PNEUMONIA DETECTION USING CHEST X-RAY WITH DEEP LEARNING

A PROJECT REPORT

Submitted by

DEEPIKA T R **(913116205011)**

KEERTHANA K **(913116205022)**

RAMYA T S **(913116205041)**

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

VELAMMAL COLLEGE OF ENGINEERING AND TECHNOLOGY, MADURAI

ANNA UNIVERSITY: CHENNAI 600 025

APRIL 2020

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report "**PNEUMONIA DETECTION USING CHEST X-RAY WITH DEEP LEARNING"** is the bonafide work of "**DEEPIKA T R (913116205011), KEERTHANA K (913116205022) and RAMYA T S (913116205041)"** who carried out the project work under my supervision.

Dr. R. Perumalraja M.E., Ph.D.,

HEAD OF THE DEPARTMENT

Department of Information Technology,

Velammal College of Engg & Tech,

Madurai – 625 009.

Dr.S. Kamalesh M.E.,Ph.D.,

SUPERVISOR

Assistant Professor

Department of Information Technology,

Velammal College of Engg & Tech,

Madurai – 625 009.

Submitted for the Project Viva-Voce Examination held on _____

at Velammal College of Engineering and Technology, Madurai.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We thank the Almighty for giving us moral strength to work on the project for the past few months.

We would like to express our sincere thanks to **Shri. M.V. Muthuramalingam,** Founder of institution who had been a guiding light for us in all activities.

We express our sincere thanks to our respected Principal **Dr. N. Suresh Kumar,** for providing more facilities, to do this project work.

Our heartfelt gratitude to **Dr. R. Perumalraja**, Head of the Department, Department of Information Technology, for the valuable guidance, inspiration and encouraging appreciations, which helped us a lot in completing this project in time.

We would like to express our token of thanks to our guide, **Dr.S. Kamalesh,** Assistant Professor, Department of Information Technology for his constant source of encouragement and for the valuable guidance and support for the smooth completion of our project work.

We wish to extend our sincere thanks to **all faculty members** of Information Technology who have contributed directly and indirectly for doing this project work. We also thank to our parents and friends who provided moral and physical support.

ABSTRACT

Pneumonia is a common infectious disease in the world. Its main diagnostic method is chest X-Ray (CXR) examination. However, the high visual similarity between a large number of pathologies in CXR makes the interpretation and differentiation of pneumonia a challenge. In this paper, we propose an improved convolutional neural network (CNN) model for pneumonia detection. In order to guide the CNN to focus on disease-specific attended region, the pneumonia area of image is erased and marked as a non-pneumonia sample. In addition, transfer learning is used to segment the interest region of lungs to suppress background interference. The experimental results show that the proposed method is superior to the state-of-the-art object detection model in terms of accuracy and false positive rate. Chest X-ray (CXR) is the most suitable imaging modality to diagnose pneumonia. Usually, pneumonia manifests as an area or areas of increased opacity on CXR. However, the imaging reviews on CXR are complicated because a number of factors, such as positioning of the patient and depth of inspiration, can alter the appearance of CXR. In addition, some other conditions of the lungs can also affect the analysis and diagnosis of pneumonia, such as infiltration, mass, volume loss, or post-radiation and surgical changes. These conditions have greatly increased the difficulty for clinicians to read and analyze CXR images. Recently, deep learning has played an increasingly important role in the automatic analysis and clinical diagnosis of medical images. In particular, some methods based on convolutional neural networks (CNN) have been successfully applied to classify diseases, locate abnormal regions or segment lesions in CXR images.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

Machine learning is a category of algorithm that allows software applications to become more accurate in predicting outcomes without being explicitly programmed. The basic premise of ML is to build algorithms that can receive input data and use statistical analysis to predict an output while updating outputs as new data becomes available.

There are basically two types of ML algorithms. They are Supervised ML algorithms and Unsupervised ML algorithms. Supervised learning is a learning in which we teach or train the machine using data which is well labeled that means some data is already tagged with correct answer. After that, machine is provided with new set of examples so that supervised learning algorithm analyses the training data and produces a correct outcome from labeled data. Regression and Classification are supervised ML algorithms. A regression problem is when the output variable is a real value, such as weight or price. A classification problem is when the output variable is a category, such as 0 and 1 or ‘Pneumonia and ‘Normal’. Unsupervised learning is the training of machine using information that is neither classified nor labeled and allowing the algorithm to act on that information without guidance. Here the task of machine is to group unsorted information according to similarities, patterns and differences without any prior training of data. Clustering and Association are unsupervised ML algorithms.

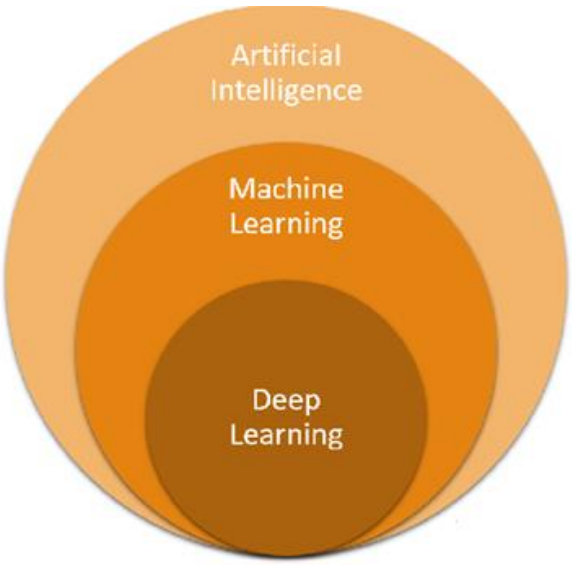


Figure 1.1 Logical view of Artificial Intelligence

Deep learning is a branch of machine learning which is completely based on artificial neural networks, as neural network is going to mimic the human brain so deep learning is also a kind of mimic of human brain. In deep learning we don't need to explicitly program everything. Concept of deep learning is not new. It has been around for a couple of years now. It's on hype nowadays because earlier we did not have that much processing power and lot of data. As in last 20 years the processing power increases exponentially, deep learning and machine learning came in picture.

"Deep learning is a particular kind of machine learning that achieves great power and flexibility by learning to represent the world as nested hierarchy of concepts, with each concept defined in relation to simpler concepts, and more abstract representations computed in terms of less abstract ones."

Neural Networks:

A neural network is a series of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. In this sense, neural networks refer to systems of neurons, either organic or artificial in nature. Neural networks can

adapt to changing input; so the network generates the best possible result without needing to redesign the output criteria.

A neural network works similarly to the human brain's neural network. . A "neuron" in a neural network is a mathematical function that collects and classifies information according to a specific architecture. The network bears a strong resemblance to statistical methods such as curve fitting and regression analysis.

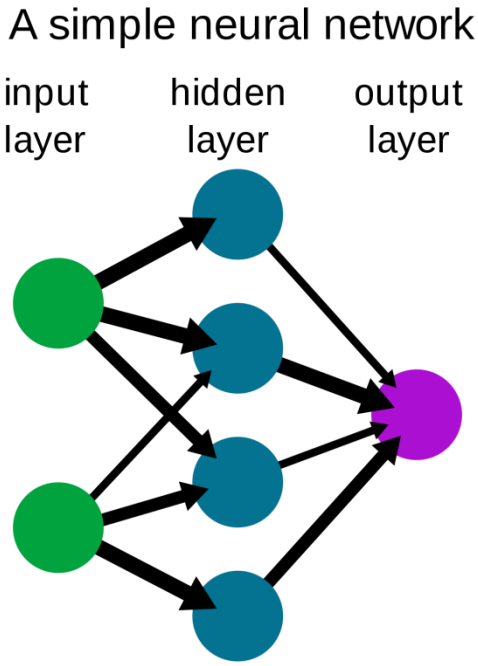


Figure 1.2 Simple Neural Network

A neural network contains layers of interconnected nodes. Each node is a perceptron and is similar to a multiple linear regression. The perceptron feeds the signal produced by a multiple linear regression into an activation function that may be nonlinear.

In a multi-layered perceptron (MLP), perceptrons are arranged in interconnected layers. The input layer collects input patterns. The output layer has classifications or output signals to which input patterns may map.

