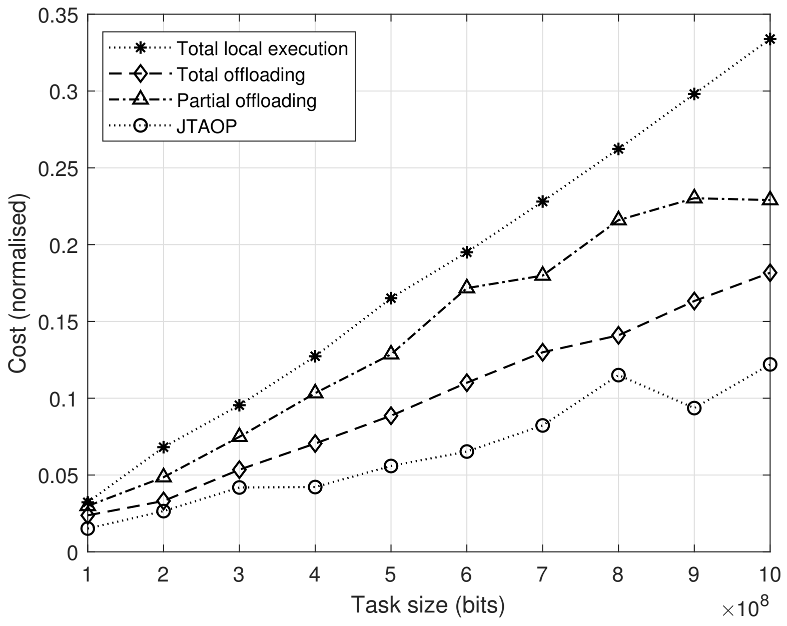

Figure 5. Cost vs. task size.

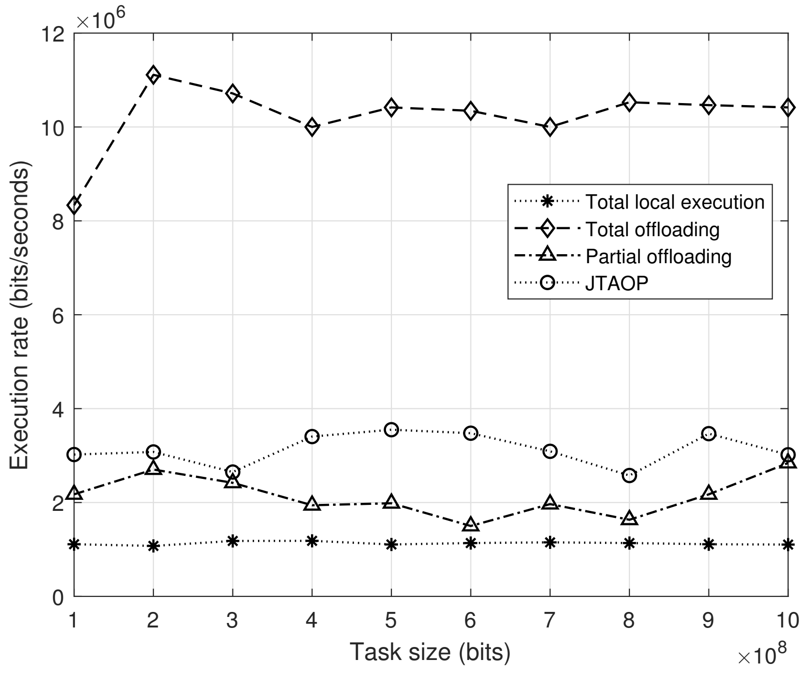

Figure 6. Execution rate vs. task size.

In Figure 7, we present the relation of delay with task size. Increasing the task size enforces all the mentioned benchmark schemes to increase their time of task execution. While minimum delay is observed in the case of total offloading as the computational power at MES is high, the cost and energy consumption performance of MD in this case is not promising, as compared to JTAOP. The JTAOP algorithm jointly considers energy consumption and optimal time policy for an MD via a trained DNN. The trained DNN decreases the computational time delay considerably, which leads to a fine performance, and produces a minimum cost. Figure 8 shows the accuracy for three different neural networks with varying dataset size. As depicted in the figure, a two layered network is the optimal choice for this type of data.

