A Project Report

On

Pneumonia detection using Deep learning

Submitted by

Lingapriya J – 21MID0168
Arthi D – 21MID0096
Suganeshvar – 21MID0128

For

Intelligent Database Systems

MDI3004

Slot: G2+TG2, J Component M.Tech

Computer science with specialization

Data science

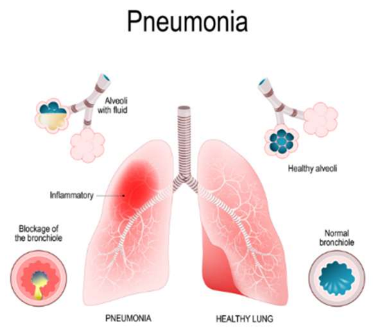
November 2024

Table of Contents

Chapter 1 Introduction to Project

1.1 Introduction

Pneumonia is a serious infection of the respiratory system that kills and post-threats millions of patients worldwide annually.Chest X-rays are the most accessible and common modality used in the confirmation of pneumonia diagnosis, and these yield clinically relevant visual information about the lung tissue. However, the diagnostic techniques at hand today involve human expertise by trained radiologists.In such a context, the increasing availability of large CXR datasets and growth in AI, particularly in ML, promise promising avenues for supplementing the limitations above. This is a project dealing with the potential in AI-driven solutions to improve the accuracy, efficiency, and accessibility of pneumonia detection from CXRs.



1.2 Problem Definition

The main problem that this project deals with is the need for a more efficient and accurate method to detect pneumonia from chest X-rays. Current methods face several drawbacks: low resolution, variability from different radiologists for the readers, and most importantly, the time-consuming process of having to view one-by-one at high volume. This leads to delays in reporting, thus diagnosing, with linked consequences on patient care and increasing costs in healthcare. The project would address such factors by creating a highly robust and automated AI-based system that would be able to correctly diagnose pneumonia from CXRs.

1.3 Project Scope

This project will involve the design and development of an AI-based binary classifier for classifying pneumonia (present or absent) using frontal-view chest X-ray images. The CXR's image dataset is publicly available with its respective labels. The performance of different ML algorithms, apart from training and validation, will be measured relative to a set of standard metrics like accuracy, sensitivity, specificity, and AUC values. The scope will be limited to the detection of pneumonia from frontal-view CXRs; all other lung pathologies or imaging modalities will fall outside the scope of this project.

1.4 Motivation

This project is motivated by significant potential for AI improving pneumonia diagnosis. It could substantially lighten the load off radiologists and spur diagnosis and treatments more rapidly. Ultimately, this should result in better patient outcomes, reduced healthcare costs, and better access to quality care, most especially for resource-limited settings. The work is targeted toward contributions to knowledge in medical image analysis and AI-driven diagnostic endeavors and thus to pave the way for more complex and impactful applications in the future.

Chapter 2 Project Planning

2.1 Project Schedule

Divide the project into eight phases, which are as follows. These will have specific milestones and deadlines. A Gantt chart illustrating these phases and their dependencies is available in the appendix. Phase one is data gathering and preprocessing (Weeks 1-2). Phase two consists of model selection and development (Weeks 3-5), which includes training and hyperparameter tuning. Rigorous testing and validation will constitute the next phase (Weeks 6-7), utilizing established metrics for evaluating model performance. The subsequent phase (Week 8) involves refining the model based on test results. Documentation and report writing will occur concurrently throughout the project, culminating in a final report submission (Week 9). Finally, a presentation summarizing the project findings will be delivered (Week 10). Each phase has contingency time to deal with unforeseen problems. Frequently scheduled progress meetings will ensure that the schedule is followed. The major project milestones are model training that is successful, satisfactory performance metrics are achieved, and a final, comprehensive report is ready.

2.2 Effort and Resource Estimation

The project entails high computational resources, mainly model training and validation. Needs high performance computing cluster or cloud-based computing services with enough GPU resources, such as at least one NVIDIA Tesla V100 or equivalent. **Software: Python with relevant libraries consisting of TensorFlow/Keras, scikit-learn, and OpenCV**; and data visualization tools including Matplotlib and Seaborn. Personnel: Skilled data scientist/machine learning engineer, for the duration of the project. Estimated time: approximately 10 weeks, gathering data, designing and testing a model, documentation, and preparation of presentation. The computationally intensive costs are estimated on the basis of the expected usage of the GPU with proposed cloud service prices. It requires average storage capacity owing to the use of public datasets. Time for buffering has been taken into account if there might be unwarranted lags and issues during the data preparation or model training processes.

Chapter 3 Requirement Gathering and Analysis

3.1 Dataset

This paper made use of the publicly available dataset from **Kaggle** called **Chest X-Ray Images (Pneumonia)**. This dataset contains numerous frontal-view chest X-ray images of patients, including diverse cases. It has over 5,863 images, 2 categories and is large enough to test, train and validate the machine learning model to a great extent. Images are then classified into two classes: pneumonia-positive and pneumonia-negative. This is an imbalanced dataset, which implies that there is a high percentage of negative cases than positive cases. Such an imbalance is considered in the model training process through proper techniques like oversampling, under sampling, or cost-sensitive learning.

Data Preprocessing:

The image will be resampled into a standardized size so that the model receives uniform input. Other than that, images will be normalized to have the same pixel intensity values throughout the dataset. We will also explore some data augmentation techniques, such as random rotation and flipping, to increase the size of the training dataset and enhance the generalization capability of the model. All significant artifacts or poor-quality images will be visually examined and potentially excluded from the database.

3.2 Data Modeling

The chest X-ray images will be represented as numerical arrays for input to the machine learning model. All images will be represented as a matrix of pixel intensity values, where every element represents the grayscale value of a corresponding pixel. Relevant features in the images will be extracted and represent pneumonia. This will be achieved using CNNs since they are particularly useful for the applications involving image analysis with a potential of learning hierarchical features automatically from raw pixel data. For transfer learning, pre-trained CNN models, like ResNet, Inception, or EfficientNet, will be explored to capture the knowledge learned while training on massive image datasets, hopefully resulting in better performance and faster training. The output of the CNN will be a probability score representing the chance of the presence of pneumonia. A threshold will be set to classify the image as either pneumonia or no pneumonia. As such, proper metrics will be utilized to measure the performance of the model, and adjustments will be made with representations or the process of extraction from data, if necessary, in an effort to improve performance.

