UNIVERSITÄT SIEGEN

WBS WM
Wissensbasierte Systeme & Wissensmanagement

# CXR EXPLANATION BASED PNEUMONIA DETECTION

**Studenten Arbeit**


**Faculty IV**

**Department of Electrical Engineering and Computer Science**

**Institute of Knowledge-Based Systems and Knowledge Management**


Submitted on the: XX.XX.XXXX


Submitted by: Gupta, Amit Kumar

Matriculation-Nr.: 1537224

Study program: Mechatronics (Master of Science)

First Examiner: Prof. Dr.-Ing. Madjid Fathi

Second Examiner: M.Sc. Hasan Abu Rasheed

**Erklärung**

Hiermit versichern wir, dass wir die vorliegende Studienarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt haben, insbesondere keine anderen als die angegebenen Informationen aus dem Internet.

Diejenigen Paragraphen der für uns gültigen Prüfungsordnung, welche etwaige Betrugsversuche betreffen, haben wir zur Kenntnis genommen.

Der Speicherung unserer Masterprojektarbeit zum Zwecke der Plagiatsprüfung stimmen wir zu. Wir versichern, dass die elektronische Version mit der gedruckten Version inhaltlich übereinstimmt.

---------- ----------

(Ort, Datum) (Unterschriften)

## Abstract

Explainable AI uses Deep Learning algorithms which can perform the classification or regression tasks and explains the reasons for the classification or regression has been made by the algorithms. Here the CXR EXPLANATION BASED PNEUMONIA DETECTION takes input CXR as images and classifies it whether it is having Pneumonia or it is Normal. On top of the classification of the images, a couple of techniques are used to explain the model output and show the most affected region in the image which are leading to this particular classification.

For classification two pre-trained models of deep learning are used and compared based on the accuracy and AUC score. Then the best model is chosen for the main classification tasks. The ResNet50 and VGG16 are used for classification purposes. The ResNet50 gives the accuracy as **83.04%** and AUC score as **0.9064** whereas the VGG16 model gives accuracy as **75%** and AUC score as **93.34%**. After having a closer look and comparing both the metrics of each, ResNet50 has been chosen for classification.

Since, the classification has been made now two methods were used to explain the classification to the user done by the model. First, a complete masking based algorithm used for computing the importance of the masked region in the classification. Second, a new shallow CNN model is trained on the same dataset and used for calculating the weights of each pixel of the image. These weights have been plotted and it shows the regions which are contributing the most to the classification. The main metrics for this model was AUC score which was **0.9334**.

Im Deutsch

Explainable AI verwendet Deep-Learning-Algorithmen, die die Klassifizierungs- oder Regression Aufgaben ausführen können und die Gründe für die Klassifizierung oder Regression durch die Algorithmen erklärt. Hier nimmt die CXR EXPLANATION BASED PNEUMONIA DETECTION die eingegebene CXR als Bilder und klassifiziert sie, ob sie eine Lungenentzündung hat oder normal ist. Zusätzlich zur Klassifizierung der Bilder werden einige Techniken verwendet, um die Modellausgabe zu erklären und die am stärksten betroffene Region im Bild anzuzeigen, die zu dieser speziellen Klassifizierung führt.

Für die Klassifizierung werden zwei vor trainierte Modelle des Deep Learning verwendet und basierend auf der Genauigkeit und dem AUC-Score verglichen. Dann wird das beste Modell für die Haupt Klassifikation Aufgaben ausgewählt. Zu Klassifizierende Zwecken werden ResNet50 und VGG16 verwendet. Das ResNet50 gibt eine Genauigkeit von **83,04 %** und einen AUC-Wert von **0,9064** an, während das VGG16-Modell eine Genauigkeit von **75 %** und einen AUC-Wert von **93,34 %** angibt. Nach genauerer Betrachtung und Vergleich der beiden Metriken wurde ResNet50 für die Klassifizierung ausgewählt.

Da die Klassifizierung nun vorgenommen wurde, wurden zwei Methoden verwendet, um dem Benutzer die durch das Modell vorgenommene Klassifizierung zu erklären. Zuerst ein vollständig maskierung basierter Algorithmus, der zum Berechnen der Bedeutung des maskierten Bereichs in der Klassifizierung verwendet wird. Zweitens wird ein neues flaches CNN-Modell auf demselben Datensatz trainiert und zum Berechnen der Gewichte jedes Pixels des Bildes verwendet. Diese Gewichte wurden aufgetragen und zeigen die Regionen, die am meisten zur Klassifizierung beitragen. Die Hauptmetrik für dieses Modell war der AUC-Score von **0,9334** .

**Contents**

**Illustration Index**

**Table Index**

# 1 Introduction and Problem Description

In the medical world the use of AI is still limited. The main reason behind this is the lack of trust of doctors on the output of the AI models. This lack of trust rises because of the fact that AI models are generally a black box. One can get the output but can not explain it to the user. Since a mistake in the medical world can lead to huge loss of life and resources. Hence explaining the output of the AI model in medical use cases is extremely important to build trust among the users.

In this work an attempt has been made to explain the classification of the CXR as Pneumonia or Normal and show the user the most affected region/s in the image which is leading to the classification of the CXR images. This will increase the reliability of the model and help the users to understand the classification.

## 1.1 5.1Motivation

AI has been adopted in most of the domains, if not all. But there is one sector where people are skeptical in using AI and that field is Medical domain. The main reason behind this skepticism is the black box nature of the AI algorithms. Then I contacted Mr.Hasan who was working on the Explainable AI use case which was based on NLP. I feel AI has power to do medical imaging and perform the diagnosis on those images and help the doctors in the process. Currently we live in a pandemic and have witnessed the dire need of AI in the preliminary diagnosis of patients. The hospitals were full of patients and doctors did not have the time for new patients. Here if we had the AI for preliminary diagnosis purposes then the workload on the doctors would have dramatically decreased.

