Lung Disease Detection

Group-19

Lakshmi Sai Hanuhitha Dondapati (ldondapa)

Sahithi Vasireddy (svasire)

Rishika Samala (rsamala)

INTRODUCTION

Problem Statement

Aim: To identify the the type of lung disease using chest X-ray images.

- Lung infections as these 5 classes, are hard to differentiate because of common symptoms.
- Early detection is necessary.

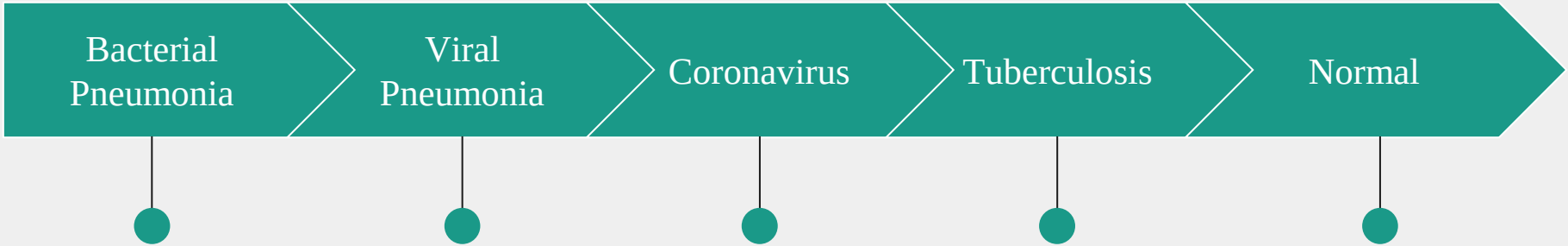
Our Approach: We are using Convolutional Neural Network for our disease image classification problem.

Research Questions

- Why CNN works better for this dataset for classification?
- Analyzing the performance of dataset with different CNN architectures.
- Analyzing the metrics,learning curves of dataset with different optimizers.
- How Transfer learning is affecting the classification metrics for this dataset?

DATA FROM KAGGLE

- Total data - 10,000 images
- Balanced dataset



PRIOR WORK

- Hybrid deep learning for detecting lung diseases from X-ray images.[3]
- Tuberculosis detection using chest X-ray images with Deep Learning [2]
- Detection and classification of lung disease for pneumonia and covid -19 using deep learning.[3]
- Unsupervised representation learning for lung disease classification using chest CT images- GANs [4]

[1] https://www.sciencedirect.com/science/article/pii/S2352914820300290
[2] https://ieeexplore.ieee.org/abstract/document/9224622
[3] https://link.springer.com/article/10.1007/s12652-021-03464-7#ref-CR24
[4] https://ieeexplore.ieee.org/abstract/document/9524697

PROPOSED APPROACH

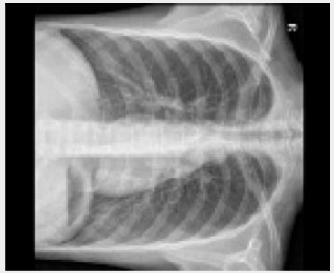
1. **Data Augmentation and Pre-Processing:** Normalizing the images, Implementing script for data loading.
2. **Baseline Modeling:** Implemented a Convolutional Neural Network (CNN) architecture for the classification task. Trained the model using keras tensorflow.
3. **Evaluation Metrics:** Defined and analyzed accuracy of test data using F1 score, Precision and Recall.
4. **Learning Curves:** Analyzed the learning curves obtained.
5. **Tuning Hyper parameters:** Tuned hyper parameters by experimenting with different CNN architectures, optimizers, learning rates and pre-processing techniques.
6. **Transfer Learning:** Used different transfer learning methods to check the performance.
7. **Results:** Compared the results and figured which model performs the best.

