République Algérienne Démocratique et Populaire
ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université de Belhadj Bouchaib d'Ain-Temouchent

Institut des Sciences
Département des Mathématiques et de l'Informatique

Thesis

To get master's degree in computer science
Option:
Network and data engineering

Presented by:
Mme. MANKOURI Amani
Mr. ZENASNI Elhachemi

Detecting Covid in chest X-ray using neural networks

Supervisor:
Dr BENDIABDALLAH Mohammed Hakim
Associate professor "B" `a U.B.B.A.T.

19/06/2022

Infront of the jury composed of:

President: Mme ABDERRAHIM Naziha	M.C.B	U.A.T.B.B
Examinators: Mr BOUAFIA Zohir	M.A.A	U.A.T.B.B.

2021/2022

Dedicate:

We dedicate this modest thesis to:

Our dear parents:

That no dedication can express what we owe them, for all their sacrifices, their kindness, their love, their tenderness, their support, and their prayers throughout our studies. May this work be the testimony of our deep love and our gratitude "May God keep you".

Our dear sisters and dear brothers:

For their constant encouragement and moral support, we dedicate this

this work in testimony of our great love and our infinite gratitude.

All our Families:

To all the MANKOURI & MOUFFAK and BENMEDJAHED & ZENASNI families, for their support throughout our academic journey, May this work be the fulfillment of your much alleged wishes, and the flight of your unwavering support.

All our friends:

For their help and moral support during the preparation of the dissertation.

Appreciation:

Above all, we thank the good GOD for having helped us to accomplish this modest work.

We take this opportunity to express our most sincere thanks and our gratitude to:

Our supervisor: Mr. BENDIABDALLAH Mohammed Hakim for his availability, his advice, guidance, and encouragement throughout our research.

Members of the jury Mme ABDERAHIM Naziha and Mr BOUAFIA Zouhir who agreed to evaluate and to review this work.

Our families and friends who, through their prayers and encouragement, were able to overcome all obstacles.

Finally, we cannot end these thanks without associating all the people who participated in the execution of this modest work.

Abstract:

The covid-19 virus infected the world's population, the doctors found themselves exhausted, and could not treat all this large number of patients, which led to great loss of human life.

The use of artificial intelligence and deep learning techniques help doctors perform medical diagnosis, allowing them to treat a greater number of patients and minimizing the error rate due to fatigue.

We have proposed deep artificial neural network architectures, allowing a rapid and efficient detection and localization of regions affected by covid-19 in chest X-ray images. A web platform has been implemented to allow patients to make a pre-diagnosis at any time. This project will not only help patients, but it will also help doctors to make the diagnosis.

Key words: Covid detection, X-rays, Artificial intelligence, Machine learning, Deep learning, Convolutional Neural Network, Flask, TensorFlow, Flask, Python, Html, CSS, JS.

Résumé:

Le virus du covid-19 a infecté la population mondiale, les médecins se sont trouvés épuisés, et ne pouvaient pas traiter tout ce grand nombre de malades, ce qui a conduit à de grandes pertes de vie humaine.

L'utilisation des techniques de l'intelligence artificielle et de l'apprentissage profond permettent d'aider les médecins a effectué le diagnostic médical, en les permettant à traiter un plus grand nombre de malades et en minimisant le taux d'erreur dû à la fatigue.

Nous avons proposé des architectures de réseaux de neurones artificiels profond, permettant la détection et la localisation rapide et efficace des régions affectées par le covid-19 dans les images radios thoraciques. Une plateforme web a été implémenté afin de permettre aux patients de faire un pré-diagnostic à tout moment. Ce projet ne va pas juste aider les patients mais il va aussi aider les médecins à effectuer le diagnostic.

Mots clés : Détection du covid , Radios thoraciques, Intelligence Artificielle , Machine learning, Deep learning, Convolutional Neural Network, Flask, TensorFlow, Python, Html, CSS, JS.

ملخص:

أصاب فيروس كوفيد 19 العالم بكله ، و وجد الأطباء أنفسهم منهكين ، لم يتمكنوا من علاج كل هذا العدد الكبير من المرضى ، مما أدى إلى خسائر كبيرة في الأرواح البشرية.

يساعد استخدام الذكاء الاصطناعي وتقنيات التعلم العميق الأطباء في إجراء التشخيص الطبي، مما يسمح لهم بمعالجة عدد أكبر من المرضى وتقليل معدل الخطأ الناتج عن التعب.

لقد اقترحنا بنى شبكات عصبية اصطناعية عميقة ، مما يسمح بالكشف السريع والفعال عن المناطق المتأثرة بـ كوفيد 19 وتحديد موقعها في صور الأشعة السينية للصدر. تم تنفيذ منصة الويب للسماح للمرضى بإجراء تشخيص مسبق في أي وقت. لن يساعد هذا المشروع المرضى فحسب ، بل سيساعد الأطباء أيضًا على إجراء التشخيص.

كلمات مفتاحية:

كشف عن كوفيد، كورونا، الأشعة السينية، الذكاء الاصطناعي، التعلم الآلي، التعلم العميق، الشبكات العصبية الصناعية، فلاسك، بايثون، أش تي ام أل، سي اس اس، جافا سكريبت

Table of Contents

Illustrations table

General Introduction

In the era of value-based healthcare, digital innovation, and big data, clinical decision support systems have become vital for organizations seeking to improve care delivery.

“A lot of these mathematical techniques have been around for a long time, but the reason that we're seeing them come to fore now is that things have gone digital,”. [1]

“We used to do radiology on film. Now we have imaging digitally. We used to have a patient chart that was paper in a folder, now charts are electronic. We have computing that is much faster than what we had, say, 20 years ago, when training machine learning models was very computationally intensive.” [1]

Number of research studies suggesting that AI can perform as well as or better than humans at key healthcare tasks, such as diagnosing disease. Today, algorithms are already outperforming radiologists at spotting malignant tumors, and guiding researchers in how to construct cohorts for costly clinical trials. However, for a variety of reasons, we believe that it will be many years before AI replaces humans for broad medical process domains. In this project, we describe both the potential that AI offers to automate aspects of care and some of the barriers to rapid implementation of AI in healthcare.

A coronavirus is a large group of viruses that includes the Middle East respiratory syndrome coronavirus (MERS-CoV), the severe acute respiratory syndrome coronavirus (SARS-CoV), and the most recent one, SARS-CoV-2, also known as coronavirus disease 2019 (COVID-19). In early December 2019, SARS-CoV-2 originated and eventually affected more than 250 million people globally.

In our project our goal is to help this pandemic by offering a website who helps patients and doctors to do the fasts examinations, we aim to create an artificial neural network that detects covid in chest X-Rays.

This report is divided in four chapters. In the first chapter we will talk about the artificial intelligence and how it became so important in human life. In the second chapter we will introduce the machine learning, deep leaning, and artificial neural networks. In the third chapter we will see our implementation of the artificial neural network with a heatmap. In the last chapter we will talk about the tools that we used to create our neural network, heatmap and the website.

Chapter 1: Artificial intelligence

I. Introduction

Since the first century BC, man always tried to create machine capable of imitate human reasoning. The notion of artificial intelligence has been found by the mathematic Alan Turing in 1950 in his book Computing Machinery and Intelligence. In this book raises question of giving the machines a form of intelligence. Turing has introduced "Turing Test" that developers still use nowadays.

The term of "Artificial intelligence" has been created recently by John McCarthy in 1955. In 1956 Jhon and his collaborators had organized a lecture in the name of « Dartmouth Summer Research Project on Artificial Intelligence » which gave birth to **"Machine learning", "Deep Learning", "Predictive analyses"** and **"Prescriptive analyses"** with that new domain has appeared "**Data science"**.

Today, Humans and machines generate data way faster than only human brain. The artificial intelligence is the base of all the learning by computer and represent the future complex decision-making processes. Computers can calculate all the different combinations and the best possible permutation effectively to make the best decision.

In this chapter we will detail what is Artificial intelligence, its benefits, Turing test and the main algorithms of AI.

II. Definition

AI which stands for artificial intelligence is the simulation of the human intelligence by the machine. This technology is particularly wide with-it advanced involvement for the humanity. In the simplest terms, AI refers to systems or machines that imitate human intelligence to perform tasks and can iteratively improve themselves based on the information they collect. AI manifests in several forms. A few examples are: [1]

- Chatbots use AI to understand customer problems faster and provide more efficient answers
- Intelligent assistants use AI to parse critical information from large free-text datasets to improve scheduling
- Recommendation engines can provide automated recommendations for TV shows based on users' viewing habits

In the process of creating this simulation we have three phases. [2]

- First, "Learning" by the acquisition of the information and the usage rules.
- Then, "Reasoning" by using the given rules to get approximate or a definitive decision.
- Finally, "Autocorrection" in the last phase the machine autocorrects its results.

AI is much more about the process and the capability for superpowered thinking and data analysis than it is about any format or function. Although AI brings up images of high-functioning, human-like robots taking over the world, AI isn't intended to replace humans. It's intended to significantly enhance human capabilities and contributions. That makes it a very valuable business asset. [1]
AI needs 3 necessary components: [3]

- It Systems
- Data avec Control Systems. so that the machine can act as the human, it needs huge amount of data and strong treatment capacity
- Advanced algorithms

AI has become a catchall term for applications that perform complex tasks that once required human input such as communicating with customers online or playing chess. The term is often used interchangeably with its subfields, which include **machine learning** and **deep learning**. For example, **machine learning** is focused on building systems that learn or improve their performance based on the data they consume. It's important to note that although **all machine learning is AI**, **not all AI is machine learning.** [1]

To get the full value from AI, many companies are making significant investments in data science teams. Data science, an interdisciplinary field that uses scientific and other methods to extract value from data, combines skills from fields such as statistics and computer science with business knowledge to analyze data collected from multiple sources. [1]

III. Turing Test

The Turing Test is a method of inquiry in artificial intelligence (AI) for determining whether or not a **computer is capable of thinking like a human being**. The test is named after **Alan Turing**, the founder of the Turing Test and an English computer scientist, cryptanalyst, mathematician, and theoretical biologist. [4]

Turing proposed that a computer can be said to possess artificial intelligence if it can imitate human responses under specific conditions. The original Turing Test requires **three terminals**, each of which is physically separated from the other two. **One** terminal is operated by a **computer**, while the other **two** are operated by **humans.** [4]

During the test, one of the human's functions as the questioner, while the second human and the computer function as respondents. The questioner interrogates the respondents within a specific subject area, using a specified format and context. After a preset length of time or number of questions, the questioner is then asked to decide which respondent was human and which was a computer. [4]

The test is repeated many times. If the questioner makes the correct determination in half of the test runs or less, the computer is considered to have artificial intelligence because the questioner regards it as "just as human" as the human respondent. [4]

IV. The benefits and challenges of operationalizing AI

There are numerous success stories that prove AI's value. Organizations that add machine learning and cognitive interactions to traditional business processes and applications can greatly improve user experience and boost productivity. Three factors are driving the development of AI across industries:

- **Affordable, high-performance computing capability is readily available.** The abundance of commodity compute power in the cloud enables easy access to affordable, high-performance computing power. Before this development, the only computing environments available for AI were non-cloud-based and cost prohibitive.
- **Large volumes of data are available for training.** AI needs to be trained on lots of data to make the right predictions. The emergence of different tools for labeling data, plus the ease and affordability with which organizations can store and process both structured and unstructured data, is enabling more organizations to build and train AI algorithms.
- **Applied AI delivers a competitive advantage.** Enterprises are increasingly recognizing the competitive advantage of applying AI insights to business objectives and are making it a business wide priority. For example, targeted recommendations provided by AI can help businesses make better decisions faster. Many of the features and capabilities of AI can lead to lower costs, reduced risks, faster time to market, and much more. [1]

V. Artificial Intelligence applications

We can find artificial intelligence in different technologies, here are some of them: [5]

1. Automatization

What makes a system run automatically. For example, RPA (Robotic Process Automation) can be programmed to run repetitive tasks faster than humans

2. Computer vision

This technology captures and analyze visual information with a camera lens. We used this in signature identification or medical images analyze

3. NLP (Natural language processing)

The treatment of naturel language is the treatment of human language by a machine. For example, detection of spams is an old one. However, actual approaches are based on machine learning. Including text translation, sentiments analyze and vocal recognition

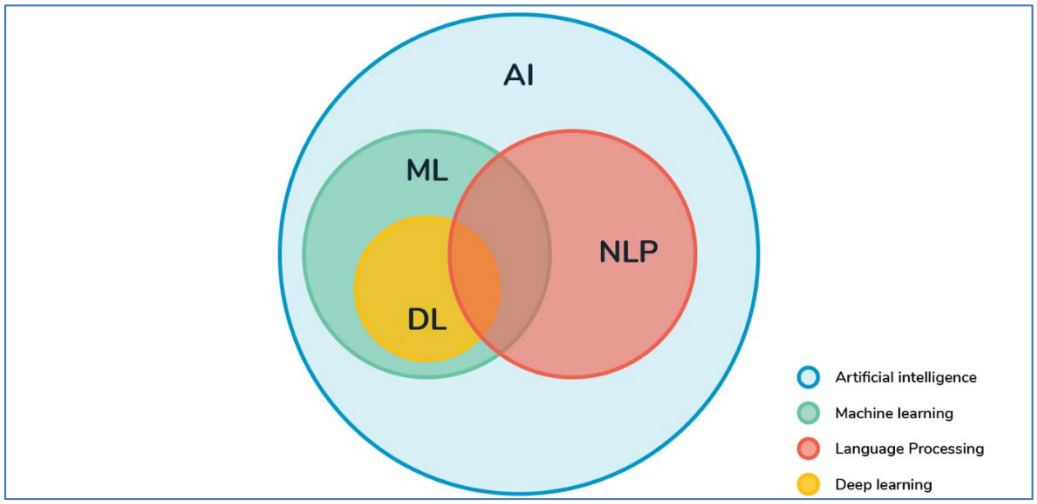

Figure 1 : AI and NLP

4. Robotics

It's about the conception and construction of robots, which are used in assembly lines of automobile production, or even by the NASA (*National Aeronautics and Space Administration)* to move big objects in the space. Scientists tried to involve the machine learning to construct these robots so that they would have the ability to interact in social context. [5]

5. Fraud's detection

In finance, AI is used in two manners.

- Application for credits demands using AI to evaluate the creditworthiness of consumers.
- More powerful AI engine are charged to supervise and detect fraud payment by bank cart in real time

6. Virtual Client Service (VCS)

Call centers uses an VCS to predict clients demands and answer human intervention. The speech recognition and Human dialogue simulator make the first point of interaction with the client service. The more complexed demands need to be done by a human. [5]

7. Autonomous vehicles

An autonomous car is a vehicle capable of sensing its environment and operating without human involvement. A human passenger is not required to take control of the vehicle at any time, nor is a human passenger required to be present in the vehicle at all. An autonomous car can go anywhere a traditional car goes and do everything that an experienced human driver does.

The Society of Automotive Engineers (SAE) currently defines 6 levels of driving automation ranging from Level 0 (fully manual) to Level 5 (fully autonomous). These levels have been adopted by the U.S. Department of Transportation. [6]

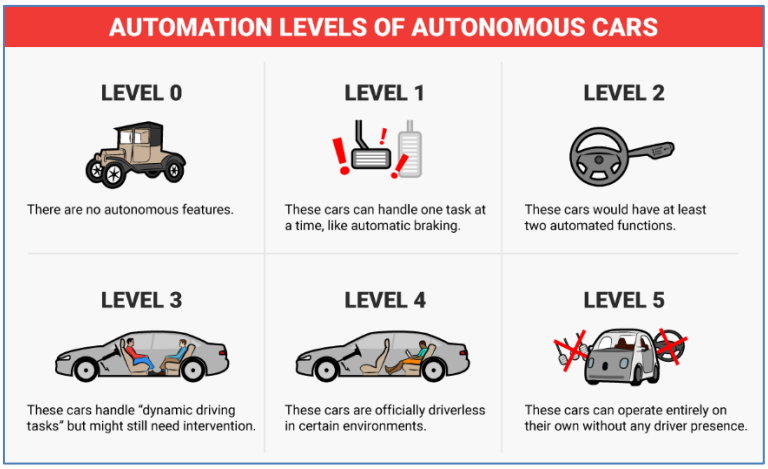

Figure 2 : Levels of Driving Automation

Autonomous cars rely on sensors, actuators, complex algorithms, machine learning systems, and powerful processors to execute software.

8. Health care

Most of AI technologies have immediate relevance to the healthcare field, but the specific processes and tasks they support vary widely.

a) Clinical decision support

CDS stands for **Clinical decision support** is a sophisticated health IT component which provides clinicians, staff, patients or other individuals with knowledge and person-specific information, intelligently filtered or presented at appropriate times, to enhance health and health care. [7]

It requires computable biomedical knowledge, person-specific data, and a reasoning or inferencing mechanism that combines knowledge and data to generate and present helpful information to clinicians as care is being delivered. This information must be organized in a way that supports the current workflow, allowing the user to make an informed decision quickly and act. Different types of CDS may be ideal for different processes of care in different settings. [7]

CDS encompasses a variety of tools to enhance decision-making in the clinical workflow. These tools include computerized alerts and reminders to care providers and patients by: [7]

- clinical guidelines.
- condition-specific order sets.
- focused patient data reports and summaries.
- documentation templates.
- diagnostic support, and contextually relevant reference information, among other tools.

CDS tools can analyze large volumes of data and suggest next steps for treatment, flagging potential problems and enhancing care team efficiency. The majority of CDS applications operate as components of comprehensive EHR systems, although stand-alone CDS systems are also used.

Clinical decision support systems are computer-based programs that analyze data within EHRs (electronic health record) to provide prompts and reminders to assist health care providers in implementing evidence-based clinical guidelines at the point of care. [8]

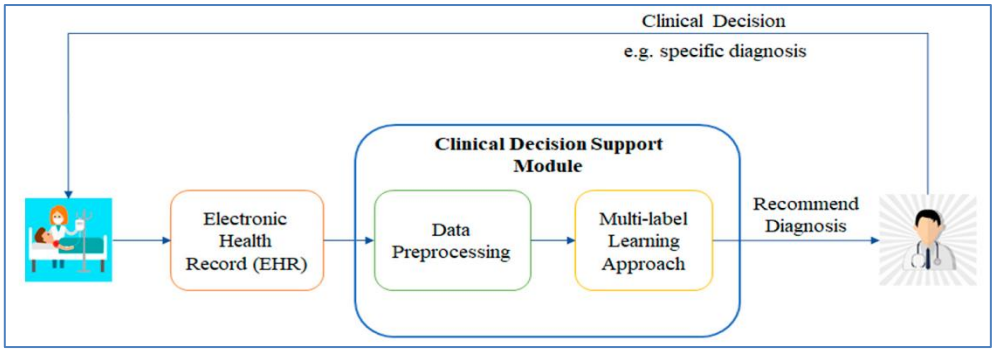

Figure 3 :Clinical decision support system

b) *Types* of Clinical decision support systems

CDSS are frequently classified as knowledge-based or non-knowledge based.

In **knowledge-based** systems, rules are created (IF-THEN statements), with the system retrieving data to evaluate the rule, and producing an action or output. Rules can be made using literature-based, practice-based, or patient-directed evidence. [9]

Non-knowledge based still require a data source, but the decision leverages artificial intelligence (AI), machine learning (ML), or statistical pattern recognition, rather than being programmed to follow expert medical knowledge. Non-knowledge based CDSS, although a rapidly growing use case for AI in medicine, are rife with challenges including problems understanding the logic that AI uses to produce recommendations (black boxes), and problems with data availability. They have yet to reach widespread implementation. Both types of CDSS have common components with subtle differences. [9]

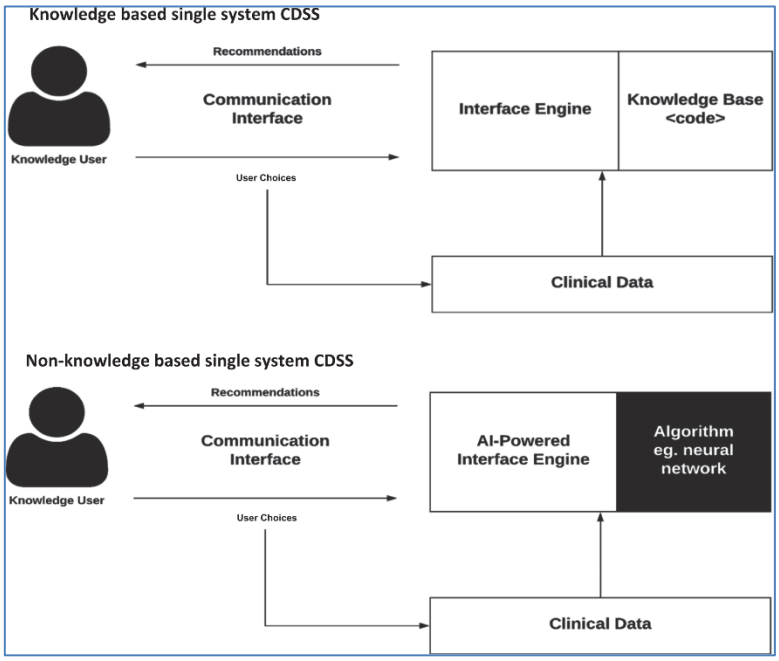

Figure 4 :Knowledge vs non-knowledge CDSS

c) *Electronic health records*

An electronic health record (EHR) is a digital version of a patient's paper chart. EHRs are **real-time**, patient-centered records that make information available instantly and securely to authorized users.

While an EHR does contain the medical and treatment histories of patients, an EHR system is built to go beyond standard clinical data collected in a provider's office and can be inclusive of a broader view of a patient's care. [10]

EHRs are a vital part of health IT and can contain a patient's:

- medical history,
- diagnoses,
- medications,
- treatment plans,
- immunization dates,
- allergies,
- radiology images, and laboratory and test results

Allow access to evidence-based tools that providers can use to make decisions about a patient's care, automate and streamline provider workflow

One of the key features of an EHR is that health information can be created and managed by authorized providers in a digital format capable of being shared with other providers across more than one health care organization. EHRs are built to share information with other health care providers and organizations such as laboratories, specialists, medical imaging facilities, pharmacies, emergency facilities, and school and workplace clinics so they contain information from all clinicians involved in a patient's care. [10]

d) AI in healthcare

Artificial intelligence simplifies the lives of patients, doctors and hospital administrators by performing tasks that are typically done by humans, but in less time and at a fraction of the cost. There is numerous ways AI can positively impact the practice of medicine, whether it's through speeding up the pace of research or helping clinicians make better decisions. Here are some examples of how AI could be used:

- **Disease detection and diagnosis:**

AI can predict and diagnose disease at a faster rate than most medical professionals. healthcare applications. In healthcare, delays can mean the difference between life and death, helping care teams react faster with AI-powered healthcare solutions. The AI products can detect issues and notify care teams quickly, enabling providers to discuss options, provide faster treatment decisions and ultimately save lives.

AI never needs to sleep. Machine learning models could be used to observe the vital signs of patients receiving critical care and alert clinicians if certain risk factors increase. While medical devices like heart monitors can track vital signs, AI can collect the data from those devices and look for more complex conditions, such as sepsis. [11]

- **Personalized disease treatment:**

Precision medicine could become easier to support with virtual AI assistance. Because AI models can learn and retain preferences, AI has the potential to provide customized real-time recommendations to patients around the clock. Rather than having to repeat information with a new person each time, a healthcare system could offer patients around-the-clock access to an AI-powered virtual assistant that could answer questions based on the patient's medical history, preferences and personal needs. [11]

- **AI in medical imaging:**

AI is already playing a prominent role in medical imaging. Research has indicated that AI powered by artificial neural networks can be just as effective as human radiologists at detecting signs of breast cancer as well as other conditions. In addition to helping clinicians spot early signs of disease, AI can also help make the staggering number of medical images that clinicians have to keep track of more manageable by detecting vital pieces of a patient's history and presenting the relevant images to them.

- **Clinical trial efficiency:**

A lot of time is spent during clinical trials assigning medical codes to patient outcomes and updating the relevant datasets. AI can help speed this process up by providing a quicker and more intelligent search for medical codes. [11]

- **Accelerated drug development:**

Drug discovery is often one of the longest and most costly parts of drug development. AI could help reduce the costs of developing new medicines in primarily two ways:

  - creating better drug designs
  - finding promising new drug combinations.

With AI, many of the big data challenges facing the life sciences industry could be overcome. [11]

e) Benefits of AI in medicine

- **Informed patient care**

Integrating medical AI into clinician workflows can give providers valuable context while they're making care decisions. A trained machine learning algorithm can help cut down on research time by giving clinicians valuable search results with evidence-based insights about treatments and procedures while the patient is still in the room with them. [11]

- **Error reduction**

There is some evidence that AI can help improve patient safety. [11]

- **Reducing the costs of care**

AI could reduce costs across the healthcare industry. Some of the most promising opportunities include reducing medication errors, customized virtual health assistance, fraud prevention, and supporting more efficient administrative and clinical workflows. [11]

- **Increasing doctor-patient engagement**

AI can help provide around-the-clock support through chatbots that can answer basic questions and give patients resources when their provider's office isn't open. AI could also potentially be used to triage questions and flag information for further review, which could help alert providers to health changes that need additional attention. [11]

- **Providing contextual relevance**

One major advantage of deep learning is that AI algorithms can use context to distinguish between different types of information. For example, if a clinical note includes a list of a patient's current medications along with a new medication their provider recommends, a well-trained AI algorithm can use natural language processing to identify which medications belong in the patient's medical history. [11]

VI. AI and Covid-19

Owing to the COVID-19 pandemic, non-contact healthcare has recently gained immense importance. Although wearable devices can be effectively used to monitor SARS-CoV-2 affected patients, they are at the cost of potential exposure to the virus. Healthcare workers, particularly nurses, may easily be affected when attaching, changing, or removing wearable devices. All possible precautions such as wearing masks or gloves, when taken by the healthcare providers, will reduce the risk of contracting the virus; nevertheless, using non-contact technology will successfully eliminate all possibilities through which the virus can be transmitted. [12]

Healthcare technology merged with AI can significantly reduce the work pressure on hospital staff. For instance, radio frequency sensing technology can collect information from a patient's body, and the passing of this information through AI algorithms will yield valuable results without any direct involvement of hospital workers.

Remote non-contact sensing technologies integrated with smart machine learning algorithms can provide correct outcomes in real time, which can easily be used by the clinician to monitor and diagnose a disease. Non-contact sensing can assist healthcare staff in detecting and monitoring SARS-

CoV-2 affected individuals. This will enable healthcare workers to rapidly diagnose the disease and make the right decisions on time.

To fight against SARS-CoV-2, the monitoring of crucial signs is important. These signs include respiratory problems, cough, fever, and in some cases, the affected cardiovascular system. Shortness of breath is a vital sign when a patient experiences serious difficulty. To detect abnormal respiration, non-contact techniques can be used for monitoring diseased patients. In addition, non-contact methods can also be used to monitor the heart rate of patients suffering from the affected cardiovascular system.