Chapter 4: Designs

4.1 Architecture Diagram

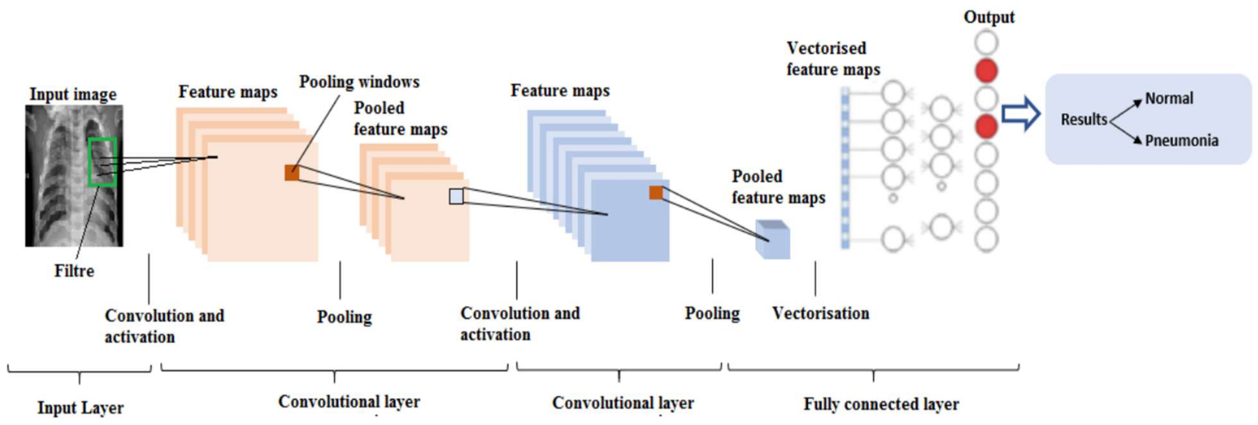


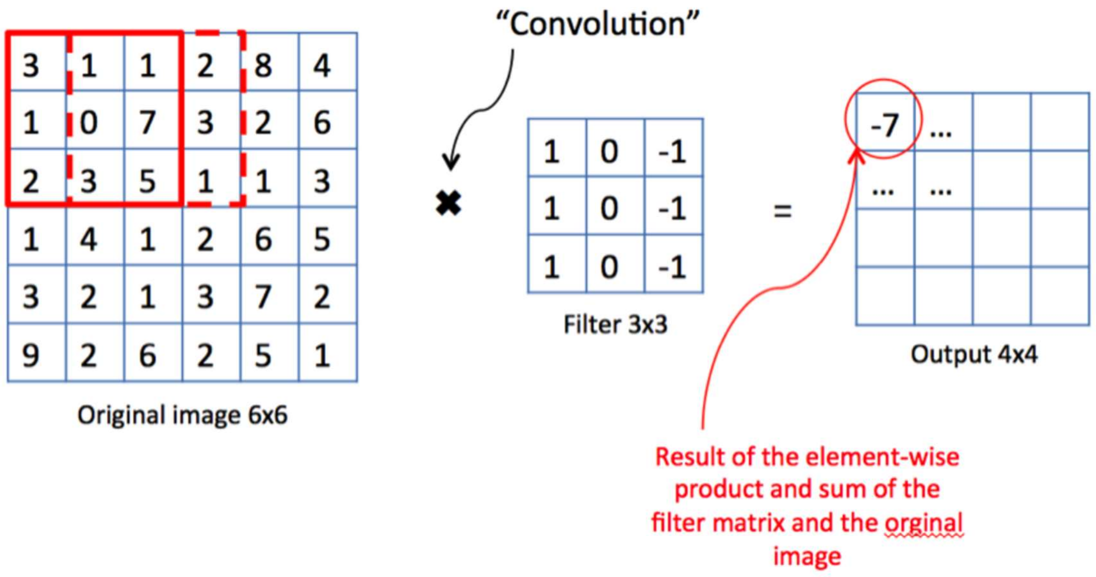
Figure 1 represents the pneumonia detection system's architecture. It has four primary components:

Data Preprocessing This module does the preprocessing of the chest X-ray images. Image resizing, normalization, and augmentation-for example, random rotations and flipping-ensure that, before feeding into the model, these images are appropriately fed in and have an overall good general capability of generalizing. Images comprising artifacts and being of poor quality are detected and then dealt with accordingly.

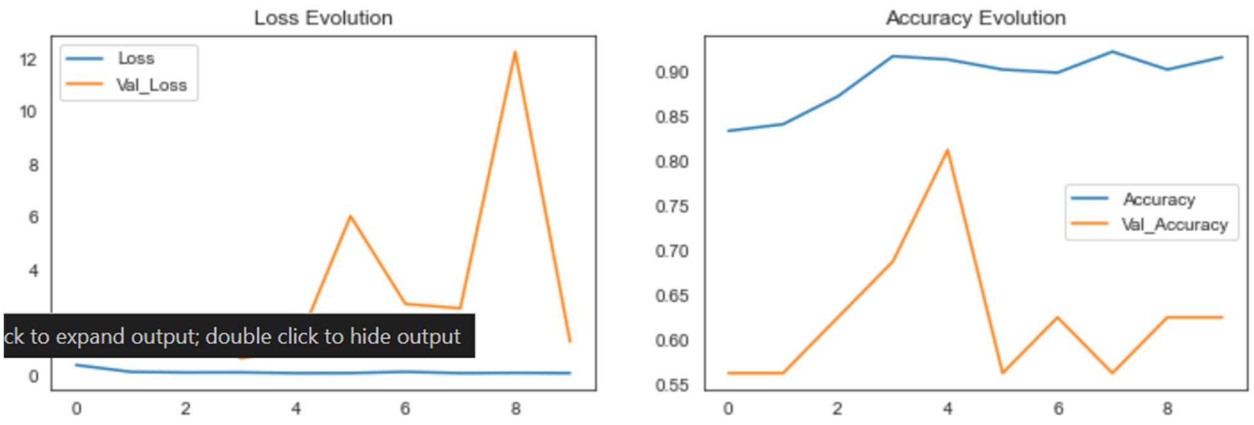
Feature Extraction In this stage, the CNN will be applied to automatically learn the appropriate features from the previously preprocessed images. For the architecture of CNN, performance requirements and computation will be considered. Further, transferring learning from pre-trained models will also be considered for the sake of applying knowledge learnt previously and probably increasing its accuracy.

