Internet of Things 7 (2019) 100070

Contents lists available at ScienceDirect

Internet of Things

journal homepage: www.elsevier.com/locate/iot

Research article

SecOFF-FCIoT: Machine learning based secure offloading in Fog-Cloud of things for smart city applications

Adam A. Alli[a,1], Muhammad Mahbub Alam[b]

[a] Islamic University in Uganda, Uganda
[b] Islamic University of Technology, Bangladesh 1704 Board Bazar, Gazipur

ARTICLE INFO

Article history:
Received 28 December 2018
Revised 18 April 2019
Accepted 2 June 2019
Available online 7 June 2019

Keywords:
Secure computation offloading
Internet of Things
Fog-Cloud
Neuro-Fuzzy systems
Smart city application
Reinforcement learning

ABSTRACT

Computation offloading is one of the important application in Internet of Things (IoT) ecosystem. Computational offloading provides assisted means of processing large amounts of data generated by abundant IoT devices, speed up processing of intensive tasks and save battery life. In this paper, we propose a secure computation offloading scheme in Fog-Cloud-IoT environment (SecOFF-FCIoT). Using machine learning strategies, we accomplish efficient, secure offloading in Fog-IoT setting. In particular, we employ Neuro-Fuzzy Model to secure data at the smart gateway, then the IoT device selects an optimal Fog node to which it can offload its workload using Particle Swarm Optimization(PSO) via the smart gateway. If the fog node is not capable of handling the workload, it is forwarded to the cloud after being classified as either sensitive or non-sensitive. Sensitive data is maintained in private cloud. Whereas non-sensitive data is offloaded using dynamic offloading strategy. In PSO, the availability of fog node is computed using two metrics; i) Available Processing Capacity (APC), and ii) Remaining Node Energy (RNE). Selection of cloud is based on Reinforcement Learning. Our proposed approach is implemented for smart city applications using NS-3 simulator with JAVA Programming. We compare our proposed secure computation offloading model with previous approaches which include DTO-SO, FCFS, LOTEC, and CMS-ACO. Simulation results show that our proposed scheme minimizes latency as compared to selected benchmarks.

1. Introduction

The emergency of ubiquitous and pervasive things have resulted in production of very extensive amount of data; as a result, data processing requirements in IoT ecosystem is growing much more faster than processing power, memory, cache, and battery life of the devices [1]. Cisco Global Cloud Index estimates that nearly 850 ZB will be generated by all people, machines, and things by 2021 up from 220 ZB generated in 2016. Of the 850 ZB expected only 10% will be useful and the rest will be ephemeral in nature. Again, useful data will be four times greater than the existing capacity of data centers of the time [2]. The trends of data cited above present opportunities for the use of Edge and Fog computing.

Fog computing environment enables computational offloading, data aggregation and storage. This allows Internet of Things(IoT) devices to provide users with satisfactory quality of service and quality of experience [3–5]. Numerous data intensive applications have been developed to employ the use of smart devices. Some examples of applications include

E-mail addresses: adamalli@iut_dhaka.edu (A.A. Alli), mma@iut-dhaka.edu (M.M. Alam).

[1] This piece of work is Part of my PhD at Islamic University of Technolog(IUT), 1704 Board Bazar, Gazipur, Dhaka, Bangladesh.

https://doi.org/10.1016/j.iot.2019.100070
2542-6605/© 2019 Elsevier B.V. All rights reserved.

self-parking cars, wearable devices, trackers and domestic appliances [6]. Most of these devices are small in size, they are battery powered, and house limited processing, storage, and memory supplies. Their size act as a bottleneck to implement and run powerful applications on them. We also note that the amount of data generated by IoT devices is increasing gradually [7], thereby increasing the network burden in terms of network congestion. Hence, it is necessary to utilize on-demand resourceful cloud paradigm that enables to process the data generated from IoT devices. This paradigm is often called Cloud of Things (CoT) [8,9].

Cloud computing paradigm provides extensive processing power and infinite storage that permits fast processing and bulk storage. It has been found useful in many applications that are not delay sensitive and do not require immediate responsiveness. However, the use of cloud computing paradigm exclusively may not be attractive in applications that require immediate processing, high responsiveness, and real-time analysis of IoT clients requests [10]. To this purpose, fog computing has been designed to mitigate issues of latency, fast responsiveness, provide real-time transactions, and bridge the network unreliability concerns.

Cloud computing extends the concept of fog computing for better utilization and minimum energy consumption [2,5]. Combining Cloud of Things to Fog Computing minimizes the service delay for IoT applications. In the [11], a novel service called Offload as a Service (OaaS) has been presented. OaaS provide the capability to extend limitations of mobile resources such as GPU, CPU, storage etc. However, the open issues in fog computing environment such as dynamic offloading, scalability, security, the use of minimum number of fog nodes to achieve efficiency and effectiveness still remain open issues [12,13].

To overwhelm all the aforesaid issues in the Cloud Integrated Fog-IoT paradigm, machine learning based approaches have been proposed in [3,14–16]. In our study, we choose a pipeline of machine learning strategies to achieve better performance. Machine learning has demonstrated latent power in solving complex problems in science, engineering, industrial and professional practices [17–21]. For example, in banking and finance machine learning can be used by executive managers to make informed decision. Machine learning can help banks spot potential business partners and their expenditure. Using Machine learning algorithm smart machines can be trained to monitor trends in the market and react in real time [22].

In medicine, machine learning has been used in diagnosis of complicated diseases. Moreover, it has been used in analysis of clinical parameters and their combination of the prognosis [23]. It can help in integrating computer based systems in health care industry. Additionally, machine learning has demonstrated ability to assist in solving complex problems in aerospace engineering. In [24], machine learning has been used to recognize specific defects of aerospace structures, decrease approximation errors and compute the closest possible outcome. The limits due to the great amount of data and the complexity of data processing make machine learning as defect classifier in aerospace structures very useful [24,25]. Machine learning has also been used to optimize parameters so as to find best options that maximize use of resources in logistic and supply management. Here, machine learning allows improved planning for unexpected events and help predict orders along the supply chain with at most accuracy [26].

In this paper, we propose a machine learning based secure offloading herein referred to as SecOFF-FCIoT framework. Our proposal is a software solution that approaches offloading data to the Fog or the cloud using machine learning approaches. Our dynamic and secure offloading reduces offloading latency and minimizes energy consumption.

1.1. Paper major contributions

The main contributions of this study are summarized as follows:

- We propose secure offloading that exploits the use of Particle Swarm Optimization(PSO) and Reinforcement Learning(RL) to obtain more efficient offloading in Fog-IoT environment. PSO is used at IoT level to select optimal node to offload its workload. Whereas, RL is used at the fog level to select suitable cloud.
- We introduce the use of Neuro-fuzzy model to isolate IoT nodes that try to congest the network by sending invalid data. If the IoT device is sensed to have sent invalid data for offloading, the data is dropped. By so doing nodes that exhibit low trust are stopped from transacting maliciously on the network.

1.2. Paper layout

The rest of this paper is organized as follows: preliminary literature related to computation offloading and fog computing is presented in section 2. Related studies are reviewed in Section 3. We present our solution in Section 4. In Section 5, experimental setting with comparative study is given. Conclusion are drawn in Section 6.

2. Preliminaries

2.1. Computational offloading mechanism

Secure Offloading is technically challenging problem in Edge/cloud enabled IoT environment, especially when dealing with wireless connected systems in which resources required for communication are highly dynamic.

In an IoT related environment, a device (smart phones, surveillance cameras, robot, smart-meter etc.) use applications installed on then to execute tasks. When the resources used by the device get below threshold the task is dropped. For instance, in a video chart, a call is placed through video application. During the call process the device continues to consume battery, memory, and storage space. When any of the resources (battery level, space or processing power) to support the call gets low, it is dropped. In this way the device is saved from dying out.

Instead of letting the device drop the process, a portion of data in the storage space may be relocated to a remote device to create more space on the device, or a part of the process which over utilizes the processor causing battery drainage can be migrated to remote systems. This helps the device to save battery as opposed to dropping the task. The above described process is known as computational offloading.

Computational offloading mechanism can be viewed as either fine grained or coarse grained procedure [27].

In coarse grain[2] computational offloading, the task is migrated to the cloud in whole and there is no need to estimate resources overhead. The decision needed in this coarse grain is either to execute the whole workload on the mobile device or it is sent to be executed at the cloud, while in fine grain computational offloading[3] a portion of the task is executed on the mobile device and the other portion is executed on the server (edge or cloud device).

In the fine grained offloading, little code is dynamically transmitted and only the computational hungry part of the code is offloaded to the remote device. Isolating processes in devices according to their processing needs can reduce unnecessary transmission overhead, improve performance and energy utilization [28].

The decision taken by an IoT device to offload part of its workload to remote device depends on the time taken to process the workload.

Assume, there exists some workload (ω) at the user equipment(UE) side say a mobile device(m), whose processing speed is s_m. This workload can be partitioned into two; one part that will always be executed at the the user equipment side e.g. user interface and code that manages peripheral (camera, temperature sensors, proximity sensor, accelerometer, etc.), and the other part that may be offloaded. If the offloadable workload is locally executed on the user equipment then Eq. (1) represents the time (τ_m) required to execute the workload;

$$\tau_m = \frac{\omega}{s_m} \tag{1}$$

otherwise, if the workload (ω) is executed on the server[4] whose processing speed is s_{se} then the time required to complete the workload (τ_{se}) is;

$$\tau_{se} = \frac{\omega}{s_{se}} \tag{2}$$

to execute the workload on the server equipment, it is necessary to transmit the workload over a network channel. To transmit the workload to remote it takes transmission time (τ_c). Transmission time can be computed as a ratio of data shared between the two equipment (UE and the Server) (d_s) and the bandwidth (B_w) between them.

$$\tau_c = \frac{d_s}{B_w} \tag{3}$$

Moreover, the energy required in completing the workload at the UE(En_m),server (En_{se}), and transmitting shared data to and from UE and server (En_c) can be computed in terms of τ_m, τ_s, τ_c as follows;

Let p_m be the processing power of UE and p_{se} be the processing power of the server equipment and p_c be the transmission power of the transmitting equipment for both the uplink and downlink,[5] then

$$En_m = p_m \times \frac{\omega}{s_m} \tag{4}$$

$$En_{se} = p_{se} \times \frac{\omega}{s_{se}} \tag{5}$$

$$En_c = p_c \times \frac{d_s}{B_w} \tag{6}$$

The total energy consumed during offloading process is the sum of energy required to process the portion of workload at user equipment, energy required to maintain the offloading process, and energy required to transmit the remainder of workload to the server.

