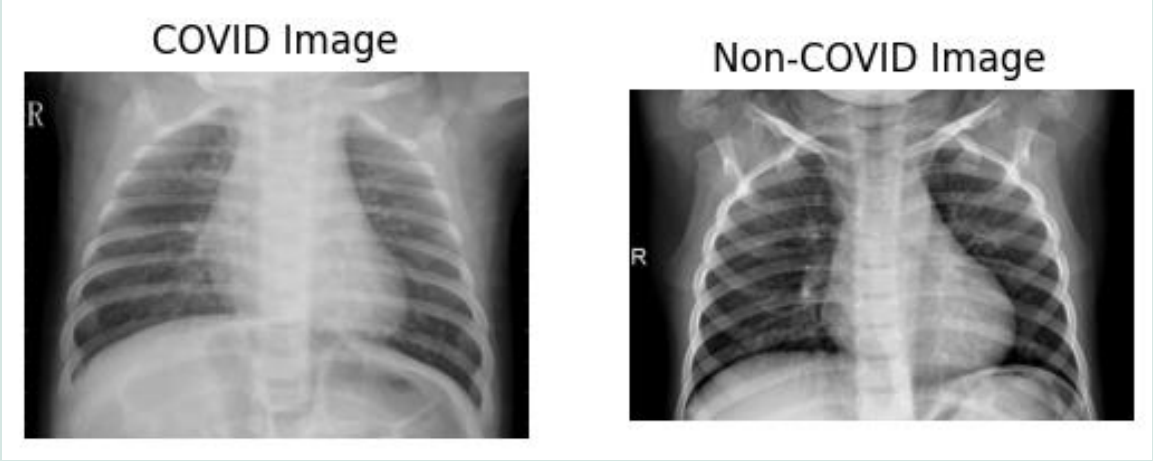


COVID Detection using X-ray Images

SUBMITTED BY:

TEAM NO: 11
ALLOCATED PROJECT NO: 13
Devam Patel, B22CH021

Karan Pratap singh, B22CH013

Shikar dave, B22CH032

Abhimanyu gupta, B22BB001

Raj Vijayvargiya, B22ES004

SUBMITTED TO:
PROF. AVINASH SHARMA
avinashsharma@iitj.ac.in

Code link is attached here

Abstract

Medical image processing and classification have become integral components in modern healthcare, aiding clinicians in the diagnosis and treatment of various medical conditions. This report presents an exhaustive analysis of Python-based code snippets focused on the processing, enhancement, and classification of medical images. Leveraging popular libraries such as OpenCV, Matplotlib, TensorFlow, and scikit-learn, the code addresses critical tasks ranging from data visualization and image enhancement to feature extraction and machine learning-based classification.

1. Introduction

Medical imaging has revolutionized the way healthcare professionals diagnose and treat diseases. With advancements in technology, the abundance of medical image data has created opportunities for leveraging machine learning techniques to assist in the analysis and interpretation of these images. This report explores a series of code snippets designed to tackle various aspects of medical image processing and classification, providing a thorough understanding of the implemented techniques.

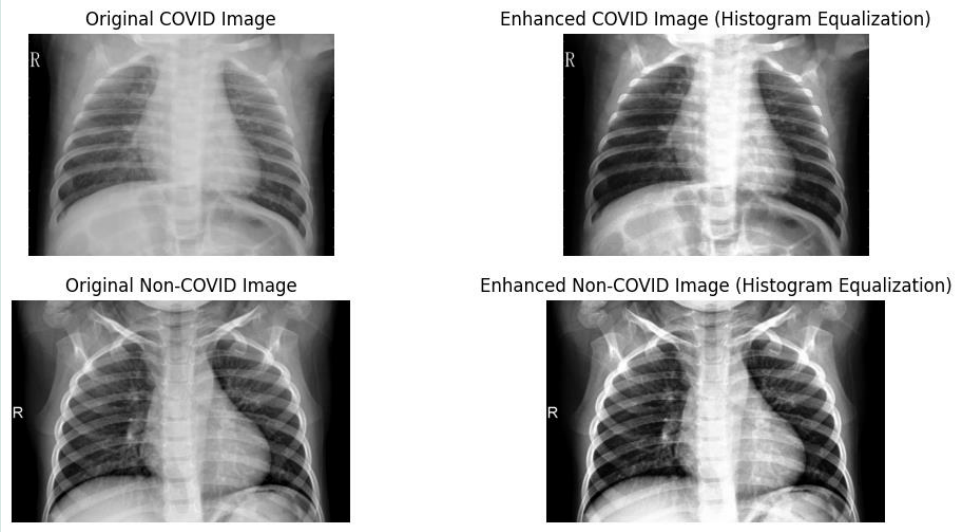
2. Data Visualization and Image Enhancement

