Multimedia Tools and Applications (2022) 81:8235–8258
https://doi.org/10.1007/s11042-022-12223-5

Effective prediction and resource allocation method (EPRAM) in fog computing environment for smart healthcare system

Fatma M. Talaat[1]

Received: 26 September 2021 / Revised: 30 December 2021 / Accepted: 10 January 2022 /
Published online: 2 February 2022

Abstract
Recently, many concepts in technology has been changed. According to the digital transformation trends, Internet of Things (IoT) represents an interested research issue. As the IoT grows, the data and the processes will need more space. The data in cases like healthcare, smart cities, autonomous vehicles, smart agriculture, etc. needs to be analyzed and processed in real-time. Cisco refers to the dependence of edge and cloud as "The Fog". The data can be analyzed at the fog layer to maximize data utilization. This paper presents a new Effective Prediction and Resource Allocation Methodology (EPRAM) for Fog environment, which is suitable for Healthcare applications. Resource Allocation (RA) represents a hard mission as it involves a set of various resources and fog nodes to achieve the required computations for IoT systems. EPRAM tries to achieve effective resource management in Fog environment via real-time resource allocating as well as prediction algorithm. EPRAM is composed of three main modules, namely: (i) Data Preprocessing Module (DPM), (ii) Resource Allocation Module (RAM) and (ii) Effective Prediction Module (EPM). The EPM uses the PNN to predict a target field, using one or more predictors. In order to detect the probability of the heart attack, PNN is trained using the training dataset. Then PNN will be tested using the user's sensing data coming from the IoT layer to predict the probability of heart attack and then take the most appropriate action accordingly. The main goal of the system is to achieve a low latency while improving the Quality of Service (QoS) metrics such as (the allocation cost, the response time, bandwidth efficiency and energy consumption). Unlike other RA techniques, EPRAM employs deep Reinforcement Learning (RL) algorithm in a new manner. It also uses the PNN for the prediction algorithm. It has achieved such acceptable performance due to using deep RL and PNN. Deep RL has shown impressive promises in resource allocation. PNN generates accurate predicted target and is much faster than multilayer perceptron networks. Comparing the EPRAM with the state-of-the-art algorithms, EPRAM achieved the minimum Makespan as compared to previous LB algorithms,

✉ Fatma M. Talaat
fatma.m.talaat@gmail.com

[1] Faculty of Artificial Intelligence, Kafrelsheikh University, Kafrelsheikh, Egypt

while maximizing the Average Resource Utilization (ARU) and the Load Balancing Level (LBL). Accordingly, EPRAM is a suitable algorithm in the case of real-time systems in FC which leads to load balancing. ERAM is effective in monitoring and predicting the status of the patient accurately and quickly.

Keywords Resource Allocation · Internet of Things (IoT) · Fog Computing (FC) · Digital Technology · Digital Transformation · Deep Learning (DL) · smart Healthcare System · Prediction · Reinforcement Learning · Mobile HEALTH Dataset

1 Introduction

Depending on the phenomenal development of the digital technology in recent years, the Internet of Things (IoT) has a great impact in our lives [16]. IoT can be considered as a set of devices equipped with sensors, actuators, and processors that communicate with each other to serve a reliable objective. IoT innovates an integrated system that combines different systems to provide intelligent performances in each task. It has created a new growth of cell phones, home and other embedded applications that are all connected to the internet. Consequently, different IoT-enabled systems such as smart healthcare, smart city, smart home, smart factory, smart transport and smart agriculture are getting significant attention across the world. Cloud Computing (CC) is considered as the standard infrastructure, platform and services to develop IoT systems [11]. However, Cloud datacenters locate at remote distance from IoT devices which leads to high latency. This issue adversely affects the response time for real time applications such as critical health monitoring systems, traffic monitoring, and emergency fire. In addition, IoT sources are geographically extended and can generate a large amount of data sent to Cloud for processing which lead to overloading. The edge computational resources can handle the previous mentioned challenges in IoT systems [21]. Fog Computing (FC) can be defined as edge computing that aims to deploy services at the edge network. The FC achieves the location-awareness and reduces the latency [2].

In FC, there are various devices such as Cisco IOx networking equipment, micro-datacenter, Nano-server, smart phone, personal computer and Cloudlets. The fog device is usually named as Fog Node (FN). These Fog Nodes create a generous distribution of services to process IoT-data near to the source. FC and CC paradigms most often work in an integrated manner as shown in the Fig. 1 to meet Quality of Service (QoS) and resource requirements of wide standard IoT systems [2]. In FC, Resource Allocation (RA) represents a hard mission as it involves a set of various resources and fog nodes to achieve the required computations for IoT systems [38]. Many previous RA methods have been proposed such as Least Connection (LC), Round Robin (RR), Weighted Round Robin (WRR), and Adaptive Weighted Round Robin (AWRR). These algorithms will illustrated in details in section 2 (the related work section). However, they have many disadvantages such as; (i) they do not take into account the heterogeneity of the computational resources, (ii) low performance due to process migration. (iii) High latency. (iv) The lack of unified standard for FC [1]. The integration between Fog, Cloud, and various IoT devices increases the difficulty of RA issue [22]. The real-world implementation of FC environment with IoT and Cloud datacenters is very expensive and the modification is tiresome issue so the solution is the empirical analysis using simulation. There are a certain number of simulators like Edgecloudsim [28], SimpleIoTSimulator [25] and iFogSim [12] for modelling FC environment.

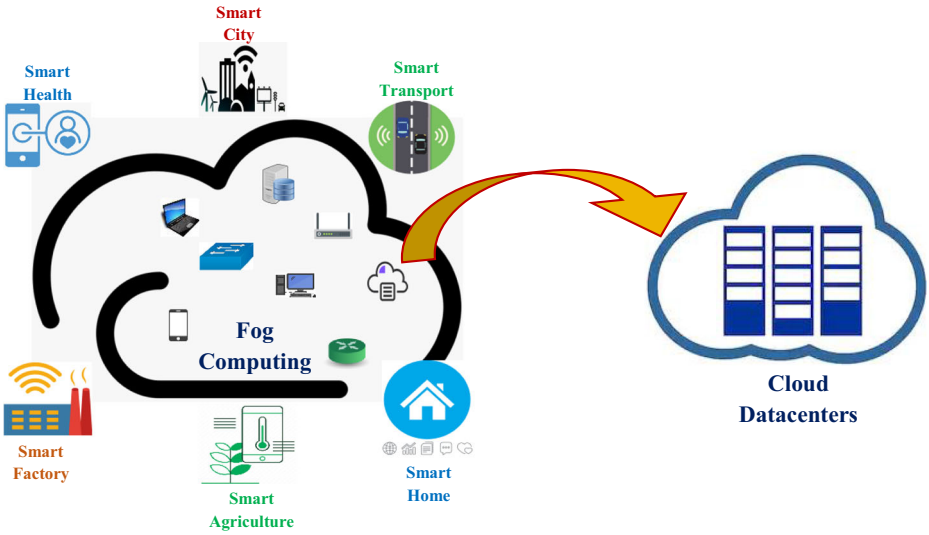

Fig. 1 Interactions among IoT-enabled systems, Fog and Cloud Computing

The rest of this section introduces some concepts in the field of stratified sampling, resource management, Deep Learning (DL) algorithm, Deep Neural Network (DNN), Probabilistic Neural Networks (PNN), and Reinforcement Learning (RL).

Figure 1 shows the relation between IoT, big dta, cloud, and fog. It also illustrates some examples of IoT applications.

1.1 Stratified sampling

In stratified sampling, the data is first divided into subgroups (or strata) with same probability. For example, you can stratify by gender to ensure that men and women are sampled in equal proportions, or by region to ensure that each region within an urban population is represented. It is also possible to specify a different sample size for each stratum. For example, when you want to sample for a survey, you can give men a higher probability of being sampled than women when you think that the willingness to participate in the survey is lower for men than for women. It is used when we might reasonably expect the measurement of interest to vary between the different subgroups, and we want to ensure representation from all the subgroups. For example, in a study of stroke outcomes, we may stratify the population by gender, to ensure equal representation of men and women.