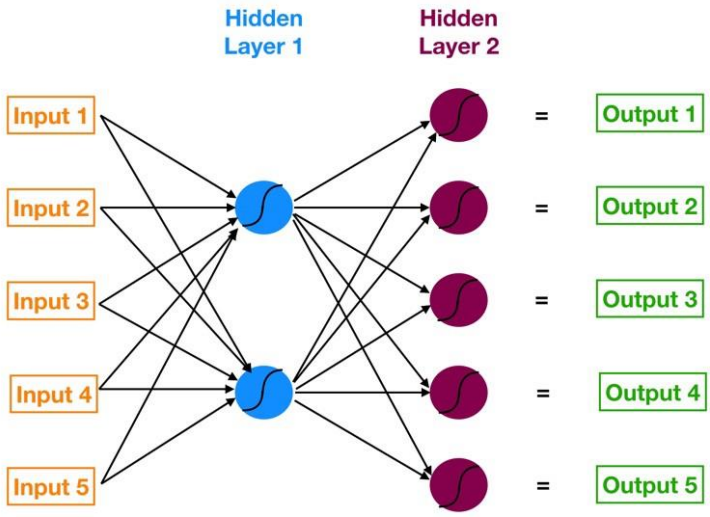


Figure 1.3 Multi Layer Perceptron Network

Convolutional Neural Network:

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (Learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

CNNs are regularized versions of multilayer perceptrons. Convolutional networks were inspired by biological processes[7][8][9][10] in that the connectivity pattern between neurons resembles the organization of the animal visual cortex. Convolution is a specialized kind of linear operation. Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers.

1.2 MOTIVATION

Patients will be able to share their medical report securely and receive quick results. Test the report and different results using available trained Machine Learning and Deep Learning backend endpoints and analyze the results. ML/AI algorithm will train in real time using Convolutional Neural Network from the prediction given by various reports. The patients will also be able to track their Medical Records and view it any time without any hassle.

1.3 OBJECTIVES

The main objective of this project is to deliver an end product that receives input from user, use them to predict and classify and give them the results predicted by the neural network model. We create an application that accepts Chest X-Ray image as input from the user using upload button. The Pneumonia is predicted using the analysis button on the screen. The model is trained with the training dataset, which is loaded every time user uses it.

1.4 SCOPE OF THE PROJECT

- Creating a model that best suits the Pneumonia dataset.
- Obtaining correct weightage to independent variables.
- Maximizing accuracy and minimize prediction error.
- Get futuristic predictions based on the given input features.

1.5 LIMITATIONS

- Requires installation of lot of R packages.
- If the values are off the chart from the training data set, the results would like to have high error rates.

1.6 IMPORTANCE OF THE PROJECT

Pneumonia is a life-threatening infectious disease affecting one or both lungs in humans commonly caused by bacteria called Streptococcus Pneumonia. Pneumonia is among the top diseases which cause most of the deaths all over the world. Virus, bacteria and fungi can all cause pneumonia. This study proposes a convolutional neural network model trained from scratch to classify and detect the presence of pneumonia from a collection of chest X-ray image samples. We constructed a convolutional neural network model from scratch to extract features from a given chest X-ray image and classify it to determine if a person is infected with Pneumonia.

The risk of pneumonia is immense for many, especially in developing nations where billions face energy poverty and rely on polluting forms of energy. WHO estimates that over 4 million premature deaths occur annually from household air pollution-related diseases including Pneumonia[1]. Over 150 million people get infected with Pneumonia on an annual basis especially children under 5 years old[2]. In such regions, the problem can be further aggravated due to the dearth of medical resources and personnel. For example, in Africa's 57 nations, a gap of 2.3 million doctors and nurses exists[3,4]. For these populations, accurate and fast diagnosis means everything. It can guarantee timely access to treatment and save much needed time and money for those already experiencing poverty. Chest X-ray (CXR) is the most suitable imaging modality to diagnose pneumonia. Pneumonia manifests as an area or areas of increased opacity[5] on CXR.

Deep learning has played an increasingly important role in the automatic analysis and clinical diagnosis of medical images. In particular, some methods based on convolutional neural networks (CNN) have been successfully applied

to classify diseases, locate abnormal regions or segment lesions in CXR images[6].

One of the most conventional medical techniques used to diagnose the disease is chest x-ray. As the concentrated beam of electrons, called x-ray photons, go through the body tissues, an image is produced on the metal surface (photographic film). When interpreting chest X-rays for Pneumonia, the radiologist will look for white spots in the lungs called infiltrates that identify an infection. the limited color scheme of x-ray images consisting of shades of black and white, cause drawbacks when it comes to determining whether there is an infected area in the lungs or not. However, such cloudy patterns would also be observed in TB Pneumonia and severe cases of bronchitis too. For conclusive diagnosis, further investigations such as complete blood count (CBC), Sputum test, and Chest computed tomography (CT) scan etc. may be needed. Therefore, we are only attempting to detect possibility of pneumonia from Chest X-rays, by looking for cloudy region in the same. Conclusive detection will depend on pathological tests.

CHAPTER 2
LITERATURE SURVEY

2.1 An Efficient Deep Learning Approach to Pneumonia Classification in Healthcare

Author: Okeke Stephen, Mangal Sain

This study proposes a convolutional neural network model trained from scratch to classify and detect the presence of pneumonia from a collection of chest X-ray image samples. Unlike other methods that rely solely on transfer learning approaches or traditional handcrafted techniques to achieve a remarkable classification performance, we constructed a convolutional neural network model from scratch to extract features from a given chest X-ray image and classify it to determine if a person is infected with pneumonia. This model could help mitigate the reliability and interpretability challenges often faced when dealing with medical imagery. Unlike other deep learning classification tasks with sufficient image repository, it is difficult to obtain a large amount of Pneumonia dataset for this classification task; therefore, we deployed several data augmentation algorithms to improve the validation and classification accuracy of the CNN model and achieved remarkable validation accuracy.

We developed a model to detect and classify pneumonia from chest X-ray images taken from frontal views at high validation accuracy. The algorithm begins by transforming chest X-ray images into sizes smaller than the original. The next step involves the identification and classification of images by the convolutional neural network framework, which extracts features from the images and classifies them. Due to the effectiveness of the trained CNN model for identifying pneumonia from chest X-ray images, the validation accuracy of our model was significantly higher when compared with other approaches. To affirm the performance of the model, we repeated the training

process of the model several times, each time obtaining the same results. To validate the performance of the trained model on different chest X-ray image sizes, we varied the sizes of the training and validation dataset and still obtained relatively similar results. This will go a long way in improving the health of at-risk children in energy-poor environments. The study was limited by depth of data. With increased access to data and training of the model with radiological data from patients and non patients in different parts of the world, significant improvements can be made.

2.2 Detection of Pneumonia clouds in Chest X-ray using Image processing Approach

Author: Abhishek Sharma, Daniel Raju, Sutapa Ranjan

Chest X-ray (CXR), is an important tool for diagnosing Pneumonia and many clinical decisions rely heavily on its radiological findings. Also it is relatively cheap compared to other imaging diagnostics and can be afforded by masses. Some work has been done on automated pneumonia detection through natural language processing and artificial neural network. However, such tools require elaborate hardware and software setup. In order to develop a software that can be easily and freely ported to mobile devices, with no financial or license related obligations, so we have undertaken a purely an image processing approach as in [4], using free open source tools such as Python and its image processing libraries.

When interpreting chest X-rays for Pneumonia, the radiologist will look for white spots in the lungs called infiltrates that identify an infection. However, such cloudy patterns would also be observed in TB Pneumonia and severe cases of bronchitis too. For conclusive diagnosis, further investigations such as complete blood count (CBC), Sputum test, and Chest computed tomography (CT) scan etc. may be needed. Therefore, we are only attempting to

detect possibility of pneumonia from Chest X-rays, by looking for cloudy region in the same. Conclusive detection will depend on pathological tests.

For the purpose of coding, we have used Python version 2.7 with the OpenCV libraries. We have developed indigenous algorithms for cropping of images to remove abdomen region and for lung boundary identification. We have also looked for Japan Society of Radiological Technology (JSRT) image dataset as mentioned in, but we felt that image quality of that dataset is really very good compared to ground truth images. So we have used analog CXR images that match rural scenario situations. As per availability, our dataset contains only 40 adult CXR.

Computer assisted detection of diseases from CXR are always very helpful at places where there is shortage of skilled radiologist. In countries like India where, we do not have experienced radiologists in rural areas, such tools can be of immense help by automatically screening people who need urgent medical care and further diagnosis. At this stage of the project, we have identified the lung region by rib cage boundary identification and would be computing the area of this region. We have also used Otsu thresholding to segregate the pneumonia cloud from the healthy lung in the lung area, still we are working on other methods that can be adopted for thresholding the CXR images which can yield better results.

2.3 Pneumonia Detection using CNN based Feature Extraction

Author: Dimpy Varshni, Kartik Thakral

Developing an automatic system for detecting pneumonia would be beneficial for treating the disease without any delay particularly in remote areas. Due to the success of deep learning algorithms in analyzing medical images, Convolutional Neural Networks (CNNs) have gained much attention for disease classification. In addition, features learned by pre-trained

CNN models on large-scale datasets are much useful in image classification tasks. In this work, we appraise the functionality of pre-trained CNN models utilized as feature-extractors followed by different classifiers for the classification of abnormal and normal chest X-Rays. We analytically determine the optimal CNN model for the purpose. Statistical results obtained demonstrates that pretrained CNN models employed along with supervised classifier algorithms can be very beneficial in analyzing chest X-ray images, specifically to detect Pneumonia.