2.1 Purpose of Data Visualization

Medical images, especially those related to conditions like COVID-19, often require careful examination and analysis. The initial code snippets focus on data visualization, employing Matplotlib to showcase both original and enhanced images side by side. Visualization serves as a crucial step in understanding the characteristics of the images and the potential impact of image enhancement techniques.

2.2 Histogram Equalization for Image Enhancement

Histogram equalization is a widely used technique to enhance the contrast of images. The code demonstrates the application of histogram equalization to both COVID and non-COVID images, highlighting its effectiveness in improving the visibility of key features.



3. Enhancing All Images

Expanding from individual image enhancement to processing entire datasets, this section of the code introduces a systematic approach. By creating separate directories for enhanced COVID and non-COVID images, the code streamlines the preprocessing pipeline. This scalable solution becomes particularly valuable when dealing with large volumes of medical image data.

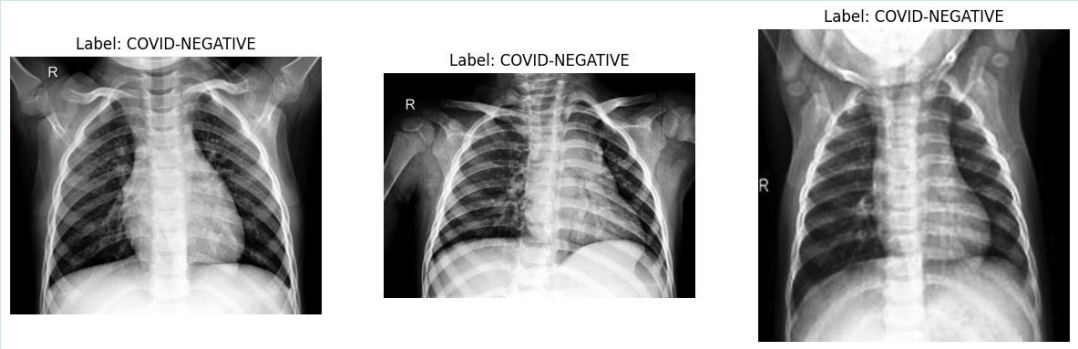
4. Data Preprocessing

4.1 Importance of Data Preprocessing

Data preprocessing is a critical phase in any machine learning project. In the context of medical image classification, it involves organizing data, assigning labels, and preparing the dataset for subsequent tasks. The code assigns labels to COVID-positive and COVID-negative cases, establishing a structured Data Frame that facilitates efficient data handling.

4.2 Creating a Structured Data Frame

The structured Data Frame, created using Pandas, serves as the backbone for subsequent analyses. It enables easy indexing, slicing, and manipulation of the data, laying the foundation for the training and evaluation of machine learning models.



5. Image Augmentation

5.1 Role of Data Augmentation

Data augmentation is a crucial strategy to diversify the dataset and improve the generalization capability of machine learning models. The implementation of TensorFlow's ImageDataGenerator introduces variations such as rotation, shifting, flipping, and zooming, creating a more robust dataset for training.

5.2 Ensuring Model Robustness

By introducing diverse perspectives of each image through augmentation, the code aims to ensure that the trained models are robust to variations commonly encountered in real-world scenarios. This strategy contributes to the model's ability to generalize well beyond the training dataset.