The study sample is then obtained by taking equal sample sizes from each stratum. In stratified sampling, it may also be appropriate to choose non-equal sample sizes from each stratum. For example, in a study of the health outcomes of nursing staff in a county, if there are three hospitals each with different numbers of nursing staff (hospital A has 500 nurses, hospital B has 1000 and hospital C has 2000), then it would be appropriate to choose the sample numbers from each hospital proportionally (e.g. 10 from hospital A, 20 from hospital B and 40 from hospital C). This ensures a more realistic and accurate estimation of the health outcomes of nurses across the county, whereas simple random sampling would over-represent nurses from hospitals A and B. The fact that the sample was stratified should be taken into account at the analysis stage. Stratified sampling improves the accuracy and representativeness of the

results by reducing sampling bias. However, it requires knowledge of the appropriate characteristics of the sampling frame (the details of which are not always available), and it can be difficult to decide which characteristic(s) to stratify by.

1.2 Resource management

The resource management is divided into; (i) resource assignment, and (ii) task placement [14, 34]. Resource assignment defines the number of incoming processes and determines the resource to be allocated to each process. For example, a Spark application selects the resource to be allocated based on the available memory and the data size. Task placement involves the determination of the location for each incoming task. For example, Delay Scheduling algorithm [41] in Hadoop assigns tasks based on data locality.

1.3 Deep learning (DL)

Deep learning (DL) (also known as deep structured learning or hierarchical learning) is a branch of machine learning methods based on Artificial Neural Networks (ANN) [39]. DL is named deep as it uses the deep neural networks [4, 26]. It is applied in fields of object detection, classification, and speech recognition [6]. Unlike the traditional learning algorithms, the performance of DL algorithm increases with the increase of the amount of data as shown in Fig. 2.

1.4 Deep neural network (DNN)

A Deep Neural Network (DNN) is an ANN with multiple layers between the input and output layers [29]. It is constructed with a set of connected layers which are: (i) Input Layer, (ii) Output Layer, and (iii) Hidden Layers (all layers in between). It finds the correct mathematical method to convert the input into the output. The word "deep" means the network joins the neurons in more than two layers (hidden layers) [3] as shown in Fig. 3.

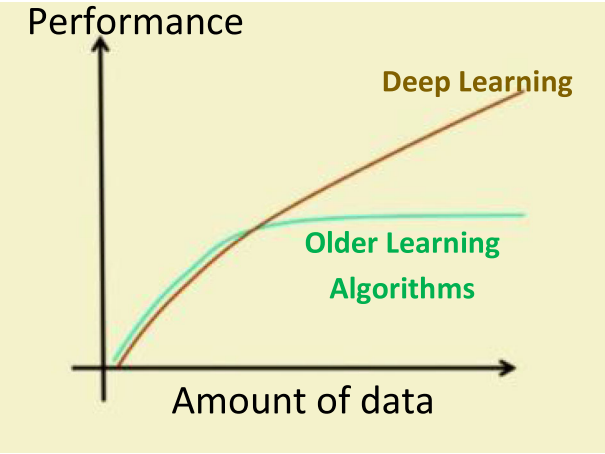

Fig. 2 Why deep learning

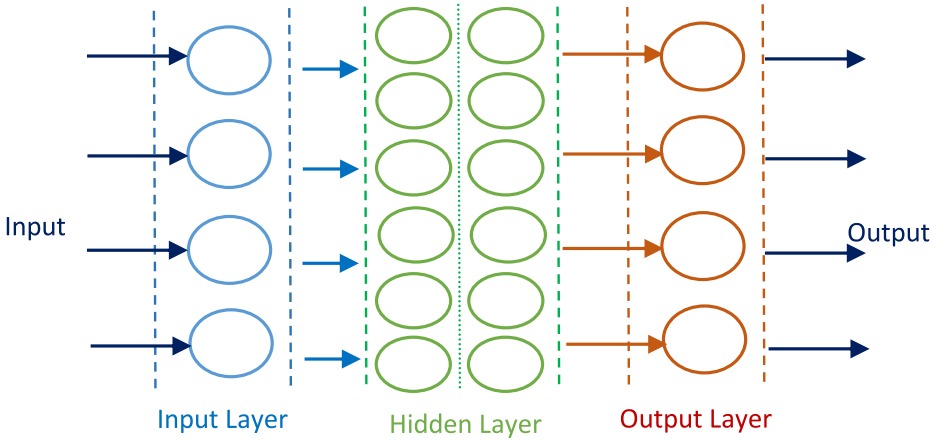

Fig. 3 DNN architecture

In DNN, each layer represents a deeper level of knowledge, i.e., the hierarchy of knowledge. At first, the DNN creates a map of virtual neurons and assigns random numerical values, or "weights", to connections between them. The weights and inputs are multiplied and return an output between 0 and 1. If the network did not accurately recognize a particular pattern, an algorithm would adjust the weights [15]. That way the algorithm can make certain parameters more influential, until it determines the correct mathematical manipulation to fully process the data.

1.5 Probabilistic neural networks (PNN)

Probabilistic Neural Networks (PNN) is organized into a multilayered feed forward network containing four layers [17, 35]: (i) Input layer: contains nodes with a set of measurements. (ii) Pattern layer: contains one neuron for each case in the training data set. It computes the Euclidean distance of the test case from the neuron's center point and then applies the Radial Basis Function (RBF) kernel function using the sigma values. (iii) Summation layer: performs a sum operation of the outputs from the second layer for each class. (iv) Output layer: takes all the outputs of the summation nodes and outputs the max, e.g. the label node that had the highest score. PNN has several advantages such as [17]: (i) PNNs are much faster than multilayer perceptron networks. (ii) PNN networks are insensitive to outliers. (iii) PNNs can be more accurate than multilayer perceptron networks. (iv) PNN networks generate accurate predicted target probability scores. (v) Guaranteed to converge to an optimal classifier as the size of the representative training set increases. In general, it is obvious that PNNs have high speed of learning and training. The main advantage of a PNN is its ability to output probabilities in multi-classification. However, PNN has disadvantages such as it requires more memory space to store the model. It requires a representative training set.

The originality of this paper is to introduce a new Effective Resource Allocation Methodology (ERAM) for Fog environment, which is suitable for Healthcare applications. ERAM tries to achieve effective resource management in Fog environment via real-time resource allocating as well as prediction algorithm. ERAM is composed of three main modules, namely: (i) Data Preprocessing Module (DPM), (ii) Resource Allocation Module (RAM) and (ii) Effective Prediction Module (EPM). The DPM is responsible for sampling, partitioning, and

Table 1 Related RA methods for Fog Computing and Cloud Computing

Year	Ref.	Implementation	Strength	Weakness
2020	[30]	Authors in [30] proposed a dynamic RA method based on Reinforcement learning and genetic algorithm. It observes the traffic in the network continuously, collects the information about each node, manages the incoming tasks, and distributes them between the available nodes equally using DRA method.	It is efficient in real-time systems in FC environment such as in the case of smart healthcare system. It is close to achieve perfection.	This method consumes more time for the process migration methodology. The process migration suffers from some issues such as; (i) complexity, (ii) Time, (iii) Inherit poor data quality.
2019	[7]	Dubey Sh et al. [7] used Round Robin (RR) to list the available nodes and assign the incoming processes equally to each node in order.	It is simple and easy to understand and implement.	If the servers have different processing capacities, one of them can become overloaded and crashed.
2019	[13]	Gupta S et al. [13] used Weighted Round Robin (WRR) for resource allocation process. It is similar to the RR in the cyclical assigning manner but it differs in that the node with the higher weight will be given the highest number of requests. In WRR, each server is allocated with a weight based on its capacity.	Able to send more requests to servers with higher capability and handling load.	Calculating the capacity is difficult in some cases such as in case with varying packet size.
2018	[27]	Singh G, Kaur K [27] used Least Connection (LC) to achieve the load balancing. It chooses the node with the least number of active transactions and modifies its data periodically depending number of connections.	It prevents the overloading.	Does not take the server capacity into account when determining number of current connections.
2018	[8]	Authors in [8] use the best Signal-to-Noise Ratio (SNR) and distributed alpha algorithm to measure the load on each node to achieve the load balancing.	It achieves the load balancing in FC environment.	It focuses on the delay of wireless communications.
2018	[36]	Wei Y et al. [36] use the Reinforcement Learning algorithm to find the best policy for scheduling in nonhomogeneous networks to maximize the energy efficiency.	It maximize the energy efficiency.	High Latency.
2017	[37]	Authors in [37] used Multi-agent Reinforcement Learning to increase the network resource utilization.	It increases the network resource utilization.	High Latency.
2017	[40]	Authors in [40] proposed a policy to minimize the latency of the IoT services. They aimed to assign each task according to the response time.	It assigns each task according to the response time and achieves LB.	The limitations of this method are: (i) It explores various scenarios in a distributed manner. (ii) It is not reasonable to decide whether to assign task to fog or to cloud depending on the processing time.

