Distributed Intelligence Model for Internet of Things Applications

by

Baha Rababah

A thesis submitted to
The Faculty of Graduate Studies of
The University of Manitoba
in partial fulfillment of the requirements
of the degree of

Master of Science

Department of Computer Science
The University of Manitoba
Winnipeg, Manitoba, Canada
April 2021

Thesis advisor

Rasit Eskicioglu

Author

Baha Rababah

Distributed Intelligence Model for Internet of Things Applications

Abstract

Increasing the implication of Internet of Things (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. 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. When the gateway is unable to process a task locally, the data and task are offloaded to the cloud.

Contents

List of Figures

List of Tables

Acronyms

ANNs Artificial Neural Networks. iii, iv, 5, 6, 11, 24, 52

BLE Bluetooth Low Energy. 1

DI Distributed Intelligence. 5, 12, 30

DIM Distributed Intelligence Model. 5, 6, 36

GRUs Gated Recurrent Units. 25, 27–29

IaaS Infrastructure as a Service. 22

IoT Internet of Things. ii, iii, 1–19, 21, 23, 24, 30–32, 35, 60, 66, 71, 72, 75

IP Internet Protocol. 17

LoWPAN Low power Wireless Personal Area Networks. 18

LSTMs Long-Short Term Memory. 25, 27, 29

MLPNNs Multi-layer Perceptron Neural Networks. iii, 25

NFC Near Field Communication. 15

PaaS Platform as a Service. 22, 30, 46

RFID Radio Frequency Identification. 1, 14–16, 71

RNNs Recurrent Neural Networks. iii, 26–28

SaaS Software as a Service. 22

UDP User Datagram Protocol. 18

WSN Wireless Sensor Networks. 15, 17

Acknowledgments

I would like to thank Dr Rasit Eskicioglu, my advisor, for his advice during this work. I would like to thank him for his helpful orientations, enthusiastic prompting and for nobly giving me his valuable time. I give a humble thanks to the committee members Dr. Ruppa Thulasiram and Dr. Ahmed Ashraf for the comments and suggestions they provided. I also give a thank to the staff at the Department of Computer Science, University of Manitoba for the great job they have done for supporting me to receive my Masters Degree in Computer Science.

This work is dedicated to my beloved daughter Leen.

Chapter 1

Introduction

1.1 Problem Description

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. 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 [2].

Cloud computing enables using distant servers to handle and process data instead of a local server. Scholars have started to look at edge computing [3, 4], as it is

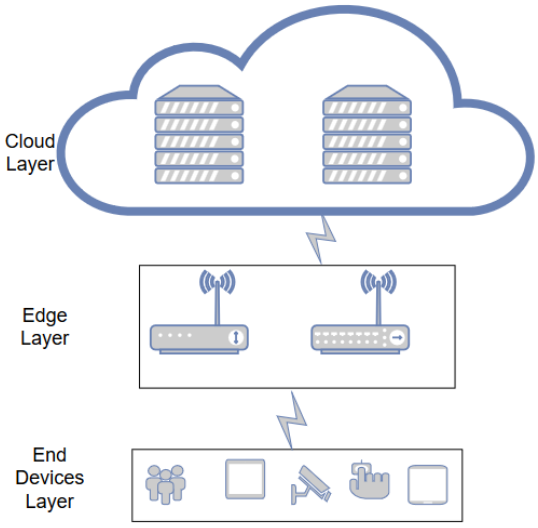

Figure 1.1: IoT and Cloud structure.

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.

Some of the solutions, such as Mozilla gateway [5], have employed rules-based intelligence in edge computing. However, rules-based intelligence does not scale well with the requirement of IoT applications [6], 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 [7].

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 (Figure 1.1). 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 translation to buffer an incoming message and forward in another format.

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, 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[2].

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 [2, 4]. 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.

This work attempts to solve the problems that are occurred from sending real time IoT data to the cloud. It also try to improve the edge layer capabilities of processing IoT tasks locally using machine learning.

1.2 Motivation

The current vision is to integrate people, things, services, and context information [8]. 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.

On the one hand, 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 [9]. Therefore, integrating edge and cloud into one model is required to include the beneficial features of both into one system to reinforce IoT applications.

1.3 Contribution and Outline

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 an ANN 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 this thesis is presented as follows. Chapter 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. Chapter 3 briefly introduces relevant background information about the IoT, edge computing, cloud computing, and ANNss. Chapter 4 describes the proposed DIM implemented in a smart home as a case study that integrates cloud and edge in order to support IoT-based smart home applications. Chapter 5 provides and discusses the experimental results. Finally, Chapter 6 concludes the work and presents possible directions for future work.

Chapter 2

Related Work

The concept " Internet of Things (IoT)" was firstly introduced by Kevin Ashton, the Executive Director of Auto-ID Labs at Massachusetts Institute of Technology, in 1990, while giving a talk to Procter and Gamble [10]. During his talk, he stated that all the data available on the web were first produced by humans who have limited attention, time and accuracy. He added that we need computers to know everything about things through analysing gathered data without any human help. Later on, the IoT has been evolved into systems using a variety of technologies such as Internet, wireless communication, microelectromechanical systems, and embedded systems. This chapter discusses some of the related works about IoT architecture and bringing the intelligence to edge layer.

2.1 Cloud Based IoT Systems

Nowadays, extracting knowledge from collected raw data is one of IoT challenges. The aim of IoT was to gather raw data from things, transfer them to the cloud for further processing. The cloud processes the data, makes the decision, and sends it to the actuators. Many proposals applied the scenario of sending collected data from the environment to the cloud to store, manage, and make decisions [11–15], as it provides large storage, complex processing, and access anywhere/anytime [16]. For example, Hassanalieragh et al. [14] presented a cloud health monitoring and management system that mainly has three components. Firstly, the data acquisition component has set of wearable sensors collect physiological biomarkers. The sensors transfer the data to the network through an intermediate data aggregator such as smart phone. Secondly, the data transmission components transfer patient's records to the health data centre in a real time. Finally, cloud processes component that store, analyse, and visualize the data.

In the last decade, many researches discussed and proposed models of IoT and cloud integration. Researches in [17] and [18] demonstrate the need to integrate IoT with cloud by presenting a deep understanding of the integration between cloud and IoT. They also provide an overview of the current research related to this topic. Babu et al. [19] present an architectural design for integrating cloud and IoT called Cloud-Assisted and Agent-Oriented for IoT. This design has three main components Smart Interface Agent (SIA), Smart User Agent (SUA) and Smart Object Agent (SOA). SIA interacts with external IT systems. SUA models users in the context of specific intelligent system, users can establish a certain request services using a GUI provided

by SUA. SOA models physical environment, it is supported by cloud computing platform. Li et al. [20] employs the Topology and Orchestration Specification for Cloud Applications (TOSCA) standard for cloud service management to systematically set down the elements and configurations of IoT applications. TOSCA describes in details the topology of application components and the implementation process of the applications. All the previously mentioned models are totally cloud-based which need to transfer all raw data to the cloud for processing and making decisions. However, cloud based systems cause high level of latency for IoT applications [21–23]. Moreover, none of the previously mentioned papers discusses processing raw data collected from end devices at the edge layer to learn, make the decision, and take actions, without sending the data to the cloud. Therefore, researchers have started to look at edge computing to process the data close to things as it provides fast processing, energy efficiency, reliability, and security and privacy [24].

2.2 Enabling Edge for IoT Systems

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 focuses for enabling IoT gateway at the edge of the network to handle and manage the IoT data. For example, Mueller et al. [3] introduced 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. [25] also came up with a sensor network system that mainly consists of one main server and a number of gateways to connect several sensor networks. Designing such system

requires a lot of configurations and high hardware cost. In another work, Bimschas et al. [26] presented a middleware for smart gateway to run different applications, such as protocol translation and request caching with sensor discovery. For general IoT applications, Guoqiang et al. [27] introduced general purpose smart IoT gateway that supports several communication protocols to translate different sensor data and external interfaces for flexible software development. In order to use smartphones as a gateway, Bian et al. [28] utilized android phone to be temporary smart home gateway that can predict user behaviour to shut down unused devices. This work aims at providing a dynamic home gateway that can reduce the wasted energy of a smart home.

In healthcare IoT application domain, Shen at al. [29] presented an intelligence 6LoWPAN border router that connects the health care sensors with IP network and uses a hidden Markov Model to make local decisions of health states. Stantchev et al. [30] introduced three-tier architecture (cloud, gateway, and smart items) to enable servitization for smart healthcare infrastructure. Servitization is the trend of convergence between manufacturing and the service sector. Rahmani et al. [31] also proposed an intelligence e-health IoT gateway for remote health monitors system at the network's edge in a three layer architecture. The gateway is able to provide several services, such as real time data processing, local storage, and data mining. In another work, Azimi et al. [4] proposed a hierarchical model for IoT monitoring health systems based on the MAPE-K [32] computing model introduced by IBM. The model uses fog and cloud computing to partition and execute of machine learning data analytic.

