Intelligent energy prediction techniques for fog computing networks

Umar Farooq[a], Muhammad Wasif Shabir[a], Muhammad Awais Javed[a,*], Muhammad Imran[b]

[a]*Department of Electrical and Computer Engineering, COMSATS University Islamabad, 45550, Islamabad*
[b]*College of Computer and Information Sciences, King Saud University, Riyadh, 11451, Saudi Arabia*

Abstract

Energy Efficiency is a key concern for future fog-enabled Internet of Things (IoT). Since Fog Nodes (FNs) are energy-constrained devices, task offloading techniques must consider the energy consumption of the FNs to maximize the performance of IoT applications. In this context, accurate energy prediction can enable the development of intelligent energy-aware task offloading techniques. In this paper, we present two energy prediction techniques, the first one is based on the Recursive Least Square (RLS) filter and the second one uses the Artificial Neural Network (ANN). Both techniques use inputs such as the number of tasks and size of the tasks to predict the energy consumption at different fog nodes. Simulation results show that both techniques have a root mean square error of less than 3%. However, the ANN-based technique shows up to 20% less root mean square error as compared to the RLS-based technique.

Keywords: Fog computing, Artificial Neural Network, Energy Prediction.

1. Introduction

Electronic devices having the ability to connect with the Internet is prevalent in the current era. A huge number of devices will be connected to the Internet in the near future. These devices are monitored, controlled, and accessed via the

*Corresponding authors: Muhammad Awais Javed, email: awais.javed@comsats.edu.pk

Internet, and thus form an Internet of Things (IoT) network. It has a range of applications in areas such as health care, transportation, agriculture, industrial automation, and security [1, 2, 3, 4].

As the IoT devices are rapidly increasing in number, they are generating a huge amount of data. To process this large data, nodes with high computational capacity such as cloud servers are needed [5, 6, 7, 8, 9, 10, 11, 12, 13]. Cloud computing can reduce the computation burden on the IoT nodes, however, the latency of the computation tasks may be increased due to long-distance transmission of tasks between the IoT devices and the cloud. Since various applications such as industrial automation require short computation delay to meet the Quality of Service (QoS) requirement [14].

A potential solution to the above disadvantage of cloud computing is the use of a fog computing paradigm where several fog nodes (with relatively less computational capacity than the cloud) are placed near to the IoT devices [15, 16, 17, 18, 19, 20, 21, 22, 23, 24]. As the tasks can easily be transmitted by the IoT devices to the nearby fog nodes, task computation time is significantly reduced. Moreover, the computational load can be efficiently distributed among the fog nodes, resulting in a decentralized computation model [25, 26].

Energy is a major challenge in IoT enabled fog computing [27, 28, 29, 30, 31]. As both IoT devices and fog nodes have energy limitations [32], it is critical to devise techniques that can enhance the energy efficiency of these nodes. At IoT nodes, all tasks can not be processed efficiently because of limited computation capacity and energy constraints. Similarly, the battery of the fog nodes needs to be managed well so that lifetime of fog nodes can be increased [33].

To improve the energy efficiency of fog nodes, techniques such as energy-aware task offloading, load balancing, transmit power optimization, etc. are used. All these techniques rely on either the real-time energy consumption values to make decisions or use predicted energy consumption values to develop the optimal task offloading scheme. Thus, accurate prediction of energy consumption of fog nodes is vital to plan the computational resources. Currently, there is very little work related to energy prediction in the context of fog networks.

In this paper, we develop two energy prediction techniques for FNs based on RLS and ANN algorithms respectively. The key idea is to use data such as the number of tasks and size of tasks to predict the future energy consumption at different FNs. The major motivation of such a prediction is to develop energy-aware task offloading algorithms in the future. Simulation results show the prediction accuracy of both techniques. Finally, ANN shows a better prediction performance as compared to the RLS algorithm.

The rest of the paper is organized as follows. Section 2 presents a literature review related to energy prediction and energy-aware offloading techniques. Section 3 presents the system model followed by proposed techniques in Section 4. Section 5 presents the simulation results. Finally, conclusions are presented in Section 6.

2. Related Work

In this section, we provide an overview of energy prediction techniques in fog computing and other wireless networks. We also provide the recent work done related to energy-aware task offloading in fog computing networks.

2.1. Intelligent Energy Predication Techniques

Intelligent energy prediction techniques have been proposed in the literature related to different electrical and wireless systems such as solar systems, electrical grids, and wireless sensor networks. Authors in [34] proposed energy prediction of different devices for smart homes. Devices send data to the fog nodes for quick real-time analysis. A novel incremental learning algorithm based on decision trees is proposed to predict energy. Based on the input data, a decision table is constructed. The algorithm has three parts, the first works on feature selection, the second selects appropriate rules for decision trees, and the last part works on the development of a decision table. Results show increased accuracy of prediction and reduced time consumption.

In [35], authors considered a wind-powered energy harvesting scenario for wireless sensor networks. The amount of harvested energy varies with time and

Table 1: Recent work related to intelligent energy prediction techniques

Scenario	Key Idea	Results
Fog computing enabled Smart homes [34]	Device energy prediction in smart home Fog nodes collect and analyse data Learning algorithm using decision trees	Increased accuracy Reduced time consumption
Wind harvested energy based sensor networks [35]	Wind harvesting for sensor networks Harvested energy varies with time Current and past data considered Energy profile matching	Increased accuracy
Routing in mobile edge computing [36]	Routing in mobile edge computing Low energy nodes can not discover routes Routing based on link connectivity and energy	Reduced energy consumption Reduced end-to-end delay
RF harvested energy [37]	Learning based RF energy prediction First technique is linear regression based Second technique is decision tree based Received power experimentally evaluated	Improved results by regression

availability of sources. The proposed algorithm predicts the energy consumption for the future based on current as well as past energy conditions. The algorithm stores past energy profiles and based on the current energy condition it finds the most matching profile. Results show increased accuracy of energy prediction.

The work in [36] proposes an efficient routing algorithm for mobile edge computing networks. As energy is consumed in the route discovery process, nodes with low energy values do not take part in the route discovery process. The proposed algorithm performs routing based on the routing link lifetime connectivity and its energy consumption. The algorithm reduces energy consumption and end-to-end delay.

In [37], the authors propose a machine learning-based RF harvested energy prediction algorithm. Two techniques have been proposed for energy prediction. The first technique is based on a linear regression algorithm whereas the second technique uses decision trees. Authors experimentally evaluate RF received power in different scenarios and use it for energy prediction with the help of

the two learning algorithms. Results show improved prediction accuracy for the linear regression algorithm.

