Czech University of Life Sciences Prague

Faculty of Economics and Management

Department of Systems Engineering and Informatics
Informatics

Master's Thesis

Pneumonia Diagnostics with Explainable Artificial Intelligence

Lala Yagubova

CZECH UNIVERSITY OF LIFE SCIENCES PRAGUE

Faculty of Economics and Management

DIPLOMA THESIS ASSIGNMENT

Lala Yagubova, BSc

Systems Engineering and
Informatics Informatics

Thesis title

Artificial Intelligence methods for decision making support

Objectives of thesis

The aim of the work is to develop a methodology for testing modern decision algorithms devised in the field of Artificial Intelligence on a Jupyter laptop for a selected decision problem.

Methodology

The student explains the architecture and operation of Jupyter laptops and describes their use for pro- cessing data files. It also gives an overview of modern decision algorithms based on machine learning and describes how to implement and test them on a Jupyter laptop.

The student chooses a decision problem for which a suitable data file can be found on the Internet. He will design algorithms suitable for its solution and propose a methodology for their testing and selection of the most suitable solution. Learning and testing will be performed on a Jupyter laptop.

The resulting Jupyter file (.ipynb file) will contain, in addition to the program implementation of decision algorithms, also the analysis of the data file and the results obtained in the learning and testing process. The results, for which it will be suitable, will be presented in graphical form.

The proposed extent of the thesis

60 pages

Keywords

decision support, machine learning, Jupyter notebook

Recommended information sources

GÉRON, A. *Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems.* Beijing ; Boston ; Farnham ; Sevastopol ; Tokyo: O'Reilly, 2019. ISBN 978-1-492-03264-9.

Expected date of thesis defence

2020/21 SS – FEM

The Diploma Thesis Supervisor

doc. Ing. Arnošt Veselý, CSc.

Supervising department

Department of Information Engineering

Electronic approval: 23. 11. 2021

Ing. Martin Pelikán, Ph.D.

Head of department

Electronic approval: 25. 11. 2021

Ing. Martin Pelikán, Ph.D.

Dean

Declaration

I declare that I have worked on my master's thesis titled "Pneumonia Diagnostics with Explainable Artificial Intelligence" by myself and I have used only the sources mentioned at the end of the thesis. As the author of the master's thesis, I declare that the thesis does not break any copyrights.

In Prague on 23 of November 2022 Lala Yagubova [illegible]

Acknowledgement

I would like to thank doc. Ing. Arnošt Veselý and all other persons, for their advice and support during my work this thesis

Pneumonia Diagnostics with Explainable Artificial Intelligence

Abstract

Pneumonia is the inflammation of the fluid inside the air sacs of the lungs. Pneumonia, which is caused by viral, bacterial, and sometimes fungal diseases that enter the lungs, is extremely infectious. Pneumonia is one of the top causes of mortality among children ages 0 to 4 years old. Pneumonia is one of the leading causes of death in the COVID-19 pandemic that has devastated the globe over the last several years. In situations with a huge number of cases, algorithms based on artificial intelligence may be employed to provide a more precise and timely diagnosis. By analysing chest X-ray images using deep learning algorithms, it is possible to quickly and accurately identify hundreds of patients. As the primary focus of our work, we analysed chest x-ray images and achieved a 97 percent accuracy rate for correct diagnosis. Since the majority of algorithms now in use take the form of black boxes, it is exceedingly difficult to understand how decisions are made. Also, we endeavoured to determine which portions of the globe the AI programme used to make judgments.

Keywords: COVID-19, X-ray, Pneumonia, Convolutional Neural Networks, Grad-Cam.

Diagnostika pneumonie s vysvětlitelnou umělou inteligencí

Abstrakt

Pneumonie je zánět tekutiny uvnitř vzduchových vaků plic. Pneumonie, která je způsobena virovými, bakteriálními a někdy i plísňovými onemocněními, která se dostanou do plic, je extrémně infekční. Pneumonie je jednou z hlavních příčin úmrtnosti dětí ve věku od 0 do 4 let. Pneumonie je jednou z hlavních příčin úmrtí při pandemii COVID-19, která v posledních několika letech zpustošila svět. V situacích s velkým počtem případů mohou být použity algoritmy založené na umělé inteligenci k zajištění přesnější a včasné diagnózy. Analýzou rentgenových snímků hrudníku pomocí algoritmů hlubokého učení je možné rychle a přesně identifikovat stovky pacientů. Hlavním cílem naší práce bylo analyzovat rentgenové snímky hrudníku a dosáhnout 97 procentní přesnosti správné diagnózy. Vzhledem k tomu, že většina algoritmů, které se nyní používají, má podobu černých skříněk, je nesmírně obtížné porozumět tomu, jak jsou přijímána rozhodnutí. Také jsme se snažili určit, které části zeměkoule program AI používal k rozhodování.

Klíčová slova: COVID-19, rentgen, pneumonie, konvoluční neuronové sítě, Grad-Cam.

Table of content

1. **Introduction**

The definition of pneumonia is an infection of the lung parenchyma. Remember that the word "pneumonia" refers to a set of syndromes that are produced by a variety of organisms and display a broad spectrum of symptoms and consequences. As a result of the closure of the airways in the lungs, pneumonia may cause significant breathing problems, and the oxygen level in the blood may be much lower than required [1]. Oxygen is one of the organs' fundamental requirements, and in the event of pneumonia, the organs might be severely harmed due to a lack of oxygen [2]. Pneumonia may cause significant harm to several organs, including the brain and kidneys [3] [4]. Coronaviruses are the primary cause of the COVID-19 sickness presently sweeping the globe, and coronaviruses may cause pneumonia when they infect the lungs. In the past two decades, three major pandemics have been produced by coronaviruses: SARS, Middle East respiratory disease (MERS), and COVID-19 [5] [6]. Without treatment, pneumonia may lead to death. It may produce an unanticipated rise in the number of pneumonia patients during epidemics, making identification and treatment very challenging. This may become quite hazardous, particularly for employees. Because of this, procedures that facilitate rapid comprehension of the situation should be developed, as well as measures to aid both personnel and patients [3]. Based on the patient's medical history, physical exam, and test findings, the healthcare professional will diagnose pneumonia. Sometimes, pneumonia is difficult to diagnose since its symptoms are similar to those of the common cold or influenza [2]. The patient may not recognise the severity of his ailment until it has lasted longer than other conditions. If the healthcare professional suspects pneumonia, one or more of the following tests may be performed [5] [1].

- A chest X-ray is used to detect lung inflammation. Typically, a chest X-ray is utilised to detect pneumonia.
- Blood tests, such as a complete blood count (CBC), indicate whether or not the immune system is combating an illness.
- A pulse oximeter determines the amount of oxygen in the blood. Pneumonia may inhibit the lungs' ability to provide sufficient oxygen to the blood. To monitor levels, a tiny sensor called a pulse oximeter is connected to the finger or ear [3] [1].

Since the invention of computers, we have pondered whether they might be used for education. The impact would be significant if we could find out how to design them to develop automatically with experience. Imagine computers learning from medical records which treatments are most effective for new diseases, homes learning from experience to optimise energy costs based on the specific usage patterns of their residents, and personal software assistants learning from their users' evolving interests to highlight considerations [7]. A solid grasp of how to study computers will open the door for several new computer applications and new degrees of proficiency and personalization. If we knew more about how algorithms for machine learning process information, we could also know more about how people learn (and what stops them) [7]. The area of machine learning is concerned with the creation of computer programmes that grow organically with experience. Many successful machine learning applications have been developed in recent years, including data mining programmes that learn to detect fraudulent credit card transactions, information filtering systems that learn from users' reading preferences, and autonomous vehicles that learn to drive on public roads [8]. Simultaneously, significant advancements have been made in the theory and algorithms that constitute the backbone of this subject [9].

1.1 Purpose of the Thesis

X-rays may be used to diagnose pneumonia, particularly in epidemic settings. The specialist's analysis of the x-ray picture may reveal if the patient has pneumonia. In such scenarios, however, each patient must be treated individually, which, in epidemic settings, may be time-consuming and provide a number of difficulties for the personnel [10]. This research aims to automatically determine if a patient has pneumonia based on a chest X-ray image using artificial intelligence algorithms, or more specifically, deep learning algorithms. Within the purpose of this study, a convolutional neural network (CNN) model was constructed and evaluated using Kaggle-provided chest X-ray data. This time around, around 90 percent of net outcomes were achieved, and the transfer learning mechanism was used to improve this and shorten the training duration. We remained committed to the ResNet50 model since it gave around 97% accuracy. Using this method, we evaluated our model on brand-new data after achieving excellent performance on training, validation, and testing data that the model had never experienced. Nonetheless, at this point, we want to analyse based on which points the model determined, so that our model is no longer a black box. Grad-Cam processing was applied to this data, and as a consequence, significant spots in the images were identified.

1.2 Outline of the Thesis

The structure of this thesis is as follows: The next part provides a comprehensive literature evaluation for pneumonia diagnosis based on respiratory medicine, convolutional neural networks, explainable artificial intelligence, and deep learning. Chapter 3 discusses how we will use the knowledge gleaned from our theoretical study and the data collection we utilised for our investigation. The findings achieved by using the recommended strategies are described in detail in Chapter 4. In addition, Chapter 4 explains the usefulness and applicability of our findings, as well as their weaknesses and areas for improvement. Chapter 4 presents performance metrics for pneumonia detection systems and discusses the outcomes of the performance analysis. Fifth chapter contains the conclusion. You may also review the sources we used when doing research for Chapter 6.

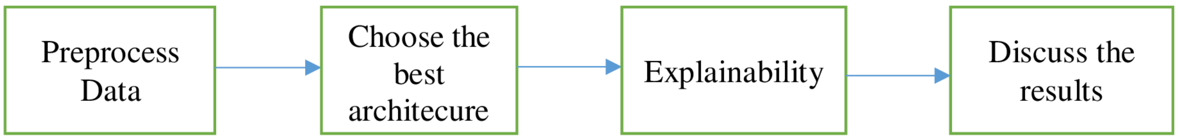
Objectives and Methodology

2.1 Objectives

Based on the strictness of the pneumonia, especially during the pandemics, the method of the diagnostics of the disease will be identified. After the explorations of the possible methods, the best methods will be suggested. Explainability of the used algorithms will be considered as well. The proposed methods will be implemented with the help of the Tensorflow and Keras frameworks.

2.2 Methodology

After the clarification of the current application fields of the Convolutional Neural Networks architectures, the dataset will be prepared for the training. The used dataset is taken from Kaggle platform, called “Pneumonia X-Ray Images” and contains significant number of X-ray images which is beneficial for the training process. Various type of architectures will be tested on the dataset and the best one will be sent to the explainability research. The general mechanism of the research is:



Literature Review

3.1 Anatomy of the lungs

The brain controls the automatic and often subconscious function of respiration. The brain determines the amount of oxygen we need and the rate at which we must breathe in order to supply adequate oxygen to our important organs (brain, heart, kidneys, liver, stomach, and intestines) as well as our muscles and joints (Stephen & Graham, 2011). To breathe well, we must use our lungs, respiratory muscles, and blood system effectively.

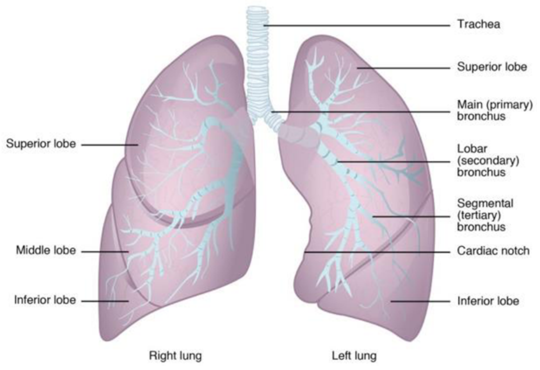


Figure 1 Basic anatomy of the lungs

Humans have two lungs, one on the right of your chest and the other on the left (Anatomy of the Lung, tarixsiz). The right lung is larger than the left due to the location of the heart (located on the left side of the chest). Lungs are covered by two thin layers of tissue which is called the pleura (Pneumonia in adults: diagnosis and management, 2022). The pleura stops the surface of the lungs from rubbing against each other as you breathe in and out. The lungs are protected by the rib cage (Hoffman, tarixsiz).

Inside the lungs is a vast network of airways (tubes) that assist transport oxygen into and carbon dioxide out of the lungs. As these tubes migrate toward the lungs, they divide into smaller tubes.

This tube links your nose and mouth to your lungs through the trachea (windpipe). The trachea is maintained by cartilage rings that assist maintain an open airway. The airway separates into two bronchi around the base of the trachea (Lung Anatomy, tarixsiz).

Each of the two bronchi (right and left) supplies its own lung. Additionally, these airways are maintained by cartilage rings (Anatomy of the Lung, tarixsiz).

These air tubes link the bronchi to the alveoli. They are the first airways without cartilage support, which might increase their risk of collapsing (Anatomy of the Lung, tarixsiz).

Alveoli are tiny air sacs (sometimes resemble clusters of grapes!) found at the end of the bronchioles. Here, oxygen and carbon dioxide are exchanged between the lungs and the blood system (Anatomy of the Lung, tarixsiz) (Lung Anatomy, tarixsiz).

3.2 Radiology of the Chest – Chest X-ray

3.2.1 X-ray

X-rays are an electromagnetic kind of radiation comparable to visible light. Unlike light, however, x-rays have a greater energy level and can penetrate through most things, including the human body. Medical x-rays are used to make pictures of the body's tissues and structures. If X-rays going through the body also pass through the X-ray detector on the other side of the patient, a picture representing the "shadows" cast by items within the body will be created (X-rays, n.d.).

One sort of x-ray detector is photographic film, although there are several other types of detectors used to create digital pictures. Radiographs are the term used to describe the X-ray pictures that come from this procedure (Chest X-ray, n.d.) (X-ray, n.d.).

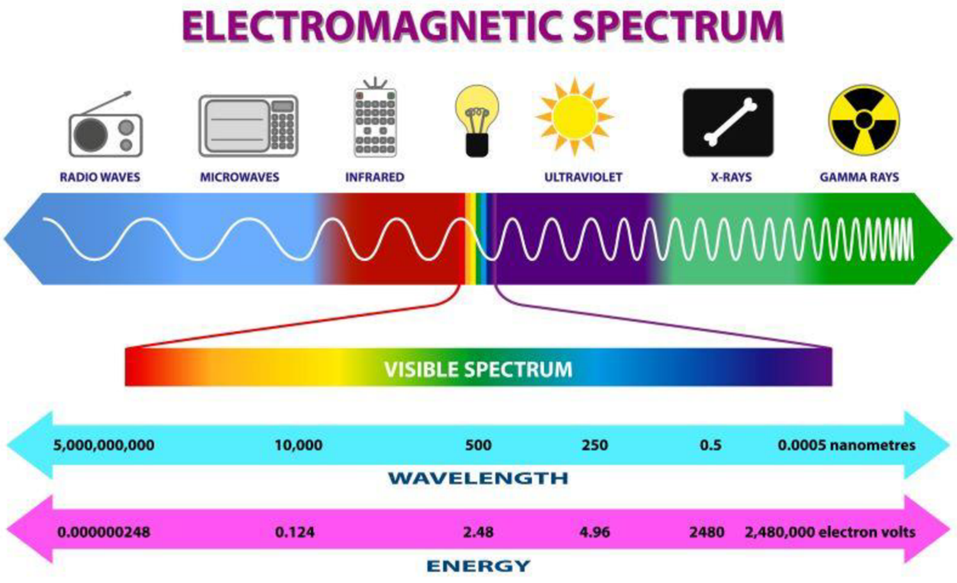


Figure 2 Electromagnetic Spectrum

X-rays are a kind of radiation capable of penetrating the human body. They are imperceptible to the naked eye and cannot be felt. Varying regions of the body absorb X-rays at different rates as they move through the body (X-rays, n.d.). After the X-rays travel through the body, a detector on the other side turns them into a picture. Dense components of your body, such as bone, where X-rays have difficulty passing, show in the picture as bright white patches. Softer regions, such as the heart and lungs, through which X-rays flow more readily seem darker (X-ray, n.d.) (How Do X-rays Work? , n.d.).

