POLITECNICO DI TORINO

Master's Degree in Computer Engineering

Politecnico di Torino

Master's Degree Thesis

Mobile application which makes diagnosis of lung diseases by detecting anomalies from X-Ray images

Supervisor:

Prof. Giovanni Malnati

Candidate:

Atabay Heydarli

Co-Supervisors:

Prof. Tomasz Trzciński (Warsaw University of Technology)

PhD. Szymon Płotka (Warsaw University of Technology)

A. Y. 2021/2022

Abstract

The COVID-19 pandemic still continues to have an enormous influence on the health and well-being of the world's population. An important step in the fight against this pandemic is using the successful screening methods of infected patients, where one of the key approaches for that is the chest X-ray radiological imaging. This study was designed to create mobile application which uses machine learning techniques to automatically detect COVID-19 and pneumonia patients, with checking their chest X-rays with maximum detection accuracy using Deep Neural Networks (DCNN). According to the studies analysed in second chapter of the thesis, COVID-19 detection accuracy utilizing CNN on chest X-rays is quite high and accurate. Despite the fact that there have been numerous studies on COVID-19 detection, no research has produced a quick and cellular-based COVID-19 detection system that uses CNN. The Single Shot Identification (SSD) MobileNet object detection model was employed in this research to generate a quick yet accurate detection of COVID-19.

For solving this issue, we provide a mobile application which uses MobileNet neural network technology to make a diagnosis from chest X-Ray images. Before developing the application, some important steps were done, such as searching for existing datasets with chest X-Ray images and the best ones were selected for using it during the model training. The model was designed by using MobileNet V2 which were trained with two datasets combined together for this thesis work. After having the trained model, which provides significantly high accuracy during the prediction, the mobile application was developed. Created mobile application has a connection with mentioned machine learning model, for predicting diagnosis based on the images taken from the camera of mobile device or uploaded from the storage of device. To conclude, the MobileNet V2 learning model was proposed to identify coronavirus-infected pneumonia patients through chest X-rays taken from mobile device and provides classification accuracy of more than 92%, where training accuracy is 95% and accuracy of validation set is 91%.

Acknowledgment

I must first express my gratitude to my research mentors, Professors Giovanni Malnati, Tomasz Trzciński, and Ph.D Szymon Płotka. This paper would not have been completed without their help and diligent participation in each step of the procedure. I want to express my gratitude to you for your help and patience over the past few months

I travelled to Warsaw University of Technology in November 2021 as part of a mobility program that lasts for four months in order to work on my thesis with Professor Tomasz Trzciński and Ph.D. Szymon Płotka. I have been quite busy during my stay at Warsaw University of Technology and working with Tomasz Trzciński and his colleagues has been a very remarkable experience. This university hosted a significant portion of the analysis for the thesis, particularly for the machine learning section. Professor Giovanni Malnati of the Politecnico di Torino provided support for the remaining portions of the thesis, which were concerned with the development of machine learning. I want to express my gratitude to all of these teachers for their assistance and support throughout this period.

Most significantly, none of this would have been possible without the support of my family, girlfriend and friends. My girlfriend, who daily encouraged me and showed her love through video calls and messages. I want to thank Nigar Huseynova for all of her kindness and compassion that she has shown me over the past few years, I love her. To say that our family has gone through some ups and downs recently would be an understatement to my parents, brother, and sister. I will always be grateful that you stopped me from giving up each time I was about to.

Table of Contents

List of Tables

List of Figures

Acronyms

AI Artificial intelligence

ML Machine Learning

PML Pixel-Based Machine Learning

SSL Semi-Supervised Learning

NN Neural Networks

MTANN Massive Training Artificial Neural Networks

CNN Convolutional Neural Networks

MNN Modular Neural Networks

CXR Chest X-Ray

MODS Microscopic Observed Drug Susceptibility

MLP Multilayer Perceptron

FPS Frames Per Second

NIH National Institutes of Health

RSNA Radiological Society of North America

DICOM Digital Imaging and Communications in Medicine

SDK Software Development Kit

API Application Programming Interface

Introduction

People all over the world suffer from various lung diseases. Lung disease makes the lungs more susceptible to certain physical problems and air pollution. As a result, lung function becomes difficult. Emphysema, asthma, pleural effusion, tuberculosis, and some other various lung diseases, as well as aspiration fibrosis, pneumonia, and lung cancer, can cause the lungs to lose their ability to adapt by reducing the amount of the air which can be practically held in the lungs [1]. the disease is contagious, it is imperative to diagnose the condition and provide the patient with proper care. Radiologists usually work with chest x-rays to identify and diagnose lung disease. A most famous one is chest x-ray, which can be used by radiologists to detect and diagnose a variety of diseases and other disorders. These are people of other evils, inflammation of the pericardium and inflammation of the bronchi [2]. More than 2 million operations are performed annually in the field of chest radiography, which is considered the most popular assessment of diseases in the world [3].

Radiation and functional tests, such as X-ray, electrocardiogram, computed tomography, magnetic resonance therapy, etc., are prescribed to accurately diagnose or check for the presence of diseases at an early stage. A technician can perform a test on medical equipment, but only a qualified highly specialized doctor can analyse the results of the research. In sparsely populated areas, there is a shortage of specialists, as a result, it is not possible to perform a study, and in hospitals in large cities, a large load on doctors, as a result, the quality of the analysis may decrease due to unintentional errors.

Examination, diagnosis, and treatment of chest diseases, one of the most popular reasons of death worldwide [4], depend on this technique. Therefore, computer technology must be used to analyse chest radiographs as efficiently as radiologists to improve workflow prioritization and clinical decision support in large-scale healthcare projects and programs around the world.

1.1 The prevalence of lung disease in the world

In the whole world, respiratory diseases take the second place in the structure of general morbidity, after diseases of the circulatory system [5, 6]. According to figures provided by the World Health Organization, pneumonia is one of the most common causes of mortality from respiratory disorders. About 75% of all respiratory disease deaths result from pneumonia. All the structural components of the lung tissue, especially the alveoli and interstitial tissue, are affected by pneumonia, an acute lung lesion of an infectious-inflammatory character. An X-ray of the lungs is one of the primary ways to identify pneumonia.

Chest X-ray is the most common radiological diagnostic method in the world, accounting for up to 45% of all radiological examinations [7]. The wide availability of the method is determined by its low cost and high diagnostic potential for such socially significant pathological processes as tuberculosis, lung cancer, and pneumonia. At the same time, radiography is an example of diagnostic ambiguity. The reason for this is that a planar image is formed as a result of the superposition of anatomical structures having different structure, composition and density. As a result, an image may contain dozens of features that occur in hundreds of pathological processes and conditions. This causes difficulties in reading and interpreting radiographic data, the occurrence of discrepancies between radiologists, which often leads to unreasonable additional clarifying examinations.

When diagnosing diseases for medical reasons, the probability of medical error is quite high: for pneumonia, about a third of the diagnoses are incorrect. Therefore, the development of services to support medical decision-making is so relevant. Machine learning algorithms are considered as part of a system for preliminary evaluation of patient test medical data. Such systems use an extensible set of supervised learning algorithms, each of which is used to preliminarily estimate the probability of a particular diagnosis. The system automatically analyses the data available to it and issues recommendations for a specialist.

1.1.2 Importance of using machine learning for medical image processing

Numerous machines learning techniques, including linear discriminant analysis, support vector machines, decision trees and random forests, neural networks, and deep learning algorithms, have been used over the years. Deep learning has shown notable progress in medicine compared to other machine learning techniques. Layered networks are employed in the deep learning subfield of machine learning to assess intricate patterns in source picture data. Convolutional neural networks are one of the deep learning methods used for pattern recognition.

Deep learning allows you to directly analyse visual data, without additional image feature extraction. However, the quality of deep learning algorithms depends on the amount of labelled data. In most cases, to achieve the required recognition level requires a large number of images (tens and hundreds of thousands). However, the volume of medical image sets rarely exceeds 1000. Therefore, the main problem when applying deep learning to medical images is the limited number of training samples available to create deep models that are not subject to overfitting [8]. To solve this problem, research groups have developed various strategies, such as:

- feature extraction from images, which allows using less data and simpler machine learning models for training.

- expanding the data set by artificially generating samples through affine transformation (i.e., increasing the amount of data), and then training the network from scratch using the expanded data set.
- initialization of the model parameters with the parameters of pre-trained models from non-medical or natural images, with further fine tuning of the network parameters using images corresponding to this task [9].

Despite the fact that many publications on machine learning appear every year, there are still only a few methods that can process a wide range of visual signs of similar disease states. For example, the use of convolutional neural networks to distinguish between interstitial lung diseases involves subtle changes in texture-type structures, which is very different from the classification of ordinary photographs of cats and dogs.

The range of representations of various disease states, the need for large labeled clinical datasets, and the complex structure of many machine learning methods mean that research and development must continue before they can be implemented and used in clinical settings. The use of deep learning for medical problems is also of concern due to its "black box" principle; however, there are methods for evaluating trained parameters in neural networks to uncover decision-making methodology.

1.2 Thesis objectives

This work, presented as a Master Thesis, which tackles the problem of image classification using mobile devices. The aim of the created mobile application is to take a picture of patient's X-Ray image and detect if there is lung disease or not. If yes, application should detect, which type of disease does the patient has. For solving this issue, some classic machine learning techniques were used, especially the classification techniques based on the images provided in dataset.

Breakthrough techniques for the automated processing of medical data, including diagnostic images, have emerged in recent years. Particularly, the employment of neural networks, parabolic, and vector regression models was suggested for the diagnosis of lung illnesses. A neural network approach and a synthetic immune system were presented to diagnose chronic broncho-obstructive illnesses and pneumonia [10].

Decision trees and segmentation were used to create algorithms for the diagnosis of tuberculosis, lung cancer, and pneumonia [11]. The aforementioned techniques were effective at classifying pathological illnesses, but they lagged behind deep machine learning techniques in terms of performance, accuracy, specificity, and sensitivity. Competitive, backpropagation, and convolutional neural networks have demonstrated benefits over humans in terms of interpretation accuracy and speed. High-performance models might be built thanks to the following use of neural network capabilities to address the issues of identifying specific pathological states in medical photos.

The major objective of this work is to offer a solution to the problem we have identified utilizing tried-and-true multimodal learning methods. We initially offer our unimodal representations, then we review the most recent cutting-edge techniques used in each research field and present better versions of those techniques. On the basis of that, we provide our final multimodal technique for classifying CXR images. Additionally, provided product is having some more functionalities, rather than image classification. Mobile application will provide to users the possibility for checking total numbers of people who suffers from Covid, statistics based on the country and some other Covid case statistics, also user can see the nearest hospitals which accept patients who are infected with this virus.

1.3 Contribution

The main contribution of this Master Thesis work is as follows:

- Introduction of the new research problem of CXR image classification, as realizing the classification of this medical images, which were taken from mobile device camera with already defined techniques are insufficient for this task.
- Definition of the problem as multimodal learning task and propose the machine learning, computer vision solution with a pipeline which specifically designed for this purpose. In order to create our pipeline, we first determined which research fields our issue comes under and then examined the most recent methods for each modality, by reading and analysing the newest publications and research related to the topic of this thesis work.
- For the computer vision part, we propose to use convolutional neural network in a lite mode for increasing the possibility of using deep learning techniques in the mobile device.
- We modify the training procedures and apply the deep learning techniques for CXR image classification to our pipeline in order to provide superior digital medical image representations for the thesis work's proposed solution.
- In order to evaluate the method used for the solution of this work, we provide digital image representation, of the medical images from CXR dataset, that used in this thesis, which were fully consistent with the previous version.
- For showing the results of created model, we developed mobile application, which has functionalities to take or upload a photo of X-ray image and see the diagnosis, based on the trained model. This application has also some additional functionalities, which can be helpful to the user of it.

1.4 Outline

In the following chapters of the thesis work the key components of proposed solution will be described. The thesis work is organised in the following manner:

- **Chapter 2** outlines the previously written papers and proposed approaches based on the topic of current thesis work. In this chapter some previously written academic papers and technical publications will be analysed, their pros and cons will be mentioned. Additionally, will described the positive parts of analysed work, which will be used during the development of this work
- **Chapter 3** outlines the ideas for mobile classification of CXR images, with the purpose of making a diagnosis. Includes the technical aspects about related work and the explanation of various methods and techniques which are used throughout this work, during the development of Machine learning algorithm.
- **Chapter 4** introduces the proposed timeline for the classification of CXR images using mobile device, by first describing each unimodal representation and later introducing the methods which are using to combine these separate representations to create one multimodal approach, for solving the issue. Additionally in this chapter there is the evaluation of proposed and developed machine learning solution, by the qualitative comparison using the implementation of both unimodal and multimodal approaches.
- **Chapter 5** will describe the developed mobile application, without taking into consideration the machine learning part of the project. Here will be shown used techniques during the development of the application, used technologies, explanation of additional functionalities that application has and also the way, how to run and use the mobile application
- **Chapter 6** summarise the proposed solution and all the work done during this thesis work, also it delineates the considered steps which will be extremely useful for the future work.

Chapter 2

Related Work

Over time, a variety of ML (Machine Learning) methods have been employed, including support vector machines, decision trees, random forests, neural networks, and deep learning algorithms. In comparison to other machine learning methods, deep learning has a significant edge in the field of medicine. In the machine learning field of deep learning, layered networks are used to analyse complex patterns in source image data.

Without further image feature extraction, deep learning enables immediate visual data analysis. However, the quantity of labelled data affects how well deep learning algorithms perform. The majority of the time, tens of thousands or more photos are needed to reach the desired level of recognition. The number of medical image sets, however, hardly ever reaches 1000. The lack of training samples needed to build deep models that are resistant to overfitting is thus the key issue when using deep learning to analyse medical images. Various research teams have developed approaches to solve this issue, including:

- feature extraction from images, which allows for the use of fewer data and more straightforward machine learning models for training.
- expanding the dataset by artificially generating samples through an affine transformation, which increases the amount of data, and then training the network from scratch using the expanded dataset.
- initialization of the model parameters using parameters of previously trained models from non-medical or natural images, followed by additional network parameter training (fine-tuning) using images suitable for this task.

Even though a lot of articles on machine learning are published every year, there are still very few techniques that can analyse a large range of visual indicators associated with related illness states. For instance, the categorization of common images of cats and dogs is significantly different from the application of convolutional neural networks to distinguish between interstitial lung illnesses, which entails subtle changes in texture-type structures.

Since there are many ways to describe different disease states, big labelled clinical datasets are required, and many machine learning techniques have complex structures, further research and development is required before they can be put into practice and used in clinical settings. Deep learning's "black box" principle raises concerns about its application to medical issues, yet there are ways to assess training parameters in neural networks to identify the decision-making process. The likelihood of medical error is relatively high when diagnosing diseases for therapeutic purposes; for example, pneumonia, roughly one-third of diagnoses are erroneous. As a result, the creation of services to assist in medical decision-making is crucial. Algorithms for machine learning are taken into account as a component of a system for a preliminary

analysis of patient test medical data. These systems employ a diverse range of supervised learning algorithms, each of which is utilized to make an initial estimation of the likelihood of a specific diagnosis. The program automatically examines the data at its disposal and suggests a specialist.

As this thesis work mainly focuses on lung diseases (especially the pneumonia and Covid) classification using mobile devices, we first would like to discuss some already existing screening approaches in a short way. All analysed techniques are using radiography, which means a method used in medical imaging for being obsolete. However, machine learning and digital advances are previously revived this possibility and used some new techniques in diagnosis of lung diseases [12]. Mainly these techniques allow detecting multiple shapes of cardio lessons based on X-ray scans. The popularity of machine learning applications during the medical diagnosis correlated with the accuracy of used models is a extremely great success as it leads to better disorder recognition. Recent encouraging results in deep learning applied in the field of lung diagnosis led to the usage of a GPU-based platform which is able to process a large volume of images in high-resolution within seconds and thus exceed the work of radiologists. In the following chapters, some known techniques, and previously written papers will be analysed.

2.1 Pixel-based Machine Learning

One of the most popular methods which are started to be popular after the availability of powerful machines that have extremely high computational powers, is pixel-based Machine Learning techniques. This technique is used in medical image analysis and processing. The difference of this technique in comparison of the simple ML techniques, is that it uses directly the pixel values of the input images, instead of calculating various features from the segmented areas of the proposed image. Based on that, neither feature extraction nor segmentation is required during the usage of this machine learning technique.

The performance of Pixel-Based Machine Learning (PML) can possibly get significantly higher than simple, common classification techniques [13], as PML method is able to avoid errors caused by incorrect processed feature extraction procedure and inaccurate segmentation, because in this technique we don’t need those functions.

2.1.1 Overview

Related to the popularity of PML methods during last years, medical image processing and analysis, together with computer vision, have adopted many various PMLs in completely different tasks. In the table 2.1 the brief summary of various classes which use PML will described, together with their functions and possible applications.

PMLs	Functions	Applications
Neural filters (including neural edge enhances)	Image processing	Edge enhancement from images Enhancement of edges which traced by physician [14]
Convolutional neural networks (including shift-invariant neural networks)	Classification	FP reduction in mammography FP reduction in CAD for lung nodule detection in CXR Character recognition [15] Face recognition FP reduction in CAD for detection of lung nodules in CXR
Massive-training artificial neural networks (MTANNs)	Classification, object detection, pattern enhancement	FP reduction in CAD in colonography Bone separation from soft tissue in CXR
Others	Image processing or classification	Separation of ribs from soft tissue in CXR [16] Segmenting posterior ribs in CXR

Table 2.1 Classes of PMLs with functions and possible applications [13]

Convolution neural networks, which comprise shift-invariant neural networks, neural filters, and MTANNs, or massive-training artificial neural networks, are the three categories of PMLs that we can differentiate (includes several iterations, such as vector regression of massive-training support or Laplacian eigenfunction MTANN). Neural filters were employed for many images analysis tasks. Among these are edge augmentation from noisy pictures, edge-preserving noise reduction in digital photos and radiographs, and augmentation of doctor-drawn subjective edges in cardiac ventriculograms.

Convolutional neural networks have been used for classification in tasks requiring the reduction of false positives in CAD schemes for the detection of masses in mammography, character recognition, and face recognition. A reduction in false-positive results in CAD lung nodule identification systems, as well as the separation of benign from malignant nodules, have been solved using massively trained artificial neural networks. This class of PMLs has also been used for lung nodule enhancement in X-ray computed tomography as well as suppression and pattern augmentation of bone tissue from soft tissue in CXR.

2.2 Bone separation in CXR by use of MTANNs

Chest X-Ray is one of the most popular medical image techniques used during the process of diagnosis of various lung disease such as pneumonia, tuberculosis, covid cases and mainly all other possible lung diseases. Based on the statistics [17], there are roughly one million of people, mainly adults require hospitalization, because of

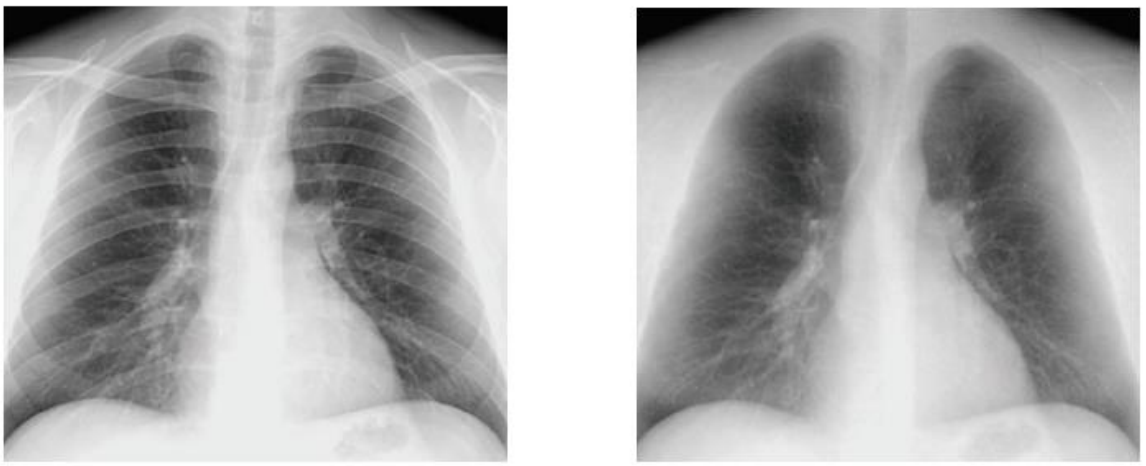
pneumonia, and 5-7% of them, which is significant to 50-70 thousand of people dies every year, only in US. Except pneumonia, for the last two years the Covid pandemics started, which leads to 6.5 million deaths all over the world, with almost 610 million cases worldwide [18].

