REPUBLIC OF TURKEY
YILDIZ TECHNICAL UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING

DETECTION OF LUNG DISEASES FROM RADIOGRAPHY IMAGES

18011092 — Arda KAŞIKÇI
19011065 — Elif MERTOĞLU

SENIOR PROJECT

Advisor
Dr. Ahmet ELBİR

June, 2023

ACKNOWLEDGEMENTS

We would like to express our gratitude to our esteemed professor, Dr. Ahmet ELBİR, for his invaluable support and experiences throughout the planning, research and implementation process of this study.

Arda KAŞIKÇI
Elif MERTOĞLU

TABLE OF CONTENTS

LIST OF SYMBOLS

F	Filter Size
L1	Lasso Regularization
L2	Ridge Regularization
Ms	Millisecond
P	Padding Size
Pe	Expected Prediction by Chance
Po	Observed Agreement Between the Classifiers
S	Stride
W	Input Image Size
Wout	Output Image Size

LIST OF ABBREVIATIONS

ANN	Artificial Neural Network
CNN	Convolutional Neural Networks
CPU	Central Processing Unit
GB	Gigabyte
KNN	K-Nearest Neighbour
KVKK	Kişisel Verilerin Korunması Kanunu
LSTM	Long Short-Term Memory
MLP	Multi-Layer Perceptron
RGB	Red-Green-Blue
RMS	Root Mean Square
RNN	Recurrent Neural Network
ROI	Region of Interest
STN	Spatial Transformer Network
SVM	Support Vector Machine
VGGNet	Visual Geometry Group Based Neural Network
XGBoost	Extreme Gradient Boosting
YOLOV3	You Only Look Once Version 3

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Detection of Lung Diseases from Radiography Images

Arda KAŞIKÇI
Elif MERTOĞLU

Department of Computer Engineering
Senior Project

Advisor: Dr. Ahmet ELBİR

The respiratory system is a fundamental system that meets the oxygen needs of the human body, and the lungs form the building blocks of this system. Lung diseases are serious conditions that can affect this system and cause life-threatening conditions. Therefore, early diagnosis of such diseases is of great importance for human health. One of the most commonly employed method in the detection of lung diseases is radiography-based X-Ray imaging technique. Doctors making wrong decisions while examining the images can lead to time loss and inefficient treatment processes. Therefore, more accurate and faster diagnosis processes will be possible through the effective use of various artificial intelligence methods, especially deep learning architectures. In this study, detection and classification of diseases on lung radiography images have been performed using hybrid models. Multiple pre-trained deep learning models have been trained. The performance of ResNet, EfficientNet and Inception architectures and different models within these architectures have been compared. High performance deep learning models were selected and used for feature extraction. The extracted features were tested with different machine learning algorithms and their performances were compared. Accuracy, precision, recall, f1-score and cohen's kappa performance metrics have been used to evaluate model performances. As a result of these performance evaluations, the hybrid model with the highest accuracy value for the data set has been determined and with this model, a decision support system has been implemented to predict lung diseases such as covid-19, pneumonia, opacity in the relevant person over lung radiography images.

Keywords: Deep Learning,Lung Diseases,Classification,Radiography,Hybrid Models

ÖZET

Radyografi Görüntülerinden Akciğer Hastalıkları Tespiti

Arda KAŞIKÇI
Elif MERTOĞLU

Bilgisayar Mühendisliği Bölümü
Bitirme Projesi

Danışman: Dr. Ahmet ELBİR

Solunum sistemi, insan vücudunun oksijen ihtiyacını karşılayan temel bir sistemdir ve akciğerler, bu sistemin yapı taşını oluşturmaktadır. Akciğer hastalıkları bu sistemi olumsuz etkileyerek hayati tehlike yaratabilecek ciddi rahatsızlıklardır. Bu nedenle, bu tür rahatsızlıklarda erken teşhis insan sağlığı açısından önem taşımaktadır. Akciğer hastalıklarının tespitinde kullanılan en yaygın yöntemlerden birisi radyografi temelli X-Ray görüntüleme tekniğidir. Doktorların görüntüleri incelerken hatalı kararlar vermesi zaman kaybına ve verimsiz tedavi süreçlerine neden olabilir. Bu nedenle, tanı süreçlerinin daha doğru ve hızlı yapılabilmesi derin öğrenme mimarileri başta olmak üzere çeşitli yapay zeka yöntemlerinin etkin kullanımı ile mümkün olacaktır. Bu çalışmada, akciğer radyografi görüntüleri üzerinde hastalıkların tespiti ve sınıflandırılması hibrit modeller kullanılarak yapılmıştır. Birden fazla önceden eğitilmiş derin öğrenme modeli eğitilmiştir. ResNet, EfficientNet ve Inception mimarileri ve bu mimariler içindeki farklı modellerle performans karşılaştırılması yapılmıştır. Özellik çıkarımı için yüksek performanslı derin öğrenme modelleri seçilmiş ve kullanılmıştır. Çıkarılan özellikler farklı makine öğrenmesi algoritmaları ile test edilmiş ve performansları karşılaştırılmıştır. Model performanslarının değerlendirilmesinde accuracy, precision, recall, f1-score ve cohen's kappa performans metrikleri kullanılmıştır. Bu performans değerlendirmeleri sonucunda veri seti için en yüksek doğruluk değerine sahip hibrit model belirlenmiş ve bu model ile akciğer radyografi görüntüleri üzerinden ilgili kişide bulunan covid-19, zatürre, opasite gibi akciğer hastalıklarını tahmin edecek bir karar destek sistemi gerçekleştirilmiştir.

Anahtar Kelimeler: Derin Öğrenme,Akciğer Hastalıkları,Sınıflandırma,Radyografi,Hibrit Modeller

1
Introduction

Pulmonary diseases pose a significant threat to our world today. It is reported that from 2003 to 2017, more than 2.2. million deaths were primarily attributed to chronic lung disease [1]. There exists many factors that cause to lung diseases including but not limited to smoking, air pollution, genetics, infections, occupational exposure, and allergies. Since people are exposed to some of these agents in their daily lives, it is important to take precautions and detect lung diseases before it becomes fatal.

One of the most important key step in detecting the lung diseases is to use radiography techniques. Under this radiography techniques, the X-ray screening is one of the most popular methodologies in detection of respiratory system diseases [2]. The images obtained from X-ray scanning are then sent to doctor for evaluation and detection. However, the detection of lung diseases is a specialized field within medicine which needs specific expertise and training. In order to alleviate the burden of diagnosis on the medical expert and allow them to focus on the treatment process, computer aided decision support systems can be used.

In this study, it is aimed to provide an automated method for detecting the lung diseases from radiography images and at the same time to implement a system that can be used in real-world diagnosis process. With this motivation, some of the image processing, convolutional neural networks (CNN), and machine learning techniques are researched and decided that hybrid modelling is the appropriate method for executing this project.

A convolutional neural network (CNN or ConvNet) is a network architecture for deep learning that learns directly from data [3]. CNN models are quite useful for finding patterns in images for both classifying and object detection, and mostly used for images and videos. CNN is great for feature extraction and has been shown to be very competent at finding patterns thought to be difficult for traditional methods like manual feature extraction or image processing methods.

The purpose of using the hybrid model is to take advantage of the successful results of machine learning methods in classification after successful feature extraction of CNN models. Shortly, hybrid modelling gives a way to reveal a more powerful and combined model by using different techniques under artificial intelligence and use them where they are the most successful.

As a part of the implementation, the COVID-19 Radiography Database is used from Kaggle repository. The data set consists of 4 different lung disease categories with total of 21165 images. Different CNN models are comparatively applied to the above mentioned data set.

In addition to these, the paper can be organized as follows: In Chapter 2, previous studies in this field is evaluated. Then, the feasibility of the project is made in the 3rd chapter. In the 4th and 5th chapters, the analysis and design of the system is explained, and in the 6th chapter the execution of the application is shared. In the 7th chapter the experimental findings are obtained, in the 8th chapter the performance analysis is evaluated and the conclusion part is discussed in chapter 9.

2
Preliminary Review

At the beginning of this project, we have done some literature research to better understand the techniques used in image classification and feature extraction. Since our data set consist of radiography images, we wanted to specifically look into other X-ray-based-image-classification problems and be aware of the methods used for this specific image type. According to our research, we have seen that with different classification models, other methodologies like image segmentation, image processing, ROI extraction, object detection and texture analysis are also used. Below, these studies are analyzed.

2.1 Literature Review

A neuro-heuristic approach for recognition of lung diseases from X-ray images [2]: In this study conducted by Qiao Ket and his team, image descriptors based on the spatial distribution of Hue, Saturation and Brightness is used with a neural network co-working with heuristic algorithms. Firstly, neural network checks whether there is a possibility of a lung disease and afterwards heuristic algorithms are used to identify the degenerated tissue in detail.

Automatic detection of major lung diseases using Chest Radiographs and classification by feed-forward artificial neural network [4]: In this study of Shubhangi Khobragade and her team, intensity based and discontinuity based image segmentation techniques are used to obtain lung boundaries. After this process, artificial neural network models are used to detect lung disease.

Image feature analysis and computer-aided diagnosis in digital radiography: Detection and characterization of interstitial lung disease in digital chest radiographs [5]: In this study conducted by Shigehiko Katsuragawa and her friends, lung texture is firstly obtained using two-dimensional Fourier transform.

The magnitude and fineness of these lung textures are then quantified by the RMS variation and first moment of the power spectrum. It is seen that the texture of abnormal lungs clearly differ from normal lung's textures.

Table 2.1 Methodology of Other Studies in the Literature

Ref.	Classes	Dataset	Methods
Bharati et al. [6]	15 classes with 1 'no findings' and 14 diseases	The NIH chest X-ray image dataset	A hybrid deep learning method called VDSNet including CNN, VGGNet, and STN is used with an accuracy of 73%.
Chen et al. [7]	bronchitis, bronchopneumonia, lobar pneumonia, pneumothorax, and normal	Not shared	The study firstly focuses on obtaining the appropriate boundaries of lungs using YOLO3. Afterwards, it uses 3 different methods for classification listed as: 1v1 scheme, 1vAll scheme and a CNN model.
Goyal et al. [8]	Covid-19, viral pneumonia, bacterial pneumonia, and normal	The C19RD and CXIP data set.	In this study, image quality enhancement and region of interest estimation is used for the lung detection part. After detection of the lungs, ANN, SVM and KNN are implemented for the classification but the highest accuracy is obtained through RNN with LSTM.

3
Feasibility

3.1 Technical Feasibility

3.1.1 Software Feasibility

The project is based on classification. Lately, convolutional neural network and transfer learning models are getting popular for classification problems. It is known that Python is popular and useful in this field, and Python is used in the project due to the existence of deep-learning-specific libraries. In the project, the Python libraries Keras and Scikit-learn were used to build, train and evaluate the deep learning models.

3.1.2 Hardware Feasibility

The coding environment is chosen to be Google Colab because of its ease-of-use and GPU support. Considering the data size and the log files of deep learning models, a storage space of 10 GB is found to be sufficient. When evaluating the memory space used while training the models, memory space of 15 GB is found to be sufficient.

3.2 Legal Feasibility

As the data set used in the project has been publicly shared, there are no legal obligations. The images used have not been associated with any individuals and do not pose a risk of violation of the Law on the Protection of Personal Data (KVKK). Therefore, there is no legal noncompliance in terms of feasibility under these conditions in the project.

3.3 Schedule Feasibility

Gantt chart for schedule feasibility is as in the Figure 3.1

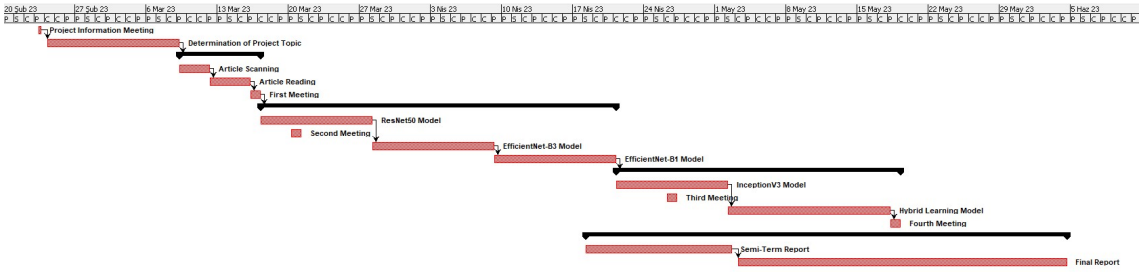

Figure 3.1 Gantt Chart for Schedule Feasibility

3.4 Economic Feasibility

Electricity consumption of personal computers and 2-month Google Colab subscription (160 TL/Month) are included as expenses within the scope of the project. No income was generated within the scope of the project.