But with this comes the problem of interpretability. Apart from the AI engineer or the person with AI know-how can only know and explain the output of the algorithms. But to make people in the medical domain trust AI outputs we have to come up with something which can make the whole system translucent or less opaque if not transparent. With this motivation in mind, I have tried to use AI to detect Pneumonia from the CXR images and explain the output or classification by comparing two approaches.

This work demonstrates the deep learning models used for classification of the CXR images as Pneumonia or Normal. Then a separate deep learning model was trained for extracting weights of each pixel of the CXR image.

## 1.2 Artificial Intelligence

Many definitions have been evolved over the past few decades for AI but John McCarthy says [2]" It is the science and engineering of making intelligent machines, especially intelligent computer programs. It is related to the similar task of using computers to understand human intelligence, but AI does not have to confine itself to methods that are biologically observable." However, for the very first time Artificial Intelligence has been used in the text by A.M Turing in 1950. He asked the question [3][4]"Can machines think?" From there, he offers a test, now famously known as the "Turing Test", where a human interrogator would try to distinguish between a computer and human text response. While this test has undergone much scrutiny since its publication, it remains an important part of the history of AI as well as an ongoing concept within philosophy as it utilizes ideas around linguistics.

Artificial Intelligence(AI) is the broad space of computer science where machines are trained based on data to think and make decisions. The intent is to mimic human beings in decision making. Basically the machines do some computations on the data and based on that data it makes decisions.

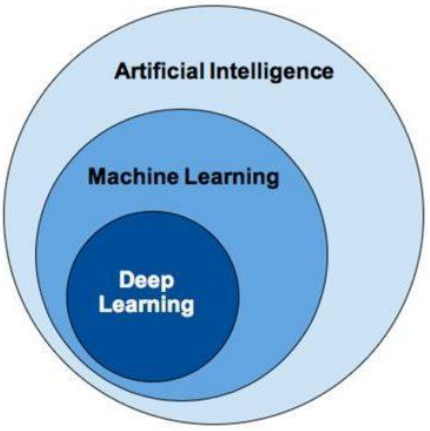



fig.1: AI vs ML vs DL [1]

Artificial Intelligence encompasses Machine Learning and Deep Learning.

## 1.3 Deep Learning

Deep Learning is the subset of Machine Learning which uses various computational layers to learn a set of features on the data. Image classification, in this classification problem various layers are used such as Convolutional layers to learn the distinctive features of the image, MaxPool layers to learn the borders of the images. But how can a deep neural network learn such features? For this task we have an algorithm known as backpropagation. There is a Loss or cost function which has to be minimised and the values corresponding to this minimum loss are the desired values of the features. In fig.2, each layer of the deep neural network has some neuron and each neuron has a set of weights associated with the input. The derivatives are calculated with respect to these weights on the output and then updated this algorithm is known as backpropagation. There is one more important parameter to understand the working of the deep learning models that is optimizers. The optimizers are extremely important for the models to reach to the minima of the loss function. The learning rate of the models can be maneuvered with tuning the optimizers.

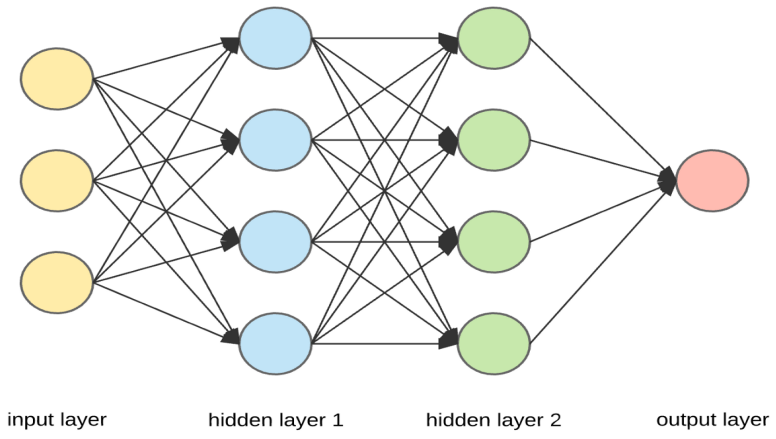



fig.2: Deep learning layers[7]

In fig.2, we can see that for a deep neural network there are different sets of layers categorized as Input layer, Hidden layers and Output layers. The input layer is basically for taking in the input. The hidden layers are the layers where all the computations take place. In this layer only the model learns the features of the data based on the backpropagation on Loss function which is nothing but the output from the output layer.

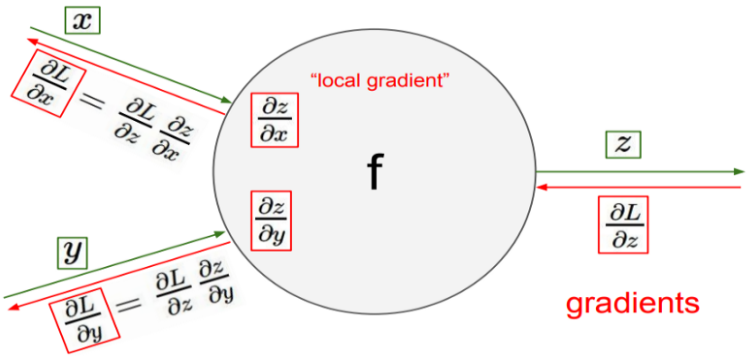



fig.3: Backpropagation[6]

## 1.4 Loss Function

It is also called a cost function. The name itself suggests the functioning of this function. In a deep neural network we have the final output layer, based on the output we get there is a penalty associated with the output. Farther the output value from ground truth larger is the loss or cost. So the main objective of any neural network training is to minimise this cost. Now minimization of the loss function is done by another algorithm known as Gradient Descent. In fig. 3, the neuron cell has a loss function L, the derivative of L with respect to output z is taken and then with the inputs x and y. Taking the derivative of the Loss function makes it possible to reach the minima. So this is the reason why we need a continuous function as our Loss function.