Except of previously analysed PML techniques which are used in medical image processing and analysing, there is one more popular technique which is developed few years ago and called bone separation from soft tissue. Previously these types of diseases are processed with the examination of lung nodules CSR, which can lead to overlooking of extremely hard diseases, such as lung cancer. However, not all of them can be visible in the retrospect. Provided studies clearly shows that some lung cancer diagnoses were not given properly, because they are missed due to partially obscured by bones, especially ribs or clavicle. For solving this issue, the new techniques will provide, which examined dual energy imaging, and focuses on producing images of two completely different tissues, such a "soft tissue" and the "bone" [19]. This technique was successful in most cases, but obviously it has many drawbacks, but the main important one is the exposure of the radiation.

For solving this problem, the MTANNs models were developed and served as a main technique which used for solving the ribs separation technique. The idea for training this type of algorithm and the model is seems quite simple, because the main idea is to provide to the model images of bones and separately the soft-tissue images which were obtained from the dual-energy radiographic medical imaging system.

Analysing the PMC article "Separation of bones from soft tissue in chest radiographs: Anatomy-specific orientation-frequency specific deep neutral network convolution" published in 2019 [20], we can see that MTANN was trained using CXR medical images as input and corresponding images which are boneless or can be named as soft-tissue lung images. This model uses the dataset of 2768 CXR images, where 729 of them are only bone images, for sure from the chest X-Ray images.

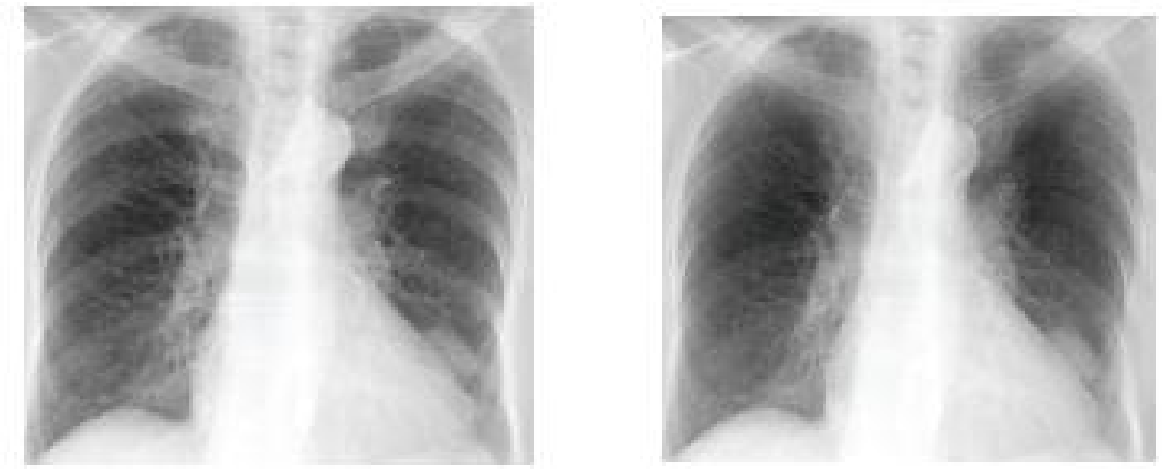
In the figure 2.1 and figure 2.2 you can see the results of the provided model by "Amin Zarshenas". As it is shown in the images, the bones, exceptionally the ribs contrast is obviously suppressed in the result image of the model. If you check the resulting images, it is clearly seen that, removing the bones from the image increase the visibility of the soft tissue areas such as lung vessels.



(a) Original Image (b) resulting image

Figure 2.1 Bones separation model's training samples.

In the Figure 2.1 the training samples were shown, which are used during the training session of the model. These images were stored in the test dataset, which contained boneless and normal CXR images. As an example, the (a) image in the 2.1 figure, represents the normal CXR image with soft-tissue and bone together, where (b) image shows only the soft-tissue image, where the bone is discarded.



(a) Input Sample (b)Generated result

Figure 2.2 Result of the trained model for bones separation.

As explained in the paper [20] after training the MTANNs model developed by the writer of the paper, the model provides the boneless images with the high accuracy. In the (a) image of the figure 2.2 you can see the input sample, which were entered to the trained model, and after using the algorithm, the resulting image was as shown in the image (b) of the figure 2.2. As you can see, the model successfully provides the images without bones.

This technique was popularly used during the previous years, because it is also obvious from the previous examples that, it is easier to distinguish the inappropriate situation inside the lungs, while checking it without the bones. This technique both help humans to better make the diagnosis, and also makes the machine learning techniques, easily find the anomalies in the CXR images.

2.3 Automatic CXR screening system

Instead of PML techniques, there are some other Machine Learning which are popular in medical image analysis and processing. One of them is automatic CXR screening system technique. This method has multiple stages of processing system which are using multistage framework. In this chapter we will look through the paper which is published in "International Journal of Computer Assisted Radiology and Surgery" [21] and describes the usage and development of automatic CXR screening system.

The provided algorithm uses almost similar intuition as the specialist radiologist in the process of lung examination, which is mainly consist of the comparison between the left and right fiends of the lung, which later helps to specialist to make the right diagnosis. Where the texture features of the medical image describe the fields which are inside the lung and the other features, mainly the shape ones focus on including the relevant characteristics, especially the geometrical characteristics.

The automatic CXR screening system contains three different stages, which are clearly shown in the Figure 2.3.

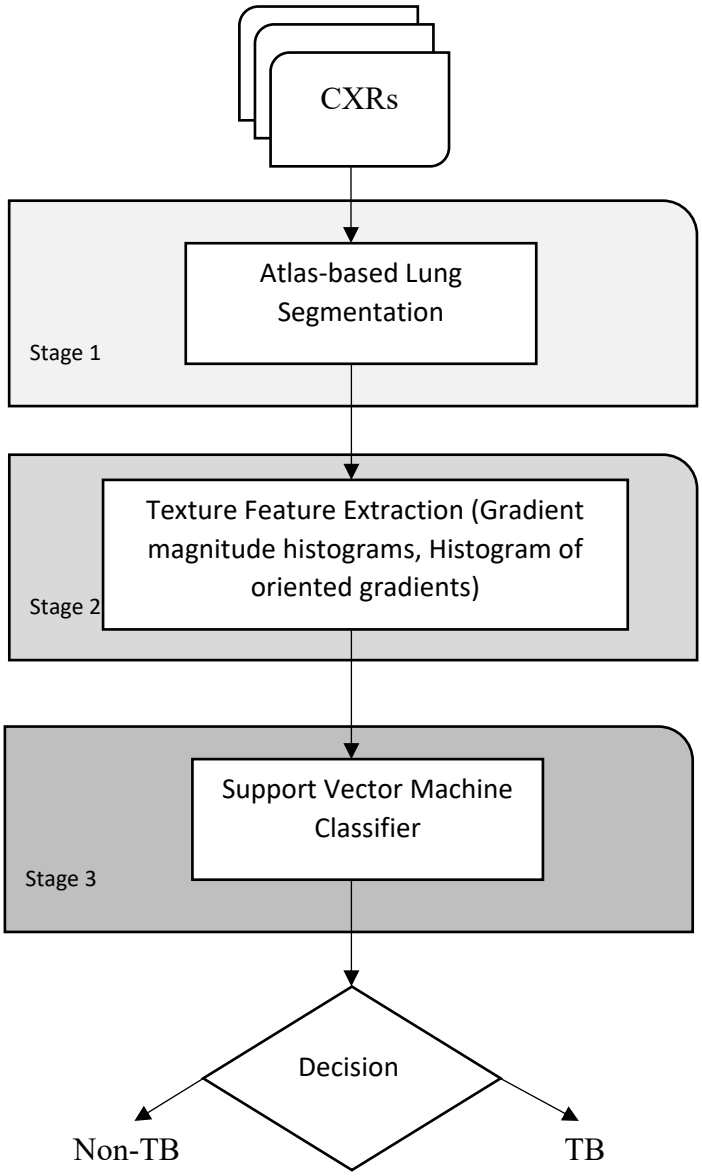


Figure 2.3: The structure of the screening system which is created for detection of tuberculosis

The method which was presented on the previous schema (figure 2.3) contains three different consecutive stages which uses various main techniques of machine learning, respectively segmentation for the first stage, feature extraction technique for the second stage and classification method for the last stage, which is the third one.

The first stage of the provided method contains three different phases inside, the first one of which is content-based retrieval of images by using the Bhattacharyya [22] shape measure of similarity and partially the Radon transform [23]. These two methods help to retrieve the medical image and make them to be in an appropriate shape that will be easier to the algorithm to work on. Next phase of the first stage helps to build an anatomic model of the patient's lung shape and finally on the last phase the method takes out the boundaries of lungs using the graph cut optimization approach.

On the next stage texture feature extraction method is using for the extracting the texture feature from the retrieved and processed image in the previous stage. This method uses intensity histogram, histogram of oriented gradients and gradient

magnitude. One of the most significant issues in the world of digital information processing is the analysis of image texture. You may identify inhomogeneities, perform segmentation, and highlight the foreground and background using the texture in the image. In this article, we'll go through a few techniques for textural analysis of images. The five categories of texture analysis techniques include statistical, geometric, structural, spectral, and modelling methods. In this article, techniques for the statistical, geometrical, spectral, and model categories will be discussed.

Finally, on the last stage of this method the classification technique is used. The classification which used in this stage is not the simple classification but the techniques which uses a support vector machine model (SVM) [24]. Multiclass classification is not natively supported by SVM. It facilitates categorizing data points into two classes and using binary classification. After dividing the multiclassification problem into numerous binary classification problems, the same method is applied to multiclass classification. To achieve mutual linear separation between every two classes, it is intended to map data points to high-dimensional space. This technique, known as a One-to-One approach, divides the multiclass classification problem into numerous binary classification problems. For each pair of classes, a binary classifier. Another strategy is called One-to-Rest. In that method, a binary classifier is used as the breakdown for each class.

After all these three stages which were briefly explained in the previous mini paragraphs, the created model is showing the result based on the final classification technique which used on the last stage of the algorithm. So, the provided algorithm by S. Jaeger, J. Siegelman, receives the input CXR images, retrieve them, then make an image processing for changing blank medical image to the shape that is easy for the model to read, then uses texture feature extraction for extracting the texture of lung images and later uses the classification inside the data, which is stored in the provided dataset, and at the end shows two possible results:

- Positive: if the algorithm detects that inserted CXR image has anomalies and make a diagnosis as tuberculosis.
- Negative: if the algorithm detects that inserted CXR image has no anomalies and shows that patient doesn’t have tuberculosis

This method is very effective in the process of making diagnosis from the CXR images, but it has some drawbacks, such as:

- Used techniques are enough complex to make the algorithm complex and time consuming to provide the result.
- There are multiple stages where each of them has some other phases which uses different techniques. This makes algorithm to be powerful but having a small issue in one of the provided algorithms could be destroy the full model.
- As mentioned in the paper [21], this method needs an extremely large dataset for showing the final result with high accuracy.

2.4 Semi-supervised Learning

After analysing the PML techniques, bone separation technique using MTANNs, automatic CXR screening methods, the last section of machine learning which will be described is the Semi-Supervised Learning (SSL) used for making diagnosis for the lung diseases based on medical images. The main idea of this technique is to use classification of the pulmonary disease, which concretely uses on the ensemble method that called CST-Voting [25].

The main idea under the algorithm used for this technique is to create a classifier with applying several Semi-Supervised Learning methods on only one dataset. In this paragraph we will look through the research made by I. Liviers, P.Pintelas and their colleagues. The paperwork written by previously mentioned researchers called as "An ensemble SSL algorithm for efficient CXR image classification" [26]. In this research publishers created an ensemble model which contains three algorithms, which are co-training, self-training and tri-training. In the following paragraphs we will look through of all these three algorithms.

Firstly, self-training algorithm is used. This algorithm implements a not complex arbitrary model, which is trained using significantly small subset which contains labelled data L and tries to predict U. After predicting the U, algorithm checks if the probability of predicted new instance is more than the defined confidence level (which is manually defined by the data scientist), it is automatically added inside the labelled data L. This operation is repeating until the provided set of U is empty, which means that all the data in the subset is labelled, or for sure algorithm can stop if there is something triggering its stopping such as various stopping criteria defined by the data scientist.

After checking the self-training algorithm used in the previously mentioned research, now we can look through the second algorithm, which is co-training. This algorithm assumes that the model has weak algorithms that can be trained with the set of labelled instances L. Later, mentioned two algorithms will classify predefined instances of unlabelled set U' and explicitly move them to the L, where the prediction process has the most confident result in comparison of other subsets. After removing samples which were in the subset U', algorithms refill mentioned set with the new instances from the U. This operation also stops as the previous algorithm, significantly co-training algorithm mainly is repeating until the provided set of U is empty, which means that all the data in the subset is labelled, or for sure algorithm can stop if there is something triggering its stopping such as various stopping criteria defined by the data scientist.

The third algorithm used in this technique is tri-training algorithms, which is an extension of the second technique mentioned before (co-training method). Similarly, to the previous co-training approach, the model has plenty of weak learners, but quite more than the co-working algorithm, at least three times increased. Here each provided classifier is trained based on labelled data in L subset and make a prediction of the class for the instances in the unlabelled set. At the end the major part makes the

resulting decision, and the sample which is classified is added to the labelled subset. This method can be viewed as one where "the majority teaches the minority" and where the outcome of the vote determines the final lesson for all students.

The encoded data from the unlabelled data set is fully utilized by those self-labelled methods functioning in various ways. The approach used to identify the unlabelled data is a key component in what distinguishes various strategies. Co-training is a multi-view algorithm, while tri-training and self-training are single-view ones. Tri-training and co-training are also ensemble methods in and of themselves.

The structure and overview of the mentioned technique is represented in the figure 2.4, where the main elements and the workflow of the technique is clearly shown. As you can see, in the first part of the figure, there is a bunch of labelled data, which are stored in a number of subsets. After all labelled data is moved to previously explained and mentioned semi-supervised algorithms, such as co-training, self-training and tri-training). Together labelled data subsets and semi-supervised algorithms build the ensemble learn by using the unlabelled and label data set. Later, the provided decision on an unlabelled sample, combines all separately shown predictions of the created semi-supervised models. Additionally, in the structure there is a not complex voting methodology is used, which significantly helps to make a decision based on the majority of provided votes.

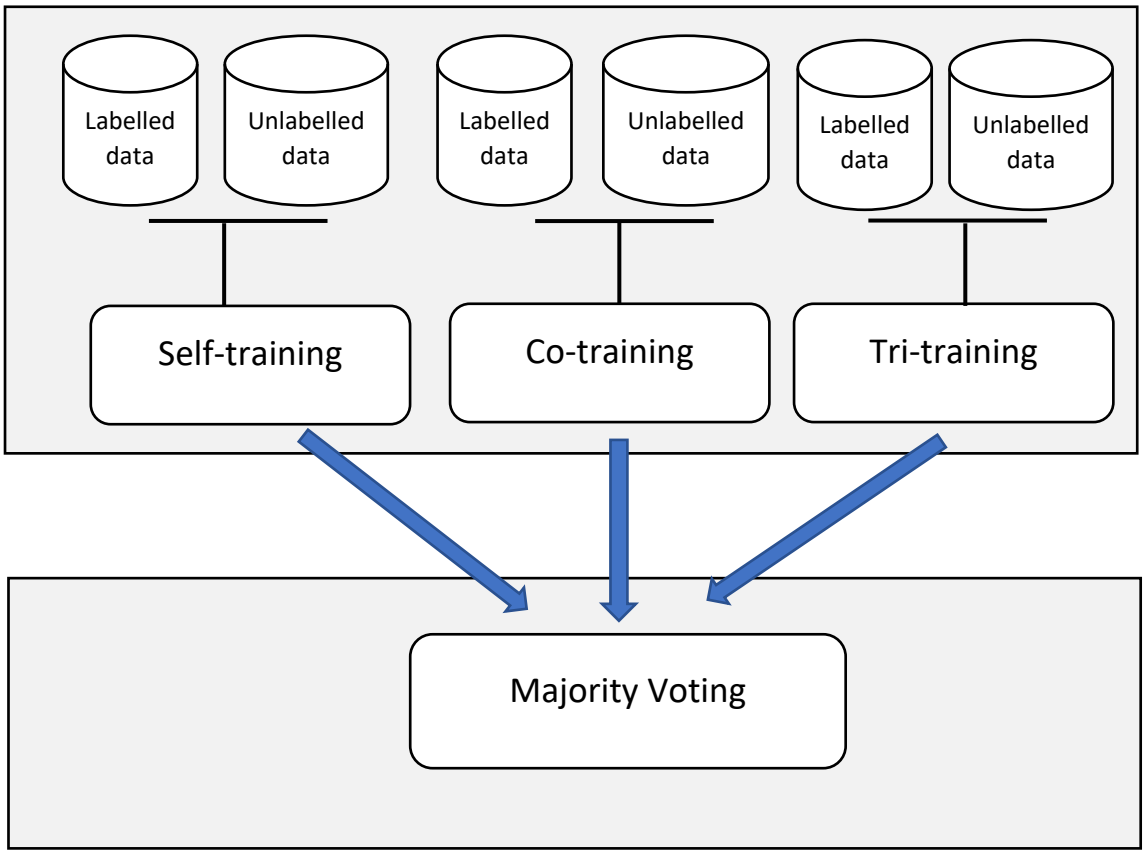


Figure 2.4 CST algorithm's structure

2.5 Automatic diagnostic using Deep learning

In this chapter automatic diagnostic system using Deep Learning will be explained. This technique is used by the researchers, who use this technique in Microscopic Observed Drug Susceptibility (MODS), where MODS is a test which is used for analysing lung diseases, especially the tuberculosis contamination. Despite its benefits, MODS is still only permitted in distant, low-asset environments since its picture-based diagnostics demand constant, trained specialized workers. From this point on, given sound mechanized investigation and elucidation of MODS societies, it is crucial to establish elective arrangements.

The research which will de analysed in this chapter is written by S.Lopez-Garner in the PLOS one journal [27]. In this research, publishers trained and later assessed a deep convolutional neural network for making a test, mentioned before which called MODS for interpretation of medical images and providing the diagnostics. Due to its exact matching of the standard requirements for edge recognition and its ease of implementation, this approach can detect a wide variety of image edges [28]. The proposed segmentation approach is utilized to divide the image into a number of homogeneous classes, and then the improved Canny operator is applied. It is used to define a shallow boundary while the image contains uniform sections and is based on the pixel gradient value. These are the steps that make up this approach. With information on the structure of the lungs collected using segmentation and signs of radiographic images, studies have shown an accuracy of 81.89% in the detection job of tuberculosis and other lung disorders [14]. To examine how segmentation affects the detection of emphysema, however, only data from a segmented part of the lung was used, along with indicators calculated solely in this region.

Based on the information provided in the research [27], data scientist used a large dataset, which contains around 14 thousand digital images for providing a MODS test. Mentioned dataset contains only two classes:

- Positive, images which contains the marks of tuberculosis, especially which had the tuberculosis cord
- Negative, images without tuberculosis cord, which means healthy patient.

During the development process researchers used a network for adaptation, which called VGG16 [28]. A model of 15 layers which contains deep and max-pooling convolutional neural network layers that are fully linked, grouped into five separated blocks of four, and ended by a fully connected layer, which can be called classifier. You can see the structure of the mentioned algorithm in the following Figure 2.5

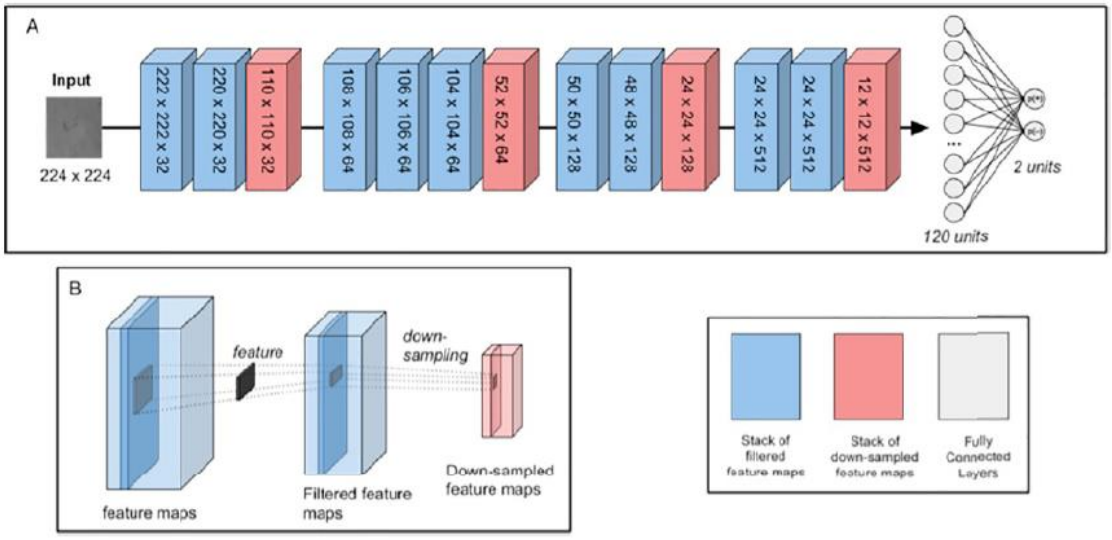


Figure 2.5 Architecture of the MODS image analysis system [27]

As you can see from the figure, there are two main stages of this architecture. In the stage A we can see that there is 224x224 image, which is converted to grayscale medical image of a MODS, because this kind of images is easy for machine learning algorithms to work on. The input image is passed all over the network and as you can see the output of the second layer is without disruption connected layer is a distribution over two opposite classes, negative and positive ones. Each block represented in the figure, which based on the first stage is a collection of the maps containing dimensions such as width times height times the total number of feature maps, simply the number of blocks.

