People's Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research

UNIVERSITY OF
ECHAHID HAMA LAKHDER EL OUED
FACULTY OF EXACT SCIENCES
Computer Science department

End of study memory

Presented for the Diploma of

ACADEMIC MASTER

Domain : **Mathematics and Computer Science**

Industry : **Computer Science**

Specialty: **Distributed Systems and Artificial Intelligence**

Presented by: ATIA Brahim
BARIKA Bachir

Them

Modeling

An Event Analysis System For IoT

.

Defended on *01-10-2020* before the jury:

Mr. Abdelkader Laouid	MCA	President
Mr. Mounir Beggas	MCA	Supervisor
Mr. Mouadh Bali	MAA	Co-Supervisor
Mr. Ismail kertiou	MAA	Reporter

Academic year 2019/2020

Made with LaTeX.

Dedicate

I am most grateful to my two parents for their love , prayers , caring, and sacrifices for educating and empowering me for my future.
With all the love, respect and gratitude I dedicate this work to my dear wife, who has endured my busyness and absence. And to my dear, uprooted children, Suleiman, Maymouna, and Safaa.
To my friend who did not get tired and was not defeated until he made me re-enroll in the study and complete my scientific career.
I will never forget that you have been part of my success.

Yours lover Brahim Atia

Dedicate

Thanks to Allah first-then the efforts of my loved ones who helped me a lot with their advice and efforts when we were going to offer this job.

Dedicate this work to my dear mother,To my dear Father.

To my brother oussama and aissa, to my sisters.

To my uncle mohammed ,to my aunt khira, To my childhood friends bachar, abdelraouf and zakaria.

To my Supervisors teacher Bali mouadh and doctor beggas monir.

To all my colleagues inside and outside university.

Barika bachir

Acknowledgments

First and foremost, thanks and glory to God, our lord, and the source of Inspiration, wisdom and understanding, the Almighty, for the blessings among all my research work to complete this research.

We would like to express our profound and sincere appreciation to our research supervisors,Dr. Mounir Beguess and Mr. Mouadh Bali , for their honesty and inspiration.

And we would like to thank both our teachers and members of the jury for their willingness to discuss and enrich the material of this study.

Our special thanks go to all the people who have supported us, encouraged us and just give us any help even with a few words to stand up again and again all the time.

B. Atia

B. Barika

Abstract

In this study, we integrate intelligence technology, the Internet of Things, and distributed systems in a single structure to form a model that incorporates the most important characteristics and advantages of these technologies and produces a product that surpasses the challenges presented by the nature of their work. To achieve this goal, we adopt the Internet of Things as a work ground. And we divide the processor architecture between fog and cloud each for the best work load. For the transfer and processing of data, we use Apache Kafka and Apache Flink for their functionality mode. And finally, we incorporate the artificial intelligence technique to generate the inference model .

The work goes through several phases, each including theoretical comparisons that made the option that responds to the largest number of characteristics of our model in its method of operation to be weighted. Except for the last step, as a tracking technique, we make an applied comparison between the different candidate options to decide the appropriate model according to the case being handled.

The result,we designed a model that can be adopted to develop a monitoring system that relies on the Internet of things as a source of data and deep learning as a core for analyzing and extracting data directives , within a complex structure of distributed system that allow scalability and availability of service To the fullest extent this is all in the context of fault tolerance.Also we approved by experimental that Multilayer perceptron neural network is the appropriate method for our use case.

Keywords: internet of things, connected object, Sensor networks, IoT Analytic, Machine Learning.

خلاصة

في هذه الدراسة ، نقوم بدمج تقنية الذكاء ، وإنترنت الأشياء ، والأنظمة الموزعة في بنية واحدة لتشكيل نموذج يدمج أهم خصائص ومزايا هذه التقنيات وينتج منتجًا يفوق التحديات التي تطرحها طبيعة هذه التقنيات.

لتحقيق هذا الهدف ، نعتمد إنترنت الأشياء كأرضية عمل. ونقسم بنية المعالج بين الضباب والسحابة للحصول على أفضل حمل عمل. لنقل البيانات ومعالجتها ، نستخدم ايثه كفك و ايثه ذلنك لوضعهما الوظيفي. وأخيرًا ، قمنا بدمج تقنية الذكاء الاصطناعي لتوليد نموذج الاستدلال.

يمر العمل بعدة مراحل ، كل منها يتضمن المقارنات النظرية التي جعلت الخيار الذي يستجيب لأكبر عدد من خصائص نموذجنا في طريقة عمله ليتم ترجيحه. باستثناء الخطوة الأخيرة ، كأسلوب يتبع ، نجري مقارنة تطبيقية بين خيارات المرشحين المختلفة لتحديد النموذج المناسب وفقًا للحالة التي يتم التعامل معها.

النتيجة ، قمنا بتصميم نموذج يمكن اعتماده لتطوير نظام مراقبة يعتمد على إنترنت الأشياء كمصدر للبيانات والتعلم العميق كأساس لتحليل واستخراج توجيهات البيانات ، ضمن بنية معقدة من النظم الموزعةالذي يسمح قابلية التوسع وتوافر الخدمة إلى أقصى حد ، كل هذا في سياق التسامح مع الخطأ. كما وافقنا من خلال التجربة على أن الشبكة العصبية متعددة الطبقات هي الطريقة المناسبة لحالة الاستخدام الخاصة بنا.

الكلمات المفتاحية: إنترنت الأشياء ، كائن متصل ، شبكات الاستشعار ، تحليل إنترنت الأشياء ،تعلم آلة، التعلم العميق.

Résumé

Dans cette étude, nous intégrons la technologie du intelligence artificial, l'Internet des objets et les systèmes distribués dans une structure unique pour former un modèle qui intègre les caractéristiques et avantages les plus importants de ces technologies et produit un produit qui surpasse les défis présentés par la nature de leur travail.

Pour atteindre cet objectif, nous adoptons l'Internet des objets comme plate-forme du travail. Et nous divisons l'architecture du processing entre le fog et le cloud chacun pour dans sa meilleure avantages. Pour le transfert et le traitement des données, nous utilisons Apache Kafka et Apache Flink pour leur mode de fonctionnalité. Et enfin, nous intégrons la technique d'intelligence artificielle pour générer le modèle d'inférence.

Le travail passe par plusieurs phases, chacune comprenant des comparaisons théoriques qui designe l'option qui répond au plus grand nombre de caractéristiques de notre modèle dans son mode de fonctionnement à pondérer. Sauf pour la dernière étape, en tant que technique a suivi, nous faisons une comparaison appliquée entre les différentes options candidates pour décider quel est le modèle approprié en fonction du cas traité.

Le résultat, nous avons conçu un modèle qui peut être adopté pour développer un système de surveillance qui s'appuie sur l'Internet des objets comme source de données et le deep learning comme noyau pour analyser et extraire les directives de données, au sein d'une structure complexe de système distribué qui permet évolutivité et disponibilité du service Dans toute la mesure du possible, tout cela est dans le contexte de la tolérance aux pannes. Il faut aussi que l'expérimental approuve que le réseau de neurones perceptron multicouche est la méthode appropriée pour notre cas d'utilisation. Nous avons également approuvé par expérimentation que le réseau de neurones perceptron multicouche est la méthode appropriée pour notre cas d'utilisation.

Mots clés: Internet des objets, objet connecté, Réseaux de capteurs, IoT Analytic, Machine learning, Deep Learning.

Contents

List of Figures

List of Tables

Part I

State of the art

General Introduction

One of the most important applications in the field of the Internet is the use of digital surveillance sensors that record the different characteristics of phenomena in forms that vary from measurement, sound, image ... etc. And the conversion of this information to computing points where it is subject to filtering, classification and in-depth analysis in order to help predict and form a pillar for decision-making.

When using digital sensing and monitoring, we will be in the process of generating a large volume of data and having to deal with it in real time , taking into account the difficulties represented in the diversity of data, the possibility of defects or distortions, and the challenges arising under these conditions of the imperative of speed of response and sensitivity.

In this work, we propose an architectural structure that allows us to implement a system that responds to the most stringent monitoring requirements and that can be developed and promoted in order to rebuild and enlarge IT infrastructures like smart cities which are considered as one of the most sophisticated and rigorous digital systems in terms of complexity and requirements.

The aim of this study is to provide a solution in the style of distributed systems over then a stand-alone application to take advantage of Available open source tools that provide: availability, scalability, fault tolerance features are therefore very important in such tools.

This thesis is structured as follows. Initially, we provide an overview of the concepts that we will follow, including the concept of digital surveillance, its spatial and temporal nature, of the objectives behind it. We also discussed the concept of IoT-based monitoring and the areas of application in which it was used. After that, we gave an overview of the standards and conditions that must be met in these digital systems, a reminder of the challenges and technical limitations that arise from them. We concluded this part by giving an overview of the various possible solutions and the different architectural structures and related work that have been carried out in this direction.

While we dedicated the second part to clarifying and explaining our proposed architectural structure, and giving our proves that supports our choices in its type, tools and the distribution of components, also functions on those parts.

The Beside, part contains the experimental side, which includes the experiments carried out on the subject chosen as the main part of the application of proposed architecture for the monitoring system, which is the type of in-depth learning, the various comparisons and workloads made to specify the impact on its performance, both positively and negatively.

Chapter 1

Internet of things (IoT)

1.1: Introduction

In this part, we discuss the definition of the term Internet of Things in its various aspects and the most famous sensors of this technology. because we will adopt this technique in order to create and install our proposed model. Initially, we provide a brief definition of IoT, then a definition of a similar technology called the Cyber-Physical System, but it differs in coverage and use in order to increase the clarity of positive aspects of the Internet of Things Technology. Secondly, we list the different aspects of applications that adopt Internet of Things technology, then discuss the used protocols for data transfer and exchange. The penultimate part is a census of Finally, we address the advantages of this technology generating data from simple signals to high-definition video. Also, we mention its main challenges in the recent researches and development area.

.

1.2: Definition

Internet of Things (IoT) encapsulates a vision of world in which billions of objects with embedded intelligence, communication means, sensing and actuation capabilities will connect over IP (Internet Protocol) networks[20].
Another definition of IoT is "The Internet of Things allows people and things to be connected Anytime, Anyplace, with Anything and Anyone, ideally using Any path/network and Any service"[41].

The main idea of IoT The main idea of IoT is to physically connect anything/everything (e.g., sensors, devices,machines, people, animals, trees) and processes over the Internet for monitoring and/or control functionality. Connections are not limited to information sites, they're actual and physical connections allowing users to reach "things" and take control when needed. Hence, connecting objects together is not an objective by itself, but gathering intelligence from such objects to enrich products and services is[44].
The IoT paradigm is defined as an intersection of three vision areas:

- Internet oriented vision;
- Things oriented vision; and
- Semantic oriented vision [14].

IoT can connect ubiquitous devices and facilities with various networks to offer effective and secure services for all applications anytime and anywhere .Also, The data generated from IoT devices can be used in finding potential research trends and investigating the impact of certain events or decisions. These data are processed using various analytic tools[61].

To generate benefits from IoT, enterprises must create a platform where they can collect, manage, and analyze a massive volume of sensor data in a scalable and cost-effective manner[45] IoT should cover all things in large-scale networks, in which various networks should coexist, and are able to interact with each other via various gateways and middle wares, supported by the complex control plane[36].

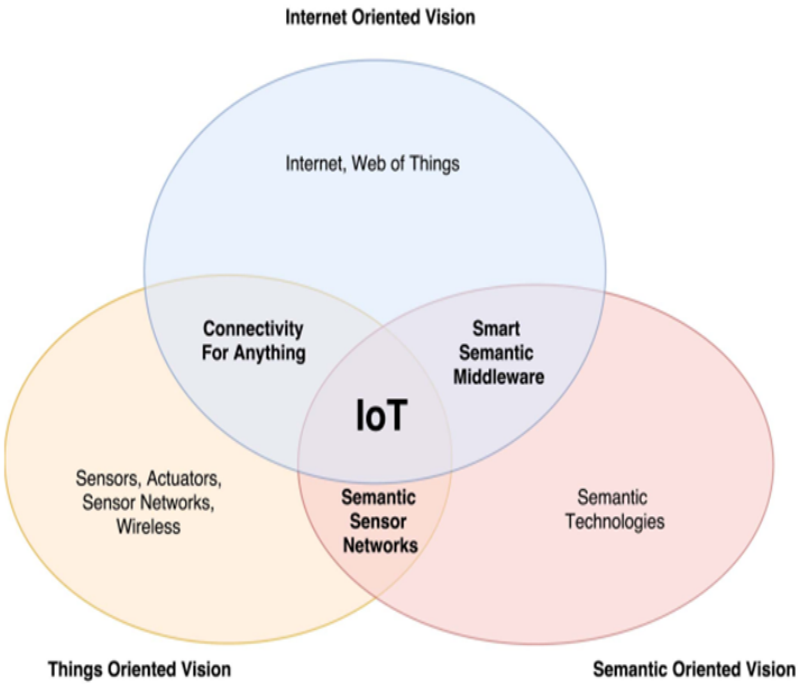

Figure 1.1: Different visions intersection: IoT [14]

1.3: Sensors

In this section we will focus only on the group of devices used in IoT that related with our work, which oriented for data collecting and measurement of features from different phenomena.

Definition

A sensor is a device (in majority electronic) that Discover or determine events or changes within its physical environment (e.g., temperature, humidity, sound, heat, pressure, flow, magnetism, chemical and biochemical elements or changes) and gives a corresponding output. Most sensors acquire analog inputs and fetch digital, often electrical, outputs. Because the sensing element, on its own, In general produces analog output by it nature, an analog-to-digital converter is often present on these devices.

Sensors are comparable to the human five senses. They form the front end of the IoT devices, i.e., "Things." Sensors are very crucial in every IoT vertical (e.g., smart cities, smart grid, healthcare, agriculture, security and environment monitoring, and smart parking) as they bridge the world's physical objects with the Internet[44].

There are two categories of Sensors,the first is very simple have only a core function to collect and transmit data, the second named smart has some sort of additional functionality to filter duplicate data and only notify the IoT gateway when very specific conditions are met. The last type requires that the sensor itself would be equipped with some programming logic. In the case of smart sensor, the device requires at least three elements—sensor(s), micro controllers, and connectivity to send filtered data to IoT gateway or other systems. Figure1.2 shows the components for smart sensor.

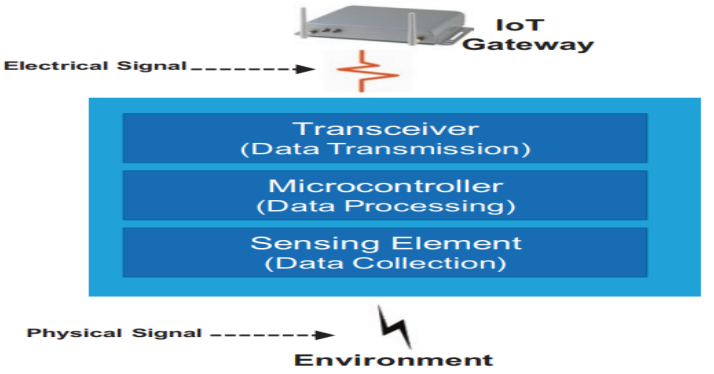

Figure 1.2: Components of smart sensors

Sensor Types

There are many types of proprietary and non proprietary sensors. The current IoT trend is to move away from proprietary and closed systems and embrace IP-based sensor networks. This allows native connectivity between wireless sensor networks and the Internet,[44].

There are many different types of sensors across various technologies. The most common of which include:

1. **Temperature Sensors:** they performing the same work of thermometer except the display of measurement.This category includes four types whom are :

   (a) Thermocouple Sensors: produces a voltage as a result of the thermoelectric effect, it is made by joining two dissimilar metals at one end.

   (b) Resistance Temperature Detector (RTD) Sensors: their resistance changes with temperature, they have been widly used in laboratory and industrial environment cause of their robustness and accuracy.

   (c) Thermistors: Similar to the RTD, but are made from semiconductor materials.

(d) Semiconductor Sensors: offer high accuracy and high linearity over an operating range of degree,and can also include signal processing circuitry within the same package as the sensor.Figure1.3 demonstrate an examples.

Figure 1.3: Examples of temperature sensors plus applications

2. **Pressure Sensors:** are used to measure the pressure of gases or liquids including water level, flow, speed, and altitude. A pressure sensor typically acts as a transducer where it generates a signal as a function of the pressure imposed.

   Touchscreen smartphones, tablets, and computers come with various pressure sensors. Whenever slight pressure is applied on the touch screen through a finger, tiny pressure sensors (typically multiple sensors located at the corners of the screen; see Fig.1.4 determine where exactly pressure is applied and consequently generate an output signal that informs the processor. Pressure sensors have also been widely used in automotive applications to measure fluid level, airbag, and anti lock braking system, in biomedical applications to sense blood pressure, in aviation to maintain a balance between the atmospheric pressure and the control systems of the airplanes, and in submarines to estimate depth and ensure proper operation of electronic systems and other components. Figure 1.4 shows examples of pressure sensors.

Figure 1.4: Examples of pressure sensors. (Source: Force Sensing & Fitbit)

3. **Flow Sensors:**are used to detect and record the rate of fluid flow in a pipe or a system. They are also used to measure the flow/transfer of heat caused by the moving medium and flow monitoring for high-purity acids.

Figure 1.5: Examples of flow sensor

4. **Level Sensors:**are used to measure the level of fluids continuously or at point values. The element to be measured can be inside a container (Fig.1.6) or can be in its natural form such as a well in an oil rig. There are many uses for level sensors. Ultrasonic level sensors, for instance, are used for non-contact level sensing of highly viscous liquids and even bulk solids. They are also widely used in water treatment applications for pump control and open-channel flow measurement.[32]

Figure 1.6: PTLevel Wireless Tank Level Monitor(Source: Amazon.ca)

5. **Imaging Sensors:** are sophisticated sensors used in digital cameras, medical imaging machines, and night vision equipment. They are utilized to measure image information by capturing and then converting variable attenuation of waves into signals (Fig. 1.7).

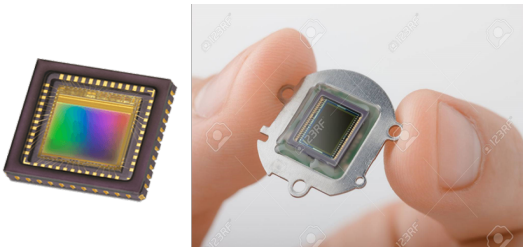

Figure 1.7: Examples of imaging sensors. (Source: e2v & 123rf.com)

6. **Noise Sensors:** Ambient noise sensors continuously monitor noise levels in surrounding environments. When the noise level change, they send electronical signal to an overall ambient noise system to take action. Such action may be an automatic action (e.g., adjust music level) or a simple notification to authorities. Many government agencies have started installing noise sensors to measure noise pollutions or the so-called noise disturbance (excessive noise that may harm humans or animals).

7. **Air Pollution Sensors:**detect and monitor the presence of air pollution in the surrounding environment. They focus on five main components: ozone, particulate matter, carbon monoxide, sulfur dioxide, and nitrous oxide.

8. **Proximity and Displacement Sensors:**detect the presence or absence of objects using electromagnetic fields, light, or sound. There are many types, each suited to specific applications and environments:

   (a) Inductive Sensors: Used for close-range detection of ferrous material.
   (b) Capacitive Sensors: Used for close-range detection of nonferrous material.
   (c) Photoelectric Sensors: Used for long-range target detection.
   (d) Ultrasonic Sensors: Used for long-range detection of targets with difficult surface.

There are so many other types of sensors. Examples include acceleration sensors, Infrared Sensors, Moisture and Humidity Sensors, Speed Sensors, biosensors, gas and chemical sensors, mass sensor, tilt sensors, and force sensors.

Most IoT applications require smaller and smarter sensors with advanced functionality to collect more data, low-power processors, longer battery life, faster response time, and shorter time to market. Sensors are expected to be dynamic in their natural surroundings with embedded ability to collect real-time data[32].

1.4: Architectures

1.4.1: Three layer Architecture

In general and Typically, the architecture of IoT is divided up into three basic layers:firstly perception layer,secondly network layer, and finally application layer, All of three are described below.

1. **Perception layer**(he sensor layer) interacts with physical devices and components through smart devices (RFID, sensors, actuators, etc.). Its main objectives are to connect things into IoT network, and to measure, collect, and process the state information associated with these things via deployed smart devices.

2. **Network layer**(transmission layer) is used to receive the processed information provided by perception layer and determine the routes to carry the data and information to the IoT hub, devices,and applications via integrated networks. The network layer is the most important layer in IoT architecture, because various devices (hub, switching, gateway, cloud computing perform,etc.), and various communication technologies (Bluetooth,WiFi, Long-Term Evolution (LTE), etc.) are incorporated in this layer.

3. **Application layer** (business layer) receives the data transmitted from network layer and uses the data to provide required services or operations.For instance, the application layer can provide the storage service to backup received data into a database, or provide the analysis service to evaluate the received data for predicting the future state of physical devices.

Yet,despite the simplicity of the multi-layer architecture of IoT,functions and operations in the network and application layers are diverse and complex. For example, the network layer not only needs to determine routes and transmit data, but also provide data services (data aggregation, computing, etc.).The application layer not only needs to provide services to customers and devices, it must also provide data services(data mining, data analytics, etc.).

1.4.2: Service-oriented Architectures (SoA)

Is a component-based model,which can be designed to connect different functional units(also known as services) of an application via interfaces and protocols. SoA can focus on designing the workflow of coordinated services, and enable the reuse of software and hardware components, improving the feasibility of SoA for use in designing IoT. architecture. Thus, in a SoA-based IoT architecture, four layers exist and interact with each other [32].

1. **the perception layer** performed as the bottom layer of the architecture, and used to measure, collect, and extract the data associated with physical devices.

2. **The network layer** used to determine routes and provide data transmission support via integrated heterogeneous networks.

3. **The service layer** located between network and application layers,and providing services to support the latter layer. It consists of the following services:

   - discovery used to discover desired service requests.
   - composition to interact with the connected objects, and divide or integrate services to meet service requests in an efficient way
   - management used to manage and determine the trust mechanisms to meet service requests.
   - interfaces are used to support interactions among all provided services.

4. **The application layer** used to support the service requests by users, it can support a number of applications, including smart grid, smart transportation, Smartcities, etc.

1.5: Communication protocols

The role of network layer is to determine the routing that Guarantee the delivery of data in the right time constraint, and offer data transmission support over integrated heterogeneous networks. In the following, some protocols that can enable the reliable and secure communication in IoT.

- **IEEE 802.15.4:** A protocol configured for the physical layer and the MAC layer in wireless personal area networks (WPANs) is to focus on low-rate wireless personal area networks (LR-WPANs) for of all things in a personal area with low energy consumption and low cost, its protocol stack is based on Open System Interconnection (OSI) model and can support bands of 868/915M and 2.4Ghz, thus the data transmission rate on these bands can achieve 20 kbps, 40kbps, and 250 kbps, respectively.

- **Low-power wireless personal area networks(LoWPAN):**Which is organized by a broad number of low-cost devices connected via wireless communications. LoWPAN has a numbe rof advantages (small packet sizes, low power, low bandwidth,etc.). As an enhancement, 6LoWPAN protocol was designed by combining IPv6 and LoWPAN, this latter is suitable to IoT, in which a large number of low cost devices are included. and it have various advantages, including a great connectivity and compatibility with legacy architectures, low-energy consumption, ad-hoc and self-organization, etc.