3.2.2 Chest X-ray

A chest X-ray utilises a very low amount of ionising radiation to obtain images of the chest's interior. It is used to examine the lungs, heart, and chest wall and aids in the diagnosis of shortness of breath, persistent cough, fever, chest discomfort, and damage. It may also be utilised to aid in the diagnosis and monitoring of a variety of lung disorders, including pneumonia, emphysema, and cancer. A chest X-ray is particularly beneficial for emergency diagnosis and treatment due to its speed and simplicity (Chest X-ray, n.d.). The purpose of a chest X-ray is to check the lungs, heart, and chest wall. Typically, a chest X-ray is the first imaging test performed to detect symptoms such as difficulty breathing, a severe or persistent cough, chest discomfort, or injury fever (Lung Anatomy, n.d.) (How Do X-rays Work? , n.d.). The examination is used to aid in the diagnosis or monitoring of therapy for disorders such as pneumonia, heart failure, and other heart issues such as emphysema, lung cancer, fluid or air accumulation around the lungs, and other medical conditions (Chest X-ray, n.d.).

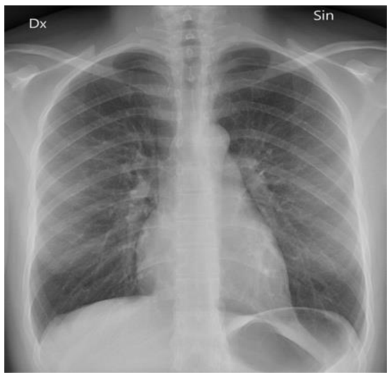


Figure 3 Chest X-ray image

3.3 Pneumonia

Pneumonia is a frequent lung illness characterised by the accumulation of pus and other fluids in the air sacs of the lungs (alveoli). Lung air sacs are structures that facilitate oxygen and carbon dioxide exchange. Pus buildup makes it harder for them to breathe (Jain, Vashisht, Yilmaz, & Bhardwaj., 2022) (Lim, Baudouin, George, & al, 2009). Pneumonia may be caused by a wide variety of microorganisms (germs), such as bacteria, viruses, fungi, and parasites. These germs are discharged into the air when an infected person coughs or sneezes, and breathing in this air causes the illness to develop (Mackenzie, 2016). Consequently, it is an infectious illness. It occurs in persons of all ages and affects millions of people around the globe. The severity of the illness varies based on the organism involved, the individual's age, and their underlying health. Community-acquired, hospital-acquired, and pneumonia in immunocompromised patients are distinct types of pneumonia (immunocompromised individuals) (Mackenzie, 2016). Typically, a bacterial, viral, fungal, or parasitic infection causes pneumonia. In adults, bacteria are the most common cause, but in children and newborns, viruses are the most common cause (Pneumonia in adults:

diagnosis and management, 2022). This illness may also be caused by physical or chemical lung injury. People who smoke, are hospitalised, and have chronic conditions such as asthma, heart disease, cancer, HIV/AIDS, lung disorders, or diabetes are more likely to get pneumonia (Shamila, Hamidreza, Payam, & Esmaeil, 2020). In addition, hospital-acquired pneumonia is prevalent. A diagnosis is based on an individual's extensive personal and medical history, as well as their signs and symptoms. Certain laboratory tests, such as a chest X-ray, are performed to confirm and assess the degree of the illness, as well as to rule out other chest infections (Shamila, Hamidreza, Payam, & Esmaeil, 2020) (Mackenzie, 2016). To determine the exact infectious agent, sputum and blood tests are performed. Pulse oximetry is used to measure the quantity of oxygen in the blood and assess the pulmonary function (Mackenzie, 2016).

3.4 Artificial Intelligence and Machine Learning

3.4.1 Artificial Intelligence

With the proliferation of computers, machines that can operate constantly, make unbiased choices, execute more faultless operations, and accomplish lengthy jobs quickly have been created. Programs operate in the background so that computers may accomplish the required activities, and one of the primary aims of the AI area has been to teach the programme the needed functions by hand. In the 1950s, Alan Turing used the terms "self-learning machines" and "thinking machines" to provide the groundwork for the modern notion of artificial intelligence. AI is a branch of engineering and science used to create computers and computer programmes that can mimic human behaviour and solve issues ranging from basic to complex (Pei, Dagmar, Colin, & Kristinn, 2019).
In other words, the purpose of AI is to make computers and computer-assisted devices think like humans. AI is described as the capacity of computers or machines to comprehend an event or scenario, make sense of it, look for solutions, generalise, and do comparable jobs to humans by leveraging their prior experience. AI is a vast area that attempts to understand and construct intelligent systems (Artificial Intelligence (AI) , n.d.).

3.4.2 Machine Learning

Arthur Samuel, an American computer scientist, initially invented the phrase "machine learning" in 1959. Arthur Samuel argued that the computer is capable of learning without being programmed and acquiring new skills (Samuel, 1959). ML refers to systems that can learn from data via experience, create predictions based on this data, explore the inner workings of algorithms, and apply how they might be enhanced without direct programming (Samuel, 1959). Learning by machines is a subfield in artificial intelligence. The goal of artificial intelligence is to programme computers to think and solve problems like people. The most significant distinction between ML and AI is that ML has the ability to improve itself (Géron, 2019). ML begins with the introduction of training data to algorithms without direct guidance from a computer or computer-assisted device; this is how learning occurs (Badillo, et al.). As a consequence of the training process, test data are used to determine if the algorithms function properly. Because of the testing procedure, a prediction rate is established. If the prediction rate is not at the intended level, the algorithms are retrained by adding more data until the desired result is achieved. With an increase in machine learning data, it is better taught to provide better outcomes. Humans are susceptible to similar scenarios (Géron, 2019). For instance, if the rate of data, expertise, and experience in ML applications improves, the performance and success outcomes of ML increase as a consequence of individuals exercising more (Smola & Vishwanathan).
ML has allowed the identification and reliable diagnosis of several illnesses. The predictive analysis of potent multi-machine algorithms provides improved patient diagnosis and therapy. The health system creates voluminous quantities of health data daily in order to learn about future illness forecasts for a patient and relies on prior health data to treat them (Alpaydın, 2010). With the utilization of health records, progress is also predicted in these areas. In addition, specialized care services are required to diagnose the condition and determine treatment decisions (Badillo, et al.). With the fast advancement of technology, the use of computer programmes has expanded significantly. The growth of technology, software, and computer programmes has given many real-world issues a new perspective. In the field of healthcare, ML algorithms are utilized in the diagnosis of vision loss, in the detection of skin and other cancer diseases, in diseases that are difficult to measure such as Alzheimer's, in the prediction of breast cancer, in the diagnosis of hepatitis disease, in the diagnosis of kidney disease, in the diagnosis of liver disease, and in the detection of sudden

cases such as heart attack. Therefore, ML may boost patient happiness when used to the healthcare industry (Alpaydın, 2010).

ML is used to optimize a process based on historical data or sample sets. In British Columbia, education is conducted via learning from data. On the computer, the required categorization operations are carried out rapidly. At the conclusion of these phases, a model is developed that can make predictions for the future (Géron, 2019). This approach may also be used for control reasons. At its most fundamental level, machine learning may be categorized into three distinct types: supervised learning, unsupervised learning, and reinforcement learning. Modeling the link between the measured qualities of the data and the labels associated with the data is required for supervised learning. Once this pattern has been identified, it may be utilized to identify unknown new data. This is subdivided further into regression and classification tasks. In classification, labels represent distinct categories, but in regression, they represent continuous values. Unsupervised learning is the process of modelling the characteristics of a dataset without the use of labels (Géron, 2019). These models incorporate clustering and dimensionality reduction operations. Clustering methods detect distinct data sets, while dimensionality reduction algorithms seek a condensed representation of the data. Reinforcement learning (RL) is a subset of machine learning that enables an AI-driven system (also referred to as an intermediate) to learn via trial and error with the help of feedback from its actions. This input is either negative or positive, signaled as punishment or reward, with the goal of, of course, optimizing the reward function. RL attempts to make its artificial intelligence as similar to natural intelligence as possible by learning from its failures (Smola & Vishwanathan).

3.4.3 Classification report

In data classification challenges, data may be separated into commercial data, texts, DNAs, and photos. This article emphasizes commercial data as the primary topic of discussion. Additionally, data categorization may be broken down into binary, multiclass, and multilabel classification. The purpose of this research is to examine the efficacy of classifiers using binary and multiclass classification utilizing evaluation criteria. In general, the evaluation metric may be characterized as a tool for measuring the classifier's performance. Various metrics analyze various characteristics of the classifier caused by the classification method (Dalianis).

Literature classifies assessment metrics into three categories: threshold, probability, and rank metric. Each of these sorts of metrics examines the classifier for distinct objectives. In addition, all of these measures are based on the scalar group technique, which presents all performance as a single point number. Thus, it promotes comparison and analysis, despite the fact that it might obscure the nuances of their behavior. In reality, threshold and ranking metrics are the most used metrics employed by academics to evaluate the effectiveness of classifiers. Typically, these indicators may be used in three distinct evaluation applications (Hossin & Sulaiman, 2015).

First, assessment criteria were employed to assess the trained classifier's generalisation capacity. In this instance, the evaluation metric measures and summarises the quality of the trained classifier when evaluated with unseen data. Accuracy, or mistake rate, is one of the most used measures used by academics to assess the generalisation capability of classifiers. By accuracy, the trained classifier is evaluated based on its total accuracy, which is the sum of samples accurately predicted by the trained classifier when tested on unseen data (Hossin & Sulaiman, 2015). Second, the selection of models was guided by assessment criteria. In this instance, the evaluation measure goal is to identify the best classifier among the several kinds of trained classifiers that emphasise the greatest future performance (optimal model) when evaluated with unseen data. Thirdly, evaluation metrics are utilised as a discriminant to differentiate and pick the most suitable solution (best solution) among all solutions created during classification training. For instance, the accuracy measure is used to identify each solution and pick the optimal answer generated by a certain classification method. With unobserved data, only the best solution, presumed to be the optimum model, will be evaluated (Dalianis).

As indicated in Table 1, discrimination assessment of the best (optimal) solution during classification training may be established for binary classification issues based on the confusion matrix. The column of the table reflects the actual class, while the row represents the anticipated class. TP and TN reflect the number of properly categorised positive and negative samples, respectively, based on this confusion matrix. FP and FN represent the number of incorrectly identified negative and positive samples, respectively (Hossin & Sulaiman, 2015).

	Real Positive	Real Negative
Predicted Positive	TP (True Positive)	FP (False Positive)

Predicted Negative	FN (False Negative)	TN (True Negative)

Table 1 Model of confusion matrix

In general, the accuracy metric quantifies the proportion of accurate predictions to the total number of examined samples (Hossin & Sulaiman, 2015):

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision is used to determine how many positive patterns were accurately predicted out of all positive patterns anticipated (Hossin & Sulaiman, 2015):

$$Precision = \frac{TP}{TP + FP}$$

Recall is used to quantify the percentage of positively classed patterns accurately identified (Hossin & Sulaiman, 2015):

$$Recall = \frac{TP}{TP + FN}$$

The F score is the harmonic mean of recall and precision (Hossin & Sulaiman, 2015):

$$F\ Score = 2\ x\ \frac{Precision\ x\ Recall}{Precision + Recall}$$

3.5 Artificial Neural Networks

Artificial Neural Networks (ANNs) are adaptable artificial systems inspired by the brain's functioning processes. They are systems that can alter their internal architecture based on their intended function. They are well suited for addressing nonlinear problems for which it is possible to rebuild fuzzy rules determining the best solution (ZHANG, LIPTON, LI, & SMOLA).

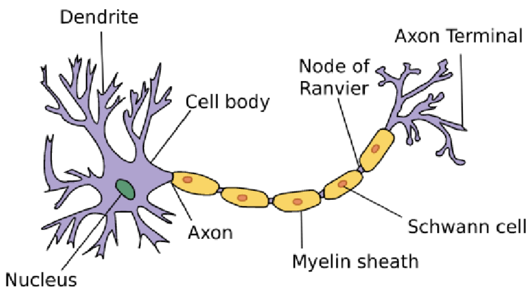


Figure 4 Picture of brain cell (neuron)

Nodes and connections, also known as processing elements (PE), are the basis of an ANN. Each node has its own input, through which it receives communications from other nodes or the environment, and its own output, through which it sends communications to other nodes or the environment. Finally, a function f exists in which each node translates its global input to output. Each link is defined by the stimulation or inhibition intensity between two nodes. Positive numbers suggest connections that are stimulating, whereas negative values indicate connections that are inhibiting. Connections between nodes are subject to alter over time. This dynamic launches a process of learning over the whole ANN. The process by which nodes alter themselves is known as the "Law of Learning." The global dynamics of an ANN are reliant on time (Chollet, Deep Learning with Python). In order for the ANN to alter its own connections, the surrounding environment must have a bigger influence on the ANN. On the ANN, data is the medium of movement. Consequently, the process of learning is one of the major processes that distinguish ANNs, which are termed adaptive processing systems. The learning process is a method for adjusting the connections of an ANN to the data structure that constitutes the environment, and thereby "knowing" the environment and the relationships that define it (Maind & Wankar). Depending on the quality and amount of input data, neurons may be organised in any topological manner (such as one- or two-dimensional layers, three-dimensional blocks, or more-dimensional structures). The most prevalent ANN structure is known as feedforward. Depending on the amount of input variables, a certain number of PEs are typically merged into a single input layer. The transmission of data to one or more hidden layers operating inside the ANN. The output layer offers the last part of this structure, the outcome. Whether the result is a single integer or a binary value, the output layer has just one PE (Hossin & Sulaiman, 2015).

