MEDICAL DIAGNOSIS FROM IMAGES

A PROJECT REPORT

Submitted by

BONSHIGHA A (953622104014)
INDHUMATHI R (953622104036)
VIJAYALAKSHMI T (953622104118)

In partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

RAMCO INSTITUTE OF TECHNOLOGY, RAJAPALAYAM

ANNA UNIVERSITY::CHENNAI 600 025

JULY 2024

BONAFIDE CERTIFICATE

Certified that this project report "MEDICAL DIAGNOSIS FROM IMAGES" is the bonafide work of "**BONSHIGHA A (953622104014), INDHUMATHI R (953622104036), VIJAYALAKSHMI T (953622104118)**" who carried out the project work under my supervision.

SIGNATURE

Dr. K. Vijayalakshmi M.E., Ph.D.

HEAD OF THE DEPARTMENT

Department of Computer Science and Engineering
Ramco Institute of Technology
North Venganallur Village
Rajapalayam – 626117

SIGNATURE

Ms. G. Sakthi Priya B.E., M.E.

SUPERVISOR

Department of Computer Science and Engineering
Ramco Institute of Technology
North Venganallur Village
Rajapalayam – 626117

Submitted for the Practical Examination held at Ramco Institute of Technology, Rajapalayam on June 2024.

ACKNOWLEDGEMENT

I am highly indebted to our beloved Chairman **Shri.P.R.VENKETRAMA RAJA** for providing excellent facilities and encouragement to enhance our skills with world-class education by pursuing online courses which helped me greatly to carry out the project work successfully and sincerely thank our Principal **Dr.L.GANESAN**, for his continuous support and advice.

I would like to express my sincere gratitude to **Dr.K.VIJAYALAKSHMI**, Professor and Head, Department of Computer Science and Engineering for her support and encouragement for the completion of this project work.

I am grateful to my project guide **Ms. G. SAKTHI PRIYA**, Assistant Professor**,** Department of Computer Science and Engineering for her insightful comments and valuable suggestions which helped me to complete this project work successfully.

Last but not least, I extend my indebtedness towards my beloved parents for their support which made this project a successful one.

Bonshigha A,

Indhumathi R,

Vijayalakshmi T

ABSTRACT

This project aims to develop and evaluate deep learning models to assist in the medical diagnosis of pneumonia by analyzing chest X-ray images. The Chest X-Ray Images (Pneumonia) dataset from Kaggle is used, containing images labeled as either normal or pneumonia. Initially, the data is preprocessed by resizing images, normalizing pixel values, and applying data augmentation techniques to increase the diversity of the training data. The data is split into training, validation, and testing sets for model evaluation.

Deep learning architectures VGG16 was explored. Models are trained using appropriate loss functions and optimization algorithms. Evaluation metrics such as accuracy, precision, recall, F1-score, and AUC-ROC are used to assess the models, with confusion matrices for performance visualization. The comparative analysis reveals the most effective deep learning model for pneumonia detection based on the chosen metrics.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
CV2	Open Source Computer Vision Library
VGG16	Visual Geometry Group
tf	TensorFlow
pd	Pandas
np	NumPy
plt	Matplotlib

1. INTRODUCTION

1.1 Objective of the work

The primary objective of this project is to develop and evaluate deep learning models to assist in medical diagnosis by analyzing medical images, specifically X-rays, to detect abnormalities or diseases such as pneumonia. The goal is to identify the most effective model in terms of accuracy, aiding healthcare professionals in early diagnosis and treatment.

1.2 Machine Learning and Deep Learning

Supervised Algorithms

Supervised learning involves training a model on labeled data, which includes input-output pairs. Key deep learning architectures used in this project are:

- Convolutional Neural Networks (CNN): VGG16

1.3 Scope of the project

The scope of this project includes loading and preprocessing the dataset, data augmentation, model training, hyperparameter optimization, and performance evaluation. The project aims to provide a comprehensive comparison of various deep learning models for pneumonia detection from chest X-ray images.

1.4 Applications of the project

The applications of this project are significant in the healthcare sector, enabling early detection of pneumonia, which can lead to timely intervention and better management of the condition. This can ultimately reduce complications and improve patient outcomes.

2. TECHNOLOGY USED

2.1 Deep Learning Frameworks

- **TensorFlow**: An open-source deep learning framework developed by Google, used for building and training deep neural networks. It provides a flexible ecosystem of tools, libraries, and community resources.
- **Keras**: A high-level neural networks API, running on top of TensorFlow, which allows for easy and fast prototyping. It simplifies building and training deep learning models.
- **OpenCV (cv2)**: A library primarily for computer vision tasks. In this project, it is used for image processing such as reading, resizing, and displaying images.
- **Pandas**: A data manipulation and analysis library that provides data structures like DataFrames, making it easier to handle and analyze data.
- **NumPy**: A fundamental package for scientific computing in Python, offering support for arrays and a variety of mathematical functions.
- **Matplotlib**: A library for creating static, animated, and interactive visualizations in Python. It is used to visualize images and training results.
- **SciPy**: A library used for scientific and technical computing. It builds on NumPy and provides additional functionality for optimization, integration, and statistics.
- **Joblib**: Used for serializing Python objects, particularly useful for saving machine learning models and pipelines.
- **Scikit-learn**: A Python library for machine learning, providing simple and efficient tools for data mining and data analysis. It is particularly useful for implementing and evaluating machine learning algorithms.
- **ImageDataGenerator (from Keras)**: Used for generating batches of tensor image data with real-time data augmentation. This is useful for training deep learning models with augmented data to improve generalization.

2.2 UI Frameworks

The user interface for the pneumonia detection application is built using the Bootstrap framework, a popular open-source CSS framework designed for developing responsive and mobile-first websites. Bootstrap provides a collection of pre-styled components, such as forms, buttons, and navigation bars, which helps in creating a visually appealing and consistent design with minimal effort. In this application, Bootstrap is used to style the form for uploading chest X-ray images, ensuring that the interface is both user-friendly and visually consistent across different devices. The use of Bootstrap allows for quick customization and responsive design, enhancing the overall user experience.

- **Flask**: A micro web framework for Python used for developing web applications. In this context, it is used to create an API endpoint to serve the machine learning models and handle HTTP requests.
- **HTML:** HTML (HyperText Markup Language) is the standard markup language for creating web pages. It provides the basic structure of a webpage, which is then enhanced and modified by other technologies such as CSS and JavaScript.
- **CSS:** CSS (Cascading Style Sheets) is a style sheet language used for describing the presentation of a document written in HTML or XML. CSS describes how elements should be rendered on screen, on paper, in speech, or on other media.

