Automatic Pneumonia Detection by observing Lung Opacity from the Chest of X-Ray Images

Final Capstone Project Report

Capstone Project-Group 1

Great Learning

Team Members:

Sanjay Kumar

Manish Kumar

Suman Nandi

Date- 15-01-2023

Mentored by:

Neha Baranwal

Nupoor Singh

Contents

1. Abstract

Analysis and diagnosis has a major importance in the medical Industry. Timely diagnosis plays a very crucial role whenever lives of the patient are concerned. Pneumonia is sometimes fatal lung infection which can be fatal many times so an early and efficient diagnosis with high accuracy is required. Real life method of detecting pneumonia are very trivial so there is need for a system by using deep-learning has been a driving force for making this project. In this paper, we propose different deep convolutional neural network(CNN) architectures to extract features from images of chest X-ray and classify the images to detect presence pneumonia in a person with a higher accuracy. For this research work we have used the real world dataset contributed by the Radiological Society of North America (RSNA). Through this paper we want to show that with the help of deep learning can help us in detecting disease in early stages which makes it ideal for treatment, which was not possible in the early era of diagnosis.

2. Introduction

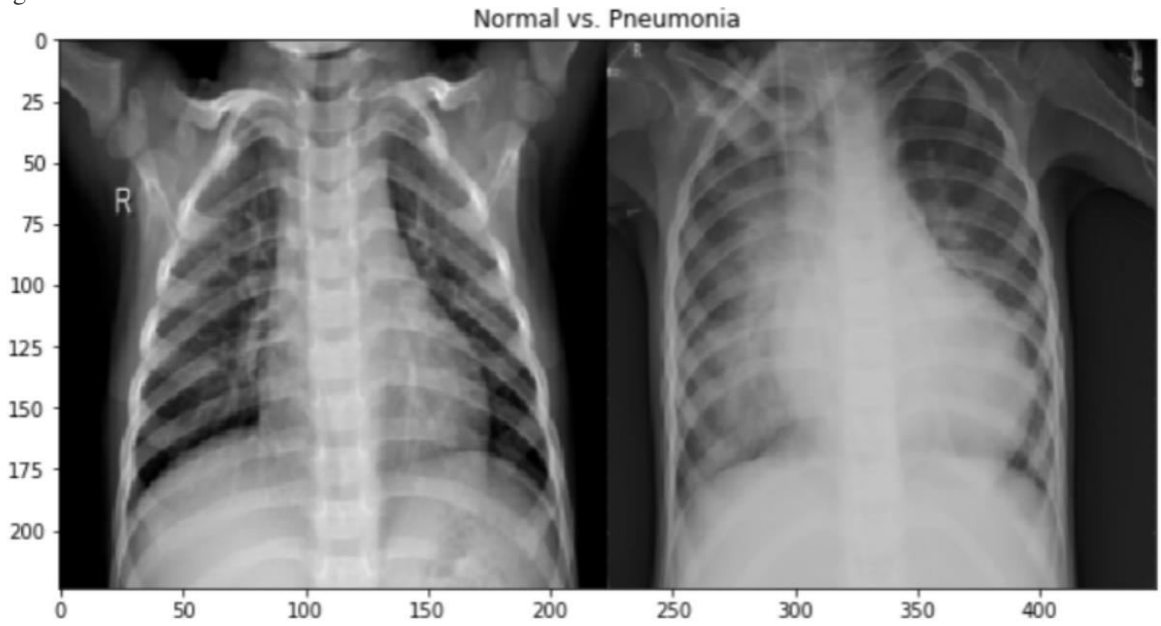
2.1 What is Pneumonia?

Pneumonia is an inflammatory condition of the lung affecting primarily the small air sacs known as alveoli. Typically symptoms include some combination of productive or dry cough, chest pain, fever, and trouble breathing. Severity is variable.

Pneumonia is usually caused by infection with viruses or bacteria and less commonly by other microorganisms, certain medications and conditions such as autoimmune diseases. Diagnosis is often based on the symptoms and physical examination. Chest X-ray, blood tests, and culture of the sputum may help confirm the diagnosis. The disease may be classified by where it was acquired with community, hospital, or health care associated pneumonia.

2.2 What Does a Normal Image Look Like

Fig 1



Pneumonia impacts all the elderly and young people everywhere. With the high growth in the popularity of neural networks, engineers and researchers have been able to find state-of-the-art products for computer vision. Artificial Intelligence helps us to automate analysis techniques, which is only possible now because of the technology of Deep Learning. The exposure to Pneumonia is quite high for many people, mainly in economically underdeveloped and developing countries where the majority are deprived of a nutritious diet. The World Health Organisation states that more than 4 million untimely deaths per year occur from diseases caused by air pollution.

2.3 Chest Radiographs basics

Radiological examination of the lungs using computed tomography (CT), magnetic resonance imaging (MRI), or radiography (X-rays) is frequently used for diagnosis. X-ray imaging constitutes a non-invasive and relatively inexpensive examination of the lungs. Fig 1 shows an example shows an example of a pneumonic and a healthy lung X-ray. The white spots in the pneumonic X-ray (indicated with red arrows), called infiltrates, distinguish a pneumonic from a healthy condition. However, chest X-ray examinations for pneumonia detection are prone to subjective variability [2, 3]. Thus, an automated system for the detection of pneumonia is required. In this study, we developed a computer-aided diagnosis (CAD) system that uses an ensemble of deep transfer learning models for the accurate classification of chest X-ray images.

3. Problem Statement, Data and Findings

3.1 Problem Statement:

Pneumonia is an infection in the lung, which requires review of a chest radiograph by highly trained specialists. Pneumonia shows up in a chest radiograph as an area of opacity. However, diagnosis of it can be complicated and much time and effort is spent by specialists in reviewing them. Chest radiograph is the most common performed diagnostic imaging study. Due to the high volume of chest radiography, it is very time consuming and intensive for the radiologists to review each image manually. As such, an automated solution is ideal to locate the position of inflammation in an image. By having such an automated pneumonia screening system, this can assist physicians to make better clinical decisions or even replace human judgement in this area.

3.2 Objective of the project

The goal of this project is constructing a convolutional neural network (CNN) to identify whether a patient has pneumonia or not by classifying their chest X-ray images. This project is adequately scoped and focuses on one specific type of disease rather than targeting multiple diagnosis. Creating a robust algorithm that provides fast and accurate diagnosis is beneficial for both patients and medical professionals.

3.3 About the given dataset

The dataset is collected from the following website: RSNA Pneumonia Detection Challenge | Kaggle. With almost 54,000 members from 146 countries, the Radiological Association of North America (RSNA) is an international society of radiologists, medical professionals, and other medical physicists.They propose that machine learning can help prioritise and accelerate the assessment of possible pneumonia cases by optimizing initial detection(imaging screening). RSNA shared 30000 plus x-ray images and its diagnoses information along with the bounding box coordinates in two different .csv files for the Kaggle competition. Every x-ray image is in Digital Imaging and Communications in Medicine (DICOM) format which is the standard medical imaging format accepted worldwide.

