JOMO KENYATTA UNIVERSITY OF AGRICULTURE AND TECHNOLOGY

DEPARTMENT OF ELECTRICAL AND ELECTRONIC

ENGINEERING

BSc Electrical and Electronic Engineering

Final Year Project Report

Title: Machine Learning-Based Pneumonia Detection System

MBUGUA GEORGE KAMUNDIA

ENE211-0008/2018

Mr. A. Muhia

2023

DECLARATION

This project is my original work, except where due acknowledgement is made in the text, and to the best of my knowledge has not been previously submitted to Jomo Kenyatta University of Agriculture and Technology or any other institution for the Award of a degree or diploma.

SIGNATURE: ……………………………………………. DATE:………………………………..

NAME: MGUBUA GEOGE KAMUNDIA

REG NO: ENE211-0008/2018

TITLE OF PROJECT: **Machine Learning-Based Pneumonia Detection System**

SUPERVISOR CONFIRMATION:

This project has been submitted to the Department of Electrical and Electronic Engineering, Jomo Kenyatta University of Agriculture and Technology, with my approval as the University supervisor:

SIGNATURE:…………………………………………. DATE:……………………………..

NAME: Mr. Muhia

DEDICATION

This project, the Machine Learning-Based Pneumonia Detection System, is dedicated to all individuals whose lives have been touched by the profound impact of respiratory illnesses. In honor of those who have faced the challenges of pneumonia, our endeavor seeks to contribute to the advancement of medical technology and healthcare solutions.

To the countless patients who have endured the uncertainty of misdiagnosis and prolonged waiting times for crucial diagnostic results, this project stands as a commitment to improving the accuracy and efficiency of pneumonia detection. It is our sincere dedication to mitigating the potential consequences of delayed diagnosis, providing healthcare professionals with a more effective tool to enhance patient care.

We extend our heartfelt gratitude to the medical professionals, researchers, and healthcare providers who tirelessly strive to enhance the quality of healthcare services. This project is dedicated to their unwavering dedication and commitment to the well-being of individuals worldwide.

May the Machine Learning-Based Pneumonia Detection System serve as a testament to the collaborative efforts and technological innovations aimed at alleviating the burden of respiratory diseases. Through dedication and innovation, we aspire to make a meaningful impact on the lives of patients and contribute to the continual improvement of healthcare practices.

This project is dedicated to a future where advanced technologies play a pivotal role in the early detection and effective management of pneumonia, fostering a healthier and more resilient global community.

ACKNOWLEDGEMENT

I would like to express my sincere gratitude and appreciation to all those who have contributed to the successful development and implementation of the Machine Learning-Based Pneumonia Detection System project.

First and foremost, I extend my heartfelt thanks to my academic advisors and mentors for their invaluable guidance and unwavering support throughout the project. Their expertise and encouragement have been instrumental in shaping the direction of the research and ensuring its successful execution.

I am immensely thankful to Jomo Kenyatta University of Agriculture and Technology for providing the necessary resources and infrastructure that facilitated the smooth progress of this project. The collaborative environment and access to cutting-edge technology have greatly contributed to the project's overall success.

I extend my appreciation to the healthcare professionals who generously shared their insights and expertise, offering valuable perspectives on the real-world implications of the Pneumonia Detection System. Their input has been crucial in ensuring the system's practicality and relevance in a clinical setting.

I am also grateful to my peers and colleagues who contributed to the project through discussions, brainstorming sessions, and collaborative efforts. Their diverse perspectives and collective efforts have enriched the project's outcomes.

Last but not least, I would like to acknowledge the support and understanding of my family and friends. Their encouragement and patience during the challenging phases of the project were indispensable.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
CBC	Complete Blood Count
CNN	Convolutional Neural Network
EHR	Electronic Health Record
EWS	Early Warning system
GAN	Generative Adversarial Network
KEMRI	Kenya Medical Research Institute
ML	Machine Learning
ReLu	Rectified Linear Unit
XAI	Explainable Artificial Intelligence

ABSTRACT

The final year project was in Digital Health as a cutting-edge computer vision project that integrated machine learning techniques to detect the presence of viral or bacterial pneumonia in the lungs using X-ray scan images. The primary aim of this project was to address two significant problems faced by the healthcare industry, particularly in Kenya.

The first problem was the high rate of misdiagnosis, which leads to severe consequences such as death or permanent disability. Statistics show that 3 in 10 Kenyan patients are misdiagnosed, a considerable cause for concern. The project aims to significantly reduce the chances of misdiagnosis by providing a more accurate and reliable diagnostic tool for healthcare professionals.

The second problem being addressed is the lack of enough radiologists, often leading to longer waiting times for patients to receive their diagnostic results. Radiologists are not always readily available, and some hospitals may not have a resident radiologist. This results in patients waiting for days to receive their results, leading to delayed results and eventually death or permanent damage. By providing a software solution that can be easily integrated into compatible machines, aims to speed up the diagnostic process and reduce waiting times.

The implementation for this project involves developing a software solution that can be easily integrated with compatible machines. This software solution utilizes computer vision techniques and machine learning algorithms to analyze X-ray scan images and accurately detect the presence of pneumonia.

CHAPTER ONE

1.0 INTRODUCTION

1.1 BACKGROUND INFORMATION

Pneumonia is a significant public health concern worldwide, causing substantial morbidity and mortality, particularly in developing countries. Developing countries' poor sanitary conditions and lack of medical personnel make them vulnerable. Accurate and timely diagnosis of pneumonia is crucial for effective treatment and patient management. X-ray imaging is widely used for diagnosing pneumonia, but the interpretation of X-ray images requires specialized medical expertise and can be time-consuming. In 2017, this disease was associated with the deaths of over 808,000 children under the age of five, worldwide, accounting for 15% of all deaths in the mentioned age group [1]. In Kenya, majority of the applications of machine learning techniques have been adopted in the finance and agricultural sector. Applying machine learning techniques in the health sector has the potential to enhance diagnostic capabilities and improve patient outcomes.

Pneumonia is an infection that inflames the air sacs in either one or both lungs, which may make the air sacs to be filled with fluid or pus hence making it difficult to breathe. Pneumonia can be caused by bacteria, fungal or viruses and can range from mild to severe [2]. In general, it can take anywhere from a few days to several weeks for pneumonia to become life-threatening. A chest X-ray is often used to diagnose pneumonia. Blood tests such as complete blood count (CBC) checks whether your immune system is fighting an infection. Pulse oximetry measures how much oxygen is in your blood.

Machine learning, a subset of artificial intelligence, has revolutionized various industries and sectors, including object detection and the medical field. By leveraging large datasets and powerful algorithms, machine learning models can be trained to detect objects accurately and make informed decisions. Machine learning has emerged as a valuable tool, enabling advancements in diagnosis, treatment, early warning systems (EWS) and patient care. Use cases of machine learning in the medical field include disease diagnosis employed in medical imaging, drug discovery in pharmaceutical research, personalized medicine by leveraging patient data, genetic information and treatment outcomes, disease outbreak prediction used in early warning systems and so much more.

1.2 PROBLEM STATEMENT

The timely and accurate diagnosis of pneumonia remains a critical challenge, particularly in resource-limited settings. The interpretation of X-ray images for pneumonia detection requires specialized medical expertise, which may not always be readily available. Pneumonia killed 740 180 children under the age of 5 in 2019, accounting for 14% of all deaths of children under 5 years old but 22% of all deaths in children aged 1 to 5 years worldwide. Additionally, the scarcity of trained radiologists and the time-consuming nature of manual image analysis hinder efficient diagnosis and patient management. In 2020, Kenya's approximate population was recorded to have 53,771,296 with only 146 radiologist workforce. Thus, there is a pressing need to develop a solution that leverages machine learning to enhance the efficiency and accuracy of pneumonia detection in X-ray images, enabling improved healthcare outcomes for patients in Kenyan hospitals.

1.3 JUSTIFICATION

Enhanced Efficiency and Timely Diagnosis: Implementing machine learning techniques for pneumonia detection in X-ray images will significantly improve the efficiency of diagnosis. By leveraging pre-trained models on large datasets, the project will enable rapid analysis of X-ray images, reducing the time required for interpretation. This accelerated diagnosis will facilitate timely interventions and treatment, improving patient outcomes.

Overcoming Expertise and Resource Limitations: Kenya, like many developing countries, faces a shortage of specialized medical expertise, particularly in radiology. The proposed project will reduce the dependency on scarce radiologists by automating the pneumonia detection process. By deploying a machine learning solution, hospitals in Kenya can leverage the power of technology to bridge the gap in expertise, thereby ensuring accurate and consistent diagnoses, even in resource-limited settings.

Increased Accuracy and Consistency: Human interpretation of X-ray images can be subjective, leading to diagnosis inconsistencies. The project will improve the accuracy and reliability of pneumonia detection by utilizing machine learning algorithms. These algorithms can be trained on diverse and extensive datasets, capturing various pneumonia characteristics and variations. As a result, the project will provide consistent and objective diagnoses, reducing misdiagnosis rates and improving patient care.

Scalability and Cost-effectiveness: Machine learning techniques have the potential for scalability and cost-effectiveness. Once the machine learning model is developed and validated, it can be deployed across multiple healthcare facilities in Kenya, reaching a larger population and impacting a broader range of patients. By leveraging existing machine learning frameworks and pre-trained models, the project offers a cost-effective solution that maximizes the use of available resources.

1.4 OBJECTIVES

1.4.1. Main objective

To implement a machine learning system to detect pneumonia.

1.4.2. Specific objectives

a) To gather and curate the requisite dataset essential for the construction of the machine learning model.

b) To perform data pre-processing and cleansing procedures to enhance the model's accuracy and reliability.

c) To develop a robust and efficient machine learning model capable of accurately identifying pneumonia from X-ray images.

d) To deploy the developed model into a practical and operational system for seamless integration into healthcare settings.

CHAPTER TWO

2.0 LITERATURE REVIEW

In the traditional approach, statistical methods have typically relied on mathematical equations to characterize patterns within data. For instance, linear regression offers a simplified representation known as the 'line of best fit'. However, the advent of Artificial Intelligence (AI) has introduced techniques that go beyond simple equations and uncover intricate associations in data. One such technique is the use of neural networks, which mimic the interconnected structure of the human brain using numerous interconnected neurons. This enables Machine Learning (ML) systems to tackle complex problem-solving tasks. These systems can observe and rapidly process an extensive array of inputs virtually without limits [3]. Furthermore, they can continually learn from each new case, quickly accumulating knowledge from many examples that surpasses what a single clinician could experience in a lifetime, all within a matter of minutes [4].

Machine learning has emerged as a transformative technology in medicine, revolutionizing various aspects of healthcare delivery, diagnosis, treatment, and patient care. Current trends in medicine show a growing interest in applying machine learning techniques to improve medical outcomes, enhance efficiency, and enable personalized healthcare [5].

One notable trend is the utilization of machine learning in medical image analysis. Deep learning algorithms and convolutional neural networks (CNNs) have demonstrated remarkable performance in image segmentation, tumor detection, and disease classification tasks. These advancements can potentially improve diagnostic accuracy, aid in early disease detection, and assist radiologists in making more informed decisions [6].

In addition to medical imaging, machine learning is applied to diverse healthcare domains, including genomics, electronic health records (EHRs), drug discovery, and personalized medicine. Machine learning models can analyze genomic data to identify disease risk factors, predict treatment responses, and enable precision medicine approaches. By analyzing EHRs, machine learning algorithms can identify patterns and risk factors for adverse events, supporting clinical decision-making and optimizing patient care [7].

In Kenya, there is a growing interest in leveraging machine learning in healthcare. The Kenya Medical Research Institute (KEMRI) and other research institutions are actively exploring machine learning algorithms to address public health challenges, including disease surveillance, outbreak prediction, and resource allocation. Furthermore, initiatives are underway to develop machine learning models that can aid in diagnosing and managing prevalent diseases in Kenya, such as malaria and tuberculosis [8].

CLASSIFICATION OF MACHINE LEARNING ALGORITHMS

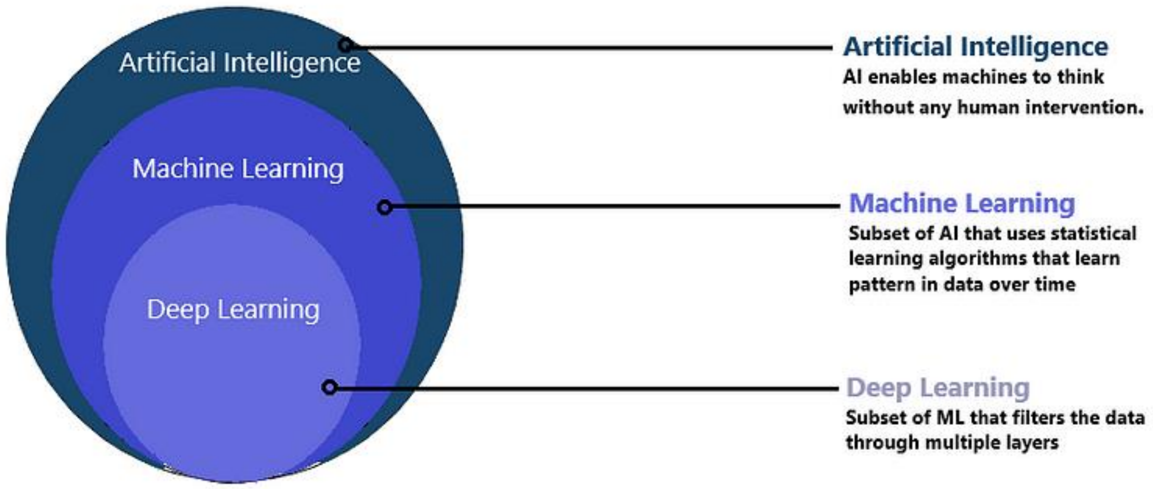

Artificial Intelligence
Machine Learning
Deep Learning
Artificial Intelligence
AI enables machines to think without any human intervention.
Machine Learning
Subset of AI that uses statistical learning algorithms that learn pattern in data over time
Deep Learning
Subset of ML that filters the data through multiple layers

Figure 2.0. 1 AI Classification

Artificial Intelligence (AI)

AI is a broad field of computer science that seeks to create intelligent machines that can think and act like humans [9]. AI encompasses a wide range of techniques and approaches, including:

- Search algorithms: These algorithms are used to find the best solution to a problem, such as the shortest path between two points or the most relevant documents in a search query.
- Knowledge representation: This is the study of how to represent knowledge in a computer so that it can be used by AI systems.
- Reasoning: This is the ability to use knowledge to make logical inferences.
- Learning: This is the ability to improve performance on a task by learning from experience.

Machine Learning (ML)

ML is a subset of AI that focuses on the development of algorithms that can learn from data. ML algorithms are able to identify patterns in data and make predictions or decisions based on those patterns. There are three main types of ML algorithms:

- Supervised learning: These algorithms are trained on a labeled dataset, where each example is labeled with the correct output. The algorithm learns to map the input data to the output labels. Some common supervised learning algorithms include classification and regression algorithms.
- Unsupervised learning: These algorithms are not trained on a labeled dataset. Instead, they learn to identify patterns in the data without any guidance. Some common unsupervised learning algorithms include clustering and dimensionality reduction algorithms.
- Reinforcement learning: This is a type of machine learning that enables an agent to learn in an interactive environment by trial and error. The agent interacts with its environment by taking actions and receiving rewards or penalties. The agent's goal is to maximize its cumulative reward over time.

Deep Learning (DL)

DL is a technique for implementing ML algorithms that uses artificial neural networks (ANNs). ANNs are inspired by the structure of the human brain, and they are able to learn complex patterns in data. DL has been particularly successful in the field of image recognition, where it has been used to develop algorithms that can identify objects in images with high accuracy.

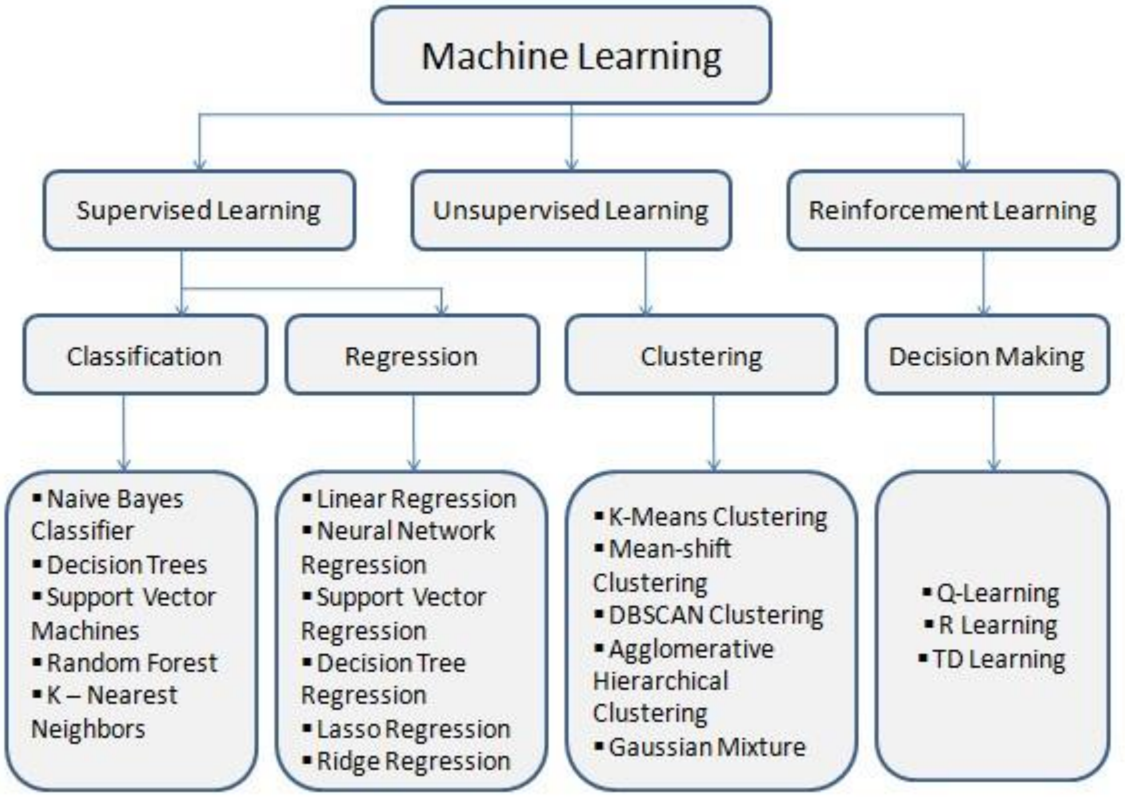


Figure 2.0. 2Machine learning

CLASSIFICATION

It is a fundamental supervised learning technique that involves assigning data points to predefined categories or classes. It is a crucial task in various domains, including machine vision, natural language processing, and medical diagnosis. The primary goal of classification is to develop a model that can accurately predict the class label of an unseen data point based on its features.

Supervised learning algorithms rely on labeled training data to learn the mapping between input features and output labels. This labeled data consists of examples where each data point is associated with its corresponding class label. During the training phase, the algorithm analyzes the training data to identify patterns and relationships between the features and the labels. This process enables the algorithm to learn a decision boundary that separates the data points into distinct classes [10].

Once trained, the classification model can be applied to unlabeled data, predicting the class label for each new data point. The accuracy of the model's predictions is evaluated using a separate set of labeled data, known as the test set. The test set provides an unbiased assessment of the model's performance on unseen data [11].

Several classification algorithms have been developed, each with its strengths and weaknesses. Some of the most widely used classification algorithms include:

- Logistic regression: A statistical model that predicts the probability of a binary outcome (e.g., spam or not spam) based on a set of input features.
- Support vector machines (SVMs): A powerful algorithm that finds the optimal hyperplane that separates data points into distinct classes.
- Decision trees: A tree-like structure that represents a series of decisions, leading to a final class label.
- K-nearest neighbors (KNN): An algorithm that classifies a data point based on the majority class of its k nearest neighbors in the training data.
- Random forests: An ensemble method that combines multiple decision trees to improve classification accuracy.

Applications of Classification

Classification algorithms have a wide range of applications across various domains, including:

- Image recognition: Classifying images into predefined categories, such as animals, vehicles, or scenes.
- Natural language processing (NLP): Classifying text documents into categories, such as spam, sentiment, or topic.
- Medical diagnosis: Identifying diseases or predicting patient outcomes based on medical data.
- Fraud detection: Identifying fraudulent transactions or activities based on financial data.
- Recommendation systems: Recommending products, movies, or other items to users based on their past preferences.

REGRESSION

Regression is a fundamental supervised learning technique that involves predicting a continuous numerical output value based on a set of input features [1]. It is a crucial task in various domains, including finance, economics, and engineering. The primary goal of regression is to develop a model that can accurately estimate the target value for a given set of input features.

Supervised learning algorithms rely on labeled training data to learn the mapping between input features and output values. This labeled data consists of examples where each data point is associated with its corresponding target value. During the training phase, the algorithm analyzes the training data to identify patterns and relationships between the features and the target values. This process enables the algorithm to learn a function that approximates the relationship between the features and the target values [10].

Once trained, the regression model can be applied to unlabeled data, predicting the target value for each new data point. The accuracy of the model's predictions is evaluated using a separate set of labeled data, known as the test set. The test set provides an unbiased assessment of the model's performance on unseen data [11].

Common Regression Algorithms

- Linear regression: A statistical model that assumes a linear relationship between the input features and the target value.
- Polynomial regression: An extension of linear regression that allows for nonlinear relationships by including higher-order terms of the input features.
- Ridge regression: A regularized linear regression model that reduces the impact of overfitting by penalizing large weights.
- Lasso regression: Another regularized linear regression model that performs both variable selection and regularization.
- Support vector regression (SVR): A non-linear regression algorithm that finds the optimal hyperplane that minimizes the prediction error.
- Random forests: An ensemble method that combines multiple decision trees to improve regression accuracy.