Ignoring other conditions such as complexity of the workload, time taken during the initial setup, the size of program, offloading to be beneficial then Eq. (7) must hold.

$$\frac{\omega}{s_m} \geq \frac{\omega}{s_{se}} + \frac{d_s}{B_w} \tag{7}$$

[2] Coarse grain offloading strategy is synonymous to static computational offloading.
[3] Fine grain computational offloading is synonymous to dynamic offloading.
[4] used synonymous to imply Fog node or cloud server.
[5] uplink and downlink power may differ from time to time depending on channel conditions.

If the server is infinitely fast then Eq. (7) reduces to

$$\frac{\omega}{s_m} \geq \frac{d_s}{B_w}. \tag{8}$$

It is important to note that no matter how fast the server processor is, as long as the transmission time is greater than the time required to process the workload at the user equipment offloading will not improve performance. Considering workload that requires heavy computation, and light data shared is in the center of fine-grained computation offloading. Identifying parts of the task that require heavy computation with light data sharing is seen to improve performance [28]. Secondly, performance improvement is dependant on the offloading tasks that require heavy computation with light data sharing to minimize transmission energy. Moreover, dynamic offloading adapt to different run-time conditions e.g. fluctuating network bandwidths and mobility of devices in the network. Lastly, dynamic offloading make the use of prediction mechanisms such as machine learning, stochastic Bayesian methods, heuristic algorithms and other optimization algorithms attractive for decision making [27,28].

During Dynamic offloading, the workload may be split into smaller portions so that part of it is processed locally and the other part is offloaded to many remote devices [16]. A number of approaches used to split tasks have been explained in [28]. The most commonly used method is the graph method.[6] The two factors latency and energy consumption play a considerable role in determining the quality of offloading via fog nodes.

2.2. Fog computing

The concept of fog computing stretches from the outer edge where data is created to eventually where it is processed and/or stored. This computing and storage location could be within the organizational data center, the edge device(Router, switch etc.) or the cloud. Fog computing forms another layer of a distributed network environment in between the cloud computing and the Internet of Things(IoT) that provides a continuum to bridge the missing link for data that needs to be handled locally closer to the edge or pushed to the cloud [29]. Fog computing paradigm contains Fog Nodes distributed within limited geographical area where certain IoT mobile devices computation tasks are executed.

Among other computational objectives, fog computing aims at reducing the data amount that requires to be forwarded to the cloud for processing and storage. This activity improves system efficiency when massive data processing, storage and analysis are required in real-time. Each IoT device performs its computational tasks or offloads them to the Fog [30,31].

At the architectural point of view, the Fog computing provides a horizontal system architecture that distributes resources and services of computing, storage, control, and networking anywhere along the continuum from the cloud to Internet of Things.

With the applications or data that may need to be processed quickly for example in use case such as manufacturing, connected machines may require to respond to an event at most immediately. Fog computing provide appropriate choice as means of responding to such event. Secondly, the Fog is required in situations that may arise due to no bandwidth to send the data for processing to the cloud, or where amount of bandwidth required to send the data to either the organizational data center or the cloud is very expensive[14,32,33]. An additional benefit is that, the fog can be used to secure the data from segmented network.

3. Review of related work

Several works have been done related to computation offloading as well as the combination of cloud with IoT, but a few contributed to security in computation offloading. Our work adds to the body of knowledge related to secure computational offloading in trusted system. In this section, we present related work to computational offloading and security in fog environment as follows;

3.1. Review on computational offloading in Internet of Things

An offloading architecture called AutoScaler was proposed in [34]. In this paper, provisioning offloading as a service was explored. They proposed large-scale offloading in IoT environment. An offloading system consisting of front-end, back-end and load simulator which generates dynamic offloading workload of multiple devices was designed. The AutoScaler front-end component introduces extra time overhead of ≈ 150 ms in the overall response time of a client request. Adjusting a trade off between the utilization price and computational capabilities of the surrogate servers at the backend, it was observed that the total time of code invocation is reduced. Furthermore, the proposed architecture was easily deployable on large scale. Introducing surrogates entities in our Fog environment is likely to improve performance.

Ma et al. [35] used a game theoretical approach to analyze the decision problem of IoT devices. Their work based on a definition of potential game. Nash equilibrium was derived and algorithm known as computational offload decision(COD) was proposed. This algorithm, portray significant reduction in the system cost. Cost was viewed in terms of processing

[6] Graph method is method used to represent tasks as a directed acyclic graph(DAG). Vertices represent the computational components and the edges represent the communication between them. Splitting the task is done by partitioning the graph.

delay and energy consumption. COD is energy-aware computation offloading in cloudlet-based mobile edge computing. The computation offloading decision (COD) algorithm is based on decentralized computation offloading strategy for IoT devices. The energy consumption of IoT devices in computation offloading to cloud through base station was found to be significant. Another mobile cloud IoT (MCIoT) paradigm was proposed in [36] which uses a new nested game model for computation offloading. Firstly, each mobile device would determine the portion of remote offloading computation using Rubinstein game theory approach. Secondly, a computation resource in the cloud was assigned dynamically for the requested computation offloading. The nested game theory approach provides an optimal solution for the computation offloading in the MCIoT paradigm. However, game theory principle consumes more time for computation offloading.

Min et al. [3] proposed learning based computational offloading for devices with energy harvesting. They constructed a hotbooting Q-learning dynamic process that chooses a portion of data to offload to mobile edge computing devices(MEC) according to the system state (bandwidth, amount of energy harvested, and battery level) to determine an offload policy. Moreover, the system was formulated as a Markov Decision Process(MDP) in which the Q-learning technique was used to attain optimal policy. However, the hotbooting mechanism take significant length of time to converge. To improve further the system performance they proposed a Fast DQN computational offloading. The proposed system use the concept of machine learning in computational offloading though it implementation doesn't consider important issues such as mobility, cost of use of MEC and security in their utility function. These factors left out of the scope of the study could have an impact on the utility function and therefore impact on the policy. In addition, dynamic changes in the network parameters such as bandwidth and mobility was not considered. In [16], a deep reinforcement learning based on computational offloading has been proposed. Similar to [3], they have designed an offloading algorithm for optimal decision making subject to dynamics of the system in terms of user and cloudlet behavior. They concentrated on the composite behavior of the cloud queue and the distance between the cloudlet and the user equipment. Unlike in [3] where offload is achieved on one MEC at an instance, Van Le and Tham [16] solve the problem of offloading to multiple cloudlets.

Zhang at el. [37] proposed a hybrid computation offloading algorithm that combines cloudlet with public clouds, to provide a more energy-efficient offloading strategy for home automation applications. Particle Swarm Optimization (PSO) based heuristic algorithm is implemented to schedule mobile services. The task scheduling mechanism uses queuing model based on First Come First Served (FCFS). In the same study, the waiting time in the cloudlet is modeled as a $M/M/m/\infty$ queue while PSO is used to select unscheduled mobile services in the work flow. Scheduling tasks using FCFS is likely to be slower as compared to other scheduling mechanism, whereas using the PSO for selecting the Fog node may improve system performance. PSO is suitable for this kind of problem since IoT devices exhibits characteristics of a swarm. Dynamic Task Offloading (DyTO), has been proposed in [38], in their work they introduced the concept of surrogate object in a computational offloading environment. A surrogate object installed on the mobile cloud takes care of information that tracks the mobile host thereby separating cohesion of resources to the mobile host as it traverses or looses connection. The surrogate object maintains the information to ensure proper delivery of data especially in situation where the network is unstable and characterized by disconnection. The proposed model is observed to save energy. Execution is faster because interruptions in processing caused by disconnection, or variations in network resources caused by mobility is avoided. Applying similar solutions in secure offloading appears feasible.

3.2. Review on security and privacy in Internet of Things

Privacy, trust, and security are important factors to consider in IoT ecosystem. Huge amount of data outsourced from a global network of things connected to data centers, Fogs and cloud make IoT networks more vulnerable to cyber attacks [39]. Authors in [40] have proposed Smart Trust management method to detect On-Off attacks caused by nodes that may perform bad behaviour randomly in order to avoid low in trust ratings. Moreover, suspicious resources may behave differently with different neighbours. They proposed a method to collaborate with IoT devices to identify On-Off attacks and broken nodes. Interaction among IoT devices are evaluated using meta data attributes. A machine learning classifier is used to classify the data. From the trust score generated by the classifier, a decision is taken to either trust or mistrust the resource. Their proposed security method detect On-Off attack with 97% precision and 96% in real record data set. The method is 95% faster. This recent work is one of those that uses machine learning generated score in trust management of resources.

In [41], privacy preservation with IoT oriented offloading. This is a method for solving data transmission security problem. It was implemented in the WMANs (Wireless Metropolitan Area Networks). In their proposal, data is offloaded to the cloudlets through access points. To deal with shortest routing problem, the Dijkstra algorithm was used to establish shortest path between access points. Also NSDE (Non-dominated Sorting Differential Evolution Algorithm) was considered to resolve multi-objective optimization problem. Zhang et al. [37] have presented privacy-preserving data aggregation from hybrid IoT devices in fog computing. The proposed scheme ensures data integrity by guaranteeing that injection data is received from legitimate IoT devices. The proposed scheme reduces communication overhead, but increases computation overhead when searching for appropriate fog device. Thus, it increases processing delay.