It is a format that has metadata, as well as pixel data or image data attached to it. Therefore, each image has metadata information like Patient ID, name, age and other image-related data.

The first step would be to examine the data available for this. The data is given in a zip file “rsna-pneumonia-detection-challenge.zip”, which contains the following items:

1. A folder “stage_2_train_images”: This folder contains all the training dataset chest radiograph DICOM images.
2. A csv file “stage_2_train_labels.csv”: This file contains the corresponding patientID images to the folder “stage_2_train_images” and contains the bounding box of areas of pneumonia detected in each image along with a target label of 0 or 1 for pneumonia detected.
3. A csv file “stage_2_detailed_class_info.csv”: This file contains the corresponding patientID images to the folder “stage_2_train_images” and contains the target class labels of the images.

4. A folder “stage_2_test_images”: This folder contains all the test dataset chest radiograph DICOM images. We will not be using this set of images as they do not contain labels.
5. A csv file “stage_2_sample_submission.csv”: This file contains the corresponding patientID images to the folder “stage_2_test_images”. We will not be using this set of file.

The first step is to unzip the zip file to open the above files to the google drive directory. Second is to verify the file format of the images provided, and they are all DICOM images in the “.dcm” file format. After that, the next step would be to inspect the csv files.

3.3.1 Details about Label and Class info

Class Info

Loading the “stage_2_detailed_class_info.csv” file into pandas dataframe, a quick glance reveals that it has only 2 columns:
patientId – which refers to the patientId’s corresponding image name
class – target label of the patientId’s image which can take 1 of 3 values: Normal, Lung Opacity and No Lung Opacity / Not Normal. The primary concern of the project would be to detect images with Lung Opacity, and the others would be in the same group labelling.

There is a total of 30,227 entries and there are no missing values in this file. Checking for duplicates, there appears to be 3,543 duplicate entries. This will be addressed later. Referring to the below image, I can see that there is a slight imbalance in the proportion of class labels, which may be exacerbated when I proceed to reduce to only having 2 labels, which are Lung Opacity and the rest. This needs to be addressed when feeding the data into the model for training to prevent bias in the machine learning algorithm.

Label Info

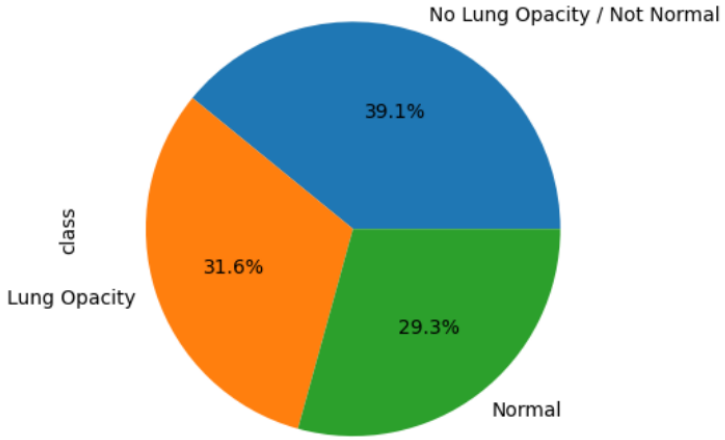
Loading the “stage_2_train_labels.csv” file into pandas dataframe, I can see that it has several fields:

- patientId – which refers to the patientId’s corresponding image name
- x - upper-left x coordinate of the bounding box
- y - upper-left y coordinate of the bounding box
- width – the width of the bounding box
- height – the height of the bounding box
- Target – binary target indicating if this image has evidence of pneumonia

3.4 Finding from given datasets

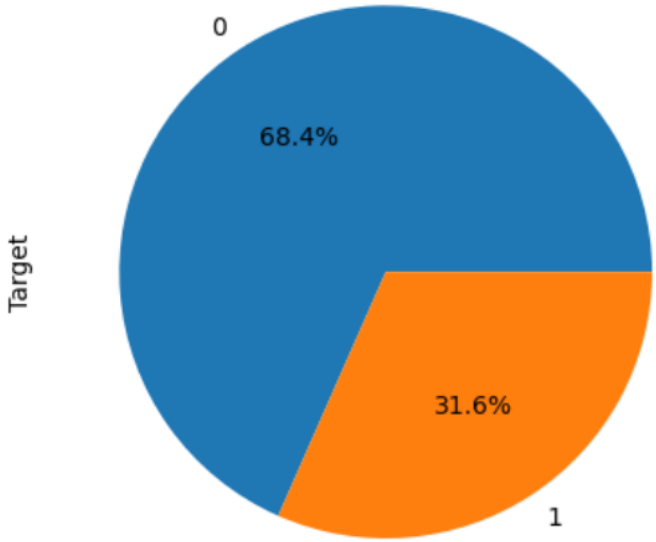
Based on analysis, some of the observations:

- Training data is having a set of patientIds and bounding boxes. Bounding boxes are defined as follows: x, y, width and height.
- There are multiple records for patients. Number of duplicates in patientID = 3,543.
- There is also a binary target column i.e. Target indicating there was evidence of pneumonia or no definitive evidence of pneumonia.
- Class label contains: No Lung Opacity/Not Normal, Normal and Lung Opacity.



Pie chart to show the % of Target label data

- Chest examinations with Target = 1 i.e. ones with evidence of Pneumonia are associated with Lung Opacity class.
- Chest examinations with Target = 0 i.e. those with no definitive evidence of Pneumonia are either of Normal or No Lung Opacity / Not Normal class.



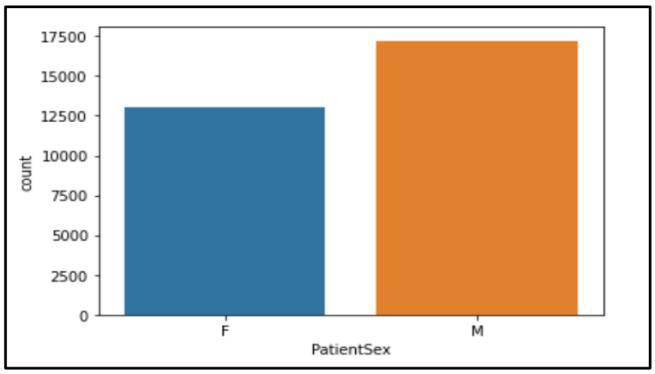
- About 23,286 patientIds (~87% of them) provided have 1 bounding boxes while 13 patients have 4 bounding boxes

```
Let's group by patient IDs and check number of bounding boxes for each unique patient ID
--------------------------------------------------------------------------------
Number of unique patient IDs in the dataset: 26684

Number of patientIDs per bboxes in the dataset
```