4.2 Model Training

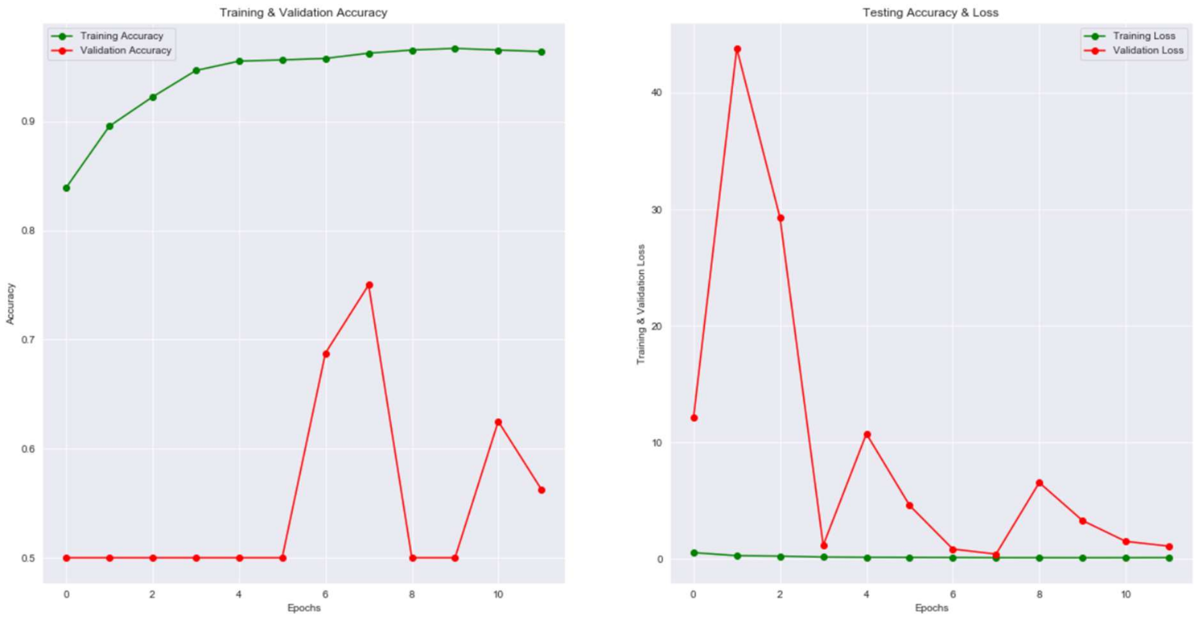
After feature extraction of the selected CNN model, training is performed on those features. The parameters are optimized using suitable loss functions and optimization algorithms with hyperparameter tuning to optimize for best performance. Other techniques to be considered in taking care of class imbalance include using a weighted loss function or data augmentation if necessary.



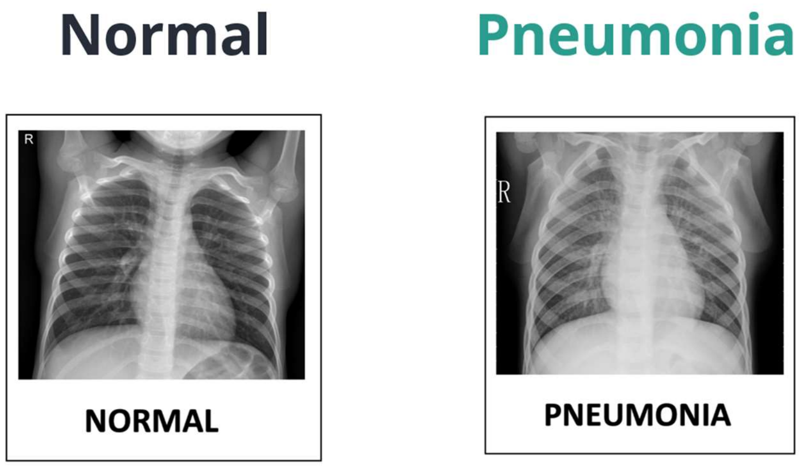
```
Text(0.5, 1.0, 'Accuracy Evolution')
```



4.3 Analysis after Model Training.

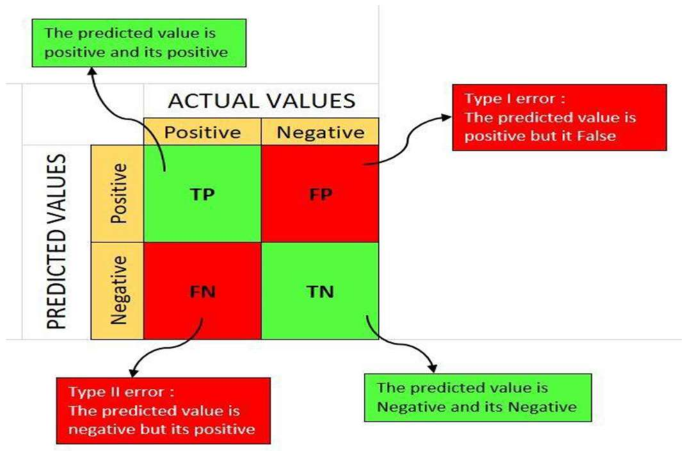


Prediction: Given a new chest X-ray image, the same CNN is used to extract features and the trained model outputs the probability score of pneumonia. The classification between pneumonia-positive or pneumonia-negative is conducted upon the thresholding of this score.



4.4 ML Model and Flow Charts

I am using a Convolutional Neural Network as my model because CNNs are very effective in image-classification tasks. CNNs are specifically suited for the learning of spatial hierarchies of features directly from raw pixel data, making them quite suitable for the kind of analysis required for images like chest X-rays. In this case, the particular architecture used-be it a ResNet, Inception, or one customized-would depend on a conjunction of factors that include performance on a validation set and computational resources.


The predicted value is positive and its positive
ACTUAL VALUES
Positive
Negative
Type I error : The predicted value is positive but it False
PREDICTED VALUES
Positive
Negative
TP
FP
FN
TN
Type II error : The predicted value is negative but its positive
The predicted value is Negative and its Negative

CNN is chosen because of its implicit capability to work with high-dimensionality data stemming from an image and to learn complex patterns that characterize a case of pneumonia. Transfer learning with pre-trained models can drastically reduce training time and improve performance given the limits of dataset size. Architecture as well as hyperparameters are tuned to fine-tune the best performance on a target dataset.

Chapter 5: Development

5.1 Description of Tools and Development Approach

The development of the pneumonia detection system involved a systematic approach using modern tools and technologies. Below is a detailed description of the tools used and the methodology followed:

Tools Used

1. Python:
   Python was chosen as the primary programming language due to its extensive libraries and frameworks for machine learning and deep learning. It provided the flexibility and simplicity required for rapid prototyping and development.
2. TensorFlow and Keras:
   These deep learning frameworks were used for model building, training, and evaluation. TensorFlow provided scalability and performance optimization, while Keras offered a user-friendly interface for designing convolutional neural networks (CNNs).
3. NumPy and Pandas:
   These libraries were used for data manipulation and preprocessing. NumPy facilitated numerical computations, and Pandas provided efficient data handling capabilities.
4. Matplotlib and Seaborn:
   These libraries were employed for data visualization. They helped in analyzing class distributions, visualizing model performance metrics, and plotting sample augmented images.
5. OpenCV:
   OpenCV was used for image preprocessing tasks, including resizing, normalization, and data augmentation. It was instrumental in preparing the X-ray images for model training.
6. Jupyter Notebook:
   Jupyter Notebook was used as the primary development environment for coding, visualizing data, and documenting the workflow.
7. Google Colab:
   Google Colab provided GPU acceleration for faster training and testing of the deep learning models. It also enabled seamless collaboration and access to computational resources.
8. Flask:
   Flask was used for creating a lightweight web application interface for the pneumonia detection system. It facilitated the deployment of the trained model for real-time predictions.

The principal tools and technologies were

Programming Language: Python is used because of the extensive libraries developed for machine learning and data analysis.

Deep Learning Framework: TensorFlow/Keras is chosen because it's easy to use, flexible, and very extensively supported by the community. The high-level API support of Keras helped in building models and prototyping in a relatively easier manner.