The dataset used is ChestX-ray14 consists of 112,120 frontal chest X-ray images from 30,085 patients. Each radiographic image in the dataset is labeled with one or more out of different 14 thoracic diseases. These labels were concluded through Natural Language Processing (NLP) by text-mining disease classification from the associated radiological reports and are expected to be more than 90% accurate.

The proposed pneumonia detection system is done using the ’Densely Connected Convolutional Neural Network’ (DenseNet-169) .The architecture of the proposed model has been divided into three different stages – the preprocessing stage, the feature extraction stage and the classification stage.

The primary goal of using Convolutional Neural Network in most of the image classification tasks is to reduce the computational complexity of the model which is likely to increase if the inputs are images. The original 3-channel images were resized from 1024×1024 into 224×224 pixels to reduce the heavy computation and for faster processing. All of the further techniques has been applied over these downsized images.

This paper primarily aims to improve the medical adeptness in areas where the availability of radiotherapists is still limited. Our study

facilitates the early diagnosis of Pneumonia to prevent adverse consequences (including death) in such remote areas. So far, not much work has been contributed to specifically to detect Pneumonia from the mentioned dataset.

2.4 OBSERVATIONS AND CRITICAL FINDINGS

Thus, comparing and inferring all these existing systems, we could come to a conclusion that data set is important to train the model. After choosing the dataset, we should perform data pre-processing. Then split the data into training and testing data set. Train the model, obtain the coefficients. Calculate the test error using any one of the prescribed methods. CXR images are obtained from the Kaggle website.

The papers have shown that we can approach this problem in multiple ways, with its merits and demerits. But our goal is to minimize the error rate, by doing so the users can get results which they can believe and it is the truth as in that case. Among reviewing multiple journals, we concluded that, neural network model can provide us what we want, by giving the optimum parameters and data. The data must be noise free and should not have any missing data, hence preprocessing is must.

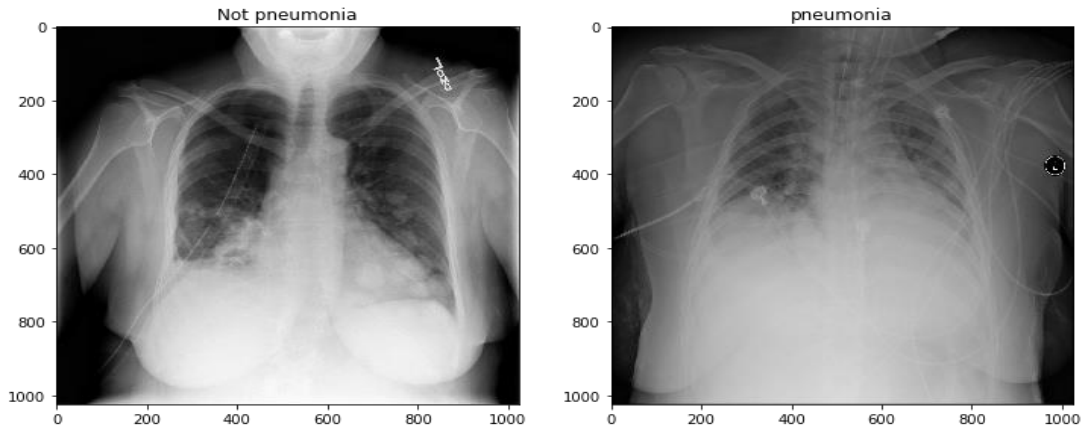


Figure 2.1 X-Ray image for Normal and Pneumonia

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 INTRODUCTION

Requirements Analysis is the process of defining the expectations of the users for an application that is to be built or modified. Requirements analysis involves all the tasks that are conducted to identify the needs of different stakeholders. Therefore, requirements analysis means to analyze, document, validate and manage software or system requirements. High-quality requirements are documented, actionable, measurable, testable, traceable, helps to identify business opportunities, and are defined to a facilitate system design.

3.2 FUNCTIONAL REQUIREMENTS

A functional requirement defines a function of a system or its component, where a function is described as a specification of behavior between outputs and inputs. Functional requirements are usually in the form of "system shall do <requirement>", an individual action or part of the system, perhaps explicitly in the sense of a mathematical function, a black box description input, output, process and control functional model or IPO Model.

3.2.1 HARDWARE REQUIREMENTS

- Installed Physical Memory (RAM) -4.0 GB
- Processor Type -64 bit PROCESSOR
- Operating system -windows 7/windows 10

3.2.2 SOFTWARE REQUIREMENTS

- **Python 3.6.5**

  Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. Python is designed to be

highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

It supports functional and structured programming methods as well as OOP. It can be used as a scripting language or can be compiled to byte-code for building large applications. It provides very high-level dynamic data types and supports dynamic type checking. It supports automatic garbage collection. It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

The latest stable version of python is 3.6, released on December 23rd, 2016. The installation process will automatically install IDLE, pip and documentation as well and will also create shortcuts and file associations so that you don't have to set up environment variables after completion of installation.

- **Jupyter NoteBook**

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

Jupyter supports over 40 programming languages, including Python, R, Julia, and Scala. Your code can produce rich, interactive output: HTML, images, videos, LaTeX, and custom MIME types.

- **Keras**

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

Keras is an API designed for human beings, not machines. It puts user experience front and center. Keras follows best practices for reducing cognitive load: it offers consistent & simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear and actionable feedback upon user error. New modules are simple to add (as new classes and functions), and existing modules provide ample examples. To be able to easily create new modules allows for total expressiveness, making Keras suitable for advanced research.

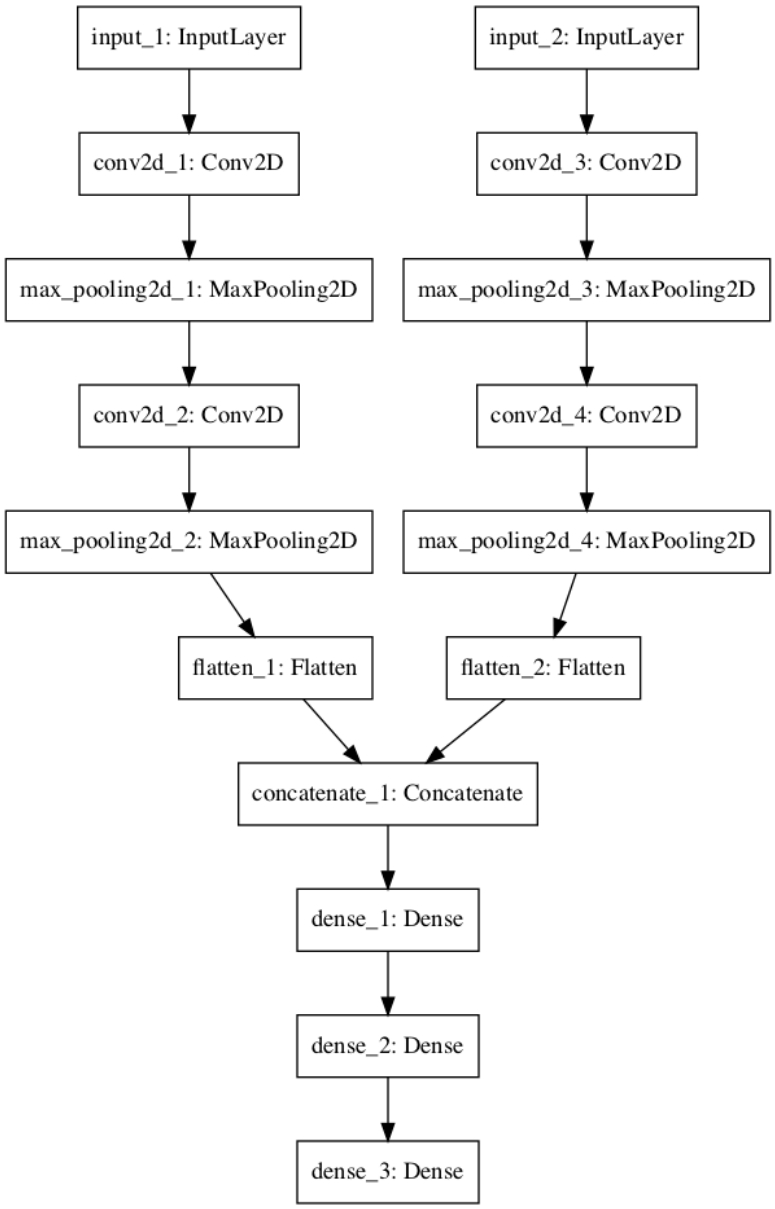


Figure 3.1 Working of Keras

- **TensorFlow**

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

It can build and train ML models easily using intuitive high-level APIs like Keras with eager execution, which makes for immediate model iteration and easy debugging.

