PEOPLES DEMOCRATIC REPUBLIC OF ALGERIA

Ministry of Higher Education and Scientific Research

University Mohamed Khider – BISKRA

Faculty of Exact and Natural Science

Computer science department

Order Number: RTIC12 /M2/2021

THESIS

Submitted in fulfilment of the requirements for the Master's degree in

Computer Science

NETWORK AND INFORMATIN AND COMMUNICATION TECHNOLOGY

QoS-driven Self-Adaptive IoT system

BY :

DEHIMI Raouane

Members of the jury:

Sahraoui Somia	MCB	President
Zernadji Tarek	MCB	Supervisor
Bendahman Toufik	MAA	Member

Session 2020-2021

This work is dedicated to:

My parents the joy of my life,

My siblings Zakaria & Dalila my pride,

Adla, Kanza, Imane, Zahia, Saliha, Rachida my aunts,

Lahcen, Fathi, Said, Ibrahim, my uncles.

My Grandmother Malika,

My Friend Chaima.

Acknowledgements

Above all, praise and thank Allah most gracious and merciful for his showers of blessings throughout my research work to complete it successfully.

I would like to express my profound and sincere gratitude to my research director,

Dr. Zernadji Tarek, for his immense knowledge, his valuable advice and his help in our work. I have truly admired her support and understanding all along.

My thanks go to all the teachers in the computer department who helped me with my education.

I am very grateful for the love, care, encouragement and prayers of my parents. I am especially grateful to all my family members, My sister and My brother and uncles for their love and support.

My thanks also go to my precious friends especially Chaima, classmates, and dear members of the Biskra Computer Science Club, having such great people around me had a huge impact on my life and pushed me forward to always give my best to everyone I love and care about.

Thank you

Abstract

IoT applications are subject to a variety of uncertainties, such as sudden changes in traffic load, communication interference. this has given rise to a big challenge to provide consistent quality of service in the network. The self-adaptation was finding to provides the means to automate tasks that humans would otherwise perform, without errors, and varying and potentially inconsistent expertise. Self-adaptation is considered to be the best solution to dynamically manage a system in the occurrence of deviations from the expected quality of service (QoS) parameters.

Currently, the Machine learning and Deep Learning techniques can be used to facilitate adaptation because they ensure that the system can learn from multiple data and improve over a period of time.

What we are suggesting is a model to be able to count on changes earlier than a QoS gap occurs. The approach constantly monitors QoS parameters and predicts capacity differences from QoS parameters primarily based entirely on old statistics, by change the values of old QoS parameter that was may cause derivation in QoS to the new values that improve the QoS.

Keywords: IoT, Uncertainties, QoS, Self-adaptation, Machine Learning, Deep Learning.

Résumé

Les applications IoT sont soumises à diverses incertitudes, telles que des changements soudains de la charge de trafic, des interférences de communication. Cela a donné lieu à un grand défi pour fournir une qualité de service constante dans le réseau. L'auto-adaptation a été trouvée pour fournir les moyens d'automatiser les tâches que les humains effectueraient autrement, sans erreurs, et avec une expertise variable et potentiellement incohérente. L'auto-adaptation est considérée comme la meilleure solution pour gérer dynamiquement un système en cas d'écart par rapport aux paramètres de qualité de service (QoS) attendus.

Actuellement, les techniques d'apprentissage automatique et d'apprentissage profond peuvent être utilisées pour faciliter l'adaptation, car elles garantissent que le système peut apprendre à partir de plusieurs données et s'améliorer sur une période de temps.

Suggérons, c'est un modèle pour pouvoir compter sur les changements avant qu'un écart de QoS ne se produise. L'approche surveille en permanence les paramètres de QoS et prédit les différences de capacité par rapport aux paramètres de QoS en se basant principalement sur d'anciennes statistiques, en modifiant les valeurs de l'ancien paramètre de QoS qui pouvait entraîner une dérivation de la QoS vers les nouvelles valeurs qui améliorent la QoS.

Mot clé : IoT, QoS, Self-adaptation, Les techniques d'apprentissage, d'apprentissage en profondeur.

ملخص

تخضع تطبيقات إنترنت الأشياء لمجموعة متنوعة من أوجه عدم اليقين، مثل التغيرات المفاجئة في عبء حركة المرور، وتداخل الاتصالات، وقد أدى ذلك إلى ظهور تحدٍ كبير لتوفير جودة خدمة متسقة في الشبكة، وتم العثور على التكيف الذاتي لتوفير الوسائل اللازمة أتمتة المهام التي قد يؤديها البشر، بدون أخطاء، وخبرات متنوعة وربما غير متسقة. يعتبر التكيف الذاتي أفضل حل لإدارة نظام الديناميكي بالآلة في حالة تدهور عن معلمات الخدمة المطلوبة. يمكن استخدام تقنيات التعلم الآلي والتعلم العميق لتسهيل التكيف لأنها تضمن أن النظام يمكن أن يتعلم من بيانات متعددة ويتحسن على مدى فترة من الزمن.

ما نقترحه هو نموذج للتمكن من الاعتماد على التغييرات في وقت أبكر من حدوث فجوة جودة الخدمة. يراقب النهج باستمرار معلمات جودة الخدمات ويتوقع اختلافات السعة من معلمات جودة الخدمات التي تعتمد بشكل أساسي بالكامل على الإحصائيات القديمة، عن طريق تغيير قيم معلمة جودة الخدمات القديمة التي قد تسبب تدهور في جودة الخدمات للقيم الجديدة التي تعمل على تحسين جودة الخدمات .

كلمات مفتاحية: إنترنت الأشياء، جودة الخدمة، عدم اليقين، التكيف الذاتي، التعلم الآلي، التعلم العميق.

Contents

List of Figures

List of Tables

General Introduction

General Introduction

1-General context

The Internet of Things (IoT) is made up of small embedded computers (motes) fitted with low-power wireless networks, sensors and actuators. These motes form networks able to monitor and control the physical world and thus connect digital processes to our physical environment. **[1]**

Recent technological innovations and the rapid merger of fields such as sensing and actuation technologies, embedded systems, wireless communication and data analytics are accelerating the growth of the Internet of Things (IoT). The IoT has undergone rapid growth over the last decade. There are an estimated more than 6 billion devices currently connected to the Internet and more than 25 billion devices connected by 2020 **[2].**

A large number of sensors implemented in the Internet of Things generate a large amount of data for various applications such as smart home, smart medical, smart manufacturing, smart transportation, smart grid, and smart agriculture. Analyzing these data to help decision-making, to increase productivity and accuracy, to improve incomes is an essential process which makes 'IoT a valuable idea for business and a paradigm for improving living standards.

In recent years, tremendous progress has been made in machine learning (ML) and its use of the Internet of Things (IoT). Deep Learning is part of a broader family of machine learning, Deep learning would play a vital role in creating a smarter IoT as it has given remarkable results in various areas. Data can be collected by sensors at the edge and used for training Deep learning models. **[3]**

2-Problematic and Objectives

IoT systems have Quality of service (QoS) requirements, like as low energy, low packet loss, Connectivity and latency. IoT environments presents challenges in finding the right network settings is hard as IoT applications are subject to a variety of uncertainties, such as sudden changes in traffic load and communication interference.

Traditionally, network configuration requires manual intervention to eliminate uncertainties that cause continuous network maintenance or inefficiency. Human operators monitor the system and make adjustments when problems or more problems are discovered. They usually see opportunities to improve system performance. Although humans understand the general background of the problem better than computers, human operators are prone to experiences of slow reaction time, fatigue, errors, and diversity and potential conflict. Self-adaptation provides a way to automate tasks that would otherwise be performed by humans.

Applying Deep Learning Model in IoT can be considered for ease of adaptation, because it enables the system to learn from multiple data and improve over time. the prediction method leads to that the various adaptation actions having to be defined prior to the deviation of the required objectives.

DeltaIoT is the first exemplar for research on self-adaptation that provides both a simulator for offline experimentation and a physical setup that can be accessed remotely for real-world experimentation.**[1]**

This work focuses on designing and implementing a system that enables IoT (Internet of Things) system to adapt to the performance changes in order to maintain its required quality goals (e.g., The average packet loss over 24 h should not exceed 10%, or energy consumption should be minimized). The developed system would have to dynamically reconfigure the IoT system variances from expected quality of service (QoS) parameters.

3-Outlines

In the first chapter, we study the basics of the IoT industry. In the second chapter, we study about Self-Adaptation field and we explain the Conceptual Model of a Self-adaptive System. In the Third chapter, the Deep Learning starting with ML to Deep Learning. The fourth chapter involved the conception and Design. In fifth chapter implementation of our approach to a self-adaptive IoT system.

Chapter 01

Internet of Things

Chapter 01: Internet of Things

1-Introduction

Over the few last decades, human has been looking for the ways could make his life comfortable and easy. Mankind had experienced the technical transformational journey with the new inventions of technology, this technology was very helpful and performed every possible work in shorter duration and better accuracy, with the new concept "smart concept", the world has been more connected "hyper-connected world".

The smart concepts include smart phones, smart devices, smart applications and smart cities, those concepts form an ecosystem of devices whose basic work is to connect various devices to send and receive data. Internet of Things is one of the dominating technologies that keeps an eye on the connected smart devices. It has laid an incredible impact on the technical, social, economic and in the lives of humans and machines.

In this chapter, our objective is present the main concept of IoT, by the definition of IoT, define different application domain, challenges, & QoS in IoT.

2-The Internet of Things

Definition1. The Internet of Things (IoT) provides information exchange and communication for device-to-device, device to-people, and device-to-environment. The IoT is a network system that connects equipped with minuscule identifying devices such as RFID, sensors and smart objects with the Internet according to the information shared by the sensing devices and the agreed protocols to realize quick, reliable and real-time information exchange and communication, achieving intelligent identification, location, tracking, monitoring, and management. The interconnected objects are inexhaustible sources of information, it creates a vast amount of data which need communication infrastructure, computational and processing unit to convert this data into useful information to enable real-time decision making. This infrastructure is placed generally in cloud. [**4**]

Definition2.The internet of things, or IoT, is a system of interrelated computing devices, mechanical and digital machines, objects, animals or people that are provided with unique identifiers (UIDs) and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction.**[5]**

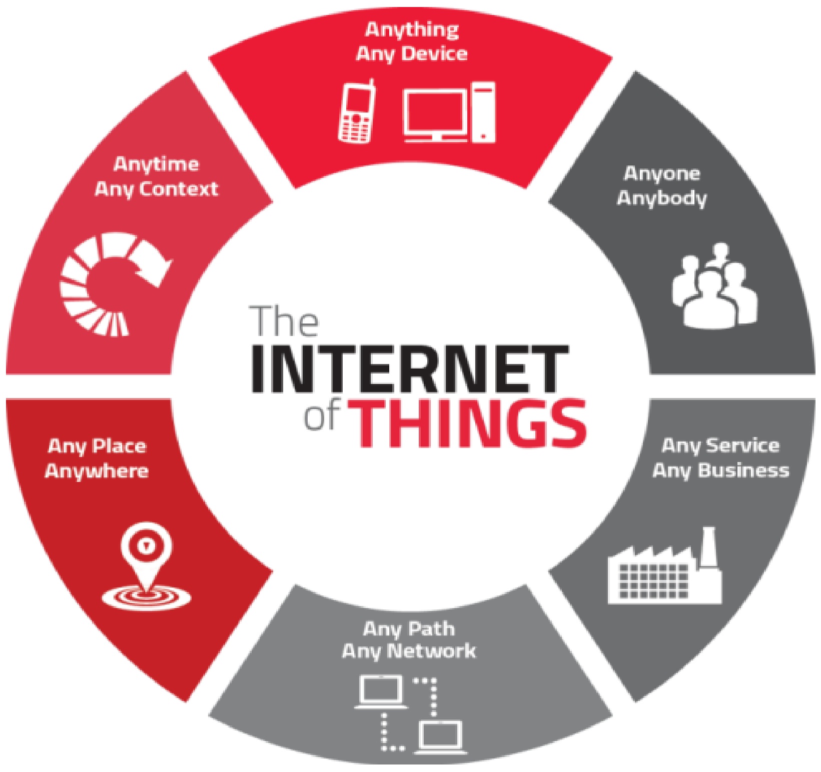


Figure 1.1. Internet of Things **[6]**

2.1-Characteristics of IoT

- **Interconnectivity**: Concerning the IoT, anything can be interconnected with the global information and communication infrastructure.
- **Things-related services**: The IoT is capable of providing thing-related services within the constraints of things, such as privacy protection and semantic consistency between physical things and their associated virtual things. To provide thing-related services within the constraints of things, both the technologies in physical world and information world will change.
- **Heterogeneity**: The devices in the IoT are heterogeneous as based on different hardware platforms and networks. They can interact with other devices or service platforms through different networks.
- **Dynamic changes**: The state of devices change dynamically, e.g., sleeping and waking up, connected and/or disconnected as well as the context of devices including location and speed. Moreover, the number of devices can change dynamically.
- **Enormous scale**: The number of devices that need to be managed and that communicate with each other will be at least an order of magnitude larger than the devices connected to the current Internet. Even more critical will be the management of the data generated and their interpretation for application purposes. This relates to semantics of data, as well as efficient data handling.
- **Safety**: As we gain benefits from the IoT, we must not forget about safety. As both the creators and recipients of the IoT, we must design for safety. This includes the safety of our data and the safety of our physical well-being. Securing the endpoints, the networks, and the data moving across all of it means creating a security paradigm that will scale.
- **Connectivity**: Connectivity enables network accessibility and compatibility. Accessibility is getting on a network while compatibility provides the common ability to consume and Produce data.**[6]**

2.2-Architecture of IoT

There is no single consensus on architecture for IoT, which is agree universally. Different researchers have proposed different architectures.

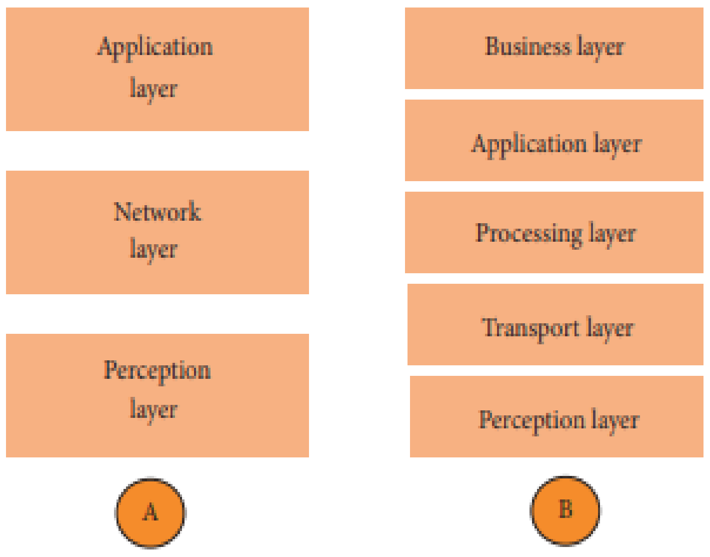


Figure 1.2. Architecture of IoT (A: three layers) (B: five layers) **[7]**

2.2.1- Three Layer Architectures

The most basic architecture is a three-layer architecture as shown in Figure 2.A. It was introduced in the early stages of research in this area. It has three layers, namely, the perception, network, and application layers.

- ✓ **The perception layer:** is the physical layer, which has sensors for sensing and gathering information about the environment. It senses some physical parameters or identifies other smart objects in the environment…
- ✓ **The network layer:** is responsible for connecting to other smart things, network devices, and servers. Its features are also used for transmitting and processing sensor.
- ✓ **The application layer:** is responsible for delivering application specific services to the user. It defines various applications in which the Internet of Things can be deployed, for example, smart homes, smart cities, and smart health.

2.2.2-Five Layer Architectures

The five layers are perception, transport, processing, application, and business layers (Figure2.B). The role of the perception and application layers is the same as the architecture with three layers. We outline the function of the remaining three layers.

- ✓ **The transport layer:** transfers the sensor data from the perception layer to the processing layer and vice versa through networks such as wireless, 3G, LAN, Bluetooth, RFID, and NFC.
- ✓ **The processing layer:** is also known as the middleware layer. It stores, analyzes, and processes huge amounts of data that comes from the transport layer. It can manage and provide a diverse set of services to the lower layers. It employs many technologies such as databases, cloud computing, and big data processing modules.
- ✓ **The business layer:** manages the whole IoT system, including applications, business and profit models, and users' privacy. The business layer is out of the scope of this paper. Hence, we do not discuss it further.**[7]**

2.3-Application of IoT

2.3.1-Smart Cities Domain

The smart city mission is to make better the old urban infrastructure in cities. The concept of smart cities is to make its existence in the development of urban development strategy. There are lot of existing technologies in urban cities, like as internet, wireless communication, infrared, Bluetooth, Wi-Fi differing from technologies and their range. Smart city is to make best use of public resources by increasing quality of services and decrease the expense.

- ❖ Effective Water Supply.
- ❖ City Lighting.
- ❖ Road Traffic.
- ❖ Citizens Safety.
- ❖ Garbage Management.
- ❖ Air Standard.
- ❖ Smart Parking.
- ❖ Crowdsensing.
- ❖ Energy Efficient Buildings.
- ❖ Trustworthy Public Transportation.

- ❖ Environment

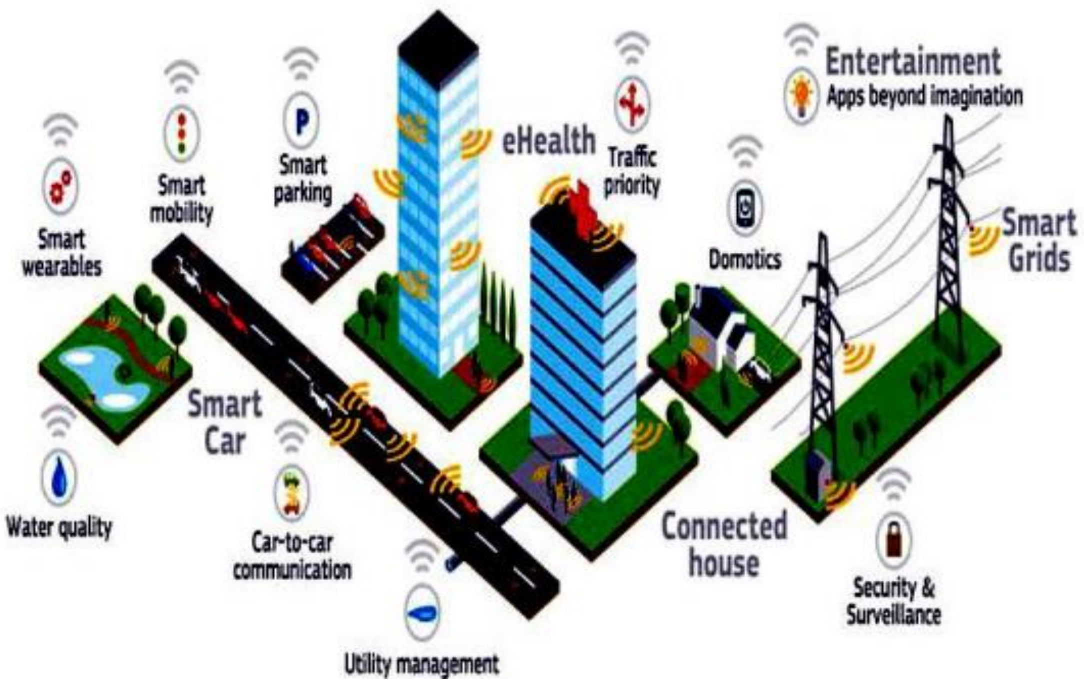


Figure 1.3. The Internet of Things Smart Cities.**[8]**

2.3.2-Smart Energy Domain

The IoT system provides a strong means of managing the consumption cost of energy and optimize the output of enterprises. IoT is considered to play a pivotal role in creating these energy management systems smarter. The Internet of Things is enabling energy efficiencies while creating a new way for energy conservation. This will make you efficient in energy utilization while keeping the costs to a minimum.

- ❖ Smart Meters
- ❖ Mending and Maintenance
- ❖ Collect Consumption Data Easy
- ❖ Monitor Water Services and Remotely
- ❖ Electricity Pole Surveillance

2.3.3-Smart Healthcare Domain

In an environment where healthcare remains notably expensive to the average citizen, the overall age of the global population is increasing, and the prevalence of chronic diseases is on the rise, the advent of the Internet of Things shows assurance in helping healthcare facilities operate more efficiently. This technology is set to change the healthcare industry within the next decade, as it has great potential and multiple potential applications, from remote monitoring to medical device integration.

- End-to-end Affordability and Connectivity
- Data Classification and Analysis
- Tracking and Warning
- Remote Medical Assistance
- Lower Costs
- Better Sick Person Experience
- Preferable Management of Drugs and Medicine
- Improved Outcomes of Treatment

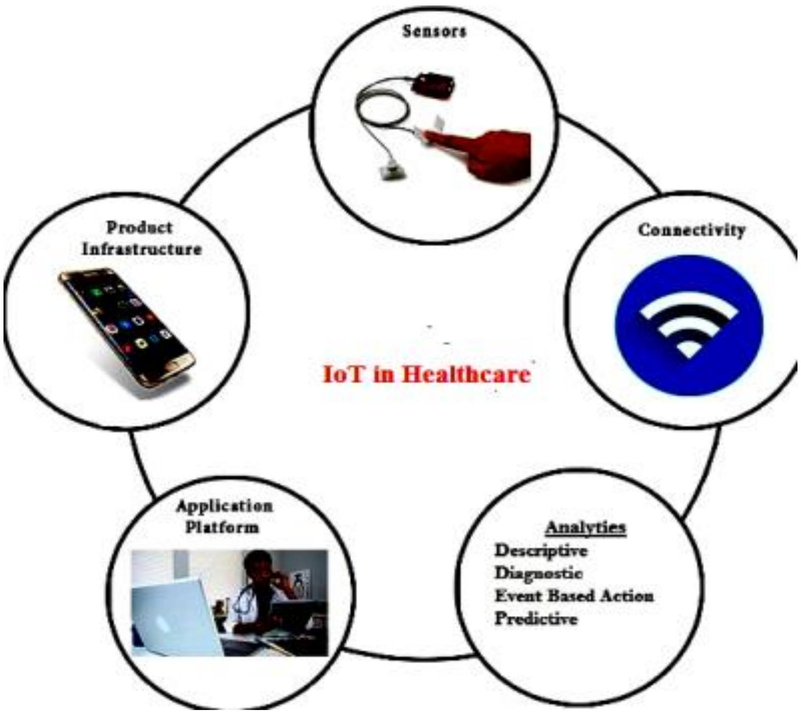


Figure1.4. The Internet of Things Smart HealthCare.**[3]**

2.3.4-Smart Homes Domain

Smart home technologies can unlock both individual and society-wide advantage in dissimilar ways. They can provide economic saving, make better convenience for consumers, contribute to more ecological and sustainable living, reinforce the buyer's sense of safety and security, and more.

- Network Providers
- Platform Providers
- Service Provide
- Distributed Architecture
- Health Care
- Security Services
- Energy
- Entertainment

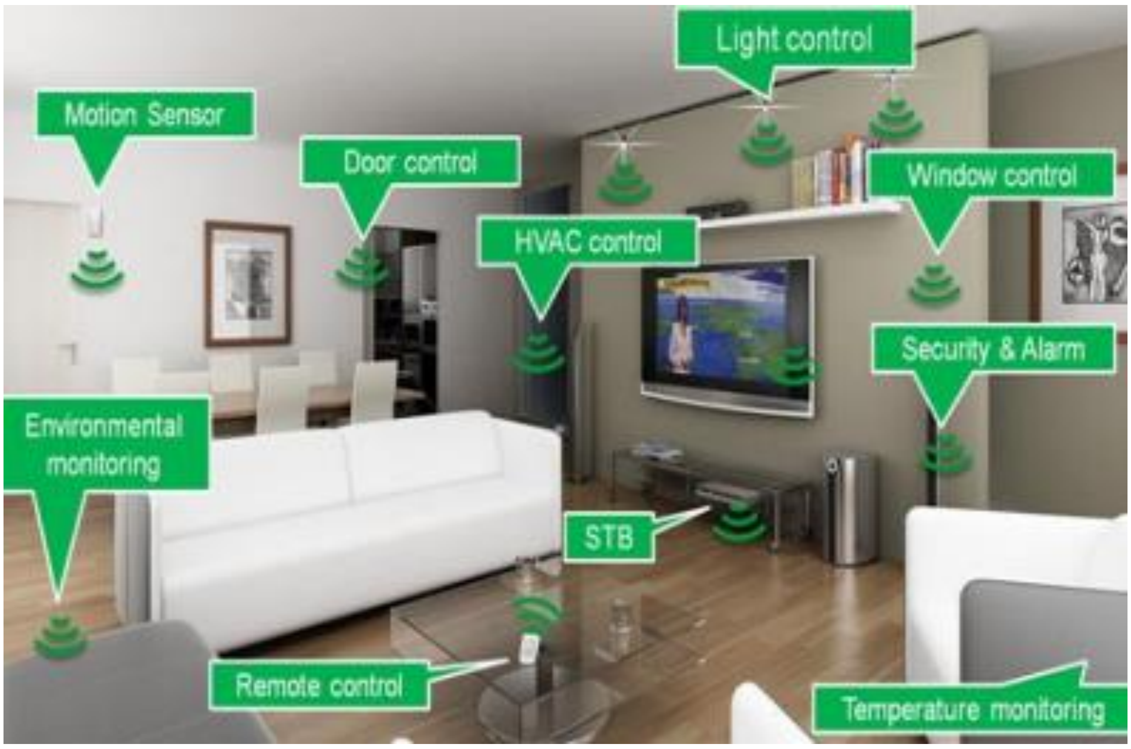


Figure 1.5. The Internet of Things Smart Homes.**[3]**

2.3.5- Internet of Things in Agriculture Domain

In IoT based smart farming, a system is built for monitoring the crop field with the assist of sensors like as light, humidity, temperature, soil moisture, and automating the irrigation system. The farmers can monitor the field conditions from anyplace. The applications of IoT based smart farming not only aim traditional, huge farming operations, but could also be new levers to uplift other growing or common trends in agricultural like organic farming, family farming and improve highly transparent farming. IoT-based smart farming can provide great advantage, including more efficient water usage, or optimization of inputs and treatments.

IoT recent trends in agriculture are using technology in order to make smarter decisions, decrease costs, and boost production. **[5]**

2.4- Challenges and recent research of IoT

2.4.1-Networking

Generally, the Networking issue has a great relevance in the Internet because of it includes some of the important factors which uses to manage networks. First of all, traffic and protocols that have a significant impact on the behavior of the network, sought to deal with networking challenges via mobile Ad-Hoc Network.