The operations between different layers of feature maps are represented in the stage B. As you can see there are feature maps, filtered feature maps and after down sampling there is a down sampled feature maps. For the convolutional and pooling layers, the kernel sizes are 3 x 3 and 2 x 2, respectively. A dataset consisting of 1126 train/validation and 2957 test images are used to train and evaluate the network. (B) A diagram illustrating the convolution and pooling processes applied to an input volume.

2.6 Summary

In the chapter 2 of this thesis work, previously designed and developed machine learning techniques for medical image processing were analysed. Extreme Learning Machines, Semi-Supervised Learning models, and the Automatic CXR screening system were briefly addressed in this chapter along with another relevant research, which help to understand which type of technology and method is proper for this type of process. Also, advantages and drawbacks of all techniques and developed solutions were described, which will help us to define which technique we can use during the solution of the problem of this thesis work. We also discussed prior deep learning

strategies for analysing chest X-rays, datasets for pulmonary diseases, methods for enhancing picture data, and various forms of classification result assessments. Additionally, the CXR image classification for mobile devices were analysed, however there is not too many examples of this technology. At the end of this chapter Deep convolutional neural networks were briefed, and we described the actions they do when analysing visual data.

Chapter 3

Proposed approach

This chapter represents the proposed approach for mobile CXR classification, which is the main contribution and the topic of this work. In the previous chapters, we analysed previously written publications and developed models for solving this problem. After analysis we define the techniques which can be most convenient for using in this thesis work, based on the positive sides and the drawbacks of each technique that used before. In the following chapters you can find the brief explanation of each technique, methodology and technologies used for developing the machine learning part of the project. As mentioned before, for this work the mobile application will be developed, but in this chapter, we wouldn't touch on that part.

For developing the machine learning approach which can classify X-Ray image of lung for positive and negative cases, we need some techniques and technologies to use. Based on the previously analysed research and ML models we decide to use:

- Convolutional Neural Networks (CNN)
- MobileNet- which is the CNN designed for mobile devices
- Dataset of CXR images with Covid Cases

In the following chapters we will briefly introduce each mentioned technology

3.1 Neural Network

Before explaining what, the Convolutional Neural Network is, it is more appropriate to define what the neural network is. To date, a wide variety of approaches have been accumulated and systematized in the application of statistical and mathematical algorithms for building AI systems, such as Bayesian methods, logistic regression, support vector machine, decision trees, ensembles of algorithms, etc. [30].

Between 2005 and 2008, there was a quantum leap in AI research. The mathematical scientific world began to actively study an approach based on the learning model of multilayer neural networks, which became the foundation for the development of another theory - deep machine learning. And the IT industry began to develop the first application systems based on these approaches and actively study them.

Recently, a number of foreign experts have come to the conclusion that most modern and really successful implementations are solutions built on the technology of deep neural networks and deep machine learning [31].

Neural networks are based on an attempt to recreate “a simplified model of nervous systems in biological organisms. In living things, a neuron is an electrically excitable cell that processes, stores, and transmits information using electrical and chemical signals through synaptic connections. The neuron has a complex structure and narrow specialization. By connecting with each other to transmit signals using synapses, neurons create biological neural networks. In the human brain, there are on average about 65 billion neurons and 100 trillion synapses [32]. In fact, this is the basic mechanism of learning and brain activity of all living beings, i.e. - their intelligence.

An innovative technology known as artificial neural networks was developed at the nexus of neurophysiology, mathematics, and informatics. They have shown to be a successful approach for resolving forecasting, classification, and pattern recognition issues that are difficult to formalize. Artificial neural networks' architecture and method of information processing are key characteristics. Adaptability is the key characteristic of neural networks.

Thus, neural networks that have been trained to behave in a certain context can be retrained and modified to function when environmental factors are fluctuating. The system will operate more steadily in a non-stationary environment the higher its adaptive capacity. For issues with image categorization, signal processing, and control, the concept of adaptability makes sense.

When it comes to the issue of picture classification, one can create a neural network that gathers data not only to identify a particular class but also to improve the accuracy of the judgment. The productivity of neural networks will rise as a result of using this knowledge to get rid of dubious choices. The activation state of the neural network is used to express knowledge in the network's own structure. All of the network's neurons have the ability to affect each of its neurons. As a result, contextual information has a direct impact on whether a neural network exists.

Due to their excellent fault tolerance, neural networks. Their performance declines under difficult circumstances, but only slightly. For instance, if a neuron or one of its connections is broken, it will make it harder to recover stored information, but it won't have a disastrous effect. The type of damage also matters greatly in this situation. Therefore, given the dispersed nature of information storage in a neural network, it may be claimed that severe damage to the network's structure will considerably affect how well it performs.

An input layer, one or more hidden layers, and an output layer make up an artificial neural network (ANN) node layer. Every artificial neuron or node is interconnected with others and has the proper weight and threshold. Any node whose output rises beyond the predetermined threshold is activated and begins sending data to the network's top level. If not, the data is not forwarded to the network's next level. Figure 3.1 depicts the simple artificial neural network's fundamental design.

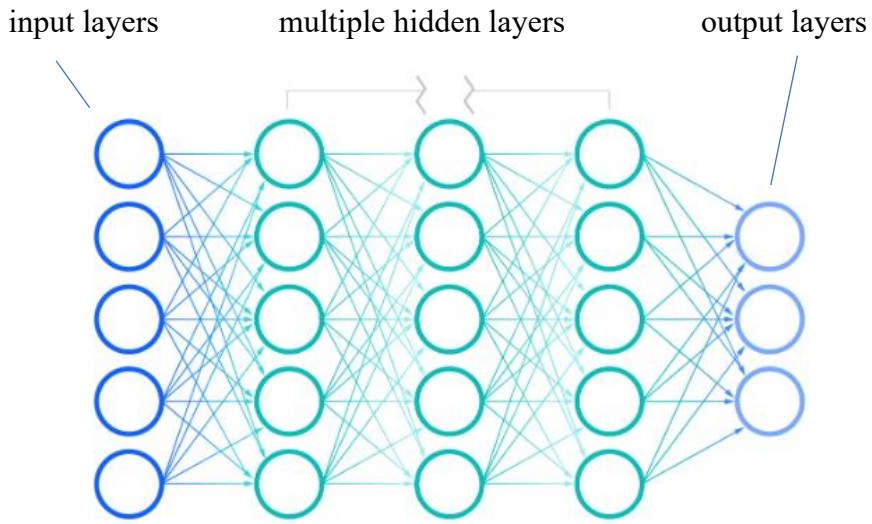


Figure 3.1 Simple architecture of Deep Neural Networks [33]

In order for neural networks to develop and become more accurate over time, training data is necessary. However, if they are adjusted for accuracy, these learning algorithms transform into potent tools for computing and artificial intelligence, enabling us to classify and organize data quickly. Speech or picture recognition tasks can be finished in minutes rather than hours, compared to manual identification by human experts. One of the most well-known neural networks is used in the Google search algorithm.

There are nine main artificial neural network types, where each of them has its own techniques to create and its own area of usage. Based on the proposed problem, data scientist can use different neural networks for gaining the sufficient result. In the following paragraphs you will find some main types between the nine types of NN mentioned by [34] and a brief explanation of them:

- **Perceptron**- it is one of the oldest and also the simplest model of the neutron. Perceptron is the smallest unit of neural network which can do and represent various computations, for sure not so complex ones, for detecting the features or business intelligence which is taken from the input data. This type of neural network accepts the weighted inputs and apply some functions mainly activation function for obtaining the output as the result.
- **Feed Forward Neural Network -** The most basic kind of neural network, where inputs only flow in one way, passing via artificial neural nodes and out through output nodes. Wherever hidden layers may or may not be present, input and output layers are present. They can then be classified as single-layer or multi-layer feed-forward neural networks depending on this. The number of levels in a function is determined by its degree of difficulty. It stretches in one direction forward but not the other.
- **Multilayer Perceptron –** with speech recognition, machine translation, and some complicated categorization, this method is quite effective. Every node in a multilayer perceptron, which has many more layers than a single perceptron and is completely coupled because of this, is tightly connected to every other neuron in

the layer below. Since this neural network has a lot of connected layers and back propagation, it can be utilized for deep learning.

- **Recurrent Neural Network –** these particular neural networks are designed to save a layer's output and transfer it to the input in order to aid in the layer's prediction. A feed-forward neural network is frequently the initial layer, followed by a recurrent neural network layer in which the memory function recalls some of the data it had in the previous time step. Direct propagation is employed here. Save the required information for later use. Small adjustments are made using the learning rate if the prediction is off. Following that, gradually raise it so that it produces an accurate prediction during backpropagation.
- **Convolutional Neural Network -** Instead of the usual two-dimensional array, convolution neural networks have a three-dimensional layout of neurons. Convolutional layer refers to the top layer. Only a small portion of the visual field is processed by each neuron in the convolutional layer. Like a filter, input features are gathered in batches. Even if it only completely understands the images in portions, the network may repeat these actions to complete the entire image processing. During processing, the image is changed from RGB or HSI scale to greyscale. The detection of edges will be aided by furthering the pixel value fluctuations, enabling the classification of images into several categories. CNN is frequently used in machine translation, speech recognition, computer vision, and image processing. The main benefit of CNN over other neural network types is that it uses a significantly less number of parameters for deep learning than fully connected layers.
- **Modular Neural Network** – this type of neural network consists of a number of distinct networks that each carry out a specific task. Throughout the calculation process, the various networks do not really communicate with or signal one another. They each contribute separately to the outcome. You can see the architecture of Modular Neural Network in the figure 3.2

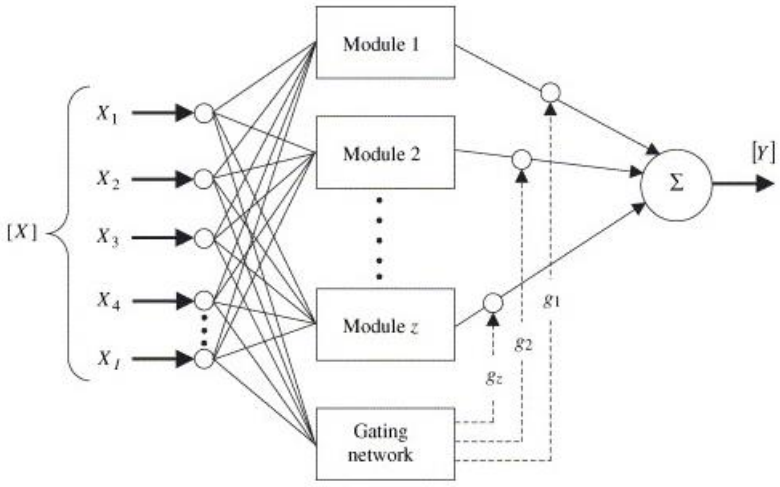


Figure 3.2 Architecture of Modular Neural Networks

Therefore, by dividing a complex computational process into separate parts, it can be completed much more quickly. Because the networks are neither connected nor even

communicating with one another, the computing speed increases. This type of neural networks is manly used for stock market prediction systems, compression of high-level input data and the Adaptive modular neural networks for the character recognition. The advantage of MNN is that this type of NN is more efficient, can perform independent training and also, they are more robust than other types.

After analysing main types of neural networks, checking their advantages and disadvantages, also looking through the previously created technologies and published papers in the second chapter of the thesis, we define that using Convolutional Neural Networks will be the best solution for creating the topic of this topic. Based on this, in the following chapter we will briefly explain the technology which stands behind CNN.

3.2 Convolutional Neural Networks (CNN)

The main technique and Machine Learning architecture which will be used for developing the solution for solving the main question of the thesis, will be Convolutional Neural Networks (CNN). The selection of this network was done based on analysed papers and developed applications, on the second chapter of this thesis, and after checking advantages and disadvantages of each Neural Network type on the previous chapter.

The CNN are Multilayer Perceptron (MLP) architectures that mimics and simulates human processing in the cortex of the brain, but mainly in the Visual Cortex. The Convolutional Neural Networks are one of the best techniques and has the best features which help to easily process the classification of image content. For the past years this method is mainly using during the classification of medical images, and that's why we decided to use that technique for this thesis. The design of this networks was inspired by the work of Wiesel and Hubel who were scientist and neurologist [35]. Nowadays these applications are extremely popular between the usage in deep learning tasks such as object tracking, image classification, text detection, text recognition, scenario classification and others. This type of neural networks has a great capacity for extracting different features with high and medium levels of abstraction from the input data which are mainly image data. Similar to other Multilayer perceptron architecture CNN have input layer, which are normally medical or other type of images, plenty of hidden layers and one output layer. Hidden layers of convolutional neutral networks contain three different types, such as:

- Convolutional Layers- main purpose of these layers is to extract the features from the input images.
- Pooling Layers (Downsampling) – these layers can affordably reduce the dimensionality of the network.
- Fully Connected Layer (FCL) – used for determining the output of the whole model.

In the figure 3.3 you can see the architecture of CNN which created for making diagnosis from chest X-ray images, which is quite similar to the topic of this thesis work. In architecture layers of CNN, especially the convolutional layers have the main purpose for extracting the features, where pooling layers can affordably reduce the dimensionality of the network. The last layer which stands at the end of the neural network, are used to determine the output of the whole model, by connecting all separate outputs of different layers and at the end making them use of an activation function.

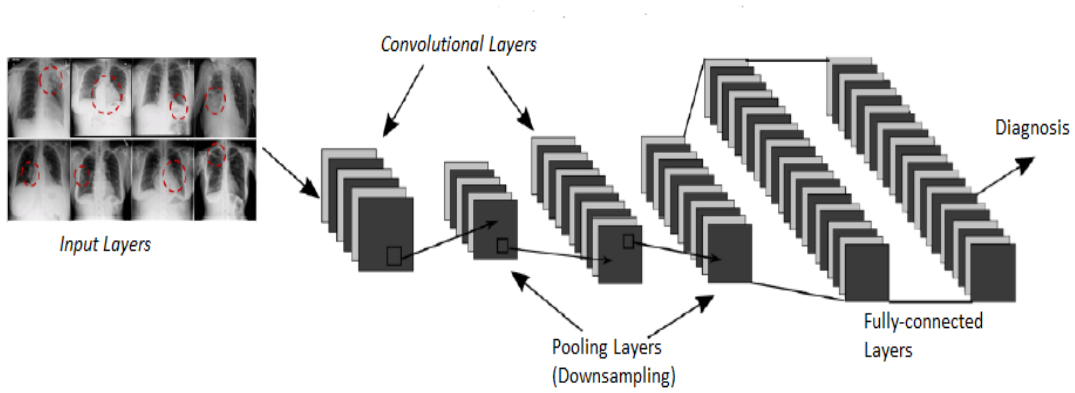


Figure 3.3 Architecture of CNN for CXR diagnosis

The input images in image classification tasks, are firstly treated as matrices. These images have various length and width of the matrix based on the predefined dimensions of the image, where the depth is determined related to the number of colour channels that form it.

One of a CNN's fundamental components, the convolutional layer is where the majority of the computations take place. Non-linear filters in this layer sequentially process the input data to create feature maps, which are matrices. These filters are automatically changed during training so that they turn on in the presence of pertinent features, such as the direction of edges, lines, or colour spots. Each convolutional layer is subjected to filters, and the feature maps are then layered to create a matrix with one additional dimension than the original dimension of the pictures.

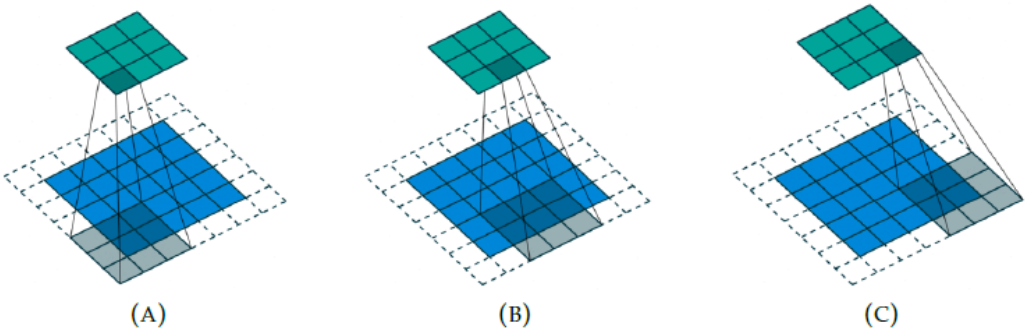


Figure 3.4 Convolutional Operation

In the figure 3.4 the convolutional operation is represented, which means the operations inside the convolutional layer of the model. You can see these operation extracts valuable input features and different functions and processes this data to the next high level, whereas it can reduce the dimensionality of the input images. Convolution can be viewed as a sequence of different functions and where single operation centred on a group of results with single output.

The outputs of the convolutional layers have the drawback of recording the precise location of the features in the input images. This means that a slight adjustment to the position of the input picture feature will produce a different activation map [37]. The image may have been cropped, moved, or rotated to cause this change in input.

The dropout is another crucial idea in the context of CNN. Dropout is a regularization technique that involves bringing back in a portion of the neurons that were randomly "shut down" after each training iteration of a layer. The network can get additional knowledge through this method. Since a neuron cannot be dependent on the precise presence of other neurons, it must possess resilient qualities. All the neurons in the FCL, which are dense layers, are linked to every neuron in the anterior layer. The FCL classifies the features collected in the convolutional and pooling layers, and at the conclusion of the process, an activation function—typically a Softmax—is used to forecast the input image's class.

The Convolutional Neural Networks are extremely popular in solving the medical image classification problem. In the next paragraphs we will describe some previously created solutions for this problem.

A strategy for categorizing chest X-ray pictures using MobileNet, a CNN model that was previously trained on the ImageNet dataset, is suggested by researchers of the paper [38]. The approach is used by the authors to diagnose the infectious disease tuberculosis, which targets the lungs. It chose 19 of the best features for Dataset 2 and 25 of the best features for Shenzen. According to the paper's findings, Dataset 2's classification accuracy was 89.1% and Shenzen's was 88.2%.

In order to classify x-ray pictures from three different patient types—common bacterial pneumonia, COVID-19, and typical incidents [39] used CNN architectures VGG19 and MobileNetV2. The study examined two datasets and found that MobileNet V2 had higher classification accuracy than VGG19 for a given classification job on Dataset 1. For Dataset 2, MobileNet V2 was able to identify COVID-19 cases from other dataset cases.

After analysing proposed solutions for medical image classification which is using the Convolutional Neural networks for mobile devices, we define that the best choice is with using the MobileNet network. This technology will be described in the next chapter of thesis.

3.3 MobileNet

For image classification with the best quality, it is necessary to solve three main tasks and problems, such as creating or finding a good dataset which contains input data, create adequate architecture of CNN and provide the learning for created mobile. As our thesis's main question is to create image classification solution for mobile devices, we should find a best Convolutional Neural Network which is appropriate for the mobile devices, which means for the devices with low power processors, rather than computer which has partly good computational power. There are plenty of solutions for mobile devices but most popular between these networks are VGG-16, Inception v3, ResNet-50, MobileNet.

With the purpose of finding the best convolutional neural network for mobile devices the researchers of Information Technologies Academy in Belarus trained models with the same dataset using neural networks mentioned before [41]. The researchers and professors of IT academy used simple dataset with only thousands of images, where 20 percent of all images are in the test dataset. After realized analysis on trained model, publishers represent the table of the results (Table 3.1).

Name of the model	Number of network parameters	Accuracy (%)	Time of processing of one frame, s
VGG-16	135	62.6	2.4
Inception v3	24	88.2	1.6
ResNet-50	22	70.6	0.8
MobileNet	5	86.7	0.2

Table 3.1 Results of the accuracy assessment of CNN for mobile devices

In this table the results of the provided analysis on different CNNs which are used for mobile devices, are shown. It is obvious that there are two main mobile networks which has comparatively high accuracy percentage than others (Inception v3 and MobileNet). But the time of processing one frame of Inception v3 model is almost 8 times more than MobileNet, and also number of network parameters are 5 times higher. After, checking these results, we can easily define the best option between all the mentioned ones, with high accuracy, lowest time of processing and lower number of network parameters- which is MobileNet. In the following paragraphs of this chapter will be the brief explanation of MobileNet neural network.

