Enhanced Pneumonia Detection from Chest X-rays via Deep Learning

Tanvi Poddar
Information Technology
National Institute of Technology, Karnataka
Surathkal, India
tanvipoddar.221it071@nitk.edu.in

Upasana Nayak
Information Technology
National Institute of Technology Karnataka
Surathkal, India
upasananayak.221it075@nitk.edu.in

Uggumudi Sai Lasya Reddy
Information Technology
National Institute of Technology Karnataka
Surathkal, India
usailasya.221it074@nitk.edu.in

***Abstract*—Lung diseases refer to many lung disorders, such as pneumonia, tuberculosis, lung cancer, and many other breathing problems. Early disease detection is paramount to mitigate fatality and reduce chronic illnesses. Traditionally, these diseases can be identified through imaging techniques like Computed Tomography (CT) and chest X-ray images of the lungs. The accurate detection and identification of lung diseases is challenging due to the high degree of similarities between different lung diseases and variations in the same disease. This project, at the core, tries to bridge the gap between the fields of biomedical imaging and signals and systems by using advanced deep-learning techniques. A comprehensive evaluation of the proposed approach using publicly available datasets demonstrates a better performance compared to the existing approaches.**

***Index Terms*—Lung Disease Detection, Biomedical Imaging, Deep Learning**

I. Introduction

Pneumonia is a respiratory condition characterized by inflammation of the air sacs in the lungs, usually caused by an infection from bacteria, viruses, fungi, or other microorganisms. This illness can range from mild to severe and is particularly dangerous for certain populations, such as the elderly, young children, and individuals with weakened immune systems.

Detecting pneumonia accurately and swiftly is crucial for timely treatment and patient management. In medical diagnostics, imaging techniques, particularly chest X-rays and more recently, computed tomography (CT) scans, play a pivotal role in identifying pneumonia. However, analyzing these images manually can be time-consuming and requires expertise, leading to a growing interest in leveraging artificial intelligence, specifically Convolutional Neural Networks (CNNs), to aid in the detection of pneumonia.

CNNs are a type of deep learning model specifically designed for processing visual data, such as images. They are proficient in automatically learning features from raw pixel data and hierarchically extracting complex patterns, making them well-suited for image classification tasks.

In the context of pneumonia detection, CNNs can be trained on large datasets of chest X-ray or CT scan images. By learning from a diverse range of pneumonia-afflicted and normal (healthy) images, CNNs can discern subtle differences in lung patterns, densities, and abnormalities associated with pneumonia. The trained CNN model can then classify new, unseen chest X-ray images as either indicative of pneumonia or healthy lung conditions based on the learned patterns and features.

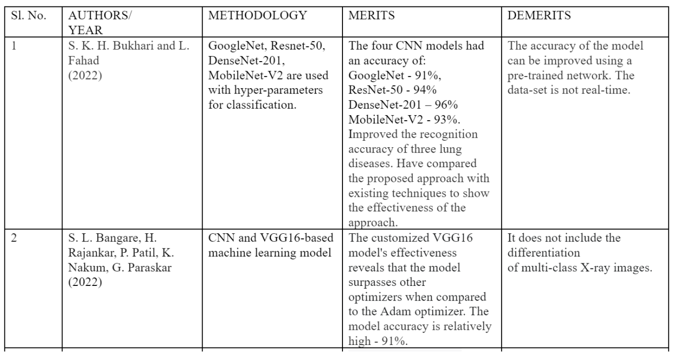
Utilizing CNNs for pneumonia detection offers several advantages:

1. Accuracy: CNNs can learn intricate patterns and features from medical images, potentially achieving high accuracy in identifying pneumonia-related abnormalities.
2. Efficiency: Automated analysis using CNNs can expedite the detection process, providing quicker assessments compared to manual inspection by healthcare professionals.
3. Scalability: Once trained, CNN models can be deployed and utilized across various healthcare settings, aiding in pneumonia detection in a scalable manner.
4. Assistive Tool: CNNs serve as an assisting tool for healthcare professionals, offering insights and supporting diagnostic decisions, potentially improving overall patient care.

In summary, CNNs represent a promising approach in the field of medical imaging for pneumonia detection. Their ability to analyze and interpret visual data enables faster and potentially more accurate identification of pneumonia-related abnormalities in chest X-ray or CT scan images, contributing to earlier diagnosis and improved patient outcomes.

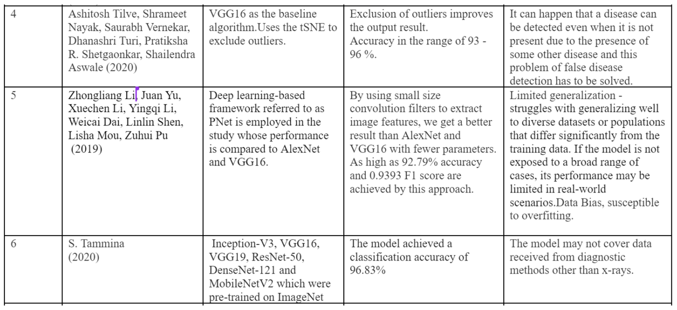
II. RELATED WORKS

The literature review covers a wide range of machine learning models used for disease detection, particularly pneumonia,



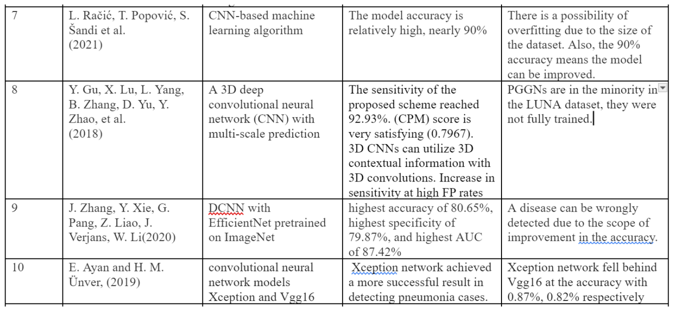
Sl. No.	AUTHORS/ YEAR	METHODOLOGY	MERITS	DEMERITS
1	S. K. H. Bukhari and L. Fahad (2022)	GoogleNet, Resnet-50, DenseNet-201, MobileNet-V2 are used with hyper-parameters for classification.	The four CNN models had an accuracy of: GoogleNet - 91%, ResNet-50 - 94% DenseNet-201 – 96% MobileNet-V2 - 93%. Improved the recognition accuracy of three lung diseases. Have compared the proposed approach with existing techniques to show the effectiveness of the approach.	The accuracy of the model can be improved using a pre-trained network. The data-set is not real-time.
2	S. L. Bangare, H. Rajankar, P. Patil, K. Nakum, G. Paraskar (2022)	CNN and VGG16-based machine learning model	The customized VGG16 model's effectiveness reveals that the model surpasses other optimizers when compared to the Adam optimizer. The model accuracy is relatively high - 91%.	It does not include the differentiation of multi-class X-ray images.

Fig. 1. Literature Review



4	Ashitosh Tilve, Shrameet Nayak, Saurabh Vernekar, Dhanashri Turi, Pratiksha R. Shetgaonkar, Shailendra Aswale (2020)	VGG16 as the baseline algorithm.Uses the tSNE to exclude outliers.	Exclusion of outliers improves the output result. Accuracy in the range of 93 - 96 %.	It can happen that a disease can be detected even when it is not present due to the presence of some other disease and this problem of false disease detection has to be solved.
5	Zhongliang Li, Juan Yu, Xuechen Li, Yingqi Li, Weicai Dai, Linlin Shen, Lisha Mou, Zuhui Pu (2019)	Deep learning-based framework referred to as PNet is employed in the study whose performance is compared to AlexNet and VGG16.	By using small size convolution filters to extract image features, we get a better result than AlexNet and VGG16 with fewer parameters. As high as 92.79% accuracy and 0.9393 F1 score are achieved by this approach.	Limited generalization - struggles with generalizing well to diverse datasets or populations that differ significantly from the training data. If the model is not exposed to a broad range of cases, its performance may be limited in real-world scenarios.Data Bias, susceptible to overfitting.
6	S. Tammina (2020)	Inception-V3, VGG16, VGG19, ResNet-50, DenseNet-121 and MobileNetV2 which were pre-trained on ImageNet	The model achieved a classification accuracy of 96.83%	The model may not cover data received from diagnostic methods other than x-rays.