- **Zigbee :** A wireless network technology, de-signed for short-term communication with low-energy con-sumption, in this protocol five layers are included rather then seven for precedent. In addition of low energy consumption, low cost,low data rate Advantages it have low complexity, reliability, and security, and support multiple topologies, including star, tree,and mesh topologies. the fre-quency band of the physical layer for this protocol can be normally 2.4GHz, and can support end-devices (slaves) up to 65000[32].

- **Z-Wave:** short-term wireless communication technology with the advantages of low cost, low energy consumption, and great reliability, it provide reliable transmission between a control unit and one or more end-devices but no more than 232 nodes (slaves) can be included whom would be controlled by the controller and have routing capability, its frequency band is less than 1GHz (908.42 MHz – 868.42MHz) and can only support 232 end-devices (slaves) but it is so simple in implementation.

1.6: Data transfer technologies

- **Message Queue Telemetry Transport (MQTT):** A messaging protocol using the publish/subscribe technique, which is used to collect measured data on remote sensors and transmit the data to servers[32]. MQTT is a simple and lightweight protocol, and supports many kind of networks with low bandwidth and high latency.It can be implemented in various platforms to connect things in IoT into the Internet, in this way obviously it can be used as a interconnecting protocol between sensors/actuators and servers,the reason why MQTT play an important role in IoT.

- **Constrained Application Protocol (CoAP):**A messaging protocol based on REpresentational State Transfer (REST) architecture. CoAP was proposed to modify some HTTP functions to meet the requirements for IoT, because most of devices in IoT are resources constrained(i.e., small storage and low computing capability), indeed CoAP is the application layer protocol in the 6LoWPAN protocol stack, it goal is to enable resources constrained devices to achieve RESTful interactions, many important features are included in CoAP :Resource observation, block-wise resource transport,resource discovery, interaction with HTTP, and security.

- **Extensible Messaging and Presence Protocol (XMPP):**An instant messaging protocol based on XML streaming protocols[32].It inherits features of XML protocol, and include the following three roles:

  1. the server can achieve the functionality of link management and message routing.
  2. the gateway is used to support the stable communication among heterogeneous systems.
  3. The client can be connected to the server based on TCP/IP protocol and transmit context based on XML streaming protocol.

  principally XMPP used in IoT to support the object-to-object communication with XML-based text messages.

- **Data Distribution Service (DDS):**A publish/subscribe protocol for supporting high performance device-to-device communication[32].And is a data centric protocol, in which multicasting can be supported to achieve great quality of service and high reliability. The broker-less publish/subscribe architecture makes DDS suitable to real-time constrained IoT and device-to-device communications.

- **Advanced Message Queuing Protocol (AMQP):**An open standard message queuing protocol used to provide mes-sage service (queuing, routing, security and reliability, etc.)in the application layer.Can be considered as a message-oriented middleware protocol. Using AMQP, clients can achieve stable communication with message middlewares,even if these clients and middlewares are produced by different programming languages. In addition, it implements various kinds of message exchange architectures,including store and forward,publish and subscribe,message distribution,message queuing, context-based routing, and point-to-point routing.

- **Others :**There are many of protocol whom can play important roles in IoT as well. For example[32]:

    1. Multicast DNS (mDNS) can support the name resolution in IoT applications.
    2. DNS Service Discovery (DNS-SD)can be used by clients to discover desired services in a special network via mDNS.
    3. Routing protocol for Low Power and Lossy networks is a link-independent routing protocol which can be deployed at resource-constrained nodes to deter-mine routes over low power and lossy links.

1.7: Foundational Services and Applications

1. **Image Recognition:** Ubiquitous mobile devices equipped with high resolution cameras facilitate generating images and videos by everyone, everywhere. Moreover, intelligent video cameras are used in many places like smart homes, campuses, and manufacturers for different applications. Image recognition/classification and object detection are among the fundamental usages of such devices.

2. **Speech/Voice Recognition:** With the massive proliferation of smart mobile devices and wearable, automatic speech recognition is becoming a more natural and convenient way for people to interact with their devices[38]. many application detects voice activity by monitoring the noise in the environment. If it is a voice, the system runs the next complexity level recognition network whose task is acoustic modeling to identify if the voice looks like speech or some kind of interest like gun shooting or glass breaks in stolen activities.Exp *Shams et al.*[48].

3. **Indoor Localization:** Providing location aware services, such as indoor navigation and location aware marketing in retailers, are becoming prevalent in indoor environments. Indoor localization may also have applications in other sectors of IoT, such as in smart homes, smart campuses, or hospitals. In a system called DeepFi [59], a DL method over fingerprinting WiFi channel state information data has been utilized to identify user positions.

4. **Physiological and Psychological State Detection:** IoT combined with DL techniques has been also employed to detect various physiological and psychological states of humans, such as pose, activity, and emotions. Many IoT applications incorporate a module for human pose estimation or activity recognition to deliver their services, e.g., smart homes, smart cars, entertainment (e.g., XBox), education, rehabilitation and health support, sports, and industrial manufacturing. For example in a more specification topic Suspicious Behavior and Crime Action Recognition Recognizing abnormal (or concerned) human actions and behaviors from surveillance video streams are critical for fighting against crimes in smart cities [48]

5. **Security and Privacy:** Large-scale surveillance systems are becoming essential part of today civil defense systems. Such systems are widely deployed in our cities and urban communities. Currently, wireless-based video surveillance approach is commonly used to insure a real-time objects monitoring. [10]

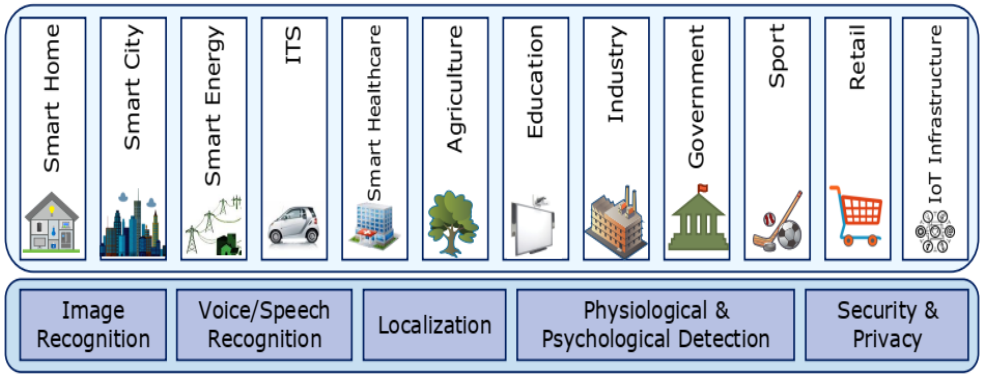

Figure 1.8: IoT foundational services and the applications [38]

In consequences to the diversities domains whom integrate IoT as a base components, the IoT market and its numbers don't stop increasing over years, figure 1.9 Shows the growth of IoT usage in the world.

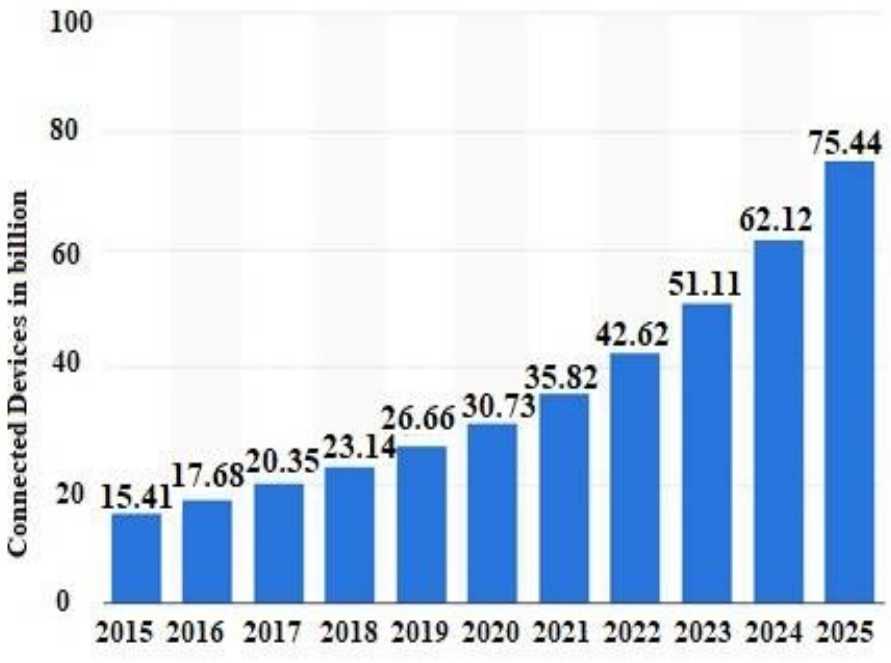

Figure 1.9: Internet of Things (IoT) connected devices from 2015 to 2025 (in billions)[8]

1.8: Artificial Intelligence In IoT Applications

This section aims to introduce and discuss Intelligent Artificial and to demonstrate how it can improve the effectiveness of IoT-based monitoring applications.

IoT devices usually generate large amounts of data and transfer it to the cloud for further processing. These data include multimedia formats such as video, images , and sounds, or structured data such as temperature, vibration, and information on the luminous flux. There are many advanced technologies for the processing of structured data and the automatic control of IoT devices. However, traditional processing technologies, which require complex computing, are not suitable for time-sensitive IoT services.

As deep learning technology improves the efficiency of processing high-volume information such as multimedia, more and more work has begun to be introduced into IoT services and applications.

1.8.1: Data Analytic IoT Features and Specifications

IoT data can be streamed continuously or accumulated as a source of big data flow or stream. Streaming data refers to the data generated or captured within tiny intervals of time this data needs to be promptly analyzed to extract immediate insights and/or make fast decisions. Big data refers to huge data sets that the commonly used hardware and software platforms are not able to store, manage, process, and analyze. These two approaches (Data Streaming and Big Data) should be treated differently since their requirements for analytic response are not the same,insight from big data analytics usually delivered after after a portion of time vary between some minutes to several days of data generation, in contrast insights from streaming data analytics should be ready in a range of few hundreds of milliseconds to few seconds.

In the same context, Developing ubiquitous environments has the Option of Data fusion and sharing which plays a critical role based on IoT data. This role is more essential for time-sensitive IoT applications where a timely operation of fusion of data is needed to bring all pieces of data together and in one single representation for analysis and therefore providing reliable and precise actionable insights. In a simple expression, many IoT applications are based on the presence of multiple pieces of data to be merged and used simultaneously by the application. The latter option suggests the presence of a method which offers the possibility of avoiding two possible situations 1-Some peace absences at an acceptable level. 2-The presence of noise deforming data values.

Data Stream Analytic

In order to achieve streaming Data Analytics many research suggest it should be done on height powered computation like cloud computing, which have theoretically illimited storage and computation. But there is others methods, like data parallelism, where large data set is partitioned into several smaller partitions and processed within parallel analytic simultaneously, also, there is Incremental processing which refers to fetching a small batch of data to be processed quickly in a pipeline of computation tasks.Although these techniques can reduce time latent period to produce a response from the streaming data analytic supporting structure, they are not the best possible solution for time-stringent IoT applications.

We suggest to bring Data analytic as near as possible to the data sources (IoT sensors) is the best solution to ovoid time latency, this could be reached by providing a method which can compute the most insight or give the best decision from the primary part of data collected and consume the minimum computational required. These characteristics are included in AI methods.

Big Data

Because IoT is no thing else then a huge number of sensors whom frequently capture status of their environments continuously , it is considered as the major source of Big Data. The extraction of insights and knowledge is the main purpose for Big Data existence and it has an extreme importance in developing strategies, making decisions for companies since it enables them to gain competitive advantages. Hilbert compares the impact of big data analytic to that of telescope's and microscope's invention for astronomy and biology, respectively. These goals could not be attempted without the IoT big data having the following "6V's" features[38]:

1. **Volume**: Data volume is a determining factor to consider a dataset as big data or traditional massive/ very large data. The quantity of generated data using IoT devices is much more than before and clearly fits this feature.

2. **Velocity**: The rate of IoT big data production and processing is high enough to support the availability of big data in real-time. This justifies the needs for advanced tools and technologies for analytics to efficiently operate given this high rate of data production.

3. **Variety**: Generally, big data comes in different forms and types. It may consist of structured, semi-structured, and unstructured data. A wide variety of data types may be produced by IoT such as text, audio, video, sensory data and so on.

4. **Veracity**: it refers to the quality, consistency, and trustworthiness of the data, which in turn leads to accurate analytics. This property needs special attention to hold for IoT applications, especially those with crowd-sensing data.

5. **Variability**: This property refers to the different rates of data flow. Depending on the nature of IoT applications, different data generating components may have inconsistent data flows. Moreover, it is possible for a data source to have different rates of data load based on specific times.

6. **Value**: Value is the transformation of big data to useful information and insights that bring competitive advantage to organizations. A data value highly depends on both the underlying processes/services and the way that data is treated.

1.8.2: Intelligent Artificial technics

Machine Learning

as described by Arthur Samuel in 1959, is a "Field of study that gives computers the ability to learn without being explicitly programmed."In 1997, Tom Mitchell gave a more formal definition, namely: "A Computer program is said to learn from an experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E."[49]

In other words, Machine learning allows computers to learn without being explicitly programmed; the programmers have the ability to change when exposed to different sets of data. Machine learning techniques are therefore more appropriate to capture hidden insights from Data. The whole process is data-driven and executes at the machine layer by applying complex mathematical calculations to Data sets. Machine learning requires large data sets batches for learning and discovering patterns in data. Commonly used machine learning algorithms are: Linear Regression, Logistic Regression, Decision Tree, Naive Bayes and Random Forest. There are several tools that have been developed for working with Big data for example Jaspersoft, Pentaho, Tableau and Karmasphere. These tools alleviate the problem of scalability by providing solutions for parallel storage and processing[22].

Deep Learning

DL consists of supervised or unsupervised learning techniques based on many layers of Artificial Neural Networks (ANNs) that are able to learn hierarchical representations in deep architectures. DL architectures consist of multiple processing layers. Each layer is able to produce non-linear responses based on the data from its input layer. The functionality of DL is imitated from the mechanisms of human brain and neurons for processing of signals[38].

In recent years, Deep Learning methods have gained more spaces, and invasion addition domains compared to the other traditional machine learning approaches. Such approaches are called now shallow-structured learning architectures versions (i.e., a limited subset) of DL. Although artificial neural networks (ANNs), the first and primitive version, have been introduced in the past decades, the growing trend for deep neural networks (DNNs) started in 2006 when G. Hinton et al. presented the concept of deep belief networks [52]. There after, the state-of-the-art performance of this technology has been observed in different fields of AI including natural language processing, search engines and information retrieval, image recognition,and image retrieval.

Deep Learning architectures

DL techniques have been developed on top of traditional ANNs. Feed-forward Neural Networks (FNNs) [25] (a.k.a Multilayer Perceptrons - MLPs) have been used in the past decades to train systems, but when the number of layers is increased, they become difficult to train [26]. The small size of training data was another factor that results in overfitted models. Moreover, the limitation in computational capabilities in those days prohibited the implementation of efficient deeper FNNs. These computational limitations have been resolved lately due to hardware advances in general and the development of Graphics Processing Units (GPUs) and hardware accelerators specifically.

A DNN consists of an input layer, several hidden layers, and an output layer. Each layer includes several units called neurons. A neuron receives several inputs, performs a weighted summation over its inputs, then the resulting sum goes through an activation function to produce an output. Each neuron has a vector of weights associated to its input size as well as a bias that should be optimized during the training process.

Generally, neural networks with two or more hidden layers that incorporate the recent advanced training algorithms are considered as deep models. One advantage of DL architectures, compared to the traditional ANNs, is that DL techniques can learn hidden features from the raw data.

Summary of Deep Learning Models

Model	Category	Learning model	Typical input data	Characteristics	Sample IoT Applications
AE	Generative	Unsupervised	Various	* Suitable for feature extraction, dimensionality reduction * Same number of input and output units * The output reconstructs input data * Works with unlabeled data	* Machinery fault diagnosis * Emotion recognition
RNN	Discriminative	Supervised	Serial, time-series	* Processes sequences of data through internal memory * Useful in IoT applications with time-dependent data	* Identify movement pattern * Behavior detection
RBM	Generative	Unsupervised, Supervised	Various	* Suitable for feature extraction, dimensionality reduction, and classification * Expensive training procedure	* Indoor localization * Energy consumption prediction
DBN	Generative	Unsupervised, Supervised	Various	* Suitable for hierarchical features discovery * Greedy training of the network layer by layer	* Fault detection classification * Security threat identification
LSTM	Discriminative	Supervised	Serial, time-series, long time dependent data	* Good performance with data of long time lag * Access to memory cell is protected by gates	* Human activity recognition * Mobility prediction
CNN	Discriminative	Supervised	2-D (image, sound, etc.)	* Convolution layers take biggest part of computations * Less connection comparedto DNNs. * Needs a large training dataset for visual tasks.	* Plant disease detection * Traffic sign detection
VAE	Generative	Semi-supervised	Various	* A class of Auto-encoders * Suitable for scarcity of labeled data	* Intrusion detection * Failure detection
GAN	Hybrid	Semi-supervised	Various	* Suitable for noisy data * Composed of two networks:a generator and a discriminator	* Localization and way finding * Image to text
Ladder Net	Hybrid	Semi-supervised	Various	* Suitable for noisy data * Composed of three networks:two encoders and one decoder	* Face recognition * Authentication

Table 1.1: Summary of Deep Learning Models [38].

Summary

In the other side, Volume, variety and velocity, which are the three main features of big data, have caused it to surpass the capabilities of its ingestion, storage, analysis and processing by conventional systems. Big Data means very large data sets, with both structured and unstructured data, that come at high speed from different sources. In addition, not all incoming data are valuable; some of it may not be of good quality and carrying a certain level of uncertainty[22].

However, when considering the fast flows and increasing volumes of data produces by IoT,it leave the traditional methods with the inability to "mine" and pull out rational actions in a timely manner. In addition to the volume's fast growth of data which cannot be handled for traditional analytics, another challenge is the variety of data types and speed at which the data arrives requests novel solutions for data processing and analytics. Compared to conventional structured data of organizations, Big Data does not always fit into neat tables of rows and columns. New data types pertaining to structured and unstructured data have come into existence and which are processed to yield business insights.

Beyond the big data analytics, IoT data calls for another new class of analytics, namely fast and streaming data analytics, to support applications with high-speed data streams and requiring time-sensitive (i.e., real-time or near real-time) actions.Indeed, applications such as autonomous driving, fire prediction, driver/elderly posture (and thus consciousness and/or health condition) recognition demands for fast processing of incoming data and quick actions to achieve their target[38].In other words, Insight from big data analytics can be delivered after several days of data generation,but insight from streaming data analytics should be ready in a range of few hundreds of milliseconds to few seconds.

Here and to respond for new exigences, and to pay-pass the limitations and constraints imply by traditional methods in Big Data Analytics, come the new types of AI technics namely Machine learning and its most advanced version Deep Learning like an alternative with many promises.

1.9: Monitoring

In this section , we will look at the terms that describe the environment that define the objectives that whole project seeks to achieve. The first of these terms is the monitoring term in such a way that we give both a brief literary and a technical meaning, followed by a collection of points which allow for the optimal visibility of these important tasks. For the evaluation model, we have to list its various characteristics and limit the reach that includes the project we have to accomplish. The next step is to review the work carried out or relevant to this project design in order to gain an inclusive approach to the target field and to analyze the recent trends and findings of previous researchers.

1.9.1: Literature definition

I- Monitor [5]. gerund or present participle: monitoring, observe and check the progress or quality of (something) over a period of time; keep under systematic review.
Exp : "equipment was installed to monitor air quality".
Similar:

- observe
- watch
- keep an eye on
- keep track of
- keep under observation
- keep under surveillance
- keep a check on
- scan
- record
- listen to and report on, *exp* :"listening devices were used to monitor conversations"

II- Monitoring [6].

- Verb Present participle of monitor.
- Noun (usually uncountable, plural monitoring)

The act of listening, carrying out surveillance on, and/or recording the emissions of one's own or allied forces for the purpose of maintaining and improving procedural standards and security, or for reference, as applicable.
The act of listening, carrying out surveillance on, and/or recording of enemy emissions for intelligence purposes. The act of detecting the presence of signals, such as electromagnetic radiation, sound, or visual signals, and the measurement thereof with appropriate measuring instruments. The act of detecting the presence of radiation and the measurement thereof with radiation measuring instruments .

About monitoring

What to monitor?

Any measurable or capture phenomenon within the target environment. Approved fields:

1. Climate: temperature, humidity, wind speed, light intensity ... etc.
2. Manufacturing: carbon dioxide, concentration sensor, temperature, scene ... etc.
3. Health care: blood pressure, blood sugar, heart rate ... etc.
4. Spacial exploration : radio frequency, radiation level, ...etc[1].

Why monitoring?

There are several reasons for the activity which usually called monitoring to gather information about events in an area. We will locate:

1. Providing an interactive system to track activities 24 hours a day, from slightly different angles, that allows a more realistic representation of the target to be drawn.
2. Create a history (analyze, know, gain insights) for potential use.
3. Learn more about the connection between the different features and their relationships.
4. Make one or more features to build classification.
5. In future, method of predicting events based on a pattern.
6. Extract more knowledge and insights.

In general, the use of a monitoring system plays an significant role in disease prediction.[19], improving production, reducing cost [8], and providing an early warning system [63].

[1]Notice that list is not limited only by these environments.

Where to find monitoring?

Theoretically, by its nature, we could include Monitoring in any corner and any environment, but in real world until the time of the writing, it's still limited to some scopes that had evaluated as very relevant and specific such as: health, industry, traffic and weather. Based on functional characteristics and prior knowledge of system parts and factors that affect the environment or target element, we distribute sensors over points that enable measurements to be taken and avoid vulnerability also noise to minimize bad data.

The monitoring role will be the foundation and fundamental service for the entire system by spreading the idea of smart cities and orienting the world's to this option of urban management and administration.

Timing of monitoring

Capture of measure for the appropriate phenomena is made at a regular specific time interval $\Delta(t) = t_1 - t_0$ ([2]), usually coherent and relevant with many constraint within the environment. The nearest the measurement interval it was , the tiny time interval (smaller than the second). the observation process is to be described as real-time monitoring.

This monitoring will be active in real time , as long as it is appropriate for its function. Components capable of obtaining and handling the entire data ensemble that arrived from various sensors should be provided.

Monitor - process

After data is captured from different distributed sensors within the environment, it should be transmitted and grouped into a central unit to detect and eliminate outliers and deformed data. This will then be processed to identify (cluster) or forecast the information, the preceding step is in fact a comparison between the current state and the predefined state of the system. The final topic is to take action or provide the rational decision-making assist.

[2] t_0 : time of precedent measurement. t_1 : time of Subsequent measurement. $\Delta(t)$: time interval

1.9.2: Monitoring based IoT

New technologies, specifically Internet of Things (IoT)-based sensors, have been used and integrated with monitoring systems. A number of research studies have been conducted in the manufacturing, health and transport sectors and have shown significant benefits from the use of IoT-based sensors for monitoring such as improvement of working conditions, prevention of errors in design, fault diagnosis[53], quality prediction[31] , and helping managers with better decision making[18].

Advantages

The use of IoT in monitoring surveillance has many benefits over conventional one. Some of these benefits are: low cost, negligible visual effects, low power consumption , high versatility, no need for infrastructure, ease of deployment of sensor nodes and the ability to track and manage the scope environment on a continuous basis.

