I. J. Computer Network and Information Security, **2021, 3, 1-14**
Published Online June 2021 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2021.03.01

Distributed Intelligence Model for IoT Applications Based on Neural Networks

Baha Rababah
Department of Computer Science, University of Manitoba, Winnipeg, R3T 2N2, Canada
E-mail: baha@cs.umanitoba.ca

Rasit Eskicioglu
Department of Computer Science, University of Manitoba, Winnipeg, R3T 2N2, Canada
E-mail: rasit.eskicioglu@cs.umanitoba.ca

Received: 02 March 2021; Revised: 24 March 2021; Accepted: 15 April 2021; Published: 08 June 2021

Abstract: Increasing the implication of IoT data puts a focus on extracting the knowledge from sensors' raw data. The management of sensors' data is inefficient with current solutions, as studies have generally focused on either providing cloud-based IoT solutions or inefficient predefined rules. Cloud-based IoT solutions have problems with latency, availability, security and privacy, and power consumption. Therefore, Providing IoT gateways with relevant intelligence is essential for gaining knowledge from raw data to make the decision of whether to actuate or offload tasks to the cloud. This work proposes a model that provides an IoT gateway with the intelligence needed to extract the knowledge from sensors' data in order to make the decision locally without needing to send all raw data to the cloud over the Internet. This speeds up decisions and actions for real-time data and overcomes the limitations of cloud-based IoT solutions. When the gateway is unable to process a task locally, the data and task are offloaded to the cloud.

Index Terms: Internet of Things, Distributed Intelligence, IoT Gateway, Machine Learning, and Neural Networks.

1. Introduction

Recently, the Internet of Things (IoT) has become very popular as Radio Frequency Identification (RFID), wireless networks, Bluetooth Low Energy (BLE), and sensing and actuating technologies have evolved. It is anticipated that 8.4 billion smart things will be associated with the Internet by 2022, and the number of machine-to-machine (M2M) connections is anticipated to reach 27 billion by 2024 [1]. Enabling the IoT has many challenges that cannot be ignored related to its availability, scalability, energy efficiency, security and privacy, and reliability [2, 3, 4]. Although there are many proposed research that tried to figure out those challenges, most of them are cloud-based and are not the best solution because they do not deal with real-time data [5].

Cloud computing means using distant servers to handle and process data instead of local servers. Recently, researchers have started to look at edge computing [6, 7], as it is closer to smart devices than the cloud. Edge computing enables technologies to allow computing to be done at the edge of the network, near the smart devices. Processing data near the smart devices will help to solve problems related to latency, security and privacy, and power consumption.

Recently, some of the solutions, such as Mozilla gateway [8], have employed rules-based intelligence in edge computing. Conversely, rules-based intelligence does not scale well with the requirement of IoT applications [9], as it needs to define plenty of rules to manage a large number of things. Such intelligence also cannot deal with uncertain events because it only offers pre-assumed intelligence [10].

In fact, many scholars have proposed combining IoT and the cloud in one model that has three layers: an end devices layer, an edge layer, and a cloud layer. At the bottom of this model, the end devices layer has sensors and actuators. Those devices are heterogeneous with regard to power consumption, communication capabilities, and processing capabilities. It can be a battery-operated sensor, a tracking device, or a smartphone.

In general, sensors collect data from the surrounding physical environment and communicate this data to the edge layer for processing. After that, the data is transferred over a local network, such as a wireless network, to the cloud layer. Along the way, the information crosses the edge layer components, such as a gateway, an edge router, or an access point. These components usually perform protocol transactions.

The data is then transmitted over the Internet to the cloud layer. The cloud layer consists of a group of connected, powerful computers that are able to process vast amounts of data. The cloud then processes the received data, enhances it, and integrates it with other information in order to convert it either into knowledge that needs to be stored or into an action that must be taken in the real world.

The action is subsequently passed to the end devices layer through the edge layer until it reaches a certain actuator.

For example, to turn on the heating system in a smart home, sensors collect data about the temperature, time, and motion in the home. It then sends this data to the cloud. The cloud processes the received data, makes a decision either to turn the heating system on or off, and it then tells the smart home application what decision was made.

However, this method of transferring all the generated IoT data to the cloud and returning the decision is inefficient. It also causes problems with latency, availability, security and privacy, and power consumption. Current network architectures and technologies are not sufficient to transfer the increasing amount of IoT data to the cloud and returning information about decisions.

Recently, scholars have been investigating the viability of processing data close to the end devices layer in order to avoid the aforementioned challenges. Many publications have suggested building an IoT gateway in the edge layer that is capable of processing data and making a decision without transferring the data to the cloud [5, 7]. Constrained devices (e.g., Arduino and Raspberry Pi) have acceptable processing capabilities that enable them to act as IoT gateways within the edge layer to process raw data, make a decision, and perform an action. However, constrained devices do not have the same computational power as the cloud, and so the cloud cannot yet be ignored.

The current vision is to integrate people, things, services, and context information [11]. In order to attain this vision, new IoT models are needed. These models should consider real-time data and make quick decisions that can be acted upon by enabling edge computing, as this would provide fast processing, energy efficiency, security, mobility, and heterogeneity.

It would be beneficial for edge computing to be provided with the required intelligence to deal with an uncertain event. On the other hand, cloud computing cannot be omitted since edge computing has limited processing and storage capabilities. Cloud computing has the advantages of a much larger storage space and better processing and data analysis capabilities [12]. Therefore, integrating edge and cloud into one model is required to include the beneficial features of both into one system to reinforce IoT applications.

This work presents a distributed intelligence model (DIM) that integrates the features of edge and cloud layers. Distributed intelligence (DI) in IoT is a paradigm that uses models, techniques, and algorithms to make decisions about whether to process IoT data in the edge layer, cloud layer, or both. In this way, DI provides the desired functionality and performance for IoT applications.

The proposed model enables an IoT gateway to extract most of the knowledge from the sensors' data and make a decision locally without sending the data to the cloud. Processing and decision-making tasks are done at the IoT gateway instead of in the cloud, thus solving the problem of latency for real-time applications. In cases when the gateway cannot reliably process data and make a decision because it is overloaded or the task is too complex, it offloads the task to the cloud.

The model will be implemented using IBM Cloud and Raspberry Pi as a gateway, sensors, and actuators. The intelligence provided to the gateway will be based on artificial neural networks (ANNs) that will let it take necessary actions after processing the raw data.

In order to validate the effectiveness of the presented model, a smart home application will be implemented to verify the model. For example, when controlling the temperature in a smart home, the sensors will collect data from temperature sensors and motion sensors at the home and then send them to the IoT gateway at the edge layer. The gateway will process the data, make a decision to turn the air condition system either on or off based on neural networks algorithm, and send the decision back to the air conditioning system at the home. However, if the gateway is overloaded and unable to process the data, it will forward the task to the cloud layer.