In [42], a practical evaluation of a high security energy-efficient gateway was considered in IoT fog computing applications. The primary contribution of this paper is increased security levels for sensed data in resource-constrained environment using Rivest Shamir Adleman (RSA) and Elliptic Curve Cryptography (ECC). Furthermore, the proposal improves throughput and energy efficiency for IoT devices. In practical evaluation, ECC outperforms than RSA, but ECC needed further improvement to organize massive sensed data for real-time scenarios. Belem et al. [43] have proposed a device-based security called PROTeCt (Privacy aRchitecture for IntegratiOn of Internet of Things and Cloud computing), which improves user privacy. In the proposed architecture, user privacy is given by cryptography based scheme in which only the interested users can access their data, which is stored on cloud in encrypted form to protect IoT network from insecure access. In this works, gateway was authenticated. Users required to register and accepted to join the network time and again which increases communication cost.

To this end, our proposed work is one of the kind that is intended to tackle deficits in previous computation offloading works. To the best of our knowledge, we have presented a secure and dynamic offloading scheme which minimizes energy consumption and latency to obtain more efficient offloading in Fog-IoT environment.

4. Proposed system

4.1. Problem statement

We consider a network of Γ IoT nodes such that $\Gamma = \{1, 2, 3 \ldots n\}$. Each of IoT devices in this network may contain computation-intensive, or delay-sensitive computation task. These IoT devices are deployed in a network which are connected through a smart gateway to the Fog nodes and the cloud respectively creating a hierarchical network. The fog nodes form a network continuum to the cloud. Given a task, the IoT evaluates the task to see if it can execute the task locally using resident resources or not. If the IoT finds that it cannot execute the task, it offloads the task to the Fog. The Fog either performs the tasks or sends it to the cloud.

Our intention is to perform dynamic offloading while maintaining users sensitive tasks in the Fog during task offloading at the same time achieve high performance in terms of throughput, delay, energy consumption, resource utilization rate and response time.

4.2. System overview

The proposed system architecture shown in Fig. 1. consists of IoT mobile devices at the IoT layer, network devices at network layer, fog devices at the Fog layer, and cloud infrastructure at the cloud layer.

4.2.1. IoT devices

The IoT devices acquire, monitor and measure data. In addition, they send, receive data to and from the Fog. The infrastructure environments created by the IoT devices allow them to monitor and filter data from environment. IoT devices are characterized by low computational power, constrained by battery life and their small form factor. They have considerably low memory and powered by small battery cells [44–46].

4.2.2. Network layer

The network layer consists of network devices such as switches, routers and gateways. They may adopt the functionality of a fog on a small scale [45,46]. To mention in this framework, a smart gateway is adopted to secure the network through evaluation of data coming from IoT device. Using Neuro-Fuzzy model, we are able to predict if an IoT device is a candidate for malicious attack through reading obtained from the devices. Based on the readings, the node can be trusted.

4.2.3. Fog layer

The fog layer consists of Fog nodes. Fog nodes are high-performance distributed systems. They report results of processing to both Cloud and IoT layer along the continuum. Unlike the IoT devices, fog nodes are more powerful and have considerable amount of storage. They can provide localized services when need arise. They can be equipped with ability to support data analytics at transactions level [44–46]. In our system,we equip the Fog nodes with surrogate entity, task classifier, and task scheduler. Fog devices may be installed on moving objects such as in cars, buses or trains(mobile).

4.2.4. Surrogate entity

Surrogate entities are software defined objects installed on the Fog Node. Their function is to collect and store information about the IoT devices which are in service. IoT device may roam from one network to another, or may face network unreliability situation which may lead to breakdown in services. As a result, the device is required to seek the service again when network becomes available. To avoid such situations, the surrogate entity [38], serves as information holder. If the IoT devices goes out of service, the service at the fog remains active. After the service is completed the fog node uploads the results to the IoT device. The entities help decouple IoT devices during service time hence improving performance, allowing for mobility and uncertainties in the network connectivity. Information in the Surrogate entity remains unchanged during service time, though some IoT device information may change from time to time due to mobility. They generate compact data sets which are less expensive to maintain.

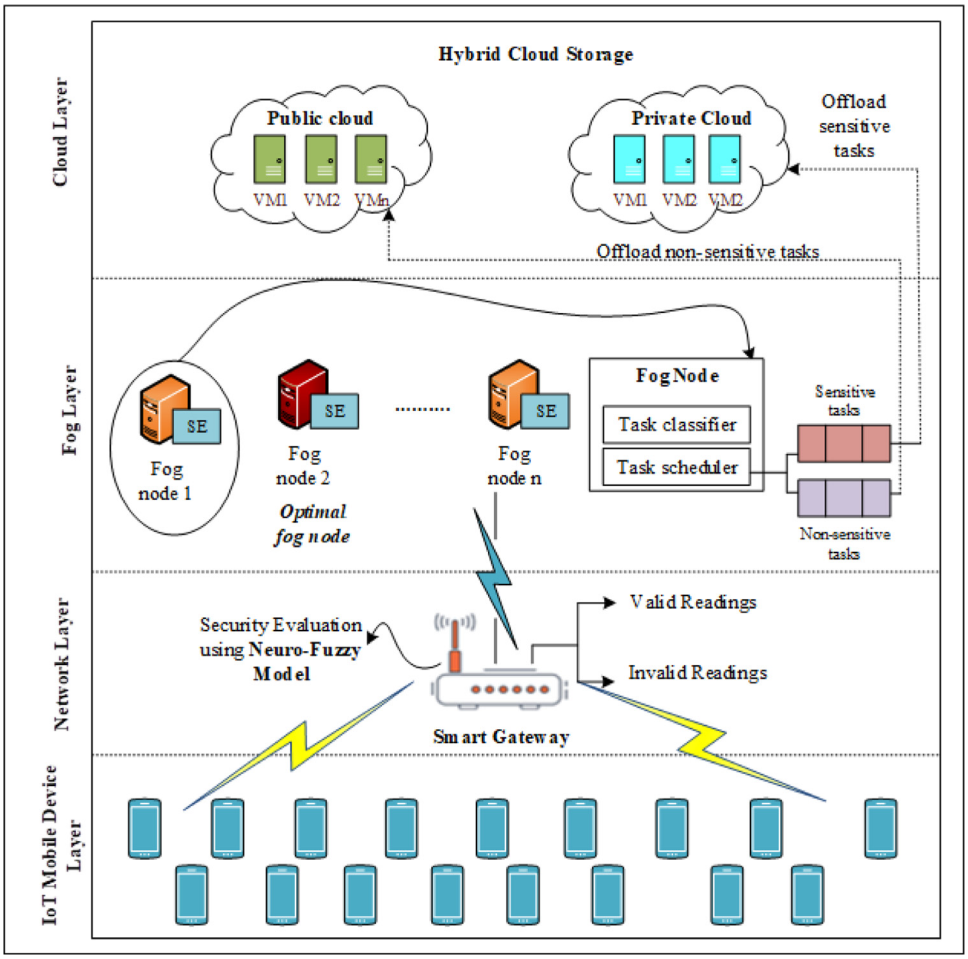

Fig. 1. Proposed system.

4.2.5. *Cloud layer*

The cloud consists of very powerful, state of art centralized infrastructure, which has infinite capacity to realize secure and heavy computation. Cloud infrastructure may be private, public or hybrid. Private cloud allow organizations to run their cloud services based on proprietary architecture, within their own data centers. They are created and maintained by an individual organization. Public clouds allow organizations to run their cloud services based on third-party architecture. With public cloud, vendor aught not to set their own infrastructure. Public clouds are based on multi-tenant architecture and pay-as-you-go pricing model. Users only pay for what is required for their use [44,47].

4.3. *Optimal fog node selection using PSO*

In IoT network, connectivity is achieved through wireless means. In such an environment resources can be dynamic. They are affected by unprecedented factors which include fluctuating bandwidth due to mobility, weather, and physical obstacles along communication path. This makes the workload at the Fog Node change frequently. Thus, an algorithms to update network information more often is required.

As the fog node current workload changes, we run the PSO algorithm [48] to update information used for selecting the optimal Fog node. The criterion for selecting the fog node is used to reduce the total processing delay between IoT mobile devices to the Fog. Optimal fog node is chosen by looking at two metrics: (i) Available Processing Capacity (APC) and (ii) Remaining Node Energy (RNE). Every node will calculate its fitness using these two metrics. When the request is made by the user device, the fog node with high APC and RNE will be chosen.

The PSO algorithm is meta heuristic algorithm which derives its intelligence from swarm. PSO has been used in many applications that exhibit characteristics of swarm similar to behavior of social creatures like birds and fish [49]. We find this algorithm suitable in this study since IoT devices behave the same way a swarm does. The mechanism used in PSO

Table 1
User request task information.

Name	Description
ID_{Task}	Task unique identifier
ApplicationType	The type of the application
Task constraints	Specific constraints of tasks such as latency
$Fognode_{id}$	Optimal fog node within the user communication range

is simple, but powerful. PSO is used in many applications in science and engineering. Examples are found in studies that involve prediction of events [50], network planning [51], alignment optimization [52] etc.

PSO uses two basic principles that is communication and learning. A particle hereafter known as an agent who is a member of a swarm creates an intelligent behaviour which is absolutely unreachable by other agents in the swarm. Thereafter, communicates this information to all agents of the swarm. From information tendered in by each agent, all agents learn the best course of action to take. The agents create a single global optimal knowledge that determines the course of action for the whole swarm. Leveraging this simple cooperation results in a level of intelligence that can be reached by all agents of the swarm [49].

