PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
Ministry of Higher Education and Scientific Research
Mohamed Khider University – BISKRA
Faculty of Exact Sciences, Natural Sciences, and Life
Computer Science department

N° d'ordre : IA01 /M2/2021

Dissertation

Presented to obtain the academic master's degree in

Computer Science

Option : **Artificial intelligence**

A smart predictive approach for telemedicine

By :

BELKIS HASSANI

Defended on June 2021, in front of the jury composed of:

Samir Bourekkache	**MCA**	**President**
OKBA KAZAR	**Prof**	**Supervisor**
Toufik Kalfali	**MAA**	**Examiner**

Acknowledgements

I would like to express my sincere gratitude to my supervisor **“OKBA KAZAR”** for the continuous support of this work, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research.
Besides my supervisor, I would like to thank the members of the discussion committee for their kind reading of this note and for the notes that have helped to promote and redress this work, And all those who have helped me do this work, And thanks to all the professors of the computer science department.

BELKIS HASSANI

Dedication

Praise be to God (Alhamdulillah), who has enabled me to accomplish this humble work.

I dedicate this work to :

My dear parents, may God protect them **Djamel & Salima**.

My dear sisters **Fatima Zahra**, **Kaouther**,and **Kalthoum**.

To the new friend **Djihene Houfani**, who was supportive of this work.

To The Batch Of **2020/2021**.

BELKIS HASSANI

Abstract

The globe is currently confronting numerous issues brought about by a new virus known as COVID-19. however, Telemedicine is experiencing an unprecedented boom, especially with the evolution of artificial intelligence techniques like deep learning, and fuzzy logic. In this work, we proposed and developed a method for predicting whether or not a person is infected with COVID-19 disease.

Keywords : COVID-19, Coronavirus, Artificial Intelligence, Deep learning,Telemedcine.

الملخص

يواجه العالم حاليًا العديد من المشكلات الناجمة عن فيروس جديد يُعرف باسم COVID-19. ومع ذلك ، يشهد الطب عن بُعد طفرة غير مسبوقة ، خاصة مع تطور تقنيات الذكاء الاصطناعي مثل التعلم العميق والمنطق الضبابي.

من خلال هذا العمل ، قمنا بإقتراح وتطوير طريقة للتنبؤ بما إذا كان الشخص مصابًا بفيروس كورونا أم لا.

الكلمات المفتاحية: COVID-19، فيروس كورونا ، الذكاء الاصطناعي التعلم العميق ، التطبيب عن بعد.

Résumé

Le monde est actuellement confronté à de nombreux problèmes provoqués par un nouveau virus connu sous le nom de COVID-19. Cependant, la télémédecine connaît un essor sans précédent, en particulier avec l'évolution des techniques d'intelligence artificielle comme l'apprentissage en profondeur et la logique floue.
Dans ce travail, nous avons proposé et développé une méthode pour prédire si une personne est infectée ou non par la maladie COVID-19.

Mots clés : COVID-19, coronavirus, Intelligence Artificielle, Deep learning, Télémédecine.

Table of Contents

List of Figures

List of Tables

General Introduction

Work context

With the advances, we are seeing in modern technologies and uses in all sectors and the health sector in particular, where he became artificial intelligence techniques with telemedicine are the keys to new therapeutic advances.

Telemedicine is a medical practice using technology and communication information to helps determine a diagnosis, or to keep track of a patient's health...etc.

COVID-19 disease is one of the most common in the present world. This stimulates more interest researchers for the design of systems such as deep learning models for predicting this disease.

Problematic and objectives

In response to the COVID-19 outbreak, the research on telehealth is a promising and very interesting field, and that may come with a challenge that we will try to get into in this work and try to provide a system for doing COVID-19 prediction with high performance and accuracy to get better results.

The following is a list of our work's objectives.

- Study the concept of telemedicine, its importance in real-world applications.
- Study Deep learning and Fuzzy logic techniques and see how we can use them for COVID-19 prediction.
- Propose a system for COVID-19 prediction, with the hybrid method.

Structure of the master project

The following is the structure of the dissertation:

The First chapter« State of the art on predictive telemedicine »: This chapter begins by introducing the notion of telemedicine, its various components, and its significance in the modern world.

The Second chapter« State of the art on Deep learning and Fuzzy logic based telemedicine systems »: In this chapter, we will present what is Deep Learning also Fuzzy logic, how they work, and we will present some related work in telemedicine and the detection of diseases and comparison between them.

The Third chapter « System design »: We'll construct an architecture for the COVID-19 prediction system, which will include all of its different elements and functionalities.

The Fourth chapter « Implementation and Results »: This part describes the software environment in which the system will be built, as well as the specifics of our application's implementation. Based on the outcome, we will also evaluate the performance and accuracy of our system.

Chapter 1

State of the art on predictive telemedicine

1.1 Introduction

Individuals and cultures are changing as a result of advances in information and communication technologies (ICTs), this technology is shaping more and more the healthcare domain. ICTs are used to improve and deliver health services and related information. The delivery of health care through e-health systems should not normally have negative, harmful or disadvantageous effects.

The main objective of this chapter is to clarify the world of telemedicine as well as to situate a fairly broad research context through its applications.

1.2 Telemedicine

Telemedicine is an umbrella term that encompasses any medical activity involving an element of distance. In its commonly understood sense, in which a doctor. patient interaction involves telecommunication, it goes back at least to the use of ship to shore radio for giving medical advice to sea captains [1].

Tele-distance medicine or health is the use of information and communication technologies (ICTs) to deliver health services where there is physical separation between care providers and or the recipients over both long and short distances [2].

1.3 E-health

Electronic Health, or 'eHealth', is the term used to describe interactions with health services that can be performed using computer-based communication technologies. It evolved from telemedicine and tele-health where telecommunication is the delivery method for health care [3].

E-Health is an emerging field in the intersection of medical informatics, public health and business, referring to health services and information delivered or enhanced through the Internet and related technologies. In a broader sense, the term characterizes not only a technical development, but also a state-of-mind, a way of thinking, an attitude, and a commitment for networked, global thinking, to improve health care locally, regionally, and worldwide by using information and communication technology [4].

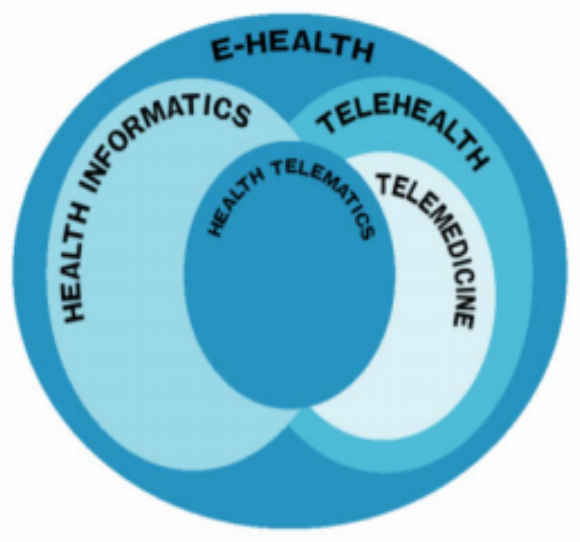

Figure 1.1: Terminologies associated with telemedicine [5]

1.4 History of telemedicine [6]

- In 1876, Alexander Graham Bell patents the telephone. A new era of telecommunications begins .
- In 1924, teledactlyl is tool that would allow doctors to see their patients through a view-screen, this idea was invented by Dr.Hugo Gernsback and printed in science and invention.
- In 1948, The first radio-logic images are sent via telephone across 24 miles in Pennsylvania .
- In 1959, Nebraska Psychiatric institute and Norfolk State Hospital establish a closed-circuit television link for psychiatric consultation .
- In 1960, NASA researches ways to incorporate telemedicine in order to provide healthcare to astronauts.
- In 1964, AT & T releases the Picture-phone, adding video to telephone calls.
- In 1973, Space Technology Applied Care (STARPAHC) brings NASA and the Indian Health Services together to the Papago India Reservation in Arizona
- In 1989, The World Wide Web is invented and expands the capabilities of telemedicine.
- In 1993, American Telemedicine Association (ATA) is created as a non-profit and has grown to be the leading telehealth association .
- In 1996, HIPAA is signed into law that provides data privacy and security provisions for safeguarding medical information
- In 1999, Medicare starts paying for telehealth consultations for patients who live in rural areas .
- In 2003, Skype was released bringing video-chat to the mainstream.
- In 2009, ARRA and HITECH Act is created to motivate the implementation of electronic health records and supporting technology in the U.S.

- In 2014, Doxy.me released to the public.Providers can now securely videochat with their patients.

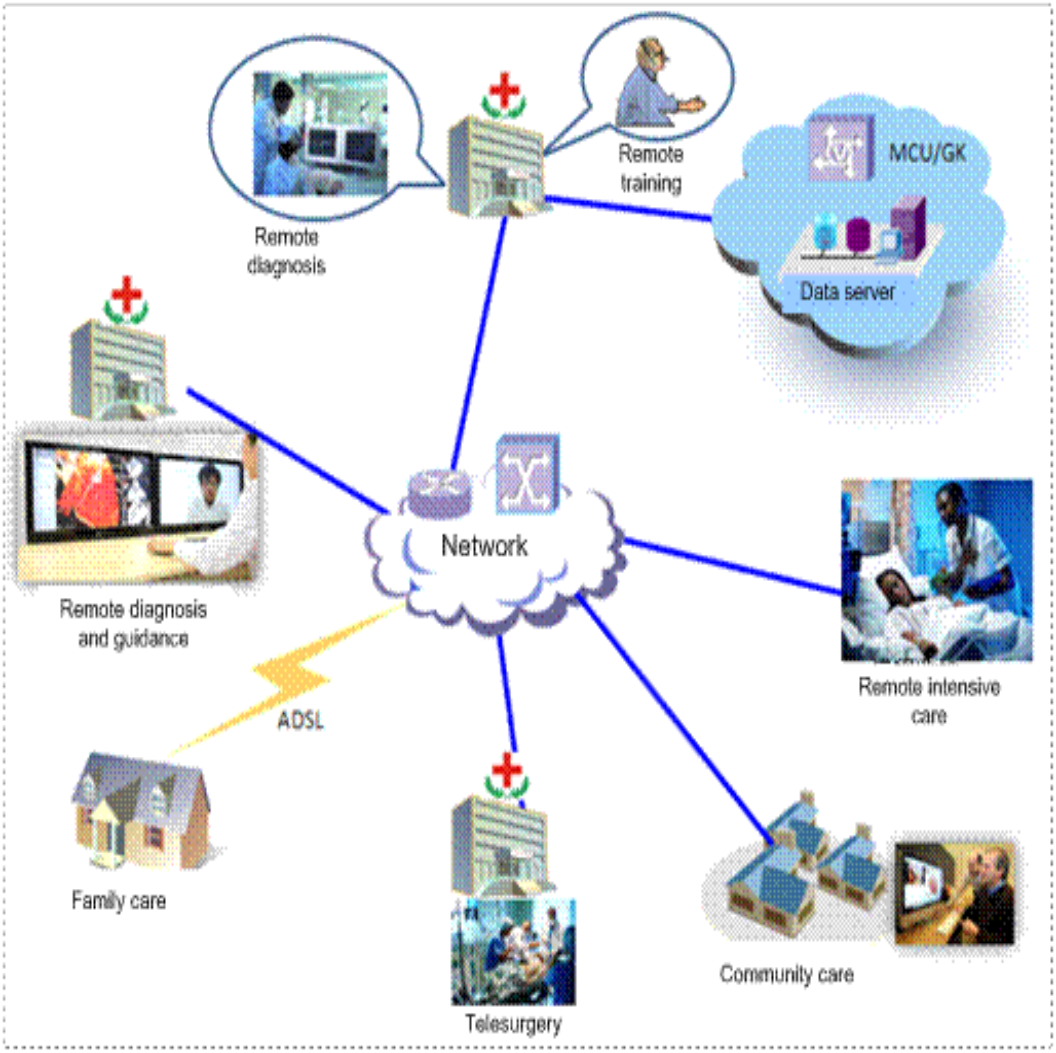

Figure 1.2: Telemedicine Architecture [7]

1.5 Telemedicine Types [2]

There are four main types of telemedicine, namely: Real time (synchronous), Store and forward (S&F) (asynchronous), Telemonitoring or remote monitoring, and Mobile health (Mhealth).

1.5.1 Real time (synchronous)

Live data or information is exchanged. One of the most common examples is video conferencing between a patient and a healthcare provider. Others include real-time ultrasound or angiogram viewing, broadcasting of operations from the operating room, and tele-stethoscope monitoring of heart sounds. This is a simple and convenient method of telemedicine, but requires high bandwidth, constant connectivity and investment in related hardware.

1.5.2 Store and forward (S&F) (asynchronous)

Data is gathered and transmitted. Depending on when access is available, it can be saved locally or on a cloud. The data can be viewed, commented on, and even incorporated into a separate server at the discretion of the other telehealth stream members. It is less reliant on continuous connectivity, but it is more difficult to handle. Choice of software and interconnectivity standards has a greater role as interpretations may differ.

1.5.3 Telemonitoring or remote monitoring

Medical devices collect and process personal data, which they send to clinicians in real time or in a stored summary form (asynchronously). Home care devices for the elderly and infirm, as well as tele-ICU, are examples.

1.5.4 Mobile health (Mhealth)

Is a special kind of digital health. Smart phones have more processing power and communication than prior specialized telemedicine services. They are low-cost, have built-in audio and video, and are versatile enough to enable real-time, store-and-forward transmission as well as streaming from virtually anywhere.

Telemonitoring via built-in or add-on sensors enables a single system to serve as a full

telehealth solution for a variety of issues. Many advanced software or smartphones may provide direct health information to patients.

1.6 Benefits of Telemedicine

The main advantages of telemedicine can be summarized as follows:

- Saving of cost and effort.
- Underserved rural, semi-urban,and remote communities provide access to specialist health care services facilities.
- Easy access to specialist doctors without the need for the doctor or patient to move.
- Remote surgery.
- Ease of monitoring the patient at home and rapid transmission of information to the carer.
- Assists in addressing shortages of HealthCare providers.

Without forgetting the benefits of telemedicine during the spread of Coronavirus (COVID-19), we mention the most important of them :

- The stay-at-home goal is met during the pandemic, protecting uninfected patients, clinicians, and the community.
- People do not flinch to the Coronavirus screening centres (COVID-19) and only the sick people are directed to these centers.
- It helps doctors in emergency rooms focus on the patients who really need them.
- Patients diagnosed with the new Coronavirus (Covid-19) who have been isolated benefit from obtaining additional information about the disease and inquiring if they experience any symptoms, as they are contacted using video calls.

1.7 What is Coronavirus?

Coronaviruses (CoVs) belong to the family of Coronaviridae, the order Nidovirales, and thegenus Coronavirus. They are the major group of viruses causing respiratory andgastrointestinal illnesses. CoVs are enclosed viruses that contain a single-stranded, non-segmented positive-sense ribonucleic acid (RNA) virus [8].