When referring to a neural network, the preferred term is "Artificial Neural Network" (ANN). The architecture of ANNs is modelled after the human brain. Typically, they are comprised of hundreds of small processing units linked through a sophisticated communication network. It is a simplified model of a genuine neuron that fires when each unit or node delivers a new signal or gets a sufficiently strong input signal from other linked nodes. Historically, it has been used to refer to a neural network as a biological network or circuit of neurons. However, in current use, the phrase often refers to ANN. ANN is a mathematical or computational model, a paradigm for information processing inspired by the course of the biological nervous system, such as the brain information system (Buduma, Buduma, & Papa). An ANN comprises of artificial neurons coupled and engineered to replicate the features of organic neurons. These neurons collaborate to address certain

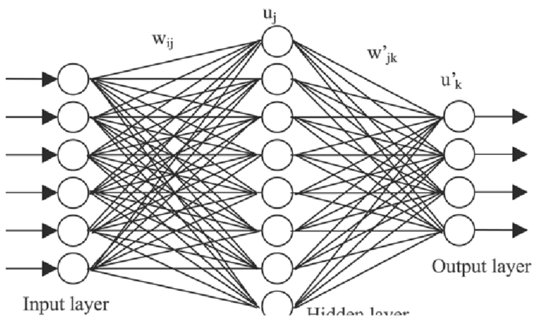


Figure 5 Artificial Neural Network model

issues. ANN is constructed to tackle challenges of artificial intelligence without developing a model of a biological system. using artificial neural networks, voice recognition, image analysis, adaptive control, etc. These applications are developed by a learning process including the adjustment of synaptic connections between neurons, similar to how biological systems learn. The same is true for ANN (Badillo, et al.).

3.5.1 Activation functions

The main task of any activation function in any neural network-based model is to map input to output; where the input value is obtained by calculating the weighted sum of the neuron's input and adding the bias to it (if there is a bias). In other words, the activation function decides whether to fire a neuron for a given input by producing the corresponding output. In ANN architecture, nonlinear activation layers are used after each layer. The nonlinear behavior of these layers allows the ANN model to learn more complex things and manage to map inputs to outputs nonlinearly (Chollet, Deep Learning with Python). The important feature of the activation function is that it is differentiable to allow error backpropagation to train the model. The most commonly used activation functions in deep neural networks are explained below:

The output of the sigmoid activation function is constrained to the interval [0,1] for inputs of real values. The sigmoid function has an S-shaped curve. The mathematical depiction of the sigmoid:

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

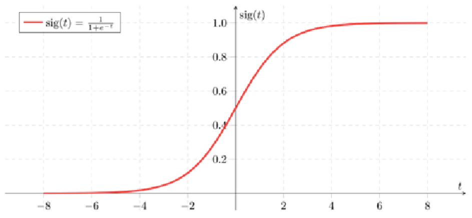


Figure 6 Sigmoid function

The tanh enables function binds input values (real numbers) inside the interval [-1, 1]. tanh's mathematical representation:

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

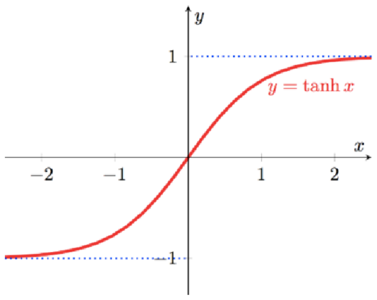


Figure 7 Tanh function

In Artificial Neural Networks, the most used activation function is Rectifier Linear Unit (ReLU). This function converts all input values to positive integers. ReLU is advantageous since it needs less processing overhead compared to other algorithms. ReLU's mathematical representation:

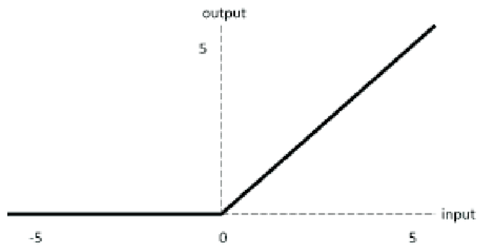


Figure 8 ReLU function

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

However, employing the ReLU activation function might sometimes lead to significant issues. Consider, for instance, a bigger gradient flowing during the error backpropagation method, which, when sent through a ReLU function, may cause the weights to be updated so that the neuron is never triggered again. This problem is sometimes referred to as the "Dying RELU issue." There are several types of ReLU that may be utilised to tackle such difficulties. Several of these are discussed below.

3.5.2 Loss Functions

In Section 3.1, the many kinds of CNN architectural layers are described. We now understand that the last layer of any CNN design (based on classification) is the output layer, where the final classification occurs. In this output layer, we use a loss function to compute the prediction error generated by the CNN model on the training data (Goodfellow, Bengio, & Courville). This prediction error indicates how near the network's predictions are to the actual output, and this error will be optimised throughout the CNN model's learning phase. The error is calculated using two parameters using the loss function. The first parameter represents the predicted output of the CNN model, while the second is the actual result (also known as the label). Different loss functions are used for various sorts of issues. Some of the most frequently used loss functions are detailed in the following sections (Chollet, Deep Learning with Python).

The cross entropy loss, also called the log loss function, is commonly used to measure the performance of a CNN model whose output is probability p ∈ {0, 1}. It is widely used as an alternative to the quadratic error loss function in multiclass classification problems. It uses softmax activations in the output layer to produce the output in a probability distribution, i.e. $(p, y) \in \mathbb{R}^N \times \mathbb{R}^N$, where p is the probability for each output category and y indicates the desired output, and the probability of each output class can be obtained as: $p_i = \frac{e^{a_i}}{\sum_{k=1}^{N} e^{a_k}}$ where N is the number of neurons in the output layer and e^{a_i} denotes each unnormalized output from the previous layer in the network (Goodfellow, Bengio, & Courville). Now finally, the cross entropy loss can be defined as: $H(p, y) = -\sum_i y_i \log(p_i)$ where $i \in [1, N]$

3.6 Foundations of CNN Model

3.6.1 Information about CNNs

In 1959, two neurophysiologists, David Hubel and Torsten Wiesel (Hubel & Wiesel, 1059), conducted experiments and subsequently released a paper titled "Receptive Fields of Single Neurons in the Cat Striatal Cortex" in which they described the layered nature of the neurons in a cat's brain. These layers learn to recognise visual patterns by first extracting local characteristics and then combining those extracted elements to create a higher level representation. This idea eventually became one of the fundamental foundations of Deep Learning. Inspired by the work of Hubel and Wiesel in 1980, Kunihiko Fukushima presented the Neocognitron, a self-organizing neural network that can recognise hierarchical visual patterns via learning; this design was the first theoretical model for CNN. LeCun et al. (LeCun, Bottou, Bengio, & Haner, 1998) provide a significant improvement over the Neocognitron design. In 1989, the MNIST handwritten digits dataset was successfully recognised by a contemporary CNN framework called LeNet-5 (LeCun, Bottou, Bengio, & Haner, 1998). LeNet-5 has been trained using an error back propagation approach and is capable of recognising visual patterns directly from unprocessed input photos without the need for feature engineering. CNN did not perform well in a variety of complicated tasks after the discovery of LeNet-5 owing to a lack of significant training data, a lack of algorithmic innovation, and inadequate computer capacity. Today, however, in the age of Big Data, we have massive labelled datasets, more advanced algorithms, and especially powerful GPU workstations. With these enhancements, Krizhevsky et al. (Krizhevsky, Sutskever, & Hinton) created AlexNet in 2012, which scored high precision in the Large Scale Image Recognition Challenge. The victory of AlexNet made it feasible to create a variety of CNN models and use them in many fields of computer vision and natural language processing.

CNN, also known as ConvNet, is a form of ANN with a feedforward design and remarkable generalisation capabilities relative to other networks with fully connected (FC) layers. It is capable of learning highly abstracted aspects of things, particularly geographical data, and it may define as. A deep CNN model consists of a limited number of processing layers capable of learning many aspects of the input data (e.g., picture) at several levels of abstraction (Géron, 2019). Initializing layers discover and extract high-level characteristics (with less abstraction), while deeper layers discover and extract low-level features (with higher abstraction). By lowering the amount of trainable network parameters, CNN's weight sharing feature allowed the model to avoid overfitting and increase generalisation (Ghosh, Sufian, Sultana, Chakrabarti, & De). In CNN, the classification and feature extraction layers co-learn, making the model's output more structured and reliant on the retrieved features (Chollet, Deep Learning with Python). Other forms of neural networks make it more difficult to implement a big network than convolutional neural networks (Ghosh, Sufian, Sultana, Chakrabarti, & De).

3.6.2 Layers of ConvNet

CNN consists of several constituent parts (known as "layers of architecture"). In this part, we discuss in detail the function of some of these CNN architectural building elements.

3.6.2.1 Convolution operation and Convolutional layer

In electrical engineering, a signal is the basic quantity that represents information. In mathematics, a signal is an information-carrying function. As a matter of fact, every quantity that can be measured in time, space, or higher dimensions may be interpreted as a signal. A signal may be any size and have any shape. Audio, video, and audio are all instances of signals. Convolution is a mathematical operation that combines two signals into a third signal. In other terms, convolution is a mathematical procedure that expresses the connection between input and output signals. Consider the $x_1(t)$ and $x_2(t)$ signals. The convolution of these two signals is therefore defined as follows:

$$y(t) = x_1(t) \otimes x_2(t) = \int_{-\infty}^{\infty} x_1(\tau)x_2(t-\tau)d\tau = \int_{-\infty}^{\infty} x_2(\tau)x_1(t-\tau)d\tau$$

The convolutional layer is the most essential layer in any CNN design. It consists of a collection of convolutional kernels (also known as filters) that combine with the input picture (N-dimensional metrics) to generate an output feature map. A kernel or filter may be described as a grid of discrete values or integers, with each value referred to as the kernel's weight. At the beginning of a CNN model's training procedure, all weights of a kernel are allocated random integers (there are different approaches for initialising the weights). The weights are then modified throughout each training cycle, and the core learns to extract significant information. The convolution formula between an image and a kernel is:

$$S_{ij} = (I \otimes K)_{ij} = \sum_{a=\left\lfloor -\frac{m}{2} \right\rfloor}^{\left\lfloor \frac{m}{2} \right\rfloor} \sum_{b=\left\lfloor -\frac{n}{2} \right\rfloor}^{\left\lfloor \frac{n}{2} \right\rfloor} I_{i-a,j-b} K_{\frac{m}{2}+a,\frac{n}{2}+b}$$

I is the image's matrix representation and contains the input pixel values. K denotes the matrix form of the filter's kernel. It contains two indices, a and b, which respectively represent rows and columns. S_{ij} is the recalculated pixel value at a given position. S is known as the feature map (Goodfellow, Bengio, & Courville).

A step is the length of a horizontal or vertical footfall. In the preceding example, we apply convolution with no padding to the input picture and add one step to the kernel. However, we may use a different step value for the convolution procedure. Increasing the step size of the convolution process leads in a lower-dimensional feature map, which is impressive. Padding is necessary to provide additional weight to the edge size information of the input picture; without padding, the border side attributes would be removed soon. Additionally, the input picture size is increased, resulting in a larger output feature map. The primary benefits of convolution layers include:

- Parameter sharing: There is no particular weight between two neurons of neighbouring layers in CNN, thus instead of dealing with each pixel of the input matrix and all the weights, we may learn a set of weights for all inputs, which dramatically lowers training time and other expenses.
- Sparse connectivity: In a fully connected neural network, each neuron in one layer is linked to each neuron with the next layer. However, in CNN, the number of weights is significantly smaller. As a consequence, the number of needed connections or weights is modest, as is the amount of memory required to store these weights, making it memory efficient. Additionally, the dot(.) operation is less expensive computationally than matrix multiplication (Ghosh, Sufian, Sultana, Chakrabarti, & De).

The size of the feature map is another crucial piece of information about the convolution layer. Using the following formula, the size of the feature map is determined:

$$H_i = \left\lfloor \frac{H_{i-1} - F + P}{S} \right\rfloor \; and \; W_i = \left\lfloor \frac{W_{i-1} - F + P}{S} \right\rfloor$$

H_i is the height of the output feature map, W_i is its width, H_{i-1} is the height of the input image, W_{i-1} is its width, F is the filter size, P is the padding of the convolution operation, and S is calles stride. Stride informs that the convolution operation will shift the filter over the image in steps of one pixel or larger (Ghosh, Sufian, Sultana, Chakrabarti, & De).

3.6.2.2 Pooling layer

Pooling layers are used to subsample feature maps (produced following convolution operations), i.e. to reduce the size of larger feature maps to smaller feature maps. While limiting the feature maps, the most dominating features (or information) in each pool stage are always maintained. Similar to the convolution operation, the pooling operation is conducted by defining the size of the pooled area and the operation step. maximum pooling, minimum pooling, average pooling, etc. Different pooling strategies are used in various pooling levels (Chollet, Deep Learning with Python). Max Pooling is the most popular and widely utilised pooling method. The primary downside of the pooling layer is that it sometimes reduces CNN's overall performance. This is due to the fact that the pooling layer aids the CNN in determining whether or not a certain feature is present in the given input picture, independent of the precise position of that feature. The formula to calculate the dimensions of the feature maps are:

$$H_i = \left\lfloor \frac{H_{i-1} - F}{S} \right\rfloor \; and \; W_i = \left\lfloor \frac{W_{i-1} - F}{S} \right\rfloor$$

H_i is the height of the output feature map, W_i is its width, H_{i-1} is the height of the input image, W_{i-1} is its width, F is the filter size, and S is its stride (Ghosh, Sufian, Sultana, Chakrabarti, & De).

3.6.2.3 Fully Connected (FC) layer

Typically, the last portion (or layers) of any CNN design (used for classification) comprises of completely linked layers, in which each neuron in a layer is connected to each neuron in the layer behind it. The last fully linked layer of the CNN architecture serves as the output layer (classifier). Fully Connected Layers (FC) is a sort of feed-forward artificial neural network (ANN) that follows the MLP concept. The FC layers take input from the final convolution or pooling layer in the form of a collection of metrics (feature maps) (Géron, 2019). These metrics are flattened to produce a vector, which is then given to the FC layers to produce the final CNN network. The following describes the general architecture of convolutional neural networks. The thick layers here are the FC layers that we discussed before. Their position is usually permanent (Ghosh, Sufian, Sultana, Chakrabarti, & De). Convolution and pooling layers are often grouped together. Multiple such groups may exist inside a single ConvNet model. FC levels come last after these groupings, and there may be one or more FC layers last. Various combinations of Convolutional Layer, Pooling Layer, and FC Layers result in many designs, and we will also discuss various architectures (ZHANG, LIPTON, LI, & SMOLA).

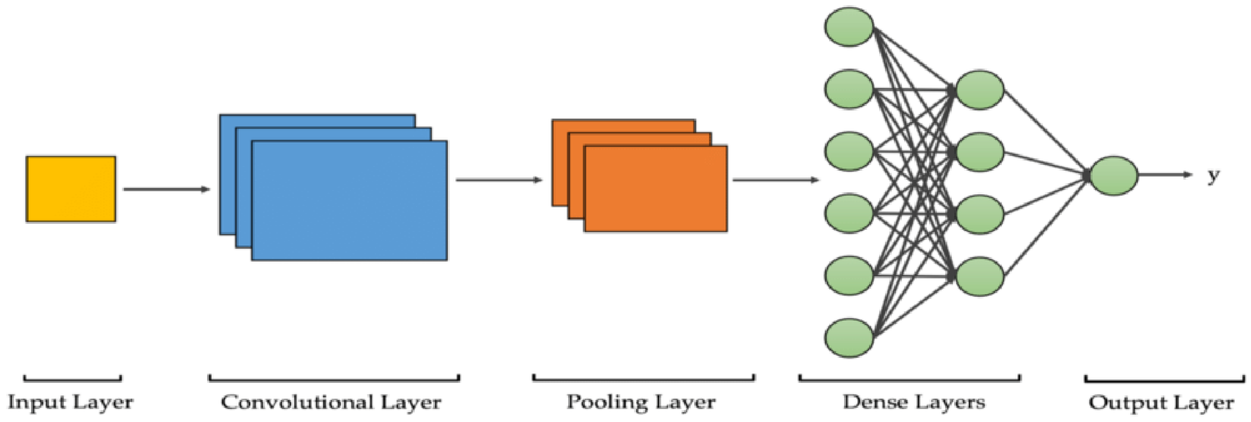


Figure 9 General ConvNet model

3.6.2.4 Dropout layer

"Dropout" refers to the removal of hidden and visible units from a neural network. Simply explained, dropout is the process of disregarding units (i.e. neurons) during the training phase of a randomly chosen group of individual neurons. By "ignore," I mean to disregard these units throughout a certain forward or backward transition (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014).