The objective of PSO algorithm is to find the optimal solution by information sharing and cooperation among individuals in a group. Assume that one population comprised of n particles and D dimensional searching space. Each particle changes its position x at time t by

$$x(t+1) = x(t) + v(t+1) \tag{9}$$

$$v(t+1) = \omega v(t) + C_1 R(0,1) * (x_{pbest} - x(t)) + C_2 R(0,1) * (x_{gbest} - x(t) \tag{10}$$

where v(t) is the velocity of the particle at time t, x(t) is the particle position at time t, ω is the inertia weight, C_1 and C_2 are the learning coefficient and acceleration coefficient respectively, R is the random variable range between 0 and 1, x_{pbest} is the particle best position, and x_{gbest} is the global best position. For each particle x_i of the $task_i$, we compute fitness value by using APC, and RNE.
Fitness function:

$$f(x_i) = w_1 APC(x_i) + w_2 RNE \tag{11}$$

where w_1 and w_2 are the two weight factors representing the importance of APC and RNE, respectively. w_1[0.1, 0.9] and $w_2 = 1 - w_1$.

Moving forward, let us consider, sz_i, cx_i, μ_i be the characteristics of $task_i$; where sz_i, cx_i denote the size complexity of the task i, and mean latency respectively. And let bs and F be the buffer the current buffer size and CPU frequency of the Fog node, then we can compute $L(x_i)$, the latency of task x_i as follows;

$$L(x_i) = \frac{sz_i \times cx_i \times \mu_i + bs(x_i)}{F(x_i)} \tag{12}$$

After the optimal node is chosen by user, the request is sent to the fog node. User request consists of information about the tasks to be offloaded to fog nodes shown in Table 1.

At the IoT level, the following proposed scheme is used to choose an optimal fog node to which offloading can be made.

Algorithm 1: offloading scheme at IoT Node.

```
Method OffloadingScheme-IoT():
    for each IoT device do
        if offloadable workload exists then
            select an optimal Fog Node using PSO
            send data through Smart Gateway
        else
            perform local execution on the IoT device
        end
    end
End IoT-Scheme
```

Table 2
Sample of knowledge base for security evaluation.

D_{ID}	SV	T_S	Prediction	output
107	75	4137ms	0.00027	invalid
108	251	1564ms	1.00002	valid
116	0	951ms	0.00030	invalid
115	230	230ms	1.00458	valid
114	245	1117ms	1.00001	valid

4.4. Neuro-Fuzzy model for security evaluation

In IoT-Fog architecture, IoT devices communicate to the upper layers through the gateways. Gateways are responsible for bridging between IoT devices, the Fog, the cloud, and user equipment (smart phone, cyber-physical devices etc.). They provide a communication link, real-time control over the IoT devices, and provide offline services. Gateways can be used to secure data that is being transported to and from the upper layers. Security is achieved by isolating resources that exhibit abnormal behavior. In many occasions, security solutions require considerable amount of processing power which are expensive in terms of energy consumption. Implementing them on IoT devices is infeasible. In our study, we secure the network at using Neuro-Fuzzy network at the smart gateway because the smart gateway houses considerable processing power, and storage as compared to IoT devices. In [53], neuro-fuzzy model has been used in similar setting for purposes of routing to achieve better energy utilization.

Nuero-Fuzzy systems are suitable when we intend to manage parameterized components of a Fuzzy system. They are used to produce systems that deal with parameters that can be tuned through training. Nuero-Fuzzy systems are tools used for prediction and classification problems. They combine characteristics of both Neural networks and fuzzy techniques. Neural network tools bring useful traits of learning, generalization, optimization, and adaptation. Whereas, Fuzzy models bring human like intelligence using IF...THEN rules, Expert knowledge, simplicity in terms of linguistic variable with no mathematical expression to the system [54].

Nuero-Fuzzy systems are broadly categorized into; Neural Fuzzy systems, Fuzzy Neural systems and Hybrid Nuero-Fuzzy systems. In Neural Fuzzy systems, Neural network are used to determine functions and mapping between fuzzy sets that are used as fuzzy rules. They change weights during training so as to minimize mean square error between the actual output and target. In these systems, fuzzification functions, fuzzy word membership, functions and fuzzy rule confidences are represented as weights in neural network, whereas in Fuzzy neural network inputs are non-fuzzy, operations are replaced by membership functions, and aggregation operations such as max, min, t-norm and t-coform are used. Lastly, in hybrid Nuero-Fuzzy systems, each technique is used independently to accomplish a task. Fuzzy rules are interpreted in the of context neural networks while fuzzy sets are interpreted as weights [55,56].

In this work, we employ Nuero-Fuzzy model at the smart gateway to evaluate the data coming from IoT devices. Two factors are considered for security evaluation i.e. Sensor value(Sv), Time (Ts). From these two values predicted value(Pv) is derived. If the predicted value is greater than 1.00, we consider the resource has valid reading otherwise the reading is invalid; therefore the resource is isolated from transaction. We assume that Neuro-fuzzy model consists of N devices. (d_1, d_2,...,d_N). Each input has two parameters that is Sv, and Ts and outputs values, which are either valid and invalid.

The sensor value(Sv) can be Small, Medium or Large. The corresponding time value can be low, moderate, or high. In our model, we consider sensor value (Sv) to be small if data size is less than 100bits, Moderate if the data size is between 100 to 350 bits, and large if the data size is more than 350bits. The corresponding time is considered Low if the Time (Ts) is below 100ms, Moderate if the Time (Ts) is between 100 and 1000 ms and high if Time Ts is greater than 1000ms. For each episode, Sv, Ts, predicted values(Pv), and output are generated. These output values are stored in the knowledge base which acts as bases of experience. Using the knowledge base the Neuro Fuzzy network is trained to adapt to incoming data from the IoT devices. Fuzzy rules are constructed and these are based on experience/knowledge in the domain as follows;

IF Sv is Small AND Ts is High THEN Pv is Invalid.
IF Sv is Small AND Ts is moderate THEN Pv is valid.
IF Sv is Small AND Ts is Low THEN Pv is valid.
IF $Sv$is medium AND Ts is High THEN Pv is valid.
IF Sv is medium AND Ts is moderate THEN Pv is valid.
IF Sv is medium AND Ts is low THEN Pv is valid.
IF Sv is Large AND Ts is High THEN Pv is valid.
IF Sv is Large AND Ts is moderate THEN Pv is valid.
IF Sv is Large AND Ts is low THEN Pv is invalid.

IoT devices are sensed to have invalid data due to incorrect sensor values and incorrect delay (ms). From the obtained results such as valid and invalid predicted values, the data form trusted devices are retained. Fig. 2, illustrates the proposed Neuro-Fuzzy model, and Table 2 shows sample of knowledge base of Neuro-fuzzy model.

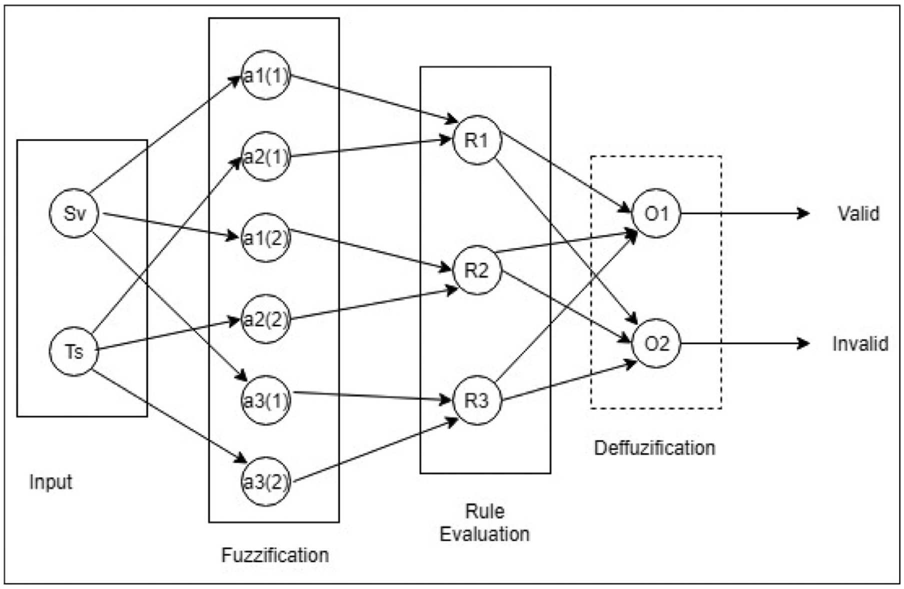

Fig. 2. Proposed Neural-Fuzzy model.

At the gateway, the following proposed scheme is used to secure data coming from IoT device.

Algorithm 2: Data validating scheme at smart gateway.

```
Method validateScheme(ioTScheme-data):
    for data received do
        validate data using Neuro-fuzzy model
        if data is not valid then
            drop the data
            inform IoT to resent
        else
            forward data for processing to selected Fog Node
        end
    end
End validateScheme
```

4.5. Dynamic task offloading using task scheduling

When the fog node cannot process all received tasks within the latency constraints, the fog offload tasks to cloud server for further processing. For this action, a dynamic task offloading is proposed. Dynamic offloading is based on reinforcement learning scheme called Q-learning.

Q-learning is a model-free reinforcement learning mechanism from which the agent learns the best course of action through experiencing the consequences of an action, without necessarily building a map of the domain [57,58]. Learning is achieved through trial an error until set of action taken yields optimal policy. This mechanism is realized by the agent trying an action at a particular state. The agent evaluates the consequences of the action in terms of an immediate scalar reward received and its estimate of future rewards with respect to the state and action which it has taken. By trying all the states repeatedly, the best state of action is learnt. Q-learning is a naive way of learning, but, as such, it forms basis of complicated operations in intelligent systems. It has found many applications in gaming, computing, science, and industrial application [59]. Examples of real world application are found in studies such as, in gaming [60], performance analysis [61], and robotics [62].