number_of_boxes	number_of_patientIDs_per_boxes
1	23286
2	3266
3	119
4	13

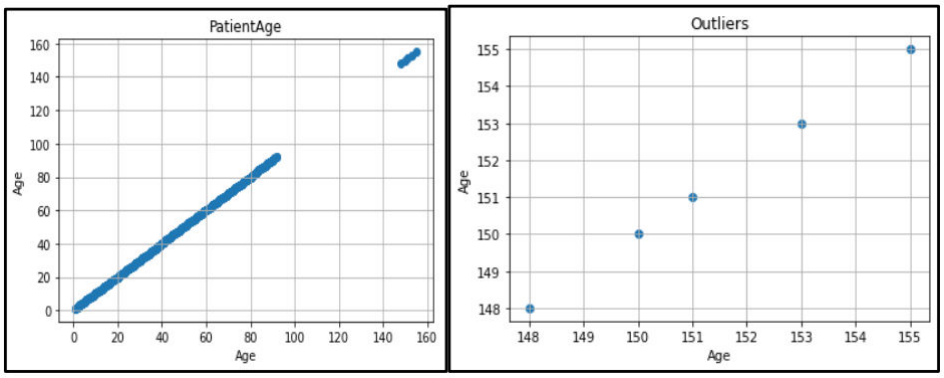
3.4.1 Univariate Analysis

1. Patient Sex



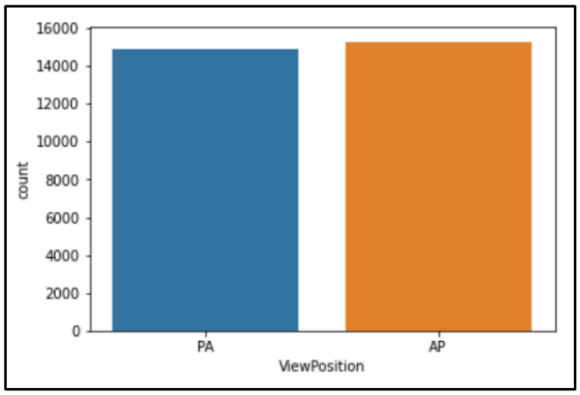
More male patient chest X-rays are present compared to the female counterpart.

2. Patient Age



The age distribution is between 0-100 years except for the outliers which can be due to data entry mistakes. Only 5 outliers' values are present which is above 148 years. Most patients are in the range of 55-58 years whereas maximum patients are 58 years old. The other concentrated range is between 40-54 years.

3. View position

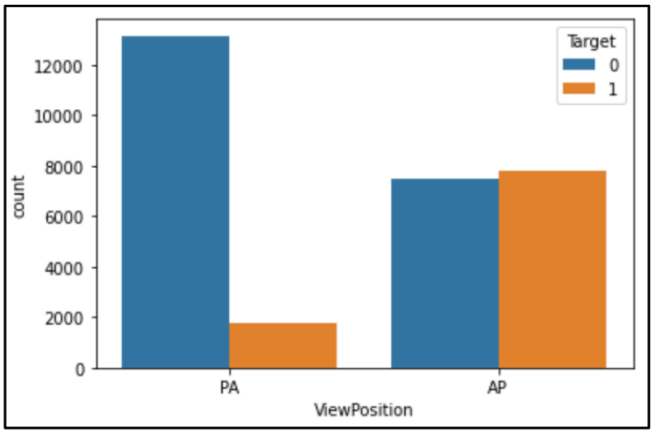


'AP' and 'PA' View Positions doesn't seem to have much of a difference in terms of the amount of data.

3.4.2 Bivariate Analysis

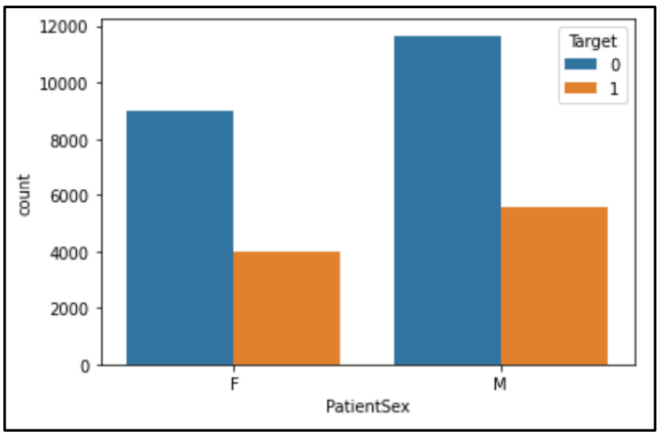
1. View position concerning target

   a. 'PA' view position is highly imbalanced for the 'Target' attribute while the 'AP' view position is balanced. Fact patients should be imaged in an upright PA position, as AP views are less useful and only reserved for very ill patients who are unable to stand erect.
   b. The inference is that patients that are imaged in an AP position are those that are more ill, and therefore more likely to have contracted pneumonia. As the absolute split between AP and PA images is about 50- 50, the above consideration is extremely significant.



'PA' view position is highly imbalanced concerning the 'Target' attribute while the 'AP' view position is balanced.

2. Patient's Sex to Target



The ratio between the Patient's Sex concerning Target seems to be the same between males and females when compared to the number of data present. But what is noticeable is that more male patients are positive with pneumonia as compared to female patients.

4. Overview of the final process

4.1 Problem methodology

4.1.1Convolutional Neural Network

In recent years, the use of deep learning in clinical diagnosis and medical images has increased rapidly; specifically, CNNs can be considered a special type of multi-layer neural network that was built to directly identify visual patterns in pixel images with minimal preprocessing. CNNs have many benefits, such as an ability to extract more significant features from images rather than handcrafted features. Researchers have proposed different CNN-based deep networks for achieving image classification, image segmentation, object detection, and localization in computer vision. Besides solving natural computer vision problems, CNNs have also been very successful and efficient in solving medical problems, such as breast cancer detection, brain tumor segmentation, diagnosing Alzheimer's disease, and classifications of skin lesions. In addition, detailed reviews about deep learning in medical image analysis have been presented. Various CNN models, such as ResNet, AlexNet, LeNet, VGG, MobileNet, Xception, PNASNet etc. and Inception were developed as pre-trained models on millions of images and can be used for image classification using transfer learning. These models have disadvantages—a very large architecture, millions of trainable parameters that require substantial computing power, and high time consumption. Moreover, when the used dataset size is small, these models may overfit the training data, resulting in poor classification accuracy.

We have used transfer learning models VGG16 and MobileNetV2 to improve the accuracy of the Pneumonia classification.

Transfer Learning

Transfer learning consists of taking features learned on one problem, and leveraging them on a new, similar problem.Transfer learning is usually done for tasks where your dataset has too little data to train a full-scale model from scratch.