Coranaviruses are a family of viruses that infect mammals and birds. Coronaviruses produce infections of the respiratory system in humans that are often mild, such as the common cold. Coronaviruses are a zoonotic virus family that spreads from animals to humans.COVID-19 is the new name for the sickness. “COVI” stands for coronavirus, “D” is for disease, and “19” is for the year it was discovered [9].

Coronavirus disease 2019 (COVID-19) is defined as illness caused by a novel coronavirus called severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2; formerly called 2019-nCoV), which was first identified amid an outbreak of respiratory illness cases in Wuhan City, Hubei Province, China [10].

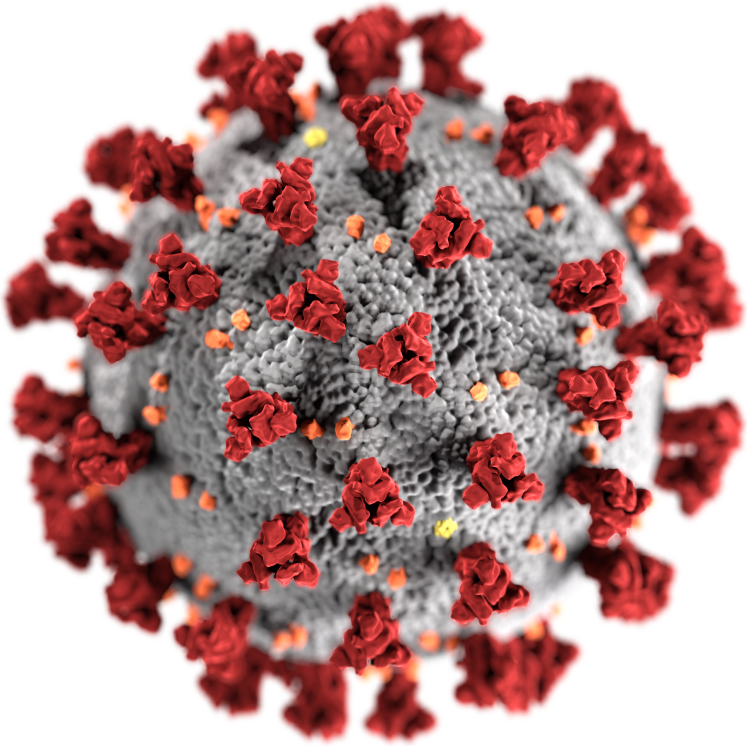

Figure 1.3: Coronaviruses / COVID-19 [11]

1.7.1 History [8]

- In 1962, CoVs have been described as novel respiratory tract viruses, which were discovered in samples taken from persons who had signs and symptoms of a respiratory tract illness.
- In 2002, The first cases of SARS were discovered in the Chinese province of Guangdong .
- In December 2019, a group of pneumonia patients in Wuhan, the capital of China's Hubei province, were found to be infected with a novel CoV (nCoV).
- The newly discovered virus, which causes a similar infection (SARS-CoV), was first dubbed nCoV 2019 (January 2020) and then COVID-19 (February 2020).
- As of February 12, 2020, a total of 43,103 cases of infection and 1,018 deaths havebeen recorded . Thousands of human infections have been confirmed in China, along withmany exported cases throughout the globe .

1.7.2 Signs and symptoms [9]

COVID-19 affects different people in different ways. Most infected people will develop mild to moderate illness and recover without hospitalization.
The following are the most common symptoms:

- Fever.
- Dry cough.
- Tiredness.

Symptoms that are less common:

- Aches and pains.
- Sore throat.

- Diarrhoea.
- Conjunctivitis.
- Headache.
- Loss of taste or smell.
- A rash on skin, or discolouration of fingers or toes.

Symptoms that are serious:

- difficulty breathing or shortness of breath.
- chest pain or pressure.
- loss of speech or movement.

It takes 5–6 days on average for symptoms to appear when a person is infected with the virus, but it can take up to 14 days.

1.7.3 Diagnosis

COVID-19 can be diagnosed temporarily based on symptoms, and this may be validated by a variety of methods, as shown in the table below.

Method available	Working principle	Advantage	Time required	Disadvantage
Next generation sequencing (NGS)	Whole genome sequencing	Highly sensitive and specific, Provide all related information; Can identify novel strain	1–2 day	High expertise Equipment dependency and high cost Highly sophisticated Lab required
RT-PCR	Specific primer-probe based detection	Fast results Higher sensitivity Needs small amount of DNA Can be performed in a single step Well established methodology in viral diagnostics	3–4 h	Higher costs due to the use of expensive consumables Expensive lab equipment Detection is also complex and time consuming
LAMP	More than two sets of specific primers pair based detection	Highly repeatable and accurate Single working temperature	1 h	Too sensitive, highly prone to false positives due to carry-over or cross-contamination
Serological (traditional)	Antigen/Antibodies IgG/ IgM	Sensitive and specific	4–6 h	Testing come after 3-4 days of infection False positive
Rapid serological	Antigen/Antibodies IgG/ IgM	POCT	15–30 min	Testing come after 3-4 days of infection False positive
CT scan	Chest images	Enhance sensitivity of detection if findings combined with RT-PCR results	1 h	Indistinguishability from other viral pneumonia and the hysteresis of abnormal CT
Virus isolation	In vitro live virus isolation and propagation	Highly (100%) specific Gold standard	5–15 days	Low sensitivity as isolation is not 100%

Figure 1.4: Diagnosis method available for COVID-19 [12]

1.8 Conclusion

In this chapter, we have presented the topic of Telemedicine by giving its definition, its types, its history, and its benefits. We have also defined the E-health term.
Recently, a new virus (COVID-19) has emerged that can cause fatal health problems for humans. And this is what the statistical data show by revealing the percentage of deaths due to the Coronavirus all over the world, and therefore there is an implicit necessity to predict the virus as soon as possible, which is the goal of our project.
The next chapter is devoted to an overview of the methods and the prediction algorithms used.

Chapter 2

State of the art on Deep learning & Fuzzy logic based telemedicine systems

2.1 Introduction

Over the past decade, artificial intelligence (AI) technologies have gained increasing importance in a variety of fields. Artificial Intelligence (AI) technologies include fuzzy systems, Artificial Neural Network (ANN) systems, genetic programming, etc. So we decided to use a hybrid approach i.e. fuzzy logic and deep learning to produce a more effective system.

The main objective of this chapter is to illustrate deep learning and fuzzy logic by learning the most common steps and methods of building a system based on both.

2.2 Deep learning

Deep learning (DL) is a particular subset of ML methodologies using artificial neural networks (ANN) slightly inspired by the structure of neurons located in the human brain Informally, the word deep refers to the presence of many layers in the artificial neural network .

DL is a real tsunami for machine learning in that a relatively small number of clever methodologies have been very successfully applied to so many different domains (image, text, video, speech, and vision), significantly improving previous state-of-the-art results achieved over dozens of years. The success of DL is also due to the availability of more training data (such as ImageNet for images) and the relatively low-cost availability of GPUs for very efficient numerical computation. Google, Microsoft, Amazon, Apple, Facebook, and many others use those deep learning techniques every day for analyzing massive amounts of data [13].

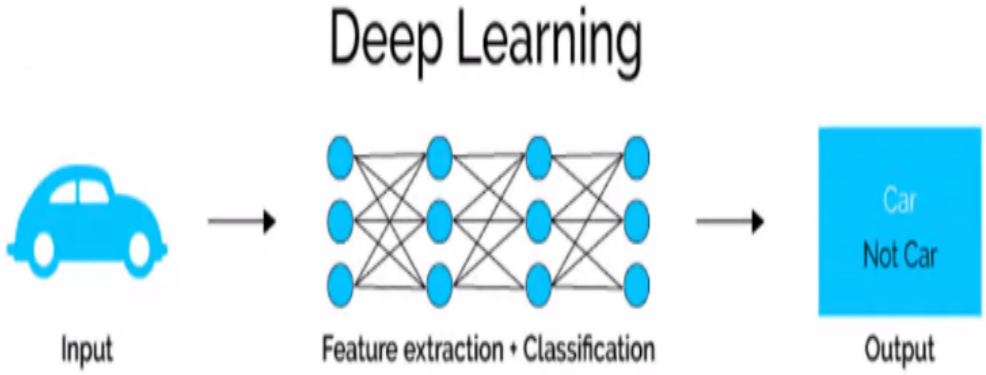

Figure 2.1: Deep learning

2.2.1 Artificial Neural Networks

Artificial neural networks (ANNs) are learning models inspired by biological neural networks that approximate functions that depend on a large number of inputs (features or data representation). Deep neural networks (DNNs) are ANNs with multiple hidden layers between the input and output layers. Thus, from a given input, they are able to learn features (hidden layers) and to give a classification result (output layer). They have been very successful in NLP for part-of-speech tagging, chunking, named entity recognition, and semantic role labeling . DNNs are good examples of DL and are the focus of this paper. In the remainder of this section, we introduce the topology of conventional neural networks and their initialization [14].

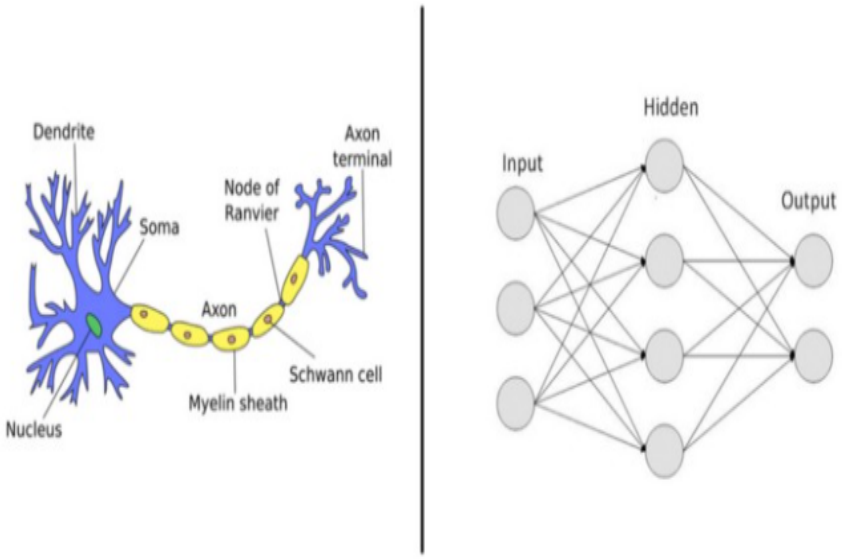

Figure 2.2: Artificial neural networks and biological neural networks [15]

2.2.2 Three Classes of Deep Learning Networks [16]

Deep learning is a broad category of machine learning techniques and architectures that are distinguished by the use of several layers of non-linear information processing that are hierarchical in nature. Most works in this field can be divided into three categories:

2.2.2.1 Deep networks for unsupervised or generative learning

which are intended to capture high-order correlation of the observed or visible data for pattern analysis or synthesis purposes when no information about target class labels is available. Unsupervised feature or representation learning in the literature refers to this category of deep networks. When used in the generative mode, may also be intended to characterize joint statistical distributions of the visible data and their associated classes when available and being treated as part of the visible data. In the latter case, the use of the Bayes rule can turn this type of generative network into a discriminative one for learning.

2.2.2.2 Deep networks for supervised learning

which are intended to directly provide discriminative power for pattern classification purposes, often by characterizing the posterior distributions of classes conditioned on the visible data. Target label data are always available in direct or indirect forms for such supervised learning. They are also called discriminative deep networks.

2.2.2.3 Hybrid deep networks

where the goal is discrimination which is assisted, often in a significant way, with the outcomes of generative or unsupervised deep networks. This can be accomplished by better optimization or/and regularization of the deep networks in category (2). The goal can also be accomplished when discriminative criteria for supervised learning are used to estimate the parameters in any of the deep generative or unsupervised deep networks in category (1) above.

2.2.3 Deep Learning Categories [17]

In this part, An up-to-date overview will be presented for each of the three main categories of neural networks, Attention , Convolutional Neural Networks, Pretrained Unspervised Networks, and Recurrent/Recursive Neural Networks.

2.2.3.1 Convolutional Neural Networks (CNNs)

CNNs are inspired by biological processes and are designed to mimic the neural connectivity found in the brain's visual cortex. They require considerably fewer data preprocessing compared to traditional image classification algorithms which require handengineered pre-processing filters. CNNs have a large range of applications in image and video recognition, recommender systems, image classification, medical image analysis, and natural language processing (NLP).

2.2.3.2 Pretrained Unsupervised Networks

Since we typically have minimal training data, data generation and feature extraction are critical applications in deep learning.To provide a larger dataset on which to train the network, various techniques are used to supplement the initial dataset. Using advanced deep learning architectures such as Generative Adversarial Networks (GANs) and Autoencoders, the researchers were able to solve the issue.To boost model learning, we may generate synthetic data based on the original dataset.

2.2.3.3 Recurrent and Recursive Neural Networks

This subcategory of deep learning structures can submit data in time stages. This class introduces four structures: Recurrent neural networks, Recursive neural networks, Long short-term memory (LSTM), and Attention .

Recurrent Neural Network (RNN), Is a deep learning subclass . The ability of RNNs to process and extract insights from sequential data is lauded. As a result, recurrent neural networks are used in video analysis, image captioning, natural language processing (NLP), and music analysis. Unlike traditional neural networks, which presume data points are independent.

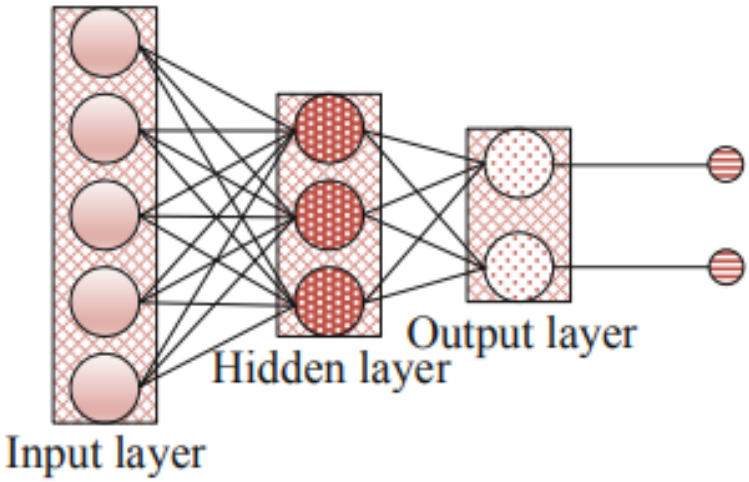

Figure 2.3: Recurrent Neural Network architecture [18]

Recursive Neural Network, are a class of non-linear adaptive models for dealing with variable-length data. They excel at processing data structure inputs in particular. In what can be interpreted as a loop, recursive networks feed the state of the network back into itself.Recursive neural networks are generally known for having a bottom-up feed-forward method and top-down propagation method. Both mechanisms are used in most recursive networks to propagate information via structure.

Long short-term memory (LSTM), is the most common RNN architecture that remembers values over arbitrary intervals. It was first introduced in 1997 by Hochreiter and Schmidhuber and works well on making predictions based on time series data, avoiding the long-term dependency problem that traditional, or vanilla, RNNs were plagued with.LSTM is also well suited to classification and processing tasks and can be found in the Google Translate, Apple Siri, and Amazon Alexa applications.