The rest of the paper is presented as follows. Section 2 discusses the related work about IoT architecture and bringing the intelligence to the edge layer. It also provides some proposed work for intelligent IoT systems. Section 3 introduces distributed intelligence with IoT. Section 4 describes the proposed DIM in a smart home as a case study that integrates cloud and edge in order to support IoT-based smart home applications. Section 5 implements the proposed model. Section 6 provides and discusses the experimental results. Finally, section 7 concludes the work and presents possible directions for future work.

2. Related Work

Nowadays, extracting knowledge from collected raw data is one of IoT challenges. A large amount of work is being performed in the field of enabling edge computing to process raw data generated and making the decision. Some of the work focus on enabling an IoT gateway at the edge of the network to process and manage the raw data. Mueller et al. [6] introduce a SwissQM/SwissGate system to program, deploy, and operate wireless sensor networks. They propose a gateway called SwissGate and apply it on smart home applications. Jong-Wang et al. [7] also came up with a sensor network system that consists of one main server and a number of servers acting as gateways and connecting with different sensor networks. Designing such system requires a lot of configurations and high hardware cost.

Many works have tried to counter the intelligence challenges at edge computing. For example, Badlani et al. [13] introduce smart home systems based on artificial neural network. It aims to reduce power consumption through analyzing the human behavior pattern. They trained a single perceptron network using random values of temperature and humidity to control the fan. In order to help disabled persons, Hussein et al. [14] came up with a self-adapting intelligence home system that helps disabled people to overcome their impediment based on neural network. They used Feed-Forward Neural Network to design intelligent fire alarm system. They also used recurrent neural network to learn the user habits. However, building such system needs a lot of configurations as it needs a server at edge to process and save the data. Furthermore, the number of trained and tested samples was small. Park et al. [15] also came up with the Residual- Recurrent Neural Network architecture for smart home to predict human activities. They evaluated the proposed system using the Massachusetts Institute of Technology (MIT) dataset.

To counter the intelligence challenges in IoT gateway, Wang et al. [16] suggests a framework of smart gateway for smart homes that consists of smart home layer, smart gateway layer, and cloud layer. While the gateway performs data collection, awareness, and reporting, the cloud stores the reported data, and adjusts the data collection and awareness policy. Recent distributed intelligent approach was presented by Rahman et al. [5], they propose a Distributed Intelligence model for IoT gateway based on belief network and reinforcement learning to learn, predict, and make a decision. This system starts based on a small number of predefined rules, after that, the system can change the rules based on past experiences. Another recent distributed intelligent approach is proposed by Allahloh et al. [14], which is an intelligent oil and gas field management and control system based on IoT. It uses several currently available technologies such as SCADA and LabVIEW that is installed on the workstations and microcontrollers connected to wireless networks.

As mentioned above, large effort is being performed in the field of IoT gateway that is located between cloud and end devices. Although there is a large amount of work being done to enable the gateway at edge layer, there is only small improvement towards providing intelligence to the gateway by extracting knowledge from the raw data at IoT gateway to make decision locally. Moreover, small number of papers discussed automatic offloading from the gateway to the cloud in overload situations. On other words, few researches performed in collaborating between the IoT gateway and the cloud when the gateway is unable to analyze the data, make the decision and take the action.

3. Distributed Intelligence

DI in IoT is a paradigm that uses various models, techniques, and algorithms to make decisions to process IoT data either in the edge layer, cloud layer, or both. The development of IoT applications has recently become easier with the availability of development kits, open hardware, and software. IoT applications have also become easy to deploy on the cloud with the availability of PaaS resources and IoT cloud services [17].

Computer boards such as Arduino and Raspberry pi have shortened the development of IoT applications because these boards can be an IoT gateway between the cloud and the end layer, at the edge layer. However, complications remain at the software level. These boards are not intelligent enough to reduce the IoT-based knowledge from data that it receives from sensors and to ultimately make a decision [18].

For current IoT approach, sensors gather raw data from the environment and forward it to the gateway at the edge layer. Thereafter, the gateway converts between the protocols and forwards the data to the cloud. Then, the cloud processes the data, extracts knowledge, and makes a decision of whether to actuate a device. Finally, the cloud sends the decision to the actuator at the end devices layer.

This existing approach of sending all raw data created by end devices to the cloud using an IoT gateway is not efficient. It negatively impacts the network's reliability, availability, robustness, and security and privacy. Regarding availability, suppose we have a smart home system that is totally cloud-based. Assume there is a fire inside the home, and the Internet connection is lost. In this case, the system will fail to connect with the cloud to process the data and perform the appropriate action.

Some contributions have proposed a basic function for the IoT gateway, such as applying predefined rules that are not efficient, especially when complex IoT applications are involved, as they require a large number of rules. IoT applications connect a considerable number of factors. As a result, the intelligence developed by predefined rules fails to scale well. Furthermore, predefined rules cannot offer intelligence in undefined conditions since they can only offer presumed intelligence.

On the other hand, intelligence should not reside only on the cloud. Nowadays, there is a need for providing intelligence both in the cloud and edge layers. Decisions should be made in either of these layers, depending on which layer will provide the desired functionality and performance. This can be achieved by distributing the intelligence between the edge and cloud layers. DI in IoT means distributing the intelligence over both the edge and the cloud layers in order to provide the required functionality and performance for IoT applications. The IoT gateway at the edge layer should be able to process data, make a decision in most cases, sending data to the cloud only for overloaded tasks.

The question, then, is, 'How can a gateway be provided with the intelligence needed to obtain knowledge from the data?' Algorithms and techniques should be running on the IoT gateway. They should also be able to process real-time IoT data and extract knowledge in order to make a decision. Finally, algorithms should be able to perform their functions even when cloud connectivity is lost.

Machine learning can be a powerful solution for processing data produced by IoT devices. Combining machine learning and IoT gateways can help when analyzing data and making decisions locally. This combination can also provide real-time prediction, security, reliability, and availability. However, many IoT gateway controllers, such as

Raspberry Pi and Arduino, have limited processing capabilities. Therefore, the gateway should use several techniques based on CPU utilization, CPU temperature, the number of tasks, and the type of application to determine where to process each incoming task. For example, the gateway might process only a certain number of incoming tasks simultaneously and offload the rest to the cloud.

4. The Proposed Model and Methodology

This section presents a DIM by offering intelligence at both the edge and the cloud layers. The model was designed in two main steps: designing edge intelligence and designing cloud intelligence.