Q-learning concept includes state space, action space and reward function. Each state s and action-pair Q(s,a) has a Q-value. If action is selected by the agent located in state s, the Q-value for state-action pair is updated according to the obtained scalar reward using Eq. (14). When choosing an action, the highest Q-value for the subsequent state s' is considered ($\epsilon - greedy$ strategy).

At the fog level, given $task_i$, an action a_i means “choose Virtual Machine (VM_i) from all the existing VM” that meets requirements of $task_i$ for offloading. The task requirements include the type of the server (private or public), The VM that may be used to perform the task within the task constraints (amount of CPU, Memory, latency and Priority attached to the task). Action space a consists of action $a = \{a_1, a_2, \dots, a_i\}$. In addition, available *VM* in the cloud server define the state space. The state space $S_m = \{VM_1, VM_2, \dots, VM_m\}$, each *VM* is characterized by the amount of CPU and memory(*VM*[*CPU*, *MEM*]). The action pair is represented as follows;

$$s = \begin{bmatrix} VM_1, a_1 & \dots & VM_1, a_i \\ \dots & \dots & \dots \\ VM_m, a_1 & \dots & VM_m, a_i \end{bmatrix} \in (S, a) \tag{13}$$

A task is assigned to any virtual machine that meets the latency and resource constraints. To determine optimal action on the current observation of both the server and task requirement, the Fog chooses appropriate cloud based on the current state and reward received from the environment. The goal of the system is to maximize rewards received and minimize latency.

Dynamic task offloading by task scheduling problem is viewed as Markov Decision Process(MDP). Herein the action space is described by a binary vector for each $task_i$. When a current $task_i$ is received by the available VMs, it is represented by 1 otherwise it is represented by 0, then the reward function is computed for state-action pair. The rewards obtained denotes the cloud servers current state (running, waiting, busy etc.). The state action pair rule is shown in (14)

$$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)] \tag{14}$$

where $\alpha \in (0, 1)$ is the learning rate, r is the reward received on taking action a while in state s, and $\gamma \in [0.1, 0.9]$ discount factor. The VM to where offloading is to happen is chosen $\epsilon-$greedily with a small chance of acting randomly, even if the update is done based on highest Q-Value.

The incoming tasks may be sensitive or non sensitive. The sensitive tasks are offloaded to the private cloud servers, non-sensitive tasks are offloaded to the public cloud. Since the type of tasks differ and data processing platform is not the same, a task classifier is employed.

The purpose of the classifier is to classify the in coming tasks in to sensitive S_i and non-sensitive tasks NS_i. Sensitivity of the task is determined according to its size, complexity and latency. During offloading, the task characteristics don't change.

In this study, the task size is scaled between 1 and 100 Kbits, and task complexity is scaled between 10 and 200 cycles per bit. Before offloading the computation tasks to cloud servers, tasks priority are assigned accordingly. Following the above proposed scheme dynamic offloading to either the private or public cloud is accomplished. The scheme is outlined in Algorithm 3

Algorithm 3: Dynamic offloading scheme.

```
Method dynaOffloadingScheme($Task_i$):
    for each $task_i$ do
        Compute execution time
        Classify tasks into $S_i$ and $NS_i$
        Assign Task priority tasks($P_i$) according to task size($sz_i$), characteristics($cx_i$), and mean latency ($\mu_i$)
        if $P_i >$ threshold then
            offload $S_i$ to private server
            offload $NS_i$ to public server
        else
            Perform $S_i$ on Fog
            offload $NS_i$ to public server
        end
        receive results from Cloud
    end
End dynaOffloadingScheme
```

4.6. The general pipeline structure of SecOFF-FCIoT

The general proposed pipeline of Machine learning based secure offloading in Fog Cloud of things environment is shown in Fig. 3. The IoT devices generate workload that may have offloadable content. The IoT device select a most suitable Fog node that can process the request and send back the results with in required threshold. PSO is used at this level as described in Section 4.3. The offloadable content is forwarded through a smart gateway to the Fog. The smart gateway secures the data as described in Section 4.4. The Fog may offload the workload coming from the IoT to the cloud using reinforcement learning mechanism described in Section 4.5. After processing the cloud sends back the results to the IoT through the Fog, and Smart gateway.

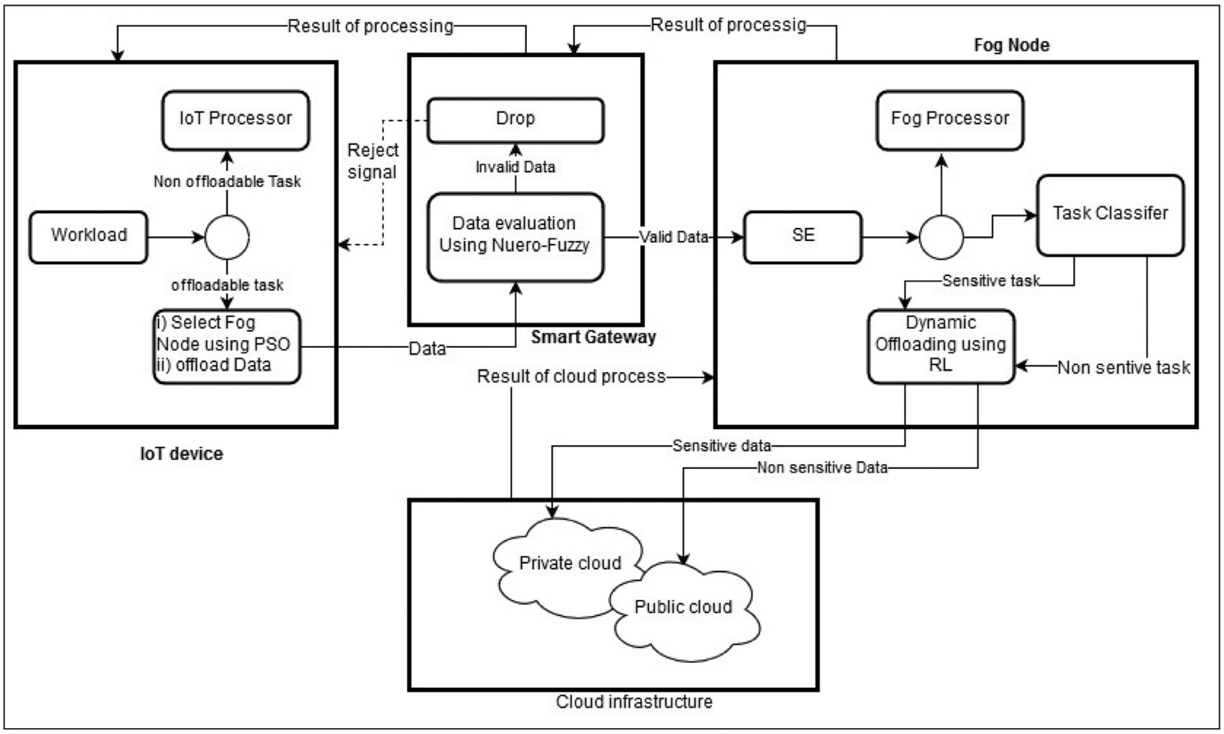

Fig. 3. The general pipeline structure of proposed secure offloading scheme.

5. Experimental setting with comparative study

The performance evaluation analysis and comparative study is presented in this section. First, we present the experiment settings and then provide experiment results. We evaluated the performance in terms of throughput, delay, energy consumption,resource utilization, and response time. Finally, we show our proposed offloading scheme is secure and scaleable.

5.1. Experiment environment and use case

5.1.1. Experiment environment

We create a Fog-Cloud IoT network which, consists of 1 smart gateway, 5–10 IoT mobile devices, 5 fog nodes and 1 hybrid cloud server. We used network simulator (NS3.26) and Java programming in our experiment environment.

The purpose of this configuration is to link the fog-cloud paradigms to the IoT mobile devices. NS-3 is an open source network simulator that supports C++ and/or python packages. Additionally, NS-3 contains Integrated Graphical User Interface (GUI) that is used to visualize performance of simulated network. Further, it includes modules that can be used to simulate network performance metrics such as throughput, energy, delay etc. It also contains modules that can be used to simulate most state of art networks based on 5G, 802.11ah, and 802.11ax standard that support fast and reliable data transmission.

In our experiments, all the simulation parameters have been set to follow uniform distribution. Each device is powered by CPU whose clock frequency ranges from 1 GHz to 1.5 GHz. The clock frequencies are set randomly. We also set available bandwidth between mobile devices to range between 100 Kb/s to 1000 Kb/s. The computing offloading require CPU cycles and tasks to be offloaded in bits. Computational tasks are categorized into complex and non complex. In order to characterize the offloading task complexity, we used load-input data ratio (LDR).[7] When LDR is high, the task is classified to be computationally-intensive otherwise the task is not. Non-computationally intensive tasks can be executed at local device or edge.

The system configuration of this experiment is shown in Table 3.

5.1.2. Use case

The proposed dynamic offloading framework with security evaluation has been simulated for realistic application scenario in smart city. The smart city concept shows the need of quality of service and experience, improved connectivity and

[7] LDR is the ratio of input computational size and computational workload. Computational workload is measured in terms of CPU cycles required to complete a task on a device.

Table 3
System configuration.

Name	Description
Simulation tool	NS-3.26
Development toolkit	JDK-1.8
Operating System	Ubuntu 14.04 LTS
Development platform	Netbeans 8.0
Processor	Pentium (R) Dual-Core CPU E5700@3.00 GHz
Installed memory	2 GB RAM

performance to realize several urban services. With the use of Information and Communication Technologies (ICT), Smart City application explores cloud-based and IoT based services for users in real-world through smart phones. Today, cloud technology is and will continue to be the backbone infrastructure for smart cities around intelligent transportation, public safety, public health and air quality monitoring programs. Cloud offers a broad set of services including storage, processing, compute, analytics, databases, networking, etc. Users can use these cloud based services to construct secure, agile and cost-effective solutions. Broadly speaking, smart city application can be categorized into: Smart Infrastructure Monitoring, Smart water monitoring and management, Smart Building and Property, Smart city Services,Smart Energy Management, and Smart Industrial Environment [63,64].