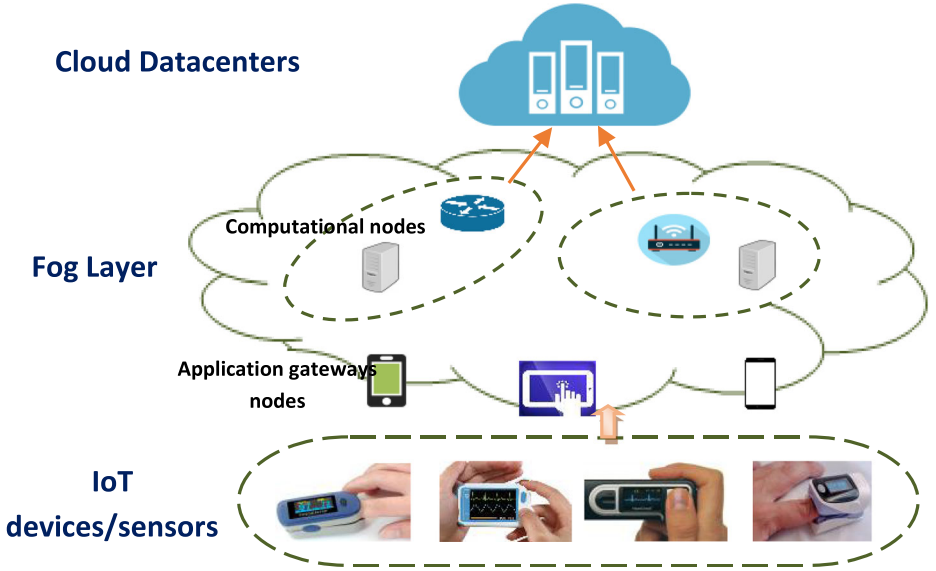

Fig. 4 Fog environment for IoT-enabled healthcare case study

balancing data to be in the appropriate form for analyzing and processing. The RAM learns to select the best FN to execute the incoming request. The RAM uses Reinforcement Learning (RL) algorithm to achieve a high LB for fog environment. The EPM uses the PNN to predict a target field, using one or more predictors. In order to detect the probability of the heart attack, PNN is trained using the training dataset. Then PNN will be tested using the user's sensing data coming from the IoT layer to predict the probability of heart attack and then take the most appropriate action accordingly.

The proposed IoT-Fog system consists of three layers, namely: (i) IoT Layer, (ii) Fog Layer, and (iii) Cloud Layer. The mission of the IoT layer is to monitor the patient symptoms. The fog layer is considered with handling the incoming requests and forwards them to the suitable server. The cloud layer is responsible for managing the transfer of data to and from the fog layer. The user data is sent to the most appropriate resource to be allocated and processed. The allocating resource is managed by a specific healthcare organization. The main goal of the system is to achieve a low latency while improving the Quality of Service (QoS) metrics such as the allocation cost, the response time, bandwidth efficiency and energy consumption. Unlike other RA techniques, EPRAM employs deep RL algorithm in a new manner. It also uses the PNN for the prediction algorithm. It has achieved such acceptable performance due to using deep RL and PNN. Deep RL has shown impressive promises in resource allocation. PNN generates accurate predicted target and is much faster than multilayer perceptron networks. Accordingly, EPRAM is a suitable algorithm in the case of real-time systems in FC which leads to load balancing.

The rest of paper is organized as follows: Section 2 introduces some of the recent previous efforts in the field of RA techniques generally. Then, it introduces the recent previous efforts in the area of RA algorithms for in CC and FC. Section 3 introduces a proposed Effective Prediction and Resource Allocation Methodology (EPRAM) for Fog environment using real-time resource allocating as well as PNN with more details about each contribution. Section 4 introduces the evaluation results and discussion. Our conclusion is discussed in Section 5.

Table 2 Server characteristics table (SCT)

FS_ID	Capacity	RAM	CPU_Usage	AW	Status
FS1	100 MB	7 MB	40 MHZ	1.2	Balanced
FS2	90 MB	8 MB	50 MHZ	1	Oveloaded
FS3	200 MB	5 MB	27 MHZ	2.5	Underloaded
FS4	240 MB	6 MB	50 MHZ	2	Underloaded
.	…	…	.	...	...
FSx	…	…	.	...	...

2 Related work

This section introduces some of the recent previous efforts in the field of RA techniques generally and in the area of RA algorithms for in CC and FC. In the last years, many researchers are working on different issues to solve the FC and IoT challenges. Fatma M. Talaat et al. [30] proposed a dynamic RA method based on Reinforcement learning and genetic algorithm. It observes the traffic in the network continuously, collects the information about each node, manages the incoming tasks, and distributes them between the available nodes equally using DRA method. It is efficient in real-time systems in FC environment such as in the case of smart healthcare system. Dubey Sh et al. [7] used Round Robin (RR) to list the available nodes and assign the incoming processes equally to each node in order. It is simple and easy to understand and implement. However, if the servers have different processing capacities, one of them can become overloaded and crashed.

Gupta S et al. [13] used Weighted Round Robin (WRR) for resource allocation process. It is similar to the RR in the cyclical assigning manner but it differs in that the node with the higher weight will be given the highest number of requests. In WRR, each server is allocated with a weight based on its capacity. Singh G, Kaur K [27] used Least Connection (LC) to

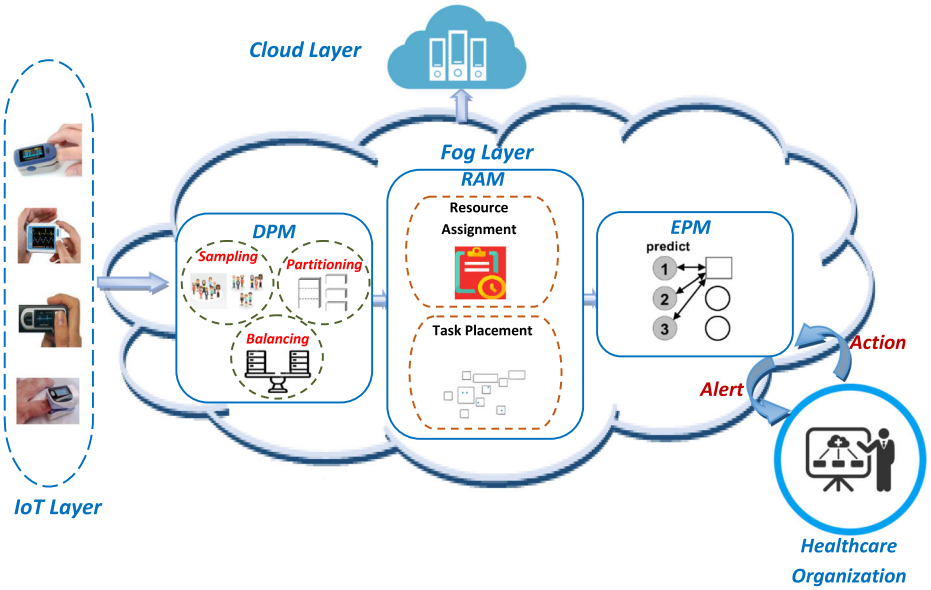

Fig. 5 Effective Resource Allocation Methodology (ERAM)

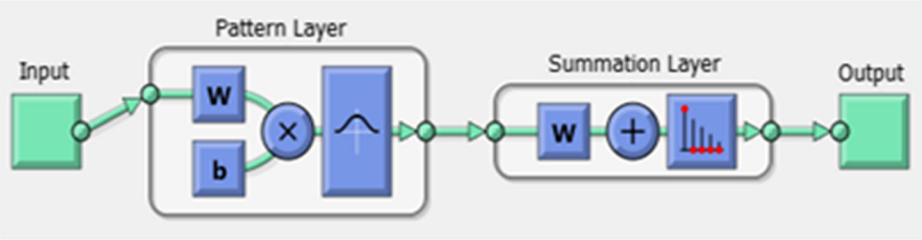

Fig. 6 PNN Layers

achieve the load balancing. It chooses the node with the least number of active transactions and modifies its data periodically depending number of connections.