The authors have used mobile ad hoc networks (MANET) interconnected to fixed networks by different gateway. In IoT, can't be predicted where the object moved, and the object may be needed to transmit from network to another. The biggest problem is in dynamic gateways change and the difficulty of Identifying the location of things.

2.4.2-Routing

Routing process means selecting the best path between the source and the destination to complete the communication process successfully. There are various ways to determine the best path based on the communication protocol type such as a number of hops, costs, and bandwidth. Can be classified routing protocols into two main categories are:

- Reactive protocols: the path is established after transmission request is made,
- Proactive protocols: initial path before the request is made.

proposed the protocol under the name of "fault-tolerant routing protocol" for IoT. This protocol has designed by using learning automate (LA) and cross-layer concept. LA dealing with optimization problems to choose optimal solutions, the need to cross-layer concept. LA dealing with optimization problems to choose optimal solutions, the need to cross-layer is saving energy of the items of IoT (i.e., RFID).

2.4.3-Heterogeneity

The IoT environment is the best-known example to represent the heterogeneity issue because it contains a plethora of the different devices in their nature; the main objective of IoT is creating a common way to abstract the heterogeneity of these devices and achieving the optimal exploitation of their functionality. In this vein, the researchers always seek to find an effective method to deal with these devices regardless of their nature.

sought to introduce solutions to some of the IoT problems such as interconnection, heterogeneity, and generate an application that allows people to interconnect services over the Internet, these solutions are represented in: creating a domain specific language (DSL), graphic editor and IoT platform Midgar software.

2.4.4-Interoperability

Interoperability concept can be defined as the ability to create systems or devices cooperating with each other in an efficient way. Sought to use the semantic level interoperability architecture for pervasive the computing and IoT; the architecture is relied on the semantic information sharing solutions called “smart-M3”.

The principal idea of the proposed architecture relies on dividing IoT environment into small spaces to facilitate their management process.

2.4.5-Cloud Computing

Cloud computing is considered as a standard framework to represent IoT, and both IoT and cloud computing possess a set of benefits and restrictions. IoT represents real world and small things, but it is limited storage in addition to traditional problems in the network such as scalability and privacy.

The integration of cloud computing with IoT became a very important point of recent researches; to produce system able to overcome the many challenges such as scalability, storage resource and virtualization; the main objective of this integration is to leverage from cloud computing in processing power which need for sensors and other things.

2.4.6- Security and privacy

The security rule aims to protect it from threats; these threats classify into two kinds are: the external threats such as attacks on system form attackers and the internal threats represented in misuse of the system or information. There are three main factors of security are: data confidentiality, privacy and truth.

Privacy is defined as a control access to personal data; and it allows keeping certain information and data confidential; the features of privacy are secrecy, anonymity and solitude. In the IoT environment the security and the privacy are important to guarantee a reliable interaction between the physical world and the cyber world**.**

2.4.7- Quality of Service

QoS is defined as "the amount of time that is taken to deliver the message from the sender and the receiver" if this time is equal or less than pre-specified time requirement the QoS is achieved. ITU re-defined QoS concept as a degree of conformance of delivering service to the user by the provider with agreement between them. For QoS assurance, must cope with service models to determine which degree of QoS for each Internet service. **[9]**

2.5- Technologies

2.5.1- Radio Frequency Identification (RFID)

RFID is the key technology for making the objects uniquely identifiable. Its reduced size and cost make it integrable into any object. It is a transceiver microchip similar to an adhesive sticker which could be both active and passive, depending on the type of application.

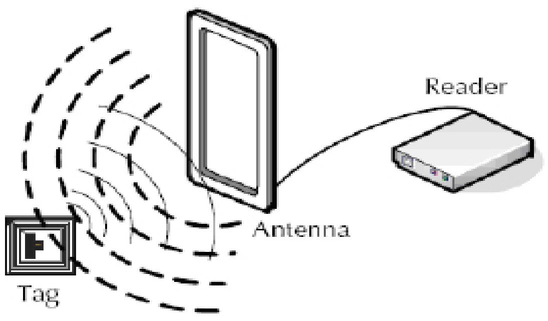


Figure 1.6. RFID Scenario.**[10]**

2.5.2- Wireless Sensor Network (WSN)

WSN is a bi-directional wirelessly connected network of sensors in a multi-hop fashion, built from several nodes scattered in a sensor field each connected to one or several sensors which can collect the object specific data such as temperature, humidity, speed etc and then pass on to the processing equipment.

A typical sensing node is shown in the figure below:

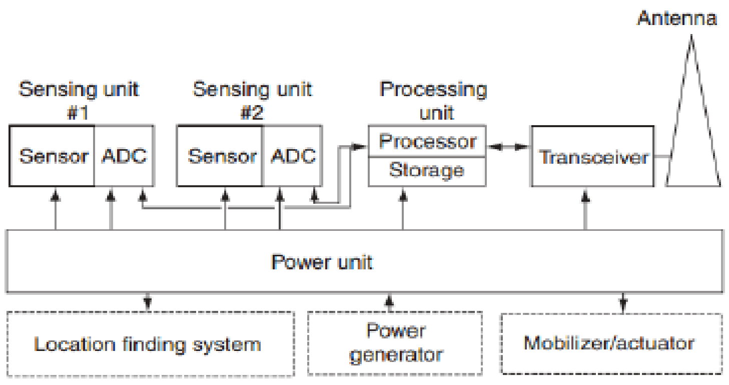


Figure1. 7. A typical sensing node.**[10]**

2.5.3- Networking Technologies

These technologies have an important role in the success of IoT since they are responsible for the connection between the objects, so we need a fast and an effective network to handle a large number of potential devices. For wide-range transmission network, we commonly use 3G, 4G etc., but as we know, mobile traffic is so much predictable since it only has to perform the usual tasks like making a call, sending a text message etc.

so, as we step into this modern era of ubiquitous computing, it will not be predictable anymore, which calls for a need of a super-fast, super-efficient fifth generation wireless system which could offer a lot more bandwidth.

Similarly, for a short-range communication network we use technologies like Bluetooth, WIFI etc.

2.5.4-Nano Technologies

Nano technologies are a cost-effective solution for improvising the communication system of IoT and other advantages like size reduction of sensors and actuators, integrated ubiquitous computing devices and higher range of frequencies etc. **[10]**

3-IoT Distribution Patterns

IoT distribution patterns classify as: centralized, collaborative, connected intranets, and distributed based on a layered architectural style.

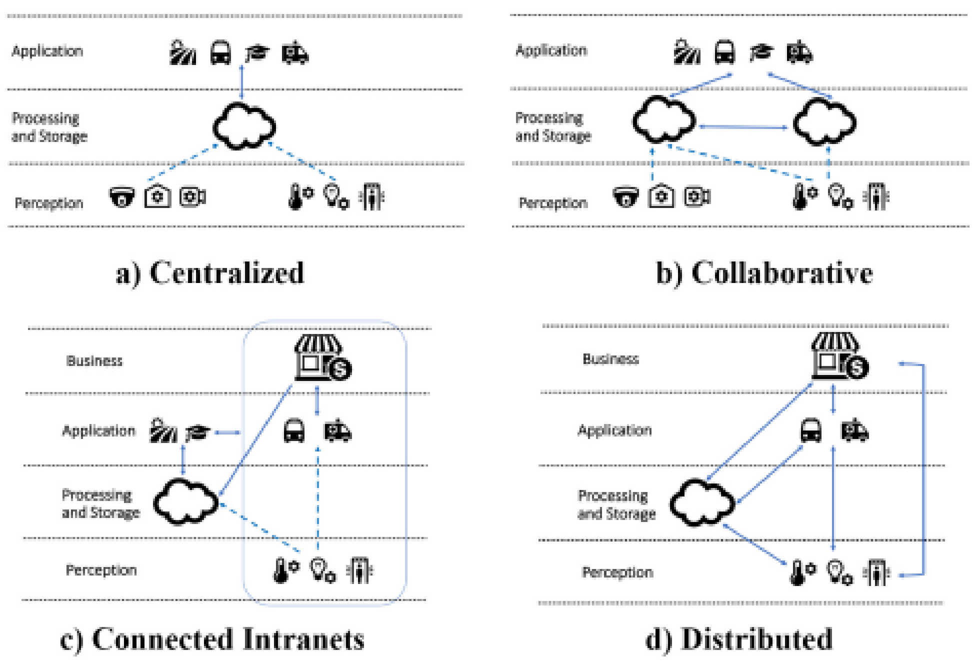


Figure1.8. IoT distribution patterns.**[11]**

- In a centralized distribution pattern, the perception layer provides data for the central processing and storage component to prepare for a service in the next layer. In order to use the IoT service, one must connect to this central component. The central component can be a server, cloud, or a fog network connected to cloud.

- In a collaborative pattern, a network of central intelligent components can communicate in order to form and empower their services.

- In a Connected Intranets pattern, sensors provide data within a local intranet to be used locally, remotely, and centrally. Here the advantage is that if the central component fails, local service is still in access. The disadvantage is that there is no fully distributed framework to facilitate the communication among components.

- In a distributed pattern, all components are fully interconnected and capable to retrieve, process, combine, and provide information and services to other components towards common goals.**[11]**

4-A QoS in IoT

QoS is one of the key factors in IoT applying. QoS helps to manage the system capabilities and its resources to provide IoT services. It enables the service providers to provide clearer visibility of their services and performance and usability of the services to the consumers' metrics will help customers to identify best IoT service for its application and will also deal with the optimization of service quality.

to satisfy the requirement of diverse applications of IoT, the need is to enhance the quality of the network services by adding value to it. The various quality metrics identified in a communication network like: Jitter, Bandwidth, Throughput and Efficiency, Monetary Cost, Availability, Security and Privacy.**[12]**

- The QoS of application and service layers is perceived directly by customers, therefore it becomes the most important standard for evaluating the QoS of IoT. (Service Time, Service Delay, Service Accuracy, Service Load, Service Priority)
- QoS indicators of network layer as well as its mechanism depend on the network type. Whether its mobile communication network, Internet or various private networks, they all have their own QoS systems. As transmitting network, it can be evaluated by the following main QoS indicators: bandwidth, delay, packet loss rate and jitter. However, the main problem is how to translate the upper layer's requirements to the QoS requirements and guarantee mechanism applied by the network.
- The QoS of perception layer guarantees the quality of monitoring, which is including sampling parameters, time synchronization, coverage and location/mobility.[13]

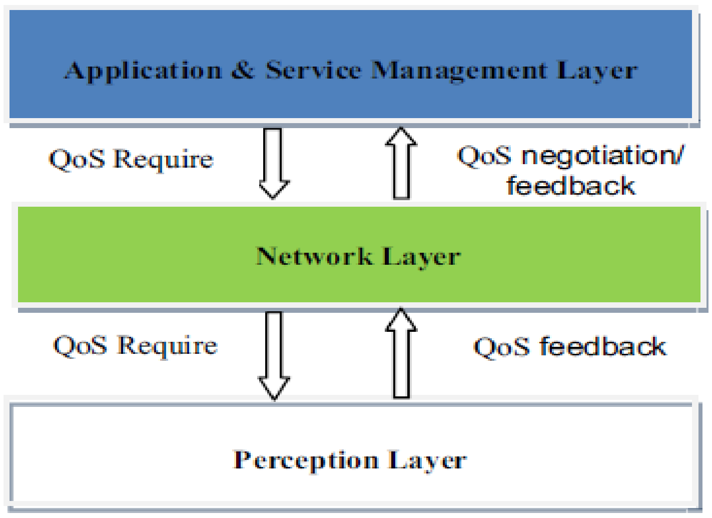

Application & Service Management Layer
QoS Require
QoS negotiation/ feedback
Network Layer
QoS Require
QoS feedback
Perception Layer

Figure1.9. Hierarchical QoS in IoT.**[13]**

5-Conclusion

In this chapter, our study was specifying on the IoT, its definition and fields application domain, then the architecture of IoT, we noted that there are two types of architecture the most used and it's the challenges, and the Technologies also we present QoS in IoT.

One of the major challenges is ensuring the QoS in IoT system, to ensure this by using self-adaptation which will present in the next chapter**.**

Chapter 02

Self-Adaptation

Chapter 02: Self-Adaptation

1-Introduction

The purpose of self-adaptation systems is to face changing environments independently. The main challenge is to adapt appropriately even though the range of potential environmental changes is unknown at the time of design. In this case, static coping mechanisms are likely not applicable or do not have the expected effect.

To deal with dynamic changes in the environment, systems should also be able to adapt their adaptive approach. However, it is difficult to detect and eliminate inapplicable or ineffective coping mechanisms, as well as to deduce new ones that may deal with the new situation upon execution.

Self-adaptation is an effective solution for complex systems that need to deal with the increasing complexity, uncertainty and dynamicity of these systems.; to control the adaptation process in a self-adaptive system, a common approach is to use an implementation of IBM's MAPE-K feedback loop, which autonomously Monitors and Analyses a managed system and plans and executes an adaptation plan if necessary. All parts of the MAPE-K loop share a common Knowledge base to exchange information.

2-Self Adaptation

Definition1.Self-adaptivity is the capability of the system to adjust its behavior in response to the environment. The ‘‘self’’ prefix indicates that the systems autonomously decide (i.e., with minimal or no interference) how to adapt or to organize them-selves so that they can accommodate changes in their contexts and environments. While some self-adaptive systems may be able to function without any human intervention, guidance in the form of higher-level objectives (e.g., through policies) is useful and realizable in many systems (Brun et al., 2009) **[14].**

Definition2. A self-adaptive system evaluates its own behavior and changes its own performance when the evaluation indicates that it is not accomplishing what the software is intended to do, or when better functionality or performance is possible (Salehie & Tahvildari, 2011).**[14]**

There two basic principles that complement one another and determine what is a self-adaptive system:

- ***External principle***: A self-adaptive system is a system that can handle changes and uncertainties in its environment, the system itself, and its goals autonomously (i.e., without or with minimal human interference).
- ***Internal principle***: A self-adaptive system comprises two distinct parts: the first part interacts with the environment and is responsible for the domain concerns (i.e., concerns for which the system is built); the second part interacts with the first part (and monitors its environment) and is responsible for the adaptation concerns (i.e., concerns about the domain concerns).**[15]**

2.1-Conceptual Model of a Self-adaptive System

The conceptual model introduces a basic vocabulary for the field of self-adaptation and serves as a guidance for organizing and focusing the knowledge of the field. The conceptual model comprises four basic elements: environment, managed system, adaptation goals, and managing system.**[15]**

- ***Environment*** The environment refers to the part of the external world with which the self-adaptive system interacts and in which the effects of the system will be observed and evaluated.
- ***The managed system*** comprises the application code that realizes the system's domain functionality. the concerns of the managed system are concerns over the domain, the managed system senses and effects the environment.
- ***The adaptation goals*** are concerns of the managing system over the managed system; they usually relate to the software qualities of the managed system.

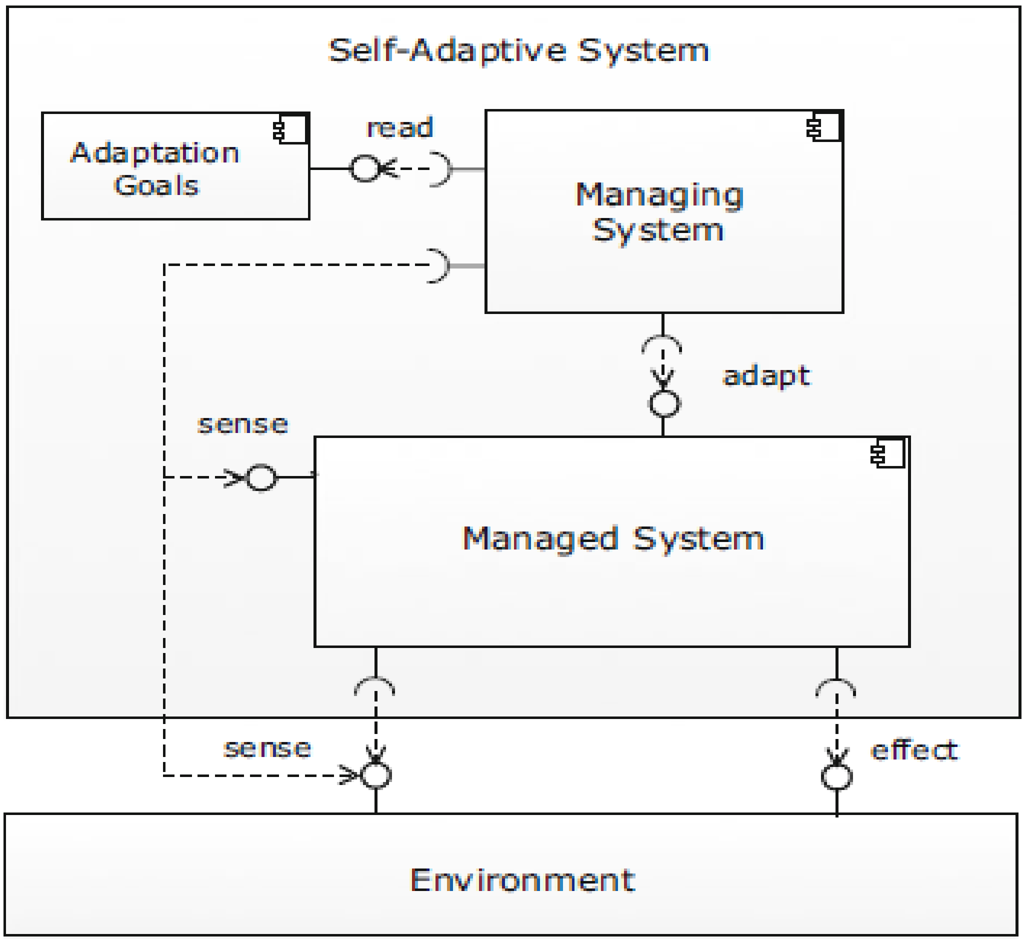


Figure 2.1. Conceptual model of a self-adaptive system.**[15]**

2.2 Waves of a Self-adaptive System

- ***Automation of management tasks:*** System management done by human operators is a complex and error-prone process. System manages itself autonomously based on high-level objectives
- ***Architecture based adaptation:*** Motivation for self-adaptation acknowledged, need for a principled engineering perspective Separation between change management (deal with change) and goal management (adaptation objectives)
- ***Runtime models:*** Architecture principles of self-adaptive systems understood; concrete realisation is complex Model-driven approach to realise self-adaptive systems Runtime models as key elements to engineer self-adaptive systems

- ***Goal driven adaptation:*** Design of feedback loops well understood, but requirement problem they intent to solve is implicit Requirements for feedback loops Languages and formalisms to specify requirements for self-adaptive systems
- ***Guarantee adaptation goals under uncertainty:*** Mature solutions for engineering self-adaptive systems, but uncertainty handled in ad hoc manner the role of uncertainty in self-adaptive systems and how to tame it Formal techniques to guarantee adaptation goals under uncertainty
- ***Control based adaptation:*** Engineering of MAPE-based self-adaption well understood, but solutions are often complex Applying principles from control theory to realise self-adaptation. Theoretical framework for (particular types of) self-adaptive systems.[15]

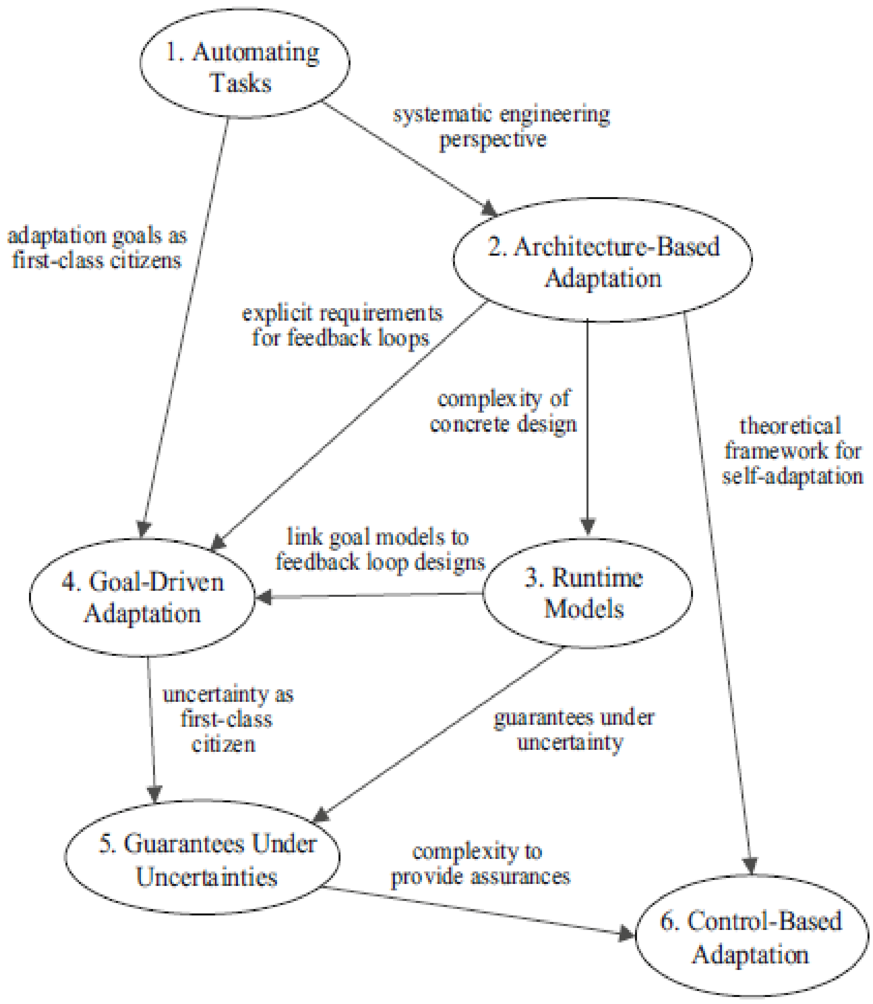

1. Automating Tasks
systematic engineering perspective
adaptation goals as first-class citizens
2. Architecture-Based Adaptation
explicit requirements for feedback loops
complexity of concrete design
theoretical framework for self-adaptation
link goal models to feedback loop designs
4. Goal-Driven Adaptation
3. Runtime Models
uncertainty as first-class citizen
guarantees under uncertainty
5. Guarantees Under Uncertainties
complexity to provide assurances
6. Control-Based Adaptation

Figure 2.2. Six waves of research in self-adaptive systems.**[15]**

3- The IBM Autonomic Framework

The IBM Autonomic Computing Initiative codified an external, feedback control approach in its Autonomic Monitor-Analyze-Plan-Execute (MAPE) Model. is either an entire system or a component within a larger system. The MAPE loop highlights four essential aspects of self-adaptation:[16]

- ***Monitor:*** The monitoring phase is concerned with extracting information properties or states out of the managed element. Mechanisms range from source-code instrumentation to non-intrusive communication interception.
- ***Analyze:*** is concerned with determining if something has gone awry in the system, usually because a system property exhibits a value outside of expected bounds, or has a degrading trend.
- ***Plan:*** is concerned with determining a course of action to adapt the managed element once a problem is detected.
- ***Execute***: is concerned with carrying out a chosen course of action and effecting the changes in the system.

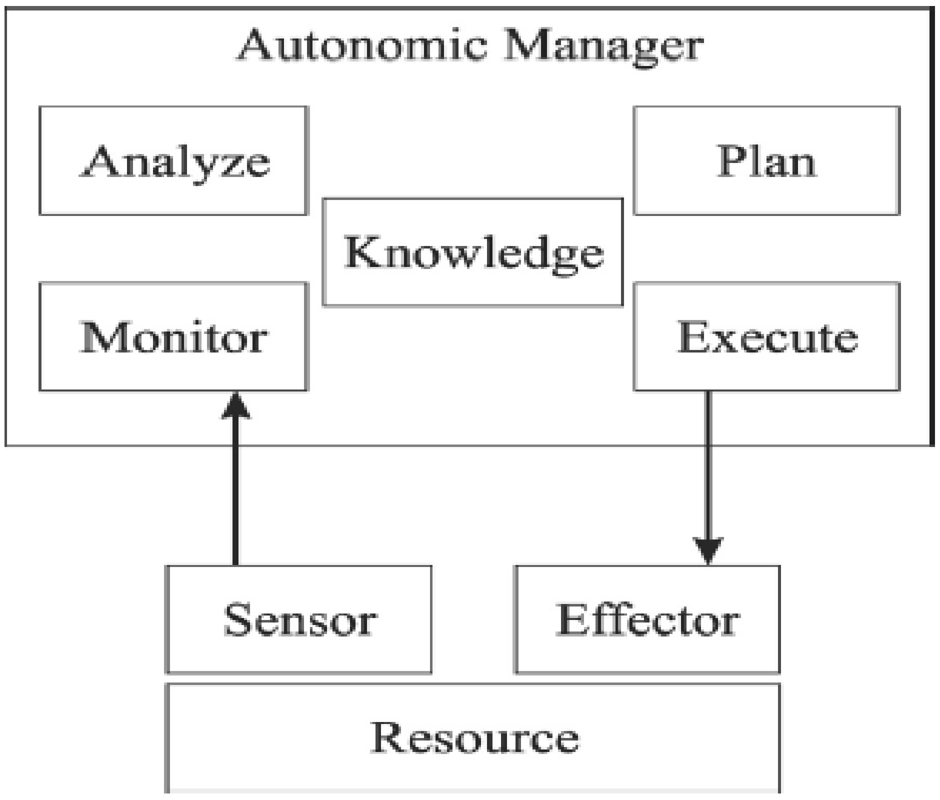


Figure 2.3. IBM autonomic computing architecture.**[17]**

4- Self-Adaptation Patterns

their six control patterns based on MAPE loop (Monitoring, Analysis, Planning, Execution) that model different types of interacting loops with different level of decentralization.

In the figure, managed subsystems (MS) manage the managed subsystems, and comprise the adaptation logic.