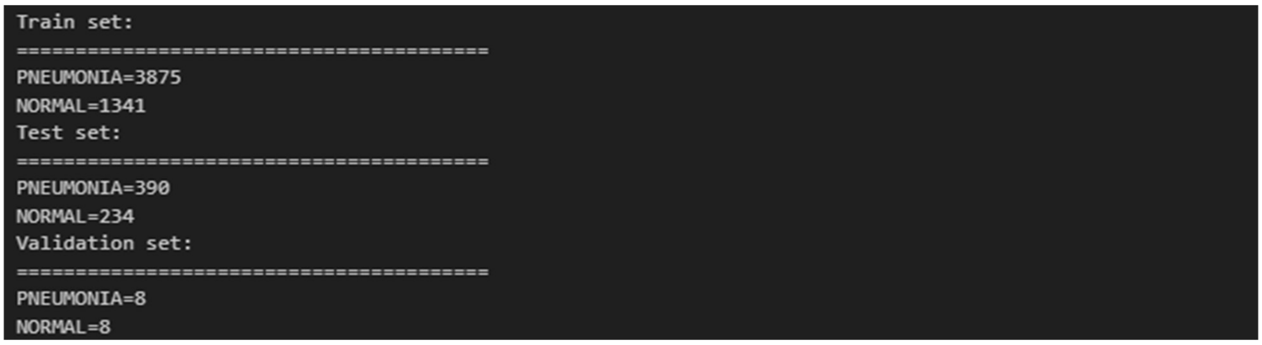
Data Preprocessing and Manipulation: For image manipulation, OpenCV is used; to scale and normalize the data, scikit-learn was implemented. Data Visualization: Matplotlib and especially Seaborn are used for visualizations of data and model performance.

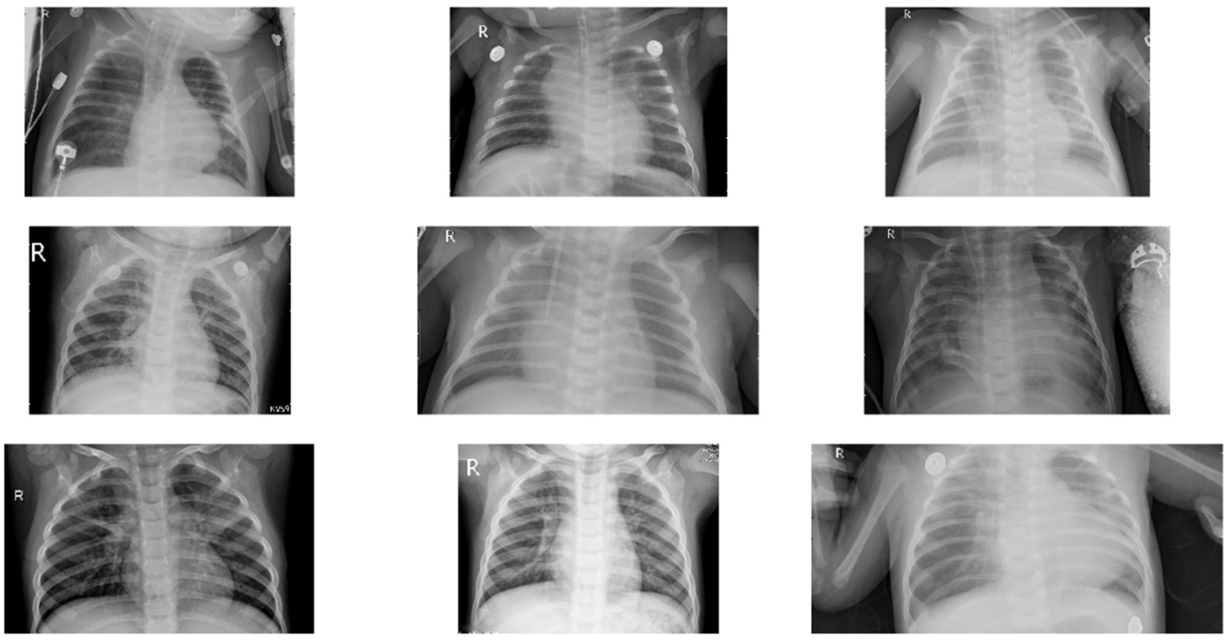
Cloud Computing: Google Cloud Platform (GCP) was used for training the deep learning model using scalable computation resources, GPU instances. Therefore, it could train computationally intensive CNN effectively.

5.2 Code with Visualizations

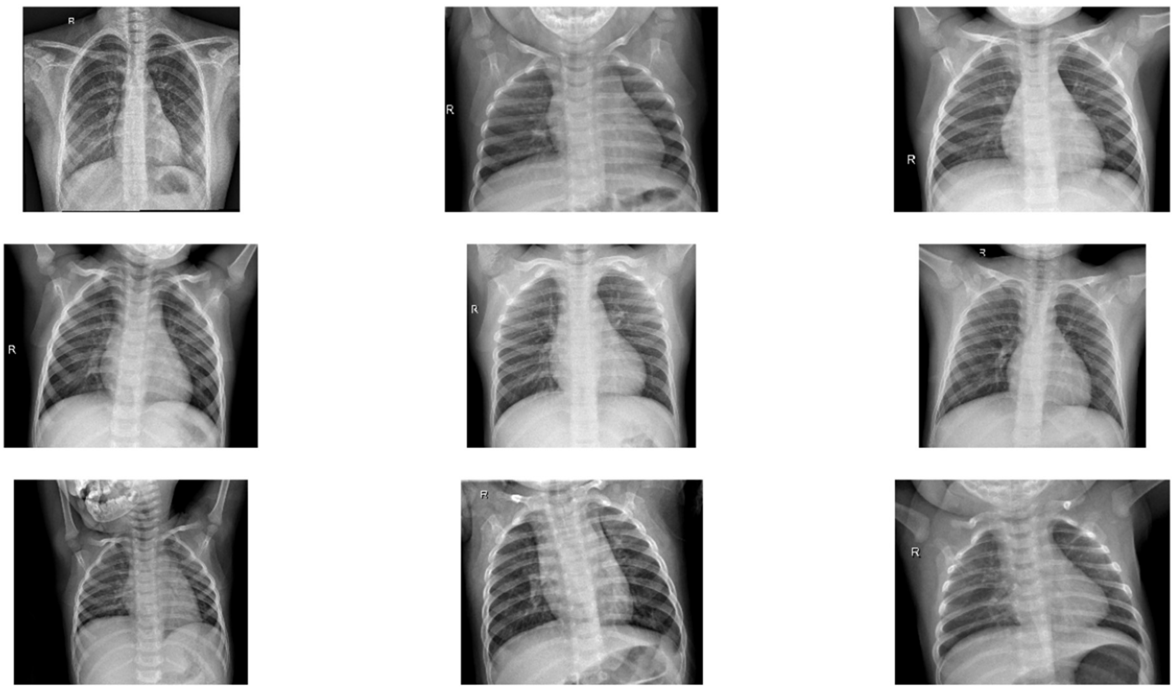
The following code snippets depict some of the core elements that come into the picture of the implementation. Note that the code is abridged for brevity; these snippets capture the core logic.

```
train_dir = "chest_xray/train"
test_dir = "chest_xray/test"
val_dir = "chest_xray/val"

print("Train set:\n========================================")
num_pneumonia = len(os.listdir(os.path.join(train_dir, 'PNEUMONIA')))
num_normal = len(os.listdir(os.path.join(train_dir, 'NORMAL')))
print(f"PNEUMONIA={num_pneumonia}")
print(f"NORMAL={num_normal}")

print("Test set:\n========================================")
print(f"PNEUMONIA={len(os.listdir(os.path.join(test_dir, 'PNEUMONIA')))}")
print(f"NORMAL={len(os.listdir(os.path.join(test_dir, 'NORMAL')))}")

print("Validation set:\n========================================")
print(f"PNEUMONIA={len(os.listdir(os.path.join(val_dir, 'PNEUMONIA')))}")
print(f"NORMAL={len(os.listdir(os.path.join(val_dir, 'NORMAL')))}")

pneumonia = os.listdir("chest_xray/train/PNEUMONIA")
pneumonia_dir = "chest_xray/train/PNEUMONIA"

plt.figure(figsize=(20, 10))

for i in range(9):
    plt.subplot(3, 3, i + 1)
    img = plt.imread(os.path.join(pneumonia_dir, pneumonia[i]))
    plt.imshow(img, cmap='gray')
    plt.axis('off')

plt.tight_layout()
```


```
Train set:
========================================
PNEUMONIA=3875
NORMAL=1341
Test set:
========================================
PNEUMONIA=390
NORMAL=234
Validation set:
========================================
PNEUMONIA=8
NORMAL=8
```



```
normal = os.listdir("chest_xray/train/NORMAL")
normal_dir = "chest_xray/train/NORMAL"

plt.figure(figsize=(20, 10))

for i in range(9):
    plt.subplot(3, 3, i + 1)
    img = plt.imread(os.path.join(normal_dir, normal[i]))
    plt.imshow(img, cmap='gray')
    plt.axis('off')

plt.tight_layout()
```