As a result of all these benefits, IoT becomes the fundamental foundation of several applications and services designed for monitoring, in the section bellow we depicted some of them.

1.9.3: domain application

IoT applications vary across myriads of fields and contexts.Figure 1.10 shows application domains of monitoring[51].:

- smart city,
- smart homes,
- smart warehouse,
- smart agriculture,
- and smart healthcare.

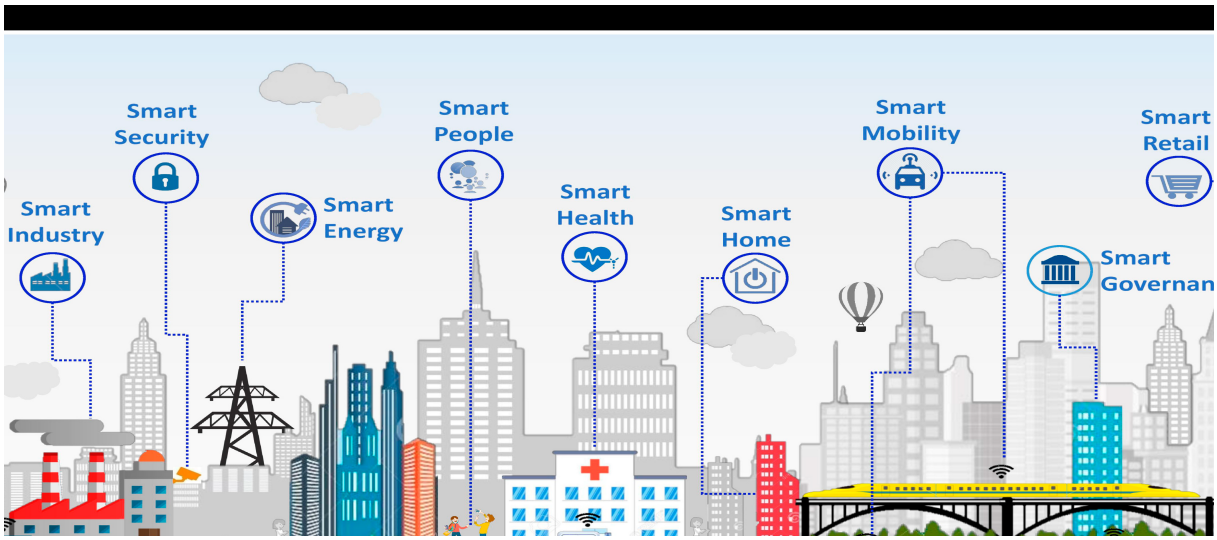

Figure 1.10: Overview of a generic smart city environment

Challenges

Firstly : Monitoring service efficiency is highly dependent on data collection, in addition to data manipulation and decision generation. Sensor inflation in order to obtain a clearer picture of the environment increases The amount of data collected across the network that takes us into Big Data situations. The effectiveness of monitoring services is, in fact, empowered by the real-time processing of data. Conventional data processing mechanisms are not capable of responding to requests that are time sensitive, in addition to accelerating data growth.

Secondly : Conventional data processing mechanisms use statistical methods to analyze large data sets in order to extract useful hidden data. Predictive analysis is also a commonly used conventional data analysis mechanism based on statistical methods. However, due to statistical insignificance in large data sets, challenges in efficient computing, and distinctive Big Data characteristics, i.e. heterogeneity, noise accumulation, false positive correlation, and incidental endogeneity, the direct application of predictive analytics has been questioned by domain experts.

According to[51], Data collected from a million smart meters within a year is calculated and presented in Table 1, assuming that a record size is 5 KB . As shown in Table 1.2, the set of smart meters collect 2920 terabytes (TB) for a period of one year with a frequency of record per every 15 min. Considering the data growth shown in the example, we can claim the necessity for real-time data processing techniques in smart cities. Smart environments of the modern era tend to rely on more sophisticated data management techniques, since conventional data processing techniques are not capable of catering for the exponential data growth.

Frequency	**1/day**	**1/hour**	**1/30 min**	**1/15 min**
Records Collected	365 m	8.75 b	17.52 b	35.04 b
Terabytes Collected	1.82 TB	730 TB	1460 TB	2920 TB

Table 1.2: Smart Meter Data Collection for One Year.

1.9.4: Main Architectures

With architecture we mean how and where the monitoring system elements are positioned and constructed physically and logically in order to achieve their objectives. This technology is older but it has its big leap with the spread of internet and the emergence of cloud computing in public spheres, and it continues to evolve and benefit from the hardware and internet enhancement. In height level we can place three types of architectures classified by method of distribution of processing as follows:

Centralized

This is the first that introduced the use of IoT along with the enormous ability of cloud computing storage and processing to provide long support for the entire network.Data sources are placed near the sensors in this fashion just to collect them in a coherent format and then send the result through internet link to the cloud. Implemented on the cloud core framework, it carries out all the tasks needed after the result is sent back to its destination again via the internet interconnections. the figure 1.11 show the structure of this architecture in general view.

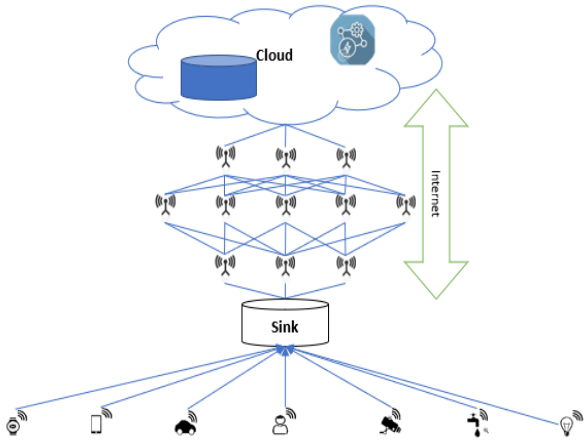

Figure 1.11: Centralized Architecture (Original)

Decentralized

Contrary to the precedent, this architecture split some of the key components between cloud and fog levels, on this latter we consider the primary computing tasks and decision making sections with partial data storage, the heavy part of the system residing on the cloud, the main storage data and the collective tasks. In this situation, the responses to sensor messages are rendered at the level of fog that decreases internet traffic and its effects. Figure 1.12 depict how it is composed.

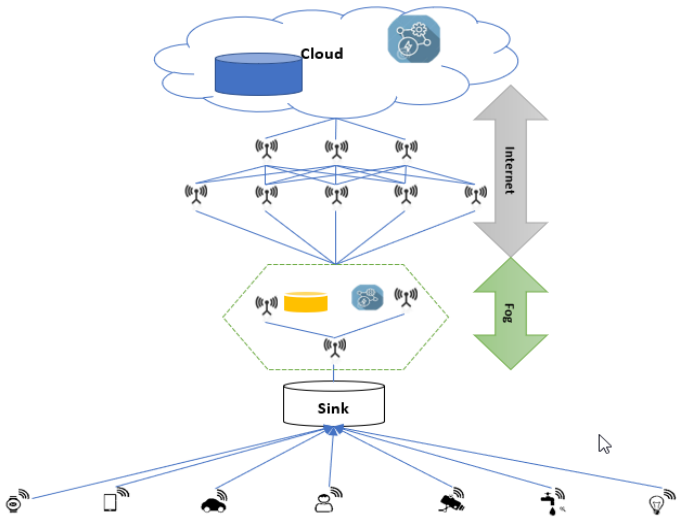

Figure 1.12: Decentralized architecture (Original)

Embedded

In this mode sensors are usually directly connected to actors and have the capacity to process data and apply the model as well as produce decisions or predictions locally, independently, this mode is often used in narrow scope and for special purposes such as gate command, light control, face recognition ... etc.

1.10: IoT vs Cyber-Physicalsystems(CPS)

Cyber-Physicalsystems(CPS) :

Is the integration of physical components, sensors, actuators, communication networks, and control centers, in which sensors are deployed to measure and monitor the status of physical components, actuators are deployed to ensure the desirable operations on physical components, and communication networks are used to de-liver measured data and feedback comments among sensors,actuators, and control centers. The latter are used to analyze measured data and sent feedback commands to actuators, ensuring the system operate in desired states[32]

The essence of CPS is the 'system' and the main objective of CPS is to measure the state information of physical devices and ensure the secure, efficient, and intelligent operation on physical devices. In CPS, the sensor/actuator layer, communication layer, and application (control) layer are present. The sensor/actuator layer is used to collect real-time data and execute commands, communication layer is used to deliver data at upper layer and commands to lower layer, and application (control) layer is used to analyze data and make decisions. Figure1.13 illustrates the three layers in CPS. From this figure, we can see that CPS is a vertical architecture.

In contrast, IoT is a networking infrastructure to connect a massive number of devices and to monitor ,control devices by using modern technologies in cyberspace. Thus, the key of IoT is 'inter-connection'. The main objective of IoT is to interconnect various networks so that the data collection,resource sharing, analysis, and management can be carried out across heterogeneous networks. By doing so, reliable, efficient,and secure services can be provided. Thus, IoT is a horizontal architecture, which should integrate communication layers of all CPS applications to achieve inter-connection, as shown in Figure1.13.

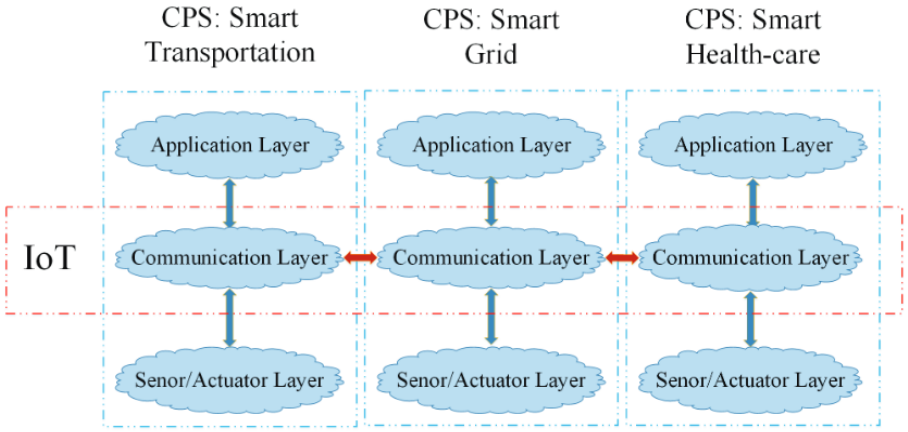

Figure 1.13: The integration between IoT and CPS [32]

For instance, in a smart city,networks of smart weather forecasting, smart transportation,and smart grid should be interconnected and interact with each other. Data from smart transportation and smart weather forecasting should be processed and extracted to be used by the smart grid to determine the states and brightness of street-lamps to ensure efficient use of energy resources, as well as traffic safety at night[32].

To summarize, the basic difference between CPS and IoT is that, CPS is considered as a 'system', while IoT is considered as 'internet'.The common requirements for both CPS and IoT are real-time, reliable, and secure data transmission. The distinct requirements for CPS and IoT are listed as follows:for CPS, effective, reliable, accurate, real-time control is primary goal, while for IoT, resource sharing and management,data sharing and management, interface among different nets,massive-scale data and big data collection and storage, data mining, data aggregation and information extraction, and high quality of network Quality of Service (QoS) are important services.

1.11: Data Challenge

IoT has emerged as a new trend in the last few years, where mobile devices, transportation facilities, public facilities, and home appliances can all be used as data acquisition equipment in IoT[34].By the way,various types of data, which are collected from many different devices, such as RFID readers, monitors, thermometers, etc.These data are different in data structures, volume, accessing methods, and some other aspects, as such; they can hardly be stored and accessed efficiently by a classic methods such as flat files or relational databases engine or in. Besides, the data volume may increase quite rapidly, so that the solution here is to found on big data technics and methods, which be able to process the data with a high throughput.

The need to adopt big data in IoT applications is compelling. These two technologies have already been recognized in the fields of IT and business. Although, the development of big data is already lagging, these technologies are inter-dependent and should be jointly developed. In general, the deployment of IoT increases the amount of data in quantity and category; hence, offering the opportunity for the application and development of big data analytics. Moreover, the application of big data technologies in IoT accelerates the research advances and business models of IoT. The relationship between IoT and big data, can be divided into three steps to enable the management of IoT data. The first step comprises managing IoT data sources, where connected sensors devices use applications to interact with one another. For example, the interaction of devices such as CCTV cameras, smart traffic lights, and smart home devices, generates large amounts of data sources with different formats. This data can be stored in low cost commodity storage on the cloud. In the second step, the generated data are called —big data,which are based on their volume, velocity, and variety. These huge amounts of data are stored in big data files in shared distributed fault-tolerant databases The last step applies analytics tools such as MapReduce, Spark, Splunk, and Skytree that can analyze the stored big IoT data sets. The four levels of analytics start from training data, then move on to analytics tools, queries, and reports[34].

1.12: Conclusion

During this chapter, we gave a comprehensive overview of the Internet of things by focusing on its definition and its most important physical characteristics represented in the abundance and diversity of sensors and software represented in the multiplicity and distribution of the communication protocols it adopts.

We also discussed the most important applications, services and areas that rely on the Internet of things as a work ground. So that we shed light on the most important points that make us surround this technology. And we take into account its requirements during the design of our model, so that we respond to the challenges that arise from it, as well as improve planning for its advantages and differences with the rules of industrial control systems.

Chapter 2

Computation paradigm and data processing

2.1: Introduction

In this chapter , we will look at the various engineering configurations for approved computing of the Internet of Things and the processing of big data in general, including cloud computing, fog and edge computing. The latter will not be examined in detail and in detail, except for the purpose of determining its position in fog computing. We focus on a number of features and a comparison between cloud and the fog computing, through it we choose a specific structure for our solution.

.

2.2: Computation paradigm

The emergence of Internet of Things (IoT) has enabled the interconnection and intercommunication among massive ubiquitous things, which caused an unprecedented generation of huge and heterogeneous amount of data, known as data explosions.
In this section we discuss the different computation paradigm used to implement solution for IoT applications specially latency-sensitive or oriented monitoring.

2.2.1: Cloud Computing

Cloud computing refers to both the applications delivered as services over the Internet and the hardware and systems software in the data centers that provide those services[12]. By storing and processing data using cloud technology, we have liberated ourselves from the relentless trouble of accessing data in a limited manner. We can now access additional features on our phones, computers, laptops, and IoT devices without needing to expand its computing power or investing in its memory storage capacity- all credit goes to the cloud computing. Most enterprises are familiar with cloud computing since it's now a defector standard in many industries.

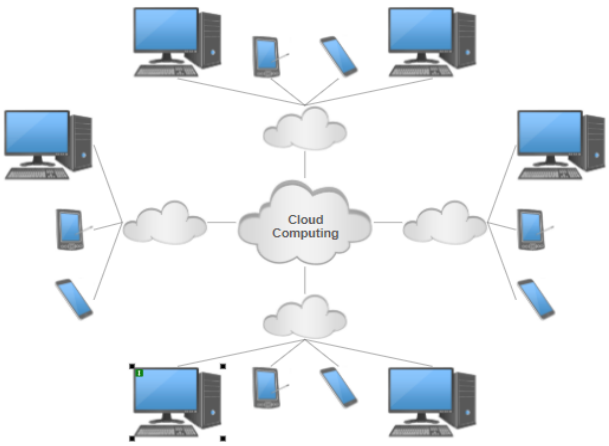

Figure 2.1: Distance between Cloud and Device connected

2.2.2: Edge computing

Also known as just "edge"—brings processing close to the data source, and it does not need to be transmitted at remote cloud or a superior systems for processing. By exclude the distance and time elements it acquires to transmit data until centralized sources, we can ameliorate the speed and performance of system response.

The term Edge and Fog computing seem similar but they aren't, and for a fact, they do share many common keys. Both of them move processing of data near as possible to the source of data generation. The principal goal of doing so is to limit the amount of data transmitted to the cloud. This leads in decreasing latency and thereby improving system response time, especially in remote mission-critical applications. By bringing the data processing closer to the source, there will also improvement of security as we don't need to send all the data across the public internet.

Both technologies leverage the power of computing capabilities within a local network to perform computation tasks that may have been carried out in the cloud easily. They can help to reduce the dependence on cloud-based platforms for data processing and storage, which often leads to latency issues, and are able to generate data-driven decisions faster.

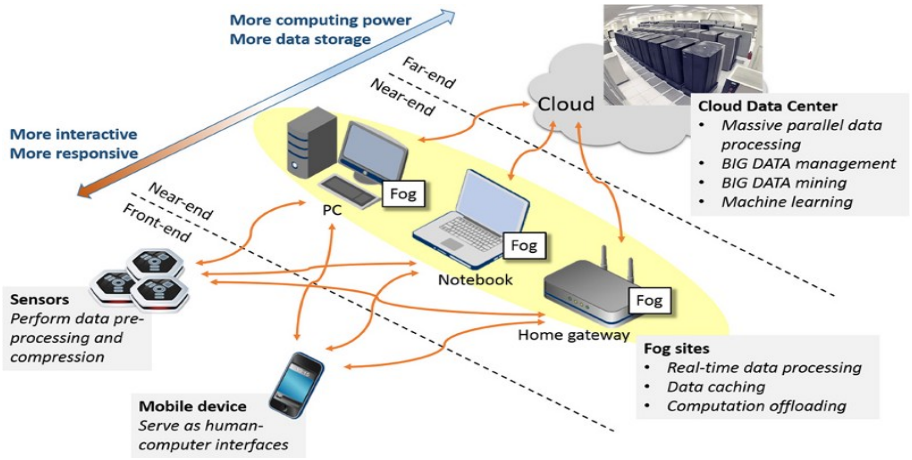

Figure 2.2: How Cloud, Fog and Edge works together[62]

The widely used computation paradigm has cloud datacentres as the only point for execution after the primary processing available at the devices.

With the explosion of data, devices and interactions, cloud architecture on its own can't handle the influx of information. While the cloud gives us access to compute, storage and even connectivity that we can access easily and cost-effectively, these centralized resources can create delays and performance issues for devices and data that are far from a centralized public cloud or data center source.

if the huge number of *IoT* devices continuously generating and sending data to the cloud for analysis would precede to scalability issues in the core network. Levels of congestion in the backbone network will increase manifold and may lead to aggravated packet loss and delay, spoiling the user experience. Furthermore, sending lots of data to the cloud for processing may lead to the cloud becoming a bottleneck, again leading to increase in response time.
Shifting computing power closer to the Edge of the network will help in reducing cost as well as improving security.

2.2.3: Fog computing

Is a term coined by professor Salvatore J. Stolfo , that has recently been picked up by Cisco. Fog computing is a paradigm that extends cloud computing and services to the edge of the network allowing applications to run in close proximity of users, be highly geo-distributed and support user mobility. Due to such characteristics, fog cuts down latency of service requests, and improves quality of service (QoS), resulting in superior user-experience[35].

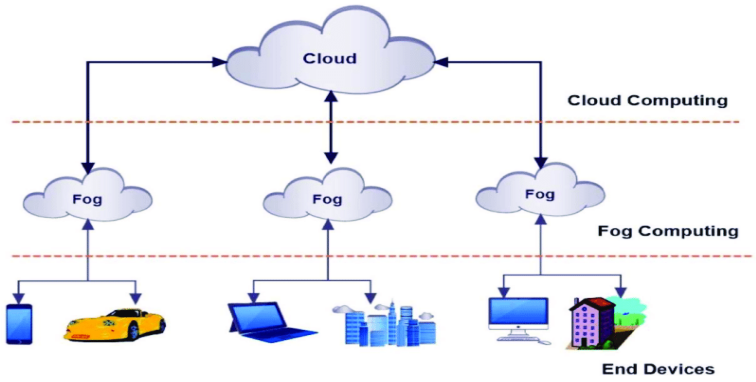

Figure 2.3: Fog-computing is an extension of the cloud but closer to end devices[13]

2.2.4: Architecture of Fog Computing

The hierarchical architecture of fog computing is composed of the following three layers:

- **Terminal layer**

  Which is the nearest layer to the end user and physical environment. It Come into contact with various IoT devices, for example, sensors,mobile phones, smart vehicles, cards readers, and so on. Specially, though the mobile phones and smart vehicles have the computing power,we only seeing them as the smart sensing devices here. These are widely geographically distributed in general. They are responsible for sensing and measuring the feature data of physical objects or events and transmitting these sensed data to upper layer for processing and storage.

- **Fog layer**

  It's the core layer and it is positioned just on the network 's edge .It is composed of multi fog nodes, whom generally are or including routers, gateways, switchers, access points, base stations, specific fog servers, etc. In a manner widely distributed, these fog nodes are situated between the end devices and the cloud, for example, Within coherent places and geographic zones or specific task like: cafes, shopping centers, bus terminals,streets, parks, etc. They can be static e.g, at a fixed location, or mobile on a moving carrier. The end devices should be connect with fog nodes to obtain services. They have the capabilities to compute, transmit and temporarily store the received sensed data.

  The real-time analysis and latency-sensitive applications can be accomplished in fog layer. Moreover,at the same time fog nodes are also in connection with cloud services by IP core net-work, and responsible for interaction and cooperation with cloud to obtain more powerful computing and storage capabilities.

- **Cloud layer**

  This layer consists of multiple high-performance servers and storage devices, and provides various application services, suchas smart home, smart transportation, smart factory, etc. It has powerful computing and storage capabilities to support for extensive computation analysis and permanently storage of an enormous amount of data. How-ever, different from traditional cloud computing architecture, not all computing and storage tasks go through the cloud. According to the demand load, the cloud core modules are efficiently managed and scheduled by some control strategies to improve utilization of the cloud resources[46].

Figure 2.4 show the presentation of these layers in graphical manner.

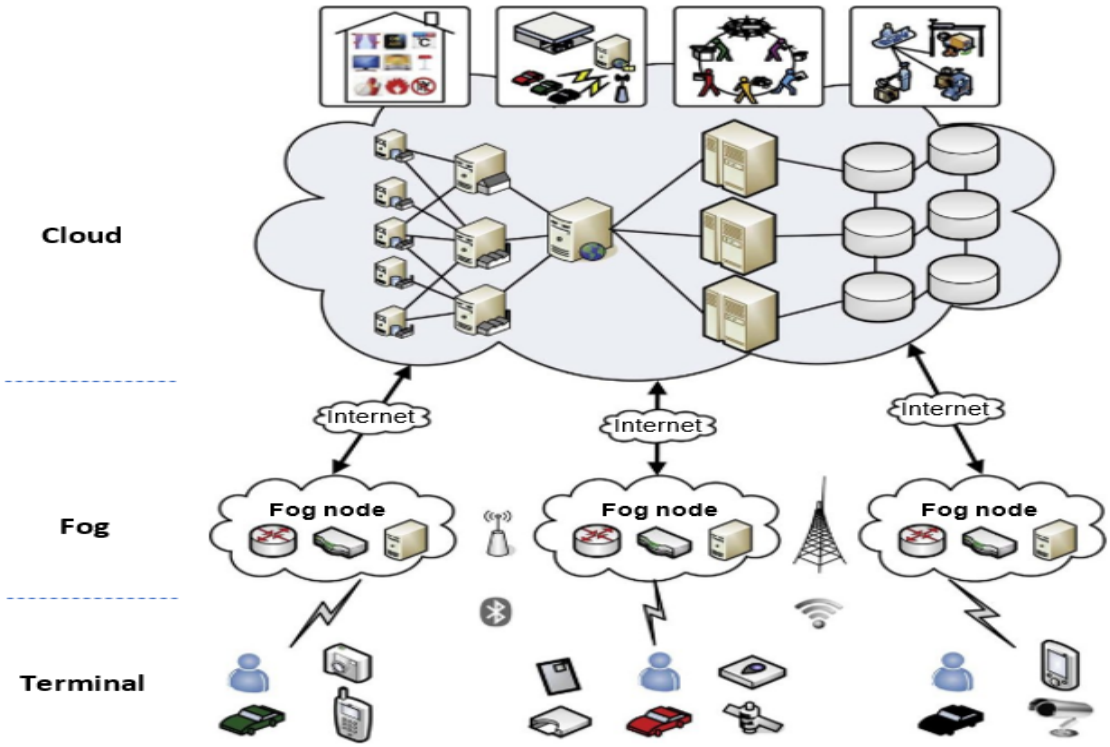

Figure 2.4: The hierarchical architecture of fog computing

As it is shown in the figure 2.4 In this architecture, each element (end device or smart object) is connected with one of the fog nodes at least by one or more of the access technologies (mainly including Wireless Local Area Network (WLAN), ZigBee, Bluetooth, 3G, 4G, etc.) or by wired connection. Also fog nodes can communicate and interconnected by wired or wireless communication technologies. And each of fog nodes is linked into the cloud by IP core network.

