CHEST XRAY PNEUMONIA DETECTION

Course Code: CS354N
Course Instructor: DR.Aruna Tiwari

TEAM MEMBERS:
PRANAY D KUMAR 210001053
SANSKAR 210001063
VELPURU NAVYA 210001075

Github Link: https://github.com/Sanskar6877/ChestXRay_Pneumonia_Detection

Problem Statement:

CNN Models for Pneumonia Detection in Chest Xray Images:

- Objective: Detect pneumonia from pediatric chest Xray images using CNN models.
- Approach: Compare various CNN architectures and configurations, including pretrained models like Inceptionv3, ResNet50, VGG16, and VGG19.
- Goal: Distinguish pneumonia from nonpneumonia cases to contribute to early detection and intervention strategies.
- Impact: Potentially reduce mortality rates among children, especially in regions with high prevalence rates of pneumonia.

Data Collection and Preprocessing:

- Dataset Overview: The dataset, "Chest X-Ray Images (Pneumonia)," sourced from Kaggle, contains 5416 training, 300 validation, and 624 testing images, totalling 2 GB. These grayscale images are 64 * 64 pixels and categorized into Normal, Bacterial Pneumonia, and Viral Pneumonia. Access the dataset(https://drive.google.com/drive/folders/1DA2ScHT5ZoxLHLhDrB5EiDY1iB4dJDS-?usp=drive_link).

- Preprocessing: Images are loaded, converted to grayscale, and resized to ensure uniformity. They are then split into training, validation, and testing sets, labeled ('PNEUMONIA' or 'NORMAL') for supervised learning.

- Data Preparation: Pixel values are normalized, and data augmentation techniques like rotation and flipping are applied to enhance model robustness. Consistent augmentation parameters are ensured during training, enhancing generalization capability.

Flow Chart of the Project :

