LUNG DISEASE DETECTION

Lakshmi Sai Hanuhitha Dondapati (ldondapa)
Rishika Samala (rsamala)
Sahithi Vasireddy (svasire)

LUDDY SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING

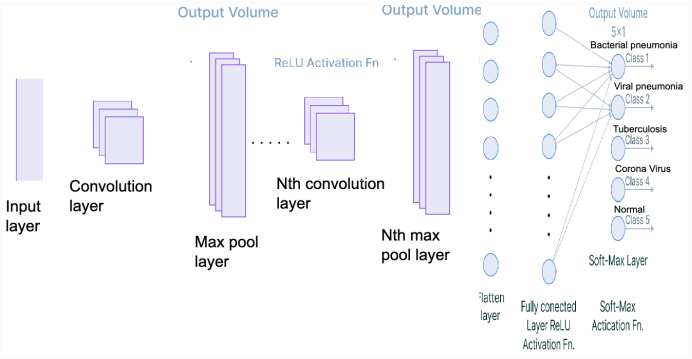
ABSTRACT

In this project, we aim to identify and foresee the type of lung disease using chest x-ray image data. Although we are aware of some common problems like asthma, lung inflammation, and other bacterial infections, we should be more concerned about critical lung disorders in recent times. Our problem statement is to categorize the lung X-ray images into 5 classes: Bacterial Pneumonia, Viral Pneumonia, Corona Virus, Tuberculosis, and images with no lung infection at all (Normal). The dataset has 10,000 image data and is further divided into the train, test, and validation sets. The dataset is balanced. The train set has bacterial pneumonia with 1205 images, coronavirus disease with 1218 images, normal with 1207 images, Tuberculosis with 1220 images, and viral pneumonia with 1204 images. The validation set has bacterial pneumonia with 401 images, coronavirus disease with 406 images, normal with 402 images, Tuberculosis with 406 images, and viral pneumonia with 401 images. The test set has bacterial pneumonia with 401 images, coronavirus disease with 407 images, normal with 404 images, Tuberculosis with 408 images, and viral pneumonia with 403 images. We are implementing a Convolutional Neural Network with the Keras library to classify the type of lung disease.

1. INTRODUCTION

Respiratory illnesses are one of the most common causes of death in the world. Lung infections such as bacterial pneumonia, tuberculosis, viral pneumonia, and coronavirus. The X-ray images of these diseases are hard to differentiate by the naked eye. Moreover, these infections have a lot of common symptoms like cough, weight loss, fever/chills, night sweats, shortness of breath, and so on. Because of this, early diagnosis of these types of diseases helps in the early treatment which may later cost a life. Radiologists find it difficult to classify abnormalities on chest X-rays. Hence a Convolutional Neural Network can be used to classify them. Convolutional neural networks are among the most powerful deep neural networks because they can include multiple hidden layers that perform convolution and subsampling to extract low to high levels of features from the input data. Computer vision, biological computation, fingerprint improvement, and other applications have all shown remarkable efficiency using this network.

In the convolutional layer, an input of size R*C is convolved with a kernel (filter) of size x*x. Each block of the input matrix is independently convolved with the kernel and generates a pixel in the output. The output image features are generated using the result of the convolution of the input image and kernel. In general, the output image features acquired by convolving the kernel and the input images are referred to as feature maps, whereas the kernel of the convolution matrix is referred to as a filter. The feature vector is the inputs and outputs to future convolutional layers of a CNN, which can include numerous convolutional layers. There are n filters in each convolution layer. The number of filters used in the convolution technique determines the depth of the generated feature maps (n).



2. RELATED WORKS AND DIFFERENTIATION

There has been a significant amount of research in detecting different lung diseases using machine learning models. There are many projects where each class that we considered here in this project, has been taken individually and was detected. To name a few, [1], [3] , [4], [5]. In 2019, CNN was used for the detection of pneumonia from chest X-ray images in Lin et al.[2]. The research was on the Kaggle dataset designed with a convolutional neural network. Furthermore, there are other projects using

computer vision and a cooperative CNN model that was proposed by Wang et al. In 2020, there exists research on covid detections using X-ray images [6] and other anomaly detection methods [7].
From these works, we can observe that all the prior research is focused on classifying one type of disease. Also, some projects considered two types of diseases and implemented classification on the model. This is a major difference as the model we are focusing on compares the diseases with similar symptoms and can make accurate predictions.