Table 2: Recent work related to energy-aware task offloading

Scenario	Key Idea	Results
Internet of Things in health care [38]	Health related data offloading URLLC used for transmission Multi-armed bandit learning algorithm used URLLC constraints considered in task offloading	Reduced energy consumption Reduced delay
D2D based mobile edge computing [39]	Task and power allocation optimized Energy, task delay and mobility considered Genetic and heuristic scheduling used	Reduced latency Improved energy consumption
IoT based mobile edge computing [40]	Minimize energy to meet delay constraints Deep reinforcement learning technique used Selection of best edge node to offload Computational resource allocation optimized Reward based on task delay and energy	Improved energy consumption Reduced latency
Mobile edge computing networks [41]	multi-level edge computing system considered offloading based on combinatorial optimization Tasks partitioned based on component call graphs Goal is to minimize the complexity of task partition Reliability by duplicate tasks computed at low speed	Improved reliability Improved energy consumption

2.2. Energy-Aware Task Offloading

Task offloading is a major application of fog computing. It is vital to consider energy while making task offloading decisions. Several works in the literature have developed energy-aware task offloading algorithms. In [38], health-related data is offloaded by sensors to the edge nodes. Ultra-Reliable Low Latency Communications (URLLC) is used for transmitting the tasks from sensors to the edge nodes. The proposed algorithm uses a multi-armed bandit learning algorithm for efficient offloading decisions. URLLC constraints are also considered in task offloading decisions. Results highlight reduced latency and improved energy consumption.

The work in [39] considers Device to Device (D2D) communications based mobile edge computing scenario. The goal of the proposed technique is to

optimize task allocation and power allocation. Factors such as energy, task delay, and mobility are considered while making allocation decisions. A genetic algorithm and heuristic energy-aware scheduling are used. Results show reduced latency and improved energy consumption.

In [40], authors aim to minimize the energy consumption of IoT-based mobile edge computing networks while meeting the task delay constraints. The deep reinforcement learning technique is used to select the best edge node for offloading. Furthermore, Computational resource allocation at edge nodes is also optimized. The reward function proposed in the paper is a utility function based on task delay and energy. Results show improved energy consumption and reduced latency.

The work in [41] considers a multi-level edge computing system. The problem of task allocation from multiple users to multiple edge nodes is solved using combinatorial optimization. Tasks are also partitioned and allocated to different edge servers based on component call graphs. The goal of the proposed technique is to minimize the complexity of task partition. The proposed algorithms also improve reliability by using a shadowing scheme where the same copy of the task is stored at a nearby edge node and computed at a low speed. Results show improved reliability and energy consumption.

To cope with the resource limitations of mobile devices, the authors in [42] proposed a lightweight process migration-based computational offloading framework for IoT-supported mobile edge cloud computing (MECC). To improve cognitive ability of edge intelligent industrial IoT, a machine learning-based framework and methods are proposed in [43]. Although the proposed framework takes into account resource constraints, however, they do not specifically consider energy prediction aspects of fog networks. To enable device-centric adaptive data management and offloading, an adaptive execution model for mobile data stream mining applications in context of MECC is proposed in [44]. The proposed model considers multiple factors such as choice of learning models, learning rates, learning modes, and limited computational resources. However, it does not make any energy predictions about fog networks.

2.3. Motivation of our work

Energy prediction of fog nodes is vital to design robust task offloading and caching algorithms. As highlighted in Section II-A, there has been very little work related to energy prediction for fog computing networks. The goal of this paper is to propose energy prediction techniques for fog nodes and evaluate their performance in a task offloading scenario.

The main contributions of this paper are as follows:

- We propose a Recursive Least Square (RLS) based energy prediction technique for fog nodes that uses time-series data
- We propose a Artificial Neural Network (ANN) based energy prediction technique for fog nodes that uses the number of tasks and the task size as inputs
- We provide detailed energy prediction simulation results for a realistic fog scenario and show that ANN-based energy prediction outperforms RLS-based energy prediction.

3. System Model

An illustration of IoTs based fog enabled network used in this paper is given in Figure 1. We consider a central controller which is controlling n number of FNs where $n = \{1, 2, 3...N\}$. Each FN assists the IoTs nodes or end-users inefficient task computation. IoT nodes are connected to the FNs only and the central controller manages these FNs. Although the FNs have much higher computation capacity than the IoT nodes, they still have limited computation capability, storage capacity, and available power. The central controller performs two major tasks, first is to collect network information such as the number of upcoming task offloading requests and current computation load at each FNs. The second task performed by the controller is to make efficient decisions about offloading and load distribution among the FNs.

We present the task offloading, delay, and energy consumption model considered in this paper in the following subsections.

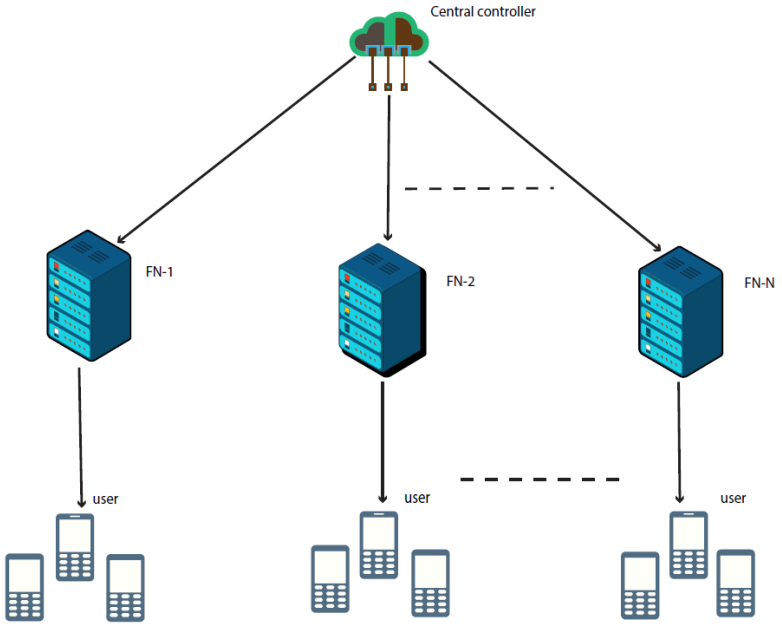

Figure 1: System Model

3.1. Task offloading Model

The number of tasks offloaded by the users to the FNs for computation is denoted by K_m and the size of each task in bits is given by l_m. Furthermore, the number of CPU cycles needed to operate or process each task at the fog node and end-user is given by η_m and η_u. It is assumed that the FNs compute tasks one by one on First Come First Serve (FCFS) basis. We assume that the number of requests at FN follows a Poisson distribution [45] with the average rate of λ.