The most common incarnation of transfer learning in the context of deep learning is the following workflow:

1. Take layers from a previously trained model.
2. Freeze them, so as to avoid destroying any of the information they contain during future training rounds.
3. Add some new, trainable layers on top of the frozen layers. They will learn to turn the old features into predictions on a new dataset.
4. Train the new layers on your dataset.

A last, optional step, is fine-tuning, which consists of unfreezing the entire model you obtained above (or part of it), and re-training it on the new data with a very low learning rate. This can potentially achieve meaningful improvements, by incrementally adapting the pretrained features to the new data.

4.1.2 MobileNetV2

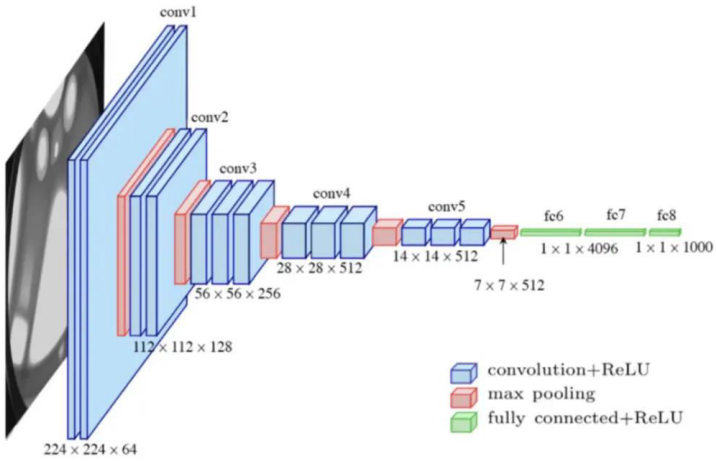
- In MobileNetV2, there are two types of blocks. One is residual block with stride of 1. Another one is block with stride of 2 for downsizing.
- There are 3 layers for both types of blocks.
- This time, the **first layer** is **1×1 convolution with ReLU6.**
- The **second layer** is the **depthwise convolution**.
- The **third layer** is another **1×1 convolution but without any non-linearity.** It is claimed that if ReLU is used again, the deep networks only have the power of a linear classifier on the non-zero volume part of the output domain

Input	Operator	Output
$h \times w \times k$	1x1 conv2d , ReLU6	$h \times w \times (tk)$
$h \times w \times tk$	3x3 dwise s=s, ReLU6	$\frac{h}{s} \times \frac{w}{s} \times (tk)$
$\frac{h}{s} \times \frac{w}{s} \times tk$	linear 1x1 conv2d	$\frac{h}{s} \times \frac{w}{s} \times k'$

- And there is an expansion factor t. And t=6 for all main experiments.
- If the input got 64 channels, the internal output would get 64×t=64×6=384 channels.

4.1.3 VGG16

VGG is a deep convolutional neural network that was proposed by Karen Simonyan and Andrew Zisserman [1]. VGG is an acronym for their group name, Visual Geometry Group, from the Oxford University. The VGG model investigates the depth of layers with a very small convolutional filter size (3 × 3) to deal with large-scale images. The authors released a series of VGG models with different layer lengths, from 11 to 19, which is presented in the following table.



In summary:

- All configurations of VGG have block structures.
- Each VGG block consists of a sequence of convolutional layers which are followed by a max-pooling layer. The same kernel size (3 × 3) is applied over all convolutional layers. Besides, the authors used a padding size of 1 to keep the size of the output after each convolutional layer. A max-pooling of size 2 × 2 with strides of 2 is also applied to halve the resolution after each block
- Each VGG model has two fully connected hidden layers and one fully connected output layer.

VGG16 is composed of 13 convolutional layers, 5 max-pooling layers, and 3 fully connected layers. Therefore, the number of layers having tunable parameters is 16 (13 convolutional layers and 3 fully connected layers). That is the reason why the model name is VGG16. The number of filters in the first block is 64, then this number is doubled in the later blocks until it reaches 512. This model is finished by two fully connected hidden layers and one output layer. The two fully connected layers have the same neuron numbers which are 4096. The output layer consists of 1000 neurons corresponding to the number of categories of the Imagenet dataset. In the next section, we are going to implement this architecture on Keras.

4.2 Data pre processing

Data augmentation

The main point of augmenting data — or more specifically augmenting train data is that we are going to increase the number of data used for training by creating more samples with some sort of randomness on each of them. These randomnesses might include translations, rotations, scaling, shearing, and flips.

Applying these small amounts of variations on the original image does not change its target class but only provides a new perspective of capturing the object in real life. And so, we use it is quite often for building deep learning models.

Keras ImageDataGenerator class provides a quick and easy way to augment your images. It provides a host of different augmentation techniques like standardization, rotation, shifts, flips, brightness change, and many more. .

```
# Create data generator for training data with data augmentation and normalizing all values by 255
image_datagen = ImageDataGenerator(rescale = 1./255,
                                   rotation_range=20,
                                   shear_range = 0.2,
                                   zoom_range = 0.3,
                                   brightness_range=(1.2,1.5),
                                   validation_split=0.2,
                                   horizontal_flip = True)

train_generator = image_datagen.flow_from_directory(img_working_dirname+"/train",
                                                    target_size = (IMG_HEIGHT, IMG_WIDTH),
                                                    batch_size = 32,
                                                    class_mode = 'binary',
                                                    subset='training',shuffle=True)
# Setting testing data generator's source directory
test_generator = image_datagen.flow_from_directory(img_working_dirname+"/train",
                                                   target_size = (IMG_HEIGHT, IMG_WIDTH),
                                                   batch_size = 32,
                                                   class_mode = 'binary',
                                                   subset='validation',shuffle=True)
```

However, the main benefit of using the Keras ImageDataGenerator class is that it is designed to provide real-time data augmentation. Meaning it is generating augmented images on the fly while your model is still in the training stage.

ImageDataGenerator class ensures that the model receives new variations of the images at each epoch. But it only returns the transformed images and does not add it to the original corpus of images. If it was, in fact, the case, then the model would be seeing the original images multiple times which would definitely overfit our model.

Another advantage of ImageDataGenerator is that it requires lower memory usage. This is so because without using this class, we load all the images at once. But on using it, we are loading the images in batches which saves a lot of memory.

5.Step by step walk through the solution

5.1 CNN (Convolutional Neural Network)

Now it's time to actually build the neural network architecture. Let's start with the input layer. So this layer basically takes all the image samples in our X data. Hence we need to ensure that the first layer accepts the exact same shape as the image size. It's worth noting that what we need to define is only (*width, height, channels*), instead of *(samples, width, height, channels)*.