This architecture can provide the technical support for the IoT, cyberphysical system (CPS) and Mobile Internet to provide efficient data processing and storing services. Especially for CPS, which combines the capabilities of computing, communications and storage in order to monitor or control the entities and objects that exist in the physical world[50], fog computing can improve the efficiency and quality of service(QoS) in the case of data explosion currently.

Many studies[27] results conclude that this computing model enhances the performance and reduces the energy consumption in mobile environment.indeed mobile fog computing is the complementary of fog computing to provide low-latency mobile services.

2.2.5: Characteristics of fog computing

In fog computing service capabilities are proximity to end users. This is the nearly basic characteristic and the most significant advantage compared with other traditional computing models. Furthermore, there are some characteristics and advantages listed as follows:

- **Low latency and real time interactions**

  by theoretical modeling of the fog computing architecture Sarkar et al.[46] proved that the service latency for a system associated with fog computing was importantly lower than that with cloud computing .Meanwhile, Hu et al.[28] applied fog computing with model into face identification and resolution field, and indicated that the system response time significantly less than that with cloud computing by experiment.

- **Save bandwidth**

  by its Principe to carry important part pf processing and storage from cloud for example, data preprocessing, redundancy removing, data cleaning and filtering, valuable information extraction, are performed locally. fog computimg save a big part of network bandwidth was used to transmit these data and results. Only part of useful data is transmitted to and from the cloud, and most of the data don't need to be transmitted over the Internet.

- **Support for mobility**

  There are diverse mobile devices (e.g., smart phones, vehicles, and smart watch) so that the spatial mobility at the terminal layer is common, while there are also some end devices remained fixed, such as traffic cameras. likewise, nodes in fog layer can also be a mobile or static computing resource platform. It can be deployed in airport and coffee shop, or on the mobile vehicles and trains. In fog computing End device itself or intermediate devices process the massive data generated by the Internet of things, and truly realizing mobile data analysis. So it can provide services for more extensive nodes.

  Fog computing architecture supports location-based mobility demands and enables administrators to control where users and mobile devices are coming in and how they access the information. This improves the performance of system and quality of service[50].

- **Geographical distribution and decentralized data analytic**

  Fog computing consists of large number widely distributed nodes, which git it the ability to track and derive the locations of end devices as a part of supporting mobility. rather of processing and storing information in centralized data center far away from end-user, this decentralized architecture ensures the proximity of data analytic to the user. This characteristic can result in faster analysis of big data, better location-based services, and more powerful capabilities of real-time decision making.

- **Heterogeneity**

  In general, fog nodes come in different form factors and are deployed in a wide variety of environment in the form of physical node or virtual node[30]. They usually range from high-performance servers, edge routers, gateways, access points, base stations, etc. These hardware platforms have varying levels of processing and storage capacities, explore various kinds of operating system (OS), and load different software applications. Fog computing is a highly virtualized platform, so fog nodes are heterogeneous and extremely dynamic at different levels of architecture hierarchy to address the requirements of widely distributed applications that need low latency.

- **Interoperability**

  by consequence of to their heterogeneous nature, many components of fog computing (nodes and end devices) come from different suppliers and are usually deployed in the divers environments. Fog computing must be able to inter-operate and cooperate with different providers to collaborate with wide range of services and seamlessly support certain of them. For example, streaming service supported by fog computing requires the cooperation of different providers, in which services are federated across domains[17].

- **Data security and privacy protection**

  Fog computing hosts services closed to end-users. So it has particular advantages in data security and privacy protection. Firstly, it can protect data by using cryptography and isolation. Fog nodes provide access control policy, encryption schemes, integrity check and isolation measures to protect the security of sensitive privacy data. Secondly, it can avoid the risks caused by avoiding the transmission of critical data over the internet to cloud and keep them at a local level.

- **Low energy consumption**

  fog nodes are dispersed geographically. So it will not generate a lot of heat due to concentration, and need not additional cooling system. In addition, short range communication mode and some optimal energy Management Policies of mobile nodes evidently reduce communication energy consumption. This will lead to reducing power use, saving energy and decreasing the cost. Fog provides a greener computing paradigm.

2.3: Data Processing

This section is dedicated to introducing and covering the part in which the data received from the sensors are processed and highlights the main characteristics that would be available there.

One of the key questions to be answered when planning a data architecture is the issue of batch versus stream processing: do we process data as it arrives, in real time or in near-real time, or do we wait for the data to be accessed? The second question is, what framework is best suited to our proposed solution? This chapter is intended to provide a high-level overview of the considerations to be taken into account when making that decision..

2.3.1: Data Processing Types

According to [21] the kind of analysis big data tools are classified into three types (1) batch analysis, (2) stream processing, (3) interactive analysis. In batch analysis, data are stored after that it is analyzed. In stream processing, as early as possible which is analysis and it give results. Interactive analysis allow users to perform their own analysis of data.

Batch Processing

In batch processing, we wait for a certain amount of raw data to be "piled up" before running a processing task. Usually, this means that the data is between an hour and several days old before it is made available for analysis. Batch processing tasks will usually be carried out on a set schedule (e.g. every 24 hours, or in some cases once the amount of data has reached a certain threshold).

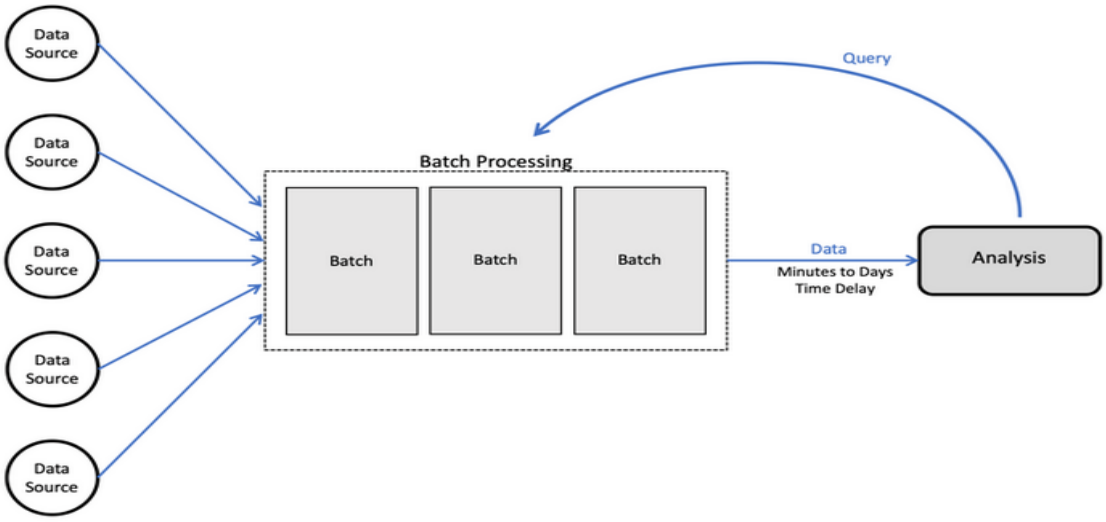

Figure 2.5: Batch Processing [1]

Batch processing, by definition, requires latencies between the time data arriving in the storage layer and the time available in the analytics or reporting tools. We should choose to recognize these delays because our processing requires a collection of information to be collected with constraints, so we have to work with batch processing frameworks. In our scenario, we will update our ANN model and training it with new set of data in case we loose accuracy, otherwise we will want to feed the model with the maximum volume of data stored and as older as possible.

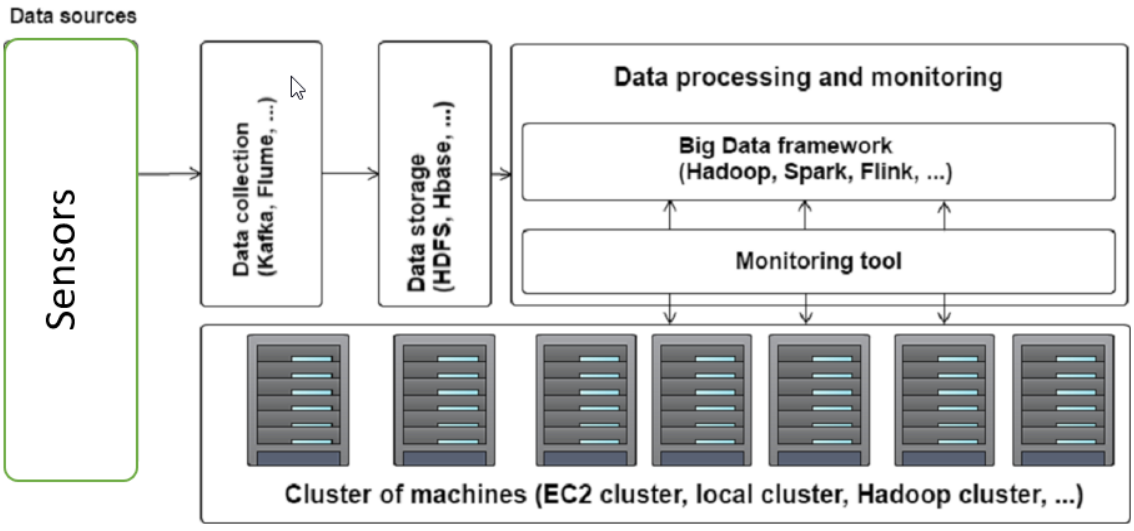

Figure 2.6: Batch Mode scenario[29]

Batch processing tools and frameworks [1]

- Open-source Hadoop frameworks such as Spark and MapReduce are a common option for processing large data.
- Relational databases such as Amazon Redshift and Google BigQuery.

Stream Processing

In stream processing, we process data as soon as it arrives in the layer of storage – which would often also be very close to the time it was generated (though this would not always be the case). Typically this will be in sub-second time frames, such that the execution takes place in real time for the end user. Typically, these operations would not be stateful, or would only be able to store a 'small' state, so generally a relatively simple transformation or calculation would be involved [**?**].

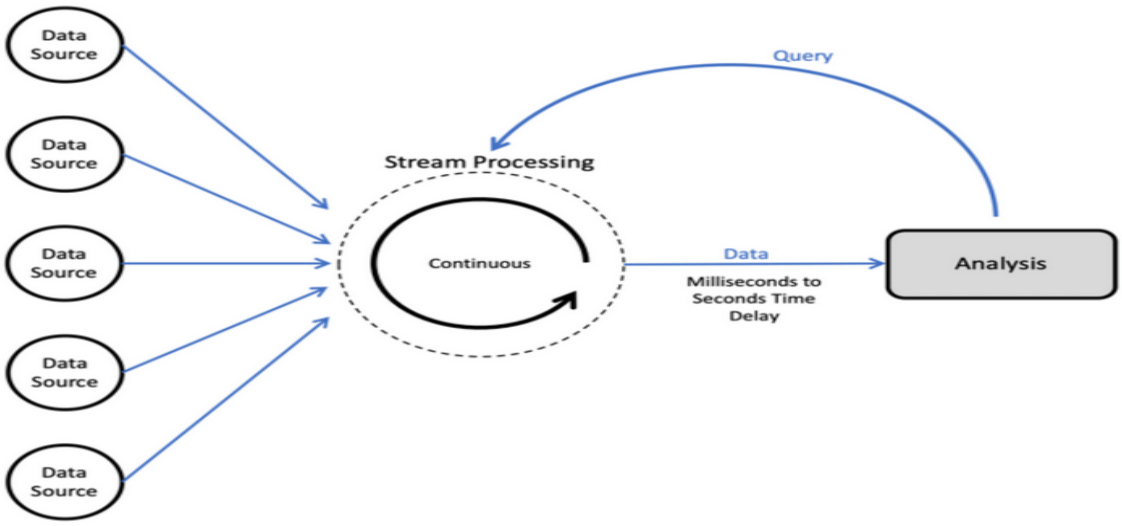

Figure 2.7: Stream Processing Schema [1]

The rule is: If we need to analyze or serve data as close as possible to when we get hold of it, we would use stream processing.

Indications that stream processing is the right approach:

- The data is generated in a continuous flow and reaches high speed.
- Sub-second latency is crucial.

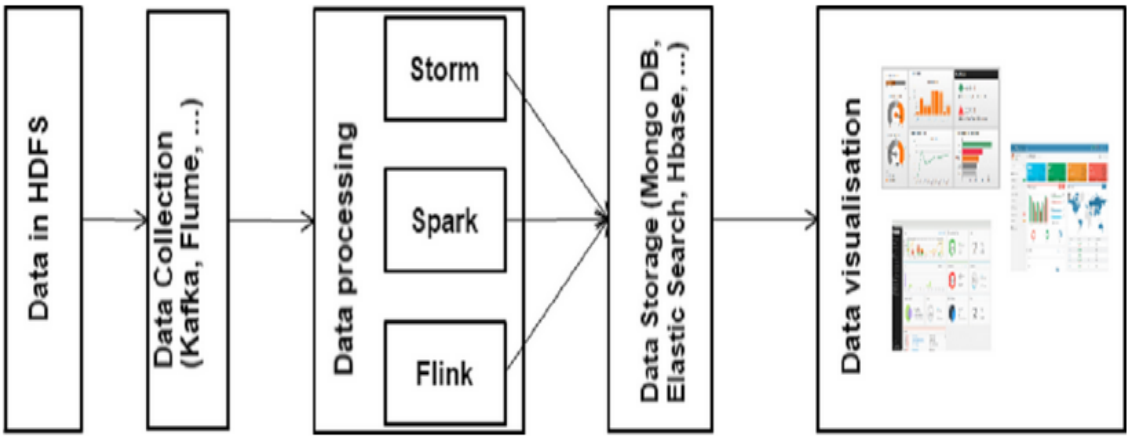

Figure 2.8: Stream Mode scenario[29]

2.3.2: Critical Stream Processing Aspects:

There are some important characteristics and terms associated with Stream processing that we should be aware of in order to recognize the strengths and weaknesses of every Streaming framework[43]:

1. **Delivery Guarantees (semantics) :**there are 3 types of message delivery[58].

   (a) a maximum of one-time (AT will most once recording), when the messages are delivered only once and may be lost.

   (b) at least one-time (AT Least once recording), when the messages can be delivered again - they are not lost, but may overlap.

   (c) exactly once (exactly once), when each message is guaranteed to be delivered only once, while the messages are not lost or duplicated.

   Obviously Exact-once is desirable but difficult to achieve in distributed systems and comes with performance drawbacks.

2. **Fault Tolerance :**In the event of failures such as node failures, network failures, etc., the system should be able to recover and start processing again from the point where it left. This is accomplished by checking the streaming status of some persistent storage from periodically.

3. **State Management :**In the case of stateful processing requirements where there is a need to preserve some state (e.g. counts of each distinct word found in records), the system should be able to provide a mechanism for preserving and updating state information.

4. **Performance :**This involves latency (how fast a record can be analyzed), throughput (processed / second records) and scalability. Latency should be as minimal as possible, while throughput should be as high as possible. It's challenging to do them at the same time.

5. **Advanced Features :**Event Time Processing, Watermarking, Windowing these really are characteristics that are expected if the requirements for stream processing are complex. For example, processing records based on the time when they were created at the source (event time processing).

6. **Data sources and receivers :**Some frameworks concentrate on interacting with various external file systems, storage and databases (Kafka, Cassandra, HDFS, OpenStack Swift, Cassandra, Amazon Kinesis, HBase, Google Cloud Platform Amazon S3, Kudu, MapR-FS), but others are restricted to working with data stored in specific cluster types.

7. **Maturity :**Particularly from the point of view of adoption, it is a positive thing if the system is already proved and the war is being evaluated on a scale by big corporations. More likely to get strong feedback from the group and aid stackoverflow.

2.3.3: Types of Stream Processing:

Now that we are aware of the terms we have just discussed, it is now easy to understand that there are two approaches to the implementation of the Streaming Framework:

1. **Native Streaming :**It means that every arriving record is processed as soon as it appears, without waiting for others. There are several continuous running processes (that we call operators / tasks / bolts depending on the framework) which run continuously and every record passes through these processes to be processed.

2. **Micro-batching :**Often known as Fast Batch. It often means that incoming records are piled with each other every few seconds and then processed in a single mini batch with a latency of a few seconds.

Both initiatives have benefits and drawbacks. Native Streaming feels more natural since every record is processed as soon as it arrives, allowing the system to achieve the lowest possible latency. But this also means that it is difficult to achieve fault tolerance without compromising throughput as for each record, we need to track and checkpoint once processed. State management is also simple, as there are long running processes that can easily sustain the necessary state.

Micro-batching , on the other hand, is quite opposite. Fault tolerance comes for free as it is essentially a batch and throughput is also high as processing and checkpointing will be done in one shot for group of records. But it will be at some cost of latency and it will not feel like a natural streaming. Also efficient state management will be a challenge to maintain[43].

2.3.4: Kafka Streams

Kafka is message broker which can be connected to any real-time framework available on the market[47].

Kafka Streams, which is a light weight library unlike other streaming systems. It is useful for streaming Kafka data, transforming it and then returning it to Kafka. We may recognize it as a Java Executor Server Thread pool-like library but with Kafka built-in help. It can be well integrated with any program, and it will operate from the box.

Because of its lightweight nature, it can be used for architecture of microservices kind. There's no efficiency comparison with Flink or Spark and Storm, but there's also no need for a separate cluster to operate, it's very convenient and easy to deploy and work. Uses Kafka Consumer Group internally and works on philosophy of the Kafka log.

Kafka Streams has one major advantage in that its processing is Exactly Once to End. It is possible because both the source and destination are Kafka and released around June 2017 from Kafka 0.11 version, Exactly once is supported. We just need to enable a flag to enable this feature so it will work out of the box [43].

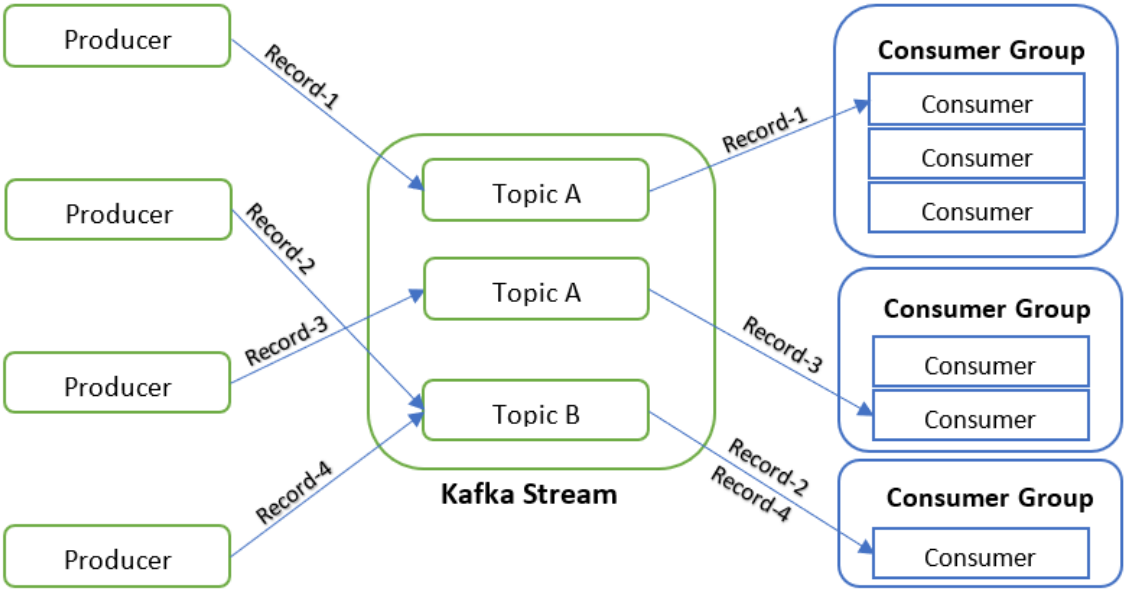

Figure 2.9: Kafka Stream Schema -Original-

2.3.5: Apache Spark

Spark system overview

Is an open source distributed processing framework that was created at the UC Berkeley AMPLab[40].Spark is designed to operate on an in-memory system in comparison to Hadoop which works on a disk. Spark is a recognized platform for analytics that ensures fast, easy-to-use, and versatile computing. This performs complex processing of large data sets. Spark runs programs up to 100x faster than Hive and Apache Hadoop by comparison, as it uses MapReduce in-memory. Spark is also based on the base Apache Hive code. Spark changes Hive 's physical execution engine to boost system performance. Additionally, Spark provides APIs to support fast development of applications in various languages including Java , Python, and Scala. Spark is able to work with all Hadoop-supported file systems.

One more option is Spark's data model is based on the Resilient Distributed Dataset (RDD) abstraction. RDDs constitutes a read-only collection of objects stored in system memory across multiple machines. Such objects are available without requiring a disk access. Furthermore, they can be rebuilt if a partition is lost.[40]

In Spark, there are two types of operations on RDDs: (1) transformations and(2) actions. Transformations consist in the creation of new RDDs from existing ones using functions like map, filter,union and join. Actions consist of final result of RDD computations[29].

Spark architecture

In Fig.2.10, We provide a description of the Spark architecture. A Spark cluster is based on three key components of a master / slave architecture:

1. **Driver Program:** It represents a slave node in a Spark cluster. It only maintains an object called Spark Context that manages and oversees running applications.

2. **Cluster Manager:**which is charged in primary for organizing the application workflow assigned to workers by the Driver Program. Secondly, it controls or supervises all resources in the cluster and forwards them to the Driver Program.

3. **Worker Nodes:**Each node in this component represents a single operation container during the execution of the Spark program.