Applications of Regression

- Predicting house prices: Estimating the value of a house based on factors such as size, location, and amenities.
- Forecasting sales: Predicting future sales figures based on historical data and market trends.
- Modeling customer behavior: Understanding customer preferences and purchasing patterns.
- Risk assessment: Evaluating the financial risk of investments or loans.
- Scientific research: Modeling and predicting complex relationships in scientific phenomena.

CLUSTERING

Clustering is a fundamental unsupervised learning technique that involves grouping data points into clusters based on their similarity [12]. It is a crucial task in various domains, including machine learning, data mining, and image analysis. The primary goal of clustering is to identify patterns and relationships in unlabeled data without any prior knowledge of the underlying structure.

Unsupervised learning algorithms do not rely on labeled training data. Instead, they explore the unlabeled data to discover patterns and groupings based on inherent similarities between data points. This process enables the algorithm to identify clusters that represent distinct groups or categories within the data [13].

Common Clustering Algorithms

- K-means clustering: A partitioning algorithm that iteratively assigns data points to clusters based on their distance to cluster centroids.
- Hierarchical clustering: A hierarchical algorithm that builds a hierarchy of clusters, merging or splitting clusters based on their similarity.
- Density-based spatial clustering of applications with noise (DBSCAN): A noise-tolerant algorithm that identifies clusters based on the density of data points.
- Self-organizing maps (SOMs): A neural network-based algorithm that projects high-dimensional data onto a lower-dimensional representation, revealing clusters in the data.
- Gaussian mixture models (GMMs): A probabilistic algorithm that assumes data points are generated from a mixture of Gaussian distributions, corresponding to different clusters.

Applications of Clustering

- Customer segmentation: Identifying groups of customers with similar characteristics for targeted marketing campaigns.
- Image segmentation: Grouping pixels in an image based on their color, texture, or intensity, identifying objects or regions of interest.
- Anomaly detection: Identifying data points that deviate significantly from the rest of the data, potentially representing anomalies or outliers.
- Document clustering: Grouping documents with similar content for information retrieval and organization.
- Gene expression analysis: Identifying groups of genes with similar expression patterns for understanding biological processes.

REINFORCEMENT

Reinforcement learning (RL) is a subfield of machine learning that focuses on training agents to make optimal decisions in an environment by interacting with it and learning from the consequences of their actions [14]. Unlike supervised learning, which relies on labeled data, RL agents learn through trial and error, receiving rewards or penalties for their actions. This allows them to adapt to new situations and continuously improve their performance [15].

Key Concepts in RL:

- Agent: The decision-maker or learner in the RL system.
- Environment: The surroundings in which the agent interacts.
- State: The current situation or observation the agent receives from the environment.
- Action: The choice the agent makes in response to the state.

- Reward: A signal indicating the desirability of an action, positive for good actions and negative for bad ones.
- Policy: The agent's strategy for selecting actions in different states.
- Value function: The expected future reward an agent can get from a state or state-action pair.

RL Process:

- The agent perceives the state of the environment.
- Based on its policy, the agent selects an action.
- The agent takes the action and receives a reward or penalty from the environment.
- The agent updates its policy based on the reward, state, and action.

Types of RL Algorithms:

1. Value-based: These algorithms learn the value of taking a particular action in a given state, such as Q-learning and Deep Q-learning.
2. Policy-based: These algorithms directly learn a policy that maps states to actions, such as Policy Gradient and Deep Deterministic Policy Gradient (DDPG).

Applications of RL:

- Robotics: Training robots to perform tasks like walking, running, and manipulating objects [15].
- Game playing: Developing agents that can master complex games like chess, Go, and Atari games.
- Resource management: Optimizing the use of resources like energy, bandwidth, and water.
- Recommendation systems: Recommending products, movies, and other items to users based on their preferences.
- Financial trading: Making investment decisions and managing portfolios.

Advantages of RL:

1. Adaptability: RL agents can learn from their mistakes and adapt to new situations.
2. Data efficiency: RL requires less labeled data compared to supervised learning.
3. Flexibility: RL can be applied to a wide range of problems.

Disadvantages of RL:

1. Computational complexity: Training RL agents can be computationally expensive.
2. Sample inefficiency: RL agents can take a long time to learn in complex environments.
3. Exploration-exploitation tradeoff: Balancing exploration of new actions and exploitation of known good actions.

DEEP LEARNING

Deep learning is a subfield of machine learning that uses artificial neural networks (ANNs) with multiple layers to learn complex patterns from data. It has achieved remarkable results in various fields including image recognition, natural language processing, and speech recognition [1].

Working of Neural network

A neural network is typically characterized by its various layers, with the initial layer known as the input layer. This layer receives and transmits input signals to the subsequent layer. The subsequent layer, referred to as the hidden layer, engages in diverse computations and feature extractions, often comprising multiple hidden layers. The final layer, termed the output layer, produces the ultimate outcome.

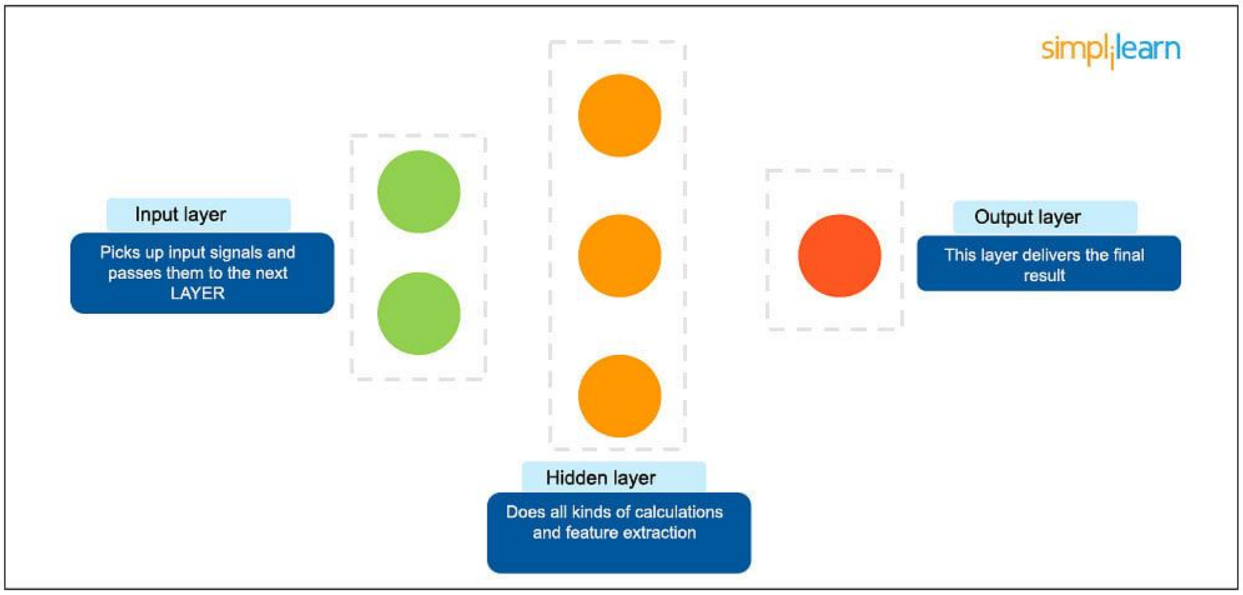

simplilearn
Input layer
Picks up input signals and passes them to the next LAYER
Hidden layer
Does all kinds of calculations and feature extraction
Output layer
This layer delivers the final result

Figure 2.0. 3 Layers in Neural networks

Consider the practical scenario of how traffic cameras discern license plates and identify speeding vehicles on the road. The image, sized at 28 by 28 pixels, is utilized as input for license plate identification. Each neuron in the network possesses an activation number, denoting the grayscale value of the corresponding pixel, ranging from 0 to 1—where 1 indicates a white pixel and 0 signifies a black pixel. Neurons activate when their activation values are proximate to 1.

Pixel data, organized as arrays, is introduced into the input layer. For images exceeding 28 by 28 pixels, resizing is necessary since the input layer's size is fixed. In this instance, the inputs are designated as X1, X2, and X3, each representing an incoming pixel. The input layer subsequently transfers these inputs to the hidden layer. The interconnections between neurons are endowed with random weights. These weights are multiplied by the input signals, and a bias is incorporated into the calculation.

The initial values of weights in a CNN vary depending on the specific initialization technique used. There are several common approaches, each with its own advantages and disadvantages:

1. Xavier Initialization:

This is a popular choice for CNNs, particularly for ReLU activation functions.

It aims to initialize weights in a way that preserves the variance of the signal across layers during backpropagation.

Mathematically, it scales the weights based on the fan-in (number of incoming connections) of a neuron.

This helps prevent exploding or vanishing gradients, which can hinder training.

2. He Initialization (Kaiming Normalization):

This is another popular choice, especially for ReLU activation functions.

It focuses on initializing weights to promote a zero-mean distribution with a specific standard deviation.

This ensures that both positive and negative activations are equally likely initially, preventing the dominance of one side and aiding gradient flow.

Compared to Xavier, He initialization might be slightly more prone to vanishing gradients for deeper networks.

3. Random Initialization:

This is a simple approach where weights are randomly drawn from a uniform or normal distribution within a predefined range.

It's less theoretically motivated than Xavier or He but can be surprisingly effective for small networks or specific tasks.

However, it can lead to exploding or vanishing gradients in deeper networks, hindering training.

4. Zero Initialization:

This initializes all weights to zero, which might seem intuitive but is generally not recommended for CNNs.

It can lead to vanishing gradients because neurons receive no initial signal, making it difficult for them to learn.

It might be used in specific cases like initializing batch normalization layers, but not for the entire network.

5. Pre-trained Weights:

This involves using weights learned from a pre-trained model like VGG16 or ResNet50.

This is a powerful approach for transfer learning, where you leverage the pre-trained features for your specific task.

The pre-trained weights can be fine-tuned with your data, often requiring fewer training iterations and achieving better performance.

- The choice of initial weight values depends on several factors, including:
- Network architecture: Deeper networks might benefit from He initialization to prevent vanishing gradients, while shallow networks might be fine with Xavier or even random initialization.
- Activation function: Xavier and He are specifically designed for ReLU activation, while other choices might require different initialization techniques.
- Computational resources: Pre-trained weights can save training time and improve performance, but come at the cost of download time and potentially larger model sizes.

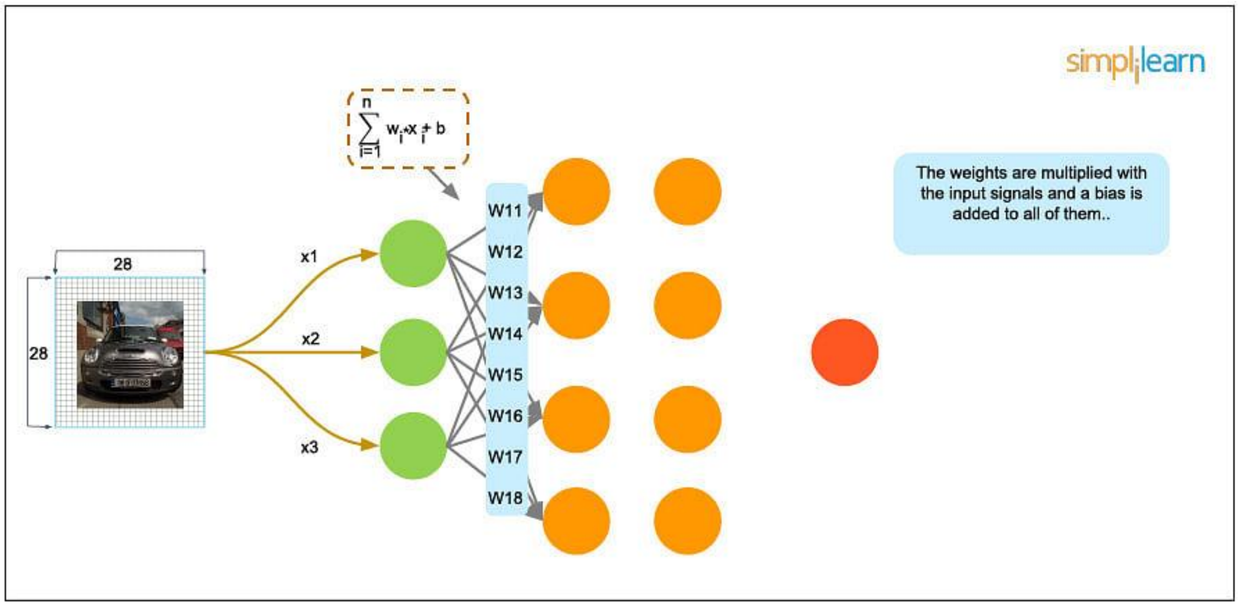

simplilearn
$\sum_{i=1}^{n} w_i x_i + b$
The weights are multiplied with the input signals and a bias is added to all of them..
28
28
x1
x2
x3
W11
W12
W13
W14
W15
W16
W17
W18

Figure 2.0. 4Weights in Neural networks

The resultant weighted sum of inputs is then presented to the activation function, determining the nodes to activate for feature extraction. As the signal progresses through the hidden layers, the weighted sum of inputs is computed and forwarded to the activation function in each layer, dictating the nodes to activate.

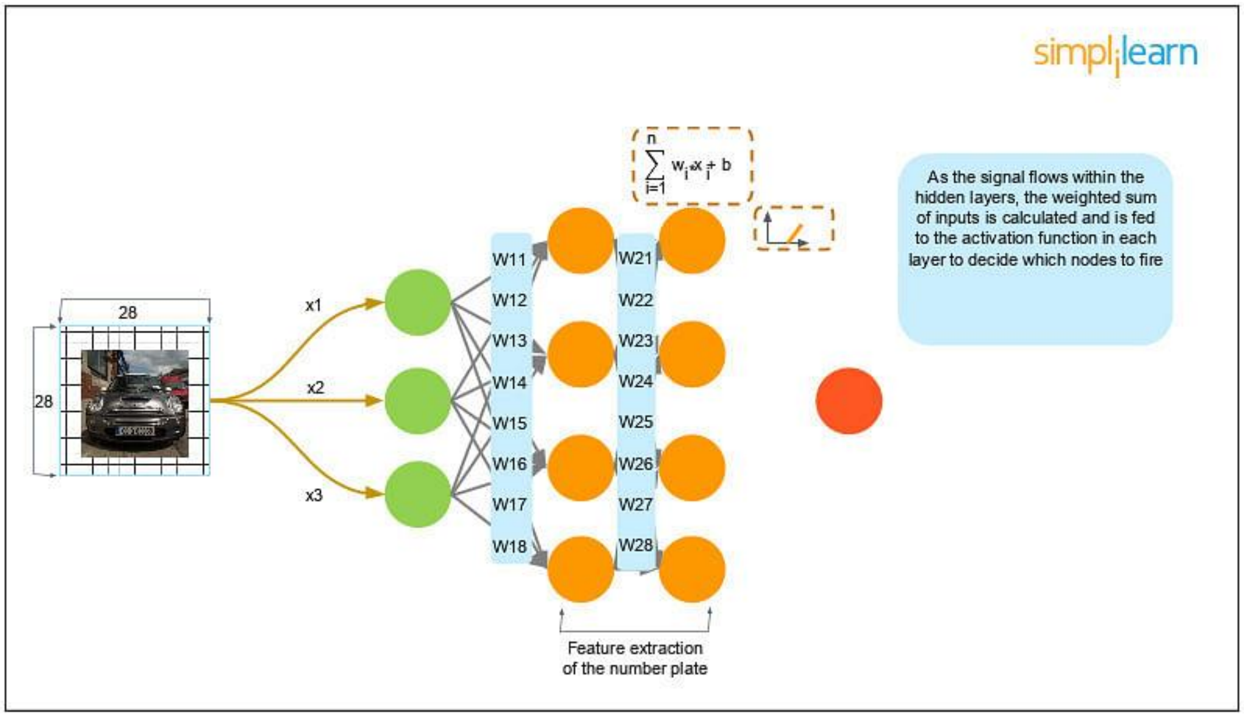

simplilearn
$\sum_{i=1}^{n} w_i x_i + b$
As the signal flows within the hidden layers, the weighted sum of inputs is calculated and is fed to the activation function in each layer to decide which nodes to fire
28
28
x1
x2
x3
W11
W12
W13
W14
W15
W16
W17
W18
W21
W22
W23
W24
W25
W26
W27
W28
Feature extraction of the number plate

Figure 2.0. 5 Activation functions in Neural networks

Activation functions

1. Sigmoid Function
   The application of the sigmoid function occurs in cases where the model is making predictions related to probability.

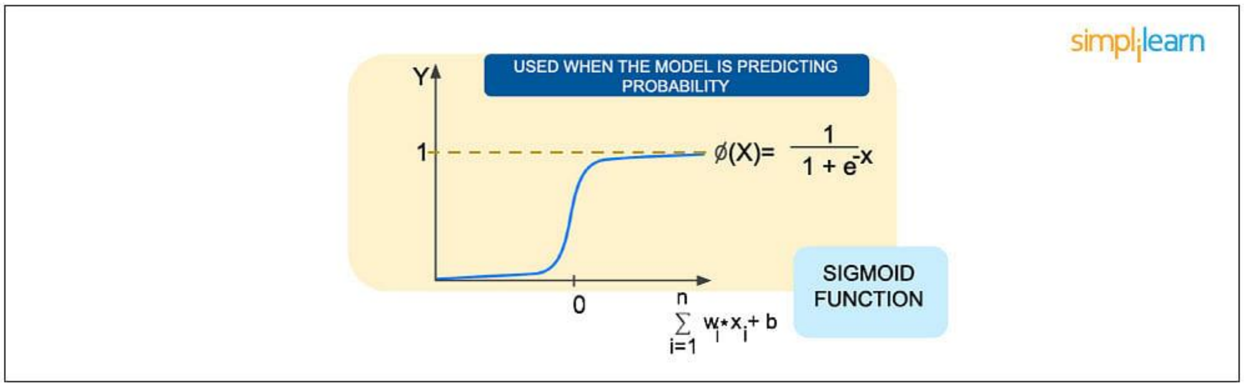


Figure 2.0. 6 Sigmoid function

2. Threshold function
   The threshold function becomes valuable in situations where concerns about uncertainty in the middle are not desired.

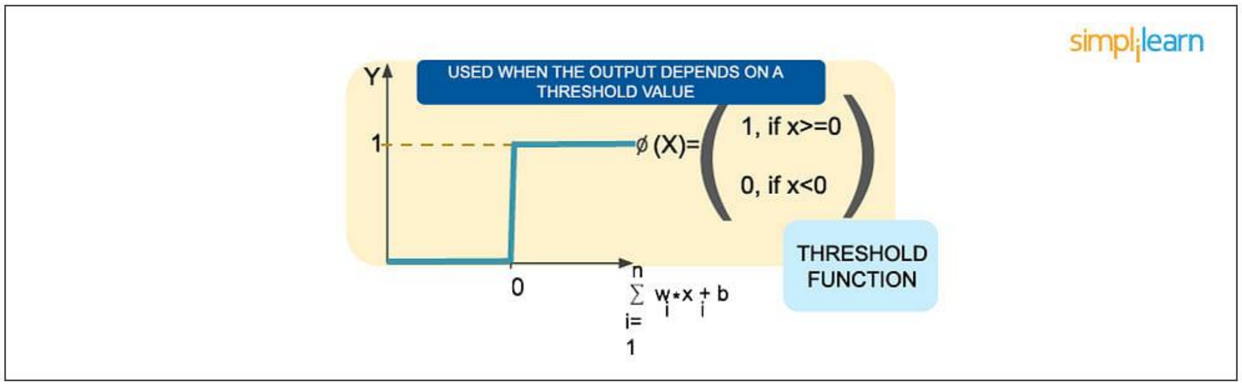

simplilearn
USED WHEN THE OUTPUT DEPENDS ON A THRESHOLD VALUE
Y
1
ø (X)=
1, if x>=0
0, if x<0
0
$\sum_{i=1}^{n} w_i * x_i + b$
THRESHOLD FUNCTION

Figure 2.0. 7 Threshold function

3.ReLU (rectified linear unit) Function

The ReLU (rectified linear unit) function gives the value but says if it's over 1, then it will just be 1, and if it's less than 0, it will just be 0. The ReLU function is most commonly used these days.

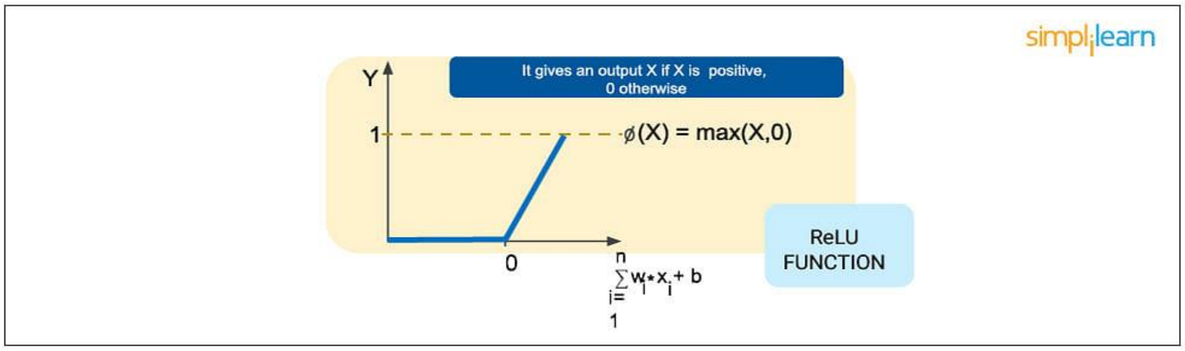

simplilearn
It gives an output X if X is positive, 0 otherwise
Y
1
ø(X) = max(X,0)
0
$\sum_{i=1}^{n} w_i * x_i + b$
ReLU FUNCTION

Figure 2.0. 8 ReLU Function

4. Hyperbolic Tangent Function

   The hyperbolic tangent function is similar to the sigmoid function but has a range of -1 to 1.

