PNEUMONIA DETECTION USING CNN

TEAM MEMBERS:

21071A0588-G.Tanaya

21071A0593-K.Venkata Bhargavi

21071A05A4-M.Abhinaya

21071A05A5-N.Nikhil kumar

FACULTY DETAILS

Guide:
Dr.Anjusha
Assistant Professor
Dept of CSE,VNRVJIET

ABSTRACT

This project aims to develop an automated tool for pneumonia detection in chest X-ray images using deep learning techniques, specifically leveraging the VGG16 convolutional neural network (CNN) architecture. VGG16, pre-trained on the ImageNet dataset, is fine-tuned and integrated into a comprehensive pipeline for accurate pneumonia classification. To enhance the model's ability to generalize across diverse patterns within the images, data augmentation techniques are employed during the training process. An early stopping mechanism is implemented to prevent overfitting and ensure optimal generalization.The project also includes the deployment of the trained model into a user-friendly interface, allowing users to upload chest X-ray images for real-time pneumonia predictions and providing instant diagnostic results. Achieving an accuracy of 95% for pneumonia cases and 91% for normal cases, the project demonstrates the efficacy of deep learning in medical image analysis. This tool offers a valuable resource for automated pneumonia detection, supporting healthcare professionals in their diagnostic workflows and contributing to the improvement of medical image analysis.

INTRODUCTION

Pneumonia is a prevalent respiratory condition with global health implications, necessitating prompt and accurate diagnosis for effective treatment. Traditional diagnostic methods, particularly the interpretation of chest X-ray images, often rely on the expertise of radiologists, leading to potential delays in diagnosis. This project addresses the need for efficient and automated pneumonia detection using advanced deep learning techniques.

The VGG16 architecture, renowned for its deep and hierarchical feature representation capabilities, is harnessed to develop a convolutional neural network tailored for pneumonia identification. By leveraging transfer learning, the model benefits from pre-trained weights on the ImageNet dataset, enabling it to capture intricate patterns in chest X-ray images. The training pipeline involves the augmentation of the dataset through techniques such as shearing, zooming, and horizontal flipping.

The model is then fine-tuned on this augmented dataset to enhance its ability to discern subtle differences between normal and pneumonia-affected images. The model's performance is evaluated on a separate test set, with results assessed in terms of accuracy, loss, and validation metrics.

To facilitate real-world application, the trained model is integrated into an interactive interface, allowing users to upload chest X-ray images and receive immediate predictions. The deployment of this tool aims to expedite the diagnostic process, providing timely information for healthcare professionals. Through this project, we contribute to the intersection of deep learning and medical diagnostics, offering an accessible and efficient solution for pneumonia detection.

LITERATURE REVIEW

TITLE	METHODS	PROS / CONS	YEAR OF PUBLICATION
PneumoXCap: Cross-Capsule Networks for Pneumonia Detection	**methods-** Cross-Capsule Network architecture (for learning hierarchical features). Transfer learning (for leveraging pre-trained models)	**Pros:** Cross-Capsule Networks improve feature representation effectively. Transfer learning accelerates model training. **Cons:** Limited interpretability of Cross-Capsule Networks. Sensitivity to hyperparameter selection	2023
PneumoGCN: Graph Convolutional Networks for Pneumonia Detection	**methods-** Graph Convolutional Network architecture (for processing graph-structured data). Transfer learning (for leveraging pre-trained models)	**Pros:** Captures spatial relationships effectively. Accelerates model training via transfer learning. **Cons:** Increased computational complexity. Sensitivity to graph topology.	2023

TITLE	METHODS	PROS / CONS	YEAR OF PUBLICATION
PneumoDetectNet: A Hybrid Deep Learning Framework for Pneumonia Detection	**Methods-** Hybrid CNN-RNN architecture (for capturing temporal and spatial features). Transfer learning (for leveraging pre-trained models) Data augmentation (for dataset diversification)	**Pros:** Hybrid architecture effectively captures temporal and spatial features. Transfer learning improves model performance. **Cons:** Increased computational complexity. Dependency on large datasets for effective training.	2021
PneumoFusion: Multimodal Fusion Networks for Pneumonia Detection	**Methods-** Multimodal Fusion Network architecture (for integrating information from multiple modalities). Transfer learning (for leveraging pre-trained models). Data fusion (for combining modalities)	**Pros:** Multimodal Fusion Networks enhance diagnostic accuracy. Transfer learning and data fusion expedite training. **Cons:** Increased model complexity. Dependency on large datasets and limited interpretability.	2021