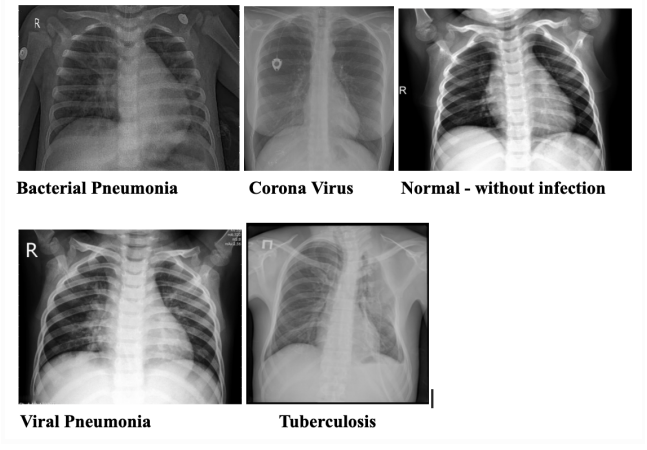
3. OUR METHODOLOGY AND APPROACH

Our project is partitioned into 5 crucial steps which are given as follows:

1. Data Pre-Processing: Normalizing the images, checking for corrupted files. Implementing script for data loading.
2. Baseline Modeling: Implement a Convolutional Neural Network (CNN) architecture for the classification task. Start training the model.
3. Define model evaluation metrics like precision, recall, and f-score.
4. Experimenting with different CNN architectures. Try transfer learning methods.
5. Comparing the results and figuring out which model performs the best.

4. BASELINE IMPLEMENTATION

We are using Keras and TensorFlow libraries for implementing the steps involved. Here we have 5 classes: Bacterial Pneumonia, Viral Pneumonia, CoronaVirus, and Tuberculosis, and images with no lung infection at all (Normal).



4.1 Data Preprocessing:

We took our dataset from Kaggle which already has train, validation, and test data separated. Started preprocessing our dataset images.

1. We used Image Data Generator to rescale them since our original images contain some RGB components and they are high for our model to process. So we made them in between 0 and 1 by scaling with 1/255. We also used width_shift_range, height_shift_range, and zoom range for our images to not be sensitive to width, height, and zoom parameters while training.
2. We used the flow_from_directory() method which takes the path of the given directory to load the training, validation, and test data, and based on the batch size given it generates batches of data. We started with a batch size of 32 initially.
3. Since the input images contain different sizes, for all the data to be uniform for the training and testing process we resized all the input data to size 224*224.
4. Our labels are categories of lung disease of 5 different classes, so we used class mode as categorical and used shuffling of data.

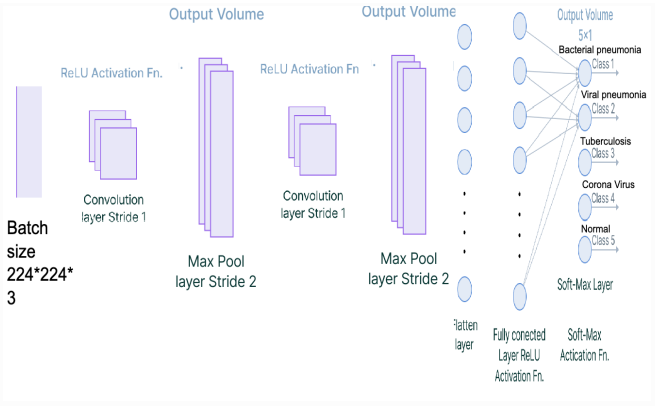
4.2 Baseline Modeling:

1. We started with defining a CNN architecture with 2 convolutional layers, 2 max-pooling layers after each convolutional layer, flatten the output of max-pooling layers, and then 1 fully connected hidden dense layer, one output layer.
2. CNN Layer1: Implemented a 2D convolutional layer with kernel size 3*3, activation function as RELU, and padding, with 32 feature layers.
3. Max Pooling Layer1: The output of the convolutional layer 1 is max pooled and then batch-normalized the max pooled outputs.
4. CNN Layer2: Implemented a 2D convolutional layer with kernel size 3*3, activation function as RELU, and padding.
5. Max Pooling Layer2: The output of the convolutional layer 2 is max pooled and then batch-normalized the max pooled outputs.
6. Flattened the outputs of batch normalization and given to 1 hidden fully connected layer with 512 output layers and activation function as RELU.
7. This output is again batch normalized and sent to a fully connected output layer with 5 output layers to detect the 5 categorical labels with activation function as softmax.
8. We implemented a dropout of 20% for each layer.
9. The optimizer used in ADAM and a loss function of cross-entropy are defined.