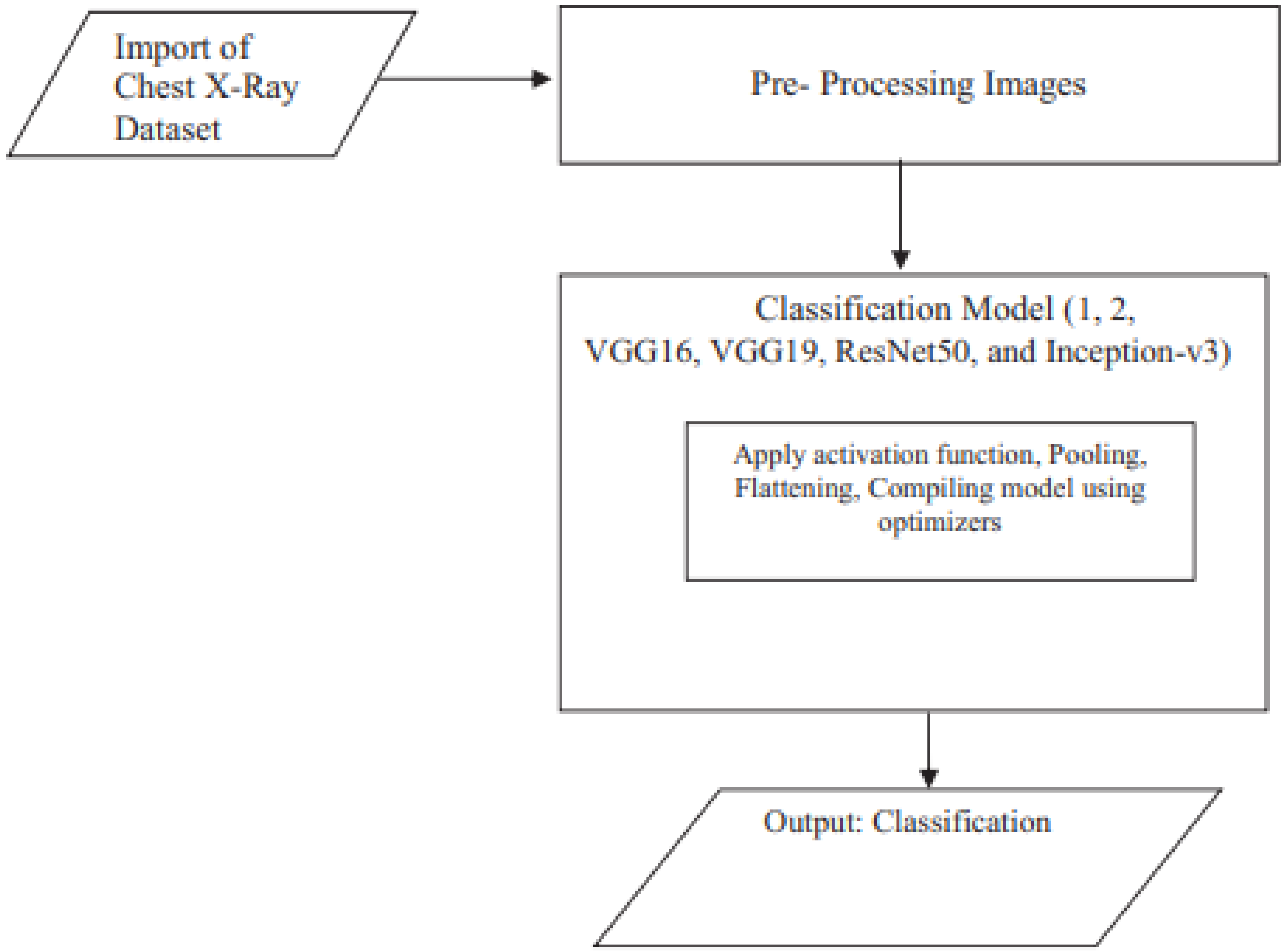


Fig 1: Stages of the pneumonia Detection Project

MODELS:

Custom CNN Model:

- 3 convolutional layers: 32, 64, and 128 feature maps.
- Max-pooling: 2x2 after each convolutional layer.
- Dense layers: 256 output perceptrons, 2 output perceptrons.
- Dropout layer included.
- Learning rate: 0.0001.

Inception-v3:

- CNN with 42 layers introduces auxiliary classifiers and batch normalization.
- Utilizes factorization to reduce parameters.
- Learning rate: 0.000001.

ResNet50:

- Utilizes shortcut connections to address issues.
- Learning rate: 0.000001.

VGG16:

- 16-layer CNN with 3x3 kernel-sized filters.
- Employs dropout after each dense layer.
- Learning rate: 0.0001.

VGG19:

- Variant of VGG16 with 19 layers.
- Learning rate: 0.00001.

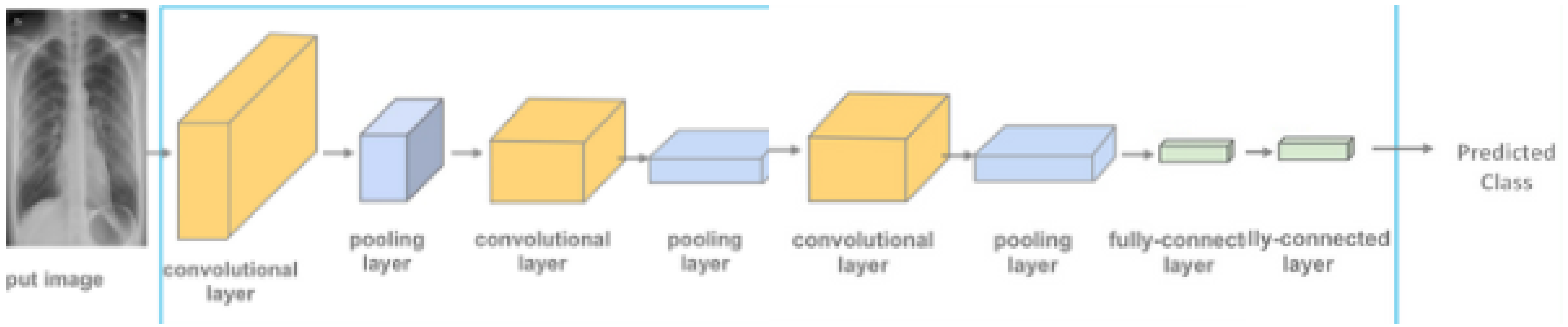


Fig 2: CNN Architecture (model 1)