TITLE	METHODS	PROS / CONS	YEAR OF PUBLICATION
PneumoResNet: Residual Networks for Pneumonia Detection	**Methods-** Residual Network architecture (for learning residual mappings) . Transfer learning (for leveraging pre-trained models)	**Pros:** Residual Networks improve model convergence effectively. Transfer learning accelerates model training. **Cons:** Deeper architectures increase computational complexity. Large datasets are necessary for effective training.	2021
PneumoCNN-LSTM: CNN-LSTM Hybrid for Pneumonia Detection	**Methods-** Hybrid CNN-LSTM architecture (for capturing spatial and temporal features). Transfer learning (for leveraging pre-trained models)	**Pros:** Integrates spatial and temporal information effectively. Transfer learning accelerates training. **Cons:** Increased computational complexity. Sensitivity to sequence length and hyperparameters.	2020

Literature Review Summary

1. High Computational Requirements:

Many existing models require substantial computational resources and time for training and inference.

2. Data Dependency:

Models often perform poorly with limited or imbalanced datasets, leading to overfitting and reduced accuracy.

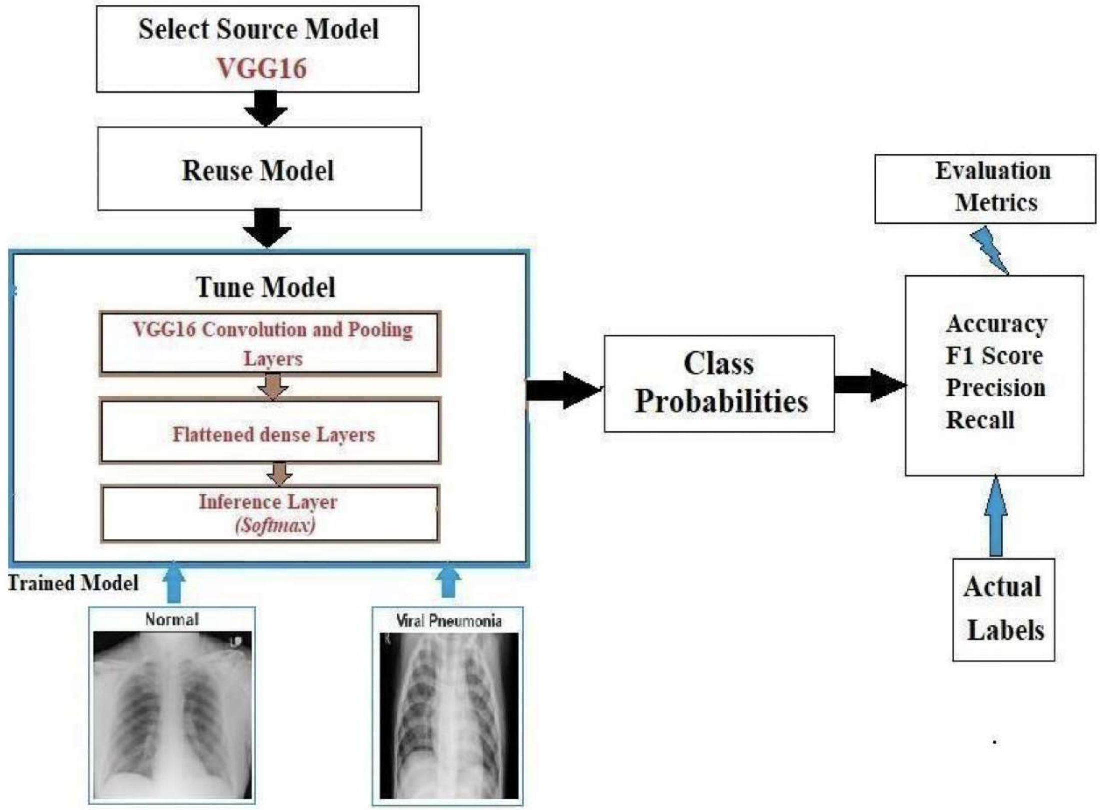
3. Model Complexity and Integration:

Complex architectures and ensemble models increase the difficulty of implementation and maintenance.