There are many Loss functions, some of them are Cross Entropy loss, Mean Squared Loss, Hinge Loss. Only one condition has to be fulfilled for selecting a loss function and that is the function has to be continuous to be differential. In this work we will be using Cross Entropy Loss.

## 1.5 Cross Entropy Loss

The cross entropy loss is the loss function which is particularly used in the classification problem. Cross entropy loss is also of two types BinaryCrossEntopy and CategoricalCrossEntropy. The former is used for binary classification problems whereas the latter is used for multi-class classification problems. when there is more than 1 class output in the final softmax layer then the problem is said to be a multi-class label problem.

$$y_p = \frac{e^{s_p}}{\sum_{i=1}^{n} e^{s_i}}$$

$$L = -\sum_{i=1}^{n} y_i \cdot \log(y_p)$$

$$Sp = softmax\ layer\ output$$

$$n = total\ classes$$

$$yi = class\ truth\ value$$

## 1.6 Gradient Descent[9]

The gradient descent is the optimization technique in deep learning. The name itself explains the working of this. In this optimization technique we descend towards the minima step by step in the opposite direction of the gradient. The step is known as the learning rate and denoted by η. There are many gradient descent algorithms developed over the years some of them are Stochastic Gradient Descent, Nesterov Gradient, Adagrad, Adadelta, Adam etc. However, for this problem Adam has been used. In fig.4, the comparison of how fast the minima is reached by each algorithm.

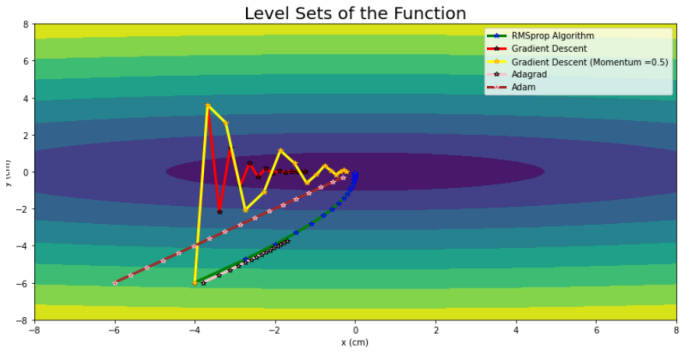



fig.4: Gradient Descent comparison[10]

Adam converges to the minima faster than any other algorithms of gradient descent.

## 1.7 Adam

Adaptive Moment Estimation(Adam) is the momentum based gradient descent algorithm which computes the adaptive learning rate for each parameter. It has been observed that there

has to be a balance between the magnitude of the derivative of the loss with respect to the parameter and the learning rate. If this balance is not maintained then there could be a problem of gradient vanishing and gradient exploding. Gradient vanishing occurs when the values of derivatives are close to zero. On the other hand gradient exploding occurs when the magnitude of gradient is large. So to avoid and control according to the magnitude of the gradient we have a learning rate. If this learning rate is adapted according to the magnitude of the gradient then this is called adaptive learning.

$$mt = \beta_1 \cdot m_{t-1} + (1 - \beta_1) g_t$$

$$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) g_t^2$$

$$\tilde{m}_t = \frac{m_t}{\left(1 - \beta_1^t\right)}$$

$$\tilde{v}_t = \frac{v_t}{\left(1 - \beta_2^t\right)}$$

$$\theta_{t+1} = \theta t - \frac{\eta}{\sqrt{\tilde{v}_t} + \in} \cdot \tilde{m}_t \qquad [9]$$

The $mt$ preserves the history of the gradients and $vt$ preserves the variance of the past gradients. $\theta t$ gives us the weight of the t-the parameter adapted according to the gradient. If the gradient is large the learning rate is a smaller value and vice-versa.

## 1.8 Transfer learning

Transfer Learning is the most powerful technique in deep learning to use the knowledge and hard work of the researchers to build something helpful for the particular purpose. Transfer Learning is like taking the help of the experienced person to learn and do things on our own by building on top of what knowledge sharing has been done from the pre-trained model. In transfer learning the weights of the layers of the pre-trained model are taken and few new layers are added on top of that and trained.

Researchers have spent months on training these models and generously left it open for further research. This not only saves the training time but also makes it possible for us to do research with little computational power.

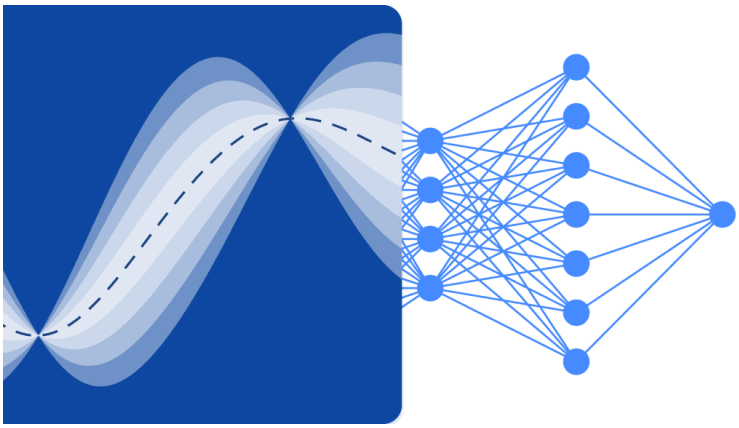


fig.5: Transfer Learning[8]

The blue part in the fig.5 has already been trained and few fully connected dense layers have been stacked upon this trained model. Since the blue half of the fig.5 shares its learnt weights and biases to the next layer, so whole architecture need not to be trained, which is optional, just few fully connected layers are needed to be trained. This saves resources and time.