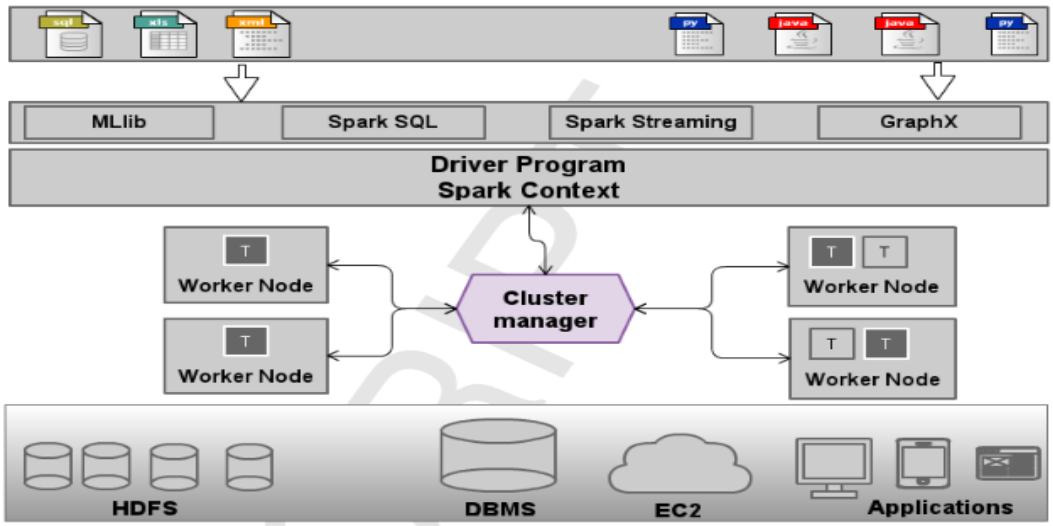

Figure 2.10: Spark system overview [29]

One of the many advantages of Spark in the number of APIs available for different applications:

1. **Spark Core[29]:** Is the underlying general execution engine for the Spark platform. All other features and extensions are built on top of it. Spark Core provides in-memory computing capabilities and a generalized execution model to support a wide variety of applications, as well as Java, Scala, and Python APIs for ease of development.

2. **Spark streaming[40]:** Is another component that provides automatic parallelization, as well as scalable and fault-tolerant streaming processing. It enables users to stream tasks by writing batch like processes in Java and Scala. It is possible to integrate batch jobs and interactive queries. It runs each streaming computation as a series of short batch jobs on in-memory data stored in RDDs.

3. **SparkSQL [29]:**offers a range of features to structure data retrieved from several sources. It allows subsequently to manipulate them using the SQL language.
   In other words, programmers can easily mix SQL commands to query external data sets with complex analytics because it unifies the two abstractions: relational tables and RDD. It facilitates fast parallel processing of data queries over large distributed data sets for this purpose. It uses a query languages called HiveQL.

4. **MLlib [40]:**Is a distributed machine learning framework built on top of Spark. For performance, MLlib provides various optimized machine learning algorithms such us classification, regression, clustering, and collaborative filtering. It offers algorithms for topic modeling and frequent pattern mining.

5. **GraphX[40] :**Constitutes a library for manipulating graphs and executing graph-parallel computations. Like Spark Streaming and Spark SQL, GraphX extends the features of Spark RDD API. Thus users can create a directed graph with arbitrary properties attached to each vertex and edge. GraphX offers different operators that support graphs manipulation (e.g., subgraph and mapVertices). It provides also a library of graph algorithms (e.g., PageRank and triangle counting).

2.3.6: Apache Storm

Storm Overview

Storm[29] is an open source framework for processing large structured and unstructured data in real time. It is a fault tolerant framework that is suitable for real time data analysis, machine learning, sequential and iterative computation. Following a comparative study of storm and Hadoop,we find that the first is geared for real time applications while the second is effective for batch applications.

Storm's Topology

As shown in Fig.2.11, a storm program/topology is represented by a directed a cyclic graphs (DAG). The edges of the program DAG represent data transfer. The nodes of the DAG are divided into two types: spouts and bolts. The spouts (or entry points) of a storm program represent the data sources. The bolts represent the functions to be performed on the data. Note that storm distributes bolts across multiple nodes to process the data in parallel.In Fig.2.11, we show a storm cluster administrated by zookeeper, a service for coordinating processes of distributed applications.

Storm is funded on two types of active services called Nimbus (located master node) and supervisor (within each slave node). Nimbus administrate the slave nodes and charges tasks to them. If it determines a node failing in the cluster, it delegates the task to another node. Each supervisor it charged to control the processing of its tasks (affected by the nimbus). It has the privileged to stop and start the spots according to instructions of Nimbus.Each topology included in Storm cluster is divided into several tasks.

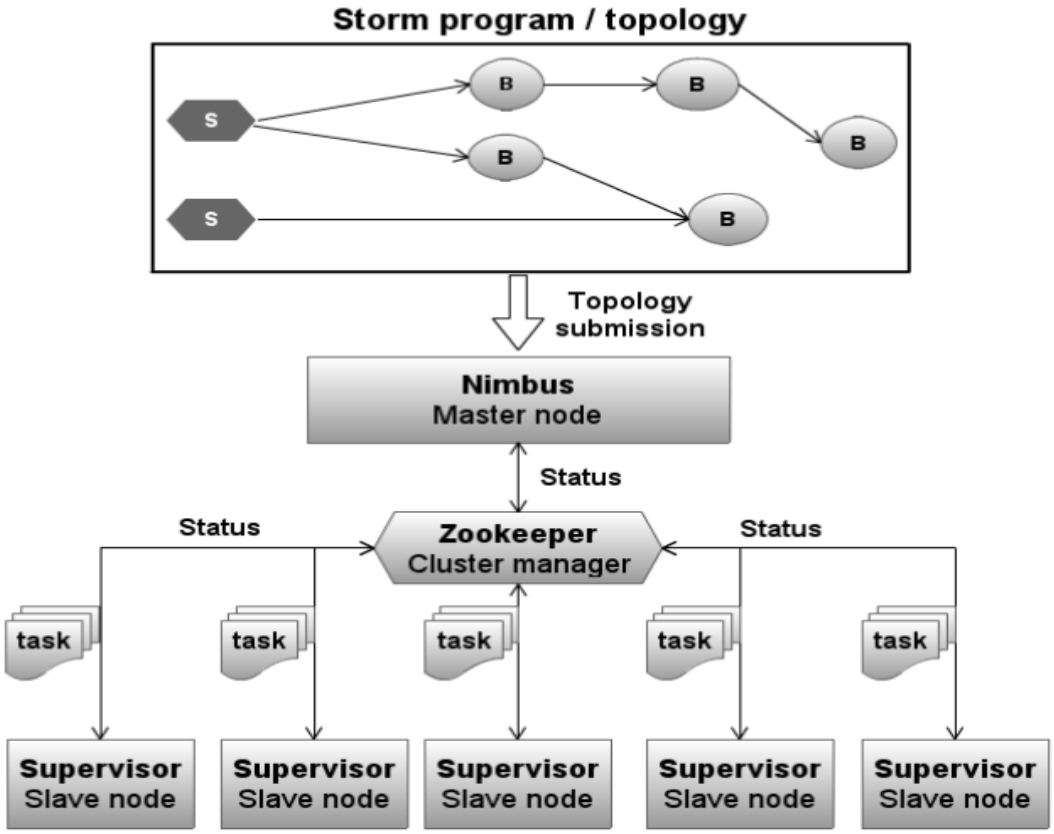

Figure 2.11: Topology of a Storm program and architecture [29]

2.3.7: Apache Samza

Apache Samza is built on the notion of a Publish / Subscribe task that listens to a stream of data, processes messages as they get there, and produces an output the output to another stream. A stream can be split into multiple partitions, and a copy of the task will be created for each partition.

Apache Samza depends on third party systems to manage:

- Data streaming among tasks (Apache Kafka, which is based on the Apache Zoo Keeper).
- Distribution of tasks between cluster nodes (Apache Hadoop YARN)

2.3.8: Apache Flink

Flink Overview

Flink has its origins in the Stratosphere project, a research project conducted by three Berlin-based Universities as well as other European Universities between 2010 and 2014. The project had already attracted a broader community base, in part through presentations at several public developer conferences including Berlin Buzzwords, NoSQL Matters in Cologne, and others. This strong community base is one reason the project was appropriate for incubation under the Apache Software Foundation.

The project completed incubation quickly, and in December 2014, Flink graduated to become a top-level project of the Apache Software Foundation. Flink is one of the 5 largest big data projects of the Apache Software Foundation, with a community of more than 200 developers across the globe and several production installations, some in Fortune Global 500 companies. At the time of this writing,34 Apache Flink meetups take place in cities around the world, with approximately 12,000 members and Flink speakers participating at big data conferences. In October 2015, the Flink project held its first annual conference in Berlin: Flink Forward.[23]

The key components of the Flink stack

Flink 's core computational fabric, labeled "Flink runtime" in Figure 2.12, is a distributed system that supports streaming data flow programs and executes them in one or even more computing devices in a fault-tolerant manner. This runtime could well run in a cluster, as a YARN (Yet Another Resource Negotiator) application, or within a single machine, which is very beneficial for debugging Flink apps.

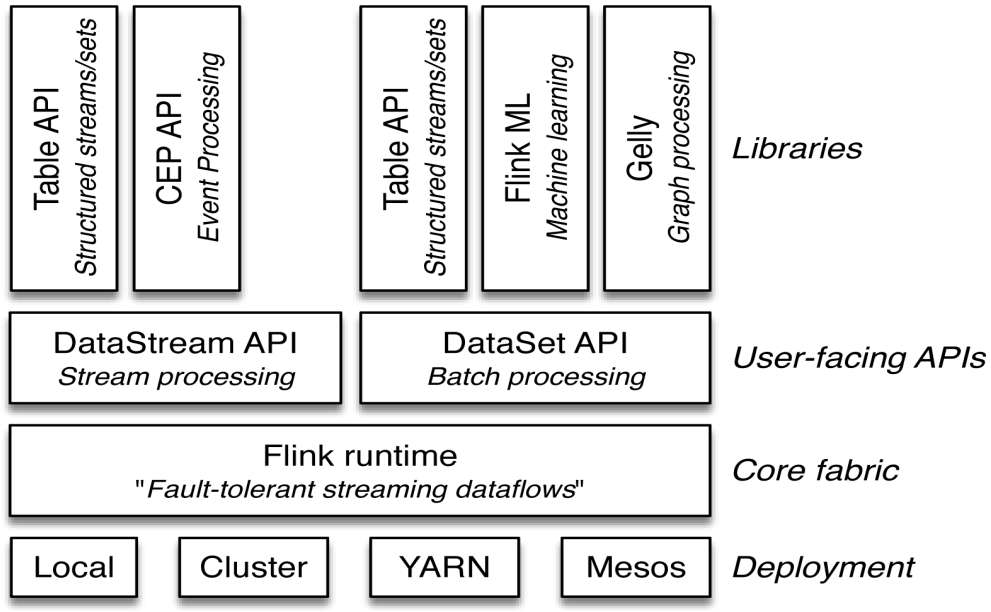

Figure 2.12: Key components of the Flink stack [23]

Notice: That both stream and batch processing APIs are included in the user-facing layer, making Flink a single data processing tool in almost any situation. Libraries include machine learning (FlinkML), complex event processing (CEP) and Gelly graph processing, and a streaming or batch mode table API.

Runtime-accepted programs are very powerful, but they are verbose and hard to program directly. For this reason, Flink provides developer-friendly APIs that layer at the top of the runtime and induces these streaming dataflow programs. The DataStream API for stream processing and the DataSet API for batch processing are also available.

The DataStream API is a fluent API that defines analytics for potentially infinite data streams. You can only use the API in Java or Scala. Users are working with DataStream, a data structure that represents distributed, possibly unlimited, streams.

Flink is distributed in the sense that it can run on dozens or hundreds of machines, distributing massive computation in small chunks, with each machine running one chunk. In the event of disruption as well as other errors or malicious reprocessing, as in the case of bug fixes or version updates, Flink code automatically restores the computation correctly. This capability has reduced the need for the programmer to worry about failures. Flink uses fault-tolerant streaming data streams internally, allowing developers to analyze never-ending streams of continuously generated data (stream processing).

2.4: Related Work

Researchers in both industry and academia have derived alternative approaches for optimizing the monitoring service compound with the benefits from internet of things(IoT) Accordingly, various experimental studies were conducted to identify the solutions to perceived challenges.

Liu et al[33]. proposed the first work to introduce deep learning into the edge computing environment. They proposed a deep-learning-based food recognition application by employing edge-computing-based service infrastructure. Their work shows that edge computing can improve the performance of deep learning applications by reducing response time and lowering energy consumption. However, this work considered mobile phones as edge nodes, which is not appropriate for IoT services since most IoT devices are equipped only with low-spec chips. Since we focus on general IoT devices without enough energy supplement and high-spec chips, the edge servers are deployed in IoT gateways, which have enough service capacity for executing deep learning algorithms.

In the study by Mohammad et *al* [53], a real-time monitoring system that utilizes IoT-based sensors, big data processing, and a hybrid prediction model is proposed. in which they developed an IoT-based sensor that collects data within ***the monitored production chain***. The characteristics of IoT-generated sensor data from the manufacturing process are: real-time, large amounts, and unstructured type. The proposing of big data is proposed and implemented with appropriate Apache software, after they implement the system in proposed schema by introducing the idea of localized the part of decision on the edge side of the network not on the cloud. Results show the efficiently of this architecture in reducing time latency.

Moreover,Tang *et al*[54]. introduced a hierarchical Fog Computing architecture for big data analysis in smart cities. In contrast to the Cloud, the Fog Computing parallelizes data processing at the edge of network, which satisfies the requirements of location awareness and low latency. The multi-layer Fog computing architecture is able to support quick response at neighborhood-wide, community-wide and city-wide levels, providing high computing performance and intelligence in future smart cities.

Mahdi et *al*[37] shows that IoT applications in general, and smart city applications in specific where context-awareness is a valuable asset can benefit immensely from unlabeled data to improve the performance and accuracy of their learning agents. Furthermore, the semi-supervised deep reinforcement learning a good solution for many IoT applications since it requires little supervision by giving a rewarding feedback as it learns the best policy to choose among alternative actions.

Alsheikh et al.[9] proposed a framework to combine deep learning algorithms and Apache Spark for IoT data analytics. The inference phase is executed on mobile devices, while Apache Spark is deployed in cloud servers for supporting data training. This two-layer design is very similar to edge computing, which shows that it is possible to offload processing tasks from the cloud.

In study [11], Ghada Alsuhly et al developed an integrated IoT system for monitoring and managing the museum ambience with the help of anti-theft capabilities. They adopt a layered approach to design, in comparison to current structures. But their layered architecture is more functional than component-based. However, this methodology steals on the basis of batch processing mode and does not meet the latency constraint needed for many situations and cases not covered by the analysis.

2.5: Conclusion

During this chapter, we cited as many strategies as possible and we link it back to our objectives in order to make comparative studies. this studies guide us to answer the posed questions about the most adopted computing paradigm and data processing platform. Also, it helps us benefit of advanced techniques and their strengths.

Part II

Contribution

Chapter 3

Proposition and Technical choices

3.1: Introduction

We devote this chapter to presenting the framework of the general model that presents its various parts with a general overview that includes the role of the part within the overall model. Then we transfer to perform theoretical comparisons between the various frameworks currently available in terms of the concepts of their function and their reaction to the various characteristics of the target function of the final model. Which results, at each point, in deciding the most suitable choice for a specific part and changing the general form by putting the options in place of the general description until the specification of all the pieces.

3.2: General Model

In this work we propose a model of an event analysis system (Monitoring tool) Based on IoT which exploit Deep learning from IA discipline. Figure 3.1 represent the general schema for this tool.

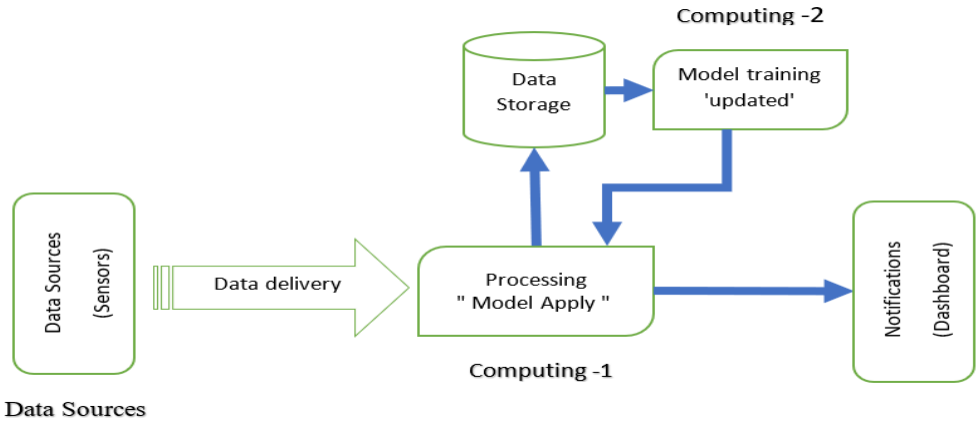

Figure 3.1: General monitoring tool architecture (Original)

The model would reply to bellow requests:

1. Guarantee the detection, selection and transmission of events generated by sensors in the proper manner.
2. Protect the data from losing in the most difficult circumstances.
3. Have the ability to manage data streaming, and volume growth.
4. Provide the best way to perform tasks, and achieve low latency by producing results in the shortest possible time.
5. Can easily scale in the future, and respond to extendable requests.
6. Offer full tolerance of fault to match case critics.
7. Include the most appropriate method of Artificial intelligent to analyze events and prove beneficial (Prediction, classification)

3.3: IoT or CPS

As stated before, the two concepts are almost similar in terms of functionality side and primary goals, but in detail and specific points we can find the differences which guides to choose one of them to be the correct implementation to get the most appropriate result.

3.3.1: Scope

Through scope, we mean the size of area to be covered by the chosen technology. It has two dimensions, vertical representing how many stages the device has components as shown in figure 1.13. The other dimension is the horizontal one that shows how the technology spreads across surfaces and locations.

Through literature we can infer that CPS is targeted towards being used in industrial environments and very limited areas. One of the functional choices is that it should have the minimum number of stages and also implemented with the security of restraint. On the other hand, the internet of things that were originally intended to include and manipulate any suitable and useful device (sensor / actuator) can be linked to the internet, its scope theoretically (vertically or horizontally) is not limited, and can combine any number of minor areas connected over internet.
Our project aimed at being applied across various scopes. In the low level it may cover a small area such as building or crossing an institution made up of several buildings, this reach may extend more and more to other areas and incorporate more high-stage services.

From all above, we chose the Internet of Things as our platform upon which to apply our solution.

3.3.2: Protocols

Because Cyber Physical System is oriented to be primarily in industry control for specific purposes,through a limited area, it usually uses a specific number and types of protocols, thus returning to high safety requirements for such systems.
In comparison to the Internet of Things, which uses the Internet as key mechanism for its communications, it uses a wide range of communication protocols and would be more compliant with a number of them, thereby allowing new devices to be more integrated.

Another big distinction between the two systems is that the IoT primarily prefers to use protocols with low power consumption as its sensors can be positioned separately and far from power sources. CPS, on the other hand , is based on devices and protocols most of which are implanted where power is permanent and constant.

3.3.3: Scalability

Scalability has two parts, firstly at the space level which covered by the system that can be done by adding more areas and buildings, secondly in the number and variety of devices that allow the collection of new types of information and increase the volumes.

Scalability in CPS is limited in the areas concerned and, for the most part, it concern the number of devices installed over the area and that is caused by the constraints that apply to this choice.

For IoT the scalability is one of the key features, it can be at many sides, at the scope, day after day the zones covered by sensors growth, in the same time the diversity of sensors and actuators is increasing to include more data types and shapes, the fact which make the hierarchy of hole system expanded to more stages offers new services and distribute tasks in order to get work balancing, fault-tolerance.

3.3.4: Flexibility

Due to the multiple constraints expected by CPS infrastructures, they can not be modified or reorganized in such an easy way as to fit new requirements. The sensitivity of the hole system to compatibility and accuracy made it difficult to frequently increase or decrease the space and service scope of the system.

On the other hand, IoT infrastructures are quite open and easy to modify, in fact, their dependence on the Internet makes their evolution in communication technologies and protocols at the same time no hard work to increase or decrease the scope.

3.3.5: Data Types

IoT can be found anywhere on the Internet , new devices (sensors, actuators) are integrated every day to collect and measure new types of data, from the very simple shape (temperature , humidity, ...) to the complex ones (image, voice , video ... etc).

Despite the fact that the CPS is a specific subset of IoT, we can find data types specified and used only by CPS and not included in data types identified by IoT, these are industrial data and formats used to process tasks.

3.3.6: Summary

In the table 3.1 below, we summarize the main points that should be considered in order to decide which framework to use in our solution.

	IoT	**CPS**
Scope	Not limited by geography	Limited to company zone
Protocols	Internet protocols	Specific to service's type
Scalability	Easy and very often	Hard and rare
Flexibilty	Very flexible	Limited
Data types	Any type even new ones	Limited Ensemble and Specific

Table 3.1: Comparative between Internet of Things and Cyber Physical System

From above, IoT was considered to be our data source for our model. The latter has been updated as shown in the figure below.

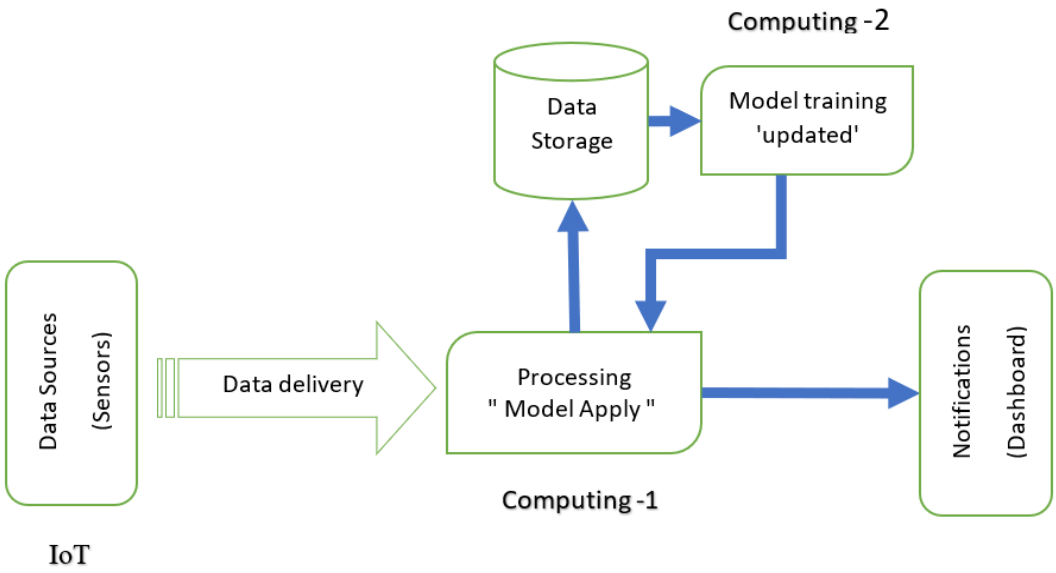

Figure 3.2: Data Source framework choice

3.4: Chosen Computing Paradigm

3.4.1: Differences between Cloud, Fog and Edge paradigm

1. **Location of Data Processing:**

   The primary difference between cloud , Fog , and Edge computing is the location where data processing occurs. In cloud , data is processed on a central cloud server, which is usually located far away from the source of information. It takes place on cloud services such as Amazon E2C instances. On the other hand,in Fog computing, the data is processed within an IoT gateway or Fog nodes that are located in the LAN network, also it shifts the Edge computing tasks to processors that are connected to the LAN hardware or the LAN directly so that they may be physically more distant from the actuators and the sensors. So, for Edge computing, the data is processed on the sensor or device itself without shifting to anywhere else.

2. **Processing Power and Storage Capabilities:**

   Cloud computing provides superior and advanced processing technological capabilities. It can store far more data than Fog computing that has the limited processing power. Similarly, the processing power and storage capabilities are even lesser in the case of Edge computing, since both of them are performed on the devices/IoT sensor itself.

