# Pneumonia Detection using Deep Transfer Learning

CS 577 S21 Deep Learning – Project Report

Department of Computer Science
Illinois Institute of Technology

May 04, 2021

Haripriya Aravapalli
A20445663
haravapalli@hawk.iit.edu

Avinash Shekar
A20447945
ashekar2@hawk.iit.edu

## Research Paper: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7345724/

Efficient Pneumonia Detection in Chest Xray Images Using Deep Transfer Learning

Mohammad Farukh Hashmi, Satyarth Katiyar, AvinashG Keskar, Neeraj Dhanraj Bokde, and Zong Woo Geem

## Data Source: https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia

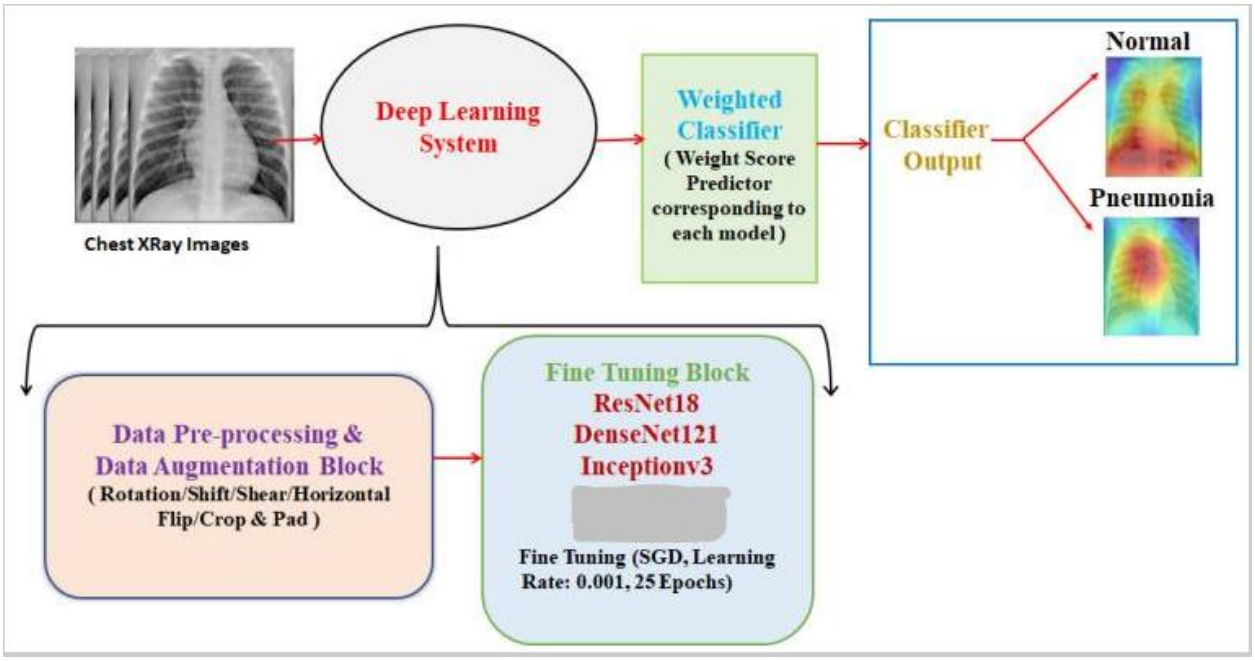
The dataset contains subfolders for each image category (Pneumonia/Normal). There are 5,863 X-Ray images (JPEG) and 2 categories (Pneumonia/Normal).

## Problem Statement:

Pneumonia affects 7% of the global population. Chest X-rays are primarily used for the diagnosis of this disease. However, even for a trained radiologist, it is a challenging task to examine chest X-rays. According to the WHO, pneumonia can be prevented with a simple intervention and early diagnosis and treatment.[1] There is a need to improve the diagnosis accuracy. Our aim is to propose a model for quick diagnosis of pneumonia and potentially save lives.