Many works have tried to counter the intelligence challenges at edge computing. For example, Badlani et al. [33] introduced smart home systems based on ANNs. It aims to reduce power consumption through analysing the human behaviour 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.[34] presented a self-adapting intelligence home system that help 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 configuration as it needs a server at edge to process and save the data. Furthermore, the number of trained and tested samples was small. In another work, Mehr at al. [35] studied the human activity detection performance using three ANNs algorithms: Batch Back Propagation, Quick Propagation, and Levenberg Marquardt. The results illustrated that Levenberg Marquardt is the best. Park et al. [36] also presented the Residual- Recurrent Neural Network architecture for smart home to predict human activities. They evaluated the proposed system using the Massachusetts Institute of Technology's dataset.

To counter the intelligence challenges in IoT gateway, Wang et al. [37] suggested a framework of smart gateway for smart homes that consists of home layer, 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. Another work introduced by Calegari et al. [38] suggested to use Logic Programming as a Service (LPaaS) to provide reasoning service for IoT

applications. They mentioned that Logic Programming as a Service (LPaaS) can enhance non-symbolic techniques in order to achieve distributed intelligence. Recent distributed intelligent approach was presented by Rahman et al. [2], they proposed 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 Intelligence (DI) approach is proposed by Allahloh et al. [39], which is an intelligent oil and gas field management and control system based on Internet of Things (IoT). It uses currently available technologies such as SCADA and LabVIEW that is installed on the workstations and microcontrollers connected to wireless networks. Alsboui et al. [40] proposed Mobile-Agent Distributed Intelligence Tangle-Based architecture that able to uphold multiple IoT applications. Tangle is a flow of interconnected and individual transactions. These transactions are stored across a decentralised network. The architecture consists of IoT devices, tangle to process transactions, Proof of Work server, and mobile agent. IoT devices are connected together through TCP/IP protocols for communication. They communicate with the Tangle to manage process and store data in an efficient way. Proof of Work (PoW) server is an IoT device that performs computations' cost on behalf of IoT devices. Mobile agent supports inter-node communications by transferring a set of transactions when passing through nodes on their path.

As mentioned above, large effort is being performed in the field of IoT gateway that is located between cloud and end devices. Although there are 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. In other words, few researches performed in collaborating between the IoT gateway and the cloud when the gateway is unable to analyse data, make decision and take action.

Chapter 3

Background

3.1 Internet of Things

Smart devices concept becomes popular in 1990s. IoT term was first introduced in 1999 by Kevin Ashton, the head of Auto-ID Centre at Massachusetts Institute of Technology [10]. He explained the possibility of creating IoT by tagging Radio Frequency Identification (RFID) and sensors to objects. After that IoT has become popular in research works and industrial fields. The recent innovative developments of information and communication technologies related to ubiquitous communication, pervasive computing, and ambient intelligence played an essential role in IoT development. Ubiquitous communication is a concept in communication and computer network where interconnection can be made anywhere and anytime. The pervasive computing is the improvement in smart devices including the computational feature process. Ambient Intelligence is the ability of objects to sense and response to physical environment.

There is no particular definition for IoT, many definitions introduced in the literature [31, 41–44]. However, the objectives of IoT are similar in most of those definitions. For this work, we will consider the definition that defines IoT as a paradigm to connect unlimited numbers of objects with their unique identifications anytime and anyplace using IoT communications protocols [31]. IoT objects might be mobile phones, clothing, machines, and doors. IoT has been involved in many applications, such as transportation [45], smart home [46], and health care [47].

3.1.1 IoT Architecture

Architecture for IoT does not have a particular standard because it is a broad concept. Several proposed architectures are presented such as International Telecommunications Union (ITU) architecture, European FP7 Research Project architecture, IoT Forum Architecture, and Qian Xiaocong, Zhang Jidong Architecture [48].

Although there is a small difference between those architectures, IoT architecture commonly has three main layers:

- The things layer with physical endpoints that sense and receive information. The physical endpoints can be smart devices (e.g smart car and smart phone), sensors, actuators and microcontrollers. These endpoints is the requisite part of IoT that uses Radio Frequency Identification (RFID), Bluetooth, Near Field Communication (NFC), ZigBee, Wireless Sensor Networks (WSN), and embedded intelligence to send the data to the next layer. The technologies that are used in this work will be discussed in the next section.

- The connectivity layer that has gateways and the core network that usually

performs protocols transactions.

- The application layer that collects sensors' data in order to store, process, measure, and take actions. In most modules, application layer resides on cloud that has the power to receive and process data coming from multiple resources.

3.1.2 Key Technologies of IoT

The IoT is a promising technology for a smarter future: smart kitchen shelves are able to search for several groceries, check and compare their prices before ordering them. Smart mattresses are able to tell you about your sleeping position, give you a relaxing massage, and more. While most of this might shortly be within our hands, we still have to care about the great technologies that makes IoT comes true. In this section, we will introduce some of those technologies that are used to build the model.

- Radio Frequency Identification (RFID): RFID is the technology that uses radio waves to detect, identify and track an object electronically through a tag that is attached to the object [49]. It can store and retrieve information using electromagnetic transmission and a radio frequency compatible integrated circuit. A simple RFID system mainly has RFID readers and RFID tags [50]. The RFID tag can be read by the RFID reader from up to several feet away and does not require being within a direct line with the reader to be tracked. RFID reader can scan several tags at the same time and transfer it to the server. RFID technology is widely used with different RFID applications, such as smart homes, public transportation, health services, and smart agriculture to identify things electronically.

- Wireless Sensor Networks (WSN): sensor is a device that is able to detect a certain type of input from the physical object or the environment such as heat, smoke, light, motion and pressure. A sensor consists of four units: power, sensing, computing and communication [51]. In general, sensor's output is an electrical signal that is transmitted to the data centre for processing and extracting the knowledge. WSN is a number of wireless sensors that communicate to collect information and detect a certain event in the physical environment [52]. Sensors in WSN are able to communicate in a centralized or decentralized approach based on several topologies such as mesh and star. Sensors can communicate together in the point to point or multi-hop models. WSN provide different beneficial data that might be used in certain applications and services such as military, healthcare, education, and agriculture. However, it has some limitations in communication and resources constraints, such as short communication range, mobility, reliability, and security. Nowadays, scholars are trying to solve these limitations by several modes such as different routing protocols and intelligent based approaches[53].

- Addressing: One of the main requirements of IoT is to identify and manage objects based on its identification, location or functions because it consists of large number of objects. In fact, addressing has achieved this requirement by adding more features to IoT such as uniqueness, reliability, persistence and scalability. Unfortunately, the current Internet Protocol (IP) version 4 is only able to identify a group of sensors based on their geographical location. The solution is to enable Internet Protocol (IP) version 6 to identify every sensor individually

based on its identification, location, or functions, as it has a larger address size in order to address and manage IoT objects efficiently [54]. Internet Engineering Task Force (IETF) standardized adaption protocol for IPv6 is called IPv6 over Low power Wireless Personal Area Networks (LoWPAN)[55]. 6LoWPAN is a short form that integrate Internet Protocol (IPv6) with Low power Wireless Personal Area Networks (LoWPAN). LoWPAN for End devices gives them the ability to transfer information wireless using Internet Protocol. It allows compression techniques to compress the IPv6 and User Datagram Protocol (UDP) packets after deleting the excessive data from the header part.

- ZigBee Protocol: Zigbee is a wireless communication protocol that is created by ZigBee Alliance to enhance the features of WSNs. It is a low power protocol, designed based on the IEEE 802.15.4 standard that has the attributes of low cost, low data rate, short transmission range, reliability, scalability, and flexible protocol design [48]. Although it has range approximately 100 meters, it can transmit data over a long distance through mesh or tree of intermediate devices. It is widely used for iot applications such as healthcare, smart agriculture, and smart home.

- Middleware: Due to objects heterogeneity, limited storage, and processing capabilities, middleware and application layer plays a key role with the main objectives of functional abstraction and device communication [56, 57]. Middleware is categorized in the layers such as object abstraction, service management, service composition and application. In generic IoT system, end devices usually collect specific information such as temperature, light and pressure. This

information passes through limited processing locally. Then, it is sent to the external gadget for computations. After that, it can be transmitted to the local area network until it reaches the Internet then the cloud-based service. The cloud-based service converts the received data into knowledge after some processing operations. This knowledge might need to be stored or action needs to be performed. In order to support IoT, middleware should handle the following challenges: interoperability between heterogeneous devices, context awareness, managing data, privacy and security, and device discovery [58].

3.2 Edge Computing

Few years ago, edge computing was defined as a set of computers, devices and network resources that produce, collect, process and send data to remote servers in the Cloud [59]. Nowadays, edge computing's role exceeds this definition with the basic role of collecting and sending data. It is defined as a paradigm of processing part of the data at the edge of the network close to the IoT devices, where the data is generated, rather than sending the IoT data to the cloud for processing [60]. The devices in the edge layer can perform both computational and communication tasks. For example, an embedded device such as Raspberry Pi could serve as an IoT gateway if it takes data from a smart home camera and performs data processing tasks that are used for fall detection.

Current cloud architecture fails to handle the huge amount of data produced by IoT devices, as it causes high latency and bandwidth utilization. Thus, edge computing is required in order to optimize computing in real time IoT applications.