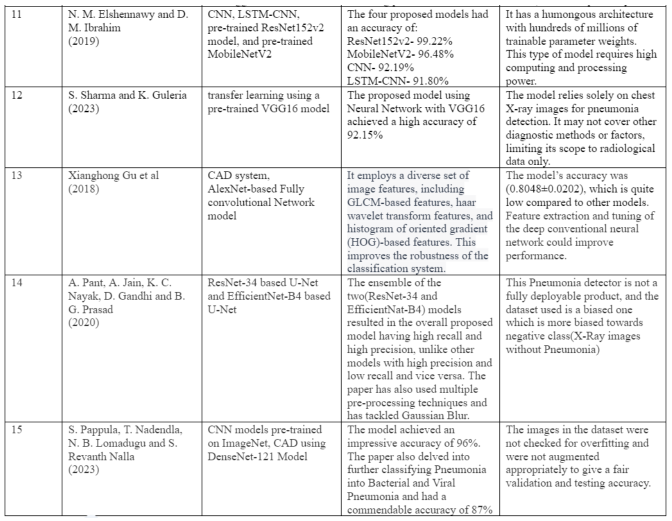
Fig. 2. Literature Review



7	L. Račić, T. Popović, S. Šandi et al. (2021)	CNN-based machine learning algorithm	The model accuracy is relatively high, nearly 90%	There is a possibility of overfitting due to the size of the dataset. Also, the 90% accuracy means the model can be improved.
8	Y. Gu, X. Lu, L. Yang, B. Zhang, D. Yu, Y. Zhao, et al. (2018)	A 3D deep convolutional neural network (CNN) with multi-scale prediction	The sensitivity of the proposed scheme reached 92.93%. (CPM) score is very satisfying (0.7967). 3D CNNs can utilize 3D contextual information with 3D convolutions. Increase in sensitivity at high FP rates	PGGNs are in the minority in the LUNA dataset, they were not fully trained.
9	J. Zhang, Y. Xie, G. Pang, Z. Liao, J. Verjans, W. Li(2020)	DCNN with EfficientNet pretrained on ImageNet	highest accuracy of 80.65%, highest specificity of 79.87%, and highest AUC of 87.42%	A disease can be wrongly detected due to the scope of improvement in the accuracy.
10	E. Ayan and H. M. Ünver, (2019)	convolutional neural network models Xception and Vgg16	Xception network achieved a more successful result in detecting pneumonia cases.	Xception network fell behind Vgg16 at the accuracy with 0.87%, 0.82% respectively

Fig. 3. Literature Review

using chest X-ray images. The models include GoogleNet, ResNet-50, DenseNet-201, MobileNet-V2, VGG16, custom CNNs, PNet, Inception-V3, VGG19, DenseNet-121, MobileNetV2, 3D CNN, EfficientNet, Xception, LSTM-CNN, ResNet152v2, and AlexNet-based Fully convolutional Network model. The models' accuracy ranges from around 80% to as high as 99.22%. Some models use pre-trained networks on ImageNet for improved performance. Various techniques like hyper-parameter tuning, exclusion of outliers, ensemble methods, and multi-scale prediction are employed to enhance the models' performance. However, several challenges are identified across the models, such as overfitting due to the size of the dataset, limited generalization to diverse datasets, data bias, and issues with false disease detection. Some models also struggle with real-time data and are limited to radiological data only. Furthermore, some models have been successful in classifying pneumonia into bacterial and viral



11	N. M. Elshennawy and D. M. Ibrahim (2019)	CNN, LSTM-CNN, pre-trained ResNet152v2 model, and pre-trained MobileNetV2	The four proposed models had an accuracy of: ResNet152v2- 99.22% MobileNetV2- 96.48% CNN- 92.19% LSTM-CNN- 91.80%	It has a humongous architecture with hundreds of millions of trainable parameter weights. This type of model requires high computing and processing power.
12	S. Sharma and K. Guleria (2023)	transfer learning using a pre-trained VGG16 model	The proposed model using Neural Network with VGG16 achieved a high accuracy of 92.15%	The model relies solely on chest X-ray images for pneumonia detection. It may not cover other diagnostic methods or factors, limiting its scope to radiological data only.
13	Xianghong Gu et al (2018)	CAD system, AlexNet-based Fully convolutional Network model	It employs a diverse set of image features, including GLCM-based features, haar wavelet transform features, and histogram of oriented gradient (HOG)-based features. This improves the robustness of the classification system.	The model's accuracy was (0.8048±0.0202), which is quite low compared to other models. Feature extraction and tuning of the deep conventional neural network could improve performance.
14	A. Pant, A. Jain, K. C. Nayak, D. Gandhi and B. G. Prasad (2020)	ResNet-34 based U-Net and EfficientNet-B4 based U-Net	The ensemble of the two(ResNet-34 and EfficientNat-B4) models resulted in the overall proposed model having high recall and high precision, unlike other models with high precision and low recall and vice versa. The paper has also used multiple pre-processing techniques and has tackled Gaussian Blur.	This Pneumonia detector is not a fully deployable product, and the dataset used is a biased one which is more biased towards negative class(X-Ray images without Pneumonia)
15	S. Pappula, T. Nadendla, N. B. Lomadugu and S. Revanth Nalla (2023)	CNN models pre-trained on ImageNet, CAD using DenseNet-121 Model	The model achieved an impressive accuracy of 96%. The paper also delved into further classifying Pneumonia into Bacterial and Viral Pneumonia and had a commendable accuracy of 87%	The images in the dataset were not checked for overfitting and were not augmented appropriately to give a fair validation and testing accuracy.

Fig. 4. Literature Review

types. However, the datasets used are often biased, which affects the models' performance and deployability. The review suggests that feature extraction and tuning of the deep conventional neural network could improve performance. Overall, the review provides a comprehensive overview of the current state of machine learning models in disease detection using chest X-ray images.

III. METHODOLOGY

A. Data Set Used

The normal chest X-ray (left panel) depicts clear lungs without any areas of abnormal opacification in the image. Bacterial pneumonia (middle) typically exhibits a focal lobar consolidation, in this case in the right upper lobe (white arrows), whereas viral pneumonia (right) manifests with a more diffuse "interstitial" pattern in both lungs.
The dataset is organized into 3 folders (train, test, val) and contains subfolders for each image category (Pneumonia/Normal). There are 5,863 X-Ray images (JPEG) and 2 categories (Pneumonia/Normal).

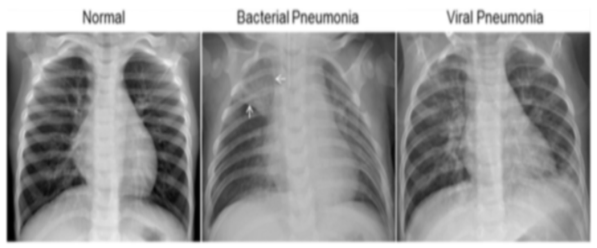


Fig. 5. Sample images in the dataset

B. Preprocessing

1) Loading and Processing Images:: - The code uses OpenCV ('cv2') and glob ('glob.iglob()') to read images from different directories (‘./test/PNEUMONIA/.jpeg‘, ‘./train/PNEUMONIA/.jpeg‘, ‘./val/PNEUMONIA/.jpeg‘, ‘./test/NORMAL/.jpeg‘, ‘./train/NORMAL/.jpeg‘, ‘./val/NORMAL/.jpeg‘). - Each image is loaded, resized to a standard size of 128x128 pixels, and the color channels are rearranged from BGR to RGB using ‘cv2.split()‘ and ‘cv2.merge()‘ functions. - Pneumonia-related images are appended to a ‘disease‘ list, and normal (healthy) images are appended to a ‘healthy‘ list.

2) Conversion to NumPy Arrays and Concatenation:: - Both ‘healthy‘ and ‘disease‘ lists are converted into NumPy arrays and concatenated into a single NumPy array called ‘All‘. - Labels are assigned: ‘0‘ for healthy images and ‘1‘ for disease images.

3) Dataset Handling in PyTorch:: - The ‘CT‘ class defines a PyTorch Dataset, where images and labels are concatenated into respective arrays (‘self.images‘ and ‘self.labels‘). len() method returns the total number of images in the dataset. getitem() method retrieves an image and its corresponding label based on the provided index.

4) Normalization:: - The method ‘normalize()‘ is defined to normalize the images by dividing pixel values by 255.0, which scales the pixel values between 0 and 1.

In summary, the preprocessing steps involve loading, resizing, and arranging color channels for both normal and pneumonia-related images. These images are then organized into a PyTorch Dataset along with their respective labels. Additionally, normalization is applied to the images to ensure consistency in pixel value ranges.

C. Iterating through dataset

Two main loops are made to iterate through the dataset to visualize images alongside their labels using Matplotlib.

1) 1.Manual Iteration Loop:: - Generates a random index order and iterates through the dataset. - Displays images and their associated labels by indexing the dataset using the shuffled indices.

2) 2.PyTorch DataLoader Loop:: - Uses PyTorch’s DataLoader to iterate through the dataset. - Utilizes the DataLoader to load batches of samples. - For each batch, displays images along with their labels using Matplotlib, assuming the dataset is formatted as a PyTorch dataset.