Afterward, this input layer is connected to several convolution-pooling layer pairs before eventually being flattened and connected to dense layers. Notice that all hidden layers in the model are using the ReLU activation function due to the fact that ReLU is faster to compute compared to sigmoid, and thus, the training time required is shorter. Lastly, the last layer to connect, with a sigmoid activation function.

```
#Test Model1
model1 = Sequential()
model1.add(Conv2D(filters = 16,kernel_size=(3,3),activation = 'relu' , input_shape = (IMG_HEIGHT,IMG_WIDTH,3)))

model1.add(Conv2D(filters = 32,kernel_size=(3,3),activation = 'relu'))
model1.add(MaxPool2D(pool_size=(2,2)))

model1.add(Conv2D(filters = 64,kernel_size=(3,3),activation = 'relu'))
model1.add(MaxPool2D(pool_size=(2,2)))

model1.add(Conv2D(filters = 128,kernel_size=(3,3),activation = 'relu'))
model1.add(MaxPool2D(pool_size=(2,2)))

#model.add(Dropout(0.25))

model1.add(Flatten())
model1.add(Dense(units=64,activation='relu'))
#model.add(Dropout(0.25))
model1.add(Dense(units = 1 , activation = 'sigmoid'))
```

after the model being constructed, now we need to compile the neural net using a categorical cross-entropy loss function and Adam optimizer. So this loss function is used since it's just the one that's commonly used

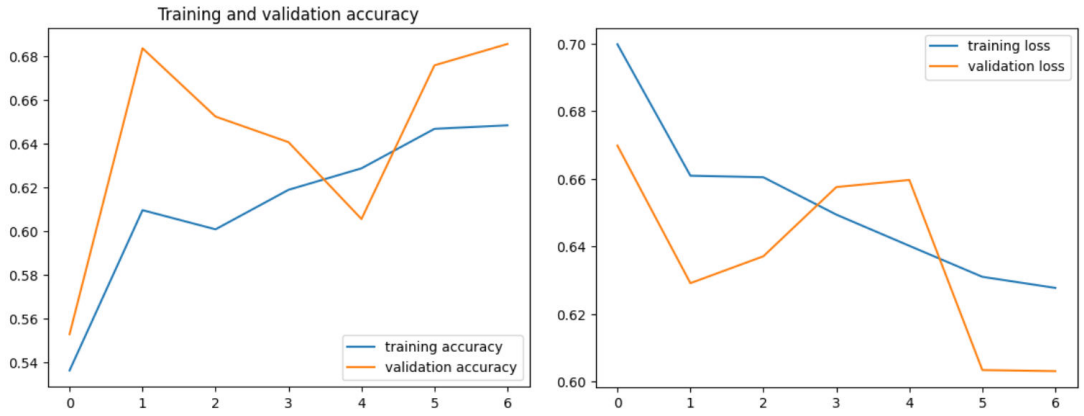
in the multiclass classification task. Meanwhile, I choose Adam as the optimizer since it's just the best one to minimize loss value in most neural network tasks.

```
optimizer = Adam()
model1.compile(optimizer='adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
model1.summary()
```

Now it's time to train the model! Here we are using *fit_generator()* instead of *fit()* because we are going to take the train data from the *train_generator* object.

```
%%time
history1 = model1.fit_generator(generator=train_generator,epochs=30,validation_data=test_generator,validation_steps=16,worker
```

What's so special with *train_generotor* is that the training process is going to be done using samples with some randomness. So all training data that we have in *X_train* is not directly fed into the neural network. Instead, those samples are going to be used as the basis of the generator to generate a new image with some random transformations.



According to the two figures above, we can say that the performance of the model keeps improving, even though both the testing accuracy and loss value look fluctuating within these 30 epochs. Another important thing to notice here is that this model does not suffer from overfitting thanks to the data augmentation method we applied in the earlier part of this project. We can see here that the accuracy on train and test data is 64% at the final iteration.

5.2 MobileNetV2

MobileNet was trained to classify images with 1000 class of images and at the end it have dense layer with 1000 nodes or activation layers and we want to take it off¶

"include_top" is kept to False as we do not want to include dense layers with 1000 nodes and we want to train our model on the top of this pretrained model and should not include 1000 classes to classify in our model "pooling" we have set as "avg" so that allow us to have one vector output

MobileNetV2 is very similar to the original MobileNet, except that it uses inverted residual blocks with bottlenecking features. It has a drastically lower parameter count than the original MobileNet. MobileNets support any input size greater than 32 x 32, with larger image sizes offering better performance.

Here we are instantiating a MobileNetV2 model where the classification layers will depend on the very last layer before the flatten operation. This can be changed by the include_top parameter but generally is not very useful because it does not retain the generality of images when compared to bottom layers.

Next, we freeze the convolutional layers to use the base model as a feature extractor by

mobilenet.trainable = False.

```
mobilenet = tf.keras.applications.MobileNetV2(
    input_shape=(IMG_HEIGHT,IMG_WIDTH,3),
    include_top=False,
    weights="imagenet",
    pooling='avg')
mobilenet.trainable = False #Keeps the weights being trained
```

Now we need to compile and train the model using a categorical cross-entropy loss function and Adam optimizer.

```
EPOCHS=50
BATCH_SIZE=35
model3.compile(
    optimizer='adam',
    loss='binary_crossentropy',
    metrics=['accuracy',tf.keras.metrics.AUC(name='auc')]
)
history3 = model3.fit(
    train_generator,
    validation_data=test_generator,
    batch_size=BATCH_SIZE,
    epochs=EPOCHS,
    callbacks=callbacks3
)
```

5.3 VGG16

While instantiating we will use the weights of the imageNet & include_top=False signifies that we do not want to classify 1000 different categories present in imageNet our problem is all about two categories Pneumonia and Normal that's why we are just dropping the first and last layers then we will just design our own layers and add it into VGG16.

```
#importing the VGG16 library as shown below and add preprocessing layer to the front of the VGG
#We will use imagenet weights. include_top ensure we dont get first and the last layer
vgg = VGG16(input_shape=[IMG_HEIGHT,IMG_WIDTH]+[3],weights='imagenet',include_top=False)
```

After importing VGG16 model, we have to make this important change. By using the for loop iterating over all layers and setting the trainable as False, so that all the layers would not be trained.

```
#ensuring all the layer dont get trained
for layer in vgg.layers:
    layer.trainable = False
```

As we deleted the first and the last columns in the previous step, We will just make a flattened layer and finally we just add our last layer with a sigmoid activation function.

```
vgg16_x = Flatten()(vgg.output)
vgg16_pred = Dense(1,activation="sigmoid")(vgg16_x)
model4 = tf.keras.Model(inputs=vgg.input,outputs=vgg16_pred)
```