Applications for embedded and mobile vision use convolutional neural networks, such as MobileNet. These lightweight deep neural networks with depth-wise separable convolutions, which can have low latency for embedded and mobile devices, are used to build them. The MobileNet model is the first mobile computer vision model for TensorFlow and is designed for use in mobile applications, as its name suggests. MobileNet employs depth-wise separable convolutions. It significantly lowers the number of parameters as compared to a network with traditional convolutions of the

same depth in the nets. The result is a class of deep neural networks that are lightweight.

Two processes are used to create a depth wise separable convolution.

- Convolution on a depth basis.
- Convolution done in points.

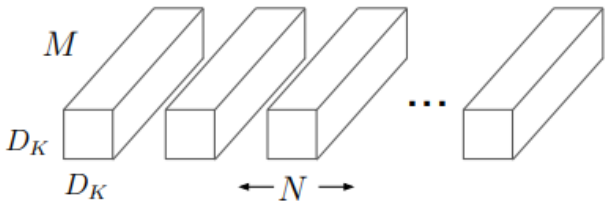
This offers us a great place to start when training our classifiers, which are ridiculously small and unbelievably quick. MobileNet is a class of CNN that was open sourced by Google [42]. The foundation of MobileNets is a set of depth-separable convolutional layers. A depthwise convolution and a pointwise convolution make up each depthwise separable convolution layer.

A MobileNet contains 28 layers if depthwise and pointwise convolutions are counted separately. By properly adjusting the width multiplier hyperparameter, it is possible to further minimize the 4.3 million parameters that make up a normal MobileNet. In the following table (Table 3.2) you can see the detailed architecture of MobileNet.

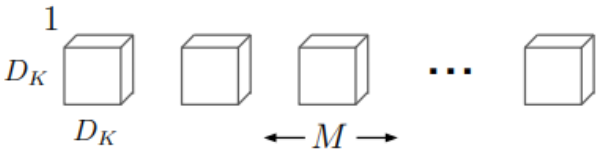
Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Table 3.2 Detailed architecture of MobileNet [43]

The MobileNet model is based on convolutions which called with depthwise, that is a form of factorized convolution, which were explained in the previous chapter. The foundation of the MobileNet model is convolutions with depth wise, which factorize a normal one to another form which is a depthwise convolution and an 1x1 convolution known as pointwise one. The explained type of convolution for MobileNets mainly applies only one filter for each input group. The outputs of the convolution are combined by the pointwise convolution using an 1x1 convolution. Standard convolutions combine inputs into a new result set in one step while also filtering the inputs.



(a) Standard Convolution Filters



(b) Depthwise Convolutional Filters

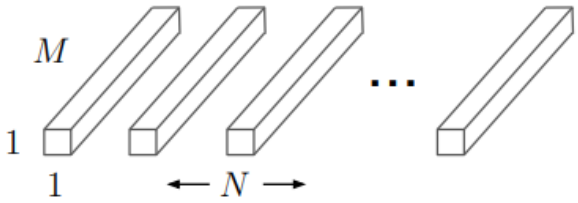


Figure 3.5 Layers of MobileNet network

The architecture is divided into two separate layers by the convolution of depthwise type, which were explained before: a layer for combining and a layer for filtering. The computation and model size are significantly decreased as a result of this factorization. In the Figure 3.5 you can see the layers of MobileNet network, where the factorization of a standard convolution shows at 3.5(a) into a depthwise convolution 3.5(b) and a 1x1 pointwise convolution 2.

The analysis provided in 2017 represents the introduction of the MobileNet class of CNN, according to [43] Its compact network design enables quicker and simpler integration into mobile device apps. Additionally, MobileNet is presented by Togaçar, Egen, and Cömert [44] as a deep learning model to be employed in low-cost hardware devices. MobileNet's Depthwise Separable Convolution technique has lowered the computational difficulty of the system.

Deep convolution network architecture is present in the MobileNetV2 FPN Lite SSD itself. Despite not having the same computing capability as general-purpose computers, these application platforms nonetheless demand high probability and timely difficulties. A depth-separated bottleneck convolution from base blocks with residuals is evaluated using a Rectified Linear Unit (ReLU) activation function in the MobileNetV2 FPN Lite SSD [40].

Figure 3.6 Possible applications of MobileNet in different recognition tasks

In the figure 3.6 the possible applications of MobileNet are shown. As you can see from the image, with MobileNet it is possible to do Object detection, face detection, fine grain classification, landmark recognition, image classification and plenty of other possible applications. In this thesis we will use MobileNet convolutional neural network, for realising the medical image classification with the purpose of making diagnosis from chest X-ray images.

But there are 2 versions of MobileNet neural networks. Before starting to use this technology, we could understand which of these two versions are better, which has the highest accuracy of the prediction. In the next chapter of this thesis, both types of MobileNet networks will be discussed, and the best one will be chosen for the future use in this thesis.

3.4 Comparison of MobileNet versions

Before using the MobileNet architecture in the proposed solution for this thesis work, we will look through different versions of this architecture. There are two main versions of MobileNet network and one new version which is still in beta version, so we wouldn't check that version. In this chapter we will check MobileNet v1 and v2, briefly explain the main concept of these technologies and compare their computational power and accuracy on proposed output. This comparison of two MobileNet versions is provided in the research paper which was published in the Journal of Physics: Conference Series [45].

Modern designs have a lot of depth; therefore, they consume a lot of memory and also the long processing time. Additionally, the majority of CNNs require top-notch graphics equipment to train and render the large amount of input images inside multiple layers. The MobileNet network can only operate on a smaller number of devices as it becomes deeper because it uses more memory and computation. Google researchers proposed MobileNet V1 and MobileNet V2, respectively with some years of difference between them. These two versions are comparably accurate to the heavier variants despite being fairly light. Despite the similarities between both models, MobileNet V2 employs more modern techniques than ResNet network which is counted as the previous model which were popularly used for machine learning solutions in mobile devices [46].

In the research paper [45], researchers developed algorithm which is using pre-shot videos and images to train the model. In the train dataset there are various type of captured videos where all possible problems were encountered which can created during installation of video devices on a vehicle, because this algorithm was designed for pedestrian detection task on the mobile devices which were installed to the bicycles or motorcycles. The entire videos were captured from the camera of mobile device (Xiaomi Redmi 8 Pro) with 720p and 30 Frames Per Second (FPS) of video settings. The duration of videos in the proposed dataset is about 10 seconds with the resolution of screen 720 x 1280px. Also added images to dataset has the same resolution of the screen. Later implemented algorithm was concretely applied to all videos and images of the dataset. Researchers of the analysed paper then have a variety of videos, some of which may or may not include pedestrians in them. A percentage of recognition is displayed over each pedestrian that is spotted reliability.

In the research [45] publishers were defined the accuracy graphs will be displayed, per each MobileNet version architecture, both on distance 5 meters and 10 meters, but in this thesis work we will display the result graph only for 5 meters approach.

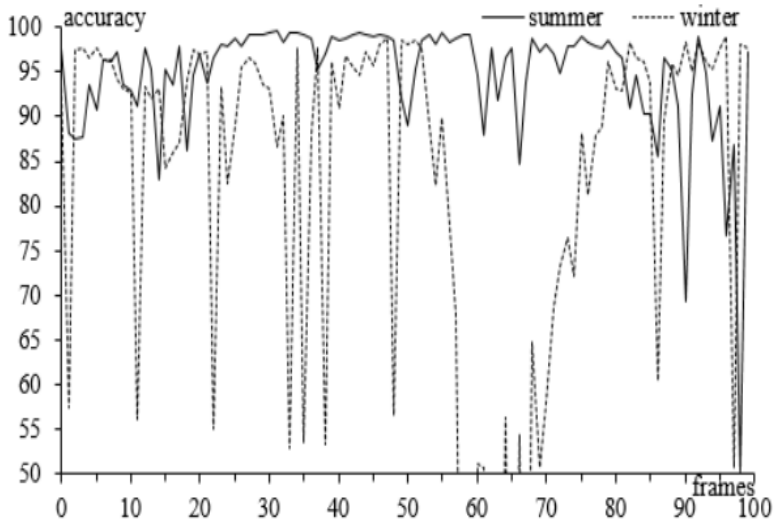


Figure 3.7 Graph of frame-by-frame recognition accuracy of the MobileNet V1

In the figure 3.7 the accuracy graph of the MobileNet V1 performance is represented. Used video frames and images for this representation, were taken and captured in 5

meters distance from the person. Also, the data scientists who wrote this paper, took into consideration the seasons, especially summer and winter. From the previous figure it's clearly seen that the solution which uses MobileNet V1 provides very low accuracy output in the winter, in some frames the accuracy is almost zero. But accuracy percentage in summer is affordable high. The time-lapse accuracy graphs show that the MobileNet V1 model performs significantly poorer at identifying a running pedestrian in winter at 5 meters. In the summer, the average recognition accuracy for recognition at a distance of 5 m is 88.7%, whereas in the winter, it is 76.8%.

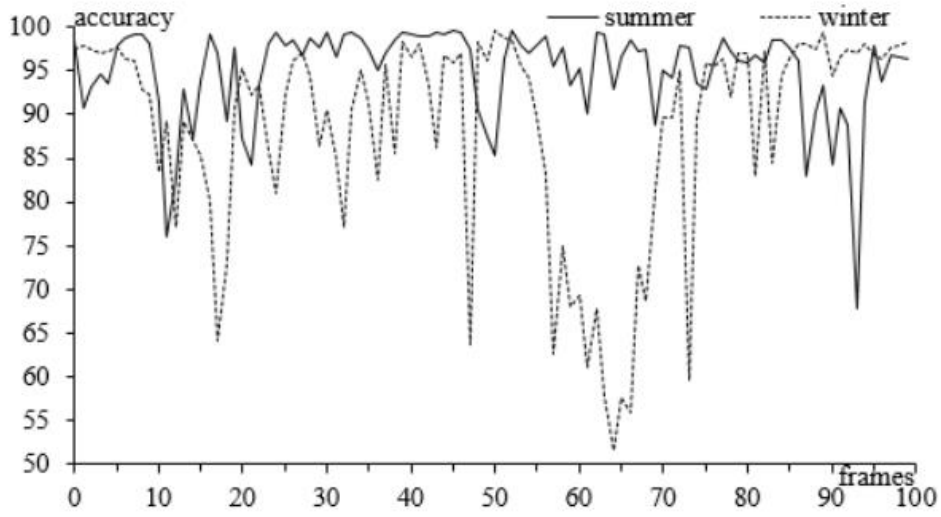


Figure 3.8 Graph of frame-by-frame recognition accuracy of the MobileNet V2

In the figure 3.8 you can see the accuracy graph, which based on frames used in the MobileNet V2 model. Used video frames and images for this representation, were taken and captured in 5 meters distance from the person, as in the previous graph which were used for MobileNet V1. Also, the data scientists who wrote this paper, took into consideration the seasons, especially summer and winter. From the previous figure it's clearly seen that the solution which uses MobileNet V2 provides very low accuracy output in the winter, in some frames the accuracy is almost zero. But accuracy percentage in summer is affordable high. The time-lapse accuracy graphs show that the MobileNet V1 model performs significantly poorer at identifying a running pedestrian in winter at 5 meters. In the summer, the average recognition accuracy for recognition at a distance of 5 meters is 94.7%, whereas in the winter, it is 87.3%.

Taking into consideration the results of qualitative analysis of accuracies of solution using both network architectures, we can clearly provide a conclusion, and define the better version of MobileNet neural networks. At a distance of 5 m, both models very well detect a running pedestrian in the summer. However, the recognition is less useful in winter shoots. For MobileNet V1, where the average accuracy of summer and winter vary by over 14%, the differential is particularly significant. With regard to MobileNet V2, the average accuracy of winter frames has decreased by 6.6%. The average accuracy of winter frames was found to be 6.3% higher than summer ones with a distance increase to 10 m. At a distance of 10 meters, MobileNet V1's average recognition accuracy for winter frames was a mediocre 57.8%.

Summing up, we can say that the model which use MobileNet V2 provide significantly better results in comparison to the model with MobileNet V1. So, for the solution of this thesis, MobileNet V2 will be used.

3.5 Datasets of CXR images

In the previous chapters the main technologies which are necessary for creating the ML solution for CXR image classification were discussed. Now we define that for solving the main question of this thesis, we will work Convolutional Neural Networks, and as we are creating the mobile solution, we will use MobileNet architecture, especially the MobileNet V2, as we already analysed the comparison between two versions of this architecture. But for training the created model it is necessary to have a dataset with CXR images, with both positive and negative Covid cases, or pneumonia cases, so that the proposed solution could classify images and make diagnosis. In this chapter we will look through available datasets which are suitable for this work and choose the most convenient one for future use in this work.

Name	**Provided by**	**Type of Images**	**Total number of images**
NIH chest X-ray dataset	NIH (National Institutes of Health)	Chest X-Ray (PNG)	100.000
COVID-19 Radiography Database	Researchers from Qatar University	Covid, Viral Pneumonia, Normal (DICOM)	3618 Covid 6010 Pneumonia 10.136 Normal
RSNA Pneumonia Detection Challenge	RSNA (Radiological Society of North America)	Pneumonia, Normal (DICOM)	16,216 posteriors, 13,628 anterior, 4,537 tests
COVID-19 Image Data Collection	Cohen et al.	Covid Positive, Normal (DICOM)	123
Chest X-Ray images (Pneumonia)	Guangzhou Women and Children's Medical centre	Bacterial Pneumonia, Viral Pneumonia, Normal (JPEG)	5863
Synthetic COVID-19 Chest X-Ray Dataset for Computer Aided Diagnosis	Cornell Medical University	Covid-Non-Covid (DICOM)	21295

Table 3.3 List of Datasets of CXR image classification

In the table 3.3 there are six different datasets, which contain CXR images, which can be used for training the model that solves the image classification tasks, exactly the medical image classification task. There are plenty of other datasets, but for this thesis

work we will look through of these 6 datasets, and based on pros and cons, we will choose the one for using during the training of our model. In the next paragraphs you can find the brief description and content of each dataset and also the comparison of them.

In the first place there will be a discussion of chest X-ray dataset provided by "National Institutes of Health (NIH)" [47]. This dataset provided by NIH in 2017 contains about 100 thousand of anonymized chest X-ray images. All the images which are in this database are in PNG format, which can be seen as an advantage, because mainly medical images have DICOM format, which helps these images to store some medical data also, but for using in image classification there is necessity to convert them to PNG format, so it is not needed for this dataset. This dataset contains chest X-ray images, which have completely various labels, such as Cardiomegaly, Hernia, Nodule, Effusion, Infiltration, Atelectasis, No finding and some other possible labels. In this dataset there are also some additional columns(data) such as patient's ID, age and gender, which can be useful during the prediction process. There are some drawbacks of this dataset, such as:

- The number of images is unnecessary high, which can lead to increase the time of training process and computational cost of the whole process. For our solution, having 100 thousand image dataset is not so useful.
- This dataset has plenty of various labels, which represents the diagnosis of the patient based on the chest X-ray. In our case having too much various labels for medical images are useless, as we need only covid cases, or having the pneumonia can be useful.
- As the dataset is quite old, it was published in 2017, so there is not any covid positive or negative possibilities, because at that time there were no such people infected to that virus, and it was not spread that much.

Taking into consideration, the drawbacks of this dataset, we can say that it is useless to use that dataset for training our model, because of its size (number of input images), absence of covid positive and negative labels of images, and absence of X-Ray images which were taken from the patients with Covid positive. So, this dataset wouldn't be used for the solution of this thesis.

The next dataset which will be analysed is 'Covid-19 Radiography Database' collected by researchers of Qatar university and the winner of the Covid-19 dataset award by Kaggle Community [48]. In 2021 a team of researchers from Qatar University with their collaborators from Malaysia and Pakistan who had the connection and collaboration with medical doctors, they created a chest X-ray dataset which contains Covid positive, Viral pneumonia and Normal image cases. There are about 19600 of images in total in the dataset, where 3618 of Covid cases, 6010 of viral pneumonia cases and the rest which is around 10 thousand are normal X-ray images. This dataset was the winner of 'COVID-19 dataset award by Kaggle community' [48], which is the powerful evidence of that, this dataset can be used in our solution for this thesis work. There are some advantages of using this dataset:

- Affordable number of images in dataset, which means that there are not too many images as before, which can increase the time of the process, and there are not too

few numbers of images, which can be not so useful during the training process of the model

- Contains Covid positive cases, together with viral pneumonia cases and normal patients. Having covid positive cases in that number, is completely useful and necessary for our solution. Also, having viral pneumonia cases is an advantage, as the model would be aware of differentiating the patients with Covid positive or having only the pneumonia
- As the dataset was the winner of Covid 19 dataset award by Kaggle, it means that it was significantly useful for other data scientist who work on the solution of medical CXR image classification. So, this dataset could be the good solution for using in the solution of this thesis.

Next dataset which will be discussed and analysed is the Chest X-ray images that provided during the challenge provided by "Radiological Society of North America (RSNA)", for pneumonia detection [49]. They were collaborated with colleagues from the Society for Thoracic Radiology and MD.ai to identify pneumonia cases in the National Institutes of Health's collection of chest x-rays (NIH). This dataset contains about 30 thousand of frontal view chest radiograph images which were taken from the public NIH dataset which was discussed in the beginning of this chapter. There are 16 thousand posterior images, 13 thousand anterior images and four thousand of test images. The Dataset mentioned in this paragraph contains two labels, such as pneumonia present, pneumonia absent. The dataset provided for the challenge with pneumonia detection topic has some drawbacks, such as:

- This dataset contains only the pneumonia cases, without having any data for covid cases. Having only the pneumonia and normal cases for medical images are useless, as we need only covid cases, or having the pneumonia can be useful but there should be combination with another dataset which contains the images of covid cases.
- As the dataset is quite old, it was published in 2018, so there is not any covid positive or negative possibilities, because at that time there were no such people infected to that virus, and it was not spread that much. This drawback is quite similar to the dataset of chest x-ray images provided by NIH that were analysed in the beginning of this chapter. So, this dataset wouldn't be the best solution for using in the solution of this thesis work.

The next dataset is "Covid-19 Image Data Collection" provided by Cohen. J Paul [50]. This dataset was created by three scientists firstly for their own commercial purpose, but later they published also for future usage. The dataset was created by assembling plenty of medical images from different websites and publications. There are two main labels on this dataset, which are Covid positive and normal one. This small dataset has some drawbacks, which avoid us to use it during the development of solution for this thesis work, such as:

- This dataset is quite small. With dataset which has just hundred of X-Ray images, it is too hard to train the model successfully and at the end the accuracy of provided predictions will be too low.

- The images used in this Dataset are not with official license, so we don't know from which source the researchers took that image. So, it is not so convenient to use images that are not licenced, and with anonymous providers, it can be risky in terms of copyright and other stuff. So, we wouldn't use this dataset during the training of developed model for solving the main question of this thesis.

The next dataset which will be discussed now is "Chest X-Ray Images (Pneumonia) Dataset" which contains medical images provided from Guangzhou Women and Children's Medical centre [51]. This dataset has both posterior and anterior chest X-Ray images, which were divided into three groups, which means that there are three labels in dataset, such as:

- Normal – the chest images where there are not any areas with abnormal opacification in the image, which means that patient doesn't have any pneumonia detected from their X-ray. The Normal chest X-ray image which doesn't have any abnormal opacifications, shown in the figure 3.9(a).
- Bacterial pneumonia – if the chest images have focal lobar consolidation, as shown in figure 3.9(b), as you can see on the second image in the right upper lobe, where the arrow is represented.
- Viral pneumonia – in the case of viral pneumonia there is more diffuse pattern in both lungs, as shown in the figure 3.9(c).