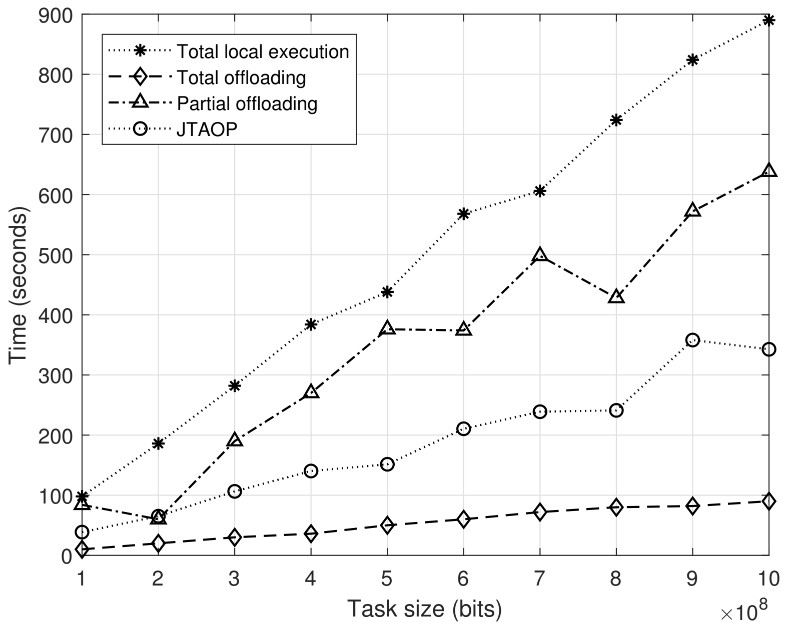

Figure 7. Delay vs. task size.

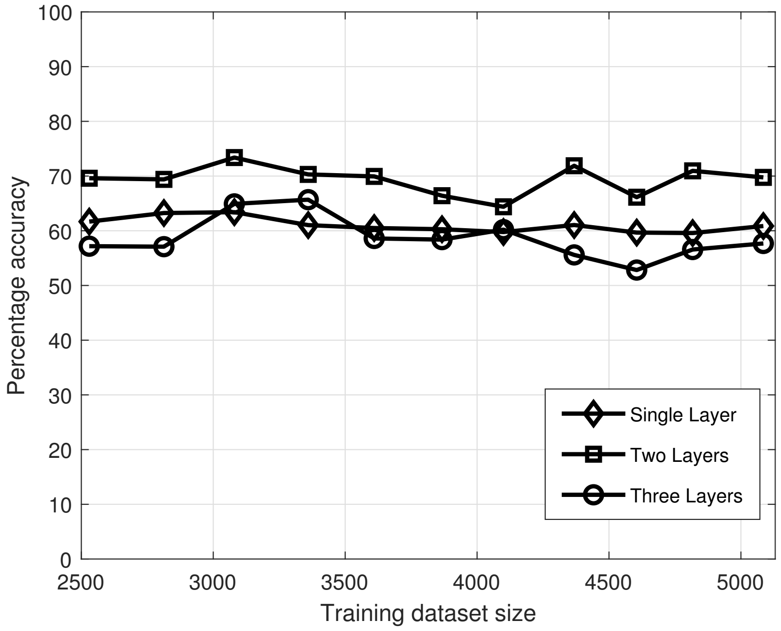

Figure 8. Neural network accuracy with different number of hidden layers.