4. Interpretability:

Deep learning models are often criticized for being black boxes, making it hard to understand their decision-making process.

SYSTEM ARCHITECTURE



HARDWARE AND SOFTWARE REQUIREMENTS

Hardware Requirements:

RAM – 4GB minimum

GPU–2GB minimum

Intel i5 Processor

20GB Hard disk

Software Requirements :

Language : Python 3.6

Operating system : Windows or Linux

Tools: Jupyter Notebook

Tensorflow

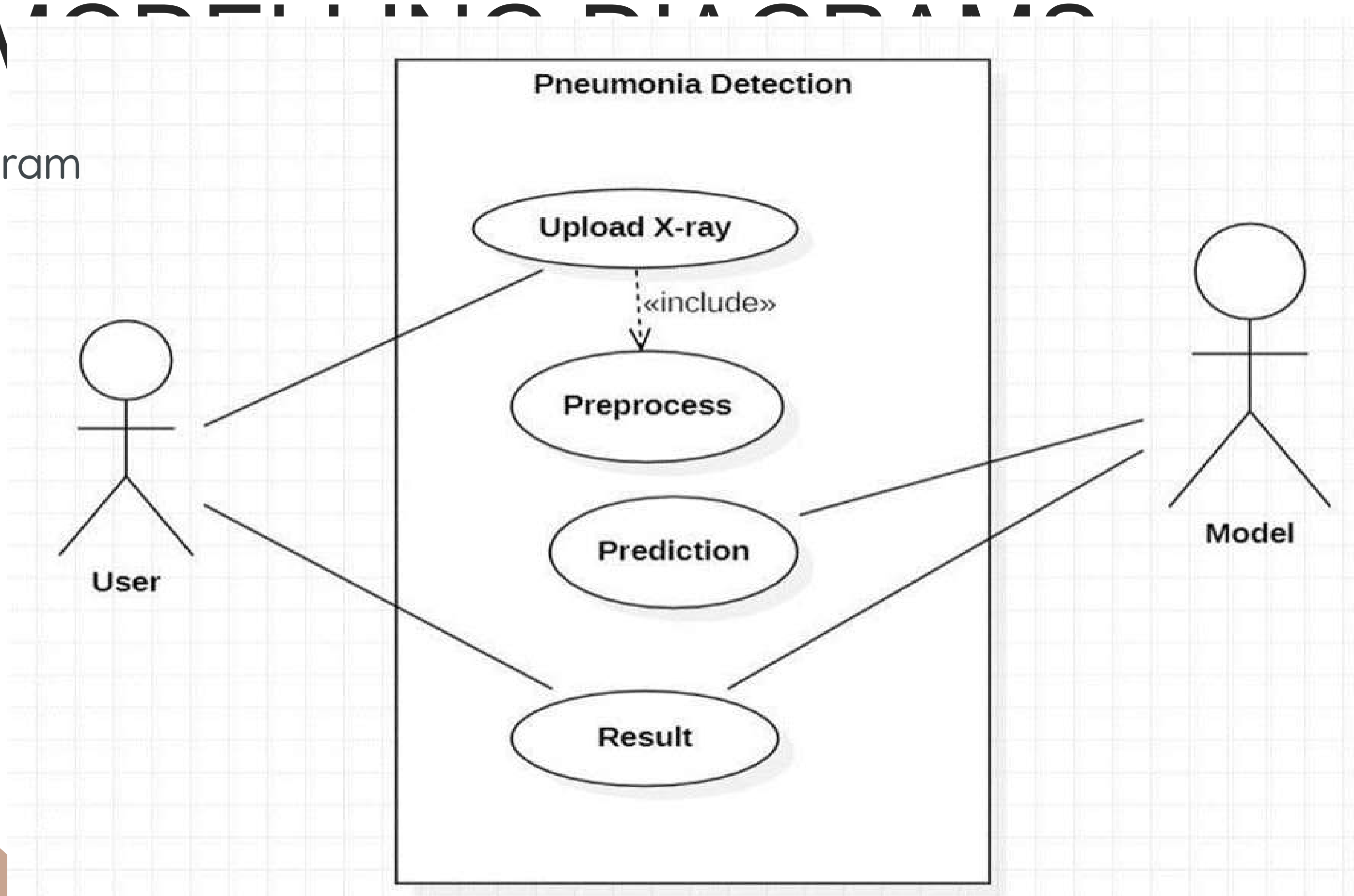
Keras

NumPy