Q. Fan et al. [8] proposed a model to minimize the communication and processing time by assigning the incoming process to the suitable source. Authors in [8] use the best Signal-to-Noise Ratio (SNR) and distributed alpha algorithm to measure the load on each node to achieve the load balancing. It reconstructs set of series of events and compares the SNR. However, the disadvantage of this method is that it focuses on the delay of wireless communications. Wei Y et al. [36] use the Reinforcement Learning algorithm to find the best policy for scheduling in nonhomogeneous networks to maximize the energy efficiency. Authors in [37] used Multi-agent Reinforcement Learning to increase the network resource utilization.

Ashkan Yousefpour et al. [40] proposed a policy to minimize the latency of the IoT services. They aimed to assign each task according to the response time. The proposed policy in [40] divides the tasks to light and heavy. If the response time on a specific fog node is less than the threshold, the task will be assigned to this node. Otherwise, the task will be forwarded to one of the neighboring nodes or to the cloud. The limitations of this method are: (i) It explores various scenarios in a distributed manner. (ii) It is not reasonable to decide whether to assign task to fog or to cloud depending on the processing time.

The RA problem in CC has gained attention for several years. However, there are little studies related to this issue in FC. Table 1 summarizes some of these related works highlighting their strength and weakness.

Many researches about PM2.5 prediction are presented in many research papers such as in [10, 19, 20]. Convolutional neural networks are frequently hampered by high computational

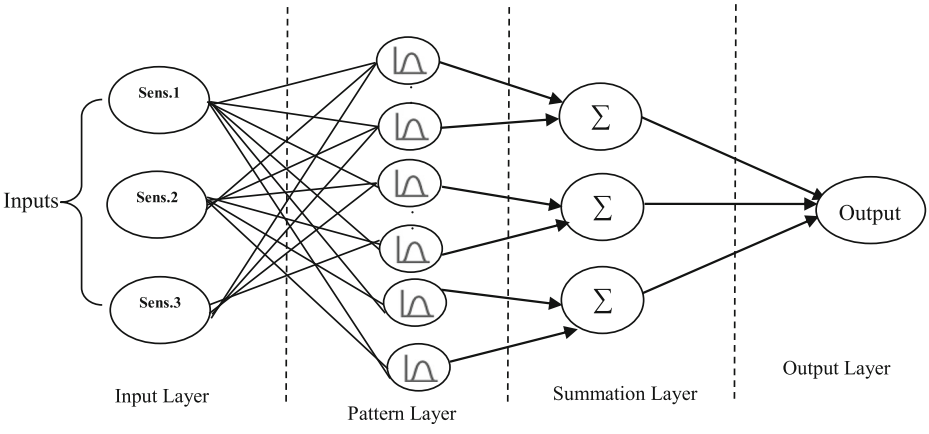

Fig. 7 PNN Architecture

Table 3 Servers' capabilities

Case	Sens.1_P	Sens.2_P	Sens.3_P	Probability
1	5	6	7	1
2	4	3	8	1
3	5	2	6	2
4	1	7	7	1
5	4	6	8	1
6	5	5	4	3
7	3	6	9	2
8	7	5	2	3
9	4	5	6	3
10	5	8	7	2

and storage needs. For several structural model pruning procedures and datasets (CIFAR-10 and ImageNet), the authors in [5] examine the accuracy-efficiency trade-off (TPUs).

2.1 Problem statement

Although, there are many RA algorithms they have many limitations such as i) most of them depend on the response time to decide whether to assign task to fog or to cloud which is not plausible. ii) They don't consider the task's requirements such as the priority of the task and the number of tasks. (iii) Calculating the capacity is difficult in some cases such as in case with varying packet size. (iv) They may cause network bottleneck.

2.2 Plan of solution

The proposed IoT-Fog system consists of three layers, namely: (i) IoT Layer, (ii) Fog Layer, and (iii) Cloud Layer. The mission of the IoT layer is to monitor the patient symptoms. The fog layer is considered with handling the incoming requests and forwards them to the suitable server. The cloud layer is responsible for managing the transfer of data to and from the fog layer. The user data is sent to the most appropriate resource to be allocated and processed. The allocating resource is managed by a specific healthcare organization.

Table 4 Input Data Set for PNN

Sens.1_P	Sens.2_P	Sens.3_P	Probability	Count_1	Count_2	Count_3
5	2	7	1	4	3	3
4	3	8	1			
3	4	7	1			
4	3	8	1			
5	2	6	2			
3	2	8	2			
5	4	7	2			
5	3	8	3			
4	2	6	3			
4	4	6	3			

Table 5 Test example

Sens.1_T	Sens.2_T	Sens.3_T
4	3	7

2.3 The proposed effective prediction and resource allocation methodology (EPRAM)

One of the most significant applications related to the aims of IoT is an efficient healthcare system. In healthcare systems, many factors should be taken into consideration such as time, privacy of data, and accuracy. The healthcare system should be reliable and available at any time. Accordingly, this paper is concerned with designing an IoT-Fog based Healthcare System as shown in Fig. 4. The proposed IoT-Fog system consists of three layers which are: (i) IoT Layer, (ii) Fog Layer, and (iii) Cloud Layer. The IoT layer combines the IoT devices (pulse oximeter, ECG monitor, etc.) to observe the user status. The fog layer is considered with handling the incoming requests and forwards them to the suitable Fog Node (FN). The fog layer is divided into set of fog regions. Each fog region has a Master Node (MN) which manages and controls the whole nodes' data. Each FN's managing software sends its characteristics information to the MN. The characteristics of the FN can be found by checking the values in each Node Characteristics Table (NCT) which is located at the MS. The main characteristics for each FN as shown in Table 2 are: (i) The cache size (capacity), (ii) The RAM size (RAM), and (iii) the usage of the CPU (CPU_Usage). A new characteristic (Adaptive Weight (AW)) will be calculated according to the mentioned parameters by (1) as:

$$AW = \alpha * \left(\frac{(Capacity * RAM)}{CPU_{Usage}} \right) \tag{1}$$

The cloud layer is responsible for managing the transfer of data to and from the fog layer.

2.4 A case study in smart healthcare

In Healthcare systems, the IoT devices are usually connected with smart phones. Examples of these devices are; pulse oximeter, ECG monitor, smart watches, etc. They send the health status of the patient via application module. The smart phones act as gateway node to preprocess the IoT sensed data. If the available resources at the gateway node meets the requirements, the process will be executed at the application. Otherwise it will be executed at FN. The application gateway node selects the appropriate node to execute the process and initiate actuators according to the result. The smart healthcare systems are useful in many cases such as: (i) Reduce the incidence of a heart attack among those with a cardiac disease, (ii) Early detection of symptoms of a particular virus, (iii). Detecting Parkinson's disease. The IoT based healthcare system architecture can be illustrated as 3-tier architecture as shown in the Fig. 4. Layer 1 combines the IoT devices (pulse oximeter, ECG monitor, etc.), layer 2 contains the fog nodes combined as set of regions, and layer 3 is the cloud datacenters.

2.5 The proposed EPRAM

This subsection introduces a new Effective Prediction and Resource Allocation Methodology (EPRAM) for Fog environment, which is suitable for Healthcare applications. ERAM tries to

Table 6 Training Calculations 1

Sens.1_P-Sens.1_T	Sens.2_P-Sens.2_T	Sens.3_P-Sens.3_T	(Sens.1_P-Sens.1_T)^2	(Sens.2_P-Sens.2_T)^2	(Sens.3_P-Sens.3_T)^2	$G_1 = \exp(-((Sens.1\_P-Sens.1\_T)^2)/2)$	$G_2 = \exp(-((Sens.2\_P-Sens.2\_T)^2)/2)$	$G_3 = \exp(-((Sens.3\_P-Sens.3\_T)^2)/2)$	$G_1+G_2+G_3$
1	−1	0	1	1	0	1.648	1.648	1	4.296
0	0	1	0	0	1	1	1	1.648	3.648
−1	1	0	1	1	0	1.648	1.648	1	4.296
0	0	1	0	0	1	1	1	1.648	3.648
1	−1	−1	1	1	1	1.648	1.648	1.648	4.944
−1	−1	1	1	1	1	1.648	1.648	1.648	4.944
1	1	0	1	1	0	1.648	1.648	1	4.296
1	0	1	1	0	1	1.648	1	1.648	4.296
0	−1	−1	0	1	1	1	1.648	1.648	4.296
0	1	-1	0	1	1	1	1.648	1.648	4.296

Table 7 Training Calculations 2

$Sum(S_1)$	$Sum(S_2)$	$Sum(S_3)$	$P_1 = Sum(S_1) / Count\_1$	$P_2 = Sum(S_2) / Count\_2$	$P_3 = Sum(S_3) / Count\_3$
15.888	14.832	12.888	3.972	4.944	4.296

achieve effective resource management in Fog environment via real-time resource allocating as well as prediction algorithm. The patient data is sent to the most appropriate server to handle it. This server is administrated by a specific healthcare organization. The main goal of the system is to achieve a low latency. The fog layer consists of three main modules as shown in the Fig. 5, namely: (i) Data Preprocessing Module (DPM), (ii) Resource Allocation Module (RAM) and (ii) Effective Prediction Module (EPM).