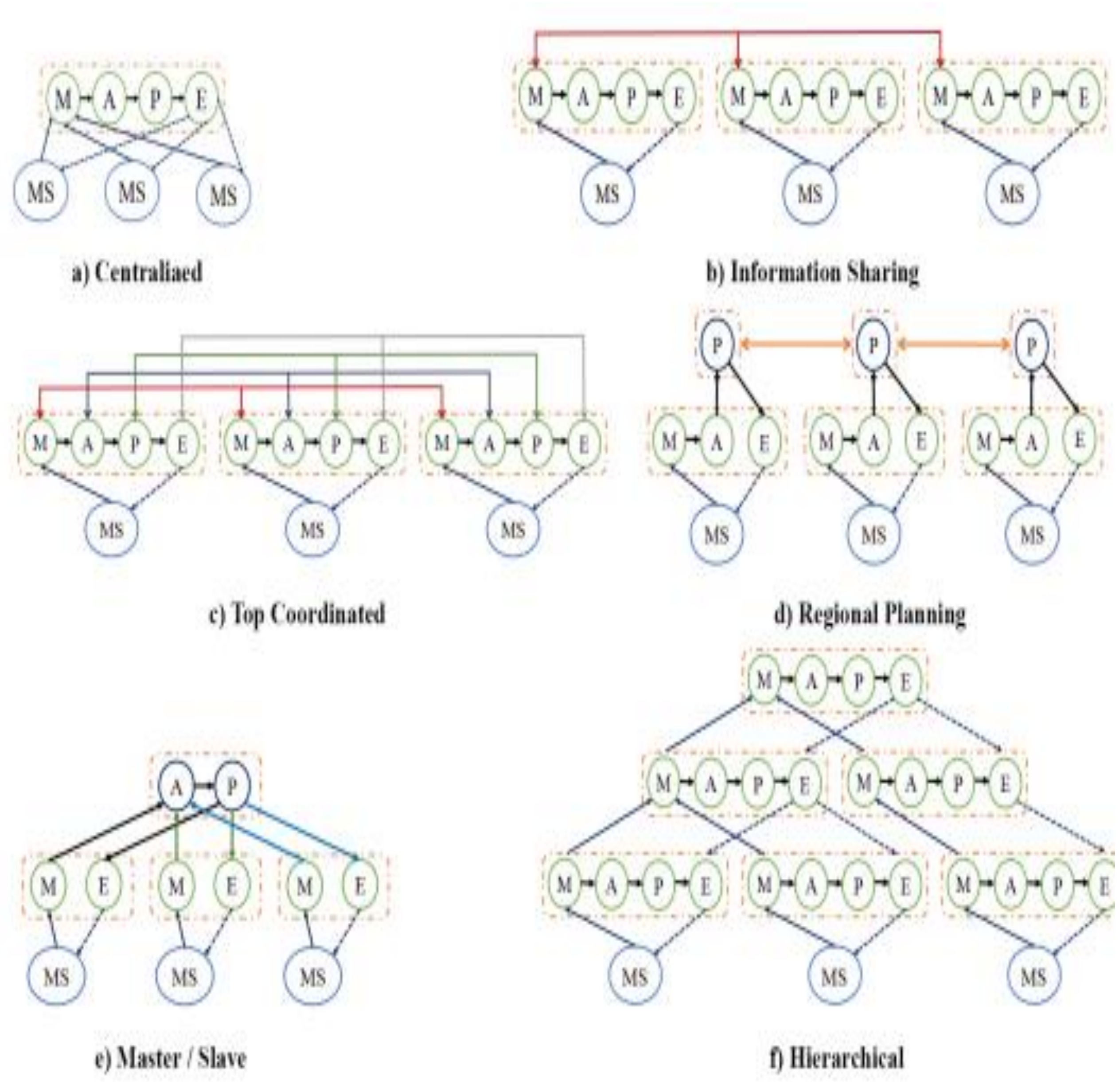


Figure 2.4. Self-adaptation patterns.**[11]**

- ***A centralized self-adaptation pattern*** performs the adaptation through a central control loop.

- ***fully decentralized approach:***
  - In the coordinated pattern, all MAPE components coordinate with their corresponding peers.
  - In information sharing, only M components communicate with one another.
- ***hierarchical distribution model:***
  - In master/slave pattern, a hierarchical relationship between one centralized master component (A and P) and multiple slave components (M and E) is created.
  - In Regional planning self-adaptation pattern, a physical space can be divided into different regions and the regions local planners coordinate to find the best adaptation solution for a local or global problem. Regional planning provides one P for each region to supervise the other elements of loop, in a way to interact different regions P one to another.
  - In The hierarchical control pattern provides a layered separation of concerns to manage the complexity of self-adaptation as a hierarchy of MAPE loops.**[11]**

5- Uncertainties

A key aspect of self-adaptation is being aware and dealing with uncertainties in a system.

Three criteria are used to classify different type of uncertainties:

1. ***The first criteria location***:

reasons about the part of the model where the uncertainty is located. They make a distinction between context uncertainty, model structural uncertainty and input parameters uncertainty. Context uncertainty concerns itself with uncertainty about information of the system (e.g.

information that was not included inside the model). Model structural uncertainty is concerned with the correctness of the model itself, in the sense of how accurately it models reality. Input parameters uncertainty describes the uncertainty of the values of the parameters used to evaluate the model.

2. ***The second criteria level***:

Five levels/orders are defined: from order 0 where there is no uncertainty up until order 4 where there is uncertainty about the order of uncertainty. In between we have orders that reason about being aware of uncertainties, lacking awareness of uncertainties and lacking process of becoming aware of uncertainties.

3. ***The final criteria nature***:

is divided into two categories: Epistemic and Aleatory. Epistemic reasons about uncertainty being introduced as a result of lack/imperfection/... of the data. Aleatory are types of uncertainty introduced due to the nature of some processes where for example randomness in the process plays a part. .**[18]**

6- Conclusion

In this chapter, our study was specifying on the self-adaptation, its definition and conceptual model of self-adaptation and its six waves, we also discussed IBM autonomic Framework using predicted information about state of the system keep off the unnecessary adaptation.

To predicted information about the IoT system we need to use some Deep Learning Model, that we will explain in the next chapter.

Chapter 03

Deep learning

Chapter 03: Deep learning

1-Introduction

Over the years, Machine Learning (ML) techniques have been applied to reduce errors and simultaneously increase the speed of data processing and lead to Internet of Things (IoT) devices. Machine learning is used to teach machines to manage data in a more effective way.

Sometimes, once we have accessed the data, we cannot interpret the model or extract information from the data. In this instance, we apply machine learning. With so many data set available, the demand for machine learning is increasing. The aim of machine learning is to learn from the Data. Numerous studies have been conducted on how to make machines learn on their own. In broad terms, learning is the process of turning experience into expertise or knowledge. Deep learning is a form of machine learning that enables computers to learn from experience and understand the world in terms of a hierarchy of concepts. Because the computer gathers knowledge from experience, there is no need for a human computer operator formally to specify all of the knowledge needed by the computer. In our work, we will use deep learning model. Using a portion of the data gathered, we train the model at first. After training the model, the relevant features of the remaining data, i.e., features of the data that is not used for learning, are used to make predictions about these configurations. The accuracy of the machine learner here is the number of right predictions relative to the total number of predictions made. In this chapter, we will outline the foundation of these concepts.

2- Machine Learning

Definition 1. ML is a method of data analysis dealing with the construction and evaluation of algorithms. It is the science that gives computers and computing machines the ability to act without being explicitly being programmed. It is defined by the ability to choose effective features for pattern recognition, classification, and prediction based on the models derived from existing data.**[19]**

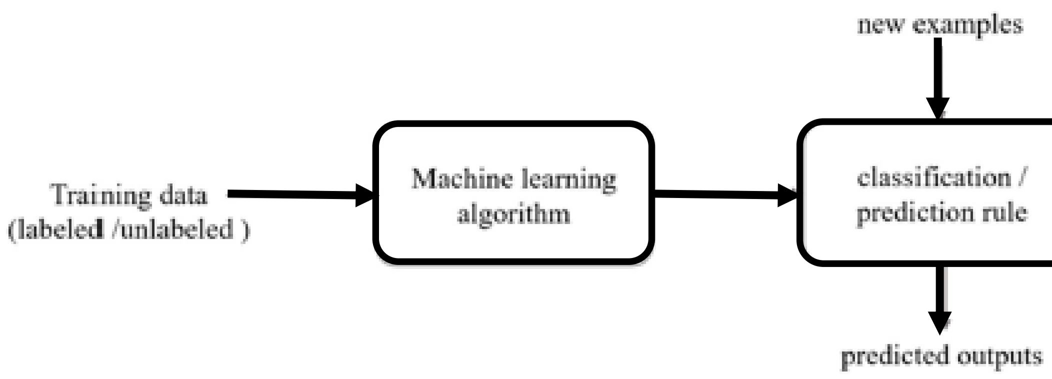


Figure 3.1. A typical machine learning approach.**[20]**

2.1- Machine Learning Classification

One may classify machine learning systems along many different dimensions. We have chosen three dimensions:

- Classification on the basis of the underlying learning strategies used. The processes themselves are ordered by the amount of inference the learning system performs on the available information.
- Classification on the basis of the representation of knowledge or skill acquired by the learner.
- Classification in terms of the application domain of the performance system for which knowledge is acquired.**[21]**

2.2- Types of Learning

2.2.1-Supervised Learning

The supervised machine learning algorithms are those algorithms which needs external assistance. The input dataset is divided into train and test dataset. The train dataset has output variable which needs to be predicted or classified. All algorithms learn some kind of patterns from the training dataset and apply them to the test dataset for prediction or classification.**[22]**

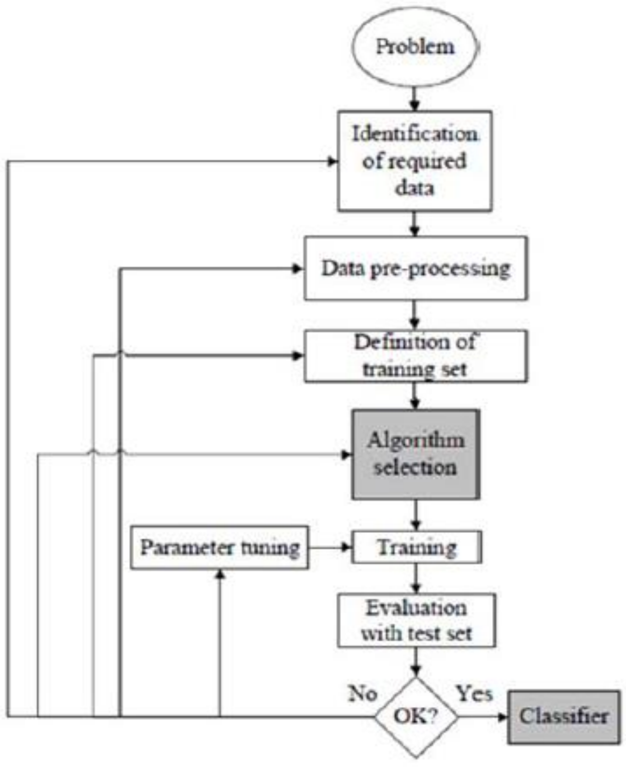


Figure 3.2. Workflow of supervised machine learning algorithm.**[22]**

The application of supervised machine learning is generally split into two classes:

- ***Regression*:** technique predicts a single output value using training data.
- ***Classification:*** means to group the output inside a class. **[22]**

The most famous supervised machine learning algorithms:

- ***Decision Tree***
- ***Naïve Bayes***
- ***Support Vector Regression*** [22]

2.2.2-Unsupervised Learning

The unsupervised learning algorithms learns few features from the data. When new data is introduced, it uses the previously learned features to recognize the class of the data. It is mainly used for clustering and feature reduction. **[22]**

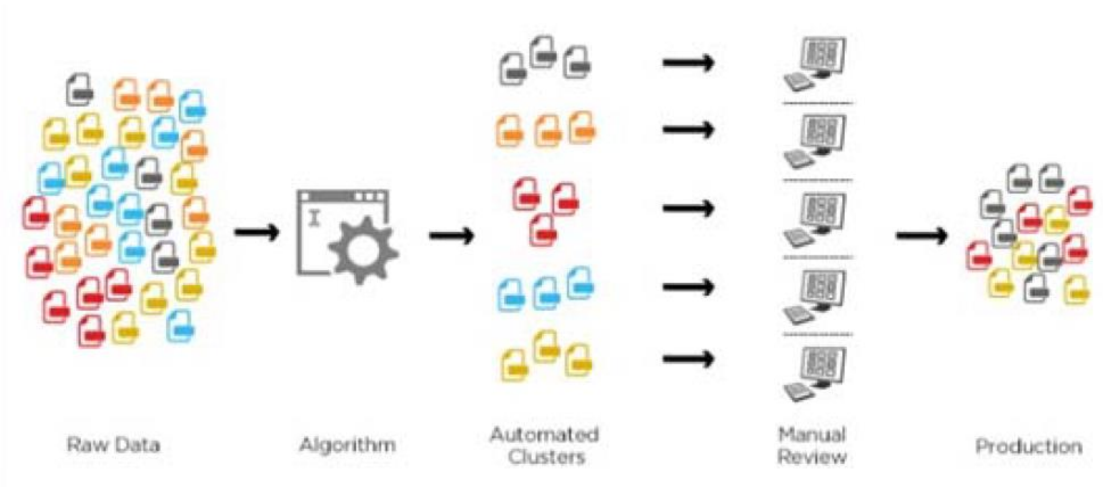


Figure 3.3. Workflow of unsupervised machine learning algorithm.[22]

The application of unsupervised machine learning is generally split into two classes:

- ***Clustering***: is an important concept when it comes to unsupervised learning. It mainly deals with finding a structure or pattern in a collection of uncategorized data.
- ***Association***: Association rules allow you to establish associations amongst data objects inside large databases.

The most famous unsupervised machine learning algorithms:

- ***K-Means Clustering***
- ***Principal Component Analysis***

2.2.3- Semi - Supervised Learning

Semi – supervised learning algorithms is a technique which combines the power of both supervised and unsupervised learning. It can be fruit-full in those areas of machine learning and data mining where the unlabeled data is already present and getting the labeled data is a tedious process. There are many categories of semi-supervised learning.

The most famous semi-supervised machine learning algorithms:

- ***Generative Models***
- ***Self-Training***
- ***Transductive SVM*** [22]

2.2.4- Reinforcement Learning

Reinforcement learning is a type of learning which makes decisions based on which actions to take such that the outcome is more positive. The learner has no knowledge which actions to take until it's been given a situation. The action which is taken by the learner may affect situations and their actions in the future. Reinforcement learning solely depends on two criteria: trial and error search and delayed outcome.**[22]**

Reinforcement Learning is not simple, and is tackled by a plethora of different algorithms. As a matter of fact, in Reinforcement Learning an agent decides the best action based on the current state of the results.**[26]**

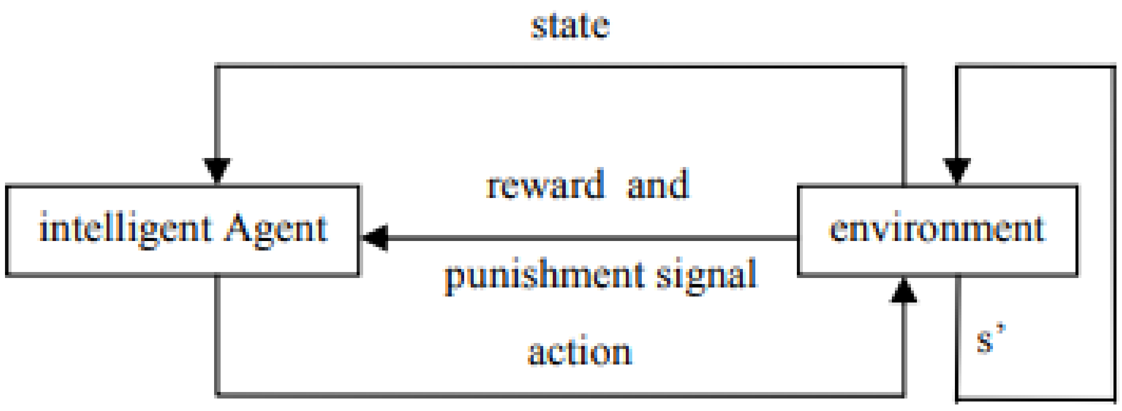


Figure 3.4. The basic model of RL.**[26]**

The most famous Reinforcement learning algorithms:

- ***Q-Learning***
- ***Monte Carlo Tree Search***
- ***Temporal Difference, or TD***
- ***Asynchronous Actor-Critic Agents (AAAC)*** [27]

3-Deep Leaning

Definition 1. The deep learning is a subset of the machine learning and based on the algorithms that are stimulated by the functioning of the brain and the way they are structured. The deep learning educates the computer system to produce results by training them on the previously available examples. The capability of the deep learning has enabled it to accomplish results that were impossible once upon a time. The remain as the pivotal mechanics behind most of the applications that are found nowadays such as the self-driving vehicles, voice control in consumer electronics such as the phone, Television, smart machines etc. **[28]**

Definition 2. A class of machine learning techniques that exploit many layers of non-linear information processing for supervised or unsupervised feature extraction and transformation, and for pattern analysis and classification.**[39]**

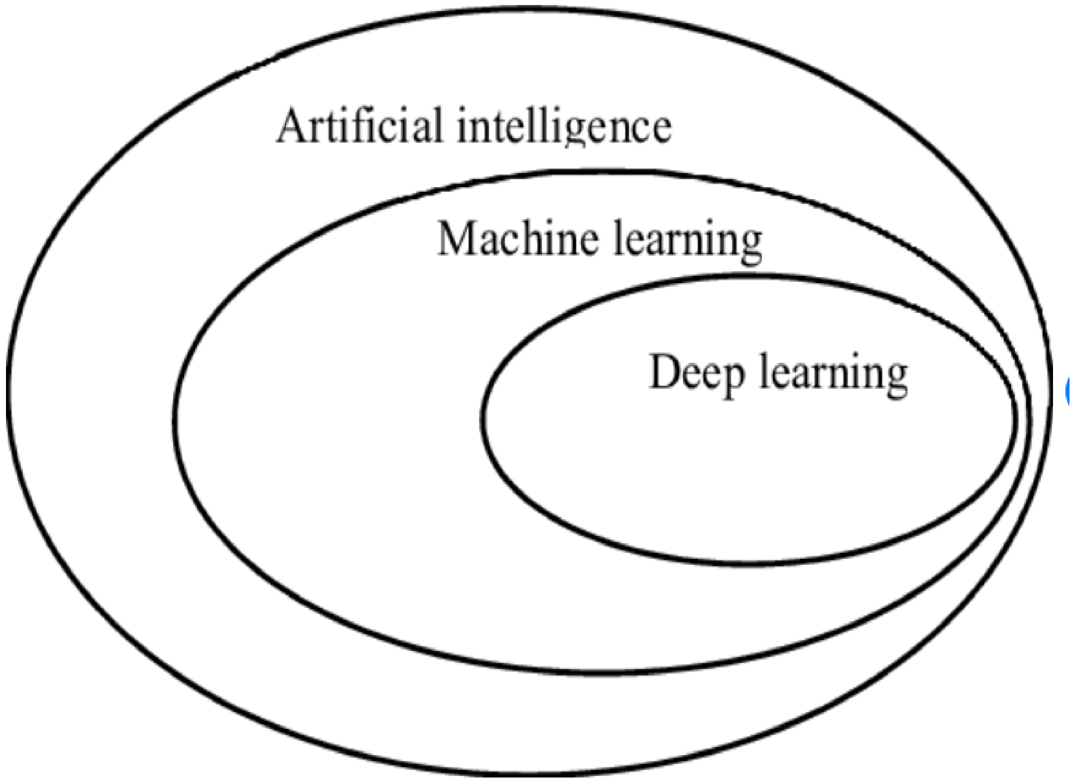


Figure 3.5. A Venn-diagram of artificial intelligence: link between artificial intelligence, machine learning and deep learnin***g***.**[29]**

3.1-Datasets for Deep Learning

- ***Training Dataset***: The sample of data used to fit the model.
  The actual dataset that we use to train the model (weights and biases in the case of a Neural Network). The model sees and learns from this data.
- ***Validation Dataset***: The sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyperparameters.
  The validation set is used to evaluate a given model, but this is for frequent evaluation. This data use to fine-tune the model hyperparameters.
  The validation set is also known as the Development set. This makes sense since this dataset helps during the "development" stage of the model.
- ***Test Dataset***: The sample of data used to provide an unbiased evaluation of a final model fit on the training dataset.
  The Test dataset provides the gold standard used to evaluate the model. It is only used once a model is completely trained (using the train and validation sets).**[40]**

Dividing the data set into two sets is a good idea, but not quite reasonable. You can greatly reduce your chances of overfitting by partitioning the data set into the three subsets. Use the validation set to evaluate results from the training set. Then, use the test set to double-check your evaluation after the model has "passed" the validation set. The following figure shows this workflow:

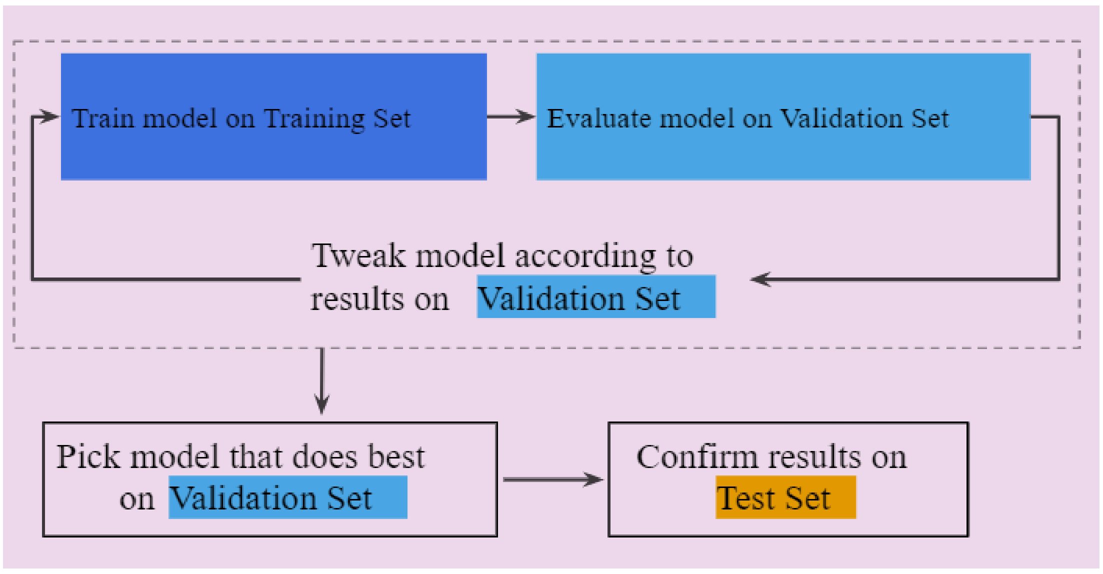

Train model on Training Set
Evaluate model on Validation Set
Tweak model according to results on Validation Set
Pick model that does best on Validation Set
Confirm results on Test Set

Figure 3.6. workflow of Datasets for Deep Learning.**[40]**

4-Neural Network

Definition.1. Neural networks, which provide the basis for deep learning, are a class of machine learning methods that are being applied to a diverse array of fields in business, health, technology, and research. Neural networks are computing systems similar to the biological neural networks in animals. **[30]**

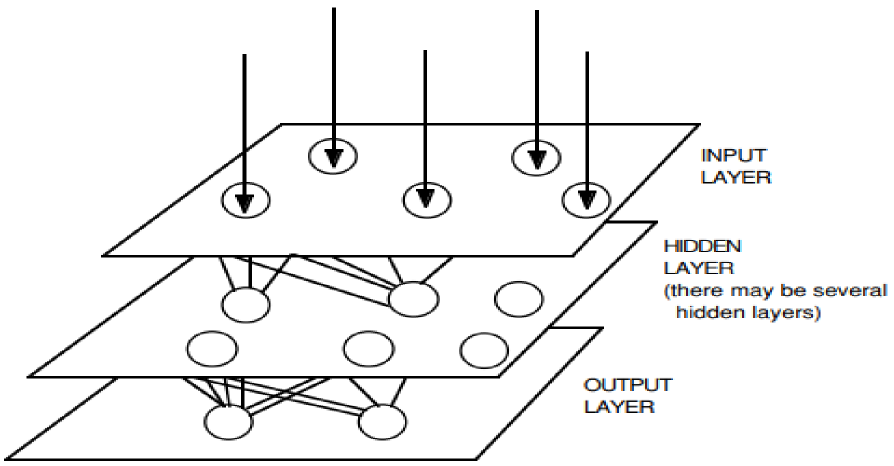


Figure 3.7. Neural Network Diagram.[31]

is derived from the biological concept of neurons. A neuron is a cell like structure in a brain. To understand neural network, one must understand how a neuron works. A neuron has mainly four parts. They are dendrites, nucleus, soma and axon.**[22]**

An artificial neural network behaves the same way. It works on three layers. The input layer takes input (much like dendrites). The hidden layer processes the input (like soma and axon). Finally, the output layer sends the calculated output (like dendrite terminals).

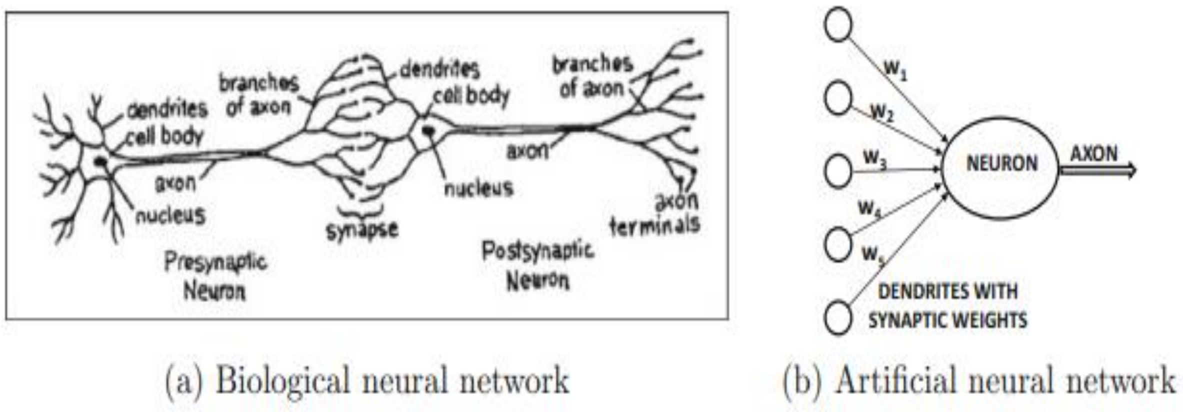


Figure 3.8. link between Biological neural network(a) and Artificial Neural Network(b).**[32]**

There are basically three types of artificial neural network **[22]**:

- ***Supervised Neural Network:*** the output of the input is already known. The predicted output of the neural network is compared with the actual output. Based on the error, the parameters are changed, and then fed into the neural network again. will summarize the process. Supervised neural network is used in feed forward neural network.
- ***Unsupervised Neural Network:*** Here, the neural network has no prior clue about the output the input. The main job of the network is to categorize the data according to some similarities. The neural network checks the correlation between various inputs and groups them.
- ***Reinforced Neural Network*****:** In reinforced neural network, the network behaves as if a human communicates with the environment. From the environment, a feedback has been provided to the network acknowledging the fact that whether the decision taken by the network is right or wrong. If the decision is right, the connections which points to that particular output is strengthened. The connections are weakened otherwise. The network has no previous information about the output.