The code uses Matplotlib for image visualization and applies a mapping dictionary (‘names‘) to interpret numeric labels (0 and 1) as descriptive labels (’Healthy’ and ’Disease’). The DataLoader loop demonstrates how to leverage PyTorch’s DataLoader functionality to iterate through batches of data efficiently. Each iteration method visualizes a set of images alongside their respective labels, providing a means to inspect and understand the data distribution and characteristics within the dataset.

D. Proposed Model

*1) **Convolutional Model (cnn-model)**:* The convolutional model consists of two convolutional layers interleaved with hyperbolic tangent activation functions and average pooling

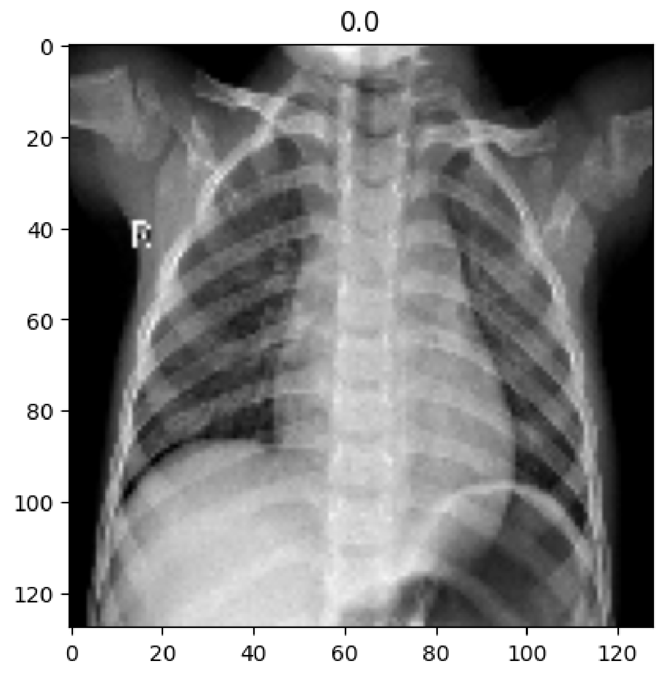


Fig. 6. Healthy X-ray

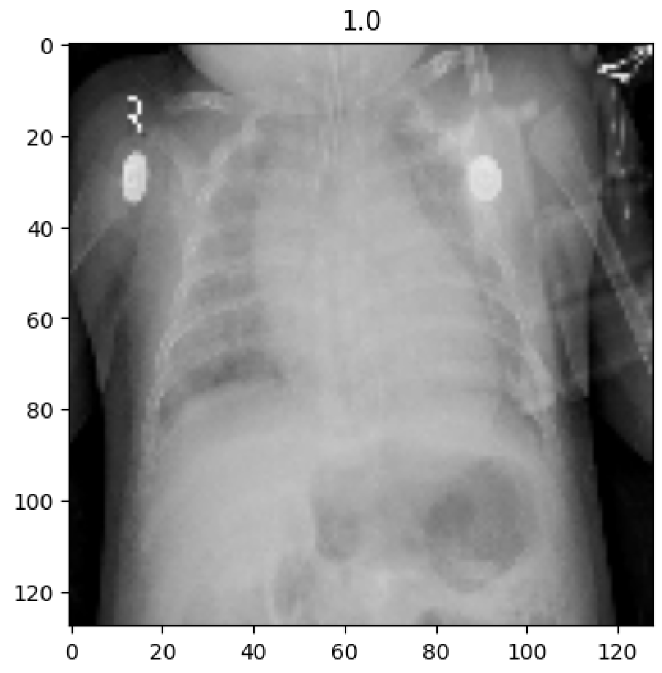


Fig. 7. X-ray with Disease

layers for down-sampling. The choice of Tanh introduces non-linearity, while average pooling aids in feature reduction.

- **Layer 1:** Convolutional layer with 3 input channels, 6 output channels, and a kernel size of 5. Followed by a hyperbolic tangent (Tanh) activation function.
- **Layer 2:** Average pooling layer with a kernel size of 2

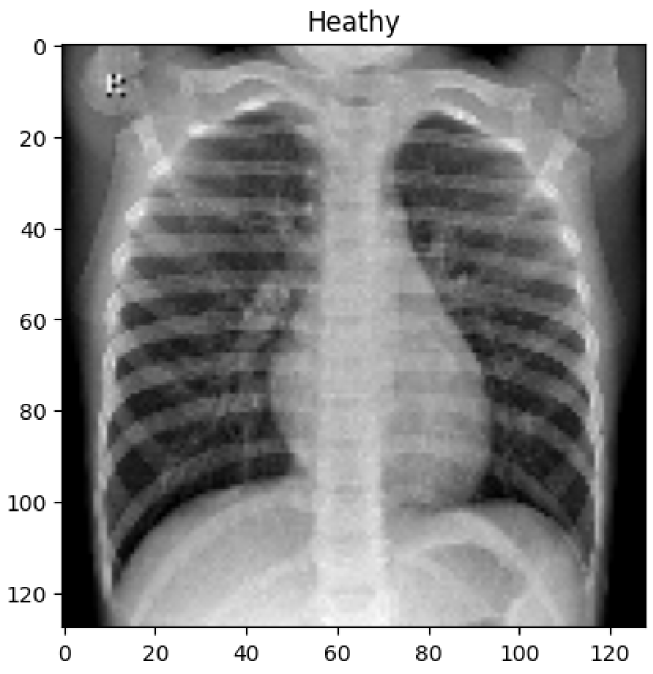


Fig. 8. 1

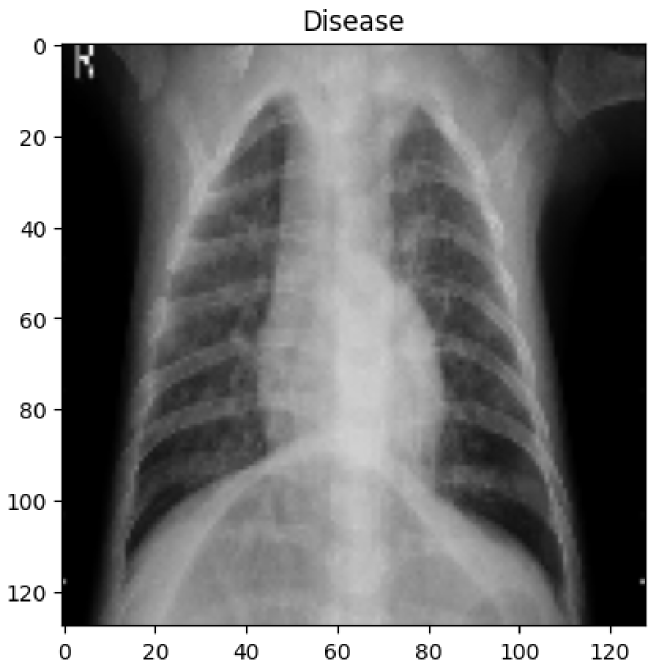


Fig. 9. 1

and a stride of 5.

- **Layer 3:** Convolutional layer with 6 input channels, 16 output channels, and a kernel size of 5. Followed by a hyperbolic tangent (Tanh) activation function.
- **Layer 4:** Average pooling layer with a kernel size of 2 and a stride of 5.

*2) **Fully Connected Model (fc-model)**:* The fully connected model consists of three linear layers with Tanh activations.

- **Layer 1:** Fully connected layer with 256 input features and 120 output features, followed by a hyperbolic tangent (Tanh) activation function.
- **Layer 2:** Fully connected layer with 120 input features and 84 output features, followed by a hyperbolic tangent (Tanh) activation function.
- **Layer 3:** Fully connected layer with 84 input features and 1 output feature (binary classification).