At each training step, individual nodes are deleted from the network with a probability of 1-p or kept with a probability of p, yielding a reduced network. The node's incoming and outgoing edges are likewise eliminated. Now that we have a basic understanding of dropping out, the question arises: why must we drop out? Why must we physically disable portions of a neural network? "Avoid overfitting" is the solution to these questions. A fully connected layer consumes the majority of the parameters, and as a result, neurons acquire

co-dependency during training, limiting the individual power of each neuron and resulting in overfitting of the training data (Ghosh, Sufian, Sultana, Chakrabarti, & De).
Now that we have a basic understanding of dropout and motivation, let's go into further depth. If you are looking for an overview of dropouts in neural networks, the two sections above should be sufficient. In this part, I will discuss further technical concerns. Regularisation is one approach to prevent overfitting (the probability of over-learning of the model) in machine learning. Regularization makes minor changes to the learning algorithm for better generalization of the model. Regularization decreases overfitting by penalising the loss functionç by adding the penalty values to the loss function. By including this penalty, the model is trained to avoid learning the interdependent feature weights. Those well-versed in Logistic Regression may be acquainted with L1 (Laplacian) and L2 (Gaussian) penalties. Dropout is a method of altering neural networks that reduces the amount neurons learn from one another (Maind & Wankar).

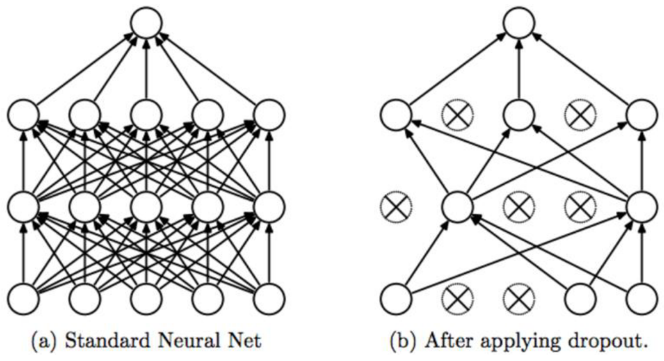


Figure 10 Dropout mechanism

3.6.2.5 Batch Normalization

For each feature, batch normalisation computes the mean and standard deviation for that feature in the mini-batch. The feature is then subtracted from the mean and divided by the mini-batch standard deviation (Ioffe & Szegedy, 2015). The inputs are transformed to have a mean of 0 and a variance of 1 as a result. This is a method for enhancing the speed, performance, and stability of artificial neural networks. By altering and scaling the activations, it is used to normalise the input layer. Batch normalisation enables us to employ much increased learning rates and to be less cautious with initialization. In certain instances, it also serves as an organiser, removing the necessity for Dropout (Ioffe & Szegedy, 2015).
Some benefits of the batch normalization layer:

- Reduces the likelihood of overfitting
- Improves the convergence duration of the neural network
- Avoids the vanishing gradient problem
- Can overcome the bad initialization of the weights problem (Ghosh, Sufian, Sultana, Chakrabarti, & De)

3.7 Famous CNN Architectures

Typical CNN designs consist of several convolution layers (each typically followed by a ReLU layer), a pooling layer, additional convolution layers (+ReLU), another pooling layer, etc. Due to convolutional layers, the picture generally becomes deeper (i.e., contains more feature mappings) as it passes through the network. A standard feed-forward neural network is at the top of the stack. It comprises of numerous fully connected layers (+ReLUs) and the final layer outputs the prediction when it is added (for example, a softmax layer that gives the estimated class probabilities) (Chollet, Deep Learning with Python).

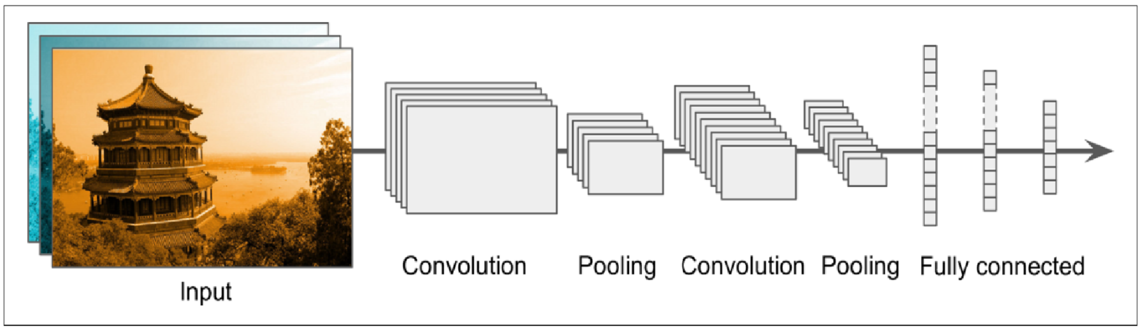


Figure 11 CNN model format

Utilizing enormous convolution kernels is a typical error. For instance, rather of utilising a convolutional layer with 55 cores, stack two layers with 33 cores; it will utilise fewer parameters, need less processing, and perform better in general. The first convolution layer is an exception: it may normally have a big core (e.g., 5 x 5), frequently with two or more steps; this decreases the spatial dimension of the picture without losing too much information, and since the input image typically has just three channels. It will not be extremely expensive (Géron, 2019).

Variants of this fundamental design have been created over the years, resulting in astounding advancements in the industry. Error rates in contests such as the ILSVRC ImageNet challenge are a strong indicator of this development. In only six years, the top five error rates for picture categorization in this competition decreased from 26% to less than 2.3%. The top five mistake rates represent the number of test photos for which none of the system's top five predictions correspond to the right response. There are 1,000 classes, some of which are rather thin, and the graphics are huge (256px in height) (try distinguishing 120 dog breeds). Observing the progression of winning submissions is an effective approach to comprehend how CNNs function (Ghosh, Sufian, Sultana, Chakrabarti, & De).

Here are descriptions of several well-known CNN architectures:

3.7.1 VGG-16 Architecture

Karen Simonyan and Andrew Zisserman (Simonyan & Zisserman, 2014) of the Oxford University Visual Geometry Group launched VGGNet in 2014. It is believed that building neural networks with additional layers would boost classification accuracy, and many VGG networks have been built for this aim. VGG networks have 11, 13, 16, or 19 layers, although the number of convolutional layers varies, and there are three completely linked layers. VGG-16 is a convolutional neural network with thirteen convolutional layers, three fully connected layers, and 138 million parameters (Géron, 2019). In VGG networks, input pictures consist of 224 × 224 RGB images. In the VGG-16 network, blocks with varying numbers of convolutional layers are generated using 3 x 3 filters in a single phase, and there are a maximum of five pooling layers between blocks that are generated using 2 x 2 filters in two stages. After the last convolution layer, there are three completely linked layers with 4096, 4096, and 1000 outputs, followed by the softmax layer.

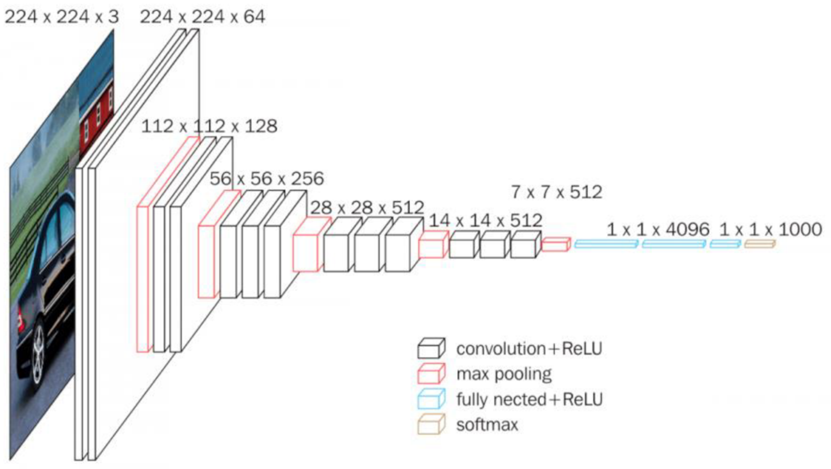


Figure 12 VGG-16 model

3.7.2 ResNet-50 architecture

ResNet is a well-known artificial neural network (ANN) also known as the residual neural network (He, Zhang, Ren, & Sun, 2015). It controls how the pyramidal cell structures of the cerebral cortex are used. Currently, neural networks do this via "bypassing connections" or employing shortcuts to move across layers. Experts use conventional residual neural network topologies with two or three layer hops, batch normalisation, and nonlinearity in between. In certain instances, data scientists also utilise a weight matrix to determine the jump weights. The phrase "highway networks" is used to characterise this circumstance. "Densenets" refers to models with several parallel leaps (Géron, 2019). Non-residual neural networks are usually referred to as flat networks while addressing residual neural networks. The primary purpose for skipping layers is to avoid fading gradients and other obstacles. As gradients propagate back to previous levels, the gradient might diminish drastically due to the repetitive nature of the process. The majority of individuals do this by using activations in the subsequent layer until they understand specific weights from the layers below. During training, these weights are shifted to the upstream layers to emphasise the previously missed layer. In the most fundamental scenario, the weights used to link neighbouring layers have a role (He, Zhang, Ren, & Sun, 2015).

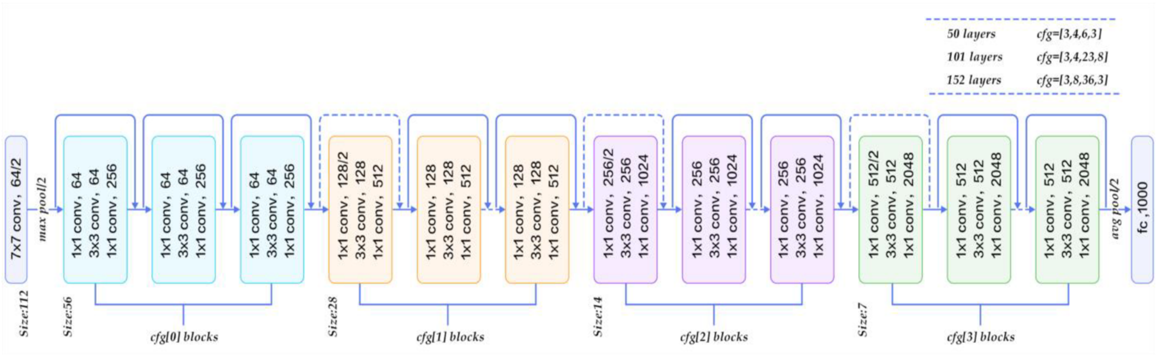

50 layers cfg=[3,4,6,3]
101 layers cfg=[3,4,23,8]
152 layers cfg=[3,8,36,3]
7x7 conv, 64/2
max pool/2
1x1 conv, 64 3x3 conv, 64 1x1 conv, 256
1x1 conv, 64 3x3 conv, 64 1x1 conv, 256
1x1 conv, 64 3x3 conv, 64 1x1 conv, 256
1x1 conv, 128/2 3x3 conv, 128 1x1 conv, 512
1x1 conv, 128 3x3 conv, 128 1x1 conv, 512
1x1 conv, 128 3x3 conv, 128 1x1 conv, 512
1x1 conv, 256/2 3x3 conv, 256 1x1 conv, 1024
1x1 conv, 256 3x3 conv, 256 1x1 conv, 1024
1x1 conv, 256 3x3 conv, 256 1x1 conv, 1024
1x1 conv, 512/2 3x3 conv, 512 1x1 conv, 2048
1x1 conv, 512 3x3 conv, 512 1x1 conv, 2048
1x1 conv, 512 3x3 conv, 512 1x1 conv, 2048
avg pool/2
fc, 1000
Size:112
Size:56
cfg[0] blocks
Size:28
cfg[1] blocks
Size:14
cfg[2] blocks
Size:7
cfg[3] blocks

Figure 13 ResNet-50 structure

However, this is only effective if the majority of interlayers are linear or overlay the nonlinear layer. Otherwise, a separate weight matrix might be beneficial for missing connections. In such situations, it is best to perform research using Highwaynet. Skipping layers simplifies the network during the first training phase by removing complexity. tenfolds the rate of learning while decreasing the influence of disappearing gradients. There are few layers to spread out because there are few layers to begin with. As the network gains knowledge of the feature space, it progressively recovers the expert layers. As the training advances and each layer grows, they approach the manifolds and comprehend everything more quickly. The neural network has a great deal more flexibility to explore the feature space when it is no longer a part of it. This renders it susceptible to perturbations, which lead it to separate from manifolds and necessitates the recovery of further training data (Géron, 2019).

3.7.3 MobileNet architecture

The MobileNet (Howard, et al., 2017) model created by Sandler and Howard (2018) is a fair balance between efficiency and computational expense. MobileNet used separate convolution and depth convolution to dramatically reduce the parameters of the convolutional layer while retaining a high level of classification performance. MobileNet now offers bottleneck blocks for the reuse of feature maps. MobileNet can thus deliver good classification accuracy with much reduced training time. MobileNet's architecture is based on profoundly separable convolutions, a sort of factored convolution that separates a conventional convolution into a depth-based convolution and a 1x1 point convolution. MobileNet's depth information convolution offers a distinct filter for each input channel. The outputs of the deep convolution are then mixed using 1x1 convolution, or even point convolution. At one point, a conventional convolution filter mixes inputs to produce a new set of outputs. It is subdivided into two layers, one for filtering and the other for combination, as a result of the convolution's separation. This factorization cuts computing time and model size considerably (Howard, et al., 2017).

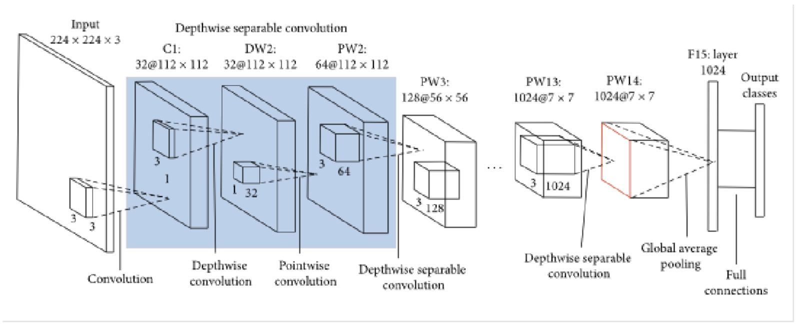


Figure 14 MobileNet architecture

With the exception of its initial layer, which is a complete convolution, the MobileNet structure is composed of strongly separable convolutions. By characterising the network in these fundamental principles, we may rapidly investigate network architectures to locate an appropriate network. Except for the last fully connected layer, which is nonlinear and feeds it to the softmax layer for classification, all model layers are followed by batch normalisation and ReLU nonlinearity. After each convolution layer, batch normalisation and ReLU nonlinearity are introduced to the factored layer using depth-based convolution, 1x1 point convolution, and batch normalisation. In deep convolutions as well as the first layer, step convolution with downsampling is used. Even before the completely linked layer, the spatial resolution is reduced to 1 by a last pooling layer. Convolutions of depth and points are independent layers (Howard, et al., 2017).