3.2. Delay Model

The task delay model incorporates two parts, first is the total computation delay (for all tasks) at a FNs denoted as d_m [46] and second is the queuing

delay faced by the tasks, denoted as as q_m. The queuing delay considers the time when the fog node is busy in other task computation. The total delay for tasks at a FNs, denoted by d_{FN} can be computed as follows:

$$d_{\mathrm{FN}} = d_m + q_m \tag{1}$$

where d_m is given as:

$$d_m = \frac{l_m \eta_m}{f_m} + \frac{l_m}{W B_m} \tag{2}$$

Here f_m is the CPU frequency of FN, W is the spectrum bandwidth for offloading, and B_m is the spectral efficiency of the established wireless link between the user and the FN. Moreover, $l_m \eta_m$ cycles are needed for the computation of offloaded tasks. Once the tasks are processed by the FN, the result is transmitted back to the users and this can further increase the total offloading delay.

3.3. Energy Consumption Model

We assume that the total energy consumption to process a task at FNs is given by E_m. We can compute E_m as follows [46]:

$$E_{\mathrm{m}} = \eta_m l_m \theta_m \tag{3}$$

Here η_m is the computing energy requirement at FNs with CPU cycles, l_m is the task size, and θ_m is the energy consumed per CPU cycle of FNs.

In this paper, we do not consider the energy required for receiving a task and energy for transmitting the task output back to the IoT nodes [45].

4. Proposed Energy Prediction Techniques

In this section, we present the proposed energy prediction techniques based on RLS Adaptive Filter and ANN.

4.1. RLS Adaptive Filters

Adoptive RLS is widely used in the area of adaptive Digital Signal Processing (DSP). The domain adaptive DSP is accelerating rapidly, these filters with little variation and remodeling are abundantly used in various real-world applications. Generally, adaptive filters are used for unspecified conditions and environments but with the help of finite or infinite impulse response along with their adaptive coefficients are utilize and these coefficients are varying with time. RLS and least mean square are the most famous algorithm used in time series prediction.

RLS is used in this paper for the prediction of energy and delay of task requests at FNs. The coefficients of RLS are adjusted recursively throughout the process according to input data. It has high computation cost and rapid convergences, the main focus of RLS is to minimize the objective function i.e. sum of squared error. For efficient exploitation of input data in the non-stationary environment the weight factor is also introduced which makes certain that minimum weight is allocated to earlier error values. For the execution of the RLS algorithm following equations are used [47] :
RLS filter final output is calculated by Eq. 4

$$y_{i-1}(i) = w^T(i-1)x(i) \tag{4}$$

where the w_T and x_i represents the weights and input of RLS filter respectively.

Furthermore the mean gain of RLS filter is given by Eqs. 5 and 6

$$u(i) = \psi_{\lambda}^{-1}(i-1)x(i) \tag{5}$$

$$K(i) = \frac{u(i)}{(\lambda + x^T(i)u(i))} \tag{6}$$

Calculation of error estimation is performed by using Eq. 7 which is defined as difference of desired output to output of filter.

$$e_{i-1}(n) = d(i) - y_{i-1}(i) \tag{7}$$

Additionally the weights vector of filter is updated by following Eq. 8

$$w(i) = w^T(i-1) + K(i)e_{i-1}(i) \tag{8}$$

and the calculation of inverse matrix performed by Eq. 9

$$\psi_{\lambda}^{-1}(i) = \lambda^{-1}(\psi_{\lambda}^{-1}(i-1) - K(i)\big[x^{T}(i)\psi_{\lambda}^{-1}(i-1)\big]) \tag{9}$$

The λ is described as the forgetting factor,whose value is from 0 to 1. Eqs. 4-9 are applied in adoptive filter to get our desired target.

4.2. ANN based energy prediction

ANN is a commonly used technique in AI that facilitates models to learn how to solve complex problems using the available data. ANN consist of a function signal and an error signal, the function signal is an input that propagates from several nodes (neurons) to reach the output layer. The error signal emerges as an output of each node which circulates backward layer by layer [48]. As shown in Figure 2, the model in ANN is fed with the input parameters. These inputs are passed through a different number of hidden layers and in each hidden layer, output data is the weighted sum of inputs based on some logic. ANN learns the values of weights using training data. As the amount of training data increases, the performance of ANN becomes better.

In our proposed energy prediction technique, we use ANN based on two inputs as follows:

- **Number of tasks at a FN:** The first input variable of ANN is the number of tasks received by the FNs for computation. The reason to take this as an input variable is that the energy consumed by the FNs is directly proportional to the number of tasks.

- **Task Size:** The second input used for ANN is the task size. As in a fog network, different number tasks are served by the FNs and each task has a different size, therefore energy consumed by the FNs also depends on the task size. As an example, tasks such as web browsing and node position computation are of less size (require low computation time) as compared to the multimedia tasks such as streaming High-Quality videos.

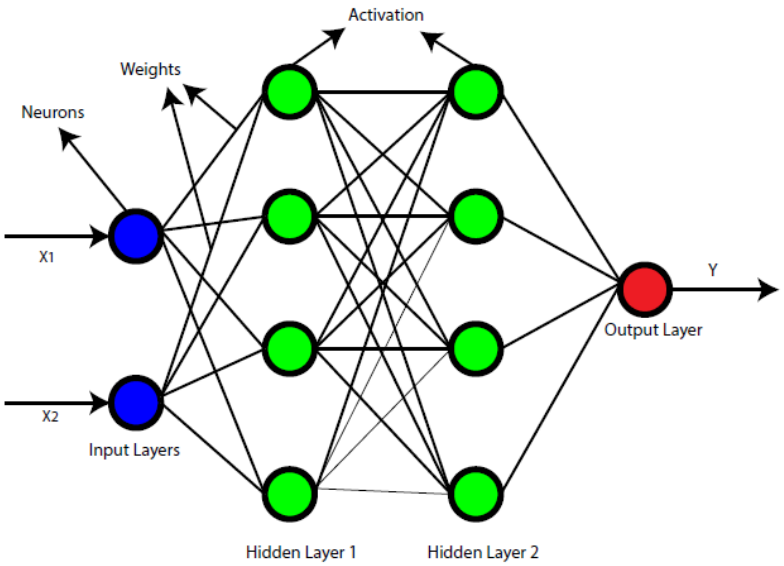

Figure 2: Structure of ANN

The model has a single output which is energy consumption at an FNs. Generally, the internal parameters of fog devices such as voltage and processing frequency of CPU per bit task size are kept constant. However, few works such as [45, 49] dynamically adapt these parameters for improving the performance of fog devices and the overall network.