4.3 Evaluation Metrics:

1. We calculated accuracy, Confusion matrix, and classification report metrics and have built our base model.

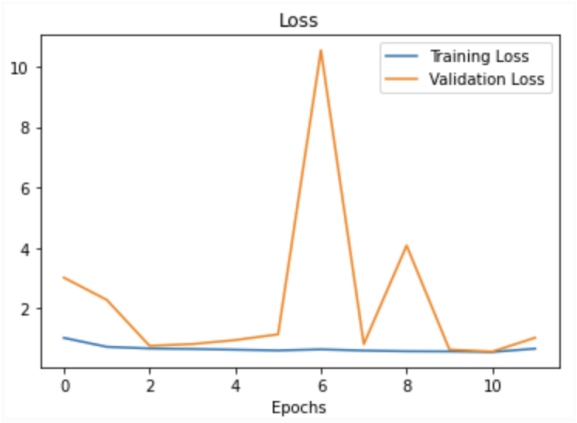
4.4 Our Network Architecture Setup:



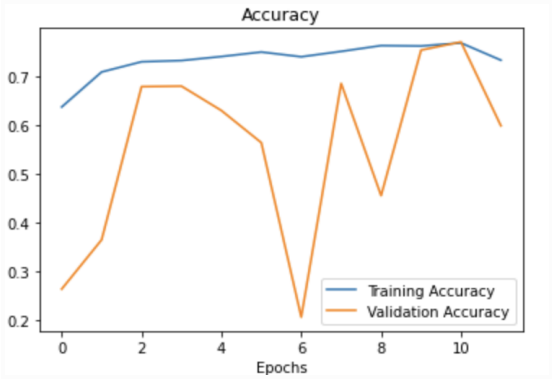
4.5 Baseline Model Results:

1. We built the base model with the above-defined CNN architecture and trained the model with 12 epochs initially.
2. Used this model with test data and found the accuracy to be 61.5%.
3. Plotted the learning curves as below.

1. Loss Curve



2. Accuracy Curve



5. FURTHER EXPERIMENTS AND SETUP

With about 60% accuracy we are implementing different tuning of the hyperparameters and finally implementing a transfer learning algorithm for better performance and to compare the results and figure which model works better for the model.

5.1 Hyperparameters Tuning:

We have experimented with different types and tuned the hyperparameter tuning as follows:

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

5.1.1 OUR EXPERIMENTS SETUP

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)	32117
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

5.2 Hyperparameter tuned model and Results:

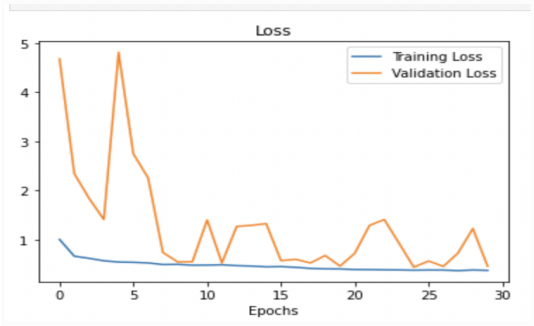
5.2.1 With ADAM Optimizer

Metrics, Confusion Matrix and Learning Curves:

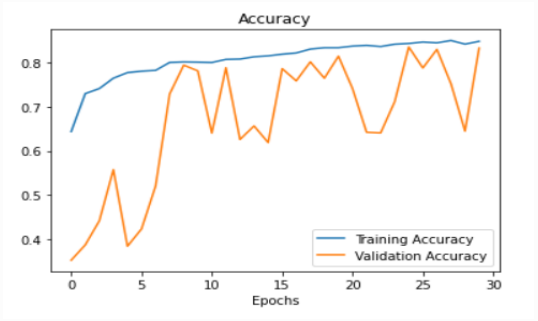
	precision	recall	f1-score	support
0	0.70	0.82	0.75	403
1	0.94	0.93	0.93	407
2	0.89	0.90	0.89	404
3	0.96	0.97	0.96	408
4	0.75	0.61	0.67	403
accuracy			0.85	2025
macro avg	0.85	0.84	0.84	2025
weighted avg	0.85	0.85	0.84	2025

```
[[330   1  13   1  58]
 [  7 379   3  14   4]
 [ 16   7 362   0  19]
 [  0  13   0 395   0]
 [120   5  31   1 246]]
```