4.1. Designing Edge Intelligence

Edge intelligence represents the intelligence offered by an IoT gateway based on the data collected from end devices. After receiving an IoT task, the first function of the gateway is to decide where the task should be executed. This is done by testing the gateway's CPU utilization. The offloading decision is performed using equation (1), where x is CPU utilization and t is the threshold value of CPU utilization. If the value of f(x) is less than or equal to the threshold value, the task will be processed locally. If the value of f(x) is greater than the threshold value, the task will be offloaded to the cloud.

$$\boldsymbol{f}(\boldsymbol{x}) = \begin{cases} 0, & \boldsymbol{if}\ 0 \le \boldsymbol{x} \le \boldsymbol{t} \\ 1, & \boldsymbol{if}\ \boldsymbol{x} > \boldsymbol{t} \end{cases} \tag{1}$$

The second function of the gateway is to process tasks and take actions locally. This is performed by implementing a neural networks algorithm. For this model, three neural networks algorithms are tuned, implemented, and evaluated to build an algorithm that helps the gateway extract the knowledge from raw data, make decisions, and take action.

The following artificial neural networks algorithms are tested: multi-layer perceptron neural networks, long-short-term memory, and gated recurrent units. One of these algorithms will be chosen to be implemented in the gateway based on the analysis results in the evaluation chapter.

Algorithm 1 is employed to provide the necessary level of intelligence to the gateway while the gateway keeps listening to a coming task. Once a task arrives, the gateway checks its CPU utilization. If it is less than or equal to the threshold, the gateway processes the task locally based on the ANN algorithm. If the task exceeds the threshold, the gateway offloads it to the cloud.

To explain further, suppose the motion sensor starts to detect motion in the living room and the CPU utilization threshold is 75%. A task comes to the gateway to control the light. The gateway checks its CPU utilization, which is 70%. Thus, the gateway decides to process the task locally. The gateway obtains the values of motion, the husband's location, the wife's location, the time (morning, afternoon, or evening), and the day (weekday or weekend). Then, the gateway inputs these values into the ANN algorithm to decide whether to turn the light on or off. The ANN algorithm should be trained using a dataset before being implemented on the gateway.

Algorithm 1: The Role of IoT Gateway

```
while new task do
    Data: get the values of parameters related to the
          task
    if CPU utilization less than or equal the threshold
     then
        use the trained neural network algorithm to
         process the task;
    else
        offload the task to the cloud;
    end
end
```

4.2. Designing Cloud Intelligence

Cloud intelligence is the intelligence offered by a cloud-based service based on the data collected from end devices. After the offloading decision is performed by the gateway, the cloud receives the task. The cloud processes the task based on the prior-belief probability distribution of the attributes controlling home appliances (e.g., light and air-conditioning).

The smart home cloud application is configured to have the prior-belief probability [2] needed for the attributes that control the light and air conditioning (Fig.1. and Fig.2.). The beliefs are calculated according to equation 2, where

P(A) is the probability that an action will be taken, and X_i and Y_i are the belief and its value, respectively. The predefined threshold for taking an action is 0.5.

The equation shows how to calculate the probability that the cloud will make one decision over another. For example, for controlling the light, if the motion sensor does not detect motion, if the husband and wife are inside, if it is the evening during the weekend, The probability of turning the light on/off is as follows: P(l) = (0.35 * 0.3) + 2 (0.2 * 0.5 * 0.6) + (0.3 * 0.6) + (0.15 * 0.6) = 0.495. as P(l) less than the threshold (0.5), the decision will be to turn the light off.

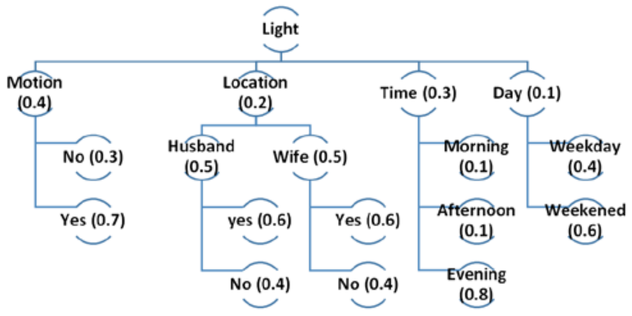

Fig.1. Probability Distribution of Light Attributes.

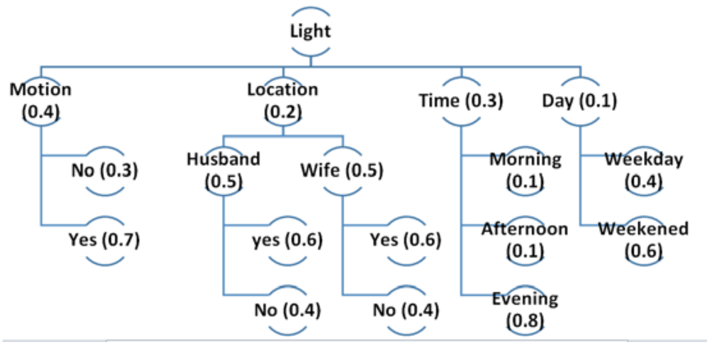

Fig.2. Probability Distribution of Light Attributes.

$$P(A) = \sum_{i=1}^{n} X_i Y_i \tag{2}$$

4.3. *Workflow*

The present work proposes a distributed intelligence model that integrates edge and cloud computing. The model enables fast local data processing of most of the real-time data to support fast decision-making and actions at the gateway. When the gateway is unable to process a task, it is offloaded to the cloud. Fig.3. illustrates the distributed intelligence model, and the workflow is explained below with controlling the light. The same steps are applied to control the temperature.

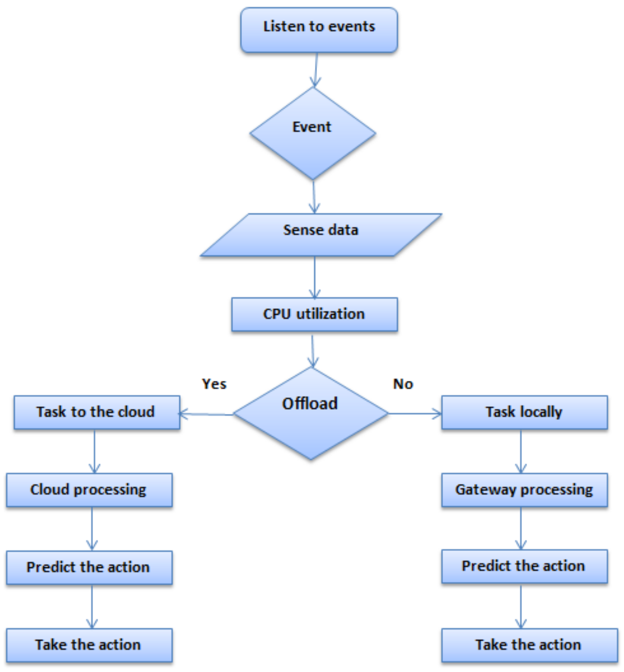

Fig.3. Workflow of the IOT Gateway.