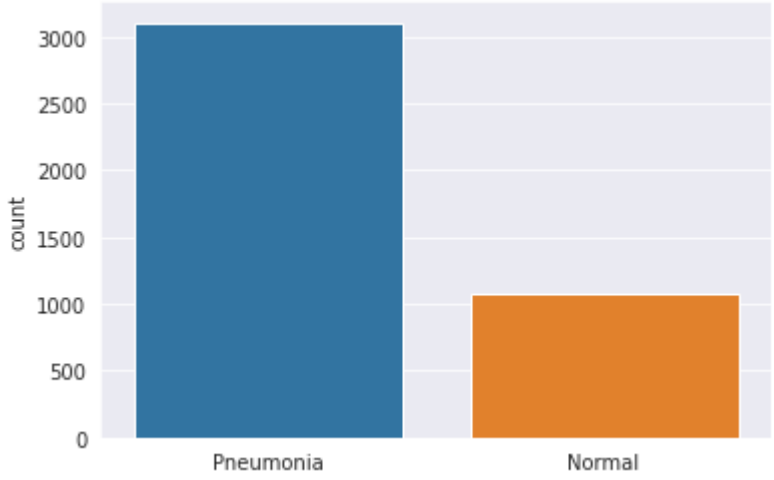
## Methodology:

1. Transform data to python readable format.
2. Dataset is already divided into train, validate, and test.
3. Perform the Exploratory Data Analysis.
4. Preprocess the data and Implement Data Augmentation.
5. Try out a base model of Deep Neural Network.
6. Compare with ResNet18, DenseNet121, and InceptionV3 pre-trained neural networks.
7. Implement a Weighted Classifier for all the models and obtain the classified output.

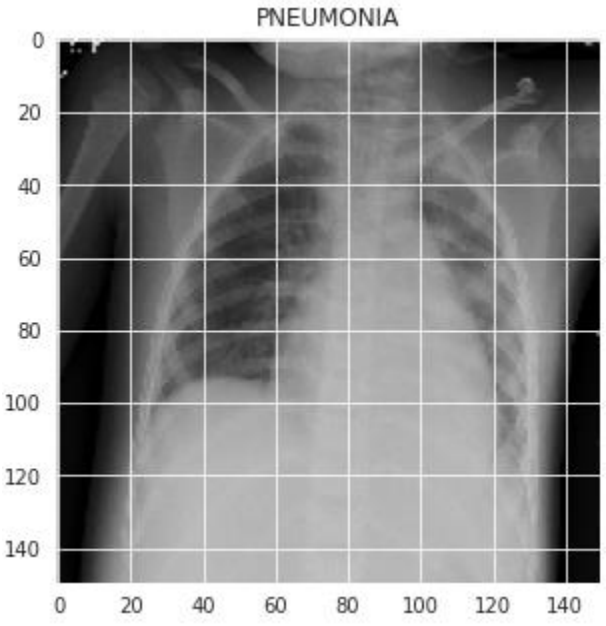


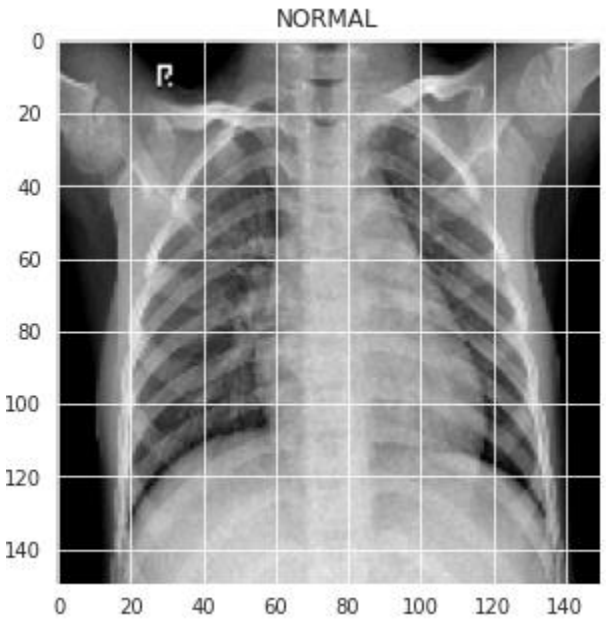

## Exploratory Data Analysis:

The distribution of Pneumonia and Normal classes is as shown below. In the dataset we have 1341 Normal images and 3875 Pneumonia images.



The images are in jpeg format. Initially we have resized all the images to 150 x 150. A random sample of images is shown below:




## Data Preparation:

Approximately 20% of dataset is used as test set and the remaining as train dataset. Each image should be preprocessed according to the Convolutional neural network used. Images are normalized and then resized as per the Deep Learning model. For CNN model we have resized the images to 150 x 150. For ResNet50 and DenseNet121 we have resized them to 224 x 224 and for InceptionV3, images are resized to 229 x 229.