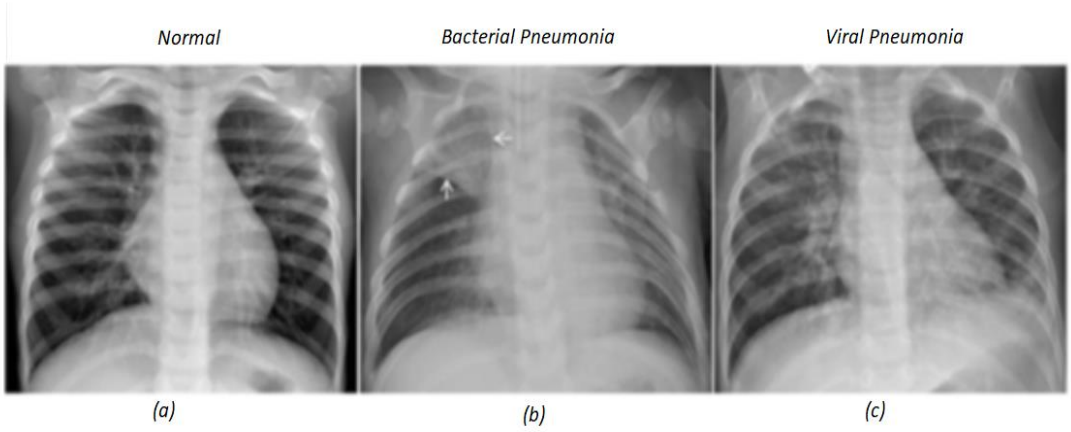


Figure 3.9 Sample images for all labels of the dataset [51]

In total in this dataset consists of 5.866 X-Ray images in all three provided folders. There are three folders in dataset, which are train set, test set and validation set. The medical images in this dataset have the JPEG format, which can be counted as advantage, as we don't need to change the format of each image. All chest radiographs were initially checked for quality control before being removed from the study of the chest x-ray pictures. Before the diagnosis for the photos could be used to train the AI system, they were graded by two experienced doctors. A third expert also reviewed the evaluation set to make sure there were no grading mistakes. This dataset is quite powerful in terms of provided images, but it has some drawbacks which avoid the usage in the solution for this thesis:

- The number of sample images is still not enough for successfully training the model, also the most percentage of total images are normal cases, which also

reduce the possibility of having high prediction accuracy after training the model and having the final output.

- This dataset is quite old and doesn't have any images related to positive or negative Covid cases, there is only bacterial and viral pneumonia. For the researchers or data scientists who want to classify the types of pneumonia, this dataset could be the best option. So, in our case using this dataset is not convenient.

The last dataset which will be analysed in this chapter is the 'Synthetic COVID-19 Chest X-Ray dataset for computer aided diagnosis' [52]. This dataset was provided by researchers Hasib Zunair and Ben Hamza in collaboration with Cornell Medical University. This dataset consists of more than 21 thousand of synthetic Covid-19 X-Ray images which is defined to be used for computer-aided diagnosis. These high-quality photos were produced using an unsupervised domain adaptation method. When employed as supplementary training data for deep learning architectures in highly asymmetrical conditions, we discover that synthetic images not only enhance performance but also accurately identify the target class. We also discover that when trained exclusively on artificial images, equivalent performance may still be attained. Additionally, striking characteristics of the artificial COVID-19 images show that the distribution differs greatly from non-COVID-19 classes, allowing for an appropriate choice boundary. We anticipate that the availability of such detailed chest X-ray pictures of COVID-19 will promote improvements in the creation of management and/or diagnostic tools. This dataset has some advantages which can be useful in the solution of this thesis, such as:

- The number of images in the provided dataset is quite enough for training a powerful model, which will have a high enough percentage of accuracy at the end, also the covid and non-covid samples divided almost equally, so at least 10 thousand samples for positive covid cases and 10 thousand for the negative ones, which can be seen as normal.
- Also, in the comparison of previously analysed datasets, this one has the greatest number of images related to positive and negative covid cases, which can be the most convenient to the model which will be designed for solving the main question of this thesis.

To conclude the dataset comparison and selection chapter, we will define the dataset that will be used during the training of the model which will be created for this thesis work. After making a comparison of previously defined datasets it is obvious that there are only two datasets, which had some advantages and can be convenient for the solution of this thesis work. Two best datasets are "COVID-19 Radiography Database" by researchers of Qatar University [48] and "Synthetic COVID-19 Chest X-Ray dataset for computer aided diagnosis" by Cornell medical university [52]. So we decide to use the combination of these datasets, to having the powerful dataset which will consist not only covid cases but also the pneumonia cases. The usage and combination of these datasets will be discussed in the following chapter.

3.5 Summary

In the third chapter of this thesis the possible technologies which can be used for developing an image classification task. Firstly, the brief explanation of neural networks was provided, the available types of neural networks are shown. After providing the comparison of mentioned networks, as a main architecture for developed solution were selected one, which was Convolutional neural networks which will be used for solving classification task. Then the brief explanation of Convolutional Neural Networks was presented. After explaining the main purpose of convolutional networks, we define and analyse the best architectures for mobile devices. The MobileNet was the winner between others. Later the comparison of two versions of MobileNet was provided and the MobileNet V2 was chosen. After defining the main technologies for creating the ML solution, we define the six different datasets, which contain the medical CXR images. Analysis and comparison of all datasets leads the selection of two datasets, which will be combined in the future use.

Chapter 4

Developed Model

The chapter 4 of thesis work will describe the Machine Learning model developed for the solution of the main question of thesis. Firstly, there will be brief explanation of the methodology and structure of developed model, the development steps will be described. Later you can find the details of collecting the data from analysed datasets and pre-processing of this data. After explaining everything related to data you can find the full logic of the created model and the ways to training it. At the end of this chapter there will be performance evaluation of developed algorithm and trained model.

4.1 Methodology

In this chapter you can see the main steps which were followed during the development stage of the Machine Learning model for solving the solution of medical image classification. In general, there are four main steps and stages in the solution provided for this thesis:

- Collecting Data
- Pre-processing Data
- Model Training
- Detection

Mentioned stages are only necessary for the development of the ML model for the solution, which is not everything for this thesis. There is also designed and developed mobile application which will be discussed in the following chapter (5) of this thesis. In the figure 4.1 you can see the visual representation of all stages mentioned before. All these stages should be done in a following order for making possible to get the final result, that is necessary for this thesis work.

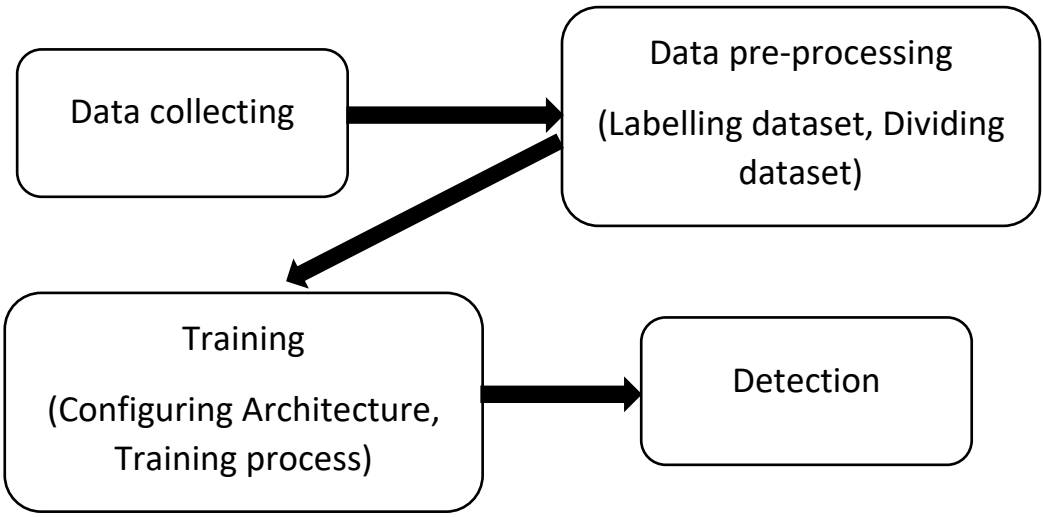


Figure 4.1 Proposed algorithm for chest X-ray detection

As you can see from the figure 4.1 there are four main stages of the algorithm which is proposed for chest X-ray detection model. All theoretical instances of the technologies which is necessary for realising these steps were discussed in the chapter three of this thesis work. In the following chapters there will be the explanation and discussion of the usage of previously explained technologies in a practical way for developing the model for chest X-Ray image classification.

4.2 Collecting and Pre-processing data

Having a great designed model without a good dataset is useless, because it wouldn't be able to predict the result with high accuracy percentage. It means that having a powerful dataset is one of the main problems during the development of machine learning approaches. For the model created for this thesis we define two different datasets, which were selecting after analysing six popular Chest X-ray datasets in the third chapter of this thesis. So, the following two datasets were selected during the analysis, as they have slightly more advantages than other datasets and can be very convenient for training the model that we created. In the following lines you can find these datasets:

- "COVID-19 Radiography Database" by researchers of Qatar University [48] with the collaboration of medical institutes of Malaysia and Pakistan
- "Synthetic COVID-19 Chest X-Ray dataset for computer aided diagnosis" by Cornell medical university [52].

A chest X-ray dataset including Covid positive, viral pneumonia, and normal image cases was generated in 2021 by a group of researchers from Qatar University working with collaborators from Malaysia and Pakistan who have connections to and collaboration with medical doctors. The dataset contains a total of roughly 19600

photos, of which 3618 are Covid instances, 6010 are viral pneumonia cases, and the remaining 10,000 are regular X-ray images. This dataset can be used in our solution for this thesis work because it was chosen as the winner of the "COVID-19 dataset award by Kaggle community" [48]. The second dataset consists of more than 21 thousand of synthetic Covid-19 X-Ray images which is defined to be used for computer-aided diagnosis. These high-quality photos were produced using an unsupervised domain adaptation method.

After combining these datasets, we have around 40 thousand of images in our dataset, where 13,627 images with Covid-19 positive cases, 6010 with viral pneumonia cases and the rest are normal images, where the number of normal images is 19,263. The resulting dataset is quite powerful and can train the model successfully, for detecting each type of anomalies (Covid-19 and Viral Pneumonia). In the figure 4.2 you can see the samples of images which contains in our dataset, namely the Covid-19 case in (a), normal chest image in (b) and viral pneumonia in (c)

Figure 4.2 Samples of CXR images of dataset

All images from both datasets have the format of DICOM (Digital Imaging and Communications in Medicine), so the final dataset also has the same format. A medical image saved in the DICOM format is known as a DICOM file. Medical image storage and distribution follow this industry-standard format [53]. A DICOM file may include one or more images, such as the findings of an MRI or ultrasound, as well as patient data. The majority of healthcare software developers and producers of medical equipment support the DICOM format, however using images with DICOM format for machine learning algorithms is not convenient. Computer Vision algorithms are not able to work with these images, so during the pre-processing process we have to change the format of images.

In computer vision, we frequently work with medical images, and practically all of our databases have image in that format. The DICOM image is not a simple image; it also includes data on the patient, the pixels, and other things. In our situation, all that is required is for us to view the image or save it in a format in which it can be viewed such as JPG or PNG file so that we can open it with any software we like. In our case we convert all the images of datasets to PNG to make it easier to analyse with other software products and also to make it easier for the machine learning model. We write the convert.py where all formats of images are converted from DICOM to PNG.

```
import numpy as np
import os
import pydicom
from PIL import Image

def get_names(path):
    names = []
    for root, dirnames, filenames in os.walk(path):
        for filename in filenames:
            _, ext = os.path.splitext(filename)
            if ext in ['.dcm']:
                names.append(filename)
    return names
def convert_dcm_png(name):
    im = pydicom.dcmread('./dataset/stage_2_train_images/'+name)
    im = im.pixel_array.astype(float)
    rescaled_image = (np.maximum(im,0)/im.max())*255 # float pixels
    final_image = np.uint8(rescaled_image) # integers pixels
    final_image = Image.fromarray(final_image)
    return final_image

names = get_names('./dataset/stage_2_train_images/')
quantity=0
for name in names:
    image = convert_dcm_png(name)
    image.save('./dataset/stage_2_train_images_png/'+name+'.png')
    quantity=quantity+1
    print(quantity, " images already converted to png")
print("All images were converted to png")
```

Figure 4.3 Converting DICOM images to PNG

The code snippet shown before is the content of convert.py file inside our application, which has the main purpose of converting the format of images of datasets to PNG. This functionality is done by using numpy, pydicom and PIL libraries. Here we have two main functions:

- Get_names – for getting names of images based on the path of directory where the folder of images contains
- Convert_dcm_png – function for getting the list of DICOM images and converting them to PNG.

At the end of the file, we add new PNG images to the new folder, from where the algorithm will take them during the training, and print the name of converted images, also at the end show the user that all images from the dataset are already converted to PNG.

After converting the image formats, wee need to pre-process the images to make it more convenient and easier for machine learning algorithm to work with. In the code snippet which is presented in figure 4.4 you can see the functions which has the purpose of changing the colour scale of the images and the size of images.

```
class CXRDataset(Dataset):
    """CXR."""
    def __init__(self,
                 img_dir: str = None,
                 annotation_path: str = None,
                 transform: bool = True,
                 x_img_resize: int = 224,
                 y_img_resize: int = 224,
                 ) -> None:
        self.img_dir = img_dir
        self.annotation_path = annotation_path
        self.transform = transform
        self.x_img_resize = x_img_resize
        self.y_img_resize = y_img_resize

        self.landmarks = pd.read_csv(self.annotation_path, delimiter=
    ",")
        self.landmarks["filename"].sort_values()
        self.images = glob.glob(self.img_dir + "/*.png")
        self.images.sort()
```

Figure 4.4 Pre-processing of images

In the code snippet you can see that we read all images of the dataset, previously converted to PNG format, and making some transformations for making them more convenient for the created model. Here we change the size of images to 224 x 224 and colour scale of image to BGR, these transformations make the system work properly, and also training will proceed successfully. The next stage of pre-processing is labelling the dataset to make it trainable. For object detection purposes, each image must be tagged, as seen in Figure 4.5. In contrast to picture classification, object detection labels objects by supplying a bounding box. The purpose of this function is to identify an object in the image that belongs to a particular class. The labelled dataset was divided into two parts: training and testing. The bounding box coordinates from the labelled dataset are produced as XML data. Each image is labelled before being transformed into a CSV file that contains all of the file names and the results of the previous tagging.

Figure 4.5 Labelling sample of CXR image

After labelling the images the dataset is fully ready to be train. The training process and used parameters for the training will be described in the following chapter of this thesis. Before ending the chapter related to data collection and pre-processing of images, you can see the example shot from the final dataset, which has three classes, Normal, Covid-19 and Viral Pneumonia.

	A	B	C
1	patientId	class	
2	0004cfab-14fd-4e49-80ba-63a80b6bddd6	Viral Pneumonia	
3	00313ee0-9eaa-42f4-b0ab-c148ed3241cd	COVID-19 positive	
4	00322d4d-1c29-4943-afc9-b6754be640eb	Viral Pneumonia	
5	003d8fa0-6bf1-40ed-b54c-ac657f8495c5	Normal	
6	00436515-870c-4b36-a041-de91049b9ab4	COVID-19 positive	
7	00436515-870c-4b36-a041-de91049b9ab4	Viral Pneumonia	
8	00569f44-917d-4c86-a842-81832af98c30	COVID-19 positive	
9	006cec2e-6ce2-4549-bffa-eadfcd1e9970	Viral Pneumonia	
10	00704310-78a8-4b38-8475-49f4573b2dbb	COVID-19 positive	
11	00704310-78a8-4b38-8475-49f4573b2dbb	COVID-19 positive	
12	008c19e8-a820-403a-930a-bc74a4053664	Viral Pneumonia	
13	009482dc-3db5-48d4-8580-5c89c4f01334	Normal	
14	009eb222-eabc-4150-8121-d5a6d06b8ebf	Normal	
15	00a85be6-6eb0-421d-8acf-ff2dc0007e8a	Normal	
16	00aecb01-a116-45a2-956c-08d2fa55433f	Viral Pneumonia	
17	00aecb01-a116-45a2-956c-08d2fa55433f	Viral Pneumonia	
18	00c0b293-48e7-4e16-ac76-9269ba535a62	Viral Pneumonia	
19	00c0b293-48e7-4e16-ac76-9269ba535a62	Viral Pneumonia	

Table 4.1 Example of the final dataset

The patientId column has the identification numbers of the patient, also the chest image of the same patient has the same image name and during the labelling process we used patient identification column as image identification. As you can see there are several classes shown in the table, which are Viral Pneumonia, Covid-19 and Normal. After labelling this column is counted as labels. After collecting and processing the data we can start to write the logic of the model and realize the training process.

4.3 Logic of the Model

In the previous chapter we discussed and show the steps for data collecting and pre-processing of images of dataset for future training. After having the dataset ready for training, the next important stage is to create the convolutional neural network that can

be trained with the input data from processed dataset. Before starting the training process, we should define some important parameters which are unnecessary from creating the convolutional neural network and train the developed model. Several parameters that are extremely important to be configured, such as epochs (length of training), the input dataset, the output results of the training and the type of model used. In the figure 4.6 the code snippet for creating the convolutional neural network using MobileNet V2 technology was presented.

```
#init creates object of class
classifier=Sequential()
#changes for thaneo
#activation fn-rectifier
classifier.add(Convolution2D(64,3,3,input_shape=(64,64,3),activation=
    'relu'))
classifier.add(MaxPooling2D(pool_size=(2,2)))
classifier.add(Convolution2D(32,3,3,input_shape=(64,64,3),activation=
    'relu'))
classifier.add(MaxPooling2D(pool_size=(2,2)))
classifier.add(Flatten())
#full connwection--output either sigmoid or softmax
classifier.add(Dense(output_dim=128,activation='relu'))
classifier.add(Dense(output_dim=1,activation='sigmoid'))
#compile
classifier.compile(optimizer='adam',loss='binary_crossentropy',
    metrics=['accuracy'])
classifier.summary()
```

Figure 4.6 Creating the Convolutional Neural Network with MobileNet V2

This code snippet represents the creation of CNN used MobileNet V2, mainly adding the layers of the model, such as convolutional layer, max pooling, flatten and some other parameters which are unnecessary to create the network. The output of this code snipper will show the created layers of the model.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 62, 62, 64)        1792
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 31, 31, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 29, 29, 32)        18464
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 14, 14, 32)        0
_________________________________________________________________
flatten_2 (Flatten)          (None, 6272)              0
_________________________________________________________________
dense_3 (Dense)              (None, 128)               802944
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 129
=================================================================
Total params: 823,329
Trainable params: 823,329
Non-trainable params: 0
```

Figure 4.7 Output of the CNN layers

After creating the layers of mobile convolutional network, the training process can be started but before that we will show the main architecture of MobileNet layers that created in the previous code snippet.

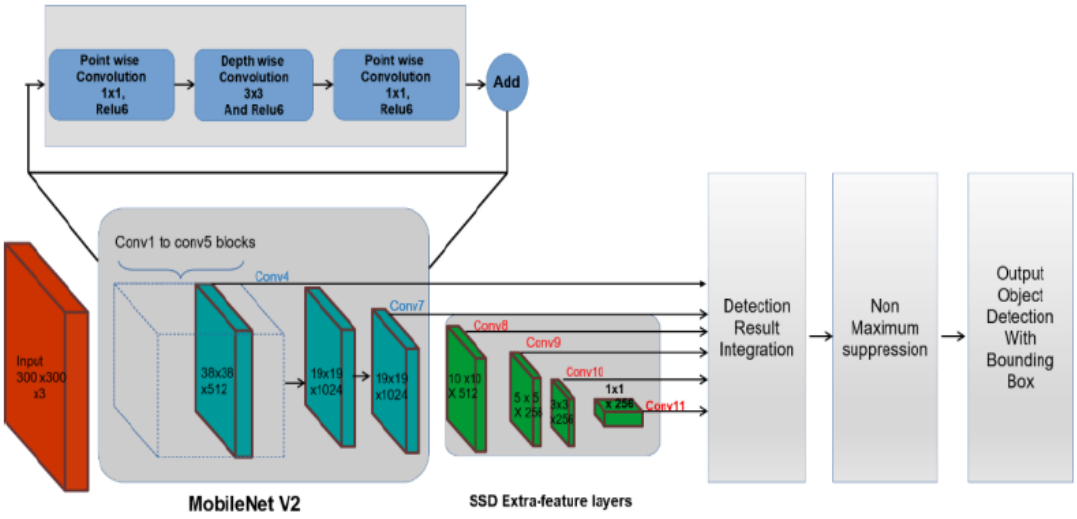


Figure 4.8 The architecture of created MobileNet V2 model

In the figure 4.8 the architecture and layers of the created MobileNet V2 model is represented. These models use the MobileNet deep network basis with an additional feature layer at the end, combining two machine learning techniques. Figure 4 displays the SSD MobileNet V2 architecture. For several categories, SSDs are quick single-shot object detectors [13]. Figure 4 shows how SSD links the expected outcomes of each convolution layer to create the output bounding box via non-maximum suppression. The core network model within MobileNet is integrated with an additional SSD feature layer by the SSD MobileNet, improving learning while speeding up computation. Also, the MobileNet V2 model is using separable depth-wise convolutional layer and pointwise functions. Reduced input size is achieved by using the bottleneck layer. The output from these models will be an image with a bounding box to represent the item detection task. This kind of detection will more clearly demonstrate the difference in the type of image observed.

The convolutional neural network is ready now, and the architecture of the created model is represented. Now, everything is ready to start the training process. The details about the training process you can find in the next chapter.

4.4 Training

The training procedure can start, as the model and dataset configuration are already finished and explained in the previous chapters. Each network's weight will keep

getting better during this model training procedure until it reaches the best value. The validation errors and accuracy show that the training procedure is producing solid outcomes. On a scale from 0 to 100%, the accuracy value is measured, the higher the accuracy value, the better the detection. A lower loss value, in contrast to the accuracy value, denotes a low error rate in its identification. The loss value is typically expressed as a decimal, and it is preferable if it is less than 1. The model's capacity to recognize each class in the dataset will be tested in this final step. By producing a bounding box and a detection score, SSD MobileNet finds items in a picture. In this work, the detection accuracy was assessed using a confusion matrix. Also measured is the model's speed at detecting the test image.