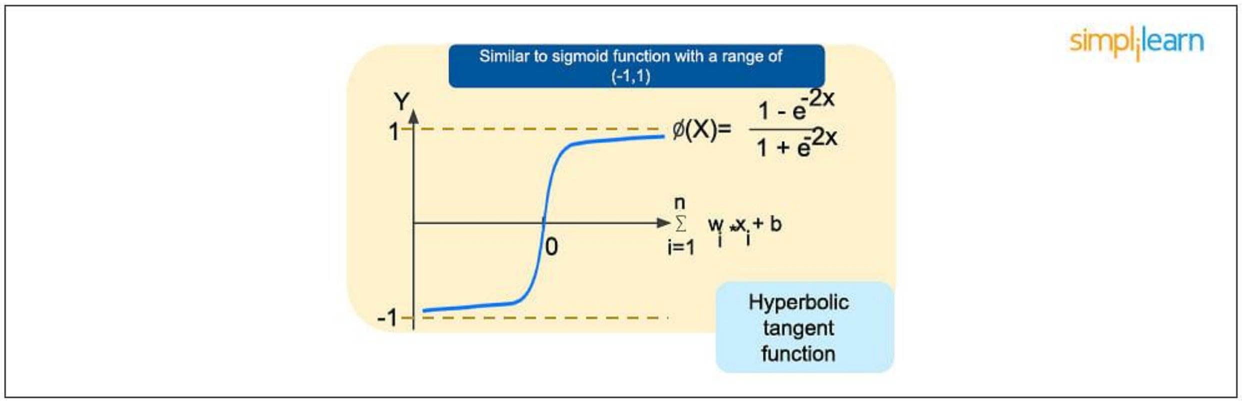


Figure 2.0. 9 Hyperbolic Tangent Function

The correction of output errors involves back-propagating the error through the network, and adjustments to the weights are made to minimize the error rate. This process is guided by a cost function that quantifies the error. The weights are iteratively adjusted until they align with the various training models introduced.

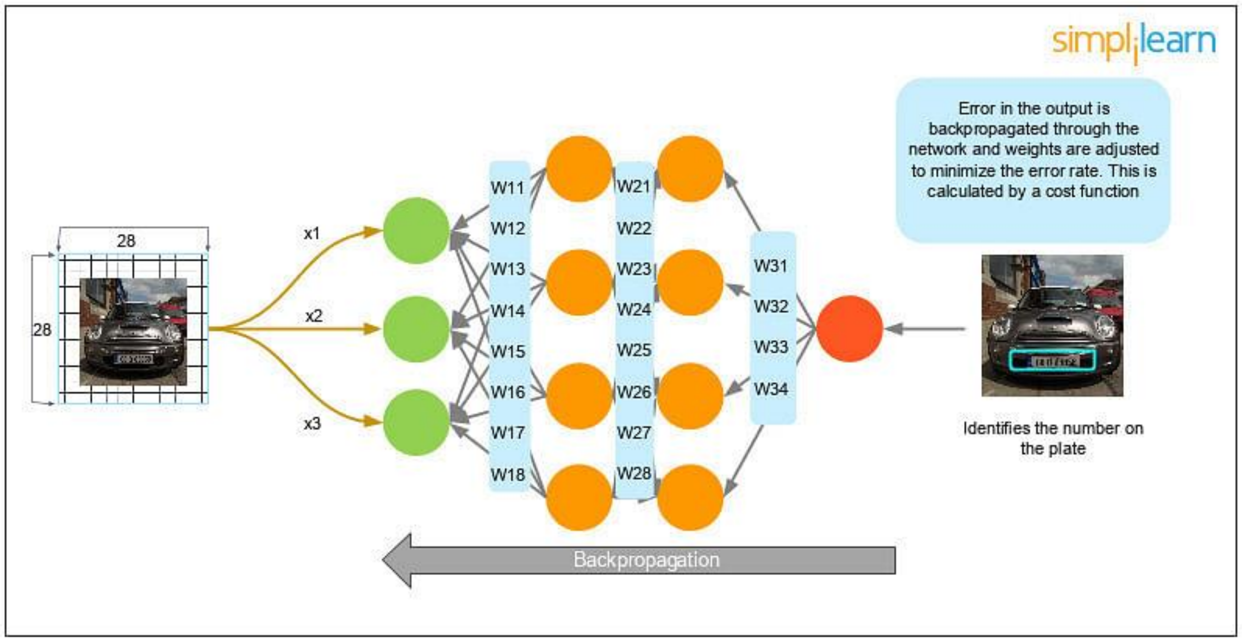

simplilearn
Error in the output is backpropagated through the network and weights are adjusted to minimize the error rate. This is calculated by a cost function
28
28
x1
x2
x3
W11
W12
W13
W14
W15
W16
W17
W18
W21
W22
W23
W24
W25
W26
W27
W28
W31
W32
W33
W34
Identifies the number on the plate
Backpropagation

Figure 2.0. 10 Backpropagation

Subsequently, the generated output is compared to the original result, and multiple iterations ensue to achieve maximum accuracy. During each iteration, the weights at every connection are fine-tuned based on the observed error. While the mathematical intricacies of this adjustment process are complex and won't be delved into here, our focus lies in understanding its implementation through code for our specific use case.

Types of Neural Networks

1. Feed-forward Neural Network:
   This represents the simplest form of artificial neural networks (ANNs) where data progresses unidirectionally from input to output. Widely utilized in applications like vision and speech recognition, it exhibits swift operational performance during use, albeit requiring considerable time for training.
2. Radial Basis Functions Neural Network:
   This neural network model classifies data points based on their distance from a central point. When lacking training data, grouping and center point creation are employed. Noteworthy applications include power restoration systems, where the network identifies and groups similar data points.
3. Kohonen Self-organizing Neural Network:
   Random input vectors are fed into a discrete map comprising neurons, also referred to as vectors, dimensions, or planes. Applied in scenarios such as medical analysis, it excels at recognizing patterns in data by identifying similarities among input vectors.
4. Recurrent Neural Network:
   In this variant, the hidden layer retains its output for subsequent predictions, incorporating the output into its new input. Notable applications encompass text-to-speech conversion, where the network's ability to preserve output aids in generating coherent predictions.
5. Convolution Neural Network:
   This neural network processes input features in batches, akin to passing them through a filter. This methodology enables the network to remember and analyze images in segments. Applications range from signal processing to facial recognition in image processing tasks.
6. Modular Neural Network:
   Comprising a collection of distinct neural networks collaborating to produce the final output, this cutting-edge approach is still in the research phase, demonstrating the evolving nature of neural network architectures [16].

IDENTIFYING PNEUMONIA IN X-RAYS

Radiologists use a combination of factors to determine whether an X-ray image shows the presence of pneumonia [17]:

Visual cues:

- Infiltrates: These are areas of increased opacity in the lung parenchyma, indicating inflammation and fluid accumulation. They can be patchy or confluent, depending on the severity of the pneumonia.
- Consolidation: This refers to a larger area of dense opacity, often involving one or more lobes of the lung. It signifies a more advanced stage of the disease.
- Air bronchograms: These are air-filled bronchi that appear as white lines within the consolidated areas. They are a hallmark feature of bacterial pneumonia.
- Pleural effusion: This is fluid accumulation in the space between the lung and the chest wall, causing blunting of the costophrenic angle and a rounded appearance of the silhouette.
- Loss of lung markings: Normally, blood vessels and other structures in the lung create a visible pattern on the X-ray. In pneumonia, this pattern can be obscured by inflammation and fluid.

Additional factors:

- Clinical context: The radiologist considers the patient's symptoms, history, and other medical information. For example, cough, fever, and shortness of breath are suggestive of pneumonia.
- Distribution of findings: The location and distribution of infiltrates and consolidation can offer clues about the type of pneumonia. Bacterial pneumonia often involves segments or lobes, while viral pneumonia can be more patchy.
- Progression: Comparing X-rays taken at different times can be helpful in diagnosing pneumonia. Worsening of findings or development of new features supports the diagnosis.

2.1 TECHNOLOGIES

Deep learning and artificial intelligence (AI) are rapidly evolving domains that constantly introduce new technologies. Among the most promising emerging trends are federated learning, generative adversarial networks (GANs), explainable AI (XAI), reinforcement learning, and transfer learning [18].

2.1.1. Federated learning

Federated learning is a machine learning approach that enables collaboration among multiple devices without sharing data with a central server. This technique addresses concerns regarding data privacy. A notable application of federated learning is seen in Google's predictive text keyboard, which improved accuracy without compromising user privacy. Unlike traditional models that rely on centralized data sources, federated learning trains models directly on user devices, eliminating the need to transfer data to a central server. This approach mitigates privacy concerns and reduces computational and storage requirements by keeping data local.

2.1.2.Generative adversarial networks (GANs)

Generative adversarial networks (GANs) are a specific class of neural networks renowned for creating novel and authentic data by leveraging existing datasets. GANs have gained prominence in diverse domains, including image generation, where they have been employed to produce lifelike depictions of individuals, animals, and even natural landscapes. The fundamental principle behind GANs involves the interplay between two neural networks: a generator network and a discriminator network. The generator network is responsible for fabricating synthetic data, while the discriminator network aims to discriminate between genuine and fabricated data. This adversarial process encourages the generator network to continuously refine its output until it becomes indistinguishable from real data, while the discriminator network strengthens its ability to differentiate between real and fake instances. This adversarial relationship fosters a dynamic and iterative learning process, ultimately producing high-quality synthetic data by GANs.

2.1.3. Explainable AI (XAI)

Explainable AI (XAI) is an approach aimed at enhancing the transparency and interpretability of machine learning models, ensuring unbiased and fair decision-making. Its importance lies in providing insights into the reasoning behind AI systems. Here is an example illustrating the application of XAI:

Let's consider a financial institution utilizing machine learning algorithms to predict the probability of loan default for loan applicants. In the case of traditional black-box algorithms, the bank may lack visibility into the decision-making process, making it challenging to explain to the loan applicant.

However, with the implementation of XAI, the algorithm can offer an explanation for its decision. This empowers the bank to validate that the decision was based on rational factors rather than erroneous or discriminatory information. For instance, the algorithm might reveal that it derived a risk score by considering the applicant's credit score, income, and employment history. This level of transparency and explainability fosters trust in AI systems, enhances accountability, and facilitates better-informed decision-making.

2.1.4. Reinforcement learning

Reinforcement learning is a specific branch of machine learning that focuses on training agents through criticism and incentives. Various domains, from robotics to gaming and even banking, have successfully applied this technique. One notable example is DeepMind's AlphaGo, which utilized reinforcement learning to enhance its gameplay continuously and ultimately surpass leading human Go players. This accomplishment serves as a remarkable illustration of the effectiveness of reinforcement learning when confronted with intricate decision-making challenges.

2.1.5. Transfer learning

Transfer learning is a valuable machine learning strategy crucial in addressing novel problems by leveraging previously trained models. This approach proves particularly useful when limited data is available for a new task.

An illustrative example of transfer learning is demonstrated in adapting image recognition models. Researchers have successfully applied transfer learning by taking models initially trained on a specific type of image, such as facial recognition, and adapting them to identify different types of images, such as animals.

By utilizing transfer learning, the knowledge gained from pre-trained models, including learned features, weights, and biases, can be effectively reused in the new task. This reuse of information significantly enhances the performance of the model while simultaneously reducing the amount of data required for training. It allows the model to leverage the prior knowledge and generalization capabilities acquired from the initial task to improve its performance in the new problem domain.

CHAPTER THREE

3.0 METHODOLOGY

The model was created using CNNs. They are generally considered the superior choice for image classification tasks compared to other machine learning algorithms like support vector machines (SVMs) and linear regression due to their ability to extract and learn complex patterns from images. Here's a detailed explanation of why CNNs are better suited for image classification:

1. Feature Extraction:
   CNNs are designed to automatically extract relevant features from images, eliminating the need for manual feature engineering.
   They employ convolutional layers that apply filters to the image data, capturing local patterns and progressively extracting higher-level features.
   This hierarchical feature extraction process allows CNNs to learn complex and meaningful representations of images, which is crucial for accurate classification.
2. Spatial Relationships:
   CNNs inherently preserve the spatial relationships between pixels in images, which is essential for understanding the context and content of an image.
   They utilize pooling layers to reduce the spatial dimension while retaining key features, allowing them to handle images of varying sizes.
   This ability to capture spatial information is crucial for tasks like object detection, scene understanding, and image segmentation.
3. Non-linearity:
   CNNs incorporate non-linear activation functions, such as ReLU, to capture non-linear relationships in the data.
   This allows them to model complex patterns that go beyond linear relationships, which is often the case in image data.
   Linear regression, on the other hand, assumes a linear relationship between the input and output, which is not suitable for capturing the intricate patterns in images.
4. Robustness to Noise:
   CNNs are relatively robust to noise and distortions in images due to their inherent feature extraction capabilities.
   They can learn to extract meaningful features even in the presence of noise, making them suitable for real-world applications where images may not be perfectly clean.
   SVMs, on the other hand, can be more sensitive to noise and may struggle with noisy image data.
5. Scalability:
   CNNs can handle large image datasets effectively due to their parallel architecture and efficient memory usage.
   They can be trained on millions of images without significant performance degradation, making them suitable for large-scale image classification tasks.
   Linear regression, on the other hand, may struggle with large datasets due to its computational complexity.

Overall, CNNs have emerged as the dominant approach for image classification due to their ability to automatically extract complex features, preserve spatial relationships, handle non-linear patterns, tolerate noise, and scale to large datasets. These advantages make them well-suited for a wide range of image-based applications.

3.1 BLOCK DIAGRAM

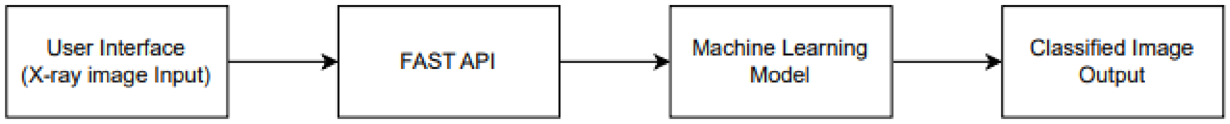


Figure 3.1 1 Block Diagram

The website's intuitive interface allowed users to upload images effortlessly. Upon upload, the image was seamlessly processed at the backend, powered by the FastAPI web framework. This backend processing involved resizing the image, normalizing its pixel values, and converting it into a NumPy array. The prepared NumPy array was then fed into the trained machine learning model, which analyzed its features and, based on its learned weights, generated a classification.

3.2 FLOWCHART

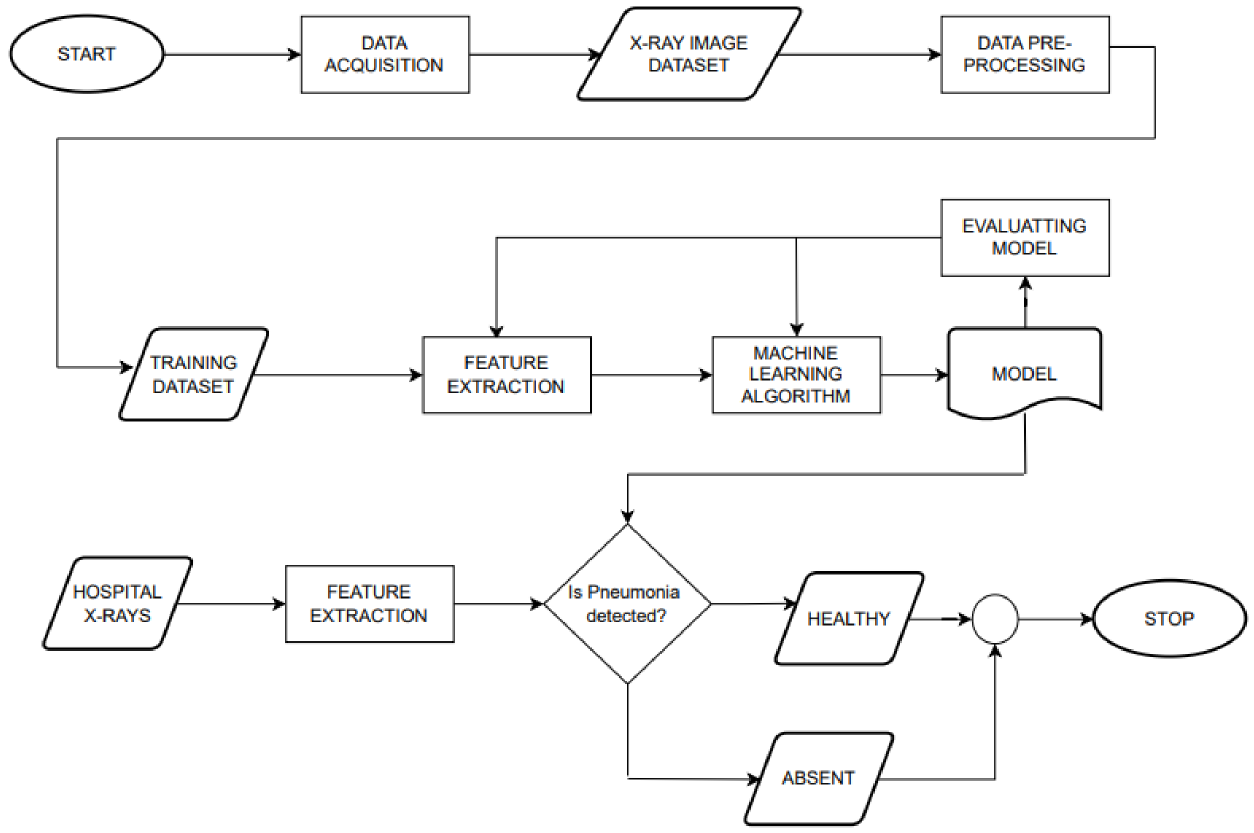


Figure 3.2. 1 Project Flowchart

<u>STEPS</u>

Data Acquisition:

Images were obtained from Kaggle website:

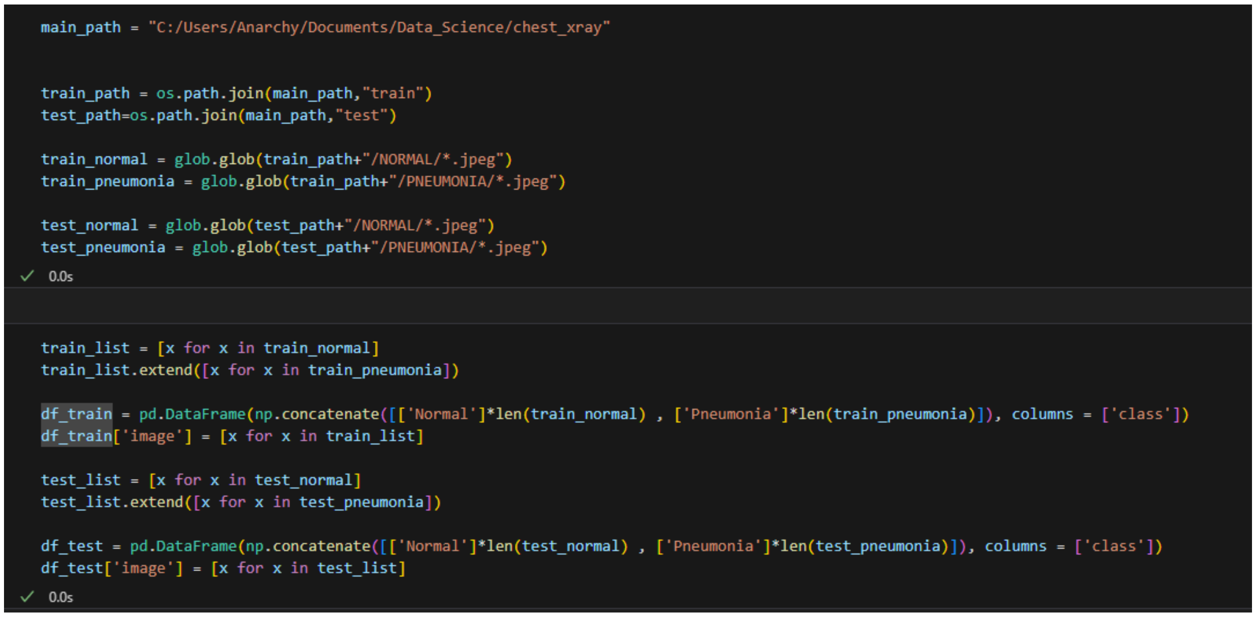
https://www.kaggle.com/code/jonaspalucibarbosa/chest-x-ray-pneumonia-cnn-transfer-learning/input

The images were downloaded and stored in the following folder path:

"C:\Users\Anarchy\Documents\Data_Science"

Data Preprocessing:

The images came in different folders labelled “Train” and “Test” each having subfolders labelled “NORMAL” and “PNEUMONIA”. In order to have both classes of images during the training process, the data was concatenated. Merging of the classes for training resulted in having 5216 images for training while having 624 images for testing.