The internal ANN structure used for energy prediction is a three-layer and two-input ANN. One of the most important tasks is to find the appropriate number of neurons in the hidden layers for which our prediction results become accurate depending on the complexity of the scenario and input data set. The multiple inputs given to the model are defined as x_i and these inputs are multiplied with weights w_{ij}. Here w_{ij} represents the weights between node i of the input layer of the model to the node j in the hidden layer. The bias b_j is also added to acquire net input as given in Eq. 10, which is then provided to the hidden layer by the Rectified Linear Unit (ReLU) function $f(V)$ as stated in

Eq. 11 to get the desired output y_j as in Eq. 12 [48, 50].

$$V_j = \sum_{i=1}^{m} x_i w_{ij} + b_j \tag{10}$$

$$f(V) = max(0, a) \tag{11}$$

$$y_j = f(V_j) \tag{12}$$

The ReLU function provides an output a, if the input is positive, otherwise, the output is zero. The nature of the ReLU function is non-linear and it is used at each layer to get the desired output. The output from the hidden layer is then passed on to the output layer where we have a k number of neurons.

$$Z'_k = \sum_{j=1}^{h} y_j w'_{jk} + b'_k \tag{13}$$

Eq. 13 computes input of k neurons of output layer where b'_k is the bias and w'_{jk} are the weights between j neurons of hidden layer to k neurons of output layer. The output Z_k is given by passing through the activation function by feed-forward (FF) propagation.

$$Z_k = f(Z'_k) \tag{14}$$

After calculating the output of model Z_k, it is then compared to the target output T_k. e_k is calculated for all targets and measures outputs until the values of error are reduced to the desired value [50].

$$e_k = \frac{1}{2}\sum_{k=1}^{K}(T_k - Z_k) \tag{15}$$

After finding e_k, using Eq. 15, the gradient error at the output layer is specified by Eq. 16. Furthermore, we also used the Backpropagation (BP) algorithm with the learning rate η as well as momentum α as given in Eq. 17. η and α indicate the stability of the overall network. If we change η to a smaller

value, the network becomes stable and convergence is slow. On the other hand, for larger η, the system becomes unstable with faster convergence [48].

$$\delta_k = (T_k - Z_k) f'(Z'_k) \tag{16}$$

$$w'_{jk}(T+1) = w'_{jk}(T) + \eta \delta_k y_j + \alpha \left(w'_{jk}(T) w'_{jk}(T-1) \right) \tag{17}$$

Equations given in Eqs. 18a and 18b are used for computation of updated weights and biases from hidden layer to output layer for BP in Eq. 17 as well as the calculation of hidden layer gradient error for Eq. 19 [50].

$$\Delta w'_{jk} = \eta \delta_k y_j \tag{18a}$$

$$\Delta b'_k = \eta \delta_k \tag{18b}$$

$$\delta_j = \sum_{k=1}^{K} \delta_k w'_{jk} f'(Z'_k) \tag{19}$$

w_{ij} and b_j are updated for input to hidden layer with the help of following Eqs. 20a and 20b. So it is deduced that the BP Eq. 17 is a generalized form, which is used for updating the input to the hidden layer and hidden to output layer weights and bias.

$$\Delta w_{ij} = \eta \delta_j x_i \tag{20a}$$

$$\Delta b_j = \eta \delta_j \tag{20b}$$

The proposed algorithm summarizes the steps which are followed by us. After initialization of training samples and testing samples. We set the number of testing and training samples in the overall data. In the third step, random initialization of weights and biases is done after that in the fourth step number of epochs are selected. The algorithm consists of two portions FF and BP, the FF pass is performed by using Eqs. 10 - 14. BP pass is performed by Eq. 17 and weights and baises are updated with the help of Eqs. 18a - 20b. This whole process from steps 5-7 will be repeated until error e_k converges.

Algorithm 1: Proposed ANN based energy prediction algorithm

Input: Training Data, weights, biases

Initialization of Training and Testing data.

Initialization of random weights and biases.

while e_k *converges to desired value* **do**

    Set number of epochs

    Ouput Computation using FF given by Eqs. 10 - 14.

    Calculation of e_k with the help of Eq.15.

    Implementation of BP Eq. 17 using Eqs. 18a - 20b.

end

Testing of training ANN on given data

5. Results and Discussion

5.1. Simulation Parameters

The input parameters as explained in Section 4.2 are the number of tasks and size of tasks given on a particular time slot $t = 1, 2, 3....$ k_i i.e. number of tasks is following the Poisson distribution with an average rate of λ, we have assumed that an FNs at a particular interval can serve (50-100) requests. To make a fog environment as realistic as possible, the task size is taken as the sum of individual requests per time slot. For instance, at a particular slot $t = 1$ there could be a possibility that an FNs can serve tasks with a different variation like fewer tasks having larger size or excess of requests with smaller size or combination of both. Therefore the task size assumed here is in the range of (0.5Mb - 16Mb).

The energy consumption E_m at FNs depends on l_m task size, η_m CPU cycles and energy consumption per CPU cycle θ_m. These internal parameters values have different ranges, η_m is ranging from (200-2000) cycle/bit and value of θ_m is taken as (10^{-10}) J/cycle. Noticing the smaller values of the internal parameters the energy consumption is measured in mJ. Before moving forward to performance analysis of the proposed algorithm, we define the different evaluation

procedures applied [50]. n is the total number of observations and X_i and X_i' are actual and predicted data respectively.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(X_i - X_i')^2} \tag{21}$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|X_i - X_i'|}{X_i} \times 100 \tag{22}$$

$$MABE = \sum_{i=1}^{n}\frac{|X_i - X_i'|}{n} \tag{23}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(X_i - X_i')^2}{\sum_{i=1}^{n} X_i} \tag{24}$$

The first evaluation approach used is Root mean square error (RMSE), also called statistical error, where the value of RMSE is positive all the time and zero for an ideal case. Next is the mean absolute percentage error (MAPE) which is used for measuring the accuracy of the model and a model will be considered highly accurate with a minimum MAPE value. Mean absolute bias error (MABE) another standard employed for checking the accuracy of measured and predicted values. The last is the R^2 coefficient of determination, the range of R^2 is from 0-1 and it tells us about how the variation of one variable is related to the other or goodness of fit.

5.2. Performance of RLS

The first prediction technique applied is adaptive RLS filter, like explain in Section 4.2 inputs (No. of task & task size) are fed into adoptive RLS, and prediction graphs are shown in Figures (3-5). The number of testing samples is on the x-axis whereas the y-axis shows the energy consumption in mJ. We tested the RLS filter with 5000 samples and the variation of energy prediction is noted on one, three, and five seconds respectively. Figure 3 shows the prediction of

one second ahead and as prediction time is changed from three to five seconds as depicted in Figures (4 & 5) respectively, the error results are degrading as well. A point should be added here that the trend of all graphs on 1-1000 sample is more disrupted because at the initial point the model's weights are not fully trained so as the model becomes more trained with these adoptive weights the behavior of error results is much better as we can see in from 2000-5000 samples.