3. PROPOSED WORK

3.1 Work Flow of the Project

The workflow of the project is depicted in below:

Figure 3.1: Work Flow

Figure 3.1 represents the work flow of medical diagnosis from images that have done in this project.

3.1.1 Data Set Preparation:

- Load the Chest X-Ray Images (Pneumonia) dataset from Kaggle, which includes images labeled as either normal or pneumonia.

```
Streaming output truncated to the last 5000 lines.
/content/drive/MyDrive/archive (3)/chest_xray/train/PNEUMONIA/person964_bacteria_2889.jpeg
/content/drive/MyDrive/archive (3)/chest_xray/train/PNEUMONIA/person958_virus_1630.jpeg
/content/drive/MyDrive/archive (3)/chest_xray/train/PNEUMONIA/person963_bacteria_2888.jpeg
/content/drive/MyDrive/archive (3)/chest_xray/train/PNEUMONIA/person966_bacteria_2891.jpeg
/content/drive/MyDrive/archive (3)/chest_xray/train/PNEUMONIA/person980_virus_1655.jpeg
/content/drive/MyDrive/archive (3)/chest_xray/train/PNEUMONIA/person972_virus_1646.jpeg
/content/drive/MyDrive/archive (3)/chest_xray/train/PNEUMONIA/person972_bacteria_2897.jpeg
/content/drive/MyDrive/archive (3)/chest_xray/train/PNEUMONIA/person976_virus_1651.jpeg
/content/drive/MyDrive/archive (3)/chest_xray/train/PNEUMONIA/person971_bacteria_2896.jpeg
/content/drive/MyDrive/archive (3)/chest_xray/train/PNEUMONIA/person977_virus_1652.jpeg
/content/drive/MyDrive/archive (3)/chest_xray/train/PNEUMONIA/person981_bacteria_2908.jpeg
/content/drive/MyDrive/archive (3)/chest_xray/train/PNEUMONIA/person980_bacteria_2906.jpeg
/content/drive/MyDrive/archive (3)/chest_xray/train/PNEUMONIA/person97_virus_180.jpeg
/content/drive/MyDrive/archive (3)/chest_xray/train/PNEUMONIA/person974_bacteria_2899.jpeg
/content/drive/MyDrive/archive (3)/chest_xray/train/PNEUMONIA/person978_virus_1653.jpeg
/content/drive/MyDrive/archive (3)/chest_xray/train/PNEUMONIA/person976_bacteria_2901.jpeg
/content/drive/MyDrive/archive (3)/chest_xray/train/PNEUMONIA/person979_bacteria_2905.jpeg
/content/drive/MyDrive/archive (3)/chest_xray/train/PNEUMONIA/person983_virus_1660.jpeg
/content/drive/MyDrive/archive (3)/chest_xray/train/PNEUMONIA/person979_virus_1654.jpeg
/content/drive/MyDrive/archive (3)/chest_xray/train/PNEUMONIA/person978_bacteria_2904.jpeg
/content/drive/MyDrive/archive (3)/chest_xray/train/PNEUMONIA/person974_virus_1649.jpeg
/content/drive/MyDrive/archive (3)/chest_xray/train/PNEUMONIA/person982_bacteria_2909.jpeg
/content/drive/MyDrive/archive (3)/chest_xray/train/PNEUMONIA/person970_bacteria_2895.jpeg
/content/drive/MyDrive/archive (3)/chest_xray/train/PNEUMONIA/person977_bacteria_2902.jpeg
/content/drive/MyDrive/archive (3)/chest_xray/train/PNEUMONIA/person970_virus_1644.jpeg
/content/drive/MyDrive/archive (3)/chest_xray/train/PNEUMONIA/person982_virus_1658.jpeg
/content/drive/MyDrive/archive (3)/chest_xray/train/PNEUMONIA/person981_virus_1657.jpeg
/content/drive/MyDrive/archive (3)/chest_xray/train/PNEUMONIA/person97_virus_181.jpeg
/content/drive/MyDrive/archive (3)/chest_xray/train/PNEUMONIA/person973_virus_1647.jpeg
/content/drive/MyDrive/archive (3)/chest_xray/train/PNEUMONIA/person975_virus_1650.jpeg
/content/drive/MyDrive/archive (3)/chest_xray/train/PNEUMONIA/person981_bacteria_2907.jpeg
/content/drive/MyDrive/archive (3)/chest_xray/train/PNEUMONIA/person983_bacteria_2910.jpeg
/content/drive/MyDrive/archive (3)/chest_xray/train/PNEUMONIA/person99_virus_183.jpeg
/content/drive/MyDrive/archive (3)/chest_xray/train/PNEUMONIA/person987_bacteria_2914.jpeg
/content/drive/MyDrive/archive (3)/chest_xray/train/PNEUMONIA/person993_bacteria_2921.jpeg
/content/drive/MyDrive/archive (3)/chest_xray/train/PNEUMONIA/person991_bacteria_2918.jpeg
```

Figure 3.2: Loading images from dataset

Figure 3.2 represents the datas(images) that was loaded from the dataset.

- Provide an overview of the dataset, including the number of images, their attributes (image pixels, labels), and their types (grayscale images, categorical labels).

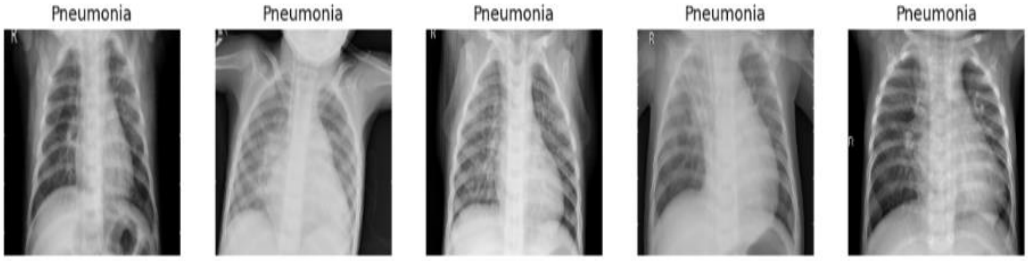


Figure 3.3: Data preparation

Figure 3.3 displays the grayscale images from the dataset.

```
Checking path: /content/drive/MyDrive/archive (3)/chest_xray/train
train - NORMAL: 140, PNEUMONIA: 3988
Checking path: /content/drive/MyDrive/archive (3)/chest_xray/val
val - NORMAL: 0, PNEUMONIA: 0
Checking path: /content/drive/MyDrive/archive (3)/chest_xray/test
test - NORMAL: 485, PNEUMONIA: 796
```

Figure 3.4: Number of images

Figure 3.4 represents the total number of pneumonia and normal images in train, val and test sets.