4.1- Neural Network Types

Some of Neural Network Type:

4.1.1- Deep Neural Networks (DNN)

Deep ANNs are most widely referred to as deep learning (DL) or deep neural networks (DNNs). They are a relatively new area of ML research allowing computational models that are composed of multiple processing layers to learn complex data representations using multiple levels of abstraction. One of the main advantages of DL is that in some cases, the step of feature extraction is performed by the model itself. DL models have dramatically improved the state-of-the-art in many different sectors and industries, including agriculture. DNN's are simply an ANN with multiple hidden layers between the input and output layers and can be either supervised, partially supervised, or even unsupervised. **[41]**

4.1.2-Convolutional Neural Networks (CNN)

are one of the most popular models used today. This neural network computational model uses a variation of multilayer perceptron's and contains one or more convolutional layers that can be either entirely connected or pooled. These convolutional layers create feature maps that record a region of image which is ultimately broken into rectangles and sent out for nonlinear processing. CNN's were first developed and used around the 1980s. **[42]**

4.1.3- Recurrent Neural Networks (RNN)

Recurrent Neural Network (RNN) are a type of Neural Network where the output from previous step is fed as input to the current step. In traditional neural networks, all the inputs and outputs are independent of each other, but in cases like when it is required to predict the next word of a sentence, the previous words are required and hence there is a need to remember the previous words. Thus, RNN came into existence, which solved this issue with the help of a Hidden Layer. The main and most important feature of RNN is Hidden state, which remembers some information about a sequence.**[43]**

4.1.4- Long Short-Term Memory (LSTM)

LSTM networks are an extension of recurrent neural networks (RNNs) mainly introduced to handle situations where RNNs fail. Nevertheless, there are drawbacks to RNNs. First, it fails to store information for a longer period of time. Thus, Long Short-Term Memory (LSTM) was brought into the picture. It has been so designed that the vanishing gradient problem is almost completely removed, while the training model is left unaltered. Long-time lags in certain problems are bridged using LSTMs where they also handle noise, distributed representations, and continuous values.[**44]**

4.1.5- Probabilistic Neural Network (PNN)

PNN was first proposed by Specht in 1990. It is a classifier that maps input patterns in a number of class levels. It can be forced into a more general function approximation. This network is organized into a multilayer feed-forward network with input layer, pattern layer, summation layer, and the output layer. PNN is an implementation of a statistical algorithm called kernel discriminant analysis**.[45]**

4.2- Neural network applications

Basically, most applications of neural networks fall into the following five categories **[31]**:

- ❖ ***Prediction:*** Use input values to predict some output (e.g., pick the best stocks in the stock market, predict the weather, identify people with cancer risks).
- ❖ ***Classification:*** Use input values to determine the classification (e.g., is the input the letter A, is the blob of video data a plane and what kind of plane is it)

- ***data association***: Like classification but it also recognizes data that contains errors (e.g., not only identify the characters that were scanned but also identify when the scanner wasn't working properly)
- ***data conceptualization*** Analyze the inputs so that grouping relationships can be inferred (e.g., extract from a data base the names of those most likely to buy a particular product)
- ***data filtering*** Smooth an input signal (e.g., take the noise out of a telephone signal)

4.3- Neural network Components

- ***Hyperparameters***: are the variables which determines the network structure and the variables which determine how the network is trained. Hyperparameters are set before training (before optimizing the weights and bias).**[33]**

  These are various Hyperparameters:

  - Number of Hidden Layers and units.
  - Dropout.
  - Network Weight Initialization.
  - Activation function.
- ***Neurons:*** are the basic units of a neural network. In an ANN, each neuron in a layer and is connected to each neuron in the next layer. When the inputs are transmitted between neurons, the weights are applied to the inputs along with the bias.
- ***Weights:*** control the signal (or the strength of the connection) between two neurons. In other words, a weight decides how much influence the input will have on the output.
- ***Biases:*** which are constant, are an additional input into the next layer that will always have the value of 1. Bias units are not influenced by the previous layer (they do not have any incoming connections) but they do have outgoing connections with their own weights. The bias unit guarantees that even when all the inputs are zeros there will still be an activation in the neuron.

$$Y = \sum (weight * input) + bias$$

- ***Activation Function:*** In a neural network, an activation function normalizes the input and produces an output which is then passed forward into the subsequent layer.

Activation functions add non-linearity to the output which enables neural networks to solve non-linear problems. In other words, a neural network without an activation function is essentially just a linear regression model. **[34]**

- ***Layers:*** Each layer contains a number of Neurons. The amount in most cases is entirely up to the creator. However, having too many layers for a simple task can unnecessarily increase its complexity and in most cases decrease its accuracy. The opposite also holds true.

  The Neural Network is constructed from 3 type of layers:

  - Input layer: initial data for the neural network.
  - Hidden layers: an intermediate layer between input and output layer and place where all the computation is done.
  - Output layer: produce the result for given inputs.**[35]**

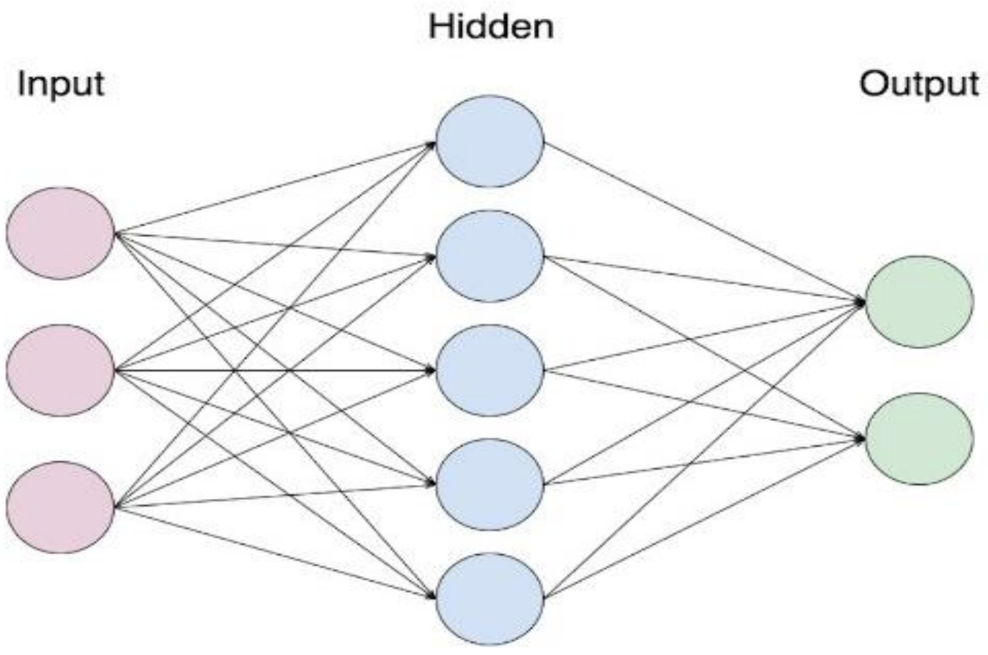


Figure 3.7. layers input /Hidden/output.**[36]**

5-Conclusion

In this chapter, we discussed machine learning techniques and the various types of machine learning techniques. Also, we talked about deep learning and Datasets for Deep Learning, also Neural Network and application and there types also Components. In the next chapter we will present the Design of our approach.

Chapter 04

Design

Chapter 04: Design

1-Introduction

IoT environments presents challenges in finding the right network settings. Which is hard as IoT applications are subject to a variety of uncertainties (e.g., Sudden changes in traffic load, communication interference...), for solving this problem of configuring the IoT network, A lot of research has been done in the field of adaptive systems. The goal of our approach is to implement adaptive IoT applications by implementing solutions based on adaptive architectures for specific cases.

The purpose of this chapter is to outline the overall architecture and description of the system components, as well as describe the general functioning of the system.

2-Objective

The aim of this work is to develop an efficient model based on machine learning techniques and deep learning technique for IoT (Internet of Things) system that allows system to adapt itself to ensure the maintenance of its quality objectives.

The system developed should dynamically reconfigure the IoT system to deviations from the predicted quality of service parameters (QoS).

3-Related Work

A number of recent efforts have explored the application of self-adaptation in IoT System.

D. Weyns et al**.** **[37]** present MARTAS approach by Applying Architecture-Based Adaptation to Automate the Management of Internet-of-Things. H. Muccini et al. **[11]** provide a literature based knowledge to learn and evaluate IoT distribution patterns, and self-adaptation control patterns. Robbe Berrevoets and Danny Weyns **[38]** investigated A QoS-aware Adaptive Mobility Handling Approach for LoRa-based IoT Systems.D. Weyns, S. Michiels **[18]** present hwo to apply Deep Learning to Reduce Large Adaptation Spaces of Self-Adaptive Systems with Multiple Types of Goals.This approach enhances the traditional MAPE-K feedback loop with a learning module that selects subsets of adaptation options from a large adaptation space to support the analyzer with performing efficient analysis.

From the related works reported above, our approach share some similar point with Applying Deep Learning to Reduce Large Adaptation Spaces of Self-Adaptive Systems with Multiple Types of Goals .we use the same datasets .Also by creating a Model ,Using a portion of the data colleting, we train the model at first.After training the model, the relevant features of the remaining data are used to make predictions approximately those configurations. The accuracy of the machine learner right here is the amount of proper predictions relative to the entire quantity of predictions made.

Table 4.1. Related work comparison

	Approach	**Technique**	**MAPE-K loop**
D.Weyns &al	Applying Architecture-Based Adaptation to Automate the Management of Internet-of-Things(MARTAS)	Statistical model cheking with stochastic automate, DeltaIoT	Yes
H.Muccini &al	provide a literature based knowledge to learn and evaluate IoT distribution patterns, and self-adaptation control patterns.	CAPS	Yes
R.Berrevoets & D.Weyns	investigated A QoS-aware Adaptive Mobility Handling Approach for LoRa-based IoT Systems	Adaptation Algorithme, DeltaIoT	No

K.Shah& M.Kumar	present Distributed Independent Reinforcement Learning (DIRL) Approach to Resource Management in Wireless Sensor Networks	Q-Learning	No
K.Vaidhyanathan &H.Muccini	apply A Machine Learning-driven Approach for Proactive Decision Making in Adaptive Architectures	LSTM, , Q-Learning, CAPS	No
D. Weyns, S. Michiels	apply Deep Learning to Reduce Large Adaptation Spaces of Self-Adaptive Systems with Multiple Types of Goals (DLASeR)	Deep Learning	yes
Our Approch	Deep learning training model .After training the model, the relevant features of the remaining data are used to make predictions approximately those configurations.	Deep Learning	no

3- The Approach

The bottom layer consists of the managed system with the network of motes and the gateway. The middle layer comprises a network engine, or we can present as client that uses the IoT network and offers an interface to it via a probe and an effector. The probe can be used to monitor the IoT network (status of motes and links, data about the packet loss, the energy consumption, etc.). And the effector adapts the mote settings (power settings of the motes, distribution of packets sent to parents, etc.). As for the top layer is added to the system that automatically adapts the adaptation goals of the IoT network.

1. The monitor tracks uncertainties (such as Network interference and noise and the changes of packets produced by the motes) and relevant properties of the managed system and the environment in which the system is deployment.

2. The collected data is used to update the models.

3. The monitor triggers the analyzer.

4. The analyzer reads the adaptation goals and also the updated models from the knowledge, as well it estimates the expected deviation of the adaptation goal.

- After The Data is set, we training the datasets in the Deep Learning Model (mean the adaptation for each mote parameters).
- The model tries to optimize the value of each parameter to avoid degradation of service quality.
- The values change to the new values after training datasets in the DL models.
- The new data represents the state of the motes after adaptation.
- Calculate the accuracy percentage of each considered QoS (latency, power consumption, packet loss).

5. the analyzer triggers the planner.

6. The planner generates a plan for the adaptation.

7. The planner triggers the executor.

8. The executor executes the plan.

a general overview of the approach to realizing the adaptive IoT application.

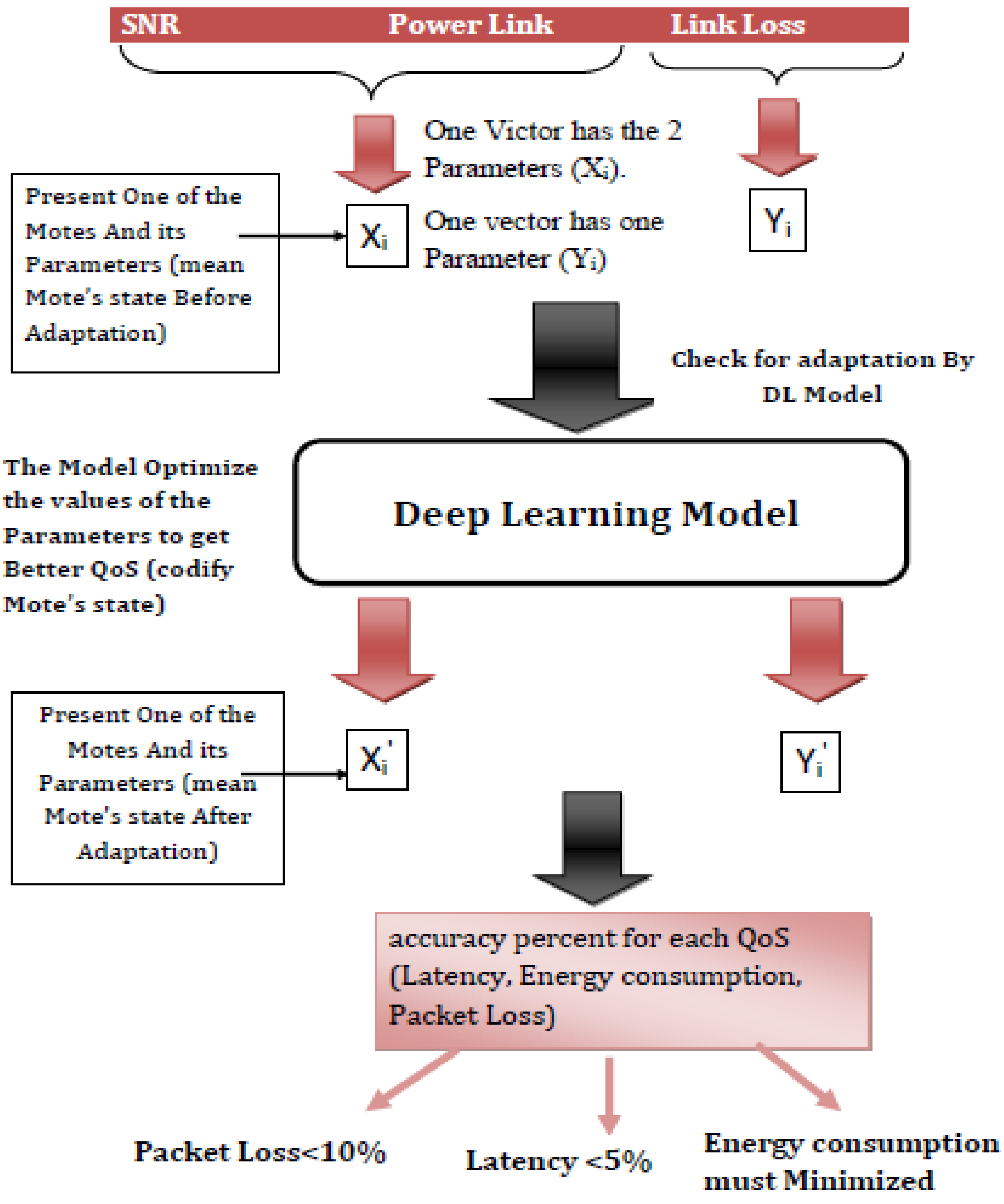

SNR
Power Link
Link Loss
One Victor has the 2 Parameters (Xi).
One vector has one Parameter (Yi)
Present One of the Motes And its Parameters (mean Mote's state Before Adaptation)
Xi
Yi
Check for adaptation By DL Model
The Model Optimize the values of the Parameters to get Better QoS (codify Mote's state)
Deep Learning Model
Present One of the Motes And its Parameters (mean Mote's state After Adaptation)
Xi'
Yi'
accuracy percent for each QoS (Latency, Energy consumption, Packet Loss)
Packet Loss<10%
Latency <5%
Energy consumption must Minimized

Figure 4.1. General overview of the approach

4-Detailed description for the approach

4.1- Support Vector Regression (SVR)

Support Vector Machine can also be used as a regression method, maintaining all the main features that characterize the algorithm (maximal margin). The Support Vector Regression (SVR) uses the same principles as the SVM for classification, with only a few minor differences. First of all, because output is a real number it becomes very difficult to predict the information at hand, which has infinite possibilities. In the case of regression, a margin of tolerance (epsilon) is set in approximation to the SVM which would have already requested from the problem. But besides this fact, there is also a more complicated reason, the algorithm is more complicated therefore to be taken in consideration. However, the main idea is always the same: to minimize error, individualizing the hyperplane which maximizes the margin, keeping in mind that part of the error is tolerated. [58]

In most linear regression models, the objective is to minimize the sum of squared errors. Take Ordinary Least Squares (OLS) for example. The objective function for OLS with one predictor (feature) is as follows:

$$MIN \sum_{i=1}^{n} (y_i - w_i x_i)^2$$

where y_i is the target, w_i is the coefficient, and x_i is the predictor.

In contrast to OLS, the objective function of SVR is to minimize the coefficients more specifically, the l2-norm of the coefficient vector not the squared error. The error term is instead handled in the constraints, where we set the absolute error less than or equal to a specified margin, called the maximum error, ϵ (epsilon). We can tune epsilon to gain the desired accuracy of our model. Our new objective function and constraints are as follows:

Minimize:

$$MIN \frac{1}{2} ||\mathbf{w}||^2$$

Constraints:

$$|y_i - w_i x_i| \leq \varepsilon$$

Example:

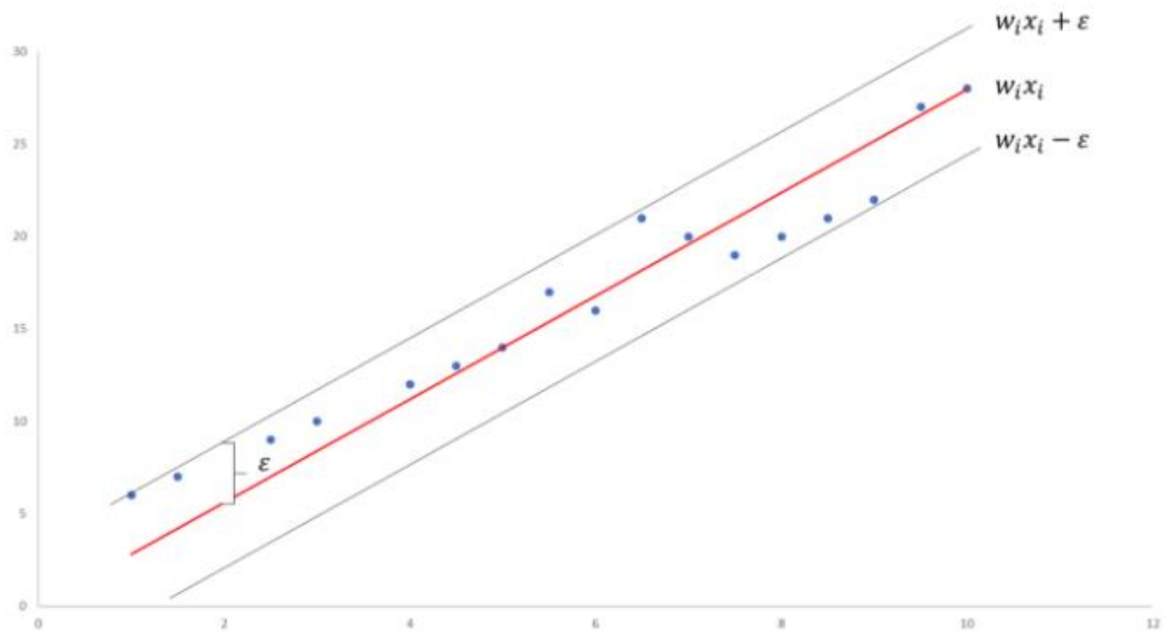


Figure 4.2. Example of Simple SVR.[58]

4.2- Used Datasets

The Data-set that used for training the model was collected from motes data generated by the Simulation process. One run in simulation means one day of process (period of 24 h). So, we collected the data from 1000 simulations (1000 days) to be trained and tested by our model. Take 70% of dataset for training and 30% of them to test.

4.3-MAKE-Loop

The MAPE-K loop is the system that automatically adapts the adaptation goals of the IoT network. The process starts with the monitor which uses the probe to track the traffic load and network interference's as well as everything related with the managed system and quality attributes. This data is used to update a set of models in the knowledge repository, including a model of the IoT system with the relevant aspects of the environment, and a set of quality models.

The IoT system model contains (status of motes and links, data about the packet loss, the energy consumption, latency of the network, etc.)

Sub-Component:

The Monitor Collects the data and settings from IoT network and updates the

knowledge repository.

The Analyzer predicts the expected deviation in adaptation goals by using the

prediction components.

The Planer Selects the adaptation action to achieve quality goals and objectives of

the network.

The Executor executes the adaptation actions to adapt the IoT system.

5- Functional Overview of the system

The process begins with the mote preparation for sending the packets that received from children or the locally generated packets. The MAPE-K Loop will monitor the collected motes data and intervene by generating an adaptation action if an adaptation required. In addition, The Deep Learning Model will be triggered to estimate the mote state by effecting the SNR and Link Power. The adaptation is estimated for each mote on the network in order to provide the adaptation goals. The mote then sends the packets in its queue to its parents one by one. As soon as the queue is empty the mote returns to the idle state.

The global overview of our system will be presented on this diagram.

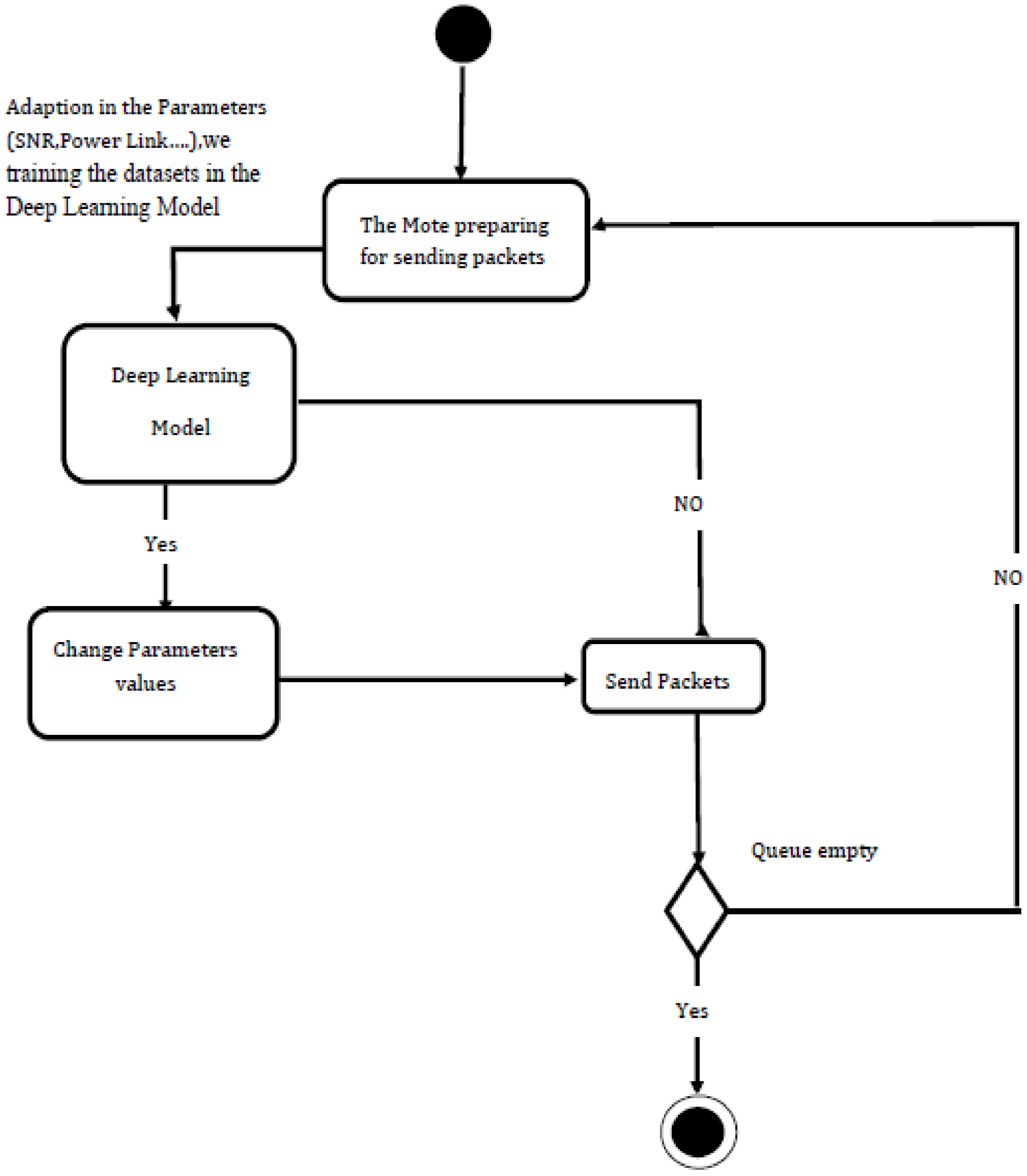

Adaption in the Parameters (SNR,Power Link....),we training the datasets in the Deep Learning Model
The Mote preparing for sending packets
Deep Learning Model
NO
Yes
NO
Change Parameters values
Send Packets
Queue empty
Yes

Figure 4.3. Global overview of the system

6-Conclusion

This chapter introduces an adaptive IoT system approach by applying architecture-based adaptation to it and a quality model (deep learning) that evaluates differences in quality attributes (Packet loss and Energy consumption and latency) to guaranty the adaptation objectives (QoS). the implementation of our system and the results of the experiment. We will present in the next chapter.

Chapter 05

Implementation

Chapter 05: Implementation

1-Introduction

Once we have detailed the architecture-based adaption approach over the IoT network.