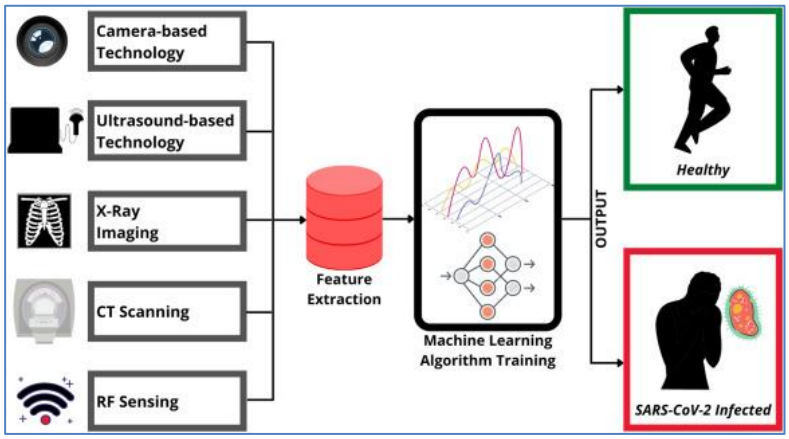

Figure 5: Covid-19 and AI

1. Camera-based technologies

One of the key symptoms of COVID-19 is shortness of breath (or respiratory problems), which affects chest movement. **Camera-based technologies** such as smartphones and thermal or depth cameras capture the video footage of the chest movements, which can then be analyzed by intelligent machine learning algorithms to detect any abnormalities in the **breathing pattern**. [12]

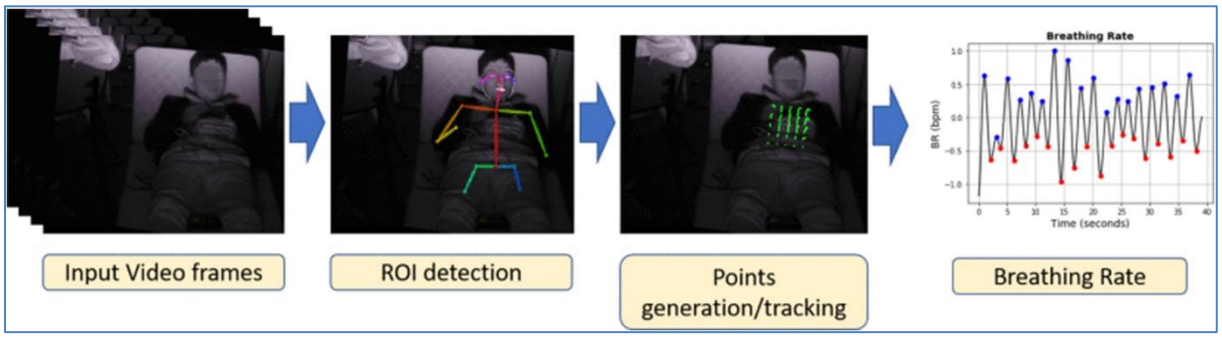

Figure 6: Camera-base tech and Covid-19

2. Ultrasound-based technology

Ultrasound-based technology uses **high-frequency sound waves** to capture the body motion. To detect **abnormal respiration**, an ultrasound machine generates sound waves that, upon bouncing off

from distinct sections of the body**, generate echoes** that are identified by the probe and are exploited to produce a stirring image. To reduce the risk of infection from COVID-19 patients, non-contact ultrasound-based technology can be utilized by monitoring abnormal respiratory activity in the lungs. [12]

3. X-radiation (X-ray) imaging

X-ray images can be used to detect and monitor the **symptoms of patients** with COVID-19. For instance, X-ray images of the lungs are used to detect abnormal respiration. In the past, X-ray imaging has been used to detect pneumonia. However, pneumonia and COVID-19, both as diseases, **affect the respiratory system**, and to detect any anomalies in the respiratory system, **X-ray imaging processed through AI algorithms** can be effectively used to monitor COVID-19 symptoms. As limitations of X-ray imaging, it requires a professional analysis, and the equipment is costly. [12] [13]

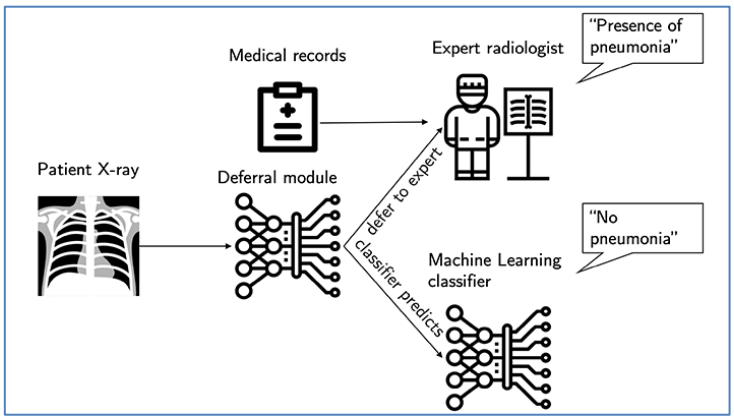

Figure 7 : X-ray and AI

4. Computerized tomography (CT) scanning

CT scanning technology involves generating X-ray imaging of a patient's chest to produce a **3D image of the lungs**. The outcome of the images **reveals any types of abnormalities** that can eventually be utilized to detect symptoms of COVID-19. Research has shown that CT scanning has an exposure sensitivity of **86%–98%** and can hence monitor any activity of SARS-CoV-2 infection in the lungs. [12]

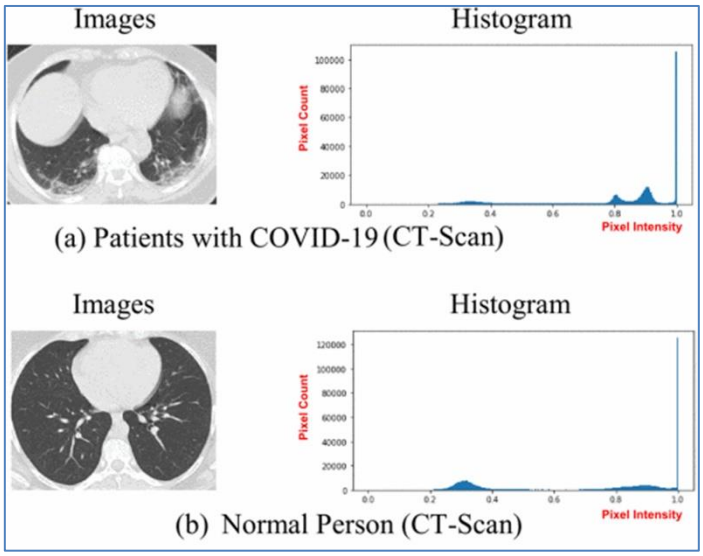

Figure 8 : CT Scanning

5. RF sensing

RF sensing primarily consists of Wi-Fi and **radar-based technologies**. Radar-based technology uses a frequency-modulated continuous wave to observe the Doppler effect when the whole body or part of the body moves. This technology can be effectively used to **monitor respiratory abnormalities**. Images taken by the radar system can be processed using intelligent machine-learning algorithms. [12]

Once trained, these algorithms can **detect anomalies in images**. Moreover, Wi-Fi-based technology comprises wireless channel information. Wireless channel information is used to define the propagation properties of the RF signal, including the abnormalities generated by the human body parts. Wi-Fi-based sensing technology used to monitor patients with COVID-19 has opened new doors for research in terms of healthcare. The received signal strength indicator obtained from Wi-Fi signals can be utilized to monitor distinct human activities. [12]

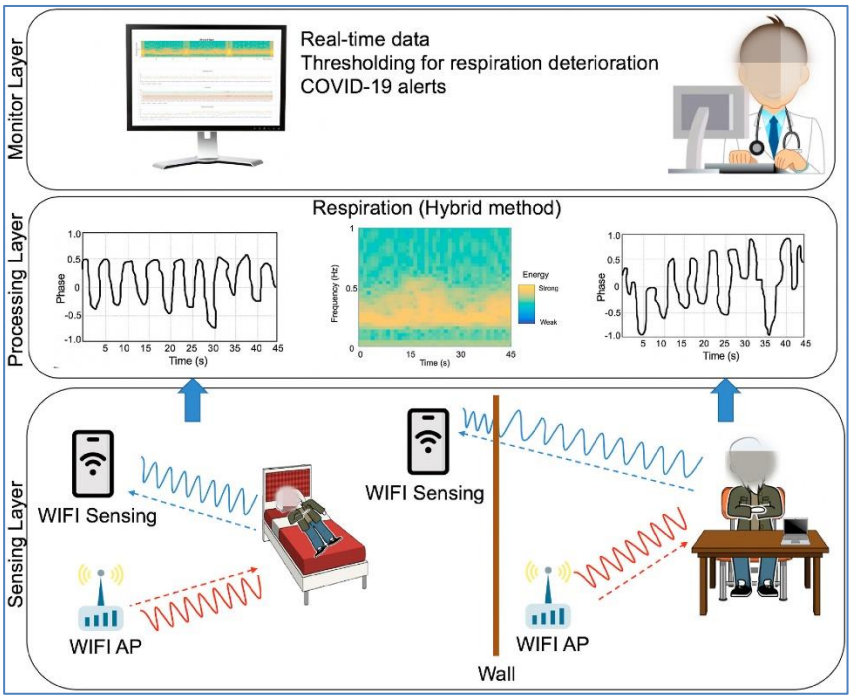

Figure 9 : RF sensing

VII. Machine Learning

Machine learning is a branch of **artificial intelligence (AI)** and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy.

Machine learning is an important component of the growing field of data science. Using statistical methods, algorithms are trained to make classifications or predictions, uncovering key insights within data mining projects. As big data continues to expand and grow, the market demand for data scientists will increase, requiring them to assist in the identification of the most relevant business questions and subsequently the data to answer them. [14]

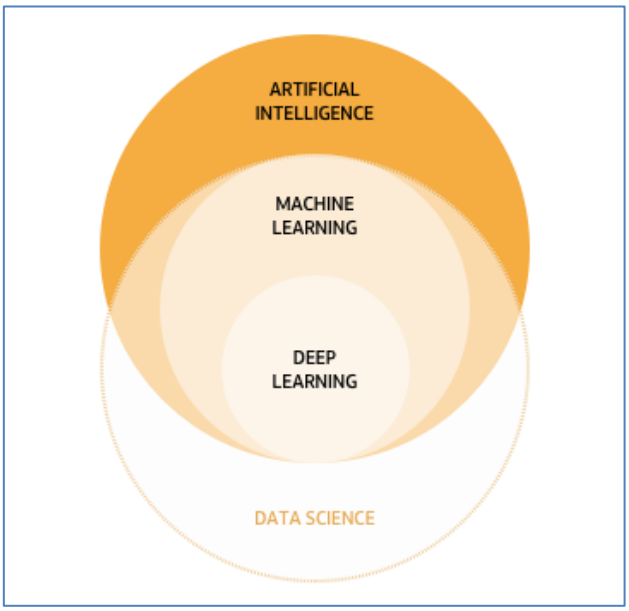

Figure 10: AI, Machine Learning, Deep learning, and Data Science relation

1. Machine Learning, Deep Learning and Neural Networks

Machine learning, deep learning, and neural networks are all sub-fields of artificial intelligence. However, deep learning is a sub-field of machine learning, and neural networks is a sub-field of deep learning. [15]

The way in which deep learning and machine learning differ is in how each algorithm learns. Deep learning automates much of the feature extraction piece of the process, eliminating some of the manual human intervention required and enabling the use of larger data sets. [15]

Classical, or **"non-deep",** machine learning is more dependent on human intervention to learn. Human experts determine the set of features to understand the differences between data inputs, usually requiring more structured data to learn. [15]

"Deep" machine learning can leverage labeled datasets, also known as supervised learning, to inform its algorithm, but it doesn't necessarily require a labeled dataset. It can ingest unstructured data in its raw form (e.g. text, images), and it can automatically determine the set of features which distinguish different categories of data from one another. Unlike machine learning, it doesn't require human intervention to process data, allowing us to scale machine learning in more interesting ways. Deep learning and neural networks are primarily credited with accelerating progress in areas, such as computer vision, natural language processing, and speech recognition. [15]

Neural networks, or artificial neural networks (ANNs), are comprised of a node layer, containing an input layer, one or more hidden layers, and an output layer. Each node, or artificial neuron, connects to another and has an associated weight and threshold. If the output of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the network. Otherwise, no data is passed along to the next layer of the network. [15]

The **"deep"** in deep learning is just referring to the depth of layers in a neural network. A neural network that consists of more than three layers—which would be inclusive of the inputs and the output—can be considered a deep learning algorithm or a deep neural network. A neural network that only has two or three layers is just a basic neural network. [15]

2. Machine learning procedure

The learning system of a machine learning algorithm into three main parts. [15]

- **Decision Process**: In general, machine learning algorithms are used to make a prediction or classification. Based on some input data, which can be labelled or unlabeled, your algorithm will produce an estimate about a pattern in the data.
- **Error Function**: An error function serves to evaluate the prediction of the model. If there are known examples, an error function can make a comparison to assess the accuracy of the model.
- **Model Optimization Process**: If the model can fit better to the data points in the training set, then weights are adjusted to reduce the discrepancy between the known example and the model estimate. The algorithm will repeat this evaluate and optimize process, updating weights autonomously until a threshold of accuracy has been met.

3. Machine learning algorithms

Machine learning algorithms are classified into 4 types: [15]

- Supervised Learning
- Unsupervised Learning
- Semi-supervised Learning
- Reinforcement Learning

a) Supervised learning

Supervised learning, also known as supervised machine learning, is defined by its use of **labeled** datasets to train algorithms that to classify data or predict outcomes accurately. As input data is fed into the model, it adjusts its weights until the model has been fitted appropriately. This occurs as part of the cross validation process to ensure that the model avoids **overfitting** or **underfitting**. [16]

Supervised learning helps organizations solve for a variety of real-world problems at scale, such as classifying spam in a separate folder from your inbox. Some methods used in supervised learning include neural networks, naïve bayes, linear regression, logistic regression, random forest, support vector machine (SVM), and more. [15]

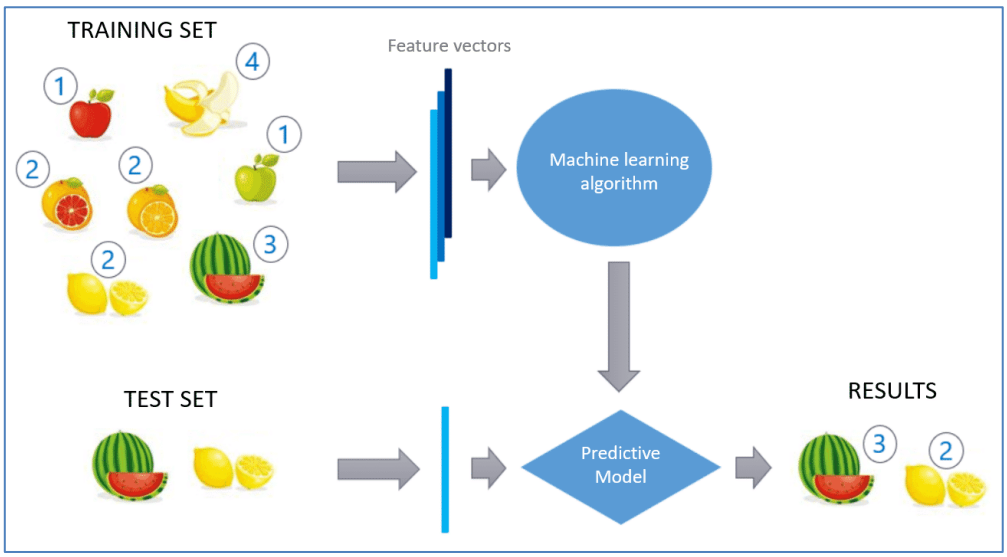

Figure 11: Supervised learning

b) *Unsupervised learning*

Unsupervised learning, also known as unsupervised machine learning, uses machine learning algorithms to analyze and **cluster** unlabeled datasets. These algorithms discover hidden patterns or data groupings without the need for human intervention. Its ability to discover similarities and differences in information make it the ideal solution for exploratory data analysis, cross-selling strategies, customer segmentation, image, and pattern recognition. [15]

It's also used to reduce the number of features in a model through the process of dimensionality reduction; **principal component analysis (**PCA) and **singular value decomposition** (SVD) are two common approaches for this. Other algorithms used in unsupervised learning include neural networks, k-means clustering, probabilistic clustering methods, and more. [16]

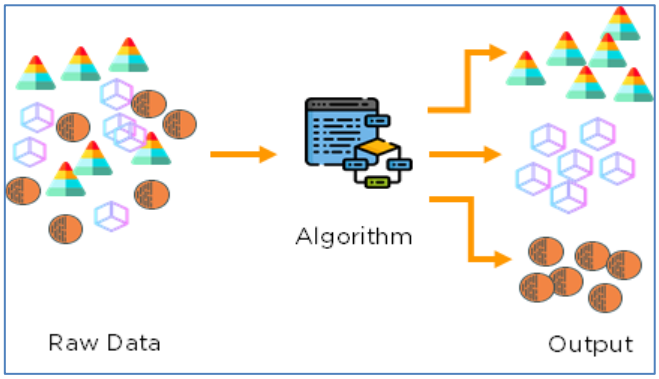

Figure 12 : Unsupervised learning

c) *Semi-supervised learning*

Semi-supervised learning offers a medium between supervised and unsupervised learning. During training, it uses a smaller labeled data set to guide classification and feature extraction from a larger, unlabeled data set. Semi-supervised learning can solve the problem of having not enough labeled data (or not being able to afford to label enough data) to train a supervised learning algorithm. [15]

The semi-supervised estimators can make use of this additional unlabeled data to better capture the shape of the underlying data distribution and generalize better to new samples. These algorithms can perform well when we have a very small number of labeled points and a large number of unlabeled points.

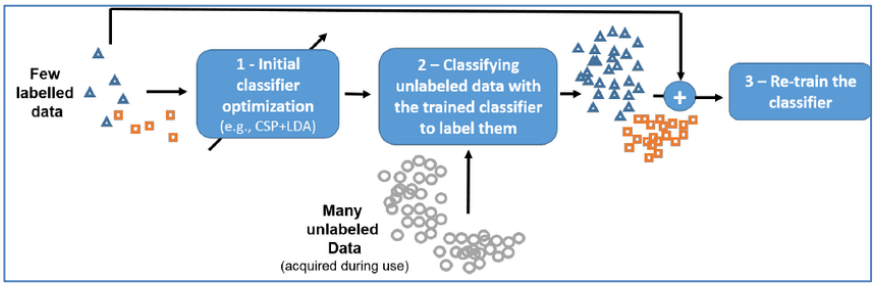

Figure 13 : Semi-supervised learning

d) Reinforcement learning

Reinforcement learning is a behavioral machine learning model that is similar to supervised learning, but the algorithm isn't trained using sample data. This model learns as it goes by using trial and error. A sequence of successful outcomes will be reinforced to develop the best recommendation or policy for a given problem. [15]

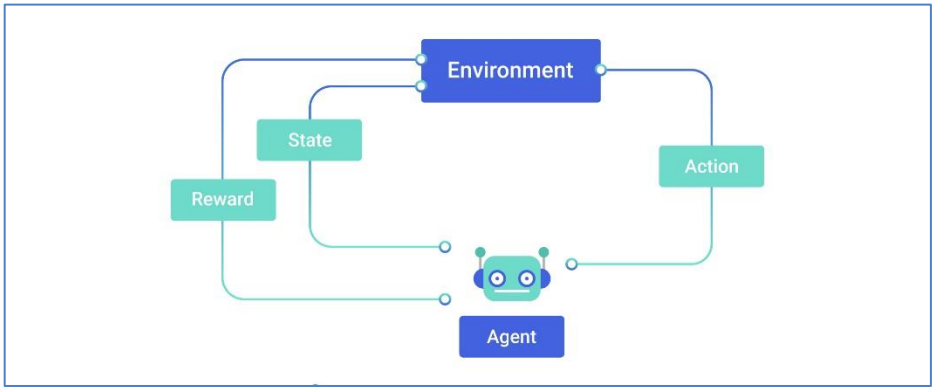

Figure 14 : Reinforcement learning

The Reinforcement Learning problem involves an **agent** exploring an unknown environment to achieve a goal. RL is based on the hypothesis that all goals can be described by the maximization of expected cumulative **reward**. The agent must learn to sense and perturb **the state of the environment** using its **actions** to derive maximal reward. The formal framework for RL borrows from the problem of optimal control of Markov Decision Processes (MDP). [17]

The main elements of an RL system are:

- The agent or the learner
- The environment the agent interacts with
- The policy that the agent follows to take actions
- The reward signal that the agent observes upon taking actions

A useful abstraction of the reward signal is the value function, which faithfully captures the 'goodness' of a state. While the reward signal represents the immediate benefit of being in a certain state, the value function captures the cumulative reward that is expected to be collected from that state on, going into the future. The objective of an RL algorithm is to discover the action policy that maximizes the average value that it can extract from every state of the system. [17]

4. Overfitting & Underfitting

a) Overfitting:

Overfitting is a concept in data science, which occurs when a statistical model fits **exactly against its training data**. When this happens, the algorithm cannot perform accurately against **unseen data**, defeating its purpose. Generalization of a model to new data is ultimately what allows us to use machine learning algorithms every day to make predictions and classify data. [18]

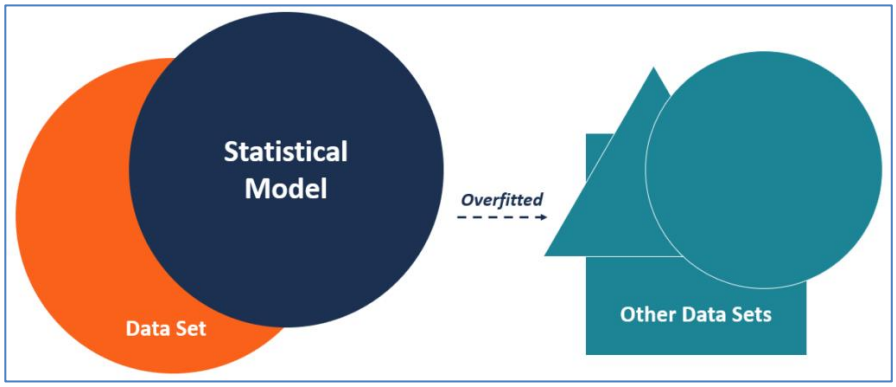

Figure 15: Overfitting

When machine learning algorithms are constructed, they leverage a sample dataset to train the model. However, when the **model trains for too long** on sample data or when the model is too complex, it can start to learn the "noise," or **irrelevant information**, within the dataset. When the model memorizes the noise and fits too closely to the training set, the model becomes **"overfitted,"** and it is unable to generalize well to new data. If a model **cannot generalize well to new data**, then it will not be able to perform the classification or prediction tasks that it was intended for. Low error rates and a high variance are good indicators of overfitting. [18]

To prevent this type of behavior, part of the training dataset is typically set aside as the "test set" to check for overfitting. If the training data has a low error rate and the test data has a high error rate, it signals overfitting.

b) Detect Overfitting:

Detecting overfitting is almost impossible before you test the data. It can help address the inherent characteristic of overfitting, which is the inability to generalize data sets. The data can, therefore, be separated into different subsets to make it easy for training and testing. The data is split into two main parts. [19]

The training set represents most of the available data (about 80%), and it trains the model. The test set represents a small portion of the data set (about 20%), and it is used to test the accuracy of the data it never interacted with before. By segmenting the dataset, **we can examine the performance of the model** on each set of data to spot overfitting when it occurs, as well as see how the training process works. [19]

The performance can be measured using the **percentage of accuracy observed** in both data sets to conclude on the presence of overfitting. If the model performs better on the training set than on the test set, it means that the model is likely overfitting. [19]

c) Prevent Overfitting

Understanding how to detect overfitting, it is important to understand how to avoid overfitting altogether. Below are several techniques that you can use to prevent overfitting:

- ***Training with more data***

One of the ways to prevent overfitting is by training with more data. Such an option makes it easy for algorithms to detect the signal better to minimize errors. As the user feeds more training data into the model, it will be unable to overfit all the samples and will be forced to generalize to obtain results. [19]

Users should continually collect more data as a way of increasing the accuracy of the model. However, this method is considered expensive, and, therefore, users should ensure that the data being used is relevant and clean. [19]

- ***Data augmentation***

An alternative to training with more data is data augmentation, which is less expensive compared to the former. If you are unable to continually collect more data, you can make the available data sets appear diverse. [19]

Data augmentation makes a sample data look slightly different every time it is processed by the model. The process makes each data set appear unique to the model and prevents the model from learning the characteristics of the data sets. [19]

Another option that works in the same way as data augmentation is adding noise to the input and output data. Adding noise to the input makes the model become stable, without affecting data quality and privacy, while adding noise to the output makes the data more diverse. However, noise addition should be done with moderation so that the extent of the noise is not so much as to make the data incorrect or too different. [19]

- ***Data simplification***

Overfitting can occur due to the complexity of a model, such that, even with large volumes of data, the model still manages to overfit the training dataset. The data simplification method is used to reduce overfitting by decreasing the complexity of the model to make it simple enough that it does not overfit. [19]

Some of the actions that can be implemented include pruning a decision tree, reducing the number of parameters in a neural network, and using dropout on a neutral network. Simplifying the model can also make the model lighter and run faster. [19]

- ***Early stopping:***

This method seeks to pause training before the model starts learning the noise within the model. This approach risks halting the training process too soon, leading to the opposite problem of underfitting. [18]

- ***Regularization:***

If overfitting occurs when a model is too complex, it makes sense for us to reduce the number of features. If we don't know which features to remove from our model, regularization methods can be particularly helpful. [18]

Regularization applies a "penalty" to the input parameters with the larger coefficients, which subsequently limits the amount of variance in the model. While there are a number of regularization methods, such as L1 regularization, Lasso regularization, and dropout, they all seek to identify and reduce the noise within the data. [18]

- ***Ensembling***

Ensembling is a machine learning technique that works by combining predictions from two or more separate models. The most popular ensembling methods include **boosting** and **bagging.** [19]

Boosting works by using simple base models to increase their aggregate complexity. It trains many weak learners arranged in a sequence, such that each learner in the sequence learns from the mistakes of the learner before it. Boosting combines all the weak learners in the sequence to bring out one strong learner. [19]

The other assembling method is **bagging**, which is the opposite of boosting. Bagging works by training many strong learners arranged in a parallel pattern and then combining them to optimize their predictions. [19]

d) Underfitting

Underfitting is a scenario in data science where a data model is **unable to capture the relationship between the input and output** variables accurately, generating a high error rate on both the training set and unseen data. It occurs when a **model is too simple**, which can be a result of a model needing more training time, more input features, or less regularization. [20]

Like overfitting, when a model is underfitted, it cannot establish the dominant trend within the data, resulting in training errors and poor performance of the model. If a model cannot generalize well to new data, then it cannot be leveraged for classification or prediction tasks. Generalization of a model to new data is ultimately what allows us to use machine learning algorithms every day to make predictions and classify data. [20]

High bias and **low variance** are good **indicators of underfitting**. Since this behavior can be seen while using the training dataset, underfitted models are usually easier to identify than overfitted ones. [20]

e) Avoid underfitting

We can detect underfitting based of the training set, we can better assist at establishing the dominant relationship between the input and output variables at the onset. By maintaining adequate model complexity, we can avoid underfitting and make more accurate predictions. Below are a few techniques that can be used to reduce underfitting: [20]

- ***Decrease regularization***

Regularization is typically used to reduce the variance with a model by applying a penalty to the input parameters with the larger coefficients. There are a number of different methods, such as L1 regularization, Lasso regularization, dropout, etc., which help to reduce the noise and outliers within a model. However, if the data features become too uniform, the model is unable to identify the dominant trend, leading to underfitting. By decreasing the amount of regularization, more complexity and variation is introduced into the model, allowing for successful training of the model. [20]

- ***Increase the duration of training***

As mentioned earlier, stopping training too soon can also result in underfit model. Therefore, by extending the duration of training, it can be avoided. However, it is important to cognizant of overtraining, and subsequently, overfitting. Finding the balance between the two scenarios will be key. [20]

- ***Feature selection***

With any model, specific features are used to determine a given outcome. If there are not enough predictive features present, then more features or features with greater importance, should be introduced. For example, in a neural network, you might add more hidden neurons or in a random forest, you may add more trees. This process will inject more complexity into the model, yielding better training results. [20]

f) Optimum:

The ideal scenario when fitting a model is to find the balance between overfitting and underfitting. Identifying that "sweet spot" between the two allows machine learning models to make predictions with accuracy. [20]

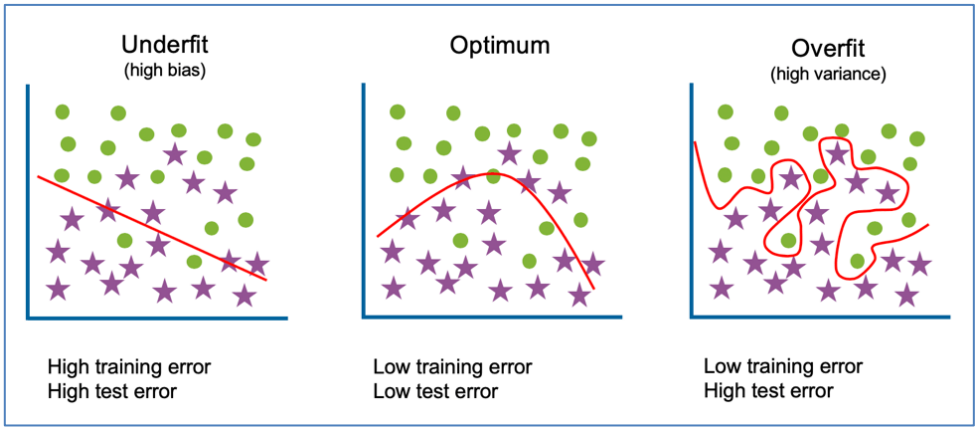

Figure 16 : Underfit, Overfit and optimum

VIII. Traditional algorithms for machine learning:

Algorithms are mathematics procedures of a resolution. It's about a systematic method which gives reliable results. Artificial intelligence algorithms can be broadly classified as:

1. Classification Algorithms

Classification algorithms are part of supervised learning. These algorithms are used to divide the subjected variable into different classes and then predict the class for a given input. For example, classification algorithms can be used to classify emails as spam or not. Let's discuss some of the commonly used classification algorithms. [21]

a) Naive Bayes

Naive Bayes algorithm works on Bayes theorem and takes a probabilistic approach, unlike other classification algorithms. The algorithm has a set of prior probabilities for each class. Once data is fed, the algorithm updates these probabilities to form something known as posterior probability. This comes useful when you need to predict whether the input belongs to a given list of classes or not. [21]

A Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

Even if these features are related to each other, a Naive Bayes classifier would consider all of these properties independently when calculating the probability of a particular outcome.