```
layer_outputs = [layer.output for layer in classifier.layers[:12]]
# Extracts the outputs of the top 12 layers
activation_model = models.Model(inputs=classifier.input, outputs=
    layer_outputs) # Creates a model that will return these outputs,
    given the model input
activations = activation_model.predict(img_tensor)
# Returns a list of five Numpy arrays: one array per layer activation
first_layer_activation = activations[0]

layer_names = []
for layer in classifier.layers[:4]:
    layer_names.append(layer.name) # Names of the layers, so you can
    have them as part of your plot
images_per_row = 10

for layer_name, layer_activation in zip(layer_names, activations): #
    Displays the feature maps
    n_features = layer_activation.shape[-1] # Number of features in
    the feature map
    size = layer_activation.shape[1] #The feature map has shape (1,
    size, size, n_features).
    n_cols = n_features // images_per_row # Tiles the activation
    channels in this matrix
    display_grid = np.zeros((size * n_cols, images_per_row * size))

    for col in range(n_cols): # Tiles each filter into a big
    horizontal grid
        for row in range(images_per_row):
            channel_image = layer_activation[0,:, :,col *
    images_per_row + row]
            channel_image -= channel_image.mean() # Post-processes
    the feature to make it visually palatable
            channel_image /= channel_image.std()
            channel_image *= 64
            channel_image += 128
            channel_image = np.clip(channel_image, 0, 255).astype('
    uint8')
            display_grid[col * size : (col + 1) * size, # Displays
    the grid
                         row * size : (row + 1) * size] =
    channel_image
    scale = 1. / size
    plt.figure(figsize=(scale * display_grid.shape[1],
                        scale * display_grid.shape[0]))
    plt.title(layer_name)
    plt.grid(False)
    plt.imshow(display_grid, aspect='auto', cmap='viridis')
```

Figure 4.9 Training process of the model

Giving a Machine Learning algorithm training data to use as a learning resource is the process of training an ML model. The model artifact produced during training is referred recognized as a "ML model." The right response, sometimes referred to as a target or target attribute, needs to be included in the training data. The learning algorithm generates an ML model that captures these patterns by looking for patterns in the training data that relate the qualities of the input data to the target (the prediction you want to make). After the training process is finished, the results of layers will be shown in the output terminal in a visual way. In the figure 4.10 you can see the convolutional layer of the model after training with the provided dataset and the parameters for training.

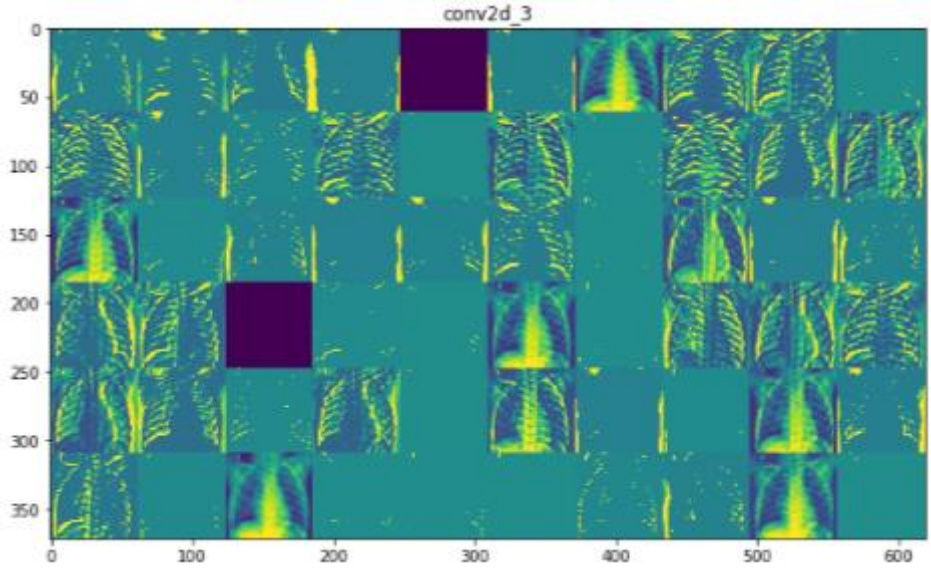


Figure 4.10 Convolutional Layer of the model after training

The visual presentation of all layers is shown after training process finished but showing the convolutional layer is enough, to understanding the main schema of the training process. The results of the performance analysis will be shown in the following chapter of the thesis.

4.4.1 Performance Evaluation

In this chapter the evaluation of the model performance after the training will be discussed. The research findings, including those from training and model testing, will be covered in this part. The Tesla PCIe-16GB Python 3 Google Compute Engine Backend (GPU) hardware and Google Collaboratory software were utilized for training and model testing in this study. Utilizing Google Collaboratory has the benefit of allowing the use of high-end, open-source machines. This research method is made easier by the Tensorflow open-source library's use. The CNN SSD MobileNet V2

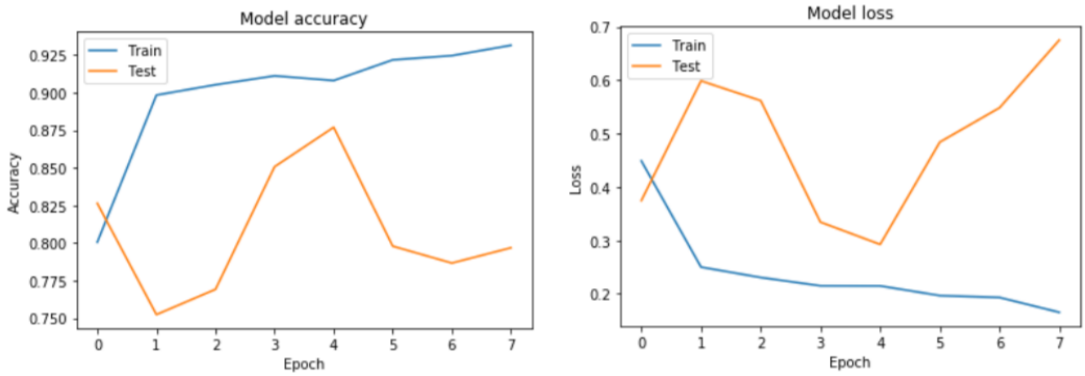
architectural models have been used for training. The model is run 20,000 times. The configuration time can be used to estimate the length of training; if it is longer than that, it will take longer. As a result, the epoch determination acts as a baseline against which other variables, such as accuracy, speed, and loss value, can be compared. Additionally, these two models' training results demonstrate good accuracy and loss values. The SSD MobileNet V1 model is utilized in the initial training procedure. This training method required a GPU with good specifications, but it still took about 14 hours due to the size of the used dataset. For showing the performance of the model, based on the accuracy and the loss, it is necessary to run the code snipper which represented in the figure 4.11

```
import matplotlib.pyplot as plt
# Plot training & validation accuracy values
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()

# Plot training & validation loss values
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```

Figure 4.11 Code snippet for representing the performance of the Model

As you can see from the figure shown before, for representing the performance graphs of the trained model we can use the pyplot module of the matplotlib library. Used history function is the call back function for the created model.



(a) (b)

Figure 4.12 Accuracy and Loss graphs of trained model

Figure 4.12 displays the outcomes of training using this model. The loss value varies from the start of the training procedure to the finish, as shown by the graph. On CNN, a decent loss value is between 0 and 1, or less than 1%. The loss value in this V2 model

training begins to fall below 1 around epoch 7000 (7k). The decreasing of the model loss from the starting of the process till the end shows that, the model is trained successfully, so that the percentage of model loss is decreasing after 7k epochs of training.

From the figures its clearly seen that, train model accuracy is slightly increasing from the beginning to the end of the training process. Which means that training process held successfully. At the beginning the accuracy percentage was around 80 percent, and after 7k epochs, this number is increased to 94 and higher, which can be counted as a successfully trained model. In this chapter the training process was discussed, and the results after performance analysis is shown. In the next chapter there will be the explanation of detection process, and visualization of the output.

4.5 Detection

In this chapter we will show the examples of detection output of the model and provide an analysis based on the detection performance of the algorithm. In the figure 4.13 you can find the code snippet which gives a test image as an input to the model and print the predicted result.

```
import numpy as np
import tensorflow as tf

from keras.preprocessing import image
from keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(
        rescale=1./255,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True)
training_set = train_datagen.flow_from_directory('C:/Users/heydarli/
    polito/master_thesis/mobileCXR/dataset/traint',
        target_size=(64, 64),
        batch_size=32,
        class_mode='binary')

model = tf.keras.models.load_model("C:/Users/heydarli/polito/
    master_thesis/mobileCXR/cnn_model.h5")
test_image=image.load_img('C:/Users/heydarli/polito/master_thesis/
    mobileCXR/1.jpeg',target_size=(64,64))
test_image=image.img_to_array(test_image)
test_image=np.expand_dims(test_image,axis=0)
result=model.predict(test_image)
training_set.class_indices
if result[0][0]==1:
    prediction='normal'
elif result[0][0]==2:
    prediction='covid-19'
else:
    prediction='pneumonia'
print(prediction)
```

Figure 4.13 Prediction process of the model

The code snippet presented in figure 4.13 has the purpose for adding a target image as an input to the model and based on the prediction of the model show the output. The provided input image has the label of Covid-19 and the trained model's output was 'covid-19' The result of the prediction is shown both in a textual form and in a visual format as shown in figure 4.14. The resulting images has the bonding box which detects the chest of the patient from the whole image and shows the predicted result with the accuracy percentage.

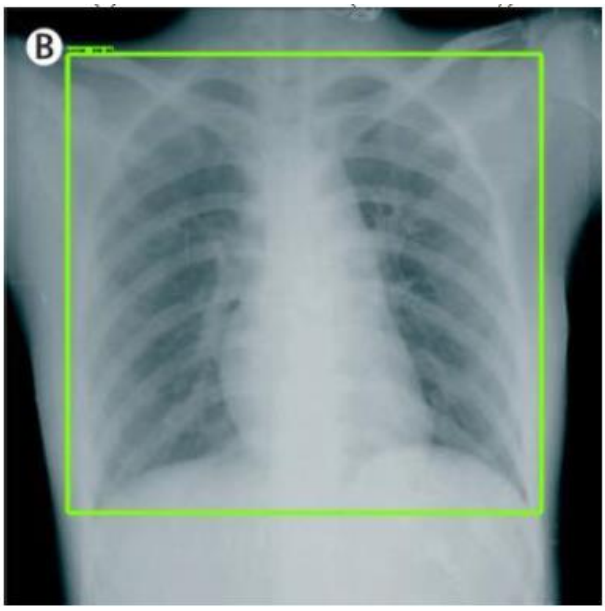


Figure 4.14 Visual output of the model

As you can see from the figure 4.14 the visual output of the trained model contains the input image, with the coloured bonding box which has the label and the accuracy percentage on it. The colour of the box is changing based on the accuracy percentage of the predicted image, if it is more than 90 percent it is green coloured, and each 10 percent of decreasing will change the colour from green to light green, yellow, orange and at the end to the red colour.

Lastly, we will provide the confusion matrix of the model based on the 50 input images for each label (class) which are presence for our model, namely the covid, normal and pneumonia classes. The matrix confusion approach aids in measuring machine learning classification performance. With this kind of model, it is possible to identify and categorize a model on the test dataset that has known true values. The phrase "confusion matrix" is both straightforward and baffling. A performance evaluation method for machine learning classification is the confusion matrix [55]. This particular type of table aids in determining how well a classification model performs when applied to a set of test data for which the true values are known. Although "confusion matrix" is a pretty straightforward term, the verbiage used to describe it can be a little difficult to understand. Here is a brief description of this method.

True Class	Predicted class			
	Covid	Normal	Pneumonia	Total
Covid	46	1	3	50
Normal	1	48	1	50
Pneumonia	0	0	50	50

Table 4.2 Confusion matrix of the model, based on predictions

The confusion matrix which is shown in the table 4.2 use the 150 images from the dataset, 50 images per each class, namely for covid, normal and pneumonia classes. Designed model can detect all 150 images in under 5 minutes or around 2 seconds per image, which can be seen as the powerful model. We can see that model can detect 46 covid cases correctly, or with average accuracy of 92% which is quite high, where the normal and pneumonia cases have even more accuracy percentage, 96 % for the normal class and 100% for the pneumonia class. But this numbers can be various based on the input 50 images. But in general, the accuracy numbers are quite high, which means that developed and trained model is quite successful.

Chapter 5

Developed Mobile Application

5.1 Overview

In the chapter four of this thesis, we discussed the machine learning model created to solving the main question of this work, described the main technologies used for ML approach and show the practical realisation of the created model. But, for this thesis work having the machine learning approach is not enough, as there is a necessity of creating a mobile application which will use previously designed model, to predict the diagnosis of the patient, based on the chest X-Ray image of the patient taken from the camera of mobile device or by uploading them from the memory of device. In the figure 5.1 the simple architecture of the developed mobile application is shown.

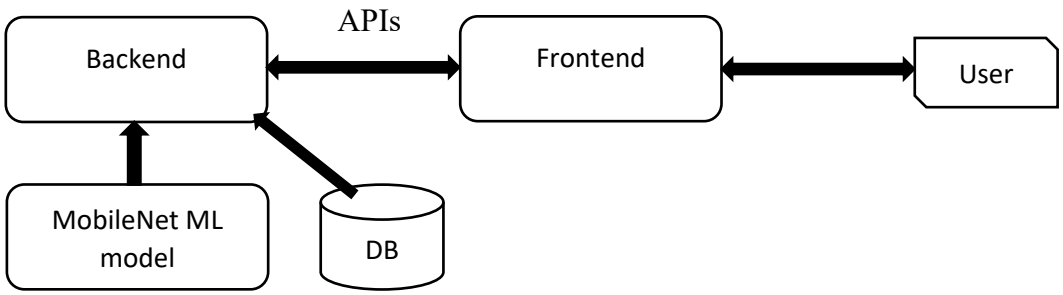


Figure 5.1 Simple Architecture of Mobile Application

As you can see from the previous figure developed mobile application will have four main parts. All these parts have the importance for having a working mobile application:

- Database – for storing the user data, for making login and authentication and storing the results of the previously scanned chest X-ray image, for making possible checking the results of previous analysis.
- Machine Learning model – main part of this application, because without this model application wouldn't be able to predict the diagnosis from the input images taken from mobile device's camera.
- Backend – where all the request and services will be written. Also, the integration of the previously designed machine learning model and the mobile application will be done in the backend of the mobile application.
- Frontend – for showing the results of the analysis to the patient or to any user who will use the application.

In the following chapters the brief explanation of each part will be represented, with the technologies used for creating that specific part. Also, some code snippets will be

shown, for practically understanding all the process of the application developed for this thesis work.

5.2 Used technologies

There are plenty of various that can be used for developing mobile application. Selection of the used technology is quite important to having a powerful and convenient application. It is important to know the compatibility of technologies used in frontend and backend, also having a proper technology which makes the model integration part much easier is quite important. In this chapter the quick comparison of possible technologies for each part of the application will be provide and the selected one will be explained. The main thing is that, checking all possible technologies for developing the mobile application is extremely useless, it is enough to look through the technologies that can be realized by the student who is writing this thesis.

Firstly, the possible mobile application development frameworks and IDEs will be discussed. Two popular frameworks and technologies will be discussed in this chapter, namely the React Native app, and the mobile app developed in Android studio with Kotlin programming language. Both technologies are quite popular in this sphere and has plenty of advantages which can be extremely helpful during the development of such application. For development of the mobile application in this thesis work will be used the React Native framework, as it has some advantages over android studio app developed with Kotlin, such as:

- React native mobile application can be compiled and run in all operating systems, at the same time on Android, IOS and Windows phone, using the same code [56]. This possibility is the big advantage for the mobile application development for this thesis, as having the app running in all devices can decrease the development time of application and help us to focus on the machine learning model.
- React native app is easier to use and the IDE which used for React native app is quite light than Android Studio.
- It is easier to integrate the Machine Learning model with the mobile application using the previously defined libraries, which has the purpose for showing the ML model's predictions in the frontend.

Based on the mentioned advantages we will use the React Native app for developing the solution for this thesis, however the mobile app development with Android Studio has significant advantages and these applications are quite complex and stronger rather than React Native app. But, as we do not need to create extremely complex application with huge functionalities, there is no necessity of using the Android Studio for developing mobile application with Kotlin. After defining the frontend technology which will be used during the development stage of application, we can choose the technology which will be used for designing the backend of our application.

There are several technologies that can be used during the development of the backend of application, but we will look through only two popular approaches, namely the Node JS with Express server and the FastApi approach. These both approaches has quite enough power and popularity, and all of them has plenty of advantages in comparison to similar technologies.

Backend created with Node JS is quite strong and it is possible to afford this approach doing hard computations and complex requests, however the Fastapi has the same possibilities. Fastapi approach has some little advantages over the Node JS technology based on the topic of our thesis, such as:

- As the FastApi is the framework of the python, and our machine learning model was created with python, it is more convenient to use Python using technology for the backend of our application.
- With fastapi there is a possibility of using new technology OpenApi which can help to generate frontend services automatically. This approach can be really helpful during the development stage, because we do not need to write each front API separately and make the connection with frontend, with OpenApi you don't need to write them manually, as all of the interfaces and the services are generated automatically.

Based on the mentioned advantages of FastApi technology, we will choose this approach for future development of the backend of our application. Also, combination of fastapi and React is quite appropriate and convenient, as the communication between two services is realising without any doubts.

To conclude this chapter, the selected technologies for future development of the application for this thesis work, are React Native for the frontend development and the FastApi approach for the backend development part. In the following chapters the brief explanation of proposed technologies will be presented.

5.2.1 Frontend

Before starting to describe the technologies used for creating the frontend of the developed mobile application, it is necessary to give a brief explanation of what is the frontend means. The frontend of the web or mobile application is the stage or part of the application to which the user interacts directly, and usually the frontend is referred to the 'client side of the application' [57]. The frontend of the application contains almost everything that user sees when making any interaction with the website or mobile application, such as represented colours, text styles, graph, images, buttons, navigation bar and everything the user can see and interact. The frontend web and mobile developers provide the structure, behaviour, the way of appearance and the content of every part of the application that appears on the screen of the browser or mobile application when they are running. One of the key points for both mobile and web frontend developers is to make the app responsive, which means that to make them work properly in all screen sizes and in all devices, unrelated to their screen

dimensions and sizes. Also, the communication with the backend should be done in a proper way, that the application could response to the requests in a small amount of time, which can avoid possible negative feedbacks from the user side. There are various frontend languages that are used now, such as HTML, CSS, Javascript, and plenty of various frameworks which makes the development process much easier and convenient for the developer.

There are three main tools and frameworks for mobile development nowadays, such as Android Studio SDK, Xcode and React Native app development. In the following paragraphs we will look through all of these frameworks and decide the one to use during the development of the application for this thesis work.

For Apple operating systems, there is a software development environment called XCode. It enables you to run programs, define graphical user interfaces and how they relate to code, and write and test code [59]. The XCode IDE This acronym stands for "integrated (or unified) development environment," which is what it means in English. The tools with everything you need for both writing code and putting together a finished project are known by this term. The setting was created with the macOS operating system in mind. It allows for the creation of projects that can operate on iOS, tvOS, and watchOS. Along with AppleScript, Python, Ruby, and Java, Xcode supports the programming languages Swift, Objective-C, C, and C++. We wouldn't use this technology during the development of the mobile app for this thesis, as having the application which is working only on one OS is not so convenient.

Android Studio SDK is the software development kit which is using for developing mobile applications for Android Operating System, using the programming languages such as Java and Kotlin. This SDK is quite powerful and can create extremely complex applications which are able to solve difficult problems and has the big functionalities. It can be the good option to create the application with Android Studio SDK, but it is not so convenient to have developed application only for one operating system, so we wouldn't use this technology during the development phase of our application for this thesis work.