There are many trained base models to be used, however picking of a model is completely task specific. Since this project is a Computer Vision project, the transfer learning models narrowed down to a few models such as VGG16, VGGNet, VGG19, ResNet50, Xception, AlexNet etc. However for this project two pre-trained models are utilized i.e. VGG16 and ResNet50. The performance of these two models have been compared and finally one best performing model was selected. So let's see what these pre-trained models are architecturally.

### 1.9 VGG16

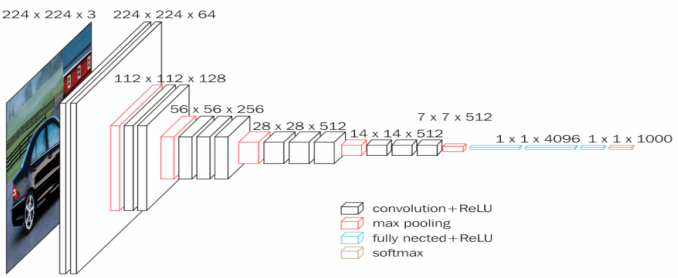



fig.6: VGGNet

This model is trained on the ImageNet dataset which has 1000 classes of images. This model takes input an image of size (224,224,3) where height = 224, width = 224 and channel = 3. Channel = 3 here signifies the image is a color image and it has 3 layers of images stacked upon each other as RGB respectably. The architecture shown in fig.6 is of VGGNet on which is the parent model of various other models such as VGG16, VGG19.

[11] The architecture of the VGG16 model is as follows:

- First layer(Input): In this layer an image of a batch of images can be sent of dimension (batch_size, 224,224,3) here batch size is 1.
- $2nd - 3rd$ layer(conv + ReLU) : These layers are convolutional layers. It has filters of (3,3) dimension and 64 in total. Third layer gives the output image of dimension (128,128,64) the size of the image reduced to half of the input image because of the fact that stride is taken as 2.
- Fourth Layer(MaxPooling2D) : This layer is known as Max Pooling layer. It is the same as a convolutional layer however the difference is that instead of element wise multiplication it picks the maximum value from the kernel when it is superimposed on the image. There 256 filters applied of shape (2,2).
- Fully-connected + ReLU : These are the layers where the flattened input is given. These layers contain 4096 neurons with a ReLU function. The output is $max(0, x)$ where x is the input.
- Final Layer(Softmax) : Since the VGG16 is trained on ImageNet[13] dataset which has 1,000 classes. So Softmax has been used. This gives the values between 0 and 1. The argmax is applied and that index is the predicted class label.

## 1.10 ResNet50 [14]

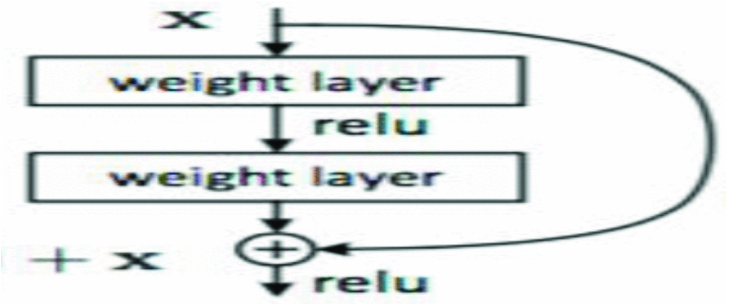


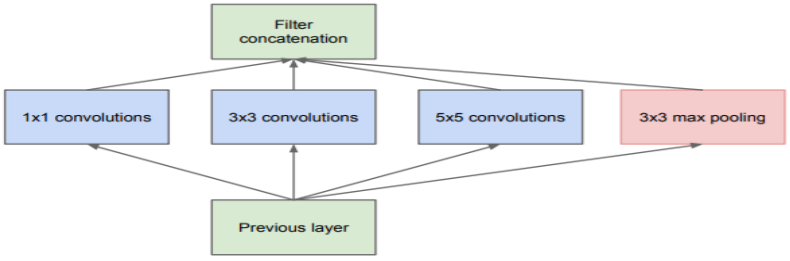
fig.7: ResNet50 skip architecture

ResNet50 is the state of the art pre-trained model for computer vision problems. In VGG16, the layers were sequential. These researchers have found that due to the very deep architecture of the VGG16, the image information was getting lost and there was a problem of gradient vanishing. To solve this gradient vanishing problem Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun researchers at Microsoft Research came up with this better architecture where they were using skip connection after every 2 sequential layers. This solved the problem of gradient vanishing.

The architecture of the ResNet differs from VGG16 in a way that it has several skip connections after every 2 convolutional layers. The input image is of the size (224,224,3). It has a total of 50 layers.

## 1.11 Inception GoogleNet [25]

Also known as GoogLeNet , it is a 22-layer network that won the 2014 ILSVRC Championship. The original intention of the design is to expand the width and depth on its basis . This is designed on the motives derived from improving the performance of the depth of the network generally can increase the size of the network and increase the size of the data set to increase, but at the same time cause the network parameters and easily fit through excessive ,computing resources inefficient and The production of high-quality data sets is an expensive issue. Its design philosophy is to change the full connection to a sparse architecture and try to change it to a sparse architecture inside the convolution. The main idea is to design an inception module, in fig.8, and increase the depth and width of the network by continuously copying these inception modules , but GooLeNet mainly extends these inception modules in depth.