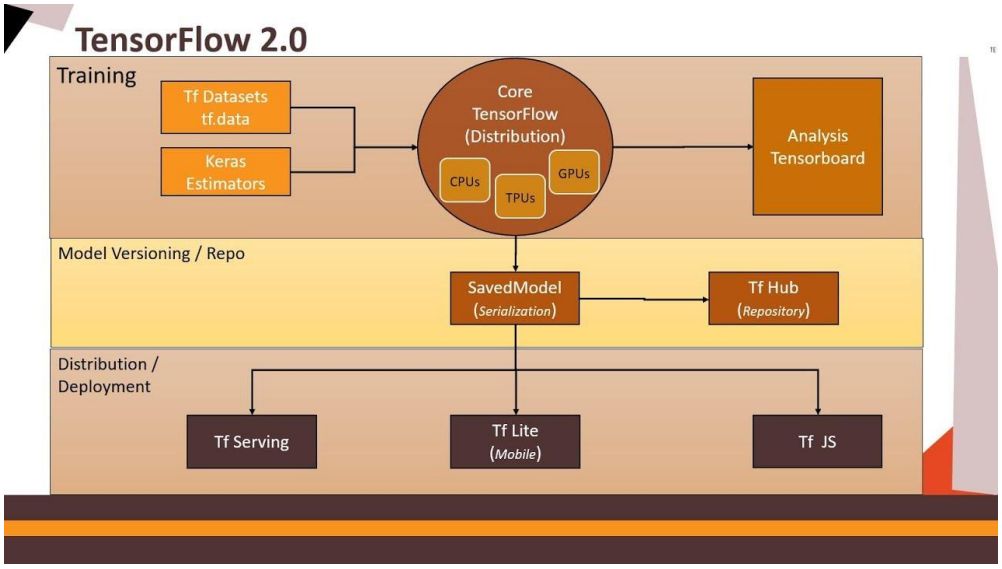


Figure 3.2 TensorFlow Architecture

- **NumPy**

NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays for the Python programming language. It is specifically useful for algorithm developers. Using NumPy, mathematical and logical operations on arrays can be performed.

NumPy is often used along with packages like **SciPy** (Scientific Python) and **Mat−plotlib** (plotting library). This combination is widely used as a replacement for MatLab, a popular platform for technical computing. However, Python alternative to MatLab is now seen as a

more modern and complete programming language. It is open source, which is an added advantage of NumPy.

- **Pandas**

  Pandas is an open-source Python Library providing high structures. The performance data manipulation and analysis tool using its powerful data structures. . The name Pandas is derived from the word Panel Data – an Econometrics from Multidimensional data.

  Prior to Pandas, Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data — load, prepare, manipulate, model, and analyze.

  Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

- **OpenCV**

OpenCV (Open Source Computer Vision Library) is an open-source BSD-licensed library that includes several hundreds of computer vision algorithms. The document describes the so-called OpenCV 2.x API, which is essentially a C++ API, as opposed to the C-based OpenCV 1.x API (C API is deprecated and not tested with "C" compiler since OpenCV 2.4 releases).

OpenCV has a modular structure, which means that the package includes several shared or static libraries. The following modules are available:

Core functionality (core) - a compact module defining basic data structures including the dense multi-dimensional array Mat and basic functions used by all other modules.

Image Processing (imgproc) - an image processing module that includes linear

and non-linear image filtering, geometrical image transformations (resize, affine and perspective warping, generic table-based remapping), color space conversion, histograms.

Video Analysis (video) - a video analysis module that includes motion estimation, background subtraction, and object tracking algorithms.

Camera Calibration and 3D Reconstruction (calib3d) - basic multiple-view geometry algorithms, single and stereo camera calibration, object pose estimation, stereo correspondence algorithms, and elements of 3D reconstruction.

2D Features Framework (features2d) - salient feature detectors, descriptors, and descriptor matchers.

Object Detection (objdetect) - detection of objects and instances of the predefined classes (for example, faces, eyes, mugs, people, cars, and so on).

High-level GUI (highgui) - an easy-to-use interface to simple UI capabilities.

Video I/O (videoio) - an easy-to-use interface to video capturing and video codecs.

- **Pickle- Python Object Serialization**

The pickle module implements a fundamental, but powerful algorithm for serializing and de-serializing a Python object structure. “Pickling” is the process whereby a Python object hierarchy is converted into a byte stream, and “unpickling” is the inverse operation, whereby a byte stream is converted back

into an object hierarchy. Pickling (and unpickling) is alternatively known as “serialization”, “marshalling,” [1] or “flattening”, however, to avoid confusion, the terms used here are “pickling” and “unpickling.

- **Tkinter**

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps −

- Import the Tkinter module.
- Create the GUI application main window.
- Add one or more of the above-mentioned widgets to the GUI application.
- Enter the main event loop to take action against each event triggered by the user.

3.3 NON-FUNCTIONAL REQUIREMENTS

A non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. The plan for implementing non-functional requirements is detailed in the system architecture, because they are usually architecturally significant requirements.

Non-functional requirements are in the form of "system shall be <requirement>", an overall property of the system as a whole or of a particular aspect and not a specific function. The system's overall properties commonly mark the difference between whether the development project has succeeded or failed. Non-functional requirements are often called "quality attributes" of a system. Other terms for non-functional requirements are "qualities", "quality goals", "quality of service requirements", "constraints", "non-behavioral requirements", or "technical requirements". Informally these are sometimes called the "utilities", from attributes like stability and portability.

Non-functional requirements are those requirements which elaborate the performance characteristic of the system and define the constraints on *how* the system will do so.

- Defines the constraints, targets or control mechanisms for the new system.
- Describes how, how well or to what standard a function should be provided.
- They are sometimes defined in terms of metrics (something that can be measured about the system) to make them more tangible.
- Identify realistic, measurable target values for each service level.
- These include reliability, performance, service availability, responsiveness, throughput and security.

3.3.1 PERFORMANCE REQUIREMENTS

Performance requirements define how well the system performs certain functions under specific conditions. Examples are speed of response, throughput, execution time and storage capacity. The service levels comprising performance requirements are often based on supporting end-user tasks. Like most quality attributes, performance requirements are key elements when designing and testing the product. Performance means system throughput under a given workload for a specific timeframe. Performance is validated by testing the scalability and the reliability of hardware, software and network. It is an ongoing process and not an end result. Performance requirements undergo massive changes as features and functionalities get added and eliminated to accommodate evolving business requirements. Performance requirements need to be considered along with other types of quality attributes (e.g., reliability, robustness, security and usability as well as availability, interoperability, safety, efficiency and flexibility).

- **AVAILABILITY**

High availability is when the application remains available and accessible without any interruption and serve their intended function seamlessly. HA is achieved when the end product web application continues to operate, for example, even if one or more servers are blown up, shut down, or simply disengaged unexpectedly from the rest of the network. Availability is the ratio of time a system or component is functional to the total time it is required or expected to function. This can be expressed as a direct proportion (for example, 9/10 or 0.9) or as a percentage (for example, 90%). It can also be expressed in terms of average downtime per week, month or year or as total downtime for a given week, month or year. Sometimes availability is expressed in qualitative terms, indicating the extent to which a system can continue to work when a significant component or set of components goes down.

- **RESPONSE TIME**

Response time is the total amount of time it takes to respond to a request for service. That service can be anything from a memory fetch, to a disk IO, to a complex database query, or loading a full web page. Ignoring transmission time for a moment, the response time is the sum of the service time and wait time. The service time is the time it takes to do the work you requested. For a given request the service time varies little as the workload increases – to do X amount of work it always takes X amount of time. The wait time is how long the request had to wait in a queue before being serviced and it varies from zero, when no waiting is required, to a large multiple of the service time, as many requests are already in the queue and have to be serviced first. Here, it is measured as the time taken for processing the input, calculating the price prediction and displaying the output. Training the data is yet another module or process that takes a lot of time for completion but it is a one-time process.

3.3.2 SAFETY REQUIREMENTS

Safety is the degree to which a software system prevents harm to people or damage to the environment in the intended context of use. Safety

requirements address the user concern for how well the system protects people and the environment from harm. When eliciting safety requirements, consider aspects related to hazard avoidance, hazard detection and removal, and minimizing the damage if an accident occurs.

3.3.3 SECURITY REQUIREMENTS

Concerns of security are defined in terms of confidentiality (protection from disclosure); integrity (protection from unauthorized modification); and availability (protection from destruction). Due to potentially sensitive information being contained within the database, the system should facilitate security and privacy.