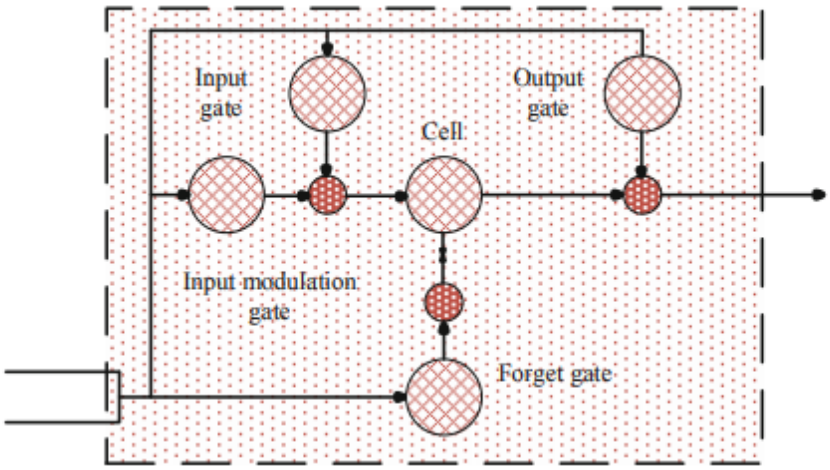

Figure 2.4: Long short-term memory [18]

Attention, Most contemporary neural network architectures utilize recurrence and convolution mechanisms along with an encoder-decoder configuration. Attention networks use an additional “attention” mechanism that is growing in popularity among numerous architectures. Attention can be thought of similarly to how we focus our attention on the task at hand. For example, if you are asked to fix paint a room, you put your attention to the area of the room you are currently painting. If you are asked to fix a damaged vehicle, then your attention is on the part of the vehicle you are currently working on. Attention networks apply the same concept by focusing on specific areas at different

time steps.

2.3 Fuzzy logic

Fuzzy logic is concerned with the formal principles of approximate reasoning, with precise reasoning viewed as a limiting case. In more specific terms, what is central about fuzzy logic is that, unlike classical logical systems, it aims at modeling the imprecise modes of reasoning that play an essential role in the remarkable human ability to make rational decisions in an environment of uncertainty and imprecision. This ability depends, in turn, on our ability to infer an approximate answer to a question based on a store of knowledge that is inexact, incomplete, or not totally reliable [19].

Fuzzy Logic is an approach to variable processing that allows for multiple values to be processed through the same variable. Fuzzy logic attempts to solve problems with an open, imprecise spectrum of data that makes it possible to obtain an array of accurate conclusions. Fuzzy logic is designed to solve problems by considering all available information and making the best possible decision given the input [20].

2.3.1 Why Fuzzy Logic?

The generality of fuzzy logic is much greater than that of bivalent logic. It is the generality of fuzzy logic that underlies much of what fuzzy logic has to offer. The following are some of fuzzy logic's most significant contributions:

FL-generalization : Any bivalent-logic-based theory, can be FL-generalized, and thus upgraded, by adding fuzzy logic concepts and techniques to T. Examples: fuzzy control, fuzzy linear programming, fuzzy probability theory and fuzzy topology.

Linguistic variables and fuzzy if–then rules : In effect, the formalism of linguistic variables and fuzzy if–then rules is a strong modeling language that is commonly used in fuzzy logic applications. Essentially, the formalism is used to summarize and compact

knowledge with the use of granulation.

Cointensive precisiation: Fuzzy logic has a high precision cointensive strength. This power is needed for the cointensive formalization of human-centric fields such as economics, linguistics, law, conflict resolution, psychology, and medicine.

NL-Computation (computing with words): The foundation for NL-Computation is fuzzy logic,that is, computation based on natural language knowledge. NL-Computation is of direct relevance to mechanization of natural language understanding and computation with imprecise probabilities. More generally, NL-Computation is needed for dealing with second-order uncertainty, that is, uncertainty about uncertainty, or uncertainty for short [21].

2.3.2 Fuzzy Logic Systems Architecture

It is divided into four main sections, as shown in figure (2.5)

RULE BASE: It contains the set of rules and the IF-THEN conditions provided by the experts to govern the decision making system, on the basis of linguistic information.Recent advances in fuzzy theory have resulted in a number of useful methods for designing and tuning fuzzy controllers.The majority of these advancements minimize the number of fuzzy rules.

FUZZIFICATION: This is a technique for transforming inputs, such as crisp numbers, into fuzzy sets. Crisp inputs, such as temperature, pressure, and rpm's, are the same inputs measured by sensors and passed into the control system for processing.

INFERENCE ENGINE: It defines the degree of matching between the current fuzzy input and each rule, as well as which rules should be fired based on the input field. Next, the fired rules are combined to form the control actions.

DEFUZZIFICATION: This step converts the fuzzy sets created by the inference en-

gine into a crisp value. There are several defuzzification methods, and the best one is used in conjunction with a specialized expert to reduce error [22].

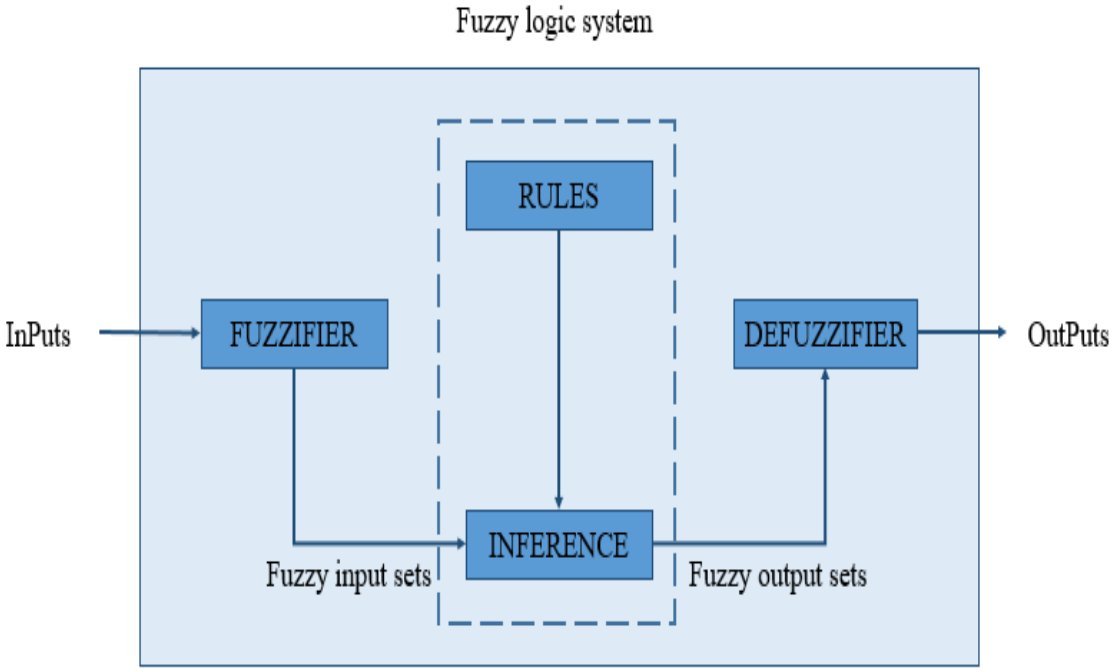

Figure 2.5: Fuzzy Logic Systems Architecture [23]

2.3.3 Application domains [24]

Fuzzy set theory has already found applications in a variety of fields. Those applications will most likely be classified as follows:

- Applications to mathematics, that is, generalizations of traditional mathematics such as topology, graph theory, algebra, logic, and so on.
- Applications to algorithms such as clustering methods, control algorithms, mathematical programming, and so on.
- Applications to standard models such as "the transportation model," "inventory control models," "maintenance models," and so on.

- Applications to real-world problems of different kinds.

2.3.4 Fuzzy Logic Theory [23]

The fuzzy logic theory is extremely adaptable, and there is no uniform formalism rule that can be applied to it. Depending on the type of application, multiple ways for associating components to a certain set might be used.[25]

2.3.4.1 Crisp Sets

Recall that a crisp set A in a universe of discourse U (which provides the set of allowable values for a variable) can be defined by listing all of its members or by identifying the element $x \subset A$. One way to do the latter IS to specify a condition by which$x \subset A$;thus A can be defined as $A = \{\ x|\ x\ meets\ some\ condition\}$. Alternatively, we can introduce a zero-one membership function (also called a characteristic function, discrimination function, or indicator function) for A,denoted $\mu A\ (x)$,), such that $A\ \Rightarrow\ \mu A\ (x) =\ 1$ IF $x\ \in\ A$ AND $A\ \Rightarrow\ \mu A\ (x) =\ 0$ IF $x\ \notin\ A$. Subset A is mathematically equivalent to its membership function $\mu A\ (x)$ in the sense that knowing $\mu A\ (x)$ is the same as knowing A itself.

2.3.4.2 Fuzzy Sets

A fuzzy set F defined on a universe of discourse U is characterized by a membership function $\mu A\ (x)$ which takes on values in the interval [0, 1]. A fuzzy set is a generalization of an ordinary subset (i.e., a crisp subset) whose membership function only takes on two values, zero or unity. A membership function provides a measure of the degree of similarity of an element in U to the fuzzy subset.

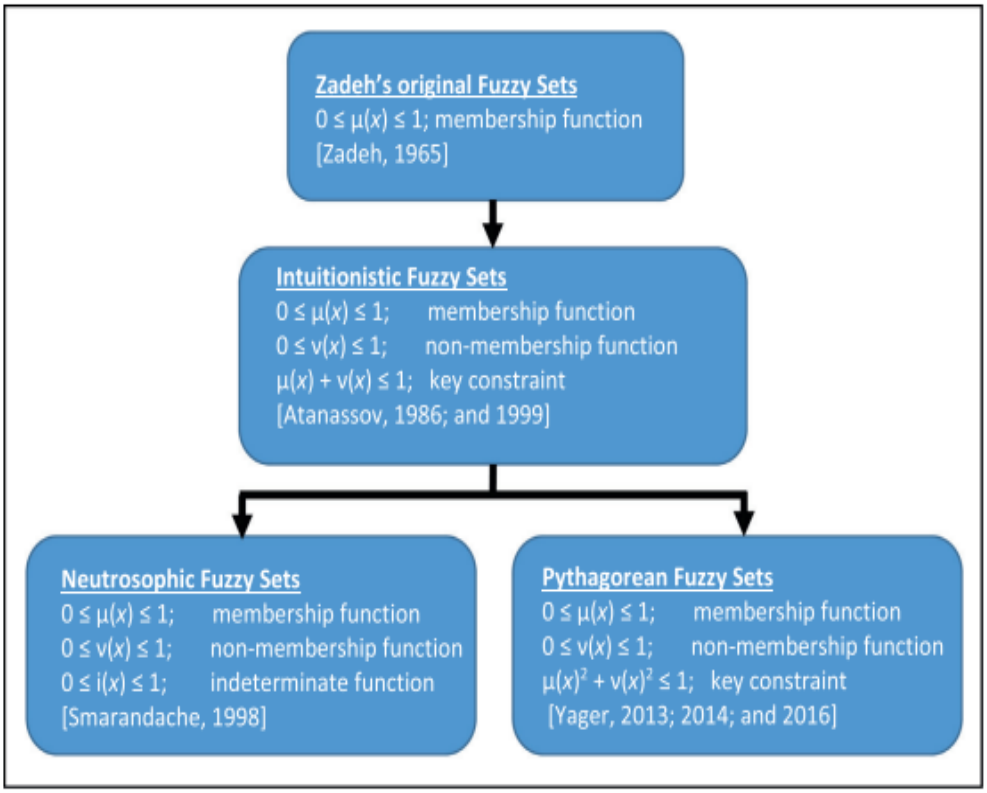

Figure 2.6: Some key developments in the introduction of fuzzy set concepts [26]

2.3.4.3 Linguistic Variables

Zadeh states "In retreating from precision in the face of overpowering complexity, it is natural to explore the use of what might be called linguistic variables, that is, variables whose values are not numbers but words or sentences in a natural or artificial language.... The motivation of the use of words or sentences rather than numbers is that linguistic characterizations are, in general, less specific than numerical ones." [27].

Let U denote the name of a linguistic variable (e.g., temperature). Numerical values of a linguistic variable U are denoted x,where $x \in U$. Sometimes x and IL are used interchangeably, especially when a linguistic variable is a letter, as is sometimes the case in engineering applications. A linguistic variable is usually decomposed into a set of terms,T(U),which cover its universe of discourse.

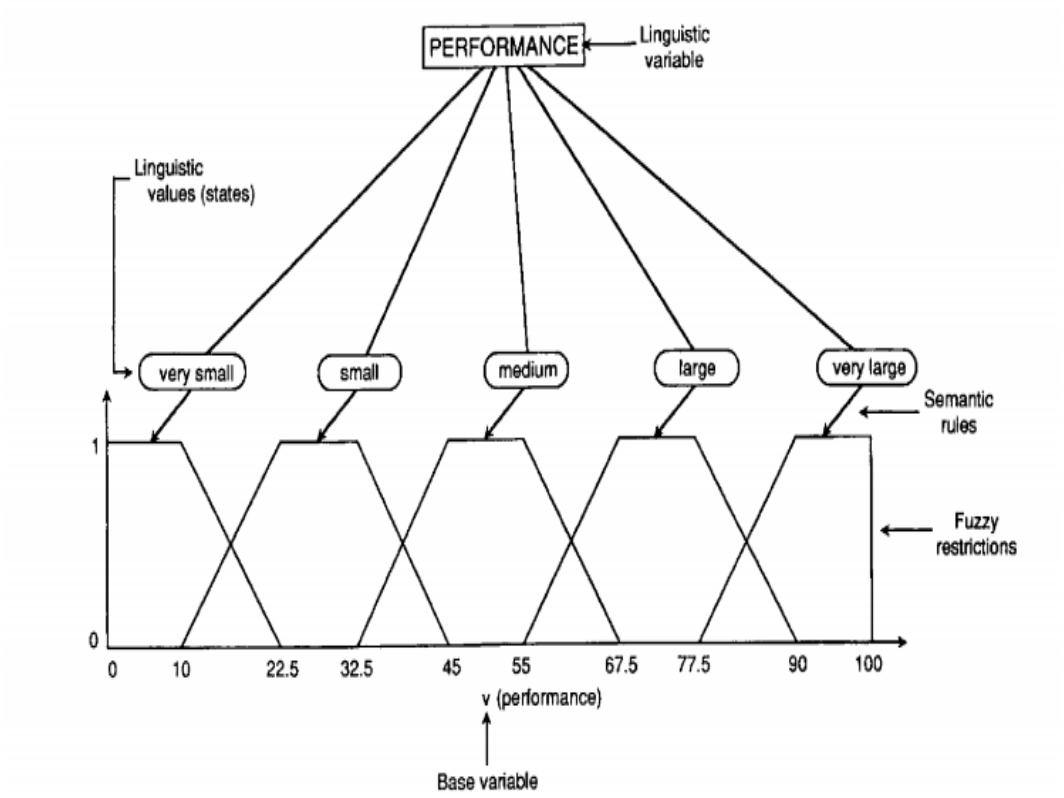

Figure 2.7: Example of linguistic variable [28]

2.3.4.4 Membership Functions

In engineering applications of fuzzy logic, membership functions,$\mu F\ (x)$, are, for the most part, associated with terms that appear in the antecedents or consequent of rules, or in phrases (e.g., foreign cars).