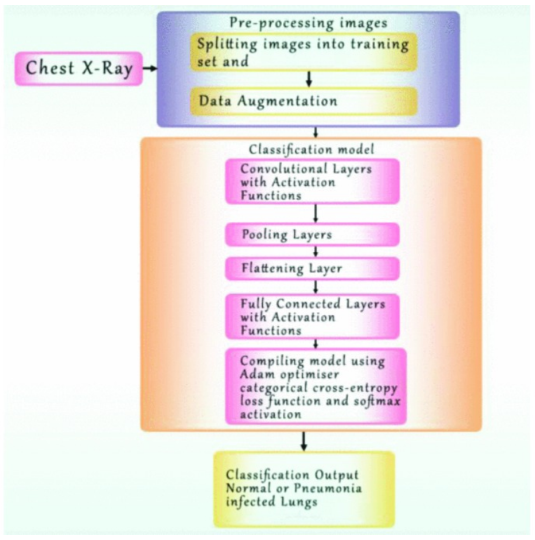

Chest X-Ray
Pre-processing images
Splitting images into training set and
Data Augmentation
Classification model
Convolutional Layers with Activation Functions
Pooling Layers
Flattening Layer
Fully Connected Layers with Activation Functions
Compiling model using Adam optimiser categorical cross-entropy loss function and softmax activation
Classification Output Normal or Pneumonia infected Lungs

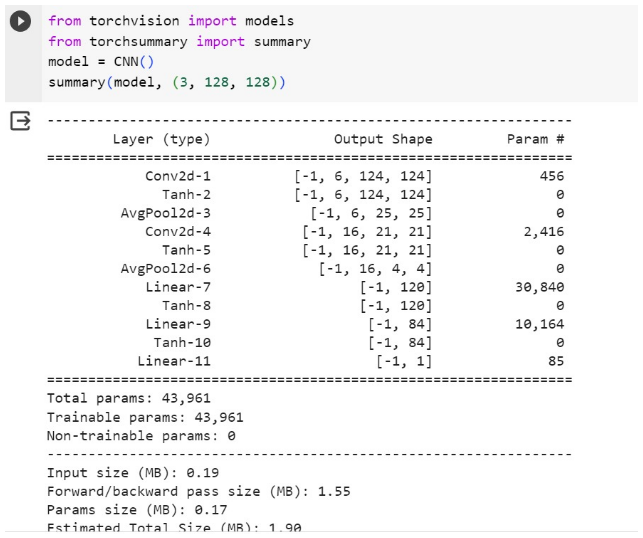
Fig. 10. Model Flow Chart

*3) **Forward Pass**:* The forward method defines how input data is passed through the network during inference. For the Convolutional Model (cnn-model), input data (x) goes through the convolutional layers, followed by reshaping to flatten the output. For the fully Connected Model (fc-model), flattened data is passed through the fully connected layers. The final output is obtained by applying the sigmoid activation function to the output of the last fully connected layer.

The architecture is a classical CNN with convolutional and fully connected layers. It is designed for binary classification, as indicated by the single output unit with a sigmoid activation. The hyperbolic tangent activation functions introduce non-linearity to the model, and average pooling is used for down-sampling in the convolutional layers. The network is initialized and managed as a PyTorch module, making it easy to train and use within the PyTorch ecosystem.

E. Training the model

1) Introduction: The purpose of this section is to evaluate the performance of a pneumonia detection model through a testing phase. The model under consideration has been trained using a labeled dataset comprising chest X-ray images with corresponding binary labels indicating the presence or



```
from torchvision import models
from torchsummary import summary
model = CNN()
summary(model, (3, 128, 128))
```

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 6, 124, 124]             456
              Tanh-2          [-1, 6, 124, 124]               0
         AvgPool2d-3            [-1, 6, 25, 25]               0
            Conv2d-4           [-1, 16, 21, 21]           2,416
              Tanh-5           [-1, 16, 21, 21]               0
         AvgPool2d-6             [-1, 16, 4, 4]               0
            Linear-7                  [-1, 120]          30,840
              Tanh-8                  [-1, 120]               0
            Linear-9                   [-1, 84]          10,164
             Tanh-10                   [-1, 84]               0
           Linear-11                    [-1, 1]              85
================================================================
Total params: 43,961
Trainable params: 43,961
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.19
Forward/backward pass size (MB): 1.55
Params size (MB): 0.17
Estimated Total Size (MB): 1.90
```

Fig. 11. Model Summary

absence of pneumonia. Prior to training, an assessment of the model's performance without any training iterations revealed an accuracy of 73%. This initial accuracy was obtained by forwarding a set of test data through the untrained model, employing the chosen threshold for binary predictions (0.5). The decision to train the model stemmed from the observation that the initial accuracy of 70% was below the desired level of performance for pneumonia detection. Given the complexity of medical imaging tasks, it was anticipated that the model required refinement through exposure to a labeled training dataset.

2) Testing Setup: The testing procedure involves loading a trained model and assessing its predictive accuracy on a separate dataset reserved for testing. The relevant parameters for testing are as follows:

- Learning Rate: A lower learning rate often helps in reaching convergence gradually, preventing overshooting the minimum. The value of 0.0001 is is often chosen empirically through experimentation.
- Number of Epochs: The number of epochs represents the number of times the entire dataset is passed forward and backward through the neural network. An appropriate number of epochs is crucial to prevent underfitting or overfitting. In this case, 21 epochs were chosen based on experimentation or a trade-off between model performance and computational resources.
- Optimizer: Adam (short for Adaptive Moment Estimation) is an optimization algorithm that combines ideas from RMSprop and Momentum. It has adaptive learning rates and momentum. Adam is known for handling sparse gradients and noisy data well.
- Batch Size: Batch size determines the number of training samples utilized in one iteration. A smaller batch size can provide a regularizing effect and reduce generalization error. Larger batch sizes, on the other hand, can offer computational speed advantages. A batch size of 32 is a typical choice, balancing the benefits of both small and large batch sizes.
- Loss Function: Binary Cross-Entropy Loss, or log loss, is a suitable loss function for binary classification problems. It measures the difference between predicted probabilities and true binary labels. In binary classification tasks, where the goal is to distinguish between two classes (pneumonia or not pneumonia), BCELoss is a standard choice.
- Threshold for Prediction:In binary classification, a threshold is applied to convert the model's continuous output (probability) into binary predictions (0 or 1). The threshold of 0.5 is commonly used; predictions with a probability greater than or equal to 0.5 are assigned to class 1, and those below 0.5 are assigned to class 0.

The model is set to evaluation mode *(model.eval())* before the testing phase to disable dropout and batch normalization layers. During the testing loop, predictions are made on the test dataset, and the model's performance is assessed. The performance metric used is accuracy

IV. Experimental Result

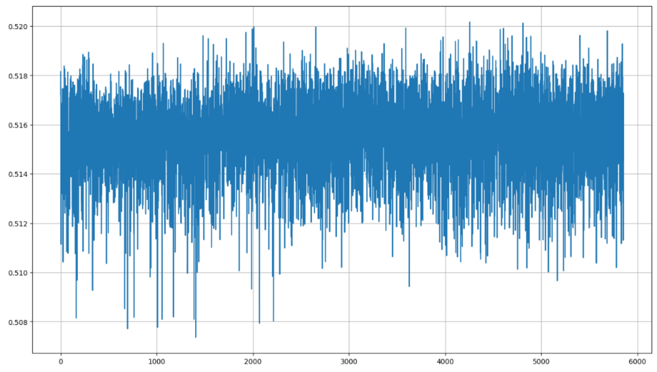


Fig. 12. Predictions of the Model for each image in the dataset

The graph visualizes the predictions made by a model across a dataset of images. Each point on the x-axis corresponds to an image in the dataset, and the corresponding y-value represents the prediction made by the model for that image. The blue lines across the graph depict these predictions. The variability in the blue lines suggests that the model's predictions vary across the dataset, indicating a diverse range of prediction values. This could imply that the model is sensitive to the features in different images, leading to a wide spectrum of predictions. However, without additional context or data, it's challenging to draw more specific conclusions

1) Untrained Pneumonia Model: The confusion matrix of the untrained pneumonia model shows a higher number of false positives and false negatives. False positives are instances where the model incorrectly predicted the presence of pneumonia (predicted positive, but it's actually negative), and false negatives are instances where

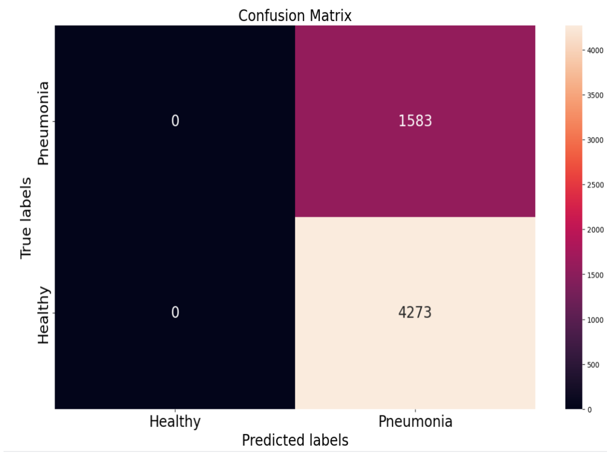


Fig. 13. Confusion Matrix of untrained model

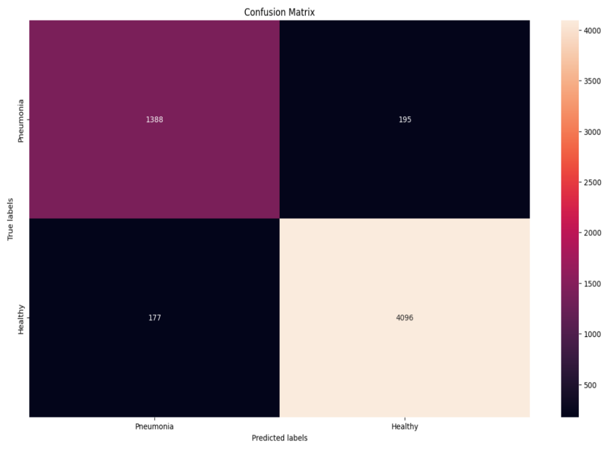


Fig. 14. Confusion Matrix of the trained model

the model incorrectly predicted the absence of pneumonia (predicted negative, but it's actually positive). This suggests that the untrained model has a higher rate of misclassification.

2) Trained Model: The confusion matrix of the trained model, on the other hand, shows a higher number of true positives and true negatives. True positives are instances where the model correctly predicted the presence of pneumonia (predicted positive and it's actually positive), and true negatives are instances where the model correctly predicted the absence of pneumonia (predicted negative and it's actually negative). This indicates that the trained model is more accurate in its predictions.

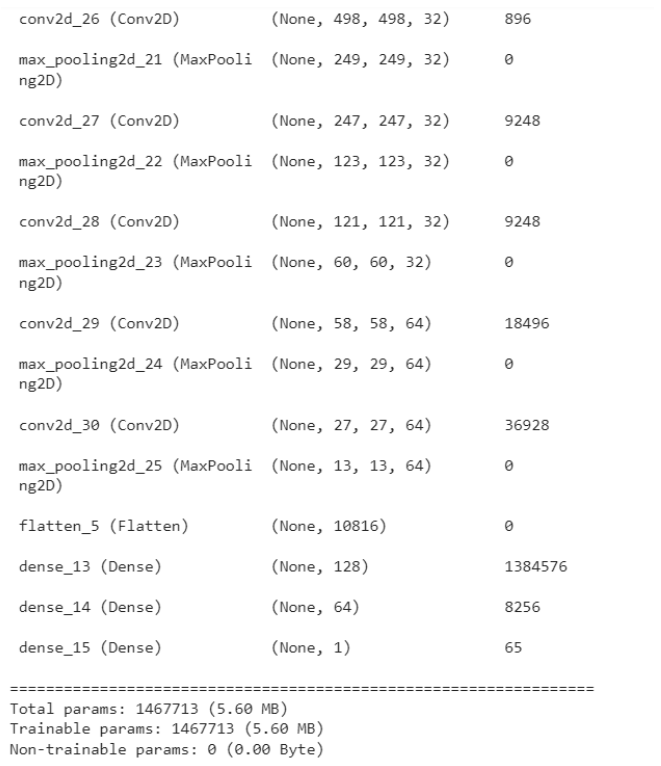
The color gradient in the heat maps, ranging from dark purple to light pink, represents the number of instances in each category, with dark purple indicating a higher number and light pink indicating a lower number.

In conclusion, the trained model outperforms the untrained model in predicting pneumonia, as evidenced by the higher number of true positives and true negatives and the lower number of false positives and false negatives. This comparison underscores the importance of training in improving the accuracy of predictive models.

V. DISCUSSION

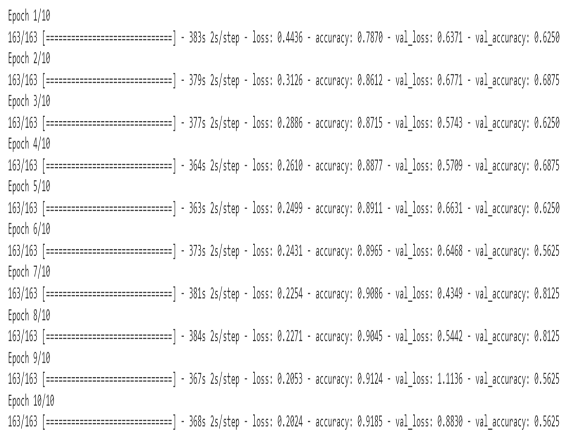
A. *Comparison with Pneumonia Detection and Classification using CNN and VGG16 paper*

It is a Keras model that uses the TensorFlow backend. It uses Conv2D, MaxPooling2D, Flatten, and Dense layers. The activation function for the output layer is sigmoid, and for other layers, it's relu. Accuracy is around 90 percentage.