A Naive Bayesian model is easy to build and useful for massive datasets. It's simple and is known to outperform even highly sophisticated classification methods. [15]

b) Decision Tree

Decision Tree algorithm in machine learning is one of the most popular algorithms in use today; this is a supervised learning algorithm that is used for classifying problems. It works well classifying for both categorical and continuous dependent variables. In this algorithm, we split the population into two or more homogeneous sets based on the most significant attributes/ independent variables. [15]

The decision tree algorithm is more of a flowchart like an algorithm where nodes represent the test on an input attribute and branches represent the outcome of the test.

c) Random Forest

A collective of decision trees is called a Random Forest. To classify a new object based on its attributes, each tree is classified, and the tree "votes" for that class. The forest chooses the classification having the most votes (over all the trees in the forest). [15]

Random forest works like a group of trees. The input data set is subdivided and fed into different decision trees. The average of outputs from all decision trees is considered. Random forests offer a more accurate classifier as compared to Decision tree algorithm. [21]

Each tree is planted & grown as follows:

- If the number of cases in the training set is N, then a sample of N cases is taken at random. This sample will be the training set for growing the tree.
- If there are M input variables, a number m<<M is specified such that at each node, m variables are selected at random out of the M, and the best split on this m is used to split the node. The value of m is held constant during this process.
- Each tree is grown to the most substantial extent possible. There is no pruning.

d) Support Vector Machines

SVM is an algorithm that classifies data using a hyperplane, making sure that the distance between the hyperplane and support vectors is maximum. [21]

In SVM you plot raw data as points in an n-dimensional space (where n is the number of features you have). The value of each feature is then tied to a particular coordinate, making it easy to classify the data. Lines called classifiers can be used to split the data and plot them on a graph. [15]

e) K Nearest Neighbors

KNN algorithm uses a bunch of data points segregated into classes to predict the class of a new sample data point. It is called "lazy learning algorithm" as it is relatively short as compared to other algorithms. [21]

This algorithm can be applied to both classification and regression problems. Apparently, within the Data Science industry, it's more widely used to solve classification problems. It's a simple algorithm that stores all available cases and classifies any new cases by taking a majority vote of its k neighbors. The case is then assigned to the class with which it has the most in common. A distance function performs this measurement. KNN can be easily understood by comparing it to real life. [15]

Things to consider before selecting K Nearest Neighbors Algorithm:

- KNN is computationally expensive
- Variables should be normalized, or else higher range variables can bias the algorithm
- Data still needs to be pre-processed.

2. Regression Algorithms

Regression algorithms are a popular algorithm under supervised machine learning algorithms. Regression algorithms can predict the output values based on input data points fed in the learning system. The main application of regression algorithms includes predicting stock market price, predicting weather, etc. The most common algorithms under this section are: [21]

I. Linear regression

It is used to measure genuine qualities by considering the consistent variables. It is the simplest of all regression algorithms but can be implemented only in cases of linear relationship or a linearly separable problem. The algorithm draws a straight line between data points called the best-fit line or regression line and is used to predict new values. [21]

To understand the working functionality of this algorithm, imagine how you would arrange random logs of wood in increasing order of their weight. There is a catch; however – you cannot weigh each log. You must guess its weight just by looking at the height and girth of the log (visual analysis) and arrange them using a combination of these visible parameters. This is what linear regression in **machine learning** is like. [15]

In this process, a relationship is established between independent and dependent variables by fitting them to a line. This line is known as the regression line and represented by a linear equation

$$\mathbf{Y= a *X + b.}$$

In this equation:

- Y – Dependent Variable
- a – Slope
- X – Independent variable
- b – Intercept

The coefficients a & b are derived by minimizing the sum of the squared difference of distance between data points and the regression line.

II. Lasso Regression

Lasso regression algorithm works by obtaining the subset of predictors that minimizes prediction error for a response variable. This is achieved by imposing a constraint on data points and allowing some of them to shrink to zero value. [21]

III. Logistic Regression

Logistic regression is mainly used for binary classification. This method allows you to analyze a set of variables and predict a categorical outcome. Its primary applications include predicting customer lifetime value, house values, and more [21]

Logistic Regression is used to estimate discrete values (usually binary values like 0/1) from a set of independent variables. It helps predict the probability of an event by fitting data to a logit function. It is also called logit regression. [15]

These methods listed below are often used to help improve logistic regression models:

- include interaction terms
- eliminate features
- regularize techniques
- use a non-linear model

IV. Multivariate Regression

This algorithm must be used when there is more than one predictor variable. This algorithm is extensively used in retail sector product recommendation engines, where customers preferred products will depend on multiple factors like brand, quality, price, review and more. [21]

V. Multiple Regression Algorithm

Multiple Regression Algorithm uses a combination of linear regression and non-linear regression algorithms taking multiple explanatory variables as inputs. The main applications include social science research, insurance claim genuineness, behavioral analysis, and more. [21]

3. Clustering Algorithms

Clustering is the process of segregating and organizing the data points into groups based on similarities within members of the group. This is part of unsupervised learning. The main aim is to group similar items. For example, it can arrange all transactions of fraudulent nature together based on some properties in the transaction. Below are the most common clustering algorithms. [21]

a) K-Means Clustering

It is the simplest unsupervised learning algorithm. The algorithm gathers similar data points together and then binds them together into a cluster. The clustering is done by calculating the centroid of the group of data points and then evaluating the distance of each data point from the centroid of the cluster. Based on the distance, the analyzed data point is then assigned to the closest cluster. 'K' in K-means stands for the number of clusters the data points are being grouped into. [21]

It is an unsupervised learning algorithm that solves clustering problems. Data sets are classified into a particular number of clusters (let's call that number K) in such a way that all the data points within a cluster are homogenous and heterogeneous from the data in other clusters. [15]

How K-means forms clusters:

- The K-means algorithm picks k number of points, called centroids, for each cluster.
- Each data point forms a cluster with the closest centroids, i.e., K clusters.
- It now creates new centroids based on the existing cluster members.
- With these new centroids, the closest distance for each data point is determined. This process is repeated until the centroids do not change.

b) Fuzzy C-means Algorithm

FCM algorithm works on probability. Each data point is considered to have a probability of belonging to another cluster. Data points don't have an absolute membership over a particular cluster, and this is why the algorithm is called fuzzy. [21]

c) Expectation-Maximization (EM) Algorithm

It is based on Gaussian distribution we learned in statistics. Data is pictured into a Gaussian distribution model to solve the problem. After assigning a probability, a point sample is calculated based on expectation and maximization equations. [21]

d) Hierarchical Clustering Algorithm

These algorithms sort clusters hierarchical order after learning the data points and making similarity observations. It can be of two types

- Divisive clustering, for a top-down approach
- Agglomerative clustering, for a bottom-up approach

5. Machine learning in healthcare

Machine learning is finding a wide range of healthcare applications, ranging from case management of prevalent chronic diseases to leveraging patient health data, in conjunction with external influences like pollution exposure and weather factors.

By crunching large volumes of data, machine learning technology can help healthcare professionals generate precise medicine solutions customized to individual characteristics.

Other potential machine learning developments in healthcare include exploring ways to use the technology in telemedicine, the report notes. Some machine learning companies are studying the ability to organize and provide doctors with patient information during a telemedicine session, as well as capturing information during the virtual visit to assist with increased efficiency and workflow.

IX. Conclusion

In this chapter we talked about how AI can add value to your business by Providing a more comprehensive understanding of the abundance of data available, relying on predictions to automate excessively complex or mundane tasks.

We also mentioned Clinical decision support (CDS) and their impact on improvements in quality, safety, efficiency, and effectiveness of health care. And how Machine learning and CDS tools are most effective when they are trained on data that is accurate, clean, and complete. After all, an algorithm's output is only as good as its input, and in the high-stakes industry of healthcare, the input has to be pretty precise.

We have talked also about machine learning and its components, types, algorithms, and we mentioned neural networks as a main part of deep learning and the cause of its depth. In the next chapter we will detail what are neural networks, how do they work, their types and much more.

Chapter 2: Neural networks

I. Introduction

In the previous chapter we talked about how artificial intelligence was found and how it helped the human to complete his daily tasks so easily. We also introduced about the machine learning and deep learning, and we talked how the machine became a human like when it comes to take decisions.

In this chapter we will talk about the artificial neural networks. How the idea started and how we managed to make it real.

This chapter is divided into five parts, In the first part we will introduce human neural network and the main parts responsible of acting and learning. We will also see the similarity between the human and artificial neural network

In the second part we will see artificial neural network with it components, parameters, how does it function and the types of activations function that we have.

In the third part, we will talk about the types of neural networks that we can find (Perceptron, RNN, CNN … etc.) and which one for which with details of how each one works

In the final part we will detail more about Convolutional neural network architectures as AlexNet, VGG, Xception…. and how they helped in improving the conception of a Convolutional neural network

We are more interested in our topic in Convolutional neural networks, in the next chapter

II. Human nervous system

The nervous system is the major controlling, regulatory, and communicating system in the body. It is the center of all mental activity including thought, learning, and memory. Together with the endocrine system, the nervous system is responsible for regulating and maintaining homeostasis. Through its receptors, the nervous system keeps us in touch with our environment, both external and internal. [22]

Like other systems in the body, the nervous system is composed of **organs**, principally the brain, spinal cord, nerves, and ganglia. These, in turn, consist of various tissues, including nerve, blood, and connective tissue. Together these carry out the complex activities of the nervous system. [22]

The various **activities of the nervous system** can be grouped together as three generals, overlapping functions: **Sensory, Integrative,** and **motor.**

1. How nervous system works

Millions of **sensory receptors** detect changes, called **stimuli**, which occur inside and outside the body. They monitor such things as temperature, light, and sound from the **external environment**. Inside the body, the **internal environment**, receptors detect variations in pressure, pH, carbon dioxide concentration, and the levels of various electrolytes. All this gathered information is called sensory input. [22]

Sensory input is **converted into electrical signals** called nerve impulses that are transmitted to the brain. There the signals are brought together to create sensations, to produce thoughts, or to add to memory; Decisions are made each moment based on the sensory input. This is integration.

Based on the sensory input and integration, the nervous system responds by sending signals to muscles, causing them to contract, or to glands, causing them to produce secretions. Muscles and glands are called effectors because they cause an effect in response to directions from the nervous system. This is the motor output or motor function. [22]

2. Nerve Tissue

Although the nervous system is very complex, there are only two main types of cells in nerve tissue. The actual nerve cell is the neuron. It is the "conducting" cell that transmits impulses and the structural unit of the nervous system. The other type of cell is neuroglia, or glial, cell. The word "neuroglia" means "nerve glue." These cells are nonconductive and provide a support system for the neurons. They are a special type of "connective tissue" for the nervous system. [22]

a) Neurons

Neurons, or nerve cells, carry out the functions of the nervous system by conducting nerve impulses. They are highly specialized and amitotic. This means that if a neuron is destroyed, it cannot be replaced because neurons do not go through mitosis. The image below illustrates the structure of a typical neuron. [22]

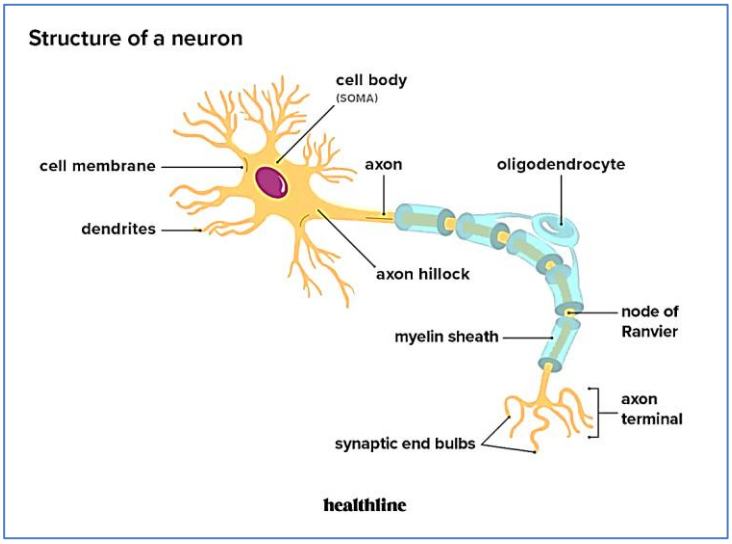

Figure 17 : Structure of a neuron

Each neuron has three basic parts: cell body (soma), one or more dendrites, and a single axon.

3. Idea of Artificial neural network

The first neural network was conceived of by Warren McCulloch and Walter Pitts in 1943. They wrote a seminal paper on how neurons may work and modeled their ideas by creating a simple neural network using electrical circuits. This breakthrough model paved the way for neural network research in two areas: [23]

- Biological processes in the brain.
- The application of neural networks to artificial intelligence (AI).

The main goal of the neural network approach was to create a **computational system** that could solve problems like a human brain. However, over time, researchers shifted their focus to using neural networks to match specific tasks, leading to deviations from a strictly biological approach. Since then, neural networks have supported diverse tasks, including computer vision, speech recognition, machine translation, social network filtering, playing board and video games, and medical diagnosis. [23]

As structured and unstructured data sizes increased to big data levels, people developed deep learning systems, which are essentially neural networks with many layers. Deep learning enables the capture and mining of more and bigger data, including unstructured data. [23]

III. Artificial neural network

1. Definition

Neural networks, also known as artificial neural networks (ANNs) or simulated neural networks (SNNs), are a subset of machine learning and are at the heart of deep learning algorithms. Their name and structure are inspired by the human brain, imitating the way that biological neurons signal to one another. [24]

Artificial neural networks (ANNs) are comprised of **node layers**, containing an **input layer**, one or more **hidden layers**, and an **output layer**. Each node, or artificial neuron, connects to another and has an associated weight and threshold. If the output of any individual node is above the specified threshold value, that **node is activated**, sending data to the next layer of the network. Otherwise, no data is passed along to the next layer of the network. [24]

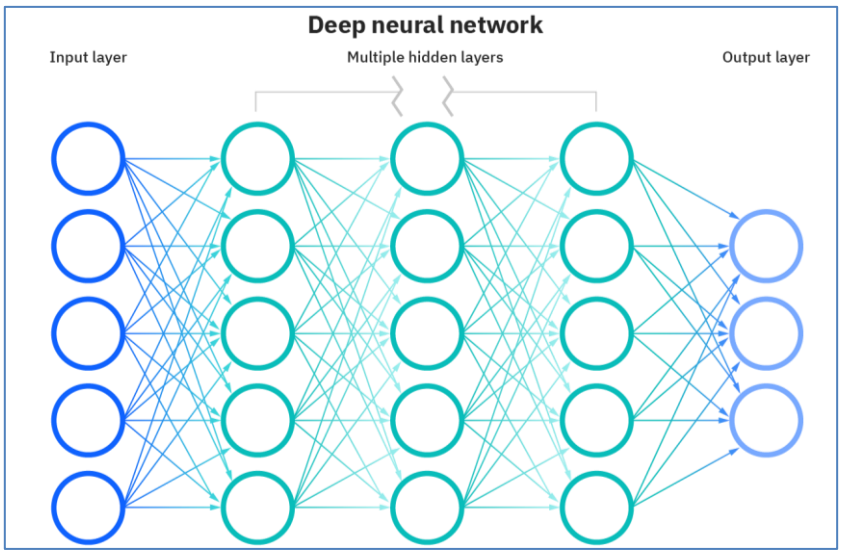

Figure 18 : Artificial neural network

Neural networks reflect the behavior of the human brain, allowing computer programs to recognize patterns and solve common problems in the fields of AI, machine learning, and deep learning.

Neural networks rely on training data to learn and improve their accuracy over time. However, once these learning algorithms are fine-tuned for accuracy, they are powerful tools in computer science and artificial intelligence, allowing us to classify and cluster data at a high velocity. Tasks in speech recognition or image recognition can take minutes versus hours when compared to the manual identification by human experts. One of the most well-known neural networks is Google's search algorithm. [24]

2. Artificial neural networks functioning

Artificial Neural Networks work in a way like that of their biological inspiration. They can be considered as **weighted directed graphs** where the neurons could be compared to the nodes and the connection between two neurons as weighted edges. The processing element of a neuron receives many signals (both from other neurons and as input signals from the external world). [25]

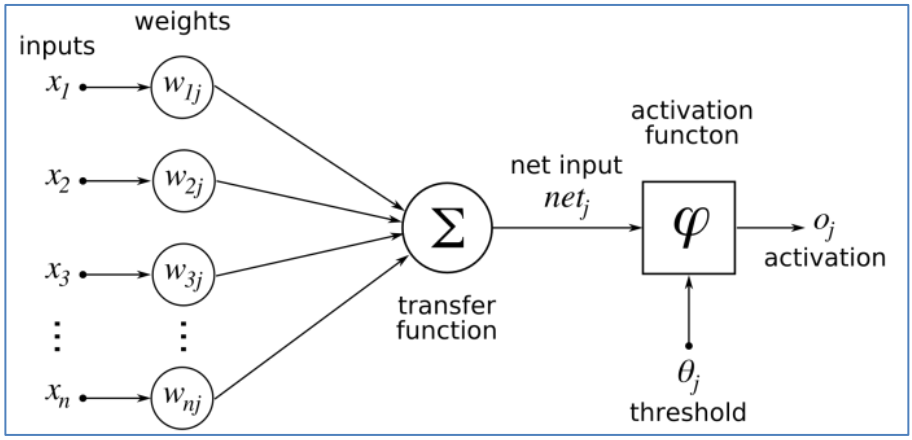

Figure 19 : Functioning of ANN

- **Input:**

It is the set of features that are fed into the model for the learning process. For example, the input in object detection can be an array of pixel values pertaining to an image. [26]

- **Weight:**

Its main function is to give importance to those features that contribute more towards the learning. It does so by introducing scalar multiplication between the input value and the weight matrix. For example, a negative word would impact the decision of the sentiment analysis model more than a pair of neutral words. [26]

- **Transfer function:**

The job of the transfer function is to combine multiple inputs into one output value so that the activation function can be applied. It is done by a simple summation of all the inputs to the transfer function. [26]

- **Activation Function:**

It introduces non-linearity in the working of perceptrons to consider varying linearity with the inputs. Without this, the output would just be a linear combination of input values and would not be able to introduce non-linearity in the network. [26]

- **Bias:**

The role of bias is to shift the value produced by the activation function. Its role is similar to the role of a constant in a linear function. [26]

When multiple neurons are stacked together in a row, they constitute a layer, and multiple layers piled next to each other are called a multi-layer neural network.

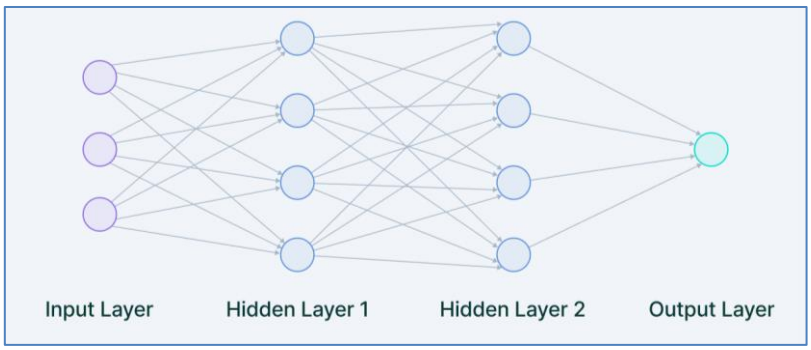

Figure 20 : Multi-layer neural network

a) Input Layer

The input layer takes raw input from the domain. No computation is performed at this layer. Nodes here just pass on the information (features) to the hidden layer. [26]

b) Hidden Layer

As the name suggests, the nodes of this layer are not exposed. They provide an abstraction to the neural network. The hidden layer performs all kinds of computation on the features entered through the input layer and transfers the result to the output layer. [26]

There can be multiple interconnected hidden layers that account for searching different hidden features in the data. On the other hand, the later hidden layers perform more complicated tasks like identifying complete objects (a car, a building, a person).

c) Output Layer

It's the final layer of the network that brings the information learned through the hidden layer and delivers the final value as a result. In the case of classification/regression models, the output layer generally has a single node.

All hidden layers usually use the same activation function. However, the output layer will typically use a different activation function from the hidden layers. The choice depends on the goal or type of prediction made by the model. [26]

There are two essential terms describing the movement of information, feedforward and backpropagation.

- **Feedforward:**

The flow of information occurs in the forward direction. The input is used to calculate some intermediate function in the hidden layer, which is then used to calculate an output. In the feedforward propagation, the Activation Function is a mathematical "gate" in between the input feeding the current neuron and its output going to the next layer. [26]

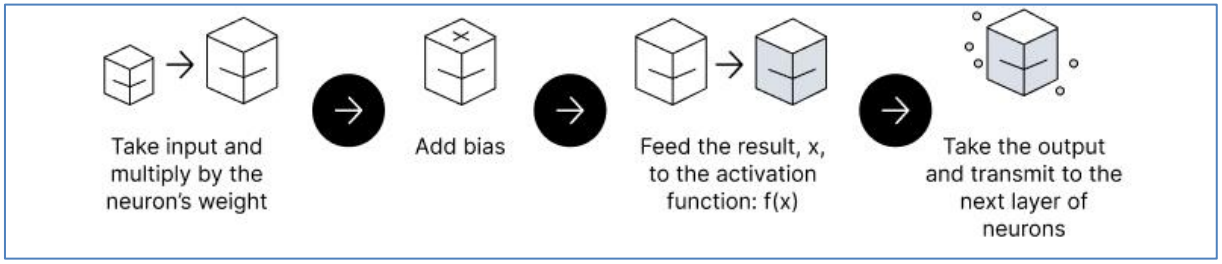

Figure 21: feedforward movement

- **Backpropagation:**

The weights of the network connections are repeatedly adjusted to minimize the difference between the actual output vector of the net and the desired output vector. Backpropagation aims to minimize the cost function by adjusting the network's weights and biases. The cost function gradients determine the level of adjustment with respect to parameters like activation function, weights, bias, etc. [26]

- **Dropout**

Random neurons are cancelled. Dropout is regularization technique to avoid overfitting (increase the validation accuracy) thus increasing the generalizing power. Generally, use a small dropout value of 20%-50% of neurons with 20% providing a good starting point. A probability too low has minimal effect and a value too high results in under-learning by the network. [27]

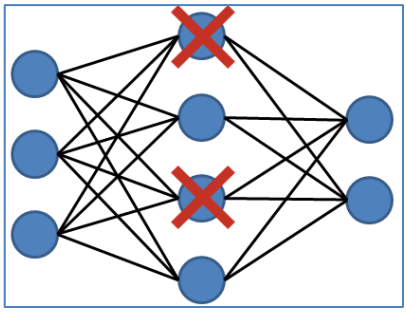

Figure 22 : Dropout

- **Learning Rate**

The learning rate defines how quickly a network updates its parameters. **Low learning rate** slows down the learning process but converges smoothly. **Larger learning rate** speeds up the learning but may not converge. Usually, a **decaying Learning rate** is preferred. [27]

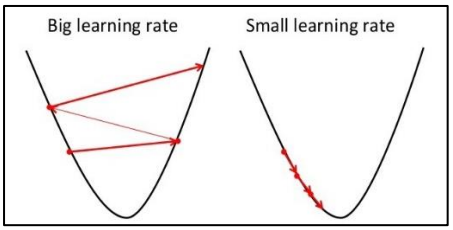

Figure 23 : Learning Rate

- **Momentum**

Momentum helps to know the direction of the next step with the knowledge of the previous steps. It helps to prevent oscillations. A typical choice of momentum is between 0.5 to 0.9. [27]

- **Number of epochs**

Number of epochs is the number of times the whole training data is shown to the network while training. Increase the number of epochs until the validation accuracy starts decreasing even when training accuracy is increasing(overfitting). [27]

- **Batch size**

Mini batch size is the number of sub samples given to the network after which parameter update happens. [27]

3. Activation functions

An activation Function decides whether a neuron should be activated or not. This means that it will decide whether the neuron's input to the network is important or not in the process of prediction using simpler mathematical operations. [28]

The role of the Activation Function is to derive output from a set of input values fed to a node (or a layer).

The primary role of the Activation Function is to transform the summed weighted input from the node into an output value to be fed to the next hidden layer or as output. [28]

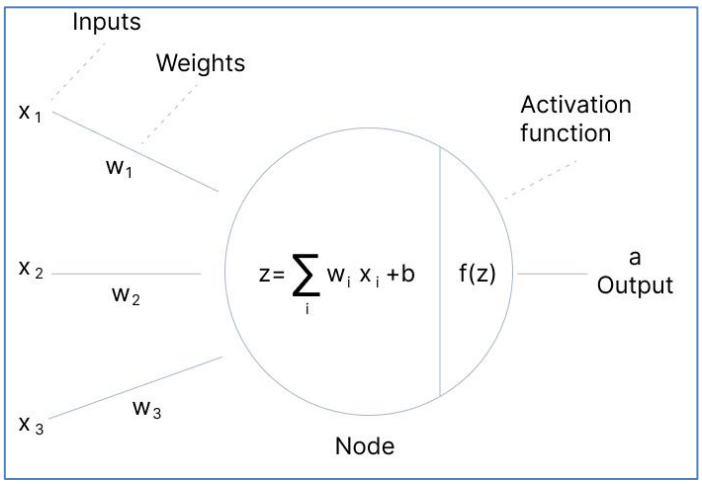

Figure 24 : Activation function

There are a lot of activation functions divided in linear and non-linear functions:

A. Binary Step Function

Binary step function depends on a threshold value that decides whether a neuron should be activated or not. The input fed to the activation function is compared to a certain threshold; if the input is greater than it, then the neuron is activated, else it is deactivated, meaning that its output is not passed on to the next hidden layer. [28]

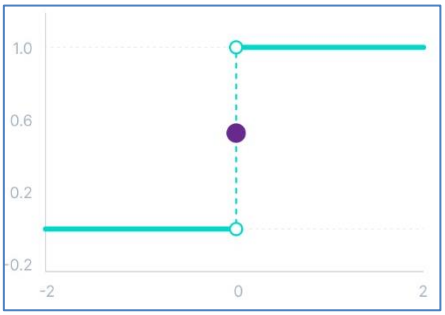

Figure 25: Binary step fucntion

Mathematically it can be represented as:

$$f(x) = \begin{cases} 0 & for\ x < 0 \\ 1 & for\ x \geqslant 0 \end{cases}$$

the limitations of binary step function:

- It cannot provide multi-value outputs
- The gradient of the step function is zero, which causes a hindrance in the backpropagation process.

B. Linear Activation Function

The linear activation function is also known as Identity Function where the activation is proportional to the input. [28]

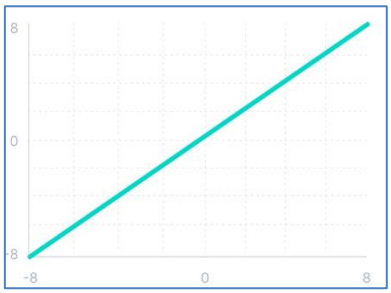

Figure 26 : Linear Activation function

Mathematically it can be represented as:

$$f(x) = x$$

linear activation function has two major problems:

- It's not possible to use backpropagation as the derivative of the function is a constant and has no relation to the input x.
- All layers of the neural network will collapse into one if a linear activation function is used. No matter the number of layers in the neural network, the last layer will still be a linear function of the first layer. So, essentially, a linear activation function turns the neural network into just one layer.