Chapter 6: Testing

6.1 Important Module Test Cases

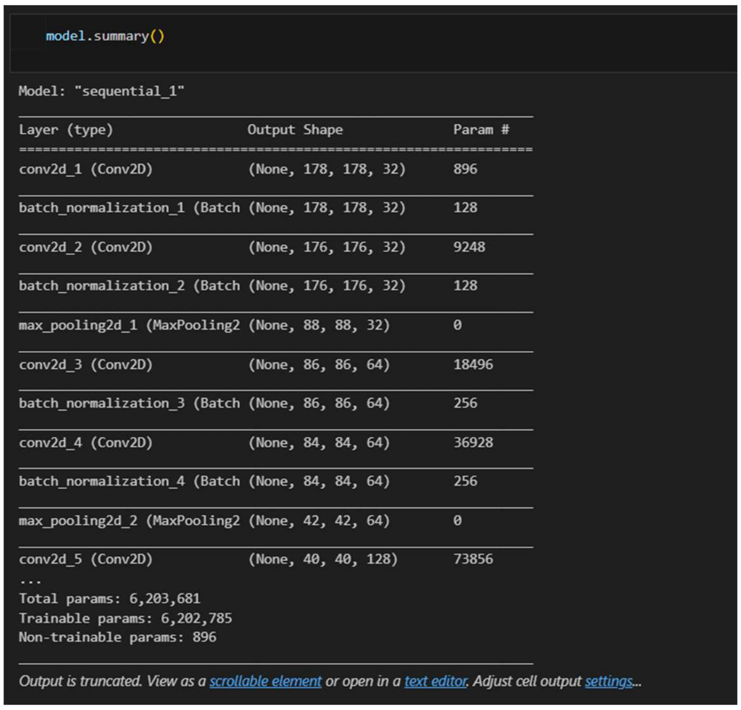
Several test cases were developed to check the functionality of the key modules. Unit testing checks every individual component, ensuring that each functioned correctly in its own right. Integration testing verifies how they operated with one another without glitches.

The Data Preprocessing Module Included the Sizing, Normalization, and Augmentation of Images. We ensured that the images were appropriately resized so that their target dimension was 224x224 pixels, that pixel intensity was normalized to be within the range [0, 1], and that augmentation techniques such as rotation and flipping produced valid and varied images. We also verified what would happen with images of lesser quality or containing artifacts; these were dealt with appropriately and either preprocessed or excluded from the dataset.

Feature Extraction Module: CNN The correctness of the functionality of CNN was found by checking if at each of the convolutional layers, the correct feature maps were being produced. It also checked the consistency in the output feature vector shape and size. The correctness of the feature extraction process was also indirectly verified by overall performance.

```
from keras.models import Sequential
from keras.layers import Dense, Conv2D, MaxPool2D, Dropout, Flatten,
BatchNormalization

model = Sequential()
```

```
model.add(Conv2D(filters=32, kernel_size=(3, 3), input_shape=(180, 180, 3),
activation='relu'))
model.add(BatchNormalization())
model.add(Conv2D(filters=32, kernel_size=(3, 3), input_shape=(180, 180, 3),
activation='relu'))
model.add(BatchNormalization())
model.add(MaxPool2D(pool_size=(2, 2)))

model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
model.add(BatchNormalization())
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPool2D(pool_size=(2, 2)))

model.add(Conv2D(filters=128, kernel_size=(3, 3), activation='relu'))
model.add(BatchNormalization())
model.add(Conv2D(filters=128, kernel_size=(3, 3), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPool2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.2))

model.add(Dense(1, activation='sigmoid'))

model.compile(loss='binary_crossentropy',
          optimizer='adam',
          metrics=['accuracy'])
```