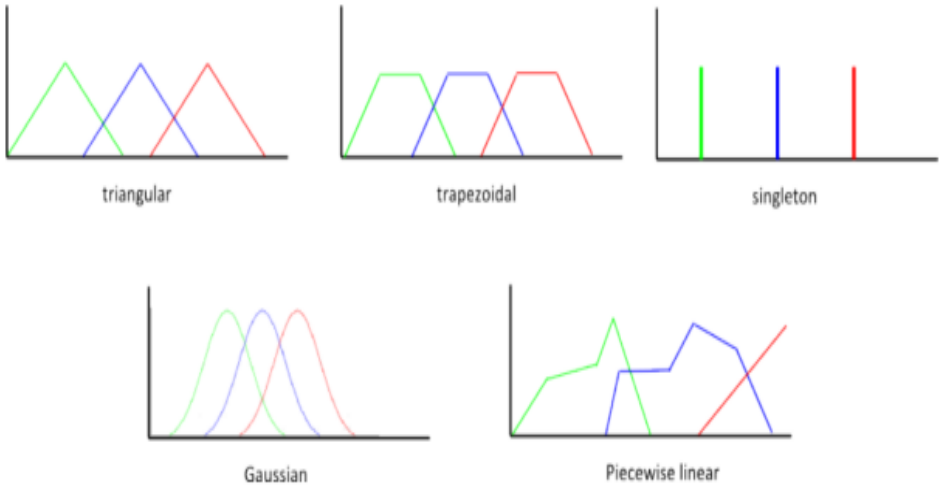

Figure 2.8: Different Types of Membership Functions [29]

2.4 Related work

Artificial intelligence is widely used in medical field. In the literature, several medical predictive systems are proposed. Most of them applied Deep learning techniques for diagnosis and prediction to reduce costs and to improve accuracy and time response. In this section, we discuss some AI based works for medical prediction.

Na L. et al.[30], proposed an intelligent classification model for breast cancer diagnosis based on a hybrid feature selection approach: gain directed simulated annealing genetic algorithm wrapper (IGSAGAW), to remove redundant and irrelevant feature from the feature space and cost sensitive support vector ma- chine (CSSVM) learning algorithm. The proposed method is applied on Wisconsin Original Breast Cancer (WBC) and WBCD. The proposed work shows a good performance and decreases the calculation complexity.

Abdel-Zaher AM. et al.[31], developed a Computer- Aided Diagnosis (CAD) scheme for detection of breast cancer using deep belief network (DBN) unsupervised path followed by back propagation supervised path. The proposed system was tested on the Wisconsin Breast Cancer Dataset (WBCD) and gave an accuracy of 99.68%.

Mehrbakhsh Nilashi et al.[32], developed a knowledge-based system for classification

of breast cancer disease using Expectation Maximization (EM), Classification and Regression Trees (CART) and Principal Component Analysis (PCA). The proposed system can be used as a clinical decision support system to assist medical practitioners in the healthcare practice.

In this paper, Ankur M. and Jaymin P. [33], proposed a predictive model for heart disease detection using Machine Learning and Data Mining techniques. The proposed approach combined between Naive Bayes (NB) and Genetic Algorithm (GA) to classify heart diseases. Data were collected from Cleveland Heart Disease Data set (CHDD) available on the UCI Repository. Nonetheless, this model could not predict specific heart disease.

Tanmay K. et al. [34], proposed a fuzzy expert system for heart disease diagnosis. They used MATLAB software is as a development tool and a database from V.A. Medical Center, Long Beach and Cleveland Clinic Foundation.

Ahmed H. et al. [35], proposed a KNN variant (KNNV) algorithm for COVID-19 classification. was implemented and tested on a COVID-19 dataset from the Italian society of medical and intervention radiology society and it was also compared to other algorithms of its category. the proposed algorithm shows good results terms of four metrics: precision, recall, accuracy, and F-Score.

Tulin O. et al. [36], designed a deep learning based model for automatic COVID-19 detection using raw chest X-ray images and The DarkNet model.

In [37], a deep Learning based network to predict the real-time COVID-19 transmission is developed. The model is based on LSTM network to overcome the limits of traditional methods (could not able to represent the spatio-temporal components simultaneously) . the used data is time serie dataset collected from Hopkins university and Canadian Health Authority. The proposed model achieves good performance with an accuracy of 92.67% for long term predictions.

Jordi L. et al.[34], developed an AI speech processing framework for COVID-19 real-time and non-invasive diagnosis from cough recordings.

Gonçalo M. et al. [38], proposed a medical decision support system based on convolutional neural network (CNN) using EfficientNet architecture, and presented two experiments: multi-class and binary classification. The system was developed using X-ray Image DataSet .

Reference	Objective	Used technique	Data	Accuracy	Limitations
Na L. et al. [30], 2019	Breast cancer diagnosis	IGSAGAW + CSSVM	WBC	95.8%	Computationaly expensive
			WBCD	95.7%	
Abdel-Zaher AM. et al. [31], 2015	Breast cancer diagnosis	DBN	WBCD	99.68%	Computationaly expensive
Nilashi M. et al.[32], 2017	Breast cancer diagnosis	EM clustering + PCA + Fuzzy logic + CART	WDBC	93.2%	SmallIEM fails on high-dimensional data sets due to numerical precision problems
			Mammographic mass dataset	94.1%	
Ankur M. et al. [33], 2015	Heart disease detection	Naive Baye Genetic algorithm	Cleveland Heart Disease Data set	-	Computationaly expensive
Tanmay K. et al. [34], 2017	heart disease diagnosis	Fuzzy logic	Numerical data	93.33%	Completely dependent to human expertise.
Ahmed H. et al. [35], 2021	COVID-19 classification	KNNV	IHC dataset	85%	heterogeneous and incomplete data
Tulin O. et al.[36], 2020	automatic COVID-19 detection	Deep learning (DarkNet model)	X-ray images	97.11%	use of a limited number of X-ray images.
Chimmula V. et al.[37], 2020	COVID-19 transmission forecasting	LSTM networks	Time serie dataset	92.67%	Small data set
Jordi L. et al. [34], 2020	COVID-19 diagnosis	Biomarkers CNN model	voice recordings	98.5%	Quality of input data issue
Gonçalo M. et al. [38], 2020	COVID-19 diagnosis	CNN model	X-ray images	binary classification:99% Multi-class : 96.6%	The system performance can be lower in early stage detection

Table 2.1: Comparison of earlier systems

The use of AI techniques in medical field to manage different diseases shows a performance improvement in terms of accuracy, speed and cost. It makes disease management more reliable by reducing diagnosis errors, and extracting useful information from large amount of data to predict health outcomes.

2.5 Conclusion

This chapter covered definitions of deep learning and fuzzy logic, and a comparative study of deep learning and fuzzy logic was the main focus of the class due to their importance and also because they are the main topic of this research and some related work. The system design and data set will be introduced in the next chapter, as well as the proposed architectures.

Chapter 3

System design

3.1 Introduction

The COVID-19 pandemic has posed unprecedented challenges to our society made us propose solutions, under the shadow of telehealth and artificial intelligence, we proposed our system of COVID-19 prediction.

In this chapter, we will study and analyze the real needs of users by highlighting the conceptual side of our application, which allows us to detail the different general architecture and scenarios as well as the detailed design. to be implemented in the phase next.

3.2 General System Architecture

In general, our coronavirus prediction system will follow a set of phases, as shown in Figure (3.1).

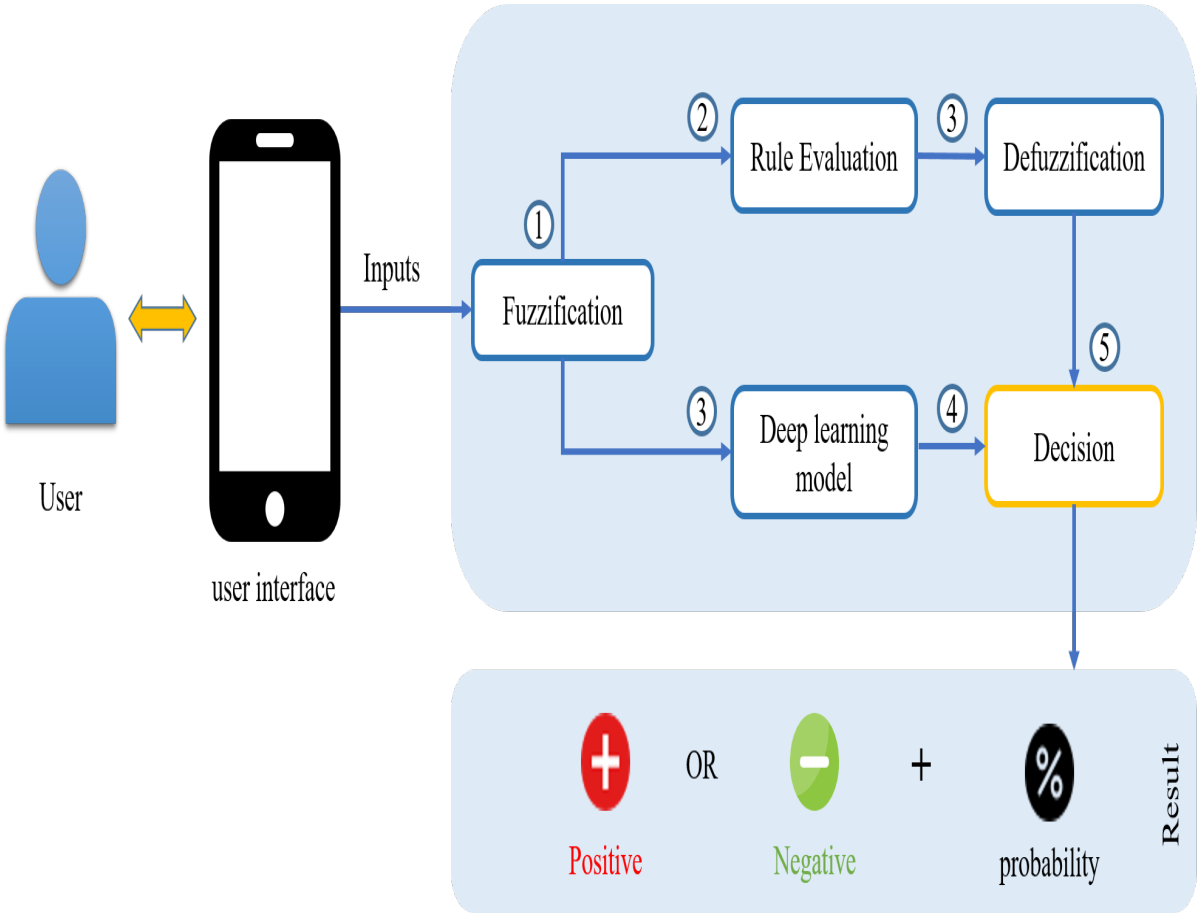

Figure 3.1: The General system architecture

The proposed approach is made up of three levels, respectively the user level, the application interface level, and the resulting level.

3.2.1 Fuzzification

In this step we define linguistic variables and terms.

Linguistic variables are basic words or sentences that serve as input and output variables.To do this, we need to create membership functions that define the degree to which a numeric data belongs to a linguistic variable. The inputs of fuzzification are the important symptoms for the detection of COVID-19.

The tables below offer an example of linguistic values utilized in our system:

Fever	
Linguistic values	Range of Linguistic values
Low	[34 , 36]
Normal	[35.5 , 37.5]
High	[37 , 40]

Table 3.1: Fever parameter in fuzzy representation

Heart Rate	
Linguistic values	Range of Linguistic values
Low	[40 , 60]
Normal	[55 , 80]
High	[75 , 95]
Very High	[90 , 100]

Table 3.2: Heart Rate parameter in fuzzy representation

Blood Oxygen	
Linguistic values	Range of Linguistic values
Risk	[10 , 90]
Normal	[80 , 100]

Table 3.3: Blood Oxygen parameter in fuzzy representation

Cough	
Linguistic values	Range of Linguistic values
Low	[1 , 4]
High	[3 , 10]

Table 3.4: Cough parameter in fuzzy representation

COVID-19	
Linguistic values	Range of Linguistic values
Low	[0 , 40]
Medium	[30 , 70]
High	[60 , 100]

Table 3.5: COVID-19 parameter in fuzzy representation

Symptoms of COVID-19 can be graphically represented by trapezoidal membership function by the following figures:

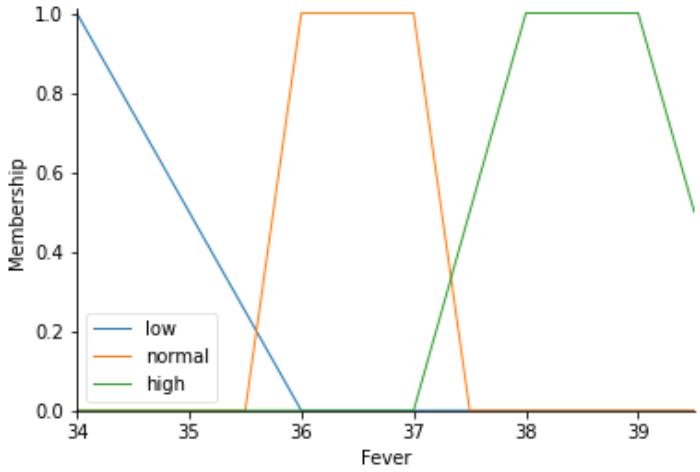

Figure 3.2: Linguistic variable Fever

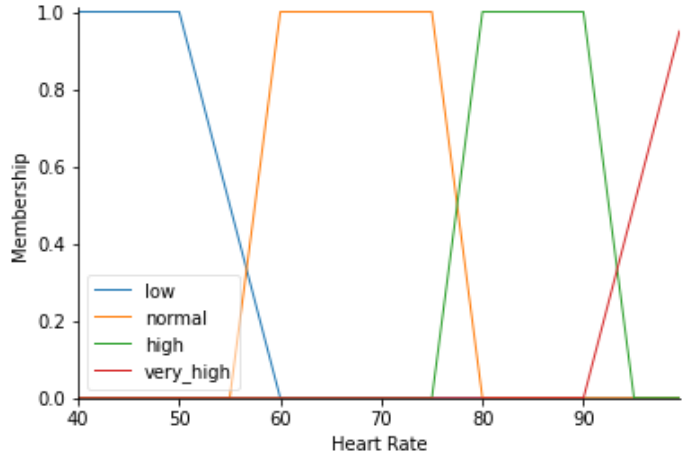

Figure 3.3: Linguistic variable Heart rate

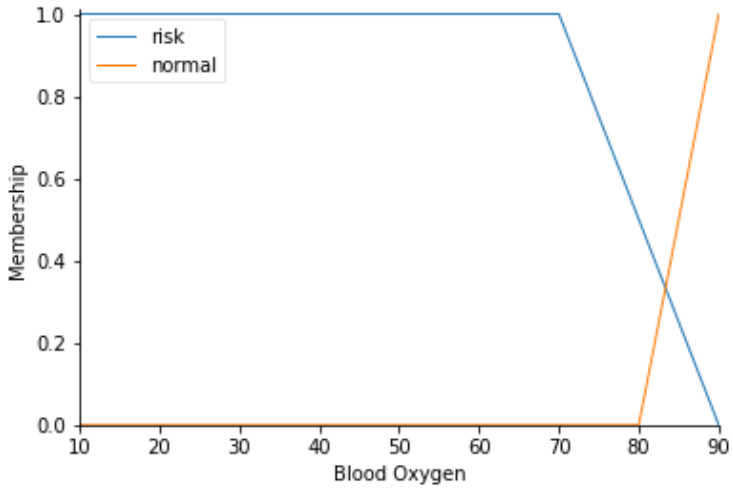

Figure 3.4: Linguistic variable Blood oxygen

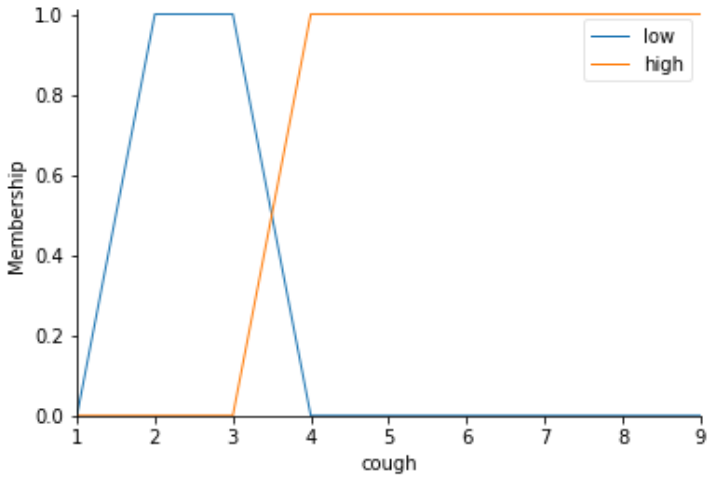

Figure 3.5: Linguistic variable Cough

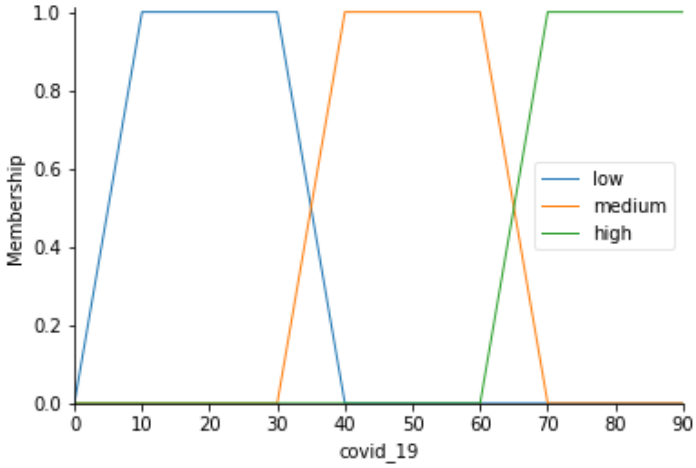

Figure 3.6: Linguistic variable COVID-19(output)

3.2.2 Rule Evaluation

It is made up of all the information we have about the process. It allows you to define membership functions and fuzzy rules that describe the behavior of the system. It is the heart of the entire system in the sense that all other components are used to interpret and combine these rules to form the final system.

The collection of fuzzy rules generated in our system is summarized in the following sentences:

1. IF fever is low AND heart rate is normal AND blood oxygen is normal AND cough is low THAN COVID-19 is lowAND.

2. IF fever is normal OR heart rate is low OR blood oxygen is normal OR cough is low THAN COVID-19 is low.
3. IF fever is normal AND heart rate is low AND blood oxygen is normal AND cough low , COVID-19 is medium.
4. IF fever is normal AND heart rate is normal AND blood oxygen is normal AND cough is low THAN COVID-19 is high.
5. IF fever is normal AND heart rate is normal AND blood oxygen is normal AND cough is low THAN COVID-19 is low.
6. IF fever is low AND heart rate is high AND blood oxygen is normal AND cough is low THAN COVID-19 is medium.
7. IF fever is low AND heart rate is high AND blood oxygen is normal AND cough is low THAN COVID-19 is high.
8. IF fever is high OR heart rate is high THAN COVID-19 is medium.
9. IF fever is normal AND blood oxygen is risk THAN COVID-19 is medium.
10. IF cough is high THAN COVID-19 is high.
11. IF fever is normal AND heart rate is very high AND blood oxygen is risk AND cough is high THAN COVID-19 is high.
12. IF fever is high AND heart rate is normal AND blood oxygen is normal AND cough is low THAN COVID-19 is high.
13. IF heart rate is normal OR blood oxygen is normal THAN COVID-19 is high .
14. IF fever is normal AND heart rate is low AND blood oxygen is risk AND cough is low THAN COVID-19 is high.
15. IF fever is normal AND heart rate is high AND blood oxygen is normal AND cough is high THAN COVID-19 is medium.
16. IF fever is normal AND heart is rate is very high AND blood oxygen is normal AND cough is low THAN COVID-19 is medium.

The next step is rules evaluation via fuzzy set operations. Max and Min are the operations used for OR and AND respectively. Combine all of the evaluation findings to create a final result. This is a fuzzy value.

3.2.3 Defuzzification

This step consists of carrying out the reverse operation of the fuzzification, that is to say, obtained physical value of the output from the surface obtained.
We used the center of area method (COA) in this step.

$$C_x = \frac{\sum_{i=1}^{n} A_i C_i}{\sum_{i=1}^{n} A_i} \tag{3.1}$$

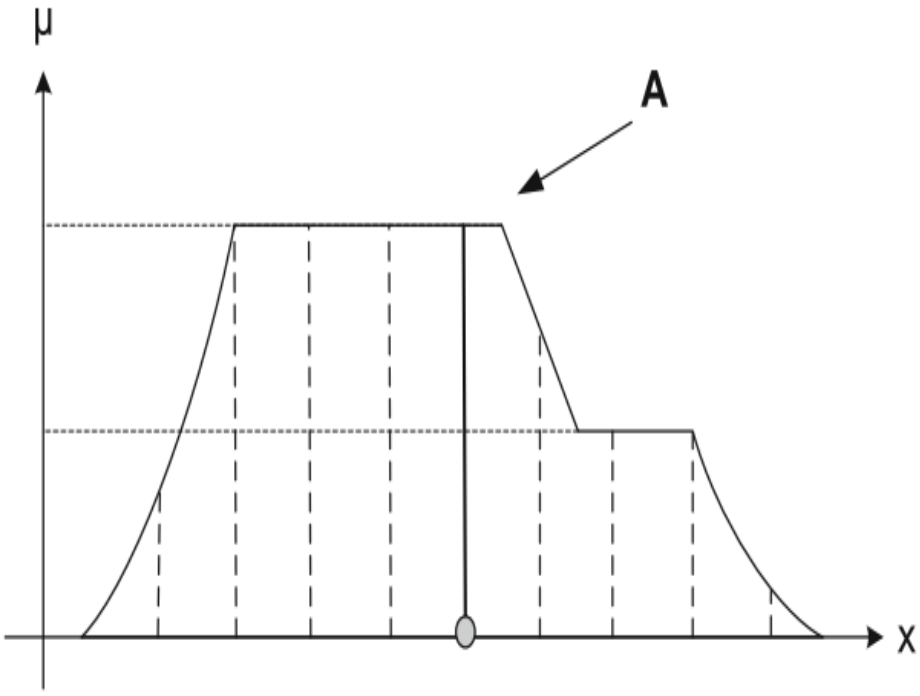

Figure 3.7: Defuzzification by the center of area [39]

The mathematical expression represented in (3.1) is how to calculate the output of defuzzification with the center of area method (COA), And figure (3.7) is an example of this method.

3.2.4 Deep learning model architecture

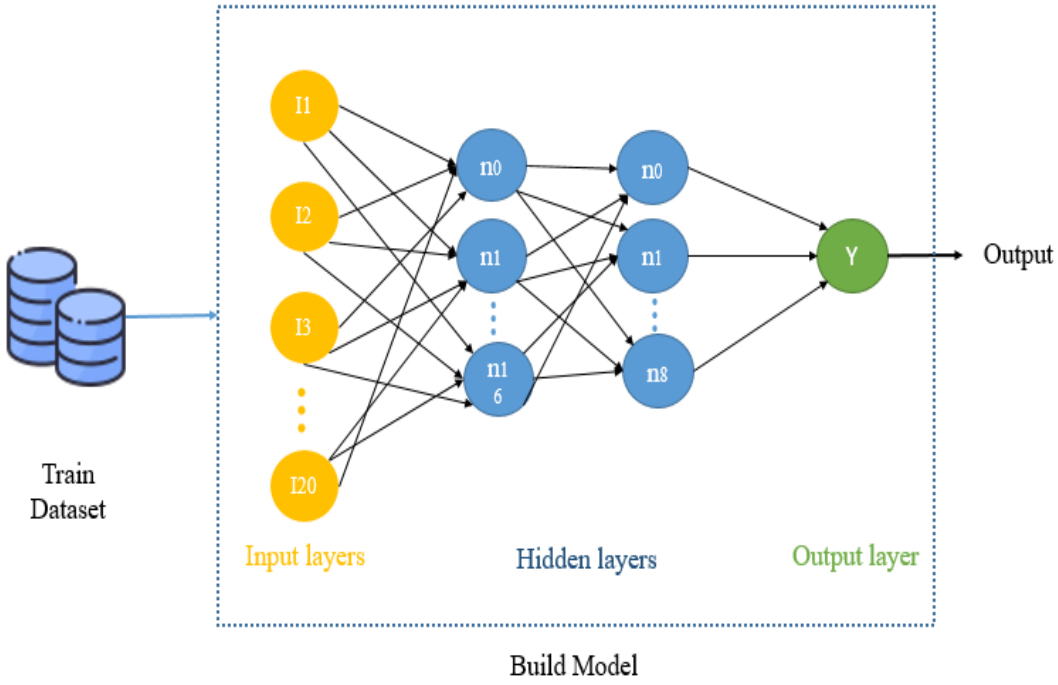

Figure 3.8: Deep learning model architecture

3.2.5 Dataset Description

These data will aid in determining whether or not a person has coronavirus illness based on a set of established symptoms.

The dataset contains 20 variables that will be having an impact on whether someone has coronavirus disease or not, Are as follows :

1. Symptoms:
   (a) Breathing Problem (true/false).
   (b) Fever (true/false).
   (c) Sore throat (true/false).
   (d) Running Nose (true/false).
   (e) Headache (true/false).

(f) Hypertension (true/false).

(g) Fatigue (true/false).

(h) Gastrointestinal (true/false).

2. Chronic diseases:

(a) Asthma (true/false).

(b) Chronic Lung Disease (true/false).

(c) Heart Disease (true/false).

(d) Diabetes (true/false).

3. Other information:

(a) Abroad travel (true/false).

(b) Contact with COVID-19 Patient (true/false).

(c) Attended Large Gathering (true/false).

(d) Family working in Public Exposed Places (true/false).

(e) Wearing Masks (true/false).

(f) Sanitization from Market (true/false).

The following (Figure 3.8) represents COVID-19 disease presence distribution in dataset.

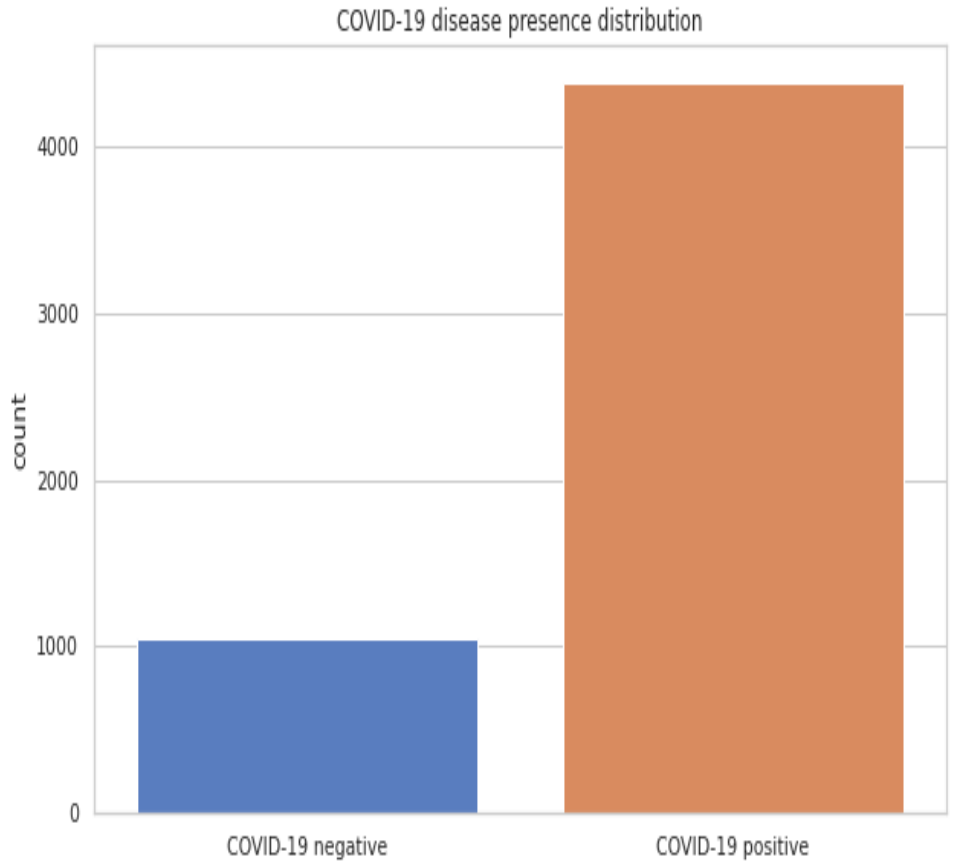

Figure 3.9: COVID-19 disease presence distribution

3.2.6 DL Model Description

The proposed model is made up of three levels, respectively the input layers, the hidden layers level and the output layer level.

3.2.6.1 Inputs

The first layer (Input layers) in a model needs to receive information about its input shape. So in our model we have 20 inputs.

3.2.6.2 Activation functions

Activation functions are a critical part of the design of a neural network, which defines the output of the node given an input or set of inputs.

The following table (Table 3.6) represents the activation functions used in our model.