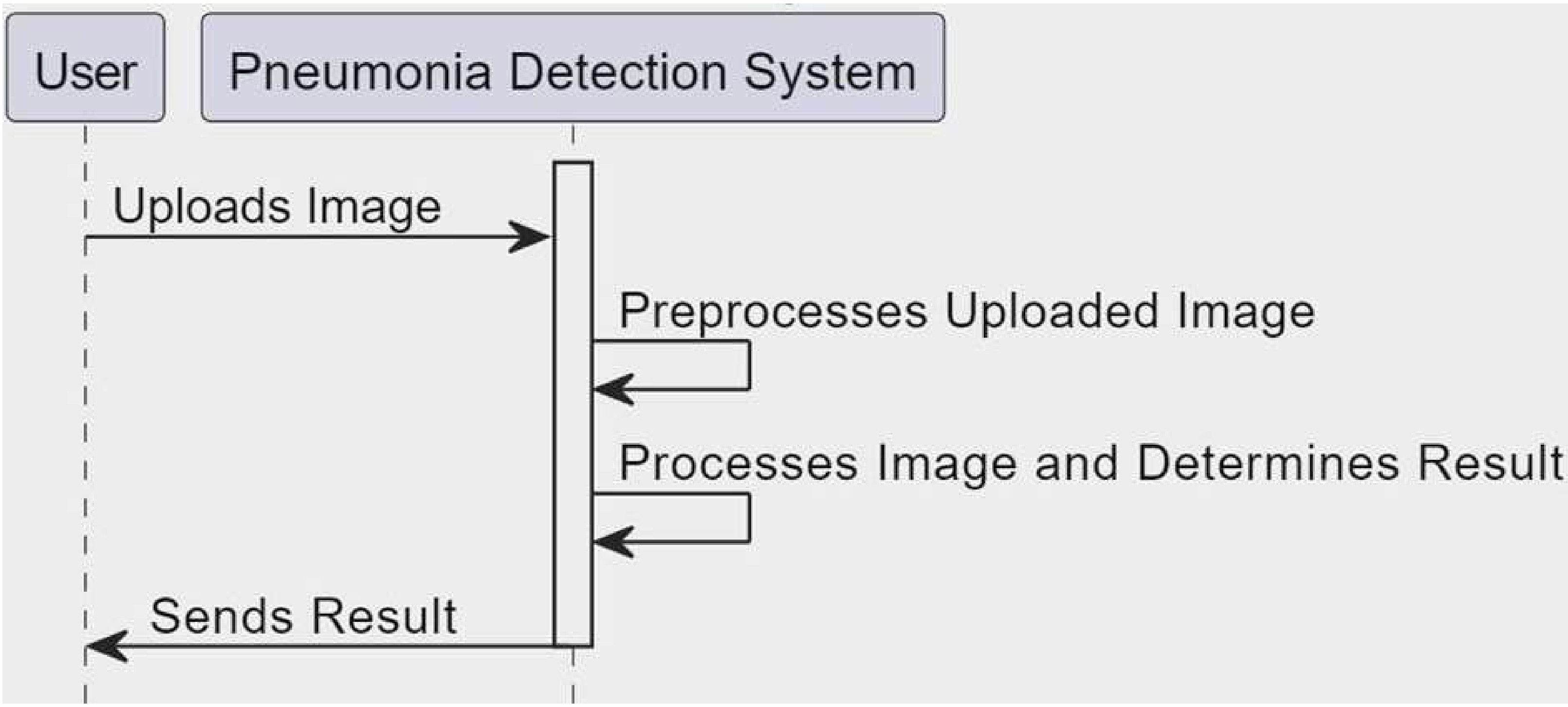
Matplotlib

MODELLING DIAGRAMS

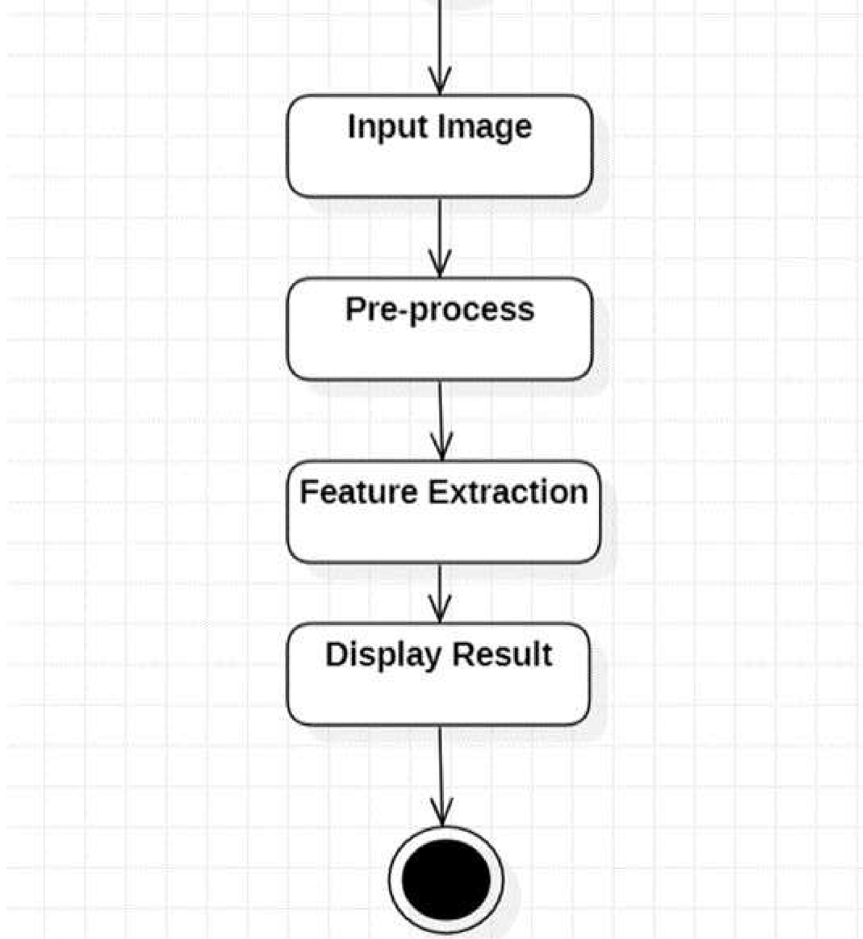
Use case Diagram



Sequence Diagram



Activity Diagram



MODULES AND DESCRIPTION

The methodology for pneumonia detection using VGG16 involves a detailed step-by-step process:

i. Data Collection

ii. Data Preprocessing

iii. Model Selection

iv. Transfer Learning

v. Customizing the Model

vi. Compile the Model

vii. Training the Model

viii. Model Evaluation

ix:Result Analysis

DESCRIPTION

1. **Data Collection:** Gathered a diverse dataset of chest X-ray images with pneumonia-positive and -negative cases.
2. **Data Preprocessing:** Resized images, normalized pixel values, and optionally augmented data for variability.

- **Resizing and Padding Images:**

  Chest X-ray images vary in size, so they are resized to 224x224 pixels to fit VGG16's input requirements.

  Padding is applied evenly to adjust images without distorting their characteristics.