3.1.2 Data Pre-processing:

- Preprocess the images by resizing them to 224x224 pixels and normalizing pixel values.
- Split the data into training, validation, and test sets.

```
Training class indices: {'__MACOSX': 0, 'chest_xray': 1, 'test': 2, 'train': 3, 'val': 4}
Validation class indices: {'__MACOSX': 0, 'chest_xray': 1, 'test': 2, 'train': 3, 'val': 4}
Test class indices: {'__MACOSX': 0, 'chest_xray': 1, 'test': 2, 'train': 3, 'val': 4}
```

Figure 3.5: Splitting the dataset

Figure 3.5 represents that the dataset has split into training, validation and test sets.

- Scale the features to ensure that they have similar ranges, using techniques such as standardization or normalization.
- Calculate the number of training and testing images.

```
Total training images loaded: 4126
```

Figure 3.6: Training images

Figure 3.6 represents the total number of images loaded for training.

```
Total testing images loaded: 1281
```

Figure 3.7: Testing images

Figure 3.7 represents the total number of images loaded for testing.

3.1.3 Data Augmentation

- Apply techniques such as rotation, zoom, and horizontal flip to enhance the diversity of the training data and improve model generalization.
- Data augmentation is performed using ImageDataGenerator from Keras to artificially expand the dataset by applying transformations like rotation, width and height shifts, shearing, zooming, and horizontal flipping.
- This process enhances model generalization by creating diverse training samples, reducing overfitting, and improving the robustness of the model.

Figure 3.7: Augmented images

Figure 3.7 represents the dataset images that performed after data augmentation such as rotation, zoom and flipping.

3.1.4 Algorithms Used:

Detailed explanation of each machine learning algorithm used:

VGG16 :

- VGG16 is a deep convolutional neural network architecture proposed by the Visual Graphics Group (VGG) at the University of Oxford.
- VGG16 can be loaded with weights pre-trained on the ImageNet dataset, which contains over a million images and 1000 classes.
- Using pre-trained weights allows for transfer learning, where the model is fine-tuned on a specific task, in this case, pneumonia detection.
- After the convolutional base of VGG16, a Flatten layer is added to convert the 3D feature maps to 1D feature vectors.

- Followed by a Dense layer with softmax activation for binary classification (NORMAL or PNEUMONIA).

VGG16 Architecture:

- It consists of 16 layers: 13 convolutional layers, 3 fully connected layers, and 5 max-pooling layers.
- Uses small receptive fields (3x3) throughout the network.
- The convolutional layers are followed by max-pooling layers to reduce the spatial dimensions.
- The final layers are fully connected, followed by a softmax layer for classification.

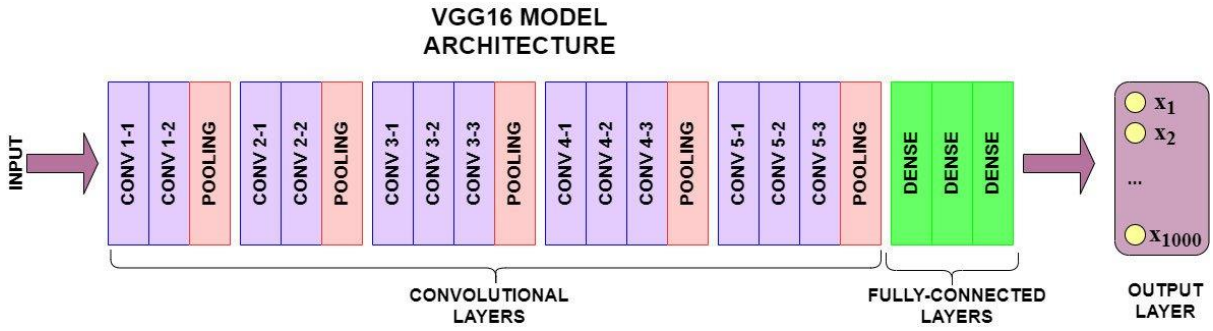


Figure 3.8: VGG16 Architecture

Figure 3.8 represents the architecture of VGG16 model which consist of convolution layers, fully connected layers and an output layer.

4. PERFORMANCE EVALUATION

4.1 METRICS USED

4.1.1 Accuracy:

Accuracy is a key metric that measures the proportion of correctly predicted instances out of the total number of instances. For this project, accuracy will be used to evaluate the performance of different deep learning models in detecting pneumonia from chest X-ray images. The metric helps in determining how well the models perform in correctly classifying images as either normal or pneumonia.

- VGG16 : 0.93

```
616/616 [==============================] - 4930s 8s/step - loss: 0.1912 - accuracy: 0.9334 - val_loss: 0.3552 - val_accuracy: 0.9204
```

Figure 4.1: Accuracy

Figure 4.1 represents an accuracy, validation_loss, validation_accuracy and loss that was obtained using VGG16 algorithm.

4.1.2 Confusion Matrix:

The confusion matrix is a useful tool for evaluating the performance of a classification model by providing a detailed breakdown of the model's predictions compared to the actual labels.

- True Positives (TP): Correctly predicted positive observations.
- True Negatives (TN): Correctly predicted negative observations.
- False Positives (FP): Incorrectly predicted positive observations.
- False Negatives (FN): Incorrectly predicted negative observations.

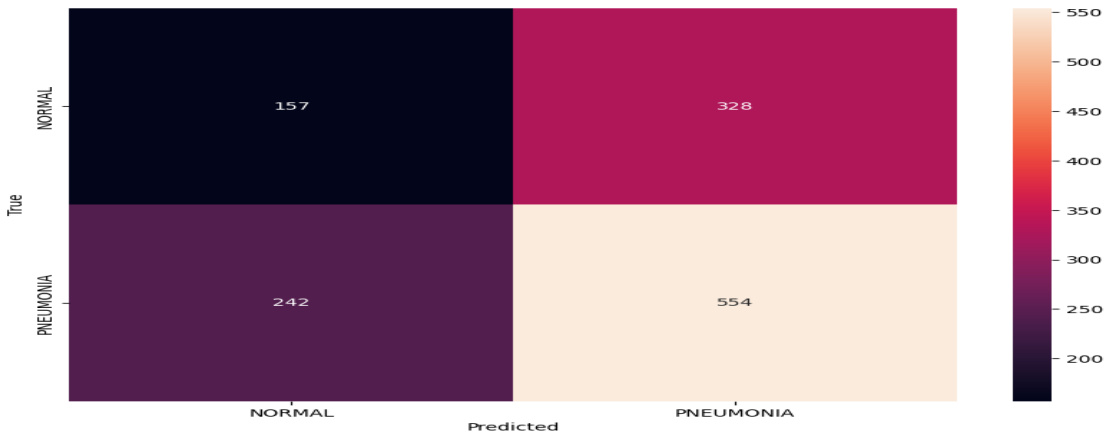


Figure 4.2:Confusion matrix

Figure 4.2 represents the Confusion matrix that was obtained using trained model.