4
System Analysis

Throughout the research that we have carried out, the supervised learning method has been found to be the most appropriate method for our problem of image classification. In supervised learning, the data set is given to a model as pre-labeled and the predictions are done using those labels. Using labeled inputs and outputs, the model can measure its own accuracy and learn over time.

During preliminary review we had the opportunity to review other studies that used supervised learning models such as but not limited to AlexNet, ResNet, EfficientNet, GoogleNet, SVM, RNN, CNN, ANN and many more. Following our research, we have decided to continue with the CNN architecture and its models, also combining our feature maps obtained from CNN with machine learning models, implementing a new hybrid model for our classification problem.

CNN's ability to extract features without losing information about neighbouring pixels makes it an excellent model for images. Having obtained features from our most successful CNN model and using our knowledge from the literature review, we decided to feed these features into a machine learning model for the classification part. We adopted a machine learning model for classification as it exhibits superior efficiency in identifying class labels and possesses greater capability in managing non-linear relationships as compared to MLPs.

In our project, we have used ResNet50, EfficientNetB1, EfficientNetB3, InceptionV3 and their different variants changing hyperparameters such as batch size, number of epochs, learning rate and so on. In all of these models, we have adopted transfer learning method due to time constraints and the size of our data set. Once we have obtained the feature maps, we have fed our data into the Random Forest, XGBoost, SVM, KNN and get the result of the classification. For the model evaluation part, we will use F1-Score, confusion matrix, precision, recall, support and Cohen's Kappa. Cohen's Kappa Score is used when the different classes have different frequencies and since our data set is not balanced, it is an important metric for our project.

5
System Design

5.1 Database Design

The dataset used in the project was taken from Kaggle repository and is publicly shared. It consist of 21,165 RGB chest X-ray images with 4 different classes. The classes and the total number of images in each of them is shown in Table 5.1.

Table 5.1 Dataset Class Distribution

COVID-19	Lung Opacity	Viral Pneumonia	Normal
3616	6012	1345	10,192

There exists an image mask for every image in a category. Image masks are used to define specific areas in an image to help machine learning algorithms better recognise different features, thereby increasing the accuracy of the algorithm. Figure 5.1 shows some examples from the dataset.

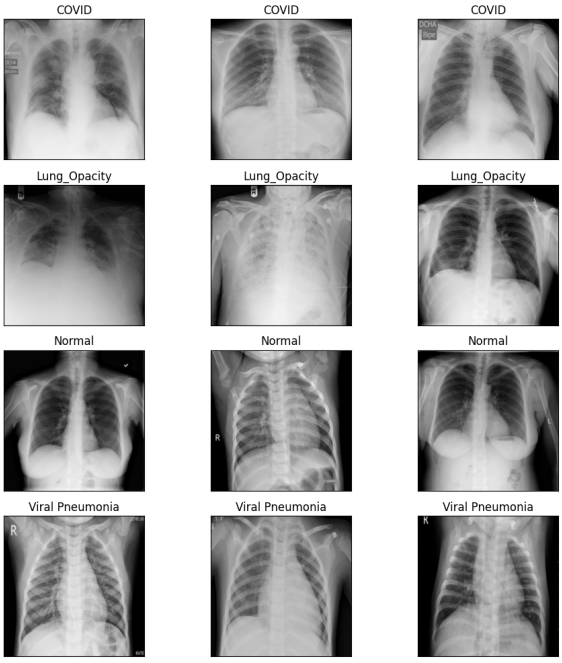

Figure 5.1 Random Images from Dataset

5.2 Software Design

This section explains the methods used for feature extraction and deep learning in subsection 5.2.1. Classification part using machine learning algorithms is explained in subsection 5.2.2. The models are: ResNet-50, EfficientNet, Inception_V3, Random Forest, XGBoost, SVM and KNN.

5.2.1 Feature Extraction

Convolutional Neural Network is an architecture for deep learning algorithms. It's working logic is similar to human brain's. Just like in the human brain, the perceptrons are interconnected, and different perceptrons get different weights for different features, allowing them to recognise those features when they see them after.

CNN differs from other neural networks with its convolutional layers and these layers increase its complexity. For the classification part, it basically uses classical neural networks, and for the feature and pattern extraction part, it uses convolutional layers. The first layers of convolutional layers are for extracting simple and more basic patterns, followed by more complex features.

CNN distinguishes one image from another by assigning different importance values to regions of the image it receives as input. Here, different importance values emphasise learnable parameters, which are basically the values of the image matrix and its channels. These parameters, referred to in the literature as 'weights and biases', change continuously at certain layers during training, allowing the model to learn. The change occurs when applying different image filters which are used to extract features.

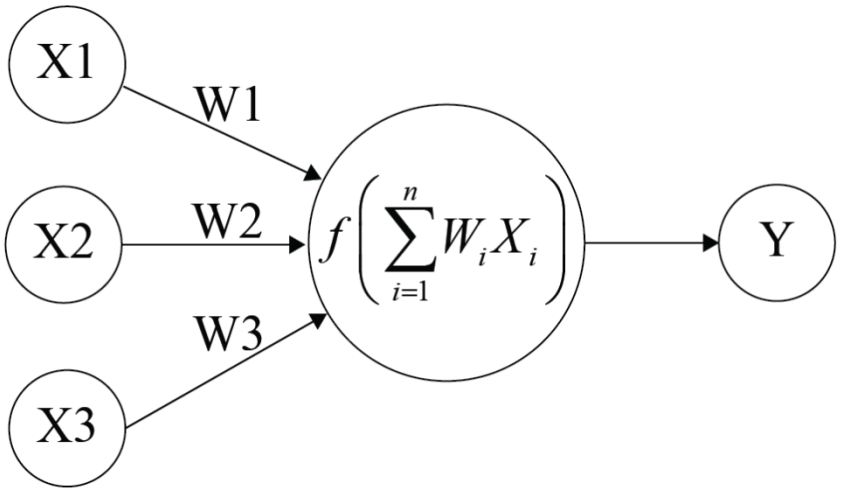

Figure 5.2 Weights and Biases [9]

Convolutional Neural Networks consist of three main layers: Convolutional, Pooling and Padding. Convolutional layers are the building blocks for a CNN model. In this layer, scalar matrix multiplication is executed between two matrices. The first matrix here represents input image and the second matrix is image filters, also known as kernels, which are relatively smaller than input image and used for feature extraction.

Convolutional neural networks have sparse interactions. This is provided by making the filter size smaller than image size. For a computer to understand an image, not all the pixel values are important and used. By using small filters, the computers can obtain the meaningful information from an image and this will affect the CPU and memory usage in a better way.

In an iteration, the filter matrix slides through the input image matrix, multiplying each matched cell and summing the results. The result is then put in the middle cell of the filter-placed image matrix. The step size as the filter slides across the image is called a *stride* [10]. Figure 5.3 shows the implementation of convolution operation.

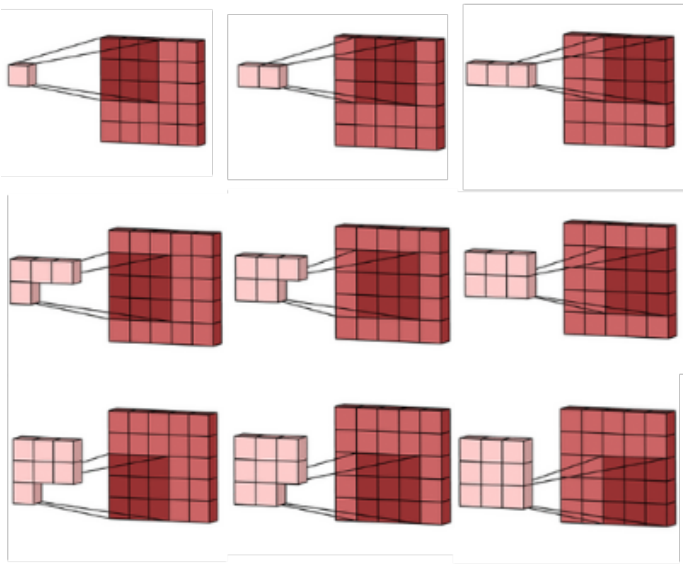

Figure 5.3 Convolution Process

One of the other main layers mentioned above is padding layer. It is seen in Figure 5.3, the convolution operation shrinks the input image size and while the values in the middle parts of the input are used many times in the scalar product, the values in the corners are used relatively less. In order to prevent this, the image is filled with values according to a certain rule, therefore making all the pixels are used equally. Figure 5.4 shows an implementation of zero padding.

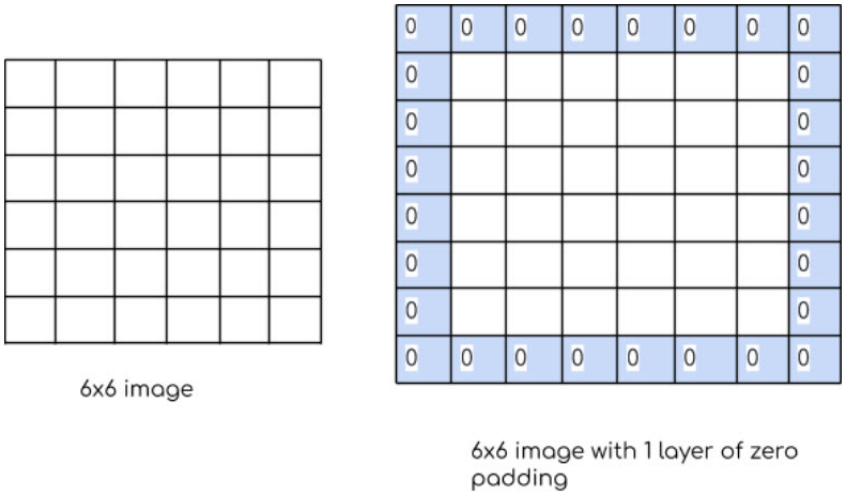

Figure 5.4 Zero Padding [11]

In order to calculate output image size after applying a filter to an input image, the below equation can be used. W, F, P and S represents input image size, filter size, padding and stride respectively.

$$W_{\text{out}} = \frac{W - F + 2P}{S} + 1 \tag{5.1}$$

The other main layer of the CNN can be considered as the pooling layer. Pooling layers are used to shrink the image size, helping to reduce required computational power. Shrinking is done by taking only the important and dominant features of the image, therefore does not cause any information loss. These layers usually have two parameters. The first one indicates the size of the frame to be shrinked and the second one indicates the stride. Figure 5.5 shows an example to max-pooling.

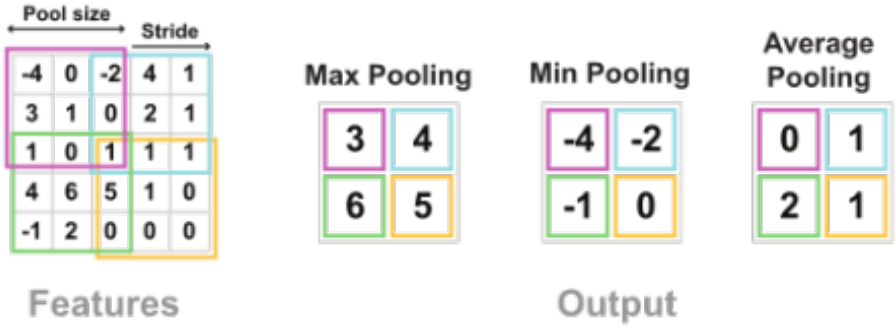

Figure 5.5 Max, Min and Average Pooling [12]

Below equation shows the output image size after applying pooling layer.

$$W_{\text{out}} = \frac{W - F}{S} + 1 \tag{5.2}$$

5.2.1.1 ResNet-50

ResNet50 belongs to ResNet model family. The number 50 emphasise the number of layers used in the model. ResNet50 has 48 convolutional layers along with 1 average pool and 1 max-pool layer. The model is considered to be the one of the most popular CNN architectures around and firstly introduced by Microsoft Research in 2015 [13]. The model's architecture is shown in Figure 5.6.