6. Data Inspection

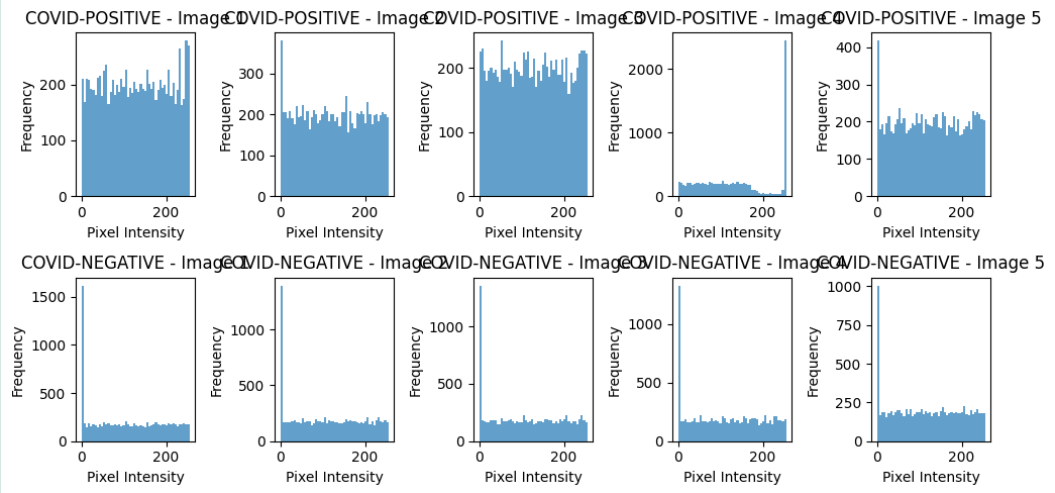
6.1 Significance of Data Inspection

Randomly selecting and displaying images from the dataset provides a qualitative assessment of the data's characteristics. This section of the code allows for a visual

inspection, helping to identify patterns, variations, and potential challenges within the dataset.

6.2 Balancing the Dataset

The random selection of images ensures that the displayed samples represent a diverse range of cases. Balancing the dataset is crucial for preventing biases in model training and ensuring that the model learns equally from both positive and negative instances.



7. Data Processing

7.1 Multithreading for Efficient Processing

Efficient data processing is essential, particularly when dealing with large datasets. The code introduces multithreading to parallelize image processing tasks, significantly improving processing speed. Error handling mechanisms are implemented to ensure the robustness of the pipeline.

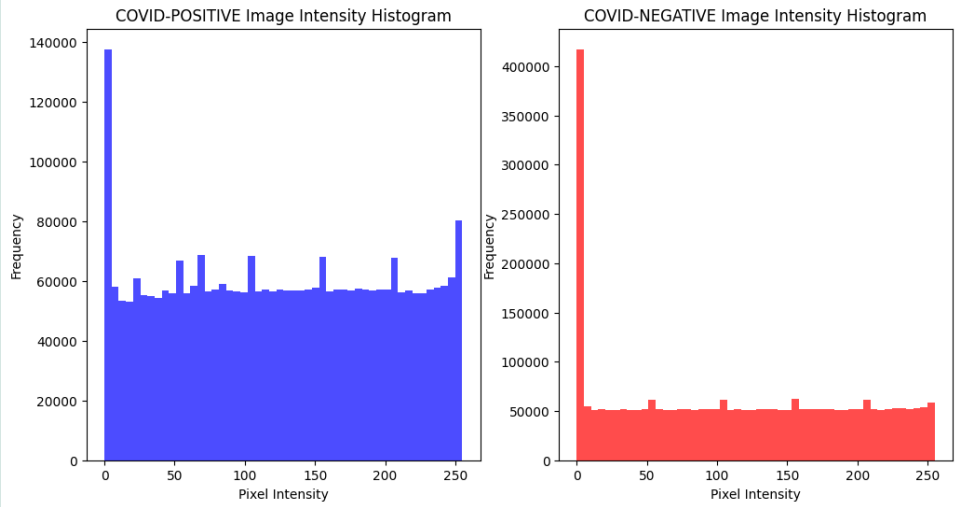
7.2 Normalizing Pixel Values

Normalizing pixel values is a standard practice in image processing and machine learning. The code ensures that pixel values are scaled to a common range, preventing any particular feature from dominating the learning process.