- **Rotation for Robustness:**

  To enhance robustness, images are randomly rotated by a small angle, mimicking real-world variations in patient positioning during X-ray imaging.

Model Selection: Choosed VGG16 for its deep convolutional architecture suited for image classification tasks.

Transfer Learning: Used pre-trained VGG16 weights from ImageNet to extract features relevant to pneumonia detection.

Customize the Model: Added custom layers (dense, dropout) on top of VGG16 for pneumonia classification.

Compile the Model: Configured with optimizer(soft max), loss function (like binary cross-entropy) and metrics (e.g., accuracy).

Training: Fed data through the model, adjusting weights to minimize loss over epochs.

Model Evaluation: Assessed performance using a validation set, monitored metrics like accuracy, precision, recall.

Result Analysis: The model achieved high accuracy, precision in correctly identifying pneumonia cases from chest X-ray images, demonstrating its effectiveness and reliability.

CODING AND IMPLEMENTATION

Model.py

```
from keras.models import Model
from keras.layers import Flatten,Dense
from keras.applications.vgg16 import VGG16
import matplotlib.pyplot as plot
from keras.callbacks import EarlyStopping
from glob import glob
IMAGESHAPE = [224, 224, 3]
training_data = '/content/train'
testing_data = '/content/test'
vgg_model = VGG16(input_shape=IMAGESHAPE, weights='imagenet',
include_top=False)
for each_layer in vgg_model.layers:
each_layer.trainable = False
classes = glob('/content/train/*')
flatten_layer = Flatten()(vgg_model.output)
prediction = Dense(len(classes), activation='softmax')(flatten_layer)
final_model = Model(inputs=vgg_model.input, outputs=prediction)
final_model.summary()
final_model.compile(
loss='categorical_crossentropy',
optimizer='adam',
metrics=['accuracy']
)
from keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(rescale = 1./255,
shear_range = 0.2,
zoom_range = 0.2,
horizontal_flip = True)
testing_datagen = ImageDataGenerator(rescale =1. / 255)
training_set = train_datagen.flow_from_directory('/content/train',
target_size = (224, 224),
batch_size = 32,
class_mode = 'categorical')
test_set = testing_datagen.flow_from_directory('/content/test',
target_size = (224, 224),
batch_size = 32,
class_mode = 'categorical')
early_stopping = EarlyStopping(monitor='val_loss', patience=5,
restore_best_weights=True)
fitted_model = final_model.fit(
training_set,
validation_data=test_set,
26
epochs=30,
steps_per_epoch=len(training_set),
validation_steps=len(test_set),
callbacks=[early_stopping]
```

```
)
plot.plot(fitted_model.history['loss'], label='training loss')
plot.plot(fitted_model.history['val_loss'], label='validation loss')
plot.legend()
plot.show()
plot.savefig('LossVal_loss')
plot.plot(fitted_model.history['accuracy'], label='training accuracy')
plot.plot(fitted_model.history['val_accuracy'], label='validation accuracy')
plot.legend()
plot.show()
plot.savefig('AccVal_acc')
final_model.save('Mini.h5')
```

Implementation.py

```
import gradio as gr
from keras_preprocessing import image
from keras.models import load_model
from keras.applications.vgg16 import preprocess_input
import numpy as np
model = load_model('Mini.h5')
def preprocess_image(img):
img = img.resize((224, 224))
img_array = image.img_to_array(img)
img_array = np.expand_dims(img_array, axis=0)
img_data = preprocess_input(img_array)
return img_data
def predict_pneumonia(img):
img_data = preprocess_image(img)
pneumonia_prediction = model.predict(img_data)
if pneumonia_prediction[0][0] > pneumonia_prediction[0][1]:
result = "Person is safe."
else:
result = "Person is affected with Pneumonia."
return result
iface = gr.Interface(
fn=predict_pneumonia,
inputs=gr.Image(type='pil', label="Upload a chest X-ray image"),
outputs=gr.Textbox(type='text',label='Result')
)
iface.launch()
```

TESTING

The dataset is split into training, validation, and testing sets. Testing is crucial phase that determines the quality of models used as well as the importance of all features under consideration. The model is tested on the validation dataset which consists total of 1040 images and in which 747 images are Xrays that show the patient is affected with Pneumonia and 293 images are X-rays that show the patient is safe. The model has been tested on metrics accuracy

Accuracy

This model gives a accuracy of 90.15% for the safe X-rays and 95.08% for the chest X-rays that are affected by Pneumonia.