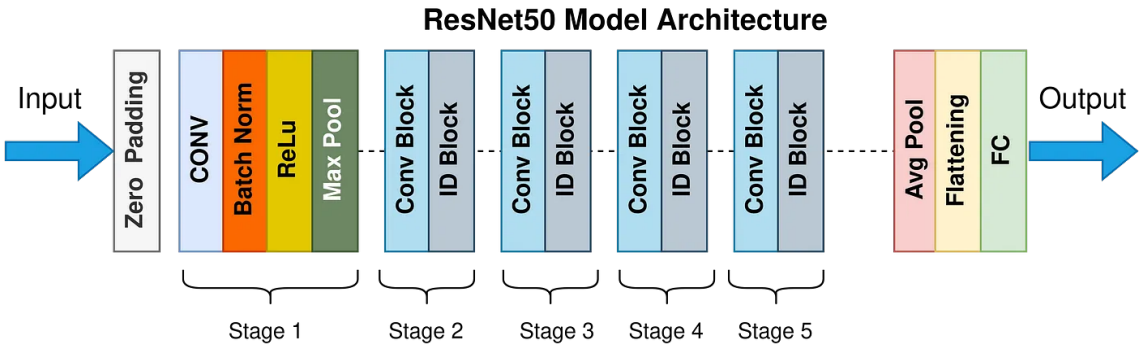

Figure 5.6 ResNet-50 Model Architecture [14]

As can be seen in Figure 5.6, in some stages there exists ID Blocks. These blocks represent *residual blocks*. These blocks are added to the network due to the vanishing/exploding gradient problem. In these layers, skip connections are used which are basically skipping some layers in the network and feeding the output of one layer as the input for the further layers. This way, during the backpropagation phase, some of the gradient calculations will be skipped and the input will be added to latter layers, preventing a vanishing gradient problem if the latter layer value is too small. Basically, if a layer causes degradation in the model's performance, it gets skipped.

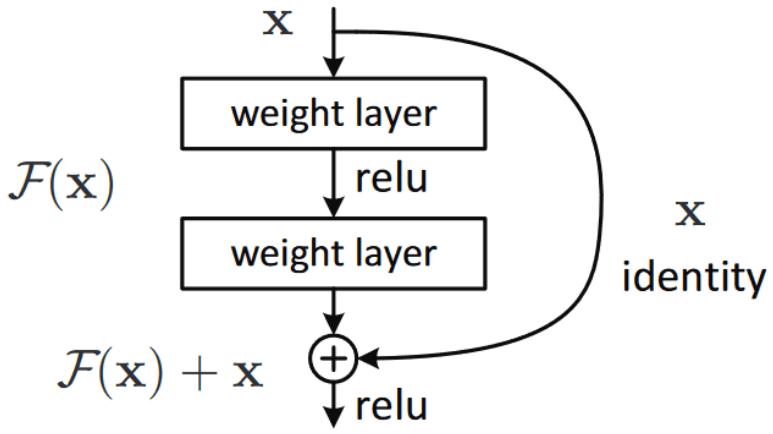

Figure 5.7 A Residual Block [15]

Figure 5.7 shows that in ResNets, the information from the initial layers is passed to deeper layers by matrix addition [16]. This allows lower-level semantic information from earlier layers to be preserved in latter layers.

5.2.1.2 EfficientNet

Model scaling is a technique used by developers when there are more computational resources available. Until EfficientNet, scaling was completely random in all dimensions. Some models were scaled depthwise, some were widthwise and some of them were just scaled the model by taking higher resolution input images. It has been observed that randomized scaling in this way does not add much effect to the model performance, but also increases the complexity of the model and the required computational power. To address this issue, Mingxing Tan and Quoc V. Le from Google Research Brain team have proposed EfficientNet in 2019 [17].

EfficientNet uses a method called *compound scaling*, which essentially scales the model uniformly in depth, width and resolution using a fixed set of scaling coefficients also known as compound coefficients.

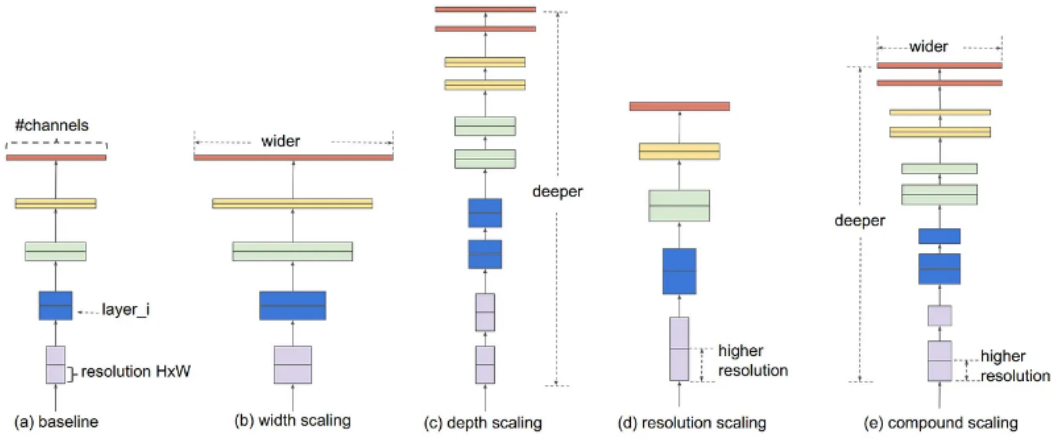

Figure 5.8 Different Scaling Techniques vs. Compound Scaling [18]

EfficientNet has 8 different models from B0 to B7 and the differences between the models come from the variants of the compound coefficients used. In our project, pre-trained EfficientNet_B0 and EfficientNet_B3 are comparatively used.

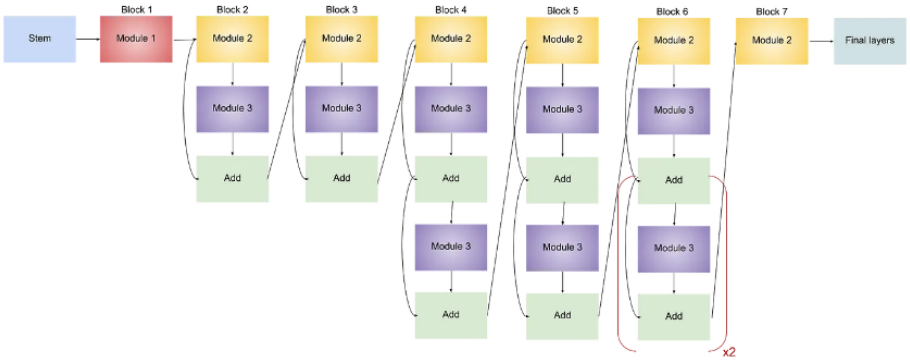

Figure 5.9 EfficientNet_B0 Architecture [19]

5.2.1.3 Inception_V3

What makes Inception one of the most successful models of CNN architecture is its ability to use parallel computation methods. Based on this concept, several versions have been developed under this model, including subversions under some of them. These versions are as follows: Inception v1. Inception v2 and Inception v3, Inception v4 and Inception-ResNet [20]. During our research, we decided to use Inception_V3 for our project because of its performance and light weight. Figure 5.10 shows an example to parallel computation.

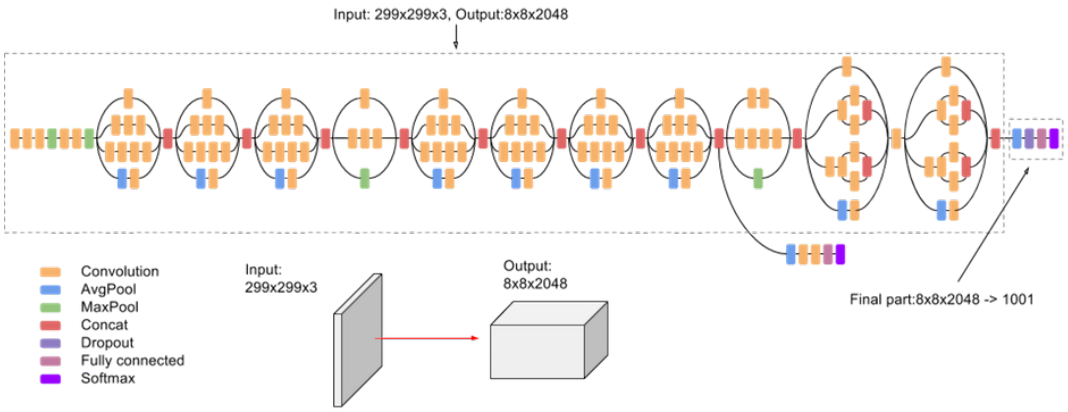

Figure 5.10 Inception_V3 Architecture [21]

There are different techniques used in Inception_V3. These are examined below [22].

- **Factorized Convolutions**: This method is used to reduce the number of parameters used by the model by converting large filters into smaller, but multiple filters.
- **Asymmetric Convolutions**: This approach works by combining filters of different sizes. This makes it more efficient to adapt to the size of the input image and also reduces the parameter size.
- **Auxiliary Classifier**: This procedure can be thought as an regularizer to network. It helps the model to make intermediate predictions, which leads to a better generalization performance.
- **Grid Size Reduction**: This technique is used to reduce the size of the feature maps, thereby reducing the amount of processing required. Stride and pooling operations are used.

5.2.2 Classification

Classification is a supervised learning technique used to classify new samples into a pre-defined set of categorised labels. There are various classification algorithms that are responsible for finding patterns in a given training dataset, and then trying to find those patterns in future unseen datasets called the 'test set'.

Classification can be done in four different ways according to the problem and the dataset [23]. These are explained below.

- **Binary Classification**: The main goal here is to classify data into one of two possible classes or categories. Although not necessarily, these two categories are usually referred to as 'negative class' and 'positive class'.
- **Multi-Class Classification**: In multi-class classification, there are more than two classes for categorizing input samples. Instances are grouped into multiple preset classes.
- **Multi-Label Classification**: In this classification task, the used algorithm may predict more than one labels for a given input instance. Object detection for image dataset can be an example to this.

Since our problem is to predict only one lung disease for an input, and due to the fact that there are more than two lung disease categories in our dataset, our task fits multi-class classification.

There are also two types of learners in classification task. These are called 'lazy learners' and 'eager learners', and the differences between them are explained below [23].

- **Lazy Learners**: In using lazy learners, the training-time-taken is usually smaller than the predicting-time because the classification is done using the training set's most appropriate data.
- **Eager Learners**: In eager learners, the training dataset is used to build a classification model, resulting much longer time taken in training part than the prediction part.

Since we want to use models that build a function representing training data to predict test data, all the models analysed in this document will be of the eager learner type.

5.2.2.1 Random Forest Classifier

In a random forest classifier, the algorithm constructs multiple decision trees, each of which is different because they are trained on a different subset of the training data and features. The prediction is made after calculating the results of all the trees and taking the most popular one. With this feature, the model is considered to be an 'ensemble', i.e. a combination of several models.