8. Data Visualization (Histogram)

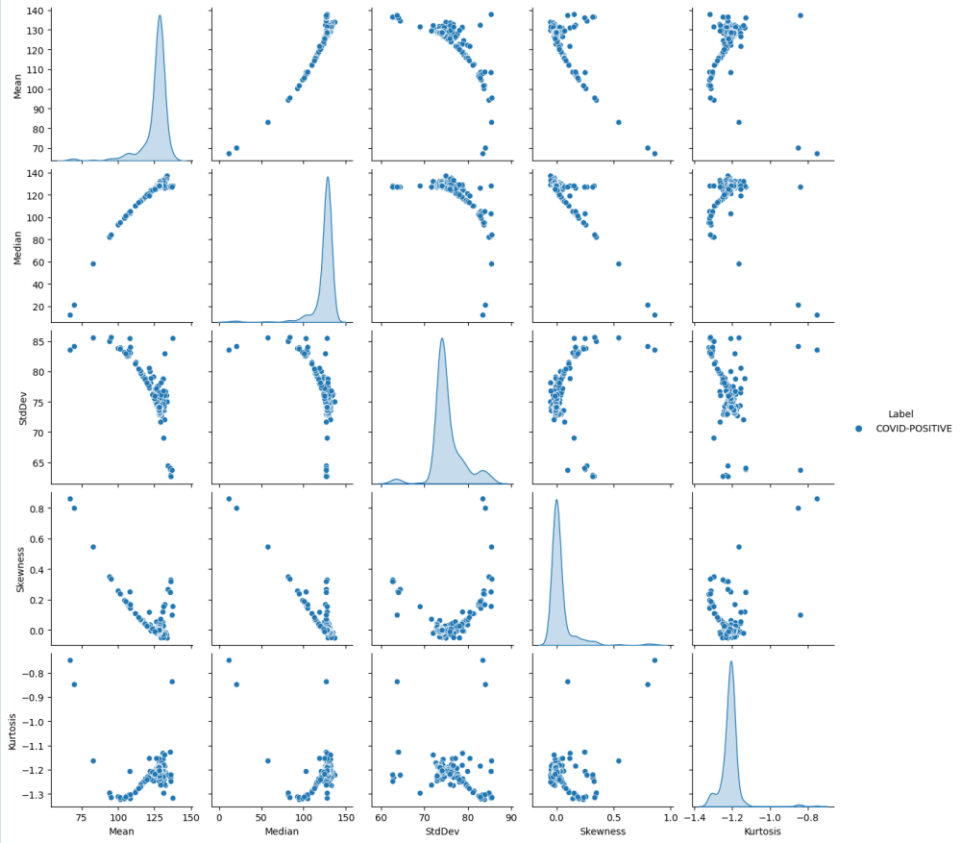
8.1 Extracting Insights from Pixel Intensity Histograms

Histograms of pixel intensities provide quantitative insights into the distribution of features within the dataset. The code generates histograms for COVID and non-COVID images, allowing for a thorough analysis of pixel intensity distributions.



8.2 Identifying Trends and Anomalies

By visualizing the distribution of pixel intensities, the code facilitates the identification of trends and anomalies within the dataset. Understanding the pixel intensity characteristics is crucial for selecting appropriate image processing and normalization strategies.