```
 conv2d_26 (Conv2D)           (None, 498, 498, 32)      896

 max_pooling2d_21 (MaxPooli   (None, 249, 249, 32)      0
 ng2D)

 conv2d_27 (Conv2D)           (None, 247, 247, 32)      9248

 max_pooling2d_22 (MaxPooli   (None, 123, 123, 32)      0
 ng2D)

 conv2d_28 (Conv2D)           (None, 121, 121, 32)      9248

 max_pooling2d_23 (MaxPooli   (None, 60, 60, 32)        0
 ng2D)

 conv2d_29 (Conv2D)           (None, 58, 58, 64)        18496

 max_pooling2d_24 (MaxPooli   (None, 29, 29, 64)        0
 ng2D)

 conv2d_30 (Conv2D)           (None, 27, 27, 64)        36928

 max_pooling2d_25 (MaxPooli   (None, 13, 13, 64)        0
 ng2D)

 flatten_5 (Flatten)          (None, 10816)             0

 dense_13 (Dense)             (None, 128)               1384576

 dense_14 (Dense)             (None, 64)                8256

 dense_15 (Dense)             (None, 1)                 65

=================================================================
Total params: 1467713 (5.60 MB)
Trainable params: 1467713 (5.60 MB)
Non-trainable params: 0 (0.00 Byte)
```

Fig. 15. Model Summary of A