EXPERIMENTS AND EVALUATION METRICS

Data Augmentation and Pre-Processing :

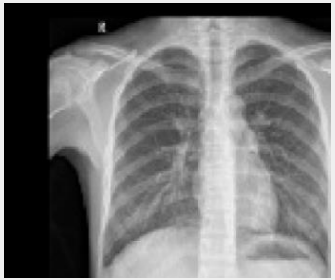
1. Rescaled the input images as they contain some RGB components.
2. We used augmentation techniques like horizontal flip, vertical flip, width shift range, height shift range, zoom range as pre processing techniques for the images to be not sensitive to flipping, shifting and zooming.
3. Resized the input images to make all of them uniform for further processing.

Original Image

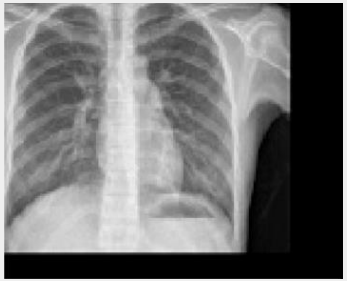


Horizontal Flip

Vertical Flip



Width Shift



Height Shift

EXPERIMENTS AND EVALUATION METRICS

CNN Baseline Model Architecture:

1. We started with a CNN architecture with 2 convolutional layers with kernel size 3*3, max-pooling layer after each convolutional layer, flatten the output of max-pooling layers, and then 1 fully connected hidden dense layer, one output layer. Used RELU activation and padding.
2. Also implemented batch normalization after each convolutional and fully connected layer for vanishing, exploding gradients problem. We implemented a dropout of 20% for each layer.
3. The optimizer used is ADAM and a loss function of cross-entropy is defined.
4. The fully connected output layer with 5 output layers is used to detect the 5 categorical labels with activation function as softmax.