(a) Inception module, naïve version

fig.8 : Inception Module[25]

There are four parallel channels in each inception module , and concat is performed at the end of the channel .

1x1 conv is mainly used to reduce the dimensions in the article to avoid calculation bottlenecks. It also adds additional softmax loss to some branches of the previous network layer to avoid the problem of gradient disappearance.

**Four parallel channels:**

- 1x1 conv: Borrowed from [ Network in Network ], the input feature map can be reduced in dimension and upgraded without too much loss of the input spatial information;
- 1x1conv followed by 3x3 conv: 3x3 conv increases the receptive field of the feature map, and changes the dimension through 1x1 convolution;
- 1x1 conv followed by 5x5 conv: 5x5 conv further increases the receptive field of the feature map, and changes the dimensions through 1x1 conv;
- 3x3 max pooling followed by 1x1 conv: The author believes that although the pooling layer will lose space information, it has been effectively applied in many fields, which proves its effectiveness, so a parallel channel is added, and it is changed by 1x1 conv Its output dimension.

**Two ways to improve network performance:**

The most direct way to improve the performance of deep neural networks is to increase their size . This includes depth, the number of levels, and their width, the size of each level unit .Another easy and safe way is to increase the size of the training data. However, both methods have two disadvantages. Larger models mean more parameters, which makes it easier for the network to overfit , especially when the number of label samples in the training data set is limited.

At the same time, because the production of high-quality training sets is tricky and expensive ,especially when some human experts do it , there is a large error rate . As shown below. Another shortcoming is that uniformly increasing the size of the network will increase the use of computing resources. For example, in a deep network, if two convolutions are

chained, any unified improvement of their convolution kernels will cause demand for resources.

If the increased capacity is inefficient, for example, if most of the weights end with 0 , then a lot of computing resources are wasted. But because the computing resources are always limited, an effective computational distribution always tends to increase the size of the model indiscriminately, and even the main objective goal is to improve the performance of the results.

The basic method to solve these two problems is to finally change the fully connected network to a sparse architecture, even inside the convolution.

The details of the GooLeNet network layer are shown in the following table:

**To sum up:**

- 128 1x1 convolution kernels are used to reduce dimensions and modify linear activation units
- A fully connected layer of 1024 units and a modified linear activation unit;
- A dropout layer that drops neuron connections with a 70% probability;
- A linear layer with softmax loss as classification Predict 1000 categories, but removed during the inference phase, However, for this purpose the model has been tuned on two classes.

**Inception-v2(2015)**

This architecture is a landmark in the development of deep network models . The most prominent contribution is to propose a normalized Batch Normalization layer to unify the output range of the network. It is fixed in a relatively uniform range. If the BN layer is not added, the value range of the network input and output of each layer is greatly different, so the size of the learning rate will be different. The BN layer avoids this situation This accelerates the training of the network and gives the network regular terms to a certain extent , reducing the degree of overfitting of the network. In the subsequent development of network models, most models have more or less added BN layers to the model.

In this paper, the BN layer is standardized before being input to the activation function. At the same time, VGG uses 2 3x3 convs instead of 5x5 convs in the inception module to reduce the amount of parameters and speed up the calculation.

Algorithm advantages:

- **Improved learning rate** : In the BN model, a higher learning rate is used to accelerate training convergence, but it will not cause other effects. Because if the scale of each layer is different, then the learning rate required by each layer is different. The scale of the same layer dimension often also needs different learning rates. Usually, the minimum learning is required to ensure the loss function to decrease, but The BN layer keeps the scale of each layer and dimension consistent, so you can directly use a higher learning rate for optimization.
- **Remove the dropout layer** : The BN layer makes full use of the goals of the dropout layer. Remove the dropout layer from the BN-Inception model, but no overfitting will occur.
- **Decrease the attenuation coefficient of L2 weight** : Although the L2 loss controls the overfitting of the Inception model, the loss of weight has been reduced by five times in the BN-Inception model.
- **Accelerate the decay of the learning rate** : When training the Inception model, we let the learning rate decrease exponentially. Because our network is faster than Inception, we will increase the speed of reducing the learning rate by 6 times.
- **Remove the local response layer** : Although this layer has a certain role, after the BN layer is added, this layer is not necessary.
- **Scramble training samples more thoroughly** : We scramble training samples, which can prevent the same samples from appearing in a mini-batch. This can improve the accuracy of the validation set by 1%, which is the advantage of the BN layer as a regular term. In our method, random selection is more effective when the model sees different samples each time.
- **To reduce image distortion**: Because BN network training is faster and observes each training sample less often, we want the model to see a more realistic image instead of a distorted image

**Inception-ResNet-v2**[26]

The main idea behind the development of Inception network was that, instead of increasing only the depth of the model they have increased the width as well making the model more robust. This idea can solve the need for a deeper model for lesser data.

The major breakthrough in the Inception came when they realized that including skip connections like ResNet's can improve the performance of the model even further. This has been proved in this project too.. Inception-ResNet-V2 has outperformed all the models.

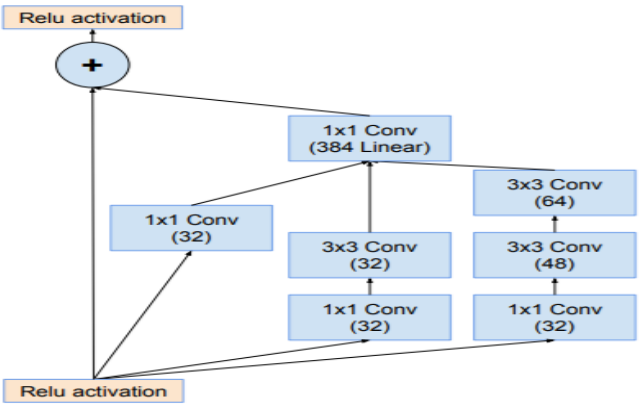



fig.9: Inception-ResNet-Module.[26]

In the fig.9, the module or you can say that it is the building block of the Inception-ResNet-v2 network. The module has 2 parallel CNN layers with 3×3 and 1×1 concolutions. There is another parallel connection with 1×1 convolution, this is the skip connection as 1×1 convolution on any image will give out the image itself after multiplying some weights.