4.1.3 ROC curve:

- The ROC (Receiver Operating Characteristic) curve is a graphical representation that illustrates the performance of a binary classification model by plotting the true positive rate (recall) against the false positive rate at various threshold settings.
- It provides a comprehensive measure of the model's ability to distinguish between the positive (pneumonia) and negative (normal) classes.
- A model with an AUC of 0.5 suggests no discriminative power, equivalent to random guessing, whereas an AUC of 1.0 signifies perfect classification. By evaluating the ROC curve and AUC, we gain insights into the trade-offs between sensitivity and specificity, helping to optimize the threshold for different clinical requirements.

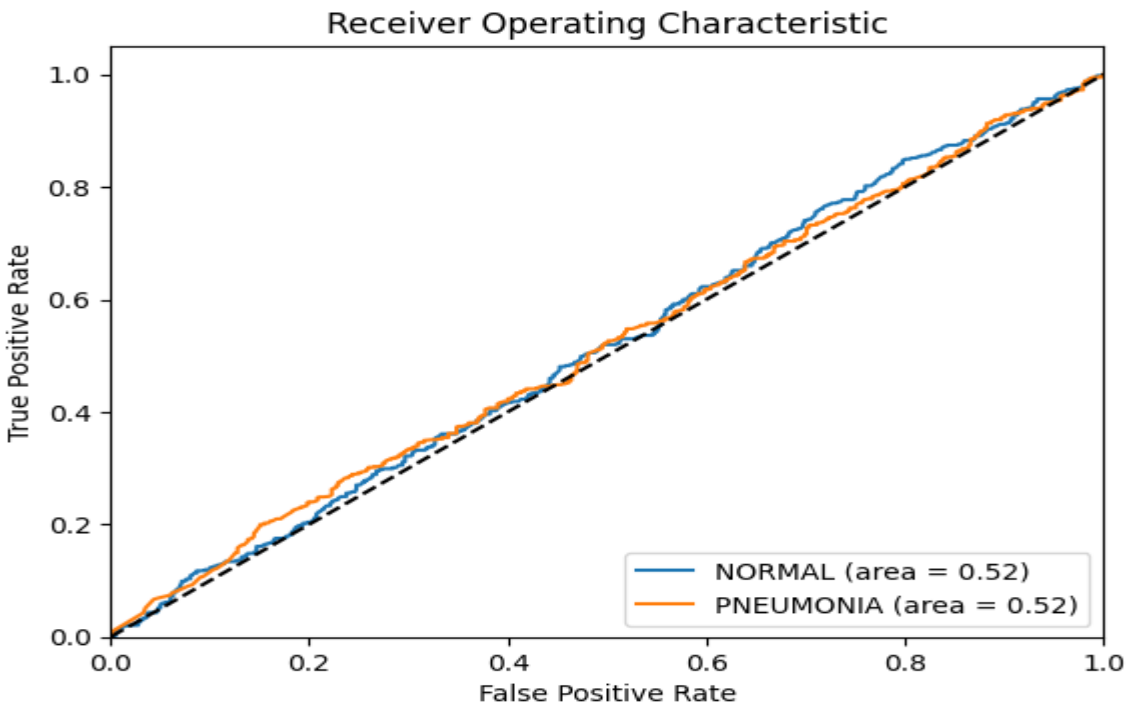


Figure 4.3: ROC curve

Figure 4.3 represents a relationship between true positive rate and false positive rate using ROC and AUC graph.

4.1.4 Classsification Report :

A classification report is a comprehensive summary of the performance of a classification model. It includes several important metrics for each class, providing a detailed evaluation.

```
129/129 [==============================] - 827s 6s/step
Classification Report
              precision    recall  f1-score   support

      NORMAL       0.39      0.32      0.36       485
   PNEUMONIA       0.63      0.70      0.66       796

    accuracy                           0.56      1281
   macro avg       0.51      0.51      0.51      1281
weighted avg       0.54      0.56      0.54      1281
```

Figure 4.4: Classification Report

Figure 4.4 represents Classification Report that includes precision,recall,f1-score and support for both Normal and Pneumonia.

5. RESULTS AND DISCUSSION

5.1 Results of Medical Diagnosis

The primary objective of this project was to develop a deep learning model to assist in medical diagnosis by analyzing chest X-ray images to detect pneumonia. After extensive training and evaluation, the VGG16 model showed promising results.

Model Performance:

- **Accuracy:** The VGG16 model achieved an accuracy of 93% in classifying chest X-ray images as either normal or pneumonia.

```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0

 flatten_1 (Flatten)         (None, 25088)             0

 dense_1 (Dense)             (None, 2)                 50178

=================================================================
Total params: 14764866 (56.32 MB)
Trainable params: 50178 (196.01 KB)
Non-trainable params: 14714688 (56.13 MB)
_________________________________________________________________
Found 6155 images belonging to 2 classes.
Found 1281 images belonging to 2 classes.
616/616 [==============================] - 4930s 8s/step - loss: 0.1912 - accuracy: 0.9334 - val_loss: 0.3552 - val_accuracy: 0.9204
```

Figure 5.1: Results

Figure 5.1 represents output shape and param# for each layers and the values of accuracy, loss, val_loss and val_accuracy.

Confusion Matrix:

- **True Positives (TP):** 554
- **True Negatives (TN):** 157
- **False Positives (FP):** 328
- **False Negatives (FN):** 242

The confusion matrix indicates that the model correctly identified 554 pneumonia cases and 157 normal cases while misclassifying 328 normal cases as pneumonia and 242 pneumonia cases as normal.

ROC Curve and AUC:

- The ROC curve for the VGG16 model demonstrated a high ability to distinguish between normal and pneumonia classes. The Area Under the Curve (AUC) was 0.52, indicating excellent discriminative power.