## Data Augmentation:

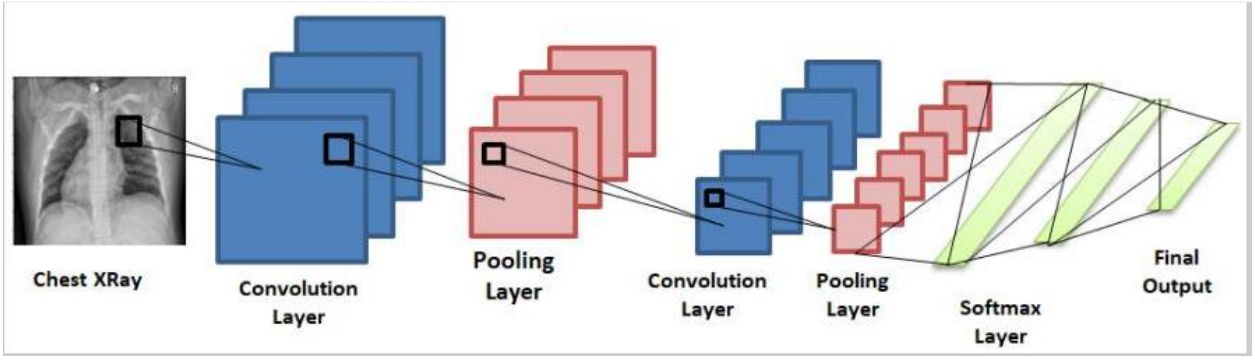
Deep Learning models trained to perform complex tasks require large number of data to learn the values during training. In order to deal with the problem of limited data we use Data Augmentation. We apply different transformations to the available data to synthesize new data. Augmented data can also be used to address the problem of class imbalance in classification. Here we have only 1341 Normal images and 3875 Pneumonia images. We have performed the following transformations to the data to get enough images of normal case in the training dataset.

Rotation of 30 degrees, vertical and horizontal shift of 0.1, zoom range of 0.2 and a Horizontal flip of the images.

## CNN Model:

We have used a convolutional neural network with 5 convolutional layers using 3 x 3 filter, Max pooling with 2 x 2, Batch Normalization and Dropout Regularization of 0.2. Resulting images are flattened and then passed through 2 Dense Layers. We have used sigmoid as activation function

for output layer with one neuron. The optimizer used is RMSProp, loss function to minimize is binary crossentropy. Metric to optimize is accuracy. We have used ReduceLROnPlateau which is a callback to reduce the learning rate when a metric has stopped improving. This will monitor Validation accuracy and if it is not improving learning rate is reduced by a factor of 0.3. Training is done for 25 epochs. The summary of the model is shown below:




```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 150, 150, 32)      896
_________________________________________________________________
batch_normalization (BatchNo (None, 150, 150, 32)      128
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 75, 75, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 75, 75, 64)        18496
_________________________________________________________________
dropout (Dropout)            (None, 75, 75, 64)        0
_________________________________________________________________
batch_normalization_1 (Batch (None, 75, 75, 64)        256
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 38, 38, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 38, 38, 64)        36928
_________________________________________________________________
batch_normalization_2 (Batch (None, 38, 38, 64)        256
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 19, 19, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 19, 19, 128)       73856
_________________________________________________________________
dropout_1 (Dropout)          (None, 19, 19, 128)       0
_________________________________________________________________
batch_normalization_3 (Batch (None, 19, 19, 128)       512
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 10, 10, 128)       0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 10, 10, 256)       295168
```

```
_________________________________________________________________
dropout_2 (Dropout)          (None, 10, 10, 256)       0
_________________________________________________________________
batch_normalization_4 (Batch (None, 10, 10, 256)       1024
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 5, 5, 256)         0
_________________________________________________________________
flatten (Flatten)            (None, 6400)              0
_________________________________________________________________
dense (Dense)                (None, 128)               819328
_________________________________________________________________
dropout_3 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 129
=================================================================
Total params: 1,246,977
Trainable params: 1,245,889
Non-trainable params: 1,088
```