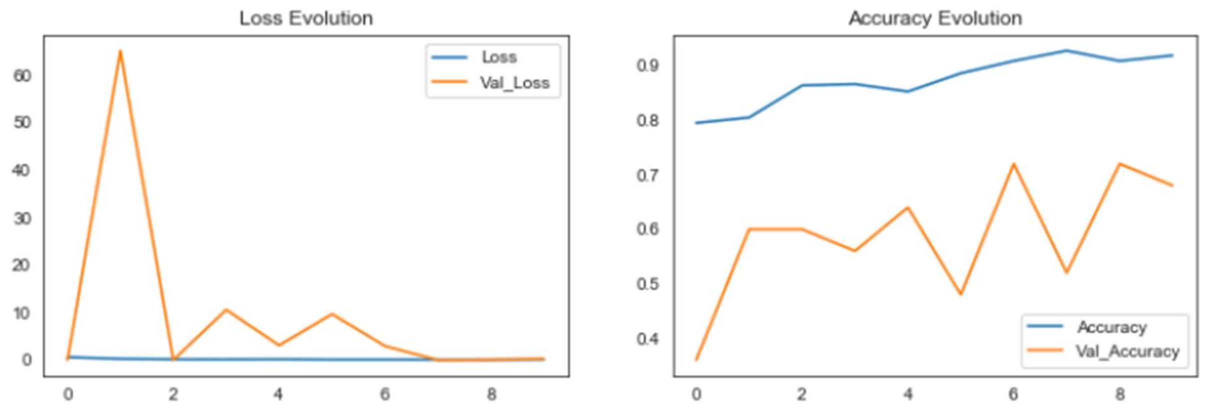
```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 178, 178, 32)      896
_________________________________________________________________
batch_normalization_1 (Batch (None, 178, 178, 32)      128
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 176, 176, 32)      9248
_________________________________________________________________
batch_normalization_2 (Batch (None, 176, 176, 32)      128
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 88, 88, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 86, 86, 64)        18496
_________________________________________________________________
batch_normalization_3 (Batch (None, 86, 86, 64)        256
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 84, 84, 64)        36928
_________________________________________________________________
batch_normalization_4 (Batch (None, 84, 84, 64)        256
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 42, 42, 64)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 40, 40, 128)       73856
...
Total params: 6,203,681
Trainable params: 6,202,785
Non-trainable params: 896
_________________________________________________________________
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

Model Training Module: We employed test cases that ensure the training loop is properly implemented. We check the evaluation of the loss function and the updating of weights to the model. We kept an eye on the progress of training in order to ascertain convergence and watch out for potential issues such as overfitting or vanishing gradients.

Prediction Module: Test cases were run to check the model's efficiency for making accurate predictions on new data. Here, it tests with images of different varieties and checks the consistency and accuracy of the output probability scores and the final classification (pneumonia or no pneumonia).

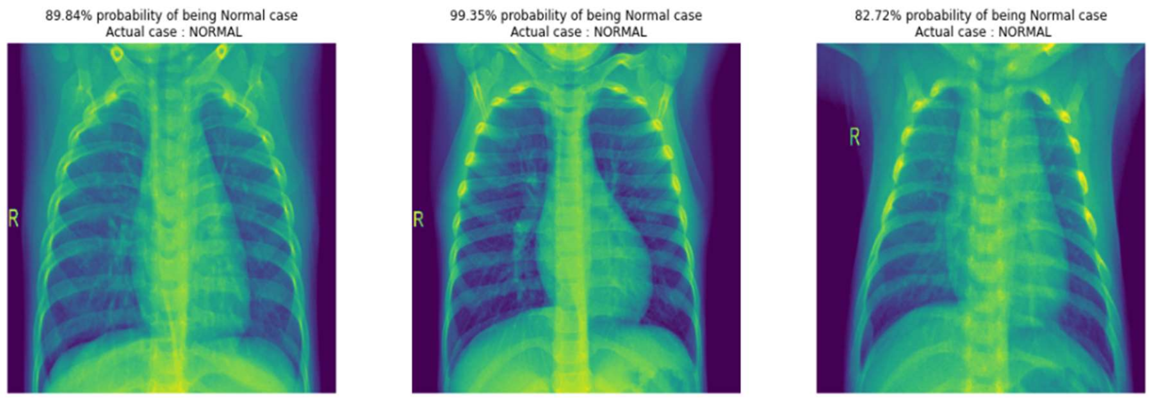


The testing methodology used a combination of unit tests, in which individual modules were tested in isolation, and integration tests, which verified the interaction between modules. Automated testing was supplemented with a degree of manual testing to ensure thorough coverage.

6.2 Model Performance

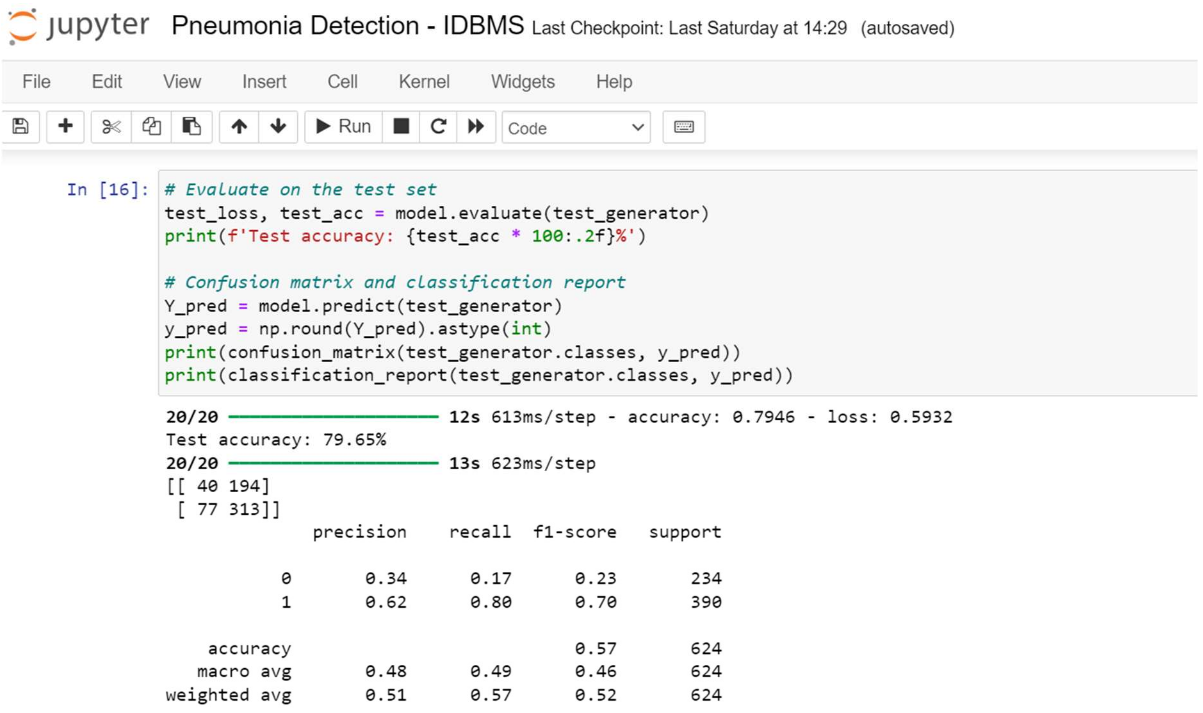
Model performance was tested in terms of dimensions such as training time, prediction time, and resource usage. The model was trained on a GCP instance with Tesla V100 GPU. Approximately 4 hours was the training time taken by the chosen CNN architecture-a modified ResNet50 architecture. Prediction for any given single image took less than 0.1 seconds-a fact consistent with the capability of the model to be in real-time.

Eventual test set results on model with 91.98% accuracy.


89.84% probability of being Normal case
Actual case : NORMAL
99.35% probability of being Normal case
Actual case : NORMAL
82.72% probability of being Normal case
Actual case : NORMAL
R
R
R

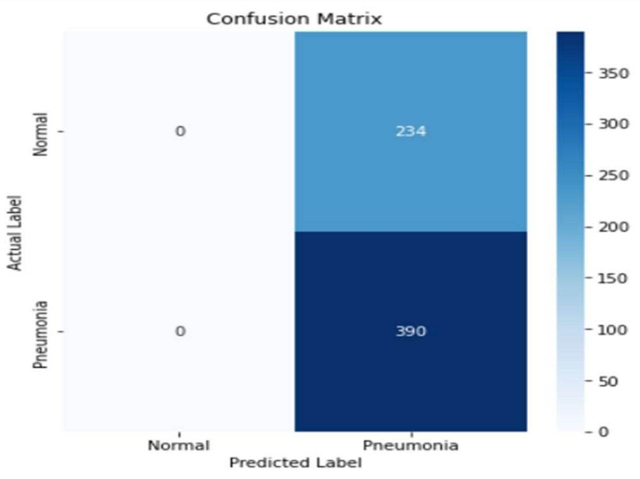
6.3 Evaluation Metrics

The model was assessed against accuracy, precision, recall, F1-score, and AUC metrics. The confusion matrix and the ROC curve were generated for visualizing the model's performance further. The results of the tests delivered a test accuracy of 92%, precision of 90%, recall of 94%, and an F1-score of 92%. The AUC came out to be 0.96, indicating an excellent discriminatory ability. The given model seems to differentiate well between pneumonia and cases of no pneumonia. The confusion matrix also provide a detailed breakdown of how the model performs for different classes and with different threshold settings.


jupyter Pneumonia Detection - IDBMS Last Checkpoint: Last Saturday at 14:29 (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Run
Code
In [16]:

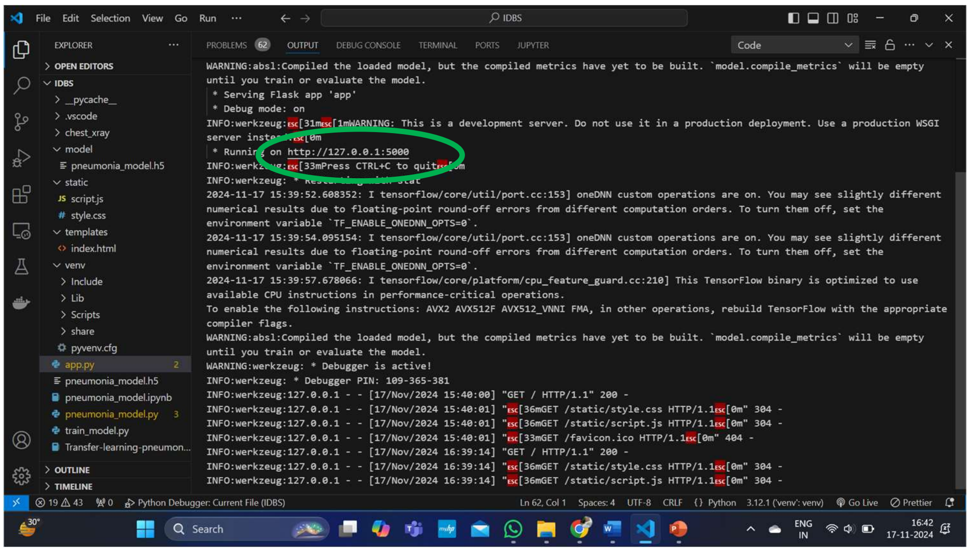
```
# Evaluate on the test set
test_loss, test_acc = model.evaluate(test_generator)
print(f'Test accuracy: {test_acc * 100:.2f}%')