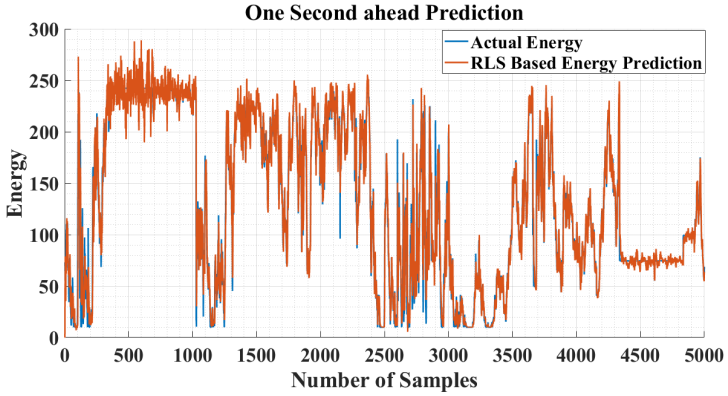

Figure 3: 1 Second ahead prediction based on RLS filter

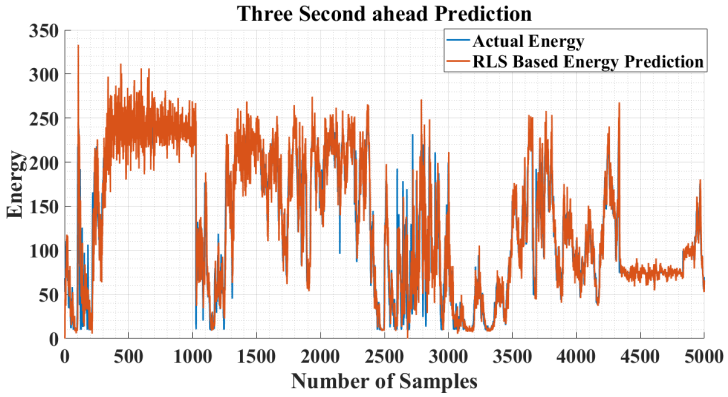

Figure 4: 3 Second ahead prediction based on RLS filter

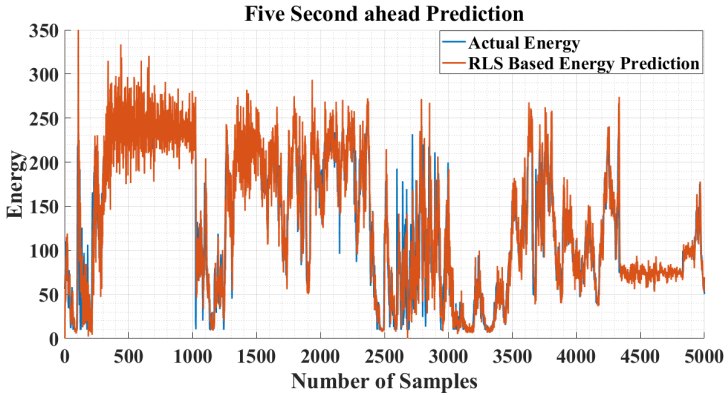

Figure 5: 5 Second ahead prediction based on RLS filter

5.3. Performance of ANN

5.3.1. Training Phase for ANN

The ANN algorithm implemented here is BP with learning rate and momentum. Comprehensive and detailed simulations are carried out for calculations of errors as shown in the latter part of this subsection. Selection of a suitable data set is one of the challenging tasks, along with the internal specifications of ANN algorithm like appropriate values for learning rate, number of neurons in the hidden layers. To do so ANN is applied to the set of changing factors, we have constructed three sets of models here according to the increasing number of input samples for training purposes. The Table (3) is associated with $20,000$ samples, the second is $50,000$ and the third has $100,000$ training samples. In the first table, the accuracy and error scores are generated for learning rates 0.01, 0.05, and 0.1 and we kept the fixed value of momentum i.e. 0.9 to get the best results. Another specification is selecting the number of hidden layers for the best performance of the model, hence the number of hidden layers is varied for individual learning rates. A similar pattern is followed for the Table 4 and Table 5 of $50,000$ and $100,000$ training samples respectively. By using the Eqs. 21 - 24 RMSE, MABE, MAPE, AND R^2 is calculated for training of data. The best model is recognized with the help of the above-mentioned performance evaluation parameters. .

Let's have a discussion on the accuracy and error scores of Model:1 as given in Table (3), the samples we used for training purposes are 20,000. These samples are counter to each learning rate and then each learning rate is subjected to a different number of hidden layer neurons.

Different error scores are mentioned in sub-section 5.1, as the hidden layers are differ from [10 5 1],[20 10 1] and [20 20 1] while keeping the learning rate 0.1 and momentum 0.9, the behaviors of $RMSE$, $MABE$, $MAPE$ and R^2 error scores also change. Similarly for learning rates of 0.05 and 0.01 same trend is followed with minor abnormalities. At last, based on these scores the best model is ranked, so the rank 1 model is with hidden layer **[20 20 1]** and learning rate 0.01 and R^2 =0.9796 and $RMSE = 0.8432$ as given in Table (3). The model just above our first ranked is very closed, given with hidden layer [20 10 1] which has $R^2 = 9800$ but when we compare $RMSE$ errors of both model its value is larger as compared to first ranked model.

Table 3: Model 1 Training error with 20,000 samples

Learning Rate = 0.1 Samples = 20000				
Layers	**RMSE**	**MABE**	**MAPE**	**R^2**
[10 5 1]	1.2435	0.8133	13.8944	0.9578
[20 10 1]	1.0324	0.6657	13.5498	0.9709
[20 20 1]	0.8829	0.5415	12.1813	0.9787
Learning Rate = 0.05				
Layers	**RMSE**	**MABE**	**MAPE**	**R^2**
[10 5 1]	0.9271	0.5765	12.6247	0.9765
[20 10 1]	0.9294	0.5716	11.9687	0.9764
[20 20 1]	0.8622	0.5129	11.3821	0.9797
Learning Rate = 0.01				
Layers	**RMSE**	**MABE**	**MAPE**	**R^2**
[10 5 1]	0.8739	0.5259	12.2456	0.9792
[20 10 1]	0.8548	0.5145	11.4708	0.9800
[20 20 1]	0.8432	0.5123	10.8221	0.9796

Further moving onto Model:2 in the Table (4), here 50,000 training samples are passed through ANN with different learning rates, and their respective error results are generated. After analyzing and comparing different error scores we

have ranked the second last Model with specifications of learning rate= 0.01, hidden layers= **[20 10 1]** and error scores of the model given by $R^2 = 0.9842$ and $RMSE = 0.8610$ is best among all the errors given in Table 4 by 50,000 samples.