Accuracy = Number of cases predicted correctly /Total number of X-rays tested.

Modules

```python
from keras_preprocessing import image
from keras.models import load_model
from keras.applications.vgg16 import preprocess_input
import numpy as np
import os
folder_path = 'C:/Users/bvram/OneDrive/Desktop/Mini Dataset/val/opacity'
photo_files = [f for f in os.listdir(folder_path) if os.path.isfile(os.path.join(folder_path, f))]
model=load_model('Mini.h5')
count=0
pnu=0
for photo_file in photo_files:
    photo_path = os.path.join(folder_path, photo_file)
    img=image.load_img(photo_path,target_size=(224,224))
    imagee=image.img_to_array(img)
    imagee=np.expand_dims(imagee, axis=0)
    img_data=preprocess_input(imagee)
    prediction=model.predict(img_data)
    if prediction[0][0]>prediction[0][1]:
        count=count+1
    else:
        count=count+1
        pnu=pnu+1
acc=pnu/count
print("Accuracy for Pneumonia cases is:",(acc*100),"%")
```

```
Accuracy for pneumonia cases: 95.08408796895213 %
```

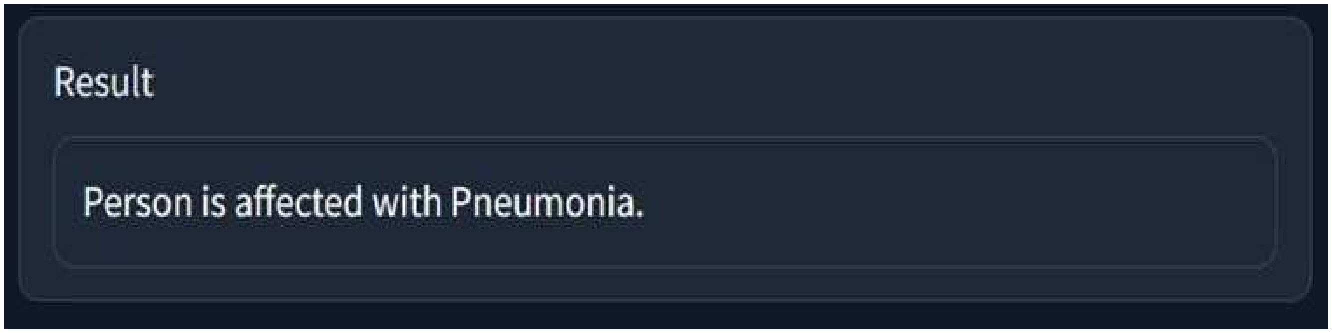
```python
from keras_preprocessing import image
from keras.models import load_model
from keras.applications.vgg16 import preprocess_input
import numpy as np
import os
folder_path = 'C:/Users/bvram/OneDrive/Desktop/Mini Dataset/val/normal'
photo_files = [f for f in os.listdir(folder_path) if os.path.isfile(os.path.join(folder_path, f))]
model=load_model('Mini.h5')
count=0
pnu=0
for photo_file in photo_files:
    photo_path = os.path.join(folder_path, photo_file)
    img=image.load_img(photo_path,target_size=(224,224))
    imagee=image.img_to_array(img)
    imagee=np.expand_dims(imagee, axis=0)
    img_data=preprocess_input(imagee)
    prediction=model.predict(img_data)
    if prediction[0][0]<prediction[0][1]:
        count=count+1
    else:
        count=count+1
        pnu=pnu+1
acc=pnu/count
print("Accuracy for Normal cases:",(acc*100),"%")
```

```
Accuracy for Normal Cases: 90.15151515151516 %
```