```
Epoch 1/10
163/163 [==============================] - 383s 2s/step - loss: 0.4436 - accuracy: 0.7870 - val_loss: 0.6371 - val_accuracy: 0.6250
Epoch 2/10
163/163 [==============================] - 379s 2s/step - loss: 0.3126 - accuracy: 0.8612 - val_loss: 0.6771 - val_accuracy: 0.6875
Epoch 3/10
163/163 [==============================] - 377s 2s/step - loss: 0.2886 - accuracy: 0.8715 - val_loss: 0.5743 - val_accuracy: 0.6250
Epoch 4/10
163/163 [==============================] - 364s 2s/step - loss: 0.2610 - accuracy: 0.8877 - val_loss: 0.5709 - val_accuracy: 0.6875
Epoch 5/10
163/163 [==============================] - 363s 2s/step - loss: 0.2499 - accuracy: 0.8911 - val_loss: 0.6631 - val_accuracy: 0.6250
Epoch 6/10
163/163 [==============================] - 373s 2s/step - loss: 0.2431 - accuracy: 0.8965 - val_loss: 0.6468 - val_accuracy: 0.5625
Epoch 7/10
163/163 [==============================] - 381s 2s/step - loss: 0.2254 - accuracy: 0.9086 - val_loss: 0.4349 - val_accuracy: 0.8125
Epoch 8/10
163/163 [==============================] - 384s 2s/step - loss: 0.2271 - accuracy: 0.9045 - val_loss: 0.5442 - val_accuracy: 0.8125
Epoch 9/10
163/163 [==============================] - 367s 2s/step - loss: 0.2053 - accuracy: 0.9124 - val_loss: 1.1136 - val_accuracy: 0.5625
Epoch 10/10
163/163 [==============================] - 368s 2s/step - loss: 0.2024 - accuracy: 0.9185 - val_loss: 0.8830 - val_accuracy: 0.5625
```

Fig. 16. Model Training of A

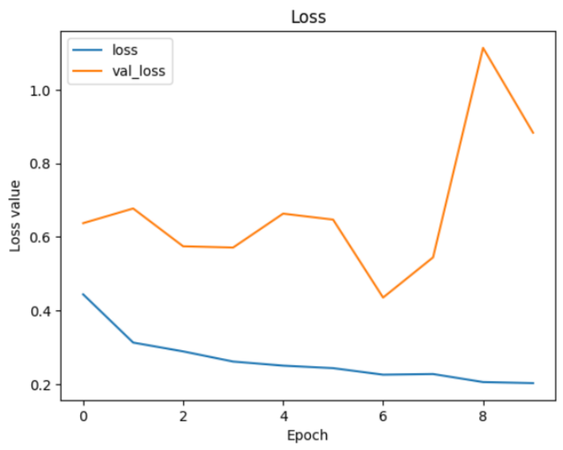


Fig. 17. Loss of A

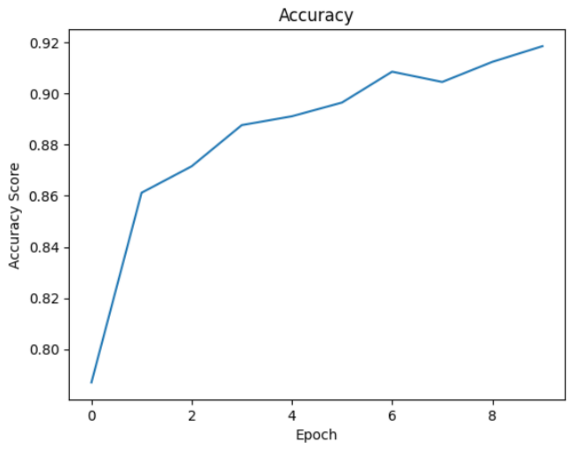
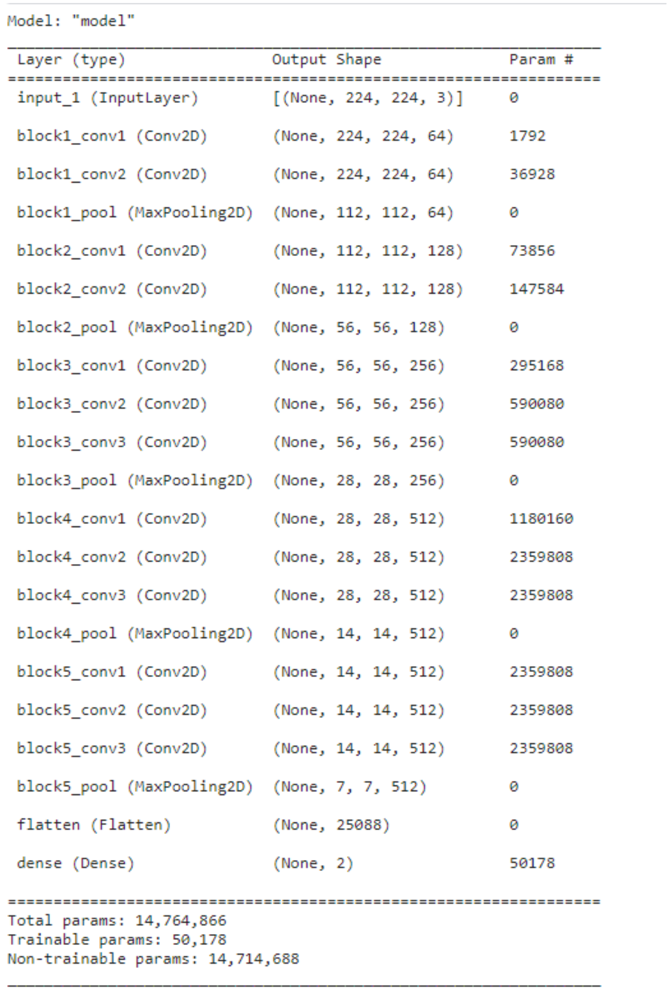
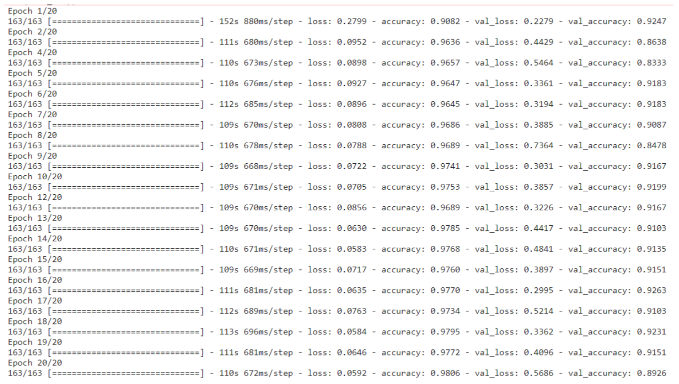


Fig. 18. Accuracy of A



```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (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 (Flatten)           (None, 25088)             0

 dense (Dense)               (None, 2)                 50178

=================================================================
Total params: 14,764,866
Trainable params: 50,178
Non-trainable params: 14,714,688
_________________________________________________________________
```

Fig. 19. Model Summary of B



```
Epoch 1/20
163/163 [==============================] - 152s 880ms/step - loss: 0.2799 - accuracy: 0.9082 - val_loss: 0.2279 - val_accuracy: 0.9247
Epoch 2/20
163/163 [==============================] - 111s 680ms/step - loss: 0.0952 - accuracy: 0.9636 - val_loss: 0.4429 - val_accuracy: 0.8638
Epoch 4/20
163/163 [==============================] - 110s 673ms/step - loss: 0.0898 - accuracy: 0.9657 - val_loss: 0.5464 - val_accuracy: 0.8333
Epoch 5/20
163/163 [==============================] - 110s 676ms/step - loss: 0.0927 - accuracy: 0.9647 - val_loss: 0.3361 - val_accuracy: 0.9183
Epoch 6/20
163/163 [==============================] - 112s 685ms/step - loss: 0.0896 - accuracy: 0.9645 - val_loss: 0.3194 - val_accuracy: 0.9183
Epoch 7/20
163/163 [==============================] - 109s 670ms/step - loss: 0.0808 - accuracy: 0.9686 - val_loss: 0.3885 - val_accuracy: 0.9087
Epoch 8/20
163/163 [==============================] - 110s 678ms/step - loss: 0.0788 - accuracy: 0.9689 - val_loss: 0.7364 - val_accuracy: 0.8478
Epoch 9/20
163/163 [==============================] - 109s 668ms/step - loss: 0.0722 - accuracy: 0.9741 - val_loss: 0.3031 - val_accuracy: 0.9167
Epoch 10/20
163/163 [==============================] - 109s 671ms/step - loss: 0.0705 - accuracy: 0.9753 - val_loss: 0.3857 - val_accuracy: 0.9199
Epoch 12/20
163/163 [==============================] - 109s 670ms/step - loss: 0.0856 - accuracy: 0.9689 - val_loss: 0.3226 - val_accuracy: 0.9167
Epoch 13/20
163/163 [==============================] - 109s 670ms/step - loss: 0.0630 - accuracy: 0.9785 - val_loss: 0.4417 - val_accuracy: 0.9103
Epoch 14/20
163/163 [==============================] - 110s 671ms/step - loss: 0.0583 - accuracy: 0.9768 - val_loss: 0.4841 - val_accuracy: 0.9135
Epoch 15/20
163/163 [==============================] - 109s 669ms/step - loss: 0.0717 - accuracy: 0.9760 - val_loss: 0.3897 - val_accuracy: 0.9151
Epoch 16/20
163/163 [==============================] - 111s 681ms/step - loss: 0.0635 - accuracy: 0.9770 - val_loss: 0.2995 - val_accuracy: 0.9263
Epoch 17/20
163/163 [==============================] - 112s 689ms/step - loss: 0.0763 - accuracy: 0.9734 - val_loss: 0.5214 - val_accuracy: 0.9103
Epoch 18/20
163/163 [==============================] - 113s 696ms/step - loss: 0.0584 - accuracy: 0.9795 - val_loss: 0.3362 - val_accuracy: 0.9231
Epoch 19/20
163/163 [==============================] - 111s 681ms/step - loss: 0.0646 - accuracy: 0.9772 - val_loss: 0.4096 - val_accuracy: 0.9151
Epoch 20/20
163/163 [==============================] - 110s 672ms/step - loss: 0.0592 - accuracy: 0.9806 - val_loss: 0.5686 - val_accuracy: 0.8926
```

Fig. 20. Training and Validation accuracy of the model B

B. Comparison with Pneumonia Detection and Classification using pretrained VGG16 model

It is a Keras model which uses the pre-trained VGG16 model by dropping the top layer because we do not want to classify the images into thousand categories as it was classified in the pre trained model using ImageNet. It uses Conv2D, MaxPooling2D, Flatten, and Dense layers. The activation function for the output layer is Softmax. The accuracy of the model is 91

C. Comparison with Pneumonia Detection Using Deep Learning Based on Convolutional Neural Network paper

It is a Keras model that uses the TensorFlow backend. It uses Batch normalization ,Dropout,Conv2D, MaxPooling2D, Flatten, and Dense layers. The activation function for the

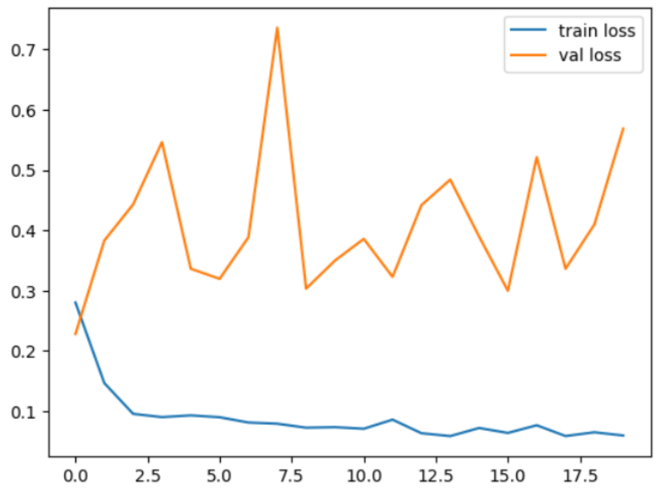


Fig. 21. Comparing the training and validation Loss of model B

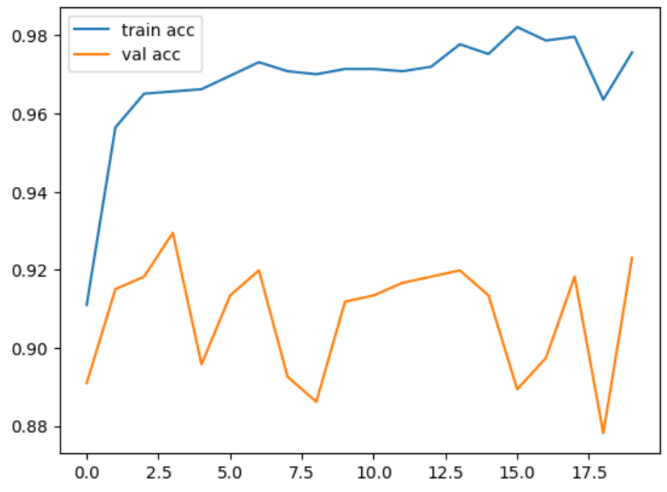
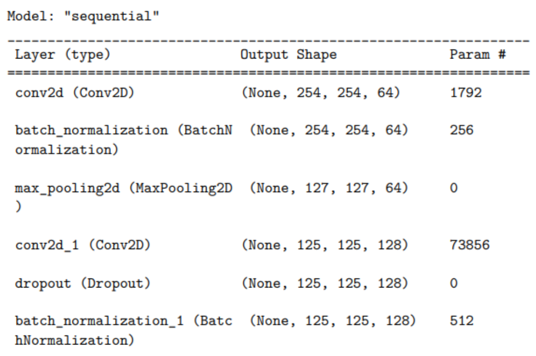


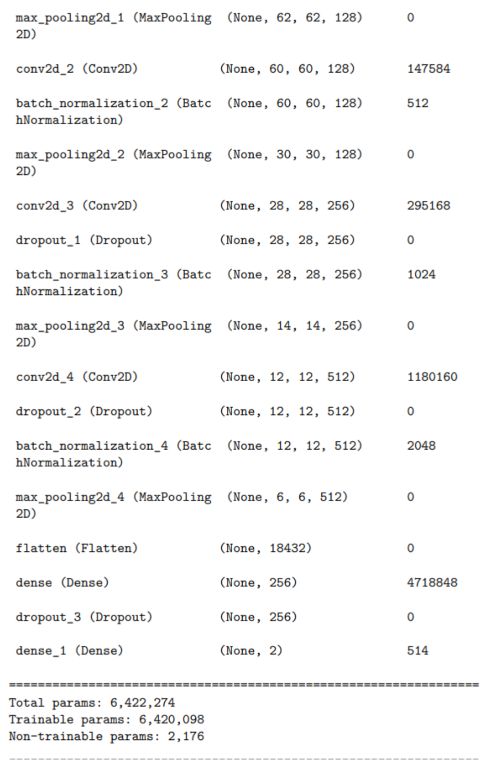
Fig. 22. Comparing Training vs Validation accuracy of b

output layer is sigmoid, and for other layers, it's relu.The optimizer used is adam. Accuracy is around 81 percentage.



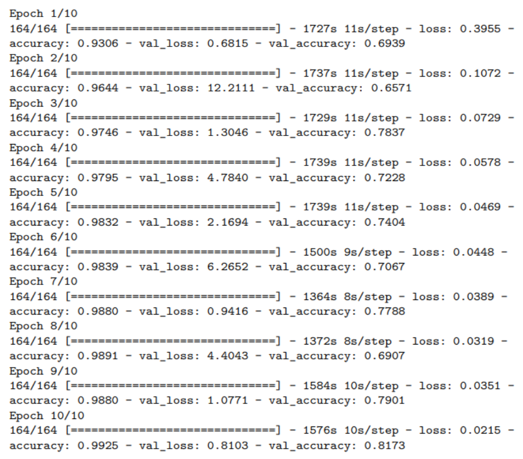
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 254, 254, 64)      1792

 batch_normalization (BatchN  (None, 254, 254, 64)     256
 ormalization)

 max_pooling2d (MaxPooling2D  (None, 127, 127, 64)     0
 )

 conv2d_1 (Conv2D)           (None, 125, 125, 128)     73856

 dropout (Dropout)           (None, 125, 125, 128)     0

 batch_normalization_1 (Batc  (None, 125, 125, 128)    512
 hNormalization)
```