This chapter is dedicated to the implementation phase, and we show how we build and implement our system. This is based on the representation of the software environment used in the tool view. Then there's the programming language.

2-Used Software and Materials

2.1- Materials

Laptop LENOVO.

Processor: Intel(R) Core (TM) i3-6006U CPU @ 2.00GHz ,2.00GHz.

Memory: 8.00GB RAM.

Graphics: Intel(R) HD Graphics 520.

Operating System: Windows 10 Professional 64-bit.

2.2- Programming language

2.2.1- Java

Java was originally developed by James Gosling at Sun Microsystems (which has since been acquired by Oracle) and released in 1995 as a core component of Sun Microsystems platform. Java is defined as an object-oriented language similar to C++, but simplified to eliminate language features that cause common programming errors.**[46]**

2.2.2- Python

Python is probably the easiest-to-learn and nicest-to-use programming language in widespread use. Python code is clear to read and write, and it is concise without being cryptic. Python is a very expressive language, which means we can usually write far fewer lines of Python code then would be required for an equivalent application written in, say, C++ or Java.

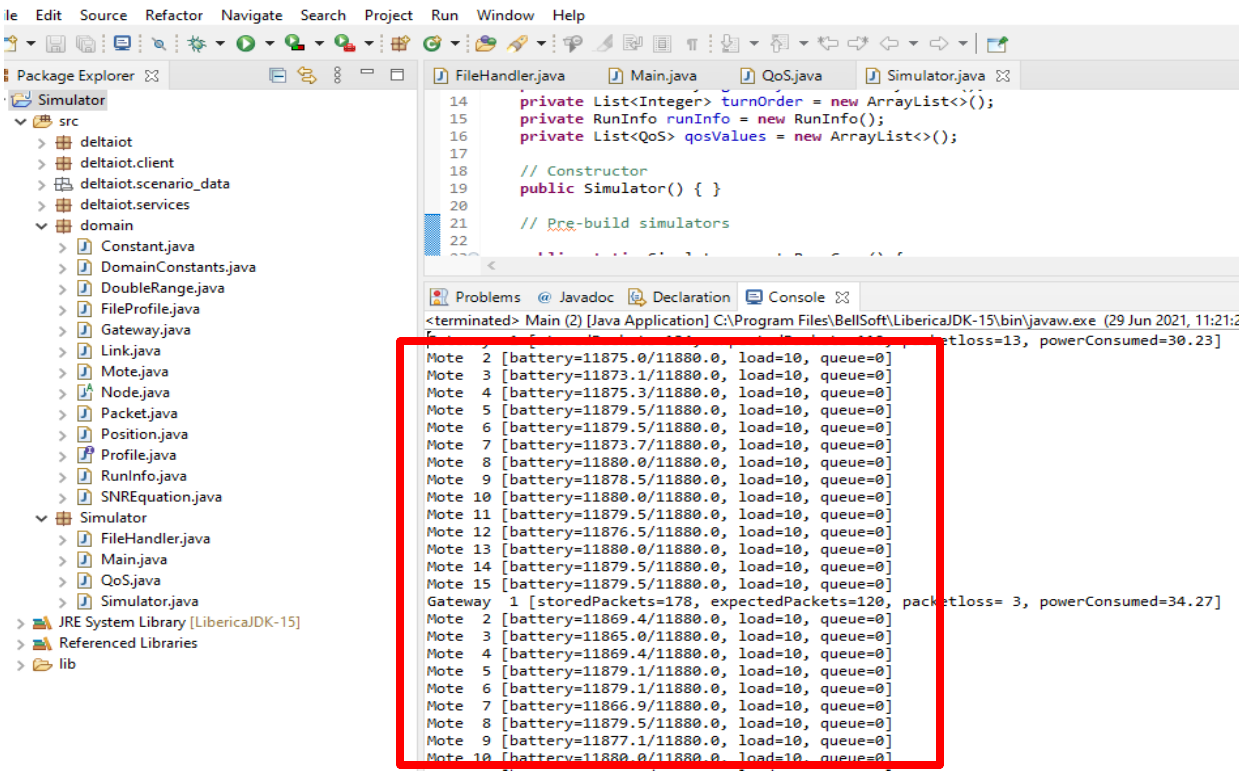
Python is a cross-platform language: In general, the same Python program can be run on Windows and Unix-like systems such as Linux, BSD, and Mac OS X, Rasbian, simply by copying the file or files that make up the program to the target machine, with no “building” or compiling necessary. It is possible to create Python programs that use platform-specific functionality, but this is rarely necessary since almost all of Python’s standard library and most third-party libraries are fully and transparently cross-platform.**[47]**

2.3- Development tools and frameworks

2.3.1-Eclipce

Eclipse is an integrated development environment (IDE) for developing applications using the Java programming language and other programming languages such as C/C++ Python, PERL, Ruby etc.

Developed using Java, the Eclipse platform can be used to develop rich client applications, integrated development environments and other tools. Eclipse can be used as an IDE for any programming language for which a plug-in is available **[48].**



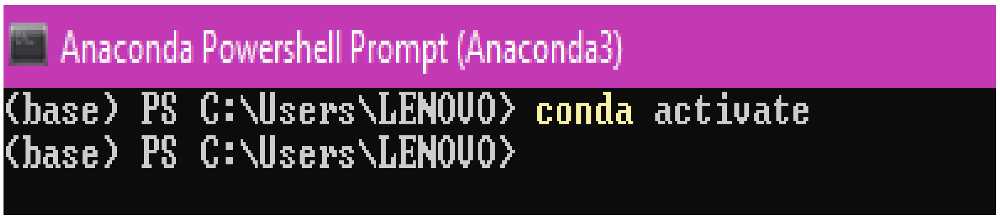
```
    private List<Integer> turnOrder = new ArrayList<>();
    private RunInfo runInfo = new RunInfo();
    private List<QoS> qosValues = new ArrayList<>();

    // Constructor
    public Simulator() { }

    // Pre-build simulators
```

```
<terminated> Main (2) [Java Application] C:\Program Files\BellSoft\LibericaJDK-15\bin\javaw.exe (29 Jun 2021, 11:21:2
                                                          etloss=13, powerConsumed=30.23]
Mote  2 [battery=11875.0/11880.0, load=10, queue=0]
Mote  3 [battery=11873.1/11880.0, load=10, queue=0]
Mote  4 [battery=11875.3/11880.0, load=10, queue=0]
Mote  5 [battery=11879.5/11880.0, load=10, queue=0]
Mote  6 [battery=11879.5/11880.0, load=10, queue=0]
Mote  7 [battery=11873.7/11880.0, load=10, queue=0]
Mote  8 [battery=11880.0/11880.0, load=10, queue=0]
Mote  9 [battery=11878.5/11880.0, load=10, queue=0]
Mote 10 [battery=11880.0/11880.0, load=10, queue=0]
Mote 11 [battery=11879.5/11880.0, load=10, queue=0]
Mote 12 [battery=11876.5/11880.0, load=10, queue=0]
Mote 13 [battery=11880.0/11880.0, load=10, queue=0]
Mote 14 [battery=11879.5/11880.0, load=10, queue=0]
Mote 15 [battery=11879.5/11880.0, load=10, queue=0]
Gateway  1 [storedPackets=178, expectedPackets=120, packetloss= 3, powerConsumed=34.27]
Mote  2 [battery=11869.4/11880.0, load=10, queue=0]
Mote  3 [battery=11865.0/11880.0, load=10, queue=0]
Mote  4 [battery=11869.4/11880.0, load=10, queue=0]
Mote  5 [battery=11879.1/11880.0, load=10, queue=0]
Mote  6 [battery=11879.1/11880.0, load=10, queue=0]
Mote  7 [battery=11866.9/11880.0, load=10, queue=0]
Mote  8 [battery=11879.5/11880.0, load=10, queue=0]
Mote  9 [battery=11877.1/11880.0, load=10, queue=0]
Mote 10 [battery=11880.0/11880.0, load=10, queue=0]
```

Figure 5.1. **DeltaIoT Simulator**

2.3.2-Anaconda

Anaconda is a distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. The distribution includes data-science packages suitable for Windows, Linux, and macOS. It is developed and maintained by Anaconda, Inc., which was founded by Peter Wang and Travis Oliphant in 2012.**[50]**



```
(base) PS C:\Users\LENOVO> conda activate
(base) PS C:\Users\LENOVO>
```

Figure 5.2. activate Anaconda.

2.3.3- TensorFlow

TensorFlow is a free and open-source software library for machine learning. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks. TensorFlow is a symbolic math library based on dataflow and differentiable programming. It is used for both research and production at Google. TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache License 2.0 in 2015.**[51]**

2.3.4-Numpy

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several other developers.**[53]**

2.3.5-Seaborn

Seaborn is a library for making statistical graphics in Python. It builds on top of matplotlib and integrates closely with pandas data structures. Seaborn helps you explore and understand your data. Its plotting functions operate on data frames and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots.**[56]**

2.3.6-Pandas

pandas are a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license. The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals.**[55]**

2.3.7-matplotlib

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK. There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged. SciPy makes use of Matplotlib.**[54]**

2.3.8-Ipython

IPython (Interactive Python) is a command shell for interactive computing in multiple programming languages, originally developed for the Python programming language, that offers introspection, rich media, shell syntax, tab completion, and history. Version 3.0 is the last "monolithic" version of IPython.**[52]**

2.3.9-Keras

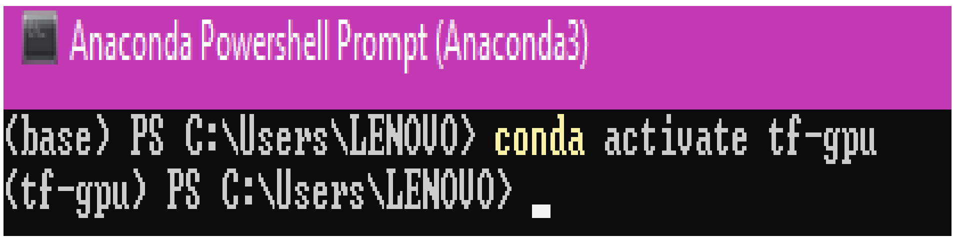
Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation.

Being able to go from idea to result as fast as possible is key to do good research. And its primary author and maintainer is François Chollet, a Google engineer. Keras is an approachable, 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 **[49].**

2.3.10- Statsmodel

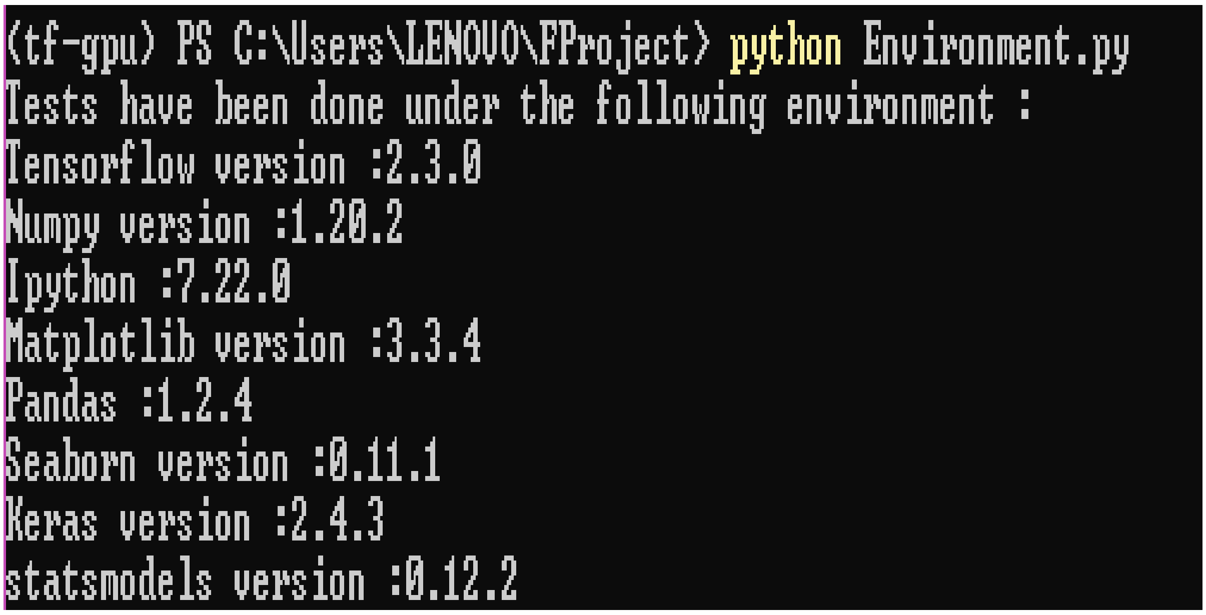
Statsmodels is a Python package that allows users to explore data, estimate statistical models, and perform statistical tests. An extensive list of descriptive statistics, statistical tests, plotting functions, and result statistics are available for different types of data and each estimator. It complements SciPy's stats module.

Statsmodels is part of the Python scientific stack that is oriented towards data analysis, data science and statistics. Statsmodels is built on top of the numerical libraries NumPy and SciPy, integrates with Pandas for data handling, and uses Patsy for an R-like formula interface. Graphical functions are based on the Matplotlib library.**[56]**



```
(base) PS C:\Users\LENOVO> conda activate tf-gpu
(tf-gpu) PS C:\Users\LENOVO> _
```

Figure 5.3. Activate TensorFlow Environment.



```
(tf-gpu) PS C:\Users\LENOVO\FProject> python Environment.py
Tests have been done under the following environment :
Tensorflow version :2.3.0
Numpy version :1.20.2
Ipython :7.22.0
Matplotlib version :3.3.4
Pandas :1.2.4
Seaborn version :0.11.1
Keras version :2.4.3
statsmodels version :0.12.2
```

Figure 5.4. Testing The Environment**.**

- as shown in ***Appendix A***.

3- Simulator

We used the DeltaIoT exemplar which offers a simulator for self-adaptation experimentation.

The DeltaIoT exemplar enables researchers to evaluate and compare new methods, techniques and tools for self-adaptation in Internet of Things (IoT). DeltaIoT is part of the smart campus initiative by imec-DistriNet, KU Leuven, Belgium.

DeltaIoT consists of a collection of 25 LoRa-based Internet of Things (IoT) motes. In each building, motes are strategically placed to provide access control to labs (via RFID sensor), to monitor the occupancy status (via Passive infrared sensor) and to sense the temperature (via temperature sensor). The sensor data from all the buildings are relayed to the IoT gateway, which is deployed at a central monitoring facility. Campus security personnel monitor the status of various buildings and labs from the monitoring facility, and take appropriate action whenever the unusual behaviour is detected in the buildings.

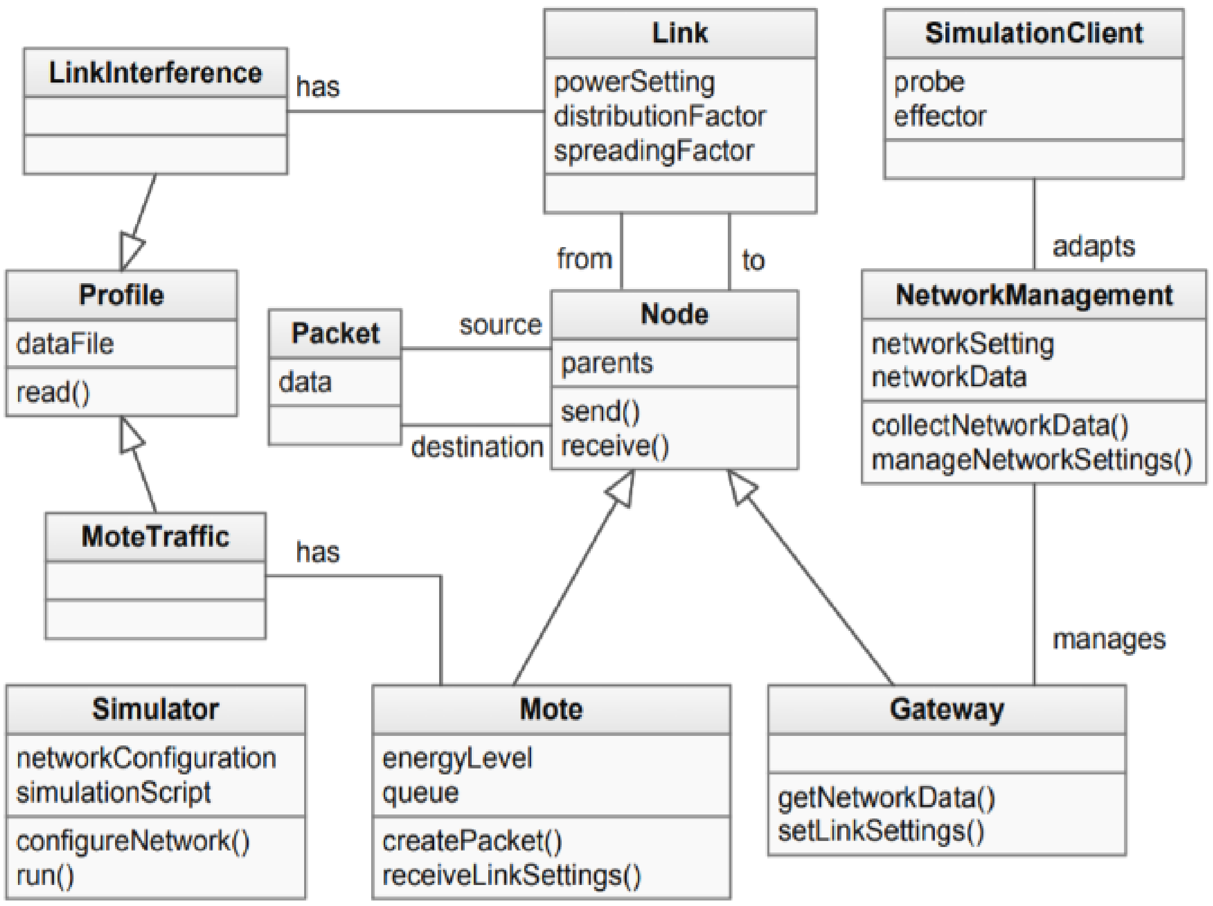


Figure 5.5. Architecture of DeltaIoT Simulator **[1]**

- ***Node, Gateway, Mote, Link***, and *Packet* are the basic elements of the IoT network. These elements correspond to the IoT Network Tier and the Gateway Tier.
- ***Network Management*** provides the functionality for collecting and processing network data and changing the network settings. This element corresponds to the Management Tier.
- The ***Simulation Client*** offers a ***probe*** and ***effector*** to apply self-adaptation to the IoT network.
- ***Profile*** to specify uncertainties in the simulated system. A profile is defined by a file that contains a series of values that represent a property of the system or its environment over time.
- ***Link Interference*** defines the levels of interference on a link over time.
- ***Mote Traffic*** defines the traffic generated by a mote over time.
- ***Simulator*** enables a user to perform a simulation of a network configuration.

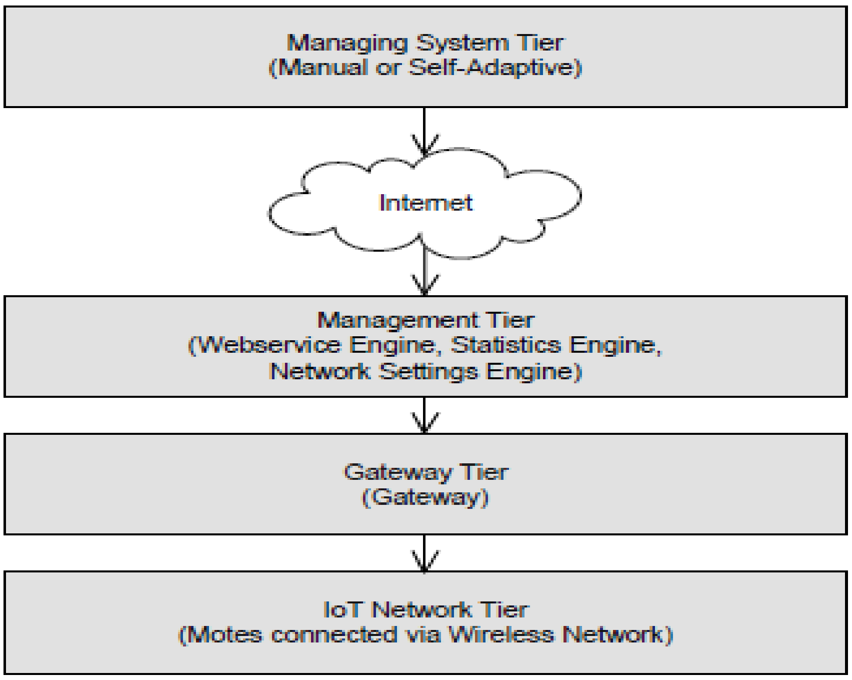

Managing System Tier
(Manual or Self-Adaptive)
Internet
Management Tier
(Webservice Engine, Statistics Engine, Network Settings Engine)
Gateway Tier
(Gateway)
IoT Network Tier
(Motes connected via Wireless Network)

Figure 5.6. Architecture of DeltaIoT Deployed at KU Leuven **[1]**

1. ***IoT Network Tier:*** The IoT network tier consists of the IoT motes connected via a wireless communication network.
2. ***Gateway Tier:*** The gateway tier is the root for the IoT network. All the IoT motes in the network transmit their data to the gateway, which is then consumed by the users based on their application requirements. Thus, all the IoT motes must have a route towards the gateway.
3. ***Management Tier:*** The IoT motes and the gateway can be monitored and managed via the management tier. The management tier consists of three key building blocks:
    a. ***Webservice Engine:*** connects DeltaIoT to the external world via Internet through a web service.
    b. ***Statistics Engine:*** is responsible for collecting and storing network statistics.
    c. ***Network Settings Engine:*** interacts with the IoT network via the gateway to collect data and adapt the network settings of the IoT motes.
4. ***Managing System Tier:*** The managing system tier contains the entity that is responsible for managing the IoT network via the management tier. In a traditional setting, network management is done manually by a system administrator.

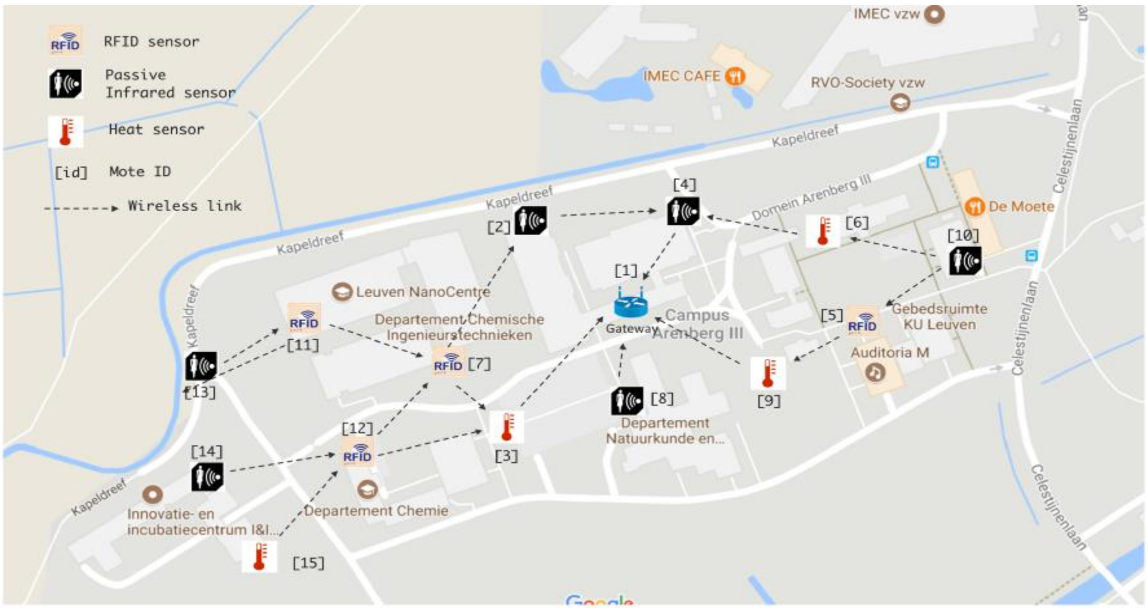


Figure 5.7. DeltaIoT network topology **[1]**

4- Building model

Support Vector Regression (SVR) One of the Most Flexible Yet Robust Prediction Algorithms. All other Machine Learning Algorithms seem to be 'BLIND' as they don't verify the correlation between variables.