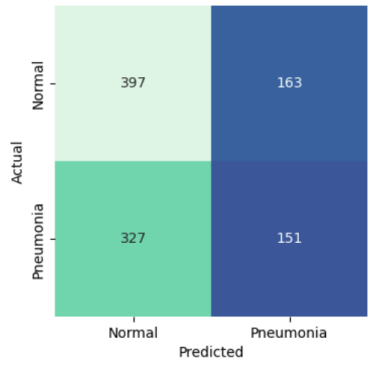
Now we will compile our model using adam optimizer and optimization metric as accuracy.

```
model4.compile(
    optimizer='adam',
    loss='binary_crossentropy',
    metrics=['accuracy',tf.keras.metrics.AUC(name='auc')]
)
history4 = model4.fit_generator(
    train_generator,
    validation_data=test_generator,
    epochs=EPOCHS,
    callbacks=callbacks4
)
```

6. Model evaluation

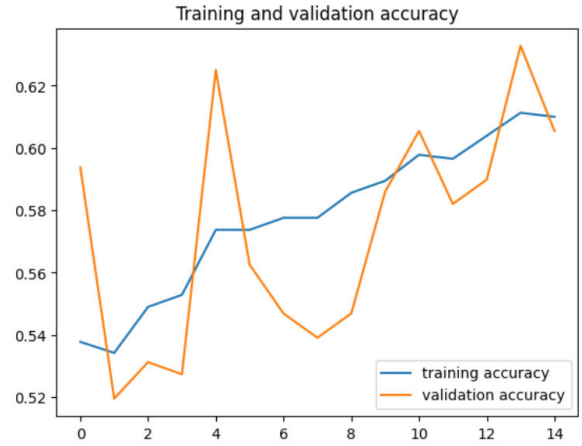
CNN

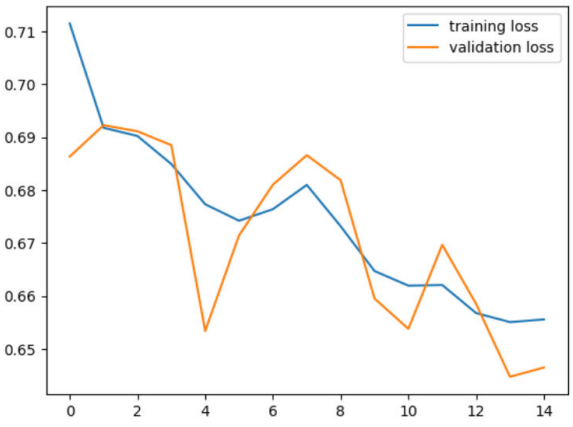
```
Classes:  {'normal': 0, 'pneumonia': 1}

================================================================
```




```
================================================================
Accuracy:       64.56%
AUC:            62.81%
Precision:      48.09%
Recall:         31.59%
```

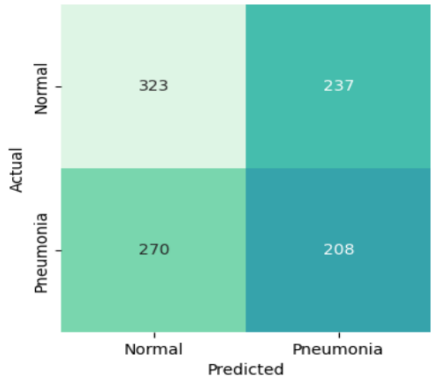
According to the confusion matrix above, we can see that 151 images are classified accurately as Pneumonia, however 397 normal cases are classified as normal. We achieved a accuracy of 64.56%.



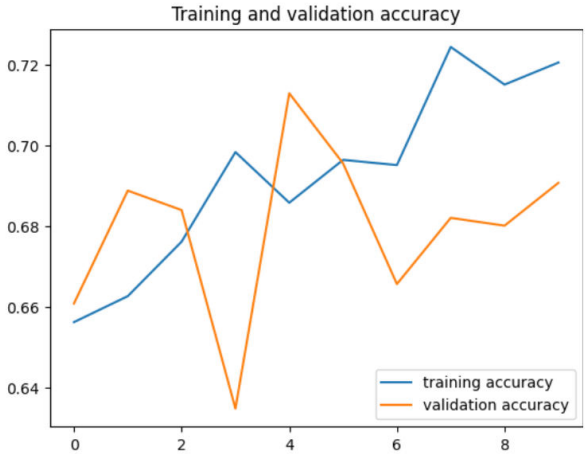


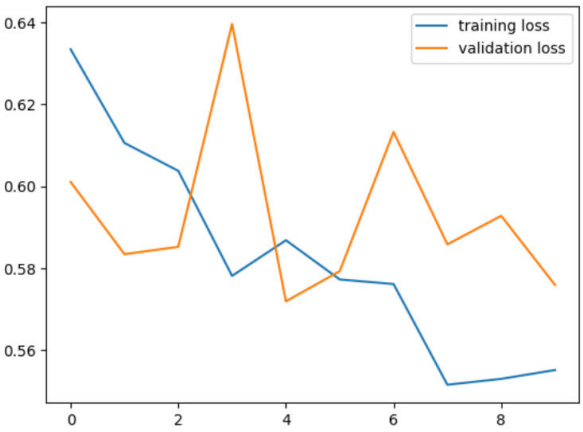
MobileNetV2

```
Classes:  {'normal': 0, 'pneumonia': 1}

================================================================
```



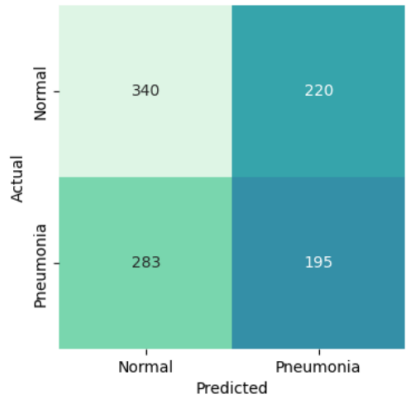

```
================================================================
Accuracy:       58.01%
AUC:            69.46%
Precision:      46.74%
Recall:         43.51%
================================================================
```



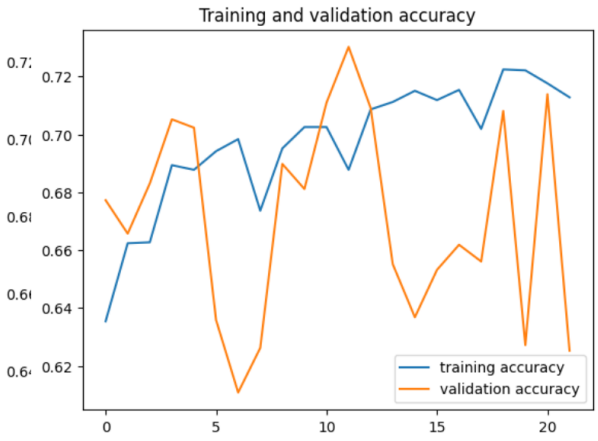


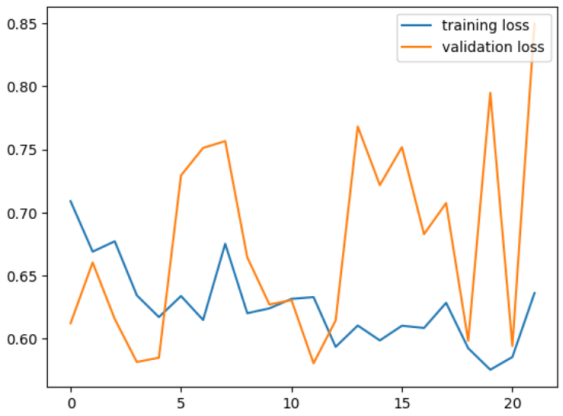
VGG16

```
Classes:  {'normal': 0, 'pneumonia': 1}

================================================================
```




```
================================================================
Accuracy:       61.52%
AUC:            71.00%
Precision:      46.99%
Recall:         40.79%
================================================================
```