C. Non-Linear Activation Functions

The linear activation function shown above is simply a linear regression model. Because of its limited power, this does not allow the model to create complex mappings between the network's inputs and outputs. [28]

Non-linear activation functions solve the following limitations of linear activation functions:

- They allow backpropagation because now the derivative function would be related to the input, and it's possible to go back and understand which weights in the input neurons can provide a better prediction.
- They allow the stacking of multiple layers of neurons as the output would now be a non-linear combination of input passed through multiple layers. Any output can be represented as a functional computation in a neural network.

- ***Sigmoid / Logistic Activation Function***

This function takes any real value as input and outputs values in the range of 0 to 1. The larger the input (more positive), the closer the output value will be to 1.0, whereas the smaller the input (more negative), the closer the output will be to 0.0, as shown below. [28]

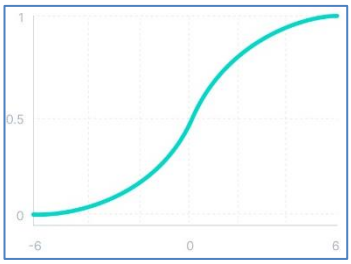

Figure 27: Sigmoid / Logistic function

Mathematically it can be represented as:

$$f(x) = \frac{1}{1 + e^{-x}}$$

Advantages of Sigmoid / Logistic function:

- It is commonly used for models where we must predict the probability as an output. Since probability of anything exists only between the range of 0 and 1, sigmoid is the right choice because of its range.
- The function is differentiable and provides a smooth gradient, i.e., preventing jumps in output values. This is represented by an S-shape of the sigmoid activation function.

The limitations of sigmoid function are discussed below:

- The output of the logistic function is not symmetric around zero. So the output of all the neurons will be of the same sign. This makes the training of the neural network more difficult and unstable.
- The derivative of the function is f'(x) = sigmoid(x)*(1-sigmoid(x)).

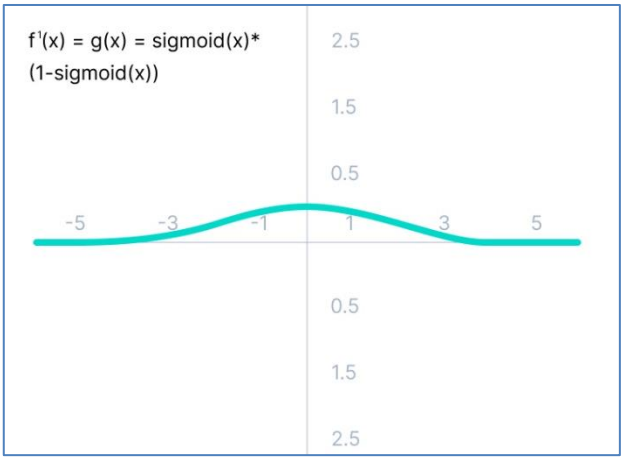

Figure 28: derivative of the Sigmoid Activation Function

The gradient values are only significant for range -3 to 3, and the graph gets much flatter in other regions. It implies that for values greater than 3 or less than -3, the function will have very small gradients. As the gradient value approaches zero, the network ceases to learn and suffers from the Vanishing gradient problem. [28]

- ***Tanh Function (Hyperbolic Tangent)***

Tanh function is very similar to the sigmoid/logistic activation function, and even has the same S-shape with the difference in output range of -1 to 1. In Tanh, the larger the input (more positive), the closer the output value will be to 1.0, whereas the smaller the input (more negative), the closer the output will be to -1.0. [28]

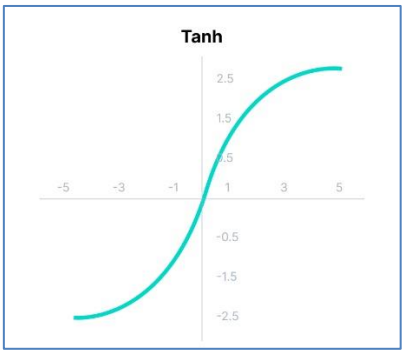

Figure 29 : Tanh Function

Mathematically it can be represented as:

$$f(x) = \frac{\left(e^{x} - e^{-x}\right)}{\left(e^{x} + e^{-x}\right)}$$

Advantages of using this activation function are:

- The output of the tanh activation function is Zero centered; hence we can easily map the output values as strongly negative, neutral, or strongly positive.
- Usually used in hidden layers of a neural network as its values lie between -1 to; therefore, the mean for the hidden layer comes out to be 0 or very close to it. It helps in centering the data and makes learning for the next layer much easier.

The limitation if tanh function:

- The gradient of the tanh activation function. it also faces the problem of vanishing gradients like the sigmoid activation function. Also, the gradient of the tanh function is much steeper as compared to the sigmoid function.

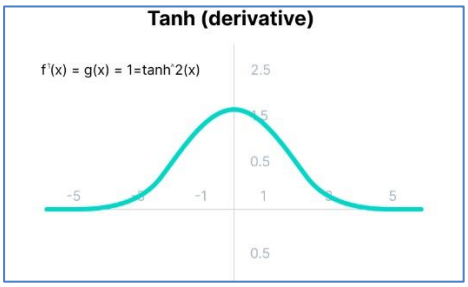

Figure 30 : Tanh (derivative)

- ***ReLU Function***

ReLU stands for Rectified Linear Unit. Although it gives an impression of a linear function, ReLU has a derivative function and allows for backpropagation while simultaneously making it computationally efficient. The main catch here is that the ReLU function does not activate all the neurons at the same time. The neurons will only be deactivated if the output of the linear transformation is less than 0. [28]

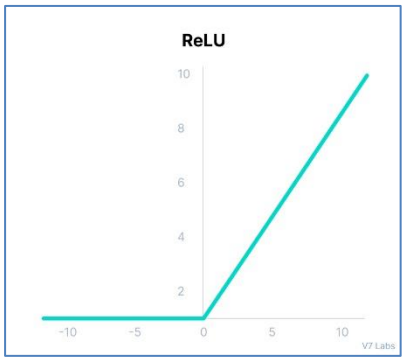

Figure 31: ReLU function

Mathematically it can be represented as:

$$f(x) = max\ (0, x)$$

The advantages of using ReLU as an activation function are as follows:

- Since only a certain number of neurons are activated, the ReLU function is far more computationally efficient when compared to the sigmoid and tanh functions.
- ReLU accelerates the convergence of gradient descent towards the global minimum of the loss function due to its linear, non-saturating property.

The limitations faced by this function are:

- The Dying ReLU problem: The negative side of the graph makes the gradient value zero. Due to this reason, during the backpropagation process, the weights and biases for some neurons are not updated. This can create dead neurons which never get activated. All the negative input values become zero immediately, which decreases the model's ability to fit or train from the data properly.

Figure 32 : Dying ReLU

- ***Leaky ReLU Function***

Leaky ReLU is an improved version of ReLU function to solve the Dying ReLU problem as it has a small positive slope in the negative area. [28]

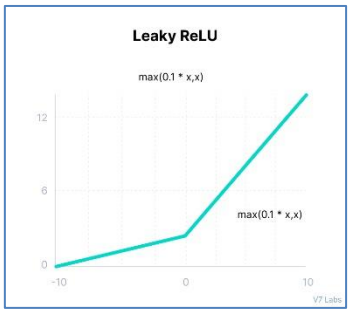

Figure 33 : Leaky ReLU Function

Mathematically it can be represented as:

$$f(x) = max\ (0.1x, x)$$

The advantages of Leaky ReLU are same as that of ReLU, in addition to the fact that it does enable backpropagation, even for negative input values. By making this minor modification for negative input values, the gradient of the left side of the graph comes out to be a non-zero value. Therefore, we would no longer encounter dead neurons in that region. [28]

The limitations that this function faces include:

- The predictions may not be consistent for negative input values.
- The gradient for negative values is a small value that makes the learning of model parameters time-consuming.

- ***Parametric ReLU Function***

Parametric ReLU is another variant of ReLU that aims to solve the problem of gradient's becoming zero for the left half of the axis. This function provides the slope of the negative part of the function as an argument a. By performing backpropagation, the most appropriate value of a is learnt.

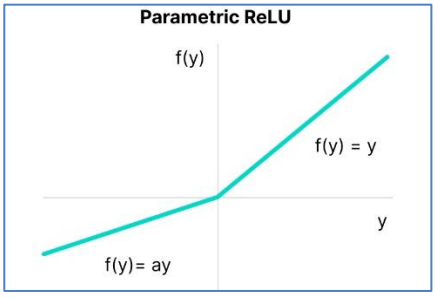

Figure 34 : Parametric ReLU Function

Mathematically it can be represented as:

$$f(x) = max\ (ax, x)$$

Where "a" is the slope parameter for negative values. The parameterized ReLU function is used when the leaky ReLU function still fails at solving the problem of dead neurons, and the relevant information is not successfully passed to the next layer. [28]

This function's limitation is that it may perform differently for different problems depending upon the value of slope parameter a.

- ***Exponential Linear Units (ELUs) Function***

Exponential Linear Unit, or ELU for short, is also a variant of ReLU that modifies the slope of the negative part of the function. ELU uses a log curve to define the negativ values unlike the leaky ReLU and Parametric ReLU functions with a straight line. [28]

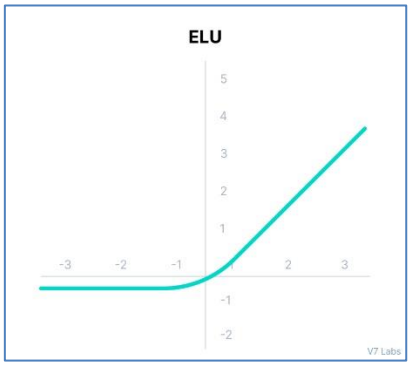

Figure 35 : ELU Function

Mathematically it can be represented as:

$$\begin{cases} x & for\ x \geqslant 0 \\ \alpha(e^x - 1) & for\ x < 0 \end{cases}$$

ELU is a strong alternative for f ReLU because of the following advantages:

- ELU becomes smooth slowly until its output equal to -α whereas RELU sharply smoothes.
- Avoids dead ReLU problem by introducing log curve for negative values of input. It helps the network nudge weights and biases in the right direction.

The limitations of the ELU function are as follow:

- It increases the computational time because of the exponential operation included
- No learning of the 'a' value takes place

- ***Softmax Function***

Before exploring the ins and outs of the Softmax activation function, we should focus on its building block—the sigmoid/logistic activation function that works on calculating probability values. [28]

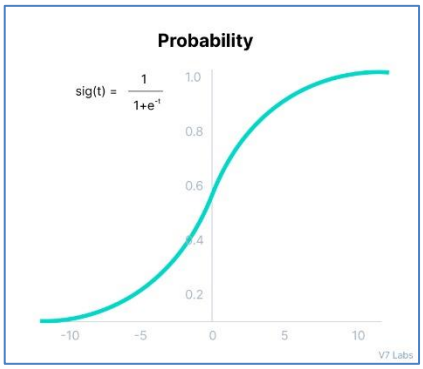

Figure 36 :Softmax fucntion

The output of the sigmoid function was in the range of 0 to 1, which can be thought of as probability.

The Softmax function is described as a combination of multiple sigmoids. It calculates the relative probabilities. Like the sigmoid/logistic activation function, the SoftMax function returns the probability of each class. It is most used as an activation function for the last layer of the neural network in the case of multi-class classification. [28]

Mathematically it can be represented as:

$$\text{softmax}(z_i) = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

- ***Swish***

It is a self-gated activation function developed by researchers at Google. Swish consistently matches or outperforms ReLU activation function on deep networks applied to various challenging domains such as image classification, machine translation etc. [28]

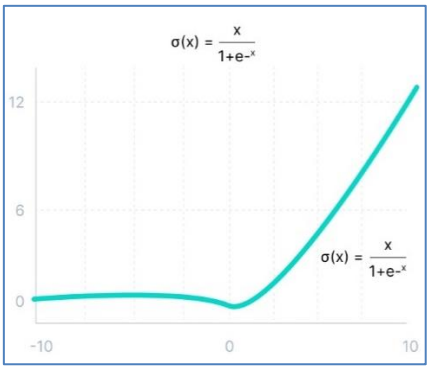

Figure 37 :Swish function

This function is bounded below but unbounded above i.e., Y approaches to a constant value as X approaches negative infinity but Y approaches to infinity as X approaches infinity. [28]

Mathematically it can be represented as:

$$f(x) = x*sigmoid(x)$$

Here are a few advantages of the Swish activation function over ReLU:

- Swish is a smooth function that means that it does not abruptly change direction like ReLU does near x = 0. Rather, it smoothly bends from 0 towards values < 0 and then upwards again.
- Small negative values were zeroed out in ReLU activation function. However, those negative values may still be relevant for capturing patterns underlying the data. Large negative values are zeroed out for reasons of sparsity making it a win-win situation.
- The swish function being non-monotonous enhances the expression of input data and weight to be learnt.

- ***Gaussian Error Linear Unit (GELU)***

The Gaussian Error Linear Unit (GELU) activation function is compatible with BERT, ROBERTa, ALBERT, and other top NLP models. This activation function is motivated by combining properties from dropout, zone out, and ReLUs. ReLU and dropout together yield a neuron's output. ReLU does it deterministically by multiplying the input by zero or one (depending upon the input value being positive or negative) and dropout stochastically multiplying by zero. RNN regularizer called zone out stochastically multiplies inputs by one. [28]

We merge this functionality by multiplying the input by either zero or one which is stochastically determined and is dependent upon the input. We multiply the neuron input x by m ~ Bernoulli(Φ(x)), where Φ(x) = P(X ≤x), X ~ N (0, 1) is the cumulative distribution function of the standard normal distribution. This distribution is chosen since neuron inputs tend to follow a normal distribution, especially with Batch Normalization. [28]

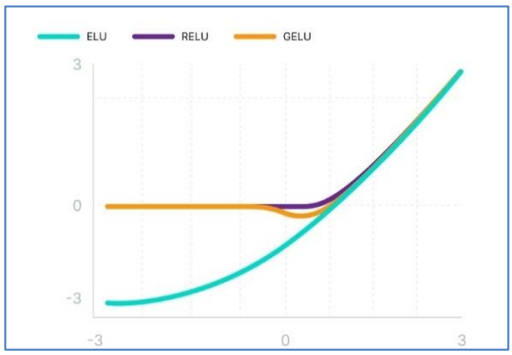

Figure 38 : Gaussian Error Linear Unit Function

Mathematically it can be represented as:

$$f(x) = xP(X \leq x) = x\Phi(x)$$
$$= 0.5x\left(1 + tanh\left[\sqrt{2/\pi}\left(x + 0.044715x^3\right)\right]\right)$$

GELU nonlinearity is better than ReLU and ELU activations and finds performance improvements across all tasks in domains of computer vision, natural language processing, and speech recognition. [28]

- ***Scaled Exponential Linear Unit (SELU)***

SELU was defined in self-normalizing networks and takes care of internal normalization which means each layer preserves the mean and variance from the previous layers. SELU enables this normalization by adjusting the mean and variance.

SELU has both positive and negative values to shift the mean, which was impossible for ReLU activation function as it cannot output negative values. Gradients can be used to adjust the variance. The activation function needs a region with a gradient larger than one to increase it. [28]

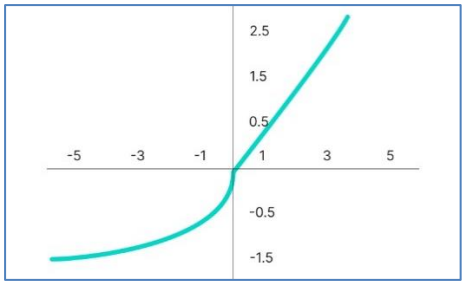

Figure 39 : Scaled Exponential Linear Unit Function

Mathematically it can be represented as:

$$f(\alpha, x) = \lambda\begin{cases}\alpha\left(e^x - 1\right) & for\ x < 0\\ x & for\ x \geqslant 0\end{cases}$$

SELU has values of alpha α and lambda λ predefined.

Here's the main advantage of SELU over ReLU:

- Internal normalization is faster than external normalization, which means the network converges faster.
- SELU is a relatively newer activation function and needs more papers on architectures such as CNNs and RNNs, where it is comparatively explored.

IV. Types of neural networks

The architecture of a neural network is of many types which include Perceptron, Feed Forward Neural Network, Multilayer Perceptron, Convolutional Neural Network, Radial Basis Function Neural Network, Recurrent Neural Network, LSTM –Long Short-Term Memory, Sequence to Sequence models, Modular Neural Network. Learning algorithms could be supervised, unsupervised or reinforcement methods.

1. The Perceptron

Perceptron is the simplest Neural Network architecture. It is a type of Neural Network that takes several inputs, applies certain mathematical operations on these inputs, and produces an output. It takes a vector of real values inputs, performs a linear combination of each attribute with the corresponding weight assigned to each of them. The weighted input is summed into a single value and passed through an activation function. [26]

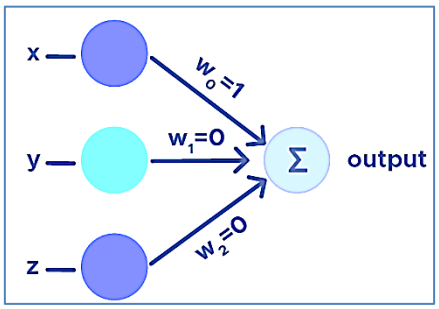

Figure 40 : Perceptron

These perceptron units are combined to form a bigger Artificial Neural Network architecture.

2. Feed-Forward Networks

It is a multi-layer Neural Network, and, as the name suggests, the information is passed in the forward direction from left to right. In the forward pass, the information comes inside the model through the input layer, passes through the series of hidden layers, and finally goes to the output layer. This Neural Networks architecture is forward in nature the information does not loop with two hidden layers. [26]

The later layers give no feedback to the previous layers. The basic learning process of Feed-Forward Networks remain the same as the perceptron.

3. Recurrent Neural Networks (RNNs)

The basic deep learning architecture has a fixed input size, and this acts as a blocker in scenarios where the input size is not fixed. Also, the decisions made by the model were based on the current input with no memory of the past. Recurrent Neural Networks work very well with sequences of data as input. Its functionality can be seen in solving NLP problems like sentiment analysis, spam filters, time series problems like sales forecasting, stock market prediction, etc. [26]

a) The Recurrent Neural Networks (RNN)

Recurrent Neural Networks have the power to remember what it has learned in the past and apply it in future predictions. [26]

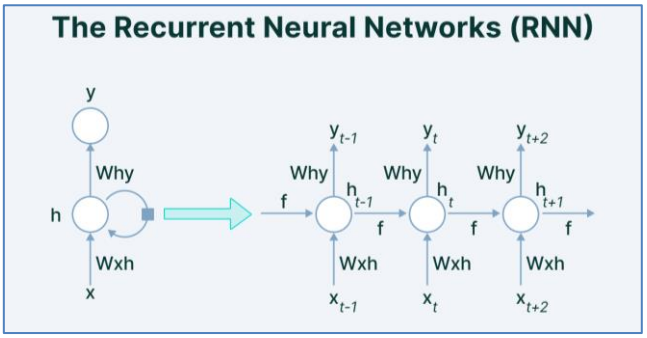

Figure 41 : Recurrent Neural Network

The input is in the form of sequential data that is fed into the RNN, which has a hidden internal state that gets updated every time it reads the following sequence of data in the input. The internal hidden state will be fed back to the model. The RNN produces some output at every timestamp. We use the same function and parameters at every timestamp. The mathematical representation is given below:

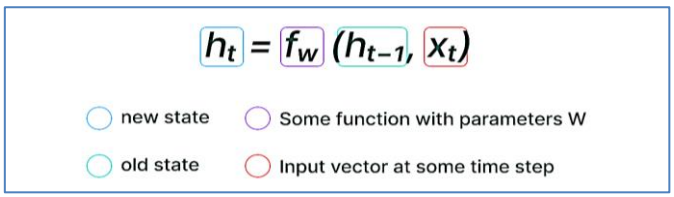

Figure 42 : RNN Formula

b) The Long Short Term Memory Network (LSTM)

In RNN each of our predictions looked only one timestamp back, and it has a very short-term memory. It doesn't use any information from further back. To rectify this, we can take our Recurrent Neural Networks structure and expand it by adding some more pieces to it. The critical part that we add to this Recurrent Neural Networks is **memory**. We want it to be able to remember what happened many timestamps ago. To achieve this, we need to add extra structures called gates to the artificial neural network structure. [26]

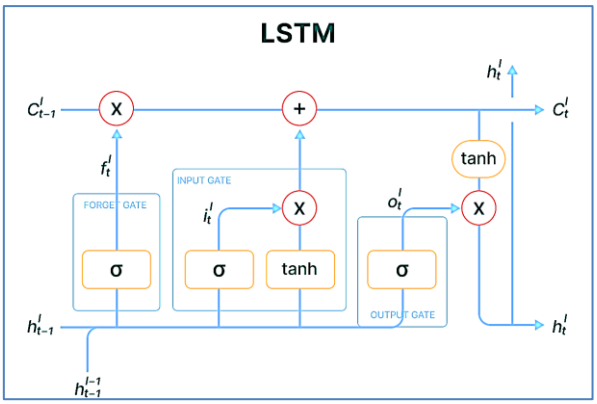

Figure 43: Long Short-Term Memory Network

- **Cell state (c_t):**

It corresponds to the long-term memory content of the network.

- **Forget Gate:**

Some information in the cell state is no longer needed and is erased. The gate receives two inputs, x_t (current timestamp input) and h_t-1 (previous cell state), multiplied with the relevant weight matrices before bias is added. The result is sent into an activation function, which outputs a binary value that decides whether the information is retained or forgotten.

- **Input gate:**

It decides what piece of new information is to be added to the cell state. It is similar to the forget gate using the current timestamp input and previous cell state with the only difference of multiplying with a different set of weights.

- **Output gate:**

The output gate's job is to extract meaningful information from the current cell state and provide it as an output.

4. Generative Adversarial Network (GAN)

Generative modeling comes under the umbrella of unsupervised learning, where new/synthetic data is generated based on the patterns discovered from the input set of data. GAN is a generative model and is used to generate entirely new synthetic data by learning the pattern and hence is an active area of AI research. It is used in scenarios like predicting the next frame in a video, text to image generation, image to image translation like style transfer, denoising of the image, etc. [26]

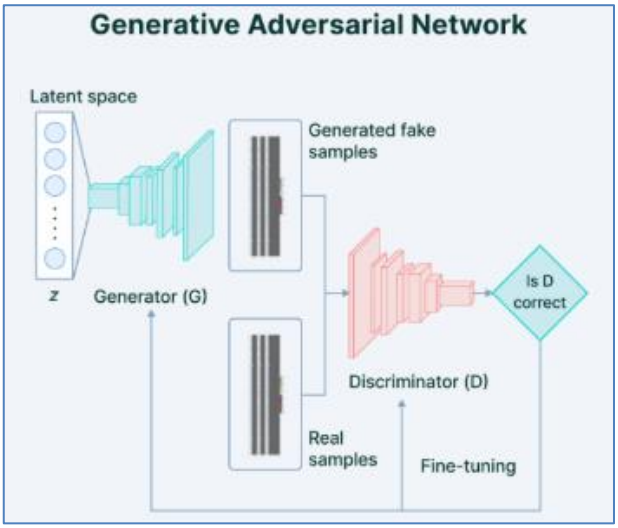

Figure 44 : Generative Adversarial Architecture

5. Convolutional Neural Networks (CNNs)

Convolutional Neural Networks is a type of Feed-Forward Neural Networks used in tasks like image analysis, natural language processing, and other complex image classification problems. A CNN has hidden layers of convolutional layers that form the base of ConvNets. Features refer to minute details in the image data like edges, borders, shapes, textures, objects, circles, etc. [26]

At a higher level, convolutional layers detect these patterns in the image data with the help of filters. The higher-level details are taken care of by the first few convolutional layers. The deeper the network goes, the more sophisticated the pattern searching becomes.

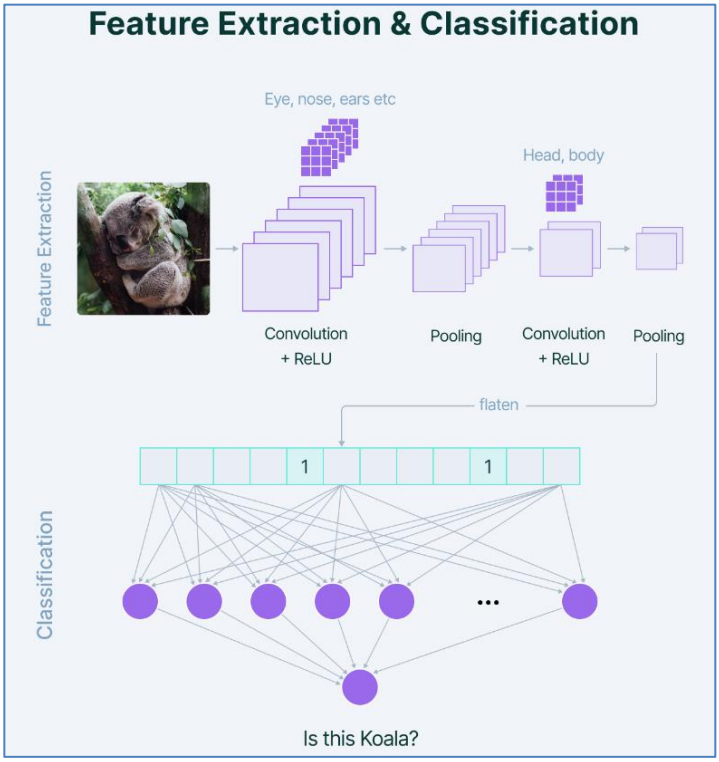

Figure 45 : Convolutional Neural Network

Convolutional neural networks are distinguished from other neural networks by their superior performance with image, speech, or audio signal inputs. They have three main types of layers, which are:

- Convolutional layer
- Pooling layer
- Fully connected (FC) layer

The convolutional layer is the first layer of a convolutional network. While convolutional layers can be followed by additional **convolutional layers** or **pooling layers**, the **fully-connected layer** is the **final layer**. With each layer, the CNN increases in its complexity, identifying greater portions of the image. Earlier layers focus on simple features, such as colors and edges.

As the image data progresses through the layers of the CNN, it starts to recognize larger elements or shapes of the object until it finally identifies the intended object.

- **Convolutional Layer**

The convolutional layer is the core building block of a CNN, and it is where most of the computation occurs. It requires a few components, which are input data, a filter, and a feature map. Let's assume that the input will be a color image, which is made up of a matrix of pixels in 3D. This means that the input will have three dimensions a height, width, and depth which correspond to RGB in an image. We also have a feature detector, also known as a kernel or a filter, which will move across the receptive fields of the image, checking if the feature is present. This process is known as a convolution. [29]

The feature detector is a two-dimensional (2-D) array of weights, which represents part of the image. While they can vary in size, the filter size is typically a 3x3 matrix; this also determines the size of the receptive field. The filter is then applied to an area of the image, and a dot product is calculated between the input pixels and the filter. This dot product is then fed into an output array. Afterwards, the filter shifts by a stride, repeating the process until the kernel has swept across the entire image. The final output from the series of dot products from the input and the filter is known as a feature map, activation map, or a convolved feature. [29]

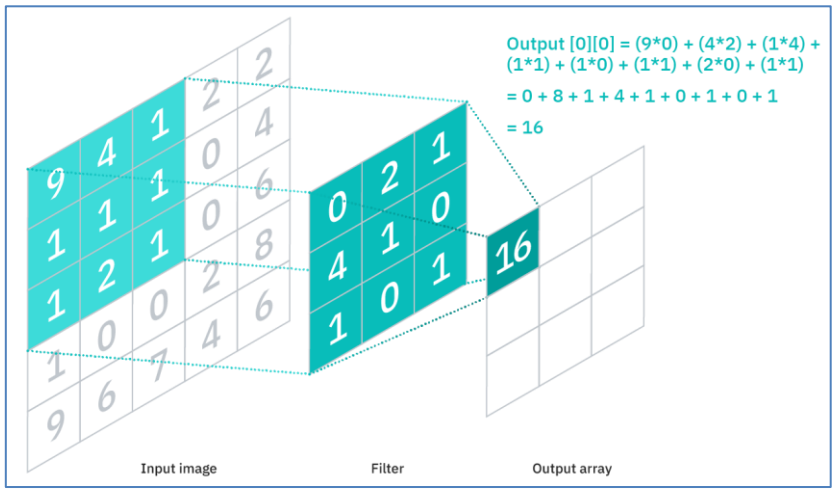

Figure 46 : Convolutional Layer

Since the output array does not need to map directly to each input value, convolutional (and pooling) layers are commonly referred to as "partially connected" layers. However, this characteristic can also be described as local connectivity.