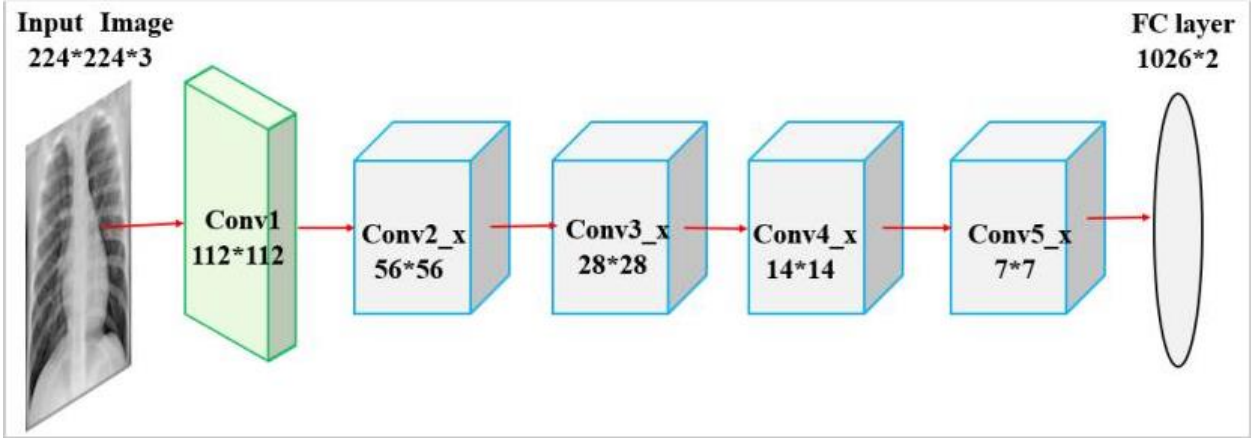
The performance of the model is validated using accuracy, precision, recall, f1-score and support. In order to calculate these values confusion matrix is obtained.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Pneumonia (Class 0) | 0.93 | 0.93 | 0.93 | 390 |
| Normal (Class 1) | 0.88 | 0.88 | 0.88 | 234 |
| accuracy | | | 0.91 | 624 |
| macro avg | 0.90 | 0.91 | 0.90 | 624 |
| weighted avg | 0.91 | 0.91 | 0.91 | 624 |

Accuracy of CNN Model: 0.9102564102564102

## ResNet50 Model:

We have used transfer learning to train the model. In transfer learning pre trained models are used as starting point for some specific tasks. When the network is too deep, the gradients of the loss function can shrink to zero resulting in vanishing gradient. ResNet model solves this problem using skip connections. Considered ResNet50 as base model. The network consists of five convolutional blocks and the original input is combined along with the output to get the result. We have used Stochastic gradient descent as optimizer for this model. The summary of model is shown below:




Below is the model trained using ResNet50 as base model for our dataset:

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
resnet50 (Functional)        (None, 7, 7, 2048)        23587712
_________________________________________________________________
flatten_1 (Flatten)          (None, 100352)            0
_________________________________________________________________
dropout_4 (Dropout)          (None, 100352)            0
_________________________________________________________________
dense_2 (Dense)              (None, 128)               12845184
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 129
=================================================================
Total params: 36,433,025
Trainable params: 36,379,905
Non-trainable params: 53,120
```

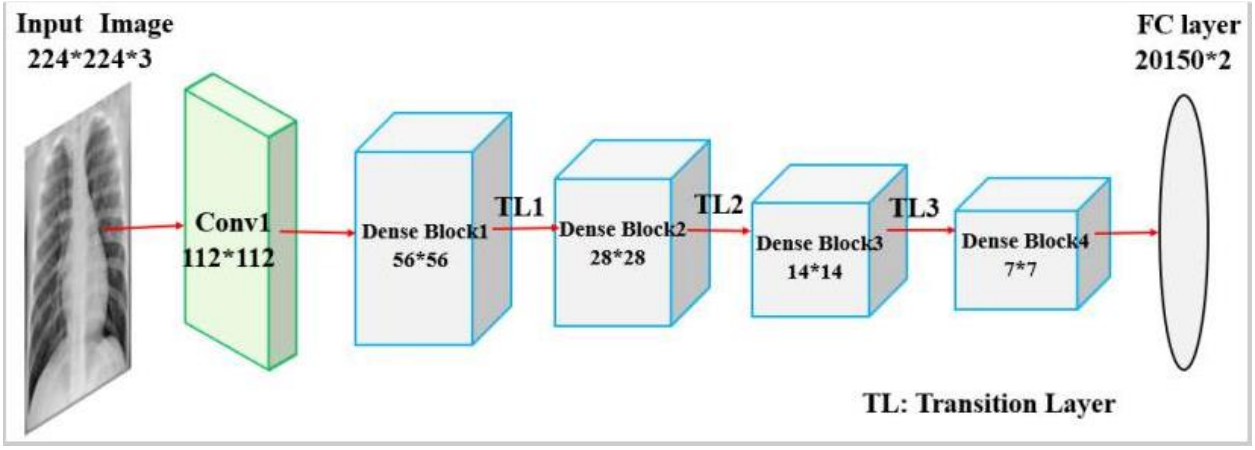
| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Pneumonia (Class 0) | 0.92 | 0.99 | 0.95 | 390 |
| Normal (Class 1) | 0.98 | 0.85 | 0.91 | 234 |
| accuracy | | | 0.94 | 624 |
| macro avg | 0.95 | 0.92 | 0.93 | 624 |
| weighted avg | 0.94 | 0.94 | 0.93 | 624 |

Accuracy of ResNet50 Model: 0.9358974358974359

After using the ResNet50 as base Model accuracy has been increased to 93%.

## DenseNet121 Model:

DenseNet model is designed to improve the declined accuracy caused due to vanishing gradient in high level neural networks. DenseNet consists of 4 Dense blocks and 3 transition layers. Here each layer is connected to all other layers and concatenates all the features. DenseNet is taken as base model to reduce the number of parameters and then trained for the current dataset. The optimizer used is stochastic gradient descent. The summary of the model is shown below:




Below is the model trained using DenseNet121 as base model for our dataset:

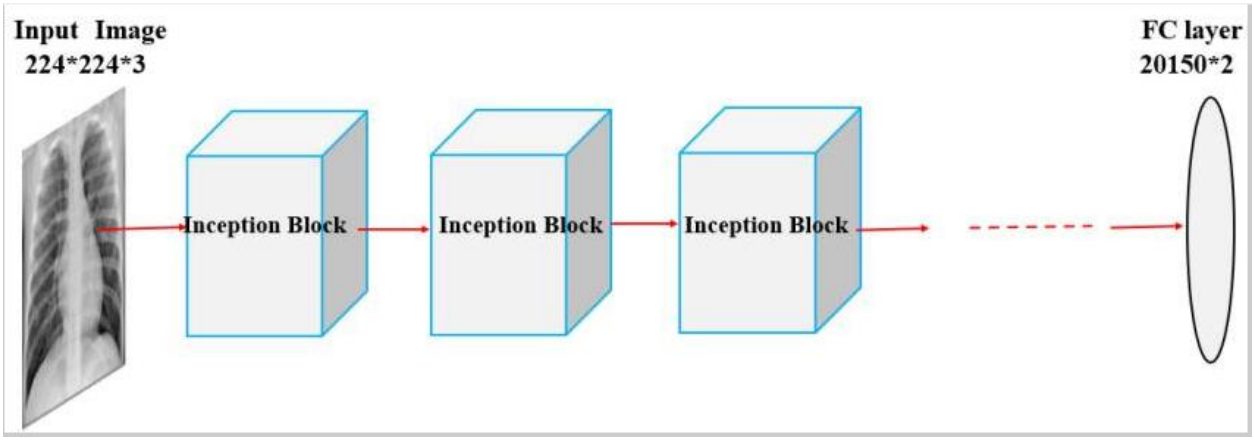
```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
densenet121 (Functional)     (None, 7, 7, 1024)        7037504
_________________________________________________________________
flatten_3 (Flatten)          (None, 50176)             0
_________________________________________________________________
dropout_6 (Dropout)          (None, 50176)             0
_________________________________________________________________
dense_6 (Dense)              (None, 128)               6422656
_________________________________________________________________
dense_7 (Dense)              (None, 1)                 129
=================================================================
Total params: 13,460,289
Trainable params: 13,376,641
Non-trainable params: 83,648
```

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Pneumonia (Class 0) | 0.94 | 0.98 | 0.96 | 390 |
| Normal (Class 1) | 0.97 | 0.90 | 0.94 | 234 |
| accuracy | | | 0.95 | 624 |
| macro avg | 0.96 | 0.94 | 0.95 | 624 |
| weighted avg | 0.95 | 0.95 | 0.95 | 624 |

After using the DenseNet121 as base Model accuracy has been increased to 95%.

## InceptionV3 Model:

InceptionV3 model addresses the problem of increasing computational costs and parameters as the learning model goes deeper. An Inception block is introduced in the network where convolutional layers are implemented parallelly. These inception blocks uses 1x1 convolutional filters to reduce the number of parameters in each block. The optimizer used is stochastic gradient descent. The summary of the model is shown below:




Below is the model trained using InceptionV3 as base model for our dataset:

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
inception_v3 (Functional)    (None, 5, 5, 2048)        21802784
_________________________________________________________________
flatten_2 (Flatten)          (None, 51200)             0
_________________________________________________________________
dropout_2 (Dropout)          (None, 51200)             0
_________________________________________________________________
dense_4 (Dense)              (None, 128)               6553728
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 129
=================================================================
Total params: 28,356,641
Trainable params: 28,322,209
Non-trainable params: 34,432
```