Layers	Activation functions
Input Layers	Relu
Hidden Layers	Relu
Output Layer	Sigmoid

Table 3.6: The activation functions

Relu activation function: ReLU stands for rectified linear unit and is a non-linear activation function that is widely used in the neural network.It can be defined mathematically as (3.1) [40]:

$$f(x) = max(0,x) \tag{3.2}$$

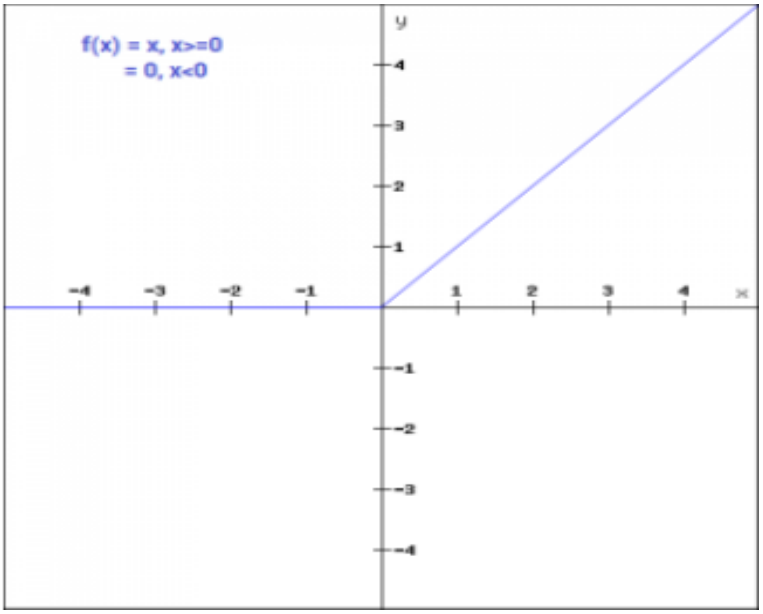

Figure 3.10: ReLU Activation Function plot [40]

Sigmoid activation function: It is the most widely used activation function as it is a non-linear function. Sigmoid function transforms the values in the range 0 to 1. It can be defined as (3.2) [40] :

$$f(x) = 1/e^{-x} \tag{3.3}$$

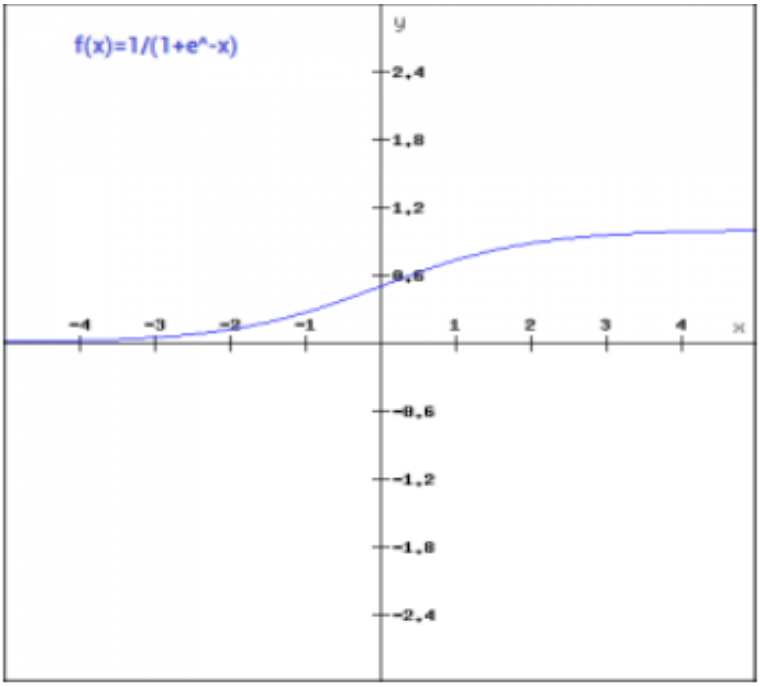

Figure 3.11: Sigmoid Activation Function plot [40]

3.3 Functional modeling

In this section, we'll simulate how our system works.

3.3.1 Sequence diagram

Is sequential representation of the course of processing and interactions between the elements of the system.

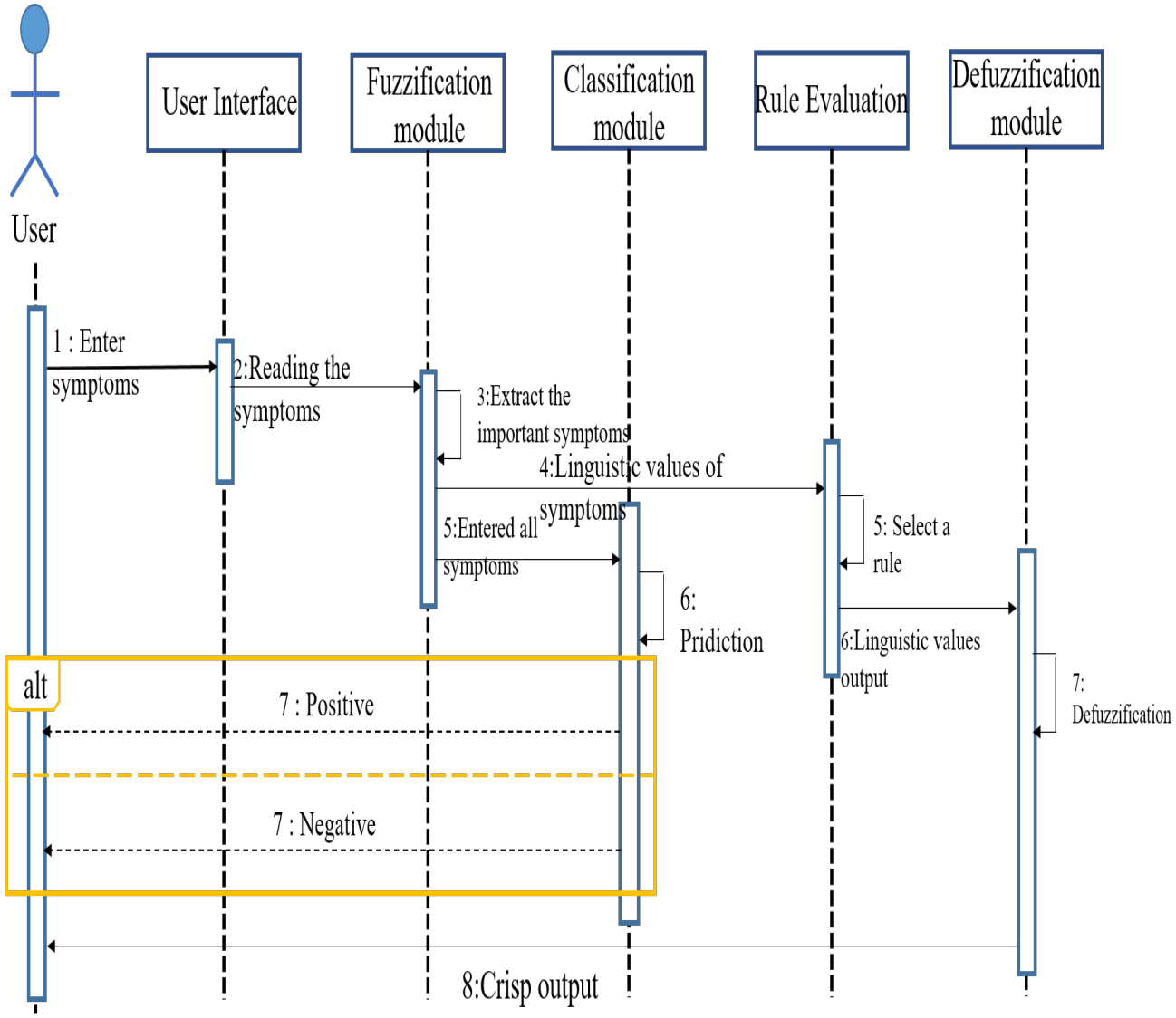

Figure 3.12: Sequence diagram of System

3.4 Algorithm of COVID-19 prediction system

The following (figure 3.13) is represented algorithm of our system.

Algorithm of COVID 19 prediction

Input :

•TR : Training dataset;

•A : Set of selected features {f_1,f_2,f_3,...,f_n};

Output :

•Positive or Negative ;

•Value of diagnosing ;

Steps :

•//Loading the Dataset ;

Data = read (TR);

//Create X and Y for traing ;

X_tring, Y_tring, Y_tring, Y_test = model_selction (X,Y);

//Create the DL model ;

Model = Create_model(X_tring, Y_tring, Y_tring, Y_test,Data);

// Serialize model to Json;

Model_josen = model.to_json();

•// Create the range of features;

// f, hr, bo, c ∈ A

ceate_range(f);

ceate_range(hr);

ceate_range(bo);

ceate_range(c);

•// Calculate membership values;

memValue(f);

memValue(hr);

memValue(bo);

memValue(c);

•//Create the set of Rules;

RS = {r_1,r_2,r_3,...,r_m};

•Read (A) ;

ld = loaded_model(Model_josen) ;

Prediction = ld.predict_classes(A) ;

•// f, hr, bo, c ∈ A;

read(f, hr, bo, c) ;

•// Calculate the fuzzy output of each rule ;

RF = Max_Min(RS) ;

•// Calculate the diagnosis value

R =diango (prediction , RF) ;

Figure 3.13: Algorithm of COVID-19 prediction system

the following (table 3.7) represents the parameters of the algorithm.

Parameter	Description
TR	Dataset
A	set of selected features f1,f2,f3,. . . ,fn
n	Number of features
RS	Rules set r1,r2,r3,. . . ,rm
m	Number of rules
f	Fever
hr	Heart rate
bo	Blood Oxygen
c	Cough
R	Result
LD	loaded model variable

Table 3.7: The algorithm parameters

3.5 Conclusion

This chapter detailed the architecture of our system and its stages, as well as demonstrating its many characteristics step by step. In the next chapter, We'll talk about the technology we used to make it happen.

Chapter 4

Implementation & Results

4.1 Introduction

After going over the architecture of our system in depth in the previous chapter. The goal of this chapter is to put the steps in place to achieve the system that has been created. We begin in the first section by describing the development frameworks and tools. Following that, we'll go over all of the proposed strategies as well as the outcomes. Finally, we'd want to show off some of the code that has been developed.

4.2 Work Environment and Development Tools

To implement our system we used the following tools and environment.

4.2.1 Programming languages

We use two programming languages (Python and Java) to code our system.

4.2.1.1 Python

Python is an easy to understand, versatile programming language. It has efficient high-level data structures and a clear but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, as well as its interpreted nature, make it an excellent language for scripting and rapid application creation across a wide range of platforms [41].

Figure 4.1: Python logo[42]

```
import sys
sys.version
```

```
'3.7.10 (default, May  3 2021, 02:48:31) \n[GCC 7.5.0]'
```

Figure 4.2: Python version used in google colab

4.2.1.2 Java

Java is a programming language and computing platform first released by Sun Microsystems in 1995. There are lots of applications and websites that will not work unless you have Java installed, and more are created every day. Java is fast, secure, and reliable. From laptops to datacenters, game consoles to scientific supercomputers, cell phones to the Internet [43].

Figure 4.3: Java logo[43]

4.2.2 Deep learning and Fuzzy logic kit

The following is the kit of deep learning and fuzzy logic used in our system.

4.2.2.1 NumPy

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more [44].

Figure 4.4: NumPy logo [44]

4.2.2.2 Pandas

Pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language [45].

Figure 4.5: Pandas logo [45]

4.2.2.3 Matplotlib

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.

Matplotlib produces publication-quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shell, web application servers, and various graphical user interface toolkits [46].

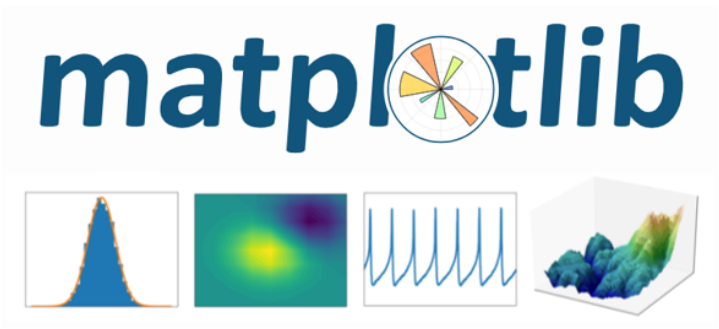

Figure 4.6: Matplotlib logo & Exemples of matplotlib works[46]

4.2.2.4 keras

Keras is a high-level neural networks library, written in Python and capable of running on top of either TensorFlow or Theano. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay is key to doing good research [47].

Figure 4.7: Keras logo [48]

4.2.2.5 Scikit-learn

Scikit-learn is a Python module integrating a wide range of state-of-the-art machine learning algorithms for medium-scale supervised and unsupervised problems. This package focuses on bringing machine learning to non-specialists using a general-purpose high-level language. Emphasis is put on ease of use, performance, documentation, and API consistency. It has minimal dependencies and is distributed under the simplified BSD license, encouraging its use in both academic and commercial settings [49].

Figure 4.8: Scikit-learn logo [50]

4.2.2.6 Scikit-fuzzy

Scikit-fuzzy (a.k.a. skfuzzy): Fuzzy Logic Toolbox for Python.
This package implements many useful tools and functions for computation and projects involving fuzzy logic, also known as grey logic [51].

Figure 4.9: Scikit-fuzzy logo [52]

4.2.3 Frameworks and tools

4.2.3.1 Google Colaboratory

Google Colaboratory more commonly referred to as "Google Colab" or just simply "Colab" is a research project for prototyping machine learning models on powerful hardware options such as GPUs and TPUs. It provides a serverless Jupyter notebook environment for interactive development. Google Colab is free to use [53].

Figure 4.10: Google Colaboratory logo [54]

4.2.3.2 Kaggle

Kaggle is an online group of data scientists and computer learners, Google LLC owns it. Kaggle began by hosting machine learning competitions and has since expanded to include a shared data platform, a cloud-based data science workbench, and short-form AI training [55].

Figure 4.11: Kaggle logo [55]

4.2.3.3 Android Studio

Android Studio is the official Integrated Development Environment (IDE) for Android app development, based on IntelliJ IDEA . On top of IntelliJ's powerful code editor and developer tools, Android Studio offers even more features that enhance your productivity when building Android apps, such as: A flexible Gradle-based build system , C++ and NDK support and extensive testing tools and frameworks, ect [56].

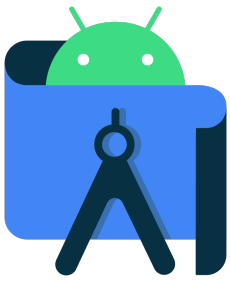

Figure 4.12: Android Studio logo [56]

4.2.3.4 Chaquopy

Chaquopy provides everything you need to include Python components in an Android app, including:

- Full integration with Android Studio's standard Gradle build system.
- Simple APIs for calling Python code from Java/Kotlin, and vice versa.
- A wide range of third-party Python packages, including SciPy, OpenCV, TensorFlow and many more.[57]

Figure 4.13: Chaquopy [57]

4.3 Implementation phases

The implementation has multiple steps, as shown in the global system design.

4.3.1 Creating the DL model

In this part, we will show how we used deep learning to classify and detect COVID-19.