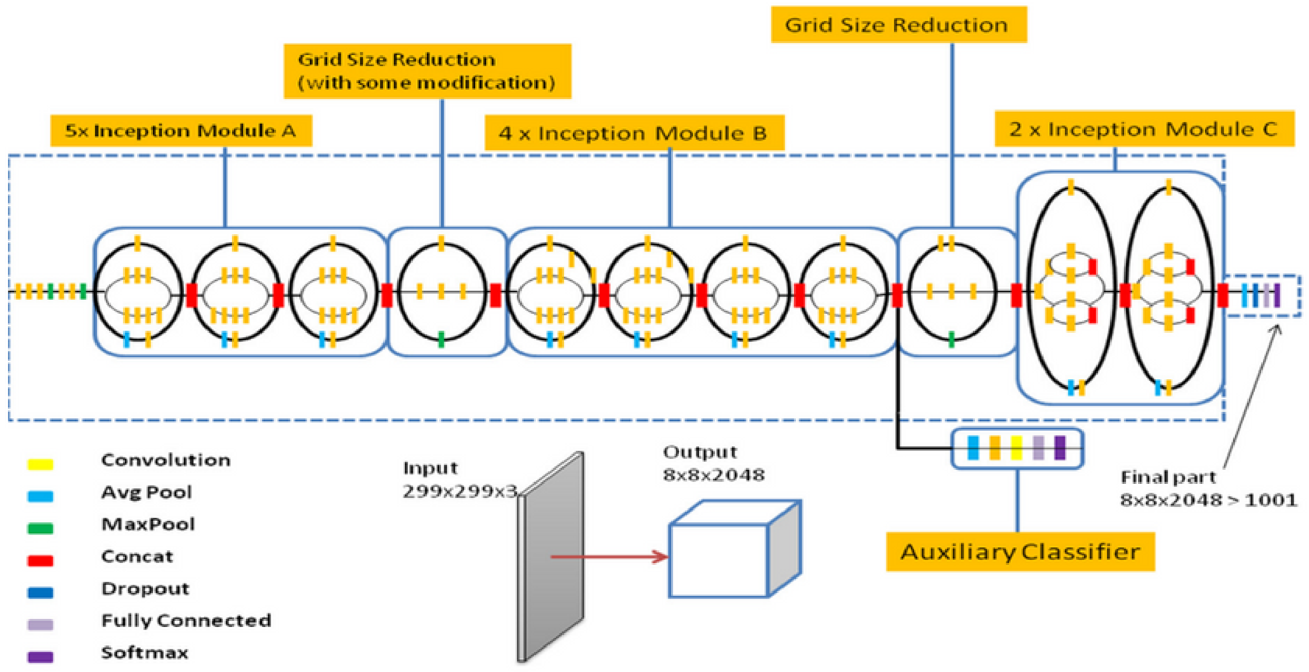


Fig 3: Inceptionv3 Architecture (model 2)