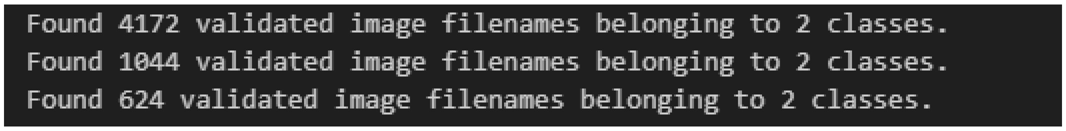
```
main_path = "C:/Users/Anarchy/Documents/Data_Science/chest_xray"


train_path = os.path.join(main_path,"train")
test_path=os.path.join(main_path,"test")

train_normal = glob.glob(train_path+"/NORMAL/*.jpeg")
train_pneumonia = glob.glob(train_path+"/PNEUMONIA/*.jpeg")

test_normal = glob.glob(test_path+"/NORMAL/*.jpeg")
test_pneumonia = glob.glob(test_path+"/PNEUMONIA/*.jpeg")
✓ 0.0s

train_list = [x for x in train_normal]
train_list.extend([x for x in train_pneumonia])

df_train = pd.DataFrame(np.concatenate([['Normal']*len(train_normal) , ['Pneumonia']*len(train_pneumonia)]), columns = ['class'])
df_train['image'] = [x for x in train_list]

test_list = [x for x in test_normal]
test_list.extend([x for x in test_pneumonia])

df_test = pd.DataFrame(np.concatenate([['Normal']*len(test_normal) , ['Pneumonia']*len(test_pneumonia)]), columns = ['class'])
df_test['image'] = [x for x in test_list]
✓ 0.0s
```

Figure 3.2. 2 Concatenation

The images came in various sizes and therefore resizing them to 224*224 was done. This is because CNN architecture require a consistent input size which is essential for weight sharing in the convolutional layers.

The train dataset was further split into 80% training and 20% validation.

The final combined dataset was as follows:



```
Found 4172 validated image filenames belonging to 2 classes.
Found 1044 validated image filenames belonging to 2 classes.
Found 624 validated image filenames belonging to 2 classes.
```

Figure 3.2. 3 Dataset split

This split is backed up by data scientists as a general rule of thumb to give your model enough data for training for sufficient learning and then enough data for testing.

The images were converted to NumPy array to be fed to the CNN.

Feature Extraction:

This process involved adding hidden layers to the CNN in order to extract high-level features from the images.

Low-level features are basic visual cues that can be directly observed from the X-ray image. These features include:

- Infiltrates: These are patchy or confluent areas of increased opacity in the lung parenchyma, indicating inflammation and fluid accumulation.
- Consolidation: This refers to a larger area of dense opacity, often involving one or more lobes of the lung.
- Air bronchograms: These are air-filled bronchi that appear as white lines within the consolidated areas.
- Pleural effusion: This is fluid accumulation in the space between the lung and the chest wall, causing blunting of the costophrenic angle and a rounded appearance of the silhouette. Normally, the costophrenic angle, where the diaphragm meets the ribs, appears sharp and well-defined on an X-ray. However, when fluid accumulates in the pleural space, it pushes the diaphragm upward, causing the angle to appear blunted or rounded. Additionally, the overall silhouette of the lung becomes more rounded due to the presence of fluid in the pleural space.
- Loss of lung markings: Normally, blood vessels and other structures in the lung create a visible pattern on the X-ray. In pneumonia, this pattern can be obscured by inflammation and fluid.

High-level features are more complex and abstract patterns that emerge from the low-level features. These features are often related to the distribution, shape, and texture of the abnormalities. Radiologists consider these features to understand the underlying pathology and assess the severity of pneumonia.

- Distribution of infiltrates: The distribution of infiltrates can provide clues about the type of pneumonia. For example, bacterial pneumonia often involves segments or lobes, while viral pneumonia can be more patchy. The term "patchy" refers to the uneven or scattered distribution of infiltrates in the lung parenchyma. Instead of forming a continuous or confluent area of opacity, patchy infiltrates appear as scattered patches or streaks of increased density, often with areas of normal lung parenchyma interspersed between them. This pattern is often associated with viral pneumonia, where the infection causes scattered areas of inflammation rather than a more localized consolidation.

- Shape of infiltrates: Irregularly shaped infiltrates can suggest a more severe form of pneumonia compared to well-defined, rounded infiltrates.
- Texture of infiltrates: Fine-grained infiltrates are typically associated with early pneumonia, while coarse-grained infiltrates may indicate a more advanced stage.
- Presence of air bronchograms: Air bronchograms are a hallmark feature of bacterial pneumonia and can be used to differentiate it from other conditions.
- Extent of pleural effusion: The extent of pleural effusion can indicate the severity of pneumonia and may suggest complications like empyema. Empyema is a serious medical condition characterized by the accumulation of pus in the pleural space, the cavity between the lung and the chest wall. This pus, a thick, yellow fluid filled with dead white blood cells, bacteria, and cellular debris,

Machine Learning Model:

Developed a machine learning model using the extracted features. The model is trained on the training dataset, learning to map input X-ray images to their corresponding labels (normal or pneumonia) based on the extracted features.

Case 1: Training using CPU

```
def get_model():

    #Input shape = [width, height, color channels]
    inputs = layers.Input(shape=(IMG_SIZE, IMG_SIZE, 3))

    # Block One
    x = layers.Conv2D(filters=16, kernel_size=3, padding='valid')(inputs)
    x = layers.BatchNormalization()(x)
    x = layers.Activation('relu')(x)
    x = layers.MaxPool2D()(x)
    x = layers.Dropout(0.2)(x)

    # Block Two
    x = layers.Conv2D(filters=32, kernel_size=3, padding='valid')(x)
    x = layers.BatchNormalization()(x)
    x = layers.Activation('relu')(x)
    x = layers.MaxPool2D()(x)
    x = layers.Dropout(0.2)(x)

    # Block Three
    x = layers.Conv2D(filters=64, kernel_size=3, padding='valid')(x)
    x = layers.Conv2D(filters=64, kernel_size=3, padding='valid')(x)
    x = layers.BatchNormalization()(x)
    x = layers.Activation('relu')(x)
    x = layers.MaxPool2D()(x)
    x = layers.Dropout(0.4)(x)

    # Head
    #x = layers.BatchNormalization()(x)
    x = layers.Flatten()(x)
    x = layers.Dense(64, activation='relu')(x)
    x = layers.Dropout(0.5)(x)

    #Final Layer (Output)
    output = layers.Dense(1, activation='sigmoid')(x)

    model = keras.Model(inputs=[inputs], outputs=output)

    return model
```

Input

inputs = layers.Input(shape=(IMG_SIZE, IMG_SIZE, 3))

- layers.Input: This refers to the input layer class provided by the Keras library. Keras is used as a high-level neural networks API, and it provides convenient classes and functions for building neural network architectures.
- shape=(IMG_SIZE, IMG_SIZE, 3): This specifies the shape of the input data that the neural network will expect. The IMG_SIZE is a variable defined earlier in your code, and it is set to 224. The (IMG_SIZE, IMG_SIZE, 3) indicates that each input sample is expected to be a 3D tensor with dimensions (IMG_SIZE, IMG_SIZE, 3). Here's what each dimension represents:
- IMG_SIZE: The width and height of the input image. In this case, it's set to 224, suggesting that the model expects input images to be 224 pixels in both width and height.
- 3: This represents the number of color channels in the image. Images with three color channels typically correspond to RGB (Red, Green, Blue) images. Each channel holds information about the intensity of a specific color.

Model architecture

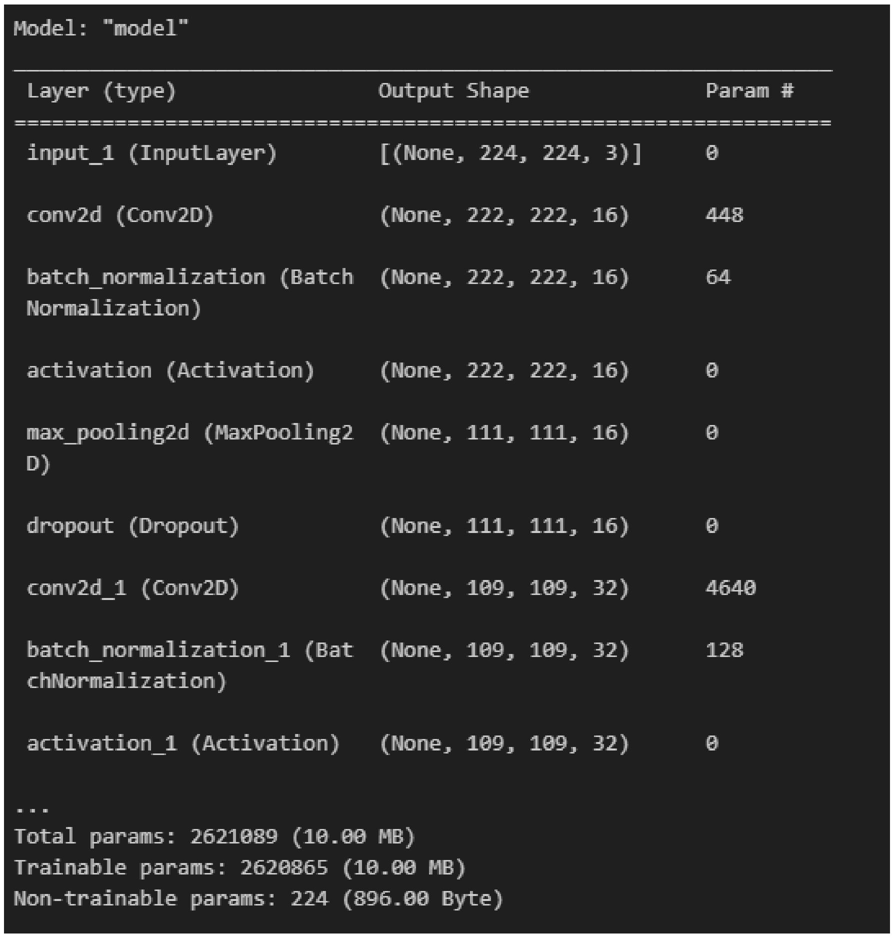

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0

 conv2d (Conv2D)             (None, 222, 222, 16)      448

 batch_normalization (Batch  (None, 222, 222, 16)      64
 Normalization)

 activation (Activation)     (None, 222, 222, 16)      0

 max_pooling2d (MaxPooling2  (None, 111, 111, 16)      0
 D)

 dropout (Dropout)           (None, 111, 111, 16)      0

 conv2d_1 (Conv2D)           (None, 109, 109, 32)      4640

 batch_normalization_1 (Bat  (None, 109, 109, 32)      128
 chNormalization)

 activation_1 (Activation)   (None, 109, 109, 32)      0

...
Total params: 2621089 (10.00 MB)
Trainable params: 2620865 (10.00 MB)
Non-trainable params: 224 (896.00 Byte)
```

Figure 3.2. 4 Model architecture using custom CNN

- Convolutional Layer:
  - Filters (Channels): 16 filters are used. These filters are responsible for learning spatial hierarchies and local patterns in the input image.
  - Kernel Size: The convolutional kernel size is set to 3x3. This means each filter looks at a 3x3 region of the input to learn local patterns.
  - Padding: 'Valid' padding is used, which means no padding is added to the input. This can lead to a reduction in spatial dimensions after convolution.
  - The Conv2D layer in Keras is initialized using the He initialization method by default. This is because the He initialization method is specifically designed for convolutional neural networks (CNNs), and it has been shown to work well in practice.
- Batch Normalization:
  Applied after convolution to normalize the activations, preventing issues like vanishing or exploding gradients during training.

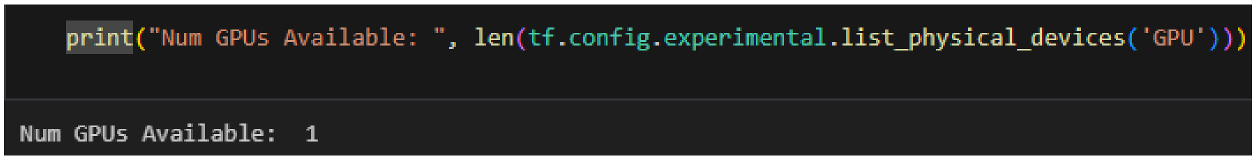
- ReLU Activation:
  Introduces non-linearity to the model by applying the Rectified Linear Unit (ReLU) activation function.
- Max Pooling:
  Reduces spatial dimensions by taking the maximum value in each local region (2x2 in this case). This helps create spatial hierarchies and reduce computational complexity.
- Dropout:
  Helps prevent overfitting by randomly setting a fraction of input units to zero during training.

Model Optimization:

Optimized the model's performance by integrating the learning process with GPU, including transfer learning and also fine-tuning. This step aimed to enhance the model's accuracy and generalization capabilities. The cases are shown below.

Case 2: Training with GPU

The input and model architecture is similar to the previous one (Using CNN with CPU). In this case, the virtual environment was configured to run on GPU (specifically an Nvidia RTX 1660 Ti, using CUDA and cuDNN for accelerated deep learning operations).



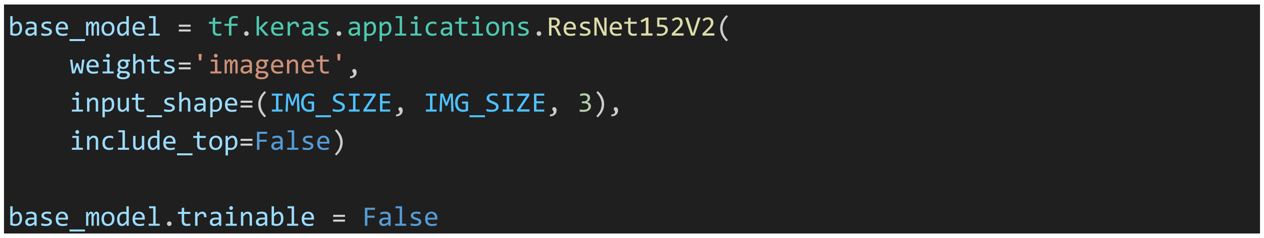
```
    print("Num GPUs Available: ", len(tf.config.experimental.list_physical_devices('GPU')))

Num GPUs Available:  1
```

Figure 3.2. 5 GPU available

The GPU is recognized.

Case 3: Training using custom CNN coupled with transfer learning and GPU

Model architecture



```
base_model = tf.keras.applications.ResNet152V2(
    weights='imagenet',
    input_shape=(IMG_SIZE, IMG_SIZE, 3),
    include_top=False)

base_model.trainable = False
```

```
def get_pretrained():

    #Input shape = [width, height, color channels]
    inputs = layers.Input(shape=(IMG_SIZE, IMG_SIZE, 3))

    x = base_model(inputs)

    # Head
    x = layers.GlobalAveragePooling2D()(x)
    x = layers.Dense(128, activation='relu')(x)
    x = layers.Dropout(0.1)(x)

    #Final Layer (Output)
    output = layers.Dense(1, activation='sigmoid')(x)

    model = keras.Model(inputs=[inputs], outputs=output)

    return model
```

The model architecture consists of a pre-trained ResNet152V2 model as the base model, followed by custom head layers for fine-tuning the model for pneumonia detection.

- ***Pre-trained ResNet152V2 Model:***
  The ResNet152V2 model is a convolutional neural network (CNN) pre-trained on the ImageNet dataset, which consists of over 14 million images.
  The pre-trained weights of the ResNet152V2 model are frozen, meaning that they are not updated during training. This is done to prevent overfitting and to ensure that the model remains generalizable to new data.
- ***Custom Head Layers:***
  The custom head layers consist of a GlobalAveragePooling2D layer, a Dense layer with 128 units and ReLU activation, a Dropout layer with a rate of 0.1, and a final Dense layer with 1 unit and sigmoid activation.
- ***GlobalAveragePooling2D Layer:***
  The GlobalAveragePooling2D layer flattens the feature maps from the ResNet152V2 model. This is done because the Dense layers that follow require a one-dimensional input.
- ***Dense Layer with 128 Units and ReLU Activation:***
  The Dense layer with 128 units and ReLU activation performs further feature extraction. The ReLU activation function adds non-linearity to the model, which is important for learning complex patterns in the data.
- ***Dropout Layer with a Rate of 0.1:***
  The Dropout layer with a rate of 0.1 is used to prevent overfitting. Overfitting occurs when a model learns the training data too well and is not able to generalize to new data. Dropout

randomly drops a certain percentage of activations during training, which forces the model to learn more robust features.

- ***Final Dense Layer with 1 Unit and Sigmoid Activation:***

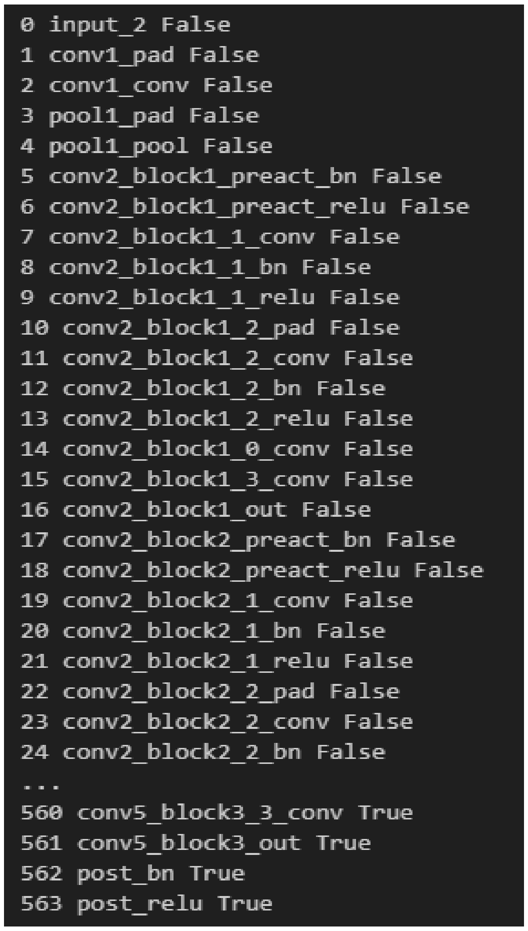
  The final Dense layer with 1 unit and sigmoid activation performs binary classification. The sigmoid activation function outputs a value between 0 and 1, which can be interpreted as the probability of pneumonia.

Case 4: Training using custom CNN coupled with transfer learning and GPU and fine tuning

The model architecture was modified by setting base_model.trainable = True and selectively unfreezing a subset of layers, allowing training for those layers during the fine-tuning process. The total number of parameters in the model remained consistent at 58,594,049, comprising both trainable (4,731,137) and non-trainable (53,862,912) parameters.

```
base_model.trainable = True

# Freeze all layers except for the
for layer in base_model.layers[:-13]:
    layer.trainable = False
# Check which layers are tuneable (trainable)
for layer_number, layer in enumerate(base_model.layers):
    print(layer_number, layer.name, layer.trainable)
```

In this fine-tuning strategy, the decision to unfreeze the last 13 layers aimed to retain knowledge from pre-training while adapting the model to the specific task. Typically, these layers include later convolutional blocks and top layers, which are more task-specific.

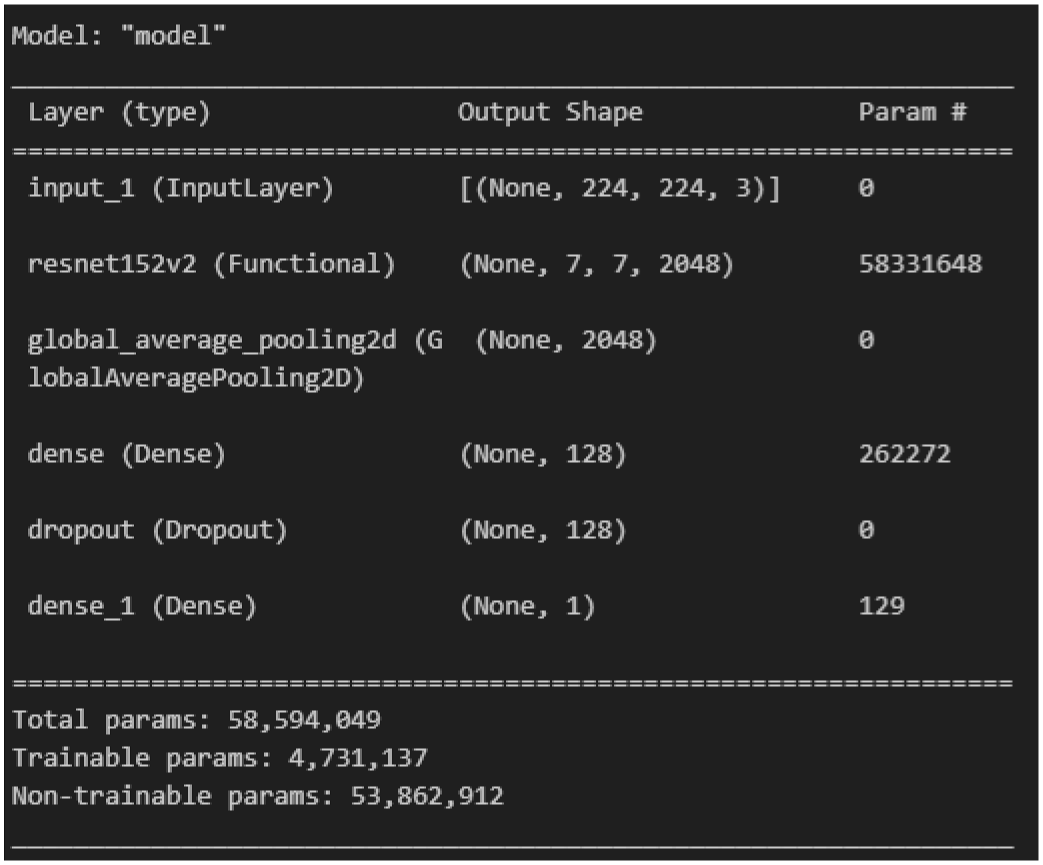


```
0 input_2 False
1 conv1_pad False
2 conv1_conv False
3 pool1_pad False
4 pool1_pool False
5 conv2_block1_preact_bn False
6 conv2_block1_preact_relu False
7 conv2_block1_1_conv False
8 conv2_block1_1_bn False
9 conv2_block1_1_relu False
10 conv2_block1_2_pad False
11 conv2_block1_2_conv False
12 conv2_block1_2_bn False
13 conv2_block1_2_relu False
14 conv2_block1_0_conv False
15 conv2_block1_3_conv False
16 conv2_block1_out False
17 conv2_block2_preact_bn False
18 conv2_block2_preact_relu False
19 conv2_block2_1_conv False
20 conv2_block2_1_bn False
21 conv2_block2_1_relu False
22 conv2_block2_2_pad False
23 conv2_block2_2_conv False
24 conv2_block2_2_bn False
...
560 conv5_block3_3_conv True
561 conv5_block3_out True
562 post_bn True
563 post_relu True
```

Figure 3.2. 6 Frozen and unfrozen layers

During compilation, a lower learning rate (2e-6) was chosen for the optimizer to prevent overfitting and maintain stability in the pre-learned weights. The binary crossentropy loss function was deemed suitable for binary classification tasks.