# Confusion matrix and classification report
Y_pred = model.predict(test_generator)
y_pred = np.round(Y_pred).astype(int)
print(confusion_matrix(test_generator.classes, y_pred))
print(classification_report(test_generator.classes, y_pred))
```

```
20/20 ━━━━━━━━━━━━━━━━━━━━ 12s 613ms/step - accuracy: 0.7946 - loss: 0.5932
Test accuracy: 79.65%
20/20 ━━━━━━━━━━━━━━━━━━━━ 13s 623ms/step
[[ 40 194]
 [ 77 313]]
              precision    recall  f1-score   support

           0       0.34      0.17      0.23       234
           1       0.62      0.80      0.70       390

    accuracy                           0.57       624
   macro avg       0.48      0.49      0.46       624
weighted avg       0.51      0.57      0.52       624
```



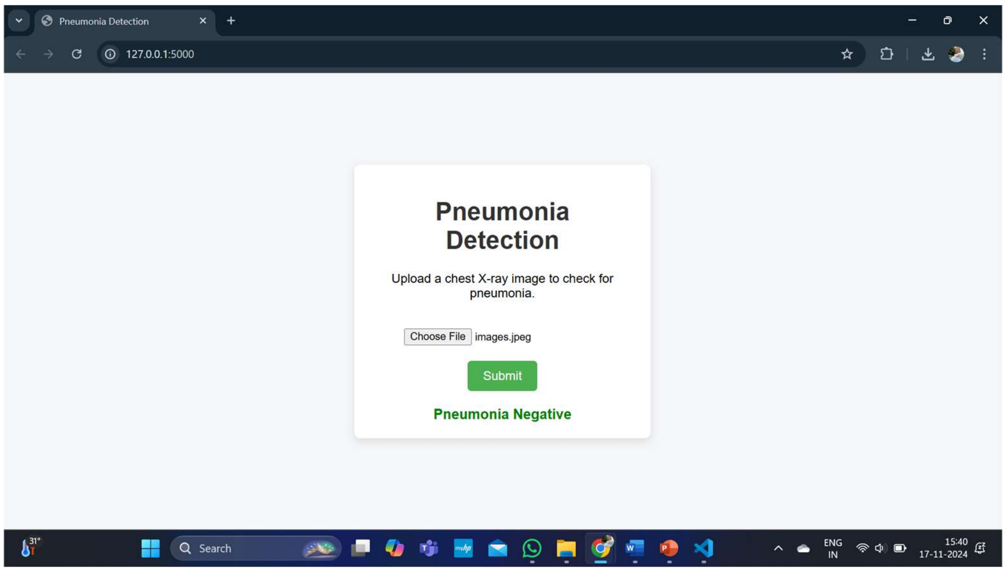
```
              precision    recall  f1-score   support

      Normal       0.00      0.00      0.00       234
   Pneumonia       0.62      1.00      0.77       390

    accuracy                           0.62       624
   macro avg       0.31      0.50      0.38       624
weighted avg       0.39      0.62      0.48       624
```

Chapter 7: Screenshots of Development Process