Table 4: Model 2 Training error scores with 50,000 samples

Learning Rate = 0.1 Samples = 50000				
Layers	**RMSE**	**MABE**	**MAPE**	**R^2**
[10 5 1]	1.0531	0.6886	12.2191	0.9765
[20 10 1]	0.9660	0.5847	10.6278	0.9802
[20 20 1]	0.8611	0.4858	9.6795	0.9842
Learning Rate = 0.05				
Layers	**RMSE**	**MABE**	**MAPE**	**R^2**
[10 5 1]	0.9191	0.5577	10.4641	0.9821
[20 10 1]	0.8711	0.4887	8.4690	0.9839
[20 20 1]	0.8649	0.4893	9.1647	0.9841
Learning Rate = 0.01				
Layers	**RMSE**	**MABE**	**MAPE**	**R^2**
[10 5 1]	0.8862	0.5205	10.7573	0.9833
[20 10 1]	0.8610	0.4856	9.5153	0.9842
[20 20 1]	0.8826	0.5023	8.8906	0.9835

Table 5 gives a demonstration of the model with 100,000 training samples, and on the indication of various values of errors, we select the best one, which is with **[20 10 1]** hidden layers, and learning rate 0.01. The values of RMSE= 0.8838 and $R^2 = 0.9856$ score are best among all the other models in 100,000 samples. After the selection of the best individual ranking of models of 20,000, 50,000, and 100,000 respectively. These selected models are then put through the testing phase and at the end, the model with 100,000 samples gives us the most accurate results.

We have selected the hidden layers [10 5 1], [20 10 1], and [20 20 1] while performing a series of simulations on training data. The same is the case for learning rates which are adjusted along with momentum values. Every factor is important so far in the training phase after the thorough analysis of the error scores of each table, we concluded that as the number of input samples is

Table 5: Model 3 Training error scores with 100,000 samples

Learning Rate = 0.1 Samples = 100000				
Layers	**RMSE**	**MABE**	**MAPE**	**Rˆ2**
[**10 5 1**]	0.9025	0.5114	9.2797	0.9850
[**20 10 1**]	0.8990	0.4982	8.9407	0.9852
[**20 20 1**]	0.903	0.5581	13.9615	0.9850
Learning Rate = 0.05				
Layers	**RMSE**	**MABE**	**MAPE**	**Rˆ2**
[**10 5 1**]	0.9012	0.5129	9.8034	0.9851
[**20 10 1**]	0.8887	0.5024	9.5128	0.9855
[**20 20 1**]	0.9049	0.5218	9.5265	0.9849
Learning Rate = 0.01				
Layers	**RMSE**	**MABE**	**MAPE**	**Rˆ2**
[**10 5 1**]	0.8948	0.5180	10.9364	0.9853
[**20 10 1**]	0.8838	0.4842	8.5535	0.9856
[**20 20 1**]	0.8968	0.5206	10.6453	0.9852

increasing the accuracy score of the model is also improving but with changing hidden layers and learning rate.

5.3.2. Testing Phase for ANN

After doing the comprehensive analysis of each training set and selecting the best model in the training phase, now in the testing phase the best model is subjected to testing samples. Energy consumption prediction for fog environment on a one second, three seconds, and five seconds is performed. One second ahead energy prediction is given in Figure. 6. On the x-axis, we have 5000 testing samples and on the y-axis energy in millijoule (mJ). Actual and ANN-based predicted energy is differentiated by different colors. The behavior of the graph emphasizes that the model is well trained and upon testing up to 100,000 samples error results are satisfactory and show a good resemblance between predicted and actual energy values.

Now as explained earlier in this section to make the fog environment as real as possible we have generated the number of requests with Poisson distribution and along with that size of the task is also generated randomly. Moreover, the

random inclination of energy is justified by our input data. The prediction on three seconds and five seconds is in the given Figures. 7 and 8 respectively. These prediction graphs follow a similar trend but moving toward a larger prediction time the results are also degrading which is very natural.

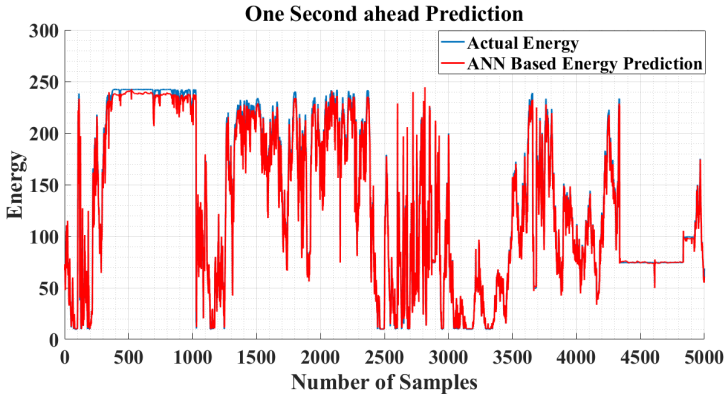

Figure 6: 1 Second ahead prediction based on ANN

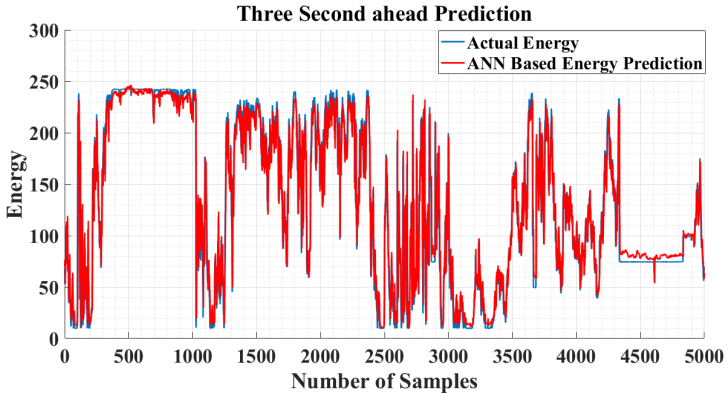

Figure 7: 3 Second ahead prediction based on ANN

5.4. Comparison between RLS and ANN

A comparison between ANN and RLS is given in Table (6). Accuracy scores are generated for both techniques for one second, three seconds, and five seconds respectively for the same number of samples. For one-second energy prediction,

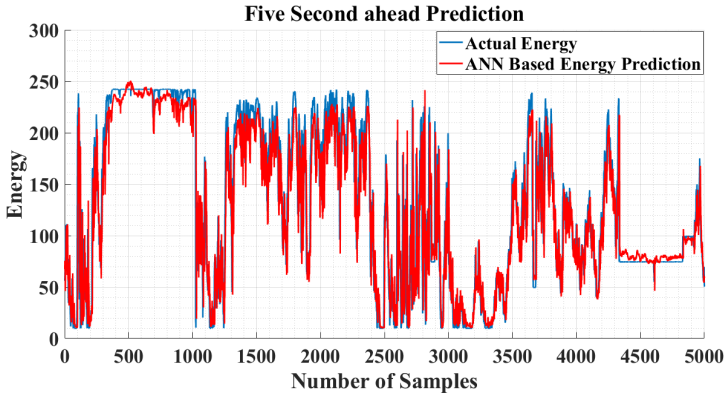

Figure 8: 5 Second ahead prediction based on ANN

the accuracy of ANN is better than RLS. Each individual score of ANN (RMSE, MABE, R^2 and MAPE) is better than RLS. Similarly for three seconds and five seconds prediction accuracy values of both approach increases (in a degrading manner) but following the same trend as in one-second testing. A major limitation of RLS is that it only considers time series data for prediction. On the other hand, ANN uses inputs such as the number of tasks and task size to predict energy consumption at fog nodes.