The edge is a layer that is located between the cloud and the data sources. For example, a gateway in health care is the edge between health care things and cloud. Edge computing is able to perform several tasks such as data pre-processing, local data storage, data processing and caching, and request and deliver services from the cloud. Edge computing has been found to improve the reliability, availability, and security and privacy of IoT applications, as the data is processed locally [61, 62]. For example, Floyer investigated data management and processing costs of a cloud based wind-farm system versus a hybrid edge cloud based system [63]. The wind-farm consists of a number of data producing sensors and devices. The hybrid edge cloud system results 36% less expensive than the cloud based one. The quantity of required data to be transmitted in the hybrid system was also 96% less, compared to the cloud based system.

In IoT applications, intelligence requires to take place in edge computing layer. Computations take place on an edge gateway close to sensors and actors. The advantages of enabling edge computing can be summarized in the following points:

- Reduce the volume of data required to be transferred to a cloud system because part of it is processed by edge devices.
- Reduce latency and enhance real time data processing because most of the data is processed at edge layer close to end devices.
- Enhance user privacy and security because edge devices could use some security and privacy techniques prior of sending data outside the network.
- Enhance the robustness because decentralization system can provide transient

services during a network failure.

In the model, we use edge gateway for several reasons as not every decision can be computed in the cloud. First of all, latency impacts several vital decisions that make a cloud route trip inefficient. Think of a smart home system that is fully managed by cloud computing, it is nice that the system can control home appliances, such as doors and windows, However, in case the house is burning, we want the system to response immediately, connection with Internet is not guaranteed 24/7. Second, required bandwidth can be too high, if the produced data is too much to be transferred to the cloud. Recently, smart home applications connect everything at homes such as shelves, sofas, mattresses, tables, lights, and fridges. Either it is technically difficult to transfer all the IoT generated data to the cloud due to the current link speed, or it is only too costly, or both. Finally, security and privacy might be impacted, when the data is transferred outside the smart home network. Think what can happen, if somebody accesses your transferred smart home data before reaching the cloud. He/She can simply figure out when you are at home or away.

On the other hand, edge computing has disadvantages of limited processing and storage capability. In particular, a single board computer that can act as an IoT gateway still have a limited computations and storage capabilities that impact their role to process the big amount of IoT data. The gateway has to be robust and flexible to care of several issues such as data processing, data management, security, and quality of service (QoS). It is challenging to accomplish all the tasks instantaneously using current resource-constrained single board computers. Think what can happen, if the CPU utilization or the CPU temperature of the gateway is too high. Of course,

it will stop and will not be able to perform its job. Thus, the solution is to offload the overloaded tasks to the cloud for processing and making the decision of actuating.

3.3 Cloud Computing

Cloud computing is a model to enable ubiquitous, convenient, on demand network access to a shared set of configurable computing resources, such as networks, servers, storage, applications, and services that can be provisioned and managed easily [64]. Over the last ten years, cloud computing started to be popular in IT industry, as it has the advantages of unlimited storage, complex processing capabilities, and access anywhere/ anytime. It also provides virtual resources on demand. Leading IT companies such as IBM, Amazon, Microsoft, and Google adopted this model to provide services over the Internet.

Cloud services are categorized into three layers as shown in (Figure 3.1). Software as a Service (SaaS), Platform as a Service (PaaS), and IaaS [9]. SaaS is the applications operating on a cloud infrastructure provided by a cloud provider, such as force.com, Microsoft and IBM.SaaS delivers applications running on Cloud environments through the browser to clients. PaaS refers to provide the operating environment including the operating system and software development frameworks. It provides a development environment that can be accessed through web browser to develop applications without taking care of the processor power and memory size. IaaS refers to provide instant computing infrastructure, storage, and network resources in order to manage the operating system, cloud applications, and cloud storage. clients has the authority to do many things such as install operating system, install a virtual

Figure 3.1: Cloud Service Models.

disk, turn on/off the server, set access policy.

IoT devices are process and storage constraint that limits their capability,but Cloud has considerable process and storage resources. For this reason, IoT can use cloud's resources to make up its limited resources. IoT can be supported by Cloud through several ways such as, communication, computation and storage. For example, IoT data can be stored on cloud and processed in a low cost and an effective way. In particular, Cloud is useful for IoT applications that are computation intensive, such as using complex analytic algorithms.

On the other hand, connecting IoT application with cloud mainly depends on the internet connectivity. Due to interrupted network connectivity, network latency is high which is inefficient for real time IoT applications. As the amount of data being generated by IoT devices is increasing, it is very challenging to transfer all the IoT data to cloud due to restricted bandwidth. Furthermore, the process of transferring data to cloud could face major security and privacy issue. For example, data transfer through intermediate networks might be compromised.

In the model, we use cloud computing since not every decision can be computed in the edge. Edge computing has limited processing and storage capability that impacts

its role to process the big amount of IoT data. So, we use cloud computing to process the data and tasks that are unable to be processed by edge computing. Think of a smart home system that is fully managed by edge computing, it is nice that an edge device can control home appliances, such as doors and windows, however, in case the edge device receives large number of tasks that is unable to process them together, simultaneously, the edge device will try to process them together and it most probably will cause deadlock. We want the system to be available 24/7.

3.4 Artificial Neural Networks (ANNs)

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 them. So instead, we provide a dataset to a machine learning algorithm and the algorithm tries to analyze the data and search for a model that can perform what the programmer has programmed it to perform. For example, it is not easy to write a program that controls smart home appliances. Even if we have a good idea how to program it, it needs to implement a considerable number of complicated rules.

The problem of writing a program with large number of complicated rules is solved by using machine learning; we provide an algorithm with many examples that assign the correct output for a specific input. The algorithm then uses these examples to produce a model that perform the job. The program created by the algorithm

may look very different from a hand written program. If we implement it correctly, the program will work for new states as efficient as the examples we trained it on. For this work, the following ANN algorithms are used: Multi-layer Perceptron Neural Networks (MLPNNs) and two feedforward neural networks algorithms based on Long-Short Term Memory (LSTMs) and Gated Recurrent Units (GRUs) architectures respectively.

3.4.1 Multi-layer Perceptron Neural Networks (MLPNNs)

A perceptron is a binary classification algorithm that helps to classify a group of input into two parts yes or no [65]. Each input has a weight to indicate how important it is, and produce an output decision of 0 or 1. However, when it is joined with other perceptrons, it forms an artificial neural network. Multi-layer Perceptron Neural Networks (MLPNNs) is a perceptron that collaborate with other perceptrons, stacked in different layers, to resolve sophisticated problems. Figure 3.2 represents an Multi-layer Perceptron Neural Networks (MLPNNs) with three layers. 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. Each signal going to each perceptron in the next layer has a particular weight.

The Perceptron Learning Process can be summarized in the four steps [66]. Firstly, multiplies the inputs by their weights, and calculate their sum. Secondly, adds a bias factor in to be able to tune the numeric output of the perceptron. Third, feeds the sum through the activation function such as sigmoid, tanh and relu, it allocate the input values to the desired output values. For instance, input values could be

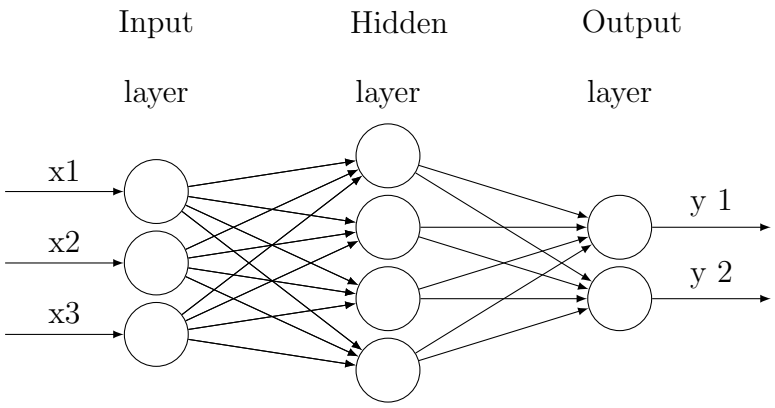

Figure 3.2: Multi-layer Perceptron Neural Network.

between 1 and 10, and outputs can be 0 or 1. Finally, the result is the perceptron output. Equation 3.1 shows how we can calculate the output for Single Perceptron Architecture, where σ represent the activation function, w is the weight of each link between input and output, x is the input, and b is the bias.

$$Y = \sigma(\sum_{n=1}^{n} w_i * x_i + b) \tag{3.1}$$

3.4.2 Recurrent Neural Networks (RNNs)

RNNs are artificial neural networks where connections between perceptrons compose a directed graph along a temporal sequence [65, 67]. This helps it to show temporal dynamic behavior. RNNs uses their internal state to handle variable length sequences of inputs. RNNs combine two properties: first, they are able to save a lot of information about the past, Second, they are able to update their hidden state in sophisticated ways. As shown if figure 3.3, RNNs can be describe as a chain of several

copies of the same network, each passing a message to a next.