5.2. Performance metrics

In this work, the following performance metrics have been used for comparative analysis

i. Throughput (R_T): It is defined by the number of tasks offloaded per unit of time T.

$$R_T = \frac{\# - task\ Offloaded}{T} \tag{15}$$

ii. Delay (T_d): Time duration for a task of the application is submitted and its results are obtained. It is also computed as follows:

$$T_d = T_{pro} + T_q + T_t + T_p \tag{16}$$

where T_{pro} denotes processing delay, T_q denotes queuing delay, T_t represents transmission delay, and T_p denotes propagation delay.

iii. Energy consumption (E_t): It is the amount of energy consumed by IoT mobile devices to perform given task.

$$E_t = E_p + E_t \tag{17}$$

where E_p is energy consumed during processing offloading a task, E_t is energy consumed during transmission and receiving the result of the task.

iv. Resource utilization rate (RU): is defined as the total amount of resources consumed as compared against the amount of resources estimated. RU is expressed as the percentage of time mobile device utilize resources in 24 h.

$$RU = \left(\frac{N(i)}{24}\right) \times 100 \tag{18}$$

v. Response time (T_R): Time interval between a user request and the reception of an action.

5.3. Comparison analysis

In this subsection, we present comparative analysis of our offloading schemes to other existing solutions in [37,38,65,66] listed in Table 5. Our proposed offloading provide scaleable and flexible solution for IoT users. In the following subsections we illustrate our results.

5.3.1. Impact on throughput

Fig. 4 shows the impact on throughput for proposed compared to DTO-SO in [38] with respect to number of requests. We set latency requirements for sensitive tasks to be 1 s and 1.5 s for non-sensitive tasks. We set number of tasks $n \in \{10, 20, 30, 40, 50\}$. We observe throughput as the number of requests increase.

Throughput increases with respect to the number of tasks transmitted by IoT device. At $n = 10$, throughput attained by our proposal is 30 KB/s for sensitive tasks, and 23 KB/s for non-sensitive task. DTO-SO attains 18 KB/s for both sensitive and non-sensitive for the same number of tasks transmitted. At $n = 50$, throughput attained by our proposal is 120 KB/s for sensitive tasks and 100 KB/s for non-sensitive task whereas, DTO-SO attains 95 KB/s for sensitive, and 80 KB/s for non-sensitive. Our proposed secure offloading scheme improves throughput by 23.2% as compared to DTO-SO. The increase of throughput is due to pipeline of machine learning offloading strategies that include PSO at the IoT nodes and dynamic offloading using Reinforcement learning at Fog node. Hence, the proposed offloading strategy is suitable for both complex and simple task computation offloading.

Table 4
Simulation measurements.

Simulation Parameters	Values
Number of nodes	10 IoT devices
Number of fog nodes	5
Number of cloud server	1 (Hybrid cloud)
Number of simulated tasks	"10, 20,30,40,50"
Number of Smart Gateway	1
Simulation area	1000 m x 1000 m
Task arrival rate	"[0-5]"
Simulation time	100 s
Initial energy of a node	5 J
Traffic type	CBR
Packet interval	0.1 s
Learning rate	0.2

Table 5
Showing benchmarks used in this study.

Reference	Target device	Major contribution	Drawbacks	Application for IoT
Zhang et al. [37]	Mobile devices	Computation offloading using PSO (FCFS) scheduling.	High latency and hybrid computation offloading increases network overhead	Smart home automation.
Gnana Jeevan et al. [38]	Mobile devices,	Dynamic task offloading using surrogate object model.	Consumes more energy and lack of security and privacy.	Mobile applications.
Tongxiang Wang et al. [65]	Mobile devices,	Cooperative multi-tasks scheduling using ACO.	Static scheduling poor resource utilization rate and not suitable for sensitive tasks	Mobile applications.
Yucen Nan et al. [66]	General IoT devices,	Lyapunov optimization for application offloading based on time and energy cost model.	High energy consumption and high response time.	Green energy consumption application.

5.3.2. Impact on delay

Fig. 5 shows the delay per user requests for both sensitive and non-sensitive tasks.

The delay of total offloading increases linearly with respect to number of requests. Delay is reduced by 20% and 60% as compared with DTO-SO and FCFS offloading approaches. In particular, considering sensitive tasks, The delay registered in our proposed scheme is up to 1.6s when $n = 50$. Whereas, the delay in DTO-SO is 1.8 s and FCFS is 2.2 s for the same amount of requests. Our proposed offloading scheme reduces delay because of optimal selection of fog node using PSO, and security evaluation at smart gateway does not increase latency. This in turn reduces time required to respond to offloading. Furthermore, task scheduling for sensitive (complex) tasks and non-sensitive tasks at the Fog is seen to reduce delay.

5.3.3. Impact on energy consumption

Energy utilization is one of the primary constraints in IoT environment. Executing complex tasks attracts massive usage of battery thereby compromising the device life. Migrating computationally intensive tasks and complex tasks by offloading such a task to the Fog and cloud server save energy and improve device lifetime. Fig. 6 shows energy consumption of the proposed secure offloading vs. previous approaches LOTEC [66] and DTO-SO [38] with respect to number of requests. The energy consumption of total offloading tasks at mobile device is linear. We observe that the energy consumption during processing non-sensitive tasks is low (0.11 J), which is 15% less than DTO-SO and 35% less than LOTEC. In previous approaches, the energy consumption rate increases gradually from $n = 10$ to $n = 50$ with higher gradient. This is because the delay sensitive and complex tasks consume more energy than non-sensitive tasks. Our proposed offloading scheme reduces energy consumption by the use of surrogate entity which decouples IoT devices from the burden of staying connected during processing of the offloaded work. Secondly, surrogate entity assigns cloud resources to device quickly thus reducing the energy required to maintaining computational offloading.

5.3.4. Impact on resource utilization rate

Fig. 7 shows resource utilization rate for proposed vs. previous approaches CMS-ACO [65], and DTO-SO [38]. Heterogeneous IoT devices request processing increases overhead and thus resource utilization rate is decreased. The proposed offloading scheme can handle real-life data, and monitoring of smart IoT devices. Thus it delivers highly scaleable sensing

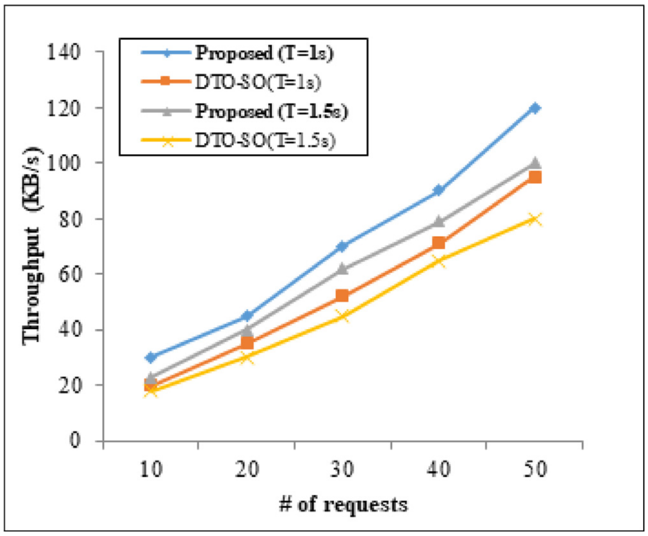

Fig. 4. Impact on throughput.

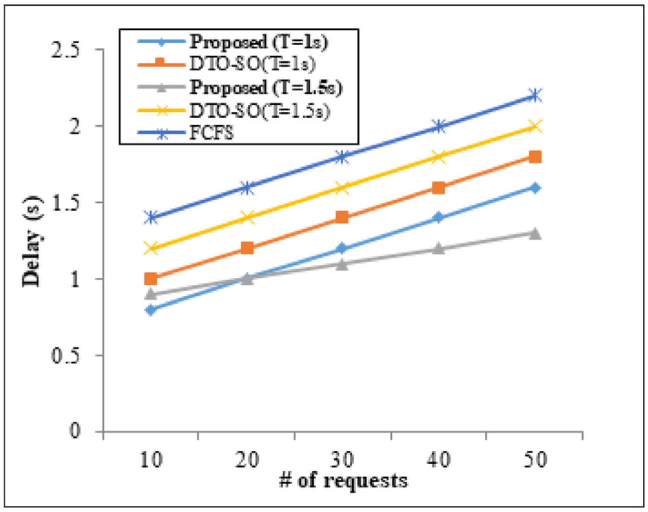

Fig. 5. Impact on delay.

information and also security is evaluated before offloading. When $n = 10$, the resource utilization rate for proposed offloading scheme is to 92% at T = 1 s while the previous approaches utilizes 89%, and 87% for DTO-SO and CMS-ACO, respectively. In previous approaches, offloading tasks of certain resource-restrained devices may not increase resource utilization rate. This reveals the trade-off between energy consumption and delay in fog-cloud IoT environment. Our proposed offloading scheme is feasible in any resource-constrained environment.

5.3.5. Impact on response time

Fig. 8 shows the result of response time for proposed offloading scheme as compared with two previous approaches LOTEC [66] and DTO-SO [38]. The response time grows gradually In LOTEC and DTO-SO. Our proposed offloading takes 0.5s to complete a task as compared to 0.9s in LOTEC, and 0.7s in DTO-SO. Therefore, our proposed scheme is faster in terms of response.