2.5.1 Data preprocessing module (DPM)

The DPM is responsible for sampling, partitioning, and balancing data to be in the appropriate form for analyzing and processing. The DPM is divided into three sub modules, namely: (i) Sampling Module (SM), (ii) Partitioning Module (PM) and (ii) Balancing Module (BM).

Sampling module (SM) In SM, the incoming data is divided into subgroups (or strata) which share a similar characteristic using stratified sampling algorithm. In stratified sampling, it may also be appropriate to choose non-equal sample sizes from each stratum. In SM, data is first sampled according to the location it comes from. Then it will be sampled according to its type. Data can be classified as the following: (i) Not or Low Critical: Examples are the logging of training activity, weight or body posture. Such data can be examined by doctor when needed. If the system fails to log some data points, the patient is still safe. (ii) Critical Data: data in critical conditions. Examples are cardiac monitoring via ECG with automatic alarms once critical situations are detected [9]. The criticality requires fast response time, i.e., real-time response. Context Management merely observes patients, devices, or employees to figure out their context and help by improving planning or taking proper decisions [32]. However, data in this case is not urgent but it gains some degree of criticality due to real-time response need. (iii) Very Critical Data Control: The detected events are not only used to alert personnel, but also to

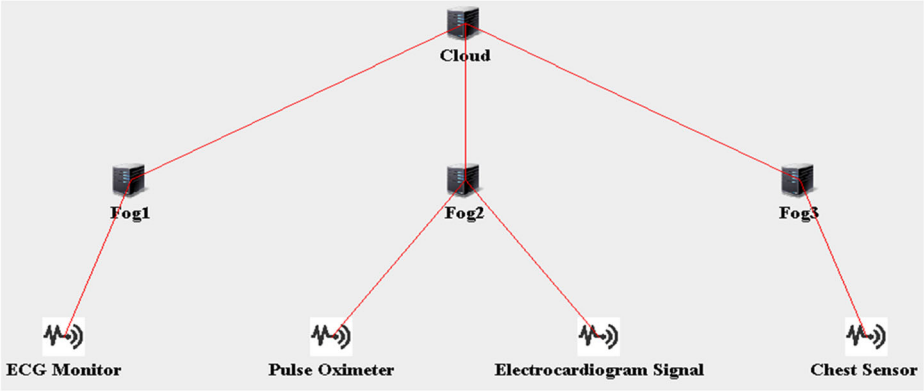

Fig. 8 Simulation of Fog Topology

control devices. This kind of data needs a feedback and real-time response. An example is a device that regulates the amount of oxygen provided to a patient [23].

Partitioning module (PM) PM splits the data into three samples. The model is built on the training set, and the model is applied to the testing set to establish its credibility. However, the testing set can then be used to further refine the model. If the performance of the model needs improvement, the parameters can be changed, and the model is then rebuilt using the training sample, after which the performance on the testing set is examined. The validation sample, which unlike the training and testing sets played no role in developing the final model, is then used to assess the model the model's performance against unseen data.

Balancing module (BM) The BM achieves the balancing of the data by discarding (reducing) records in the higher-frequency categories. The reason is that when you boost record you will run the risk of duplicating anomalies in the data.

2.5.2 Resource allocation module (RAM)

The proposed RAM is based on Reinforcement Learning (RL) algorithm to achieve a high LB for fog environment. In RL, an agent learns to interact with environment to achieve a reward. RL is an Artificial Intelligence technique in which an agent takes an action in an environment to gain rewards. The agent receives the current environment state and takes an action accordingly. The taken action leads to a change in the environment state and then the agent will be informed with the change through a reward. The RAM learns to select the best FN to execute the incoming request. The overall steps of the Resource Allocation Module (RAM) are shown in Algorithm 1.

Resource Allocation Module (RAM)

- ***Input:***
  - The data in the NCT
- ***Output:***
  - Balanced system with high performance and lowest latency.

- ***Steps:***

1: **For each new incoming process (P_i) do**
2: Create a Q-table containing two columns [State: Available Underloaded and Balanced FSs, Action: Selecting the best FS to execute P_i] initialized to zero.
3: **//Taking Action**
4: The agent interacts with the environment (FNs) and makes updates to the state-action pairs in the Q-table Q[state, action].
5: The agent uses the Q-table as a reference and view all possible actions (all possible available FNs) for a given state. Then it selects the best FN based on the lowest latency.
8: **// Updating the Q-table**
9: Update the values in the Q-table
10: **//Update the data in SCT**
11: The LBA updates the data in the SCT (update the AW and the status for each FN)
12: **Next**

Algorithm 1. (RAM) uses RL to achieve the lowest latency (the target of RL or the reward is defined as the lowest latency). The reward of RL is defined by the user as we want the agent

Table 8 MHEALTH dataset characteristics

Data Set Characteristics:	Attribute Characteristics:	Associated Tasks:	Number of Attributes:	Number of Instances:	Missing Values?
Multivariate, Time-Series	Real	Classification	23	161,280	N/A

Table 9 Used MHEALTH dataset

Heart_Attack_Probability	1	2	3
No. of Training Dataset Instances	180	175	205

Table 10 A sample of MHEALTH Dataset

chest sensor (X axis)	chest sensor (Y axis)	chest sensor (Z axis)	electrocardiogram signal (lead 1)	……	Heart_Attack_Probability
−9.7409	0.68291	0.7562	−0.06698	……	1

performs. We here define the reward as the low latency. It achieved that by training and interacting with the environment [24]+.

2.5.3 Effective prediction module (EPM)

The EPM uses the PNN to predict a target field, using one or more predictors. In order to detect the probability of the heart attack, PNN is trained using the training dataset. Then PNN will be tested using the user's sensing data coming from the IoT layer to predict the probability of heart attack and then take the most appropriate action accordingly. The architecture of PNN is

Table 11 the Performance metrics to evaluate the proposed EPRAM scheme

Metric	Definition	Notes
Makespan	It is the total execution time in which task get scheduled or completely executed. It can be called Completion Time (CT) CT: is the time at which process completes its execution.	Makespan always should be low [31].
Turn Around Time (TAT)	TAT: is the time difference between completion time and arrival time.	
Waiting Time (WT)	WT: is the time difference between turnaround time and burst time.	
Average Resource Utilization (ARU)	It is the complete utilization of each resource present in fog environment.	For better performance, ARU ratio should be high.
Load Balancing Level (LBL)		For better performance, LBL should be high [18].

shown in Figs. 6 and 7. The steps of PNN based Prediction Algorithm (PPA) are shown in Algorithm 2.

PNN based Prediction Algorithm (PPA)

- ***Input:***
  - *C is the number of classes, N is the number of examples, N_k are from class k*
  - *d is the dimensionality of the training examples, sigma is the smoothing factor*
  - *Examples[N][d] are the training examples*
  - *test_example[d] is the example to be classified*
- ***Output:***
  - *Probability of Heart Attackr*
- ***Steps:***

```
// for training:
int PNN(int C, int N, int d, float sigma, float test_example[d], float Examples[N][d]){
        int MS = -1;
        float largest = 0;
        float sum[ C ], Product[ d ], sump[ d ], SumS[ N_k ];
        float sump, ExSum;
        // The OUTPUT layer which computes the Gaussian functions (G) for each class C
        for ( int k=0; k<C; k++ ){
                sum[ k ] = 0;
                // The SUMMATION layer which accumulates the Gaussian functions (G)
                // for each example from the particular class k
                for ( int i=0; i<N_k; i++ ){
                        float product = 0;
                        // The PATTERN layer that computes eculidean distance
                        // for each parameter from the particular class
                        for ( int j=0; j<d; j++ ){
                                Product[j] = test_example[j] - Examples[i][j];
                                Product[j] = (product[j]* product[j]) / (sigma * sigma);
                                Product[j] = exp( product[j] );
                                sump += product[j];
                        }
                        SumS[i] = sump;
                        ExSum+=SumS[i];
                }
                sum[ k ] = ExSum;
                sum[ k ] /= N_k;}
        Target=sum[0];
        for ( int k=1; k<=C; k++ ){
                if ( sum[ k ] > largest ){
                        largest = sum[ k ];
                        MS = k;}
        return MS;}
```

Illustrative example

Assume there are ten cases. For each case, we consider the values from three sensors Sens.1_P, Sens.2_P, and Sens.3_P as shown in Table 3.