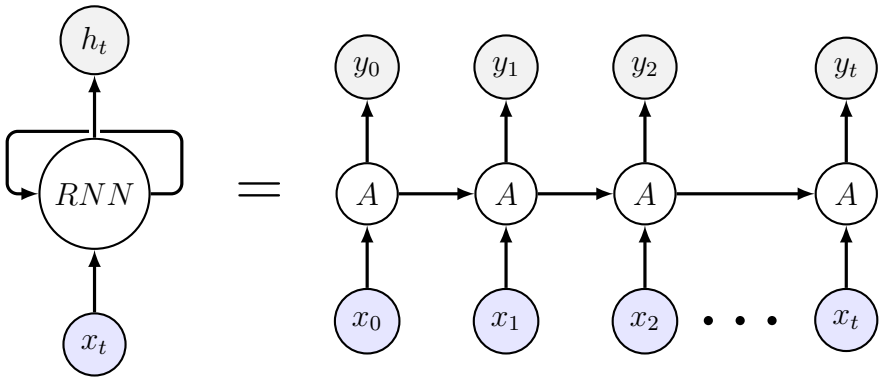

Figure 3.3: Recurrent Neural Networks.

On the other hand RNNs suffers from vanishing gradient problem where information quickly gets lost over time. The problem can be solved using Long-Short Term Memory (LSTMs) or Gated Recurrent Units (GRUs).

LSTMs networks are an extension for RNNs [65], which extends the memory. Therefore it is convenient to learn from the experiences that have extremely long time gaps in between. LSTMs is able to store and recall information over a long period of time. A LSTMs block has three gates: input, forget and output gate. Input gate allocate whether to allow new input in or not, forget gate deletes the unimportant information, output gate let it impact the output at the current time step. A LSTMs cell state passes the information among each LSTMs block. The cell state modifications are controlled with the previously mentioned three gates. Figure 3.4 presents a LSTMs cell; In particular, it illustrates the connections between the gates and the cell state itself.

Each gate and cell state are calculated according to equations 3.2, where w, x, σ, c, tanh are the weight matrix, input, sigmoid, and activation function respectively.

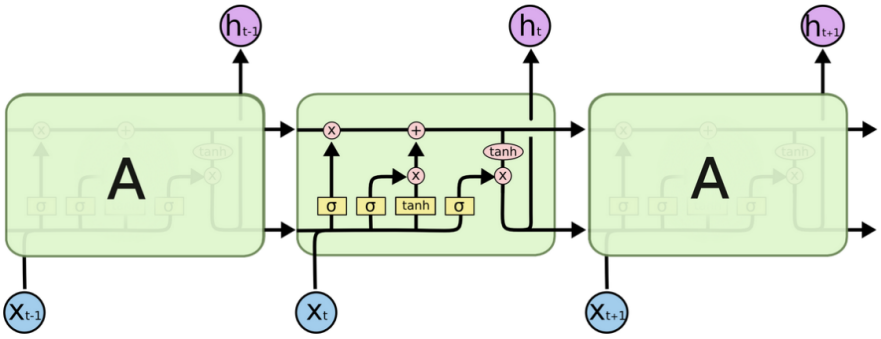

Figure 3.4: LSTM Cell [68]

The terms i, f, o, c represent the input gate, forget gate, and output gate. Finally, c represents the memory cell [68].

$$
\begin{aligned}
f_t &= \sigma(w_f.[h_{t-1}, x_t] + b_f), \\
i_t &= \sigma(w_i.[h_{t-1}, x_t] + b_i), \\
\hat{c}_t &= tanh(w_c.[h_{t-1}, x_t] + b_c), \\
c_t &= f_t * c_{t-1} + i_t * \hat{c}_t, \\
o_t &= \sigma(w_o.[h_{t-1}, x_t] + b_o), \\
h_t &= o_t * tanh(c_t)
\end{aligned}
\tag{3.2}
$$

Another extension for RNNs is Gated Recurrent Units (GRUs) which also can solve vanishing gradient problem of a standard RNNs [65, 69]. 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.

$$
\begin{aligned}
z_t &= \sigma(W_z x_t + U_z h_{t-1} + b_z), \\
r_t &= \sigma(W_z x_t + U_z h_{t-1} + b_r), \\
\hat{h_t} &= tanh(W_h x_t + r_t \odot U_h h_{t-1} + b_h), \\
h_t &= z_t \odot h_{t-1} + (1 - z_t) \odot \hat{h_t}
\end{aligned}
\tag{3.3}
$$

GRUs can be considered as a simplified version of LSTMs[70]. GRUs modifies the way the vector h_t is calculated. The GRUs components are calculated by the equations that are presented in 3.3, where z_t, r_t, σ,h_t are the update gate, the reset gate, sigmoid, and the modified hidden layer respectively.

3.5 Node-Red

Node-RED is a streaming flow based programming tool, developed by IBM and now a part of the JavaScript Foundation, for developers to wire hardware devices, online services, and APIs [71]. Node-red has a Node.js based runtime that provides developers with browser based flow editor that is used to drag nodes from a palette and connect them together to create IoT application. In the flow editor, you can create application by pulling nodes from a palette into a work space and start to wire them together. With a single click, the application is deployed back to the run time where it is executed. Several types of nodes are existed to receive, send, process, transform, and store data [72]. New nodes made by the community can be easily added and the flows can be saved and shared as a JSON file.

In our model, we use Node-red on both edge and cloud for several reasons. First of

all, it is an easy way to connect wide range of common devices and sensors. Second, it also can act as a software gateway on raspberry pi that brings additional functionality to the IoT application. Finally, it is available on several cloud platforms such as IBM and Groove, it is freely to connect end devices layer and edge layer with a cloud services.

3.6 Distributed Intelligence in IoT

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.

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 educe the IoT-based knowledge from data that it receives from sensors and to ultimately make a decision [73].

Figure 3.5 illustrates the current approach. First, 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

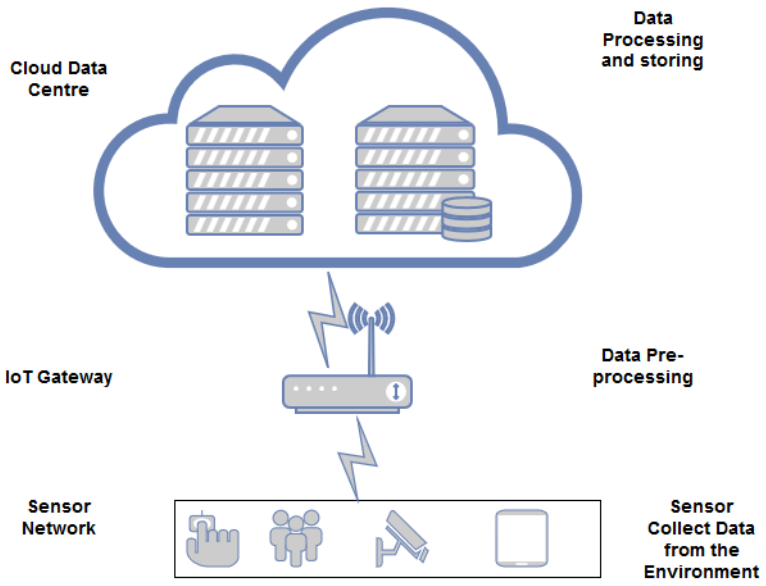

Figure 3.5: Existing Model of the integration between Cloud and IoT.

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 [74].

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 [75]. 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 processes only a certain number of incoming tasks simultaneously and offload the rest to the cloud.

The benefits of enabling DI in IoT 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 bandwidth.
- Devices can interact with each other even when the system is disconnected from the cloud.

3.7 Summary

This chapter discussed all concepts related to the DIM. Also, the IoT was described as a paradigm that connects unlimited numbers of objects with unique identifications anytime and anyplace using IoT communications protocols. IoT is the primary aspect related to our work that aims to connect and manage several smart home appliances. We also presented the edge computing concept as a paradigm for processing part of IoT data at the edge of the network (i.e., close to IoT devices), where the data is generated, rather than sending IoT data to a cloud service for processing.

The brain of our model is an IoT gateway that is located in the edge layer. This gateway receives data and tasks from end devices. It then decides whether to process each task locally or to offload it to the cloud. The gateway processes a task based on ANNs, which were also discussed in this chapter.

In addition, we presented a cloud computing architecture that enables ubiquitous, convenient, on-demand network access to a shared set of configurable computing resources, such as networks, servers, storage, applications, and services, that can be provisioned and managed easily. We highlighted its importance and power, as it processes any data and tasks that edge computing cannot handle.

We also presented Node-RED, which is a streaming-flow-based programming tool that developers use to wire hardware devices, online services, and APIs. Node-RED is used to connect all the model's components so that they can exchange data and process tasks.

Finally, we presented DI in IoT, which is the main feature of our model, as it combines edge computing and cloud computing to manage IoT devices. 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, the cloud layer, or both.

Chapter 4

Design and Implementation of the Distributed Intelligence Model

The goal of this work is to build a model that integrates cloud and edge computing in the field of IoT, thus avoiding the need to send all raw data produced by end devices to the cloud. To do this, we need a smart IoT gateway that is able to provide the necessary intelligence and make decisions closer to the end devices. The gateway should extract knowledge from the raw data that comes from sensors without sending them to the cloud.

This chapter presents a model that integrates edge and cloud computing by enabling the IoT gateway at the edge layer to process most of the tasks that come from the end layer. As we mentioned, chips that can act as a gateway have limited computation capabilities. To overcome this challenge, the gateway offloads tasks to the cloud when the gateway is unable to process them.

4.1 The Proposed Model

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.