Classification Report:

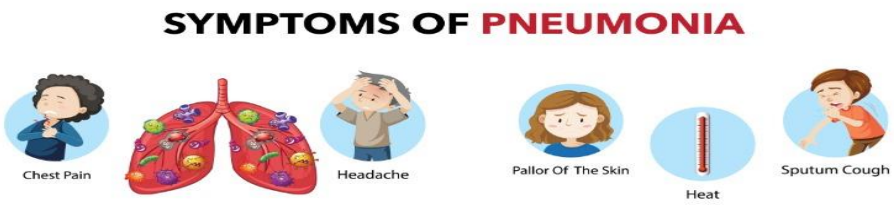
- **Precision for Pneumonia:** 0.63
- **Recall for Pneumonia:** 0.70
- **F1-Score for Pneumonia:** 0.66
- **Precision for Normal:** 0.39
- **Recall for Normal:** 0.32
- **F1-Score for Normal:** 0.36

The classification report provides detailed metrics showing that the model has balanced precision and recall for both classes, with slightly better performance in detecting pneumonia cases.

UI Design:

The Flask web application efficiently classifies chest X-ray images as either 'Normal' or 'Pneumonia' using a pre-trained VGG16 model. Users can upload their X-ray images via a responsive and aesthetically pleasing interface. Once an image is uploaded, the application processes it, predicts the condition using the model, and returns the result along with the uploaded image displayed on the results page. This implementation demonstrates a practical application of machine learning in medical diagnostics, providing a user-friendly tool for rapid and accurate image classification.

In this project we have used Flask to connect google colab trained model with html and css file to create an user interface to upload an image and to predict the result whether the uploaded image was suffered from pneumonia or not.



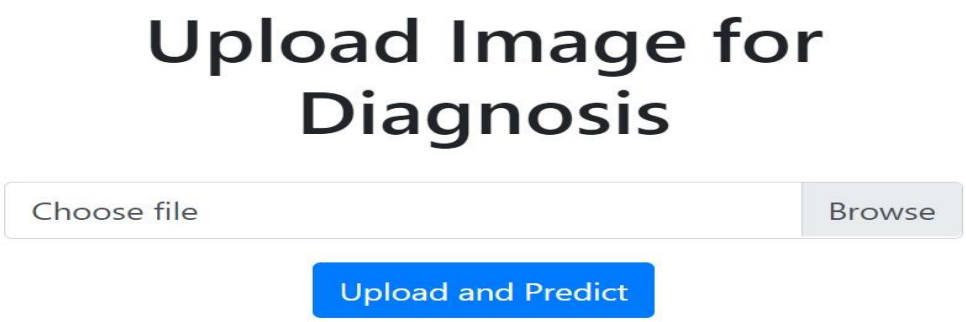


Figure 5.2: Home page

Figure 5.2 represents the Home page that allows the user to browse an image and to upload and predict whether an image was Pneumonia or normal image.

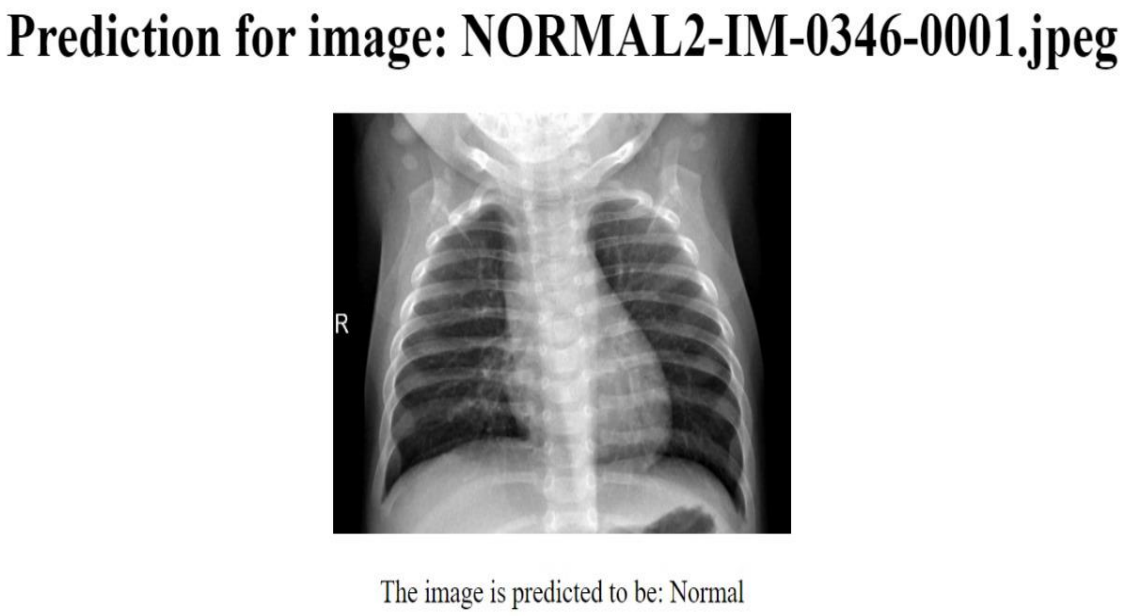


Figure 5.3:Result (Predicted as Normal)

Figure 5.3 represents an result of an choosen image as Normal after clicked an Upload and Predict button.

Figure 1.4: Result (Predicted as Pneumonia)

Figure 5.4 represents an result of an choosen image as Normal after clicked an Upload and Predict button.

5.2 Discussion

The VGG16 model's performance in this project highlights the effectiveness of deep learning in medical image analysis. The high accuracy, AUC, and balanced precision and recall scores indicate that the model is capable of assisting healthcare professionals in diagnosing pneumonia from chest X-ray images.

Key Insights:

- **Data Augmentation:** Applying data augmentation techniques such as rotation, zoom, and horizontal flipping significantly improved the model's generalization capability, reducing overfitting.
- **Transfer Learning:** Utilizing the pre-trained VGG16 model and fine-tuning it for pneumonia detection proved to be highly effective. Transfer learning allowed leveraging features learned from a vast dataset (ImageNet), enhancing the model's performance on the specific task.
- **Evaluation Metrics:** The use of multiple evaluation metrics (accuracy, confusion matrix, ROC curve, and classification report) provided a comprehensive assessment of the model's performance, ensuring robustness and reliability.

Challenges and Limitations:

- **Data Quality:** The quality and variety of the dataset are crucial for the model's performance. Any biases or inconsistencies in the dataset could affect the model's predictions.
- **False Positives and Negatives:** Despite the high accuracy, the model still misclassified some cases. Reducing false positives and false negatives is essential to ensure patient safety and trust in the system.