Table 6: Comparison of ANN and RLS Techniques

	1 Second Prediction		**3 Second Prediction**		**5 Second Prediction**	
Errors	**ANN**	**RLS Filter**	**ANN**	**RLS Filter**	**ANN**	**RLS Filter**
RMSE	1.2144	1.5331	2.2773	2.5562	2.9593	2.9647
MABE	0.6870	0.9353	1.3758	1.6587	1.8961	1.9921
R^2	0.9752	0.9604	0.9128	0.8901	0.8527	0.8522
MAPE	7.7247	12.1923	20.63	22.4640	25.9989	26.8050

6. Conclusion

In this paper, we present two energy prediction techniques for fog computing-based IoT networks. The first technique uses the RLS algorithm to predict the energy consumption at different fog nodes. The second technique relies on the

ANN algorithm and uses the number of tasks and size of tasks to train an optimal learning model. Simulation results show that the ANN-based energy prediction technique outperforms RLS based technique by 20% in terms of root mean square error. In the future, we aim to use the proposed energy prediction to develop efficient energy-aware task offloading algorithms.

References

[1] M. Rahim, M. A. Javed, A. N. Alvi, M. Imran, An efficient caching policy for content retrieval in autonomous connected vehicles, Transportation Research Part A: Policy and Practice 140 (2020) 142 – 152.

[2] M. Giordani, M. Polese, M. Mezzavilla, S. Rangan, M. Zorzi, Toward 6g networks: Use cases and technologies, IEEE Communications Magazine 58 (3) (2020) 55–61.

[3] W. Saad, M. Bennis, M. Chen, A vision of 6g wireless systems: Applications, trends, technologies, and open research problems, IEEE Network 34 (3) (2020) 134–142.

[4] S. Zhang, J. Liu, H. Guo, M. Qi, N. Kato, Envisioning device-to-device communications in 6g, IEEE Network 34 (3) (2020) 86–91.

[5] H. G. Moussa, W. Zhuang, Energy- and delay-aware two-hop noma-enabled massive cellular iot communications, IEEE Internet of Things Journal 7 (1) (2020) 558–569.

[6] G. Hattab, D. Cabric, Distributed wideband sensing-based architecture for unlicensed massive iot communications, IEEE Transactions on Cognitive Communications and Networking 5 (3) (2019) 819–834.

[7] M. Rahim, S. Ali, A. N. Alvi, M. A. Javed, M. Imran, M. A. Azad, D. Chen, An intelligent content caching protocol for connected vehicles, Transactions on Emerging Telecommunications Technologies 32 (4) (2021) e4231.

[8] Z. Sun, Z. Wei, N. Yang, X. Zhou, Two-tier communication for uav-enabled massive iot systems: Performance analysis and joint design of trajectory and resource allocation, IEEE Journal on Selected Areas in Communications (2020) 1–1.

[9] J. Liu, M. Agiwal, M. Qu, H. Jin, Online control of preamble groups with priority in massive iot networks, IEEE Journal on Selected Areas in Communications (2020) 1–1.

[10] J. Cao, P. Yu, M. Ma, W. Gao, Fast authentication and data transfer scheme for massive nb-iot devices in 3gpp 5g network, IEEE Internet of Things Journal 6 (2) (2019) 1561–1575.

[11] R. B. Sørensen, J. J. Nielsen, P. Popovski, Machine learning methods for monitoring of quasiperiodic traffic in massive iot networks, IEEE Internet of Things Journal 7 (8) (2020) 7368–7376.

[12] S. Verma, Y. Kawamoto, Z. M. Fadlullah, H. Nishiyama, N. Kato, A survey on network methodologies for real-time analytics of massive iot data and open research issues, IEEE Communications Surveys Tutorials 19 (3) (2017) 1457–1477.

[13] W. Rafique, L. Qi, I. Yaqoob, M. Imran, R. U. Rasool, W. Dou, Complementing iot services through software defined networking and edge computing: A comprehensive survey, IEEE Communications Surveys Tutorials 22 (3) (2020) 1761–1804.

[14] M. A. Javed, S. Zeadally, AI-Empowered Mobile Edge Computing in the Internet of Vehicles, IEEE Network"to appear".

[15] A. H. Sodhro, G. H. Sodhro, M. Guizani, S. Pirbhulal, A. Boukerche, Ai-enabled reliable channel modeling architecture for fog computing vehicular networks, IEEE Wireless Communications 27 (2) (2020) 14–21.

[16] P. Zhou, K. Shen, N. Kumar, Y. Zhang, M. M. Hassan, K. Hwang, Communication-efficient offloading for mobile edge computing in 5g heterogeneous networks, IEEE Internet of Things Journal (2020) 1–1.

[17] N. C. Luong, Y. Jiao, P. Wang, D. Niyato, D. I. Kim, Z. Han, A machine-learning-based auction for resource trading in fog computing, IEEE Communications Magazine 58 (3) (2020) 82–88.

[18] X. Lyu, C. Ren, W. Ni, H. Tian, R. P. Liu, Cooperative computing anytime, anywhere: Ubiquitous fog services, IEEE Wireless Communications 27 (1) (2020) 162–169.

[19] D. Wu, N. Ansari, A cooperative computing strategy for blockchain-secured fog computing, IEEE Internet of Things Journal 7 (7) (2020) 6603–6609.

[20] J. Liang, M. Zhang, V. C. M. Leung, A reliable trust computing mechanism based on multisource feedback and fog computing in social sensor cloud, IEEE Internet of Things Journal 7 (6) (2020) 5481–5490.

[21] C. Lin, G. Han, X. Qi, M. Guizani, L. Shu, A distributed mobile fog computing scheme for mobile delay-sensitive applications in sdn-enabled vehicular networks, IEEE Transactions on Vehicular Technology 69 (5) (2020) 5481–5493.

[22] M. Mukherjee, S. Kumar, C. X. Mavromoustakis, G. Mastorakis, R. Matam, V. Kumar, Q. Zhang, Latency-driven parallel task data offloading in fog computing networks for industrial applications, IEEE Transactions on Industrial Informatics 16 (9) (2020) 6050–6058.

[23] S. Luo, X. Chen, Z. Zhou, X. Chen, W. Wu, Incentive-aware micro computing cluster formation for cooperative fog computing, IEEE Transactions on Wireless Communications 19 (4) (2020) 2643–2657.