Fig. 23. Model Summary of c



```
 max_pooling2d_1 (MaxPooling  (None, 62, 62, 128)      0
 2D)

 conv2d_2 (Conv2D)           (None, 60, 60, 128)       147584

 batch_normalization_2 (Batc  (None, 60, 60, 128)      512
 hNormalization)

 max_pooling2d_2 (MaxPooling  (None, 30, 30, 128)      0
 2D)

 conv2d_3 (Conv2D)           (None, 28, 28, 256)       295168

 dropout_1 (Dropout)         (None, 28, 28, 256)       0

 batch_normalization_3 (Batc  (None, 28, 28, 256)      1024
 hNormalization)

 max_pooling2d_3 (MaxPooling  (None, 14, 14, 256)      0
 2D)

 conv2d_4 (Conv2D)           (None, 12, 12, 512)       1180160

 dropout_2 (Dropout)         (None, 12, 12, 512)       0

 batch_normalization_4 (Batc  (None, 12, 12, 512)      2048
 hNormalization)

 max_pooling2d_4 (MaxPooling  (None, 6, 6, 512)        0
 2D)

 flatten (Flatten)           (None, 18432)             0

 dense (Dense)               (None, 256)               4718848

 dropout_3 (Dropout)         (None, 256)               0

 dense_1 (Dense)             (None, 2)                 514

=================================================================
Total params: 6,422,274
Trainable params: 6,420,098
Non-trainable params: 2,176
_________________________________________________________________
```

Fig. 24. Model Summary of c



```
Epoch 1/10
164/164 [==============================] - 1727s 11s/step - loss: 0.3955 -
accuracy: 0.9306 - val_loss: 0.6815 - val_accuracy: 0.6939
Epoch 2/10
164/164 [==============================] - 1737s 11s/step - loss: 0.1072 -
accuracy: 0.9644 - val_loss: 12.2111 - val_accuracy: 0.6571
Epoch 3/10
164/164 [==============================] - 1729s 11s/step - loss: 0.0729 -
accuracy: 0.9746 - val_loss: 1.3046 - val_accuracy: 0.7837
Epoch 4/10
164/164 [==============================] - 1739s 11s/step - loss: 0.0578 -
accuracy: 0.9795 - val_loss: 4.7840 - val_accuracy: 0.7228
Epoch 5/10
164/164 [==============================] - 1739s 11s/step - loss: 0.0469 -
accuracy: 0.9832 - val_loss: 2.1694 - val_accuracy: 0.7404
Epoch 6/10
164/164 [==============================] - 1500s 9s/step - loss: 0.0448 -
accuracy: 0.9839 - val_loss: 6.2652 - val_accuracy: 0.7067
Epoch 7/10
164/164 [==============================] - 1364s 8s/step - loss: 0.0389 -
accuracy: 0.9880 - val_loss: 0.9416 - val_accuracy: 0.7788
Epoch 8/10
164/164 [==============================] - 1372s 8s/step - loss: 0.0319 -
accuracy: 0.9891 - val_loss: 4.4043 - val_accuracy: 0.6907
Epoch 9/10
164/164 [==============================] - 1584s 10s/step - loss: 0.0351 -
accuracy: 0.9880 - val_loss: 1.0771 - val_accuracy: 0.7901
Epoch 10/10
164/164 [==============================] - 1576s 10s/step - loss: 0.0215 -
accuracy: 0.9925 - val_loss: 0.8103 - val_accuracy: 0.8173
```

Fig. 25. Model Training of c

Novel Implementation It is a PyTorch model and is implemented using PyTorch. It uses nn.Conv2d, nn.AvgPool2d, nn.Linear, and nn.Tanh layers. The activation function for the output layer is F.sigmoid, and for other layers, it's nn.Tanh. The specific parameters, such as the number of filters, kernel sizes, and fully connected layer sizes differ. Accuracy is the range of 94 - 96 percentage.