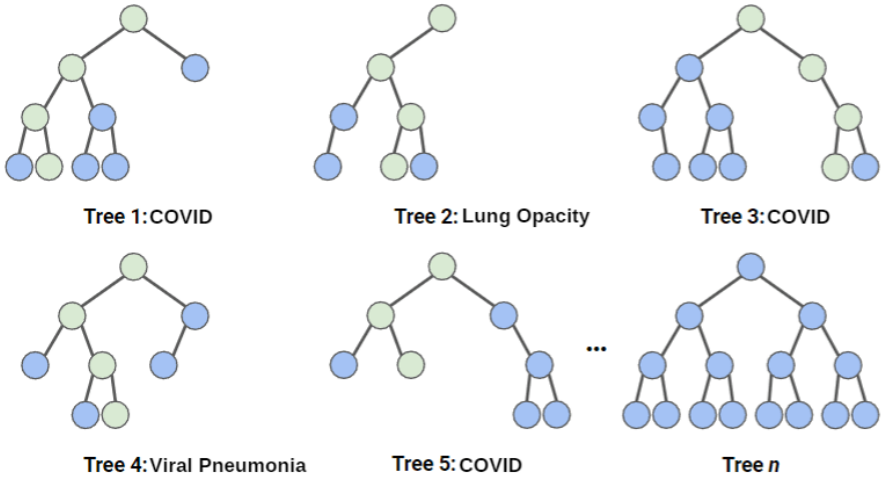

Figure 5.11 Random Forest Classifier Decision Trees [24]

There are random-forest-classifier-specific features and they are analyzed below [25].

- **Diversity**: Each tree is built different due to not using all the attributes, features or input samples.
- **Parallelization**: The trees are built seperately from each other with different subsamples of training dataset, so the CPU can be used fully with its cores.
- **Stability**: The results are stable because the result is based on majority of the voting.
- **Immune to the Curse of Dimensionality**: Since the trees are constructed using different combinations of subsampled features, the feature space is reduced.

Random forest differs from decision tree algorithm by having collection of decision trees. Therefore, the problem of overfitting in decision trees is taken care of because of the subsample usage, but random forest algorithm is comparatively slower because of the multiple decision trees.

5.2.2.2 XGBoost

XGBoost firstly proposed by the researchers at the University of Washington in 2016 in the paper titled "XGBoost: A Scalable Tree Boosting System" [26]. To gain a full understanding of the XGBoost algorithm, it is essential to understand the basic concepts of *bagging* and *boosting*, as they provide the necessary building blocks for this powerful machine learning technique.

- **Bagging**: In this approach, multiple models, which are mostly decision trees, are trained independently from each other using different randomly selected subsets of the training set. This randomized subsets are created through a process named *bootstrapping*. The models can be trained in parallel, and by having multiple models, the method becomes *ensemble*. Bagging helps reducing the variance of the model by introducing dissimilarity through different subsets of data, therefore reducing overfitting. The prediction results are obtained combining all models' results by averaging or voting.

- **Boosting**: Boosting is also an ensemble method meaning multiple models, mostly decision trees, are being used. The difference is hidden in creating these models. In this approach, multiple weak learners are sequentially trained and each following learner focuses on correcting the mistakes of the previous learners. Therefore causing to reduce the bias and variance of the model. The mentioned model XGboost belongs to this approach.

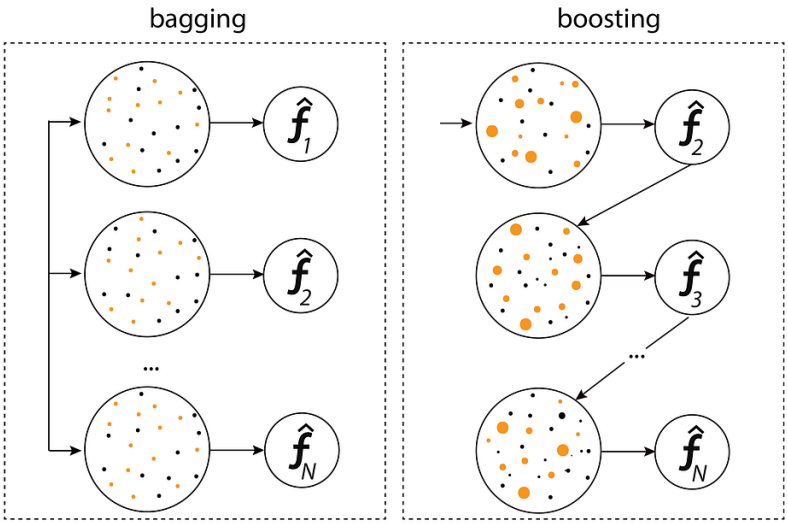

Figure 5.12 Bagging vs. Boosting [27]

From the past to the present, the evaluation of tree-based algorithms has evolved. The evolution process firstly started with *decision tree*, a single tree deciding the outcome.

Followed by *bagging*, multiple decision trees (models) ensembling to create final output by averaging or voting. Then *random forest* has emerged as a bagging-based algorithm with a difference, the randomized subset of features for each decision tree. After as an opponent to bagging, *boosting* has emerged, using feedback-based approach for finding the prediction result. Another method called *gradient boosting* added to this group, minimizing errors by gradient descent algorithm. Last but not least XGBoost has emerged as a better version of gradient boosting with it's hardware and software optimizations, giving a think of it's name 'Extreme Gradient Boosting.'

XGBoost has both system and algorithmic optimizations shown below [28].

- **Parallelization**: Although tree-building is done sequentially, XGBoost has a way to speed up this process by parallelizing it. The process of constructing trees involves two nested loops. The outer loop cannot be finished before the inner loop but inner loop takes more computational time. To overcome this issue, XGBoost employes a technique called loop interchange. By initializing the process with a global scan, the order of the loops are changed therefore XGBoost can start processing the outer loop without having to wait for the inner loop to complete its process.

- **Regularization**: Regularization is used to make the model more general and thus prevent over-fitting. It has L1 (also known as Lasso regularization) and L2 (also known as Ridge regularization) regularizations. L1 makes the model focus more on the subset of important features and L2 reduces the impact of the less important features. These regularizations can be controlled by *alpha* and *lambda* hyperparameters respectively.

- **Sparsity Awareness**: XGBoost is capable of handling sparse data by learning the best strategy for dealing with zero values in each feature during training time.

- **Weighted Quantile Sketch**: XGBoost uses weighted Quantile Sketch algorithm to find the best split points on weighted data points. Splitting is crucial for creating effective and meaningful branches in a tree.

- **Hardware Optimization**: XGBoost uses hardware components the most efficient way possible. It does this by using cache logic, puts all threads an internal buffer to store gradient statistics. Therefore the pre-calculated values will be reused and redundant calculations will not occur. Another technique it offers is 'out-of-core' computing. When dealing with large datasets, this method allows it to process data in smaller and managable chunks.

5.2.2.3 Support Vector Machine - SVM

Support Vector Machine was firstly introduced in 1995 by Corinna Cortes and Vapnik but the idea behind it belongs to earlier years [29]. Support Vector Machine can be used for the classification of both linear and non-linear data.

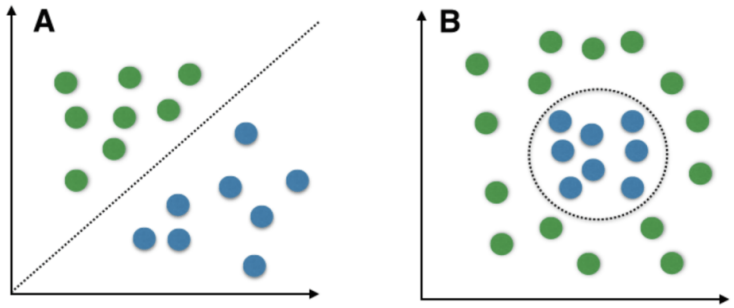

Figure 5.13 Linear and Non-Linear Data Separation [30]

The division of two classes are done by drawing a line which separates them from each other. These line is called *hyperplane*. When given a dataset, one may find infinitely many hyperplanes but the thing is to find the optimal one.

According to SVM, firstly the points closest to the line is found. These points are called *support vectors*. After, the distance between these points and the line is calculated, this value is *margin*. In order to find the most optimal hyperplane, the margin should be maximized.

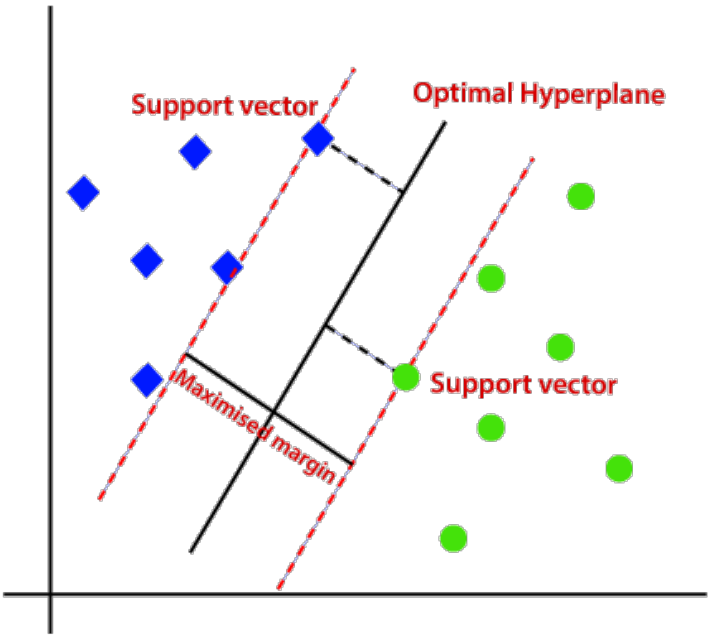

Figure 5.14 Support Vector Machine [31]

When thinking of non-linear data, a linear line is not possible to separate the classes. To classify the data, adding extra dimension to it helps to separate it. In Fig 5.15 on the left hand side, non-linear data can be seen. This data converted to linearly-separable in a higher dimension by simply adding a third dimension.

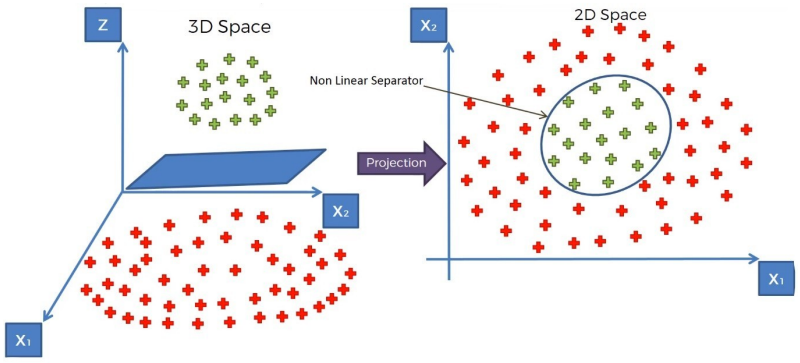

Figure 5.15 Classifying by Adding Extra Dimension [32]

There are two hyperparameters of Support Vector Machine algorithm. The first one is called *cost* and the second one is *gamma*. These two will be analyzed below.

- **Cost**: There can be different hyperplanes for a given dataset. When support vectors are very close to each other in a dataset and especially when these vectors are intertwined, there exist two approaches. One would be choosing a straight line and accepting there will be some points misclassified, or choosing a line more wiggly and intricated letting the model guess all the training data correctly. There is a trade-off here because if the line is chosen very complicated and wiggly, then the model's generalization ability reduces and if a simple straight line is chosen, then generalization increases but there will be error. This is where cost value comes into play. Large value of cost means the line will be more intricate trying to fit in all the points, and small value of cost means the line will be more straight allowing some misclassifications but more generalized.

- **Gamma**: Gamma value defines how far a single data point in the training set can have influence on the fitting line. If the value of gamma is large, then the decision boundary will be mostly affected by the points which are very close to the line, also known as support vectors. On the other hand, if the values of gamma is low then even the far points get considerable amount of weight and a more linear curve is obtained.

5.2.2.4 K Nearest Neighbors - KNN

The K-Nearest Neighbours algorithm was first introduced by Peter Hart in 1967 in the paper *The Condensed Nearest Neighbor Rule* [33]. The main concept lies behind KNN algorithm is, when a point is given which its class name is unknown, we can determine the points in our feature space which are closest to it. The number of closest points we are taking into consideration is determined by the K value.