[24] P. Roy, S. Sarker, M. A. Razzaque, M. M. Hassan, S. A. AlQahtani, G. Aloi, G. Fortino, Ai-enabled mobile multimedia service instance place-

ment scheme in mobile edge computing, Computer Networks 182 (2020) 107573.

[25] K. Tange, M. De Donno, X. Fafoutis, N. Dragoni, A systematic survey of industrial internet of things security: Requirements and fog computing opportunities, IEEE Communications Surveys Tutorials (2020) 1–1.

[26] M. Adhikari, M. Mukherjee, S. N. Srirama, Dpto: A deadline and priority-aware task offloading in fog computing framework leveraging multilevel feedback queueing, IEEE Internet of Things Journal 7 (7) (2020) 5773–5782.

[27] S. Khan, A. N. Alvi, M. A. Javed, Y. D. Al-Otaibi, A. K. Bashir, An efficient medium access control protocol for rf energy harvesting based iot devices, Computer Communications 171 (2021) 28–38.

[28] T. Huang, W. Yang, J. Wu, J. Ma, X. Zhang, D. Zhang, A survey on green 6g network: Architecture and technologies, IEEE Access 7 (2019) 175758–175768.

[29] U. Gustavsson, P. Frenger, C. Fager, T. Eriksson, H. Zirath, F. Dielacher, C. Studer, A. Pärssinen, R. Correia, J. N. Matos, D. Belo, N. B. Carvalho, Implementation challenges and opportunities in beyond-5g and 6g communication, IEEE Journal of Microwaves 1 (1) (2021) 86–100.

[30] X. Ma, Z. Chen, W. Chen, Y. Chi, Z. Li, C. Han, Q. Wen, Intelligent reflecting surface enhanced indoor terahertz communication systems, Nano Communication Networks 24 (2020) 100284.

[31] M. Polese, J. M. Jornet, T. Melodia, M. Zorzi, Toward end-to-end, full-stack 6g terahertz networks, IEEE Communications Magazine 58 (11) (2020) 48–54.

[32] K. Electronics, Wzzard - advantech/b+b smartworx battery operated iot mesh solution.

URL http://www.ksr-elec.com/iot-and-cloud-solutions/wzzard-advantech-b-b-smartworx-battery-operated-iot-solution.html

[33] U. M. Malik, M. A. Javed, S. Zeadally, S. u. Islam, Energy efficient fog computing for 6G enabled massive IoT: Recent trends and future opportunities, IEEE Internet of Things Journal"to appear".

[34] T. Li, S. Fong, X. Li, Z. Lu, A. H. Gandomi, Swarm decision table and ensemble search methods in fog computing environment: Case of day-ahead prediction of building energy demands using iot sensors, IEEE Internet of Things Journal 7 (3) (2020) 2321–2342.

[35] S. Kosunalp, An energy prediction algorithm for wind-powered wireless sensor networks with energy harvesting, Energy 139 (2017) 1275–1280.

[36] D.-G. Zhang, L. Chen, J. Zhang, J. Chen, T. Zhang, Y.-M. Tang, J.-N. Qiu, A multi-path routing protocol based on link lifetime and energy consumption prediction for mobile edge computing, IEEE Access 8 (2020) 69058–69071.

[37] F. Azmat, Y. Chen, N. Stocks, Predictive modelling of rf energy for wireless powered communications, IEEE Communications Letters 20 (1) (2016) 173–176.

[38] Z. Zhou, Z. Wang, H. Yu, H. Liao, S. Mumtaz, L. Oliveira, V. Frascolla, Learning-based urllc-aware task offloading for internet of health things, IEEE Journal on Selected Areas in Communications 39 (2) (2021) 396–410.

[39] U. Saleem, Y. Liu, S. Jangsher, Y. Li, T. Jiang, Mobility-aware joint task scheduling and resource allocation for cooperative mobile edge computing, IEEE Transactions on Wireless Communications 20 (1) (2021) 360–374.

[40] L. Ale, N. Zhang, X. Fang, X. Chen, S. Wu, L. Li, Delay-aware and energy-efficient computation offloading in mobile edge computing using deep rein-

forcement learning, IEEE Transactions on Cognitive Communications and Networking (2021) 1–1.

[41] L. Dong, W. Wu, Q. Guo, M. N. Satpute, T. Znati, D. Z. Du, Reliability-aware offloading and allocation in multilevel edge computing system, IEEE Transactions on Reliability 70 (1) (2021) 200–211.

[42] A. Yousafzai, I. Yaqoob, M. Imran, A. Gani, R. Md Noor, Process migration-based computational offloading framework for iot-supported mobile edge/cloud computing, IEEE Internet of Things Journal 7 (5) (2020) 4171–4182. doi:10.1109/JIOT.2019.2943176.

[43] B. Chen, J. Wan, Y. Lan, M. Imran, D. Li, N. Guizani, Improving cognitive ability of edge intelligent iiot through machine learning, IEEE Network 33 (5) (2019) 61–67. doi:10.1109/MNET.001.1800505.

[44] M. H. ur Rehman, C. S. Liew, T. Y. Wah, M. Imran, K. Salah, N. Nasser, D. Svetinovic, Device-centric adaptive data stream management and offloading for analytics applications in future internet architectures, Future Generation Computer Systems 114 (2021) 155–168. doi:https://doi.org/10.1016/j.future.2020.07.054.

[45] A. Karimiafshar, M. R. Hashemi, M. R. Heidarpour, A. N. Toosi, Effective utilization of renewable energy sources in fog computing environment via frequency and modulation level scaling, IEEE Internet of Things Journal 7 (11) (2020) 10912–10921.

[46] G. Zhang, F. Shen, Z. Liu, Y. Yang, K. Wang, M.-T. Zhou, Femto: Fair and energy-minimized task offloading for fog-enabled iot networks, IEEE Internet of Things Journal 6 (3) (2019) 4388–4400.

[47] R. Martinek, J. Rzidky, R. Jaros, P. Bilik, M. Ladrova, Least mean squares and recursive least squares algorithms for total harmonic distortion reduction using shunt active power filter control, Energies 12 (8).

[48] M. Chen, U. Challita, W. Saad, C. Yin, M. Debbah, Artificial neural networks-based machine learning for wireless networks: A tutorial, IEEE Communications Surveys Tutorials 21 (4) (2019) 3039–3071.

[49] S. Chen, Y. Zheng, W. Lu, V. Varadarajan, K. Wang, Energy-optimal dynamic computation offloading for industrial iot in fog computing, IEEE Transactions on Green Communications and Networking 4 (2) (2020) 566–576.

[50] Z. E. Mohamed, Using the artificial neural networks for prediction and validating solar radiation, Journal of the Egyptian Mathematical Society 26 (47) (2019) 1–13.