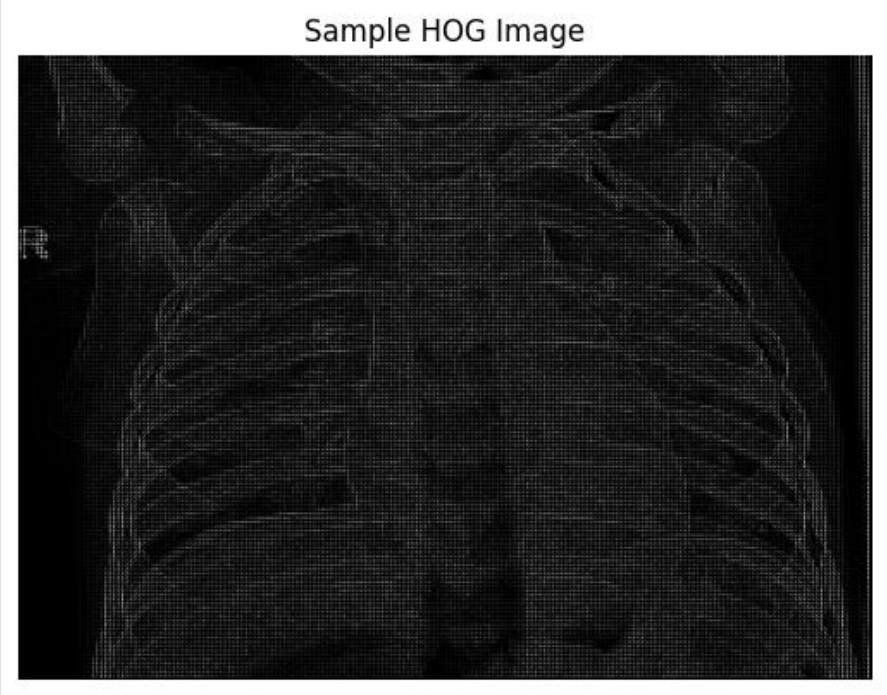
9. Feature Extraction (HOG)

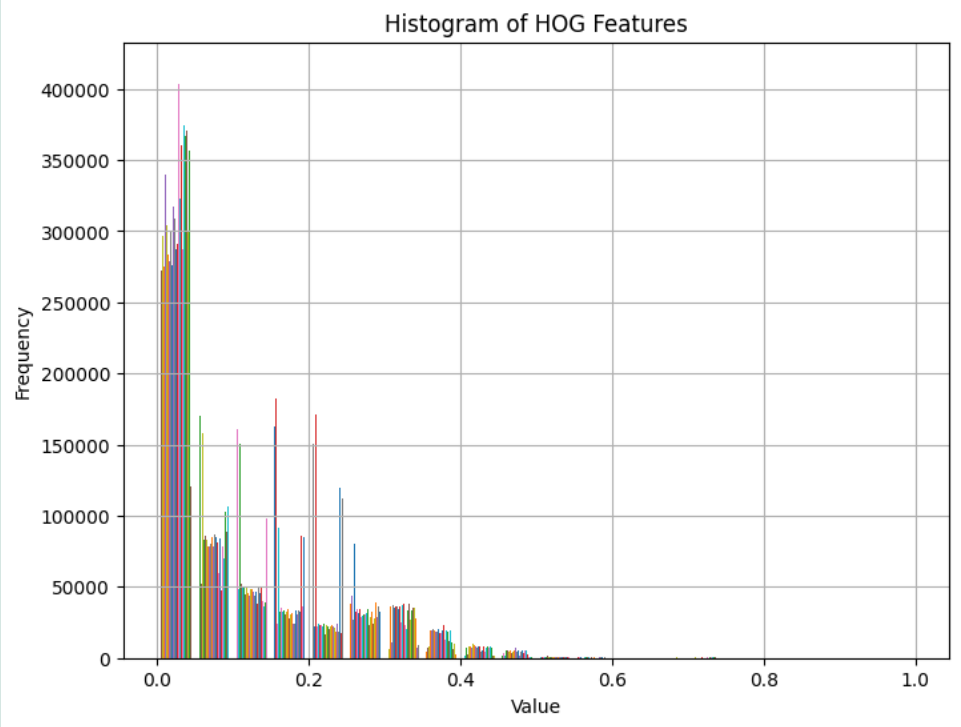
9.1 Role of Feature Extraction

Feature extraction is a crucial step in preparing data for machine learning models. The code employs Histogram of Oriented Gradients (HOG) as a feature extraction technique. HOG captures important patterns and gradients within images, providing a compact representation of key features.

9.2 Visualizing HOG Features

The code visualizes HOG features through histograms, offering a qualitative understanding of the information captured. HOG features are particularly valuable in scenarios where the spatial arrangement of features is crucial for classification.





10. Classification Models

10.1 Overview of Classification Models

This section delves into the application of various machine learning models for medical image classification. Models including Logistic Regression, Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Decision Tree, Random Forest, and Naive Bayes are implemented and evaluated.

10.2 Evaluating Model Performance

Each model's performance is assessed based on accuracy, confusion matrix, and classification reports. These metrics provide a comprehensive view of the model's ability to correctly classify COVID and non-COVID cases.