The end users should not have access to the database. The user is limited by the access, as he/she can only give the input data, and are allowed to view the results. The level of authorization to give a user is determined by examining the additional properties (metadata) associated with the user's account. Access Control is the process of enforcing the required security for a particular resource.

3.3.4 SOFTWARE QUALITY ATTRIBUTES

Quality can be defined in a different manner. The quality definition may differ from person to person. But finally, there should be some standards. Quality Assurance activities are oriented towards prevention of introduction of defects and Quality control activities are aimed at detecting defects in products and services.

Reliability

Measure if product is reliable enough to sustain in any condition. Should give consistently correct results. Product reliability is measured in terms of working of project under different working environment and different conditions.

Maintainability

Different versions of the product should be easy to maintain. For development it should be easy to add code to existing system, should be easy to upgrade for new features and new technologies time to time. Maintenance should be cost effective and easy. System be easy to maintain and correcting defects or making a change in the software.

Usability

This can be measured in terms of ease of use. Application should be user friendly. Should be easy to learn. Navigation should be simple.

Portability

This can be measured in terms of Costing issues related to porting, Technical issues related to porting, Behavioral issues related to porting.

Correctness

Application should be correct in terms of its functionality, calculations used internally and the navigation should be correct. This means application should adhere to functional requirements.

Efficiency

To Major system quality attribute. Measured in terms of time required to complete any task given to the system. For example, system should utilize processor capacity, disk space and memory efficiently. If system is using all the available resources, then user will get degraded performance failing the system for efficiency. If system is not efficient, then it cannot be used in real time applications.

Integrity or Security

Integrity comes with security. System integrity or security should be sufficient to prevent unauthorized access to system functions, preventing information loss, ensure that the software is protected from virus infection, and protecting the privacy of data entered into the system.

Testability

System should be easy to test and find defects. If required should be easy to divide in different modules for testing.

Flexibility

Should be flexible enough to modify. Adaptable to other products with which it needs interaction. Should be easy to interface with other standard 3rd party components.

Reusability

Software reuse is a good cost efficient and time saving development way. Different code libraries classes should be generic enough to use easily in different application modules. Dividing application into different modules so that modules can be reused across the application.

Interoperability

Interoperability of one system to another should be easy for product to exchange data or services with other systems.

CHAPTER 4

SYSTEM DESIGN

4.1 GENERAL CONSTRAINTS

First, it requires a relationship between the independent and dependent variables. The proportionality between the independent variable and target variable are going to help with the classification and prediction. The following figure shows the data and about its ranges stated in the dataset.

Second, the ideal classifier requires that the errors between observed and predicted values should be normally distributed. Next, there should not be any inconsistencies in the data-set, often the data-set has noise which should be removed. It can affect the results directly.

4.2 GUIDELINES

- Install all the necessary Python packages in prior. They must be the latest version. Even if not, check for any errors or warnings that show the methods that are in near extinction.
- Analyze the data, create a neural network model and fit it with the training data set. After training it, it is rigorously tested with testing data.
- Based on the accuracy, the model is tuned and improved.

4.3 SYSTEM ARCHITECTURE

A system architecture or systems architecture is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system.

A system architecture can consist of system components and the sub-systems developed, that will work together to implement the overall system. There have

been efforts to formalize languages to describe system architecture, collectively these are called architecture description languages (ADLs).

The architecture for this project is defined as follows.

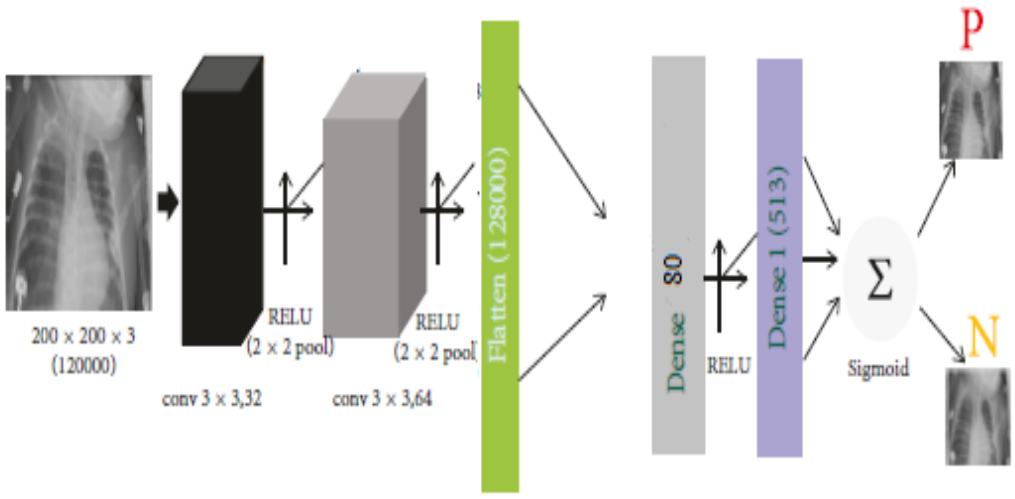


Figure 4.1 Proposed System

4.4 DEVELOPMENT METHODS

Deep learning projects are highly iterative; as you progress through the lifecycle, you'll find yourself iterating on a section until reaching a satisfactory level of performance, then proceeding forward to the next task

Planning and project setup

- Define the task and scope out requirements.
- Determine project feasibility.
- Discuss general model tradeoffs (accuracy vs speed).
- Set up project codebase.

1. Data collection and labeling
   - Define ground truth (create labeling documentation).
   - Build data ingestion.Validate quality of data.
   - Revisit Step 1 and ensure data is sufficient for the task.

2. Model exploration

- Establish baselines for model performance.
- Start with a simple model using initial data.
- Overfit simple model to training data.
- Stay nimble and try many parallel (isolated) ideas during early stages.
- Revisit Step 1 and ensure feasibility.Revisit Step 2 and ensure data quality is sufficient.

3. Model refinement

- Perform model-specific optimizations
- Iteratively debug model as complexity is added.
- Perform error analysis to uncover common failure modes.
- Revisit Step 2 for targeted data collection of observed failures.

4. Testing and evaluation

- Evaluate model on test distribution.
  - May necessitate additional training.Understand differences between train and test set distributions.
- Revisit model evaluation metric.
  - Ensure that this metric drives desirable downstream user behavior

5. Model deployment

- Expose model.
- Deploy new model to small subset of users to ensure everything goes smoothly, then roll out to all users.
- Maintain the ability to roll back model to previous versions.
- Monitor live data and model prediction distributions.

6. Ongoing model maintenance

- Understand that changes can affect the system in unexpected ways.
- Periodically retrain model to prevent model staleness.
- If there is a transfer in model ownership, educate the new team.

CHAPTER 5

SYSTEM IMPLEMENTATION

5.1 IMPLEMENTATION

Steps to built the model are as follow,

1. The architecture of our proposed model consists of Two major layers i. Feature Extraction

   ii. classifier.

2. Each layer in the feature extraction layer takes its immediate preceding layer's output as input.
3. Its output is passed as an input to the succeeding layers.
4. Our architecture consists of convolution , Relu, Max pooling and Classifier.
5. The feature extractors feature consists of Conv2D 3 X 3, 32; Relu activation function; Max pool 2D 2 X 2; conv2D 3 X 3, 64; Relu activation function; Max pool 2D 2 X 2; Flatten layer; Dense 80; Dense 1; sigmoid activation function ;
6. Classifier consist of Sigmoid Activation Function.

5.2 METHODS

5.2.1 DATASET

In this research, CXR images are obtained from the Kaggle website. The dataset[7] consists of three main folders training, testing, and validation and each contains two subfolders pneumonia (P) and normal(N) chest X-ray images. No of Pneumonia images are 4,273, and Normal images are 1,583, a total of 5,856 X-ray images of anterior-posterior chests are there.

5.2.2 PREPROCESSING AND AUGMENTATION

A smart library is made up of 5,856 x-ray images to preprocess the data. By using this smart library[8],it automatically recognizes the categories inside the dataset, resizes the images even if we do not specify the image size by default

the image size is 50 x 50. Pickle object is used to reduce the time consumption for loading the data.

5.2.3 MODEL

The architecture of our proposed model consists. Two major layers (i. Feature Extraction ii. Classifier). Each layer in the feature extraction layer takes its immediate preceding layer's output as input, and its output is passed as an input to the succeeding layers. Our architecture consists of convolution, relu, max pooling, and classifier. The feature extractors feature consists of Conv2D 3 X 3, 32; Relu activation function; Max pool 2D 2 X 2; conv2D 3 X 3, 64; Relu activation function; Max pool 2D 2 X 2; Flatten layer; Dense 80; followed by Dense 1 and sigmoid activation function that performs the classification tasks. .