3.7.4 Xception architecture

Xception (Chollet, Xception: Deep Learning with Depthwise Separable Convolutions , 2016)is a highly separable convolutional convolutional neural network model. This notion was developed by Google Expert's researchers. Google describes starter modules in convolutional neural networks as an intermediary stage between fundamental convolution and a depth-based separable convolution algorithm (a depth-level convolution followed by a point convolution). A highly separable convolution may be seen as an initial module with the maximum number of towers. As a consequence of this finding, they suggest a novel deep convolutional neural network architecture based on Inception. In instances when depth-separable convolutions alter the initiation modules (Chollet, Xception: Deep Learning with Depthwise Separable Convolutions , 2016).

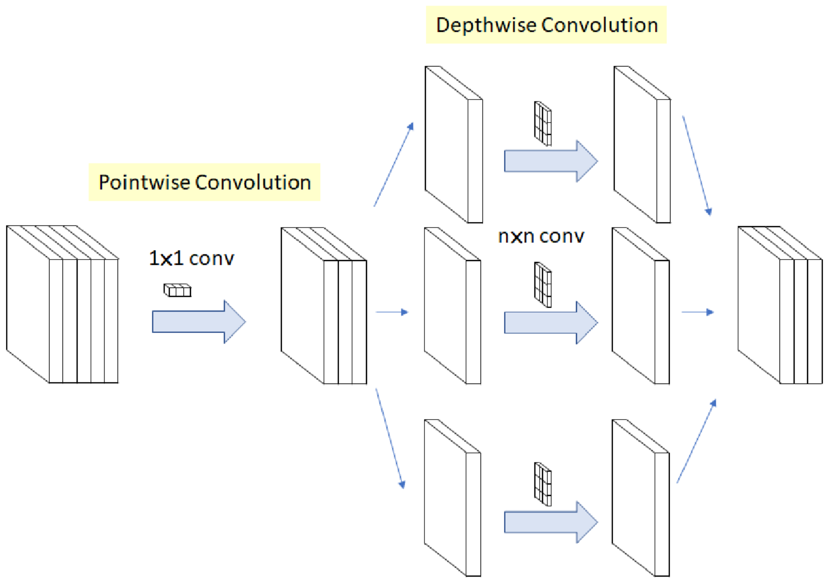


Figure 15 Xception architecture

Xception is a well-designed model based on two essential principles: Depth Separated Convolution and convolution block abbreviations, similar to ResNet. Deeply Separable Convolutions are a sort of convolution that, according to its proponents, computes much quicker than conventional convolutions. In order to lower the expense of such processes, convolutions with a high degree of separation have recently been developed. There are two primary components: depth directional convolution and point directional convolution. Similar to ResNet implementations, Xception employs a structure of Depth Direction Separable Convolution blocks with Maxpooling layers, all coupled via shortcuts. In Xception, a Point Convolution does not precede a Depth Direction Convolution. In contrast, the order is reversed (Chollet, Xception: Deep Learning with Depthwise Separable Convolutions , 2016).

3.8 Transfer Learning

Transfer learning is a study subject in machine learning that focuses on storing and then transferring the information learned while addressing one problem to a different but related problem. For instance, the information acquired when learning to identify automobiles may be employed while attempting to identify trucks (Zhuang, et al., 2019). Even though there are few formal linkages between the two domains, study in this area has some ties to the extensive history of literature on the transfer of learning in the field of psychology (Pei, Dagmar, Colin, & Kristinn, 2019).

Transfer learning is a method for machine learning in which a model learned for one job is reconfigured for a second task that is closely similar. Adapting to a new environment and transferring knowledge include applying what you've learnt in one context to assist you learn in another (Chollet, Deep Learning with Python).

Transfer learning is an optimization that facilitates quick advancement or enhanced performance while modelling the second task. Transferring learning is the practise of enhancing learning in a new activity by transferring information from a similar task previously acquired. However, transfer learning is often utilised in deep learning because it is costly and time-consuming to train deep learning models or because deep learning models are built on very large and challenging datasets (ZHANG, LIPTON, LI, & SMOLA).

In transfer learning, we first train a base network on the base dataset and task, and then redesign the learnt features or transfer them to a second target network that will be trained on a target dataset. This method is effective if the characteristics are generic, particular to the primary task location, and applicable to both the core and target activities (Zhuang, et al., 2019).

3.8 Explainable Computer Vision

According to Wiktionary, the word "explain" implies "to make obvious, clear, or intelligible for others; to eliminate ambiguity; to demonstrate the meaning." In scientific study, a scientific explanation is intended to have at least two components: 1) the thing being explained, and 2) the explanation's substance (Samek, Montavon, Vedaldi, Hansen, & Müller, 2019). AI that is explicable is not a new concept. The first work on explainable AI may be found in publications from forty years ago, in which certain expert systems explain their outcomes by the application of rules. Since the beginning of AI research, scientists have argued that intelligent systems should explain AI outcomes, particularly choices. If a rule-based expert system rejects a credit card payment, it must provide an explanation for its decision. Because human experts develop and construct the rules and information in expert systems, these rules and information are simple for humans to comprehend and interpret (Belle & Papantonis, 2021). A example approach constructed with an explicable structure is a decision tree. However, Explainable AI has emerged as a new research issue in the context of contemporary deep learning. Without wholly new explanation methods, the output of modern Deep Neural Networks (DNNs) cannot be explained by the neural network itself, an external descriptive component, or even the system's creator. CNN, RNN, and LSTM are examples of distinct DNN architectures tailored for different problem classes and input data. All of them should be regarded as black boxes whose inferential processes are unknown to observers and cannot be analysed by humans (Samek, Montavon, Vedaldi, Hansen, & Müller, 2019).
As Deep Learning (DL) models improve, it becomes more challenging to comprehend the information they acquire. Today's CNNs enable us to extract characteristics from the data we input. Nevertheless, DL models are regarded as black boxes since we do not know what they learn from the data (Belle & Papantonis, 2021).

3.9.1 Gradient-weighted Class Activation Mapping - Grad-CAM

Grad-CAM is simply a visualisation of a class label as a heatmap (visualization of data or pixels in the matrix with the help of colors) (Selvaraju, et al., 2016). Our purpose in using this method is to understand the decision making logic of the model. We will give detailed information about the mathematical dimension of the Grad-CAM mechanism below. The values we get with the help of Grad-CAM tell us the more important pixels, and the heatmap logic helps us to see the these pixels. (Bichakchi, 2021)

Step 1: Gradient Calculation:

The feature map is the output of one filter applied to the previous layer. In the first phase, the gradient is computed. The gradient value to be determined will vary based on the picture provided. The reason is the provided picture will influence both the activation map and the model's output. (Bichakchi, 2021) $\nabla F^i = \frac{\partial y^{logit}}{\partial F^i}$, where y^{logit} denotes the logit (class scores before final activation) and F^i denotes the feature map (Selvaraju, et al., 2016).

Step 2: Rating Activation Maps

In the preceding formula, a gradient formula was derived only one activation map. In general, we want to analyse the evolution of all activation maps. It is pretty simple to evaluate them all simultaneously. Since we do not know which of these activation maps are significant, therefore we must rate them (based on model prediction). We can do this by averaging the gradient's components. $\alpha_k^{logit} = \frac{1}{H\,x\,W}\sum_i \sum_j \nabla F^k$, where H and W denotes height and weight respectively (Selvaraju, et al., 2016).

Step 3: Grad-CAM Heatmap

The Grad-CAM heatmap can be obtained with the following formula:

$$L_{Grad-CAM}^{logit} = ReLU(\sum_k \alpha_k^{logit} F^k)$$

Explainability describes the model's prediction for a given input. The results attained as a consequence of this explainability may not be comprehensible to the average person. Interpretability is comparable yet distinct. Being interpretable implies that it produces predictions in a manner that people can comprehend from the beginning. (Bichakchi, 2021)

Practical Part

4.1 Dataset

A good quality dataset is needed for successful diagnosis of pneumonia cases. What we mean by a quality data set is that the number of elements in the data set is as large as necessary, the images are high-resolution and that work for the problem. The data set we used was obtained from the kaggle platform, prepared by Daniel Kermany et. al. (Kermany, et al., 2018) and re-edited by Paulo Breviglieri (BREVIGLIER, n.d.). In the version we use, there are two datasets: 4192 training (1082 healthy and 3110 sick individuals) and 1040 validation (267 healthy and 773 sick individuals).

Metadata is available for each sample, so therefore it is supervised learning task and makes our task easier. Chest X-ray images of patients with pneumonia can be seen in Figure 16

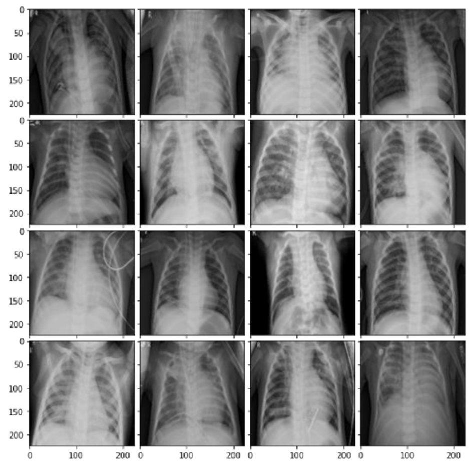


Figure 16 Chest X-ray photos of sick individuals

We have defined pneumonia in the previous sections. By closely viewing the images, it will become apparent that each patient's lungs have significant white layers. Experts detect pneumonia by analysing these images. Let us now study it. Consider the chest X-rays of healthy individuals.

We previously seen images of lungs without pneumonia. The heart, diaphragm, and lungs are plainly apparent if we look closely. The X-ray photos of healthy individuals can be observed in Figure 17.

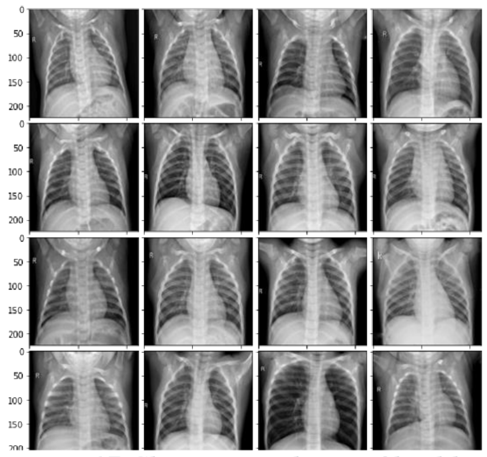


Figure 17 Chest X-ray photos of healthy individuals

During Supervised Deep Learning models, 3 types of data sets are generally used. The names of these data sets are training, validation and test data. Our main purpose in doing this is to reveal whether there is an overfitting. The main purpose of the train data is to run the training process. Validation data is to monitor the performance of our model during the training process. The main purpose of the test data is to make the final evaluation of our model. In other words, the test data determines which model is better. We mentioned earlier that we have two data sets. We will use the validation dataset for testing purposes. Training data will be be divided into 70% and 30% parts, for training and validation purposes respectively.

4.2 Libraries , Frameworks and Preparatory Coding

Python is a programming language that is often used to construct websites and applications, automate operations, and do data analysis. Python is a general-purpose programming language, which means it can be used to create a range of different programmes and is not specialised for a specific issue. This language's flexibility and friendliness to beginners have made it one of the most popular programming languages in use today. Python was the programming language utilised to implement the research project. Using notebooks for image processing tasks has several benefits. Therefore, we used Jupyter Notebook. We used Google Colab since it is user-friendly. We saved the data in Google Drive and used the GPU environment of Google Colab.

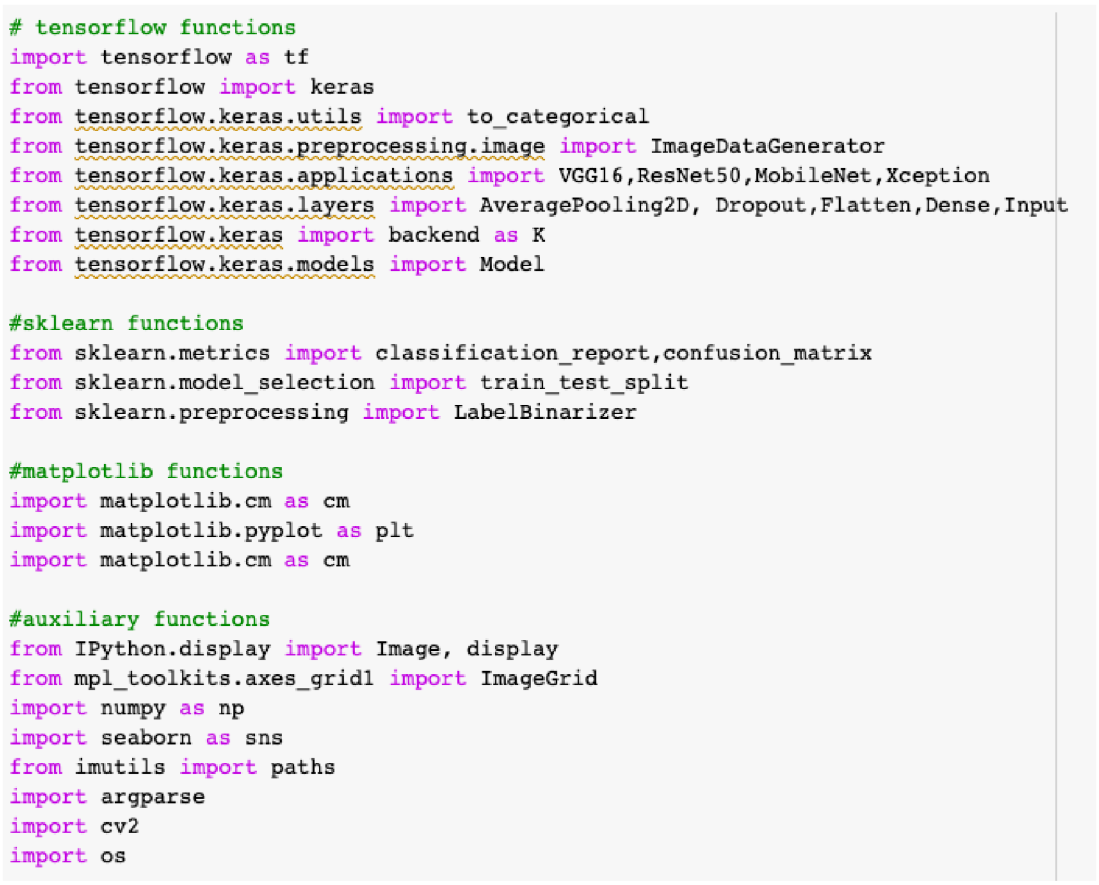
Tensorflow and Keras were used to develop deep learning models. TensorFlow enables developers to design data flow graphs, which are structures that represent how data flows in a graph or a sequence of render nodes. Each node in the graph represents a mathematical process, whereas each edge is a multidimensional data array or tensor. TensorFlow applications can execute on the majority of suitable targets, including a local system, a cloud cluster, iOS and Android smartphones, CPUs and GPUs. Using Google's cloud, you may run TensorFlow on Google's unique TensorFlow Processing Unit (TPU) hardware for additional acceleration. However, the models produced by TensorFlow can be distributed to the majority of devices, where they will be utilised to make predictions. Keras is one of the most prominent high-level APIs for neural networks. It supports several backend neural network computation engines and is implemented in Python. Given that the TensorFlow team has picked Keras as the high-level API for the next TensorFlow 2.0 release, Keras seems to be a winner, however this is not always the case. In this post, we will investigate the ideas and uses of Keras in order to comprehend why Keras is superior than low-level deep learning APIs. Even in TensorFlow 1.12, which is used in the official Getting Started with TensorFlow tutorial, TensorFlow employs the tf.keras-built high-level Keras API. In contrast, the TensorFlow Core API necessitates dealing with TensorFlow computational graphs, tensors, operations, and sessions, some of which might be challenging to comprehend for TensorFlow beginners. Using the low-level TensorFlow Core API provides various benefits, especially while debugging, but you may combine high-level and low-level TensorFlow APIs as required.