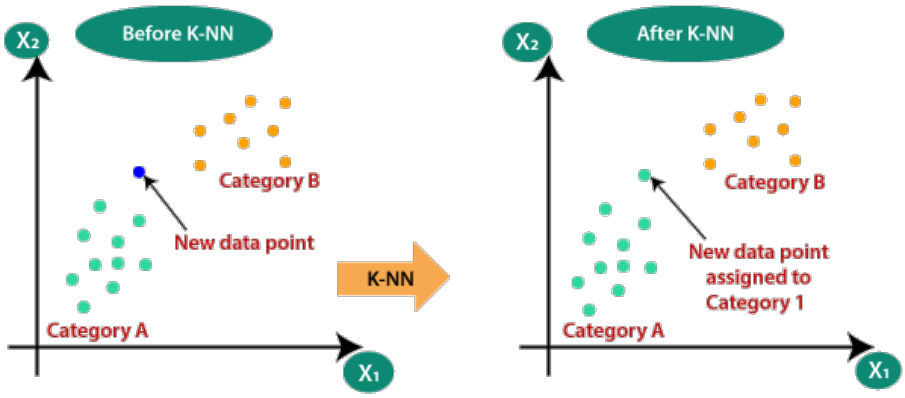

Figure 5.16 K-Nearest Neighbors [34]

Since similar points occupy similar locations in feature space, it is likely that our unknown new data point is type of the majority of its k-neighbors. The algorithm goes to feature space and checks distance with every point in the training set every time a new data point comes. This feature makes the algorithm a *lazy learner*.

In order to determine the k value, there are several approaches.

- **Brute-force Method**: In practice, K value is chosen an odd number between 3 and 10. Trying out different k values in this range and selecting the most successful one can be used to determine K value.
- **Square Root Method**: The best K value can be obtained by taking the square root of the number of samples in the training set.
- **Cross Validation Method**: By splitting the available data into train and validation, for different values of K, the one with the best result on validation data can be choosed.

To calculate the similarity (distance) between data points, several techniques can be used. These are Euclidean distance, Manhattan distance, Minkowski distance, Hamming distance and so on.

5.3 Input - Output Design

Performance metrics are used to evaluate and compare model performance. In this study, different metrics were used to evaluate the models from different perspectives after model training. These metrics are accuracy, precision, recall, f1-score, and cohen's kappa. The confusion matrix is the basis of the metrics used in this project. Therefore, in order to calculate the metrics accurately, it is necessary to generate the confusion matrix first. Confusion matrix structure is shown in the Figure 5.17

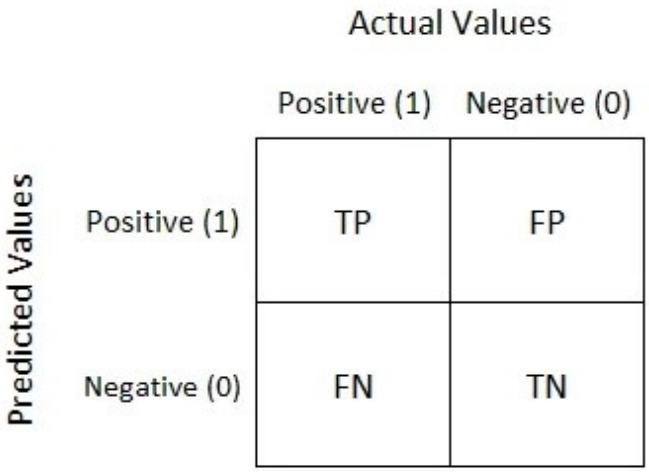

Figure 5.17 Confusion Matrix Structure

The values shown in the Figure 5.17 are:

- **True Positive (TP)**: The predicted class is positive and the prediction is true.
- **False Positive (FP)**: The predicted class is positive and the prediction is false.
- **True Negative (TN)**: The predicted class is negative and the prediction is true.
- **False Negative (FN)**: The predicted class is negative and the prediction is false.

The confusion matrix is created after these specified values are counted. Some metric equations calculated after the confusion matrix creation are as follows:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{5.3}$$

$$Precision = \frac{TP}{TP + FP} \tag{5.4}$$

$$Recall = \frac{TP}{TP + FN} \tag{5.5}$$

The above accuracy value is equal to the ratio of correctly predicted values to all predicted values. The precision value gives the ratio of how many of the values predicted as correct positive are correct and the recall value gives the ratio of how many of the values that should be predicted positively are correctly predicted. Considering the potential overfitting scenario, it is desirable for the values of these metrics to be as high as possible. After calculating the precision and recall metrics, the f1-score value is calculated as follows:

$$F1-Score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{5.6}$$

The above f1-score value is defined as the harmonic mean of precision and recall. Accuracy value is suitable when the class distribution in data set is balanced. If we have imbalanced data f1-score metric gives more accurate performance analysis then accuracy. It is because while accuracy metric gives more attention to true positives and true negatives, f1-score gives priority to false negatives and false positives. The formula of cohen's kappa, another metric used in the project, is as follows:

$$Cohen'sKappa = \frac{P_o - P_e}{1 - P_e} \tag{5.7}$$

The above cohen's kappa is a measure of the agreement between two raters. The cohen's kappa value can range from -1 to 1. Cohen's kappa score is one of the widely used performance evaluation metrics in medical diagnostic models [35]. The Po value in the equation represents the observed agreement between the classifiers, and the Pe value represents the expected prediction by chance.

The five metrics discussed in this section were used to evaluate the models. The results of these metrics played a key role in identifying the most successful model based on its performance.

6
Application

The implemented application was developed on Google Colab. It works on the command line interpreter. We generally use Python based Keras library for import, train, test our models. The Python-based Keras library is commonly utilized for the purposes of importing, training, and testing models. During the evaluation process carried out subsequent to these procedures, support was acquired from the Sci-Kit Learn library.

In the study, various model architectures were experimented with and subsequently compared. To illustrate, the architecture of the EfficientNet-B3 model employed in the study is presented in the Table 6.1 below.

Table 6.1 EfficientNet-B3 Model Architecture

Layer Number	Layer Name	Resolution	Number of Layers
1	Conv 3x3	300x300	1
2	MBConv1 3x3	150x150	2
3	MBConv6 3x3	150x150	3
4	MBConv6 5x5	75x75	3
5	MBConv6 3x3	38x38	5
6	MBConv6 5x5	19x19	5
7	MBConv6 5x5	10x10	6
8	MBConv6 3x3	10x10	2
9	Conv 1x1	10x10	1
10	Pooling	10x10	1
11	Dense Layer	10x10	1

In the Table 6.2 within the EfficientNet network, training hyperparameters are described as an example. The results of the model created with these hyperparameters are explained in the Experimental Results section.

Table 6.2 Examples of Hyperparameters Used in EfficientNet Models

Parameters	EfficientNet-B3
Batch Size	4
Epoch	20
Image Size	300x300
Optimizer	Adam
Learning Rate	0.01

The metric result outputs after the model evaluation are shown in the Figure 6.1 and Figure 6.2.

```
1059/1059 [==============================] - 29s 28ms/step
              precision    recall  f1-score   support

    Covid-19    0.98657   0.91425   0.94903       723
      Normal    0.93744   0.95584   0.94655      2038
     Opacity    0.91514   0.91438   0.91476      1203
   Pneumonia    0.94346   0.99257   0.96739       269

    accuracy                        0.93929      4233
   macro avg    0.94565   0.94426   0.94443      4233
weighted avg    0.93988   0.93929   0.93926      4233


 Cohen's Kappa Score:  0.9068338348699154
```

Figure 6.1 Classification Report Output of the Evaluated Model

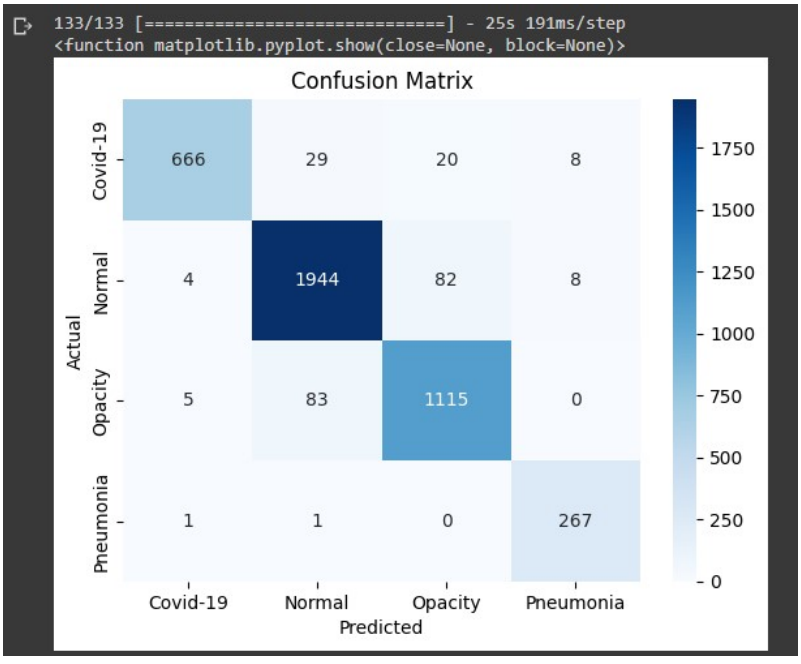

Figure 6.2 Confusion Matrix Output of the Evaluated Model

The ground truth and prediction outputs of four randomly selected images from the test data are shown in Figure 6.3 and Figure 6.4, respectively, along with the corresponding image labels indicated below each image.

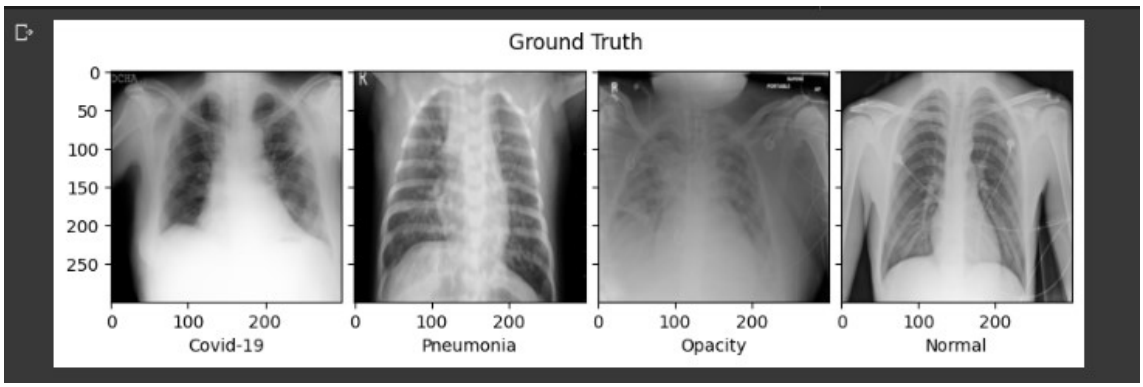

Figure 6.3 Ground Truth Images Shown to the Evaluated Model

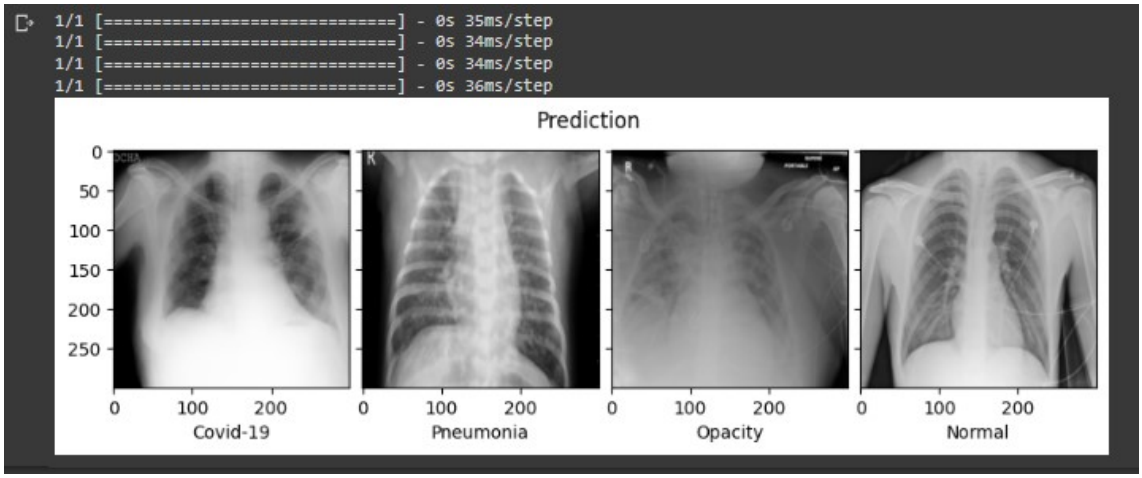

Figure 6.4 Predictions of the Evaluated Model

7
Experimental Findings

This section presents the results of all trials of the algorithms used in the feature extraction and the classification parts of the project. The Deep Learning section selects the model to be used for feature extraction and compares different deep learning models. In the Hybrid Learning section, feature extraction is performed using the best performing models selected in the Deep Learning section, and the machine learning algorithms that best classify the features from these models are compared.