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
conv2d (Conv2D)              (None, 224, 224, 32)      896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 112, 112, 32)      0
_________________________________________________________________
batch_normalization (BatchNo (None, 112, 112, 32)      128
_________________________________________________________________
dropout (Dropout)            (None, 112, 112, 32)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 112, 112, 32)      9248
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 56, 56, 32)        0
_________________________________________________________________
batch_normalization_1 (Batch (None, 56, 56, 32)        128
_________________________________________________________________
dropout_1 (Dropout)          (None, 56, 56, 32)        0
_________________________________________________________________
flatten (Flatten)            (None, 100352)            0
_________________________________________________________________
dense (Dense)                (None, 512)               51380736
_________________________________________________________________
batch_normalization_2 (Batch (None, 512)               2048
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 5)                 2565
=================================================================
Total params: 51,395,749
Trainable params: 51,394,597
Non-trainable params: 1,152
_________________________________________________________________
```

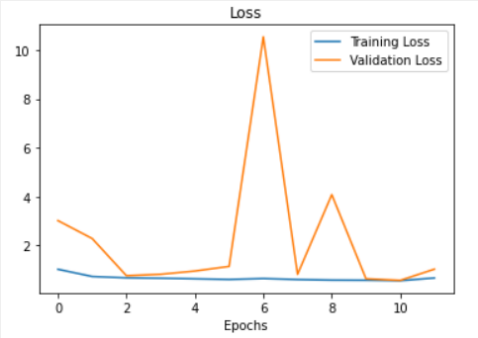
EXPERIMENTS AND EVALUATION METRICS

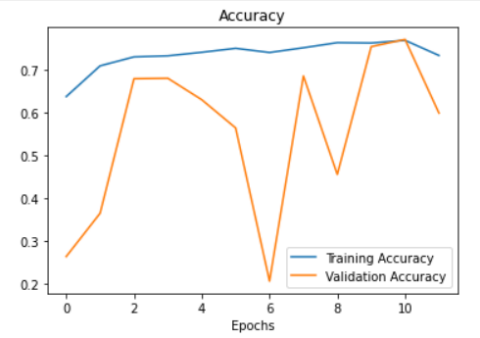
Evaluation Metrics of Baseline Model:

1. We observed the accuracy to be around 60% with precision, recall, F1 score as follows:

	precision	recall	f1-score	support
0	0.60	0.69	0.64	403
1	0.89	0.22	0.35	407
2	0.92	0.76	0.83	404
3	0.47	1.00	0.64	408
4	0.63	0.40	0.49	403
accuracy			0.62	2025
macro avg	0.70	0.62	0.59	2025
weighted avg	0.70	0.62	0.59	2025

2. We observed the learning curves as follows:





FURTHER EXPERIMENTS AND GENERATED RESULTS

Hyperparameter Tuning:

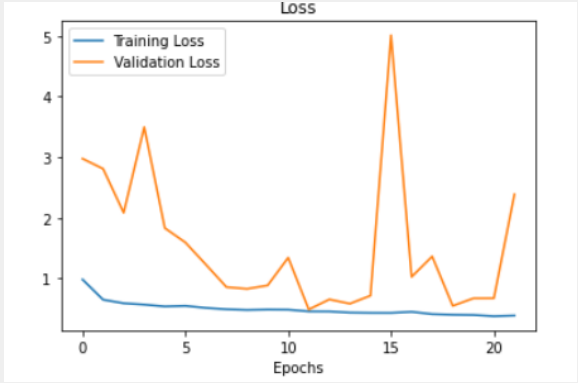