- Sensors send respective data to the gateway. For example, to control the light in the living room, the gateway needs to collect data from the motion sensor, the husband's location, the wife's location, and the light status at each time period to train the ANN model.
- The gateway saves data and keeps track of the date and time on a CSV file to train the ANN algorithm.
- The gateway keeps listening to an event. When an event happens, it collects the current status of the motion sensor, the husband's location, and the wife's location, along with the date and time.
- After that, the gateway checks its CPU utilization. For the purpose of dynamic offloading, the CPU utilization threshold of 75% recommended in [18] is used.
- If the CPU utilization is below the threshold, the gateway processes the task locally. The gateway uses the ANN trained algorithm to control the light, and then the gateway sends the decision to the smart light to actuate.
- If the CPU utilization is equal to or greater than the threshold, the gateway offloads the task to the cloud.
- The cloud receives the task with the current values of the motion sensor, the husband's location, and the wife's location, as well as the date and time.
- The cloud uses the probability distribution values of motion sensors and the husband's location, as well as the date and time (according to Fig.1.) to decide whether to turn the light on or off. If the value is equal to or greater than the predefined threshold (0.5), the decision is to turn the light on; otherwise, the decision is to turn it off.
- The cloud sends the decision to the gateway, which passes it to the smart light to actuate.

5. The Operational Model

To demonstrate our consumption, a real experiment of DIM in a smart home IoT system is conducted. The experiment combines cloud and edge computing to control the light and temperature in the living room. It is assumed that a husband and wife live in the house.

As shown in Fig.4., we have implemented a smart home system. The system includes the gateway, ZigBee USB, the husband's phone, the wife's phone, a motion and temperature sensor, a smart bulb, an air conditioning system, and a smart plug. The laptop is used to remotely access the gateway.

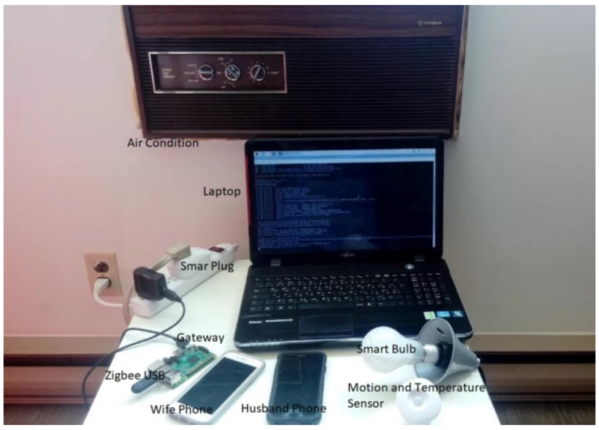

Fig.4. Gateway and End Devices in Smart Home System.

The implementation has three main phases: end devices layer implementation, edge layer implementation, and cloud layer implementation (Fig.5.). The implementation of the three phases is explained below. Table 1. also summarizes the role of each component of the operational model.

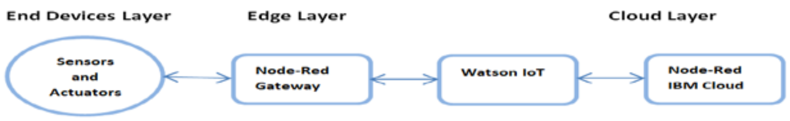

Fig.5. Operational model for Distributed Intelligence Model.

Table 1. Operational model components.

Component Name	**Component Type**	**Location**	**Description**
Sensors and actuators	Physical sensors and actuators	End devices	Uses to collect data and control appliances.
Node-RED gateway	Node-RED	Edge	Creates data flow application.
Watson IoT	IBM IoT platform	Between edge and cloud	MQTT message broker.
Node-RED IBM cloud	Node-RED	Cloud	Process IoT sensors' data.

5.1. End Devices Layer

The end devices layer mainly consists of sensors and actuators. Sensors basically detect the physical phenomena or properties that occur around them and sense several parameters according to the goals of usage, such as temperature and motion. Actuators are the component of the system that performs the actions in response to instructions given by the system, such as turning lights on or off.

In the experimental model, a motion sensor, temperature sensor, smart bulb, smart plug, and two mobile phones (one each for the husband and wife) are configured to control the light and temperature. We use one device to track both motion and temperature. The details of all devices implemented in the end device layer are shown in Table 2.

Table 2. Description of sensors and actuators.

Devices	Producer	Communication
Smart Plug	Sylvania	ZigBee
Motion sensor	Samsung	ZigBee
Temp sensor	Samsung	ZigBee
Bulb	Cree	ZigBee
iPhone 7	Apple	Wi-Fi
iPhone 8	Apple	Wi-Fi

The motion sensor is used to detect whether anyone is home. When the motion sensor detects motion, it means somebody is in the house. When the motion sensor is unable to detect motion, it means either nobody in the house or someone is in the house but is asleep. The two phones are used to detect whether the husband and wife are inside or outside based on whether their phones are connected to the home's Wi-Fi. For example, when the husband's phone is connected to the Wi-Fi, it means he is inside the house.

The temperature sensor tracks the temperature inside the house. The values are considered as low, medium, or high. The smart plug is used to connect the air conditioner to the home's electricity.

5.2. *Edge Layer*

The edge layer mainly consists of the IoT gateway. Raspberry Pi 3 Model B with a Raspbian GNU/Linux version 10 operating system is used as an IoT gateway controller in order to manage and control end layer devices. Raspberry Pi 3 model B has the following technical specifications [19]: Broadcom BCM2837 64-bit Quad Core Processor running at 1.2 GHz, 1 GB DDR2 internal RAM, Dual Core Video Core IV®Multimedia Co-Processor GPU, 16 GBytes SSD memory card for loading an operating system and storing data, and BCM43143 (802.11 b/g/n Wireless LAN and Bluetooth4.1) wireless connectivity.