7. Comparison to Benchmark

We tried various models to achieve our goal of building a model with minimal computational resources so that faster and accurate results can be brought out in the healthcare industry, especially the imaging side as it is very computationally heavy. This led us to the ResNet152 model which without much hyperparameters and not enough data pre processing gave us results of 64% accuracy which below benchmark. This opens the possibility of running the model for further iterations as the loss still keeps decreasing.

Method	Accuracy	AUC	Preceision	Recall
CNN	64.56%	62.81%	48.09%	31.59%
CNN2	62.61%	66.44%	47.20%	45.91%
MobileNetV2	58.01%	69.46%	46.74%	43.51%
VGG16	61.52%	71.00%	46.99%	40.79%

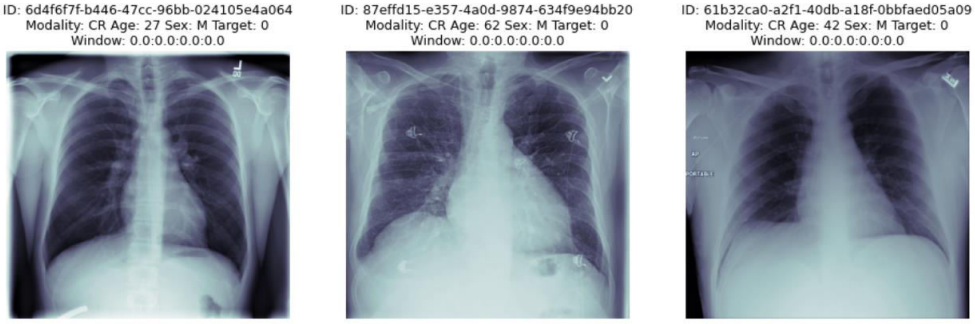
Summary of recent work used in detecting pneumonia using a convolutional neural network (CNN).

Research	Author	Technique	Accuracy	Recall	F1-Score	Precision	AUC
R1	Antin B. et al. [12]	CNN+Adam (DenseNet121)	–	–	–	–	60.9%
R2	Rajpurkar P. et al. [13]	CNN (DensNet121)	–	–	43.5%	–	–
R3	Donthi A. et al. [15]	CNN	78.9%	90.7%	–	–	71.7%
R4	Almubarok A. et al. [16]	(deep ResNet), mask RCNN +Adam + FPN	85.60%	51.52%	–	–	–
R5	Li B. et al. [17]	CNN (RetinaNet, Mask R-CNN)	26.2%	83.5%	–	61.1%	–
R6	Sirazitdinov I. et al. [18]	CNN (RetinaNet, Mask R-CNN) + FPN principle	–	79.3%	77.5%	75.8%	–
R7	Sharma H. et al. [19]	CNN (4 models)	90.68%	–	–	–	–
R8	Rahman T. et al. [1]	AlexNet, ResNet18, DenseNet201, SqueezeNet	98%	99%	98.1%	97%	

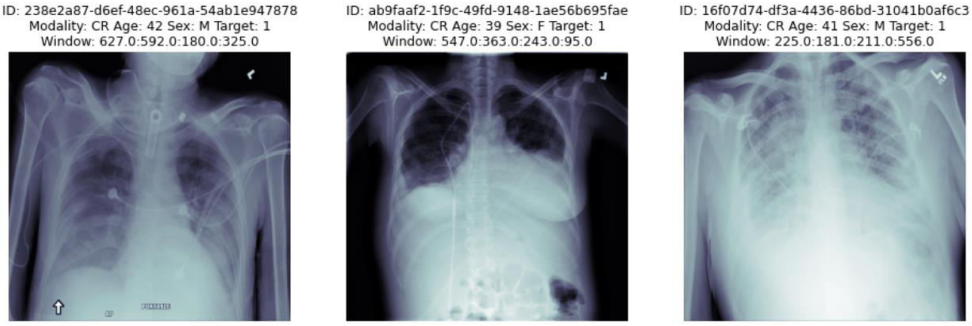
8.Visualizations

In addition to quantifying your model and the solution, please include all relevant visualizations that support the ideas/insights that you gleaned from the data