1. We experimented with **different layers of CNN** and settled with 4 layers and feature layers starting from 256, 128, 64 and 32 respectively.
2. We added **he_initializer** and **bias initializer** to resolve some of vanishing and exploding gradients problems.
3. We reduced the **drop out** probability to 10% from 20% as in baseline model.
4. Implemented **early stopping** with patience of 10 epochs checking the validation loss.
5. Implemented **learning rate changes**, started with default and used sklearn **ReduceOnPlateau** with patience 5 that is reducing by factor 0.5 once the validation loss plateaus for 5 epochs.
6. Increased the **epoch number** to 50 and then reduced to 30.
7. Checking the accuracy, F1-score, Precision, Recall and confusion matrix metrics and analyzed the learning curves.

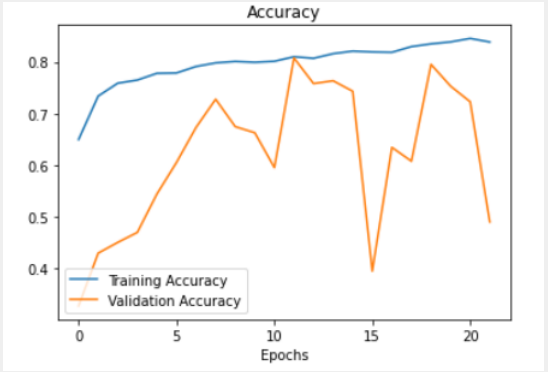
Layer (type)	Output Shape	Param #
input_6 (InputLayer)	[(None, 224, 224, 3)]	0
conv2d_20 (Conv2D)	(None, 224, 224, 256)	7168
max_pooling2d_20 (MaxPooling	(None, 112, 112, 256)	0
batch_normalization_25 (Batc	(None, 112, 112, 256)	1024
dropout_25 (Dropout)	(None, 112, 112, 256)	0
conv2d_21 (Conv2D)	(None, 112, 112, 128)	295040
max_pooling2d_21 (MaxPooling	(None, 56, 56, 128)	0
batch_normalization_26 (Batc	(None, 56, 56, 128)	512
dropout_26 (Dropout)	(None, 56, 56, 128)	0
conv2d_22 (Conv2D)	(None, 56, 56, 64)	73792
max_pooling2d_22 (MaxPooling	(None, 28, 28, 64)	0
batch_normalization_27 (Batc	(None, 28, 28, 64)	256
dropout_27 (Dropout)	(None, 28, 28, 64)	0
conv2d_23 (Conv2D)	(None, 28, 28, 32)	18464
max_pooling2d_23 (MaxPooling	(None, 14, 14, 32)	0
batch_normalization_28 (Batc	(None, 14, 14, 32)	128
dropout_28 (Dropout)	(None, 14, 14, 32)	0
flatten_5 (Flatten)	(None, 6272)	0
dense_10 (Dense)	(None, 512)	3211776
batch_normalization_29 (Batc	(None, 512)	2048
dropout_29 (Dropout)	(None, 512)	0
dense_11 (Dense)	(None, 5)	2565

Total params: 3,612,773
Trainable params: 3,610,789
Non-trainable params: 1,984

1. **WITH ADAM:** Metrics and Learning Curves

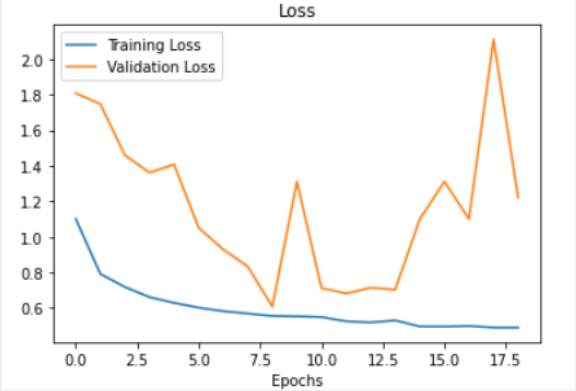
	precision	recall	f1-score	support
0	0.74	0.69	0.71	403
1	0.95	0.83	0.89	407
2	0.89	0.90	0.89	404
3	0.88	0.99	0.93	408