When new incoming sensing data arrives with values [2, 21, 22]. Count_1 is the number of examples belongs to Probability 1. Count_2 is the number of examples belongs to Probability 2. Count_3 is the number of examples belongs to Probability 3 (Tables 4, 5, 6, 7).

As P2 has the largest value, incoming sensing data arrives with values [2, 21, 22] will be detected as Probability 2.

3 Implementation and evaluation

FC runs applications in fog devices between cloud and end devices. This paradigm is used benefits of cloud and edge for distributed data and low latency. IoT sensors which are located

Table 12 EPRAM vs. previous LB algorithms

Algorithm	WT (in ms)	TAT (in ms)
LC	5.5	10.75
RR	8.5	13.75
WRR	7	12
AWRR	5	10
EPRAM	4.27	8.65

in the lower layer of the architecture, are responsible for receiving and transmitting data through the gateways to the higher layer. The actuators in the lowest level, are also responsible for system controls. In fact, FC provides filtering and analysis of data by edge devices. Each application of fog network has a different topology.

3.1 Simulation tool

iFogSim [33] is a toolkit used for simulation and modeling FC environments. It is used for evaluation of scheduling algorithms and resource management techniques in FC Environments. It can be used in different scenarios and it focuses on the effect on operational cost, power consumption, latency, and network congestion. It simulates cloud data centers, network links, and edge devices to measure performance metrics. iFogSim is available for download from: https://github.com/harshitgupta1337/fogsim. NetBeans IDE 8.0.2 can be downloaded from: https://netbeans.org/downloads/8.0.2/. Or Eclipse Modeling Tools from: http://www.eclipse.org/downloads/packages/release/Mars/2. The iFogsim simulator has a Graphical User Interface (GUI) module for designing custom and ready topologies [33]. Sensors, actuators, fog, cloud, and link elements can be added to the topology via this GUI. We create a case study with the new topology for healthcare environment and use a case study in iFogsim as shown in Fig. 8. These topologies can be read and executed by other modules in the simulator.

The Resource Allocation Module (RAM) and the Effective Prediction Module (EPM) are implemented using python.

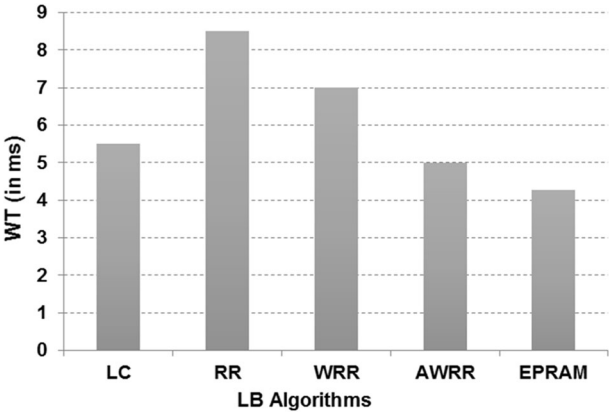

Fig. 9 WT for EPRAM vs previous LB algorithms

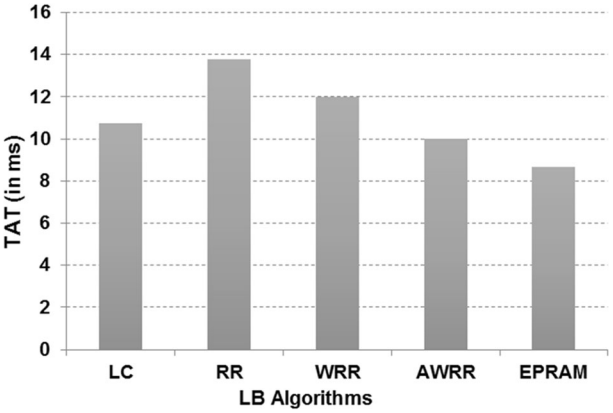

Fig. 10 TAT for EPRAM vs previous LB algorithms

3.2 Mobile HEALTH dataset

The Mobile HEALTH (MHEALTH) (https://archive.ics.uci.edu/ml/datasets/MHEALTH+Dataset) dataset contains vital signs and body motion recordings for 10 volunteers during several physical activities. Sensors placed on the subject's chest, right wrist and left ankle are used to measure the motion experienced by diverse body parts, namely, acceleration, rate of turn and magnetic field orientation. The main characteristics of MHEALTH Dataset are shown in Table 8.

In this paper, the MHEALTH is used to detect the possibility of heart attack. Hence, a new extra column (column 24) is added to have the values of the probability of attack. Column 24 has three main values which are; (i) 1 for strong probability, (ii) 2 for average probability, (iii) 3 for low probability. In order to simplify the classification, only 560 instances are selected from the MHEALTH Dataset for training and 240 instances are selected for testing. Number of training dataset instances for each probability is shown in Table 9. A sample of MHEALTH Dataset is shown in Table 10.

3.3 Performance metrics

The performance of LBOS vs. previous mentioned LB algorithms can be compared by considering the following metrics shown in Table 11.

TAT: is the time difference between completion time and arrival time as calculated in (2). WT: is the time difference between turnaround time and burst time as calculated in (3).

Table 13 Makespan analysis (in ms)

Number of tasks	Algorithm				
	LC	RR	WRR	AWRR	EPRAM
50	95.33	96	93.13	87.95	84.87
90	169.34	170.98	167.78	164.66	150.75
130	232.48	235.01	230.21	225.81	220.71
150	280.74	285.11	279.85	275.01	270.89

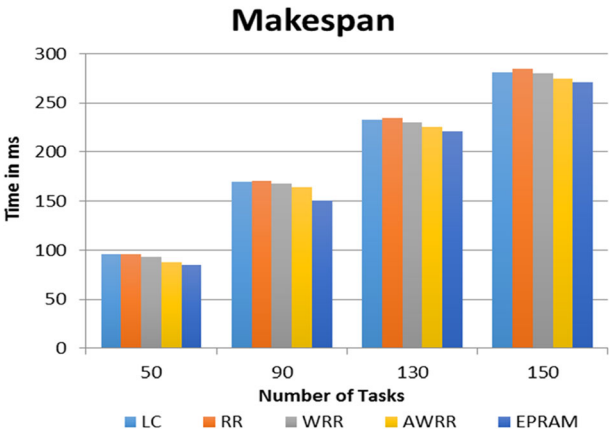

Fig. 11 Makespan for EPRAM vs Previous LB algorithms

$$TAT = CT - AT \tag{2}$$

$$WT = TAT - BT \tag{3}$$

Where, AT: is the Arrival Time which is the point of time in milli seconds at which a process arrives at the ready queue to begin the execution. BT: is the Burst Time which refers to the time required in milli seconds by a process for its execution.

ARU can be calculated as in (4) and LBL can be calculated as in (5).

$$ARU = \frac{(BS + OL)}{FSs} * 100\% \tag{4}$$

$$LBL = \frac{BS}{FSs} * 100\% \tag{5}$$

Where, BS: is the number of Balanced FNs, OL: is the number of Overloaded FNs, and FNs: is the number of all available FSs.