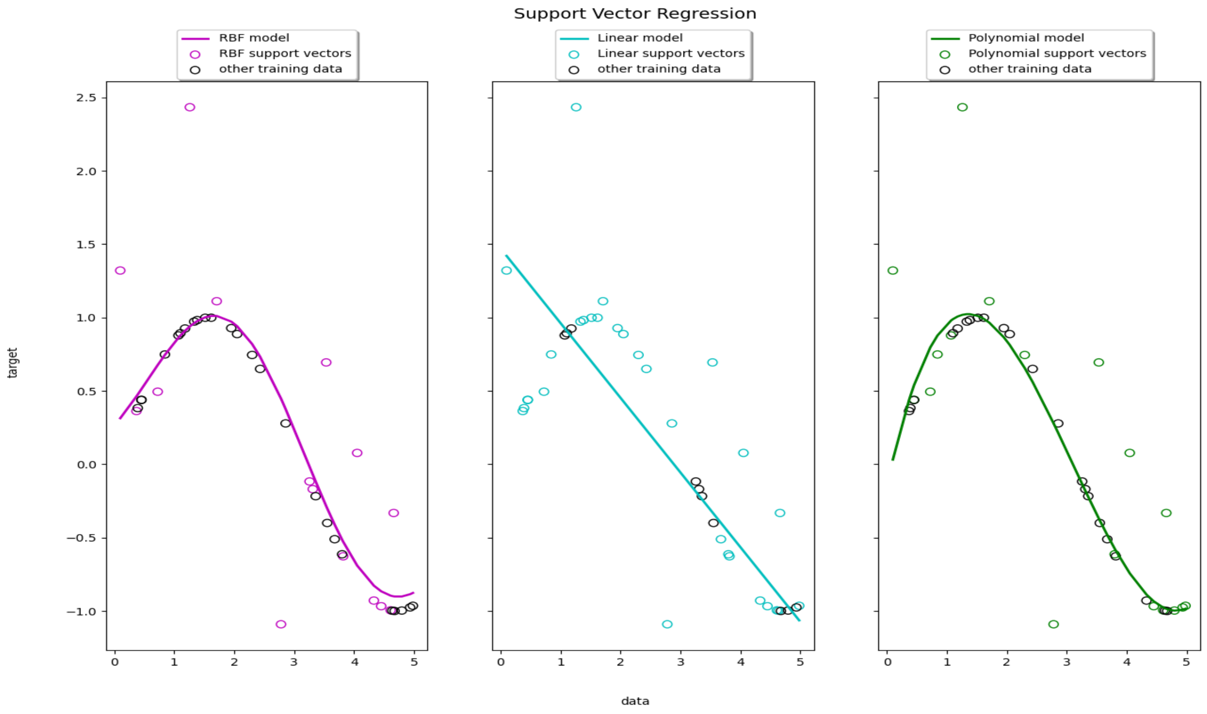
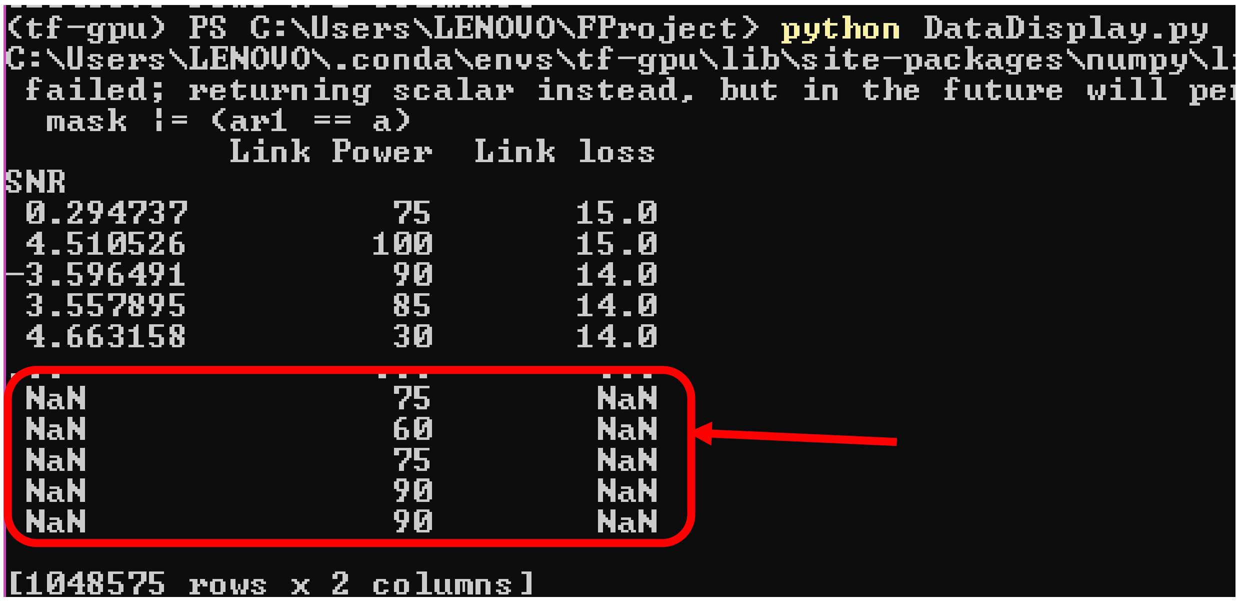


Figure 5.8. Support Vector Regression (SVR) **[57]**

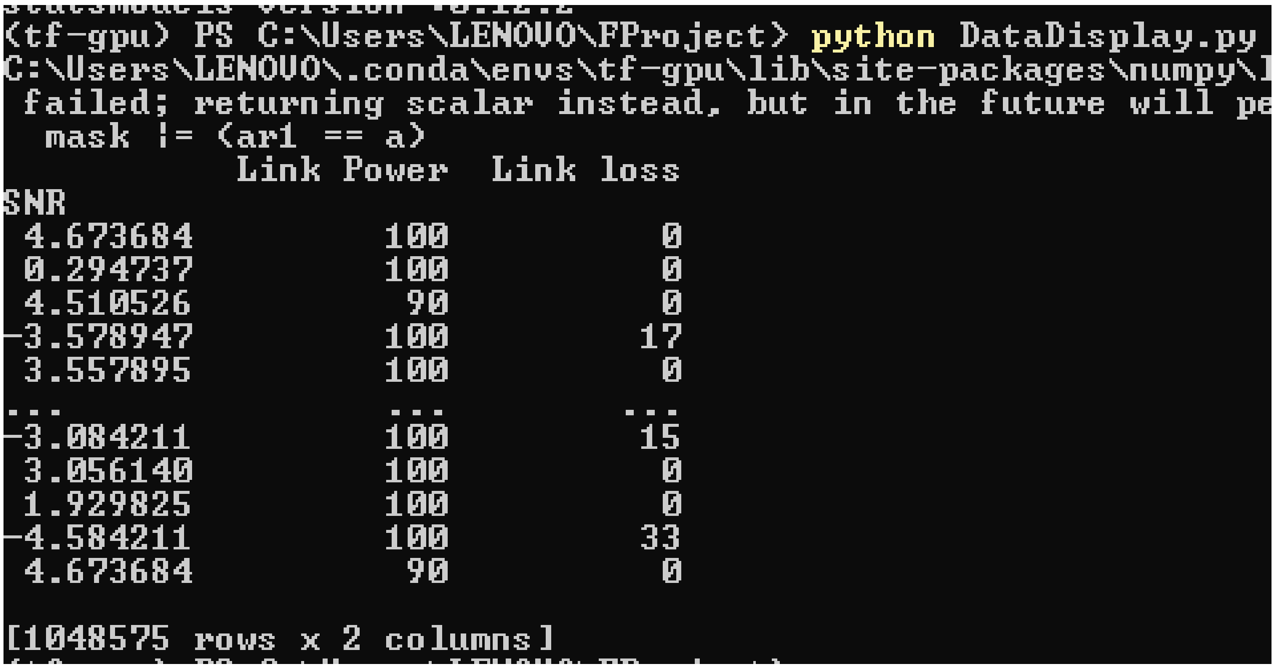
To perform the chosen ML algorithms, three datasets were given. This obstacle is known in Data Science and Machine Learning as that it destroys every data analysis process. We see NaN (Not a Number) values in dataset3 for example.



```
(tf-gpu) PS C:\Users\LENOVO\FProject> python DataDisplay.py
C:\Users\LENOVO\.conda\envs\tf-gpu\lib\site-packages\numpy\l
 failed; returning scalar instead, but in the future will pe
  mask |= (ar1 == a)
                    Link Power   Link loss
SNR
 0.294737                75         15.0
 4.510526               100         15.0
-3.596491                90         14.0
 3.557895                85         14.0
 4.663158                30         14.0
 ...                    ...          ...
 NaN                     75          NaN
 NaN                     60          NaN
 NaN                     75          NaN
 NaN                     90          NaN
 NaN                     90          NaN

[1048575 rows x 2 columns]
```

Figure 5.9. Example showing Displaying Dateset them have faulty data.

The Figure.5.10 showing the content of The Dataset showing 3 columns each one represents the consider Parameters.

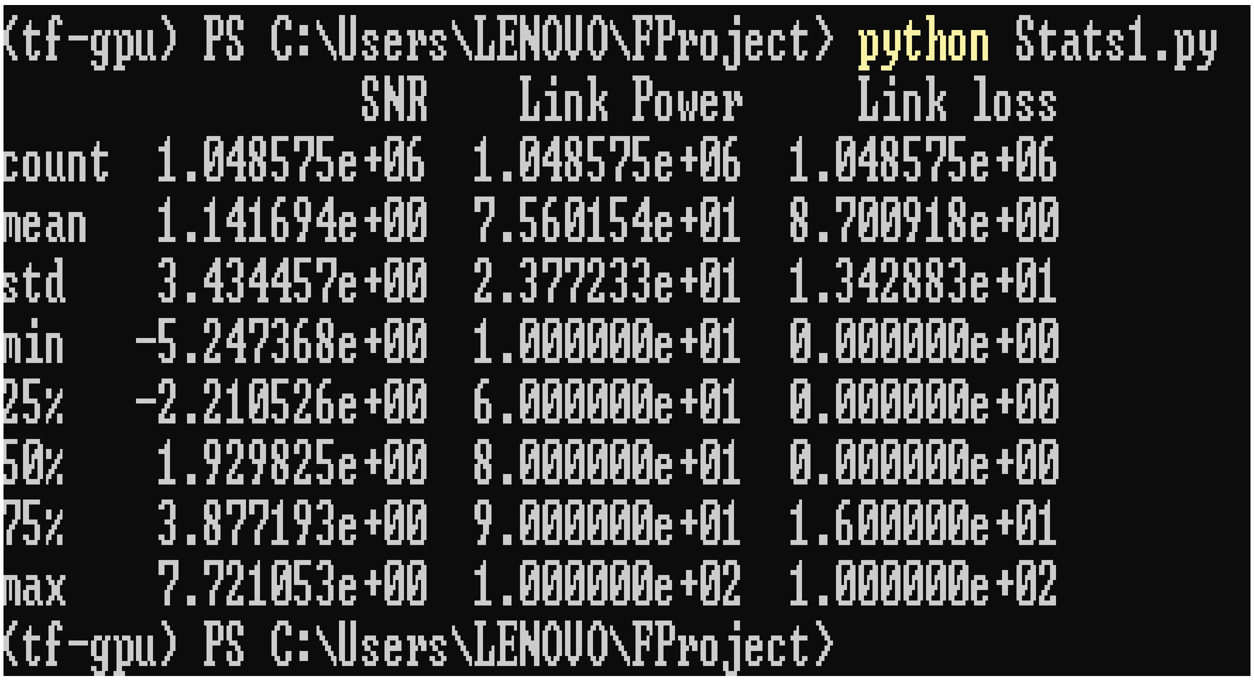


```
(tf-gpu) PS C:\Users\LENOVO\FProject> python DataDisplay.py
C:\Users\LENOVO\.conda\envs\tf-gpu\lib\site-packages\numpy\l
 failed; returning scalar instead, but in the future will pe
  mask |= (ar1 == a)
                    Link Power  Link loss
SNR
 4.673684               100          0
 0.294737               100          0
 4.510526                90          0
-3.578947               100         17
 3.557895               100          0
...                     ...        ...
-3.084211               100         15
 3.056140               100          0
 1.929825               100          0
-4.584211               100         33
 4.673684                90          0

[1048575 rows x 2 columns]
```

Figure 5.10. Example showing Displaying Dateset.

- as shown in ***Appendix C***.

The Figure.5.11 showing the Statistic of our data before adaptation



```
(tf-gpu) PS C:\Users\LENOVO\FProject> python Stats1.py
                SNR    Link Power     Link loss
count  1.048575e+06  1.048575e+06  1.048575e+06
mean   1.141694e+00  7.560154e+01  8.700918e+00
std    3.434457e+00  2.377233e+01  1.342883e+01
min   -5.247368e+00  1.000000e+01  0.000000e+00
25%   -2.210526e+00  6.000000e+01  0.000000e+00
50%    1.929825e+00  8.000000e+01  0.000000e+00
75%    3.877193e+00  9.000000e+01  1.600000e+01
max    7.721053e+00  1.000000e+02  1.000000e+02
(tf-gpu) PS C:\Users\LENOVO\FProject>
```

Figure 5.11. Example showing Statistic of the Dateset.

Count: number of records of dataset.

Min: The minimum for each one of parameters.

Std: The standard deviation is a measure of the amount of variation or dispersion of a set of values.

Max: The maximum for each one of parameters.

Mean: each parameter and its equivalent QoS (represented by Link Loss), for example the minimal value of SNR (upon whole dataset) is -5.247368, that of Link Power is 10, while the Link Loss = 0 (good connectivity)

75%: of SNR values are 3.877193, while 75% of Link Power are 90, resulting in 16 times of Loosing Connectivity (Link Loss).

- as shown in ***Appendix B***.

Applying the SVR Model as adaptation model because it determines the regression line at which all data can be predicted in advance and makes it as a rule for future situations.

After loading the dataset, the correlation (common relation) between the project parameters implies defining the X variables and the Y resulting from them, is done.

Setting the project parameters exactly as described:

The QoS (represented by Link Loss, how many times connexion is lost) is Y = F (X1, X2), where X1 = SNR and X2=Link Power.

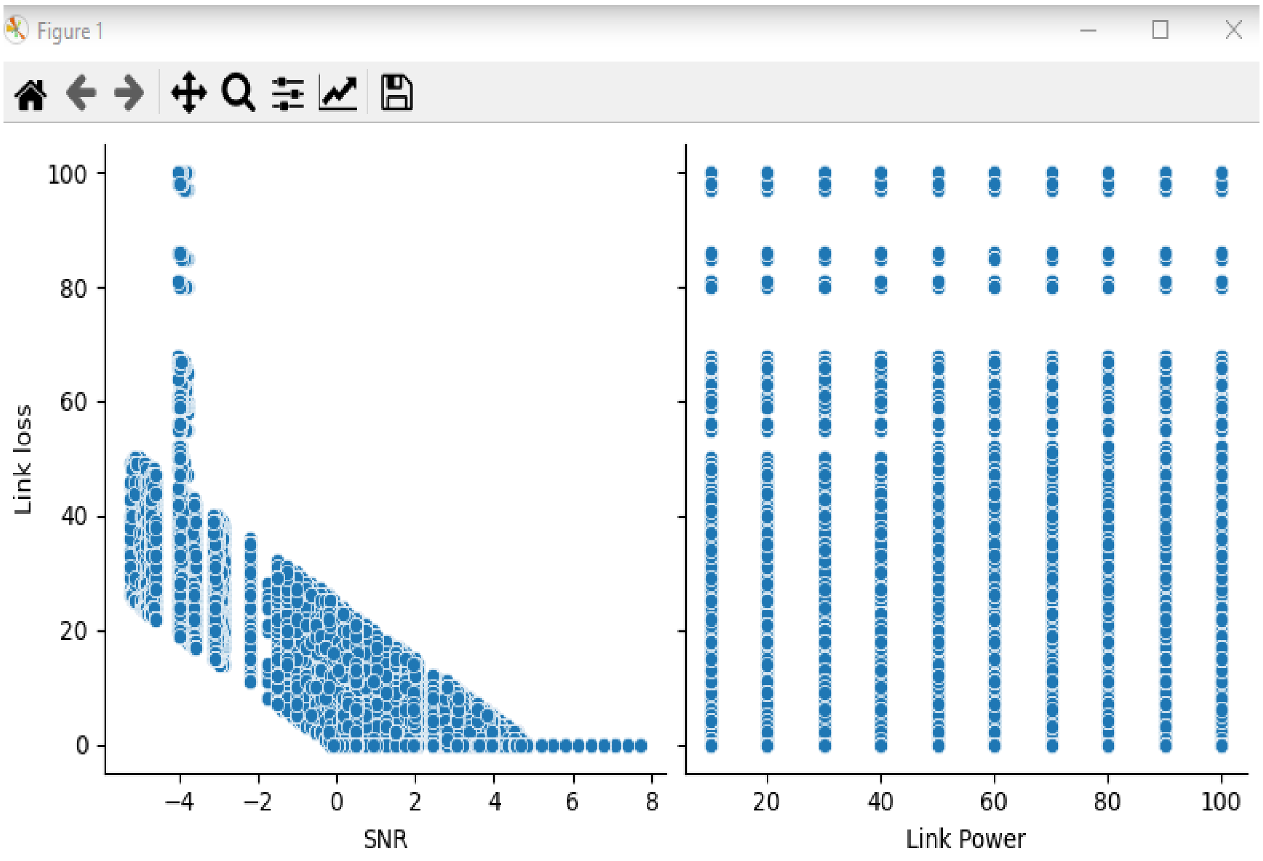


Figure 5.12. Plots of the Dateset before Adaptation.

The Figure.5.12 showing the plots of the degradation of system depending on SNR, power link, loss link.

As seen in the Figure the SNR=0 from 0 to 8 mean link loss =0, while is 100% when SNR is in range of -4.

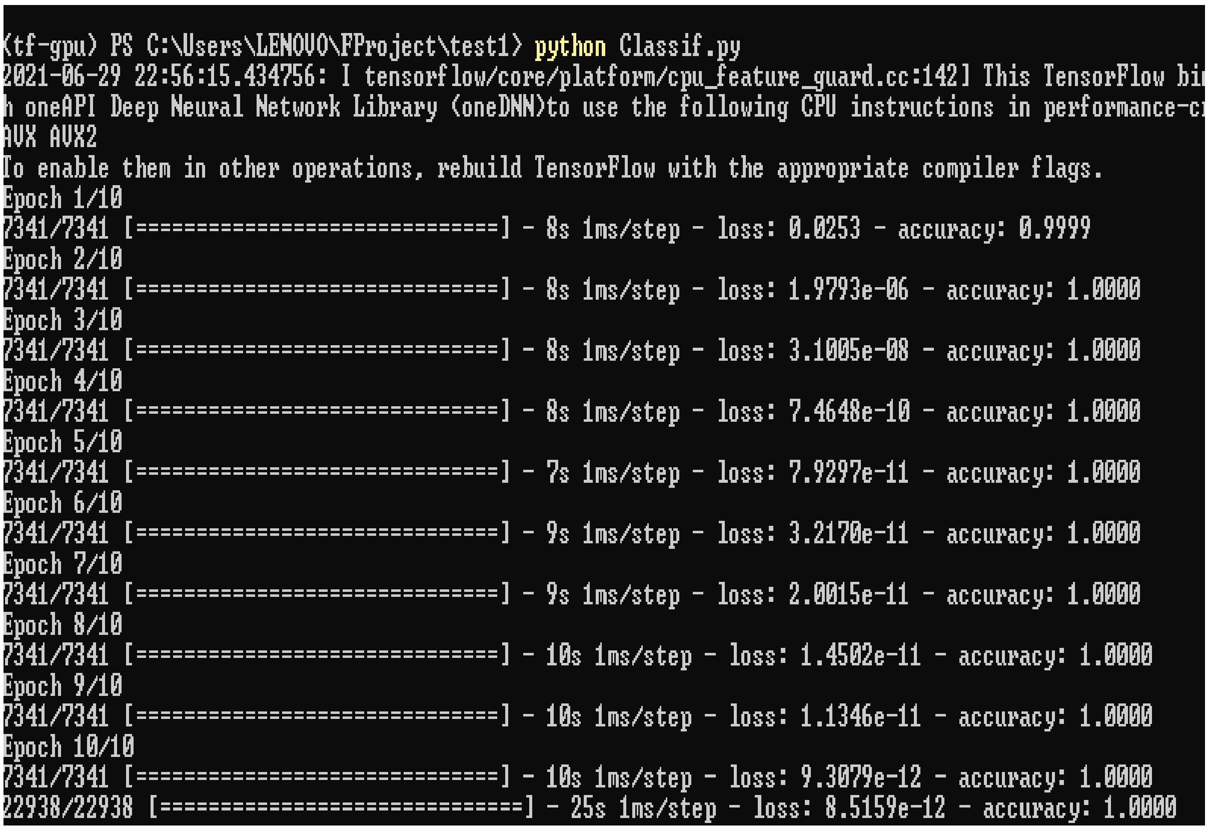
As seen in the Figure Link power always higher.

- as shown in ***Appendix F***.

5- Obtained Results

We performed a classification algorithm and Sequential one to have an idea about the expected results based on the Simulator generated dataset. With 100% of Accuracy given (by the two algorithms we understood that dataset cannot come from actual network metrics).

Test of a DL Neural Network designed for Classification to be sure that the adaptation given by Simulator is done. (Figure5.13)

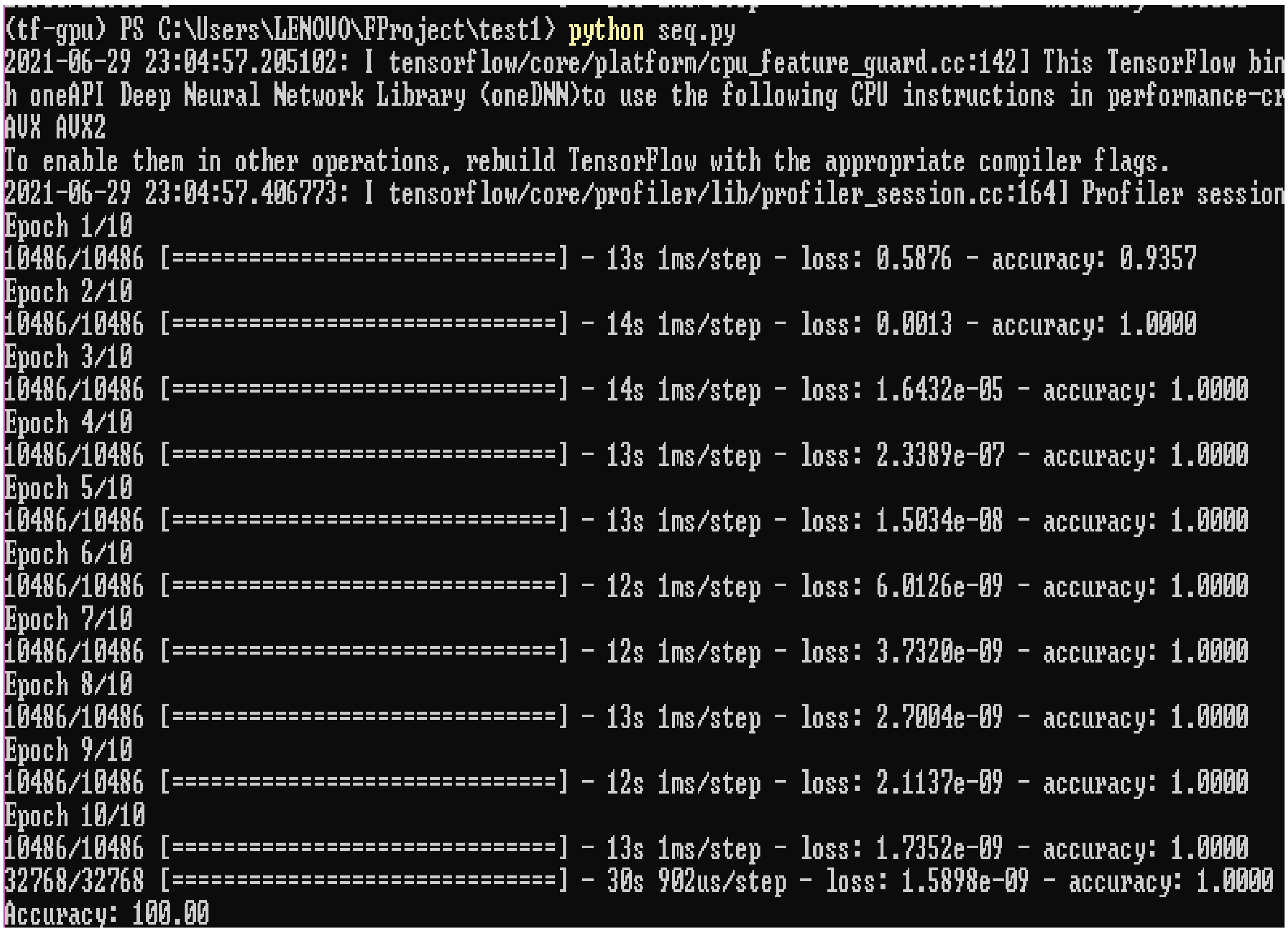


```
(tf-gpu) PS C:\Users\LENOVO\FProject\test1> python Classif.py
2021-06-29 22:56:15.434756: I tensorflow/core/platform/cpu_feature_guard.cc:142] This TensorFlow bi
h oneAPI Deep Neural Network Library (oneDNN)to use the following CPU instructions in performance-c
AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
Epoch 1/10
7341/7341 [==============================] - 8s 1ms/step - loss: 0.0253 - accuracy: 0.9999
Epoch 2/10
7341/7341 [==============================] - 8s 1ms/step - loss: 1.9793e-06 - accuracy: 1.0000
Epoch 3/10
7341/7341 [==============================] - 8s 1ms/step - loss: 3.1005e-08 - accuracy: 1.0000
Epoch 4/10
7341/7341 [==============================] - 8s 1ms/step - loss: 7.4648e-10 - accuracy: 1.0000
Epoch 5/10
7341/7341 [==============================] - 7s 1ms/step - loss: 7.9297e-11 - accuracy: 1.0000
Epoch 6/10
7341/7341 [==============================] - 9s 1ms/step - loss: 3.2170e-11 - accuracy: 1.0000
Epoch 7/10
7341/7341 [==============================] - 9s 1ms/step - loss: 2.0015e-11 - accuracy: 1.0000
Epoch 8/10
7341/7341 [==============================] - 10s 1ms/step - loss: 1.4502e-11 - accuracy: 1.0000
Epoch 9/10
7341/7341 [==============================] - 10s 1ms/step - loss: 1.1346e-11 - accuracy: 1.0000
Epoch 10/10
7341/7341 [==============================] - 10s 1ms/step - loss: 9.3079e-12 - accuracy: 1.0000
22938/22938 [==============================] - 25s 1ms/step - loss: 8.5159e-12 - accuracy: 1.0000
```

Figure.5.13. showing the Training of datasets by classification.

- as shown in ***Appendix D***.

Test of another DL model, the Sequential, only to be sure that data given in dataset are coherent. (Figure5.14)



```
(tf-gpu) PS C:\Users\LENOVO\FProject\test1> python seq.py
2021-06-29 23:04:57.205102: I tensorflow/core/platform/cpu_feature_guard.cc:142] This TensorFlow bin
h oneAPI Deep Neural Network Library (oneDNN)to use the following CPU instructions in performance-cr
AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2021-06-29 23:04:57.406773: I tensorflow/core/profiler/lib/profiler_session.cc:164] Profiler session
Epoch 1/10
10486/10486 [==============================] - 13s 1ms/step - loss: 0.5876 - accuracy: 0.9357
Epoch 2/10
10486/10486 [==============================] - 14s 1ms/step - loss: 0.0013 - accuracy: 1.0000
Epoch 3/10
10486/10486 [==============================] - 14s 1ms/step - loss: 1.6432e-05 - accuracy: 1.0000
Epoch 4/10
10486/10486 [==============================] - 13s 1ms/step - loss: 2.3389e-07 - accuracy: 1.0000
Epoch 5/10
10486/10486 [==============================] - 13s 1ms/step - loss: 1.5034e-08 - accuracy: 1.0000
Epoch 6/10
10486/10486 [==============================] - 12s 1ms/step - loss: 6.0126e-09 - accuracy: 1.0000
Epoch 7/10
10486/10486 [==============================] - 12s 1ms/step - loss: 3.7320e-09 - accuracy: 1.0000
Epoch 8/10
10486/10486 [==============================] - 13s 1ms/step - loss: 2.7004e-09 - accuracy: 1.0000
Epoch 9/10
10486/10486 [==============================] - 12s 1ms/step - loss: 2.1137e-09 - accuracy: 1.0000
Epoch 10/10
10486/10486 [==============================] - 13s 1ms/step - loss: 1.7352e-09 - accuracy: 1.0000
32768/32768 [==============================] - 30s 902us/step - loss: 1.5898e-09 - accuracy: 1.0000
Accuracy: 100.00
```

Figure.5.14. showing the Training of datasets by sequential.