The implementation starts by connecting devices in the end layer to the IoT gateway controller so that the model can control the light and temperature of the living room. The data is transmitted from the sensors and the actuators to the gateway either via Wi-Fi or ZigBee. Fortunately, the Raspberry Pi supports Wi-Fi wireless communication, and so the mobile phones are connected to the gateway through Wi-Fi. The Raspberry Pi, however, does not support ZigBee by default. A ZigBee USB (Conbee II) is attached to the gateway to connect the end devices that support ZigBee protocol. The hardware specifications of Conbee II are ATSAMR21B18 ARM®Cortex®-M0+ microcontroller, 10 mW transmission power, 200 m signal range in free line of sight, and 256 KByte flash memory.

Sensors and actuators are connected using a Node-Red platform (v1.0.3) [20] that is installed on the gateway. After connecting the sensors to the gateway, the gateway is configured to obtain the sensors' data every 10 minutes for two weeks and save it in a CSV file to create two datasets (one for light and one for temperature).

In order to build the light dataset, the gateway saves information about light status, motion, the husband's location, the wife's location, and the date and time. The day is recorded either as a weekday or weekend, and the time is recorded either as morning, afternoon, or evening. The date and time parameters are added because people's daily activities change depending on whether it a weekday or weekend. For example, people usually wake up earlier on weekdays than on weekends.

On the other hand, in order to build the temperature dataset, the gateway saves the parameters of the air conditioner's status, temperature, motion, the husband's location, the wife's location, and the time. The air condition's status is either on or off. The temperature is saved as either cold, medium, or high. The other parameters have the same settings as the light dataset.

5.3. *Cloud Layer*

After creating the end devices layer and the edge layer, a Node-Red Starter Kit application is created that runs on IBM Cloud [21]. IBM Cloud is a platform on which developers can build, run, and manage applications by integrating existing services from IBM or third-party providers. IBM Cloud is based on an open-source PaaS called cloud foundry, which offers middleware services such as data and workload management.

When the gateway is unable to process a task, it is offloaded to the cloud. The Node-Red cloud application processes the offloaded tasks based on prior beliefs for controlling the light and temperature (Fig.1. and Fig.2.). For example, when the gateway is unable to decide whether to turn the light on or off, it offloads the task to the Node-Red IBM Cloud, which will use prior beliefs to decide whether to turn the light on or off.

Next, an IoT platform service instance is created to act as asynchronous glue among all the components of the IoT DI operational model. As such, the IoT platform service (Watson IoT) [22] is connected to the Node-RED Starter application located on IBM Cloud. Thereafter, the gateway is connected to the IoT Platform.

5.4. *Intelligent IoT Gateway*

Providing intelligence to the gateway can be achieved by implementing an ANN algorithm [23]. ANNs is a technique of doing machine learning that try to realize implicit relationships in a series of data through a process that imitate the human brain. ANN is like others machine learning algorithms, used for tasks that are very sophisticated for human to program directly. Some tasks are very complicated that it is difficult for humans to formulate and code for them. So instead, we provide a dataset to a machine learning algorithm and the algorithm try to analyze the data and search fora model that can perform what the programmer have program it to perform.

In this work, ANN algorithms are tuned, implemented, and evaluated to build an algorithm that will help the gateway extract the knowledge from raw data, make decisions, and take appropriate actions. The following ANN algorithms are tested: multi-layer perceptron neural networks (MLPNNs) and two feedforward neural networks (FFNN1 and FFNN2) based on long-short-term memory (LSTMs) and gated recurrent units (GRUs) architectures, respectively. We built the two feedforward neural networks based on RNNs (LSTMs and GRUs) architectures because they performed well in some works of non-sequential data [24, 25], Although RNNs are designed for processing sequential data.

MLPNNs are perceptrons that collaborate with other perceptrons, stacked in different layers, to resolve sophisticated problems. MLPNNs have three layers: input layer, hidden layer, and output layer. Each perceptron in the input layer, sends outputs to all the perceptrons in the hidden layer, and all perceptrons in the hidden layer send outputs to the output layer.

LSTMs networks are an extension for RNNs, which extend the memory. Therefore, they are convenient to learn from the experiences that have extremely long-time gaps in between. LSTMs are able to store and recall information over a long period of time. The fundamental element to LSTMs is the cell state, the line crosses the top of the diagram. LSTMs have the ability to remove or add information to the cell state, carefully arranged by structures called gates. Gates are a way to pass information cross. LSTMs have three of these gates to protect and control the cell state that are forget, input and output gate.

Another extension for RNNs is Gated Recurrent Units (GRUs) which are also can solve vanishing gradient problem of a standard RNNs. The update gate and reset gate are used to solve the vanishing gradient problem. Both gates determine what information has to be passed to the output. They can be trained to hold information for long time, without taking it out through time.

The algorithms are implemented in Python. In general, building An ANN classifier includes two essential steps: (a) building the network by determining the appropriate numbers of layers and neurons and (b) training the network. Every implemented algorithm consists of three layers: an input layer, a hidden layer, and an output layer. For controlling the light, the input layer has five neurons, which is equal to the number of parameters used to control the light. The five input neurons correspond to motion, the husband's location, the wife's location, time, and weekday/weekend. For controlling the temperature, the input layer has five neurons, which is also equal to the number of parameters used to control the temperature. The five input neurons correspond to motion, temperature, the husband's location, the wife's location, and time.

For MLPNNs, the number of neurons in the hidden layer is chosen experimentally using cross-validation. After testing all the values, from the number of neurons in the input layer to less than twice the number of neurons in the input layer [26], the network with the highest accuracy is selected as the best one.

For FFNN1 and FFNN2, the number of units in the hidden layer is specified using equation 3 [27], where Ni is the number of input time steps (input), No is the number of output nodes (features), Ns is the number of rows (samples) in the training data, and α is a scaling factor between 2 and 10. After testing α values ranging from 2 to 10 (representing the number of units in the hidden layer), the network with the highest accuracy is selected as the best one.

$$N_h = \frac{N_S}{\alpha\left(N_i - N_O\right)} \tag{3}$$

In the output layer, for controlling the light, one neuron is used in each algorithm that represents the light status or air conditioning status. The next step is training, which involves using a learning algorithm to estimate network weight to reduce the overall error between the value estimated by the trained network and the target value. Light and air conditioning had the same training set up.

The loss function used for all the algorithms is categorical crossentropy, while adam is used optimizer. Next, we fit the training data into the network. Ten iterations are chosen before training is stopped. A batch size of 5, a verbosity mode of 1, and a validation split of 20% were also chosen. Table 3. shows the configurations of the algorithms' parameters.

After collecting sensors' data every 10 minutes for two weeks, the best ANN model is implemented and executed on the gateway. In general, training ANN models can take up to several weeks, so we saved the model's weights after training in order to make predictions again later.

Table 3. Algorithms parameters configuration.