The classifier is placed at the far end of the proposed convolutional neural network (CNN) model. Classifier requires individual features (vectors) to perform computations like any other classifier. Therefore, the output of the feature extractor (CNN part) is converted into a 1D feature vector for the classifiers. .is process is known as flattening where the output of the convolution operation is flattened to generate one lengthy feature vector for the dense layer to utilize in its final classification process. The classification classification layer contains a flattened layer, two dense layers of size 80 and 1, respectively, a RELU between the two dense layers and a sigmoid activation function that performs the classification tasks.

Function	***Explanations***
Convolution2D	Convolutional layer, sliding window convolution to 2-dimensional input information;
MaxPooling2D	Maximum pooling layer, imposing a maximum pooling on the spatial domain signal;
RELU	Rectified Linear Unit, which performs linear rectification activation on the input vector of the upper layer neural

	network and outputs nonlinear results.
Flatten	The Flatten layer is used to translate the multidimensional input information into one-dimensional information.
Dense	It feeds all outputs from the previous layer to all its neurons, each neuron providing one output to the next layer.
Sigmoid	It is an activation function for multi-class neural network output.

5.3 SPLITTING TRAINING AND TESTING DATA

The dataset can be divided into two subsets:

- **training set**—a subset to train a model.
- **test set**—a subset to test the trained model

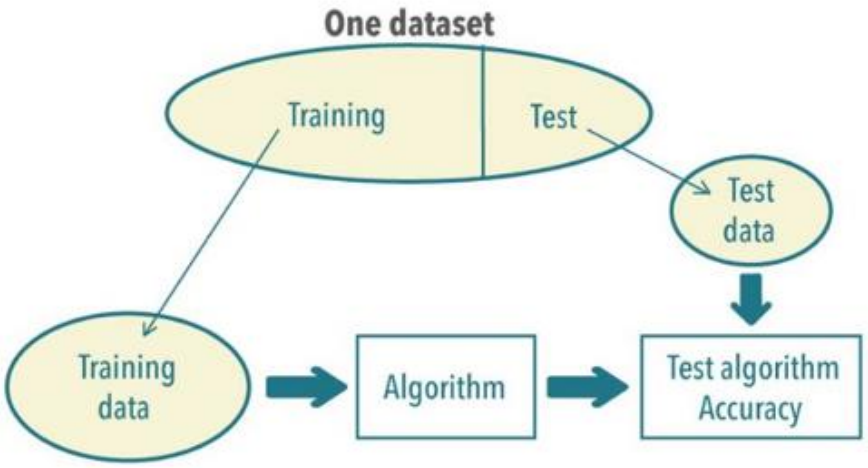


Figure 5.1 Splitting the dataset into training and testing data

Make sure that your test set meets the following two conditions:

- Is large enough to yield statistically meaningful results.
- Is representative of the data set as a whole. In other words, don't pick a test set with different characteristics than the training set.

Assuming that your test set meets the preceding two conditions, your goal is to create a model that generalizes well to new data. Our test set serves as a proxy for new data. For example, consider the following figure. Notice that the model learned for the training data is very simple. This model doesn't do a perfect job—a few predictions are wrong. However, this model does about as well on the test data as it does on the training data. In other words, this simple model does not overfit the training data.

5.4 MODEL

5.4.1 FITTING THE MODEL

A model that is well-fitted produces more accurate outcomes, a model that is overfitted matches the data too closely, and a model that is underfitted doesn’t match closely enough. Each machine learning algorithm has a basic set of parameters that can be changed to improve its accuracy. During the fitting process, you run an algorithm on data for which you know the target variable, known as “labeled” data, and produce a machine learning model. Then, you compare the outcomes to real, observed values of the target variable to determine their accuracy. Next, you use that information to adjust the algorithm’s standard parameters to reduce the level of error, making it more accurate in uncovering patterns and relationships between the rest of its features and the target.

5.4.2. ACCURACY AND ERROR RATE

A prediction error is the failure of some expected event to occur. When predictions fail, humans can use metacognitive functions, examining prior predictions and failures and deciding, for example, whether there are correlations and trends, such as consistently being unable to foresee outcomes accurately in particular situations. Applying that type of knowledge can inform decisions and improve the quality of future predictions.

Predictive analytics software processes new and historical data to forecast activity, behavior and trends. The programs apply statistical analysis techniques,

analytical queries and machine learning algorithms to data sets to create predictive models that quantify the likelihood of a particular event happening.

$$\textbf{Accuracy (\%)} = \frac{TruePositive + TrueNegative}{Total\ no.\ of\ instances} \text{X } 100$$

These terminologies are explained below:

1. **True Positive:** No. of instances predicted positive and are actually positive.
2. **True Negative:** No. of instances predicted positive but are actually negative.
3. **Total no. of Instances:** The sum of all the instances that have been classified by the classifier.

Dataset	*Accuracy of Proposed Model*
Training	96.82%
Test	98.53%
Validation	98. 91%

5.4.3. TUNING AND IMPROVEMENT

The flow is a loop, as discussed in the design phase. Once the model is developed, it will be tested to check the accuracy. Then again rigorous data analysis, we will uncover more insights about the data. In this case, as a first step the data set is directly applied to the neural network. Then the feature extraction and selection techniques and data preprocessing were done and applied to the neural network model. As we saw drastic improvements, since the data analysis plays a vital role in deep learning projects.