7.1 Deep Learning

7.1.1 ResNet-50

Table 7.1 ResNet-50 Trials with Different Hyperparameters

	Input Shape	Learning Rate	Epoch	Batch Size	# of Custom Dense Layer
ResNet-50 First Trial	224	5e-2	20	4	1
ResNet-50 Second Trial	224	5e-2	20	4	None
ResNet-50 Third Trial	224	2e-2	40	4	None

Table 7.2 ResNet-50 Trials Evaluations

	Accuracy	Precision	Recall	F-Measure	Cohen's Kappa
ResNet-50 First Trial	0.48	0.12	0.25	0.16	0.22
ResNet-50 Second Trial	0.88	0.90	0.86	0.88	0.85
ResNet-50 Third Trial	0.90	0.91	0.91	0.90	0.85

The most successful variation of ResNet-50 (ResNet-50 Third Trial) is chosen to show the confusion matrix in Fig 7.1 and classification results in Fig 7.2.

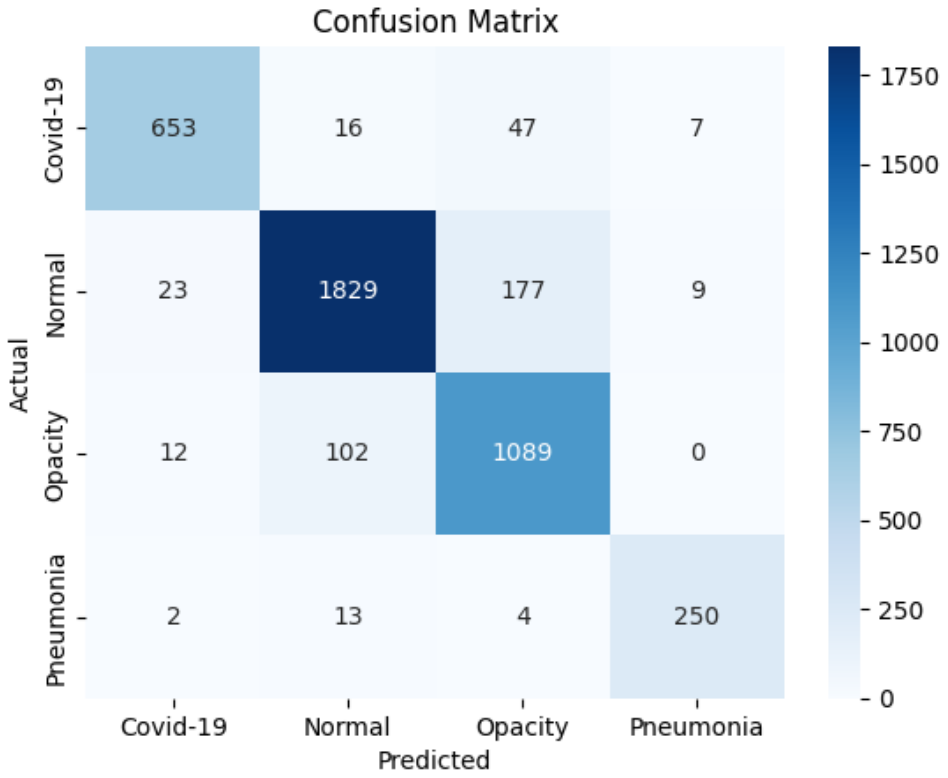

Figure 7.1 Confusion Matrix of ResNet-50

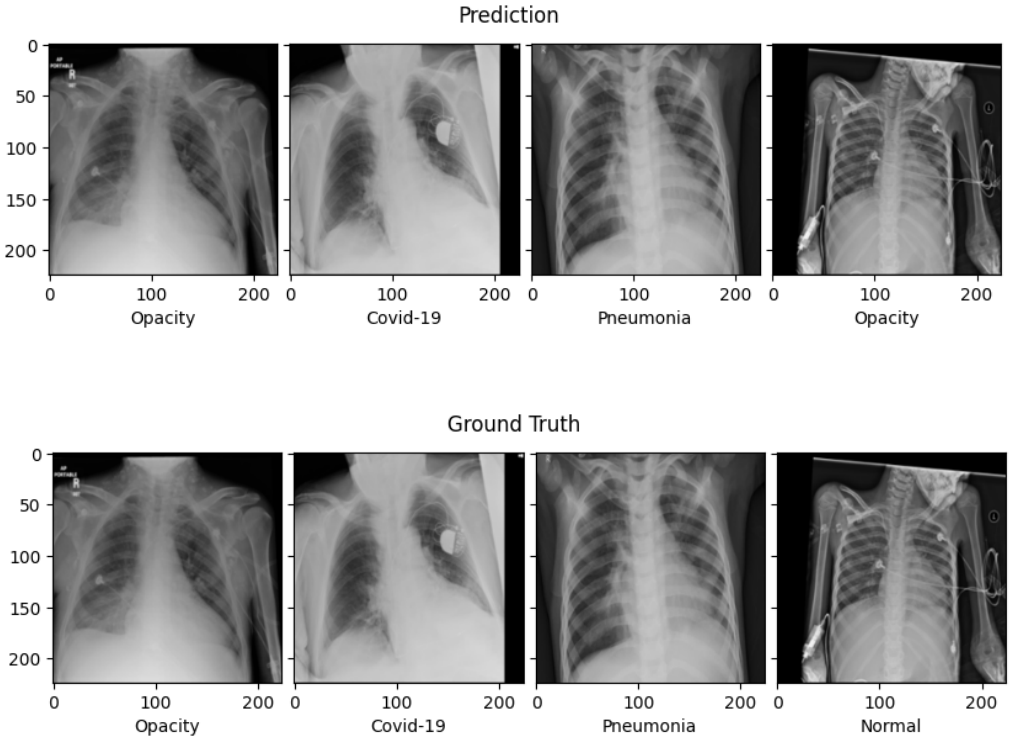

Figure 7.2 Prediction and Ground Truth Results of ResNet-50

7.1.2 EfficientNet

Table 7.3 EfficientNet Trials with Different Hyperparameters

	Input Shape	Learning Rate	Epoch	Batch Size
EfficientNet_B1 First Trial	240	1e-2	20	4
EfficientNet_B1 Second Trial	300	2e-2	20	4
EfficientNet_B3 First Trial	300	2e-2	20	4
EfficientNet_B3 Second Trial	300	1e-2	20	4

Table 7.4 EfficientNet Trials Evaluations

	Accuracy	Precision	Recall	F-Measure	Cohen's Kappa
EfficientNet_B1 First Trial	0.90	0.90	0.92	0.91	0.87
EfficientNet_B1 Second Trial	0.89	0.88	0.91	0.90	0.86
EfficientNet_B3 First Trial	0.88	0.91	0.86	0.88	0.86
EfficientNet_B3 Second Trial	0.91	0.92	0.92	0.92	0.88

The most successful variation of EfficientNet_B1 (First Trial) is chosen to show the confusion matrix in Fig 7.3 and classification results in Fig 7.4.

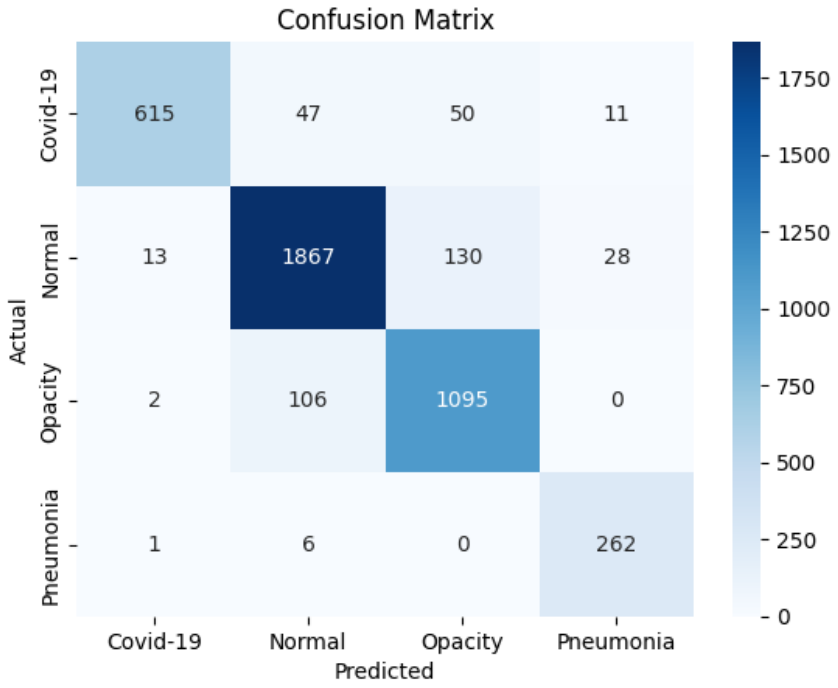

Figure 7.3 Confusion Matrix of EfficientNet_B1

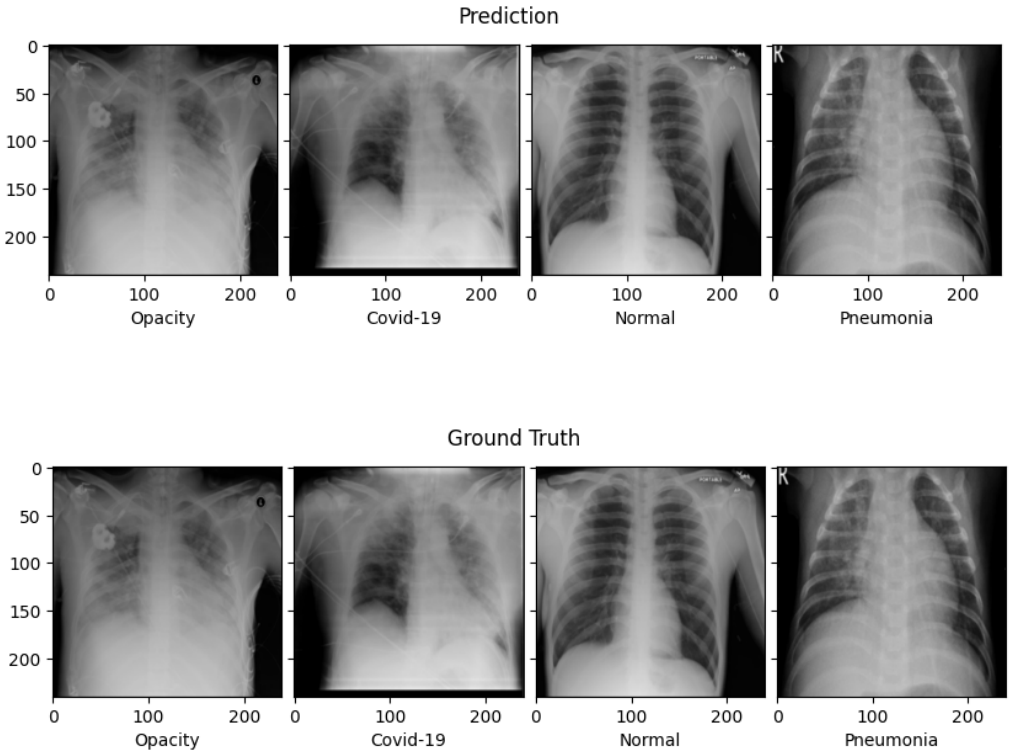

Figure 7.4 Prediction and Ground Truth Results of EfficientNet_B1

The most successful variation of EfficientNet_B3 (Second Trial) is chosen to show the confusion matrix in Fig 7.5 and classification results in Fig 7.6.