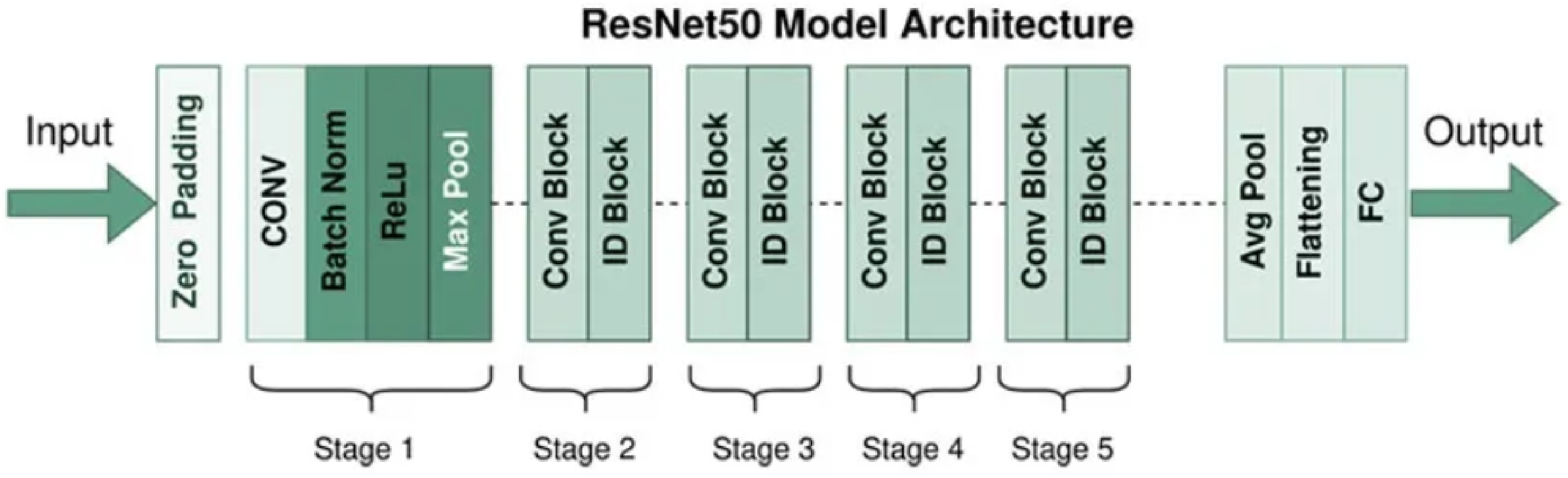


Fig 4: ResNet 50 Architecture (model 3)

ALGORITHM:

Step 1: Convolve 64x64 images with 32 feature maps using ReLU activation.
Step 2: Pool the previous layer's output with 2x2 max pooling.
Step 3: Resize input to 64x64 and convolve with 64 feature maps using ReLU activation.
Step 4: Pool the previous layer's output with 2x2 max pooling.
Step 5:
a) (Model 1): Convolve input with 128 feature maps, ReLU activation, and 2x2 max pooling. Flatten output.
b) (InceptionV3): Utilize pre-trained InceptionV3 with ImageNet weights and global average pooling.
c) (ResNet50): Utilize pre-trained ResNet50 with ImageNet weights and global average pooling.
d) (VGG16 and VGG19): Utilize pre-trained VGG16 or VGG19 with ImageNet weights and fully connected layers.
Step 6: Pass flattened output (Model 1) or last layer's output through a dense layer with 256 perceptrons and ReLU activation.
Step 7: Compile model using Adam optimizer (learning rate: 0.001), categorical cross-entropy loss, and softmax activation for binary classification.

EXPERIMENTATION AND RESULTS:

Validation and training loss:

Fig 5: plot of model 1

Fig 5: plot of Inceptionv3

Fig 5: plot of RESNET50

Fig 5: plot of VGG16

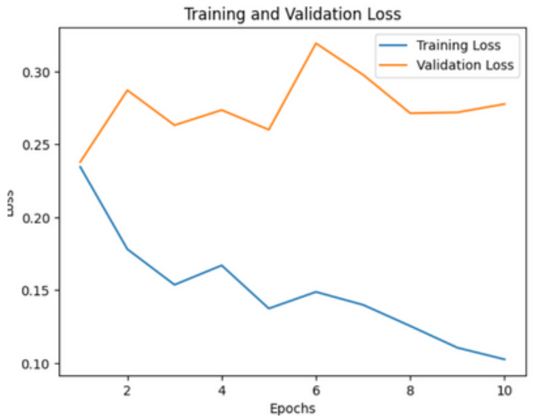
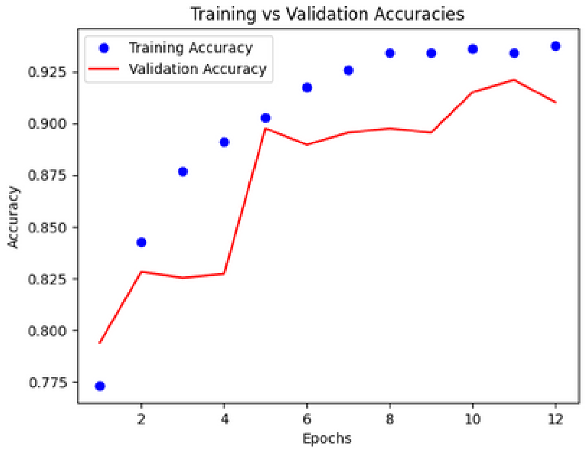


Fig 5: plot of VGG19

<u>Validation and training accuracy:</u>

Fig 5: plot of model 1



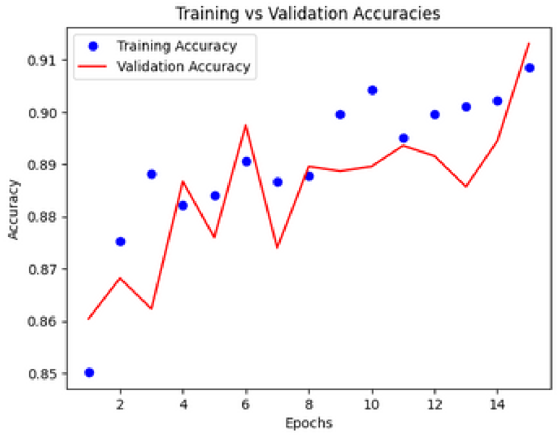
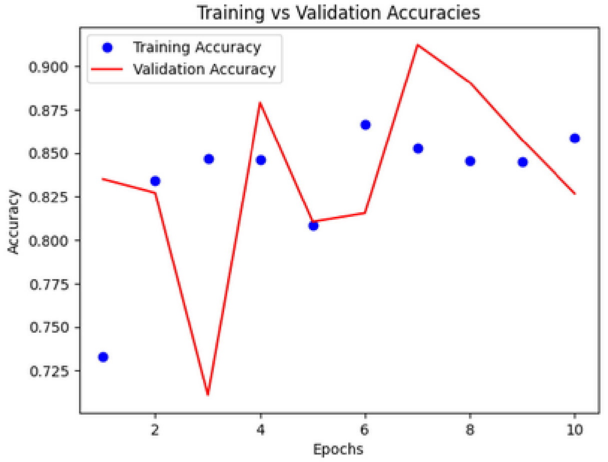