Parameters	**MLPNN**	**FFNN1**	**FFNN2**
Hidden layers	1	1	1
Activation function	sigmoid	relu	relu
Nodes	9	50	50
Optimizer	adam	adam	adam
Loss	crossentropy	crossentropy	crossentropy
epochs	10	10	10
Batch size	5	5	5
Verbose	1	1	1

5.5. *Intelligent Cloud*

After creating end devices and edge layers, a Node-Red Starter Kit application is created and run in IBM Cloud. IBM Cloud is a platform on which developers can build, run, and manage applications by integrating existing services from IBM or third-party providers. IBM Cloud is based on an open-source PaaS called cloud foundry, which offers middleware services such as data and workload management.

First, the Node-Red cloud application will process the offloaded tasks based on the prior-belief probability distribution of the attributes controlling the light and air condition. The Node-Red cloud is configured to have prior-belief probability [2] for the attributes that control the light and air conditioning (Fig.1. and Fig.2.). The beliefs are calculated according to equation 2, where P(A) is the probability of taking an action, X_i and Y_i are the belief and its value, respectively, and the threshold for deciding to perform an action is 0.5. The equation shows how to calculate the probability that the cloud will decide to perform an action.

6. Evaluation

This section discusses the evaluation method of the proposed model. As mentioned above, this model is tested using a smart home application. It is assumed that two people (a husband and wife) live in the house. Data regarding the motion sensor, the husband's location, the wife's location, and the date and time are used to control the light using an ANN algorithm. A temperature sensor, the husband's location, the wife's location, and the date and time are used to control the air conditioning system.

6.1. Intelligence Algorithms

In this work, 40% of the data is tested through the algorithms. Comparisons are made regarding true positive (TP), false positive (FP), accuracy, precision, recall, and F1-score. The classifier that performs the best is implemented in the gateway to control home appliances.

For evaluating the algorithms mentioned above, this work considers some preliminary results. For deciding whether to turn the light on or off, for example, a confusion matrix is used to describe the performance of the classification models on a set of test data for which the true values are known.

In order to understand the confusion matrix, we define the most basic terms related to it. The first is TP (i.e., we predicted yes, and the decision actually was yes). The second is TN (i.e., we predicted no, and the decision actually was no). The third is FP (i.e., we predicted yes, but the decision was actually no). Finally, the fourth is FN (i.e., we predicted no, but the decision was actually yes).

Table 4. Confusion Matrix of the Neural Networks Algorithms.

	Predicted: No			Predicted: Yes		
	MLPNN	FFNN1	FFNN2	MLPNN	FFNN1	FFNN2
Actual: No	626	612	603	1	15	24
Actual: Yes	98	20	17	39	117	120

Table 4. represents the confusion matrix, which summarizes the prediction results of the algorithms, which are used to evaluate the algorithms. Although MLPNNs exhibit the highest TP (626), the TP of FFNN1 and FFNN2 are close to it (612 and 603, respectively). Both FFNN2 and FFNN1 are almost three times higher in terms of TN (120 and 117) than MLPNNs (39). Furthermore, MLPNNs exhibit almost five times higher TP (98) than FFNN2 (17) and FFNN1 (20). According to the confusion matrix, FFNN2 and FFNN1 are better for deciding whether to turn the light on or off.

Table 5. Comparison between the three algorithms.

Parameters	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
MLPNN	87	28	80	44
FFNN1	95	85	89	87
FFNN2	95	88	83	85

Table 5. also shows a comparison between the algorithms with regard to accuracy, precision, recall and F1-score. Accuracy is simply a ratio of correct predictions to the overall observations. FFNN2 and FFNN1 were the most accurate, with a value of 0.95 recorded for each. Precision is the ratio of correctly predicted positive observations to all predicted positive observations. FFNN2 exhibit the highest precision, with 0.88. Recall is a ratio of correctly predicted positive observations to all observations in the actual 'yes' class. FFNN1 exhibits the highest recall (0.88). F1-score is a weighted average of precision and recall. FFNN1 exhibits the highest F1-score (0.87). Based on these findings, FFNN1 is deemed the best algorithm, although the parameters of FFNN2 are very close.

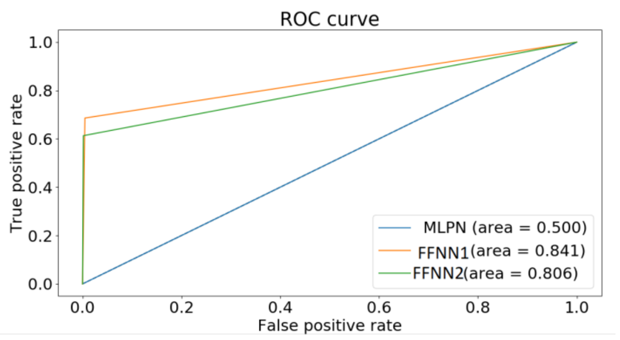

Fig.6. Receiver Operating Characteristic (ROC) curve.

In order to summarize the performances of the three classifiers, a receiver operating characteristic (ROC) curve (a type of graph plot) is used to identify the extent to which a binary classifier can distinguish between classes. The ROC curve indicates the TP rate against the FP rate for various threshold settings.

As shown in Fig.6., the ROC curve's x-axis acts as the FP rate while the y-axis represents the FN rate. The areas that appear under the curves of MLPNNs, FFNN1, and FFNN2 are 50%, 84%, and 80%, respectively. This results that FFNN1 is the most appropriate classifier.

The results in this section show that FFNN1 exhibits the highest percentage for most of the metrics discussed here. Thus, FFNN1 is implemented at the gateway of the proposed model for task processing.

6.2. *IoT Gateway Performance*

In order to demonstrate the effectiveness of offloading strategy and the benefits it provides; we evaluate the performance of the IoT gateway in the model by running the smart home application. This process starts with four parallel tasks (two light-control tasks and two temperature-control tasks). When one of the tasks ends, the gateway chooses a random delay (1-20 seconds) to start another task.

The evaluation starts by measuring the processing time and CPU utilization of the gateway when offloading is not enabled. Then, offloading is deployed, and the operational characteristics are measured. Later, we will compare offloading (DIM) and no-offloading (totally local processing) scenarios with regard to the operational characteristics.

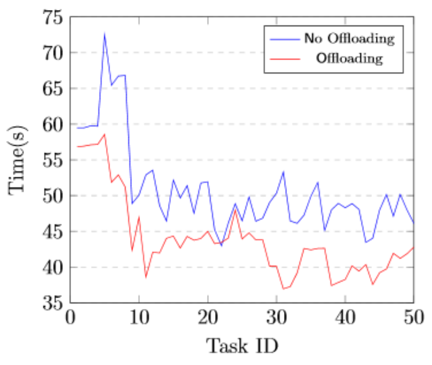

Fig.7. Tasks Processing Time

Fig.7. presents the processing times of the first 50 tasks processed locally for both no-offloading and offloading. For no-offloading, the task processing time is around 60 s at the start of the execution before increasing to 72 s for task 5 and then decreasing sharply to about 48 s for task 9. For all remaining tasks, the duration fluctuates between 42 s and 53 s.