10.3 Considerations for Model Selection

The choice of classification models is crucial and depends on factors such as dataset characteristics, interpretability requirements, and computational resources. The code provides a comparative analysis to guide the selection of an appropriate model for the given task.

```
Model: Logistic Regression
Accuracy: 0.935251798561151
Confusion Matrix:
[[298  17]
 [ 10  92]]
Classification Report:
               precision    recall  f1-score   support

COVID-NEGATIVE       0.97      0.95      0.96       315
COVID-POSITIVE       0.84      0.90      0.87       102

      accuracy                           0.94       417
     macro avg       0.91      0.92      0.91       417
  weighted avg       0.94      0.94      0.94       417

----------------------
Model: Support Vector Machine
Accuracy: 0.947242206235012
Confusion Matrix:
[[308   7]
 [ 15  87]]
Classification Report:
               precision    recall  f1-score   support

COVID-NEGATIVE       0.95      0.98      0.97       315
COVID-POSITIVE       0.93      0.85      0.89       102

      accuracy                           0.95       417
     macro avg       0.94      0.92      0.93       417
  weighted avg       0.95      0.95      0.95       417

----------------------
Model: K-Nearest Neighbors
Accuracy: 0.947242206235012
Confusion Matrix:
[[312   3]
 [ 19  83]]
```

```
Classification Report:
                precision    recall  f1-score   support

COVID-NEGATIVE       0.94      0.99      0.97       315
COVID-POSITIVE       0.97      0.81      0.88       102

      accuracy                           0.95       417
     macro avg       0.95      0.90      0.92       417
  weighted avg       0.95      0.95      0.95       417

----------------------
Model: Decision Tree
Accuracy: 0.920863309352518
Confusion Matrix:
[[300  15]
 [ 18  84]]
Classification Report:
                precision    recall  f1-score   support

COVID-NEGATIVE       0.94      0.95      0.95       315
COVID-POSITIVE       0.85      0.82      0.84       102

      accuracy                           0.92       417
     macro avg       0.90      0.89      0.89       417
  weighted avg       0.92      0.92      0.92       417

----------------------
Model: Random Forest
Accuracy: 0.947242206235012
Confusion Matrix:
[[312   3]
 [ 19  83]]
```

11. Saving a Model

11.1 Importance of Model Persistence

Saving a trained model is essential for its deployment in real-world scenarios. The code demonstrates how to save a trained Support Vector Machine (SVM) model using TensorFlow/Keras, ensuring that the model can be seamlessly integrated into healthcare systems for automated image classification.

11.2 Deployability and Integration

The ability to save and load trained models enhances their deployability, enabling healthcare professionals to utilize the models in practical settings. Integration with existing healthcare systems ensures a smooth transition from research to application.

12. Conclusion

12.1 Recapitulation of Key Findings

In conclusion, the presented code snippets offer a comprehensive workflow for medical image processing and classification. From initial data visualization and enhancement to feature extraction and model training, each step contributes to the overarching goal of creating reliable and efficient tools for medical image analysis.

```
Classification Report:
                precision    recall  f1-score   support

COVID-NEGATIVE       0.94      0.99      0.97       315
COVID-POSITIVE       0.97      0.81      0.88       102

      accuracy                           0.95       417
     macro avg       0.95      0.90      0.92       417
  weighted avg       0.95      0.95      0.95       417

----------------------
Model: Naive Bayes
Accuracy: 0.8776978417266187
Confusion Matrix:
[[285  30]
 [ 21  81]]
Classification Report:
                precision    recall  f1-score   support

COVID-NEGATIVE       0.93      0.90      0.92       315
COVID-POSITIVE       0.73      0.79      0.76       102

      accuracy                           0.88       417
     macro avg       0.83      0.85      0.84       417
  weighted avg       0.88      0.88      0.88       417

----------------------
```

12.2 Implications for Healthcare

The successful implementation of these code snippets provides a foundation for building robust tools for COVID detection using X-ray images. The insights gained from data visualization, preprocessing, and model evaluation have implications for improving healthcare diagnostics through automated image analysis.