3. **Internet dependencies :**

   It would also be worthwhile to mention here that cloud computing requires 24x7 internet access, while the other two can work even without the internet. Thus, they are more apt for the use cases where the IoT sensors may not have seamless connectivity to the internet.

4. **Security concerns:**

   In Fog, the data remains distributed among nodes. Thus, it is difficult to manipulate data as compared to the centralized structure of Cloud computing. In Edge computing, the data remains on the device itself, making it more secure out of the three. So, in the cases, where security is a major concern, Fog and Edge are preferable.

5. **Disaster concerns:**

   Again, because the data is distributed among nodes in Fog computing, the downtime in any disaster event will be minimal as compared to cloud computing, where everything is stored in one place and if anything goes wrong to it or with the communication lines, it result in total dysfunction takes down the whole system. But even if one node goes down in Fog computing, other nodes remain operational, making it the right choice for the use cases that require zero downtime.

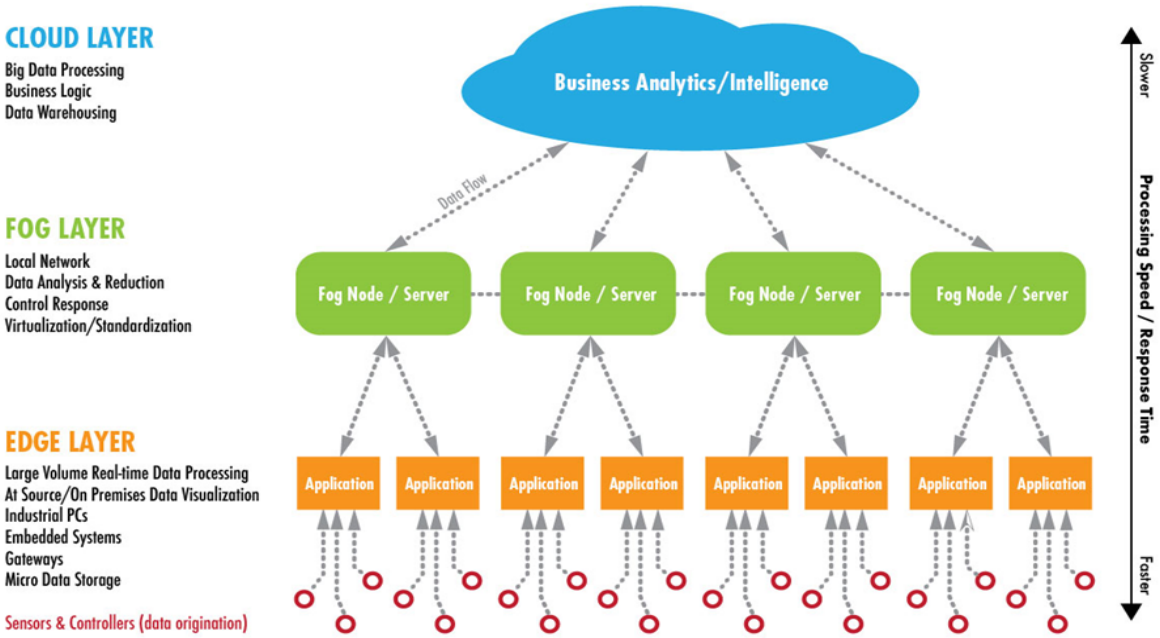

Figure 3.3: Industrial IoT Data Processing Layer Stack

Fog computing architecture adds an extra resource-rich layer between end devices and cloud to meet these challenges in the low latency, high reliability and security, high performance, mobility, and interoperability[27].

3.4.2: Comparison cloud computing Vs Fog computing

To present its properties, fog is compared with Cloud computing mode in this section. Cloud specifically because it is the first choice to appear in our scenario and in our situation from the point of view of availability and familiarity.

The table below shows the comparison between the main characteristics of the two architectures.

	Cloud Computing	**Fog computing**
Latency	significant	careless
Real time interactions	obtainable	obtainable
Mobility	restricted	widely Supported
Location awareness	Partially supported	Supported
Number of server nodes	limited	Large
Geographical distribution	Centralized	Decentralized and distributed
Distance to end devices	Far (multiple network hops)	Near (single network hop or few network hops)
Location of service	At provider servers	At the edge of the local network
Working environment	Specific data center building with conditioning systems	Outdoor (streets,base stations, etc.) or indoor (houses, cafes, etc.)
Communication mode	Over internet network	Over any kind of network LAN, MAN, WAN
Dependence on the quality of core network	Strong related	Related only on its specific network
Bandwidth costs	High	Low
Computation and storage capabilities	Unlimited	Limited
Energy consumption	High (especially the energy consumption of data center coolant system)	Low

Table 3.2: Comparison of cloud computing and fog computing

3.4.3: Summary

Both Edge and Fog computing are oriented to deal with main problem optimization of performance. Notice that Edge computing is more preferred by middle-ware companies and telecoms that work with backbone network and radio networks, Fog computing is more desired in case of data processing and service providers.

The most important difference between the IoT device or application collaborating with a cloud versus a node is that the bi-directional communication with a cloud server can take up to several minutes, while it may only take up to a few milliseconds when interacting with 'nodes' placed near the device.

While cloud computing still remains the first preference for storing, analyzing, and processing data, companies and customers are gradually moving towards Edge and Fog computing to get environments more appropriate with sensitive time applications, privacy and reduce costs.
The fundamental idea of adapting these two architectures is not to replace the Cloud completely but to segregate crucial information from the generic one. Smart applications and IoT based devices require instant decision-making tools, and while researchers and developers are adding new, enhanced, much better technics and methods that help in quick decisions, there's still a latency or lack of decisive nature, which calls for the implementation of Fog and Edge computing.

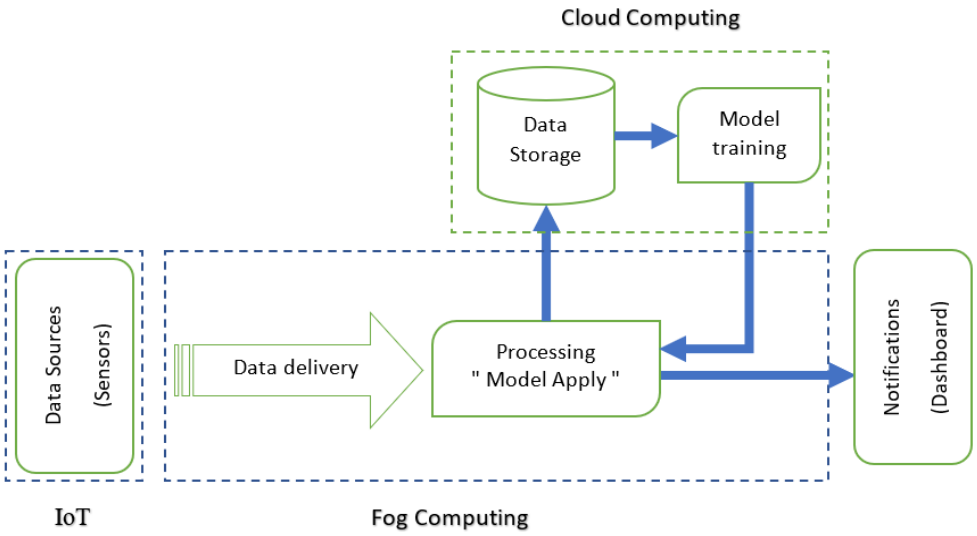

Figure 3.4: Computing paradigm selection

3.5: Chosen Data Processing framework

3.5.1: Apache Kafka

We'll explore Kafka's Advantages and Disadvantages. Since, before using any technology, it is very important to know the drawbacks, the same in the case of advantages.

Advantages:

So, here we list a few advantages of Kafka's. Such Kafka advantages basically make Kafka ideal for implementing our model. Let's continue with the descriptions of Kafka's advantages, then:

1. **High-throughput** Kafka is able to manage high-velocity and high-volume data without not requiring the massive hardware. Even, able to accept thousands of messages per second over message transmission.

2. **Low Latency** it can handle these messages with the very low latency of the millisecond range, needed by our use case.

3. **Fault-Tolerant** Kafka has an intrinsic potential to be node / machine failure prone within a cluster which is one of the main advantages of fault tolerance.

4. **Durability** That refers to data / message persistence on disk. Replication of messages is also one of the explanations for the longevity, since messages are never lost

5. **Lightweight** library, perfect for micro-services, IoT applications.

6. **Supports Stream joins** and uses rocksDb internally to preserve state.

7. **Exactly Once** (explained in part one) over Kafka 0.11).

Disadvantages

It is important to recognize the shortcomings of Kafka even though its benefits seem more common than its inconveniences. Yet do that only when it's too persuasive to ignore advantages. Here's another requirement that certain drawbacks may be more applicable to a particular use case but not really related to ours. So, here we mention some of the downside that Kafka has to offer:

1. **No Complete Set of Monitoring Tools**It is seen that a full collection of management and control tools is missing. Enterprise support staff therefore felt nervous or scared to select Kafka and support her in the long run.

2. **Message Tweaking Issues** Kafka 's efficiency decreases dramatically if there is any modification to the message. But, if the message is unchanged, it can work very well, as it uses the system capabilities.

3. **Reduces performance** Brokers and consumers are starting to compress and decompress these messages as the size increases. Because of this, when decompressed, the memory of the node is slowly being used. Also, compressing happens when the data is flowing through the pipeline. It affects throughput and performance as well.

4. Not for work as heavy lifting as Spark Streaming, Flink.

Storm's Advantages

The main strong points in Storm are [40]:

- The ISpout interface of Storm can possibly hold up any incoming data.In fact, by using Storm , users can ingest data from various real time synchronous and asynchronous systems (e.i, JMS, Kafka, Shell and Twitter).

- Based on Bolts,Storm enables to write data to any output system. because Storm provides the IBolt interface that supports any type of output systems like JDBC (to store data to any relational database), Sequence Files, Hadoop components like HDFS, Hive, HBase, and other messaging system.

- The storm cluster and Hadoop cluster are apparently similar. However, in Storm, it is possible to run different topologies for different storm tasks. Instead,in Hadoop platform, the only option consists in implementing Map Reduce jobs for the corresponding applications.

- One main difference between Map Reducejobs and topologies is the following. MapReduce job stops but a topology continues to process messages either all the time or until user terminate.

- Easy-to-use, rapid, scalable and fault-tolerant system, if one or more processes fails, Storm will automatically restart it. If the process fails repeatedly, Storm will reroute it to another machine and restart it there. It can be used for many cases such us real-time analytics, online machine learning, continuous computation and distributed RPC.
- It can process million tuples per second. Like MapReduce, and provides a simplified programming model, which hides the complexity of developing distributed applications.

3.5.2: Summary

After analyzing the differences and similarities between the five most popular frameworks for distributed streaming computing (Apache Kafka Streams, Spark Streaming, Flink, and Storm) in this chapter, we try comparing them with 10 criteria and note which considerations are most relevant to our objective choice.

Criterion / Framework	**Kafka streams** #	**Spark** #	**Flink** #	**Storm** #	**Samza** #
Data Processing Approaches	Streaming only	Micro batch only	Streaming and batch	Streaming and batch using the Trident API	Streaming only
Delay (latency The)	Low (no more than 1 millisecond)	High (>1 sec)	Low (no more than 1 millisecond)	Low (no more than 1 millisecond)	Low (<1 millisecond)
Stream primitive	The message (message) of the stream partition (stream partition) corresponding to the topic record Kafka	Discretized stream (DStream), a micro packet from several RDD (resilient distributed dataset) - fault tolerant distributed datasets	Data Stream (DataStream)	Tuple - a data stream in the form of immutable sets of key-value pairs	The message (message) of the stream partition (stream partition) corresponding to the topic record Kafka
Computing Primitive	Stream Processor	Task	Task	Bolt - stream handler	Task
Message Delivery Semantics	exactly once	exactly once	exactly once	at-least-once and exactly once using the Trident API	at-least-once
Saving application state (stateful)	Yes, the state is saved in the internal repositories of Kafka topics	Yes, the state is saved in an external distributed file system, such as HDFS	Yes, the state is saved in local storage and then moved to external long-term storage	Initially not (stateless), but the Trident API allows you to save state	Yes, the state is saved in the built-in storage of key-value pairs
Data Sources and Receivers	Apache kafka	Apache Kafka , Cassandra , HDFS , OpenStack Swift, Amazon S3,Kudu, MapR -FS	Apache Kafka , Cassandra , Amazon Kinesis, HBase , HDFS , Google Cloud Platform	Apache Kafka , Kestrel, Amazon Kinesis, HBase , Cassandra , RabbitMQ, JMS	Apache kafka
Resiliency Tools	message transaction mechanism	checkpoints mechanism	checkpoints mechanism	automatic restart of background tasks on the same cluster node or on another, in case of its failure	message transaction mechanism
Selective Data Processing	Time and Session Windows	Time windows	Time windows	No (No "windows" mechanism)	No (no "windows" mechanism)
Supported Languages	JVM languages only (Scala, Java)	Scala, Java, R, Python	Scala, Java	Any (thanks to Apache Thrift): Java, Scala, Python, C #, Ruby, etc.	JVM languages only (Scala, Java)

Table 3.3: Comparative of the most popular big data streaming frameworks [57]

In the following table we show Framework recommendation for different streaming application categories,

Category	**Description**	**Recommended framework**
ETL	Extract, transform and load	Storm, Spark Streaming
Stream ML	Stream machine learning	Spark Streaming
CEP	Complex event processing	Flink
IVP	Image and video processing	Storm

Table 3.4: Framework recommendation for different streaming application categories [39].

Development of real-time stream processing technology with Apache Storm helped address the latency issue, but not as a complete solution. For one thing, Storm did not guarantee state consistency with exactly-once processing and did not handle event time processing. People who had these needs were forced to implement these features in their application code.

As is the case with Storm and Spark Streaming, other new technologies in the field of stream processing offer some useful capabilities,but it's hard to find one with the combination of traits that Flink offers. Apache Samza, for instance, is another early open source processor for streaming data, but it has also been limited to at-least once guarantees and a low-level API.

To define a streaming topology in Samza, prior to compilation, you must explicitly define the inputs and outputs of the Samza tasks. The topology is set after the program has been compiled as the description is embedded in the application package distributed to YARN. In other words, An upgrade to topology will involve:

- Stop the YARN tasks already in progress.
- The program package is recompiled.
- The new program package will be distributed to YARN.

Which is not respectable in our case.

Similarly, Apache Apex provides some of the benefits of Flink, but not all (e.g., it is limited to a low-level programming API, it does not support event time, and it does not have support for batch computations). And none of these projects have been able to attract an open source community comparable to the Flink community

3.6: discussion

Apache kafka is chosen as the main data transmission framework for its lightweight and scalable characteristics and greater compatibility between the model parts in the transmission of data. In the first part, ensure that the data coming from the sensors are collected on the relevant topics and forwarded to the consistent consumer in the Stream processing part and a copy to the cloud storage location. In the second part, it delivers an updated model to a coherent stream processing engine cluster for the model.

The answer to the question in this chapter about the most satisfied data stream processing engine for our proposal is clearly 'Apache Flink' in both parts, the first stream processing and the second part of the model training. These two parts are connected to the second Apache Kafka server.

The visual presentation of these choices is shown in the figure 3.5 which shows the update of our model by including data processing options.

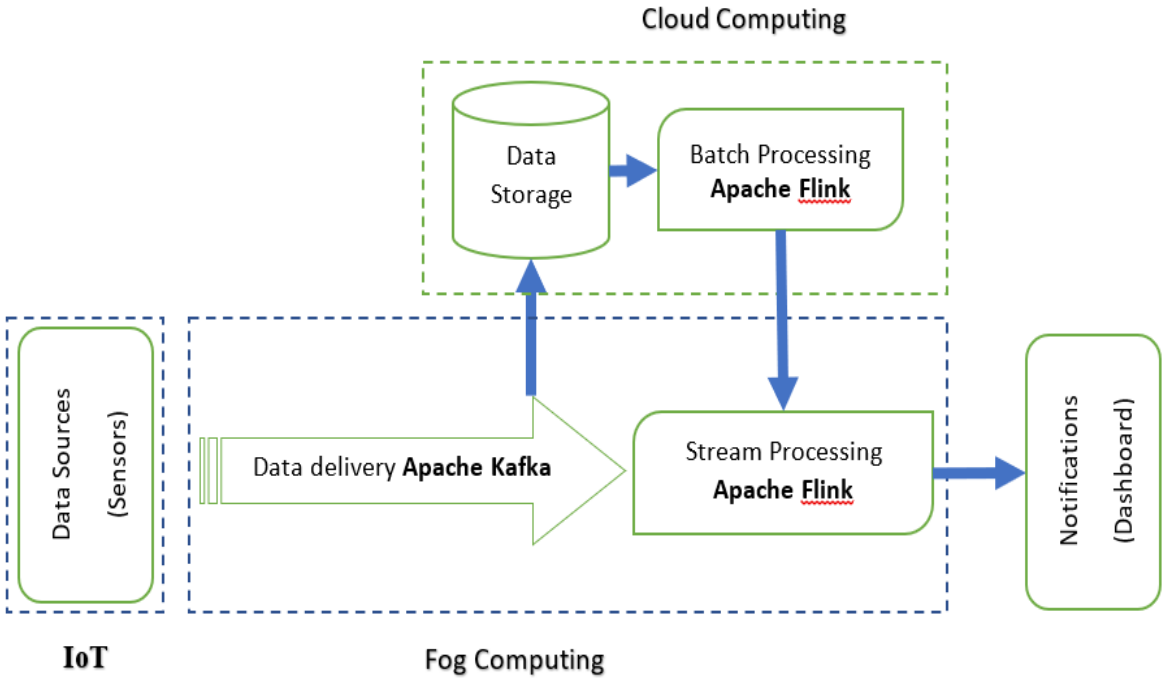

Figure 3.5: Data stream processing choices

3.6.1: Uses cases of Flink Apache

NetFlix :

- Various use cases including preparing data to feed their recommendation engine and various ETL jobs(batch and streaming).
- Flink process 3 trillion events by day.

Alibaba groups

- Runs Flink in production on thousands of nodes, peaking 2.5B events/sec. 100 TB state, sub second latency.
- Real-time search ranking on singles Day: 30% increase in conversion rate.

Uber

- Flink is deployed as company-wide. SQL streaming base platform.
- Cross-team applications: growth, alerting, forecasting and more.

3.7: Conclusion

Through this chapter. We compared the various technologies that provide us with the requirements for the final model. These comparisons were made specifically on the basis of business principles, the basis for building these technologies, and the functions and characteristics available to them. And it wasn't just performance and speed based.

This methodology allowed us to determine which techniques are available that are most responsive and appropriate to the requirements of the final model design. Which was Apache Kafka for the data transmission part and Apache FLink as the data processing engine. So that this combination can respond to all possibilities of scalability and provide the maximum of availability and fault-tolerance.

Chapter 4

Experiments and evaluation

4.1: Used Frameworks and Methods

Having implemented many DL frameworks in recent years has facilitated the exponential growth of interest in utilizing DL architectures in various domains. -Each system has its own strength on the basis of its supported DL architectures, optimization algorithms and ease of creation and deployment[15]. Several of these systems have been commonly used in research for the effective preparation of DNNs. In bellow the two frameworks used in our studies.

4.1.1: Keras

Keras is the high-level API of TensorFlow 2.0: an approchable, highly-productive interface for solving machine learning problems, with a focus on modern deep learning. It provides essential abstractions and building blocks for developing and shipping machine learning solutions with high iteration velocity[55].

Keras empowers engineers and researchers to take full advantage of the scalability and cross-platform capabilities of TensorFlow 2.0: you can run Keras on TPU or on large clusters of GPUs, and you can export your Keras models to run in the browser or on a mobile device[55] . In addition, it has been widely adopted by researchers and industry groups over the last year[25].

4.1.2: Tensorflow

Initially developed for Google Brain project, Tensorflow is an open source library for machine learning systems using various kinds of DNNs [7]. It is used by many Google products including Google Search, Google Maps and Street View, Google Translate, YouTube and some other products. Tensorflow uses graph representations to build neural network models. Developers can also take advantage of TensorBoard, which is a package to visualize neural network models and observe the learning process including updating parameters. Keras 2 also provides a high level of programming abstraction for Tensorflow[38].

4.1.3: Multi-layer Perceptron (MLP)

Multi-Layer Perceptron is a feed-forward artificial neural network which consists of a number of neurons connected by linking weights. MLP maps a group of inputs into a set of desired outputs.

MLP consists of three main parts of an input layer, a hidden layer and an output layer. The input layer receives the input data from the outside, then passes it to the first hidden layer, which will be forwarded until it finally reaches the output layer. This process is commonly known as a forward pass[24].

Input and output can be directly accessed, but hidden layer cannot. Each layer consists of several neurons. Neurons are connected between different layers using weight and bias[60]. The powerful characteristics of this type are :

- Multi-layer perceptrons generalize by detecting features of the input pattern that have been learn to be significant, and so coded into the internal units. Thus an unknown pattern is classified with others that share the same distinguishing features.
- It is this generalization ability that allows multiplayer perceptron to perform more successfully on real-world problems than other pattern recognition or expert system methods.
- Multi-layer perceptron networks are intrinsically fault-tolerant, since they are distributed parallel processing elements, with each node contributing to the final output response[16].

4.1.4: Convolutional Neural Network(CNN)

For vision-based tasks, DNNs with a dense connection between layers are hard to train and do not scale well. One important reason is the translation-in-variance property of such models. They thus do not learn the features that might transform in the image (e.g., rotation of hand in pose detection). CNNs have solved this problem by supporting translation equivariance computations. A CNN receives a 2-D input (e.g., an image or speech signal) and extracts high level features through a series of hidden layers.

The hidden layers consist of convolution layers as well as fully connected layers at the end. The convolution layer is at the core of a CNN and consists of a set of learnable parameters, called filters, that have the same shape as the input's shape but with smaller dimensions. In the training process, the filter of each convolutional layer goes through the whole input volume (e.g., in case of an image, it goes across the width and length of the image) and calculates an inner product of the input and the filter[38]. The main characteristics of CNN are :

- Convolution layers take biggest part of computations.
- Less connection compared to DNNs.
- Needs a large training dataset for visual tasks[38].

4.1.5: Recurrent Neural Network (RNN)

In certain tasks, prediction relies on a number of previous samples, such that, in addition to classifying individual samples, we do need to evaluate input sequences. For such applications, a neural feed-forward network is not applicable because it implies no connection between input and output layers. RNNs have been developed to solve this issue in sequential (e.g. speech or text) or time-series problems (data of the sensor) of different lengths[38].

Detection of driver habits in smart cars, detection of human movement patterns, input to the RNN consists of both the current sample and the previous sample. In other words,the output of RNN at time step $t-1$ would influence the output at time step t.

Each neuron is equipped with a feedback loop which returns the current output as input for the next step. This arrangement can be represented in such a way that each neuron in the RNN has an internal memory that holds the knowledge on the computation from the previous data[38]. RNNs are characterized by :

- Processes sequences of data through internal memory.
- Useful in IoT applications with time-dependent data.

4.1.6: Long Short-Term Memory (LSTM)

LSTM is an extension of RNNs. Different variations of LSTM have been proposed, though most of them have followed the same design of the original network[26].