On the other hand, for offloading, task duration starts at around 56 s. It peaks at almost 58 s for task 5 before falling to about 42 s for task 9. For all remaining tasks, the duration fluctuates between 36 s and 46 s.

In general, it is obvious that the task duration of the offloading scenario is than that of the no-offloading scenario. This is because enforcing the gateway to process all coming tasks locally causes either a delay or a connection loss when it is overloaded. Offloading some tasks to the cloud (when the gateway is busy) prevents the gateway from being overloaded and keeps it under control.

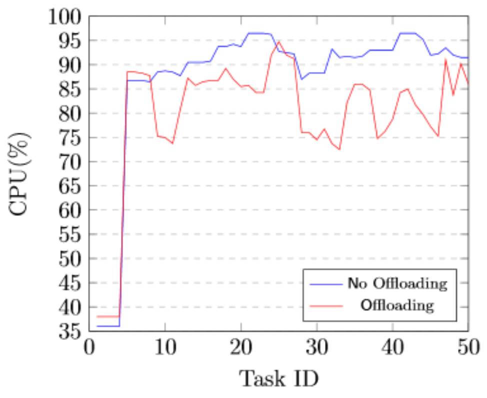

Fig.8. Gateway CPU Utilization.

As mentioned at the beginning of the section, the system starts with four parallel tasks (two light-control tasks and two temperature-control tasks). When one of the tasks is finished, the gateway chooses a random delay (between 1 and 20 seconds) to start another task (when a shorter delay period was chosen, the system went down).

Fig.8. presents the CPU utilization while the first 50 tasks were being processed locally, both for no-offloading and offloading. For no-offloading, CPU utilization is around 36% at the start of the execution. It then goes up sharply to 86% for tasks 5-8. After that, it goes up again, fluctuating between 87% and 96% for all remaining tasks.

On the other hand, for offloading, CPU utilization starts at around 38%. It then goes up sharply to 88% for tasks 5-7 before it falls to about 73% for task 11. After that, it goes up again to 89% at task 18. From this point on, it fluctuates between 72% and 94%.

It is obvious that the CPU utilization with offloading is less than 90% most of the time, while the CPU utilization without offloading is usually greater than 90%. The no-offloading scenario enforces all new tasks to be processed locally without testing the operation of the gateway. The gateway might be unable to process more tasks because doing so would cause service interruptions. The offloading threshold guarantees that the CPU utilization never reaches 100%, which ensures that the gateway runs smoothly and continuously without any service interruptions.

The observations made from the experiment carried out have showed the effectiveness of the DIM in IoT that are summarized below:

- Most real-time data is processed locally to speed up decisions and actions.
- Tasks are offloaded to the cloud when the gateway is unable to process them locally.
- The amount of data transmitted to the cloud is reduced, which improves security and privacy.
- IoT networks are more resourceful in terms of energy and communication band-width.
- Devices can interact with each other even when the system is disconnected from the cloud.

7. Conclusion and Future Work

The primary goal of this work was to investigate a proposed model that integrates edge and cloud computing to support Internet of Things (IoT) applications. This is important because cloud-based IoT applications suffer from several problems. Moreover, current network architectures and technologies are not sufficient for sending large amounts of IoT data to a cloud service and returning with a decision for controlling devices. At the same time, current edge technologies are unable to process the increasing amounts of IoT-generated data. Therefore, the IoT gateway at the edge layer should be able to process IoT data so that it can control IoT applications.

In this work, the concepts of edge computing, IoT gateways, cloud computing, neural networks, and DI in the context of the IoT were presented. We investigated the ability of an IoT gateway and cloud to process IoT data. Enabling an IoT gateway at the edge layer could overcome many challenges currently experienced by IoT systems, such as issues related to latency, robustness, availability, energy-efficiency, and security and privacy. We also discussed the role that DI plays in supporting IoT systems.

We conducted a proof of concept implementation of an IoT system that includes our demonstration of the DIM. We built an IoT gateway in the edge layer to demonstrate the ability of this layer to serve as an intermediate processing layer that can decide whether to perform a task locally or to offload it to the cloud. Our model was applied to a smart home system designed to control home appliances. The model demonstration involves all the data flow processes from data collected at the sensor nodes to the cloud.

We also configured and evaluated three types of ANNs for processing local tasks based on past experiences. The results showed that the FFNN1 algorithm performs better than MLPNNs and FFNN2. We later implemented FFNN1 on the gateway to process local tasks for controlling home appliances based on past experiences. For processing offloaded tasks on the cloud, we implemented a prior-belief probability distribution.

We constructed two scenarios for evaluating our model: an edge-based smart home system and a DIM-based smart home system. We compared the scenarios with regard to task processing time and gateway CPU utilization. The results showed that the DIM outperformed the edge-based model, as the collaboration between the gateway and the cloud helps the gateway run smoothly and continuously without service interruptions. Moreover, providing the gateway with the intelligence required to make decisions for controlling end devices was one of the significant capabilities of the DIM. The DIM combines the advantages of edge and cloud-based computing in one model to support IoT systems. It also tackles many challenges associated with edge and cloud-based computing. The DIM can be applied in a wide range of IoT systems, such as smart home, smart health, smart farming, and waste management systems.

More works can be performed in the future to enhance the presented model. For example, researchers might try to implement an ANN algorithm on the cloud that would enable the cloud to process offloaded tasks based on machine learning instead of prior beliefs. Researchers could also try to investigate other offloading criteria and compare them for supporting service level. Finally, the model's application in large-scale IoT systems, such as traffic management systems, could be tested.

References

[1] R. Kandaswamy and D. Furlonger, Blockchain-based transformation: A gartner trend insight report, https://www.gartner.com/en/doc/3869696-blockchain-based-transformation-a-gartner-trend-insight-report#a-1126710717, Last accessed on 2020-04-04, May 2013.

[2] Zainab Javed, Waqas Mahmood, "A Survey Based Study on Fog Computing Awareness", International Journal of Information Technology and Computer Science, Vol.13, No.2, pp.49-62, 2021.

[3] Mohamed M. Samy, Wagdy R. Anis., Ahmed A. Abdel-Hafez, Haitham D. Eldemerdash, "An Optimized Protocol of M2M Authentication for Internet of Things (IoT)", International Journal of Computer Network and Information Security, Vol.13, No.2, pp.29-38, 2021.