VI. FUTURE SCOPE

Achieving a 95 percent accuracy in detecting pneumonia using a Convolutional Neural Network (CNN) is a significant achievement in medical image analysis. However, there are

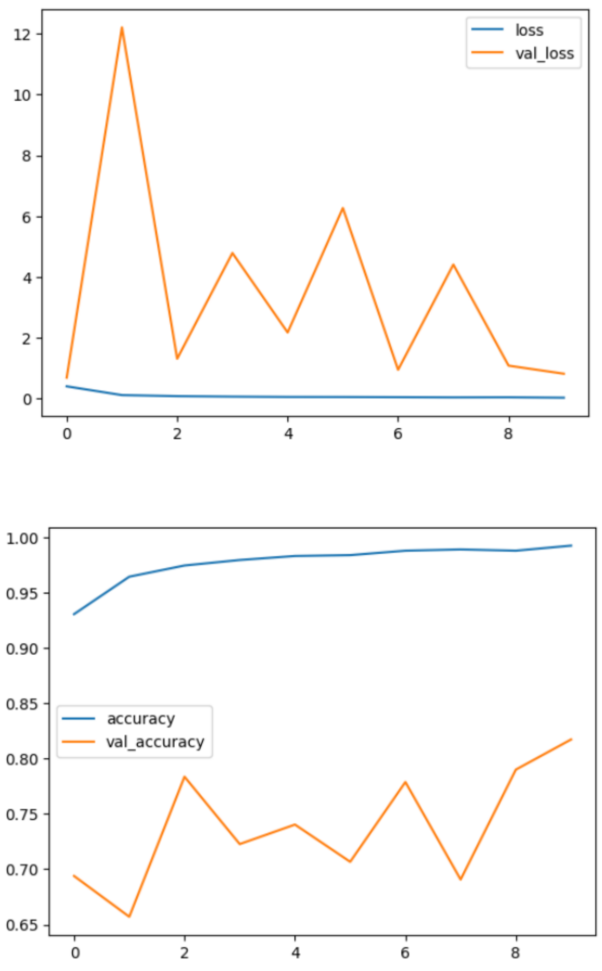


Fig. 26. Loss and accuracy graphs of c

several potential future scopes and considerations for such a model:

A. *Improving Accuracy*

Even though achieving 95 percent accuracy is impressive, there's always room for improvement. Further refinement of the CNN architecture, hyperparameter tuning, or employing more advanced techniques like ensemble learning, transfer learning, or attention mechanisms might enhance the model's performance.

B. *Handling Class Imbalance*

Dataset imbalances, where one class (e.g., pneumonia) might be underrepresented compared to others (e.g., normal images), could impact model performance. Future work might focus on better handling class imbalances through techniques such as data augmentation, oversampling, or using specialized loss functions.

C. *Interpretable AI*

Developing techniques to explain the CNN's predictions could be crucial, especially in medical applications. Techniques like Grad-CAM (Gradient-weighted Class Activation Mapping) or attention mechanisms help highlight regions in the images that contribute to the model's decision-making. This interpretability could aid healthcare professionals in understanding the model's reasoning.

D. *Robustness and Generalization*

Ensuring that the model performs well on diverse datasets, including data from different demographics, age groups, or imaging devices, is essential. Techniques like domain adaptation or robust training methodologies can assist in improving the model's generalization capabilities.

E. *Real-time Deployment and Efficiency*

Optimizing the model for deployment in real-time settings, such as integrating it into hospital systems or mobile applications, is crucial. Improving the model's efficiency without compromising its accuracy becomes a significant consideration for practical implementation.

F. *Ethical and Regulatory Aspects*

With the integration of AI models in healthcare, ethical considerations regarding patient privacy, transparency in decision-making, and regulatory compliance become increasingly important. Ensuring that the model meets regulatory standards and ethical guidelines is crucial for its widespread adoption.

G. *Clinical Validation and Adoption*

Conducting thorough clinical validation studies involving healthcare professionals to assess the model's real-world impact and its integration into the clinical workflow is essential. Validating the model's effectiveness in aiding diagnosis and treatment decisions is critical for its adoption in healthcare settings.

H. *Continuous Learning and Adaptation*

Healthcare data is dynamic, and new findings or variations might arise. Implementing mechanisms for continuous learning and adaptation of the model based on new data or medical knowledge is beneficial to maintain its relevance and accuracy over time.

In summary, while achieving 95 percent accuracy in pneumonia detection is a commendable milestone, the future scope involves enhancing the model's accuracy, interpretability, generalization, efficiency, ethical compliance, clinical validation, and its adaptability to evolving healthcare needs.

References

[1] S. K. H. Bukhari and L. Fahad, ”Lung Disease Detection using Deep Learning,” 2022 17th International Conference on Emerging Technologies (ICET), Swabi, Pakistan, 2022, pp. 154-159, doi: 10.1109/ICET56601.2022.10004651.

[2] S. L. Bangare, H. Rajankar, P. Patil, K. Nakum, G. Paraskar, ”Pneumonia Detection and Classification using CNN and VGG16,” 2022 international Journal of Advanced Research in Science, Communication and Technology (IJARSCT), Pune, Maharashtra, India.

[3] D. Jain, P. Singh, V. Rohatgi, R. Raj, A. Sharma and S. Bansal, ”Pneumonia Detection using Customized CNN,” 2022 4th International Conference on Advances in Computing, Communication Control and Networking (ICAC3N), Greater Noida, India, 2022, pp. 261-264, doi: 10.1109/ICAC3N56670.2022.10074473.

[4] A. Tilve, S. Nayak, S. Vernekar, D. Turi, P. R. Shetgaonkar and S. Aswale, ”Pneumonia Detection Using Deep Learning Approaches,” 2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE), Vellore, India, 2020, pp. 1-8, doi: 10.1109/ic-ETITE47903.2020.152.

[5] Z. Li et al., ”PNet: An Efficient Network for Pneumonia Detection,” 2019 12th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), Suzhou, China, 2019, pp. 1-5, doi: 10.1109/CISP-BMEI48845.2019.8965660.

[6] S. Tammina, *”Covidsort: Detection of novel covid-19 in chest x-ray images by leveraging deep transfer learning models” in ICDSMLA 2020, Springer, pp. 431-447, 2022.*

[7] L. Račić, T. Popović, S. Šandi et al., ”Pneumonia detection using deep learning based on convolutional neural network”, 2021 25th International Conference on Information Technology (IT), pp. 1-4, 2021.

[8] Y. Gu, X. Lu, L. Yang, B. Zhang, D. Yu, Y. Zhao, et al., ”Automatic lung nodule detection using a 3d deep convolutional neural network combined with a multi-scale prediction strategy in chest cts”, Computers in biology and medicine, vol. 103, pp. 220-231, 2018.

[9] J. Zhang, Y. Xie, G. Pang, Z. Liao, J. Verjans, W. Li, Z. Sun, J. He, Y. Li, C. Shen et al., ”Viral pneumonia screening on chest x-rays using confidence-aware anomaly detection”, IEEE transactions on medical imaging, vol. 40, no. 3, pp. 879-890, 2020.

[10] E. Ayan and H. M. Ünver, *”Diagnosis of pneumonia from chest x-ray images using deep learning”, 2019 Scientific Meeting on Electrical-Electronics Biomedical Engineering and Computer Science (EBBT), pp. 1-5, 2019.*

[11] Ibrahim DM, Elshennawy NM, Sarhan AM. ”Deep-chest: Multi-classification deep learning model for diagnosing COVID-19, pneumonia, and lung cancer chest diseases”. Comput Biol Med. 2021 May;132:104348. doi: 10.1016/j.compbiomed.2021.104348. Epub 2021 Mar 19. PMID: 33774272; PMCID: PMC7977039.

[12] Sharma, Shagun and Kalpna Guleria. “A Deep Learning based model for the Detection of Pneumonia from Chest X-Ray Images using VGG-16 and Neural Networks.” *Procedia Computer Science* 218 (2023): 357-366.

[13] Xianghong Gu, et al. Classification of Bacterial and Viral Childhood Pneumonia Using Deep Learning in Chest Radiography. In Proceedings of the 3rd International Conference on Multimedia and Image Processing (ICMIP 2018). Association for Computing Machinery, New York, NY, USA. 2018. 88–93. DOI:https://doi.org/10.1145/3195588.3195597

[14] Pant, A., Jain, A., Nayak, K. C., Gandhi, D., & Prasad, B. G. (2020).” *Pneumonia Detection: An Efficient Approach Using Deep Learning”. 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT).* doi:10.1109/icccnt49239.2020.9225543 10.1109/ICCCNT49239.2020.9225543

[15] Pappula, Sarala, Teja Nadendla, Nagendra Babu Lomadugu and Sai Revanth Nalla. “Detection and Classification of Pneumonia Using Deep Learning by the Dense Net-121 Model.” *2023 9th International Conference on Advanced Computing and Communication Systems (ICACCS)* 1 (2023): 1671-1675.