## 1.12 Explainability

The explainability can be interpreted in many ways. Many algorithms and methods have already been developed such as LIME, Gradient Bases explanation, XAI etc. However,

the methods can be customized and problem specific. The explanation can also be made by leveraging the concept and properties of the subject. Here my subject is Pneumonia, normal CXR and Pneumonia CXR can be distinguished by a concept called “silhouette separation” .

### 1.12.1 Silhouette Separation based explanation [15]

In radiology, the **silhouette sign** refers to the loss of normal borders between thoracic structures. It is usually caused by an intrathoracic radiopaque mass that touches the border of the heart or aorta.

fig.10: Normal CXR


fig.11: Pneumonia CXR

The X-Ray is a powerful electromagnetic ray that passes through the body and produces reflection of parts it is passing through. Based on the density of the body parts it produces color gradients in grayscale. if X-Ray passes completely then there it produces black reflection. The denser the part, the less greyish the reflection. Since bones are the most dense material in our chest area, we get clear white reflection. This color gradient separates the different parts in the X-Ray image.

In fig.10, this is the CXR for a normal chest. Here one can clearly see the borders and separations of different parts in the chest. The ribs clearly separates itself from the lung tissues which are white and grey color respectively. This is nothing but a silhouette separation.

In fig.11 this is the CXR for a person having Pneumonia. The borders are not clearly visible and everything kind of mixes into one color around the central part of the chest. This is how silhouette separation can help users understand the classification of the CXR.

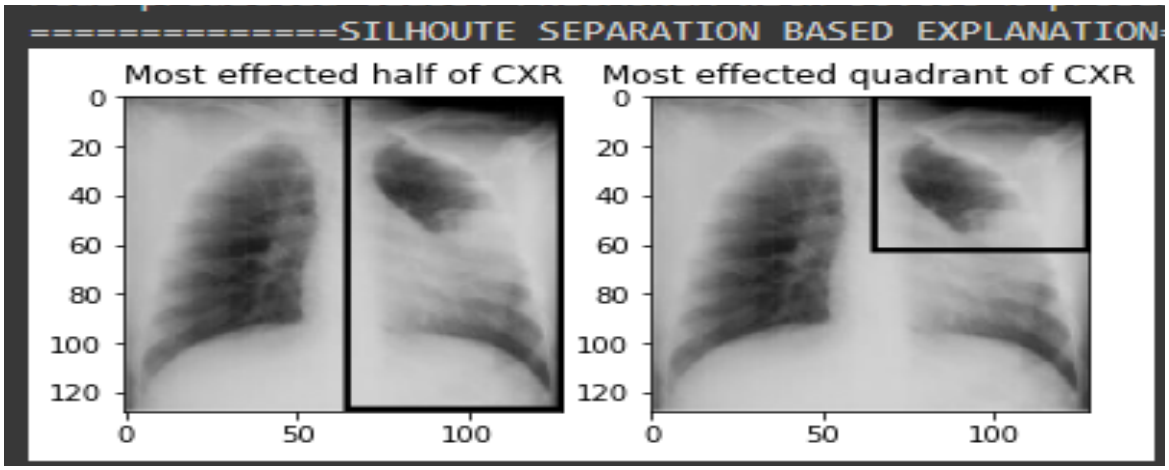



fig.12: Silhouette separation based explanation

### 1.12.2 Gradient Based Explanation [16]

The backpropagation algorithm is at the core of the Gradient Based Explanation. In fig.12, for one neuron cell the backpropagation algorithm is shown. There are two parameters x and y on which this cell is trying to learn something. It is being penalized if the output of the cell goes far from the reality. The farther the output the more is the penalty z. This is just the feedforward method. Now there are weights assigned to each x and y which the cell is trying to learn. The partial gradient w.r.t each input is computed and subtracted after multiplying a learning rate coefficient to the gradients. These new weights replace the older weights.

$$W^1{}_x = W^1{}_x - \eta . \left( \frac{\partial z}{\partial x} \right)$$

$$W^1{}_y = W^1{}_y - \eta . \left( \frac{\partial z}{\partial y} \right)$$

How are these weights helpful in explaining the output? $W_x$ shows how important is x for the output and the same for $W_y$. However, we are dealing with image classification tasks and CNN models are used so instead of neuron cells we have something called kernels.

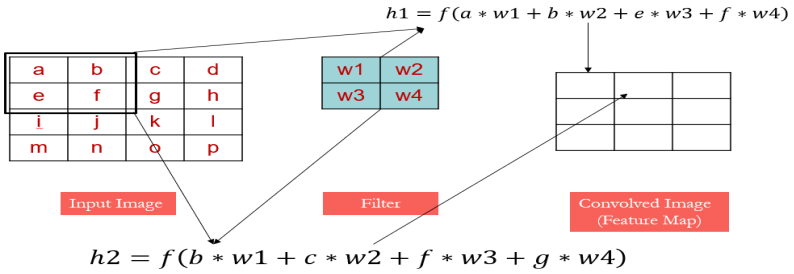



fig.13: Convolutional Operation

In fig.13, the input image has 4 by 4 dimension and has pixel values as a, b, c, d, e, f, etc. Then there is a kernel of size 2, this contains weights associated to each of the pixels which would be learnt by backpropagation. The kernel moves around the image by shifting a number of pixels, this number of pixels which are shifted by the kernel is called strides. The convoluted image is the result of element wise multiplication of the kernel weights and the corresponding pixels of the image.