4.1.1 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 4.1, 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.

$$f(x) = \begin{cases} 0 & 0 \leq x \leq t \\ 1 & x > t \end{cases} \tag{4.1}$$

The second function of the gateway is to process tasks and take actions locally. This is performed by implementing an ANNs algorithm. ANNs uses multiple layers of neural networks to decode higher-level information at other layers based on the input data. For this model, three ANNs 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 three ANNs algorithms are tested: multi-layer perceptron neural networks, and two feedforward neural networks. One of these algorithms will be chosen to be implemented in the gateway based on the analysis results in the evaluation chapter.

Algorithm 1 The Role of IoT Gateway

```
while There is task do
    Data: input values of presenters related to the task
    if CPU utilization less than or equal the threshold then
        use the trained learning algorithm to control the home appliances
    else
        offload the task to the cloud
    end
end
```

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 learning 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

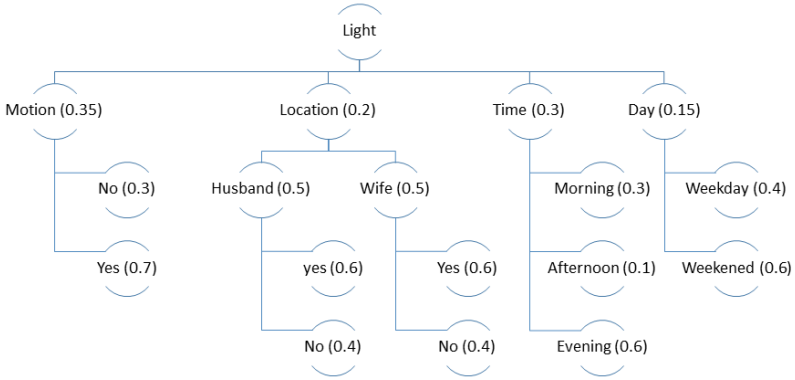

Figure 4.1: Probability Distribution of Light Attributes.

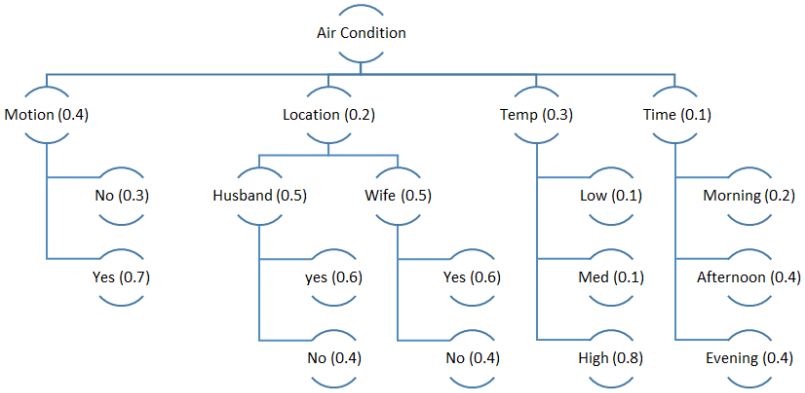

Figure 4.2: Probability Distribution of Air Condition Attributes.

into the ANN algorithm to decide whether to turn the light on or off. The learning algorithm should be trained using a dataset before being implemented on the gateway.

4.1.2 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 (Figures 4.1 and 4.2). The beliefs are calculated according to equation 4.2, 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.

$$P(A) = \sum_{n=1}^{n} x_i * y_i \tag{4.2}$$

4.1.3 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. Figure 4.3 illustrates the distributed intelligence model, and the workflow is explained below with controlling the light. The same steps are applied to control the temperature.

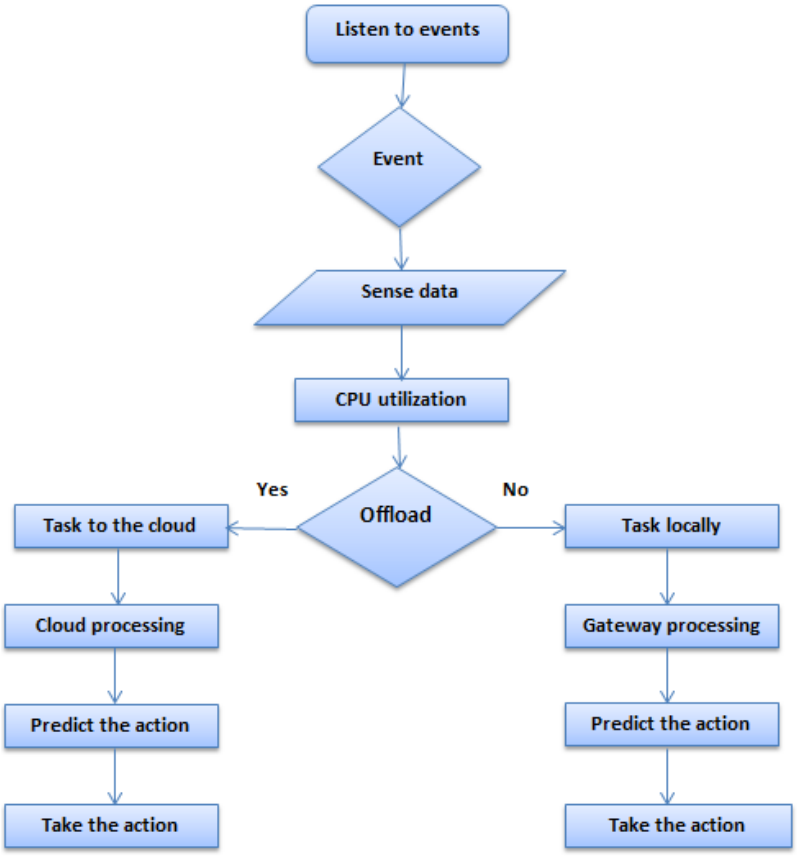

Figure 4.3: Workflow of the IoT Gateway.

- 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 learning 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 [76] is used.
- If the CPU utilization is below the threshold, the gateway processes the task locally. The gateway uses the learning 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 Figure 4.1) 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.

4.2 The Operational Model

To demonstrate our model , a real experiment of DI in a smart home IoT system is conducted. The experiment combines cloud and edge computing to control the

Figure 4.4: Gateway and End Devices in Smart Home System.

light and temperature in the living room. It is assumed that a husband and wife live in the house.

As shown in Figure 4.4, 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.

The implementation has three main phases: end devices layer implementation, edge layer implementation, and cloud layer implementation (Figure 4.5). The implementation of the three phases is explained below. Table 4.1 also summarizes the role of each component of the operational model.

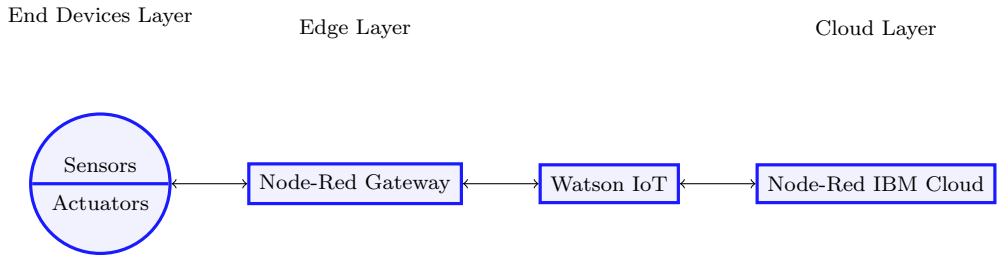

Figure 4.5: Operational Model for Distributed Intelligence.

Table 4.1: Operational model components

Component Name	Component Type	Location	Description
Sensors and Actuators	Physical sensors and actuators	End devices	Used to collect data and control appliances.
Node-RED Gateway	Node-RED	Edge	Creates a data flow application
Watson IoT	IBM IoT Platform	Between edge and cloud	MQTT message broker
Node-RED IBM Cloud	Node-RED	Cloud	Processes IoT sensor data

4.2.1 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 4.2.

Table 4.2: Description of sensors and actuators.

Devices	Producer	Communication
Smart Plug	Sylvania	ZigBee
Motion Sensor	Samsung	ZigBe
Temp Sensor	Samsung	ZigBe
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 either low, medium, or high. The smart plug is used to connect the air conditioner to the home's electricity.

4.2.2 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 [77]: 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) 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.

In order to build the temperature dataset, the gateway saves the parameters of the air condition's status, temperature, motion, the husband's location, the wife's location, and the time. The air conditioner'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.

4.2.3 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 [78]. 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 (Figures 4.1 and 4.2). 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) is connected to the Node-RED Starter application located on IBM Cloud. Thereafter, the gateway is connected to the IoT Platform.

4.2.4 Intelligent IoT Gateway

Providing intelligence to the gateway can be achieved by implementing an ANN algorithm. In this work, three artificial neural networks (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. We built the two feedforward neural networks based on RNNs (e.g LSTMs and GRUs) architectures because they performed well in some works of non-sequential data [79, 80], Although RNNs are designed for processing sequential data.

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 [81], the network with the highest accuracy is selected as the best one.

For the two feedforward neural networks (FFNN1 and FFNN2), the number of units in the hidden layer is specified using equation 4.3 [82], where N_i is the number of input time steps (input), N_o is the number of output nodes (features), N_s 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.

$$N_h = \frac{N_s}{(\alpha(N_i + N_o))} \tag{4.3}$$

Table 4.3: 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	binary crossentropy	binary crossentropy	binary crossentropy
Epochs	10	10	10
Batch size	5	5	5
verbose	1	1	1

In the output layer, for controlling the light, one neuron is used in each algorithm that represents the light status or air condition 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 the three algorithms is categorical crossentropy. The adam optimizer is also used for optimizing the model for the three algorithms. 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 4.3 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.