5. Discussion

The limited battery and computational capacity of an MD is addressed with the help of wireless energy transfer in a mobile edge computing system. In WP-MEC systems, for efficient performance of MD, the cost involved in the process of wireless energy transfer is targeted to be minimum for the optimal time allocation and offloading policy. All the previous works considered either the energy consumption or the time delay separately, mainly focusing on maximizing the execution rate. In this paper, we considered a partial offloading scheme for one MD and proposed an algorithm, JTAOP, which considers the energy consumption by the MD and the time delay simultaneously via trained DNN. Our JTAOP algorithm is scalable in terms of resolution of time. We considered 10^{-4} millisecond resolution for time. However, if a scenario needs more or less resolution, then by changing the resolution in the training dataset we can get the required trained DNN. JTAOP is also versatile in terms of different numbers of components of a single task. We considered three components of a single task; however, if a scenario needs more components, our technique can generate the required dataset by changing only the number of components per task. Simulation results show that our algorithm gives minimum cost, considering energy consumption and time delay, with a considerable execution rate, as compared to other benchmark schemes. However, our algorithm, in dealing with partial offloading, only considers those tasks that can be divided into different numbers of components and does not consider non divisible tasks.

6. Conclusions

This paper investigates optimal time allocation and offloading policies in a wireless powered MEC system. We proposed a scalable algorithm in terms of the resolution of time fraction and the number of components of a task, and solved the partial offloading scheme by considering a deep learning approach. Minimization of cost and energy consumption of MD are studied by considering an MD and double antenna HAP with the help of a trained DNN. Simulation results show that our proposed scheme is more cost efficient as compared with the other considered schemes. The trade-off between energy consumption and time delay is also studied to find the optimal policy that gives the minimum cost and energy consumption, simultaneously. For future work, more realistic factors, like MES resource limitation could be considered in the cost function to make it more practical.

Author Contributions: Conceptualization, A.I., Z.H.A. and Z.A.; methodology, A.I. and Z.A.; software, G.A. and T.B.; validation, Z.A., G.A., Z.H.A., T.B. and D.A.-J.; Formal analysis, Z.H.A., G.A., T.B. and D.A.-J.; investigation, A.I., G.A., T.B. and D.A.-J.; resources, G.A., T.B. and D.A.-J.; data curation, A.I., Z.A., Z.H.A. and G.A.; writing—original draft preparation, A.I.; writing—review and editing, G.A., Z.A., Z.H.A., T.B. and D.A.-J.; visualization, A.I. and T.B.; supervision, Z.A., G.A. and Z.H.A.; project administration, G.A. and Z.H.A.; funding acquisition, G.A., T.B., and D.A.-J. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ratasuk, R.; Prasad, A.; Li, Z.; Ghosh, A.; Uusitalo, M. Recent advancements in M2M communications in 4G networks and evolution towards 5G. In Proceedings of the 18th International Conference on Intelligence in Next Generation Networks, Paris, France, 17–19 February 2015; pp. 52–57.
2. Ren, J.; Yu, G.; He, Y.; Li, G.Y. Collaborative cloud and edge computing for latency minimization. *IEEE Trans. Veh. Technol.* **2019**, *68*, 5031–5044. [CrossRef]
3. Ali, Z.; Jiao, L.; Baker, T.; Abbas, G.; Abbas, Z.H.; Khaf, S. A deep learning approach for energy efficient computational offloading in mobile edge computing. *IEEE Access* **2019**, *7*, 149623–149633. [CrossRef]
4. Wu, H.; Tian, H.; Nie, G.; Zhao, P. Wireless powered mobile edge computing for industrial internet of things systems. *IEEE Access* **2020**, *8*, 101539–101549. [CrossRef]
5. Li, C.; Song, M.; Zhang, L.; Chen, W.; Luo, Y. Offloading optimization and time allocation for multiuser wireless energy transfer based mobile edge computing system. *Mob. Netw. Appl.* **2020**. [CrossRef]
6. Yu, S.; Chen, X.; Yang, L.; Wu, D.; Bennis, M.; Zhang, J. Intelligent edge: Leveraging deep imitation learning for mobile edge computation offloading. *IEEE Wirel. Commun.* **2020**, *27*, 92–99. [CrossRef]
7. Zhou, F.; Wu, Y.; Hu, R.Q.; Qian, Y. Computation rate maximization in UAV-Enabled wireless-powered mobile-edge computing systems. *IEEE J. Sel. Areas Commun.* **2018**, *36*, 1927–1941. [CrossRef]
8. Wang, F.; Xu, J.; Wang, X.; Cui, S. Joint offloading and computing optimization in wireless powered mobile-edge computing systems. *IEEE J. Sel. Areas Commun.* **2018**, *17*, 1784–1797. [CrossRef]
9. Bi, S.; Zhang, Y.J. Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading. *IEEE Trans. Wirel. Commun.* **2018**, *17*, 4177–4190. [CrossRef]
10. Li, C.; Chen, W.; Tang, H.; Xin, Y.; Luo, Y. Stochastic computation resource allocation for mobile edge computing powered by wireless energy transfer. *Ad Hoc Netw.* **2019**, *93*, 101897. [CrossRef]