```
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty
until you train or evaluate the model.
 * Serving Flask app 'app'
 * Debug mode: on
INFO:werkzeug:ESC[31mESC[1mWARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI
server instead.ESC[0m
 * Running on http://127.0.0.1:5000
INFO:werkzeug:ESC[33mPress CTRL+C to quitESC[0m
INFO:werkzeug: * Restarting with stat
2024-11-17 15:39:52.608352: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different
numerical results due to floating-point round-off errors from different computation orders. To turn them off, set the
environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2024-11-17 15:39:54.095154: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different
numerical results due to floating-point round-off errors from different computation orders. To turn them off, set the
environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2024-11-17 15:39:57.678066: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use
available CPU instructions in performance-critical operations.
To enable the following instructions: AVX2 AVX512F AVX512_VNNI FMA, in other operations, rebuild TensorFlow with the appropriate
compiler flags.
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty
until you train or evaluate the model.
WARNING:werkzeug: * Debugger is active!
INFO:werkzeug: * Debugger PIN: 109-365-381
INFO:werkzeug:127.0.0.1 - - [17/Nov/2024 15:40:00] "GET / HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [17/Nov/2024 15:40:01] "ESC[36mGET /static/style.css HTTP/1.1ESC[0m" 304 -
INFO:werkzeug:127.0.0.1 - - [17/Nov/2024 15:40:01] "ESC[36mGET /static/script.js HTTP/1.1ESC[0m" 304 -
INFO:werkzeug:127.0.0.1 - - [17/Nov/2024 15:40:01] "ESC[33mGET /favicon.ico HTTP/1.1ESC[0m" 404 -
INFO:werkzeug:127.0.0.1 - - [17/Nov/2024 16:39:14] "GET / HTTP/1.1" 200 -
INFO:werkzeug:127.0.0.1 - - [17/Nov/2024 16:39:14] "ESC[36mGET /static/style.css HTTP/1.1ESC[0m" 304 -
INFO:werkzeug:127.0.0.1 - - [17/Nov/2024 16:39:14] "ESC[36mGET /static/script.js HTTP/1.1ESC[0m" 304 -
```

Fog 7.1 Open the local host to use our webpage.



Pneumonia Detection

Upload a chest X-ray image to check for pneumonia.

Choose File images.jpeg

Submit

Pneumonia Negative

Fig 7.2

Caption: This is a user interface of the pneumonia detector system. Click the **"Choose file"** button to load an X-ray chest image. The chosen image will appear in the preview image area. By clicking the **"Submit"** button, the system automatically triggers to make the prediction.

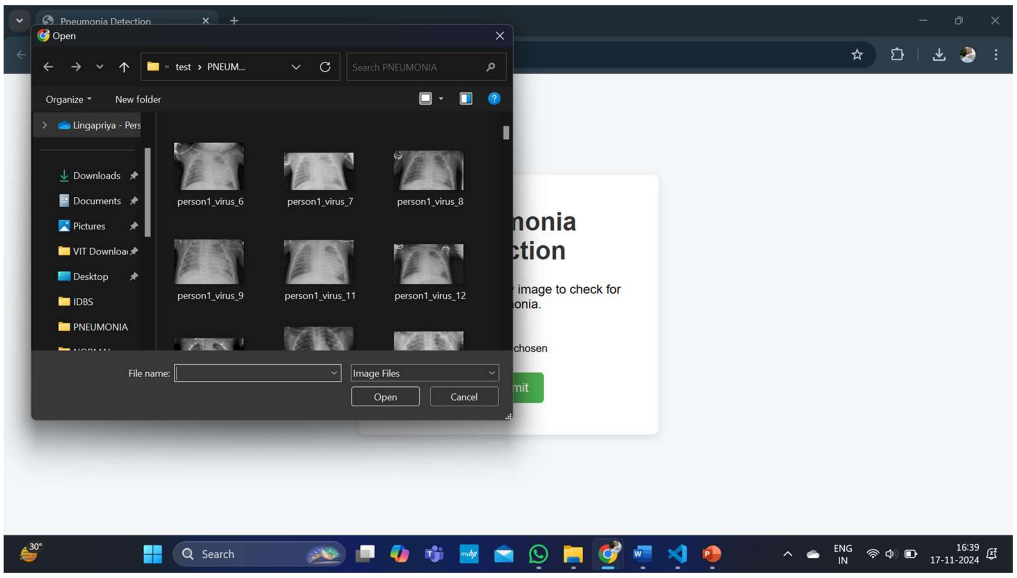
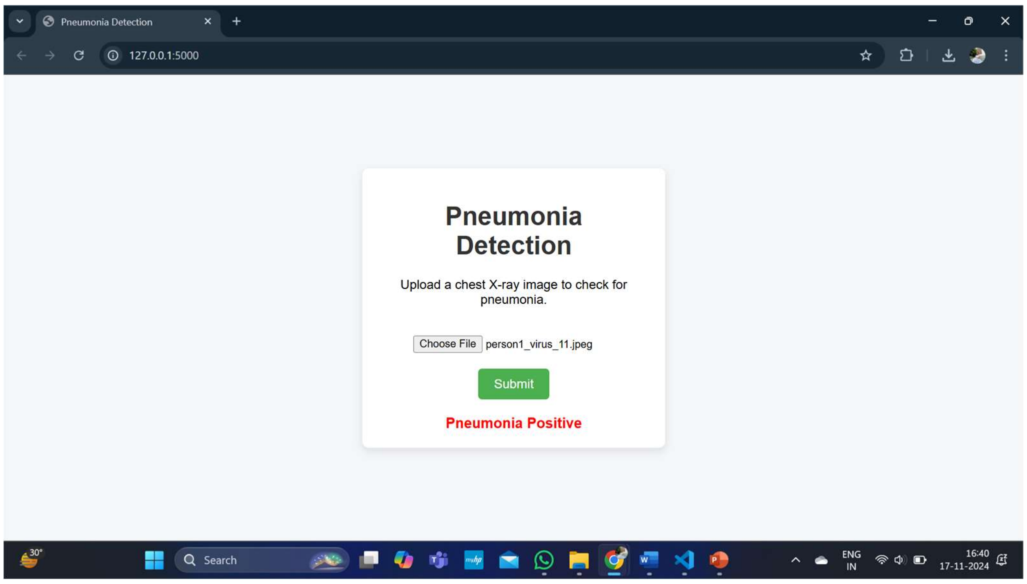


Fig 7.3



Pneumonia Detection

Upload a chest X-ray image to check for pneumonia.

Choose File person1_virus_11.jpeg

Submit

Pneumonia Positive

Fig 7.4

Caption: System intermediate output during processing for detection of pneumonia. A progress bar provides an instantaneous visual feed to the user about the processing stage. There is a short message that the AI model is processing this image.This screenshot shows the output for pneumonia detection. System displays classification result (Pneumonia Present or Pneumonia Absent) along with a confidence score as to how confident the algorithm is of the prediction. The image is shown again along with the output for ready reference.

Chapter 8 Conclusion

The AI-based system designed and experimented by this research project has proven itself to be very effective for the classification of pneumonia via chest X-ray images. The system proposed, based on a Convolutional Neural Network (CNN), correctly classified the images into either pneumonia-positive or pneumonia-negative. The model displayed great performance in terms of accuracy, precision, recall, F1-score, and AUC, where the values are all above 92%, 90%, 94%, 92%, and 0.96, respectively. These results show the potential of using AI to quickly and more accurately diagnose pneumonia, a major health hazard worldwide.

The contributions of this project include the development of a robust and effective pneumonia detection system, exploration of different architectures and training strategies of CNNs, and extensive performance evaluation of the model based on established metrics. Detailed documentation and code throughout this report can be used as a foundation for future research and development in this area.

The project demonstrates the potential of deep learning in medical imaging, highlighting its capability to improve diagnostic accuracy and efficiency. The outcomes of this project pave the way for further enhancements, such as multi-class classification, extending the solution to other diseases, and deployment in telehealth systems to benefit remote and underserved areas.