4.2.5 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 (Figures 4.1 and 4.2). The beliefs are calculated according to equation 4.2, 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.

4.3 Summary

This chapter presented the steps for implementing the DIM that integrates cloud and edge computing in order to develop a robustness model that can deal with the

increasing size of IoT data. The implementation starts by configuring several smart home sensors and devices. The IoT gateway in the edge layer, which provides intelligence and makes decisions closer to the end devices than the cloud can, was built. Node-RED was subsequently installed and configured to connect all the layers. As we mentioned, the Raspberry Pi, which acts as a gateway in the model, has limited computation capabilities. To overcome these challenges, a cloud layer is required. Accordingly, a cloud layer was implemented on IBM Cloud. This way, when the gateway is unable to process a task locally, it can offload the task to the cloud. After receiving a task, the cloud processes them and decides how to control the home appliances.

Chapter 5

Evaluation

This chapter discusses the evaluation method of the DI smart home system shown in Figure 5.1. 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.

5.1 Artificial Neural Networks Algorithms

In this work, 20% of the data is classified through MLPNNs, FFNN1, and FFNN2. 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.

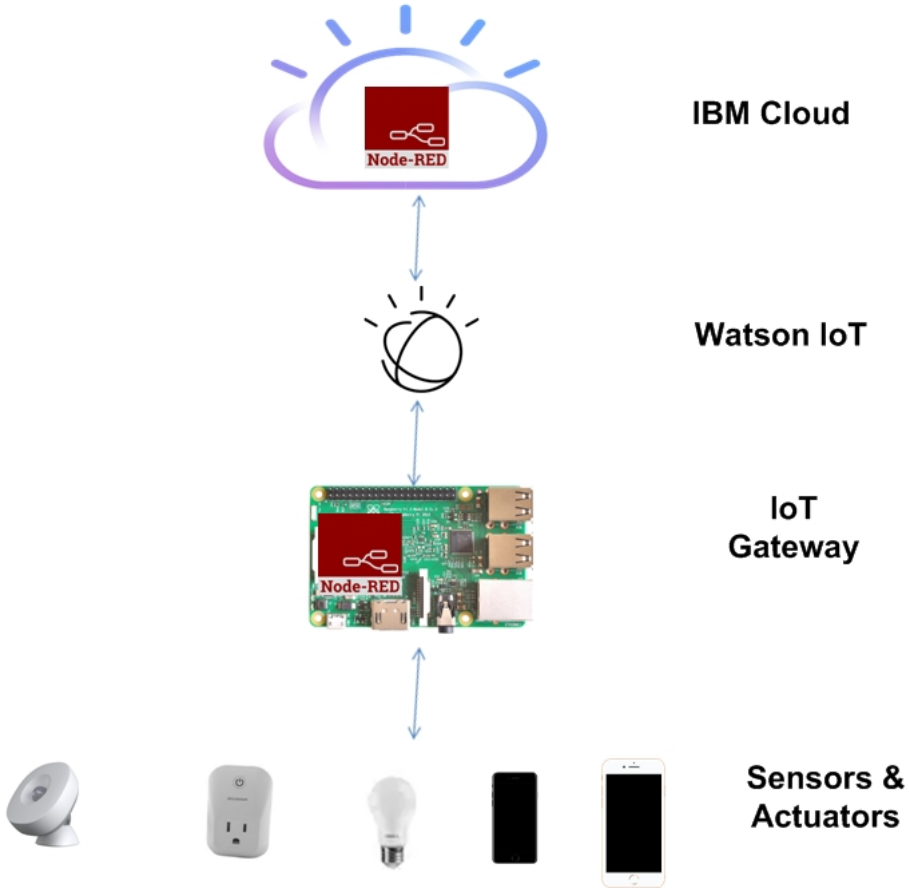

Figure 5.1: DI smart home system.

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 [83]. The ROC curve indicates the TP rate against the FP rate for various threshold settings.

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 TN (i.e., we predicted no, and the decision actually was no). The second is FP (i.e., we predicted yes, but the decision was actually no). The third is FN (i.e., we predicted no, but the decision was actually yes). Finally, TP (i.e., we predicted yes, and the decision actually was yes).

Table 5.1: Confusion Matrix of ANN Algorithms.

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

Table 5.1 represents the confusion matrix, which summarizes the prediction results of the algorithms, which are used to evaluate the algorithms. Although MLPNNs exhibit the highest TN (626), the TN of FFNN1 and FFNN2 are close to it (612 and 603, respectively). Both FFNN2 and FFNN1 are almost three times higher in terms of TP (120 and 117) than MLPNNs (39). Furthermore, MLPNNs exhibit almost five times higher FN (98) than FFNN2 (17) and FFNN1 (20). According to the confusion matrix, FFNN1 and FFNN2 are better for deciding whether to turn the light on or off.

Table 5.2 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. FFNN1 and FFNN2 were the most accurate, with a value of 0.95 recorded for each. Precision is the ratio of correctly predicted positive observations to

Table 5.2: Comparison of the three algorithms.

Parameters	Accuracy	Precision	Recall	F1-Score
MLPNN	0.87	0.28	0.80	0.44
FFNN1	0.95	0.85	0.89	0.87
FFNN2	0.95	0.88	0.83	0.85

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.

As shown in Figure 5.2, 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.

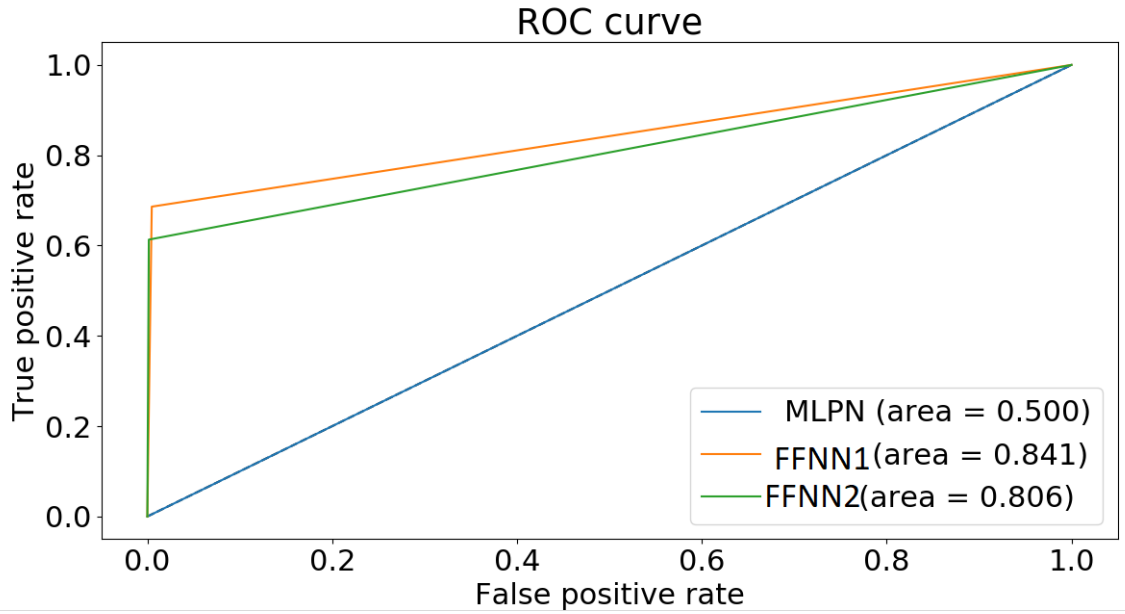

Figure 5.2: ROC Curve for the ANN Algorithms.

5.2 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 and no-offloading scenarios with regard to the operational characteristics.

Figure 5.3 presents the processing times of the first 50 tasks processed locally for

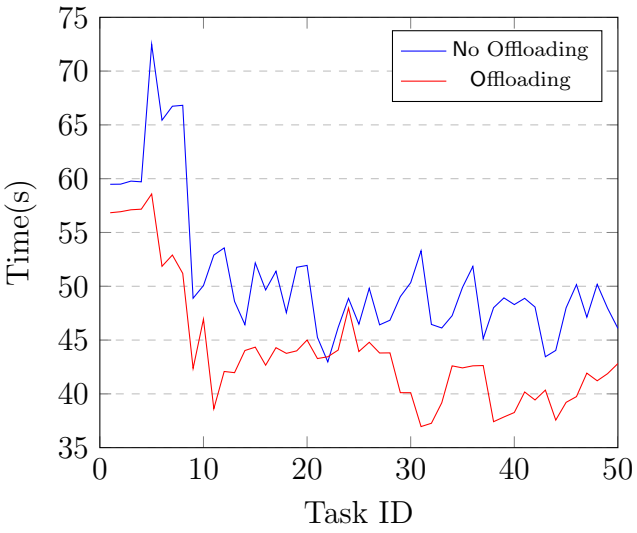

Figure 5.3: Tasks Processing Time.

both no-offloading and offloading. For both curves, the tasks processing time is too high at the beginning because the system was establishing connections with the end devices and IBM cloud.