11. Wu, D.; Wang, F.; Cao, X.; Xu, J. Wireless powered user cooperative computation in mobile edge computing systems. In Proceedings of the 2018 IEEE Globecom Workshops (GC Wkshps), Abu Dhabi, United Arab Emirates, 9–13 December 2018; pp. 1–7.
12. Mao, S.; Leng, S.; Yang, K.; Huang, X.; Zhao, Q. Fair energy-efficient scheduling in wireless powered full-duplex mobile-edge computing systems. In Proceedings of the IEEE Global Communications Conference, Singapore, 4–8 December 2017; pp. 1–6.
13. Mao, S.; Leng, S.; Maharjan, S.; Zhang, Y. Energy efficiency and delay tradeoff for wireless powered mobile-edge computing systems with multi-access schemes. *IEEE Trans. Wirel. Commun.* **2020**, *19*, 1855–1867. [CrossRef]
14. Bi, S.; Ho, C.K.; Rui, Z. Wireless powered communication: Opportunities and challenges. *IEEE Commun. Mag.* **2014**, *53*, 117–125.
15. Yang, M.; Wen, Y.; Cai, J.; Foh, C.H. Energy minimization via dynamic voltage scaling for real-time video encoding on mobile devices. In Proceedings of the IEEE International Conference on Communications, (ICC'12), Ottawa, ON, Canada, 10–15 June 2012; pp. 2026–2031.
16. Koech, K. Softmax Activation Function-How It Actually Works. Available online: https://towardsdatascience.com/softmax-activation-function-how-it-actually-works-d292d335bd78 (accessed on 1 January 2021).
17. Huang, L.; Bi, S.; Zhang, Y.J.A. Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks. *IEEE Trans. Mob. Comput.* **2020**, *19*, 2581–2593. [CrossRef]
18. Liu, B.; Xu, H.; Zhou, X. Resource allocation in wireless-powered mobile edge computing systems for internet of things applications. *Electronics* **2019**, *8*, 206. [CrossRef]
19. Wang, F.; Xu, J.; Cui, S. Optimal energy allocation and task offloading Policy for wireless powered mobile edge computing systems. *IEEE Trans. Wirel. Commun.* **2020**, *19*, 2443–2459. [CrossRef]
20. Wu, Q.; Chen, W.; Ng, D.W.K.; Li, J.; Schober, R. User-centric energy efficiency maximization for wireless powered communications. *IEEE Trans. Wirel. Commun.* **2016**, *15*, 6898–6912. [CrossRef]
21. Mao, S.; Wu, J.; Liu, L.; Lan, D.; Taherkordi, A. Energy-efficient cooperative communication and computation for wireless powered mobile-edge computing. *IEEE Syst. J.* **2020**, *6*, 1–12.
22. Back to Basics: The Shannon-Hartley Theorem-Ingenu. Available online: https://www.ingenu.com/2016/07/back-to-basics-the-shannon-hartley-theorem/ (accessed on 10 December 2020).