Finally, we present average response time (a_{RT}) and average execution time (a_{ET}) of all the tasks for secure offloading and conventional offloading schemes presented in Table 6.

The results show that our proposed system achieves response time of 184.54(ms) as compared to 285.64(ms) for DTO-SO, 295.45(ms) for CMS-ACO, 305.78(ms) for LOTEC and 312.46(ms) for FCFS. At the same time execution time registered for our proposal is 956.23(ms) as compared to 1670.46(ms), 1700.23(ms), 1750.56(ms) and 1790.9(ms) for DTO-SO, CMS-ACO, LOTEC and FCFS. All experiments are done under similar environment.

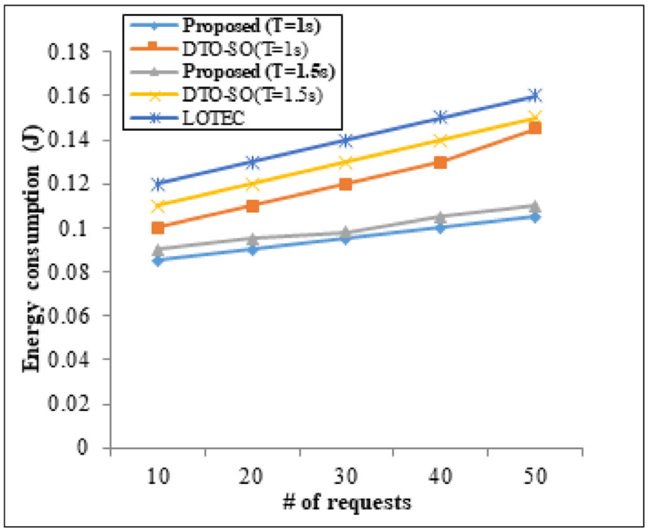

Fig. 6. Impact on energy consumption.

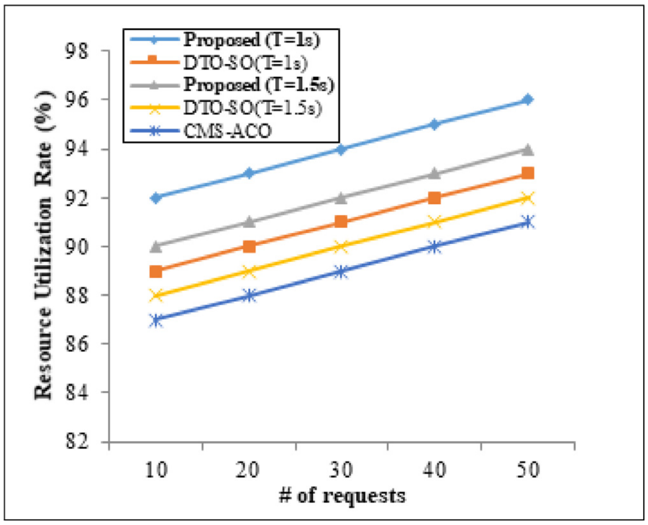

Fig. 7. Impact on resource utilization rate.

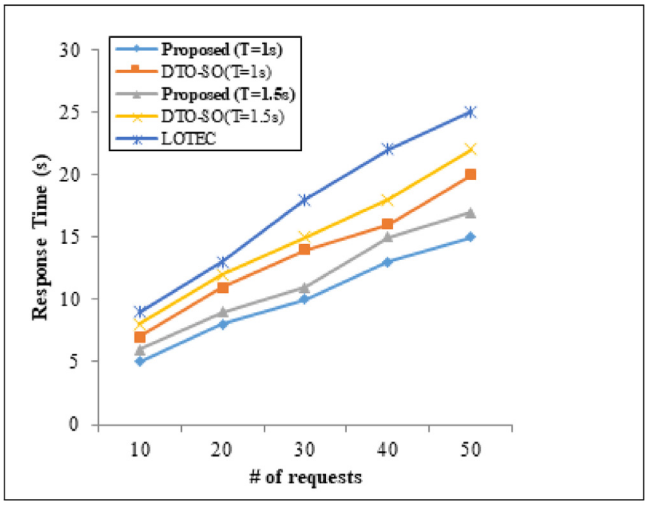

Fig. 8. Impact on response time.

Table 6
Average response time and average execution time.

Metrics	State-of-the-art on offloading approaches				
	DTO-SO	CMS-ACO	LOTEC	FCFS	Proposed
$a_{RT}(ms)$	285.64	295.45	305.78	312.46	184.56
$a_{ET}(ms)$	1670.46	1700.23	1750.56	1790.9	956.23

The performance of various metrics vary depending on user request (task size, delay constraint and complexity). In all the Figs. 4–8, graphical results were presented. Our results confirm that the proposed offloading scheme reduces delay and energy consumption than DTO-SO, FCFS, LOTEC, and CMS-ACO.

6. Conclusions

In this paper, we proposed a secure offloading to minimize latency and energy consumption. The proposed layered system architecture for offloading scheme is secure and effective for balancing the trade off between latency and energy consumption. The Neuro-Fuzzy model is proposed to eliminate the invalid resources and also optimal fog node is chosen by PSO. The results of our implementation show that the delay is marginal and has negligible energy consumption. This makes the proposed approach robust.

In future we conduct simulation for large scale environment and also build cluster based network to further reduce energy consumption in IoT mobile devices. Furthermore, introducing new technologies such as blockchain in decentralizing security, privacy and trust issues at all levels of continuum from IoT devices to the cloud in addition to offloading could be challenging but interesting.

Declaration of interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

We would like to thank the reviewers of this article for their constructive comment that enabled us to improve our work.

References

[1] C.-P. Lopez, M. Santórum, J. Aguilar, Autonomous cycles of collaborative processes for integration based on industry 4.0, in: Proceedings of the International Conference on Information Technology & Systems, Springer, 2019, pp. 177–186.
[2] Cisco, Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 20162021 White Paper - Cisco. URL https://www.cisco.com/c/en/us/solutions/700collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html.
[3] M. Min, D. Xu, L. Xiao, Y. Tang, D. Wu, in: Learning-based computation offloading for IoT devices with energy harvesting, 2017. arXiv: 1712.08768
[4] A. Munir, P. Kansakar, S.U. Khan, IFCIoT: Integrated fog cloud IoT: a novel architectural paradigm for the future Internet of Things, IEEE Consum. Electron. Mag. 6 (3) (2017) 74–82.
[5] V. Cozzolino, A.Y. Ding, J. Ott, D. Kutscher, Enabling fine-grained edge offloading for IoT, in: Proceedings of the SIGCOMM Posters and Demos, ACM, 2017, pp. 124–126.
[6] B. Di Martino, M. Rak, M. Ficco, A. Esposito, S. Maisto, S. Nacchia, Internet of Things reference architectures, security and interoperability: a survey, Internet Things 1 (2018) 99–112.
[7] T. Stack, Internet of Things (IoT) Data Continues to Explode Exponentially. Who Is Using That Data and How? (2018). URL https://blogs.cisco.com/datacenter/internet-of-things-iot-data-continues-to-explode-exponentially-who-is-using-that-data-and-how.
[8] M. Aazam, I. Khan, A.A. Alsaffar, E.-N. Huh, Cloud of things: integrating Internet of Things and cloud computing and the issues involved, in: Proceedings of the Eleventh International Bhurban Conference on Applied Sciences & Technology (IBCAST) Islamabad, Pakistan, IEEE, 2014, pp. 414–419. 14th-18th January, 2014
[9] J. Zhou, T. Leppanen, E. Harjula, M. Ylianttila, T. Ojala, C. Yu, H. Jin, L.T. Yang, Cloudthings: a common architecture for integrating the Internet of Things with cloud computing, in: Proceedings of the IEEE Seventeenth International Conference on Computer Supported Cooperative Work In Design (CSCWD), IEEE, 2013, pp. 651–657.
[10] S. Kaushik, C. Gandhi, Fog vs. cloud computing architecture, in: Advancing Consumer-Centric Fog Computing Architectures, IGI Global, 2019, pp. 87–110.
[11] D.H. Tran, N.H. Tran, C. Pham, S.A. Kazmi, E.-N. Huh, C.S. Hong, Oaas: offload as a service in fog networks, Computing 99 (11) (2017) 1081–1104.
[12] S. El Kafhali, K. Salah, Efficient and dynamic scaling of fog nodes for IoT devices, J. Supercomput. 73 (12) (2017) 5261–5284.
[13] R. Roman, J. Lopez, M. Mambo, Mobile edge computing, fog et al.: a survey and analysis of security threats and challenges, Future Gener. Comput. Syst. 78 (2018) 680–698.
[14] Y. Son, J. Jeong, Y. Lee, An adaptive offloading method for an IoT-cloud converged virtual machine system using a hybrid deep neural network, Sustainability 10 (11) (2018) 3955.
[15] L. Xiao, X. Wan, X. Lu, Y. Zhang, D. Wu, IoT security techniques based on machine learning: how do IoT devices use AI to enhance security? IEEE Signal Process. Mag. 35 (5) (2018) 41–49.
[16] D. Van Le, C.-K. Tham, A deep reinforcement learning based offloading scheme in ad-hoc mobile clouds, in: Proceedings of the IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), IEEE, 2018, pp. 760–765.
[17] S.B. Kotsiantis, I. Zaharakis, P. Pintelas, Supervised machine learning: a review of classification techniques, Emerg. Artif. Intell. Appl. Comput. Eng. 160 (2007) 3–24.
[18] R. Carbonneau, K. Laframboise, R. Vahidov, Application of machine learning techniques for supply chain demand forecasting, Eur. J. Oper. Res. 184 (3) (2008) 1140–1154.