How is this helpful for the project? For this project the size of image if $128 \times 128 \times 3$, for this 128 number of kernels of size 1 and stride 1 needed to represent the importance of each pixel in 2-D image representation. A CNN model just to get this kernel weights trained has been trained with few dense layers on top to avoid underfitting. After training the weights with 128 kernel extracted which gives total weights as 128*128. These weights are a representation of each pixel in CXR and shows how important each pixel is. These real numbers weights are now converted into images and we can clearly see the most important pixel popping out with the brightest color.

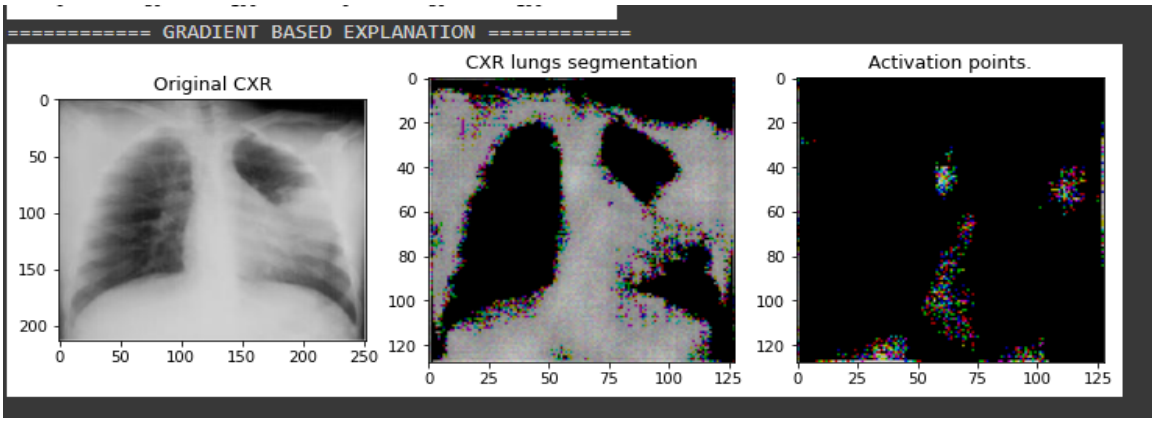



fig.14: Gradient plotting

## 1.13 PROCESSES FOLLOWED

Everything has been done using some very important Python packages and frameworks. It starts with importing the packages. The next step is the data ingestion process. The data ingestion is done using the Keras API and a class in keras known as ImageDataGenerator. After this we need the model to train. The transfer learning is used here and on top of the base model we have few convolutional and dense layers. Training the model for several epochs we stop when the model converges to the minima. Model testing is done after the model training with new unseen data.

### 1.13.1 Importing of necessary packages

- [17]**Pandas** : This library is mostly used for handling the textual data. Reading the textual data and analysis can be easily one with this. In this work, Pandas is not used much but to present the comparison among the different models and present them into the tables.
- [18]**Numpy** : The most important library in Python for Machine Learning and Deep Learning is NumPy. We handle the data in the form of matrices when we are working with a Machine Learning or Deep Learning problem. To handle the matrices and different operations on them can be easily and efficiently done using NumPy. However, operations on matrices can be performed using Python as well but what makes NumPy standout is the efficiency and speed of execution is significantly better. Here in this work NumPy has been used intensively. The main usage of NumPy in this project is in the masking of the images.
- [19]**Tensorflow** : It is a framework for designing machine learning algorithms and implementation of those. Tensorflow allows us to create deep learning architecture seamlessly. It provides easy access to the bunch of operations on tensors. From model creation to training everything has been done using tensorflow. However, keras is used on top of this.
- [20]**Keras :** Keras is a high level interface for Deep learning. Each and every layer can be called with a single line of code. The model architecture implementation is done using Keras.

- [21]**pickle :** This library in Python is used for saving and loading the file. This comes handy when we are handling large data and we do not want to do everything from scratch.
- [22]**tqdm** : This is used to monitor the progress of a loop.

**1.13.2 Data Ingestion and preprocessing** : Data ingestion is the process of data reading and loading onto the server for further process. Data can be in textual form as well as in visual form. Visual data comprises video and static images. For our purpose the static chest X-Rays are used. The image data is loaded using the ImageDataGenerator from tensorflow from the local system to Google Colab server.

Google Colab[23] is the cloud based alternative for Jupyter notebook developed by Google research. Along with all the functionalities of the Jupyter notebook it also offers some awesome features such as GPU access free of cost. This is the boon for the students to study and explore and implement heavy computational codes.

Data preprocessing in the case image dataset comprises image resizing, grayscale to rgb or vice versa as per the problem requirements. There is one special requirement for this problem statement. To analyze the affected region in the image masking has been done on each image. This is done under the method of silhouette separation.

**1.13.3 Data Augmentation :** Data augmentation is the process of diversifying the data with realistics transformations on the data. The realistic transformation consists of various processes such as rotation, each image is rotated by certain degrees based on the argument given by the user. The other most common process is shifting, in this each image is shifted right and left. Vertical and horizontal flip are the processes to rotate the images by 180 degrees horizontally and vertically respectively. The image is rescaled or normalized by stating rescale as 1/255.. One should notice that there is a dot after 255, this dot is given to make every pixel values as float.