```
model_pretrained.compile(loss='binary_crossentropy'
                , optimizer = keras.optimizers.Adam(learning_rate=2e-6),
metrics='binary_accuracy')

model_pretrained.summary()
```



```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0

 resnet152v2 (Functional)    (None, 7, 7, 2048)        58331648

 global_average_pooling2d (G  (None, 2048)             0
 lobalAveragePooling2D)

 dense (Dense)               (None, 128)               262272

 dropout (Dropout)           (None, 128)               0

 dense_1 (Dense)             (None, 1)                 129

=================================================================
Total params: 58,594,049
Trainable params: 4,731,137
Non-trainable params: 53,862,912
_________________________________________________________________
```

Figure 3.2. 7 Model architecture using transfer learning

Table 3.2. 1 Model architecture

Layer	Type	Output Shape	Param #	Description
input_1	InputLayer	(None, 224, 224, 3)	0	Receives the input image with dimensions (height, width, color channels).
resnet152v2	Functional	(None, 7, 7, 2048)	58,331,648	The pre-trained ResNet152V2 model, which extracts features from the input image.
global_average_pooling2d	GlobalAveragePooling2D	(None, 2048)	0	Reduces the spatial dimensions of the feature maps to a single vector.
dense	Dense	(None, 128)	262,272	A fully connected layer that transforms the feature vector into a smaller representation.
dropout	Dropout	(None, 128)	0	Randomly drops a certain percentage of neurons during training to prevent overfitting.
dense_1	Dense	(None, 1)	129	The final output layer that produces a single value representing the probability of pneumonia.

Model Testing:

Evaluated the performance of the trained model using the testing dataset, which is distinct from the training and validation sets. This showed how well the model performed to unseen data therefore revealing whether the model is overfitted or underfitted.

Code for printing Test loss, Test accuracy, Val loss and Val accuracy:

```python
score = model_pretrained.evaluate(ds_test, steps = len(df_test), verbose = 0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
score = model_pretrained.evaluate(ds_val, steps = len(val_df)/BATCH, verbose = 0)
print('Val loss:', score[0])
print('Val accuracy:', score[1])
```

Code for Learning curve (Accuracy):

```python
fig, ax = plt.subplots(figsize=(20,8))
sns.lineplot(x = history.epoch, y = history.history['binary_accuracy'])
sns.lineplot(x = history.epoch, y = history.history['val_binary_accuracy'])
ax.set_title('Learning Curve (Accuracy)')
ax.set_ylabel('Accuracy')
ax.set_xlabel('Epoch')
ax.set_ylim(0.90, 1.0)
ax.legend(['train', 'val'], loc='best')
plt.show()
```

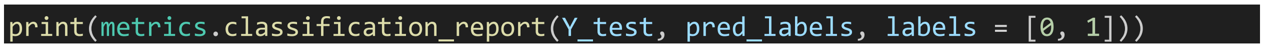
Code for Learning curve (Loss):

```python
fig, ax = plt.subplots(figsize=(20,8))
sns.lineplot(x = history.epoch, y = history.history['loss'])
sns.lineplot(x = history.epoch, y = history.history['val_loss'])
ax.set_title('Learning Curve (Loss)')
ax.set_ylabel('Loss')
ax.set_xlabel('Epoch')
ax.set_ylim(0, 0.3)
ax.legend(['train', 'val'], loc='best')
plt.show()
```

Evaluation:

Calculated evaluation metrics such as accuracy, loss, precision, recall to assess the model's accuracy in pneumonia detection.

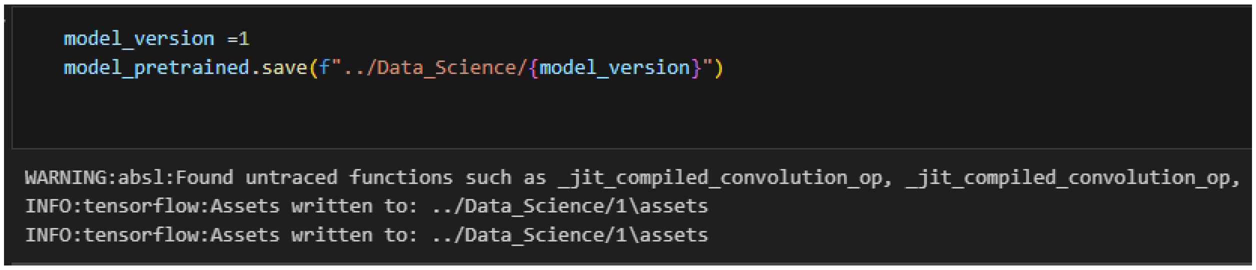
Code for printing the model’s general report:



```
print(metrics.classification_report(Y_test, pred_labels, labels = [0, 1]))
```

Model Deployment & Integration:

The trained model was saved as version 1 so as to be used for classifying images.



```
model_version =1
model_pretrained.save(f"../Data_Science/{model_version}")
```

```
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_compiled_convolution_op,
INFO:tensorflow:Assets written to: ../Data_Science/1\assets
INFO:tensorflow:Assets written to: ../Data_Science/1\assets
```

Figure 3.2. 8 Model saving

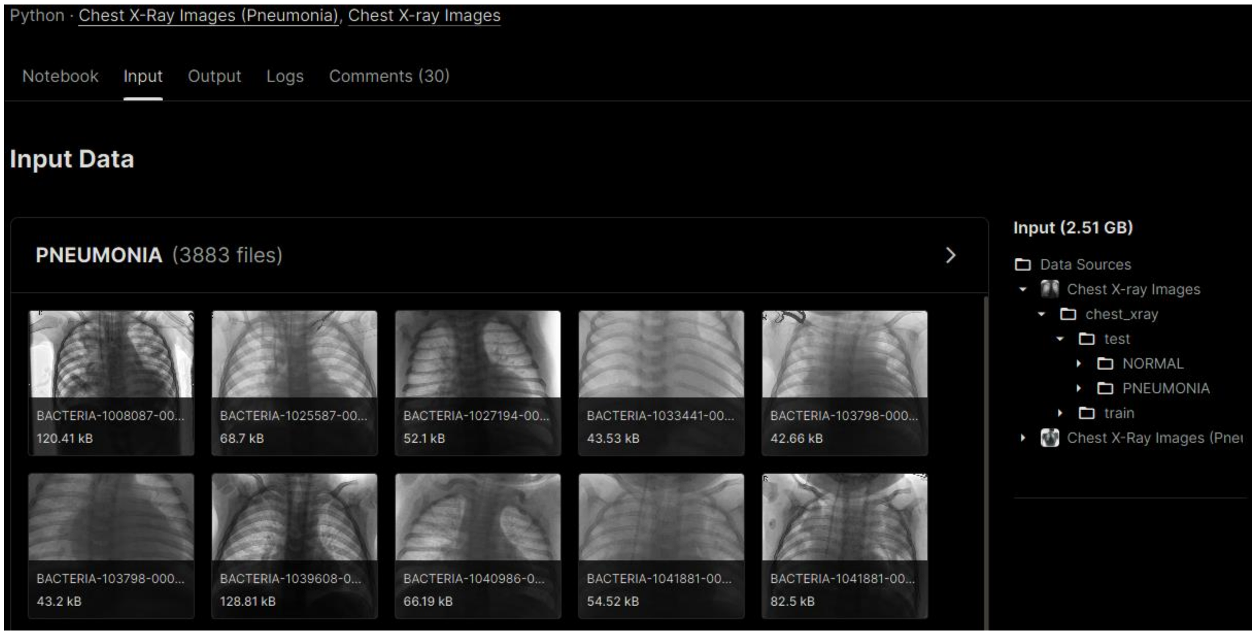
Developed a website to provide a user-friendly interface for healthcare professionals to input X-ray images and obtain diagnostic results promptly.

The website was built using FastAPI where the saved model ran in the back-end and the front-end enabled a user to upload an image for classification. This also enables easy integration into healthcare settings. Two softcopy images and two hardcopy images were provided by a hospital and each was labelled as either “Healthy” or “Sick” by a radiologist so as to test the model with real-life cases.

CHAPTER FOUR

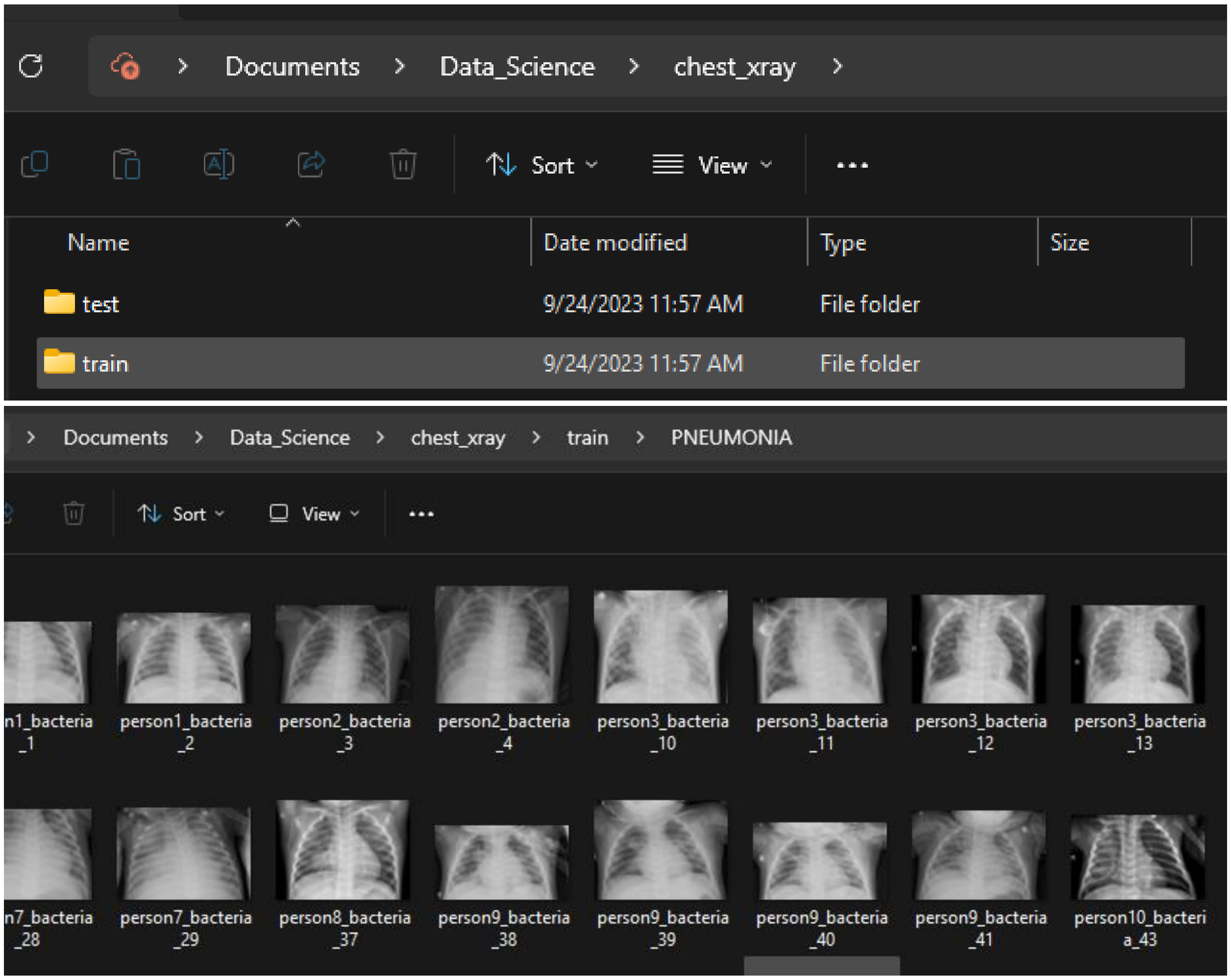
4. RESULTS

4.1 Data acquisition



Python · Chest X-Ray Images (Pneumonia), Chest X-ray Images
Notebook
Input
Output
Logs
Comments (30)
Input Data
PNEUMONIA (3883 files)
BACTERIA-1008087-00... 120.41 kB
BACTERIA-1025587-00... 68.7 kB
BACTERIA-1027194-00... 52.1 kB
BACTERIA-1033441-00... 43.53 kB
BACTERIA-103798-000... 42.66 kB
BACTERIA-103798-000... 43.2 kB
BACTERIA-1039608-0... 128.81 kB
BACTERIA-1040986-0... 66.19 kB
BACTERIA-1041881-00... 54.52 kB
BACTERIA-1041881-00... 82.5 kB
Input (2.51 GB)
Data Sources
Chest X-ray Images
chest_xray
test
NORMAL
PNEUMONIA
train
Chest X-Ray Images (Pne

Figue 4.1 1 Kaggle website



Documents > Data_Science > chest_xray >

Name	Date modified	Type	Size
test	9/24/2023 11:57 AM	File folder	
train	9/24/2023 11:57 AM	File folder	

> Documents > Data_Science > chest_xray > train > PNEUMONIA

```
n1_bacteria_1   person1_bacteria_2   person2_bacteria_3   person2_bacteria_4   person3_bacteria_10   person3_bacteria_11   person3_bacteria_12   person3_bacteria_13
n7_bacteria_28  person7_bacteria_29  person8_bacteria_37  person9_bacteria_38  person9_bacteria_39   person9_bacteria_40   person9_bacteria_41   person10_bacteria_43
```

Figue 4.1 2 Directory containing images

4.2 Data pre-processing procedures

This involved the following:

1. Concatenation

Merging of the datasets.

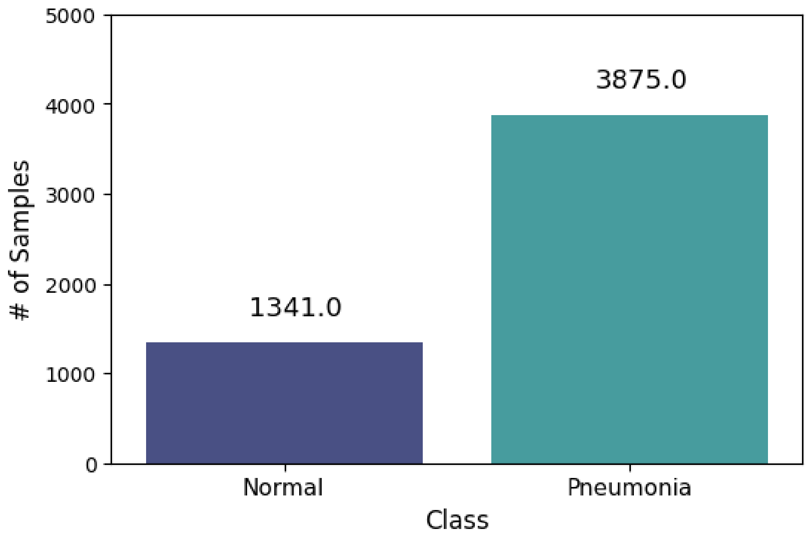


Figure 4.2. 1 Train dataset

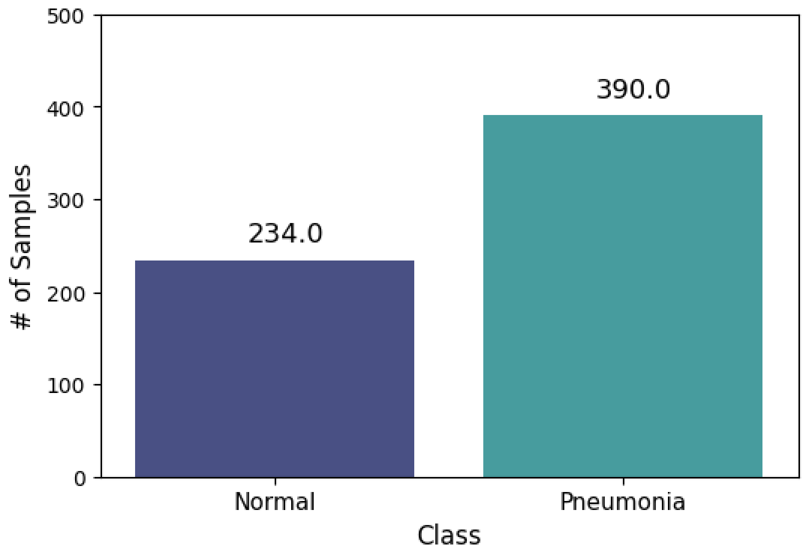


Figure 4.2. 2 Test Dataset

Merging of the classes for training resulted in having 5216 images for training while having 624 images for testing.

2. Resizing

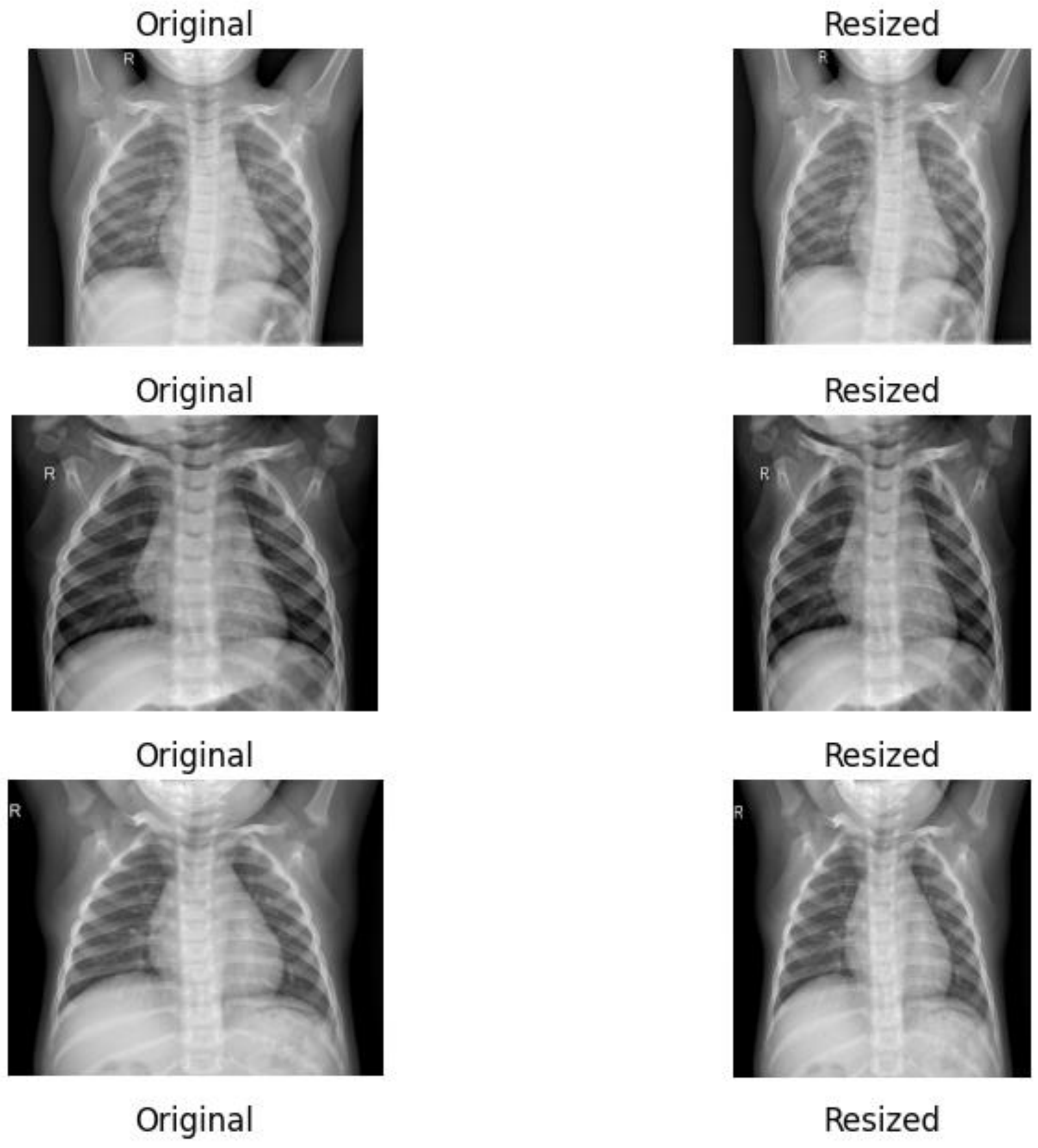
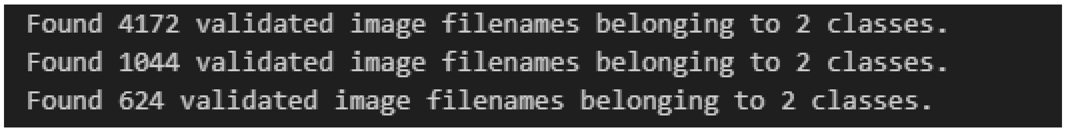


Figure 4.2. 3 Resized Images

3. Train, validation & test split