1. Loss Curve:



2. Accuracy Curve:



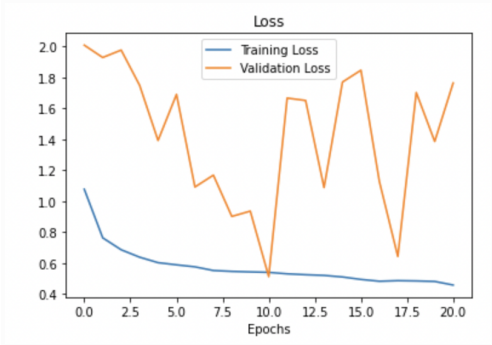
5.2.2 With SDG Optimizer

Metrics, Confusion Matrix and Learning Curves:

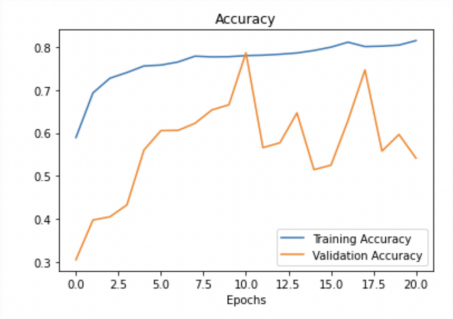
	precision	recall	f1-score	support
0	0.64	0.81	0.72	403
1	0.95	0.78	0.86	407
2	0.82	0.93	0.87	404
3	0.87	0.99	0.93	408
4	0.76	0.48	0.59	403
accuracy			0.80	2025
macro avg	0.81	0.80	0.79	2025
weighted avg	0.81	0.80	0.79	2025

```
[[328   0  28   3  44]
 [ 19 318   7  56   7]
 [  9   7 377   0  11]
 [  0   3   0 405   0]
 [157   5  47   0 194]]
```

1. Loss Curve:



2. **Accuracy Curve:**



5.3 Transfer Learning:

1. We experimented with transfer learning architectures like VGG19, ResNet50, DenseNet 201, and Xception Models.
2. Fixed the top layers and implemented training of only flatten and fully connected dense layers defined by us.
3. Checked by using ADAM Optimizer.
4. All pre-processing, early stopping, and learning rate changes are implemented the same as the hyperparameter tuned model.

5.3.1 Results of VGG19 Transfer Learning:

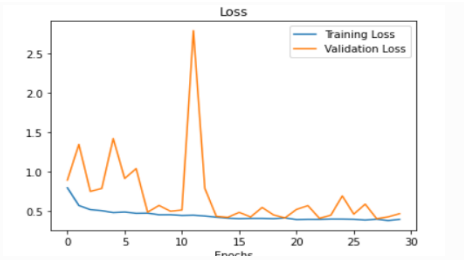
1. The accuracy is around 85% with precision, recall, F1 score, Confusion Matrix as follows:

	precision	recall	f1-score	support
0	0.72	0.71	0.71	403
1	0.98	0.92	0.95	407
2	0.82	0.98	0.89	404
3	0.97	0.98	0.98	408
4	0.72	0.63	0.67	403
accuracy			0.84	2025
macro avg	0.84	0.84	0.84	2025
weighted avg	0.84	0.84	0.84	2025

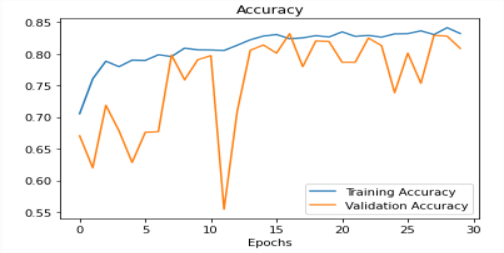
```
[[285   3  28   0  87]
 [  8 374   9  11   5]
 [  1   0 396   0   7]
 [  0   5   4 399   0]
 [102   0  48   0 253]]
```

2. Learning curves:

1. **Loss Curve:**



2. **Accuracy Curve:**



5.3.2 Results of ResNet Transfer Learning:

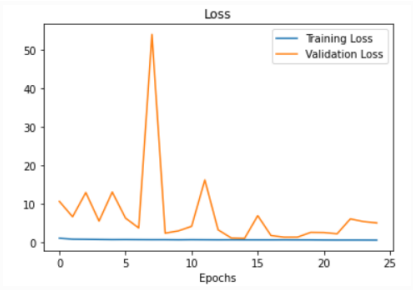
1. The accuracy is 59% with precision, recall, F1 score, confusion matrix as follows:

	precision	recall	f1-score	support
0	0.44	0.91	0.59	403
1	0.91	0.10	0.18	407
2	0.79	0.89	0.84	404
3	0.61	0.97	0.75	408
4	0.68	0.06	0.11	403
accuracy			0.59	2025
macro avg	0.68	0.59	0.49	2025
weighted avg	0.68	0.59	0.49	2025