4	0.64	0.67	0.66	403
accuracy			0.82	2025
macro avg	0.82	0.82	0.82	2025
weighted avg	0.82	0.82	0.82	2025

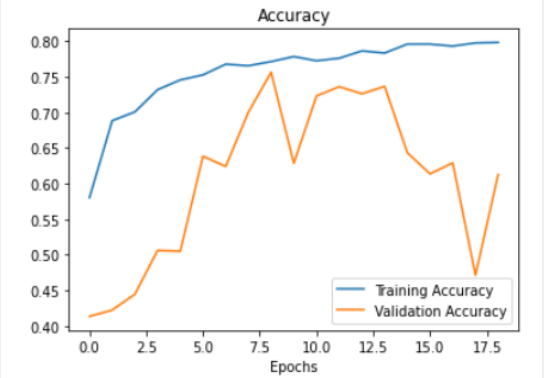




2. **WITH SGD:** Metrics and Learning Curves

	precision	recall	f1-score	support
0	0.60	0.81	0.69	403
1	0.83	0.86	0.84	407
2	0.83	0.91	0.87	404
3	0.94	0.82	0.88	408
4	0.72	0.47	0.57	403
accuracy			0.77	2025
macro avg	0.78	0.77	0.77	2025
weighted avg	0.78	0.77	0.77	2025





FURTHER EXPERIMENTS AND GENERATED RESULTS

Transfer Learning:

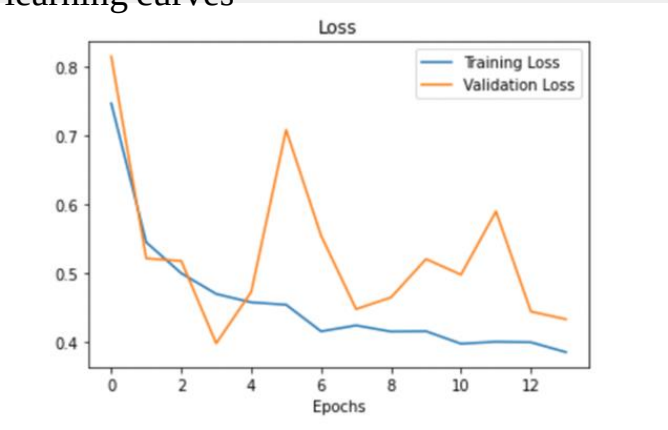
1. We experimented with transfer learning architectures like VGG19, ResNet50, DenseNet 201, Xception Models.
2. Fixed the top layers and implemented training of only flatten and fully connected dense layer defined by us.
3. Checked by using ADAM Optimizer.
4. All pre processing, early stopping, learning rate change are implemented similar to hyper parameter changed model.

Results of VGG19 Transfer Learning:

1. The accuracy is 85% with precision, recall, F1 score as follows

	precision	recall	f1-score	support
0	0.77	0.69	0.73	403
1	0.91	0.94	0.93	407
2	0.86	0.97	0.91	404
3	0.99	0.90	0.94	408
4	0.70	0.72	0.71	403
accuracy			0.85	2025
macro avg	0.85	0.85	0.84	2025
weighted avg	0.85	0.85	0.84	2025

2. The learning curves

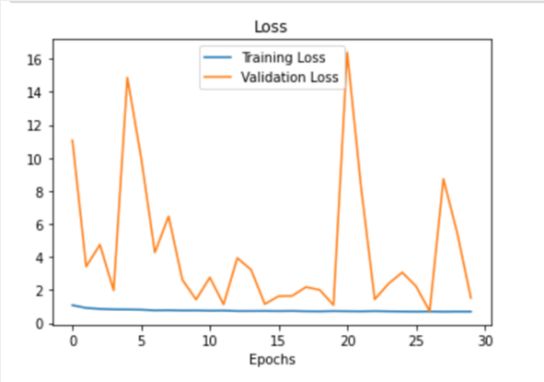


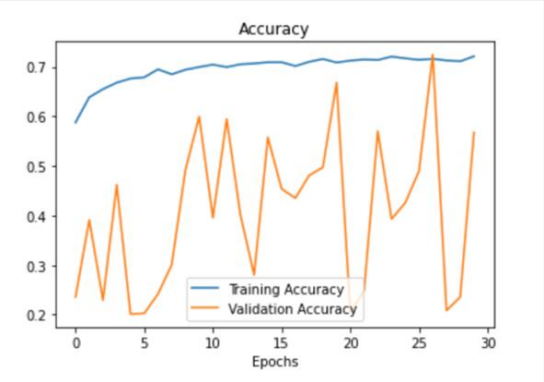
Results of ResNet Transfer Learning:

1. The accuracy is 57% with precision, recall, F1 score as follows

	precision	recall	f1-score	support
0	0.58	0.08	0.15	403
1	0.50	0.96	0.66	407
2	0.72	0.93	0.81	404
3	1.00	0.16	0.27	408
4	0.49	0.73	0.59	403
accuracy			0.57	2025
macro avg	0.66	0.57	0.49	2025
weighted avg	0.66	0.57	0.49	2025

2. The learning curves



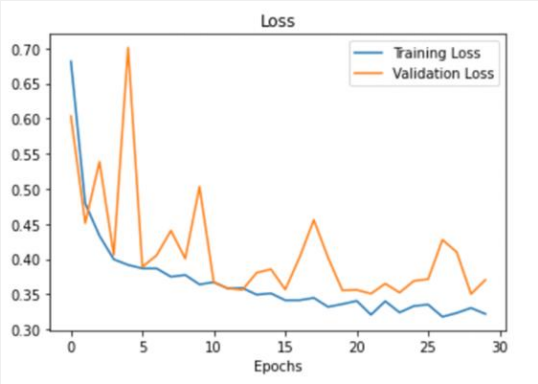


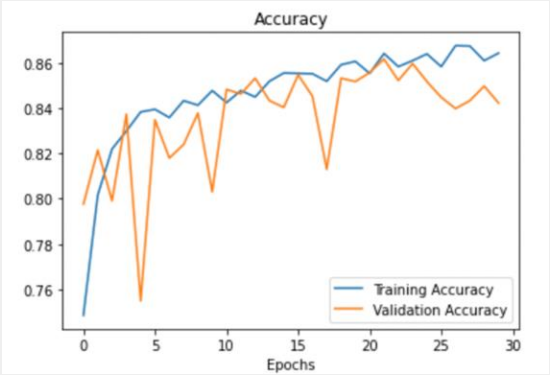
Results of DenseNet Transfer Learning:

1. The accuracy is 86% with precision, recall, F1 score as follows

	precision	recall	f1-score	support
0	0.79	0.63	0.70	403
1	0.99	0.94	0.96	407
2	0.88	0.98	0.93	404
3	0.96	0.99	0.98	408
4	0.68	0.77	0.72	403
accuracy			0.86	2025
macro avg	0.86	0.86	0.86	2025
weighted avg	0.86	0.86	0.86	2025

2. The learning curves





RESULTS DISCUSSION AND RESEARCH QUESTION 1