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 lower 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

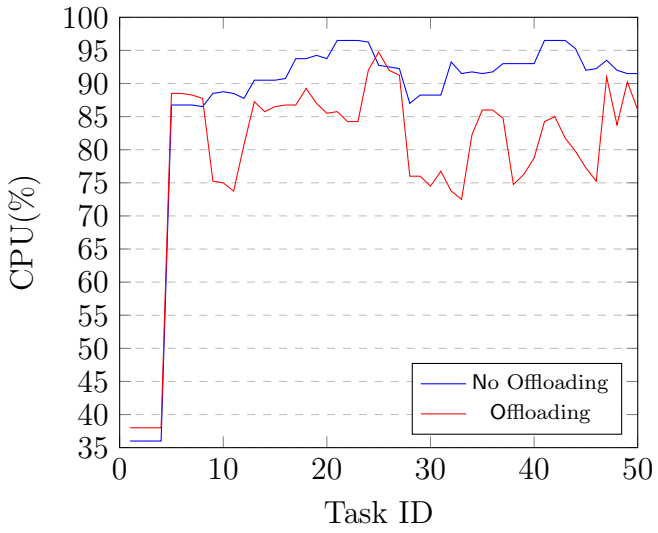

Figure 5.4: Gateway CPU Utilization.

overloaded. Offloading some tasks to the cloud (when the gateway is busy) prevents the gateway from being overloaded and keeps it under control.

As mentioned at the beginning of the chapter, 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).

Figure 5.4 presents the CPU utilization while the first 50 tasks were being processed locally, both for no-offloading and offloading. For both curves, the CPU utilization is too high at the beginning because the system was establishing connections with the end devices and IBM cloud.

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.

Chapter 6

Conclusion and Future Work

The primary goal of this thesis 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 DI-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, and smart farming.

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.

Appendix A

Research Publications

- Rababah B, Alam T, Eskicioglu R. The Next Generation Internet of Things Architecture Towards Distributed Intelligence: Reviews, Applications, and Research Challenges. Journal of Telecommunication, Electronic and Computer Engineering (JTEC), 12(2), 2020 Jul 1.

- Rababah B, Eskicioglu R. Distributed Intelligence Model for IoT Applications Based on Neural Networks. International Journal of Computer Network and Information Security(IJCNIS), accepted.

Bibliography

[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] H. Rahman and R. Rahmani, "Enabling distributed intelligence assisted future internet of things controller (fitc)," *Applied computing and informatics*, vol. 14, no. 1, pp. 73–87, 2018.

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

[4] I. Azimi, A. Anzanpour, A. M. Rahmani, T. Pahikkala, M. Levorato, P. Liljeberg, and N. Dutt, "Hich: Hierarchical fog-assisted computing architecture for healthcare iot," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 16, no. 5s, pp. 1–20, 2017.

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

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

[7] P. Mukalov, O. Zelinskyi, R. Levkovych, P. Tarnavskyi, A. Pylyp, and N. Shakhovska, "Development of system for auto-tagging articles, based on neural network.," in *3rd International Conference on Computational Linguistics and Intelligent Systems*, CEUR-WS, Kharkiv, Ukraine, 2019, pp. 106–115.

[8] 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, 2015.

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

[10] K. Ashton, "That 'internet of things' thing," *RFID journal*, vol. 22, no. 7, pp. 97–114, 2009.

[11] B. P. Rao, P. Saluia, N. Sharma, A. Mittal, and S. V. Sharma, "Cloud computing for internet of things & sensing based applications," in *2012 Sixth International Conference on Sensing Technology (ICST)*, IEEE, 2012, pp. 374–380.

[12] S. Babu, M Chandini, P Lavanya, K. Ganapathy, and V Vaidehi, "Cloud-enabled remote health monitoring system," in *2013 International Conference*

on Recent Trends in Information Technology (ICRTIT), IEEE, 2013, pp. 702–707.

[13] R. Yu, Y. Zhang, S. Gjessing, W. Xia, and K. Yang, "Toward cloud-based vehicular networks with efficient resource management," *IEEE Network*, vol. 27, no. 5, pp. 48–55, 2013.

[14] M. Hassanalieragh, A. Page, T. Soyata, G. Sharma, M. Aktas, G. Mateos, B. Kantarci, and S. Andreescu, "Health monitoring and management using internet-of-things (IoT) sensing with cloud-based processing: Opportunities and challenges," in *2015 IEEE International Conference on Services Computing*, IEEE, 2015, pp. 285–292.

[15] A. Bagula, C. Lubamba, M. Mandava, H. Bagula, M. Zennaro, and E. Pietrosemoli, "Cloud based patient prioritization as service in public health care," in *2016 ITU Kaleidoscope: ICTs for a Sustainable World (ITU WT)*, IEEE, 2016, pp. 1–8.

[16] A. Darwish, A. E. Hassanien, M. Elhoseny, A. K. Sangaiah, and K. Muhammad, "The impact of the hybrid platform of internet of things and cloud computing on healthcare systems: Opportunities, challenges, and open problems," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, no. 10, pp. 4151–4166, 2019.

[17] E. Cavalcante, J. Pereira, M. P. Alves, P. Maia, R. Moura, T. Batista, F. C. Delicato, and P. F. Pires, "On the interplay of internet of things and cloud computing: A systematic mapping study," *Computer Communications*, vol. 89, pp. 17–33, 2016.

[18] M. Aazam, I. Khan, A. A. Alsaffar, and E.-N. Huh, "Cloud of things: Integrating internet of things and cloud computing and the issues involved," in *Proceedings of 2014 11th International Bhurban Conference on Applied Sciences & Technology (IBCAST) Islamabad, Pakistan, 14th-18th January, 2014*, IEEE, 2014, pp. 414–419.

[19] S. M. Babu, A. J. Lakshmi, and B. T. Rao, "A study on cloud based internet of things: Cloudiot," in *2015 global conference on communication technologies (GCCT)*, IEEE, 2015, pp. 60–65.

[20] F. Li, M. Vögler, M. Claeßens, and S. Dustdar, "Towards automated iot application deployment by a cloud-based approach," in *2013 IEEE 6th International Conference on Service-Oriented Computing and Applications*, IEEE, 2013, pp. 61–68.

[21] F. Computing, "The internet of things: Extend the cloud to where the things are," *Cisco White Paper*, 2015.

[22] F. Bonomi, R. Milito, J. Zhu, and 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*, 2012, pp. 13–16.

[23] Y.-Y. Shih, W.-H. Chung, A.-C. Pang, T.-C. Chiu, and H.-Y. Wei, "Enabling low-latency applications in fog-radio access networks," *IEEE network*, vol. 31, no. 1, pp. 52–58, 2016.

[24] K. X. X. F. H. W. Ning Zhaolong and X. Wang, "Green and sustainable cloud of things: Enabling collaborative edge computing," *IEEE Communications Magazine*, vol. 57, no. 1, pp. 72–78, 2018.

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

[26] D. Bimschas, H. Hellbrück, R. Mietz, D. Pfisterer, K. Römer, and T. Teubler, "Middleware for smart gateways connecting sensornets to the internet," in *Proceedings of the 5th International Workshop on Middleware Tools, Services and Run-Time Support for Sensor Networks*, 2010, pp. 8–14.

[27] S. Guoqiang, C. Yanming, Z. Chao, and Z. Yanxu, "Design and implementation of a smart iot gateway," in *2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*, IEEE, 2013, pp. 720–723.

[28] J. Bian, D. Fan, and J. Zhang, "The new intelligent home control system based on the dynamic and intelligent gateway," in *2011 4th IEEE International Conference on Broadband Network and Multimedia Technology*, IEEE, 2011, pp. 526–530.

[29] W. Shen, Y. Xu, D. Xie, T. Zhang, and A. Johansson, "Smart border routers for ehealthcare wireless sensor networks," in *2011 7th International Conference on Wireless Communications, Networking and Mobile Computing*, IEEE, 2011, pp. 1–4.

[30] V. Stantchev, A. Barnawi, S. Ghulam, J. Schubert, and G. Tamm, "Smart items, fog and cloud computing as enablers of servitization in healthcare," *Sensors & Transducers*, vol. 185, no. 2, p. 121, 2015.

[31] A. M. Rahmani, T. N. Gia, B. Negash, A. Anzanpour, I. Azimi, M. Jiang, and P. Liljeberg, "Exploiting smart e-health gateways at the edge of healthcare internet-of-things: A fog computing approach," *Future Generation Computer Systems*, vol. 78, pp. 641–658, 2018.

[32] A. Computing *et al.*, "An architectural blueprint for autonomic computing," *IBM White Paper*, vol. 31, no. 2006, pp. 1–6, 2006.

[33] A. Badlani and S. Bhanot, "Smart home system design based on artificial neural networks," in *Proceedings of the World Congress on Engineering and Computer Science*, IAENG, vol. 1, San Francisco, USA, 2011, pp. 19–21.

[34] A. Hussein, M. Adda, M. Atieh, and W. Fahs, "Smart home design for disabled people based on neural networks," *Procedia Computer Science*, vol. 37, pp. 117–126, 2014.

[35] H. D. Mehr, H. Polat, and A. Cetin, "Resident activity recognition in smart homes by using artificial neural networks," in *4th international istanbul smart grid congress and fair (ICSG)*, IEEE, Istanbul, Turkey, 2016, pp. 1–5.