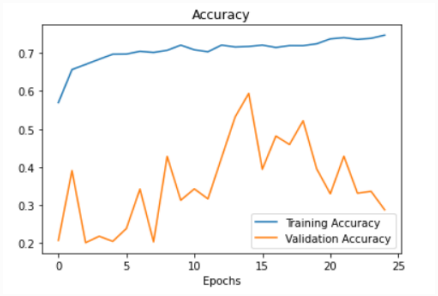
```
[[367   0  20   8   8]
 [103  41  26 236   1]
 [ 37   2 360   4   1]
 [  5   2   5 394   2]
 [327   0  47   4  25]]
```

2. Learning Curves

1. **Loss Curve:**



2. **Accuracy Curve:**



5.3.3 Results of DenseNet Transfer Learning:

1. The accuracy is 88% with precision, recall, F1 score, confusion matrix as follows:

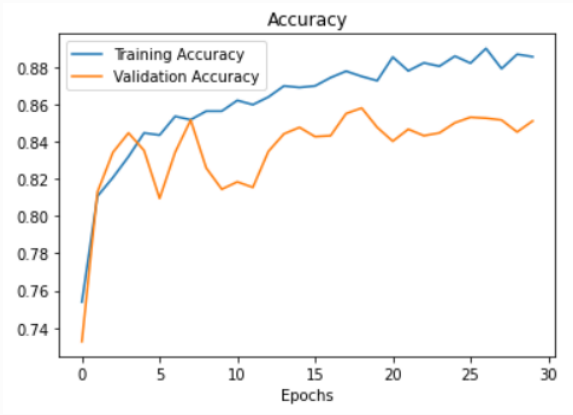
	precision	recall	f1-score	support
0	0.77	0.74	0.75	403
1	0.98	0.97	0.97	407
2	0.89	0.98	0.93	404
3	0.99	0.99	0.99	408
4	0.75	0.71	0.73	403
accuracy			0.88	2025
macro avg	0.87	0.88	0.87	2025
weighted avg	0.88	0.88	0.88	2025

```
[[298   2  11   0  92]
 [  0 393   8   6   0]
 [  3   0 396   0   5]
 [  1   5   0 402   0]
 [ 85   0  31   0 287]]
```

2. Learning Curves

1. Loss Curve:

2. Accuracy Curve:



6. RESEARCH QUESTIONS AND PROJECT TAKEAWAYS

6.1 Why does CNN work better for this dataset for classification?

1. We have chosen CNN for our image classification problem.
2. As lung disease classification is a very crucial part of treating the patients as early as possible.
3. 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.
4. Images have spatial information which is lost when flattened, so extracting all the features without losing much information and also removing unnecessary features which otherwise account for a lot of data processing is possible using only Convolutional Neural Networks.
5. Also with different augmentation techniques, CNN learns the features precisely.

6.2. Why does CNN work better for this dataset for classification?

1. Analyzing the performance of dataset with different CNN architectures:
    a. 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 an accuracy of around 60%
    b. But we observed that in the baseline model 4 CNN layers with 256, 128, 64, and 32 feature maps respectively, He initialization, bias initialization, and drop out of 0.1 probability was giving an accuracy of around 78%.

6.2.1 INFERENCE:

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

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

SGD OPTIMIZER INFERENCE:

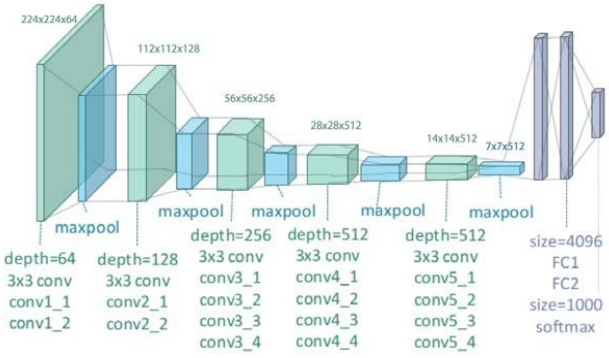
1. We initially started with an SGD optimizer and observed an accuracy of 80%. This skips the local minimums but takes a lot of time to converge.
2. We tried adjusting the learning rate but could not achieve greater accuracy.
3. Since it has a lot of oscillations, setting a learning rate was difficult. This can be observed in the learning curves too.