LSTM uses the concept of gates for its units, each computing a value between 0 and 1 based on their input. In addition to a feedback loop to store the information, each neuron in LSTM (also called a memory cell) has a multiplicative forget gate, read gate, and write gate. These gates are introduced to control the access to memory cells and to prevent them from perturbation by irrelevant inputs. When the forget gate is active, the neuron writes its data into itself. When the forget gate is turned off by sending a 0, the neuron forgets its last content. When the write gate is set to 1, other connected neurons can write to that neuron. If the read gate is set to 1, the connected neurons can read the content of the neuron.

An important difference of LSTMs compared to RNNs is that LSTM units utilize forget gates to actively control the cell states and ensure they do not degrade. The gates can use sigmoid or tanh as their activation function. In fact, these activation functions cause the problem of vanishing gradient during back-propagation in the training phase of other models using them[38]. This kind of ANNs is characterized by :

- Good performance with data of long-time lag.
- Access to memory cell is protected by gates.

4.2: Used dataset

4.2.1: Power System Datasets

It is original from Mississippi State University and Oak Ridge National Laboratory dated on 4/15/2014. Produced by doctors Uttam Adhikari, Shengyi Pan, and Tommy Morris in collaboration with Raymond Borges and Justin Beaver of Oak Ridge National Laboratories (ORNL) have created this datasets which include measurements related to electric transmission system normal, disturbance, control, cyber attack behaviors. Measurements in the dataset include synchrophasor measurements and data logs from Snort, a simulated control panel, and relays.

The power system datasets have been used for multiple works related to power system cyber-attack classification[3].

- Pan, S., Morris, T., Adhikari, U., Developing a Hybrid Intrusion Detection System Using Data Mining for Power Systems, IEEE Transactions on Smart Grid.
- Pan, S., Morris, T., Adhikari, U., Classification of Disturbances and Cyber-attacks in Power Systems Using Heterogeneous Time-synchronized Data, IEEE Transactions on Industrial Informatics.
- Pan, S., Morris, T., Adhikari, U., A Specification-based Intrusion Detection Framework for Cyber-physical Environment in Electric Power System, International Journal of Network Security (IJNS), Vol.17, No.2, PP.174-188, March 2015.
- Beaver, J., Borges, R., Buckner, M., Morris, T., Adhikari, U., Pan, S., Machine Learning for Power System Disturbance and Cyber-attack Discrimination, Proceedings of the 7th International Symposium on Resilient Control Systems, August 19-21,2014, Denver, CO, USA.

4.2.2: Dataset generator system

The figure below shows the power system framework configuration used in generating these scenarios. In the network diagram we have several components, firstly, G1 and G2 are power generators. R1 through R4 are Intelligent Electronic Devices (IEDs) that can switch the breakers on or off.

These breakers are labeled BR1 through BR4. We also have two lines. Line One spans from breaker one (BR1) to breaker two (BR2) and Line Two spans from breaker three (BR3) to breaker four (BR4). Each IED automatically controls one breaker. R1 controls BR1, R2 controls BR2 and son on accordingly.

The IEDs use a distance protection scheme which trips the breaker on detected faults whether actually valid or faked since they have no internal validation to detect the difference. Operators can also manually issue commands to the IEDs R1 through R4 to manually trip the breakers BR1 through BR4. The manual override is used when performing maintenance on the lines or other system components [3].

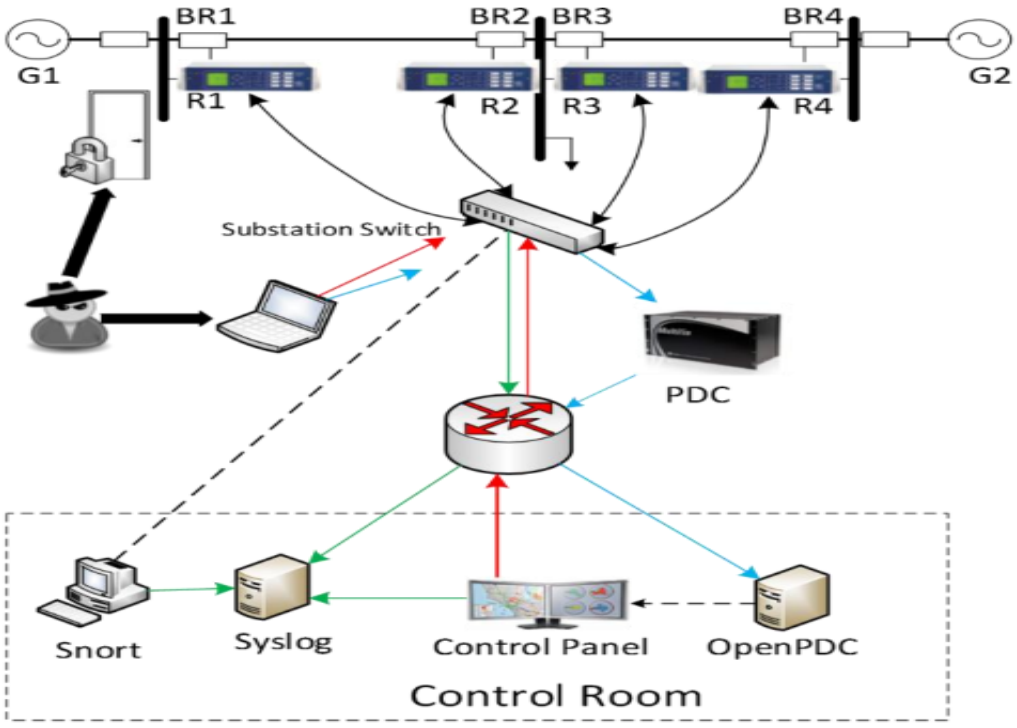

Figure 4.1: Power system framework configuration used in generating this DataSet[3]

4.3: Implemented modules

4.3.1: Data Preprocessing

In any Machine Learning process , Data Preprocessing is the stage that transforms or encodes the data to such a state that the computer could easily interpret it. In other words, the data features can be widely interpreted by an algorithm.

A dataset can be described as a series of data objects, often also referred to as records, points, vectors, patterns, events, instances, tests, observations, or entities. Data objects are characterized by a variety of features recording an object's basic characteristics, such as the mass of a physical object or the time an event occurred, etc. Attributes are also referred to as factors, features, fields, attributes, or dimensions.

At the first place.In our case, the vector collects at a specific point in time pieces of data obtained from sensors, each column (feature) is important and can not be ignored. For this reason we face the restriction to avoid column falling under the laws of hard correlation. This constraint has posed the hardest problem to deal with in the model development process.

Secondly, data transformation means modifying data in the form that improves computing and gaining more information without losing any portion of it, two methods have been used in the development of the model, so they are: standard scalar and transformer, both of which are portion of the ski-learn library. The main dataset's preprocessing steps are :

- Handling with missing value and replace Nan for mean.
- Apply normalization in datasets
- Apply standralization after normalization in datasets

4.3.2: Missing value handler

Missing values can be in two forms: infinite or null, they can be the origin of a device fault or a connection failure. From all options for dealing with these situations we choose to replace them with the mean value of the feature in question to avoid drop option that is not allowed because the model will handle the data in real time and individual event manner.

Code

```
df[df==np.inf] = np.nan
df.fillna(df.mean(), inplace=True)   #df: is datasets
```

4.3.3: Feature scaling

a. Normalizer scales data for each sample to a unit norm. This scaling technique scales both NumPy arrays and SciPy sparse matrix[42].

For example, if the feature values were x, y and z then the transformed normalized value for x would be as shown below[42]:

$$z = \frac{x_i}{\sqrt{(x_i^2 + y_i^2 + z_i^2)}}$$

Code:

```
from sklearn.preprocessingimport Normalizer
transformer = Normalizer().fit(X) # X:is the datasets
S transformer.transform(X)
```

b. StandardScaler [4] transforms the dataset such that the resulting distribution's mean value is zero and the standard deviation is one. Transformed value is obtained by subtracting mean value from the original value and dividing by the standard deviation.The formula given below is used for the transformation :

$$z = \frac{(x - \mu)}{\sigma}$$

Where :

- z the transformed value of the feature.
- x the original value.
- μ the mean.
- σ the standard deviation.

In our experiment,x the features's number in the dataset used is equal 128,the dataset has a shape (78377 rows and 128 columns).

Code:

```
from sklearn.preprocessing import StandardScaler
ss = StandardScaler()
x = pd.DataFrame(ss.fit_transform(S)) # S : is results the normalization datasets
```

4.3.4: Loss functions

Binary cross entropy: It is the loss function which identifies the loss based on the probability value. The probability value lies between 0 and 1. A perfect model with zero loss will have probability value 0. Below given is the formula used for binary classification[56].

$$BCE = -(y \log(p) + (1 - y) \log(1 - p))$$

4.4: Settings of implemented modules :

Data set characteristics :

- Data sets Size :78 mega bytes.
- Features number : 129.
  - 128 for input features.
  - 1 for target feature.
- Examples Number : 78377.

To accomplish step of building and testing models, we had use the following settings for :

Algorithms	Number of epochs	X_train shape	X_test shape	Activation function		
				Input layer	Hidden layers	Output layer
MLP	250	(70539,128)	(7838,128)	tanh	tanh	sigmoid
RNN	250	(70539,1,128)	(7838,1,128)	tanh	tanh	sigmoid
LSTM	250	(70539,1,128)	(7838,1,128)	tanh	tanh	sigmoid
CNN 1D	250	(70539,1,128)	(7838,1,128)	tanh	tanh	sigmoid
CNN 2D	250	(70539,8,16,1)	(7838,8,16,1)	tanh	tanh	sigmoid

Table 4.1: settings different algorithms for build model and fitting model

4.5: Results and discussion

4.5.1: Multi-layer Perceptrons (MLP)

Pseudo code

```
i=x_train.shape[1] #number of features
import time
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
model = keras.Sequential([
layers.Dense(512, activation='tanh', input_shape=[i,],kernel_initializer='he_uniform' ),
    layers.Dense(1024, activation='tanh',kernel_initializer='he_uniform',use_bias=False    ),
    layers.Dense(512, activation='tanh',kernel_initializer='he_uniform',use_bias=False    ),
    layers.Dropout((0.01)),
    layers.Dense(512, activation='tanh',kernel_initializer='he_uniform',use_bias=False   ),
layers.Dense(256, activation='tanh',kernel_initializer='he_uniform',use_bias=False),
    layers.Dense(256, activation='tanh',kernel_initializer='he_uniform',use_bias=False),
```

```
    layers.Dense(128, activation='tanh',kernel_initializer='he_uniform',use_bias=False),
    layers.Dropout((0.001)),
    layers.Dense(2, activation='sigmoid',kernel_initializer='he_uniform' )
  ])
optimizer = tf.keras.optimizers.Adam(0.001, decay=0.000005)
model.compile(loss=tf.losses.BinaryCrossentropy(),
              optimizer=optimizer,
              metrics=['accuracy'])
```

MLP's Result

For the MLP models we found a training performance mean greater than 0.98. In general, thus, MLP models are able to match the training results.
The low standard deviations suggest the model's robustness with respect to the data set for measurement. Figure 4.2 represents the past of the 250 epochs of school.

The top curves reflect the mean precision of training and validation, whilst the down side curves display the mean lack of training and validation. While the loss of training steadily declines, the loss of validity begins A substantial rise from epoch 100 on, to reach the value after 250 epochs greater than 0.64. If we follow the training failure curve, when after 250 epochs it consistently decreases to a value below 0.004, Figure 4.3.

The steep curve for the lack of training demonstrates the model 's failure to meet the training results. Although we use dropout as a means of regularization and this is due not to the application of kernel regularization L1 or L2, we are not able to prevent the model from being over-fitted. The explanation for model overfitting is that we have relatively fewer training data, and over time the model tends to see the same instances in each epoch. One approach to overcome this is by the technique of increasing data volume throughout the time of model preparation. The model has reached training performance peaking at 0.992. The model achieved verification accuracy, which peaked at 0.931.

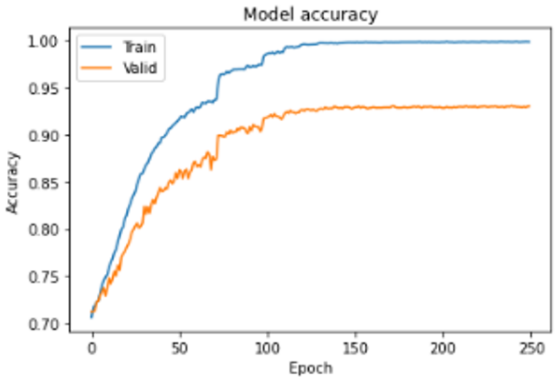

Figure 4.2: MLP Accuracy results

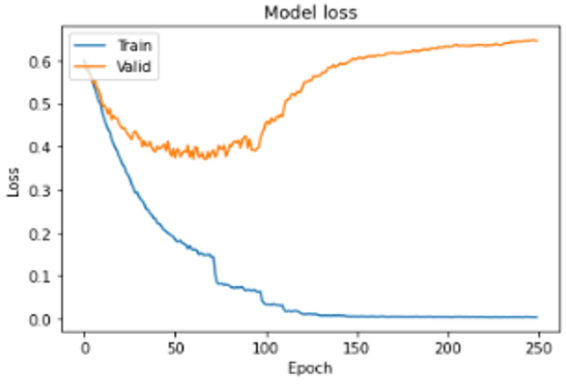

Figure 4.3: MLP loss results

4.5.2: Convolutional Neural Network 1D (CNN-1D)

Pseudo code

```
i=x_train.shape[1]
```

```
model = models.Sequential()
#conv1
model.add(layers.Conv1D(256,  kernel_size=7,
input_shape=(1, i), padding="same",activation='tanh'))
model.add(layers.BatchNormalization())
#conv2
model.add(layers.Conv1D(256,  kernel_size=5, padding="same", activation='tanh'))
model.add(layers.BatchNormalization())
#conv3
model.add(layers.Conv1D(128, kernel_size=5, padding="same" ,activation='tanh'))
model.add(layers.BatchNormalization())
model.add(layers.Conv1D(256,  kernel_size=7, padding="same",activation='tanh'))
model.add(layers.BatchNormalization())
model.add(layers.GlobalAveragePooling1D())
model.add(layers.Dense(512, activation='tanh',kernel_initializer='he_uniform'))
model.add(layers.Dense(2, activation='sigmoid',kernel_initializer='he_uniform'))
optimizer = tf.keras.optimizers.Adama(0.001,decay=0.000005)
model.compile(loss=tf.keras.losses.BinaryCrossentropy(),optimizer=optimizer,metrics=['accuracy'])
```

CNN-1D's Result

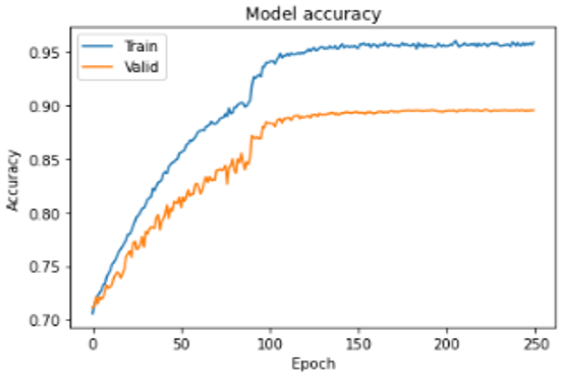

Figure 4.4: CNN-1D Accuracy results

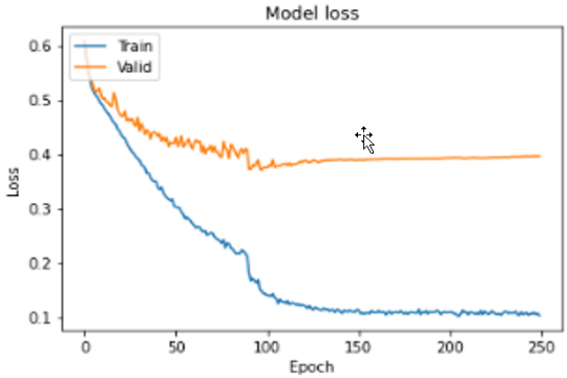

Figure 4.5: CNN-1D loss results

4.5.3: Convolutional Neural Network 2D (CNN-2D)

CNN-2D Pseudo code

```
model = Sequential()
model.add(Conv2D(filters = 64,padding="same", kernel_size = (5, 5), activation='tanh',
            input_shape = (8, 16, 1)))##convert x to 3d(128,1) features to (8,16,1)
model.add(Conv2D(filters = 64,padding="same", kernel_size = (7, 7), activation='tanh'))
model.add(Conv2D(filters = 128,padding="same", kernel_size = (7, 7),strides=(2,2), activation='tanh'))

model.add(MaxPooling2D(pool_size=(3,3),padding="same",strides=(2,2)  ))
model.add(Dropout(0.1))
model.add(Conv2D(filters = 64,padding="same", kernel_size = (5, 5), activation='tanh'))
```

```
model.add(Conv2D(filters = 64,padding="same", kernel_size = (5, 5),strides=(2,2) , activation='tanh'))

model.add(MaxPooling2D(pool_size=(3,3),padding="same",strides=(2,2)))
model.add(Conv2D(filters = 128,padding="same", kernel_size = (3, 3), activation='tanh'))
model.add(Flatten())
model.add(Dense(512, activation='tanh',kernel_initializer="he_uniform"))
model.add(Dense(2, activation='sigmoid',kernel_initializer="he_uniform"))
model.compile(loss=tf.keras.losses.BinaryCrossentropy(),
optimizer =tf.keras.optimizers.Adam(0.001,decay=0.000005), metrics=["accuracy"])
```

CNN-2D's result

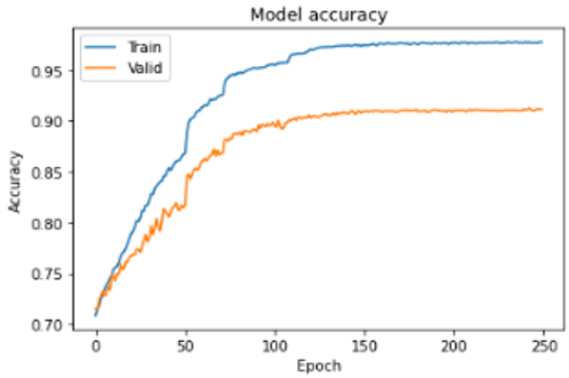

Figure 4.6: CNN-2D Accuracy results

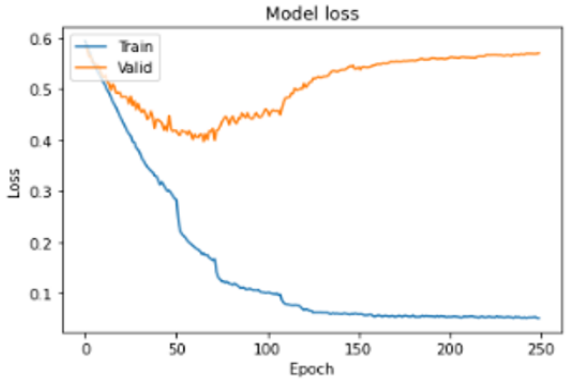

Figure 4.7: CNN-2D loss results

4.5.4: Recurrent Neural Network (RNN)

Pseudo code

```
i=x_train.shape[1]
rnn_model = keras.Sequential([
    layers.SimpleRNN(128, activation="tanh", input_shape=[1,i] ,return_sequences=True,
kernel_initializer="he_uniform"),
  layers.SimpleRNN(256,activation="tanh",return_sequences=True,kernel_initializer="he_uniform"),
  layers.SimpleRNN(128,activation="tanh",return_sequences=False,kernel_initializer="he_uniform"),
   layers.Dropout(0.01),
   layers.Dense(2, activation="sigmoid",kernel_initializer="he_uniform")
 ])
optimizer = tf.keras.optimizers.Adam(0.001,decay=0.000005)
rnn_model.compile(loss=tf.keras.losses.BinaryCrossentropy(),optimizer=optimizer,metrics=['accuracy'])
```

RNN's Result

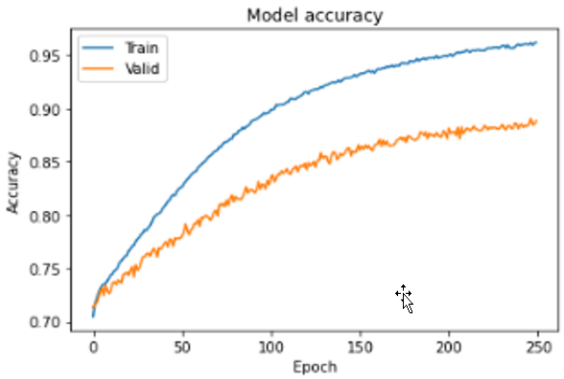

Figure 4.8: RNN Accuracy results

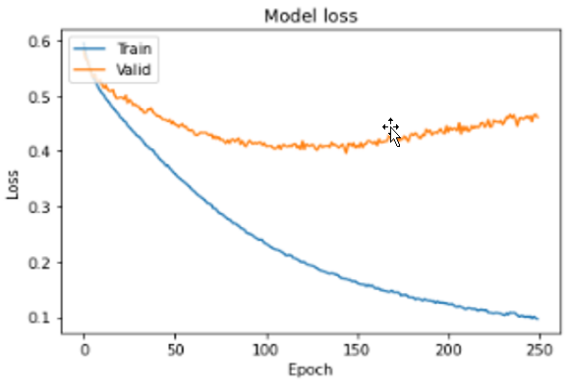

Figure 4.9: RNN loss results

4.5.5: Long Short-Term Memory (LSTM)

Pseudo code

```
i=x_train.shape[1]
model = tf.keras.Sequential()
model.add(layers.LSTM(128, activation='tanh',input_shape=(1, i), return_sequences=True,
kernel_initializer='he_uniform'))
model.add(layers.LSTM(256, activation='tanh', re-
turn_sequences=True,kernel_initializer='he_uniform'))

model.add(layers.LSTM(256, activation='tanh', re-
turn_sequences=True,kernel_initializer='he_uniform'))
```

```
model.add(layers.LSTM(128, activation='tanh', re-
turn_sequences=False,kernel_initializer='he_uniform'))

model.add(layers.Dense(2,activation='sigmoid',kernel_initializer='he_uniform'))
model.compile(loss="binary_crossentropy", optimizer=tf.optimizers.Adam(0.001,
 decay=0.000005), metrics=['accuracy'])
```

LSTM's Result

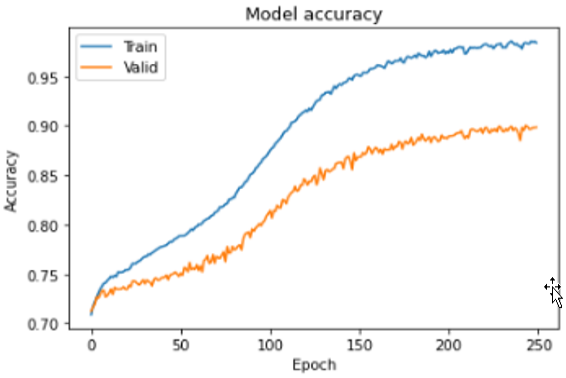

Figure 4.10: LSTM Accuracy results

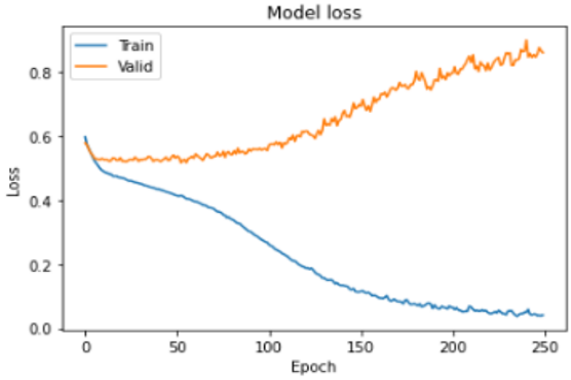

Figure 4.11: LSTM loss results

4.5.6: Code source

The full version of the code source of the modules implemented above, and the data set used, are available for consultation and use through this link. [2]