We utilised methods from the sklearn package to evaluate the models' performance. Python's most helpful and robust package for machine learning is Scikit-learn (Sklearn). It provides a consistent Python interface to a collection of valuable machine learning and statistical modelling techniques, including classification, regression, clustering, and dimensionality reduction.

The OpenCV library was used to analyse and prepare photos for modelling. crucial in contemporary systems It can analyse photos and movies to recognise a person's items, faces, and handwriting using this technology. When coupled with other libraries like as Python and NumPy, it is capable of analysing the OpenCV array structure. We employ vector space to determine the visual pattern and its many components, then use these components to perform mathematical operations.

Using the matplotlib software, the graphic pictures were created. Matplotlib is an extensive Python toolkit for producing static, animated, and interactive visualisations. Matplotlib makes the simple simple and the difficult attainable.

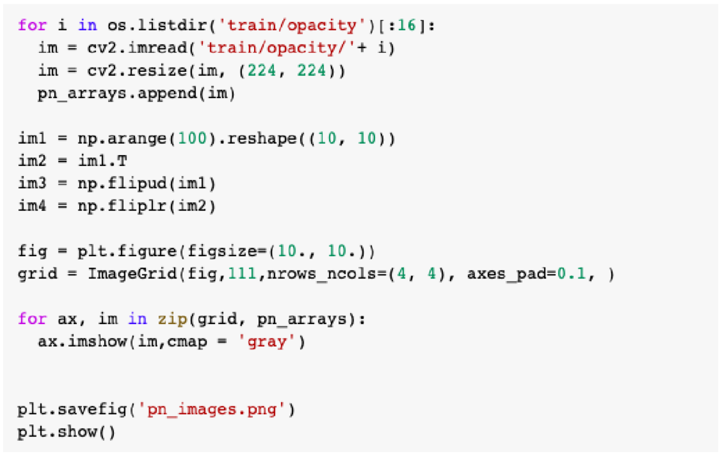
The general list of used libraries is given as follows.



```python
# tensorflow functions
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import VGG16,ResNet50,MobileNet,Xception
from tensorflow.keras.layers import AveragePooling2D, Dropout,Flatten,Dense,Input
from tensorflow.keras import backend as K
from tensorflow.keras.models import Model

#sklearn functions
from sklearn.metrics import classification_report,confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelBinarizer

#matplotlib functions
import matplotlib.cm as cm
import matplotlib.pyplot as plt
import matplotlib.cm as cm

#auxiliary functions
from IPython.display import Image, display
from mpl_toolkits.axes_grid1 import ImageGrid
import numpy as np
import seaborn as sns
from imutils import paths
import argparse
import cv2
import os
```

Figure 18 Screenshot of the used libraries

Each picture has distinctive proportions. We must convert them all to the same dimensions (224 x 224). To provide an overview of the photographs, it may be useful to arrange them in a grid. The following code was used to display photos in a grid format.

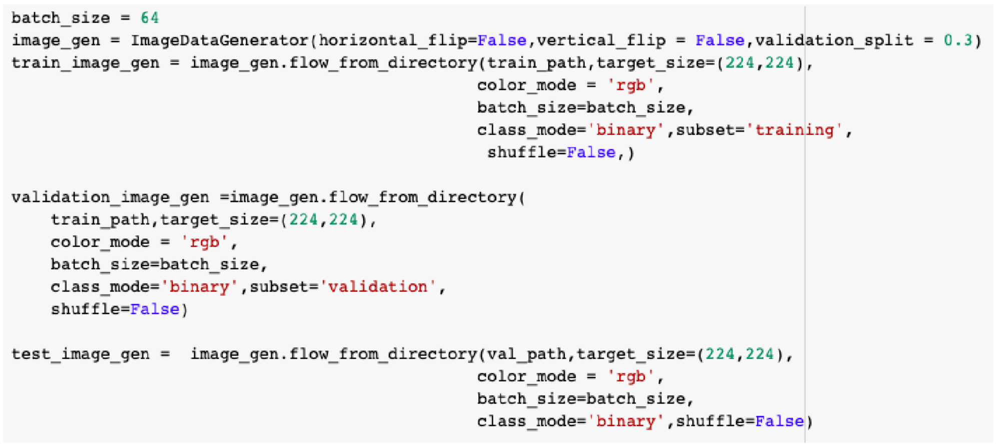


```python
for i in os.listdir('train/opacity')[:16]:
  im = cv2.imread('train/opacity/'+ i)
  im = cv2.resize(im, (224, 224))
  pn_arrays.append(im)

im1 = np.arange(100).reshape((10, 10))
im2 = im1.T
im3 = np.flipud(im1)
im4 = np.fliplr(im2)

fig = plt.figure(figsize=(10., 10.))
grid = ImageGrid(fig,111,nrows_ncols=(4, 4), axes_pad=0.1, )

for ax, im in zip(grid, pn_arrays):
  ax.imshow(im,cmap = 'gray')


plt.savefig('pn_images.png')
plt.show()
```

Figure 19 Image grid creator

4.3 Implementation of the Classification and Transfer Learning

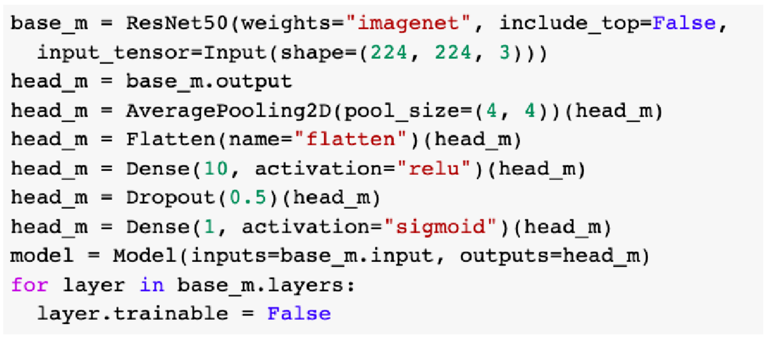
Before modelling, we must first thoroughly prepare the data for fitting. To do this, we must readfirst split the data. Additionally, the labels for the data must be created. The relevant line of code is displayed below. The code snippet below also scales all images to the same size and fully prepares them for modelling.



```
batch_size = 64
image_gen = ImageDataGenerator(horizontal_flip=False,vertical_flip = False,validation_split = 0.3)
train_image_gen = image_gen.flow_from_directory(train_path,target_size=(224,224),
                                               color_mode = 'rgb',
                                               batch_size=batch_size,
                                               class_mode='binary',subset='training',
                                                shuffle=False,)

validation_image_gen =image_gen.flow_from_directory(
    train_path,target_size=(224,224),
    color_mode = 'rgb',
    batch_size=batch_size,
    class_mode='binary',subset='validation',
    shuffle=False)

test_image_gen =  image_gen.flow_from_directory(val_path,target_size=(224,224),
                                               color_mode = 'rgb',
                                               batch_size=batch_size,
                                               class_mode='binary',shuffle=False)
```

Figure 20 Data preparation

In the preceding paragraphs, we discussed transfer learning. When we develop our models, we utilise ready-made models. Their general parameters may be valuable to us, therefore we use transfer learning on the model's front portions and solely train the model's back portions. This helps us save time and get excellent outcomes. The architectures we utilise are VGG16, ResNet50, MobileNet, and Xception. The ResNet50 model's code will serve as an example. The rationale used by the others is same.



```
base_m = ResNet50(weights="imagenet", include_top=False,
  input_tensor=Input(shape=(224, 224, 3)))
head_m = base_m.output
head_m = AveragePooling2D(pool_size=(4, 4))(head_m)
head_m = Flatten(name="flatten")(head_m)
head_m = Dense(10, activation="relu")(head_m)
head_m = Dropout(0.5)(head_m)
head_m = Dense(1, activation="sigmoid")(head_m)
model = Model(inputs=base_m.input, outputs=head_m)
for layer in base_m.layers:
  layer.trainable = False
```

Figure 21 Model implementation

After our model has been installed, it must be compiled and fitted. Before commencing the training process, we establish our loss functions, optimization strategies, batch size, and other parameters. Our loss function is binary cross entropy since our last layer has one neuron (binary classification). We may use accuracy as a metric since the but f score will be taken into account, since we have slight class imbalance. The piece of code used to implement these operations is given below.

```
model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
history = model.fit(
    train_image_gen,
    validation_data = validation_image_gen,batch_size = 64,shuffle = True,
    epochs = 100)
```

Figure 22 Model compiling and fitting

After the modelling is complete, evaluation is a vital step for us. It is equally vital to us that our model provides accurate findings for data sets it has never seen. This measurement may be accomplished using the following piece of code.

```
preds = model.predict(test_image_gen)
predictions = preds > 0.5
print(classification_report(test_image_gen.classes, predictions))
```

Figure 23 Prediction

Determining precision, recall, and accuracy figures is one of the most essential tasks. The classification report function displays the results per class, however we want the overall results as well. Therefore, we also used several score methods from the sklearn package. In addition, we will utilise the confusion matrix approach to analyse the model's outputs more thoroughly. The seaborn library may assist with the confusion matrix. This code may be used to plot these statistics and the confusion matrix.

```
cm = confusion_matrix(test_image_gen.classes, predictions)
precision = precision_score(test_image_gen.classes,predictions)
recall = recall_score(test_image_gen.classes, predictions)
f1 = f1_score(test_image_gen.classes, predictions)
print("Precision: {:.4f}".format(precision))
print("Recall: {:.4f}".format(recall))
print("F1 Score: {:.4f}".format(f1))
sns.heatmap(cm, annot=True);
```

Figure 24 Confusion matrix

Next, we need a graph that illustrates the evolution of our margin of error and precision at each epoch period. The code below may be used to visualise these graphs.

```
N = 100
plt.figure(figsize = (10,10))
plt.plot(np.arange(0, N), history.history["loss"], label="train_loss")
plt.plot(np.arange(0, N), history.history["val_loss"], label="val_loss")
plt.plot(np.arange(0, N), history.history["accuracy"], label="train_accuracy")
plt.plot(np.arange(0, N), history.history["val_accuracy"], label="val_accuracy")
# plt.ylim(0,1.25)
plt.title("MobileNet Error and Accuracy Rate")
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")
plt.legend(loc="lower left")
plt.show()
```

Figure 25 Model history sketcher

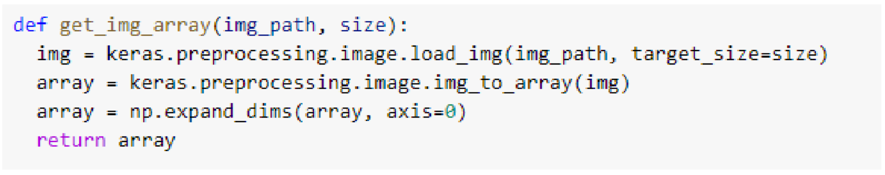
This piece of code allows us to monitor the evolution of our model's accuracy and error margin over time, both in the training and extra validation versions (where TensorFlow chooses the training time).

The value 100 in the above code section represents our epoch number. In other words, the number of times the whole data set will be iterated during model training.

4.4 Implementation of the Grad-CAM

In addition to using the Grad-CAM method, this is a key objective of the project. This approach will reveal the factors that our algorithm considered most heavily while classifying each picture. This will provide us insight into how the algorithm makes decisions.

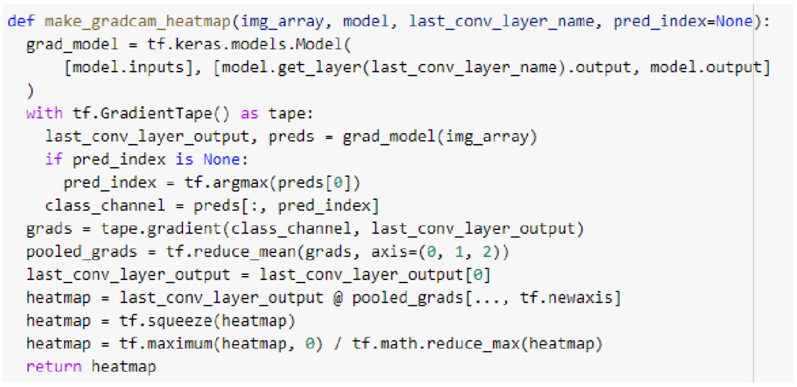
First, we must create a function that reads the picture from the specified location and returns it as an array. This method will resize our picture to the specified dimensions and convert it to an array for usage by our model.



```
def get_img_array(img_path, size):
  img = keras.preprocessing.image.load_img(img_path, target_size=size)
  array = keras.preprocessing.image.img_to_array(img)
  array = np.expand_dims(array, axis=0)
  return array
```