4.3.1.1 Loading and split the dataset

To load the data we use pd.read_csv from panda library (Figure 4.13).

```
import pandas as pd
data=pd.read_csv("/content/drive/MyDrive/Colab Notebooks/Dataset/covid19.csv")
```

Figure 4.14: Importing Panda and loaling dataset code

Then, we split the original dataset into train and test sets (Figure 4.14).

```
# create X and Y datasets for training
X=np.array(data.drop(['COVID-19'],1))
Y=np.array(data['COVID-19'])
X_train,X_test,Y_train,Y_test =model_selection.train_test_split(X,Y,test_size=0.4)
```

Figure 4.15: Split the dataset code

4.3.1.2 The DL model

The following figure (4.15) represents the function of creating binary model code .

```
def create_binary_model():
    # create model
    model = Sequential()
    model.add(Dense(20, input_dim=20, kernel_initializer='normal',activation='relu'))
    model.add(Dense(17, kernel_initializer='normal',  kernel_regularizer=regularizers.l2(0.001),activation='relu'))
    model.add(Dropout(0.3))
    model.add(Dense(9, kernel_initializer='normal',  kernel_regularizer=regularizers.l2(0.001),activation='relu'))
    model.add(Dropout(0.2))
    model.add(Dense(1, activation='sigmoid'))

    # Compile model
    adam = Adam(lr=0.001)
    model.compile(loss='binary_crossentropy', optimizer=adam, metrics=['accuracy'])
    return model
binary_model = create_binary_model()
binary_model.summary()
```

Figure 4.16: Binary model implementation code

Binary model arguments:

- **input_dim:** is the number of the features.
- **kernel_initializer:** is a term which refers to the statistical distribution or function that should be used to initialize the weights.
- **kernel_regularizer:** the regularization penalty to the loss in relation to that layer's weights.
- **Dropout Layer:** We utilized a dropout layer, which eliminates or ignores neurons during the training phase of a randomly selected set of neurons. This is done to stop overfitting.
- The adam optimizer is used to compile the model with a binary cross entropy entropy loss function.

```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_20 (Dense)             (None, 20)                420
_________________________________________________________________
dense_21 (Dense)             (None, 17)                357
_________________________________________________________________
dropout_10 (Dropout)         (None, 17)                0
_________________________________________________________________
dense_22 (Dense)             (None, 9)                 162
_________________________________________________________________
dropout_11 (Dropout)         (None, 9)                 0
_________________________________________________________________
dense_23 (Dense)             (None, 1)                 10
=================================================================
Total params: 949
Trainable params: 949
Non-trainable params: 0
```

Figure 4.17: The summary of model

The textual summary (Figure 4.16) offers information on the model's layers and their order.

- Each layer's output shape.
- Each layer's number of parameters (weights).
- The model's total number of parameters (weights).

4.3.1.3 Training the model

The training process is sped up by using a Google colab's GPU, and we set the number of epochs to 80. To train the models, we use the fit function (Figure 4.17).

```
# fit the binary model on the training data
history=binary_model.fit(X_train, Y_train, validation_data=(X_test, Y_test), epochs=80, batch_size=10)
```

Figure 4.18: The fit function code

To store training epochs and work information, we utilize the variable 'history'.
In the results section, the figures and values from the model and dataset training will be discussed.

4.3.1.4 Model evaluation

Finally, we use certain statistics to assess our model's performance, as seen in the code below (Figure 4.18), which represents the binary model's classification report. The value of the F1-score matrix for negative COVID-19 is 0.93 and 0.98 for the positive COVID-19, where the macro average is 0.96 and the weighted average is 0.97. The value of the recall matrix for negative COVID-19 is 0.89 and 0.99 for the positive COVID-19, where the macro average is 0.94 and the weighted average is 0.97. The value of the precision matrix for negative COVID-19 is 0.96 and 0.98 for the positive COVID-19, where the macro average is 0.97 and the weighted average is 0.97. Finally, the value of accuracy is 0.97 (97%).

```
#__________test_____________________________
binary_pred_ts = np.round(binary_model.predict(X_test)).astype(int)
print('Results for Binary Model (test)')
print(accuracy_score(Y_test, binary_pred_ts)*100)
print(classification_report(Y_test, binary_pred_ts))
```

```
Results for Binary Model (test)
97.37810487580496
              precision    recall  f1-score   support

           0       0.96      0.89      0.93       406
           1       0.98      0.99      0.98      1768

    accuracy                           0.97      2174
   macro avg       0.97      0.94      0.96      2174
weighted avg       0.97      0.97      0.97      2174
```

Figure 4.19: Testing binary model accuracy

We also evaluate the binary model's performance using the AUC–ROC curve, which is a model selection metric for bi–class classification problems. On the X-axis is the False Positive Rate (FPR), and on the Y-axis is the True Positive Rate (TPR).

The larger the area between the orange line (ROC) and the axis, the more accurate the deep learning models are at differentiating between the various classes. In our model, we observe the area is large, and the percentage to differentiate between the two classes is 98%.(Figure 4.19)

```
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
fpr,tpr,thresholds=roc_curve(Y_test,binary_pred_ts)
print('ensemble test roc-auc:{}'.format(roc_auc_score(Y_test,binary_pred_ts)))
fpr,tpr,thresholds=roc_curve(Y_test,binary_pred_ts)
def plot_roc_curve(fpr,tpr):
  plt.plot(fpr,tpr,color='orange',label='ROC')
  plt.plot([0,1],[0,1],color='darkblue',linestyle='--')
  plt.xlabel('false positive Rate')
  plt.ylabel('true positive Rate')
  plt.title('Receiver Operating Characteristic (ROC) curve')
  plt.legend()
  plt.show()
plot_roc_curve(fpr,tpr)
```

```
ensemble test roc-auc:0.9873808188446438
```

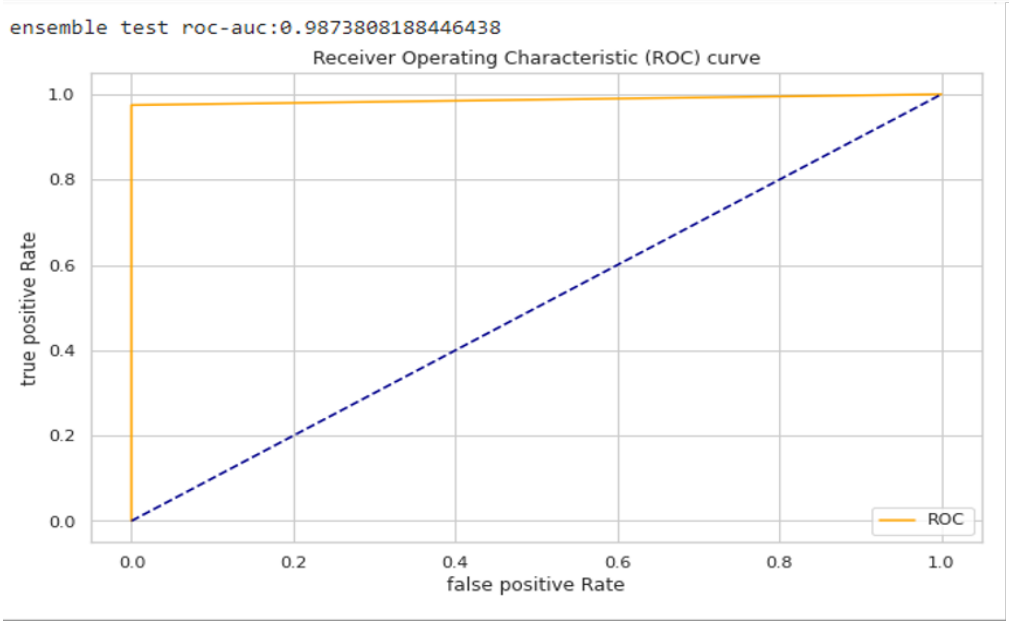

Figure 4.20: The AUC-ROC curve for our the model

4.3.2 Implementation of a fuzzy system part

To simulate this, we use the skfuzzy control system API. Let's start by defining fuzzy variables.

- **Step 1: (Fuzzification)** Create universe variables for inputs(fever, heart_rate, blood_oxygen and cough) and output (covid_19) variables (Figure 4.20).

```
import skfuzzy as fuzz
from skfuzzy import control as ctrl
import numpy as np

fever = ctrl.Antecedent(np.arange(34, 40, 0.5), 'Fever')
heart_rate = ctrl.Antecedent(np.arange(40, 100, 0.5), 'Heart Rate')
blood_oxygen= ctrl.Antecedent(np.arange(10, 100, 10), 'Blood Oxygen')
cough= ctrl.Antecedent(np.arange(1, 10, 1), 'cough')
covid_19= ctrl.Consequent(np.arange(0, 100, 10), 'covid_19')
```

Figure 4.21: Create universe variables code

- **Step 2:** Custom the membership functions. (Figure 4.21)

```
#fever
fever['low'] = fuzz.trapmf(fever.universe, [34,34 ,34, 36])
fever['normal'] = fuzz.trapmf(fever.universe, [35.5, 36, 37,37.5])
fever['high'] = fuzz.trapmf(fever.universe, [37, 38,39 ,40])
#heart_rate
heart_rate['low'] = fuzz.trapmf(heart_rate.universe, [40, 40,50, 60])
heart_rate['normal'] = fuzz.trapmf(heart_rate.universe, [55,60, 75,80])
heart_rate['high'] = fuzz.trapmf(heart_rate.universe, [75, 80,90, 95])
heart_rate['very_high'] = fuzz.trapmf(heart_rate.universe, [90, 100,100,100])
#blood_oxygen
blood_oxygen['risk'] = fuzz.trapmf(blood_oxygen.universe, [10, 10,70, 90])
blood_oxygen['normal'] = fuzz.trapmf(blood_oxygen.universe, [80, 90,100, 100])
#cough
cough['low'] = fuzz.trapmf(cough.universe, [1, 2,3, 4])
cough['high'] = fuzz.trapmf(cough.universe, [3, 4,10, 10])

# covid_19
covid_19['low'] = fuzz.trapmf(covid_19.universe, [0,10,30,40])
covid_19['medium'] = fuzz.trapmf(covid_19.universe, [30,40,60,70])
covid_19['high'] = fuzz.trapmf(covid_19.universe, [60,70,90, 100])
```

Figure 4.22: Custom the membership functions code

- **Step 3: (Rule Evaluation)** Now, to make these membership functions useful, we define the fuzzy relationship between input and output variables (rules), that has been identified in the system design (Figure 4.22).

```
#rules
r1 = ctrl.Rule(fever['low'] & heart_rate['normal'] & blood_oxygen['normal']& cough['low'], covid_19['low'])
r2 = ctrl.Rule(fever['normal'] | heart_rate['low'] | blood_oxygen['normal']|cough['low'], covid_19['low'])
r3 = ctrl.Rule(fever['normal'] & heart_rate['low'] & blood_oxygen['normal']&cough['low'], covid_19['medium'])
r4 = ctrl.Rule(fever['normal'] & heart_rate['normal'] & blood_oxygen['normal']&cough['low'], covid_19['high'])
r5 = ctrl.Rule(fever['normal'] & heart_rate['normal'] & blood_oxygen['normal']&cough['low'], covid_19['low'])
r6 = ctrl.Rule(fever['low'] & heart_rate['high'] & blood_oxygen['normal']&cough['low'], covid_19['medium'])
r7 = ctrl.Rule(fever['low'] & heart_rate['high'] & blood_oxygen['normal']&cough['low'], covid_19['high'])
r8 = ctrl.Rule(fever['high'] , covid_19['medium'])
r9 = ctrl.Rule( heart_rate['high'] , covid_19['medium'])
r10 = ctrl.Rule(fever['normal']  & blood_oxygen['risk'], covid_19['medium'])
r11 = ctrl.Rule(cough['high'], covid_19['high'])
r12 = ctrl.Rule(fever['normal'] & heart_rate['very_high'] & blood_oxygen['risk']&cough['high'], covid_19['high'])
r13 = ctrl.Rule(fever['high'] & heart_rate['normal'] & blood_oxygen['normal']&cough['low'], covid_19['high'])
r14 = ctrl.Rule( heart_rate['normal'] | blood_oxygen['normal'], covid_19['high'])
r15 = ctrl.Rule(fever['normal'] & heart_rate['low'] & blood_oxygen['risk']&cough['low'], covid_19['high'])
r16 = ctrl.Rule(fever['normal'] & heart_rate['high'] & blood_oxygen['normal']&cough['high'], covid_19['medium'])
r17 = ctrl.Rule(fever['normal'] & heart_rate['very_high'] & blood_oxygen['normal']&cough['low'], covid_19['medium'])
```

Figure 4.23: The ruleset code

- **Step 3: (Defuzzification)** Now that we have our rules defined, we can simply create a control system covid19_ctrl. To simulate this control system, we will create a ControlSystemSimulation. For defuzzification, We would create another ControlSystemSimulation when we're trying to apply our covid19_ctrl for COVID-19 because the inputs would be different (Figure 4.23).

```
#create a control system
covid19_ctrl = ctrl.ControlSystem([r1,r2,r3,r4,r5,r6,r7,r8,r9,r10,r11,r12,r13,r14,r15,r16,r17])
covid19 = ctrl.ControlSystemSimulation(covid19_ctrl)
```

Figure 4.24: The Control System code

4.3.3 Preliminary results of COVID-19 prediction system

The following figure (4.24) represents an example preliminary result of our system hybrid.

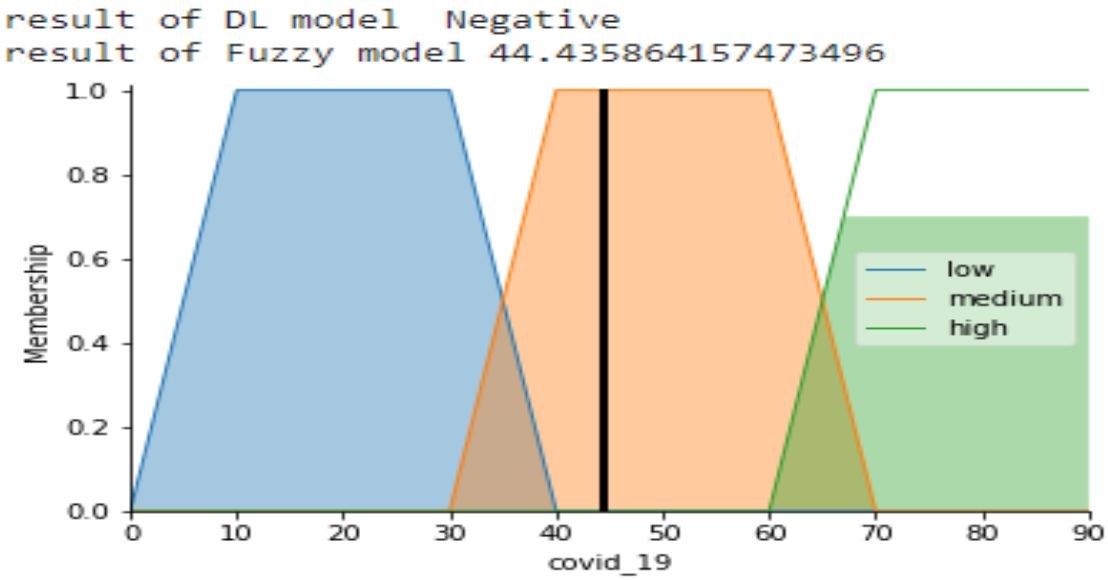

Figure 4.25: Example result of COVID-19 prediction system

4.4 COVID-19 prediction mobile app interface

4.4.1 User interface preview

The following (figure 4.26)is the user interface our mobile application built for providing a simple implementation of our hybrid pretrained model.