4.5.7: Comparison

Algorithms	Training accuracy	Test accuracy	Training Time (s)	Execution Time (s)
MLP	99.20%	93.10%	255.77	0.034
RNN	96.80%	89.30%	295.37	0.038
LSTM	97.80%	90.30%	3408.09	0.044
CNN 1D	97.10%	89.30%	227.95	0.038
CNN 2D	98.30%	91.00%	750.66	0.036

Table 4.2: Results Comparison

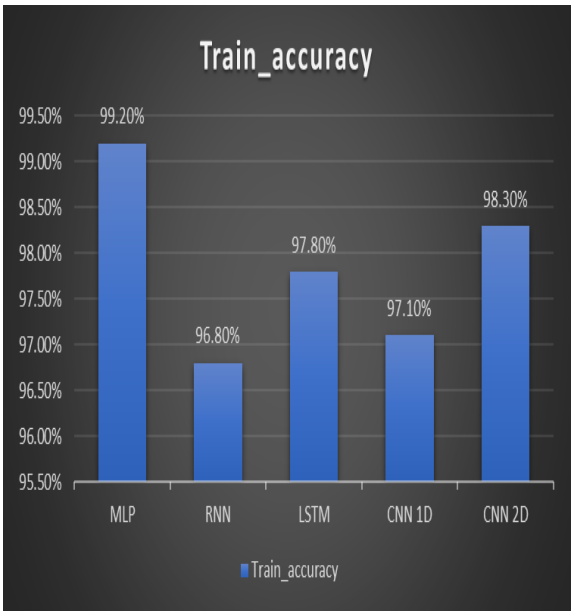

Figure 4.12: Train accuracy

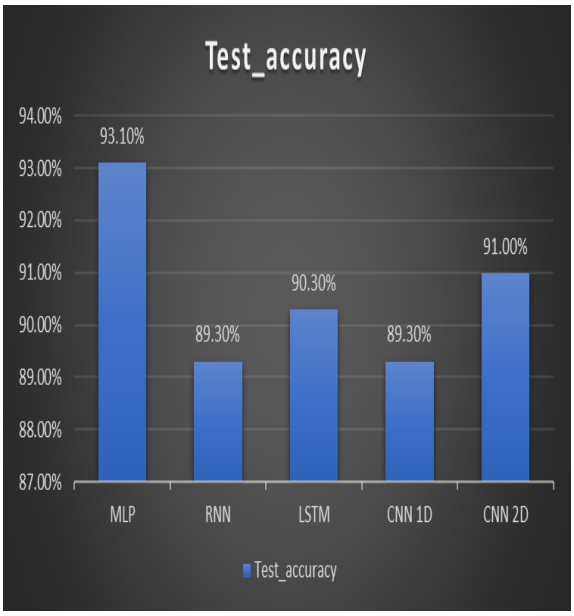

Figure 4.13: Test Accuracy

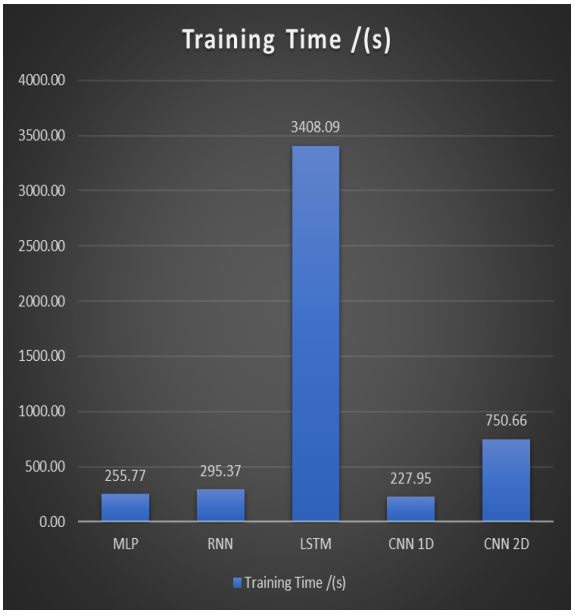

Figure 4.14: Comparison Training Time

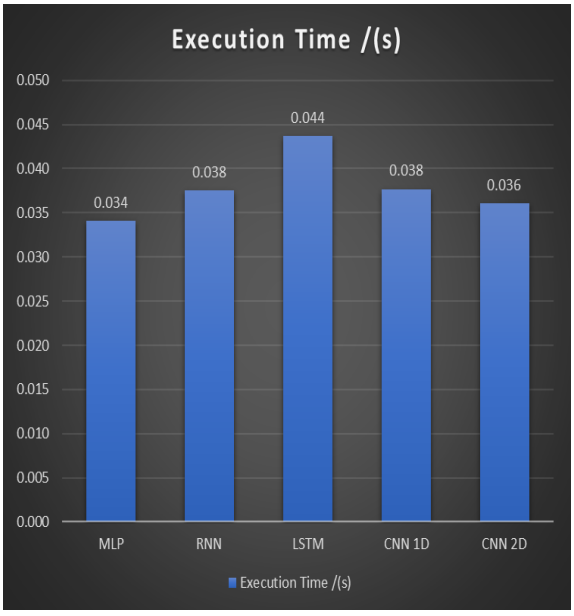

Figure 4.15: Comparison Execution Time

4.5.8: Conclusion

Every type of neural network has its domain preferred, we tested different ones on the same DataSet to determine the one to use in our case, the results for Accuracy, Loss and Time are shown in the figures above.

In time-sensitive applications, time is the first consideration which made the choice of coherent model, MLP shows the lowest time reported for the training and execution phases.

During the test process, the accuracy factor reached 93.10, which is similar to the actual situation. It's the best outcome of all those tested. Even in time execution the same model gave results in less time compared to others.

These findings are explained by the fact that the dataset used is closed to the original MLP domain. In other words, our dataset is a linear data set. which made Multi-layer perceptrons the closest option to fulfill the mission.

Other models have not even provided satisfaction results for our case, but stealing could be the best way for other DataSets to use cases that involve other types of data, such as voice or images. For this reason, we have developed an approach to comparing the test between different methods from different families in order to obtain a model that allows the best record in accuracy and time of execution.

Advantages of the proposed system over the used methods

a. In the proposed model there is no need to perform feature extraction separately.
b. Network learns the data by itself and passes the relevant information through multiple layers.
c. As the amount of data increases; as deep neural networks give better and better results.

4.6: General Conclusion

We have successfully proposed a modal prototype for time-sensitive applications in our research, which can reduce the delay by a considerable amount. Based on this framework, we can introduce IoT-based real-world applications as the main source of data, use deep learning to gain insight and add more computational capacity to fog nodes. Our theory study shows that deep learning is the most tested approach to use in monitoring systems on the other hand, our experimental findings show the approach to consider and highlight fundamental elements in selecting the right deep learning method that can help to avoid severe delays and could be detrimental to delay-sensitive applications such as health care where the delay could be life threatening, or manufacturing control where decision might done in real time manner.

Our proposed architecture is one of the very few prototypes to address the open question – how to design and build a truly global data monitoring system that benefits from AI methods and responds to the most difficult constraints.

Our work addresses the expansion of IoT systems and their incorporation with fog-based cloud computing for improved and more efficient service delivery to the customer. We pay to use the bandwidth of the cloud. Therefore, if we can access data from the fog instead of heading to the cloud, it is also economically advantageous. However, fog computing can not fully replace cloud computing, as it will still be used for storage and high-end computing intensive work that is very popular in the business world.

We can therefore conclude that cloud computing and fog computing will complement each other with their own advantages and disadvantages. Edge computing plays a key role in the Internet of Things (IoT). Fog computing helps emerging network paradigms that require fast processing with minimal delay, while cloud computing would serve the business community to meet their high-end computing demands for big data processing based on the utility pricing model.

Fog computing works better than cloud computing to meet the demands of fast data transfer to the client and smart decision-making in emergency situations for time-sensitive applications. The greatest benefit of fog computing is that it decreases the delay relative to the cloud, which is evident from our analysis and comparison findings. This is important for time-sensitive applications where the data needs to enter the client at the least possible time.

4.7: Future Work

Our work leads to a simple framework for implementing a IoT-based data monitoring model for time-sensitive applications exploring AI method. Our work also provides experimental results that demonstrate the approach to choose the best approach to get low latency and to ensure data accuracy. Although our architecture provides a clear insight into the fog computing model for time-sensitive applications, there are opportunities for more study and many improvements can be made

- More Fog Intelligence – In our work, fog nodes serve as ephemeral storage and data relay nodes. However, the number of emergency situations in various industries, such as healthcare , manufacturing, etc., is rising. It is becoming increasingly necessary to generate alerts for such emergency situations in a fast and easy manner. Therefore, we would like to expand our work to provide the fog nodes with multiple intelligence and computational authority to get a fast and efficient way of notifying fog node emergencies.

- Data aggregation and compression – Actually, we have not used any data aggregation and compression schemes in our work. However, there could be varieties of data from various industries , such as manufacturing and social media, that may be substantially larger in size. Trimming and pre-processing data before being sent to the cloud is very necessary for better and quicker service delivery. To help reduce the cloud and network burden, we would like to introduce an effective data aggregation and compression algorithm between fog and cloud when dealing with large data samples.

- More effective Node Discovery algorithm – The architecture that we have suggested here is a rather simplified framework. However, there could be thousands of intermediate nodes in a wide real-world scenario. Maintaining a list of these nodes and working through the list to find the right one could be computationally costly which could lead to a delay that we can not afford in delay-sensitive applications. Therefore, we would like to work on a more effective node discovery algorithm that could locate the near-optimal node much faster and more efficiently without causing much delay. The algorithm will be dynamic as well as capable of efficiently handling the addition and deletion of node from the network.

- The future aim will be to add real sensors and a large number of machines to introduce and test the original parallel architecture that we proposed. In addition, instead of emulating the distance from the data center, we would like to link our system to the actual data centers and re-run our experiments.

Bibliography

[1] Batch, stream, and micro-batch processing: A cheat sheet | upsolver. https://www.upsolver.com/blog/batch-stream-a-cheat-sheet. (Accessed on 09/21/2020). xi, 36, 37

[2] Github - bachir-barika/system-modeling-and-monitoring-based-iot: we are constribute for system exploite data and processing of strategy deep learning. https://github.com/bachir-barika/system-modeling-and-monitoring-based-IoT?fbclid=IwAR1CJlCHvaShMCIBoJtDIC2VbWnjvroF2c-p3KUO2DZfP4mlZfRf1RlBLxU. (Accessed on 09/22/2020). 75

[3] Industrial control system (ics) cyber attack datasets - tommy morris. https://sites.google.com/a/uah.edu/tommy-morris-uah/ics-data-sets. (Accessed on 09/01/2020). xii, 66, 67

[4] sklearn.preprocessing.standardscaler — scikit-learn 0.23.2 documentation. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html?highlight=standardscaler#sklearn.preprocessing.StandardScaler. (Accessed on 09/01/2020). 69

[5] Monitor | definition of monitor by oxford dictionary on lexico.com also meaning of monitor. https://www.lexico.com/en/definition/monitor, 03 2020. (Accessed on 06/02/2020). 21

[6] Monitoring dictionary definition | monitoring defined. https://www.yourdictionary.com/monitoring, 03 2020. (Accessed on 06/02/2020). 21

[7] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016. 62

[8] T. Alam. A reliable communication framework and its use in internet of things (iot). *International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT)*, 3(5):450–456, 2018. xi, 14

[9] M. A. Alsheikh, D. Niyato, S. Lin, H.-P. Tan, and Z. Han. Mobile big data analytics using deep learning and apache spark. *IEEE network*, 30(3):22–29, 2016. 50

[10] M. A. Alsmirat, Y. Jararweh, I. Obaidat, and B. B. Gupta. Internet of surveillance: a cloud supported large-scale wireless surveillance system. *The Journal of Supercomputing*, 73(3):973–992, 2017. 13

[11] G. Alsuhly and A. Khattab. An iot monitoring and control platform for museum content conservation. In *2018 International Conference on Computer and Applications (ICCA)*, pages 196–201, 2018. 50

[12] 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*, 53(4):50–58, 2010. 27

[13] H. Atlam, R. Walters, and G. Wills. Fog computing and the internet of things: A review. *Big Data and Cognitive Computing*, 2, 04 2018. xi, 29

[14] L. Atzori, A. Iera, and G. Morabito. The internet of things: A survey. *Computer networks*, 54(15):2787–2805, 2010. xi, 2, 3

[15] S. Bahrampour, N. Ramakrishnan, L. Schott, and M. Shah. Comparative study of deep learning software frameworks. *arXiv preprint arXiv:1511.06435*, 2015. 62

[16] R. Beale and T. Jackson. *Neural Computing-an introduction*. CRC Press, 1990. 63

[17] 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*, pages 13–16, 2012. 34

[18] A. J. Calderón Godoy and I. González Pérez. Integration of sensor and actuator networks and the scada system to promote the migration of the legacy flexible manufacturing system towards the industry 4.0 concept. *Journal of Sensor and Actuator Networks*, 7(2):23, 2018. 24

[19] W.-F. Cheung, T.-H. Lin, and Y.-C. Lin. A real-time construction safety monitoring system for hazardous gas integrating wireless sensor network and building information modeling technologies. *Sensors*, 18(2):436, 2018. 22

[20] S. Cirani, G. Ferrari, M. Picone, and L. Veltri. *Internet of Things: Architectures, Protocols and Standards*. John Wiley & Sons, 2018. 2

[21] N. Dey, A. E. Hassanien, C. Bhatt, A. Ashour, and S. C. Satapathy. *Internet of things and big data analytics toward next-generation intelligence*. Springer, 2018. 35

[22] T. Fowdur, Y. Beeharry, V. Hurbungs, V. Bassoo, and V. Ramnarain-Seetohul. Big data analytics with machine learning tools. In *Internet of Things and Big Data Analytics Toward Next-Generation Intelligence*, pages 49–97. Springer, 2018. 17, 20

[23] E. Friedman and K. Tzoumas. *Introduction to Apache Flink: Stream Processing for Real Time and Beyond*. " O'Reilly Media, Inc.", 2016. xi, 46, 47

[24] N. Harun, W. L. Woo, and S. Dlay. Performance of keystroke biometrics authentication system using artificial neural network (ann) and distance classifier method. In *International Conference on Computer and Communication Engineering (ICCCE'10)*, pages 1–6. IEEE, 2010. 63

[25] W. G. Hatcher and W. Yu. A survey of deep learning: Platforms, applications and emerging research trends. *IEEE Access*, 6:24411–24432, 2018. 62

[26] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 65

[27] P. Hu, S. Dhelim, H. Ning, and T. Qiu. Survey on fog computing: architecture, key technologies, applications and open issues. *Journal of network and computer applications*, 98:27–42, 2017. 32, 52

[28] P. Hu, H. Ning, T. Qiu, Y. Zhang, and X. Luo. Fog computing based face identification and resolution scheme in internet of things. *IEEE transactions on industrial informatics*, 13(4):1910–1920, 2016. 32

[29] W. Inoubli, S. Aridhi, H. Mezni, M. Maddouri, and E. M. Nguifo. An experimental survey on big data frameworks. *Future Generation Computer Systems*, 86:546–564, 2018. xi, 36, 38, 42, 43, 44, 45

[30] K. Kai, W. Cong, and L. Tao. Fog computing for vehicular ad-hoc networks: paradigms, scenarios, and issues. *the journal of China Universities of Posts and Telecommunications*, 23(2):56–96, 2016. 33

[31] J. Lee, S. D. Noh, H.-J. Kim, and Y.-S. Kang. Implementation of cyber-physical production systems for quality prediction and operation control in metal casting. *Sensors*, 18(5):1428, 2018. 24

[32] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao. A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications. *IEEE Internet of Things Journal*, 4(5):1125–1142, 2017. xi, 6, 7, 8, 10, 11, 12, 28, 29

[33] C. Liu, Y. Cao, Y. Luo, G. Chen, V. Vokkarane, M. Yunsheng, S. Chen, and P. Hou. A new deep learning-based food recognition system for dietary assessment on an edge computing service infrastructure. *IEEE Transactions on Services Computing*, 11(2):249–261, 2017. 49

[34] M. Marjani, F. Nasaruddin, A. Gani, A. Karim, I. A. T. Hashem, A. Siddiqa, and I. Yaqoob. Big iot data analytics: architecture, opportunities, and open research challenges. *IEEE Access*, 5:5247–5261, 2017. 30

[35] E. Markakis, G. Mastorakis, C. X. Mavromoustakis, and E. Pallis. *Cloud and Fog Computing in 5G Mobile Networks: Emerging Advances and Applications*. Institution of Engineering and Technology, 2017. 29

[36] M. H. Miraz, M. Ali, P. S. Excell, and R. Picking. A review on internet of things (iot), internet of everything (ioe) and internet of nano things (iont). In *2015 Internet Technologies and Applications (ITA)*, pages 219–224. IEEE, 2015. 2

[37] M. Mohammadi, A. Al-Fuqaha, M. Guizani, and J.-S. Oh. Semisupervised deep reinforcement learning in support of iot and smart city services. *IEEE Internet of Things Journal*, 5(2):624–635, 2017. 50

[38] M. Mohammadi, A. Al-Fuqaha, S. Sorour, and M. Guizani. Deep learning for iot big data and streaming analytics: A survey. *IEEE Communications Surveys & Tutorials*, 20(4):2923–2960, 2018. xi, xiii, 12, 13, 16, 17, 19, 20, 62, 64, 65

[39] H. Nasiri, S. Nasehi, and M. Goudarzi. Evaluation of distributed stream processing frameworks for iot applications in smart cities. *Journal of Big Data*, 6(1):52, 2019. xiii, 57

[40] A. Oussous, F.-Z. Benjelloun, A. A. Lahcen, and S. Belfkih. Big data technologies: A survey. *Journal of King Saud University-Computer and Information Sciences*, 30(4):431–448, 2018. 41, 43, 56

[41] C. Perera, A. Zaslavsky, M. Compton, P. Christen, and D. Georgakopoulos. Semantic-driven configuration of internet of things middleware. In *2013 Ninth International Conference on Semantics, Knowledge and Grids*, pages 66–73. IEEE, 2013. 2

[42] D. M. Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. 2011. 68

[43] C. Prakash. Spark streaming vs flink vs storm vs kafka streams vs samza : Choose your stream processing framework. `https://medium.com/@chandanbaranwal/spark-streaming-vs-flink-vs-storm-vs-kafka-streams-vs-samza-choose-your-stream-processing-91ea3f04675b`, Mai 2018. (Accessed on 06/02/2020). 38, 40

[44] A. Rayes and S. Salam. The things in iot: Sensors and actuators. In *Internet of Things From Hype to Reality*, pages 57–77. Springer, 2017. 2, 3, 4

[45] F. J. Riggins and S. F. Wamba. Research directions on the adoption, usage, and impact of the internet of things through the use of big data analytics. In *2015 48th Hawaii International Conference on System Sciences*, pages 1531–1540. IEEE, 2015. 2

[46] S. Sarkar and S. Misra. Theoretical modelling of fog computing: a green computing paradigm to support iot applications. *Iet Networks*, 5(2):23–29, 2016. 31, 32

[47] S. Saxena and S. Gupta. *Practical real-time data processing and analytics: distributed computing and event processing using Apache Spark, Flink, Storm, and Kafka*. Packt Publishing Ltd, 2017. 40

[48] S. Shams, S. Goswami, K. Lee, S. Yang, and S.-J. Park. Towards distributed cyberinfrastructure for smart cities using big data and deep learning technologies. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, pages 1276–1283. IEEE, 2018. 12, 13

[49] U. S. Shanthamallu, A. Spanias, C. Tepedelenlioglu, and M. Stanley. A brief survey of machine learning methods and their sensor and iot applications. In *2017 8th International Conference on Information, Intelligence, Systems & Applications (IISA)*, pages 1–8. IEEE, 2017. 17

[50] J. Shi, J. Wan, H. Yan, and H. Suo. A survey of cyber-physical systems. In *2011 international conference on wireless communications and signal processing (WCSP)*, pages 1–6. IEEE, 2011. 32, 33

[51] B. N. Silva, M. Khan, C. Jung, J. Seo, D. Muhammad, J. Han, Y. Yoon, and K. Han. Urban planning and smart city decision management empowered by real-time data processing using big data analytics. *Sensors*, 18(9):2994, 2018. 24, 26

[52] D. Svozil, V. Kvasnicka, and J. Pospichal. Introduction to multi-layer feed-forward neural networks. *Chemometrics and intelligent laboratory systems*, 39(1):43–62, 1997. 18

[53] M. Syafrudin, G. Alfian, N. L. Fitriyani, and J. Rhee. Performance analysis of iot-based sensor, big data processing, and machine learning model for real-time monitoring system in automotive manufacturing. *Sensors*, 18(9):2946, 2018. 24, 49

[54] B. Tang, Z. Chen, G. Hefferman, T. Wei, H. He, and Q. Yang. A hierarchical distributed fog computing architecture for big data analysis in smart cities. In *Proceedings of the ASE BigData & SocialInformatics 2015*, pages 1–6. 2015. 49

[55] K. Team. Keras documentation: About keras. `https://keras.io/about/`. (Accessed on 08/21/2020). 62

[56] D. Thara, B. PremaSudha, and F. Xiong. Auto-detection of epileptic seizure events using deep neural network with different feature scaling techniques. *Pattern Recognition Letters*, 128:544–550, 2019. 69

[57] A. Vichugova. Big data streaming framework: Kafka streams, spark, flink, storm, samza. `https://www.bigdataschool.ru/bigdata/choice-factors-4-big-data-streaming.html`, 10 2019. (Accessed on 06/02/2020). xiii, 57

[58] A. Vichugova. Comparison of 5 popular big data streaming frameworks. `https://www.bigdataschool.ru/bigdata/big-data-streaming-kafka-streams-vs-spark-vs-flink-vs-storm-vs-samza.html`, October 2019. (Accessed on 06/02/2020). 38

[59] X. Wang, L. Gao, S. Mao, and S. Pandey. Deepfi: Deep learning for indoor fingerprinting using channel state information. In *2015 IEEE wireless communications and networking conference (WCNC)*, pages 1666–1671. IEEE, 2015. 13

[60] I. R. Widiasari, L. E. Nugroho, et al. Deep learning multilayer perceptron (mlp) for flood prediction model using wireless sensor network based hydrology time series data mining. In *2017 International Conference on Innovative and Creative Information Technology (ICITech)*, pages 1–5. IEEE, 2017. 63

[61] I. Yaqoob, I. A. T. Hashem, A. Gani, S. Mokhtar, E. Ahmed, N. B. Anuar, and A. V. Vasilakos. Big data: From beginning to future. *International Journal of Information Management*, 36(6):1231–1247, 2016. 2

[62] P. Zandl. Edge computing and fog computing; or, on the benefits of fogging » insight. `https://www.energomonitor.com/insight/edge-computing-fog-computing-benefits-of-fogging/`, 01 2016. (Accessed on 09/27/2020). xi, 28

[63] X. Zhang, J. Zhang, L. Li, Y. Zhang, and G. Yang. Monitoring citrus soil moisture and nutrients using an iot based system. *Sensors*, 17(3):447, 2017. 22