Figure 26 Image array creator

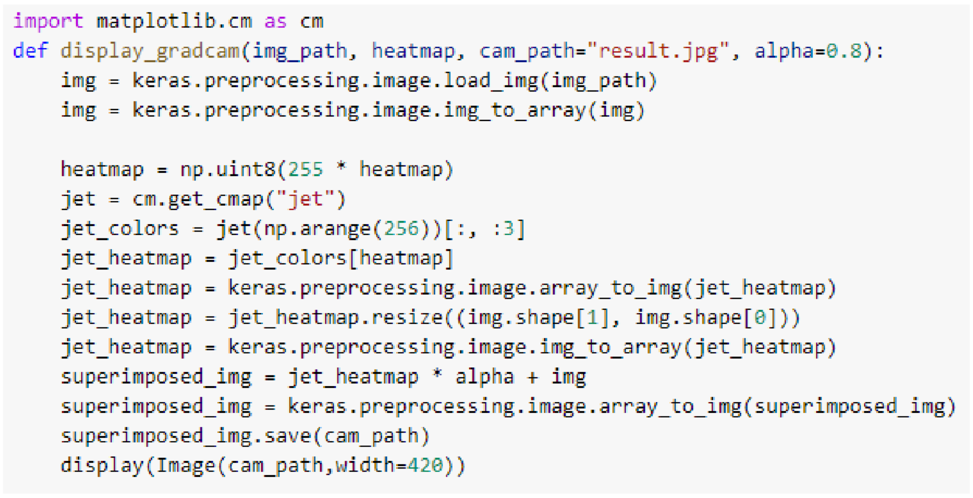
The last convolution layer utilised before the model's input, output, and prediction is necessary for the Grad-CAM approach. These three essential components are collected and a new Grad-CAM model is developed. This model is produced, after which its class cores are determined. The gradient values and heatmap are then produced. The code sample below illustrates these operations in practise.



```
def make_gradcam_heatmap(img_array, model, last_conv_layer_name, pred_index=None):
  grad_model = tf.keras.models.Model(
      [model.inputs], [model.get_layer(last_conv_layer_name).output, model.output]
  )
  with tf.GradientTape() as tape:
    last_conv_layer_output, preds = grad_model(img_array)
    if pred_index is None:
      pred_index = tf.argmax(preds[0])
    class_channel = preds[:, pred_index]
  grads = tape.gradient(class_channel, last_conv_layer_output)
  pooled_grads = tf.reduce_mean(grads, axis=(0, 1, 2))
  last_conv_layer_output = last_conv_layer_output[0]
  heatmap = last_conv_layer_output @ pooled_grads[..., tf.newaxis]
  heatmap = tf.squeeze(heatmap)
  heatmap = tf.maximum(heatmap, 0) / tf.math.reduce_max(heatmap)
  return heatmap
```

Figure 27 Heatmap

This method returns a heatmap as its result. The heatmap will then be applied to the original picture to get the desired results. After obtaining the most recent heatmaps, we must finally apply them to the photos. To create this application, we must adjust the pixel dimensions of the heatmap to match those of the image. We must next choose the credit unit. This colour has been selected as the contour jet. This colour map offers a colour pallet ranging from deep blue to deep red, and colours such as red, yellow, and orange represent sensitive areas. Finally, this heatmap is mixed with our picture, and the desired opacity value is achieved. The applicable code is as follows:



```
import matplotlib.cm as cm
def display_gradcam(img_path, heatmap, cam_path="result.jpg", alpha=0.8):
    img = keras.preprocessing.image.load_img(img_path)
    img = keras.preprocessing.image.img_to_array(img)

    heatmap = np.uint8(255 * heatmap)
    jet = cm.get_cmap("jet")
    jet_colors = jet(np.arange(256))[:, :3]
    jet_heatmap = jet_colors[heatmap]
    jet_heatmap = keras.preprocessing.image.array_to_img(jet_heatmap)
    jet_heatmap = jet_heatmap.resize((img.shape[1], img.shape[0]))
    jet_heatmap = keras.preprocessing.image.img_to_array(jet_heatmap)
    superimposed_img = jet_heatmap * alpha + img
    superimposed_img = keras.preprocessing.image.array_to_img(superimposed_img)
    superimposed_img.save(cam_path)
    display(Image(cam_path,width=420))
```

Figure 28 Grad-CAM display function

The last piece of Grad-CAM code will also output the findings and save them on our PC. This method is applied to a single picture, but the concept is the same, and the results may be displayed in grid format. This function's output provides a straightforward indication of the decision method.

Results and Discussions

5.1 Classification Results

The project has two essential components: classification and explainability. In the classification step, we train and choose the optimal model. After selecting the optimal model, the Grad-CAM approach is used to comprehend the model's operational logic.
The first step is to choose the most appropriate model for the project. Transfer Learning is implemented using four distinct models. VGG16, ResNet50, MobileNet, and Xception are these models. We will use the evaluation metrics available for each model, and then examine the trace timings. Below is a table including the evaluation metrics for each architecture. These measurements include accuracy, precision, recall, and the F1 score.

Table 2 General classification report

	Accuracy	Precision	Recall	F1-Score
VGG16	0.9490	0.9523	0.9806	0.9662
ResNet-50	0.9615	0.9778	0.9702	0.9740
MobileNet	0.9423	0.9575	0.9651	0.9613
Xception	0.9307	0.9431	0.9651	0.9540

This chart is insufficient to determine the optimal model. Because we have not yet encountered the confusion matrix. Likewise, the confusion matrix may reveal a great deal. We have a limited number of versions, and the technological equipment was assembled poorly.
Examining the various confusion matrices for each design. Confusion matrices serve a crucial part in the analysis of designs' findings.

Confusion matrix for the VGG-16 model is the following:

Table 3 Confusion matrix of VGG-16 model

	0	1
0	TP = 229	FP = 38
1	FN = 15	TN = 758

Confusion matrix for the ResNet-50 model is quite better than the previous one.

Table 4 ResNet-50 model confusion matrix

	0	1
0	TP = 250	FP = 17
1	FN = 23	TN = 750

Let's see the effectiveness of the MobileNet model:

Table 5 MobileNet model confusion matrix

	0	1
0	TP = 230	FP = 33
1	FN = 27	TN = 750

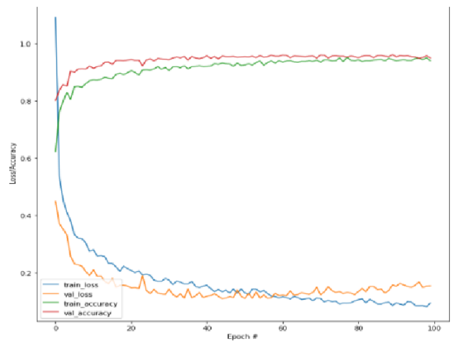
Xception model shows slightly different results compared to the other models:

Table 6 Xception model confusion matrix

	0	1
0	TP = 219	FP = 45
1	FN = 27	TN = 749

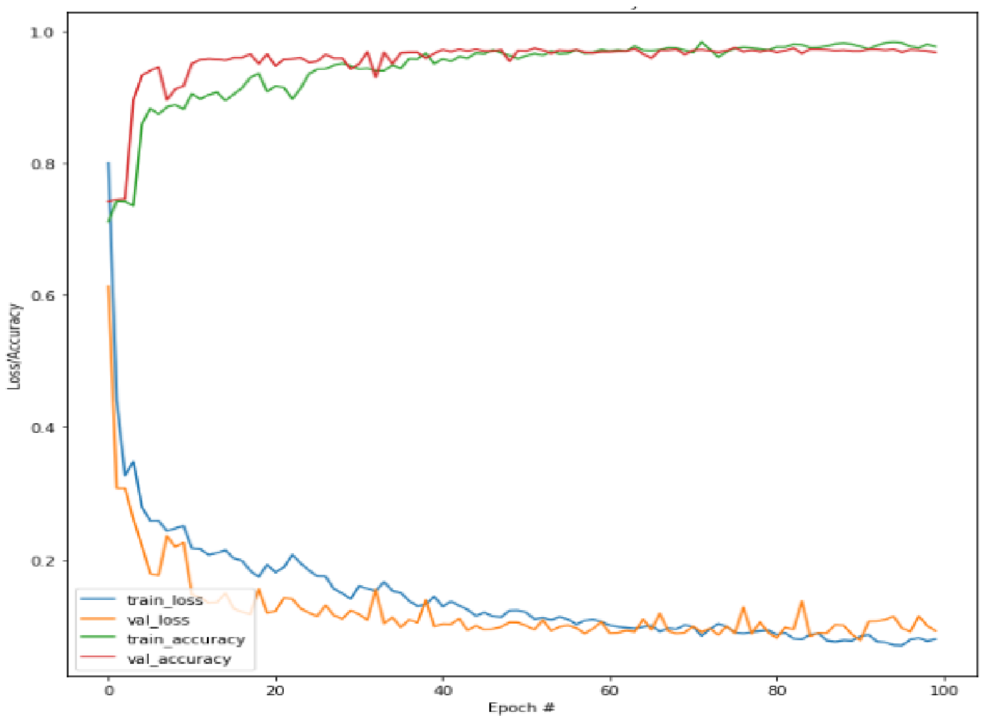
Thus, we have seen that almost all of our models provide astoundingly accurate outcomes. Now, let's investigate how the error margins and accuracy margins of the models fluctuate with each epoch period.

The error per epoch graph for the VGG-16 model can be observed as:



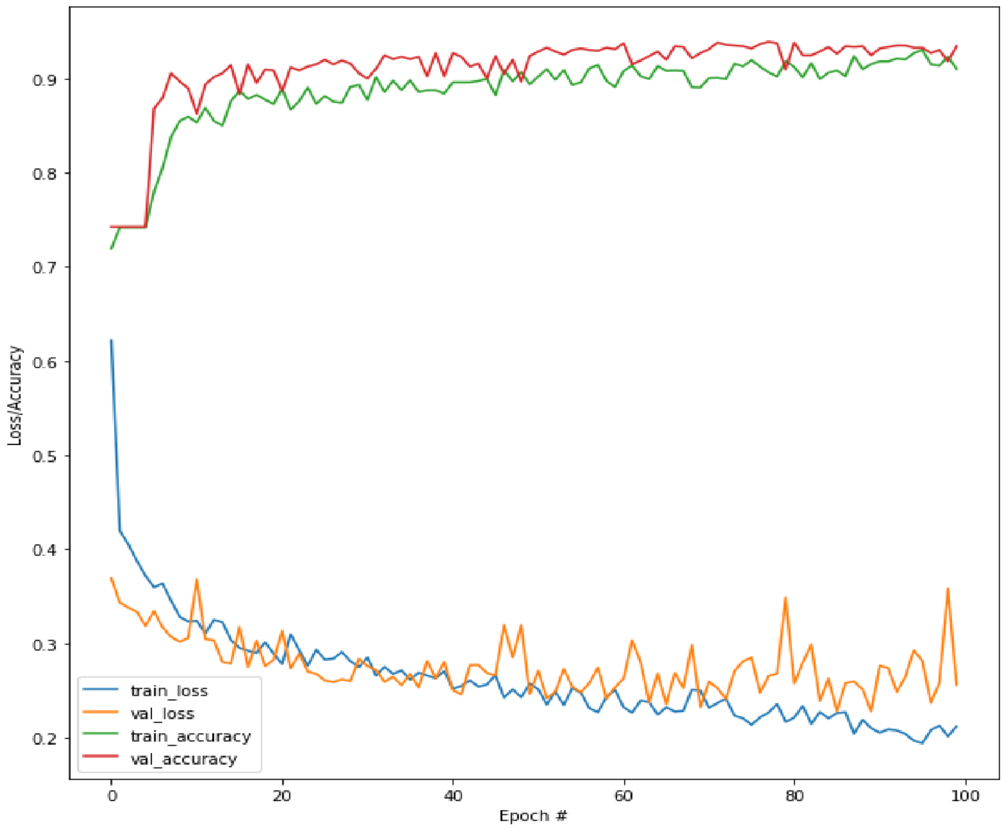
Graph 1 VGG-16 Loss/Accuracy per epoch

It can be seen that, accuracy is converging depending on the epoch, however, loss values are still fluctuating. Let's analyze the graph for the ResNet-50 model:



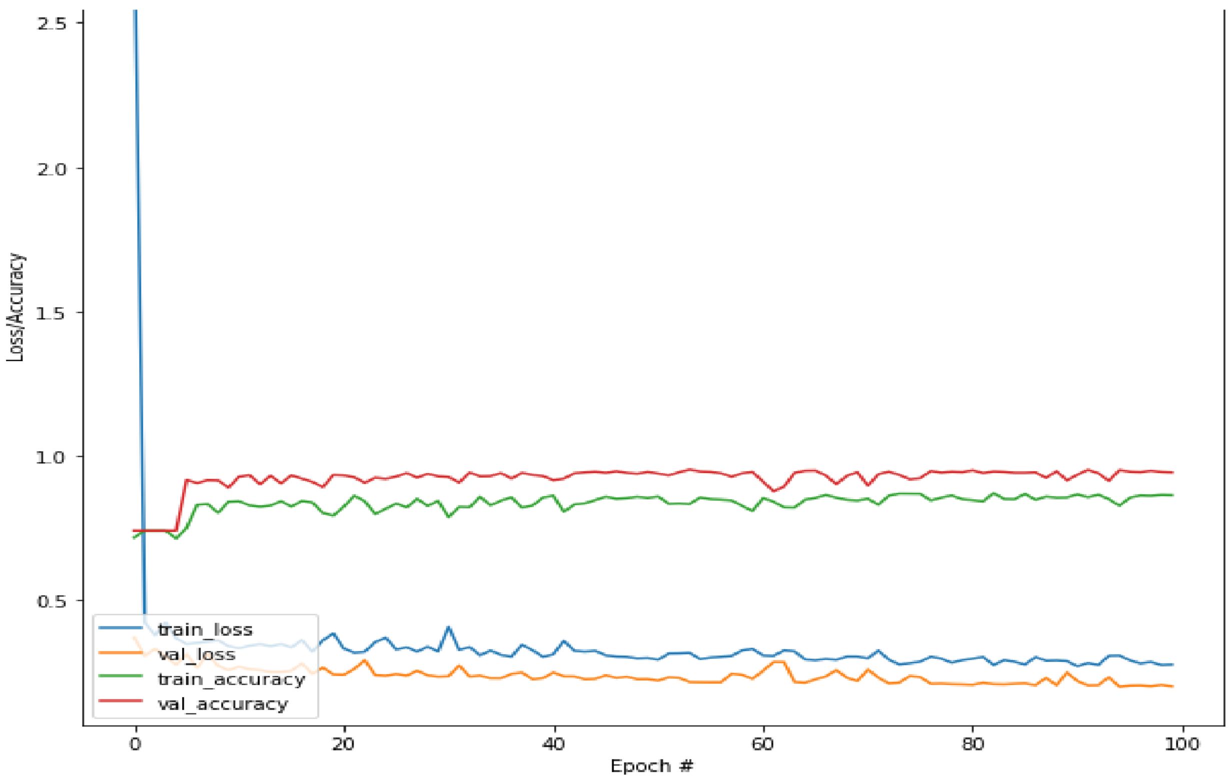
Graph 2 ResNet-50 Loss/Accuracy per epoch

Both loss values and accuracy values are slightly converding upon time. These can be pretended as good results. Now, let's have a look at the results of the MobileNet model:



Graph 3 MobileNet Loss/Accuracy per epoch

Resultant graph for the MobileNet shows that, its performance is even worse than the two previous models. For the Xception model, the grap gis in the following form:



Graph 4 Xception Loss/Accuracy per epoch

The same comments can also be applied to the Xception model.Some models are quite complicated, whilst others provide convergent outcomes.

Although our models return complicated curves, we can say that each of them is successful. Because both Accuracy values and F score values are over 90 percent. However, the most successful among them is the ResNet-50 model. Because, the F-score value is very high and converging is observed in the loss curve. Therefore, we will build explainability concepts based on the ResNet-50 model.

5.2 Grad-CAM results

So far, we have collaborated with the classification part, and according to their study, the ResNet-50 architecture provides the best performance. Let us now explore. Determine the factor the ResNet-50 model considers when reaching a conclusion. So let's get an X-ray image of a person who we know has pneumonia. And let's feed the X-ray picture of this person into the ResNet-50 model for classification. It should also be emphasised that our model has never before seen a photograph of this patient.