```
Found 4172 validated image filenames belonging to 2 classes.
Found 1044 validated image filenames belonging to 2 classes.
Found 624 validated image filenames belonging to 2 classes.
```

Figure 4.2. 4 Split Dataset

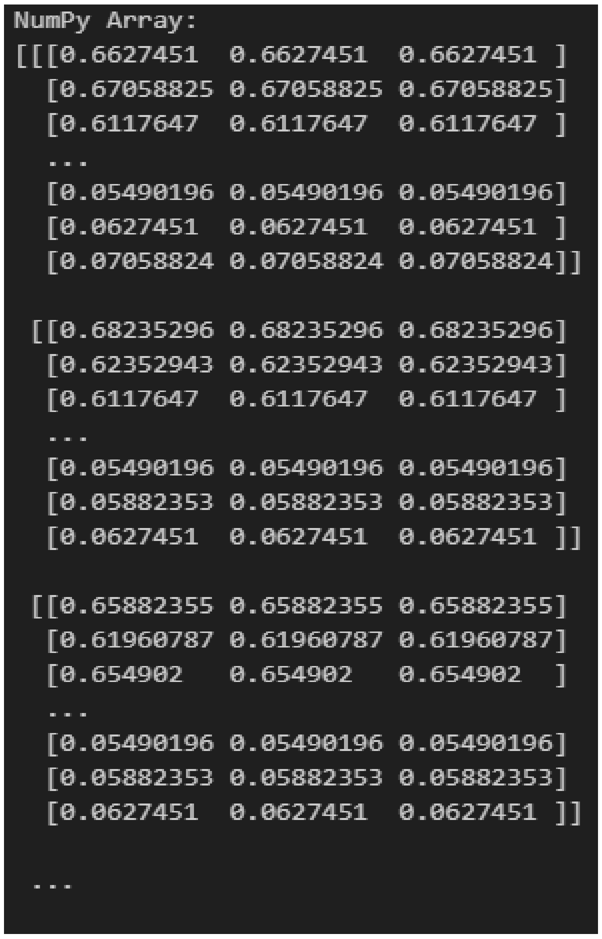
A 70:20:10 split was done

Training dataset consisted of 4172 images.

Validation dataset consisted of 1044 images.

Test dataset consisted of 624 images.

4. Converting the images to NumPy array



```
NumPy Array:
[[[0.6627451  0.6627451  0.6627451 ]
  [0.67058825 0.67058825 0.67058825]
  [0.6117647  0.6117647  0.6117647 ]
  ...
  [0.05490196 0.05490196 0.05490196]
  [0.0627451  0.0627451  0.0627451 ]
  [0.07058824 0.07058824 0.07058824]]

 [[0.68235296 0.68235296 0.68235296]
  [0.62352943 0.62352943 0.62352943]
  [0.6117647  0.6117647  0.6117647 ]
  ...
  [0.05490196 0.05490196 0.05490196]
  [0.05882353 0.05882353 0.05882353]
  [0.0627451  0.0627451  0.0627451 ]]

 [[0.65882355 0.65882355 0.65882355]
  [0.61960787 0.61960787 0.61960787]
  [0.654902   0.654902   0.654902  ]
  ...
  [0.05490196 0.05490196 0.05490196]
  [0.05882353 0.05882353 0.05882353]
  [0.0627451  0.0627451  0.0627451 ]]

 ...
```

Figure 4.2. 5 Images converted to NumPy Array

4.3 Model training and Evaluation

Case 1: Using CNN with CPU

- Epoch 1/50
  130/130 [==============================] - 241s 2s/step - loss: 0.2152 - binary_accuracy: 0.9108 - val_loss: 0.1674 - val_binary_accuracy: 0.9282 - lr: 6.0000e-06
- Epoch 2/50
  130/130 [==============================] - 239s 2s/step - loss: 0.2057 - binary_accuracy: 0.9180 - val_loss: 0.1978 - val_binary_accuracy: 0.9090 - lr: 6.0000e-06
- Epoch 3/50
  131/130 [==============================] - ETA: -1s - loss: 0.2073 - binary_accuracy: 0.9207
- Epoch 3: ReduceLROnPlateau reducing learning rate to 1.1999999514955563e-06.
  130/130 [==============================] - 231s 2s/step - loss: 0.2073 - binary_accuracy: 0.9207 - val_loss: 0.2239 - val_binary_accuracy: 0.8956 - lr: 6.0000e-06
- Epoch 4/50
  130/130 [==============================] - 230s 2s/step - loss: 0.1969 - binary_accuracy: 0.9204 - val_loss: 0.1930 - val_binary_accuracy: 0.9119 - lr: 1.2000e-06
- Epoch 5/50
  131/130 [==============================] - ETA: 0s - loss: 0.1927 - binary_accuracy: 0.9243
- Epoch 5: ReduceLROnPlateau reducing learning rate to 2.3999998575163774e-07.
  130/130 [==============================] - 216s 2s/step - loss: 0.1927 - binary_accuracy: 0.9243 - val_loss: 0.1992 - val_binary_accuracy: 0.9100 - lr: 1.2000e-06
- Epoch 6/50
  130/130 [==============================] - 185s 1s/step - loss: 0.2002 - binary_accuracy: 0.9202 - val_loss: 0.2017 - val_binary_accuracy: 0.9100 - lr: 2.4000e-07

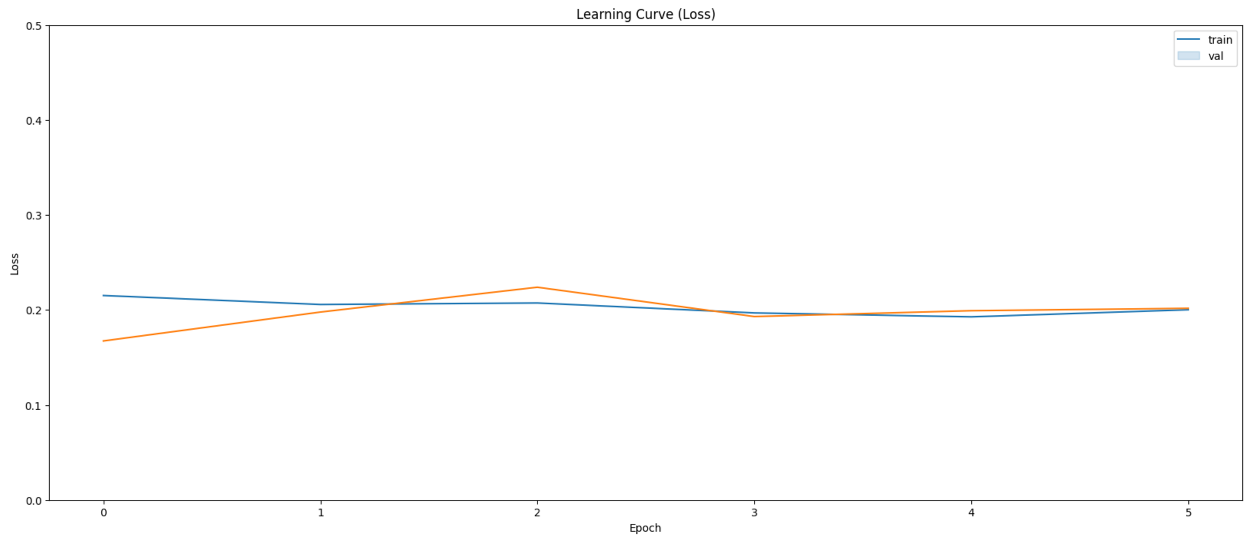


Figure 4.3. 1 Learning curve (Loss) using CPU

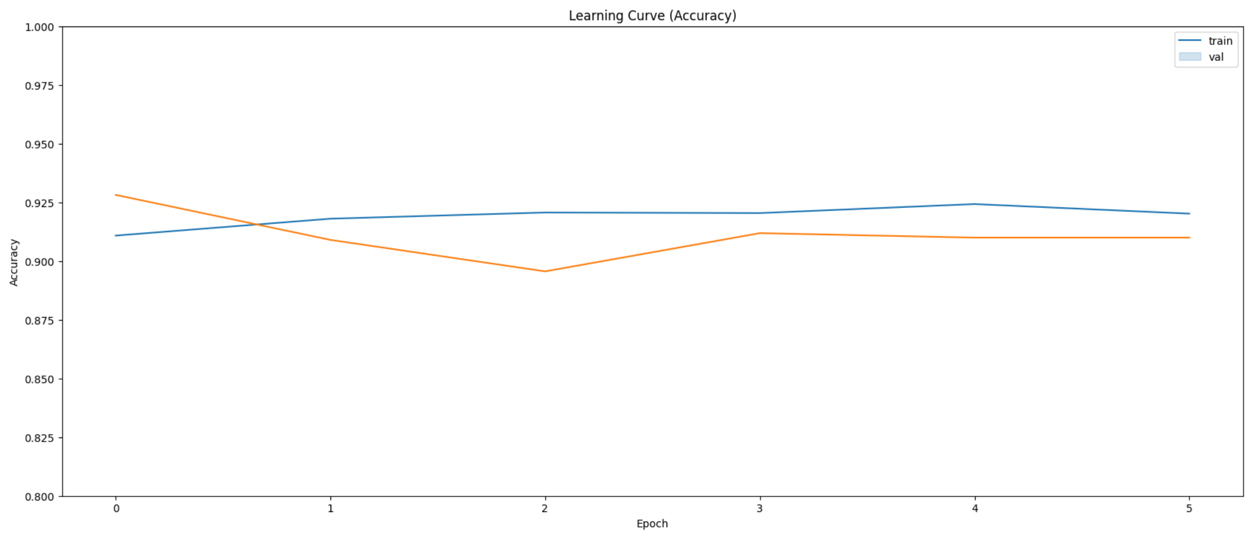


Figure 4.3. 2 Learning curve (Accuracy) using CPU

Interpretation

The learning curves pertaining to both accuracy and loss exhibit a discernible plateau, suggesting a quasi-constant trend. This observation is ascribed to the model's perceived inadequacy in capturing the intricacies of the dataset due to insufficient complexity. Additionally, the notable elevation in accuracy to 92.82% implies a potential concern of overfitting, wherein the model may excessively tailor itself to the training data, thereby compromising its ability to generalize effectively to new, unseen data instances.

Val loss: 0.16736143827438354
Val accuracy: 0.9281609058380127
Test loss: 0.3962566554546356
Test accuracy: 0.8493589758872986

Case 2: Using CNN with GPU

Result

- Epoch 1/50
  130/130 [==============================] - 284s 2s/step - loss: 0.4680 - binary_accuracy: 0.7876 - val_loss: 1.1304 - val_binary_accuracy: 0.7433 - lr: 3.0000e-05
- Epoch 2/50
  130/130 [==============================] - 178s 1s/step - loss: 0.2947 - binary_accuracy: 0.8773 - val_loss: 2.0763 - val_binary_accuracy: 0.7433 - lr: 3.0000e-05

```
• Epoch 3/50
  131/130 [==============================] - ETA: 0s - loss: 0.2624 -
  binary_accuracy: 0.8878
• Epoch 3: ReduceLROnPlateau reducing learning rate to 5.9999998484272515e-06.
  130/130 [==============================] - 174s 1s/step - loss: 0.2624 -
  binary_accuracy: 0.8878 - val_loss: 1.9693 - val_binary_accuracy: 0.7433 - lr: 3.0000e-05
• Epoch 4/50
  130/130 [==============================] - 178s 1s/step - loss: 0.2267 -
  binary_accuracy: 0.9108 - val_loss: 1.1964 - val_binary_accuracy: 0.7433 - lr: 6.0000e-06
• Epoch 5/50
  130/130 [==============================] - 197s 2s/step - loss: 0.2143 -
  binary_accuracy: 0.9096 - val_loss: 0.2497 - val_binary_accuracy: 0.8851 - lr: 6.0000e-06
• Epoch 6/50
  130/130 [==============================] - 185s 1s/step - loss: 0.2190 -
  binary_accuracy: 0.9072 - val_loss: 0.1578 - val_binary_accuracy: 0.9444 - lr: 6.0000e-06
• Epoch 7/50
  130/130 [==============================] - 194s 1s/step - loss: 0.2070 -
  binary_accuracy: 0.9197 - val_loss: 0.1850 - val_binary_accuracy: 0.9176 - lr: 6.0000e-06
• Epoch 8/50
  131/130 [==============================] - ETA: 0s - loss: 0.2100 -
  binary_accuracy: 0.9159
• Epoch 8: ReduceLROnPlateau reducing learning rate to 1.1999999514955563e-06.
  130/130 [==============================] - 178s 1s/step - loss: 0.2100 -
  binary_accuracy: 0.9159 - val_loss: 0.1882 - val_binary_accuracy: 0.9119 - lr: 6.0000e-06
• Epoch 9/50
  130/130 [==============================] - 174s 1s/step - loss: 0.2078 -
  binary_accuracy: 0.9190 - val_loss: 0.2013 - val_binary_accuracy: 0.9071 - lr: 1.2000e-06
• Epoch 10/50
  131/130 [==============================] - ETA: 0s - loss: 0.2038 -
  binary_accuracy: 0.9156
• Epoch 10: ReduceLROnPlateau reducing learning rate to 2.3999998575163774e-07.
  130/130 [==============================] - 181s 1s/step - loss: 0.2038 -
  binary_accuracy: 0.9156 - val_loss: 0.2105 - val_binary_accuracy: 0.9004 - lr: 1.2000e-06
• Epoch 11/50
  130/130 [==============================] - 206s 2s/step - loss: 0.1938 -
  binary_accuracy: 0.9264 - val_loss: 0.2126 - val_binary_accuracy: 0.9013 - lr: 2.4000e-07
```

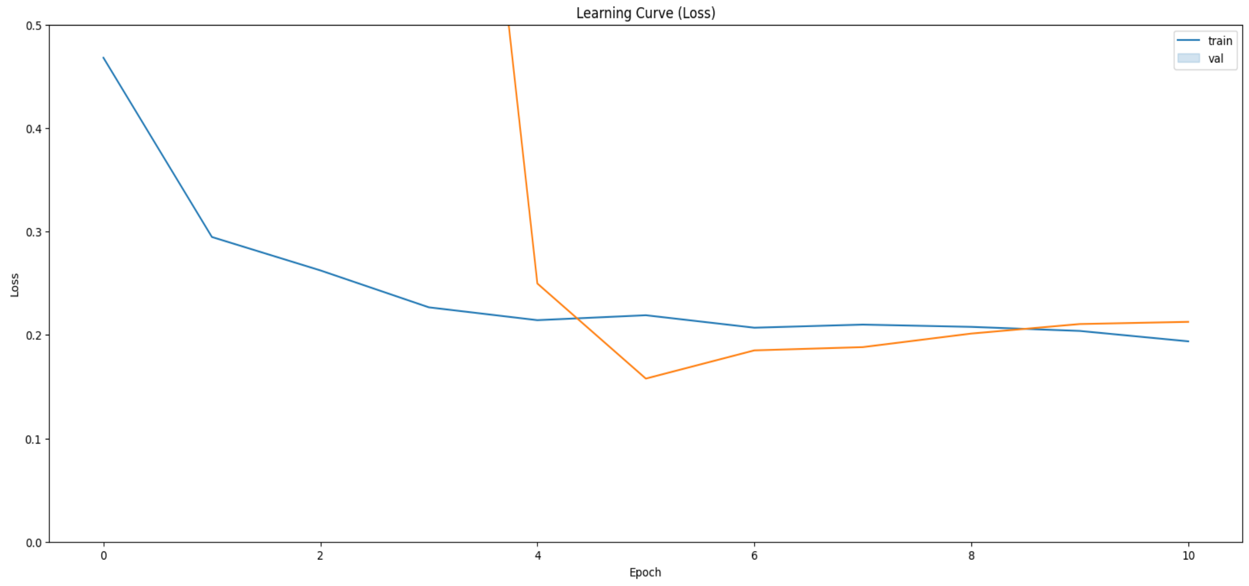


Figure 4.3. 3 Learning curve (Loss) using GPU

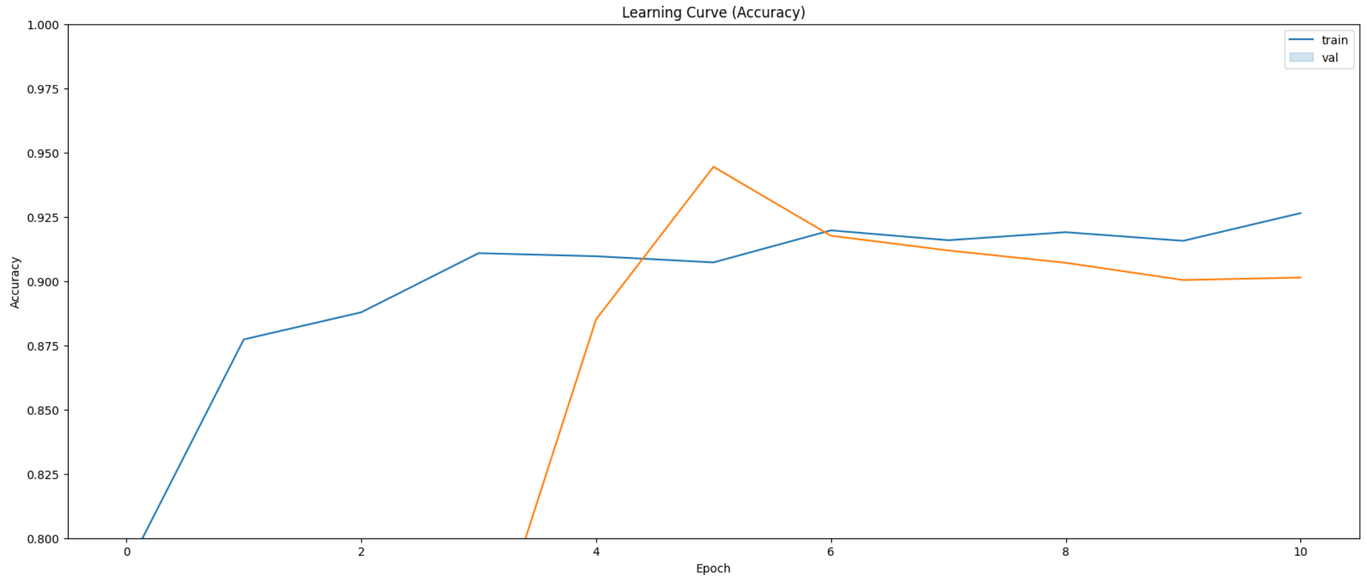


Figure 4.3. 4 Learning curve (Accuracy) using GPU

Val loss: 0.15775328874588013
Val accuracy: 0.9444444179534912

Test loss: 0.45926719903945923

Test accuracy: 0.8205128312110901

Interpretation

The model's training loss decreases over time, while its training and validation accuracy increase.

The ReduceLROnPlateau callback is triggered twice during training, which indicates that the model's learning rate is being reduced in order to prevent overfitting. This is a good sign, as it suggests that the model is learning effectively and not simply memorizing the training data.

The learning curve plot shows that the model's training and validation losses are converging, which means that the model is learning to generalize well to new data. The training accuracy is slightly higher than the validation accuracy, which typically suggests that the model is learning well from the training data but may not generalize as effectively to new, unseen data.

Case 3: Using custom CNN coupled with transfer learning and GPU

Result

- Epoch 1/50
  130/130 [==============================] - 164s 1s/step - loss: 0.3573 - binary_accuracy: 0.8387 - val_loss: 0.1985 - val_binary_accuracy: 0.9224 - lr: 5.0000e-05
- Epoch 2/50
  130/130 [==============================] - 111s 851ms/step - loss: 0.2020 - binary_accuracy: 0.9288 - val_loss: 0.1685 - val_binary_accuracy: 0.9272 - lr: 5.0000e-05
- Epoch 3/50
  130/130 [==============================] - 110s 842ms/step - loss: 0.1708 - binary_accuracy: 0.9334 - val_loss: 0.1393 - val_binary_accuracy: 0.9454 - lr: 5.0000e-05
- Epoch 4/50
  130/130 [==============================] - 109s 840ms/step - loss: 0.1488 - binary_accuracy: 0.9463 - val_loss: 0.1326 - val_binary_accuracy: 0.9492 - lr: 5.0000e-05
- Epoch 5/50
  130/130 [==============================] - 110s 844ms/step - loss: 0.1413 - binary_accuracy: 0.9473 - val_loss: 0.1181 - val_binary_accuracy: 0.9540 - lr: 5.0000e-05
- Epoch 6/50
  130/130 [==============================] - 110s 843ms/step - loss: 0.1276 - binary_accuracy: 0.9528 - val_loss: 0.1046 - val_binary_accuracy: 0.9626 - lr: 5.0000e-05
- Epoch 7/50
  130/130 [==============================] - 110s 844ms/step - loss: 0.1220 - binary_accuracy: 0.9600 - val_loss: 0.0953 - val_binary_accuracy: 0.9655 - lr: 5.0000e-05
- Epoch 8/50
  130/130 [==============================] - 110s 843ms/step - loss: 0.1164 - binary_accuracy: 0.9614 - val_loss: 0.0967 - val_binary_accuracy: 0.9655 - lr: 5.0000e-05
- Epoch 9/50
  130/130 [==============================] - 110s 843ms/step - loss: 0.1195 - binary_accuracy: 0.9557 - val_loss: 0.0948 - val_binary_accuracy: 0.9665 - lr: 5.0000e-05
- Epoch 10/50
  130/130 [==============================] - 110s 845ms/step - loss: 0.1088 - binary_accuracy: 0.9612 - val_loss: 0.1233 - val_binary_accuracy: 0.9531 - lr: 5.0000e-05
- Epoch 11/50
  131/130 [==============================] - ETA: 0s - loss: 0.1039 - binary_accuracy: 0.9621
- Epoch 11: ReduceLROnPlateau reducing learning rate to 9.999999747378752e-06.
  130/130 [==============================] - 110s 845ms/step - loss: 0.1039 - binary_accuracy: 0.9621 - val_loss: 0.0997 - val_binary_accuracy: 0.9626 - lr: 5.0000e-05
- Epoch 12/50
  ...
- Epoch 23: ReduceLROnPlateau reducing learning rate to 1.5999999902760466e-08.