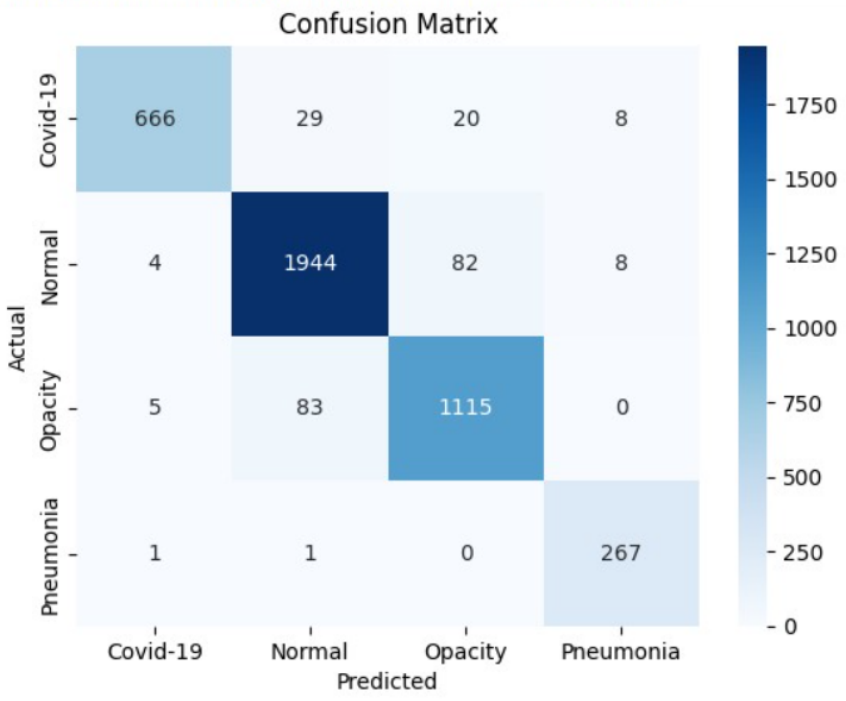

Figure 7.5 Confusion Matrix of EfficientNet_B3

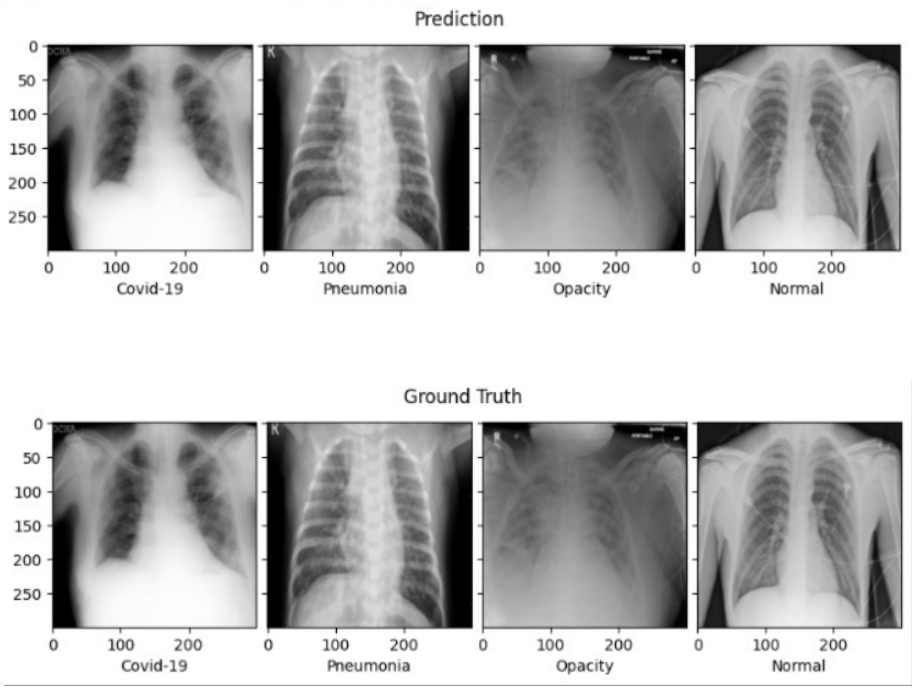

Figure 7.6 Prediction and Ground Truth Results of EfficientNet_B3

7.1.3 Inception_V3

Table 7.5 Inception_V3 Trials with Different Hyperparameters

	Input Shape	Learning Rate	Epoch	Batch Size
Inception_V3 First Trial	224	1e-2	20	4
Inception_V3 Second Trial	224	1e-2	30	4
Inception_V3 Third Trial	224	2e-2	25	4
Inception_V3 Fourth Trial	224	5e-3	20	4

Table 7.6 Inception_V3 Trials Evaluations

	Accuracy	Precision	Recall	F-Measure	Cohen's Kappa
Inception_V3 First Trial	0.89	0.91	0.89	0.90	0.84
Inception_V3 Second Trial	0.88	0.88	0.88	0.87	0.82
Inception_V3 Third Trial	0.75	0.82	0.80	0.79	0.64
Inception_V3 Fourth Trial	0.90	0.92	0.91	0.91	0.86

The most successful variation of Inception_V3 (Fourth Trial) is chosen to show the confusion matrix in Fig 7.7 and classification results in Fig 7.8.

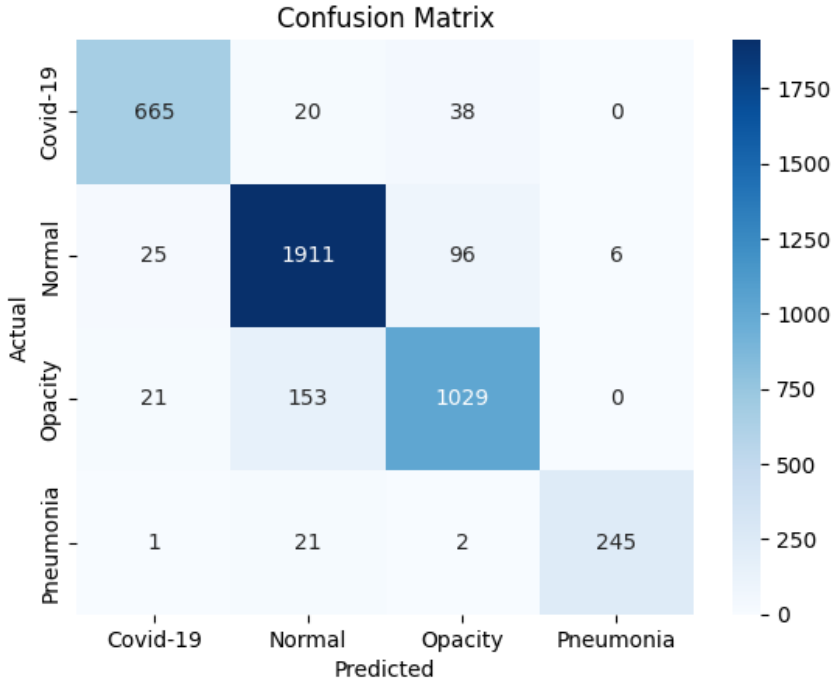

Figure 7.7 Confusion Matrix of Inception_V3

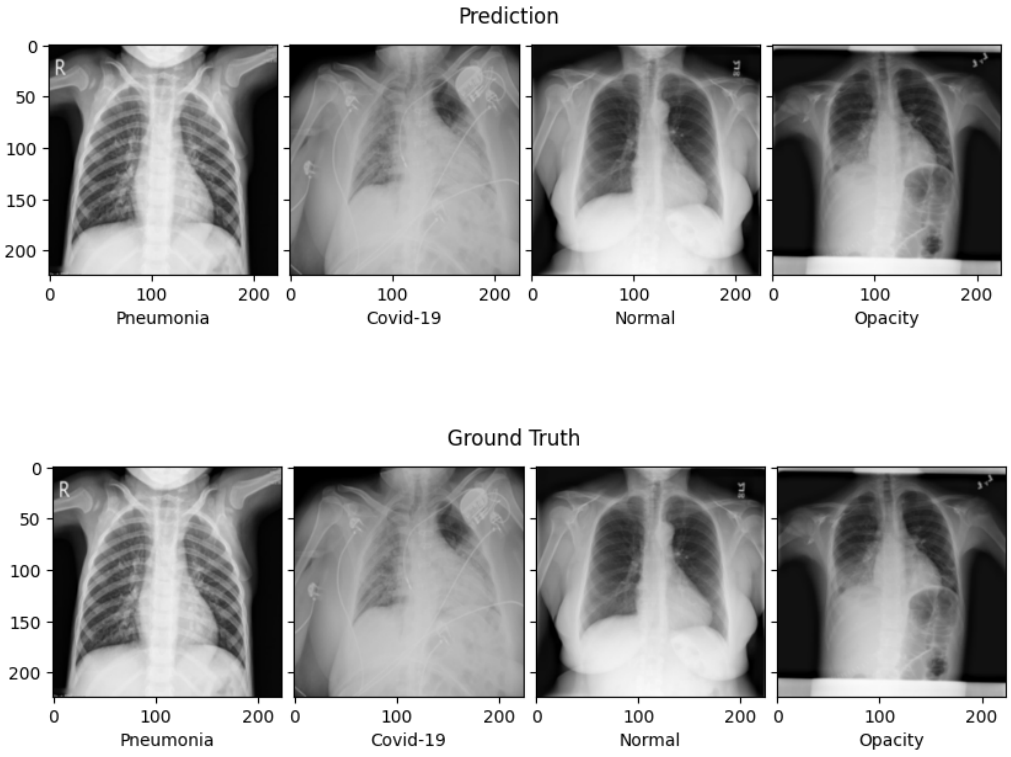

Figure 7.8 Prediction and Ground Truth Results of Inception_V3

7.2 Hybrid Learning

In this section, the performances of machine learning models in comparison with deep learning models were evaluated. Hyperparameter optimization is executed for every machine learning model and these parameters are used for every execution.

Table 7.7 Hyperparameters Used in SVM

Parameters	C	Gamma	Kernel
SVM	10	Scale	RBF

Table 7.8 Hyperparameters Used in XGBoost

Parameters	Objective	Max_depth	Learning Rate	Subsample	N_estimators
XGBoost	Multi:softmax	7	0.1	0.8	1000

Table 7.9 Hyperparameters Used in Random Forest

Parameters	N_estimators	Min Samples Split	Min Samples Leaf	Criterion
Random Forest	100	10	1	Gini

Table 7.10 Hyperparameters Used in KNN

Parameters	Algorithm	N_neighbors	Weights
KNN	Auto	7	Distance

7.2.1 ResNet-50

Table 7.11 ResNet-50 Hybrid Learning Evaluations

	Accuracy	Precision	Recall	F-Measure	Cohen's Kappa
SVM	0.92	0.94	0.93	0.93	0.88
XGBoost	0.91	0.92	0.90	0.91	0.85
Random Forest	0.84	0.88	0.80	0.83	0.75
KNN	0.86	0.90	0.84	0.87	0.78

According to the results shown in the Table 7.11, the machine learning algorithm that best classified the features extracted using ResNet-50 was determined to be SVM.

7.2.2 EfficientNet_B3

Table 7.12 EfficientNet_B3 Hybrid Learning Evaluations

	Accuracy	**Precision**	**Recall**	**F-Measure**	**Cohen's Kappa**
SVM	0.92	0.94	0.93	0.93	0.87
XGBoost	0.90	0.92	0.90	0.91	0.84
Random Forest	0.83	0.87	0.80	0.83	0.72
KNN	0.84	0.89	0.83	0.86	0.76

According to the results shown in the Table 7.12, the machine learning algorithm that best classified the features extracted using EfficientNet_B3 was determined to be SVM.

7.2.3 Inception_V3

Table 7.13 Inception_V3 Hybrid Learning Evaluations

	Accuracy	**Precision**	**Recall**	**F-Measure**	**Cohen's Kappa**
SVM	0.83	0.85	0.80	0.82	0.73
XGBoost	0.85	0.86	0.82	0.84	0.77
Random Forest	0.79	0.82	0.72	0.75	0.67
KNN	0.80	0.82	0.75	0.78	0.68

According to the results shown in the Table 7.13, the machine learning algorithm that best classified the features extracted using Inception_V3 was determined to be XGBoost.