Table 14 ARU Analysis (%)

Number of tasks	Algorithm				
	LC	RR	WRR	AWRR	EPRAM
50	42.85	42.85	57.14	71.42	72.58
90	57.14	42.85	57.14	71.42	76.28
130	71.42	71.42	71.42	85.71	86.89
150	71.42	71.42	85.71	85.71	88.11

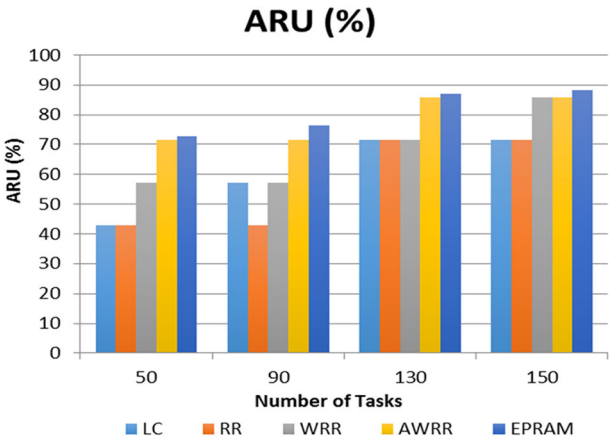

Fig. 12 ARU for EPRAM vs Previous LB algorithms

3.4 EPRAM implementation

Comparing the performance of the proposed EPRAM algorithm and the previous LB algorithms (LC, RR, WRR, and AWRR) is shown in Table 12. The value of WT for EPRAM compared to previous LB algorithms is shown in Fig. 3. The value of WT for EPRAM compared to previous LB algorithms is shown in Fig. 4.

From Figs. 9 and 10, it's observed that the EPRAM algorithm takes the lowest WT and the lowest TAT because the RL develops an agent that can assign tasks fast and efficiently [24]. It also uses the PNN for the prediction algorithm. It has achieved such acceptable performance due to using deep RL and PNN. Deep RL has shown impressive promises in resource allocation. PNN generates accurate predicted target and is much faster than multilayer perceptron networks [17]. EPRAM algorithm has been compared with LC, RR, WRR, and AWRR. The values of makespan are shown in Table 13 and in Fig. 11. The values of ARU are shown in Table 14 and in Fig. 12. The values of LBL are shown in Table 15 and in Fig. 13.

Figure 11 explained that EPRAM algorithm gives lower Makespan as compared to previous LB algorithms. Figures 12 and 13 explained that EPRAM algorithm gives better result as compared to previous LB algorithms as it achieved the higher ARU and higher LBL. Hence all the above results have shown that EPRAM algorithm performs better for makespan, ARU and LBL as compared to LC, RR, WRR, and AWRR.

Table 15 LBL Analysis (%)

Number of tasks	Algorithm				
	LC	RR	WRR	AWRR	EPRAM
50	28.57	28.57	42.85	57.14	59.04
90	42.85	28.57	57.14	71.42	72.45
130	57.14	57.14	71.42	71.42	73.45
150	71.42	57.14	71.42	85.71	86.14

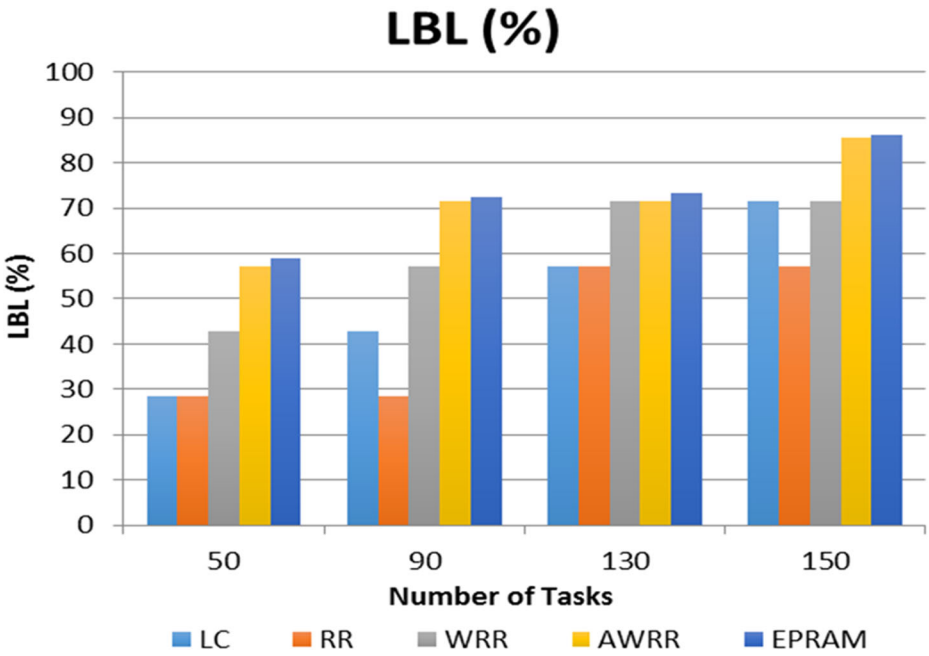

Fig. 13 LBL for EPRAM vs Previous LB algorithms

4 Conclusions and future work

This paper presented a new Effective Resource Allocation Methodology (ERAM) for Fog environment, which is suitable for Healthcare applications. The proposed IoT-Fog system consists of three layers, namely: (i) IoT Layer, (ii) Fog Layer, and (iii) Cloud Layer. The mission of the IoT layer is to monitor the patient symptoms. The fog layer is considered with handling the incoming requests and forwards them to the suitable server. The cloud layer is responsible for managing the transfer of data to and from the fog layer. The user data is sent to the most appropriate resource to be allocated and processed. The allocating resource is managed by a specific healthcare organization. ERAM achieved an effective resource management in Fog environment via real-time resource allocating as well as prediction algorithm. ERAM is composed of three main modules, namely: (i) Data Preprocessing Module (DPM), (ii) Resource Allocation Module (RAM) and (ii) Effective Prediction Module (EPM). The DPM is responsible for sampling, partitioning, and balancing data to be in the appropriate form for analyzing and processing. The RAM learned to select the best FN to execute the incoming request. The RAM uses Reinforcement Learning (RL) algorithm to achieve a high LB for fog environment. The EPM used the PNN to predict a target field, using one or more predictors. In order to detect the probability of the heart attack, PNN is trained using the training dataset. Then PNN has been tested using the user's sensing data coming from the IoT layer to predict the probability of heart attack and then take the most appropriate action accordingly. The main goal of the system is to achieve a low latency while improving the Quality of Service (QoS) metrics such as (the allocation cost, the response time, bandwidth efficiency and energy consumption). Unlike other RA techniques, EPRAM employed deep RL algorithm in a new manner. It also used the PNN for the prediction algorithm. It has achieved such acceptable

performance due to using deep RL and PNN. Deep RL has shown impressive promises in resource allocation. PNN generates accurate predicted target and is much faster than multilayer perceptron networks. Comparing the EPRAM with the state-of-the-art algorithms, EPRAM achieved the minimum Makespan as compared to previous LB algorithms, while maximizing the Average Resource Utilization (ARU) and the Load Balancing Level (LBL). Accordingly, EPRAM is a suitable algorithm in the case of real-time systems in FC which leads to load balancing. Then EPRAM has been tested using the user's sensing data coming from the IoT layer to predict the probability of heart attack and then take the most appropriate action accordingly. In future work, we aim to test EPRAM using various datasets to calculate the data transfer rate and compare it with the previous algorithms in different scenarios. In future work, EPRAM will also be tested to be used at different hierarchical levels with current investigations on how to make it more distributed.

Funding Open access funding provided by The Science, Technology & Innovation Funding Authority (STDF) in cooperation with The Egyptian Knowledge Bank (EKB).