The weights in the feature detector remain fixed as it moves across the image, which is also known as parameter sharing. Some parameters, like the weight values, adjust during training through the process of backpropagation and gradient descent. However, there are three hyperparameters which affect the volume size of the output that need to be set before the training of the neural network begins. [29]

The parameters of a convolutional layer can be:

Kernel Size: The kernel size defines the field of view of the convolution. A filter of size F×F applied to an input containing C channels is a F×F×C volume that performs convolutions on an input of size I×I×C and produces an output feature map (also called activation map) of size O×O×1. [30]

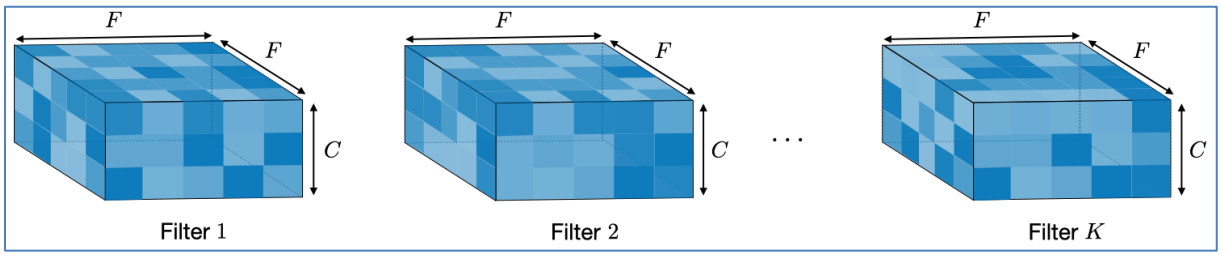

Figure 47 : Kernel Size

Stride: The stride defines the step size of the kernel when traversing the image. While its default is usually 1, we can use a stride of 2 for down sampling an image like MaxPooling. [30]

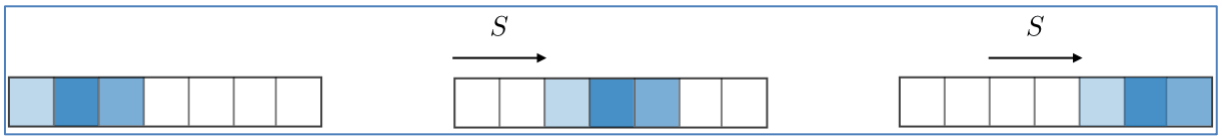

Figure 48: Stride

Padding: the process of adding P zeroes to each side of the boundaries of the input. This value can either be manually specified or automatically. The padding defines how the border of a sample is handled. A (half) padded convolution will keep the spatial output dimensions equal to the input, whereas unpadded convolutions will crop away some of the borders if the kernel is larger than 1. [30]

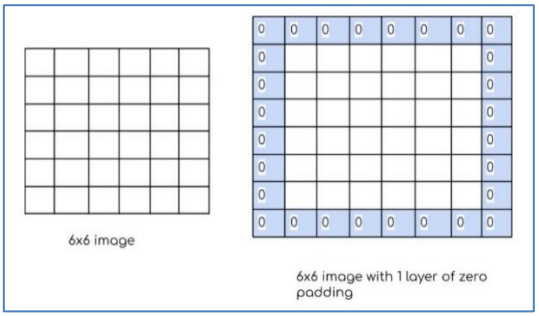

Figure 49 : Padding

Input & Output Channels: A convolutional layer takes a certain number of input channels (I) and calculates a specific number of output channels (O). The needed parameters for such a layer can be calculated by I*O*K, where K equals the number of values in the kernel. [30]

There are many types of convolutions, we will introduce the most common ones:

i. Normal Convolutions | 3x3 Convolution:

This is simply a 3x3 kernel running over an input image or channel to be specific. The resultant size is reduced by 2 pixels on all sides. [31]

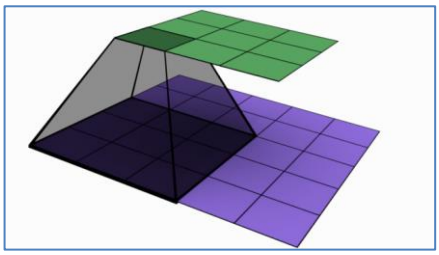

Figure 50 : Normal Convolution

ii. Pointwise Convolutions | 1x1 Convolutions:

1x1 convolutional kernels are used to reduce the number of channels (also increase if needed). [31]

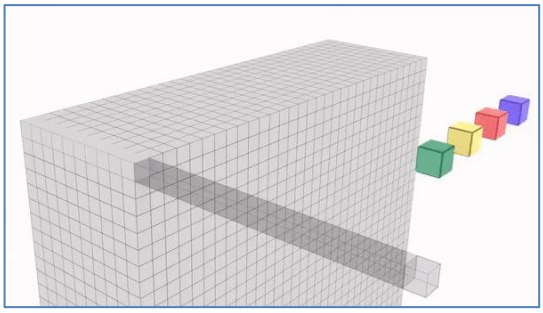

Figure 51 :Pointwise Convolution

iii. Dilated convolutions | Atrous Convolutions:

Dilated convolutions are a way of increasing the receptive field of the network. These helps in understanding the overall picture rather than finer details. Thus, they cut the compute cost and provide a global view of the network in much lesser depth (Indeed they are also our answer to the previous question we just asked earlier). [31]

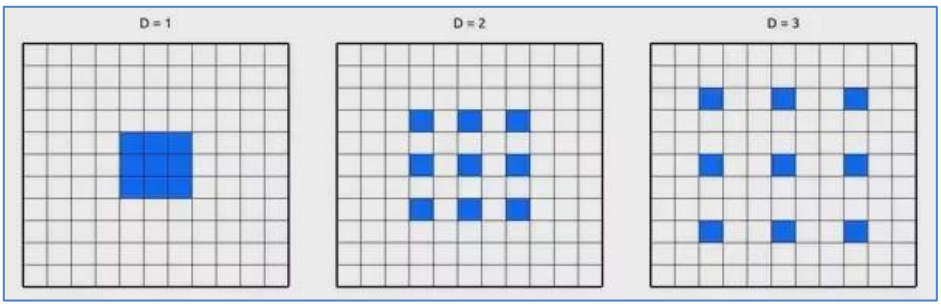

Figure 52 : Dilated convolution

With D=1, we get our normal 3x3 receptive field using a 3x3 kernel. When D=2, we get a receptive field of 5x5 using 3x3 kernel & with D=3, we get a receptive field of 7x7 with 3x3 kernel.

Dilated convolutions can be used where the object size is almost equal to the size of the image. In these cases, learning a few features of the object will give the right prediction. Hence we will not need every feature to make predictions. Thus, dilation convolutions will cut the computation cost & faster predictions. Also, it can be used in Pixel segmentations, Image super resolutions, denoising, keypoint detection, etc. [31]

iv. Deconvolution | Transpose convolution | Fractionally Strided Convolution:

Deconvolutional Neural Networks are CNNs that work in a reverse manner. When we use convolutional layers and max-pooling, the size of the image is reduced. To go to the original size, we use up sampling and transpose convolutional layers. Up sampling does not have trainable parameters—it just repeats the rows and columns of the image data by its corresponding sizes. [26]

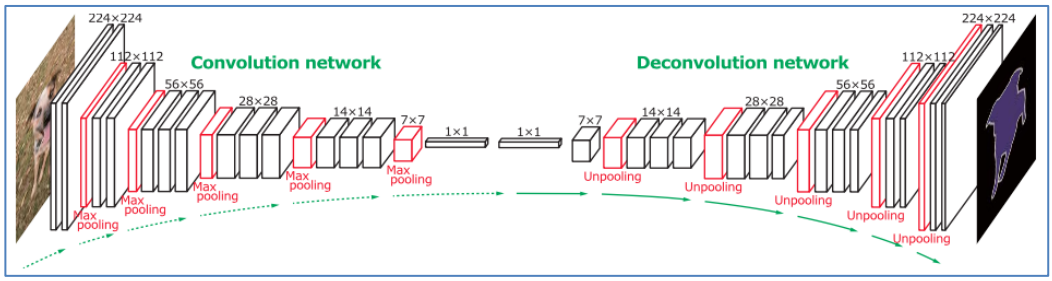

Figure 53 : Deconvolutional Neural Network

Transpose Convolutional layer means applying convolutional operation and up sampling at the same time. It is represented as Conv2DTranspose (number of filters, filter size, stride). If we set stride=1, we do not have any up sampling and receive an output of the same input size. [26]

v. Pixel Shuffle convolution for high resolution

The main aim of the pixel shuffle is for super-resolution. The pixel shuffle technique follows 2 simple steps. These are: [31]

Step1: Take the input image (MxNxC), run over a 1x1 kernel to extract the CxC number of channels.

Step2: From these 1,2, 3,…, N channels, where N is CxC, take one pixel at a time from each consecutive layer from 1,2 to N and then shuffle them to get the output.

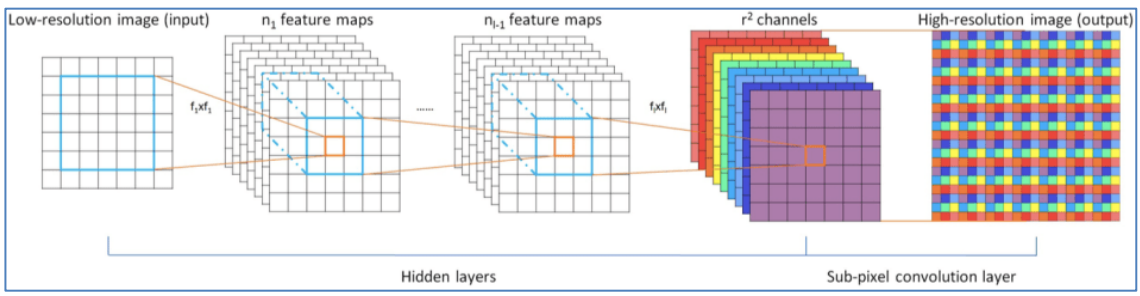

Figure 54 : Pixel Shuffle convolution

The input image size is 6x6 and 0's are padded on the border (Final input size: 7x7xC). Now we use 1x1xC kernels over it to get 7x7x(CxC) feature map. The final feature map is from 2nd to the right. Now if you observe the final output image, it consists of shuffled pixels. [31]

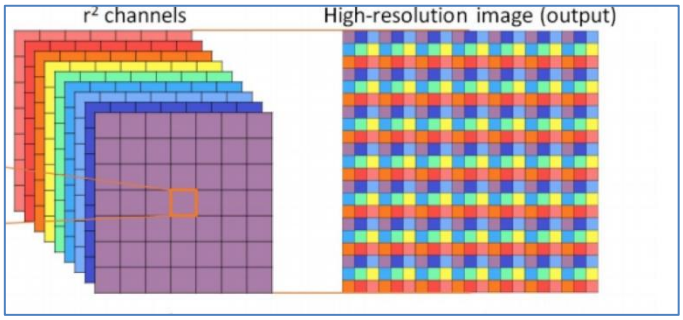

Figure 55 :Pixel Shuffle

Each pixel from the consecutive layer is chosen and arranged in a 3x3 subsection in the output. So, the 1st pixel of each feature map is chosen and arranged as a 3x3 grid. Then the 2nd pixel is chosen from all the feature maps and arranged as another 3x3 grid. Since we have a 7x7x9 feature map, our output is 21x21x1 because we took each pixel from 9 feature maps and arranged each pixel into 3x3 grids. So, we get 7*3 = 21 output size. Look at this animation now. [31]

vi. Depthwise Separable Convolution

Depthwise convolution says that instead of a 7x7x3 input image, we can say its 3 7x7x1 images. Hence, we would now need a 3x3 kernel with just 1 channel for each 7x7x1 image. [31]

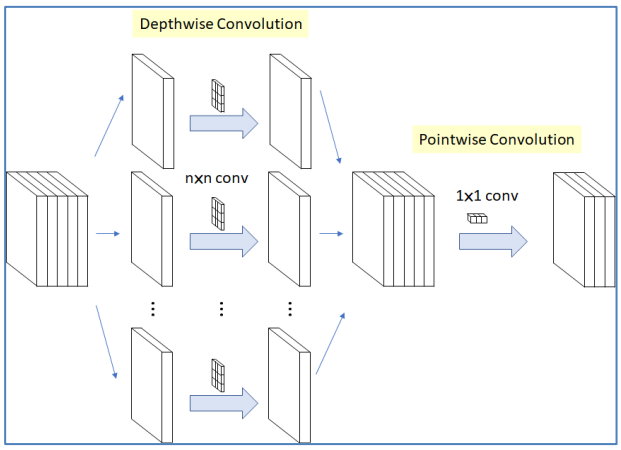

Figure 56 : Depthwise convolution

The input image is split into multiple images and then the kernel is convolved over it to get multiple feature maps. These feature maps are stacked over each other and then a 1x1xC (pointwise convolution) is used to get the desired number of channels. One of the main advantages of Depthwise convolution is that it reduced the number of parameters by 2 without even losing a significant amount of information.

vii. Spatially Separable Convolution

Spatially separable convolution is like the Depthwise convolution. It is also used when the number of parameters is a matter of concern. Spatially Separable convolution makes use of 2 different kinds of kernels to produce the output. [31]

viii. Grouped Convolutions

Grouped convolutions were initially mentioned in AlexNet and later reused in ResNext. The main motivation of such convolutions is to reduce computational complexity while dividing features into groups. Each group is expected to find out specific features from the network. These group features are then combined to form a single and denser feature map.

In grouped convolution, the filters are separated into different groups. Each group is responsible for conventional 2D convolutions with a certain depth. The following examples can make this clearer. [31]

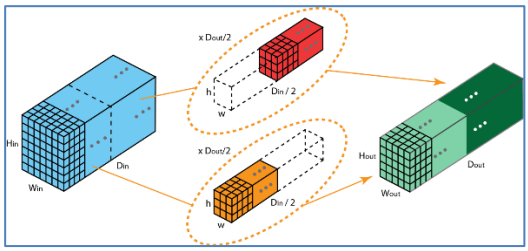

Figure 57 : Grouped convolution

- **ReLU Layer**

After each convolution operation, a CNN applies a Rectified Linear Unit (ReLU) transformation to the feature map, introducing nonlinearity to the model. [29]

Another convolution layer can follow the initial convolution layer. When this happens, the structure of the CNN can become hierarchical as the later layers can see the pixels within the receptive fields of prior layers. As an example, let's assume that we're trying to determine if an image contains a bicycle. You can think of the bicycle as a sum of parts. It is comprised of a frame, handlebars, wheels, pedals, et cetera. Each individual part of the bicycle makes up a lower-level pattern in the neural net, and the combination of its parts represents a higher-level pattern, creating a feature hierarchy within the CNN. [29]

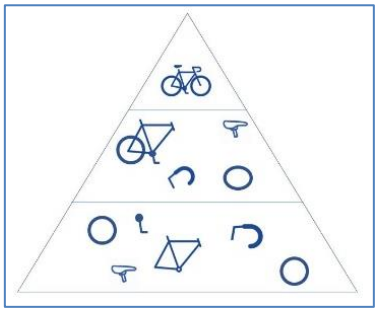

Figure 58 : ReLU Layer

Ultimately, the convolutional layer converts the image into numerical values, allowing the neural network to interpret and extract relevant patterns.

- **Pooling Layer**

Pooling layers, also known as downsampling, conducts dimensionality reduction, reducing the number of parameters in the input. Like the convolutional layer, the pooling operation sweeps a filter across the entire input, but the difference is that this filter does not have any weights. Instead, the kernel applies an aggregation function to the values within the receptive field, populating the output array. There are two main types of pooling: [29]

- **Max pooling:** As the filter moves across the input, it selects the pixel with the maximum value to send to the output array. As an aside, this approach tends to be used more often compared to average pooling.
- **Average pooling:** As the filter moves across the input, it calculates the average value within the receptive field to send to the output array.

While a lot of information is lost in the pooling layer, it also has a number of benefits to the CNN. They help to reduce complexity, improve efficiency, and limit risk of overfitting.

- **Flattering layer:**

Flattening is converting the data into a 1-dimensional array for inputting it to the next layer. We flatten the output of the convolutional layers to create a single long feature vector. And it is connected to the final classification model, which is called a fully connected layer. [32]

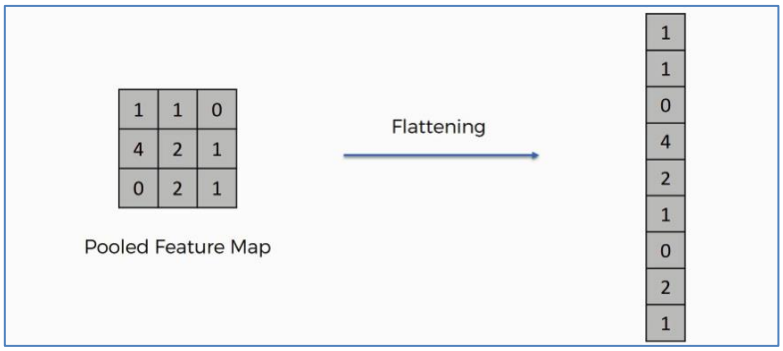

Figure 59 : Flattering layer

- **Fully-Connected Layer:**

The name of the full-connected layer aptly describes itself. As mentioned earlier, the pixel values of the input image are not directly connected to the output layer in partially connected layers. However, in the fully connected layer, each node in the output layer connects directly to a node in the previous layer. [29]

This layer performs the task of classification based on the features extracted through the previous layers and their different filters. While convolutional and pooling layers tend to use ReLu functions, FC layers usually leverage a softmax activation function to classify inputs appropriately, producing a probability from 0 to 1. [29]

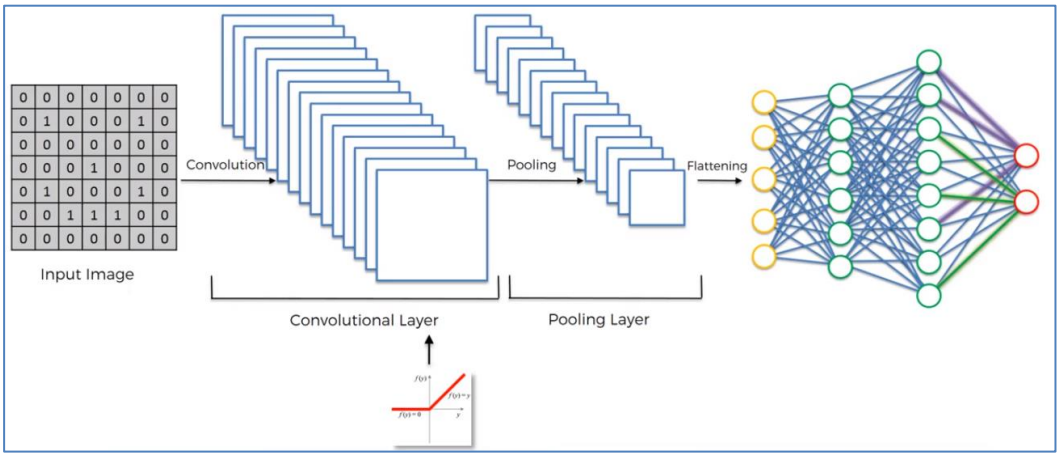

Figure 60 : full-connected layer

6. Autoencoder neural networks

Autoencoders are a specific type of feedforward neural networks where the input is the same as the output. They compress the input into a lower-dimensional code and then reconstruct the output from this representation. The code is a compact "summary" or "compression" of the input, also called the latent-space representation. [33]

An autoencoder consists of 3 components: encoder, code and decoder. The encoder compresses the input and produces the code, the decoder then reconstructs the input only using this code.

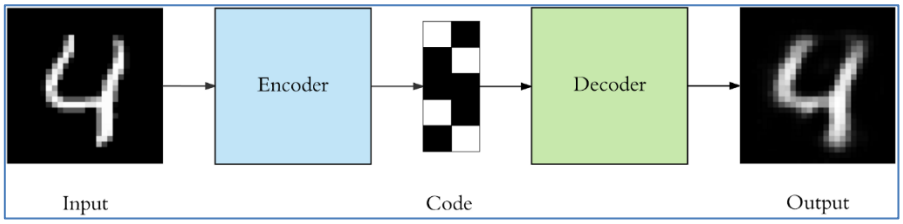

Figure 61 : Autoencoder Neural network

To build an autoencoder we need 3 things: an encoding method, decoding method, and a loss function to compare the output with the target. We will explore these in the next section. [33]

Autoencoders are mainly a dimensionality reduction (or compression) algorithm with a couple of important properties:

- Data-specific: Autoencoders are only able to meaningfully compress data similar to what they have been trained on. Since they learn features specific for the given training data, they are different than a standard data compression algorithm like gzip. So we can't expect an autoencoder trained on handwritten digits to compress landscape photos.

- Lossy: The output of the autoencoder will not be the same as the input, it will be a close but degraded representation. If you want lossless compression, they are not the way to go.
- Unsupervised: To train an autoencoder we don't need to do anything fancy, just throw the raw input data at it. Autoencoders are considered an unsupervised learning technique since they don't need explicit labels to train on. But to be more precise they are self-supervised because they generate their own labels from the training data.

V. Convolutional neural network architectures

c) AlexNet

AlexNet was trained on the Imagenet dataset with 15 million high-resolution images with 256*256*3. It has multiple convolutional layers and is deeper than the LeNet artificial neural network. [26]

The characteristics of AlexNet are :

- Dropout is added in this architecture to prevent overfitting.
- Data augmentation was performed as a pre-training process.
- ReLU activation function was used for the first time instead of sigmoid, Softmax.
- GPU learning was carried out for the first time
- Overlapping pooling was done in order to prevent information loss.
- It had five convolutional-pooling layer blocks followed by three fully connected dense layers for classification.

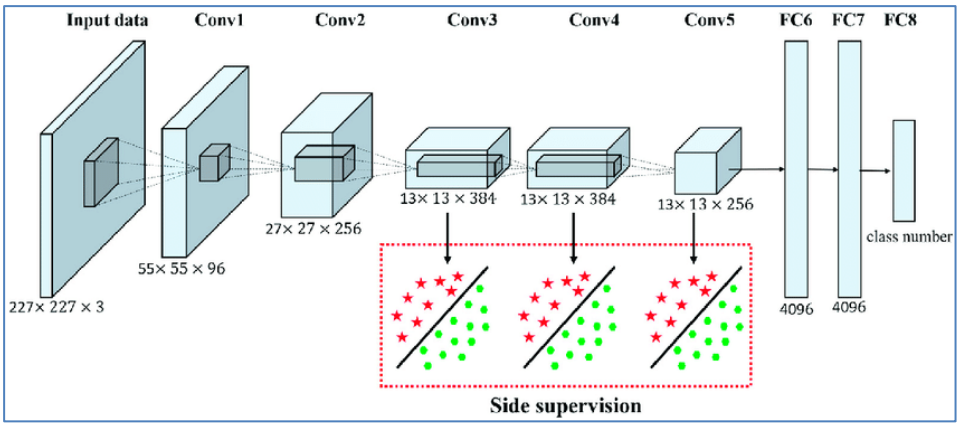

Figure 62 : AlexNet

d) Overfeat

This Neural Networks architecture explores three well-known vision tasks of classification, localization, and detection using a single framework. It trains the models on all three tasks simultaneously to boost up the accuracy. It is a modification of AlexNet. It predicts bounding boxes at each spatial location and scale. For localization, the classification head is replaced by a regression network. [26]

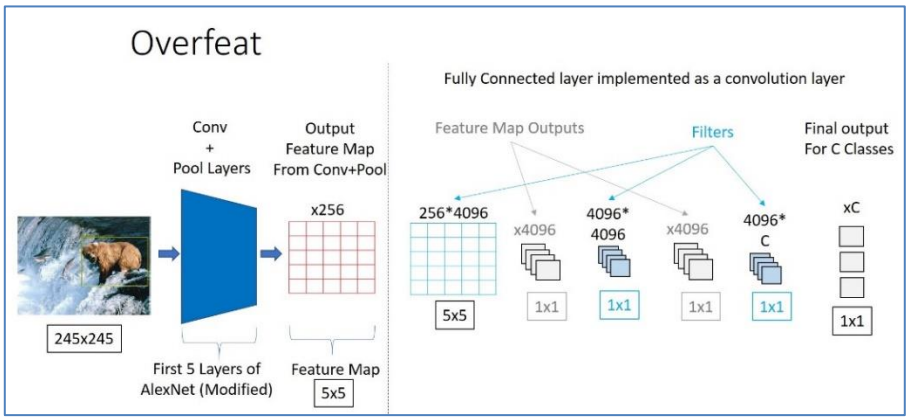

Figure 63: Overfeat

e) Visual Geometry Group (VGG)

VGG stands for Visual Geometry Group. One of the paths that we could take was to add more dense layers. This would bring with it more computations. The next possible approach was to have more convolutional layers. But this didn't work out as it was very tiring to define each convolutional layer separately. The best of all the solutions was to group convolutional layers into blocks. Eventually, the researchers concluded that more layers of narrow convolutions were more powerful than smaller numbers of wider convolutions. [26]

A VGG-block had a bunch of 3x3 convolutions padded by 1 to keep the output size the same as that of input, followed by max-pooling to half the resolution. The architecture had n number of VGG blocks followed by three fully connected dense layers. [26]

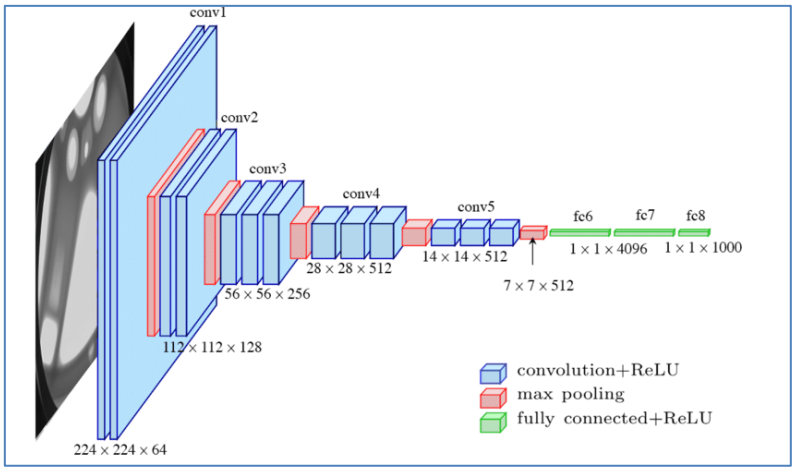

Figure 64 :VGG Architecture

f) Network-in-network

Convolutional layers need fewer parameters. It is the last few layers of fully connected neurons that bring a huge spike in the number of parameters. One way to solve this is to get rid of the fully connected layers. [26]

Convolutions and pooling reduce the resolutions, but at some point, we still need to map it to corresponding classes. Therefore, the idea was to reduce the resolution as we go deeper and increase the number of channels by using 1*1 convolutions. This gives us high-quality information per channel.