[36] J. Park, K. Jang, and S.-B. 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, Singapor, 2018, pp. 155–160.

[37] P. Wang, F. Ye, and X. Chen, "A smart home gateway platform for data collection and awareness," *IEEE Communications magazine*, vol. 56, no. 9, pp. 87–93, 2018.

[38] R. Calegari, G. Ciatto, S. Mariani, E. Denti, and A. Omicini, "Lpaas as micro-intelligence: Enhancing iot with symbolic reasoning," *Big Data and Cognitive Computing*, vol. 2, no. 3, p. 23, 2018.

[39] A. S. Allahloh and S. Mohammad, "Development of the intelligent oil field with management and control using iiot (industrial internet of things)," in *2018 2nd IEEE International Conference on Power Electronics, Intelligent Control and Energy Systems (ICPEICES)*, IEEE, 2018, pp. 815–820.

[40] T. Alsboui, Y. Qin, R. Hill, and H. Al-Aqrabi, "Enabling distributed intelligence for the internet of things with iota and mobile agents," *Computing*, pp. 1–19, 2020.

[41] X. Huang, P. Craig, H. Lin, and Z. Yan, "Seciot: A security framework for the internet of things," *Security and communication networks*, vol. 9, no. 16, pp. 3083–3094, 2016.

[42] D. Lund, C. MacGillivray, V. Turner, and M. Morales, "Worldwide and regional internet of things (iot) 2014–2020 forecast: A virtuous circle of proven value and demand," *International Data Corporation (IDC), Tech. Rep*, vol. 1, p. 9, 2014.

[43] M. Ben-Daya, E. Hassini, and Z. Bahroun, "Internet of things and supply chain management: A literature review," *International Journal of Production Research*, vol. 57, no. 15-16, pp. 4719–4742, 2019.

[44] I. Lee and K. Lee, "The internet of things (iot): Applications, investments, and challenges for enterprises," *Business Horizons*, vol. 58, no. 4, pp. 431–440, 2015.

[45] S Muthuramalingam, A Bharathi, N Gayathri, R Sathiyaraj, B Balamurugan, *et al.*, "Iot based intelligent transportation system (iot-its) for global perspective: A case study," in *Internet of Things and Big Data Analytics for Smart Generation*, Springer, 2019, pp. 279–300.

[46] B. L. R. Stojkoska and K. V. Trivodaliev, "A review of internet of things for smart home: Challenges and solutions," *Journal of Cleaner Production*, vol. 140, pp. 1454–1464, 2017.

[47] N. Saleh, A. Kassem, and A. M. Haidar, "Energy-efficient architecture for wireless sensor networks in healthcare applications," *IEEE Access*, vol. 6, pp. 6478–6486, 2018.

[48] S. T. Somayya Madakam R. Ramaswamy, "Internet of Things (IoT): A literature review," *Journal of Computer and Communications*, vol. 3, no. 05, p. 164, 2015.

[49] C. M. Roberts, "Radio Frequency Identification (RFID)," *Computers & security*, vol. 25, no. 1, pp. 18–26, 2006.

[50] X. Jia, Q. Feng, T. Fan, and Q. Lei, "RFID technology and its applications in Internet of Things (IoT)," in *2012 2nd international conference on consumer electronics, communications and networks (CECNet)*, IEEE, 2012, pp. 1282–1285.

[51] M. A. Vieira, A. B. da Cunha, and D. C. da Silva, "Designing wireless sensor nodes," in *International Workshop on Embedded Computer Systems*, Springer, 2006, pp. 99–108.

[52] B. Prabhu, N Balakumar, and A Antony, "Evolving constraints in military applications using wireless sensor networks," *International Journal of Innovative Research in Computer Science & Technology (IJIRCST)*, vol. 5, no. 1, 2017.

[53] F. Kiani and A. Seyyedabbasi, "Wireless sensor network and internet of things in precision agriculture," *Int. J. Adv. Comput. Sci. Appl*, vol. 9, no. 8, pp. 220–226, 2018.

[54] R Gilligan, S Thomson, J Bound, and W Stevens, *Rfc2553: Basic socket interface extensions for ipv6*, 1999.

[55] G. Montenegro, N. Kushalnagar, J. Hui, D. Culler, *et al.*, "Transmission of ipv6 packets over ieee 802.15. 4 networks," *Internet proposed standard RFC*, vol. 4944, p. 130, 2007.

[56] N. Park, H. Hu, and Q. Jin, *Security and privacy mechanisms for sensor middleware and application in internet of things (IoT)*, 2016.

[57] S. Bandyopadhyay, M. Sengupta, S. Maiti, and S. Dutta, "A survey of middleware for internet of things," in *Recent trends in wireless and mobile networks*, Springer, 2011, pp. 288–296.

[58] Y. Yang, Y. Huang, X. Ma, and J. Lu, "Enabling context-awareness by predicate detection in asynchronous environments," *IEEE Transactions on Computers*, vol. 65, no. 2, pp. 522–534, 2015.

[59] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, "Ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments," *Software: Practice and Experience*, vol. 47, no. 9, pp. 1275–1296, 2017.

[60] M. Whaiduzzaman, M. Sookhak, A. Gani, and R. Buyya, "A survey on vehicular cloud computing," *Journal of Network and Computer applications*, vol. 40, pp. 325–344, 2014.

[61] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[62] M. T. Beck and M. Maier, "Mobile edge computing: Challenges for future virtual network embedding algorithms," in *Proc. The Eighth International Conference on Advanced Engineering Computing and Applications in Sciences (ADVCOMP 2014)*, vol. 1, 2014, p. 3.

[63] D. Floye, *The vital role of edge computing in the internet of things.* `https://wikibon.com/the-vital-role-of-edge-computing-in-the-internet-of-things/`, Last accessed on 2021-01-23, 2015.

[64] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, *et al.*, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[65] J. Le, *The 10 neural network architectures machine learning researchers need to learn*, `https://medium.com/cracking-the-data-science-interview/a-`

gentle-introduction-to-neural-networks-for-machine-learning-d5f3f8987786, year=2018, Accessed: 2020-07-04,

[66] A. Obuchowski, *Understanding neural networks 2: The math of neural networks in 3 equations*, https://becominghuman.ai/understanding-neural-networks-2-the-math-of-neural-networks-in-3-equations-6085fd3f09df, year=2020, Accessed: 2020-08-04,

[67] N. Donges, *A guide to rnn: Understanding recurrent neural networks and lstm*, https://builtin.com/data-science/recurrent-neural-networks-and-lstm?fbclid=IwAR3HrntzKpV1lX_hjqGU2u4WOFkNxQvlnmavl9Xrzan_lWg_x41dRaxJk6s, year=2019, Accessed: 2020-07-04,

[68] C. Olah, *Understanding lstm networks*, (accessed April 20, 2020). [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[69] M. Phi, *Illustrated guide to lstm's and gru's: A step by step explanation*, (accessed April 23, 2020). [Online]. Available: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21.

[70] S. Kostadinov, *Understanding gru networks*, (accessed April 29, 2020). [Online]. Available: https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be.

[71] nodered.org, *About: Node-red*, (accessed April 14, 2020). [Online]. Available: https://nodered.org/about/.

[72] C. Rattanapoka, S. Chanthakit, A. Chimchai, and A. Sookkeaw, "An mqtt-based iot cloud platform with flow design by node-red," in *2019 Research, Invention, and Innovation Congress (RI2C)*, IEEE, 2019, pp. 1–6.

[73] I. T. Handoyo and D. I. Sensuse, "Knowledge-based systems in decision support context: A literature review," in *2017 4th International Conference on New Media Studies (CONMEDIA)*, IEEE, 2017, pp. 81–86.

[74] R. Challen, J. Denny, M. Pitt, L. Gompels, T. Edwards, and K. Tsaneva-Atanasova, "Artificial intelligence, bias and clinical safety," *BMJ Qual Saf*, vol. 28, no. 3, pp. 231–237, 2019.

[75] F. Zantalis, G. Koulouras, S. Karabetsos, and D. Kandris, "A review of machine learning and IoT in smart transportation," *Future Internet*, vol. 11, no. 4, p. 94, 2019.

[76] R. Sosa, C. Kiraly, and J. D. P. Rodriguez, "Offloading execution from edge to cloud: A dynamic node-red based approach," in *10th IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, IEEE, Nicosia, Cyprus, 2018, pp. 149–152.

[77] raspberrypi, *Raspberry pi 3 model b*, `https://www.raspberrypi.org/products/raspberry-pi-3-model-b/`, Accessed: 2020-08-04.

[78] IBM, *Ibm cloud*, `https://www.ibm.com/cloud`, Accessed: 2020-4-15.

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

[80] 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.

[81] M. J. Berry and G. Linoff, "Data mining techniques, john wiley & sons," *Inc., 1997. 464*, 1997.

[82] K. Eckhardt, *Choosing the right hyperparameters for a simple lstm using keras*, `https://towardsdatascience.com/choosing-the-right-hyperparameters-for-a-simple-lstm-using-keras-f8e9ed76f046`, year=2018, Accessed: 2020-4-15.

[83] T. Fawcett, "An introduction to roc analysis," *Pattern recognition letters*, vol. 27, no. 8, pp. 861–874, 2006.