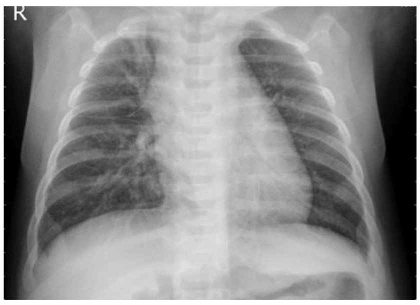


Figure 29 X-ray of thick individual

When presented with this image, our model generates the number nearly close to 1. This value represents a likelihood. This percentage shows that this person has pneumonia 99 percent. In the near future, we will evaluate how our model functions on a healthy person.

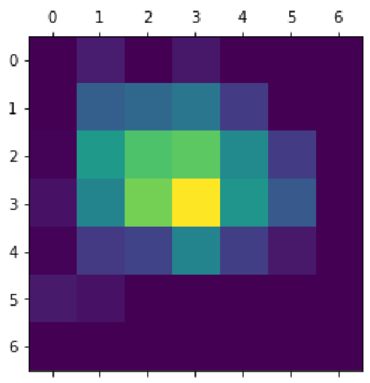


Figure 30 Heatmap for thick individual

In earlier sections, we discussed the working logic of the Grad-CAM approach and claimed that this heatmap is required for a proper feature selection. The planned heatmap for this ill person is shown. We are almost aware of the significant pixel pitches in this heatmap. Let's combine this with our image to make it even more interactive and see the outcome.

Combining our heatmap with the original picture yielded the image seen below. Red indicates the most significant aspects of the model throughout the decision phase. The sensitivity decreases from red to blue. During the decision phase, our model paid the greatest attention to the core locations, as seen by this diagram. From the centre to the outskirts, the significance diminished. First, the colours regress toward yellow, and then they regress toward blue. This demonstrates that our model did not take edge components into account throughout the decision-making process. The examination of healthy persons is just as significant as the study of ill ones.

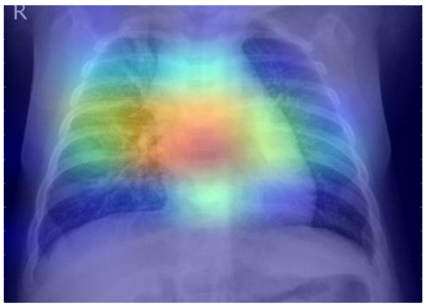


Figure 31 Explainable picture of the thick individual

The image depicts a person who does not have an issue with pneumonia. Even yet, if we examine the heart and lungs, we can see that they are quite clean and distinct, as are the veins. When this image is provided as input to our model, the resulting value is nearly 0.23. This indicates that the likelihood of pneumonia in the lungs is low.

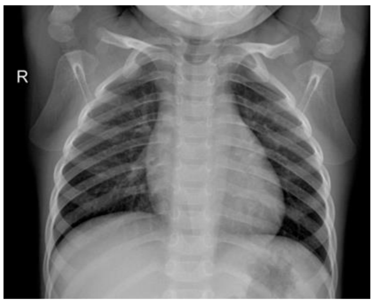


Figure 32 X-ray of healthy individual

Our model has never seen this image before, yet its prediction was accurate. Now let's examine what factors our algorithm considered while creating a recommendation for this person. Let's first examine the heatmap for this. When we examine the heatmap before going on to the whole image, we see that the hotspots are not centred. In other words, our approach continues to examine the essential factors throughout the decision phase.

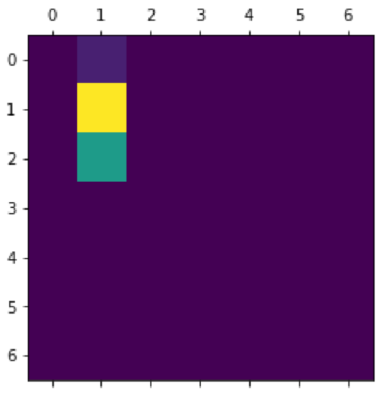


Figure 33 Heatmap for healthy individual

Now, let's examine the image in its whole. In this image, the delicate spots are not seen on the middle side and just over the heart. In other words, this model's judgement on these issues was the clearest. In this image, we can observe that the middle sides are not sensitive. This suggests that the edge dots play a more significant part in decision making in this image.

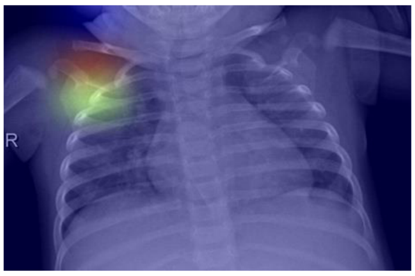


Figure 34 Explainable picture for healthy individual

Our programme could accurately predict both healthy and ill persons that it had never seen before.

5.3 Discussions.

Some of the models we used for Classification performed very well, while others performed poorly. Therefore, the model whose explainability is important is the model that performs best. Because the explainability of the poorly performing model will not be important and will be faulty. As we have seen above, class imbalance appears in our data set. Class imbalance is the significant difference between the number of positive diagnoses and the number of negative diagnoses. Therefore, when we choose our model, we care not only about accuracy but also F-score value. Medical research needs to be even more careful in this regard, because misdiagnosis can cost the patient's life. As can be seen from the tables, our models performed well overall. None of our models performed 100 percent, which is very normal. Since the methods we use are stochastic(random) and we are working with probabilistic models, we will not get 100 percent results. However, among them, the ResNet-50 model showed the best performance. Both the accuracy value and the F-value are very high. When we look at the error graph, we will clearly see that there is convergence. When we look at the confusion matrix, we will see that the TP and TN data are very high and the FP and FN values are very small. Therefore, we can say that the ResNet model is successful. But, our ReNet-50 model is still a black-box algorithm. This means that we have no information about the cause of the diagnosis made by our model. Therefore, it may not be ethically correct to use this model directly. With the help of the

Grad-CAM algorithm, we can have information about the reasons for the prediction of our model.

As an example, when we examine the X-ray image of a sick individual, we will see that our model deals with the central pixels of the image. In the case of pneumonia, inflammation occurs in the lungs, and such inflammations are usually reflected in the X-ray in white. In X-ray images, white colors are the majority in the center and our model is more focused on the central points. So, our model looks for pneumonia in the white colored areas, which is quite logical.

In general, our findings appear pretty sensible

Conclusion

In the absence of prompt diagnosis and treatment, pneumonia, an infection of the lungs, may be fatal. Pneumonia was the leading cause of human mortality during the COVID-19 pandemic. X-rays are one method of diagnosing pneumonia. Using these images, experts can evaluate pneumonia. However, when the number of patients is excessive, these analyses may be performed with the aid of artificial intelligence to safeguard the professionals and shorten the examination time. Computer Vision is used to evaluate these photos, and we intended to conduct this analysis using deep learning techniques; if the model yields satisfactory results, we intended to do this analysis using this logic (Grad-CAM explainability). Initially, a variety of deep learning architectures were evaluated, with the Xception model yielding the best results. In reality, all of the evaluated models produced excellent results, but the ResNet-50 model was better to the others since it had the lowest margin of error, the highest accuracy value, and the highest TP and TN, and lowest FP and FN scores. The ResNet-50 model is, roughly speaking, 97 percent correct. In the next stage, we must comprehend the decision-making processes used by ResNet-50 model. It is helpful for us to know which pixels get more attention and which ones are superfluous. Using the Grad-CAM method, we noticed that despite the fact that various techniques are used for each picture, the weight tends to be concentrated in the centre. Because the heart and trachea are white in colour, and inflammation is often seen in white on an X-ray, the heart and trachea will seem white. Using a red-to-blue colour palette, the images emphasise the algorithm's working logic by displaying the images' highlights. This initiative has potential for expansion. This need a far more extendable data collection and more robust technological apparatus. Additionally, it is feasible to work on other deep learning architectures. This study transcends black-box reasoning and provides knowledge regarding decision-making systems.

References

[1] J. B. Stephen and P. Graham, Respiratory Medicine, WILEY_BLACKWELL, 2011.

[2] “Anatomy of the Lung,” National Cancer İnstitute, [Online]. Available: https://training.seer.cancer.gov/lung/anatomy/.

[3] “Pneumonia in adults: diagnosis and management,” National Institute for Health and Care Excellence, 07 July 2022. [Online]. Available: https://www.nice.org.uk/guidance/cg191.

[4] M. Hoffman, “Picture of the Lungs,” WebMD, [Online]. Available: https://www.webmd.com/lung/picture-of-the-lungs.

[5] “Lung Anatomy,” Physiopedia, [Online]. Available: https://www.physio-pedia.com/Lung_Anatomy.

[6] “X-rays,” National Institute of Biomedical Imaging and Bioengineering, [Online]. Available: https://www.nibib.nih.gov/science-education/science-topics/x-rays.

[7] “Chest X-ray,” RadiologyInfo, [Online]. Available: https://www.radiologyinfo.org/en/info/chestrad.

[8] “X-ray,” NASA SCIENCE, [Online]. Available: https://science.nasa.gov/ems/11_xrays.

[9] “How Do X-rays Work?,” Independent Imaging, [Online]. Available: https://www.independentimaging.com/x-rays-work/#:~:text=An%20X%2Dray%20is%20produced,atoms%20in%20the%20metal%20plate..

[10] V. Jain, R. Vashisht, G. Yilmaz and A. Bhardwaj., “Pneumonia Pathology,” National Library of Medicine, 12 04 2022. [Online]. Available: https://www.ncbi.nlm.nih.gov/books/NBK526116/.

[11] W. Lim, S. Baudouin, R. George and e. al, “BTS guidelines for the management of community acquired pneumonia in adults:,” *Thorax,* Vols. 64:iii1-iii55., 2009.

[12] G. Mackenzie, “The definition and classification of pneumonia,” *BioMed Central,* 2016.

[13] D. A. Shamila, J. Hamidreza, T. Payam and M. Esmaeil, “Immunopathogenesis of Pneumonia in COVID-19,” National Library of Medicine, 2020. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7680509/.

[14] W. Pei, M. Dagmar, W. P. L. Colin and R. ,. T. Kristinn, “On Defining Artificial Intelligence,” *Journal of Artificial General Intelligence,* 2019.

[15] “Artificial Intelligence (AI),” IBM Cloud Education, [Online]. Available: https://www.ibm.com/uk-en/cloud/learn/what-is-artificial-intelligence.

[16] A. Samuel, “Some Studies in Machine Learning Using the Game of Checkers,” 1959.

[17] A. Géron, Hands-On Machine Learning with Scikit-learn, Keras & Tensorflow, O'REILLY, 2019.

[18] S. Badillo, B. Banfai, F. Birzele, I. I. Davydov, L. Hutchinson, T. Kam-Thong, J. Siebourg-Polster, B. Steiert and J. D. Zhang, “An Introduction to Machine Learning”.

[19] A. Smola and S. V. Vishwanathan, Introduction to Machine Learning, Cambridge University Press.

[20] E. Alpaydın, Introduction to Machine Learning, London: MIT Press, 2010.

[21] H. Dalianis, “Evaluation Metrics and Evaluation,” in *Clinical Text Mining*.

[22] M. Hossin and M. Sulaiman, “A REVIEW ON EVALUATION METRICS FOR DATA CLASSIFICATION EVALUATIONS,” *International Journal of Data Mining & Knowledge Management Process,* 2015.

[23] A. ZHANG, Z. LIPTON, M. LI and A. J. SMOLA, Dive into Deep Learning.

[24] F. Chollet, Deep Learning with Python, MANNING.

[25] B. S. Maind and P. Wankar, “Research Paper on Basic of Artificial Neural Network,” *International Journal on Recent and Innovation Trends in Computing and Communication,* vol. 2.

[26] N. Buduma, N. Buduma and J. Papa, Fundamentals of Deep Learning, O'REILLY.

[27] I. Goodfellow, Y. Bengio and A. Courville, Deep Learning, MIT Press.

[28] D. Hubel and T. Wiesel, “Receptive fields of single neurones in the cat's striate cortex,” *The Journal of Phsiology,* 1059.

[29] Y. LeCun, L. Bottou, Y. Bengio and P. Haner, “Gradient Based Learning Applied to Document Recognition,” 1998.

[30] A. Krizhevsky, I. Sutskever and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks”.

[31] A. Ghosh, A. Sufian, F. Sultana, A. Chakrabarti and D. De, “Fundamental Concepts of Convolutional Neural Network”.

[32] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting,” *Journal of Machine Learning Research,* 2014.

[33] S. Ioffe and C. Szegedy, “Batch Normalization: Accelerating Deep Network Training b y Reducing Internal Covariate Shift,” 2015.

[34] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” 2014.

[35] K. He, X. Zhang, S. Ren and J. Sun, “Deep Residual Learning for Image Recognition,” 2015.

[36] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto and H. Adam, “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications,” 2017.

[37] F. Chollet, “Xception: Deep Learning with Depthwise Separable Convolutions,” 2016.

[38] F. Zhuang, Q. Zhiyuan, K. Duan, D. Xi, H. Zhu, Y. Zhu, H. Xiong and Q. He, “A Comprehensive Survey on Transfer Learning,” 2019.

[39] W. Samek, G. Montavon, A. Vedaldi, L. ,. Hansen and K.-R. Müller, Explainable AI: Interpreting, Explaining and Visualizing Deep Learning, 2019.

[40] V. Belle and I. Papantonis, “Principles and Practice of Explainable Machine Learning,” *Frontiers,* 2021.

[41] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh and D. Batra, “Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization,” 2016.

[42] K. Bichakchi, “Grad-CAM ile Modeli Açıklamaya Çalışmak | Part 3,” Medium, 27 November 2021. [Online]. Available: https://medium.com/machine-learning-t%C3%BCrkiye/grad-cam-ile-modeli-a%C3%A7%C4%B1klamaya-%C3%A7al%C4%B1%C5%9Fmak-part-3-c52144de7253#:~:text=K%C4%B1saca%20Grad%2DCAM%20nedir%3F,sahip%20olan%20label'%C4%B1%20kullanaca%C4%9F%C4%B1z..

[43] S. D. Kermany, M. Goldbaum, W. Cai, A. Lewis, H. Xia and K. Zhang, “Identifying Medical Diagnoses and Treatable Diseases by Image-Based Deep Learning,” *Cell,* 2018.

[44] P. BREVIGLIER, “Pneumonia X-Ray Images,” Kaggle, [Online]. Available: https://www.kaggle.com/datasets/pcbreviglieri/pneumonia-xray-images.

[45] “St. John's Research Institute in India,” [Online]. Available: https://www.sjri.res.in/sites/default/files/Pneumonia.pdf.

[46] S. Alipoor, H. Jamaati, P. Tabarsi and E. Mortaz, “Immunopathogenesis of Pneumonia in COVID-19.,” *Tanaffos,* 2020.

[47] M. S. Mahoney, “The History of Computing in the History of Technology,” *Princeton University, Princeton, NJ.*

[48] G. James, D. Witten, T. Hastie and R. Tibshirani, An Introduction to Statisical Learning, Springer.

[49] G. Maguolo and L. Nanni, “A Critic Evaluation of Methods for COVID-19 Automatic Detection from X-Ray Images”.

List of pictures, tables, graphs and abbreviations

1.3 List of figures

1.4 List of tables

1.5 List of graphs