CHAPTER 6

RESULTS

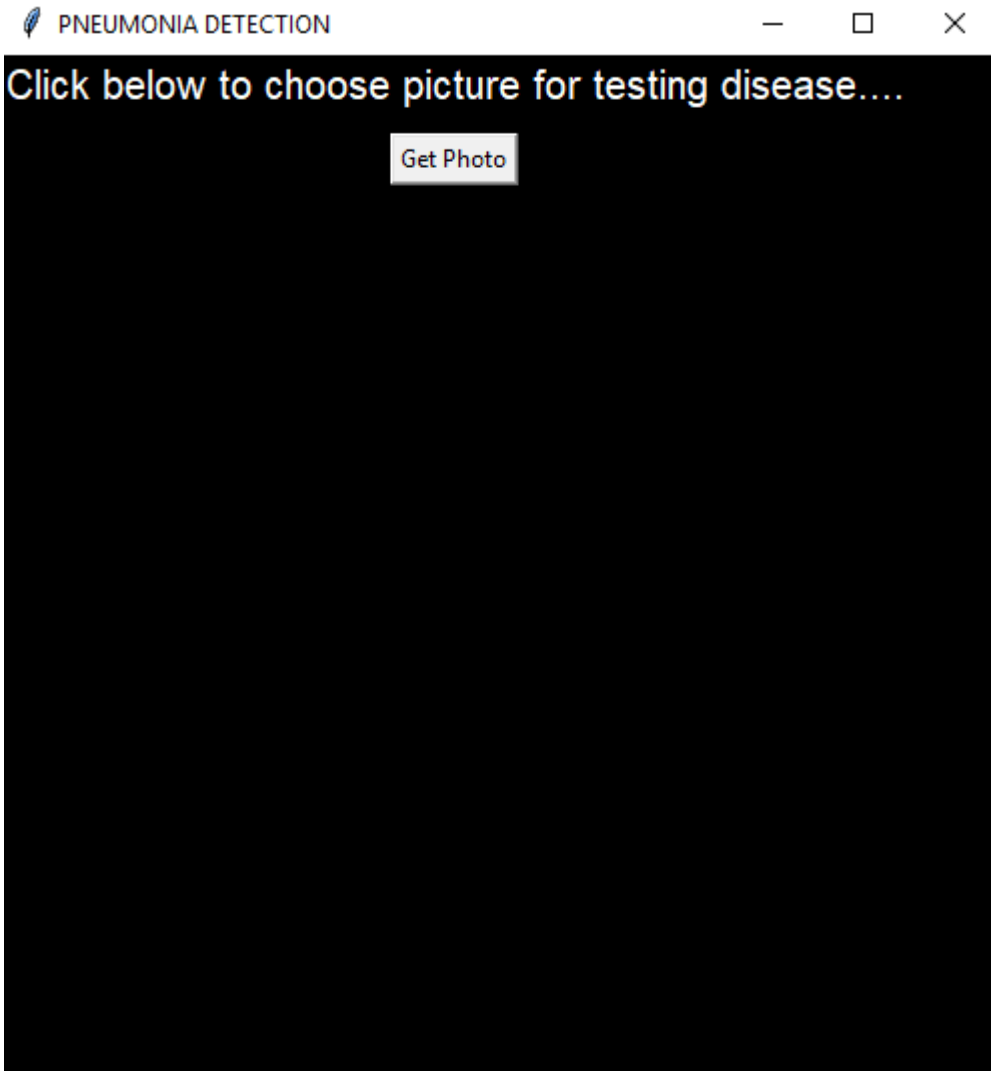


Figure 6.1 Home page

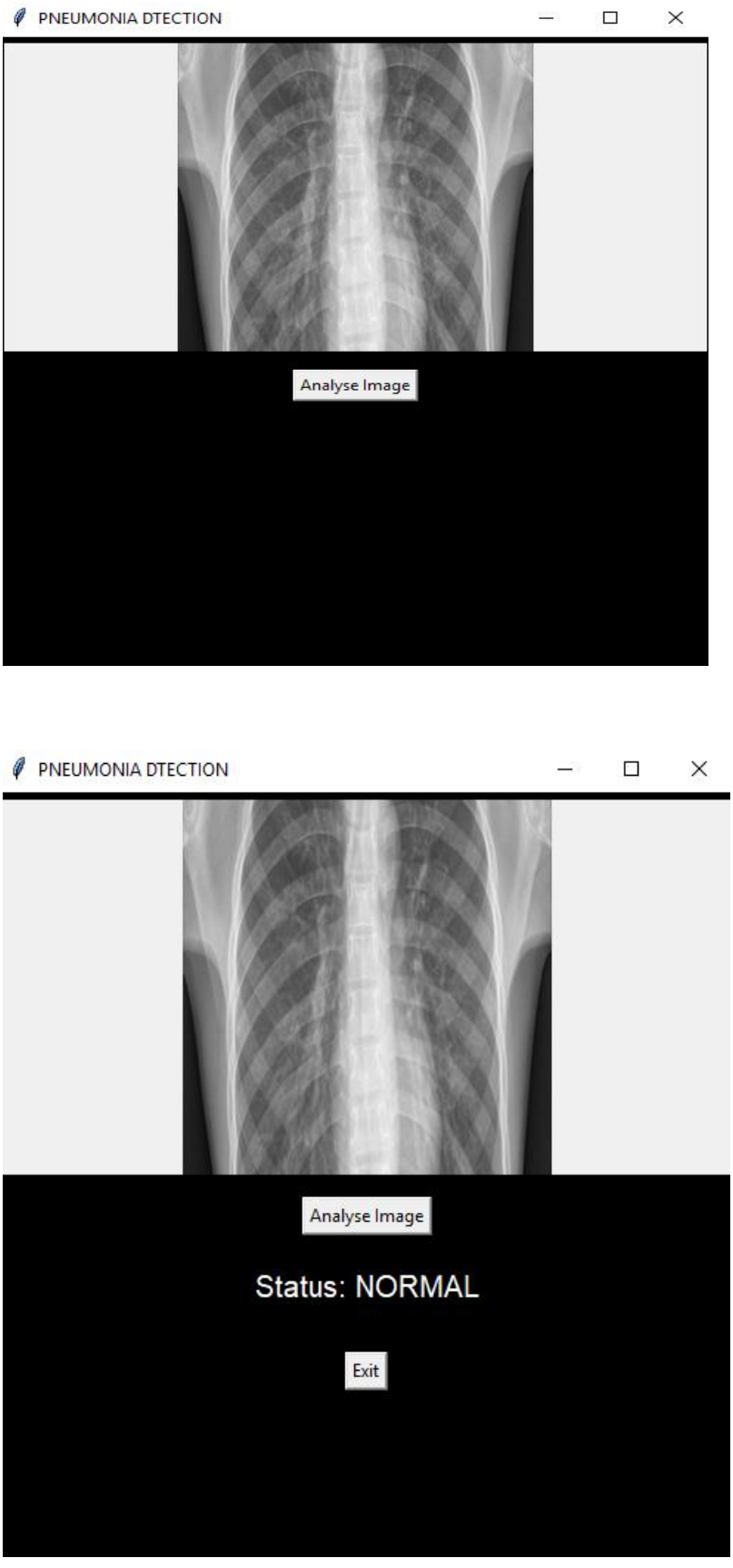
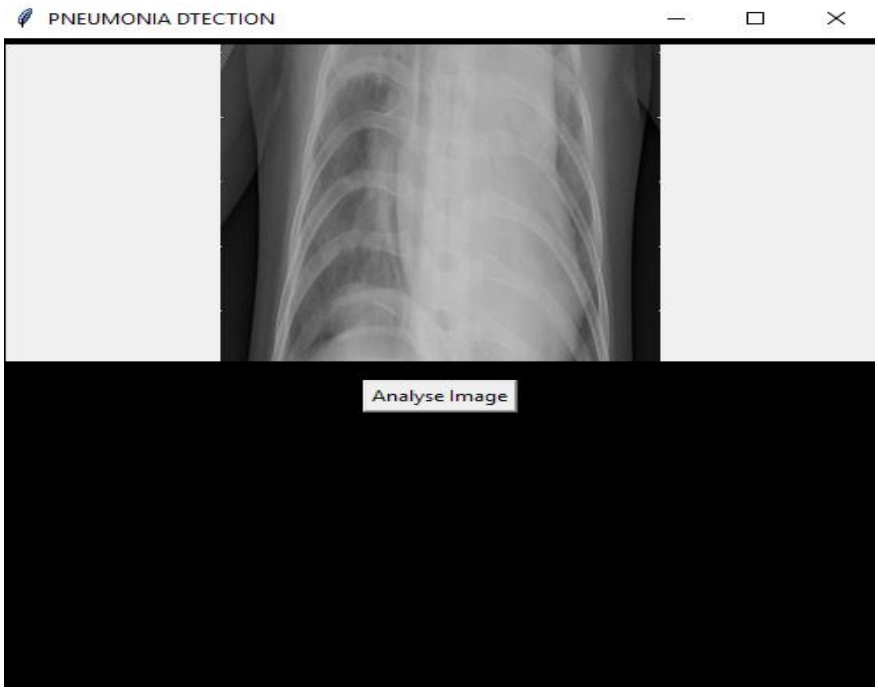


Figure 6.2 Output for Normal CXR



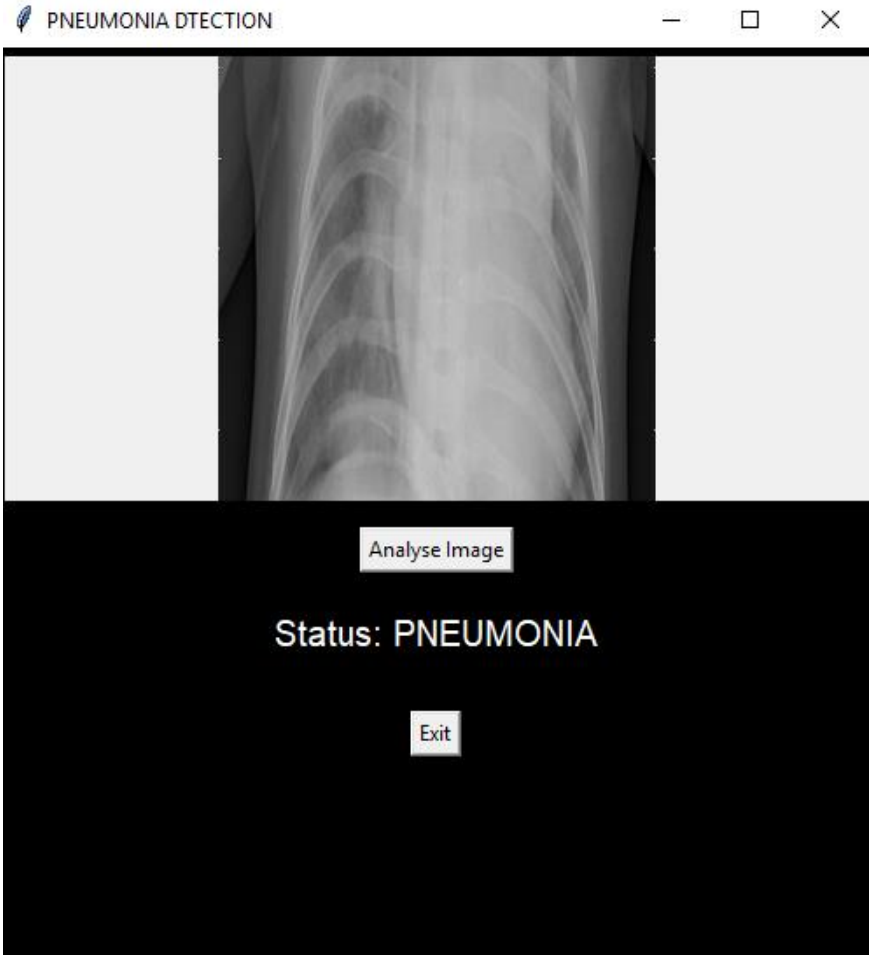


Figure 6.2 Output for Pneumonia CXR

CHAPTER 7

FUTURE ENHANCEMENT

We have proposed a model that classifies positive and negative pneumonia data from a collection of X-ray images. Our model is built from scratch, which separates it from other methods that rely heavily on transfer learning approach.