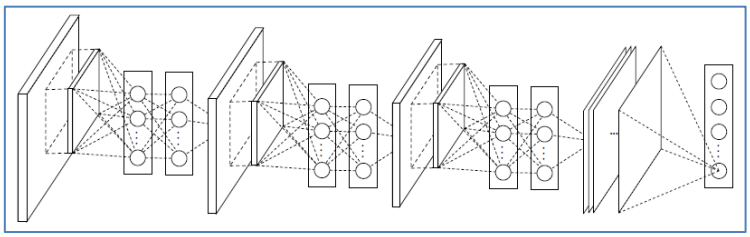

Figure 65 : Network-in-network architecture

In network-in-network architecture, the last fully connected layer is replaced by a global max-pooling layer making the model light.

g) GoogLeNet and Inception

Inception Neural Networks architecture has three convolutional layers with different size filters and max-pooling. Every layer has different size filters for parallel learning. There are different size filters to take care of huge variations in the location of information, which makes it very difficult to choose the right size filter. The small filter size convolutional layer takes care of a small information area. [26]

GoogleNet architecture consists of inception blocks that have 1x1, 3x3, 5x5 convolutional layers followed by 3x3 max pooling with padding (to make the output of the same shape as the input) on the previous layer, followed by concatenations of their output. [26]

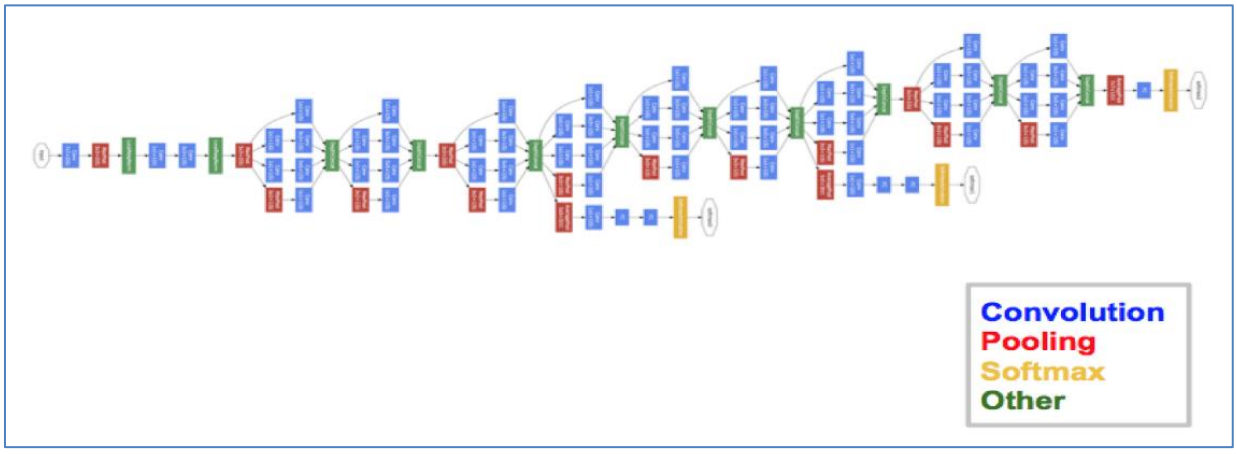

Figure 66 : Inception Architecture

h) SqueezeNet

It aims for smaller CNNs so that there is less communication across servers during distributed training. The changes it performs on the AlexNet architecture are as follows: [26]

- Replace 3*3 filters with 1*1 filters to reduce the number of parameters.
- Downsample later in the architecture so that the convolutional layers have large activation maps
- They squeeze the features with squeeze layers consisting of 1*1 convolutional layers and then they expand it with a combination of 1*1 and 3*3 convolutional layers. Each squeeze-expand block is placed together and is known as a fire module.

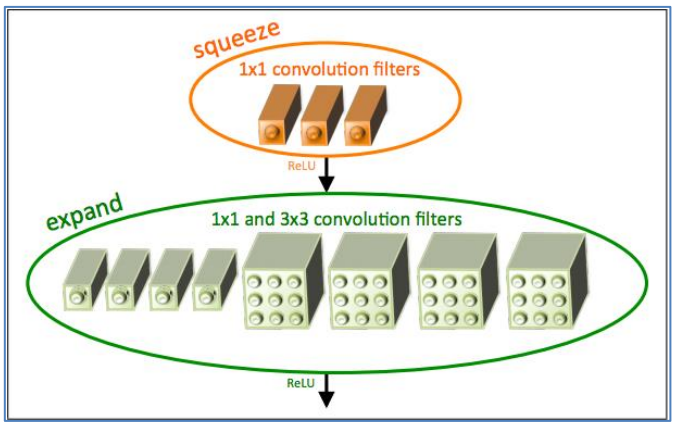

Figure 67 : SqueezeNet Architecture

i) Xception

The convolutional layer that is the basic building block of all CNN's involves a convolution operation. Each convolution operation involves the sliding of a filter through all the patches of the input pixel array. Each time the number of multiplications performed is equal to the number of elements present in the filter. [26]

In standard convolution, filters across all input channels and the combination of these values are done in a single step. Depthwise separable convolutions that are proposed in Xception architecture break down this operation into two parts:

- Depthwise convolution
- Pointwise convolution

j) MobileNets

MobileNets use depth-wise separable convolutions to build lightweight deep Neural Networks. They develop very small, low latency models that are used for applications like robots, self-driving cars, etc.

In a simple CNN structure, a filter is a block that is superimposed on the input image block, and the dot product is calculated between the two overlapping components. The details inside one channel are calculated along with the relationship between different channels. [26]

Instead of having one large filter, MobileNets have two filters:

- One goes through one channel at a time to detect how all the pixels in a channel are related
- The other goes through all the channels at the same time to see how one pixel is related to every other pixel that comes behind it.

k) Residual Networks (ResNet)

Very deep Neural Networks are extremely difficult to train due to vanishing and exploding gradient problems. ResNets provide an alternate pathway for data to flow to make the training process much faster and easier. This is different from the feed-forward approach of earlier Neural Networks architectures.

The core idea behind ResNet is that a deeper network can be made from a shallow network by copying weight from the shallow counterparts using identity mapping. The data from previous layers is fast-forwarded and copied much forward in the Neural Networks. This is what we call skip connections first introduced in Residual Networks to resolve vanishing gradients.

l) Capsule Networks

Instead of invariance, the network should strive for equivariance. It means that no matter what position or rotation a subsampled image is in, the neural network responds in the same way. It should also change accordingly to adapt to such sub-images. [26]

Neural Networks where instead of adding a layer, it nests a new layer inside a layer. This nested layer is called a capsule which is a group of neurons. Instead of making the structure deeper in terms of layers, a Capsule Network nests another layer within the same layer.

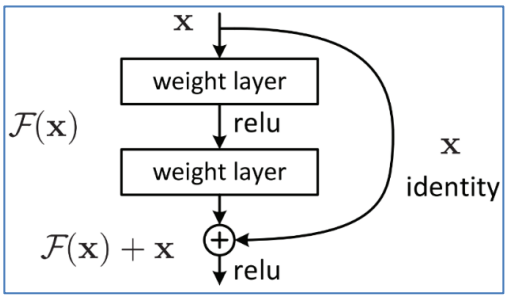

Figure 68 : Residual Network

VI. Grad Cam

After we saw what Convolutional neural network architectures are. We have grad cam algorithm to give us the opportunity to see how these architectures are making decisions.

While deep learning has facilitated unprecedented accuracy in image classification, object detection, and image segmentation, one of their biggest problems is model interpretability, a core component in model understanding and model debugging.

In practice, deep learning models are treated as "black box" methods, and many times we have no reasonable idea as to:

- Where the network is "looking" in the input image
- Which series of neurons activated in the forward-pass during inference/prediction
- How the network arrived at its final output

That raises an interesting question how can you trust the decisions of a model if you cannot properly validate how it arrived there? To help deep learning practitioners visually debug their models and properly understand where it's "looking" in an image, Selvaraju and al. created Gradient-weighted Class Activation Mapping, or more simply, Grad-CAM: [57]

Grad-CAM uses the gradients of any target concept, flowing into the final convolutional layer to produce a coarse localization map highlighting the important regions in the image for predicting the concept."

Using Grad-CAM, we can visually validate where our network is looking, verifying that it is indeed looking at the correct patterns in the image and activating around those patterns. [57]

If the network is not activating around the proper patterns/objects in the image, then we know:

- Our network hasn't properly learned the underlying patterns in our dataset
- Our training procedure needs to be revisited
- We may need to collect additional data
- And most importantly, *our model is not ready for deployment.*

Grad-CAM is a tool that should be in any deep learning practitioner's toolbox.

- **Grad-CAM++:**

Grad-CAM++ produces heatmaps at all locations of a class in case there are more than one, where the class can be positioned in a scattered or attached manner in the image. Also, the heatmap generated better explains the model's behavior when multiple instances of a single class are present in an image. [57]

Grad-CAM++ can localize the predicted class more accurately than Grad-CAM, which increases faithfulness to the model. We generated heatmaps for both the techniques (Grad-CAM and Grad-CAM++) and fused it with the actual image via point-wise multiplication. The behavior of the confidence score (of the deep network) for that class, when presented with the original image and heatmap-fused image, can be analyzed to conclude which method generates a more class-specific heatmap. A lower or no drop-in class score would indicate a higher localization of class-discriminative regions for a particular class. We provide results for this experiment in Section 4.1 which show a better localization capacity of Grad-CAM++ over Grad-CAM. [57]

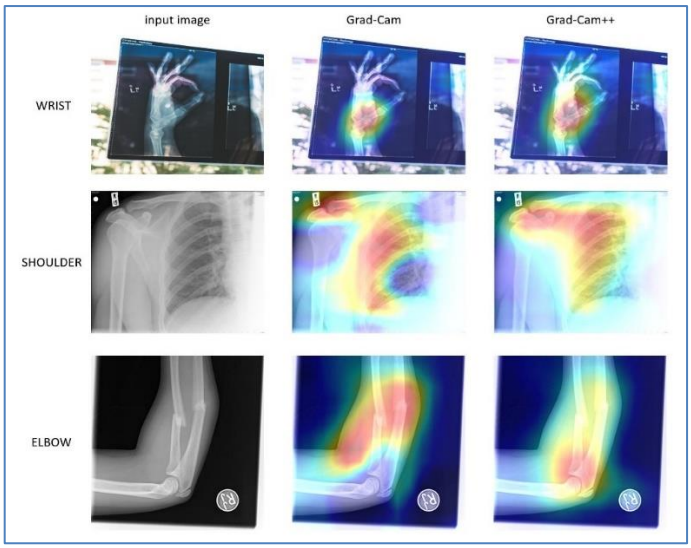

Figure 69 : Grad-Cam & Grad-cam++

VII. Conclusion

Each Neural Networks architecture has its own set of pros and cons. Standard Neural Networks like Feed-Forward Neural Networks are most commonly used in solving classification and regression problems related to simple structured data.

Recurrent Neural Networks are much more powerful in terms of remembering information for a long time and are used in sequential data like text, audio, video etc. Recent research shows that Transformers based on Attention Mechanism outperform RNNs and have almost replaced RNNs in every field. For complex data like images, we can use ConvNets in classification tasks and for generation of images or style transfer related tasks Generative Adversarial Networks performs the best.

Neural networks are also ideally suited to help people solve complex problems in real-life situations. They can learn and model the relationships between inputs and outputs that are nonlinear and complex. make generalizations and inferences; reveal hidden relationships, patterns and predictions; and model highly volatile data (such as financial time series data) and variances needed to predict rare events (such as fraud detection).

In the next chapter we will use these neural networks to make our Convolutional neural network for detecting Covid in chest X-Ray. We will se the impact of each architecture and choose the most accurate one.

Chapter 3: Conception of neural network for covid-19 x-ray

I. Introduction

In the previous chapter we learned about the neural networks, their activations functions, and other parameters. We also saw the differences between architectures and when to use each one.

In this chapter we will choose the most accurate known architectures and we trained them with the data that we collected. We will see the impact of each architecture and how accurate is it and we will apply a heat map to see how every model works.

In the end we will choose the best model and use it for our detecting website after taking consideration of a specialist opinion.

II. Model Conception

a) The used dataset for training and testing

To start the conception of our model we needed some data for training and testing. We found a dataset in the platform Kaggle (Chest X-Ray Images (Pneumonia)) it was divided in two sections: [34]

- Normal Chest X-Ray
- Pneumonia Chest X-Ray

We combined these two categories in one and we named it “None Covid Chest X-Ray” So that even if our patient has pneumonia the model doesn’t confuse it with Covid.

For Covid Chest X-Ray images, we found a dataset in GitHub. This dataset contains Covid Chest X-Rays. [35] We also used Covid Chest X-Rays form another Dataset from “radiopaedia”. [36]

In total we had:

	Covid X-Rays	**None Covid X-Rays**
Train	1618	1741
Test	102	624

b) CNN with VGG19

VGG19 architecture is included in keras, we used it with imageNet weights so that our model would have the basic definition of a chest X-Ray and the Training will focus more on the symptoms of covid. We trained our model till we got a satisfying result.

We Can resume the Training process in Accuracy plot:

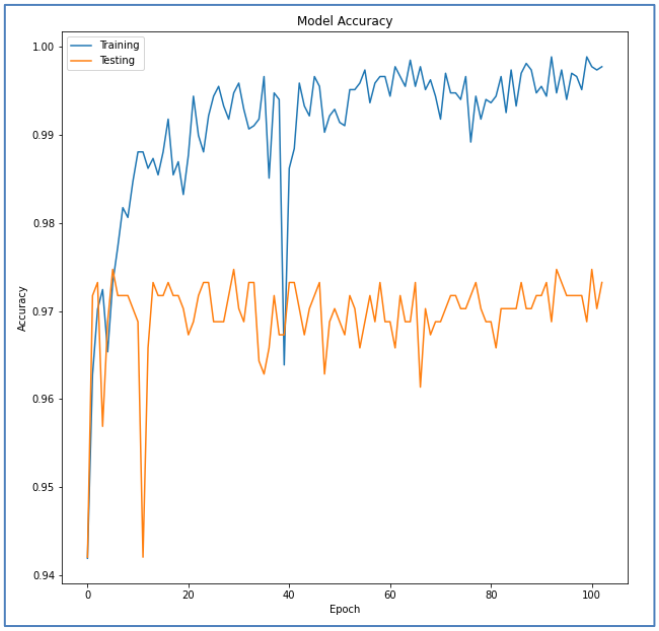

Figure 70 : Accuracy while training model based on VGG19

In this plot we can see that the accuracy has raised in the first 20 epochs. After that the training accuracy was around 99% and the test accuracy was around 97%.

After the training we Tested the accuracy on the Test data Set, and it gave us an accuracy of 92%

```
model.evaluate(X_test, Y_test)

== Test accuracy:
7/7 [==============================] - 1s 83ms/step - loss: 0.5637 - accuracy: 0.9200
```

Figure 71 : VGG19 model accuracy

We tested two random pics one with covid and one without we applied the model on these two pictures and we can see the results above the pictures.

- Chest X-Ray without Covid:

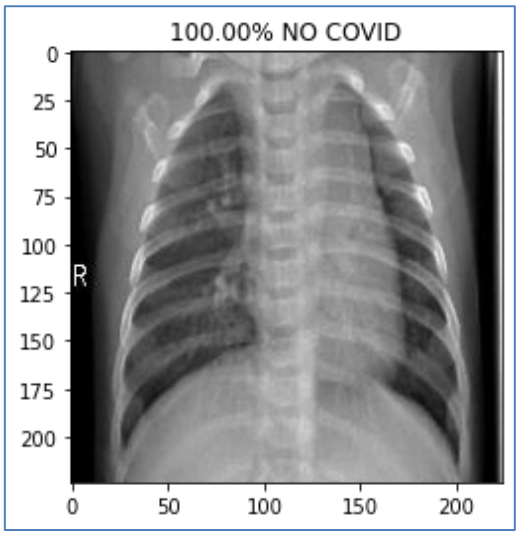

Figure 72 : Normal Chest X_Ray result with VGG19

- Chest X-Ray with Covid:

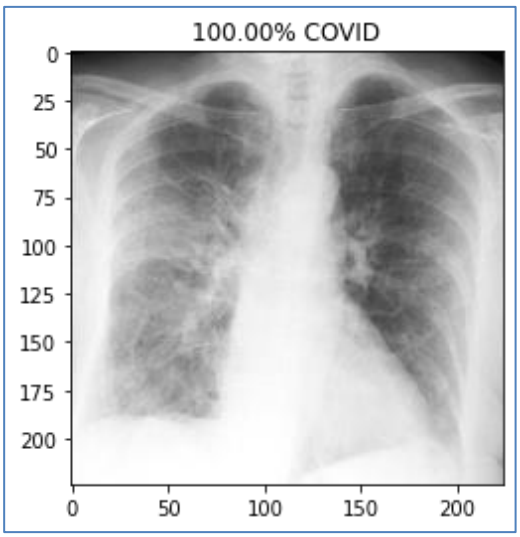

Figure 73 : Covid Chest X-Ray result with VGG19

Finally, we wanted to see how our model detected covid in this chest X-Ray, so we used Grad-Cam Algorithm which shows where the model was more activated to decide that it was covid. The most activated places are in red the least activated areas are with blue. As result the red areas are the places of covid.

The picture on the right is the original picture and the left one is with grad cam algorithm. As we can see the yellow and red parts are where covid is detected.

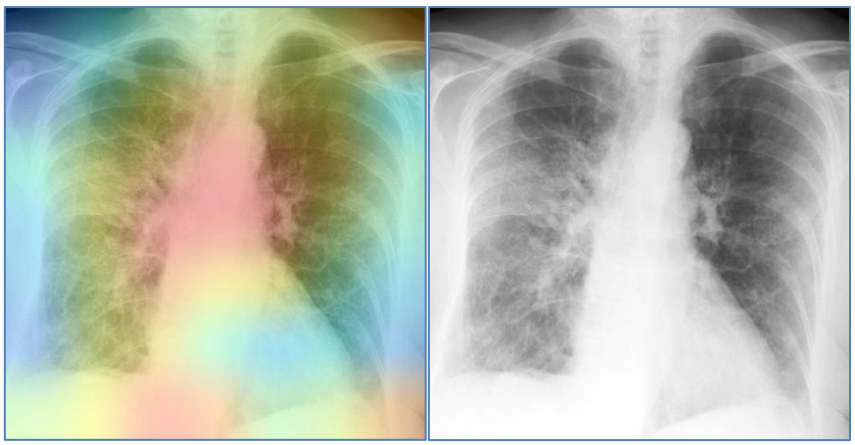

Figure 74 : Grad-Cam based on VGG19 model

c) CNN with EfficientNet

EfficientNet architecture is also included in keras, we used it with imageNet weights. We trained our model till we got a satisfying result.

We Can resume the Training process in Accuracy plot:

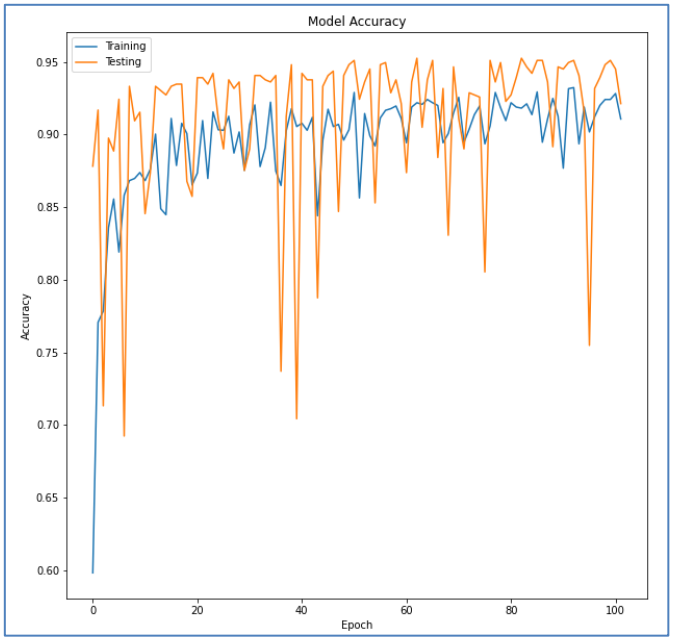

Figure 75 : Accuracy while training model based on EfficientNet

In this plot we can see that the accuracy has raised in the first 10 epochs. After that the training accuracy was around 90% and the test accuracy was around 95%.

After the training we Tested the accuracy on the Test data Set, and it gave us an accuracy of 89.5%

```
model.evaluate(X_test, Y_test)

== Test accuracy:
7/7 [==============================] - 7s 244ms/step - loss: 0.7158 - accuracy: 0.8950
```

Figure 76 : EfficientNet model accuracy

We tested two random pics one with covid and one without we applied the model on these two pictures and we can see the results above the pictures.

- Chest X-Ray without Covid:

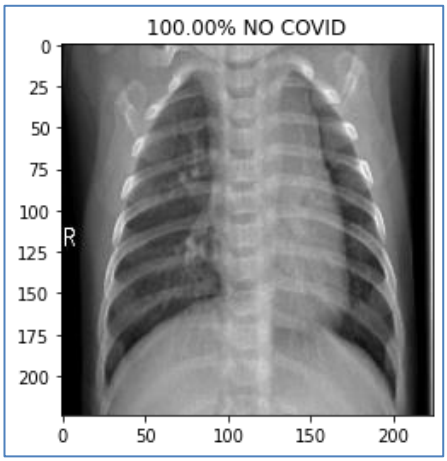

Figure 77 : Normal Chest X_Ray result with EfficientNet

- Chest X-Ray with Covid:

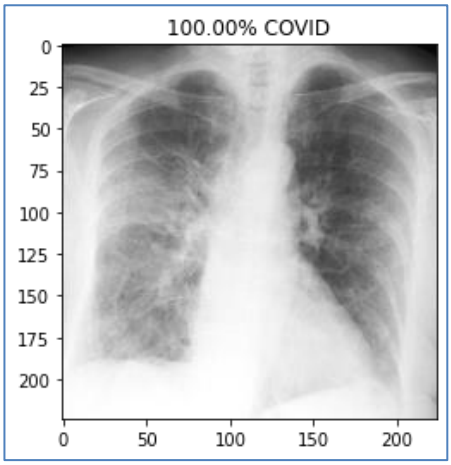

Figure 78 : Covid Chest X-Ray result with EfficientNet

Finally, we wanted to see how our model detected covid in this chest X-Ray, so we used Grad-Cam Algorithm which shows where the model was more activated to decide that it was covid. The most activated places are in red the least activated areas are with blue. As result the red areas are the places of covid.

The picture on the right is the original picture and the left one is with grad cam algorithm. As we can see the yellow and red parts are where covid is detected.

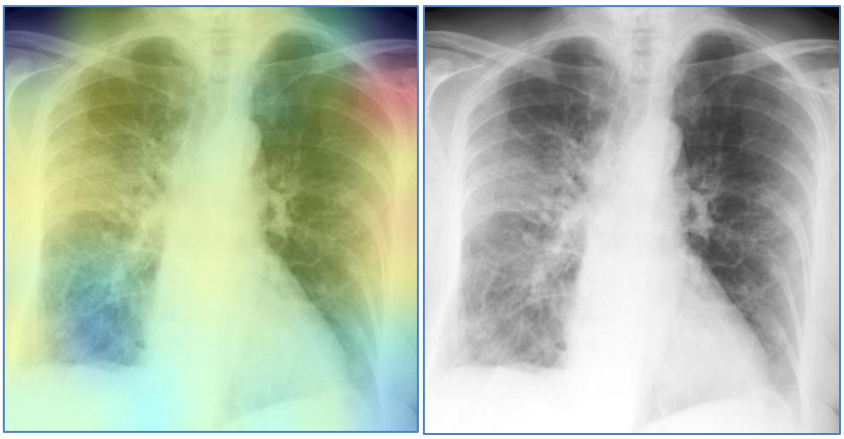

Figure 79 : Grad-Cam based on EfficientNet model

d) CNN with InceptionResNet

InceptionResNet architecture is also included in keras, we used it with imageNet weights. We trained our model till we got a satisfying result.

We Can resume the Training process in Accuracy plot:

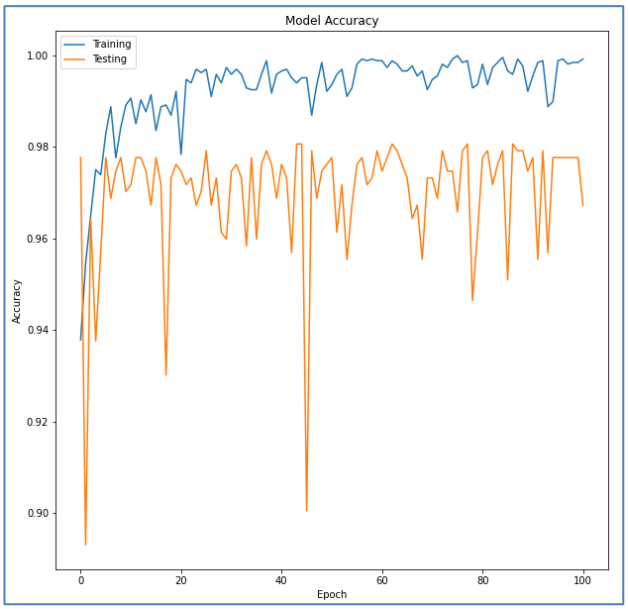

Figure 80 : Accuracy while training model based on InceptionResNet

In this plot we can see that the accuracy has raised in the first 10 epochs. After that the training accuracy was around 100% and the test accuracy was around 98%.

After the training we Tested the accuracy on the Test data Set, and it gave us an accuracy of 96%

```
== Test accuracy:
7/7 [==============================] - 1s 131ms/step - loss: 0.9460 - accuracy: 0.9600
[0.9459609985351562, 0.9599999785423279]
```

Figure 81: InceptionResNet model accuracy

We tested two random pics one with covid and one without we applied the model on these two pictures, and we can see the results above the pictures.

- Chest X-Ray without Covid:

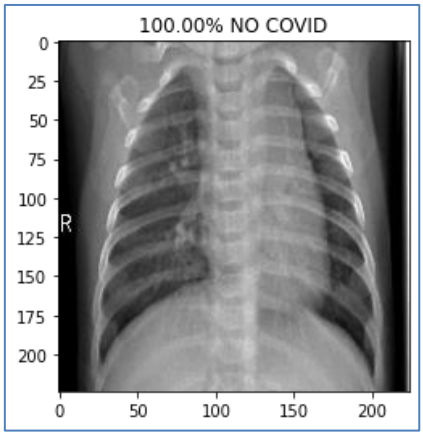

Figure 82 : Normal Chest X_Ray result with EfficientNet

- Chest X-Ray with Covid:

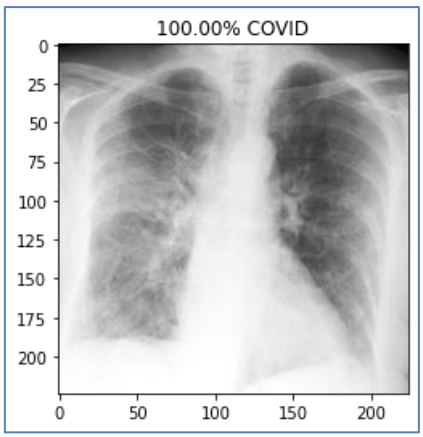

Figure 83 : Covid Chest X-Ray result with EfficientNet

Finally, we wanted to see how our model detected covid in this chest X-Ray, so we used Grad-Cam Algorithm which shows where the model was more activated to decide that it was covid. The most activated places are in red the least activated areas are with blue. As result the red areas are the places of covid. The picture on the right is the original picture and the left one is with grad cam algorithm. As we can see the yellow and red parts are where covid is detected.

Figure 84 : Grad-Cam based on InceptionResNet model

e) Comparison

We can resume the results of the three architectures in this graphics:

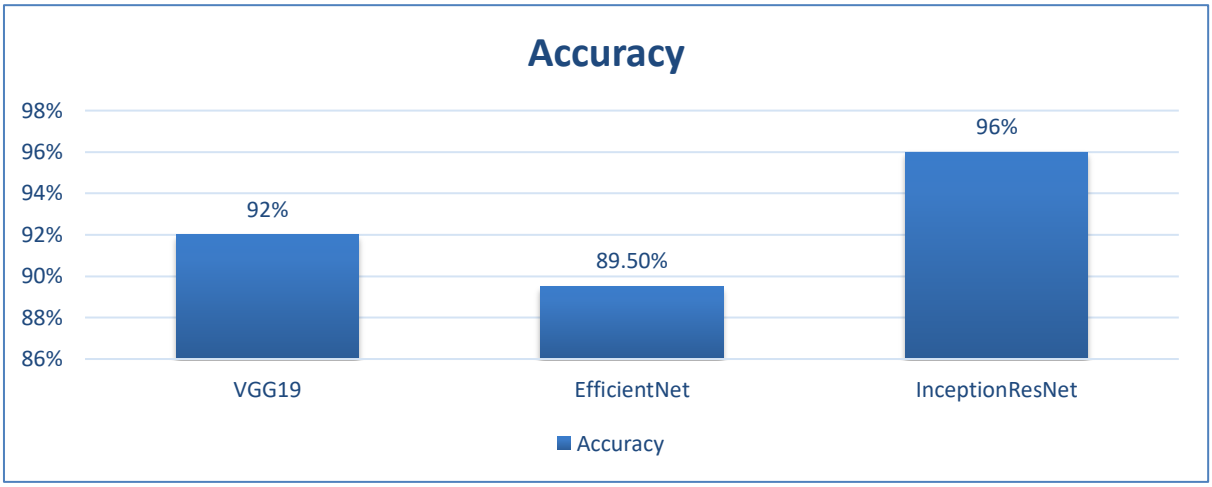

Figure 85 : Accuracy of all architectures

We trained the model with 100 epochs with early stopping. The training process gave us these results:

	VGG19	**EfficientNet**	**InceptionResNet**
Training Accuracy	100%	92%	100%
Testing Accuracy	92%	89.5%	96%

Figure 86 : Accuracy after training the model with 100 epochs

	VGG19	**EfficientNet**	**InceptionResNet**
Grad -Cam			

Figure 87 : Grad-Cam of all architectures

As we said before. The redness represents where the neural network was activated. For the three architectures VGG19 and InceptionResNet has similarities. We even asked a specialist about those results and the most accurate one was InceptionRest Net.