8
Performance Evaluation

This section evaluates the performance analysis of the deep learning and machine learning algorithms used. To achieve the best results, hyperparameter tuning is performed on each machine learning models as can be seen in Table 7.7, Table 7.8, Table 7.9 and Table 7.10. The hyperparameters for deep learning models are found out by trial and error as can be seen in Table 7.1, Table 7.3 and in Table 7.5.

Throughout the evaluation, for the algorithms used in feature extraction and classification, the metrics used are accuracy, precision, recall, f1-score and Cohen's kappa, as can be seen in section 7. Accuracy, precision and recall metrics alone are not sufficient for evaluation, so the evaluation will be mostly done considering f1-score and Cohen's kappa. Cohen's kappa measures the consistency and harmoniousness between the classifiers while f1-score measures classification accuracy.

The best resulting deep learning model is EfficientNet_B3 with an f1 score of 0.92 and a Cohen's kappa of 0.88 as can be seen in Table 7.4 and it's confusion matrix in Fig 7.5 gives the highest values in diagonal as expected. However, because the project is implemented using a hybrid model, it can be misleading to look at the result of the deep learning model or machine learning model alone. Therefore, the table 8.1 evaluates the models as a pair, rather than just the machine learning models.

Table 8.1 Best-Resulting Hybrid Models

	Accuracy	**Precision**	**Recall**	**F-Measure**	**Cohen's Kappa**
ResNet-50 & SVM	**0.92**	**0.94**	**0.93**	**0.93**	**0.88**
EfficientNet_B3 & SVM	0.92	0.94	0.93	0.93	0.87
Inception_V3 & XGBoost	0.85	0.86	0.82	0.84	0.77

The Table 8.1 is obtained by collecting the best resulting hybrid model pairs from tables 7.11, 7.12 and 7.13. It can be seen that **ResNet-50 with SVM** is slightly better than EfficientNet_B3 with SVM considering Cohen's kappa measure. And the other important point is their feature extraction times. ResNet-50 wins this race again with approximately 0,22 ms compared to SVM with 0,35 ms. In contrast, when we observe each deep learning model individually, we have seen that EfficientNet_B3 resulted the best among them, but not the best in hybrid model. This may be due to the compatibility of the extracted feature maps with machine learning models.

From Table 8.1, it is also seen that the results of the feature maps obtained from InceptionV3 used with machine learning models lagged behind other hybrid models, although it performed close to the other deep learning algorithms when analyzed individually on the dataset.

Another consequence is that using ResNet-50 with SVM increased each evaluation metric by approximately 2.5%. This shows that machine learning models are better at classifying samples than deep learning models. Therefore, the hybrid approach performs better results for this project.

9
Conclusion

The aim of this study is to provide an automated method for detecting and classifying lung diseases from X-ray images and to implement a system that can be used in a real-world diagnostic process. A data set of four different lung diseases have been chosen for this task from Kaggle repository and examined in section 5.1. Throughout the code implementation of the project, Python and Google Colab are chosen as the programming language and the environment due to the reasons explained in section 3.1.1.

The literature review in section 2.1 shows that CNN-based algorithms are the most accurate for understanding images and their features, but classification gives the best results when performed using machine learning models. We therefore decided to carry out this project using hybrid models, meaning using of deep learning and machine learning algorithms for feature extraction and classification parts respectively. Deep learning models which are ResNet, EfficientNet and Inception are used in combination with machine learning models which are Random Forest, XGBoost, SVM and KNN. These models and the logic behind them are explained under section 5.2.

The evaluation metrics used in this project are presented in section 5.3 and the experimental findings in section 7 are carried out accordingly. The performance evaluation in section 8 showed that using ResNet-50 with Support Vector Machine gave the best results, so they were chosen as the executable models for our project.

With the implementation of this project, doctors can make use of this artificial intelligence-based decision support system as a time-saving and reliable source of lung disease detection.

References

[1] P. Central. "Place of death for individuals with chronic lung disease." (2020), [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8173769/ (visited on 04/18/2023).

[2] Q. Ke *et al.*, "A neuro-heuristic approach for recognition of lung diseases from x-ray images," *Expert Systems with Applications*, vol. 126, pp. 218–232, 2019, ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2019.01.060. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417419300776.

[3] MathWorks. "What is a convolutional neural network?" (2023), [Online]. Available: https://www.mathworks.com/discovery/convolutional-neural-network-matlab.html (visited on 04/18/2023).

[4] S. Khobragade, A. Tiwari, C. Patil, and V. Narke, "Automatic detection of major lung diseases using chest radiographs and classification by feed-forward artificial neural network," in *2016 IEEE 1st International Conference on Power Electronics, Intelligent Control and Energy Systems (ICPEICES)*, 2016, pp. 1–5. DOI: 10.1109/ICPEICES.2016.7853683.

[5] M. Physics. "Image feature analysis and computer-aided diagnosis in digital radiography: Detection and characterization of interstitial lung disease in digital chest radiographs." (2023), [Online]. Available: https://aapm.onlinelibrary.wiley.com/doi/abs/10.1118/1.596224 (visited on 04/26/2023).

[6] S. Bharati, P. Podder, and M. R. H. Mondal, "Hybrid deep learning for detecting lung diseases from x-ray images," *Informatics in Medicine Unlocked*, vol. 20, p. 100 391, 2020, ISSN: 2352-9148. DOI: https://doi.org/10.1016/j.imu.2020.100391. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352914820300290.

[7] S. Reports. "Diagnosis of common pulmonary diseases in children by x-ray images and deep learning." (2020), [Online]. Available: https://www.nature.com/articles/s41598-020-73831-5#Abs1 (visited on 04/26/2023).

[8] S. Link. "Detection and classification of lung diseases for pneumonia and covid-19 using machine and deep learning techniques." (2023), [Online]. Available: https://link.springer.com/article/10.1007/s12652-021-03464-7 (visited on 04/26/2023).

[9] "Weights and biases." (2023), [Online]. Available: https://subscription.packtpub.com/book/data/9781788397872/1/ch01lvl1sec12/weights-and-biases (visited on 06/22/2023).

[10] S. Kim. “A beginner’s guide to convolutional neural networks (cnns).” (2019), [Online]. Available: https://towardsdatascience.com/a-beginners-guide-to-convolutional-neural-networks-cnns-14649dbddce8 (visited on 05/10/2023).

[11] M. Kumar. “How padding helps in cnn ?” (2020), [Online]. Available: https://www.numpyninja.com/post/how-padding-helps-in-cnn (visited on 06/22/2023).

[12] sphinx-quickstart. “Pooling (cnn).” (2021), [Online]. Available: https://epynn.net/Pooling.html (visited on 06/22/2023).

[13] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[14] S. Mukherjee. “The annotated resnet-50.” (2022), [Online]. Available: https://towardsdatascience.com/the-annotated-resnet-50-a6c536034758 (visited on 06/22/2023).

[15] S. Sahoo. “Residual blocks — building blocks of resnet.” (2018), [Online]. Available: https://towardsdatascience.com/residual-blocks-building-blocks-of-resnet-fd90ca15d6ec (visited on 06/22/2023).

[16] A. Vidhya. “All you need to know about skip connections.” (2021), [Online]. Available: https://www.analyticsvidhya.com/blog/2021/08/all-you-need-to-know-about-skip-connections/ (visited on 04/30/2023).

[17] M. Tan and Q. Le, “Efficientnet: Rethinking model scaling for convolutional neural networks,” May 2019. DOI: https://doi.org/10.48550/arXiv.1905.11946.

[18] M. Tan and Q. V. Le, *Efficientnet: Rethinking model scaling for convolutional neural networks*, 2019. arXiv: 1905.11946 [cs.LG].

[19] V. Agarwal. “Complete architectural details of all efficientnet models.” (2020), [Online]. Available: https://towardsdatascience.com/complete-architectural-details-of-all-efficientnet-models-5fd5b736142 (visited on 06/22/2023).

[20] B. Raj. “A simple guide to the versions of the inception network.” (2018), [Online]. Available: https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202 (visited on 05/10/2023).

[21] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, *Rethinking the inception architecture for computer vision*, 2015. arXiv: 1512.00567 [cs.CV].

[22] V. Kurama. “A review of popular deep learning architectures: Resnet, inceptionv3, and squeezenet.” (2020), [Online]. Available: https://blog.paperspace.com/popular-deep-learning-architectures-resnet-inceptionv3-squeezenet/ (visited on 05/10/2023).

[23] M. Banoula. “Classification in machine learning: What it is & classification models.” (2023), [Online]. Available: https://www.simplilearn.com/tutorials/machine-learning-tutorial/classification-in-machine-learning (visited on 05/30/2023).

[24] A. Shafi. "Random forest classification with scikit-learn." (2023), [Online]. Available: https://www.datacamp.com/tutorial/random-forests-classifier-python (visited on 06/22/2023).

[25] S. E. R. "Understand random forest algorithms with examples." (2021), [Online]. Available: https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/ (visited on 05/30/2023).

[26] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," ser. KDD '16, San Francisco, California, USA: Association for Computing Machinery, 2016, pp. 785–794, ISBN: 9781450342322. DOI: 10.1145/2939672.2939785. [Online]. Available: https://doi.org/10.1145/2939672.2939785.

[27] A. Şenozan. "Ensemble: Boosting, bagging, and stacking machine learning." (2023), [Online]. Available: https://medium.com/@senozanAleyna/ensemble-boosting-bagging-and-stacking-machine-learning-6a09c31df778 (visited on 06/22/2023).

[28] V. Morde. "Xgboost algorithm: Long may she reign!" (2019), [Online]. Available: https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d (visited on 06/04/2023).

[29] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, pp. 273–297, 1995. DOI: 10.1007/BF00994018.

[30] alokesh985. "Introduction to support vector machines (svm)." (2022), [Online]. Available: https://www.geeksforgeeks.org/introduction-to-support-vector-machines-svm/ (visited on 06/22/2023).

[31] A. Saini. "Support vector machine(svm): A complete guide for beginners." (2021), [Online]. Available: https://www.analyticsvidhya.com/blog/2021/10/support-vector-machinessvm-a-complete-guide-for-beginners/ (visited on 06/22/2023).

[32] H. Parvez. "Support vector machine(svm) made easy with python | machine learning." (2021), [Online]. Available: https://www.aionlinecourse.com/tutorial/machine-learning/support-vector-machine (visited on 06/22/2023).

[33] P. Hart, "The condensed nearest neighbor rule (corresp.)," *IEEE Transactions on Information Theory*, vol. 14, no. 3, pp. 515–516, 1968. DOI: 10.1109/TIT.1968.1054155.

[34] A. Ataşoğlu. "K-nn (k-nearest neighbors) algoritması (python)." (2020), [Online]. Available: https://ahmetatasoglu98.medium.com/k-nn-k-nearest-neighbors-algoritmas%C4%B1-python-678c86a90558 (visited on 06/22/2023).

[35] M. McHugh, "Interrater reliability: The kappa statistic," *Biochemia medica : časopis Hrvatskoga društva medicinskih biokemičara / HDMB*, vol. 22, pp. 276–82, Oct. 2012. DOI: 10.11613/BM.2012.031.

Curriculum Vitae

FIRST MEMBER

Name-Surname: Arda KAŞIKÇI
Birthdate and Place of Birth: 18.06.2000, İstanbul
E-mail: ardakskc@gmail.com
Phone: 0534 613 51 83
Practical Training: Ziraat Teknoloji A.Ş.
TÜBİTAK BİLGEM
Migros ONE

SECOND MEMBER

Name-Surname: Elif MERTOĞLU
Birthdate and Place of Birth: 11.06.2001, İstanbul
E-mail: elif.mertoglu@yildiz.edu.tr
Phone: 0545 774 23 94
Practical Training: Yapı Kredi Teknoloji
Türk Havacılık ve Uzay Sanayii

Project System Informations

System and Software: Windows İşletim Sistemi, Python
Required RAM: 15GB
Required Disk: 25GB