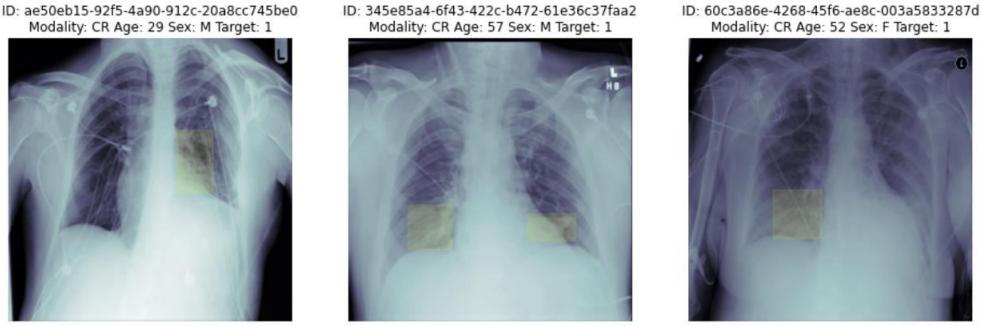
Images with Normal condition



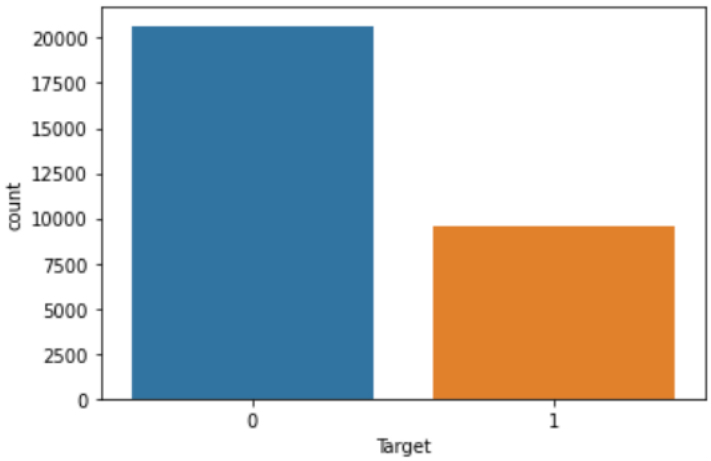
Images with Pneumonia



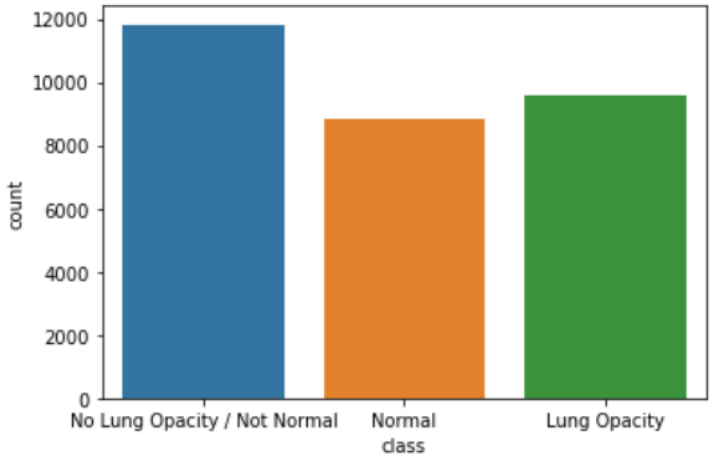
Images with Pneumonia with bounding boxes



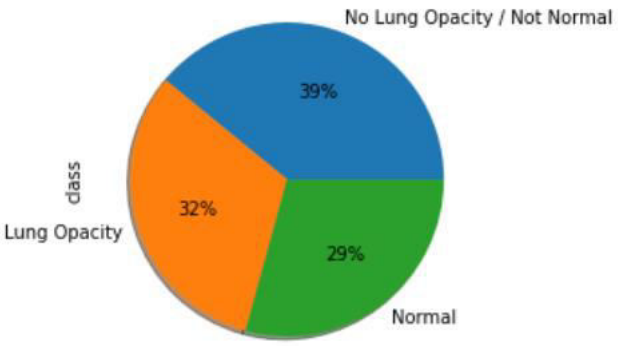
Normal and Pneumonia group segregation



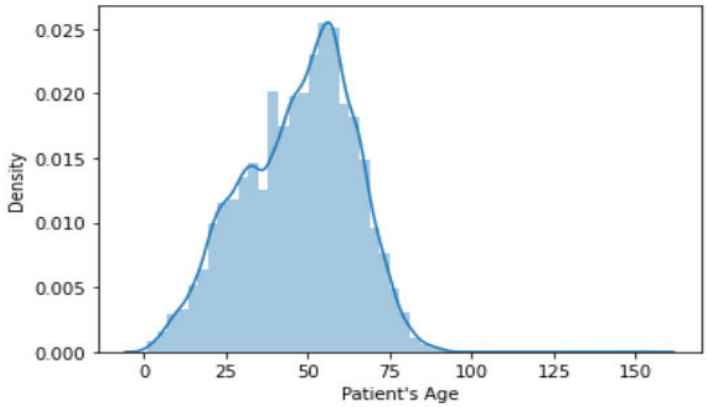
Based on normal, not normal and Pneumonia data



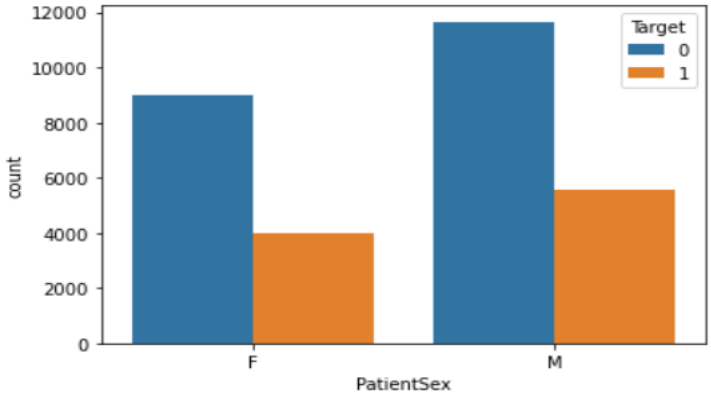
Data in percentage of split



Data present in age wise split



Gender wise data analysis



9.Implications

The medical field is the most sensitive of all the domains ever known, for the simple reason that it deals with humans. This project presents an automated approach used to classify the chest X-Ray into pneumonia and the healthy class using Deep Learning architectures. In the right context, the experiments were conducted using chest X-Ray RSNA dataset, and performances were evaluated using various performance metrics. Likewise, the obtained results show that the CNN, VGG16, MobileNetV2 gave average performance (accuracy is more than 62%) against other architectures cited in this study (accuracy is around 95%). This automated approach can perform binary classification without manual feature extraction with an accuracy of 62%. Due to the high performance achieved by this model, we believe that these results help medical experts to make decisions pertinent. Moreover, this approach may be employed as a supplementary tool in screening Pneumonia patients in emergency medical support services with the rt-PCR machine.

10.Limitations

A limitation of the project is the use of a limited number of our current datasets images. We intend to make our model more robust and accurate by using more such images.

The results generated from the experiments in several numbers epochs. It is concluded that more amount epochs are needed to improve the accuracy level.

Since the model exercises a lot of convolutional layers, the model needs very high computational power otherwise it'll eat up a lot of time in computations. Therefore, one of our future works is to develop deeper collaborative relations with hospitals and clinics to acquire more data.

11.Closing Reflections

For future works, we aim to develop a full system for pneumonia via deep learning detection, segmentation, and classification. In addition, the results may be improved using more datasets that can be added to improve the model performance, more experiments on hyperparameters settings in order to improve the approach, and more sophisticated feature extraction techniques based on deep learning that was developed for the biomedical image. Finally, we plan to improve the result visualization using class mind mapping in order to give a better understanding of the result.

12. References:

[1] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7554804/

[2] https://www.researchgate.net/publication/361505948_Pneumonia_Detection_Using_Deep_Learning_Methods

[3] https://www.kaggle.com/competitions/rsna-pneumonia-detection-challenge/data

[4] https://towardsdatascience.com/review-mobilenetv2-light-weight-model-image-classification-8febb490e61c

[5] https://www.analyticsvidhya.com/blog/2020/08/image-augmentation-on-the-fly-using-keras-imagedatagenerator/

[6] https://keras.io/guides/transfer_learning/

[7] https://www.igi-global.com/gateway/article/full-text-html/299391&riu=true

[8] https://arxiv.org/ftp/arxiv/papers/2204/2204.03618.pdf