The last mobile application development framework is the React Native, which combines the best valuable parts mobile development with popular web framework React, which is the one of the best frameworks for developing user interfaces. The main advantage of development with this framework is that it is completely easy to develop a mobile application for both operating systems writing a same code, not needed even to compile them separately for each OS. It is possible to build native mobile apps for iOS and Android using the well-liked JavaScript-based mobile app framework known as React Native (often referred to as RN). Using the same codebase, the framework enables you to develop applications for several platforms [60].

In 2015, Facebook originally made React Native available as an open-source project. It has quickly risen to the top of the list of tools used for mobile development. Some of the most popular mobile applications in the world, such as Instagram, Facebook, and Skype, are powered by React Native development.

In the figure 5.2 you can see the simple architecture of the mobile application developed with React Native.

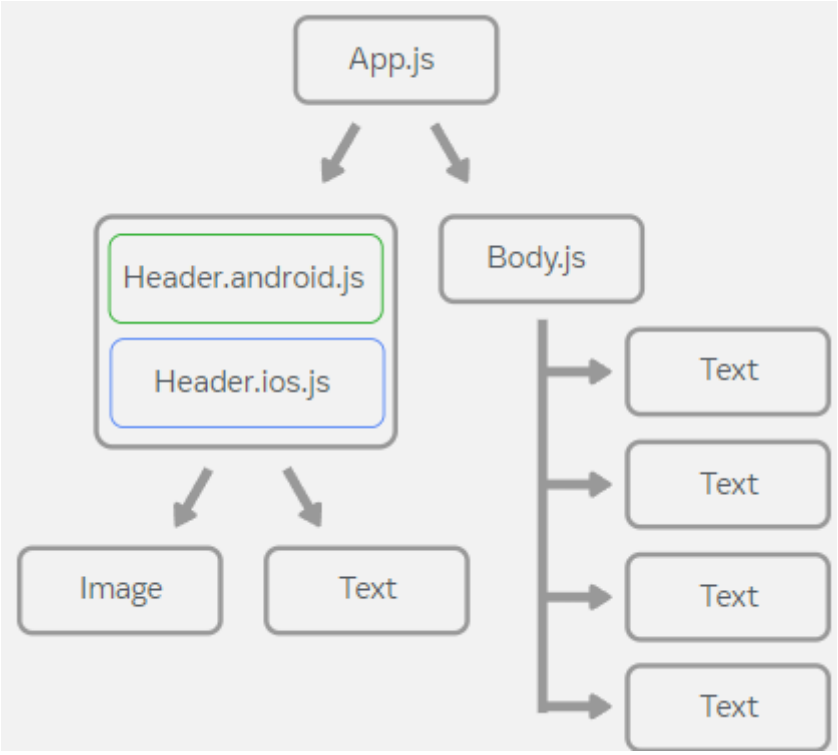


Figure 5.2 Simple architecture of the React Native App

Taking into consideration the advantages of React Native over other mobile app development frameworks, we decide to use React Native for the development of our mobile application for this thesis work. In the next chapter of this thesis, there will be the comparison of various backend frameworks, and brief explanation of each framework.

5.2.2 Backend

In the previous chapter we looked through the meaning of frontend in web and mobile applications and also briefly explain the main advantages of used mobile app development framework, namely the React Native app. But having a frontend is not enough for creating a full application which can realise so complex functions. For developing complex applications which can make a communication with server and also the Machine Learning model as needed for this solution, we should have a proper backend. In this chapter we will look through the theoretical aspects of backend, backend development, will compare the possible backend languages and frameworks, and also will show some practical aspects from the development of the backend part of application.

The backend of the mobile or web application is also known as a server-side of the developed application where the frontend is client-side, as mentioned in the previous chapter of the thesis [57]. The backend stores and organizes the data, which is necessary for the application, as well as ensuring that every part on the frontend (client side) of the developed application works properly. In comparison with frontend, backend doesn't have the direct communication and interaction with the user, however it plays the main and extremely important role behind the scenes with adding all the main functionality to a site. You can remember that without a clean and properly developed backend the frontend of the application is quite useless and wouldn't work properly.

Writing APIs, building libraries, and interacting with system components devoid of user interfaces are all backend tasks. The backend typically controls the essential features of web applications. For instance, the backend controls the real financial activities when you go through the checkout process when making a purchase from an online retailer. The backend communicates with services that are not part of your own app or website, such your bank app or PayPal, while the frontend ensures that the "checkout" button is placed correctly on the page and directs you to the following page. The entirety of this transaction's process takes place in the background, so you cannot see it in action.

There are plenty of different languages and frameworks for backend development of web and mobile applications. In the following paragraphs you can see the brief explanation of backend languages and frameworks, and also the comparison between these languages.

JavaScript code can be run outside of a browser using NodeJS, where NodeJS is an open-source cross-platform runtime environment. It's crucial to note that NodeJS isn't a standalone programming language or framework. Although NodeJS itself is not a programming language, it is made to run another one called JavaScript. An interactive web page can be created using JavaScript, a text-based computer language which can be used for developing server side or client side of the application.

C++ is a complex programming language which can be used for solving various types of tasks. Mostly this language is used for developing operating systems, some internet browsers, and video games, instead of mentioned usage, C++ supports plenty of programming paradigms, including functional, object-oriented, and procedural programming. Because C++'s backend development is so complex and challenging in comparison to other frameworks, it hasn't been as popular in recent years despite being robust and adaptable.

Swift is a software IDE where it is possible to create mobile applications which can be run for mobile phones which uses IOS operating system, and other products of Apple such as Mac, apple watch and also apple TV. With it, programmers will have more freedom than ever before. Since the usage of Apple devices has been steadily rising over the past few years, there has been an increased demand for applications developed using the Swift framework.

Java is the preferred complex programming language for creating native Android apps. Java primarily using for developing Android mobile applications and for creating backend APIs of web application. The Android Native Development Kit (NDK) allows developers to create Android apps in C and C++, however Google does not recommend it because it may affect the app's performance. Last years this language is getting less popular in backend development for mobile applications, because now android studio uses the Kotlin Language mainly.

There are some different frameworks for backend development that uses the Python programming language, such as Django, Flask and others but the there is also the new framework which called FastApi. A new Python web framework called FastAPI is robust and fun to use. FastAPI is worth considering because of the following qualities: One of the quickest Python web frameworks is FastAPI. In fact, it runs as quickly as Go and Node.js [58]. FastApi has various advantages rather than other backend frameworks, such as

- Having a clear and comprehensive developer documentation, which is the great advantage, as problem solving time can be decreased several times, having the great structured documentation.
- Get free data conversion and validation by typing in your code.
- The speed is quite fast, based on the provided tests and some analysis the speed of this framework is higher than NodeJS, Go and Java based backends, which means that it is a quite fast technology.
- Utilizing dependency injection, creating plugins is simple.
- Having an opportunity to use the OpenApi platform, which helps directly create the frontend services and interfaces, without writing any code in frontend, just with running a command which is necessary to generating the frontend client based on the developed backend structure.
- Having the backend development framework which uses the python programming language can be an advantage while using the machine learning model, as the model is created with python programming language. It is much easier to communicate with the model if the backend framework uses the same language as the model created.

Based on the mentioned advantages that FastAPI has, we will proceed the backend development of the app that is needed for the solution of this thesis work, with the FastAPI backend development framework.

The structure of the developed backend with FastAPI should have some main components and folders for working in a proper way. The simple architecture of the backend should be in a following way, which is represented in figure 5.3.

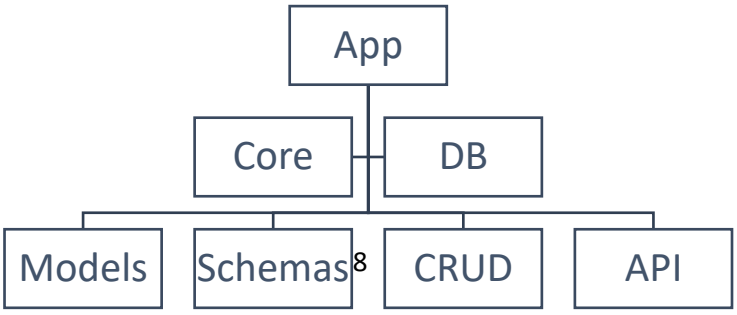


Figure 5.3 The structure of Backend with FastAPI

In the figure 5.3 the simple structure of the backend developed with the FastAPi framework is shown. In the following paragraphs the brief explanation of each folder and its content will be described:

- Core – in this folder of the backend architecture, the main settings and configuration of the backend is provided.
- Db – in the database folder, as it is shown from the name of folder, all connections related to the database will be done. Also, the base class for creating the database tables is provided inside of this folder. Without having the content of this folder, it is not possible to have a connection with database.
- Models – here the models of database are provided. The model of DB means that the tables for database, so each model created in this folder is convenient to the table created in the database. So, table creation of database is also provided with alembic revisions.
- Schemas – in the schemas folder you can find the schema for each created model, which means the behaviour of the model during different operations, such as reading, updating, creating and etc.
- CRUD – in this folder the crud operations will be provided, which were created using the models and schemas created before.
- API – last folder of the fastapi backend architecture, contains the created APIs, for making possible the usage of the created services both in backend and in frontend. Without having the created APIs, we couldn't use previously defined Crud operations.

In this chapter we discussed the technologies and main factors of different frameworks which are necessary during the frontend and backend of the web or mobile application. In the following chapter the main features of the developed mobile application will be described, which is created based on the technologies defined in this chapter, significantly for mobile app frontend and the backend.

5.3 Features of the Mobile Application

After defining the technologies for frontend and backend of mobile application development, we can start for creating the mobile application for this thesis work. Mobile application is created using React Native for frontend, and the Fastapi Python framework for the backend.

The frontend created with React Native has a simple structure, with some components and folders. The simple structure of React Native App, contains the following folders, where each of the has different purpose:

- App.js – main component of application which runs first after running the whole application. In general, App.js has imported components of all application, possible routes and all navigation parameters needed for application
- Components – here the separate component of the application is provided.
- Navigation- in this folder there is the MainContainer.js which contains all components and the navigation parameters, which creates the bottom bar navigation in the mobile app. The code snippet for the MainContaine.js is represented in the figure 5.4.
- Openapi – this folder contains all interfaces and services automatically generated after running the client generation code. These services and interfaces are generated related to the models, schemas, crud operations and APIs created in the backend of the application.
- State – here all the state system of the application is stored, by using the redux react toolkit.

```
// Screen names
const homeName= "Home";
const xRayName= "Xray";
const mapName = "Map";
const listName= "List";
const Tab= createBottomTabNavigator()
export default function MainContainer(){
    return(
        <> <NavigationContainer>
            <Tab.Navigator
            initialRouteName={homeName}
            screenOptions = {({route})=>({
                tabBarIcon: ({focused, color,size})=>{
                    let iconName;
                    let rn = route.name;
                    if(rn=== homeName){
                        iconName=focused ? 'home': 'home-outline'
                    } else if (rn=== xRayName){
                        iconName = focused ? 'camera' : 'camera-
    outline'
                    } else if (rn===listName){
                        iconName = focused ? 'list' : 'list-
    outline'
                    }
                    else if (rn===mapName){
                        iconName = focused ? 'map' : 'map-outline'
                    }
                    return <Ionicons name={iconName} size={size}
    color={color}/>
                },
            })}
            tabBarOptions = {{
                activeTintColor : '#385399',
                inactiveTintColor : 'grey',
                labelStyle: {paddingBottom: 5, fontSize:10},
            }}
            >
            <Tab.Screen name={homeName} component={HomePage}/>
            <Tab.Screen name = {xRayName} component={CheckXRpage
    }/>
            <Tab.Screen name = {mapName} component={
    HospitalsMapPage}/>
            <Tab.Screen name = {listName} component={ListPage}/>
            </Tab.Navigator>
        </NavigationContainer>
        </>)}
```

Figure 5.4 Main navigation container of the application

As you can see from the main navigation container of the developed mobile application, there will be four main pages, such as:

- Home Page – where the main statistics related to the Covid-19 infection is representing, such as total number of infected people, total deaths and total number of recovered people. The implementation and representation of this page will be shown in this chapter.
- Check X-Ray page – where user can upload or take the chest X-ray image, and check if uploaded x-ray image has the covid infection, pneumonia or nothing is detected in the patient.

- Hospitals Map page – where user can see the nearest hospitals based on the geolocation of the mobile app.
- List Page – here user can see previously analysed chest X-Ray images, where the results and the date of detection is showing.

In the following pages some pages of the application with its functionalities will be represented. Let's start with the Home Page of application. This page is created, to welcome the user and show the main statistics related to the Covid-19 virus. Here user can see the total number of covid cases, total death, total recovered people and also the statistics of each country. The statistic of Covid-19 cases is gotten from the public API which called: api.covid19api.com [61]. The code snippet in figure 5.5 shows the way using the data from the public API.

```
1    const url = "https://api.covid19api.com/summary";
2    const [data, setData] = useState();
3    const [isLoading, setIsloading] = useState(false);
4    useEffect(() => {
5        const fetchCovidData = async () => {
6            setIsloading(true);
7            try {
8                const result = await fetch(url);
9                const response = await result.json();
10               setData(response)
11               setIsloading(false);
12           }
13           catch (e) {
14               console.log(e)
15           }
16       }
17       fetchCovidData();
18   }, []);
```

Figure 5.5 Usage of Covid19 public API

After getting the whole data from the API we can call this data state and use their children components which are: total confirmed, total death, total recovered and other aggregate numbers related to Covid-19 Cases. Later, by using previously mentioned data we show these statistics in a frontend. The visual representation of the Home page is shown in **Figure 5.6.**

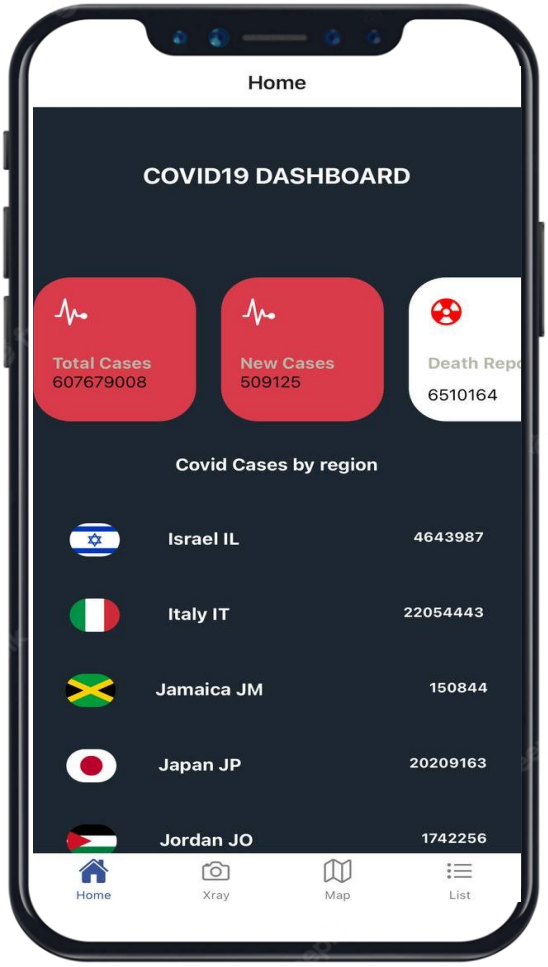


Figure 5.6 Home Page of the application

Rather than total statistics of the Covid-19 cases such as total cases, new cases, total recovered people, there is possibility to see the total statistics of each country. User can see the flags of all countries near to their names and at the right side the total number of covid cases in the mentioned country.

Developed mobile application has some more functionalities such as showing the nearest hospitals based on the geolocation of the user, the archive page where the list of previously predicted X-Ray images is shown. As you can see from the figure 5.6 application has the bottom bar, with four buttons, each of them represents the page, and by clicking to the opens the significant page. In the next chapter we will look through the X-Ray page which uses the previously created machine learning model for predicting the diagnosis based on the uploaded or captured chest X-Ray image from the mobile device.

5.4 Integration with the Machine Learning model

In the previous chapter some features of developed mobile application is shown, such as showing the statistics of covid cases to the user, showing the nearest hospitals in the map and some other features. In this page we will look through the integration of developed Machine Learning model with the mobile application. For the integration of the Machine learning model in the mobile application we need to create a function in backend to creating the API which uses the model for getting the results of the prediction from the model. In the Figure 5.7 you can see the backend function which communicates with the model and predicts the result.

```
1  @router.post("/{patient_id}/session/{session_id}", response_model=
       schemas.RequestOutput)
2  async def create_request(
3      *,
4      session_id: int,
5      send_time: str,
6      file_size: int,
7      db: Session = Depends(deps.get_db),
8      file: UploadFile = File(...),
9      current_user: models.User = Depends(deps.
       get_current_active_user),
10     background_tasks: BackgroundTasks
11 ) -> Any:
12     filename, folder = request_initialize(send_time)
13     data = await file.read()
14     img = image_from_bytes(data)
15     to_save = img.copy()
16     infos, result_img = model.evaluate(img) # Using the Model for
       evaluating
17     results_filename = folder + "result_picture.png"
18     background_tasks.add_task(cv2.imwrite, filename, to_save)
19     response = request_response(session_id=session_id,
       image_to_save=to_save, infos=infos,result_img=result_img, db=db
       , original_image_path=filename,
20                                 result_image_path=results_filename,
       image_size=file_size)
21     return response
```

Figure 5.7 Communication with the model from the Backend of application

In the figure 5.7 the request which communicates with the machine learning model created before is represented. As you can see in the line 12, we called the “model.evaluate” function, with the parameter of the input image. The result of this function is the response from the model which has the resulting image with bonding box and the final label. Later we can use this API in the frontend for showing the result of the Machine learning model to the user through the mobile application interface.

For having the frontend client of the APIs and services created in the backend there we should run the code for generating the frontend client with the openapi. After

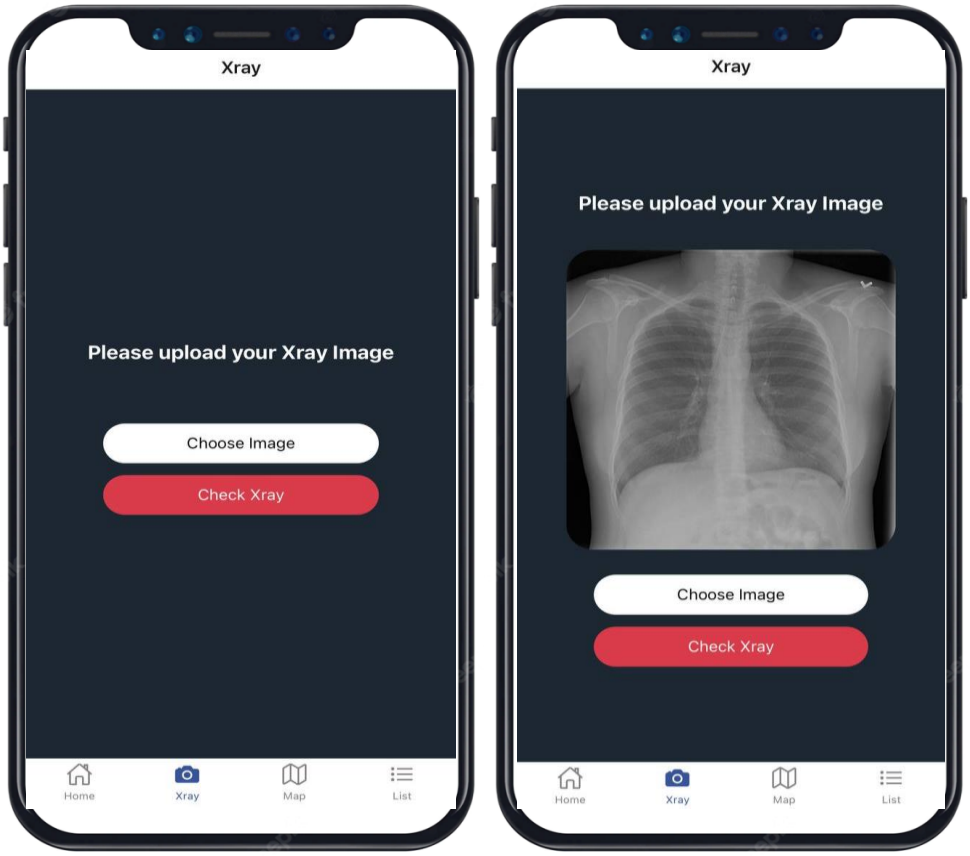
running this code, we will have the service in the frontend which will look like in the following code snippet (figure 5.8)

```
    public static createRequestApiV1RequestsSessionSessionIdPost(
        sessionId: number,
        sendTime: string,
        fileSize: number,
        formData:
    ): CancelablePromise<RequestOutput> {
        return __request(OpenAPI, {
            method: 'POST',
            url: '/api/v1/requests/{patient_id}/session/{
session_id}',
            path: {
                'session_id': sessionId,
            },
            query: {
                'send_time': sendTime,
                'file_size': fileSize,
            },
            formData: formData,
            mediaType: 'multipart/form-data',
            errors: {
                422: `Validation Error`,
            },
        });
    }
```

Figure 5.8 Frontend service for communicating with the model

The figure 5.8 represents the frontend service for communicating with the backend for using the machine learning model. Later we can just import this function inside the component and use the result data, which is the output of the model and show them to the user in the frontend. As you can see from the previous figure the url of the request is the 'api/v1/requests/{patient_id}/session/{session_id}' which is the same to the request created in the backend of application as shown in the figure 5.7. After having this service, it is really simple to use the results of the machine learning model in the frontend.