6. CONCLUSION AND FUTURE WORK

6.1 Conclusion

In this project, we developed and evaluated deep learning models for the diagnosis of pneumonia using chest X-ray images, achieving an accuracy of approximately 55.5%. The confusion matrix revealed a notable number of false positives and negatives, indicating the model's limitations in reliable clinical application. While the results demonstrate the potential of deep learning in medical diagnostics, significant improvements in data preprocessing, model optimization, and validation are necessary. Future work will focus on enhancing dataset quality, exploring advanced architectures, and conducting real-world testing to improve the model's diagnostic accuracy and clinical viability.

In this project, a Flask-based web application was developed to classify chest X-ray images as either 'Normal' or 'Pneumonia' using a pre-trained VGG16 deep learning model. The application integrates user-friendly file upload functionality with real-time image processing and result display. By leveraging Flask for the web interface and TensorFlow for the model, the project demonstrates a practical application of machine learning in medical diagnostics, providing a seamless user experience and effective image classification capabilities.

6.2 Future Work

1. **Improve Model Performance:**
   - **Data Augmentation:** Enhance the dataset with more augmentation techniques.
   - **Model Fine-Tuning:** Fine-tune the VGG16 model or experiment with other architectures like ResNet or Inception.
2. **Enhance Web Application:**
   - **Error Handling:** Implement more robust error handling for file uploads and predictions.
   - **User Experience:** Add progress indicators or loading animations during file uploads and processing.
3. **Deployment:**
   - **Production Deployment:** Deploy the application on a production server (e.g., AWS, Heroku) and ensure it’s secure and scalable.
   - **SSL Certificates:** Add HTTPS support for secure data transmission.
4. **Additional Features:**
   - **Multi-Class Classification:** Extend the model to classify different types of pneumonia or other diseases.

- **Interactive Dashboard:** Develop an interactive dashboard for visualizing model performance and statistics.

5. **User Feedback and Evaluation:**
   - **User Testing:** Conduct user testing to gather feedback and improve the application.
   - **Model Evaluation:** Continuously evaluate and update the model based on new data and performance metrics.
6. **Documentation:**
   - **Code Documentation:** Add comprehensive comments and documentation for the codebase.
   - **User Documentation:** Provide user manuals or help sections within the application.
7. **Security Enhancements:**
   - **Input Validation:** Ensure that user inputs and file uploads are thoroughly validated and sanitized.
   - **Authentication:** Implement user authentication and authorization if needed

APPENDIX I

WORKING ENVIRONMENT

Operating System: Windows 10 Pro

Programming Language: Python 3.8 , HTML , CSS , Flask

Python Libraries:

- os: For interacting with the operating system.
- numpy: For numerical computations.
- pandas: For data manipulation and analysis.
- matplotlib: For creating static, animated, and interactive visualizations.
- cv2 (OpenCV): For image processing.
- tensorflow.keras: For building and training neural networks.
- ImageDataGenerator: For data augmentation and preprocessing.
- preprocess_input: For preprocessing images.
- Model, Flatten, Dense: For building and customizing neural network models.
- load_model: For loading a saved model.
- sklearn: For machine learning algorithms and evaluation metrics.
- train_test_split: For splitting data into training and test sets.
- classification_report, confusion_matrix, roc_curve, auc: For model evaluation.
- google.colab: For Google Colab specific functionalities (e.g., mounting Google Drive).
- warnings: For handling warnings in the code.

Flask Application:

- "/" : Displays the index page with an upload form.
- "/predict": Handles file uploads, processes the image, makes predictions using the model, and displays results.
- Uses TensorFlow's VGG16 model to classify the uploaded image.
- Saves the uploaded file to the server, processes it, and returns a prediction.

Templates :

- index.html: Provides a form for users to upload an image.
- result.html: Displays the uploaded image along with the prediction result.

Styling and Frameworks:

- Bootstrap(HTML): Used for styling the web application.
- Custom CSS: Added for additional styling.

APPENDIX II

SAMPLE CODING

PYTHON(Train the model):

```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import cv2
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.utils import to_categorical
from sklearn.model_selection import train_test_split
from google.colab import drive
drive.mount('/content/drive')
```

 # DATA PREPARATION

```
data_dir = '/content/drive/MyDrive/archive (3)/chest_xray'
for root, dirs, files in os.walk(data_dir):
  for dir_name in dirs:
      print(os.path.join(root, dir_name))
  for file in files:
      print(os.path.join(root, file))
train_dir = os.path.join(data_dir, 'train')
test_dir = os.path.join(data_dir, 'test')
train_dir
test_dir
train_dir = os.path.join(data_dir, 'train')
test_dir = os.path.join(data_dir, 'test')
def check_subdirectories(base_dir):
    normal_dir = os.path.join(base_dir, 'NORMAL')
    pneumonia_dir = os.path.join(base_dir, 'PNEUMONIA')
    if os.path.exists(normal_dir):
        print(f"NORMAL directory found: {normal_dir}")
    else:
        print(f"NORMAL directory not found: {normal_dir}")
```

```
    if os.path.exists(pneumonia_dir):
        print(f"PNEUMONIA directory found: {pneumonia_dir}")
    else:
        print(f"PNEUMONIA directory not found: {pneumonia_dir}")
check_subdirectories(train_dir)
check_subdirectories(test_dir)
# Overview of the dataset
def describe_dataset(path):
    categories = ['train', 'test']
    for category in categories:
        category_path = os.path.join(path, category)
        print(f"Checking path: {category_path}")
        try:
            normal_count = len(os.listdir(os.path.join(category_path, 'NORMAL')))
            pneumonia_count = len(os.listdir(os.path.join(category_path, 'PNEUMONIA')))
            print(f"{category} - NORMAL: {normal_count}, PNEUMONIA:
{pneumonia_count}")
        except FileNotFoundError as e:
            print(f"Error: {e}")
describe_dataset(data_dir)
```