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Pneumonia (Class 0) | 0.84 | 0.99 | 0.91 | 390 |
| Normal (Class 1) | 0.99 | 0.68 | 0.81 | 234 |
| accuracy | | | 0.88 | 624 |
| macro avg | 0.91 | 0.84 | 0.86 | 624 |
| weighted avg | 0.90 | 0.88 | 0.87 | 624 |

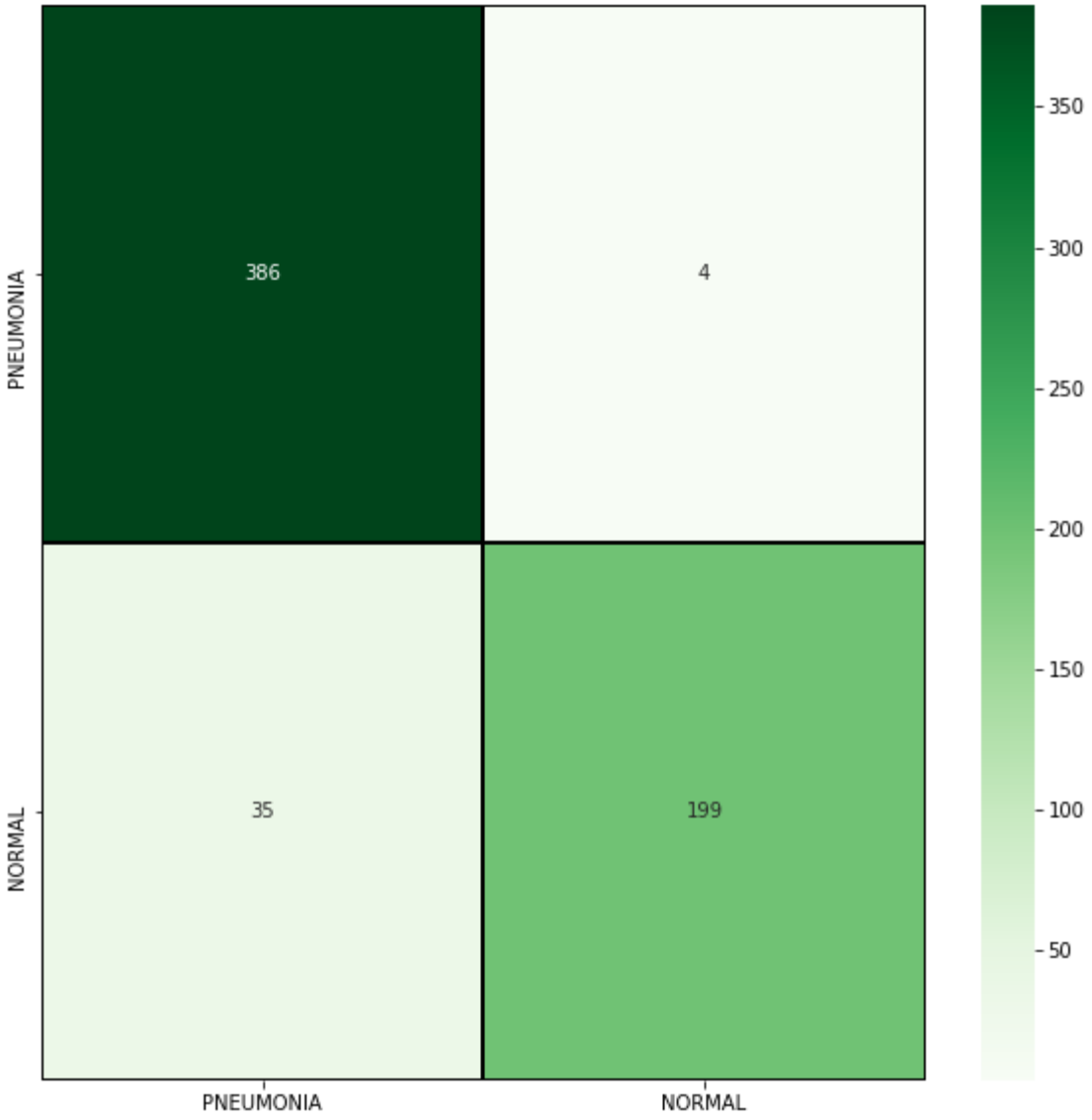
After using the InceptionV3 as base Model accuracy has been reduced to 88%.