1. **Why CNN works better for this dataset for classification?**

- We have chosen CNN for our image classification problem.
- As lung disease classification is a very crucial part in treating the patients as early as possible.
- The **False positives and False negatives** are very important as they can create a panic among the healthy patients and may make the disease very dangerous respectively. So detecting correctly is very appropriate.
- Images have spatial information which is lost when flattened, so in order to extract all the features without losing much information and also removing unnecessary features which otherwise accounts for lot of data processing is possible using only Convolutional Neural Networks.
- Also with different augmentation techniques CNN learns the features precisely.

RESULTS DISCUSSION AND RESEARCH QUESTION 2

2. Analyzing the performance of dataset with different CNN architectures:

1. We observe that a baseline model with 2 CNN layers , 32 feature maps, batch normalization, RELU activation, drop out of 0.2 probability was giving accuracy around 60%
2. But we observed that to the baseline model 4 CNN layers with 256, 128, 64, 32 feature maps respectively, He initialization, bias initialization, drop out of 0.1 probability was giving accuracy around 78%

INFERENCE:

As the number of layers increase with good bias and weights initialization, batch normalization, the vanishing or exploding gradients problem is solved and hence output will be estimated appropriately.

RESULTS DISCUSSION AND RESEARCH QUESTION 3

3. Analyzing the metrics, learning curves of dataset with different optimizers:

SGD OPTIMIZER INFERENCE:

- We initially started with SGD optimizer and observed an accuracy of 77%. This skips the local minimums but takes lot of time to converge.
- We tried adjusting the learning rate but could not achieve a greater accuracy.
- Since it has lot many oscillations, setting a learning rate was difficult. This can be observed in the learning curves too.

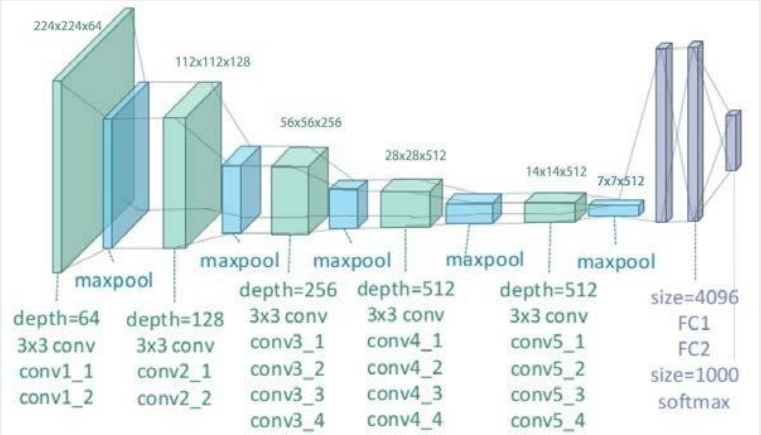
ADAM OPTIMIZER INFERENCE:

- ADAM uses both the adaptive learning rate and momentum of Gradient Descent Optimizer features. We achieved an accuracy of 82%.
- This was performing very well and converged to global minimum faster than SGD hence the computational time is reduced.

RESULTS DISCUSSION AND RESEARCH QUESTION 4

4. How Transfer learning is affecting the classification metrics for this dataset?

VGG19:

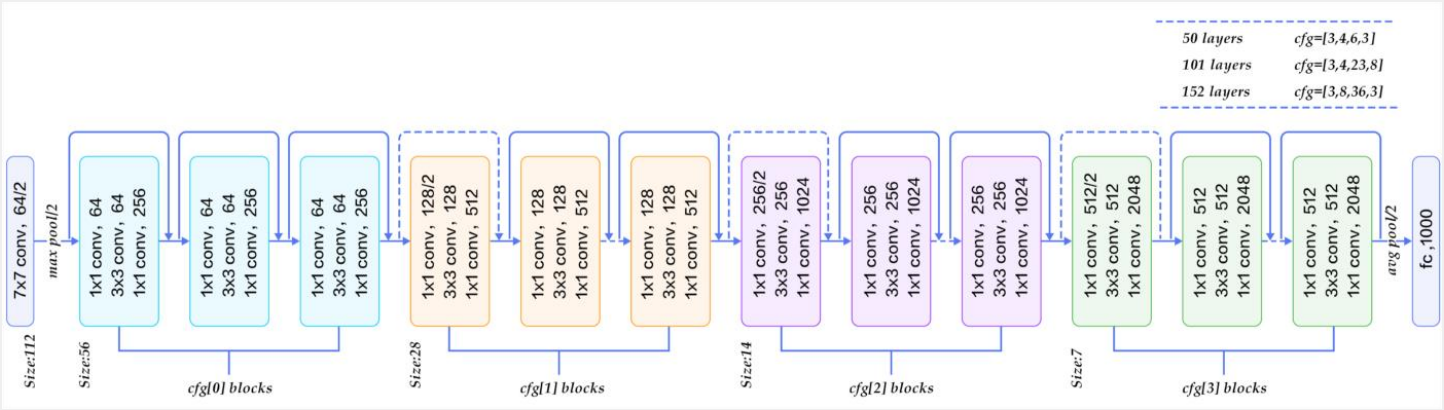


- VGG is Visual Geometry Group, a standard deep convolutional neural network layer with multiple layers. VGG-19 consists of 19 convolutional layers.
- By using this network we got an accuracy of 85%

RESULTS DISCUSSION AND RESEARCH QUESTION 4

4. How Transfer learning is affecting the classification metrics for this dataset?

ResNet50:

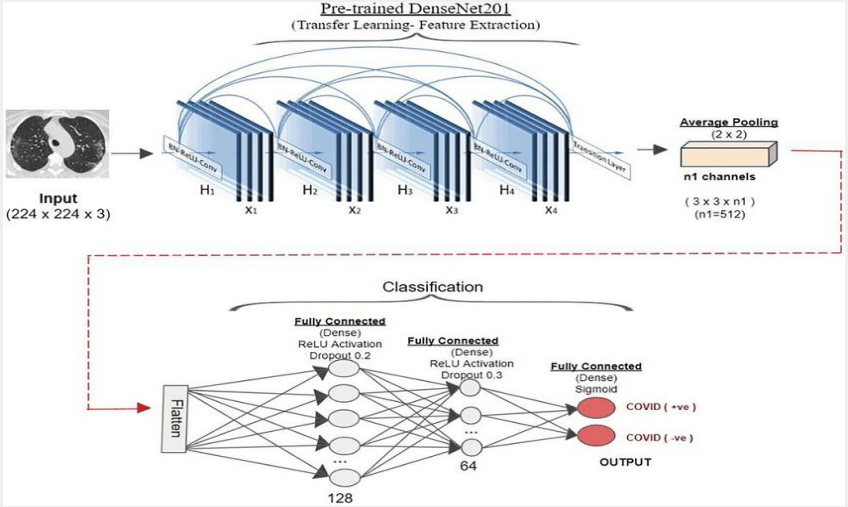


- ResNet is a Residual Network with 50 layers. This is a residual model used for solving accuracy saturation problem occurred in models with lot of layers. It has identity layers added to deeper layers to reduce any higher training error.
- By using this network we got an accuracy of 57%

RESULTS DISCUSSION AND RESEARCH QUESTION 4

4. How Transfer learning is affecting the classification metrics for this dataset?

DenseNet 201:



- In DenseNet 201 each layer receives extra inputs from preceding layers and passes their own feature maps to all subsequent layers. Each layer gets collective knowledge from all preceding layers. It has 201 layers.
- By using this network we got an accuracy of 86%
- We also tried Xception transfer learning model but got an accuracy of 82%.

CONCLUSION

- To conclude, we observe that for our lung disease detection, transfer learning is giving a better result with 86% accuracy with DenseNet and 85% with VGG19.
- If data is not so complex then a normal CNN architecture with decent number of layers and good optimizer with normalization techniques would give good results instead of using transfer learning methods.
- For any image data, transfer learning methods always ensures a good amount of performance.
- Therefore there is always a balance between computational time and accuracy obtained. Based on our application and resources we have we choose it.

THANK YOU