In the future, this work will be extended to detect and classify X-ray images consisting of COVID-19 and pneumonia. Distinguishing X-ray images that contain COVID-19 and pneumonia has been a big issue in recent times, and our next the approach will tackle this problem.

CHAPTER 8

CONCLUSION

This project helps the patients to clear their doubts on their case, about the tumor whether it is safe or not. It improves the results from the manual method, which is prone to human errors. The model is perfectly balanced, hence won’t lie on any subjective feature. In other words, there won’t be a biased model, which relies on the single attribute. The error rates have been improved, but not to 100%. There is always a small percent of possibility that it will give wrong results.

APPENDIX

9.1 LIBRARY FILES

```
import tensorflow as tf
import numpy as np
import os
import sys
import cv2
import matplotlib.pyplot as plt
import pickle
import random
import pandas as pd

import tensorflow as tf
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense,Dropout,Activation,
Flatten,Conv2D,MaxPooling2D
import pickle

from sklearn.model_selection import train_test_split

import matplotlib.pyplot as plt
%matplotlib inline
```

9.2 PREPROCESSING

```
class MasterImage(object):

    def __init__(self,PATH='', IMAGE_SIZE = 50, CATEGORIES=[]):
        self.PATH = PATH
        self.IMAGE_SIZE = IMAGE_SIZE
        self.CATEGORIES = CATEGORIES
        self.image_data = []
        self.x_data = []
        self.y_data = []

    def Process_Image(self):

        """
        Return Numpy array of image
        :return: X_Data, Y_Data
        """

        for categories in self.CATEGORIES:
# Iterate over categories
```

```
            train_folder_path = os.path.join(self.PATH, categories)
# Folder Path
            class_index = self.CATEGORIES.index(categories)
# this will get index for classification

            for img in os.listdir(train_folder_path):
# This will iterate in the Folder
                new_path = os.path.join(train_folder_path, img)
# image Path

                try:        # if any image is corrupted
                    image_data_temp =
cv2.imread(new_path,cv2.IMREAD_GRAYSCALE)                 # Read
Image as numbers
                    image_temp_resize =
cv2.resize(image_data_temp,(self.IMAGE_SIZE,self.IMAGE_SIZE))

self.image_data.append([image_temp_resize,class_index])
                except:
                    pass

        data = np.asanyarray(self.image_data)

        # Iterate over the Data
        for x in data:
            self.x_data.append(x[0])        # Get the X_Data
            self.y_data.append(x[1])        # get the label

        X_Data = np.asarray(self.x_data) / (255.0)      # Normalize
Data
        Y_Data = np.asarray(self.y_data)

        return X_Data,Y_Data

    def pickle_image(self):

        """
        :return: None Creates a Pickle Object of DataSet
        """

        X_Data,Y_Data = self.Process_Image()

        pickle_out = open('X_Data','wb')
        pickle.dump(X_Data, pickle_out)
        pickle_out.close()
```

```
        pickle_out = open('Y_Data', 'wb')
        pickle.dump(Y_Data, pickle_out)
        pickle_out.close()

        print("Pickled Image Successfully ")

        return X_Data,Y_Data

    def load_dataset(self):

        try:
            X_Temp = open('X_Data','rb')
            X_Data = pickle.load(X_Temp)

            Y_Temp = open('Y_Data','rb')
            Y_Data = pickle.load(Y_Temp)

            print('Reading Dataset from PIckle Object')

            return X_Data,Y_Data

        except:
            print('Could not Found Pickle File ')
            print('Loading File and Dataset  ..........')
            X_Data,Y_Data = self.pickle_image()
            return X_Data,Y_Data

a = MasterImage(PATH='/Users/ELCOT/Desktop/prjct/val',
                    IMAGE_SIZE=80,
                    CATEGORIES=['NORMAL','PNEUMONIA'])

X_Data,Y_Data = a.load_dataset()
X_Data = X_Data.reshape(-1,80,80,1)
print(X_Data.shape)
```

9.3 SPLITTING THE TRAINING AND TESTING DATA

```
X_Train, X_Test, Y_Train, Y_Test = train_test_split(X_Data, Y_Data,
test_size=0.3,random_state=101)
```

9.4 TRAINING THE MODEL

```
model = Sequential()
model.add(Conv2D(200, (3, 3), input_shape=X_Data.shape[1:]))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
```

```
model.add(Conv2D(100, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(80))
model.add(Dense(1))
model.add(Activation('sigmoid'))
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

model.fit(X_Data, Y_Data, batch_size=40, epochs=5,
validation_split=0.3)
model.save('model.h5')
```

9.5 EVALUATING THE MODEL

```
model.evaluate(X_Test,Y_Test,batch_size=40)
```

9.6 LOADING THE MODEL

```
model = tf.keras.models.load_model('model.h5')
```

9.7 USER INTERFACE

```
import tkinter as tk
from tkinter.filedialog import askopenfilename
import shutil
import os
import sys
from PIL import Image, ImageTk

window = tk.Tk()

window.title("PNEUMONIA DETECTION")

window.geometry("500x510")
window.configure(background ="black")

title = tk.Label(text="Click below to choose picture for testing
disease....", background = "black", fg="white", font=("", 15))
title.grid()
def exit():
    window1.destroy()

def prepare(filepath):
```

```
    training_date = []

    img_array = cv2.imread(filepath,cv2.IMREAD_GRAYSCALE)
    new_array = cv2.resize(img_array,(80,80))
    new_image =  new_array.reshape(-1,80,80,1)
    return new_image

def analysis():

    #model = tf.keras.models.load_model('model.h5')
    test = model.predict([prepare(filepath=filepath)])
    CATEGORIES=['NORMAL','PNEUMONIA']
    print(CATEGORIES[int(test[0][0])])
        #model_out = model.predict([data])[0]
    message = tk.Label(text='Status: '+CATEGORIES[int(test[0][0])],
background="black",
                           fg="white", font=("", 15))
    message.grid(column=0, row=3, padx=10, pady=10)
    button = tk.Button(text="Exit", command=exit)
    button.grid(column=0, row=9, padx=20, pady=20)

def openphoto():
    global filepath
    global window1
    window.destroy()
    window1 = tk.Tk()
    window1.title("PNEUMONIA DTECTION")
    window1.geometry("500x520")
    window1.configure(background="black")

    filepath =
askopenfilename(initialdir='C:/Users/ELCOT/Desktop/prjct/val',
title='Select image for analysis ',
                           filetypes=[('image files', '.jpeg')])
    load = Image.open(filepath)
    load= load.resize((250,490),Image.ANTIALIAS)
    render = ImageTk.PhotoImage(load)
    img = tk.Label(image=render, height="250", width="490")
    img.image = render
    img.place(x=0, y=0)
    img.grid(column=0, row=0, padx=5, pady = 5)
    #title.destroy()
    #button1.destroy()
    button2 = tk.Button(text="Analyse Image", command=analysis)
    button2.grid(column=0, row=2, padx=10, pady = 10)
    window1.mainloop()
```

```
button1 = tk.Button(text="Get Photo", command = openphoto)
button1.grid(column=0, row=1, padx=10, pady = 10)
window.mainloop()
```

REFERENCES

[1] World Health Organization, *Household Air Pollution and*

Health [Fact Sheet], WHO, Geneva, Switzerland, 2018, http://

www.who.int/newa-room/fact-sheets/detail/household-airpollution-and-health.

[2] I. Rudan, L. Tomaskovic, C. Boschi-Pinto, and H. Campbell, “Global estimate of the incidence of clinical pneumonia among children under five years of age,” *Bulletin of the World Health Organization*, vol. 82, pp. 85–903, 2004.

[3] V. Narasimhan, H. Brown, A. Pablos-Mendez, et al.,

“Responding to the global human resources crisis,”.*e*

Lancet, vol. 363, no. 9419, pp. 1469–1472, 2004.

[4] S. Naicker, J. Plange-Rhule, R. C. Tutt, and J. B. Eastwood, “Shortage of healthcare workers in developing countries,” *Africa, Ethnicity & Disease*, vol. 19, p. 60, 2009.

[5] Franquet, Tomás. "Imaging of community-acquired

pneumonia." *Journal of thoracic imaging* 33.5 (2018): 282-294.

[6] Bingchuan Li and et al.,” Attention Guided Convolution Neural Network for Detecting Pneumonia on Chest X-Rays”(2019).

[7] 2018. Kaggle URL: (2018).https://www.kaggle.com/nih-chestxrays/data

[8] https://github.com/soumilshah1995/Smart-Library-to-load-image-Dataset-for-Convolution-Neural-Network-Tensorflow-Keras-