In the figure 5.9 you can see the Check X-Ray page, which is created for taking or uploading a picture of chest X-Ray and checking if that patient has the covid-19, pneumonia infections, or the patient has not any detections. In the left image (a) of the figure 5.9 you can see the mobile interface of the Check X-Ray page, where the user should press to the button choose image, after that they can choose the image from the gallery of their mobile device or take the photo from the camera of device. After selecting the image, the screen will be look like in the left image (b), where the uploaded or taken photo is shown, and user can press the button "Check X-Ray"



(a) (b)

Figure 5.9 X-Ray image checking page

In the figure 5.9 you can see the main and most important page of developed mobile application. In this page all the logic and main question of the thesis is applied as the machine learning model is used there. After uploading or taking the image of the chest X-Ray image, the user can press the button 'Check X-ray', which will call the function explained in the figure 5.8, which has the main purpose of calling the API of the model, where as the response we get the result of the prediction of model. The accuracy of the prediction while uploading the image is quite high, where it differs from 90-100 percent, as there are not any quality problems that can be appear due to the lack of camera quality of the mobile device. But when the image is taken from the camera of mobile device there could be a problem, which are related to the camera performance of the mobile devices. In this case the accuracy of the prediction can differ from 70-95, which can be improved, by training the model with new data and also train the model with some blurry images, which can be helpful to teach the model to distinguish this type of images.

After pressing the button 'Check X-Ray', the service which calls the request to communicate with the machine learning model, is called by on click function. The response of this service is the label and the final image which is returned from the model. This process can take around 2-4 seconds to get the data from the model and show it in frontend. During this waiting time, user will see the loading bar, which can avoid the possible misunderstanding by the user. After receiving the final response

from the model, in the screen of the mobile application the resulting label will be displayed.

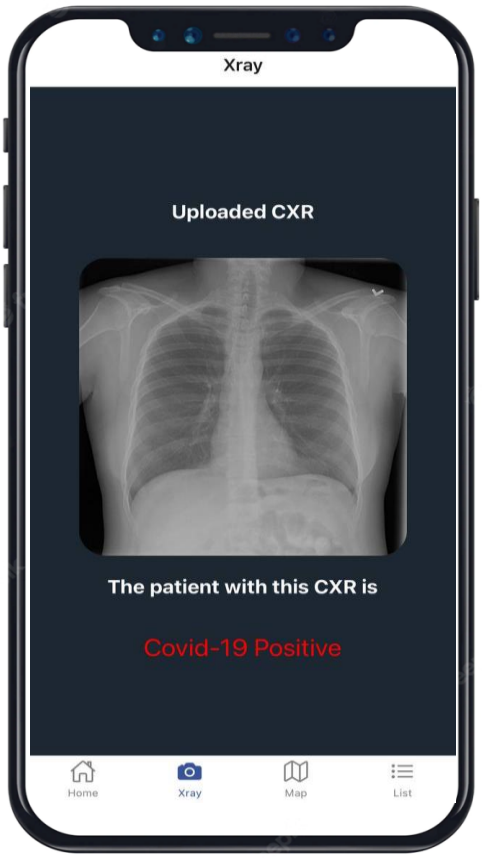


Figure 5.10 Result of the Prediction in the mobile application

After getting the response from the request which communicates with the machine learning model, we show the text 'Covid-19 Positive' if the response label is covid, 'Normal' if the response label is normal, and 'Viral Pneumonia positive' if the response label is viral pneumonia. Later the result of the prediction is written to the Database and user can check them when they want just to open the Archive page, which is in the right side of the bottom bar.

In this chapter the communication of the machine learning model with mobile application is explained. We describe the backend request creating the communication with model and giving the prediction as the response, and at the end check the function for helping to show the results of this request in the frontend of our application, for making them visible to the users. Also, at the end the visual representation of the mobile interfaces of X-Ray page were shown.

5.5 Summary

In the chapter five the main focus was on the developing the mobile application for solving the main question of the thesis. Firstly, we start by making an overview of the

created application, with the main components of the simple mobile application, which can communicate with the machine learning model. Then the possible technologies for frontend and backend development were discussed, by making a comparison of mobile app development frameworks and backend frameworks. Taking into consideration the results of comparisons and checking the advantages of all mentioned technologies we choose the most appropriate frameworks for mobile app frontend development (React Native) and backend development (FastApi with Python). After defining the structures of frontend and backend we start the development phase, and show some important functions, with their codes and visual representation of the code. Some main pages of mobile application were shown visually, along with the most important page of the application, which is the Checking X-Ray Page, and with that we finish the description of the proposed solution for this thesis.

Conclusion

The main goal of the thesis was to create a mobile solution which could make a diagnosis based on Chest X-Ray images. During the thesis work firstly, we took a look to previously designed solutions which are similar to the main purpose of this work, define the advantages of the solutions and find the drawbacks which can be improved during this work. Then we make research on possible technologies which can be used during the development of the machine learning approach, which will predict patient diagnosis from the provided chest X-Ray images. After defining the technologies there were a brief comparison of technologies and at the end the best approaches were selected for using during the designing of machine learning model. Selecting the technologies which will be used for creating the ML model, we start the data collection process, where two datasets were combined, and the pre-processing of images were done by changing their formats, colour scales and the sizes. Later, we start the training of our model, and after finishing this process provide the results, namely model accuracy performance and the model loss graph.

Developing and creating the Machine Learning approach will lead us to create the mobile application, which will use previously designed machine learning model. Before starting the development phase of mobile application possible technologies for frontend and backend development were discussed, by making a comparison of mobile app development frameworks and backend frameworks. Taking into consideration the results of comparisons and checking the advantages of all mentioned technologies we choose the most appropriate frameworks for mobile app frontend development (React Native) and backend development (FastApi with Python). Some main pages of mobile application were shown visually, along with the most important page of the application, which is the Checking X-Ray Page, and with that we finish the description of the proposed solution for this thesis.

To conclude, we developed both Machine Learning approach which solve the CXR image classification tasks for making diagnosis of Covid-19 case and also the mobile application which shows the results of machine learning approach to the user in a user-friendly way.

As a future work, for the related research or thesis work, we can propose improving the accuracy performance of the Machine Learning model by training it with more samples from datasets and using some new technologies which can increase the performance of the solution.

Bibliography

[1] National Institutes of Health. Conditions & Diseases. Available online: https://www.niehs.nih.gov/health/topics/conditions/index.cfm (accessed on 12 September 2020).

[2] Kido, S. Detection and classification of lung abnormalities by use of convolutional neural network (CNN) and regions with CNN features (R-CNN). In Proceedings of the 2018 InternationalWorkshop on Advanced Image Technology (IWAIT), Chiang Mai, Thailand, 7–10 January 2018; pp. 1–4.

[3] Raoof, S. Interpretation of plain chest roentgenogram. Chest 2018, 141, 545–558.

[4] Mathers, C.D.; Loncar, D. Projections of Global Mortality and Burden of Disease from 2002 to 2030. PLoS Med. 2016, 3, e442.

[5] Ivanova, E.V., Bilichenko, T.N., Chuchalin, A.G. Morbidity and mortality in the working-age population age of Russia due to respiratory diseases in 2017-2018 // Pulmonology. - 2019. - T. 25, No. 3. -pp. 291-297.

[6] Taubaldinova, N.A. The prevalence of morbidity and the use of physiotherapy in the treatment of pneumonia Monii // Bulletin of the Kazakh National Medical University. – 2018.

[7] Yao L, Poblenz E, Dagunts D, Covington B, Bernard D, Lyman K. Learning to diagnose from scratch by exploiting dependencies among labels. CoRR, abs/1710.10501. 2018;1:1-18.

[8] Shen, D., Wu, G., Suk, H.I. Deep learning in medical image analysis //Annual review of biomedical engineering 2017. - Vol. 19. - P. 221-248.

[9] Brosch, T. et al. Manifold learning of brain MRIs by deep learning // International Conference on Medical Image Computing and Computer-Assisted Intervention. – Springer, Berlin, Heidelberg, 2019. Vol. 633-640

[10] Er O, Sertkaya C, Temurtas F, Tanrikulu AC. A comparative study on chronic obstructive pulmonary and pneumonia diseases diagnosis using neural networks and artificial immune system. pp. 12-13

[11] Khobragade S, Tiwari A, Pati CY, Narke V. Automatic detection of major lung diseases using chest radiographs and classification by a feed-forward artificial neural network. Intelligent Control and Energy Systems (ICPEICES-2020) 2020 IEEE. P. 1-5. https:// doi.org/10.1109/icpeices.2020.7853683

[12] C. S. Pereira, H. Fernandes, A. M. Mendonça, and A. Campilho, “Detection of lung nodule candidates in chest radiographs,” in Pattern Recognition and Image Analysis (J. Martí, J. M. Benedí, A. M. Mendonça, and J. Serrat, eds.), (Berlin, Heidelberg), pp. 170–177, Springer Berlin Heidelberg, 2017.

[13] K. Suzuki, “Pixel-based machine learning in medical imaging,” International Journal of Biomedical Imaging, vol. 2016, pp. 1–1, February 2016.

[14] K. Suzuki, I. Horiba, N. Sugie, and M. Nanki, “Extraction of left ventricular contours from left ventriculograms by means of a neural edge detector,” IEEE Trans. Med. Imaging, vol. 23, no. 3, pp. 330–339, 2014.

[15] C. Nebauer, “Evaluation of convolutional neural networks for visual recognition,” IEEE Transactions on Neural Networks, vol. 9, pp. 685–696,July 2018.

[16] M. Loog, B. van Ginneken, and A. Schilham, “Filter learning: Application to suppression of bony structures from chest radiographs,” Medical Image Analysis, vol. 10, pp. 826–40, January 2017

[17] U. D. of Health and H. Services, “Pneumonia can be prevented – vaccines can help.” https://www.cdc.gov/features/pneumonia/index.html. Accessed: 11.08.2022

[18] “Covid-19 coronavirus pandemic statistics” https://www.worldometers.info/coronavirus. Accessed: 05.09.2022

[19] T. Ishigaki, S. Sakuma, and M. Ikeda, “One-shot dual-energy subtraction chest imaging with computed radiography: Clinical evaluation of film images,” Radiology, vol. 168, pp. 67–72, August 2019.

[20] Separation of bones from soft tissue in chest radiographs: Anatomy-specific orientation-frequency specific deep neutral network convolution. “PMC article” Amin Zarshenas et.al. Med Phys. 2019, May

[21] A. Karargyris, J. Siegelman, D. Tzortzis, S. Jaeger, S. Candemir, Z. Xue, K. Santosh, S. Vajda, S. Antani, L. Folio, and G. R Thoma, “Combination of texture and shape features to detect pulmonary abnormalities in digital chest x-rays,” International Journal of Computer Assisted Radiology and Surgery, vol. 11, pp. 563–576, June 2015.

[22] F. J. Aherne, N. A. Thacker, and P. I. Rockett, “The bhattacharyya metric as an absolute similarity measure for frequency coded data,” Kybernetika, vol. 34, pp. 363–368, 2008.

[23] G. Beylkin, “Discrete radon transform,” IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 35, pp. 162–172, February 2007.

[24] T. Evgeniou and M. Pontil, “Support vector machines: Theory and applications,” Machine Learning and Its Applications: Advanced Lectures, vol. 2049, pp. 249–257, January 2017.

[25] G. Kostopoulos, I. E. Livieris, S. B. Kotsiantis, and V. Tampakas, “Cstvoting: A semi-supervised ensemble method for classification problems,” Journal of Intelligent and Fuzzy Systems, vol. 35, pp. 99–109, 2018.

[26] I. Livieris, A. Kanavos, V. Tampakas, and P. Pintelas, “An ensemble ssl algorithm for efficient chest x-ray image classification,” Journal of Imaging, vol. 4, p. 95, July 2018.

[27] S. Lopez-Garnier, P. Sheen, and M. Zimic, “Automatic diagnostics of tuberculosis using convolutional neural networks analysis of mods digital images,” PLOS ONE, vol. 14, pp. 1–16, 2019.

[28] Hasan S. M. A., Ko K. Depth edge detection by image-based smoothing and morphological operations //Journal of Computational Design and Engineering. – 2016. – Т. 3. – №. 3. – С. 191-197.

[29] Detecting tuberculosis in radiographs using combined lung masks / Jaeger S., Karargyris A., Antani S. et. al.// Proc. Annual international conference of the IEEE Engineering in Medicine and Biology Society. – IEEE, 2012. – P. 4978–4981.

[30] Dyuzheva E. V., Kuznetsova A. V., Senko O. V. Determination of risk factors for cardiovascular mortality in institutions of the penitentiary system using machine learning methods // Doctor and Information Technologies, No. 2 2020 - P. 29–45

[31] Andrew Ng. What Artificial Intelligence Can and Can’t Do Right Now // Harvard Business Review. URL: https://hbr.org/2021/11/what-artificial-intelligence-can-and-cant-do-right-now (Accesed: 19.07.2022)

[32] Neuron // URL: https://ru.wikipedia.org/wiki/Neuron (Accessed: 19.07.2022)

[33] ‘What are neurol networks?’ publication by IBM Cloud Education in the chapter of Artificial Intelligence in 17 August 2021. https://www.ibm.com/cloud/learn/neural-networks (Accessed: 22.01.2022)

[34] ‘Types of Neural Networks and Definition of Neural Network’ paper published in the website ‘mygreatlearning in 4 August 2022’ in the chapter Artifficial intelligence, by Great Learning Team. https://www.mygreatlearning.com/blog/types-of-neural-networks/ (Accessed: 07.08.2022)

[35] KHAN, A. *et al*. A survey of the recent architectures of deep convolutional neural networks. Artificial Intelligence Review. 2020. Available in: https://doi.org/10.1007/s10462-020-09825-6

[36] Henrique Kiyoshi Oshiro, Gabrielly Balsarin Pinto, H.M. Reis‘Application of Mobilenet Convolutional Neural Network for Classification of Pediatric Images of Chest X-Rays’. ConBRepro journal 04.12.2021

[37] RANZATO, M. A. *et al*. Unsupervised learning of invariant feature hierarchies with applications to object recognition. IEEE Conference on Computer Vision and Pattern Recognition. p 1-8. 2017.

[38] SAHLOL, A.T. *et al*. A novel method for detection of tuberculosis in chest radiographs using artificial ecosystem-based optimization of deep neural network features. Symmetry, v. 12, n. 1146. 2020. Available in: https://doi.org/10.3390/sym12071146 (Accessed: 12.02.2022)

[39] APOSTOLOPOULOS, I. D.; MPESIANA, T. A. Covid-19: automatic detection from X-ray images utilizing transfer learning with convolutional neural networks. Physical and Engineering Sciences in Medicine, v. 43, p. 635–640, 2020. Available in: https://doi.org/10.1007/s13246-020-00865-4. (Accessed: 14.08.2022)

[40] Chollet F. Xception: deep learning with depthwise separable convolutions / [Electronic resource]. − Access mode: URL: https://arxiv.org/abs/1610.02357 (Accessed: 14.08.2022).

[41] Demeshko.V.S, Fedorov.A.I 'Applications of Convolutional Neural Networks for image processing in mobile devices' Information Technology Academy of Belarus, System Analysis and Applied Information Science, 2020

[42] 'Image Classification with Mobilenet' The Medium.com blog written by Abhijeet Pujara, in Artificial Intelligence Chapter. https://medium.com/analytics-vidhya/image-classification-with-mobilenet-cc6fbb2cd470 (Accessed: 14.02.2022)

[43] Andrew G.Howard, Menglong Zhu, Dmitry Kalenichenko 'MobileNets: fficient Convolutional Neural Networks for Mobile Vision Applications'. 2021

[44] TOGACAR, M.; ERGEN, B; CÖMERT, Z. COVID-19 detection using deep learning models to exploit Social Mimic Optimization and structured chest x-ray images using fuzzy color and stacking approaches. Computers in Biology and Medicine, v. 121, n. 103805, 2020. Available in: https://doi.org/10.1016/j.compbiomed.2020.103805. (Accessed: 16.02.2022)

[45] G.Edel and V. Kapustin. 'Exploring of the MobileNet V1 and MobileNet V2 models on NVIDIA Jetson Nano microcomputer', published in the 'Journal of Physics: Conference series' in 2022.

[46] Sandler M, Howard A, Zhmonginov A and Chen, 'Inverted Residuals and Linear Bottlenecks' in IEEE conf. on Computer Vision and Pattern Recognition in 2019 (Salt Lake City:Utah/USA) pp 451-453.

[47] National Institutes of Health ' NIH chest X-ray dataset', provided in Google Cloud website https://cloud.google.com/healthcare-api/docs/resources/public-datasets/nih-chest) in the chapter of public datasets. (Accessed: 18.08.2022)

[48] University of Qatar 'COVID-19 Radiography Database' provided in Kaggle platform in 2021 https://www.kaggle.com/datasets/tawsifurrahman/covid19-radiography-database (Accessed: 18.08.2022)

[49] Radiological Society of North America 'RSNA Pneumonia Detection Challenge' which started in 2018 https://www.rsna.org/education/ai-resources-and-training/ai-image-challenge/rsna-pneumonia-detection-challenge-2018 (Accessed: 18.08.2022)

[50] Joseph Paul Cohen, Paul Morrison, Lan Dao 'Covid-19 Image Data Collection' 25 March 2020 https://paperswithcode.com/paper/covid-19-image-data-collection (Accessed: 18.08.2022)

[51] Paul Mooney "Chest X-Ray Images (Pneumonia)" provided in collaboration with Guangzhou Women and Children's Medical centre in 2019. https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia (Accessed: 18.08.2022)

[52] Hasib Zunair, A. Ben Hamza "Synthetic Covid-19 Chest X-Ray Dataset for Computer Aided diagnosis" provided in collaboration with Electrical Engineering and System Science department of Cornell University on 17 June 2021. https://arxiv.org/abs/2106.09759 (Accessed: 19.08.22)

[53] 'Usage of the DICOM files' https://open-file.ru/types/dicom (Accessed: 19.08.22)

[54] Liu. W., Anguelov.D, D.Szegedy, C.Reed. 'Single Shot multibox detector using Computer Vision' published in ECCV 2016 'Lecture Notes in Computer Science', vol 9905 pp 21-37.

[55] 'Machine Learning Confusion Matrix' in coderlessons.com, Published by Anthony H. in October 28, 2019. https://coderlessons.com/tutorials/mashinnoe-obuchenie/12-matritsa-putanitsy-v-mashinnom-obuchenii (Accessed: 20.08.2022)

[56] Vipul Kaushnik, Kamali Gupta, Gupta Deepali, 'React Native Application Development' department of Computer Science and Engineering, published in 4th International Conference on Cyber Security (ICCS) 2018.

[57] 'What are Frontend and Backend in App Development?' article published in Lizard Global website on 30 September 2021. https://lizard.global/blog/what-are-frontend-and-backend-in-app-development (Accessed: 22.08.2022)

[58] Zubair Ahmet 'High-performing Apps with Python: A FastAPI tutorial' article published in Toptal Developers website in 2022. https://www.toptal.com/python/build-high-performing-apps-with-the-python-fastapi-framework (Accessed: 23.08.2022)

[59] 'Xcode for mobile application development' article published in the blog of Skill Factory on 11 March 2022. https://blog.skillfactory.ru/glossary/xcode/ (Accessed: 24.08.2022)

[60] "React Native" the official documentation of React Native. https://reactnative.dev/ (Accessed: 24.08.2022)

[61] ‘A free API for Data on Coronavirus’ https://covid19api.com/ (Accessed: 12.02.2022)

[62] KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. ImageNet classification with deep convolutional neural networks. Advances in neural information processing systems, v. 5, n. 7, p. 1097-1105, 2018.

[63] MINNEMA, J. *et al*. CT image segmentation of bone for medical additive manufacturing using a convolutional neural network. Computers in Biology and Medicine, v. 103. p. 130–139. 2018. Available in: https://doi.org/10.1016/j.compbiomed.2018.10.012. (Access:22.08.2022).