[4] Asifa Nazir, Sahil Sholla, Adil Bashir, "An Ontology based Approach for Context-Aware Security in the Internet of Things (IoT)", International Journal of Wireless and Microwave Technologies, Vol.11, No.1, pp.28-46, 2021.

[5] H. Rahman, R. Rahmani, "Enabling distributed intelligence assisted future internet of things controller (fitc)", Applied computing and informatics, Vol. 14, No. 1, pp. 73--87, 2018.

[6] R. Mueller, J. S. Rellermeyer, M. Duller, and G. Alonso, "A generic platform for sensor network applications," in2007 IEEE International Conference on Mobile Adhoc and Sensor Systems, IEEE, 2007, pp. 1–3.

[7] J.-W. Yoon, Y.-k. Ku, C.-S. Nam, and D.-R. Shin, "Sensor network middle-ware for distributed and heterogeneous environments," in2009 International Conference on New Trends in Information and Service Science, IEEE, 2009, pp. 979–982.

[8] Mozilla, Web of things, https://iot.mozilla.org/about/, [Accessed: 2020-04-04].

[9] S. Ismail, R. Ismail, and T. E. N. T. Sifizul, "A breast disease pre-diagnosis using rule-based classification," in International Conference on Ubiquitous In-formation Management and Communication, Springer, Phuket, Thailand, 2019, pp. 1037–1044.

[10] P. Mukalov, O. Zelinskyi, R. Levkovych, P. Tarnavskyi, A. Pylyp, and N.Shakhovska, "Development of system for auto-tagging articles, based on neural network", in3rd International Conference on Computational Linguistics and Intelligent Systems, CEUR-WS, Kharkiv, Ukraine, 2019, pp. 106–115.

[11] J. Miranda, N. Mäkitalo, J. Garcia-Alonso, J. Berrocal, T. Mikkonen, C. Canal, and J. M. Murillo, ``From the Internet of Things to the Internet of People," IEEE Internet Computing, vol. 19, no. 2, pp. 40-47, March 2015.

[12] A. Botta, W. De Donato, V. Persico, and A. Pescap ́e, "Integration of cloud computing and internet of things: A survey," Future generation computer systems, vol. 56, pp. 684–700, 2016.

[13] A. Badlani, S. Bhanot, Surekha, "Smart home system design based on artificial neural networks", In Proceedings of the World Congress on Engineering and Computer Science, San Francisco, USA, Vol.1, IAENG, pp. 106-111, Oct 2011.

[14] A. Hussein, M. Adda, M. Atieh, W. Fahs, "Smart home design for disabled people based on neural networks", In Procedia Computer Science, Vol.37, pp. 117--126, 2014.

[15] J. Park, K. Jang, S. Yang, "Deep neural networks for activity recognition with multi-sensor data in a smart home," In IEEE 4th World Forum on Internet of Things (WF-IoT), IEEE, pp. 155--160, Feb 2018.

[16] P. Wang, Pan, F. Ye, Feng, X. Chen, "A smart home gateway platform for data collection and awareness", IEEE Communications magazine, Vol. 56, No. 9, IEEE, pp. 87--93, 2018.

[17] A. Allahloh, S. Mohammad, "Development of The Intelligent Oil Field With Management and Control using IIoT (Industrial Internet of Things)", In the 2nd IEEE International Conference on Power Electronics, Intelligent Control and Energy Systems (ICPEICES), pp.815--820, 2018.

[18] R. Sosa, C. Kiraly, J. Rodriguez, "Offloading execution from edge to cloud: a dynamic node-red based approach", In the 10th IEEE International Conference on Cloud Computing Technology and Science (CloudCom), pp. 149--152, Dec 2018.

[19] Raspberry Pi.``Raspberry Pi 3 Model B." [online]. Available:\url{https://www.raspberrypi.org/products/raspberry-pi-3-model-b/}, [Accessed: 04- Aug- 2020].

[20] Node-RED. "About Node-RED". [online]. Available: \url{https://nodered.org/about/}, [Accessed: 04- Aug- 2020].

[21] IBM. "IBM Cloud". [online]. Available: {\url{https://www.ibm.com/cloud}}, [Accessed: 13- Aug- 2020].

[22] IBM,``Watson IoT,"[online]. Available: {\url{https://www.ibm.com/cloud/watson-iot-platform}}, [Accessed: 20- Aug- 2020].

[23] M. Berry, G. Linoff, "Data mining techniques: for marketing, sales, and customer support", John Willey and Sons, P.464, 1997.

[24] J. Mao, W. Xu, Y. Yang, J. Wang, Z. Huang, and A. Yuille, "Deep captioning with multimodal recurrent neural networks (m-rnn)", arXiv preprintarXiv:1412.6632, 2014.

[25] C. Chopra, S. Sinha, S. Jaroli, A. Shukla, and S. Maheshwari, “Recurrent neural networks with non-sequential data to predict hospital readmission of diabetic patients,” in Proceedings of the 2017 International Conference on Computational Biology and Bioinformatics, Association for Computing Machinery, 2017, pp. 18–23.
[26] K. Eckhardt, “Choosing the right Hyper parameters for a simple LSTM using Keras”, [online]. Available: {\url{https://towardsdatascience.com/choosing-the-right-hyperparameters-for-a-simple-lstm-using-keras-f8e9ed76f046}}, [Accessed: 20- Apr- 2020].
[27] J. Le,The 10 neural network architectures machine learning researchers need to learn, https://data-notes.co/a-gentle-introduction-to-neural-networks-for-machine-learning-d5f3f8987786, year=2018,[Accessed: 25-Apr-2020].

Authors’ Profiles

Baha Rababah is a graduate student at computer science department, University of Manitoba, Canada. His research interests are in Internet of Things, cloud computing, edge computing, and machine learning.

Dr. Rasit Eskicioglu is an Associate Professor at computer science department, University of Manitoba, Canada. His research interests are primarily in Experimental Systems. He is interested in Computer systems; Systems software; Operating systems; Distributed, cluster and grid computing; High speed network interconnects, Mobile networks, and Pervasive computing. Most recently, I am looking at wireless sensor networks (WSNs) and their applications to real world problems; Indoor localization, monitoring, and tracking; and Internet of Things (IoT).

How to cite this paper: Baha Rababah, Rasit Eskicioglu, "Distributed Intelligence Model for IoT Applications Based on Neural Networks", International Journal of Computer Network and Information Security(IJCNIS), Vol.13, No.3, pp.1-14, 2021. DOI: 10.5815/ijcnis.2021.03.01