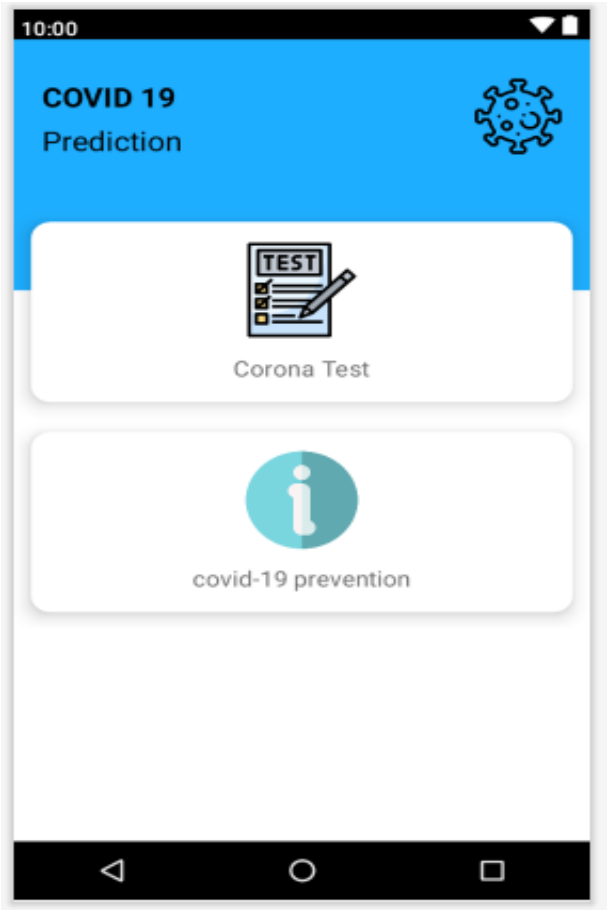

Figure 4.26: COVID-19 prediction app interface

4.4.2 User interface Detailed Explanation

In this section, we will explain the components of our previous interface, as we have it displayed and dissected in the below figure (4.27).

Figure 4.27: COVID-19 prediction app interface components

1. This is the coronavirus test button; when you click it, another interface will display (figure 4.28).

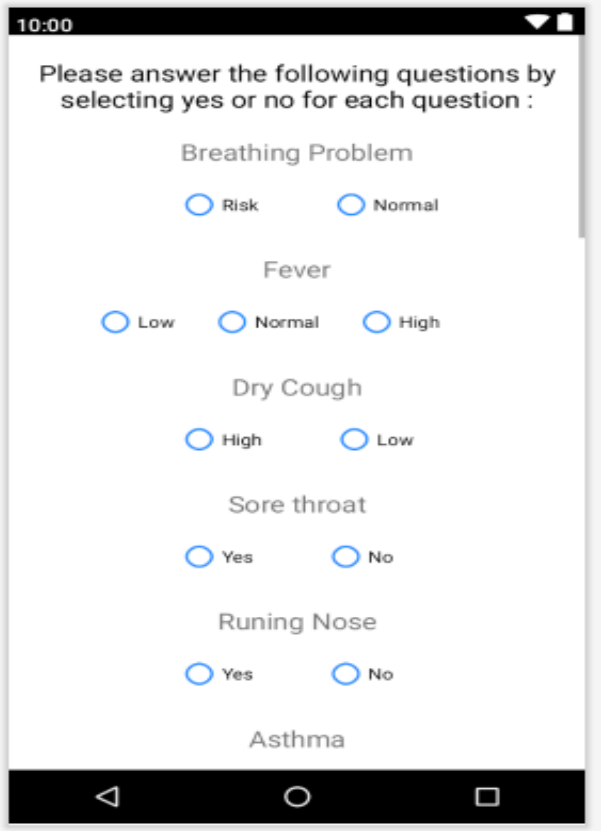

Figure 4.28: The coronavirus test interface

2. This is the COVID-19 prevention button; when you click it, another interface will display (figure 4.29).

Figure 4.29: The COVID-19 prevention interface

4.5 Conclusion

We have attempted to create **COVID-19 prediction** utilizing Deep Learning and Fuzzy logic throughout the sections of this chapter, demonstrating the technologies employed and their implementation step by step.

General conclusion and future work

Telemedicine or so-called Telehealth is a very important field in the medicine domain that has provided us opportunities to build a stronger health system. Especially during the COVID-19 epidemic, in this work, we have tried to tackle detection of this coronavirus with high technologies in deep learning and fuzzy logic as they are most effective which provides high-performance and accurate results.

Considering new technologies and research in COVID-19 illness prediction are emerging and being created, we believe it is necessary to expand our system. To do so, we propose certain suggestions:

- Attempting to get more data and retrain our model.
- Adding Arabic and Algerian dialect languages to the system.
- We'd look into related problems and put our model to the test on a variety of datasets to diagnose more diseases.

Bibliography

[1] Richard Wootton. “Telemedicine”. In: *Bmj* 323.7312 (2001), pp. 557–560.

[2] Shashi Gogia. “Rationale, history, and basics of telehealth”. In: *Fundamentals of Telemedicine and Telehealth*. Elsevier, 2020, pp. 11–34.

[3] Scott G Cunningham et al. “Definitions of eHealth”. In: *eHealth, care and quality of life*. Springer, 2014, pp. 15–30.

[4] Hans Oh et al. “What is eHealth?: a systematic review of published definitions”. In: *World Hosp Health Serv* 41.1 (2005), pp. 32–40.

[5] C Dario et al. “Opportunities and Challenges of eHealth and Telemedicine via Satellite”. In: (2004).

[6] Courtney Larson. *The History of Telemedicine - doxy.me*. May 2019. URL: https://doxy.me/en/blog/articles/history-of-telemedicine-2/ (visited on 03/21/2021).

[7] Siddharth Nair and Sasikala Mohan. “DESIGN & DEVELOPMENT OF VITAL SIGN MONITOR BASED ON WIRELESS SENSOR NETWORK AND TELEMEDICINE TECHNOLOGY”. PhD thesis. June 2014. DOI: 10.13140/RG.2.2.16453.14561.

[8] Mahendra Pal et al. “Severe acute respiratory syndrome coronavirus-2 (SARS-CoV-2): an update”. In: *Cureus* 12.3 (2020).

[9] *Coronavirus*. URL: https://www.who.int/health-topics/coronavirus/coronavirus#tab=tab_1 (visited on 05/29/2021).

[10] David J Cennimo. “Coronavirus Disease 2019 (COVID-19)”. In: (May 2021). URL: https://emedicine.medscape.com/article/2500114-overview#a1 (visited on 05/29/2021).

[11] *COVID-19*. URL: https://en.wikipedia.org/wiki/COVID-19 (visited on 05/29/2021).

[12] Ramesh Kumar et al. “COVID-19 diagnostic approaches: different roads to the same destination”. In: *Virusdisease* 31.2 (2020), pp. 97–105.

[13] Antonio Gulli and Sujit Pal. *Deep learning with Keras*. Packt Publishing Ltd, 2017.

[14] Lina Maria Rojas-Barahona. “Deep learning for sentiment analysis”. In: *Language and Linguistics Compass* 10.12 (2016), pp. 701–719.

[15] Ismail Ghallou. *Build your perceptron neural net from scratch*. Aug. 2017. URL: https://medium.com/@ismailghallou/build-your-perceptron-neural-net-from-scratch-e12b7be9d1ef (visited on 04/10/2021).

[16] Li Deng and Dong Yu. “Deep Learning: Methods and Applications”. In: *Foundations and Trends in Signal Processing* 7 (Jan. 2013). DOI: 10.1561/2000000039.

[17] Witold Pedrycz and Shyi-Ming Chen. *Deep Learning: Concepts and Architectures*. Springer, 2020, pp. 4–20.

[18] Amir Mosavi, Sina Ardabili, and Annamária R. Várkonyi-Kóczy. “List of Deep Learning Models”. In: *Engineering for Sustainable Future*. Ed. by Annamária R. Várkonyi-Kóczy. Cham: Springer International Publishing, 2020, pp. 202–214.

[19] Lotfi A Zadeh. “Fuzzy logic”. In: *Computer* 21.4 (1988), pp. 83–93.

[20] GORDON SCOTT. *Fuzzy Logic*. June 2019. URL: https://www.investopedia.com/terms/f/fuzzy-logic.asp (visited on 04/12/2021).

[21] Lotfi A Zadeh. “Is there a need for fuzzy logic?” In: *Information sciences* 178.13 (2008), pp. 2751–2779.

[22] *Fuzzy Logic | Introduction*. Oct. 2019. URL: https://www.geeksforgeeks.org/fuzzy-logic-introduction/ (visited on 04/26/2021).

[23] J.M. Mendel. “Fuzzy logic systems for engineering: a tutorial”. In: *Proceedings of the IEEE* 83.3 (1995), pp. 345–377. DOI: 10.1109/5.364485.

[24] Hans-Jürgen Zimmermann. *Fuzzy set theory—and its applications*. Springer Science & Business Media, 2011, pp. 139–181.

[25] X.E. Gros. "2 - Data Fusion – A Review". In: *NDT Data Fusion*. Ed. by X.E. Gros. Oxford: Butterworth-Heinemann, 1997, pp. 5–42. ISBN: 978-0-340-67648-6. DOI: https://doi.org/10.1016/B978-034067648-6/50004-9. URL: https://www.sciencedirect.com/science/article/pii/B9780340676486500049.

[26] Juri Yanase and Evangelos Triantaphyllou. "A systematic survey of computer-aided diagnosis in medicine: Past and present developments". In: *Expert Systems with Applications* 138 (2019), p. 112821.

[27] Lotfi Asker Zadeh. "The concept of a linguistic variable and its application to approximate reasoning—I". In: *Information sciences* 8.3 (1975), pp. 43–80.

[28] GEORGE J KLIR. "From classical mathematics to fuzzy mathematics: Emergence of a new paradigm for theoretical science". In: *Fuzzy logic in chemistry*. Elsevier, 1997, pp. 31–63.

[29] Jane Musengya. "COMMON PILOT CHANNEL POWER CONTROL FOR 3G CELLULAR NETWORKS TRAFFIC LOAD BALANCING BASED ON FUZZY LOGIC CONTROL". PhD thesis. Mar. 2017, pp. 18–19.

[30] Na Liu et al. "A novel intelligent classification model for breast cancer diagnosis". In: *Information Processing & Management* 56.3 (2019), pp. 609–623.

[31] Ahmed M Abdel-Zaher and Ayman M Eldeib. "Breast cancer classification using deep belief networks". In: *Expert Systems with Applications* 46 (2016), pp. 139–144.

[32] Mehrbakhsh Nilashi et al. "A knowledge-based system for breast cancer classification using fuzzy logic method". In: *Telematics and Informatics* 34.4 (2017), pp. 133–144.

[33] Ankur Makwana and Jaymin Patel. "Decision Support System for Heart Disease Prediction using Data Mining Classification Techniques". In: *International Journal of Computer Applications* 975.8887 (2008), pp. 1–5.

[34] Tanmay Kasbe and Ravi Singh Pippal. "Design of heart disease diagnosis system using fuzzy logic". In: *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*. 2017, pp. 3183–3187. DOI: 10.1109/ICECDS.2017.8390044.

[35] Ahmed Hamed, Ahmed Sobhy, and Hamed Nassar. "Accurate Classification of COVID-19 Based on Incomplete Heterogeneous Data using a K NN Variant Algorithm". In: *Arabian Journal for Science and Engineering* (2021), pp. 1–12.

[36] Tulin Ozturk et al. "Automated detection of COVID19 cases using deep neural networks with Xray images". In: *Computers in biology and medicine* (2020).

[37] Vinay Kumar Reddy Chimmula and Lei Zhang. "Time series forecasting of COVID-19 transmission in Canada using LSTM networks". In: *Chaos, Solitons & Fractals* 135 (2020), p. 109864.

[38] Goncalo Marques Marques, Deevyankar Agarwal, and Isabelde la Torre Diez. "Automated medical diagnosis of COVID19 through EfficientNet convolutional neural network". In: *Applied Soft Computing* (2020).

[39] Mohammed Ben-Mubarak et al. "Fuzzy Logic Based Self-Adaptive Handover Algorithm for Mobile WiMAX". In: *Wireless Personal Communications* 71 (July 2012), pp. 1421–1442. DOI: 10.1007/s11277-012-0883-0.

[40] Sagar Sharma and Simone Sharma. "Activation functions in neural networks". In: *Towards Data Science* 6.12 (2017), pp. 310–316.

[41] Guido van Rossum and the Python development team. *Python Tutorial Release 3.7.0*. Python Software Foundation, 2018.

[42] *python logo*. URL: https://www.python.org/ (visited on 05/14/2021).

[43] *What is Java technology and why do I need it?* URL: https://java.com/en/download/help/whatis_java.html (visited on 05/14/2021).

[44] *What is NumPy?* URL: https://numpy.org/doc/stable/user/whatisnumpy.html (visited on 05/14/2021).

[45] *Pandas documentation*. URL: https://pandas.pydata.org/docs/ (visited on 05/14/2021).

[46] *matplotlib Project description*. URL: https://pypi.org/project/matplotlib/ (visited on 05/14/2021).

[47] *Keras: Deep Learning library for Theano and TensorFlow*. URL: https://faroit.com/keras-docs/1.2.0/ (visited on 05/15/2021).

[48] *keras documentation*. URL: https://keras.io/ (visited on 05/15/2021).

[49] Fabian Pedregosa et al. "Scikit-learn: Machine learning in Python". In: *the Journal of machine Learning research* 12 (2011), pp. 2825–2830.

[50] *Scikit-learn documentation*. URL: https://scikit-learn.org/stable/ (visited on 05/15/2021).

[51] The scikit-image team. *The scikit-fuzzy Documentation, Release 0.3dev*. 2016.

[52] *Scikit-fuzzy documentation*. URL: https://pythonhosted.org/scikit-fuzzy/overview.html (visited on 05/15/2021).

[53] Ekaba Bisong. "Google colaboratory". In: *Building Machine Learning and Deep Learning Models on Google Cloud Platform*. Springer, 2019, pp. 59–64.

[54] *Google Colaboratory documentation*. URL: https://colab.research.google.com/notebooks/intro.ipynb (visited on 05/15/2021).

[55] *Your Machine Learning and Data Science Community*. URL: https://www.kaggle.com/ (visited on 05/15/2021).

[56] *Meet Android Studio*. URL: https://developer.android.com/studio/intro (visited on 05/15/2021).

[57] *The easiest way to use Python in an Android app*. URL: https://chaquo.com/chaquopy/ (visited on 06/13/2021).