We can clearly see that transfer learning has improved the performance and in case of DenseNet because of weights it gave the highest accuracy.

## Weighted Classifier:

Here, weights for each model are estimated and the weighted average is calculated for all the predictions. Weights are made equal for all the four models and the weighted predictions are passed through a weighted classifier to get the final weighted prediction.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Pneumonia (Class 0) | 0.92 | 0.99 | 0.95 | 390 |
| Normal (Class 1) | 0.98 | 0.85 | 0.91 | 234 |
| accuracy | | | 0.94 | 624 |
| macro avg | 0.95 | 0.92 | 0.93 | 624 |
| weighted avg | 0.94 | 0.94 | 0.94 | 624 |




Accuracy of the weighted classifier is 94%.

We can say that when equal weights were assigned to every model, the testing accuracy of the weighted classifier was less than that of DenseNet121. This is because even the models with less testing accuracy were assigned the same weights as that to the models with higher testing accuracy.

## Responsibilities:

| | Responsibility/Member | Haripriya Aravapalli | Avinash Shekar |
|---|---|---|---|
| **Preprocessing** | Data Preparation | Team effort | |
| | EDA | Team effort | |

| | | | |
|---|---|---|---|
| **Model build** | Base model | Team effort | |
| | CNN | | Individual |
| | ResNet18 | | Individual |
| | DenseNet121 | Individual | |
| | InceptionV3 | Individual | |
| **Test** | Test CNN, ResNet18 | Individual | |
| | Test DenseNet121, InceptionV3 | | Individual |
| **Results** | Documentation | Team effort | |
| | Presentation | Team effort | |

Most common tasks/responsibilities are combined team effort. For CNN and transfer learning, we worked on each one individually. Testing is switched up so we get to have another set of eye looking into each model.

## References:


1. WHO Pneumonia is the Leading Cause of Death in Children. [(accessed on 31 December 2019)];2011 Available online: https://www.who.int/maternal_child_adolescent/news_events/news/2011/pneumonia/en.
2. He K., Zhang X., Ren S., Sun J. Deep residual learning for image recognition; Proceedings of the Conference on Computer Vision and Pattern Recognition; Las Vegas, NV, USA. 26 June–1 July 2016; pp. 770–778
3. Huang G., Liu Z., Van Der Maaten L., Weinberger K.Q. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Jul, 2017. Densely connected convolutional networks; pp. 4700–4708.
4. Szegedy C., Vanhoucke V., Ioffe S., Shlens J., Wojna Z. Rethinking the inception architecture for computer vision; Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition; Las Vegas, NV, USA. 26 June–1 July 2016; pp. 2818–2826.
5. https://towardsdatascience.com/keras-transfer-learning-for-beginners-6c9b8b7143e
6. https://www.researchgate.net/figure/The-architecture-of-ResNet50-and-deep-learning-model-flowchart-a-b-Architecture-of_fig1_334767096
7. https://towardsdatascience.com/creating-densenet-121-with-tensorflow-edbc08a956d8#:~:text=DenseNet%20(Dense%20Convolutional%20Network)%20is,shorter%20connections%20between%20the%20layers.
8. https://machinelearningmastery.com/how-to-implement-major-architecture-innovations-for-convolutional-neural-networks/
9. https://towardsdatascience.com/deep-learning-for-detecting-pneumonia-from-x-ray-images-fc9a3d9fdba8