Therefor, after the value of accuracy and the Grad Cam results we choose to work with InceptionResNet because it's the most accurate and real from all above.

III. WebApp

We created a simple webapp to present our model. This webapp has three pages:

- The first one is the home page where the patient or the doctor can insert the Chest X-Ray that he wants to consult:

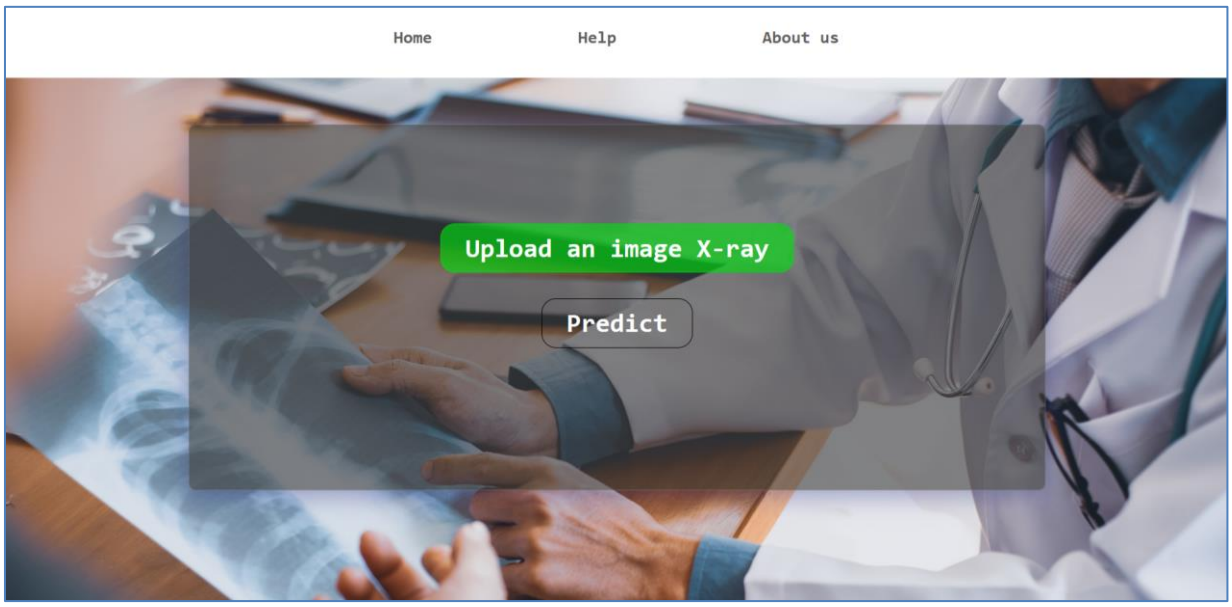

Figure 88 : Home Page

- This web page will redirect to the result page which shows if he has covid or not. And if so the heatmap over the chest X-Ray will appear:

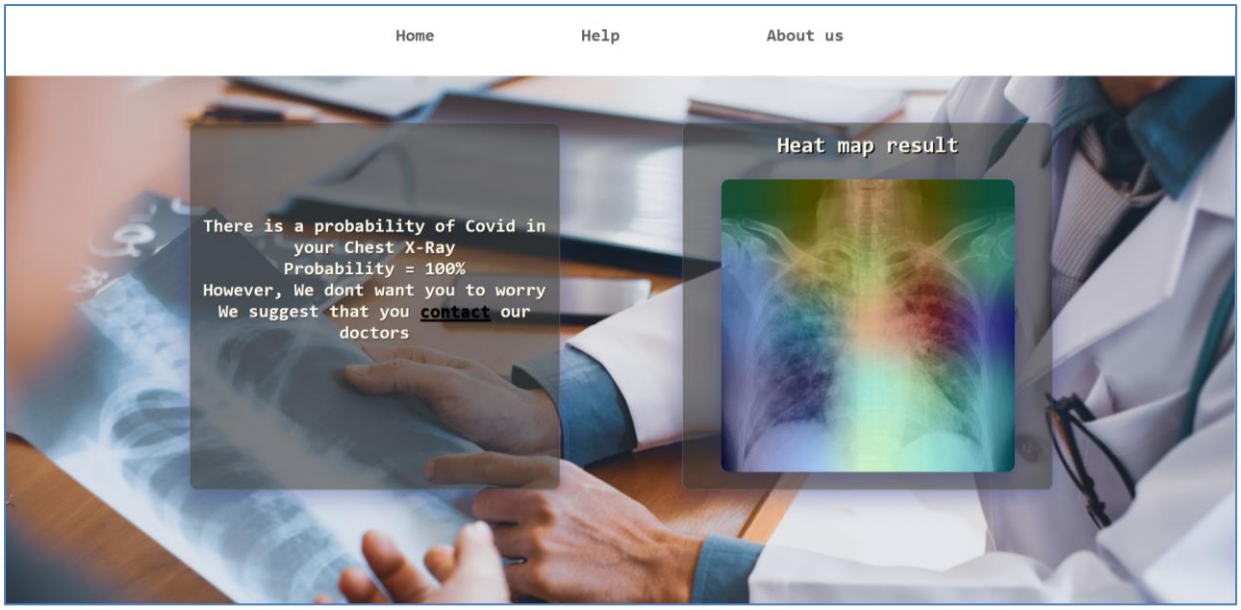

Figure 89 : Prediction Page for covid

The website will show the user where the chest is more damaged by covid.

- This is in case that he does not have covid. The web page will appear:

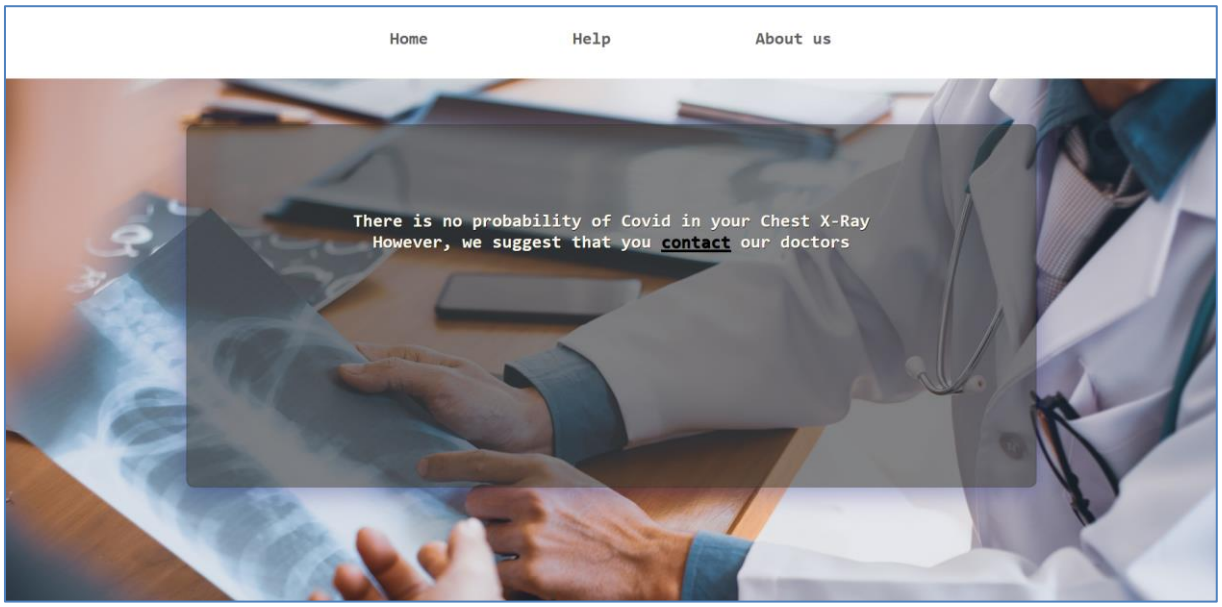

Figure 90 : Prediction Page for non covid

- Last, we have the contact page, if the patient wants to contact any doctors, we offer him some of the best doctors in Covid section:

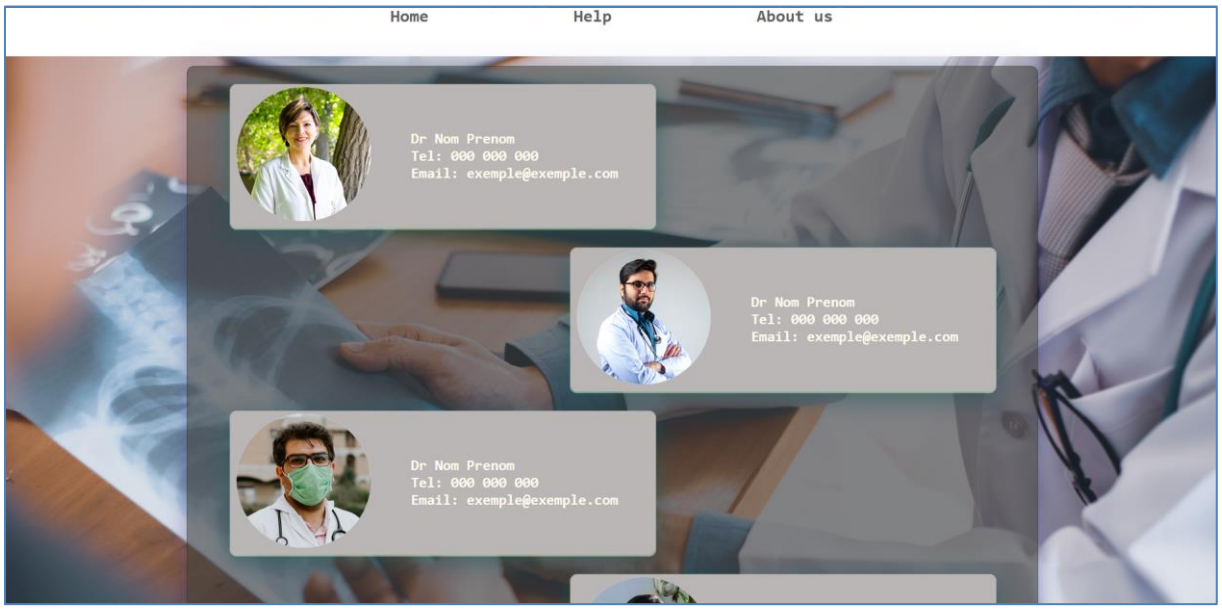

Figure 91 : Contact page

Appendix

In this section we will introduce you to some of the tools that we used to make our project.

I. Google collab

Colaboratory, or **"Colab"** for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education. More technically, Colab is a hosted Jupyter notebook service that requires no setup to use, while providing access free of charge to computing resources including GPUs.

Colab resources are not guaranteed and not unlimited, and the usage limits sometimes fluctuate. This is necessary for Colab to be able to provide resources free of charge. For more details, see Resource Limits. Users who are interested in more reliable access to better resources may be interested in Colab Pro. [37]

Figure 92 : Google colab logo

II. Python

Python is a programming language that lets you work more quickly and integrate your systems more effectively. Python can be easy to pick up whether you're a first-time programmer or experienced with other languages. Python is developed under an OSI-approved open-source license, making it freely usable and distributable, even for commercial use. [38]

AI projects differ from traditional software projects. The differences lie in the technology stack, the skills required for an AI-based project, and the necessity of deep research. To implement your AI aspirations, you should use a programming language that is stable, flexible, and has tools available. Python offers all of this, which is why we see lots of Python AI projects today. [39]

From development to deployment and maintenance, Python helps developers be productive and confident about the software they're building. Benefits that make Python the best fit for machine learning and AI-based projects include simplicity and consistency, access to great libraries and frameworks for AI and machine learning (ML), flexibility, platform independence, and a wide community. These add to the overall popularity of the language. [39]

Python offers concise and readable code. While complex algorithms and versatile workflows stand behind machine learning and AI, Python's simplicity allows developers to write reliable systems.

Developers get to put all their effort into solving an ML problem instead of focusing on the technical nuances of the language. [39]

Additionally, Python is appealing to many developers as it's easy to learn. Python code is understandable by humans, which makes it easier to build models for machine learning.

It's generally accepted that Python is suitable for collaborative implementation when multiple developers are involved. Since Python is a general-purpose language, it can do a set of complex machine learning tasks and enable you to build prototypes quickly that allow you to test your product for machine learning purposes. [39]

To reduce development time, programmers turn to several Python frameworks and libraries. A software library is pre-written code that developers use to solve common programming tasks. Python, with its rich technology stack, has an extensive set of libraries for artificial intelligence and machine learning. Here are some of them:

- **Keras, TensorFlow, and Scikit-learn** for machine learning
- **NumPy** for high-performance scientific computing and data analysis
- **SciPy** for advanced computing
- **Pandas** for general-purpose data analysis
- **Seaborn** for data visualization

With these solutions, you can develop your product faster. Your development team won't have to reinvent the wheel and can use an existing library to implement necessary features.

Python as the best language for AI development

Spam filters, recommendation systems, search engines, personal assistants, and fraud detection systems are all made possible by AI and machine learning, and there are more things to come. Product owners want to build apps that perform well. This requires coming up with algorithms that process information intelligently, making software act like a human. [39]

III. Other libraries

a) Numpy

NumPy stands for Numerical Python 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. It also has functions for working in domain of linear algebra, Fourier transform, and matrices. [40]

NumPy was created in 2005 by Travis Oliphant. It is an open-source project, and you can use it freely. In Python we have lists that serve the purpose of arrays, but they are slow to process. NumPy aims to provide an array object that is up to 50x faster than traditional Python lists. The array object in NumPy is called NDarray, it provides a lot of supporting functions that make working with NDarray very easy. Arrays are very frequently used in data science, where speed and resources are very important. [41]

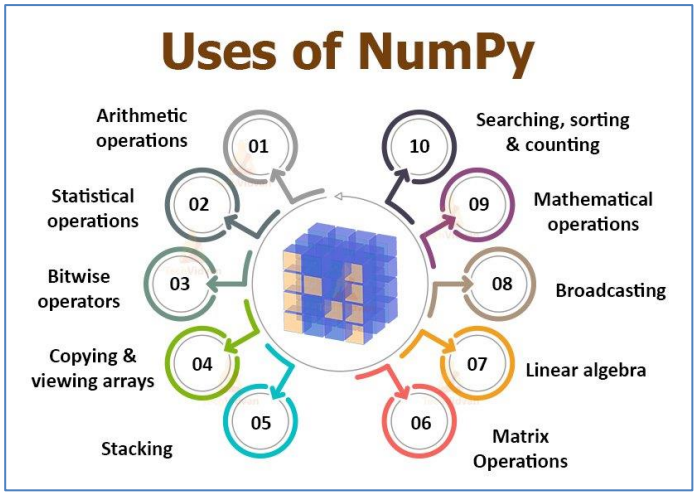

Figure 93 : Uses of NumPy

IV. Pandas

Pandas, stands for "Python Data Analysis Library". is quite a game changer when it comes to analyzing data with Python and it is one of the most preferred and widely used tools in data munging/wrangling if not THE most used one. Pandas is an open source, free to use (under a BSD license) and it was originally written by Wes McKinney (here's a link to his GitHub page). [42]

The special thing about Pandas is that it takes data (like a CSV or TSV file, or a SQL database) and creates a Python object with rows and columns called data frame that looks very similar to table in a statistical software (think Excel or SPSS for example. People who are familiar with R would see similarities to R too). [42]

Pandas provides: [43]

- A fast and efficient DataFrame object for data manipulation with integrated indexing.
- Tools for reading and writing data between in-memory data structures and different formats: CSV and text files, Microsoft Excel, SQL databases, and the fast HDF5 format.

- Intelligent data alignment and integrated handling of missing data: gain automatic label-based alignment in computations and easily manipulate messy data into an orderly form.
- Flexible reshaping and pivoting of data sets.
- Intelligent label-based slicing, fancy indexing, and subsetting of large data sets.
- Columns can be inserted and deleted from data structures for size mutability.
- Aggregating or transforming data with a powerful group by engine allowing split-apply-combine operations on data sets.
- High performance merging and joining of data sets.
- Hierarchical axis indexing provides an intuitive way of working with high-dimensional data in a lower-dimensional data structure.
- Time series-functionality: date range generation and frequency conversion, moving window statistics, date shifting and lagging. Even create domain-specific time offsets and join time series without losing data.
- Highly optimized for performance, with critical code paths written in Cython or C.

Python with pandas is in use in a wide variety of academic and commercial domains, including Finance, Neuroscience, Economics, Statistics, Advertising, Web Analytics, and more. [43]

Pandas aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open-source data analysis / manipulation tool available in any language.

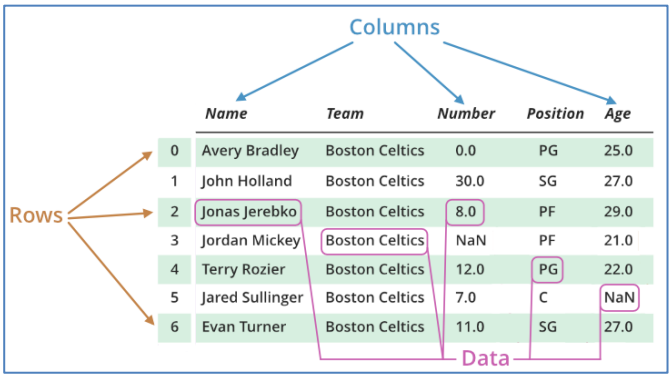

Figure 94 : Dataframes with pandas

V. Matplotlib

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible. [44]

- Create publication quality plots.
- Make interactive figures that can zoom, pan, update.
- Customize visual style and layout.

- Export to many file formats .
- Embed in JupyterLab and Graphical User Interfaces.
- Use a rich array of third-party packages built on Matplotlib.

Many packages extend and build on Matplotlib functionality, including several higher-level plotting interfaces (seaborn, HoloViews, ggplot, ...), and a projection and mapping toolkit (Cartopy).

- **Seaborn:**

Seaborn is a high-level interface for drawing statistical graphics with Matplotlib. It aims to make visualization a central part of exploring and understanding complex datasets. [45]

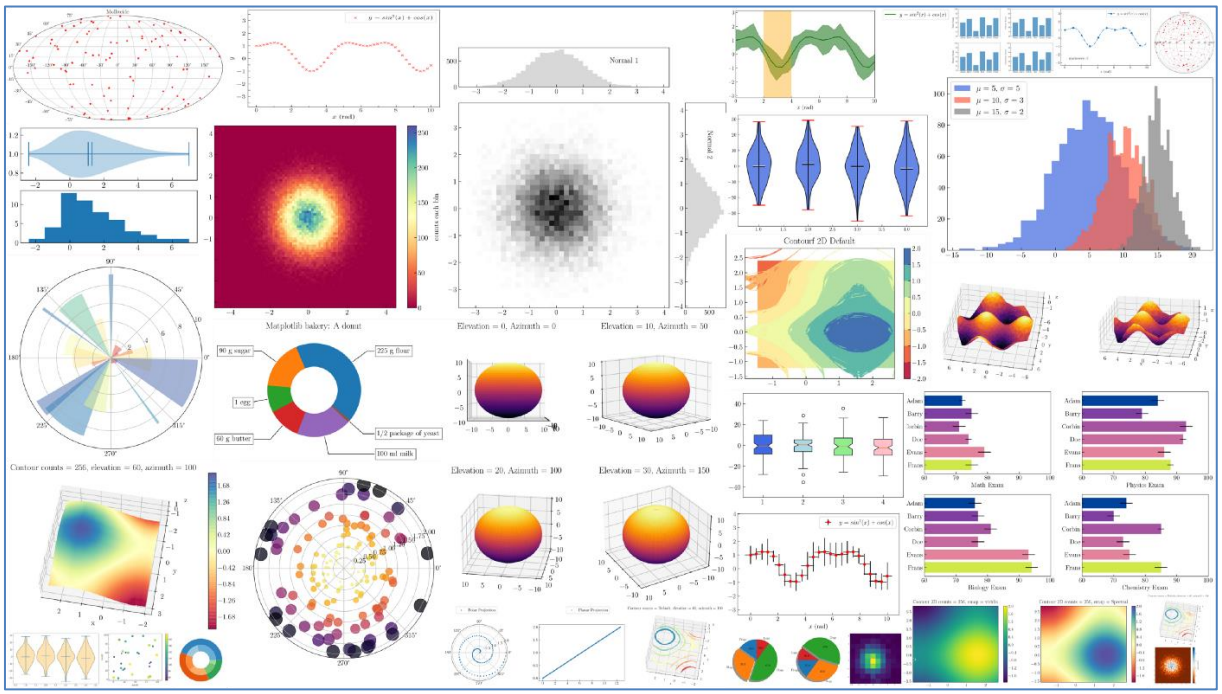

Figure 95 : Seaborn graphics

VI. Tensorflow

In the past, developing deep neural networks like CNNs has been a challenge because of the complexity of available training and inference libraries. TensorFlow, a machine learning framework that was open sourced by Google in November 2015, is designed to simplify the development of deep neural networks. [46]

TensorFlow provides high-level interfaces to different kinds of neuron layers and popular loss functions, which makes it easier to implement different CNN model architectures. TensorFlow models are also portable: the framework supports model execution natively on mobile devices ("AI on the edge") or in servers hosted remotely in the cloud. This enables a "create once, run anywhere" approach for model execution across many different platforms, including web-based and mobile. [46]

TensorFlow is an end-to-end open-source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries, and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML-powered applications. [47]

It provides:

- Efficiently executing low-level tensor operations on CPU, GPU, or TPU.
- Computing the gradient of arbitrary differentiable expressions.
- Scaling computation to many devices, such as clusters of hundreds of GPUs.
- Exporting programs ("graphs") to external runtimes such as servers, browsers, mobile and embedded devices.

TensorFlow was originally developed by researchers and engineers working on the Google Brain team within Google's Machine Intelligence Research organization to conduct machine learning and deep neural networks research. The system is general enough to be applicable in a wide variety of other domains, as well. [47]

TensorFlow provides stable Python and C++ APIs, as well as non-guaranteed backward compatible API for other languages. TensorFlow offers multiple levels of abstraction so you can choose the right one for your needs. Build and train models by using the high-level Keras API, which makes getting started with TensorFlow and machine learning easy. [48]

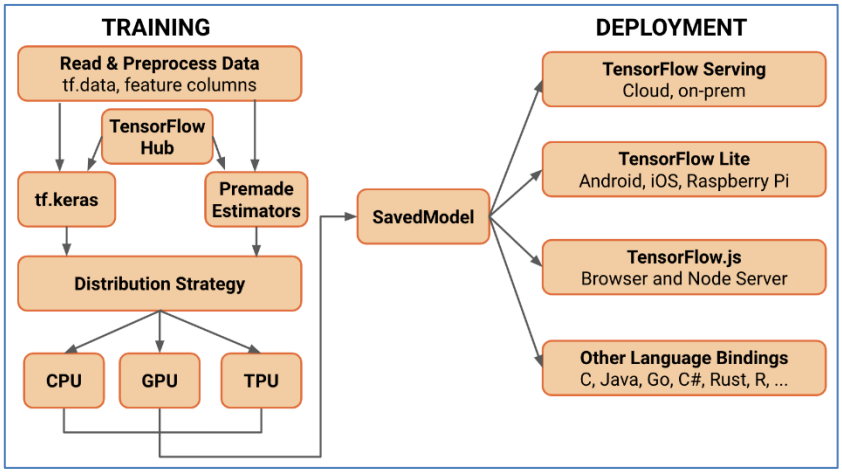

Figure 96 : TensorFlow uses

VII. keras

Keras is an API designed for human beings, not machines. Keras follows best practices for reducing cognitive load: it offers consistent & simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear & actionable error messages. It also has extensive documentation and developer guides. [49]

Keras is the most used deep learning framework among top-5 winning teams on Kaggle. Because Keras makes it easier to run new experiments, it empowers you to try more ideas than your competition, faster. And this is how you win. [49]

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 doing good research. [50]

Keras is:

- **Simple** but not simplistic: Keras reduces developer cognitive load to free you to focus on the parts of the problem that really matter.
- **Flexible**: Keras adopts the principle of progressive disclosure of complexity: simple workflows should be quick and easy, while arbitrarily advanced workflows should be possible via a clear path that builds upon what you've already learned.
- **Powerful**: Keras provides industry-strength performance and scalability: it is used by organizations and companies including NASA, YouTube, and Waymo.

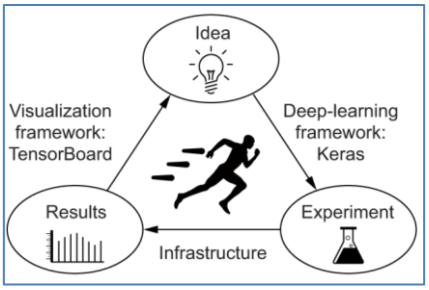

Figure 97 : Keras

Keras is the high-level API of TensorFlow 2: 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. [50]

Keras empowers engineers and researchers to take full advantage of the scalability and cross-platform capabilities of TensorFlow 2:

- run Keras on TPU or on large clusters of GPUs,
- export your Keras models to run in the browser or on a mobile device.

a) Batch size

Batch size is a term used in machine learning and refers to the number of training examples utilized in one iteration. The batch size can be one of three options: [51]

- **batch mode**: where the batch size is equal to the total dataset thus making the iteration and epoch values equivalent
- **mini-batch mode**: where the batch size is greater than one but less than the total dataset size. Usually, a number that can be divided into the total dataset size.
- **stochastic mode**: where the batch size is equal to one. Therefore, the gradient and the neural network parameters are updated after each sample.

Generally, the larger the batch size, the quicker our model will complete each epoch during training. This is because, depending on our computational resources, our machine may be able to process much more than one single sample at a time.

The trade-off, however, is that even if our machine can handle very large batches, the quality of the model may degrade as we set our batch larger and may ultimately cause the model to be unable to generalize well on data it hasn't seen before.

In general, the batch size is another one of the hyperparameters that we must test, and tune based on how our specific model is performing during training. This parameter will also have to be tested regarding how our machine is performing in terms of its resource utilization when using different batch sizes. [52]

A larger batch size will give you a better gradient and will help to prevent jumping around. Too large of a batch_size can produce memory problems, especially if you are using a GPU. Once you exceed the limit, dial it back until it works. This will help you find the max batch-size that your system can work with. [53]

b) Steps Per Epoch

It is useful if you have a huge data set or if you are generating random data augmentations on the fly. *steps_per_epoch* is batches of samples to train. It is used to define how many batches of samples to use in one epoch. It is used to declaring one epoch finished and starting the next epoch. If you have a training set of the fixed size, you can ignore it. [54]

Traditionally, the steps per epoch is calculated as *train_length* or *batch_size*, since this will use all of the data points, one batch size worth at a time.

If you are augmenting the data, then you can stretch this a tad (sometimes I multiply that function above by 2 or 3 etc. But, if it's already training for too long, then I would just stick with the traditional approach.

c) Validation Steps

Validation steps are like steps_per_epoch but it is on the validation data instead of the training data. If you have a validation dataset fixed size, you can ignore it. [54]

It is only relevant if validation_data is provided and is a tf.data dataset object. If validation_steps is specified and only part of the dataset will be consumed, the evaluation will start from the beginning of the dataset at each epoch. This ensures that the same validation samples are used every time.