```
130/130 [==============================] - 115s 884ms/step - loss: 0.1023 -
binary_accuracy: 0.9616 - val_loss: 0.0872 - val_binary_accuracy: 0.9693 - lr: 8.0000e-08
```

- Epoch 24/50

```
130/130 [==============================] - 115s 881ms/step - loss: 0.1011 -
binary_accuracy: 0.9624 - val_loss: 0.0872 - val_binary_accuracy: 0.9693 - lr: 1.6000e-08
```

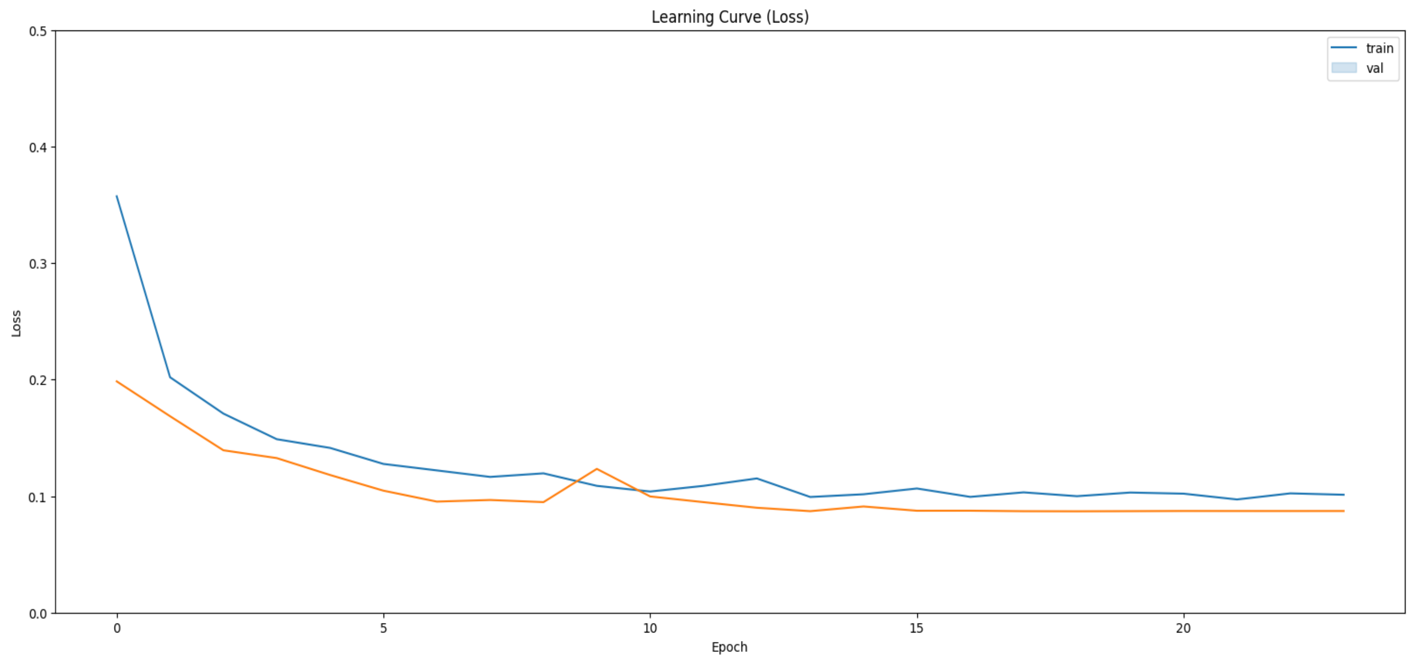


Figure 4.3. 5 Learning curve (Loss) Using custom CNN coupled with transfer learning and GPU

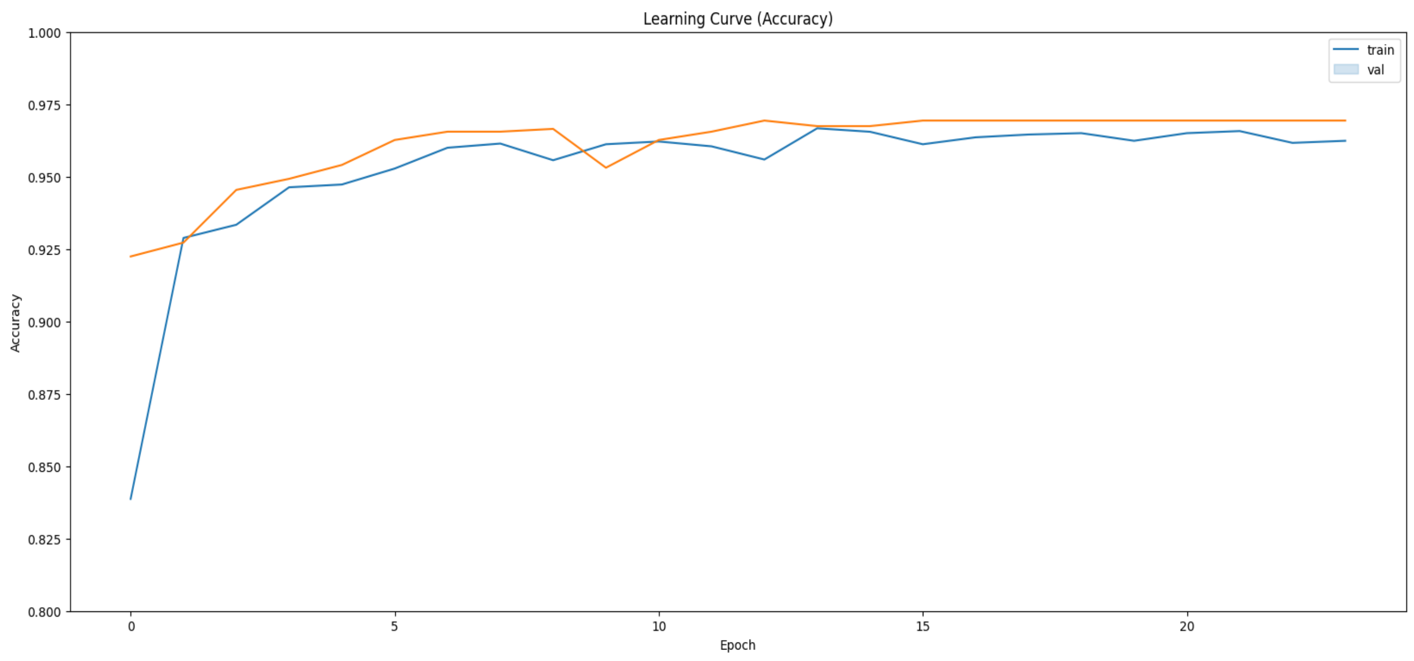


Figure 4.3. 6 Learning curve (Accuracy) Using custom CNN coupled with transfer learning and GPU

```
Val loss: 0.08693970739841461
Val accuracy: 0.962348669052124

Test loss: 0.4487508535385132
Test accuracy: 0.862814102411127014
```

Interpretation

The training results for the model with transfer learning are much better than the results for the model without transfer learning. This is evident from the following observations:

- The training loss for the model with transfer learning is lower than the training loss for the model without transfer learning.
- The training accuracy for the model with transfer learning is higher than the training accuracy for the model without transfer learning.
- The validation accuracy for the model with transfer learning is significantly higher than the validation accuracy for the model without transfer learning.

The better performance of the model with transfer learning is likely due to the fact that the pre-trained ResNet152V2 model has already learned to extract useful features from images. The custom head layers in the model with transfer learning can then focus on fine-tuning these features for the specific task of pneumonia detection.

Case 4: Using custom CNN coupled with transfer learning and GPU and fine tuning

Results

```
•  Epoch 1/50
   130/130 [==============================] - 129s 924ms/step - loss: 0.1964 -
   binary_accuracy: 0.9226 - val_loss: 0.1487 - val_binary_accuracy: 0.9579 - lr: 2.0000e-06
•  Epoch 2/50
   130/130 [==============================] - 117s 896ms/step - loss: 0.1586 -
   binary_accuracy: 0.9473 - val_loss: 0.1359 - val_binary_accuracy: 0.9617 - lr: 2.0000e-06
•  Epoch 3/50
   130/130 [==============================] - 115s 882ms/step - loss: 0.1441 -
   binary_accuracy: 0.9545 - val_loss: 0.1239 - val_binary_accuracy: 0.9646 - lr: 2.0000e-06
•  Epoch 4/50
   130/130 [==============================] - 113s 872ms/step - loss: 0.1444 -
   binary_accuracy: 0.9501 - val_loss: 0.1167 - val_binary_accuracy: 0.9626 - lr: 2.0000e-06
•  Epoch 5/50
   130/130 [==============================] - 113s 867ms/step - loss: 0.1345 -
   binary_accuracy: 0.9559 - val_loss: 0.1164 - val_binary_accuracy: 0.9626 - lr: 2.0000e-06
•  Epoch 6/50
   130/130 [==============================] - 113s 865ms/step - loss: 0.1284 -
   binary_accuracy: 0.9571 - val_loss: 0.1161 - val_binary_accuracy: 0.9626 - lr: 2.0000e-06
•  Epoch 7/50
   130/130 [==============================] - 117s 897ms/step - loss: 0.1277 -
   binary_accuracy: 0.9552 - val_loss: 0.1121 - val_binary_accuracy: 0.9646 - lr: 2.0000e-06
•  Epoch 8/50
   130/130 [==============================] - 115s 884ms/step - loss: 0.1246 -
   binary_accuracy: 0.9564 - val_loss: 0.1079 - val_binary_accuracy: 0.9665 - lr: 2.0000e-06
•  Epoch 9/50
   130/130 [==============================] - 118s 909ms/step - loss: 0.1143 -
   binary_accuracy: 0.9616 - val_loss: 0.1101 - val_binary_accuracy: 0.9626 - lr: 2.0000e-06
•  Epoch 10/50
   131/130 [==============================] - ETA: 0s - loss: 0.1111 -
   binary_accuracy: 0.9621
•  Epoch 10: ReduceLROnPlateau reducing learning rate to 3.999999989900971e-07.
   130/130 [==============================] - 117s 896ms/step - loss: 0.1111 -
   binary_accuracy: 0.9621 - val_loss: 0.1084 - val_binary_accuracy: 0.9626 - lr: 2.0000e-06
•  Epoch 11/50
   130/130 [==============================] - 116s 893ms/step - loss: 0.1055 -
   binary_accuracy: 0.9616 - val_loss: 0.1076 - val_binary_accuracy: 0.9636 - lr: 4.0000e-07
•  Epoch 12/50
   ...
•  Epoch 21: ReduceLROnPlateau reducing learning rate to 1.5999999902760466e-08.
```

```
130/130 [==============================] - 116s 891ms/step - loss: 0.1080 -
binary_accuracy: 0.9590 - val_loss: 0.1037 - val_binary_accuracy: 0.9665 - lr: 8.0000e-08
```

- Epoch 22/50

```
130/130 [==============================] - 118s 907ms/step - loss: 0.1052 -
binary_accuracy: 0.9621 - val_loss: 0.1027 - val_binary_accuracy: 0.9674 - lr: 1.6000e-08
```

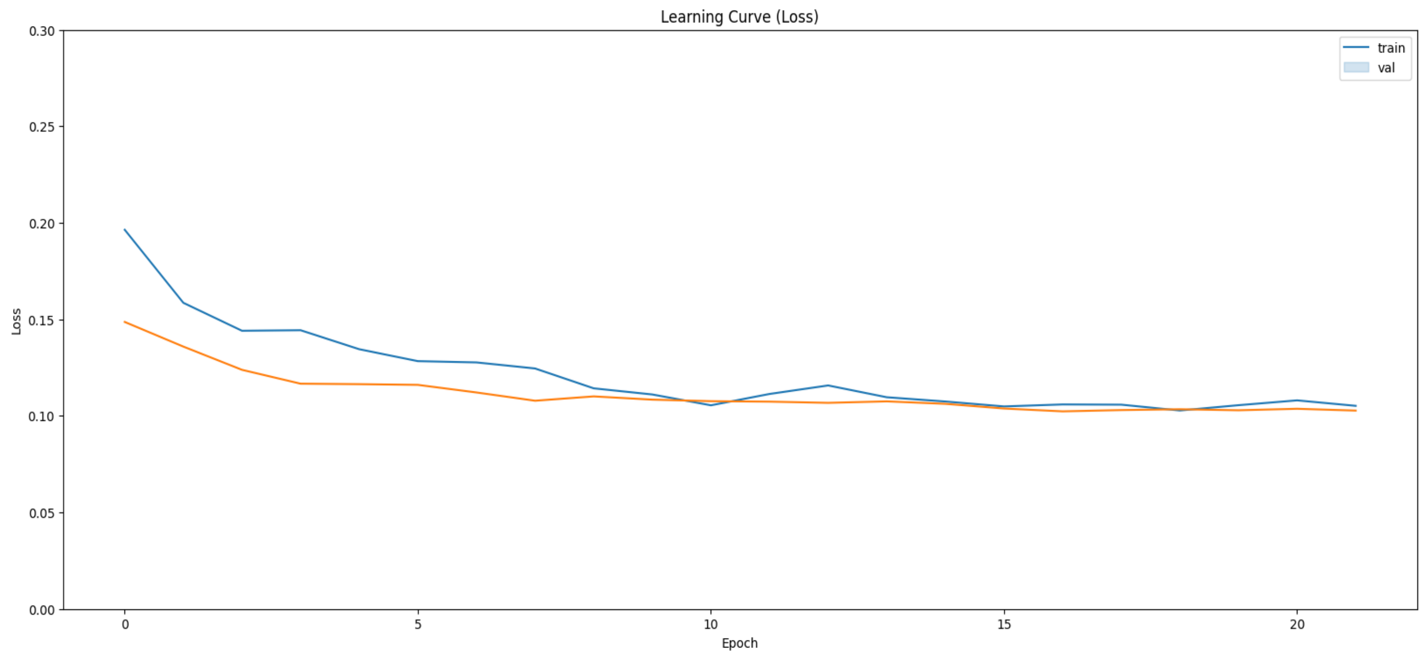


Figure 4.3. 7 Learning curve (Loss) Using custom CNN coupled with transfer learning and GPU and fine tuning

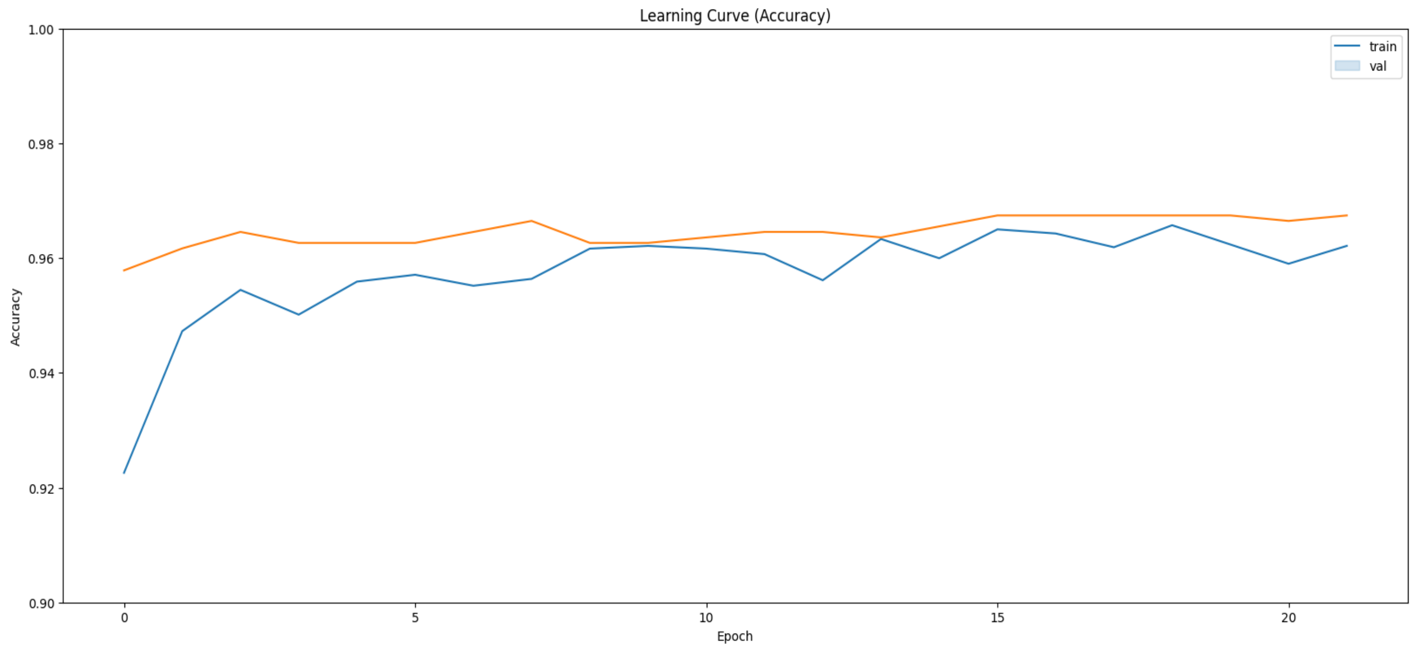


Figure 4.3. 8 Learning curve (Accuracy) Using custom CNN coupled with transfer learning and GPU and fine tuning