Fig 5: plot of Inceptionv3

Fig 5: plot of RESNET50



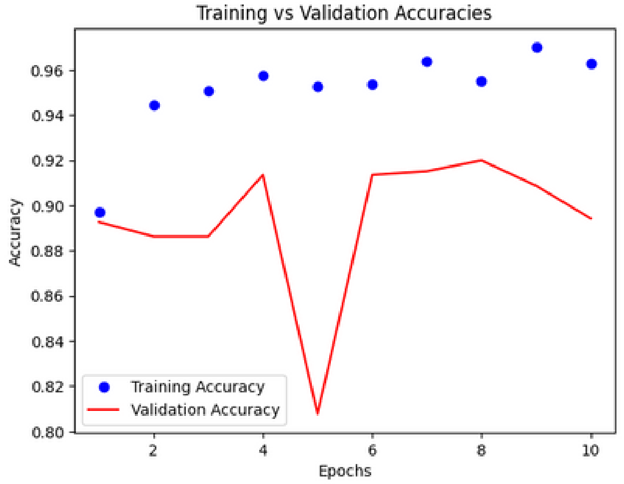


Fig 5: plot of VGG16

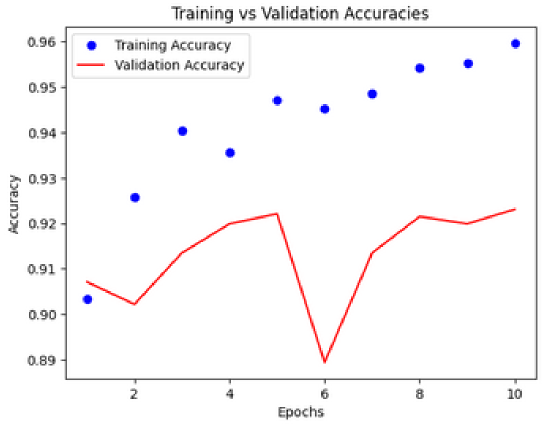


Fig 5: plot of VGG19

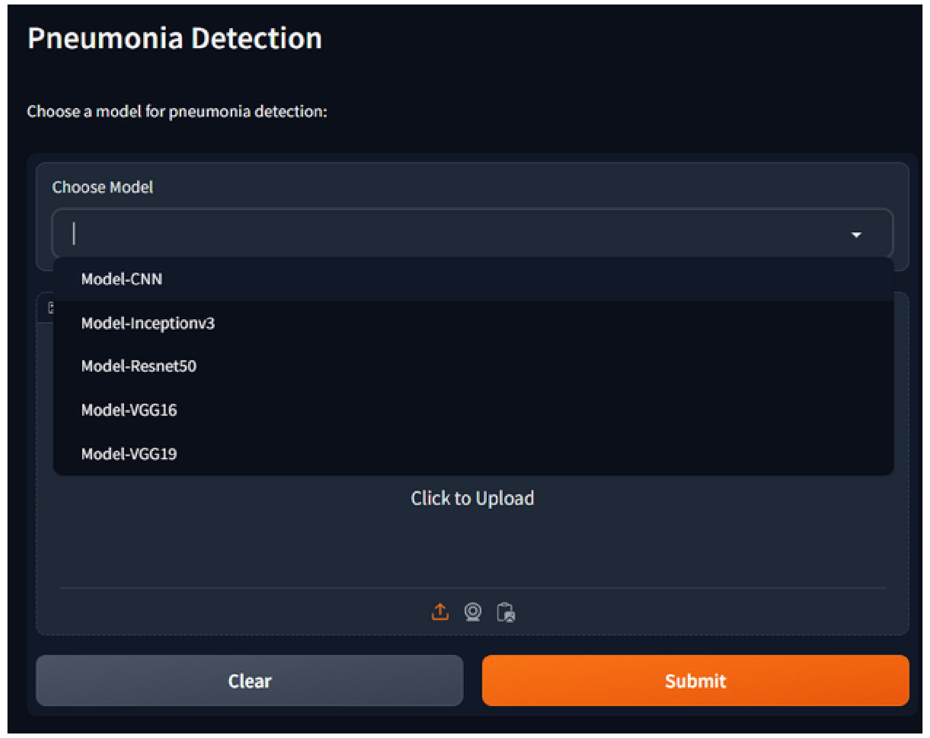
Evaluation metrics of Testing data:

Model	Accuracy	Precision	F1 score	Recall
model 1	0.90224	0.92869	0.92263	0.912232
Inceptionv3	0.82211	0.82869	0.832232	0.84263
Resnet50	0.73237	0.73458	0.81289	0.73659
VGG16	0.89423	0.91035	0.95698	0.92365
VGG19	0.92307	0.93248	0.91556	0.90644