Declarations

Funding and/or conflicts of interests/competing interests We wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

References

1. Alam M, Khan Z A (2017) Issues and challenges of load balancing algorithm in cloud computing environment. Indian journal of science and Technology. 10(25). DOI: https://doi.org/10.17485/ijst/2017/v10i25/105688
2. H. F. Atlam, R. J. Walters, G. B. Wills, "Fog Computing and the Internet of Things: A Review", Big Data Cogn. Comput., 2), (2018).
3. Bengio Y Learning Deep Architectures for AI (PDF). Foundations and Trends in Machine Learning 2(1):1–127. CiteSeerX 10.1.1.701.9550. https://doi.org/10.1561/2200000006 Archived from the original (PDF) on 2016-03-04. Retrieved 2015-09-03
4. Bengio Y, Courville A, Vincent P (2013) Representation Learning: A Review and New Perspectives. IEEE Transactions on Pattern Analysis and Machine Intelligence 35(8):1798–1828. arXiv:1206.5538. https://doi.org/10.1109/tpami.2013.50
5. Chen K, Franko K, Sang R Structured Model Pruning of Convolutional Networks on Tensor Processing Units. Computer Science, Machine Learning arxiv.org/abs/2107.04191
6. Deng L, Yu D (2014) Deep learning: Methods and applications. Found. Trends Signal Process 7:197–387
7. Dubey Sh, Dahiya M, Jain S (2019) Implementation of load balancing algorithm with cloud collaboration for logistics. Journal of Engineering and Applied Sciences 14(2): 507-515. https://doi.org/10.36478/jeasci.2019.507.515

8. Fan Q, Ansari N (2018) Towards workload balancing in fog computing empowered IoT. IEEE Transactions on Network Science and Engineering
9. Gia TN, Jiang M, Rahmani A-M, Westerlund T, Liljeberg P, Tenhunen H (2015) Fog computing in healthcare Internet of Things: A case study on ECG feature extraction. In: Proc. IEEE Int. Conf. Comput. Inf. Technol., Ubiquitous Comput. Commun., Dependable, Auto. Secur. Com-put., Pervasive Intell. Comput. (CIT/IUCC/DASC/PICOM), Oct., pp 356–363
10. Gu K, Liu H, Xia Z, Qiao J, Lin W, Thalmann D (2021) $PM_{2.5}$ monitoring: use information abundance measurement and wide and deep learning. IEEE Trans Neural Netw Learn Syst 32(10). https://doi.org/10.1109/TNNLS.2021.3105394
11. Gubbi J, Buyya R, Marusic S, Palaniswami M (2013) Internet of things (IoT): a vision, architectural elements, and future directions. Futur Gener Comput Syst 29(7):1645–1660
12. H. Gupta, A. Dastjerdi , S. Ghosh, and R. Buyya, iFogSim: a toolkit for modeling and simulation of resource management techniques in internet of things, edge and fog computing environments, software: practice and experience (SPE), 47(9): 1275–1296, 2017
13. Gupta S, Rani S, Dixit A, Dev H (2019) Features exploration of distinct load balancing algorithms in cloud computing environment. Int J Advanced Networking and Applications 11(1):4177–4183 ISSN: 0975-0290
14. Hindman B, Konwinski A, Zaharia M, Ghodsi A, Joseph AD, Katz R, Shenker S, Stoica I (2011) Mesos: A platform for fine-grained resource sharing in the data center. In: Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation, Boston, MA, USA, 30 March–1 April 2011; USENIX Association: Berkeley, CA, USA, pp 295–308
15. Hof RD (2018) Is artificial intelligence finally coming into its own? MIT Technology Review, Retrieved
16. Hua P, Dhelima S, Ninga H, Qiud T (2017) Survey on fog computing: architecture, key technologies, applications and open issues. Journal of Network and Computer Applications 98:27–42
17. Karthikeyan B, Gopal S, Venkatesh S (2008) Partial discharge pattern classification using composite versions of probabilistic neural network inference engine. Expert Syst. Appl 34:1938–1947
18. Kaur R, Luthra P (2014) Load balancing in cloud system using max min and min min ALGORITHM. International Journal of Computer Applications (0975 –8887) National Conference on emerging trends in computer technology (NCETCT-2014): 31-34
19. Ke G, Qiao J, Li X (2019) Highly Efficient Picture-Based Prediction of PM2.5 Concentration. IEEE Transactions on Industrial Electronics 66(4). https://doi.org/10.1109/TIE.2018.2840515
20. Ke G, Xia Z, Qiao J (2020) Stacked Selective Ensemble for PM2.5 Forecast. IEEE Transactions on Instrumentation and Measurement 69(3). https://doi.org/10.1109/TIM.2019.2905904
21. Mahmud R, Ramamohanarao K, Buyya R (2017) In: Beniamino DM, Laurence Y, Li K-C, Antonio E (eds) ISBN 978–981–10-5861-5Fog computing: A taxonomy, survey and future directions, Internet of Everything: Algorithms, Methodologies, Technologies and Perspectives. Springer, Singapore
22. Mahmud R, Koch FL, Buyya R (2018) Cloud-Fog Interoperability in IoT-enabled Healthcare Solutions. In: Proceedings of the 19th International Conference on Distributed Computing and Networking (ICDCN '18), pp. 1–10, Varanasi, India, Jan. 4–7
23. Masip-Bruin X, Marín-Tordera E, Alonso A, Garcia J (2016) Fog-to-cloud computing (F2C): The key technology enabler for dependable ehealth services deployment. In: Proc. Medit. Ad Hoc Netw. Workshop (Med-Hoc-Net), June, pp 1–5
24. Matthew B (2019) Reinforcement learning, fast and slow. Trends Cogn Sci 23(5):408–422. https://doi.org/10.1016/j.tics.2019.02.006
25. Online (n.d.) https://www.smplsft.com/SimpleIoTSimulator.html
26. Schmidhuber J (2015) Deep Learning in Neural Networks: An Overview. Neural Networks 61:85–117. arXiv:1404.7828. https://doi.org/10.1016/j.neunet.2014.09.003
27. Singh G, Kaur K (2018) An improved weighted least connection scheduling algorithm for load balancing in web cluster systems. International Research Journal of Engineering and Technology (IRJET). 5(3). E-ISSN: 2395-0056
28. Sonmez C, Ozgovde A, Ersoy C (2017) Edgecloudsim, An environment for performance evaluation of edge computing systems. In: Proceedings of the Second International Conference on Fog and Mobile Edge Computing (FMEC'17), pp. 39–44, Valencia, Spain, May. 8–11
29. Szegedy C, Toshev A, Erhan D (2013) Deep neural networks for object detection. Advances in Neural Information Processing Systems:2553–2561
30. Talaat Fatma M, Mohamed S, Ahmed S, Hesham A, Shereen A (2020) A load balancing and optimization strategy (LBOS) using reinforcement learning in fog computing environment. Journal of Ambient Intelligence and Humanized Computing. https://doi.org/10.1007/s12652-020-01768-8
31. Tan Y, Liu W, Qiu Q (2009) Adaptive power management using reinforcement learning. In: Proceedings of the 2009 international conference on computer-aided design, ACM: 461-467. https://doi.org/10.1145/1687399.1687486

32. Tentori M, Favela J (2007) Activity-aware computing in mobile collaborative working environments. In: Proc. 13th Int. Conf. Groupw., Design Implement. (CRIWG), Berlin, Germany, September, pp 337–353
33. Vaquero LM, Rodero-Merino L (2014) Finding your way in the fog. ACM SIGCOMM Comput. Commun. Rev 44(5):27–32
34. Vavilapalli VK, Murthy AC, Douglas C, Agarwal S, Konar M, Evans R, Graves T, Lowe J, Shah H, Seth S et al (2013) Apache hadoop YARN: Yet another resource negotiator. In: Proceedings of the 4th Annual Symposium on Cloud Computing, Santa Clara, CA, USA, 1–3 October. ACM, New York, NY, USA, pp 5: 1–5:16
35. Venkatesh S, Gopal S (2011) Robust Heteroscedastic Probabilistic Neural Network for multiple source partial discharge pattern recognition—Significance of outliers on classification capability. Expert Syst. Appl 38:11501–11514
36. Wei Y, Yu FR, Song M, Han Z (2018) User scheduling and resource allocation in HetNets with hybrid energy supply: an actor-critic reinforcement learning approach. IEEE Trans Wirel Commun 17(1):680–692. https://doi.org/10.1109/TWC.2017.2769644
37. Yan M, Feng G, Qin S (2017) Multi-RAT Access Based on Multi-Agent Reinforcement Learning. In: 2017 IEEE global communications conference. https://doi.org/10.1109/GLOCOM.2017.8254980
38. Yi S, Li C, Li Q (2015) A survey of fog computing: concepts, applications and issues. In: Proceedings of the 2015 Workshop on Mobile Big Data. ACM, pp 37–42
39. Yoshua B, Yann L, Geoffrey H (2015) "Deep Learning". Nature 521(7553): 436–444. Bibcode: 2015Natur.521..436L. https://doi.org/10.1038/nature14539
40. Yousefpour A, Ishigaki G, Jue JP (2017) Fog computing: Towards minimizing delay in the internet of things. In: 2017 IEEE International Conference On Edge Computing (EDGE), pp 17–24
41. Zaharia M, Borthakur D, Sen Sarma J, Elmeleegy K, Shenker S, Stoica I (2010) Delay scheduling: A simple technique for achieving locality and fairness in cluster scheduling. In: Proceedings of the 5th European Conference on Computer Systems; ACM: New York, NY, USA, pp 265–278