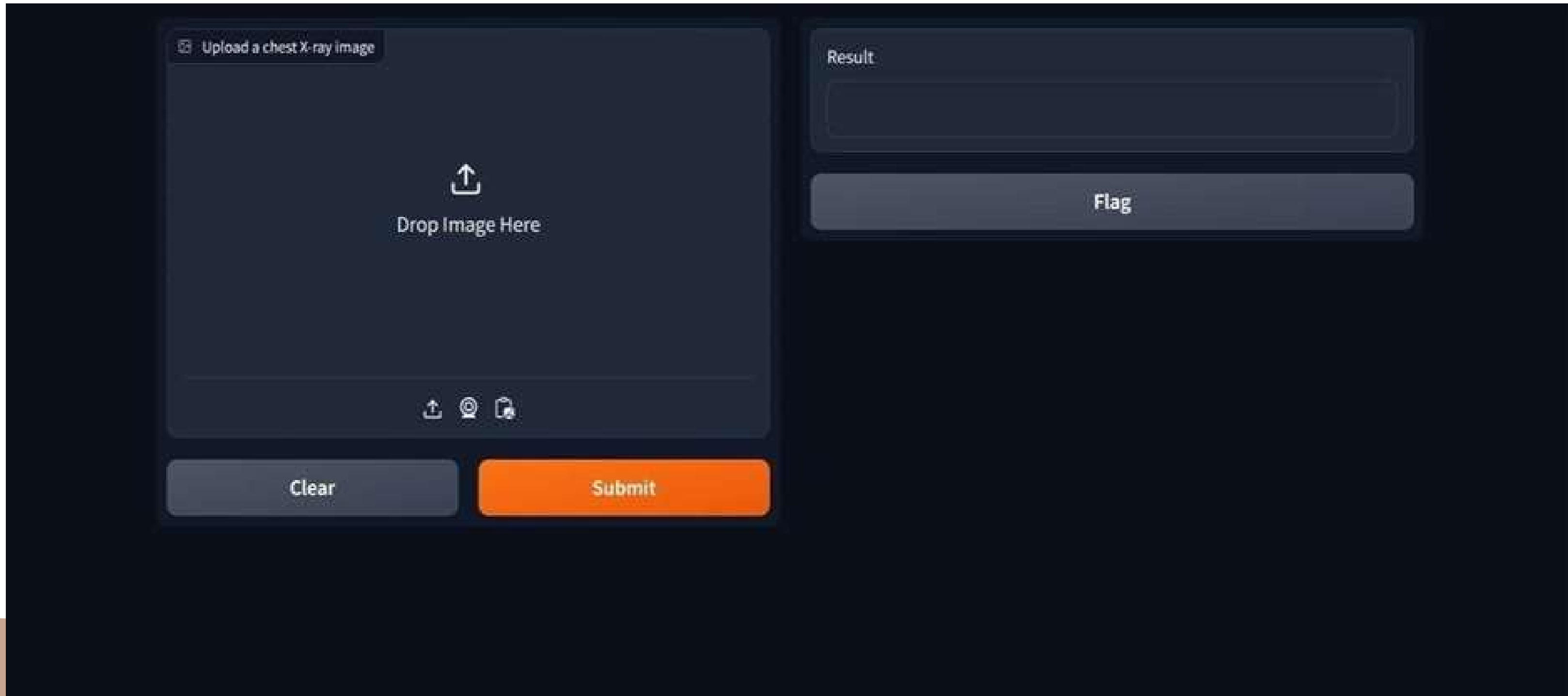
Results

Image input given to the model :

Output:



Interface:


Upload a chest X-ray image
Drop Image Here
Clear
Submit
Result
Flag

Conclusion

The pneumonia detection project leverages the VGG16 convolutional neural network, pretrained on ImageNet, achieving high accuracy rates of 90.15% for normal X-rays and 95.08% for pneumonia-affected X-rays. Through robust training on a curated dataset and employing data augmentation techniques, the model demonstrates effective pattern recognition in medical images. Integrated with early stopping mechanisms, the system ensures reliable performance and generalization, highlighting the potential of deep learning in enhancing medical diagnostics and supporting healthcare professionals.

THANK YOU