- **Calculate steps_per_epoch and validation_steps**

By default, both parameters are None is equal to the number of samples in your dataset divided by the batch size or 1 if that cannot be determined. [54]

If the input data is a tf.data dataset object, and steps_per_epoch is None, the epoch will run until the input dataset is empty. This argument is not supported with array inputs.

If you want to your model passes through all of your training data one time in each epoch you should provide steps per epoch equal to a number of batches like this:

- ***Validation Split***

The fraction of the training data to be used as validation data. It is a float between 0 and 1 and will evaluate the loss and any model metrics on this data at the end of each epoch. [54]

Validation data is always fixed and taken from the bottom of the training dataset when you set the *validation_ split*. The function is designed to ensure that the data is separated in such a way that it always trains on the same portion of the data for each epoch. All shuffling is done after split training data. The model will not train on this fraction of the validation data.

However, for some datasets, the last few instances are not useful, specifically if the dataset is regroup based on class. Then the distribution of your classes will be skewed. For this, I always like to use the sklearn function *train_test_split* and you confirm that this is a better method since it will randomly get test/validation data from the dataset.

from sklearn.model_selection import train_test_split
train_test_split(X, Y, test_size=0.2, random_state=42)

This argument is not supported when x is a dataset, generator or keras.utils.Sequence instance. Based on what you said it sounds like you need a larger batch_size, and of course there are implications with that which could impact the steps_per_epoch and number of epochs.

- **When to reduce epochs**

If your train error is very low, and your test/validation is very high, then you have over-fit the model with too many epochs. [53]

The best way to find the right balance is to use early stopping with a validation test set. Here you can specify when to stop training and save the weights for the network that gives you the best validation loss. (I highly recommend using this always)

VIII. Optimizers

Optimizers are algorithms or methods used to minimize an error function (loss function) or to maximize the efficiency of production. Optimizers are mathematical functions which are dependent on model's learnable parameters (Weights & Biases). Optimizers help to know how to change weights and learning rate of neural network to reduce the losses. [55]

We have multiple types of optimizers, we will introduce a few of them:

a) Gradient Descent

This optimization algorithm uses calculus to modify the values consistently and to achieve the local minimum. Gradient descent works best for most purposes. However, It is expensive to calculate the gradients if the size of the data is huge. Gradient descent works well for convex functions, but it doesn't know how far to travel along the gradient for nonconvex functions. [56]

$$x\_new = x - alpha * f'(x)$$

Figure 98 : Gradient Descent formula

The above equation means how the gradient is calculated. Here alpha is step size that represents how far to move against each gradient with each iteration.

Gradient descent works as follows:

- It starts with some coefficients, sees their cost, and searches for cost value lesser than what it is now.
- It moves towards the lower weight and updates the value of the coefficients.
- The process repeats until the local minimum is reached. A local minimum is a point beyond which it cannot proceed.

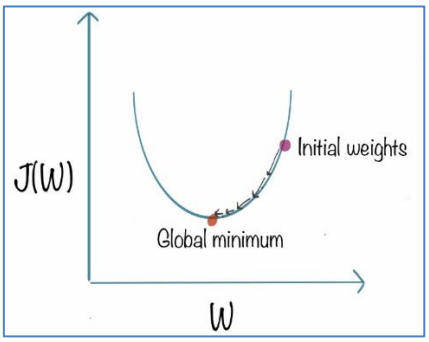

Figure 99 : Gradient Descent

How big/small the steps are gradient descent takes into the direction of the local minimum are determined by the learning rate, which figures out how fast or slow we will move towards the optimal weights. [55]

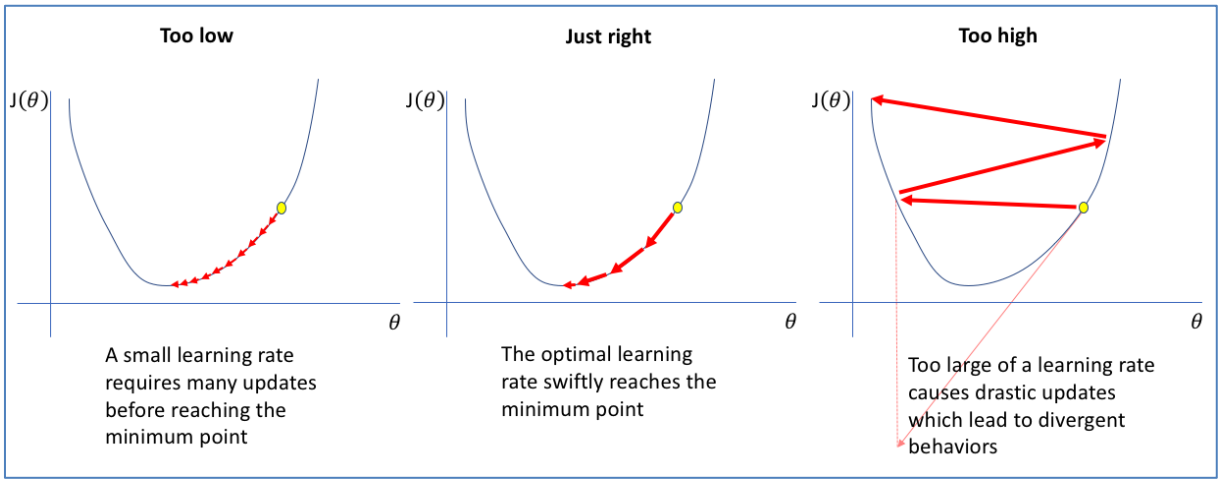

Figure 100 : Learning Rate

b) Stochastic Gradient Descent

The term stochastic means randomness on which the algorithm is based upon. In stochastic gradient descent, instead of taking the whole dataset for each iteration, we randomly select the batches of data. That means we only take few samples from the dataset. [56]

$$w := w - \eta \nabla Q_i(w).$$

Figure 101 : Stochastic Gradient formula

The procedure is first to select the initial parameters w and learning rate n. Then randomly shuffle the data at each iteration to reach an approximate minimum.

Since we are not using the whole dataset but the batches of it for each iteration, the path took by the algorithm is full of noise as compared to the gradient descent algorithm. Thus, Stochastic Gradient Descent uses a higher number of iterations to reach the local minima. Due to an increase in the number of iterations, the overall computation time increases. But even after increasing the number of iterations, the computation cost is still less than that of the gradient descent optimizer. So, the conclusion is if the data is enormous and computational time is an essential factor, stochastic gradient descent should be preferred over batch gradient descent algorithm. [56]

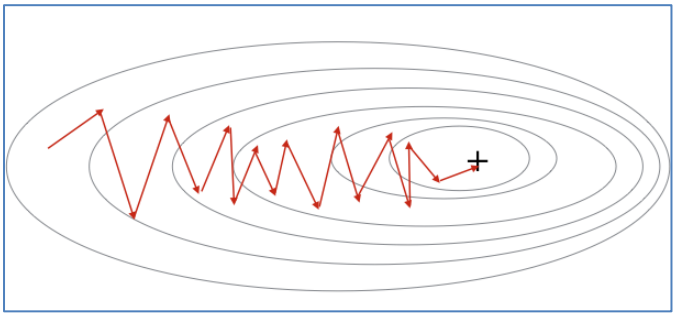

Figure 102 : Stochastic Gradient Descent

c) Stochastic Gradient Descent with Momentum

SGD with Momentum is a stochastic optimization method that adds a momentum term to regular stochastic gradient descent. Momentum simulates the inertia of an object when it is moving, that is, the direction of the previous update is retained to a certain extent during the update, while the current update gradient is used to fine-tune the final update direction. In this way, you can increase the stability to a certain extent, so that you can learn faster, and have the ability to get rid of local optimization. [55]

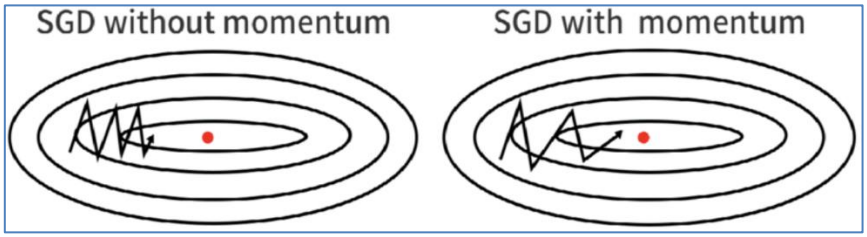

Figure 103 : SGD with Momentum

$$\nu_{new} = \eta * \nu_{old} - \alpha * \frac{\partial(Loss)}{\partial(W_{old})}$$

Figure 104 : Momentum Formula

d) Mini Batch Gradient Descent

It is a combination of the concepts of SGD and batch gradient descent. It simply splits the training dataset into small batches and performs an update for each of those batches. This creates a balance between the robustness of stochastic gradient descent and the efficiency of batch gradient descent. it can reduce the variance when the parameters are updated, and the convergence is more stable. It splits the data set in batches in between 50 to 256 examples, chosen at random. [56]

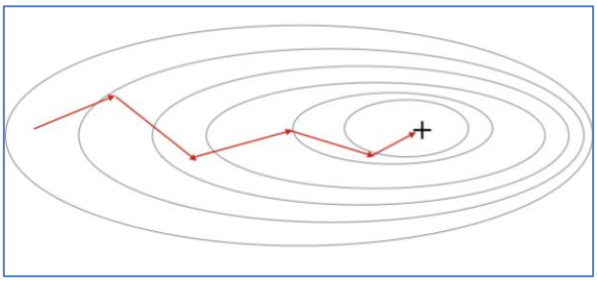

Figure 105 : Mini Batch Gradient Descent

e) Adagrad (Adaptive Gradient Descent)

The intuition behind AdaGrad is can we use different Learning Rates for each and every neuron for each and every hidden layer based on different iterations.

$$W_{new} = W_{old} + \frac{\alpha}{\sqrt{cache_{new}} + \epsilon} * \frac{\partial(Loss)}{\partial(W_{old})}$$

Figure 106 : Adagrad Formula

The benefit of using Adagrad is that it abolishes the need to modify the learning rate manually. It is more reliable than gradient descent algorithms and their variants, and it reaches convergence at a higher speed. [55]

One downside of AdaGrad optimizer is that it decreases the learning rate aggressively and monotonically. There might be a point when the learning rate becomes extremely small. This is because the squared gradients in the denominator keep accumulating, and thus the denominator part keeps on increasing. Due to small learning rates, the model eventually becomes unable to acquire more knowledge, and hence the accuracy of the model is compromised. [56]

f) RMS Prop(Root Mean Square)

RMS-Prop is a special version of Adagrad in which the learning rate is an exponential average of the gradients instead of the cumulative sum of squared gradients. RMS-Prop basically combines momentum with AdaGrad. [55]

$$cache_{new} = \gamma * cache_{old} + (1 - \gamma) * (\frac{\partial(Loss)}{\partial(W_{old})})^2$$

Figure 107 : AdaGrad Formula

The algorithm mainly focuses on accelerating the optimization process by decreasing the number of function evaluations to reach the local minima. The algorithm keeps the moving average of squared gradients for every weight and divides the gradient by the square root of the mean square.

The problem with RMS Prop is that the learning rate has to be defined manually and the suggested value doesn't work for every application.

g) AdaDelta

AdaDelta can be seen as a more robust version of AdaGrad optimizer. It is based upon adaptive learning and is designed to deal with significant drawbacks of AdaGrad and RMS prop optimizer. The main problem with the above two optimizers is that the initial learning rate must be defined manually. One other problem is the decaying learning rate which becomes infinitesimally small at some point. Due to which a certain number of iterations later, the model can no longer learn new knowledge.

To deal with these problems, AdaDelta uses two state variables to store the leaky average of the second moment gradient and a leaky average of the second moment of change of parameters in the model.

h) Adam

The name adam is derived from adaptive moment estimation. this optimizer is one of the most popular and famous gradient descent optimization algorithms. It is a method that computes adaptive learning rates for each parameter. It stores both the decaying average of the past gradients, like momentum and also the decaying average of the past squared gradients, similar to RMS-Prop and Adadelta. Thus, it combines the advantages of both the methods. [55]

$$w_t = w_{t-1} - \frac{\eta}{\sqrt{S_{dw_t} - \varepsilon}} * V_{dw_t}$$

$$b_t = b_{t-1} - \frac{\eta}{\sqrt{S_{db_t} - \varepsilon}} * V_{db_t}$$

Figure 108 : Adam formula

This optimization algorithm is a further extension of stochastic gradient descent to update network weights during training. Unlike maintaining a single learning rate through training in SGD, Adam optimizer updates the learning rate for each network weight individually. [55]

i) How to choose optimizers

- If the data is sparse, use the self-applicable methods, namely Adagrad, Adadelta, RMSprop, Adam.
- RMSprop, Adadelta, Adam has similar effects in many cases.
- Adam just added bias-correction and momentum on the basis of RMSprop,
- As the gradient becomes sparse, Adam will perform better than RMSprop.

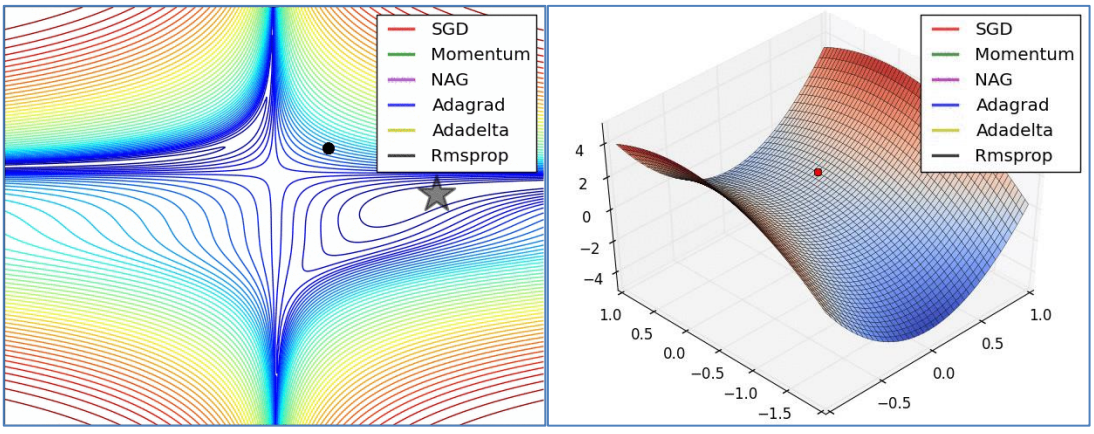

Figure 109 : Differences between optimizers

IX. Flask

Flask is a light-weight framework popularly categorized as a micro framework. Flask comes with some standard functionalities and allows developers to add any number of libraries or plugins for an extension. If you have a simple, innovative use case to be added to an existing application, Flask should be your choice as it offers flexibility. Flask comes with a small set of easy to learn API. [58]

Some features of Flask are:

- Gives you (developer) the full control of decisions to build the application during the development (implementation) stage.
- Comes with a built-in development server and fast debugger
- Coherent and neat API
- Easy and flexible configurations
- RESTful and HTTP request handling
- Integrated Unit testing support

General Conclusion

In this age of Artificial Intelligence, machine learning is a hot topic. Computer vision and predictive analytics are breaking new ground that no one could have foreseen. We are increasingly seeing both in our daily lives, such as facial recognition in smartphones, language translation software, and self-driving cars. What may appear to be science fiction is becoming a reality, and Artificial General Intelligence is only a matter of time before we achieve it.

Our main objective was to learn how machine learning and deep learning works and understand all their aspects. Our second main objective was to participate and help in the pandemic by creating an artificial neural network that detects covid in chest X-Rays.

To achieve our objective, our first step was knowing Artificial Intelligence and how machine leaning became was found. We also knew the differences between them and knew their goals and perspectives.

The second step was to get into machine learning and by knowing the deep learning. In this step we learned how to create a simple neural network. Then we tried to create a convolutional network, we have implemented several popular architectures, while using transfer learning to converge quickly to good results. Inception-ResNet gave the best result, we justify that as The ResNet architecture does not allow the vanishing gradient problem to occur. Last, We have implemented Grad-CAM heat-maps for the last convolutional layer in our model to display the most accurate visual explanation of the the areas being classified by the model.

Although we were able to create a web application that helps patients and doctors in prediction of covid and we aim to develop this platform more so that it can help the patient after finding out that he has covid by guiding him to the nearest covid center next to him and to even send him help if he needs to. To also look after him and helping him finding nearest sources of oxygen.

We could like also to get out from the box of covid and make our platform more generalized so it can detect multi diseases and help patients around the world with all the conditions.

Bibliography:

[1] Andriole, Radiology, Harvard Medical School.

[2] Oracle, "What is artificial intelligence," Oracle, [Online]. Available: https://www.oracle.com/artificial-intelligence/what-is-ai/. [Accessed 28 05 2022].

[3] actualite informatique, "Qu'est-ce que l'Intelligence Artificielle (IA)?," [Online]. Available: https://actualiteinformatique.fr/intelligence-artificielle/qu-est-ce-que-intelligence-artificielle-ia. [Accessed 28 05 2022].

[4] NetApp, "Qu'est-ce que l'intelligence artificielle ?," [Online]. Available: https://www.netapp.com/fr/artificial-intelligence/what-is-artificial-intelligence/. [Accessed 28 05 2022].

[5] S. G. Benjamin and . S. G. Alexander, "Turing Test," Tech Target, [Online]. Available: https://www.techtarget.com/searchenterpriseai/definition/Turing-test. [Accessed 28 05 2022].

[6] T. Davenport and R. Kalakota , "The potential for artificial intelligence in healthcare," *Future Healthc*, vol. 6, no. 2, pp. 94-98, 2019.

[7] synopsys, "What is an Autonomous Car?," synopsys, [Online]. Available: https://www.synopsys.com/automotive/what-is-autonomous-car.html . [Accessed 28 05 2022].

[8] HealthIT, "Clinical Decision Support," 10 04 2018. [Online]. Available: https://www.healthit.gov/topic/safety/clinical-decision-support. [Accessed 08 05 2022].

[9] Centers of Diseases Control and Prevention, "Implementing Clinical Decision Support Systems," Centers of Diseases Control and Prevention, 22 07 2021. [Online]. Available: https://www.cdc.gov/dhdsp/pubs/guides/best-practices/clinical-decision-support.htm. [Accessed 28 05 2022].

[10] T. S. Reed, P. David , C. B. Daniel , C. S. Daniel , N. F. Richard and I. K. Karen , "An overview of clinical decision support systems: benefits, risks, and strategies for success," Nature, 06 02 2020. [Online]. Available: https://www.nature.com/articles/s41746-020-0221-y. [Accessed 28 05 2022].

[11] HealthIT, "What is an electronic health record (EHR)?," HealthIT, 10 09 2019. [Online]. Available: https://www.healthit.gov/faq/what-electronic-health-record-ehr. [Accessed 28 05 2022].

[12] IBM, "What is artificial intelligence in medicine?," IBM, [Online]. Available: https://www.ibm.com/topics/artificial-intelligence-medicine. [Accessed 08 05 2022].

[13] Umer Saeed, Syed Yaseen Shah, Jawad Ahmad, Muhammad Ali Imran, Qammer H. Abbasi and Syed Aziz Shah, "Machine learning empowered COVID-19 patient monitoring using non-contact sensing: An extensive review," *Pharmaceutical Analysis*, vol. 12, no. 2, pp. 193-204, 2022.

[14] M. Miliard, "New AI diagnostic tool knows when to defer to a human," MIT researchers say , 04 08 2020. [Online]. Available: https://www.healthcareitnews.com/news/new-ai-diagnostic-tool-knows-when-defer-human-mit-researchers-say. [Accessed 28 05 2022].

[15] BM Cloud Education, "Machine Learning," IBM, 15 07 2020. [Online]. Available: https://www.ibm.com/cloud/learn/machine-learning. [Accessed 28 05 2022].

[16] S. Tavasoli, "Top 10 Machine Learning Algorithms for Beginners: Supervised, Unsupervised Learning and More," Simple learn, 2022. [Online]. Available: https://www.simplilearn.com/10-algorithms-machine-learning-engineers-need-to-know-article . [Accessed 28 05 2022].

[17] C. Toh and J. P. Brody, , "Applications of Machine Learning in Healthcare" in Smart Manufacturing - When Artificial Intelligence Meets the Internet of Things., London, United Kingdom: IntechOpen, 2021.

[18] Piyush Verma and Stelios Diamantidis , "What is Reinforcement Learning?," 27 04 2021. [Online]. Available: https://www.synopsys.com/ai/what-is-reinforcement-learning.html. [Accessed 28 05 2022].

[19] IBM Cloud Education, "Overfitting," 03 03 2021. [Online]. Available: https://www.ibm.com/cloud/learn/overfitting. [Accessed 28 05 2022].

[20] Corporate Finance Institute, "Overfitting," [Online]. Available: https://corporatefinanceinstitute.com/resources/knowledge/other/overfitting/ . [Accessed 28 05 2022].

[21] IBM Cloud Education, "Underfitting," 2021 03 2021. [Online]. Available: https://www.ibm.com/cloud/learn/underfitting.

[22] UpGrad, "Types of Artificial Intelligence Algorithms You Should Know," 13 11 2019. [Online]. Available: https://www.upgrad.com/blog/types-of-artificial-intelligence-algorithms/ | [accessed 28-May-2022]. [Accessed 28 05 2022].

[23] National Cancer institute, "Introduction to the Nervous System," SEER Training modules, [Online]. Available: https://training.seer.cancer.gov/anatomy/nervous/. [Accessed 30 05 2022].

[24] SAS, "Neural Networks , What they are & whu they matter," sas, [Online]. Available: https://www.sas.com/pl_pl/insights/analytics/neural-networks.html. [Accessed 15 05 2022].

[25] IBM Cloud Education, "Neural Networks," IBM, 17 08 2020. [Online]. Available: https://www.ibm.com/cloud/learn/neural-networks. [Accessed 01 05 2022].

[26] K. V, "Artificial Neural Network, Its inspiration and the Working Mechanism," *Data Science Blogathon.*, 2021.

[27] P. Baheti, "The Essential Guide to Neural Network Architectures," V7 labs, 2022. [Online]. Available: https://www.v7labs.com/blog/neural-network-architectures-guide. [Accessed 15 05 2022].

[28] P. Radhakrishnan, "What are Hyperparameters ? and How to tune the Hyperparameters in a Deep Neural Network?," Towards Data Science, 9 08 2017. [Online]. Available: https://towardsdatascience.com/what-are-hyperparameters-and-how-to-tune-the-hyperparameters-in-a-deep-neural-network-d0604917584a. [Accessed 26 05 2022].

[29] P. Baheti, "12 Types of Neural Network Activation Functions: How to Choose?," v7 Labs, 2022. [Online]. Available: https://www.v7labs.com/blog/neural-networks-activation-functions. [Accessed 19 04 2022].

[30] IBM Cloud Education, "Convolutional Neural Networks," IBM, 20 10 2020. [Online]. Available: https://www.ibm.com/cloud/learn/convolutional-neural-networks. [Accessed 29 04 2022].

[31] Afshine Amidi and Shervine Amidi, "Convolutional Neural Networks cheatsheet," stanford university, [Online]. Available: https://stanford.edu/. [Accessed 15 03 2022].

[32] Praveen Kumar and Nilesh Singh, "Introduction to Deep Learning with Computer Vision—Types of Convolutions & Atrous Convolutions," Towards Data Science, 16 02 2020. [Online]. Available: https://medium.com/hitchhikers-guide-to-deep-learning/10-introduction-to-deep-learning-with-computer-vision-types-of-convolutions-atrous-convolutions-3cf142f77bc0. [Accessed 10 05 2022].

[33] J. Jeong, "The Most Intuitive and Easiest Guide for Convolutional Neural Network," Towards Data Science, 04 01 2019. [Online]. Available: https://towardsdatascience.com/. [Accessed 15 03 2022].

[34] A. Dertat, "Applied Deep Learning - Part 3: Autoencoders," Towards Data Science, 03 10 2017. [Online]. Available: https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798. [Accessed 15 03 2022].

[35] Chattopadhay A, Sarkar A, Howlader P and Bala, "Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks," *winter conference on applications of computer vision* , pp. 839-847, 2018.

[36] P. Mooney, "Chest X-Ray Images (Pneumonia) DataSet," Kaggle, 2018. [Online]. Available: https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia. [Accessed 30 05 2022].

[37] U. o. M. Ethics, "COVID-19 image data collection," GitHub, 2021. [Online]. Available: https://github.com/ieee8023/covid-chestxray-dataset. [Accessed 30 05 2022].

[38] D. D. J. Bell, "COVID-19 DataSet," radiopaedia, 2022. [Online]. Available: https://radiopaedia.org/articles/covid-19-4?lang=us. [Accessed 30 05 2022].

[39] Google, "Colaboratory," Google, [Online]. Available: https://research.google.com/colaboratory/faq.html. [Accessed 30 05 2022].

[40] Python, "Python," [Online]. Available: https://www.python.org/. [Accessed 30 05 2022].

[41] A. Beklemysheva, "Why Use Python for AI and Machine Learning?," Steel kiwi, [Online]. Available: https://steelkiwi.com/blog/python-for-ai-and-machine-learning/. [Accessed 30 05 2022].

[42] NumPy, "Documentation," NumPy, [Online]. Available: https://numpy.org/. [Accessed 30 05 2022].

[43] W3 Schools, "NumPy Introduction," W3 Schools, [Online]. Available: https://www.w3schools.com/python/numpy/numpy_intro.asp. [Accessed 30 05 2022].

[44] A. Bronshtein, "A Quick Introduction to the “Pandas” Python Library," Towards Data Science, 18 04 2017. [Online]. Available: https://towardsdatascience.com/a-quick-introduction-to-the-pandas-python-library-f1b678f34673. [Accessed 30 05 2022].

[45] Pandas, "Pandas," [Online]. Available: https://pandas.pydata.org/about/index.html. [Accessed 30 05 2022].

[46] Matplotlib, "Matplotlib," [Online]. Available: https://matplotlib.org/. [Accessed 30 05 2022].

[47] Seaborn, "Seabor," [Online]. Available: https://seaborn.pydata.org/. [Accessed 30 05 2022].

[48] Google Blog, "How Machine Learning with TensorFlow Enabled Mobile Proof-Of-Purchase at Coca-Cola," 2017. [Online]. Available: https://developers.googleblog.com/2017/09/how-machine-learning-with-tensorflow.html. [Accessed 30 05 2022].

[49] M. Amini, "TensorFlow," GitHub, [Online]. Available: https://github.com/tensorflow/tensorflow. [Accessed 30 05 2022].

[50] "TensorFlow," [Online]. Available: https://www.tensorflow.org/about. [Accessed 30 05 2022].

[51] "Keras," Keras, [Online]. Available: https://keras.io/. [Accessed 30 05 2022].

[52] GitHub Team, "Keras," [Online]. Available: https://github.com/keras-team/keras. [Accessed 30 05 2022].

[53] A. Murphy, "Batch size (machine learning)," Radiopaedia, 2 05 2019. [Online]. Available: https://radiopaedia.org/articles/batch-size-machine-learning. [Accessed 30 05 2022].

[54] Deeplizard, "Batch Size in a Neural Network explained," [Online]. Available: https://deeplizard.com/learn/video/U4WB9p6ODjM. [Accessed 30 05 2022].

[55] "Choosing number of Steps per Epoch," Stack overflow, [Online]. Available: https://stackoverflow.com/questions/49922252/choosing-number-of-steps-per-epoch. [Accessed 30 05 2022].

[56] Knowledge Transfer, "How to set steps_per_epoch,validation_steps and validation_split in Keras's fit method?," androidkt, 07 01 2020. [Online]. Available: https://androidkt.com/how-to-set-steps-per-epoch-validation-steps-and-validation-split-in-kerass-fit-method/. [Accessed 30 05 2022].

[57] Mustafa, "Optimizers in Deep Learning," MLearning.ai, 27 03 2021. [Online]. Available: https://medium.com/mlearning-ai/optimizers-in-deep-learning-7bf81fed78a0. [Accessed 30 05 2022].

[58] A. Gupta, "A Comprehensive Guide on Deep Learning Optimizers," analytics vidhya, 7 10 2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/. [Accessed 30 05 2022].

[59] V. Singh, "Flask vs Django in 2022: Which Framework to Choose?," Hackr.io, 2022. [Online]. Available: https://hackr.io/blog/flask-vs-django. [Accessed 30 05 2022].