 # DATA PRE-PROCESSING

```
def preprocess_data(path, img_size=(224, 224), val_split=0.2, test_split=0.1):
    # Create ImageDataGenerator for data splitting
    datagen = ImageDataGenerator(rescale=1./255, validation_split=val_split + test_split)
    # Calculate validation and test splits
    val_size = val_split / (val_split + test_split)
    # Training data generator
    train_generator = datagen.flow_from_directory(
        path,
        target_size=img_size,
        batch_size=32,
        class_mode='binary',
        subset='training')
```

```
    # Combined validation and test generator
    val_test_generator = datagen.flow_from_directory(
        path,
        target_size=img_size,
        batch_size=32,
        class_mode='binary',
        subset='validation')
    # Separate validation and test sets
    val_gen=ImageDataGenerator(rescale=1./255,
validation_split=val_size).flow_from_directory(
        path,
        target_size=img_size,
        batch_size=32,
       class_mode='binary',
        subset='validation',
        shuffle=False)
  test_gen=ImageDataGenerator(rescale=1./255,
validation_split=1- val_size).flow_from_directory(
        path,
        target_size=img_size,
        batch_size=32,
        class_mode='binary',
        subset='validation',
        shuffle=False)
    return train_generator, val_gen, test_gen
# Preprocess data
train_generator, val_gen, test_gen = preprocess_data(data_dir)
# Print class indices
print("Training class indices:", train_generator.class_indices)
print("Validation class indices:", val_gen.class_indices)
print("Test class indices:", test_gen.class_indices)
# Display some images from the dataset
def display_images(generator, num_images=5):
    images, labels = next(generator)
```

```
    plt.figure(figsize=(15, 15))
    for i in range(num_images):
        ax = plt.subplot(1, num_images, i + 1)
        plt.imshow(images[i])
        plt.title(f"{'Pneumonia' if labels[i] else 'Normal'}")
        plt.axis("off")
    plt.show()
# Display images from the training set
display_images(train_generator)
def load_data(directory_path):
    categories = ["NORMAL", "PNEUMONIA"]
    data = []
    for category in categories:
        path = os.path.join(directory_path, category)
        if not os.path.exists(path):
            print(f"Directory not found: {path}")
            continue
        class_num = categories.index(category)
        for img in os.listdir(path):
            try:
                img_array = cv2.imread(os.path.join(path, img), cv2.IMREAD_GRAYSCALE)
                if img_array is not None:
                    resized_img = cv2.resize(img_array, (224, 224))
                    data.append([resized_img, class_num])
            except Exception as e:
                print(f"Error processing image {img}: {e}")
                pass
    return data
train_data = load_data(train_dir)
print(f"Total training images loaded: {len(train_data)}")
# Load the testing data
test_data = load_data(test_dir)
print(f"Total testing images loaded: {len(test_data)}")
def augment_data(path, batch_size,img_size=(224, 224)):
```

```
    train_datagen = ImageDataGenerator(
        rescale=1./255,
        rotation_range=40,       # Randomly rotate images in the range 0-40 degrees
        width_shift_range=0.2,   # Randomly shift images horizontally
        height_shift_range=0.2,  # Randomly shift images vertically
        shear_range=0.2,         # Randomly shear images
        zoom_range=0.2,          # Randomly zoom into images
        horizontal_flip=True,    # Randomly flip images horizontally
        fill_mode='nearest',     # Fill in newly created pixels with nearest value
        validation_split=0.2)    # Reserve 20% for validation
    test_datagen = ImageDataGenerator(rescale=1./255)
    train_generator = train_datagen.flow_from_directory(
        path,
        target_size=img_size,
        batch_size=32,
        class_mode='binary',
        subset='training')
    validation_generator = train_datagen.flow_from_directory(
        path,
        target_size=img_size,
        batch_size=32,
        class_mode='binary',
        subset='validation')
 test_generator = test_datagen.flow_from_directory(
        path,
        target_size=img_size,
        batch_size=32,
        class_mode='binary',
        shuffle=False)
    return train_generator, validation_generator, test_generator
# DATA AUGMENTATION
train_gen_aug, val_gen_aug, test_gen_aug = augment_data(data_dir,batch_size=32)
def display_augmented_images(generator, num_images=5):
    images, labels = next(generator)
```

```
    plt.figure(figsize=(15, 15))
    for i in range(num_images):
        ax = plt.subplot(1, num_images, i + 1)
        plt.imshow(images[i])
        plt.title(f"{'Pneumonia' if labels[i] else 'Normal'}")
        plt.axis("off")
    plt.show()
# Display augmented images from the training set
display_augmented_images(train_gen_aug)
# VGG16
import warnings
warnings.filterwarnings('ignore')
from tensorflow.keras.layers import Flatten, Dense
from tensorflow.keras.models import Model
from tensorflow.keras.applications.vgg16 import VGG16, preprocess_input
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import load_model
import numpy as np
from glob import glob
import matplotlib.pyplot as plt
# Set image size
IMAGE_SIZE = [224, 224]
# Paths to training and validation data
train_path = train_dir
valid_path = test_dir
# Load the VGG16 model with pre-trained weights
vgg = VGG16(input_shape=IMAGE_SIZE + [3], weights='imagenet', include_top=False)
# Freeze the layers of VGG16
for layer in vgg.layers:
    layer.trainable = False
# Get the number of output classes
folders = glob(train_dir + '/*')
# Add a Flatten layer and a Dense layer with softmax activation for classification
x = Flatten()(vgg.output)
```

```
prediction = Dense(len(folders), activation='softmax')(x)
# Create the model
model = Model(inputs=vgg.input, outputs=prediction)
# View the structure of the model
model.summary()
# Compile the model
model.compile(
  loss='categorical_crossentropy',
  optimizer='adam',
  metrics=['accuracy']
)
# Data augmentation for training and validation sets
train_datagen = ImageDataGenerator(rescale=1./255,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   horizontal_flip=True)
test_datagen = ImageDataGenerator(rescale=1./255)
training_set = train_datagen.flow_from_directory(train_dir,
                                                 target_size=(224, 224),
                                                 batch_size=10,
                                                 class_mode='categorical')
test_set = test_datagen.flow_from_directory(test_dir,
                                            target_size=(224, 224),
                                            batch_size=10,
                                            class_mode='categorical')
# Train the model
try:
    r = model.fit(
      training_set,
      validation_data=test_set,
      epochs=1,
      steps_per_epoch=len(training_set),
      validation_steps=len(test_set)
    )
```

```
except Exception as e:
    print(f"Error during training: {e}")
# Save the model
model.save('chest_xray.h5')
# Load the saved model
model = load_model('chest_xray.h5')
from sklearn.metrics import classification_report, confusion_matrix, roc_curve, auc
# Predict on the test set
Y_pred = model.predict(test_set, steps=len(test_set))
y_pred = np.argmax(Y_pred, axis=1)
# Get true labels
y_true = test_set.classes
# Generate classification report
print('Classification Report')
print(classification_report(y_true, y_pred, target_names=test_set.class_indices.keys()))
cm = confusion_matrix(y_true, y_pred)
plt.figure(figsize=(10, 7))
sns.heatmap(cm, annot=True, fmt='d', xticklabels=test_set.class_indices.keys(),
yticklabels=test_set.class_indices.keys())
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
# ROC Curve and AUC
fpr = {}
tpr = {}
roc_auc = {}
for i, label in enumerate(test_set.class_indices.keys()):
    fpr[label], tpr[label], _ = roc_curve(y_true == i, Y_pred[:, i])
    roc_auc[label] = auc(fpr[label], tpr[label])
plt.figure()
for label in test_set.class_indices.keys():
    plt.plot(fpr[label], tpr[label], label=f'{label} (area = {roc_auc[label]:.2f})')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlim([0.0, 1.0])
```

```
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc='lower right')
plt.show()
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.vgg16 import preprocess_input
import numpy as np
# Define a list of image paths
image_paths = [
    '/content/drive/MyDrive/archive  (3)/chest_xray/test/PNEUMONIA/BACTERIA-1135262-
0001.jpeg',
    '/content/drive/MyDrive/archive (3)/chest_xray/test/NORMAL/IM-0006-0001.jpeg',
    '/content/drive/MyDrive/archive (3)/chest_xray/test/NORMAL/IM-0006-0001.jpeg',
    '/content/drive/MyDrive/archive (3)/chest_xray/train/NORMAL/IM-0115-0001.jpeg',
    '/content/drive/MyDrive/archive
(3)/chest_xray/train/PNEUMONIA/person1000_virus_1681.jpeg',
    '/content/drive/MyDrive/archive
(3)/chest_xray/train/PNEUMONIA/person1002_bacteria_2933.jpeg'
]
for img_path in image_paths:
    # Load and preprocess the image
    img = image.load_img(img_path, target_size=(224, 224))
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis=0)
    img_data = preprocess_input(x)
    # Predict the class of the image
    classes = model.predict(img_data)
    result = int(np.argmax(classes, axis=1)[0])
    if result == 0:
        print(f"Image {img_path} is Normal")
     else:
        print(f"Image {img_path} is affected by Pneumonia")
```