- as shown in ***Appendix E***.

As we have approximatively 1,5 million of dataset records, we took 10000 for fast running of script –from 0 to 9999).

And then, applying the required splitting of data (into Train and Test) (it is mandatory), the training of first parameter is done (XP1, YP1).

Then, using the statsmodels library, the regression line (y = a x + b) is determined by its coefficients, here a and b are determined by lr. parameters and X=XP1= SNR at 10% of Simulated one.

Interpretation of F(SNR) =Link Loss

When applying the first constraint, ensuring 10% SNR from tested data, and we training it.

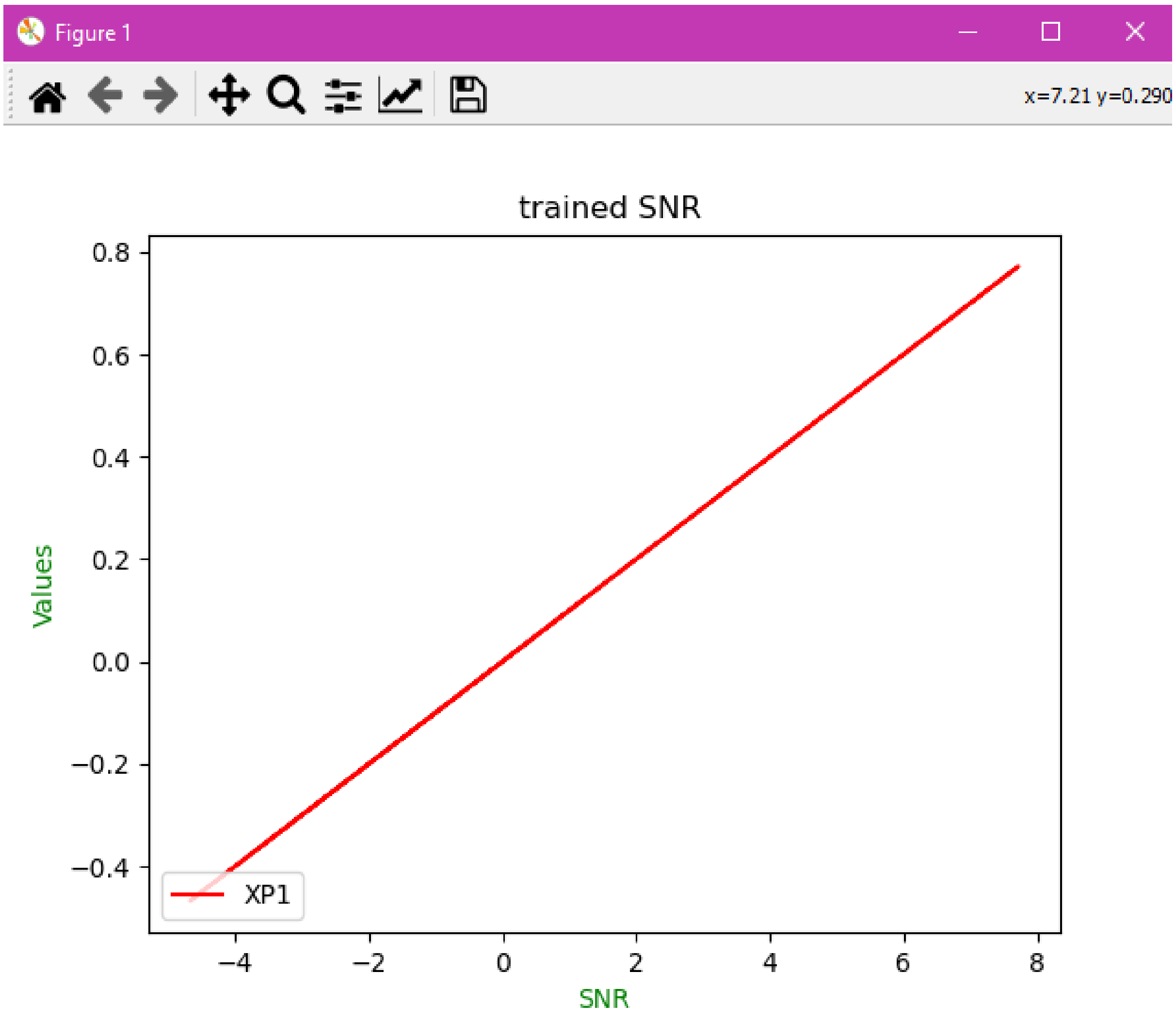


Figure.5.15. showing the evolution of SNR.

- Xp1 resent the new value of SNR after training.
- as shown in ***Appendix F***.

Interpretation of F (Link Power) = Link Loss

The stats models applied to the secund constraint, ensure minimal value of Link Power.

Applied to a line equation, y = ax + b, b=0.

The trained Link Power at its mean value reached the initial value (adaptation by prediction of the regression line)

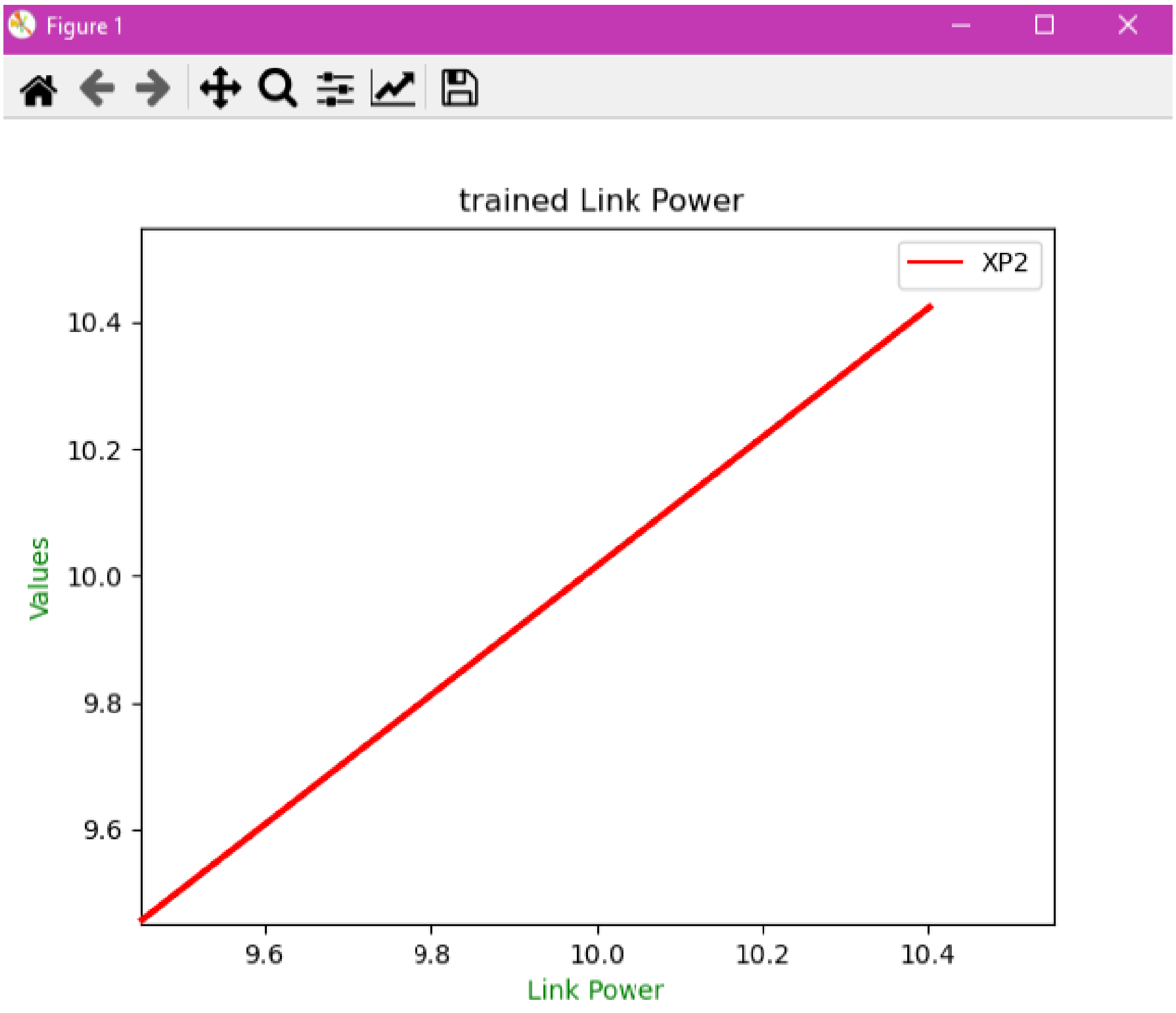
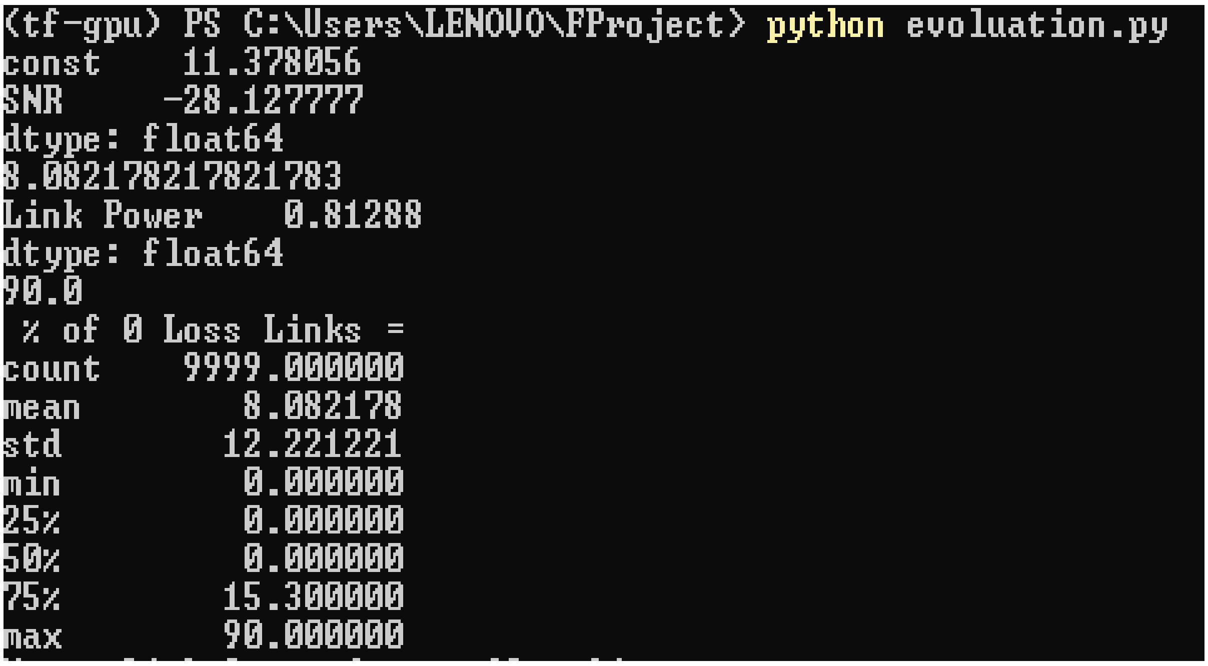


Figure.5. 16. showing the evolution of Link Power.

- Xp2 resent the new value of Link Power after training.
- as shown in ***Appendix F***.



```
(tf-gpu) PS C:\Users\LENOVO\FProject> python evoluation.py
const     11.378056
SNR      -28.127777
dtype: float64
8.082178217821783
Link Power     0.81288
dtype: float64
90.0
 % of 0 Loss Links =
count     9999.000000
mean         8.082178
std         12.221221
min          0.000000
25%          0.000000
50%          0.000000
75%         15.300000
max         90.000000
```

Figure.5. 16. Total QoS ensured.

we can determine the QoS (represented by Link Loss) as the trained YP1.

0 Link Loss = perfect connection = very good QoS, here we found it at 90% as required by project constraints.

- as shown in ***Appendix F***.

6- Conclusion

In order to verify the design of the system, appropriate tools must be used to implement it.

In this chapter, we introduced in detail some details of the implementation of adaptation based on the QoS architecture in the Internet of Things system, and combined with machine learning algorithms.

Conclusion

Conclusion

1-General context

The Internet of Things (IoT) is one of the leading technologies that related smart devices pay attention to. It has an extraordinary impact on the technical, social, economic and internal aspects of human and machine life, and because the Internet of Things structure requires quality of service (QoS), it makes life safer and more comfortable (such as low power consumption, low data packets Loss, connection, delay) should prioritize QoS indicators on IoT computers.

In order to meet the QoS requirements in IoT, Mote must dynamically adjust its configuration in the system.

In our work, we put three parameters in consider (SNR, link Power, Link Loss), they represent basic element for ensure qualities for IoT systems with battery-powered mobile end nodes.

To automate the management of Internet-of-Things (IoT), this approach used architecture-based adaptation on IoT system with a feedback loop on top of it. This MAPE-K loop employs runtime models and used its data to adapt the system to ensure the required goals.

To achieve the adaptation goals, first we implemented SVR algorithm to each mote in the system to improve the values of parameters which leads to provide the required goals.

Choosing the SVR algorithm to perform prediction and adaptation problems, SVR can also be used as multioutput algorithm, by taking the mean of each trained set of independent variables. SVR is also the most controllable algorithm.

Also, we implemented test for Deep learning Neural Network designed for classification in order to ensure the adaptation given by Simulator is performed. Another DL model, only to be sure that data given in dataset are coherent.

All this data will finally reach the MAPE-K loop to handle uncertainties of the system (The interference of the network links in IoT system or the traffic generated by the motes)

And generate the best adaptation action to provide the quality goals.

The main advantage of our automated IoT system management method is that we can process changes faster every time and extend the life of the system.In terms of risk, a predictive component that predicts the entire Internet of Things network through the use of various machine learning algorithms is needed.

2-Future Work

we anticipate further study adjustment issues with more different kinds of uncertainties, such as uncertainties of security attacks, mote mobility, connectivity etc...

we also further to create IoT System to work on using NS3.

3-challenges

as this project belongs to four Computer Science domains:

- Networking
- Artificial Intelligence
- Bigdata
- Descriptive Statistics

We got a big idea about each of them by this practical project but we noticed that the huge information they require is time consuming and such projects require much more time and previous training in languages and AI libraries and robust computers.

References

References

[1] M. Usman Iftikhar, Gowri Sankar Ramachandran & al, DeltaIoT: A Self-Adaptive Internet of Things Exemplar, Linnaeus University & KU Leuven, 2017, p.1.

[2] Janice Ca˜nedo, Anthony Skjellum, « Using Machine Learning to Secure IoT Systems », Auburn Cyber Research Center Samuel Ginn College of Engineering Auburn University, 2016, p.1.

[3] Nathalie Baracaldo, Bryant Chen & al, « Detecting Poisoning Attacks on Machine Learning in IoT Environments », IEEE International Congress on Internet of Things, 2018, p.57.

[4] Safa MERAGHNI, Labib Sadek TERRISSA and al,"An IoT-based solution for Post-Prognostics Decision in Cloud computing environment,2016, 11th International Conference on Modeling, Optimization and Simulation - MOSIM'16",p.2-3.

[5] Rouse, Margaret "internet of things (IoT)", IOT Agenda Retrieved 14 August 2019.

[6] Carlos Salazar and al, "Internet of Things-IOT: Definition, Characteristics, Architecture, Enabling Technologies, Application & Future Challenges", International Journal of Engineering Science and Computing, May 2016, p.3.

[7] Pallavi Sethi and Smruti R. Sarangi," Internet of Things: Architectures, Protocols, and Applications", 26 January 2017, p.2-3.

[8] Yusuf Perwej, Firoj Parwej," The Internet of Things (IoT) and its Application Domains", International Journal of Computer Applications, April 2019, p.37-47.

[9] Zainab H. Ali & al," Internet of Things (IoT): Definitions, Challenges and Recent Research Directions", International Journal of Computer Application, Volume 128, October 2015, p.40-42.

[10] M.U. Farooq & al, « A Review on Internet of Things (IoT) », International Journal of Computer Applications, Volume 113, March 2015, p.3-4.

[11] Henry Muccini and al,” Self-adaptive IoT Architectures an Emergency Handling Case Study”, ECSA '18: Proceedings of the 12th European Conference on Software Architecture: Companion Proceedings,2018, p.2.

[12] Manisha Singh, Gaurav Baranwal, « Quality of Service (QoS) in Internet of Things », IEEE,2018, p.3.

[13] Ren Duan and al,” A QoS Architecture for IOT”, IEEE International Conferences on Internet of Things, and Cyber, Physical and Social Computing,2011, p.718.

[14] Frank D. Macias-Escriva & al, « Self-adaptive systems: A survey of current approaches, research challenges and applications », Expert Systems with Applications, 22 July 2013, p.13.

[15] D.wayne, « Software Engineering of Self-adaptive Systems », Handbook of Software Engineering,2019, p.402-404.

[16] David Garlan & al, « Software Architecture-Based Self-Adaptation », Autonomic Computing and Networking, April 2009, p.34.

[17] Frank D. Macias-Escriva & al, « Self-adaptive systems: A survey of current approaches, research challenges and applications », Expert Systems with Applications, 22 July 2013, p.15.

[18] D.wayne, S.Michiels, "Learning to efficiently analyze large adaptation spaces of Internet of Things applications, Thesis nominated for obtaining the degree of Master of Science in: computer science, Software engineering,2018-2019, p.5-6.

[19] Shampa Sen, Leonid Datta and Sayak Mitra,” Machine Learning and IoT”, Edition 1,2019, p.2.

[20] Konstantinos G. Liakos and all,” Machine Learning in Agriculture: A Review”, sensors, 14 August 2018, p.4.

[21] Ryszard S. Michalski Jaime G. Carbonell Tom M. Mitchell “Machine Learning an Artificial Intelligence Approach”, Edition Number 1, p.7

[22] Ayon Dey, Machine Learning Algorithms: A Review, (IJCSIT) International Journal of Computer Science and Information Technologies, Volume. 7, p.1.

[23] Supervised Machine Learning: What is, Algorithms, Example, Available: www.

guru99.com/supervised-machine-learning.html.

[24] Unsupervised Machine Learning: What is, Algorithms, Example, Available: www.guru99.com/unsupervised-machine-learning.html.

[25] Ronald van Loon, Machine learning explained: Understanding supervised, unsupervised, and reinforcement learning,2018, Available: bigdata-madesimple.com/machine-learning-explained-understanding-supervised-unsupervised-and-reinforcement-learning.

[26] Wang Qiang,” Reinforcement Learning model, algorithms and its application”, International Conference on Mechatronic Science, Electric Engineering and Computer (MEC),2011, p.1.

[27] An Introduction to Machine Learning Algorithms,2019, available: litslink.com/blog/an-introduction-to-machine-learning-algorithms.

[28] Abul Bashar,” SURVEY ON EVOLVING DEEP LEARNING NEURAL NETWORK

ARCHITECTURES”, Journal of Artificial Intelligence and Capsule Networks, Volume 1, 2019, p.74.

[29] Adel Mellit and all,” Advanced Methods for Photovoltaic Output Power Forecasting: A Review”, applied sciences,2020, p.4.

[30] Anisha Bhatnagar and all,” Machine Learning Techniques to Reduce Error in the

Internet of Things”, IEEE,2019, p.405.

[31] Dave Anderson and George McNeill, “ARTIFICIAL NEURAL NETWORKS TECHNOLOGY",1992, p.2-7.

[32] Charu C. Aggarwal,” Neural Networks and Deep Learning”,2018, p.1.

[33] P.Radhakrishnan, what are hyperparameters? ,2019, Available: towardsdatascience.com/ what-are-hyperparameters-and-how-to-tune-thehyperparameters-in-a-deep-neural-network-d0604917584a.

[34] David Panys, Artificial Intelligence Wiki,2020, Available: docs.paperspace.com/ machine-learning/wiki/weights-and-biases.

[35] Mukul Malik, Basics of Neural Network, 2019, Available: becominghuman.ai/basics-of-neural-network-bef2ba97d2cf.

[36] Irfan Danish, Introduction to Neural Networks and Their Key Elements,2020, Available: towardsai.net/p/machine-learning/introduction-to-neural-networks-and-their-key-elements-part-c-activation-functions-layers-ea8c915a9d9.

[37] W.Danny, & al , "Applying architecture-based adaptation to automate the management of internet-of-things", in European Conference on Software Architecture, Springer, 2018, p.49.

[38] B. Robbe and W. Danny, "A QoS-aware adaptive mobility handling approach for Lora-based IoT systems", 2018, p. 130.

[39] Li Deng and al,Deep Learning: Methods and Applications,Volume.7,2014, p.199.

[40] Tarang Shah,"About Train, Validation and Test Sets in Machine Learning" ,2017, available: www.kaggle.com/getting-started/143685

[41] LIAKOS and al. Machine learning in agriculture: A review. Sensors, 2018, vol. 18, p.6.

[42] Difference between ANN, CNN and RNN,2020, Available: www.geeksforgeeks.org/ difference-between-ann-cnn-and-rnn.

[43] Introduction to Recurrent Neural Network,2018, Available: www.geeksforgeeks.org/ introduction-to-recurrent-neural-network.

[44] Understanding of LSTM Networks,2020, Available: www.geeksforgeeks.org/ understanding-of-lstm-networks.

[45] Satapathy, S. K., Dehuri, S., Jagadev, A. K., & Mishra, S. (2019). Introduction. EEG Brain Signal Classification for Epileptic Seizure Disorder Detection,2019, p.13

[46] V. Beal, what is java programming language?2020, Available: www.Webopedia .com/TERM/J/Java.html.

[47] M. Summerfield, "Programming in python 3: A complete introduction to the python language", Edition 3,2010, p.1.

[48] "Eclipse – overview", 2020, Available: www.tutorialspoint.com/eclipse/eclipse_ overview. html

[49] "About keras", 2020, Available: https://keras.io/about.

[50] "About Anaconda",April 2020, Available: www.anaconda.com/media-kit/.

[51] Dean, and al, "TensorFlow: Large-scale machine learning on heterogeneous systems", November 2015, p.2.

[52] ". Fernando Perez,"The IPython notebook: a historical retrospective". Fernando Perez ,2012, Available : http://blog.fperez.org/2012/01/ipython-notebook-historical.html.

[53] Charles R Harris and al,"Array programming with NumPy",2020, p. 357.

[54] "API Overview", 2019, Available: matplotlib.org.

[55] "Package overview: pandas' documentation", 2020, Available: https://pandas.pydata.org/pandas-docs/stable/getting_started/overview.html

[56] "Statistical computations and models for use with SciPy",2019, Available: www.statsmodels.org/stable/

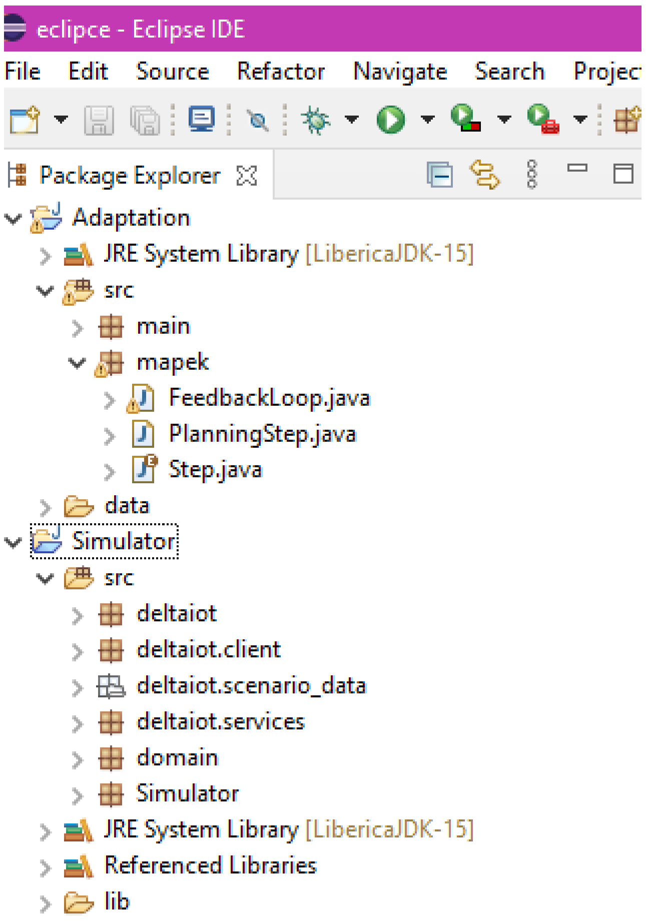
[57]" Support Vector Regression (SVR) using linear and non-linear kernels",2020, Available: scikit-learn.org/stable/auto_examples/svm/plot_svm_regression.html.

[58] Tom Sharp, An Introduction to Support Vector Regression (SVR),2020, Available:

https://towardsdatascience.com/an-introduction-to-support-vector-regression-svr-a3ebc1672c2.

Annex

Annex



```
Adaptation
  JRE System Library [LibericaJDK-15]
  src
    main
    mapek
      FeedbackLoop.java
      PlanningStep.java
      Step.java
  data
Simulator
  src
    deltaiot
    deltaiot.client
    deltaiot.scenario_data
    deltaiot.services
    domain
    Simulator
  JRE System Library [LibericaJDK-15]
  Referenced Libraries
  lib
```

In our project we use two of programming language (java, python).

1) ***Java Part:***

Eclipse was using for running the simulator DeltaIoT.

The file continues the package that define the simulator.

The simulator represents IoT system with 25 Motes and 1 Getaway

2) ***Python Part***

Anaconda Was using for coding python part it helpful in many environments to install like TensorFlow and every environment was describing above.

Appendices

Appendix A: Script for environment.

Appendix B: Script for Dataset Statistics.