```
Val loss: 0.1023356169462204
Val accuracy: 0.964032975769043
Test loss: 0.3490997552871704
Test accuracy: 0.8669871687889099
```

Interpretation

The results show that fine-tuning the pre-trained ResNet152V2 model by unfreezing all layers and then refreezing the first 13 layers has improved the overall performance of the model compared to the model without fine-tuning.

Both the validation loss and validation accuracy have increased with fine-tuning, indicating that the model is better at generalizing to new data. The test loss has also decreased and the test accuracy has increased, suggesting that the model is more effective at predicting pneumonia in real-world X-ray images.

Overall, the fine-tuning approach has been successful in improving the performance of the pneumonia detection model. This demonstrates the effectiveness of transfer learning in leveraging the power of pre-trained models while adapting them to specific tasks.

The validation loss and accuracy curves for the fine-tuned model are smoother than the curves for the model without fine-tuning. This suggests that the fine-tuned model is more stable and less likely to overfit the training data.

Table 4.3. 1 Validation and Test Accuracy

CASE	**Validation accuracy (%)**	**Test accuracy (%)**
1	92.82	84.94
2	94.44	82.05
3	96.23	86.28
4	96.40	86.70

The difference between the test accuracy from Case 3 to 4 was 0.42 and the difference between the validation accuracy was 0.17. The differences were small and thus the model was not adjusted further.

The final model was evaluated as shown below:

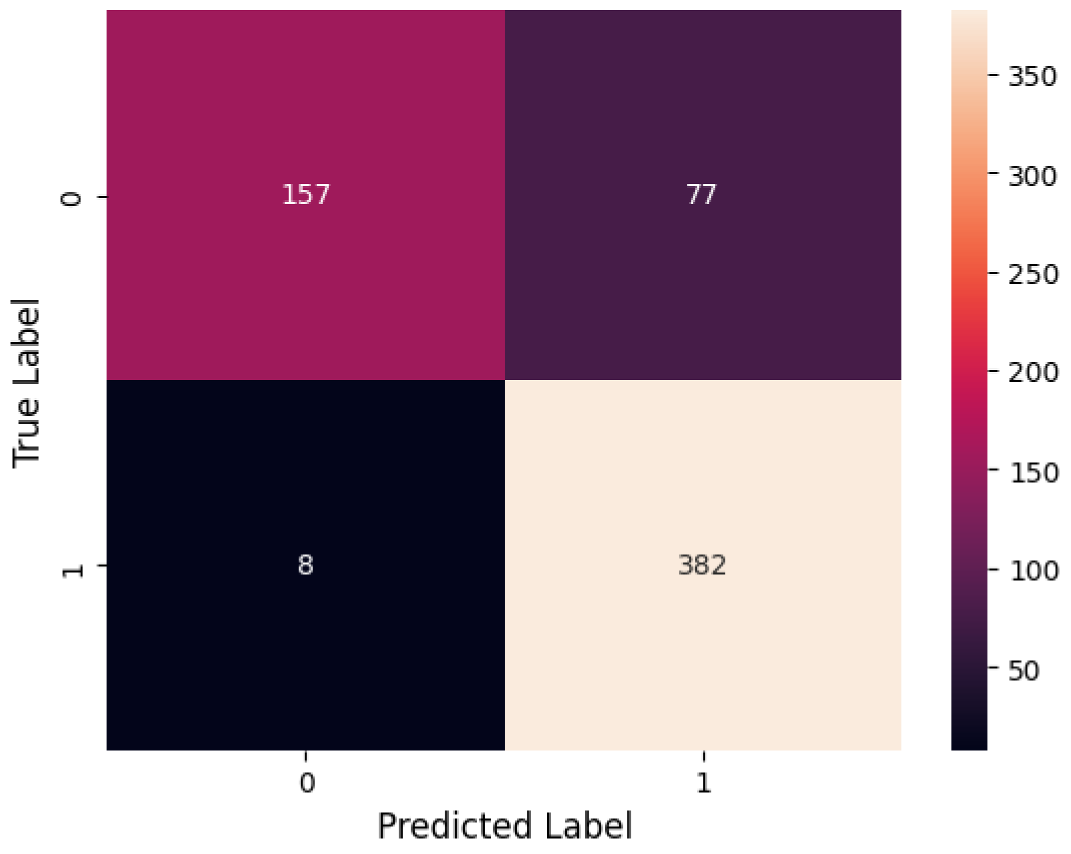


Figure 4.3. 9 Confusion matrix

Table 4.3. 2 Evaluation of the final model

	precision	recall	F1-score	Support
0	0.95	0.68	0.79	234
1	0.83	0.98	0.90	390
Accuracy			0.86	624

- The overall accuracy of the model is 86.7%, which means that it correctly predicted the label for 87% of the test data.
- The model is very good at predicting the absence of pneumonia (class 0), with a precision of 95% and a recall of 68%. This means that 95% of the time the model predicts that a patient does not have pneumonia, the prediction is correct. Additionally, 68% of the time a patient actually does not have pneumonia, the model correctly predicts that they do not.
- The model is also good at predicting the presence of pneumonia (class 1), with a precision of 84% and a recall of 98%. This means that 84% of the time the model predicts that a patient has pneumonia, the prediction is correct. Additionally, 98% of the time a patient actually has pneumonia, the model correctly predicts that they do.

The precision and recall metrics are complementary. Precision measures how accurate the model is when it makes a positive prediction, while recall measures how complete the model is in finding all positive cases.

F1-score: The F1-score is a harmonic mean of precision and recall. It is a good measure of the model's overall performance. In this case, the F1-score for class 0 is 0.79 and the F1-score for class 1 is 0.90. This means that the model is performing well at both identifying and predicting both the presence and absence of pneumonia.

F1-score > 0.9: Excellent performance, often indicating the model is very good at both precision and recall.

0.7-0.9: Good performance, especially for imbalanced datasets or complex tasks.

0.5-0.7: Moderate performance, might be acceptable for some applications if other factors are favorable.

< 0.5: Poor performance, suggests the model needs significant improvement.

4.4 Model deployment and Testing

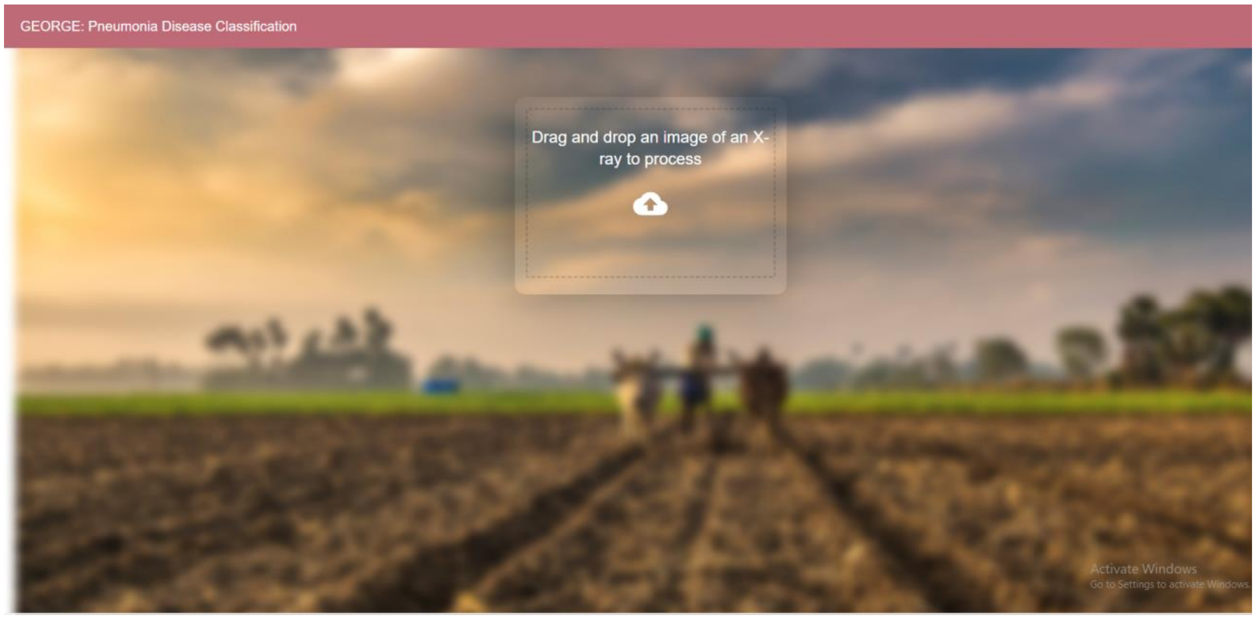

GEORGE: Pneumonia Disease Classification
Drag and drop an image of an X-ray to process
Activate Windows
Go to Settings to activate Windows.

Figure 4.4. 1 Upload page of website

A website was built to provide a user-friendly interface for classifying the images.

The softcopy images were tested out first

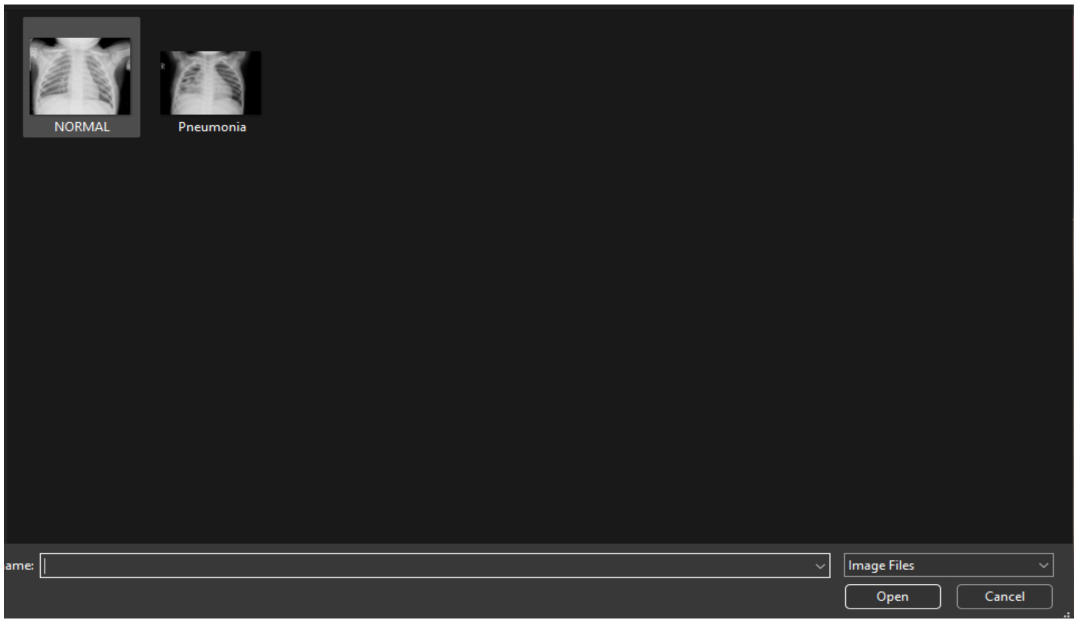


Figure 4.4. 2 Selecting the x-ray images for upload

The files are selected

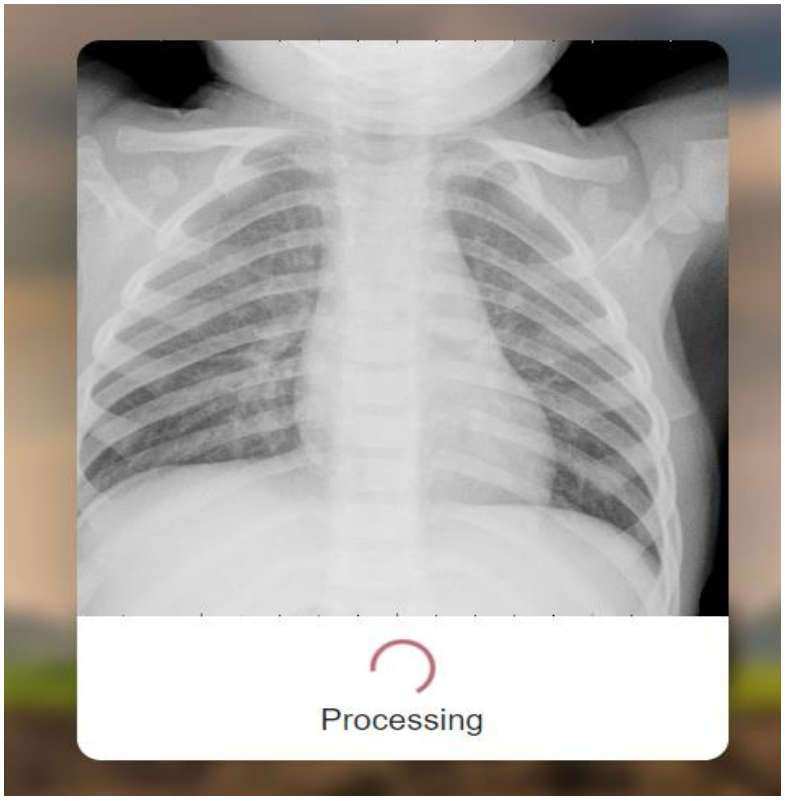
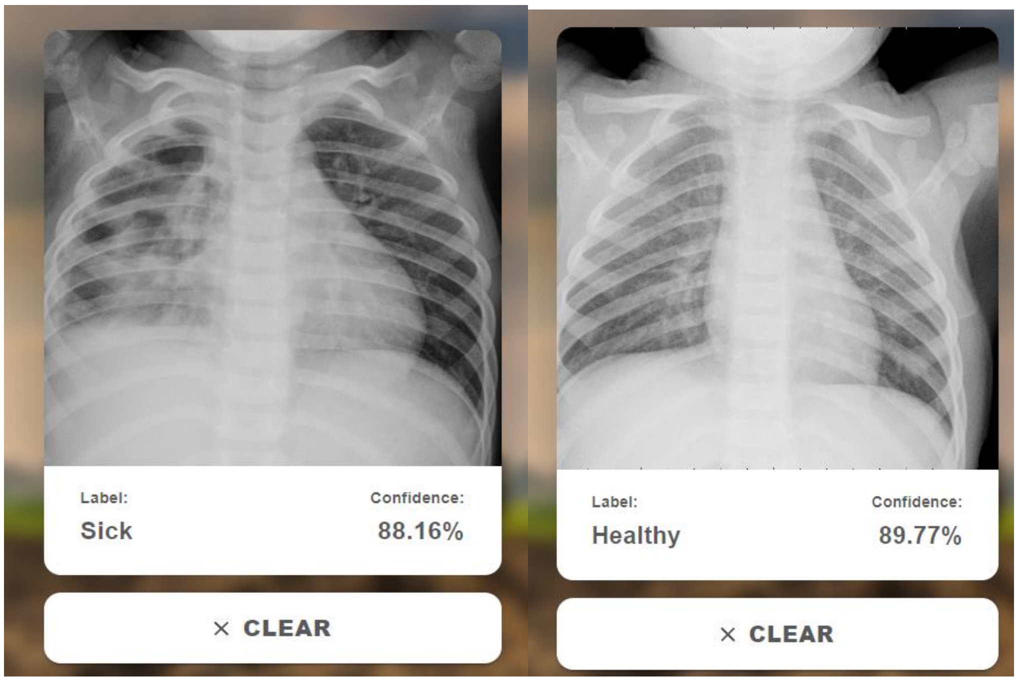


Figure 4.4. 3 Images are processed after being uploaded

The image is processed by the model running in the back-end and then it is classified as either "Sick" or "Healthy".

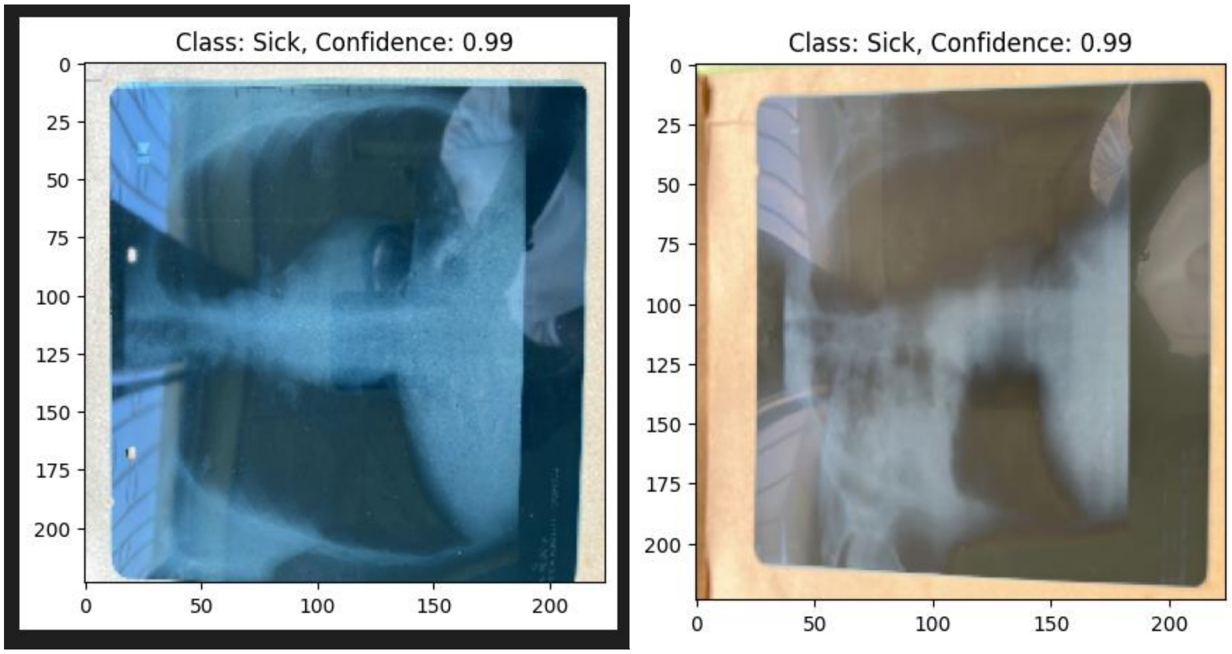

Label: Sick   Confidence: 88.16%

Label: Healthy   Confidence: 89.77%

Figure 4.4. 4 Classified Images

The classifications were made accurately.

The hardcopy x-ray films were tested out by taking a photo of them using a phone camera and the images uploaded. The results were as follows:


Class: Sick, Confidence: 0.99

Class: Sick, Confidence: 0.99

Figure 4.4. 5 Hardcopy film images

Both images were classified as “Sick” with a very high confidence level. The images have glare from the surrounding windows and reflections of the person taking the photos. This resulted to incorrect classification of the healthy case and also the results had very high confidence levels. This was interpreted as an error.

CHAPTER FIVE

5.0 CONCLUSION AND RECOMMENDATION

Based on the results obtained, the following can be concluded:

5.1 CONCLUSION

- Kaggle website provided the necessary images required for model training.
- The data pre-processing procedures were successfully implemented to provide the necessary inputs for the CNN.
- For the model, the F1-score for class 0 is 0.79 and the F1-score for class 1 is 0.90. This means that the model is performing well at both identifying and predicting both the presence and absence of pneumonia which showcases a good performance according to F1-score guidelines.
- The model successfully classified the test images obtained from the hospital.
- The model was deployed using a website framework that provided a user-friendly interface for classifying images.

5.2 CHALLENGES

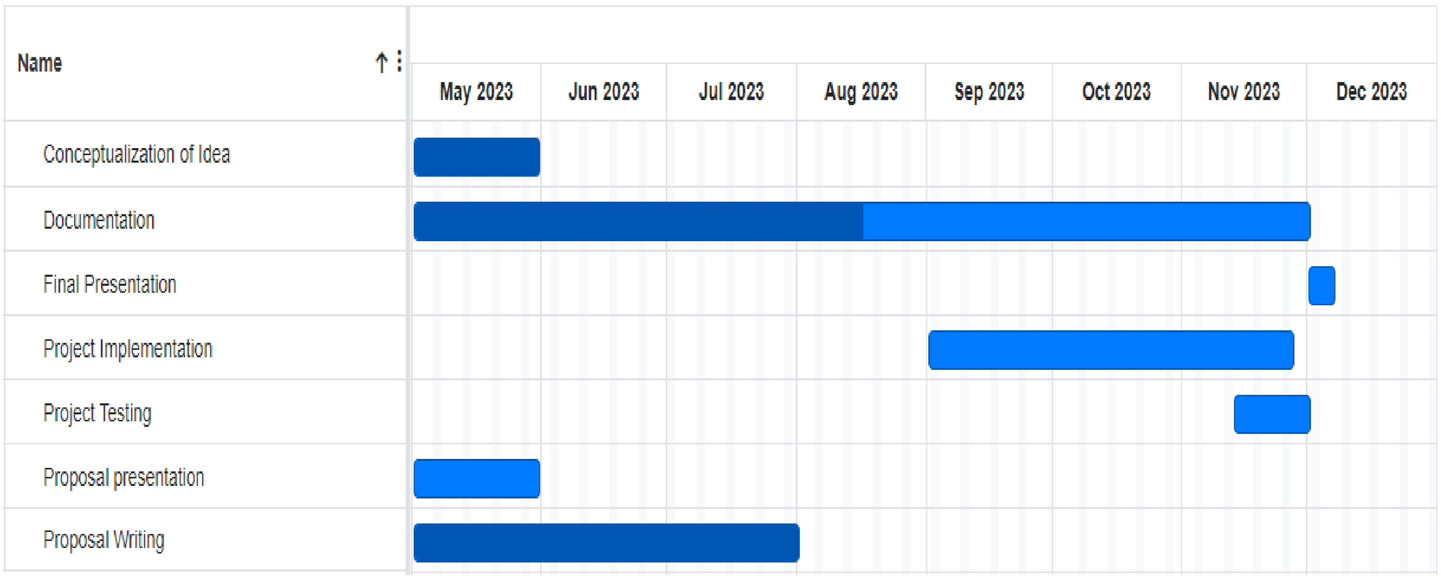
- Overfiitng while training the model using CPU
- Taking photos using a camera for the x-ray films resulted in incorrect results

5.3 RECOMMENDATIONS

- Using a GPU accelerated models as GPUs are the preferred choice for training image-based models due to their parallel architecture, high memory bandwidth, specialized hardware like tensor cores, and better efficiency for handling large models and complex computations. While CPUs can still be used for smaller models or simpler tasks, GPUs offer significant performance advantages for most practical applications in the realm of image-based deep learning.
- Using the ready softcopy x-ray images directly instead of printing the x-rays and later taking images of them. This is because the reflection on the x-ray films alter the films producing incorrect results.

PROJECT TIME PLAN

Table 5.0. 1 Project Time Plan



BUDGET

Table 5.0. 2 Proposed Budget

Item	Description	Quantity	Rate	Amount
1	X-ray films	2	500	1000
2	Miscellaneous		2500	2500
TOTAL				3500

REFERENCES

[1] R. H. D. B. Roser M., "Child and Infant Mortality," *Our World in Data,* 2019.

[2] "How Long Before Pneumonia Kills You?," NaoMedical, [Online]. Available: https://naomedical.com/how-long-before-pneumonia-kills-you/#:~:text=In%20general%2C%20it%20can%20take,fluid%20buildup%20around%20the%20lungs. . [Accessed 19 June 2023].

[3] B. M., "The NHS is trialling an AI chatbot to answer your medical questions.," 5 January 2017. [Online]. Available: http://www.wired.co.uk/article/babylon-nhs-chatbot-app. [Accessed 21 Jun 2023].

[4] A. Esteva, B. Kuprel, R. Novoa and e. al., "Dermatologist-level classification of skin cancer with deep neural networks," *Nature,* vol. 542, no. 7639, pp. 115-118, 2017.

[5] A. D. J. &. K. I. Rajkomar, "Machine learning in medicine," *The New England Journal of Medicine,* vol. 380, no. 14, pp. 1347-1358, 2019.

[6] A. K. B. N. R. K. J. S. S. B. H. &. T. S. Esteva, "Dermatologist-level classification of skin cancer with deep neural networks," *Nature,* vol. 542, no. 7639, pp. 115-118, 2017.

[7] F. W. S. W. R. Miotto, "Clinical prediction from electronic health records: Opportunities and challenges," *Journal of the American Medical Informatics Association,* vol. 25, no. 4, pp. 386-394, 2018.

[8] K. M. R. I. (KEMRI), "Research Themes," 2021. [Online]. Available: https://www.kemri.org/research-themes/. [Accessed 21 June 2023].

[9] R. O. &. H. P. Duda, Pattern classification and Scene analysis, 1973.

[10] G. W. D. H. T. &. T. R. James, "An introduction to statistical learning," 2013.

[11] I. B. Y. &. C. A. Goodfellow, "Deep learning," 2016.

[12] J. K. M. &. P. J. (. D. m. Han, "Data mining: Concepts and techniques," 2011.

[13] D. &. W. D. C. Xu, "Survey of clustering algorithms," *IEEE Transcations on neural networks,* vol. 16, no. 3, pp. 645-678, 2005.

[14] R. S. &. B. A. G. Sutton, "Reinforcement learning," in *An introduction*, MIT press, 2018.

[15] K. D. M. P. B. M. &. B. A. A. (. Arulkumaran, A brief survey of deep reinforcement learning, arXiv preprint arXiv , 2017.

[16] "An Ultimate Tutorial to Neural Networks," simplilearn, 20 July 2023. [Online]. Available: https://www.simplilearn.com/tutorials/deep-learning-tutorial/neural-network. [Accessed 30 November 2023].

[17] "Chest X-ray showing pneumonia," Mayo Clinic, [Online]. Available: https://www.mayoclinic.org/diseases-conditions/pneumonia/multimedia/chest-x-ray-showing-pneumonia/img-20005827. [Accessed 30 November 2023].

[18] "SafeMoon Hacker Agrees to Return 80% of Stolen Funds," Cointelegraph, 2020. [Online]. Available: https://cointelegraph.com/news/safemoon-hacker-agrees-to-return-80-of-stolen-funds-finance-redefined. [Accessed 21 June 2023].