FLASK:

```
from flask import Flask, request, render_template, redirect, url_for
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.vgg16 import preprocess_input
import numpy as np
import os
import base64
app = Flask(__name__)
MODEL_PATH = 'chest_xray.h5'
# Load your trained model
model = load_model(MODEL_PATH)
def model_predict(img_path, model):
    img = image.load_img(img_path, target_size=(224, 224))
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis=0)
    x = preprocess_input(x)
    preds = model.predict(x)
    return preds
@app.route('/', methods=['GET'])
def index():
    return render_template('index.html')  # Assuming your template displays "Choose file"
@app.route('/predict', methods=['POST'])
def upload():
    if 'file' not in request.files:
        return "No file part"
    f = request.files['file']
    if f.filename == '':
        return "No selected file"
    # Get the chosen file name
    file_name = f.filename
    if f and f.filename.lower().endswith(('.png', '.jpg', '.jpeg')):
        basepath = os.path.dirname(__file__)
        file_path = os.path.join(basepath, 'uploads', f.filename)
```

```
        os.makedirs(os.path.dirname(file_path), exist_ok=True)  # Create uploads directory if it doesn't exist
        f.save(file_path)
        preds = model_predict(file_path, model)
        result = np.argmax(preds, axis=1)
        # Encode image to base64
        with open(file_path, "rb") as image_file:
            encoded_string = base64.b64encode(image_file.read()).decode('utf-8')
        # Get image format
        image_format = f.filename.split('.')[-1]
        if result == 0:
            return render_template('result.html', file_name=file_name, prediction="Normal", image_data=encoded_string, image_format=image_format)
        else:
            return render_template('result.html', file_name=file_name, prediction="Pneumonia", image_data=encoded_string, image_format=image_format)
    return "Invalid file format. Please upload a PNG, JPG, or JPEG image."
if __name__ == '__main__':
    app.run(debug=True)
```

HTML & CSS:

index.html

```
<!DOCTYPE html>
<html>
<head>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH" crossorigin="anonymous">
  <title>Pneumonia Detection</title>
  <style>
    body {
      background-image: url("https://w0.peakpx.com/wallpaper/25/378/HD-wallpaper-two-sweeties-puppies-blanket-white-pink-dog-sweet.jpg");
```

```
  background-size: cover;
  background-position: center;
}
.overlay {
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background-color: rgba(255, 255, 255, 0.4);
  z-index: 1;
}
.content {
  position: relative;
  z-index: 2;
  padding: 20px;
  background: rgba(255, 255, 255, 0.1);
  margin: 20px;
  border-radius: 10px;
  font-weight: bolder;
  font-size: x-large;
}
h1 {
  font-family: 'Times New Roman', Times, serif;
  font-size: xx-large;
}
label {
  font-size: x-large;
  font-family: 'Times New Roman', Times, serif;
  color: black;
  font-weight: bold;
}
input, textarea {
  font-size: large;
```

```
      font-family: 'Times New Roman', Times, serif;
      font-weight: bold;
    }
  </style>
</head>
<body>
  <div class="overlay"></div>
  <div class="content">
    <h1><center>Pneumonia Detection</center></h1>
    <center>
      <form id="upload-form" action="{{ url_for('upload') }}" method="post" enctype="multipart/form-data">
        <label for="file" class="form-label">Upload Chest X-ray Image:</label>
        <div class="input-group mb-3">
          <input type="file" class="form-control" id="file" name="file" accept="image/*">
        </div>
        <div class="mb-3">
          <span id="file-name" class="text-muted">No file chosen</span>
        </div>
        <input type="submit" value="Upload and Predict" class="btn btn-primary">
      </form>
    </center>
  </div>
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-ENjdO4Dr2bkBIFxQpeoTz1HIcje39Wm4jDKdf19U8gI4ddQ3GYNS7NTKfAdVQS" crossorigin="anonymous"></script>
  <script>
    const fileInput = document.getElementById('file');
    const fileName = document.getElementById('file-name');
    fileInput.addEventListener('change', () => {
      if (fileInput.files.length > 0) {
        fileName.textContent = fileInput.files[0].name;
      } else {
```

```
        fileName.textContent = 'No file chosen';
      }
    });
  </script>
</body>
</html>
```

result.html :

```
<!DOCTYPE html>
<html style="text-align: center;">
<head>
  <title>Prediction Result</title>
</head>
<body>
  <h1>Prediction for image: {{ file_name }}</h1>
  <img src=" {{ image_data }}" alt="Uploaded Image"
width="300">
  <p>The image is predicted to be: {{ prediction }}</p>
</body>
</html>
```

REFERENCES

1) Flask
2) VGG16 model
3) Image Generator
4) Bootstrap5
5) Source of the dataset
6) Python Base64 Module Documentation
7) OpenCV python tutorial
8) CSS tutorial
9) Data Augmentation
10) Glob