ADAM OPTIMIZER INFERENCE:

1. ADAM uses both the adaptive learning rate and momentum of Gradient Descent Optimizer features. We achieved an accuracy of 85%.
2. This was performing very well and converged to a global minimum faster than SGD hence the computational time is reduced.

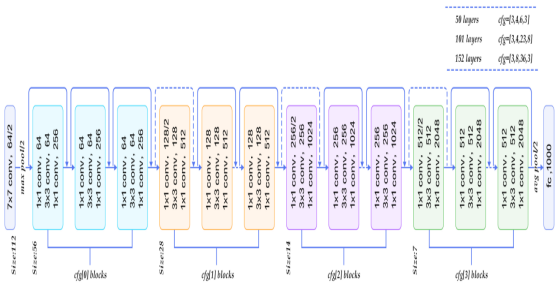
6.4. How Transfer learning is affecting the classification metrics for this dataset?

6.4.1 VGG19:



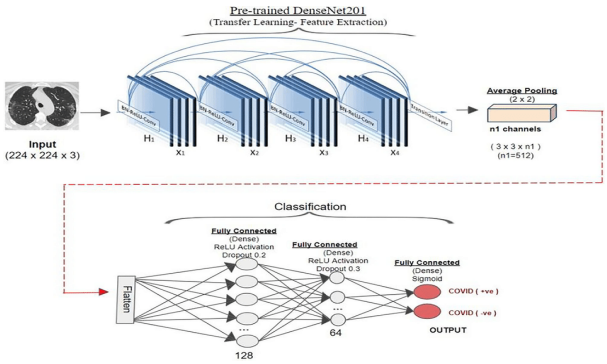
1. VGG is Visual Geometry Group, a standard deep convolutional neural network layer with multiple layers. VGG-19 consists of 19 convolutional layers.
2. By using this network we got an accuracy of 85%.

6.4.2 ResNet50:



1. ResNet is a Residual Network with 50 layers. This is a residual model used for solving accuracy saturation problems that occurred in models with a lot of layers. It has identity layers added to deeper layers to reduce any higher training error.
2. By using this network we got an accuracy of 59%.

6.4.3 DenseNet 201:



1. In DenseNet 201 each layer receives extra inputs from preceding layers and passes its feature maps to all subsequent layers. Each layer gets collective knowledge from all preceding layers. It has 201 layers.
2. By using this network we got an accuracy of 86%
3. We also tried the Xception transfer learning model but got an accuracy of 88%.

7. FUTURE DIRECTIONS

1. We experimented with 4 convolutional layers and various Transfer Learning methods. But we got the maximum F1 score to be 75.
2. In the medical field, this is not enough, we need to achieve as high an F1 score as possible to reduce

the False Positives and False Negatives.

3. So our future plan is to implement different optimizers and different network architectures to try to achieve a good F1 score.
4. We also plan to try different Transfer Learning methods by making the model learn the upper transfer layers also and try to improve the metrics.

8. CONCLUSION

1. Lung X-ray detection classification is a very challenging problem and we observe that for our lung disease detection, transfer learning is giving a better result with 88% accuracy and 75 F1 Score with DenseNet Transfer Learning.

2. If data is not so complex then a normal CNN architecture with a decent number of layers and a good optimizer with normalization techniques would give good results instead of using transfer learning methods.

3. For any image data, transfer learning methods always ensure a minimum good amount of performance.

4. But if we tried to implement a network with a 7-8 number of layers and a different number of feature maps to improve the F1 Score, the algorithm was taking 9-10 hours to converge.

5. Therefore there is always a tradeoff between computational time and accuracy obtained. Based on our application and the resources we have we can choose them.

9. REFERENCES

[1] https://www.sciencedirect.com/science/article/pii/S2352914820300290

[2] https://link.springer.com/article/10.1007/s12652-021-03464-7#ref-CR24

[3] https://ieeexplore.ieee.org/abstract/document/9224622

[4] https://www.mdpi.com/2075-4426/11/1/28

[5] https://ieeexplore.ieee.org/abstract/document/9524697

[6] https://arxiv.org/pdf/2004.09803.pdf

[7] https://covid-19.conacyt.mx/jspui/handle/1000/4271

Dataset:

[8] https://www.kaggle.com/omkarmanohardalvi/lungs-disease-dataset-4-types