Testing through User interface:

Below are some user testing interface images (includes input and output):

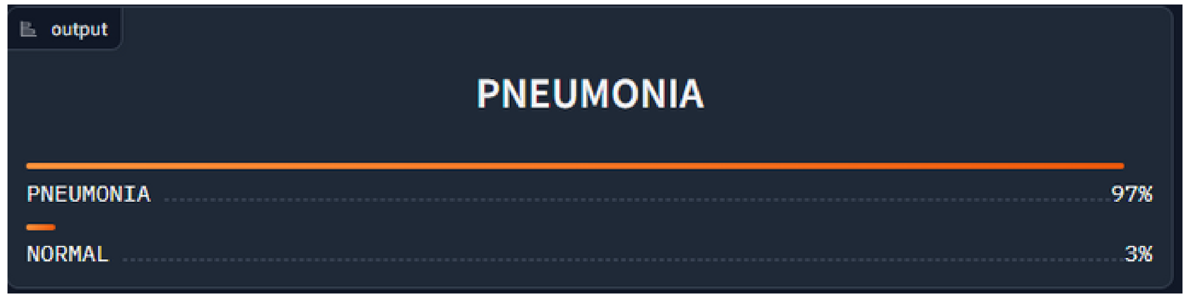
User Interface

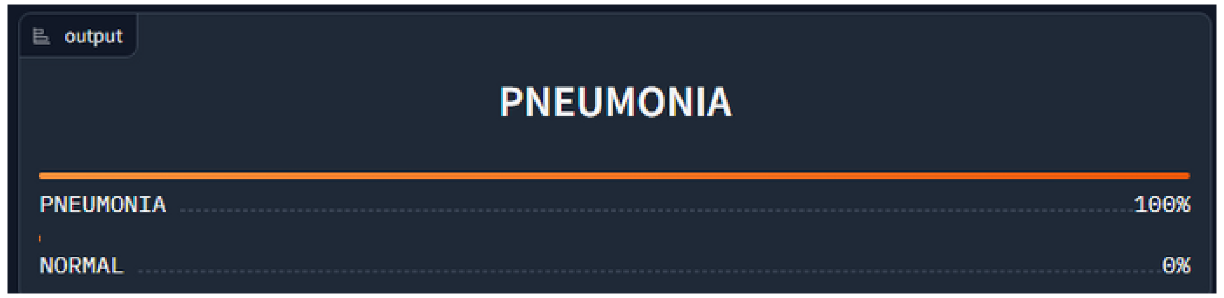


Pneumonic input image

Results are as below for above input:

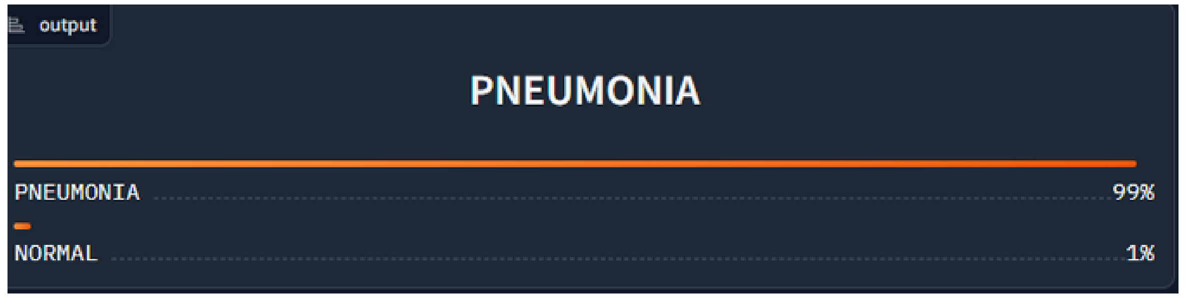
Model 1

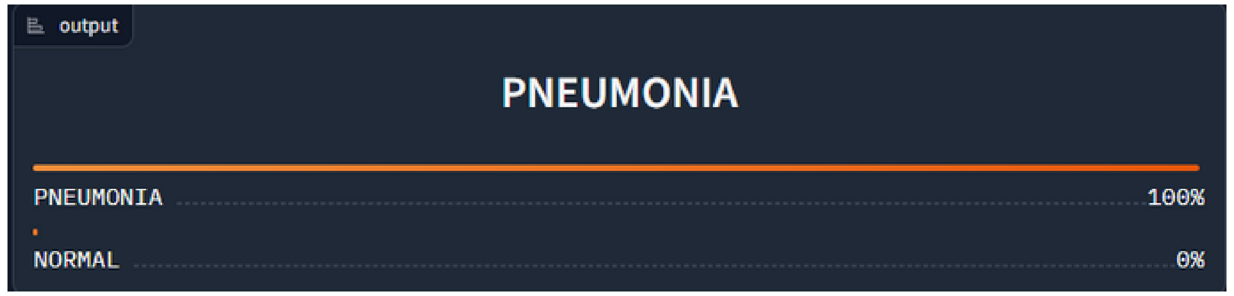




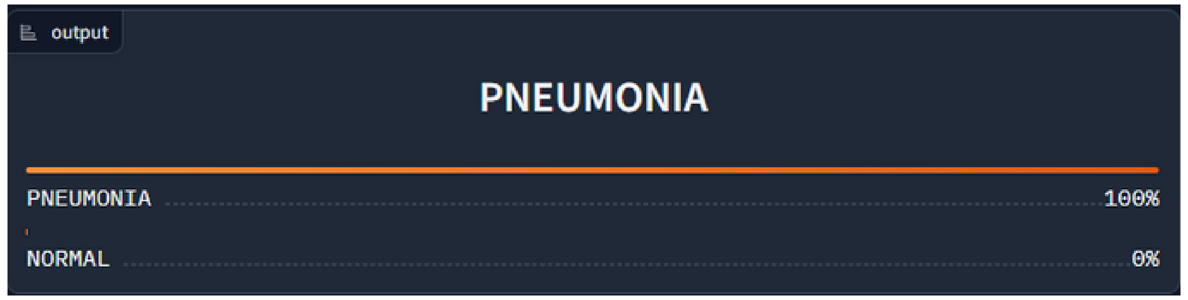
Inceptionv3

ResNet50





VGG16



VGG19

Conclusion:

- Developed and evaluated multiple deep-learning models for pneumonia detection using chest X-ray images.
- Experimented with various architectures including CNN, InceptionV3, ResNet50, VGG16, and VGG19.
- Each model was trained, validated, and tested on a dataset of chest X-ray images containing pneumonia and normal cases.
- Precision, recall, and F1-score metrics were calculated to evaluate the models' performance in classifying pneumonia cases.
- Deployed a user-friendly web application using Gradio for pneumonia detection, accessible via (https://github.com/Sanskar6877/ChestXRay_Pneumonia_Detection).
- Another multimodel pneumonia detection web application is available (https://huggingface.co/spaces/vnavya2004/PNEUMONIA_DETECTION_MULTIMODEL), allowing users to compare and analyze results conveniently.
- Overall, the project demonstrates the potential of deep learning models in aiding pneumonia detection from chest X-ray images, with practical applications through web interfaces.

Project related links:

- Github Link: https://github.com/Sanskar6877/ChestXRay_Pneumonia_Detection
- Dataset Link: https://drive.google.com/drive/folders/1DA2ScHT5ZoxLHLhDrB5EiDY1iB4dJDS-?usp=drive_link
- User interface Link: https://huggingface.co/spaces/vnavya2004/PNEUMONIA_DETECTION_MULTIMODEL
- Reference Link: https://www.sciencedirect.com/science/article/pii/S0263224120305844

THANK YOU

Team members:

- 210001053
- 210001063
- 210001075