[19] K. Kourou, T.P. Exarchos, K.P. Exarchos, M.V. Karamouzis, D.I. Fotiadis, Machine learning applications in cancer prognosis and prediction, Comput. Struct. Biotechnol. J. 13 (2015) 8–17.
[20] G.I. Webb, M.J. Pazzani, D. Billsus, Machine learning for user modeling, User Model. User-Adapted Interact. 11 (1–2) (2001) 19–29.
[21] T.T. Nguyen, G.J. Armitage, A survey of techniques for internet traffic classification using machine learning., IEEE Commun. Surv. Tutor. 10 (1–4) (2008) 56–76.
[22] J.M. Mulvey, Machine learning and financial planning, IEEE Potentials 36 (6) (2017) 8–13.
[23] R.C. Deo, Machine learning in medicine, Circulation 132 (20) (2015) 1920–1930.
[24] G. D'Angelo, S. Rampone, Diagnosis of aerospace structure defects by a HPCs implemented soft computing algorithm, in: Proceedings of the IEEE Metrology for Aerospace (MetroAeroSpace), IEEE, 2014, pp. 408–412.
[25] G. D'Angelo, S. Rampone, Feature extraction and soft computing methods for aerospace structure defect classification, Measurement 85 (2016) 192–209.
[26] T.A. Estlin, R.J. Mooney, Learning to improve both efficiency and quality of planning, in: Proceedings of the IJCAI, 1997, pp. 1227–1233.
[27] S. Yu, X. Wang, R. Langar, Computation offloading for mobile edge computing: a deep learning approach, in: Proceedings of the IEEE Twenty-Eighth Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC), IEEE, 2017, pp. 1–6.
[28] K. Kumar, J. Liu, Y.-H. Lu, B. Bhargava, A survey of computation offloading for mobile systems, Mob. Netw. Appl. 18 (1) (2013) 129–140.
[29] F. Bonomi, R. Milito, J. Zhu, S. Addepalli, Fog computing and its role in the internet of things, in: Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, ACM, 2012, pp. 13–16.
[30] P. Hu, S. Dhelim, H. Ning, T. Qiu, Survey on fog computing: architecture, key technologies, applications and open issues, J. Netw. Comput. Appl. 98 (2017) 27–42.
[31] Z. Kuang, S. Guo, J. Liu, Y. Yang, A quick-response framework for multi-user computation offloading in mobile cloud computing, Future Gener. Comput. Syst. 81 (2018) 166–176.
[32] F. Zhang, J. Ge, Z. Li, C. Li, Z. Huang, L. Kong, B. Luo, Task offloading for scientific workflow application in mobile cloud, in: Proceedings of the Second International Conference on Internet of Things, Big Data and Security (IoTBDS 2017), 2017, pp. 136–148.
[33] D. Kozyrev, A. Ometov, D. Moltchanov, V. Rykov, D. Efrosinin, T. Milovanova, S. Andreev, Y. Koucheryavy, Mobility-centric analysis of communication offloading for heterogeneous internet of things devices, Wirel. Commun. Mob. Comput. 2018 (2018) 1–11.
[34] H. Flores, X. Su, V. Kostakos, A.Y. Ding, P. Nurmi, S. Tarkoma, P. Hui, Y. Li, Large-scale offloading in the Internet of Things, in: Proceedings of the IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), IEEE, 2017, pp. 479–484.
[35] X. Ma, C. Lin, H. Zhang, J. Liu, Energy-aware computation offloading of IoT sensors in cloudlet-based mobile edge computing, Sensors 18 (6) (2018) 1945.
[36] S. Kim, Nested game-based computation offloading scheme for mobile cloud IoT systems, EURASIP J. Wirel. Commun. Netw. 2015 (1) (2015) 229.
[37] J. Zhang, Z. Zhou, S. Li, L. Gan, X. Zhang, L. Qi, X. Xu, W. Dou, Hybrid computation offloading for smart home automation in mobile cloud computing, Pers. Ubiquitous Comput. 22 (1) (2018) 121–134.
[38] A.G. Jeevan, M.M. Mohamed, Dyto: Dynamic task offloading strategy for mobile cloud computing using surrogate object model, Int. J. Parallel Program. (2018) 1–17, doi:10.1007/s10766-018-0563-0.
[39] G. D'Angelo, S. Rampone, Cognitive distributed application area networks, in: Security and Resilience in Intelligent Data-Centric Systems and Communication Networks, Elsevier, 2018, pp. 193–214.
[40] J. Caminha, A. Perkusich, M. Perkusich, A smart trust management method to detect on-off attacks in the Internet of Things, Secur. Commun. Netw. 2018 (2018) 1–10.
[41] Y. Zhang, J. Zhao, D. Zheng, K. Deng, F. Ren, X. Zheng, J. Shu, Privacy-preserving data aggregation against false data injection attacks in fog computing, Sensors 18 (8) (2018) 2659.
[42] M. Suárez-Albela, T.M. Fernández-Caramés, P. Fraga-Lamas, L. Castedo, A practical evaluation of a high-security energy-efficient gateway for IoT fog computing applications, Sensors 17 (9) (2017) 1978.
[43] L. Belem Pacheco, E. Pelinson Alchieri, P. Mendez Barreto, Device-based security to improve user privacy in the Internet of Things, Sensors 18 (8) (2018) 2664.
[44] Zahoor, S., Mir, R.N. Resource management in pervasive Internet of Things: A survey. Journal of King Saud University – Computer and Information Sciences (2018), https://doi.org/10.1016/j.jksuci.2018.08.014.
[45] H. Atlam, R. Walters, G. Wills, Fog computing and the Internet of Things: a review, Big Data Cognit. Comput. 2 (2) (2018) 10.
[46] M. Ficco, C. Esposito, Y. Xiang, F. Palmieri, Pseudo-dynamic testing of realistic edge-fog cloud ecosystems, IEEE Commun. Mag. 55 (11) (2017) 98–104.
[47] B.P. Rimal, E. Choi, I. Lumb, A taxonomy and survey of cloud computing systems, in: Proceedings of the Fifth International Joint Conference on INC, IMS and IDC, IEEE, 2009, pp. 44–51.
[48] Y. Shi, R.C. Eberhart, Empirical study of particle swarm optimization, in: Proceedings of the Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), 3, IEEE, 1999, pp. 1945–1950.
[49] Y. Shi, et al., Particle swarm optimization: developments, applications and resources, in: Proceedings of the Congress on Evolutionary Computation (IEEE Cat. No. 01TH8546), 1, IEEE, 2001, pp. 81–86.
[50] A.A. Alnaqi, H. Moayedi, A. Shahsavar, T.K. Nguyen, Prediction of energetic performance of a building integrated photovoltaic/thermal system thorough artificial neural network and hybrid particle swarm optimization models, Energy Convers. Manag. 183 (2019) 137–148.
[51] X. Ye, B. Chen, L. Jing, B. Zhang, Y. Liu, Multi-agent hybrid particle swarm optimization (MAHPSO) for wastewater treatment network planning, J. Environ. Manag. 234 (2019) 525–536.
[52] H. Pu, T. Song, P. Schonfeld, W. Li, H. Zhang, J. Hu, X. Peng, J. Wang, Mountain railway alignment optimization using stepwise & hybrid particle swarm optimization incorporating genetic operators, Appl. Soft Comput. 76 (2019) 41–57.
[53] K. Thangaramya, K. Kulothungan, R. Logambigai, M. Selvi, S. Ganapathy, A. Kannan, Energy aware cluster and Neuro-Fuzzy based routing algorithm for wireless sensor networks in IoT, Comput. Netw. 151 (2019) 211–223.
[54] G.S. Atsalakis, K.P. Valavanis, Surveying stock market forecasting techniques–part II: soft computing methods, Expert Syst. Appl. 36 (3) (2009) 5932–5941.
[55] N. Walia, H. Singh, A. Sharma, ANFIS: adaptive Neuro-Fuzzy inference system-a survey, Int. J. Comput. Appl. 123 (13) (2015).
[56] J. Vieira, F.M. Dias, A. Mota, Neuro-Fuzzy systems: a survey, in: Proceedings of the Fifth WSEAS NNA International Conference on Neural Networks and Applications, Udine, Italia, 2004, pp. 87–92.
[57] C.J. Watkins, P. Dayan, Q-learning, Mach. Learn. 8 (3–4) (1992) 279–292.
[58] H. Van Hasselt, A. Guez, D. Silver, Deep reinforcement learning with double q-learning, in: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, 13, 2016, pp. 2094–2100. 7
[59] S. Lange, M. Riedmiller, A. Voigtländer, Autonomous reinforcement learning on raw visual input data in a real world application, in: Proceedings of the International Joint Conference on Neural Networks (IJCNN), IEEE, 2012, pp. 1–8.
[60] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller, in: Playing Atari with deep reinforcement learning, 2013. arXiv:1312.5602
[61] R.H. Crites, A.G. Barto, Improving elevator performance using reinforcement learning, in: Proceedings of the Advances in Neural Information Processing Systems, 1996, pp. 1017–1023.
[62] P. Kormushev, S. Calinon, D. Caldwell, Reinforcement learning in robotics: applications and real-world challenges, Robotics 2 (3) (2013) 122–148.
[63] K. Su, J. Li, H. Fu, Smart city and the applications, in: Proceedings of the International Conference on Electronics, Communications and Control (ICECC), IEEE, 2011, pp. 1028–1031.

[64] J. Jin, J. Gubbi, S. Marusic, M. Palaniswami, An information framework for creating a smart city through internet of things, IEEE Internet Things J. 1 (2) (2014) 112–121.
[65] T. Wang, X. Wei, C. Tang, J. Fan, Efficient multi-tasks scheduling algorithm in mobile cloud computing with time constraints, Peer-to-Peer Netw. Appl. 11 (4) (2018) 793–807.
[66] Y. Nan, W. Li, W. Bao, F.C. Delicato, P.F. Pires, Y. Dou, A.Y. Zomaya, Adaptive energy-aware computation offloading for cloud of things systems, IEEE Access 5 (2017) 23947–23957.