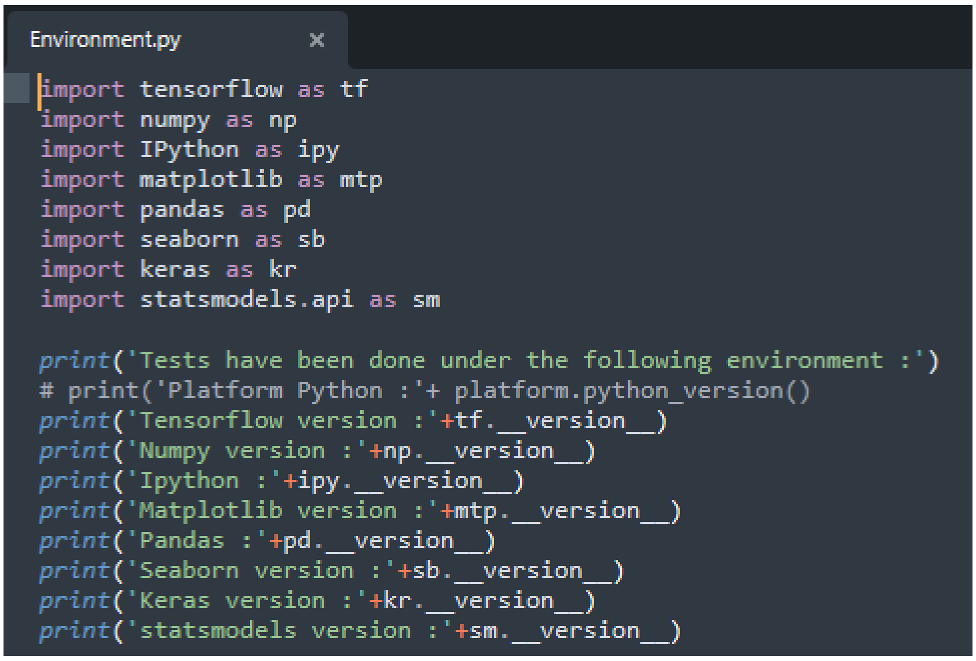
Appendix C: Script for displaying datasets.

Appendix D: Script training data by classification.

Appendix E: Script training data by sequential.

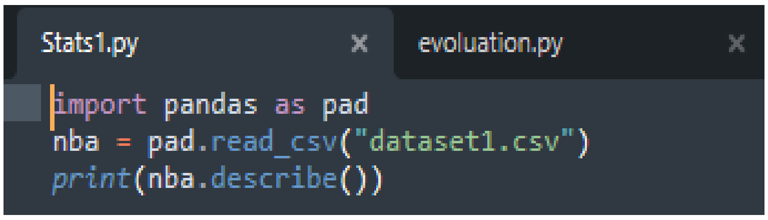
Appendix F: Script for evaluation of accuracy.

➢ **Appendix A: *Environment.py:***


Environment.py

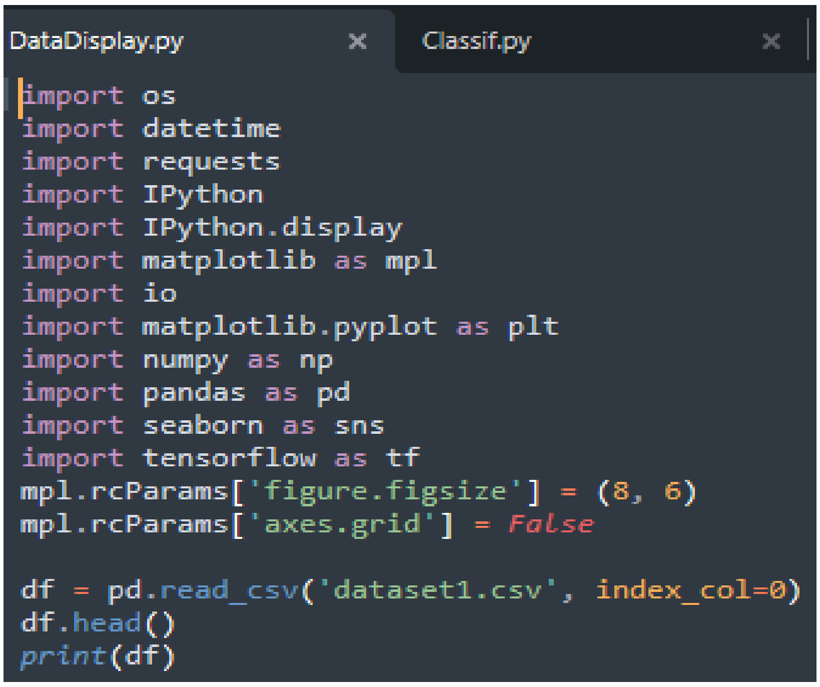
```
import tensorflow as tf
import numpy as np
import IPython as ipy
import matplotlib as mtp
import pandas as pd
import seaborn as sb
import keras as kr
import statsmodels.api as sm

print('Tests have been done under the following environment :')
# print('Platform Python :'+ platform.python_version()
print('Tensorflow version :'+tf.__version__)
print('Numpy version :'+np.__version__)
print('Ipython :'+ipy.__version__)
print('Matplotlib version :'+mtp.__version__)
print('Pandas :'+pd.__version__)
print('Seaborn version :'+sb.__version__)
print('Keras version :'+kr.__version__)
print('statsmodels version :'+sm.__version__)
```

➢ **Appendix B: *Satats1.py:***


Stats1.py evoluation.py

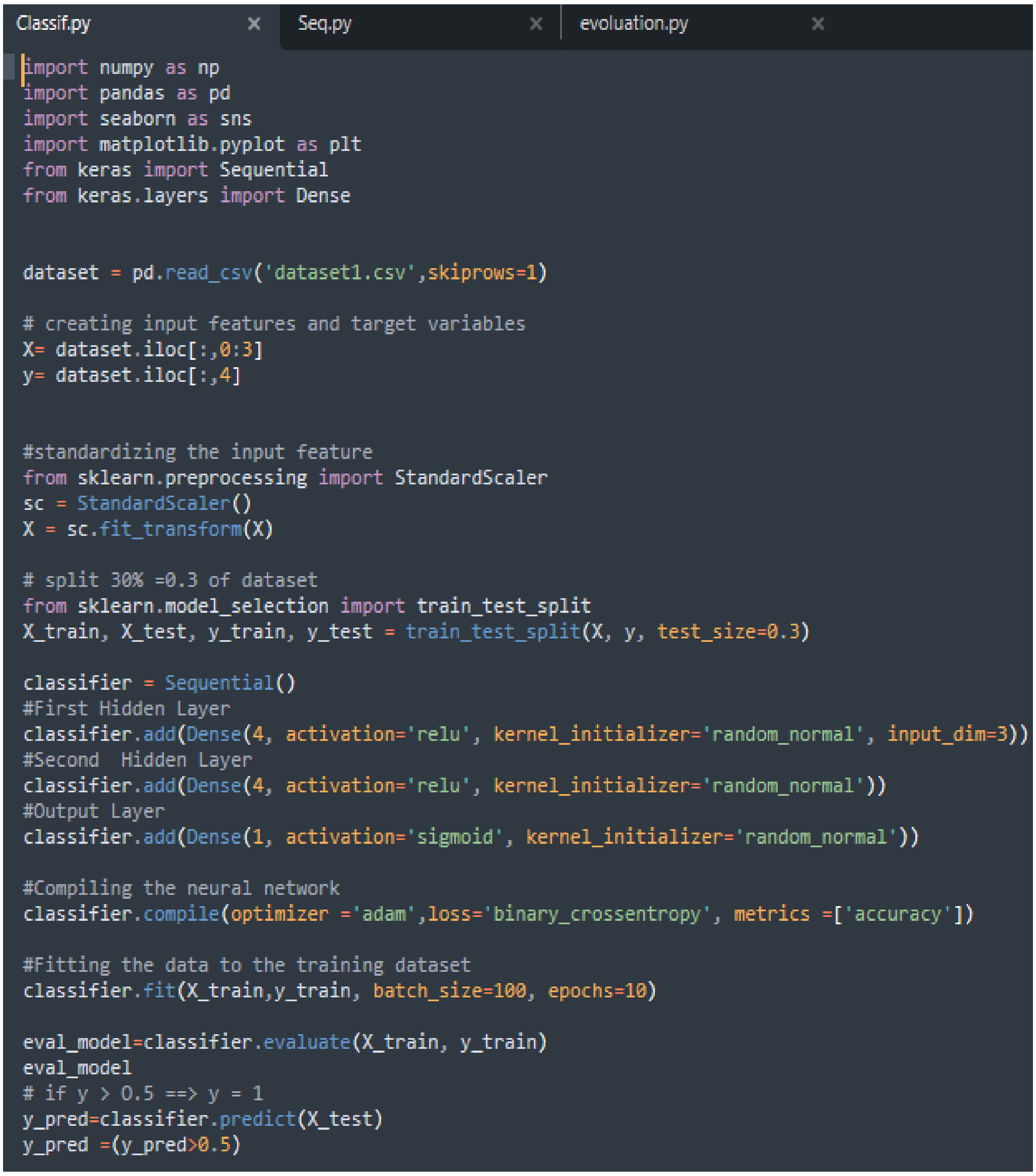
```
import pandas as pad
nba = pad.read_csv("dataset1.csv")
print(nba.describe())
```

➢ **Appendix C**: ***DataDisplay.py:***


DataDisplay.py Classif.py

```
import os
import datetime
import requests
import IPython
import IPython.display
import matplotlib as mpl
import io
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import tensorflow as tf
mpl.rcParams['figure.figsize'] = (8, 6)
mpl.rcParams['axes.grid'] = False

df = pd.read_csv('dataset1.csv', index_col=0)
df.head()
print(df)
```

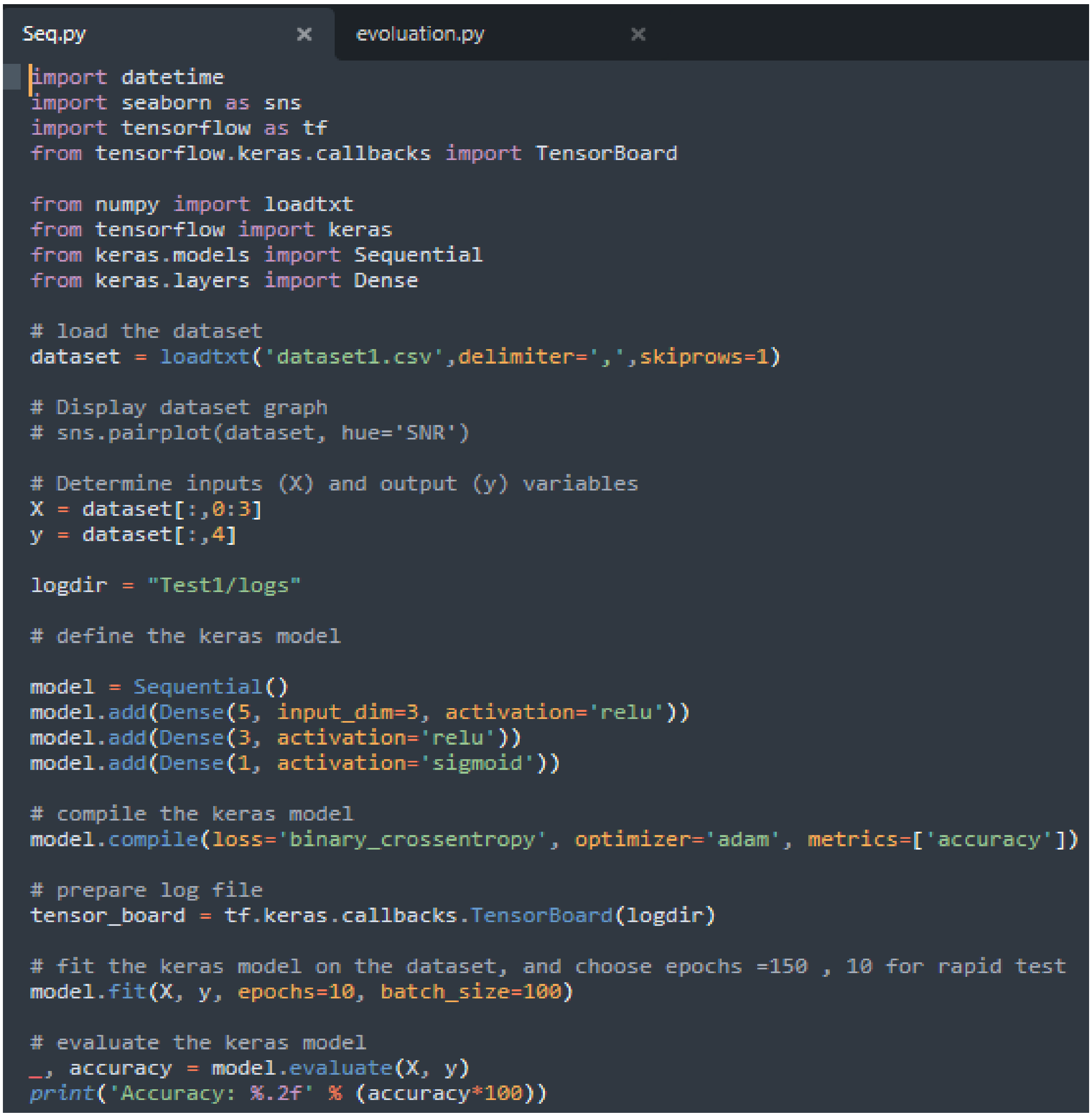
➤ **Appendix D: *Classif.py:***



Classif.py × Seq.py × evoluation.py ×

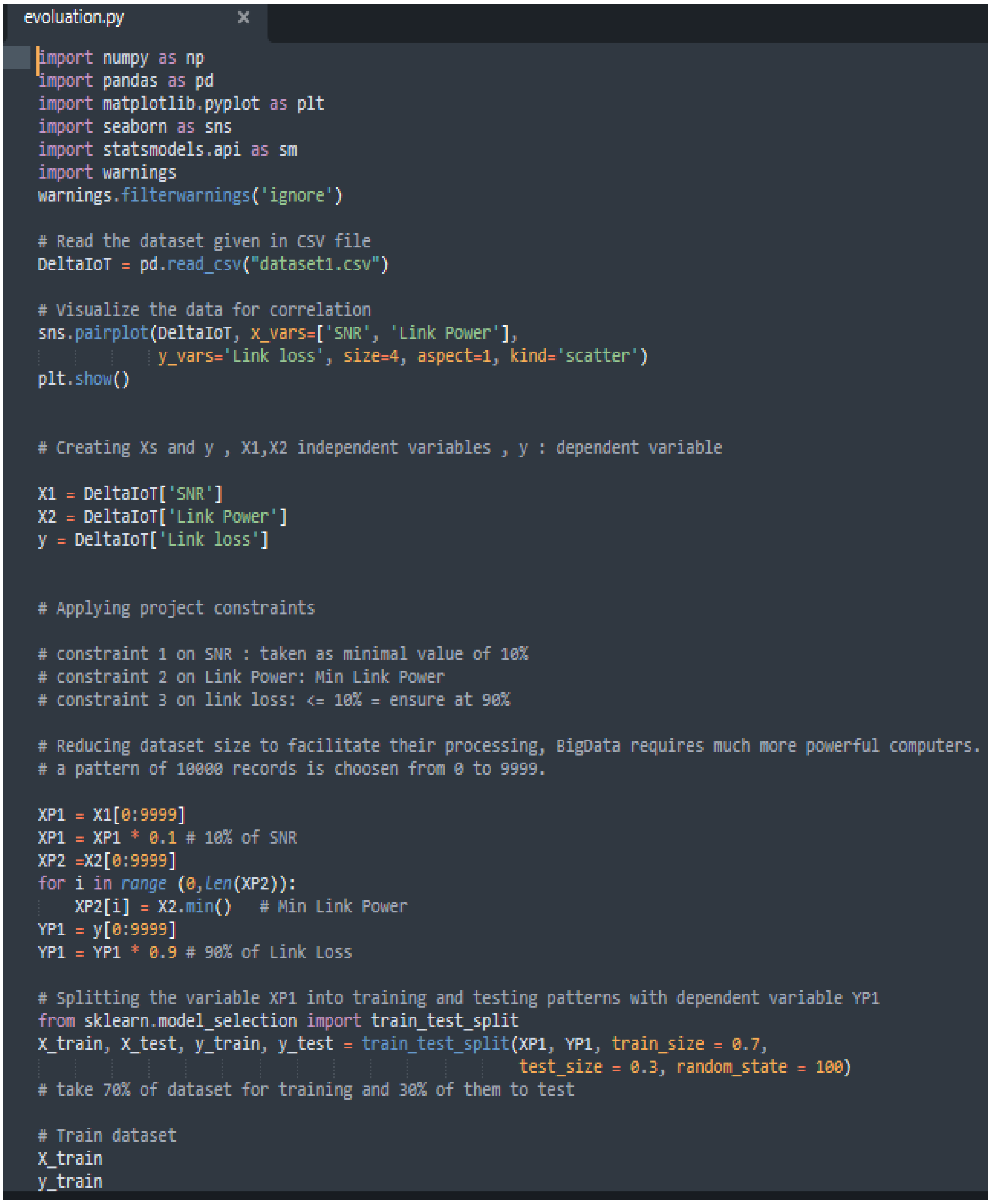
```python
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from keras import Sequential
from keras.layers import Dense


dataset = pd.read_csv('dataset1.csv',skiprows=1)

# creating input features and target variables
X= dataset.iloc[:,0:3]
y= dataset.iloc[:,4]


#standardizing the input feature
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X = sc.fit_transform(X)

# split 30% =0.3 of dataset
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

classifier = Sequential()
#First Hidden Layer
classifier.add(Dense(4, activation='relu', kernel_initializer='random_normal', input_dim=3))
#Second  Hidden Layer
classifier.add(Dense(4, activation='relu', kernel_initializer='random_normal'))
#Output Layer
classifier.add(Dense(1, activation='sigmoid', kernel_initializer='random_normal'))

#Compiling the neural network
classifier.compile(optimizer ='adam',loss='binary_crossentropy', metrics =['accuracy'])

#Fitting the data to the training dataset
classifier.fit(X_train,y_train, batch_size=100, epochs=10)

eval_model=classifier.evaluate(X_train, y_train)
eval_model
# if y > 0.5 ==> y = 1
y_pred=classifier.predict(X_test)
y_pred =(y_pred>0.5)
```

➢ ***Appendix E: Seq.py:***

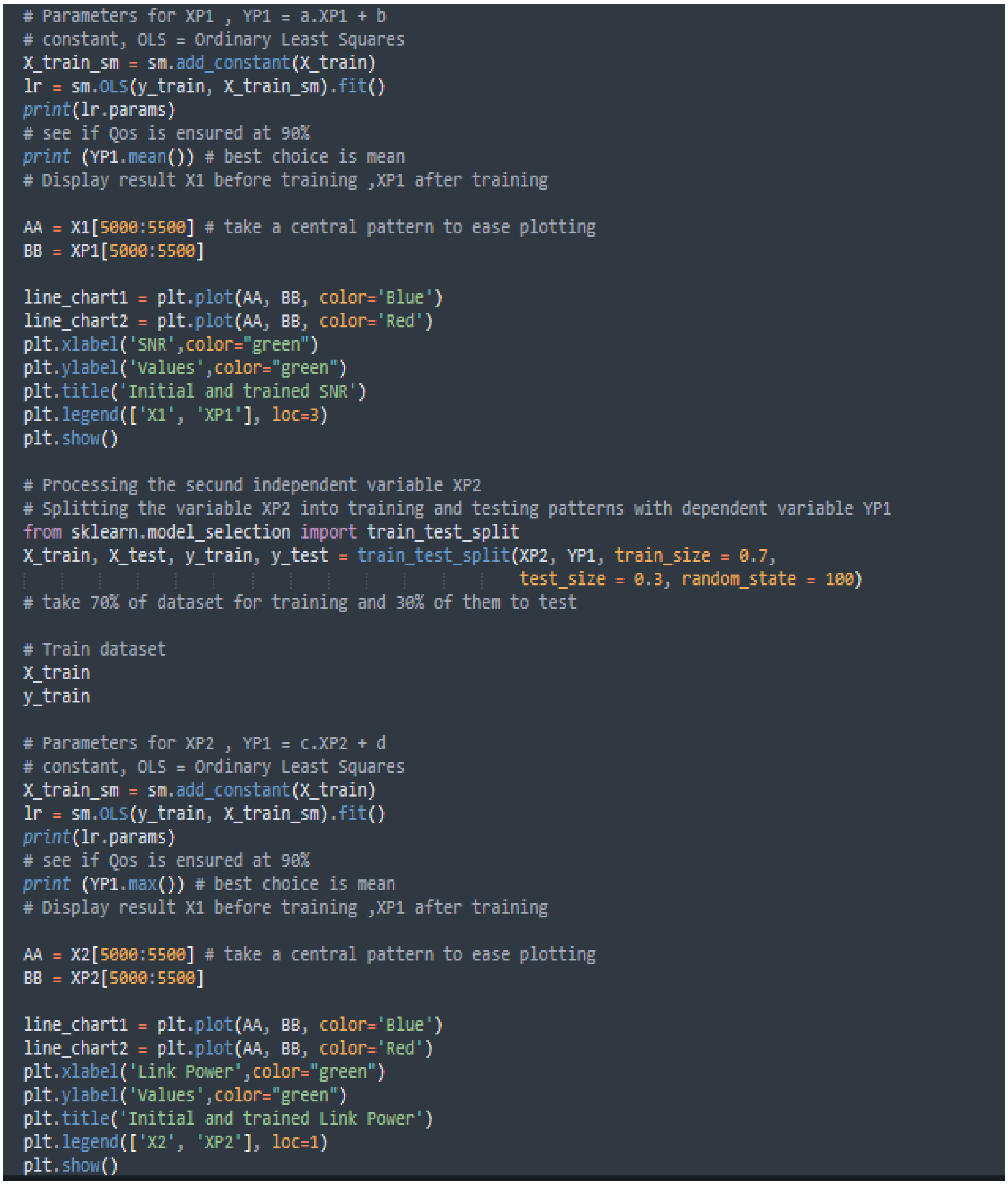


```
import datetime
import seaborn as sns
import tensorflow as tf
from tensorflow.keras.callbacks import TensorBoard

from numpy import loadtxt
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Dense

# load the dataset
dataset = loadtxt('dataset1.csv',delimiter=',',skiprows=1)

# Display dataset graph
# sns.pairplot(dataset, hue='SNR')

# Determine inputs (X) and output (y) variables
X = dataset[:,0:3]
y = dataset[:,4]

logdir = "Test1/logs"

# define the keras model

model = Sequential()
model.add(Dense(5, input_dim=3, activation='relu'))
model.add(Dense(3, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

# compile the keras model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

# prepare log file
tensor_board = tf.keras.callbacks.TensorBoard(logdir)

# fit the keras model on the dataset, and choose epochs =150 , 10 for rapid test
model.fit(X, y, epochs=10, batch_size=100)

# evaluate the keras model
_, accuracy = model.evaluate(X, y)
print('Accuracy: %.2f' % (accuracy*100))
```

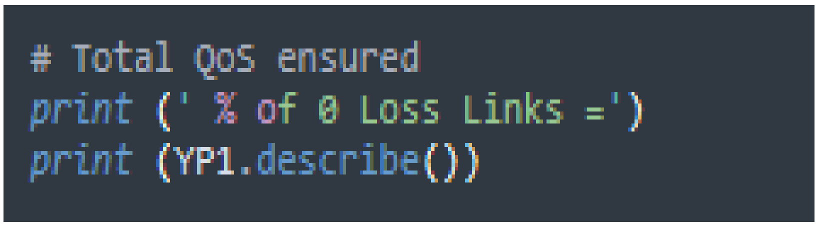
➢ **Appendix F: *Evolution.py:***


evoluation.py

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
import warnings
warnings.filterwarnings('ignore')

# Read the dataset given in CSV file
DeltaIoT = pd.read_csv("dataset1.csv")

# Visualize the data for correlation
sns.pairplot(DeltaIoT, x_vars=['SNR', 'Link Power'],
             y_vars='Link loss', size=4, aspect=1, kind='scatter')
plt.show()


# Creating Xs and y , X1,X2 independent variables , y : dependent variable

X1 = DeltaIoT['SNR']
X2 = DeltaIoT['Link Power']
y = DeltaIoT['Link loss']


# Applying project constraints

# constraint 1 on SNR : taken as minimal value of 10%
# constraint 2 on Link Power: Min Link Power
# constraint 3 on link loss: <= 10% = ensure at 90%

# Reducing dataset size to facilitate their processing, BigData requires much more powerful computers.
# a pattern of 10000 records is choosen from 0 to 9999.

XP1 = X1[0:9999]
XP1 = XP1 * 0.1 # 10% of SNR
XP2 =X2[0:9999]
for i in range (0,len(XP2)):
    XP2[i] = X2.min()   # Min Link Power
YP1 = y[0:9999]
YP1 = YP1 * 0.9 # 90% of Link Loss

# Splitting the variable XP1 into training and testing patterns with dependent variable YP1
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(XP1, YP1, train_size = 0.7,
                                                    test_size = 0.3, random_state = 100)
# take 70% of dataset for training and 30% of them to test

# Train dataset
X_train
y_train
```



```
# Parameters for XP1 , YP1 = a.XP1 + b
# constant, OLS = Ordinary Least Squares
X_train_sm = sm.add_constant(X_train)
lr = sm.OLS(y_train, X_train_sm).fit()
print(lr.params)
# see if Qos is ensured at 90%
print (YP1.mean()) # best choice is mean
# Display result X1 before training ,XP1 after training

AA = X1[5000:5500] # take a central pattern to ease plotting
BB = XP1[5000:5500]

line_chart1 = plt.plot(AA, BB, color='Blue')
line_chart2 = plt.plot(AA, BB, color='Red')
plt.xlabel('SNR',color="green")
plt.ylabel('Values',color="green")
plt.title('Initial and trained SNR')
plt.legend(['X1', 'XP1'], loc=3)
plt.show()

# Processing the secund independent variable XP2
# Splitting the variable XP2 into training and testing patterns with dependent variable YP1
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(XP2, YP1, train_size = 0.7,
                                                    test_size = 0.3, random_state = 100)
# take 70% of dataset for training and 30% of them to test

# Train dataset
X_train
y_train

# Parameters for XP2 , YP1 = c.XP2 + d
# constant, OLS = Ordinary Least Squares
X_train_sm = sm.add_constant(X_train)
lr = sm.OLS(y_train, X_train_sm).fit()
print(lr.params)
# see if Qos is ensured at 90%
print (YP1.max()) # best choice is mean
# Display result X1 before training ,XP1 after training

AA = X2[5000:5500] # take a central pattern to ease plotting
BB = XP2[5000:5500]

line_chart1 = plt.plot(AA, BB, color='Blue')
line_chart2 = plt.plot(AA, BB, color='Red')
plt.xlabel('Link Power',color="green")
plt.ylabel('Values',color="green")
plt.title('Initial and trained Link Power')
plt.legend(['X2', 'XP2'], loc=1)
plt.show()
```



```
# Total QoS ensured
print (' % of 0 Loss Links =')
print (YP1.describe())
```