The instance of the class [24]ImgaeDataGenerator is defined with all the attributes. The augmentation we need has to be true. Then there is a flow_from_directory method which

loads the images from the local system. The images are resized according to our need by stating the attribute target_size. For this project the target_size is kept as (128,128).

ImageDataGenerator returns the data loaded from the source in batches. In my case the batch was specified as 32 so the returned shape for one batch of the data was (32,128,128,3) and labeled as (32,2). I would like to take a moment to explain the labeled outcome. Since in fig.15 we can see that there are label folders in the data directory. However the folders are made for a particular class label. There are two subfolders Normal and Pneumonia in the test and train directory. So ImageDataGenerator assumes each subfolder as a class label.

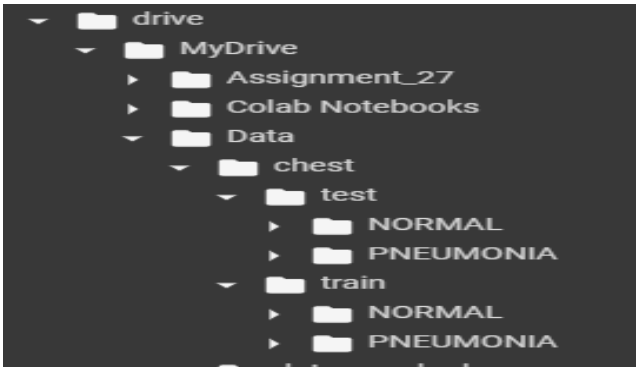



fig.15: Data Directory

**1.13.2 Model Design and Exploration :** The dataset is pre processed and ready to be fed into the model. The design of the model takes a lot of experimentation with the different layers. However, layers such as Conv2D, MaxPolling2D, Dropout, BatchNormalization, Flatten, ReLU and Dense are used to design the model. The parameters of each layer are hyperparameters and the correct value has to be found out using cross-validation. In the cross-validation method we check and experiment with some values of parameters and based on the metrics the best parameters are selected.

**Conv2D**: This is the Convolutional layer which takes in 2D/3D tensors as input. The main parameters for this layer are kernel_numbers(filters), kernel_size and strides.

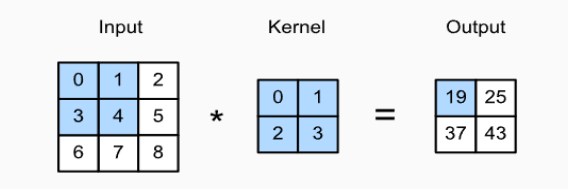



fig.16: Conv2D layer [17]

In a CNN layer, each step covers the element-wise multiplication of the kernel and the corresponding image matrix of the same shape. Convolution returns the sum of element-wise multiplication of these two matrices. In the next step the kernel jumps strides number of columns and does the same process as before. This stride number of rows jumping is done till the kernel has reached to the end of the image. Once the columns are covered then the kernel jumps strides number of rows and by repeating these steps the complete image is covered and convolution is done.

$$\sum_{j=0}^{n} (kernel * image), here\ j = j + s$$

This equation above gives us the element for each convolution and the resultant shape of the image can be computed by the following equation.

$$Output = ((n - k) / s) + 1$$

$$n \times n : shape\ of\ input$$

$$k \times k : shape\ of\ kernel$$

$$s : stride$$

**Filters:** These are the 2D matrix of numbers whose main task is to learn the features in an image matrix. The kernels in initial layers of a CNN network learn the basic features like horizontal and vertical boundaries.

**Stride:** Strides are the number of steps taken by the kernel to move after each convolution.

**MaxPooling2D:** This layer picks the maximum value from sub parts of the image. Main parameter for this is pool_size. This layer picks the maximum element from the given pool_size of the matrix.

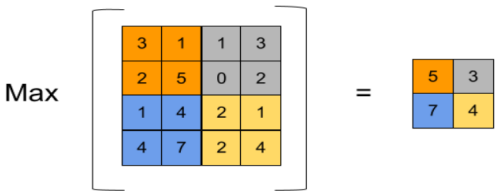



fig.17: MaxPolling2D layer

This layer is the same as that of the Convolutional layer but there is one major difference. In convolutional layer kernel and input image are multiplied element-wise but here in MaxPooling layer a unity matrix kernel is convoluted with the image and instead of summing the elements the maximum element is pooled out as is done in the fig 17. A (2,2) unity kernel is chosen here and convoluted with the orange part of the input which gives the result as 5 which is the maximum element in the orange submatrix.

The MaxPooling layer is basically used for making the CNN layer's kernel to learn more relevant features by picking up the most dominant pixel in the kernel sized image. The other main function of this layer is to downsize the image, if stride is 2 the image is downsized by half of previous size.

**Dropout** : This layer is a regularization layer for deep learning models. This introduces some randomization into the training of the model. In this layer a percentage of neurons deactivated, this percent is the hyperparameter.

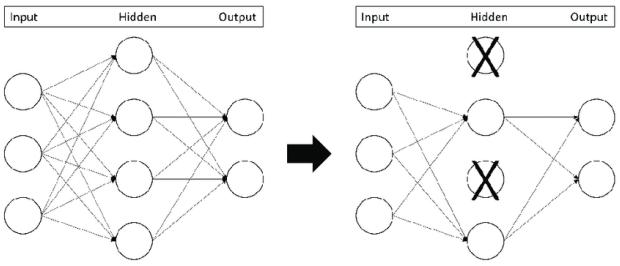



fig.18: Dropout

In fig.18, the marked neurons are deactivated and the rest of them are kept as it is. Here the dropout rate is 0.5 meaning 50% of the total neurons are deactivated.

**ReLU :** This is the activation function and sometimes used as a separate layer to return all positive values. The purpose behind using this is that the image pixel values have to be kept between 0 and 1.

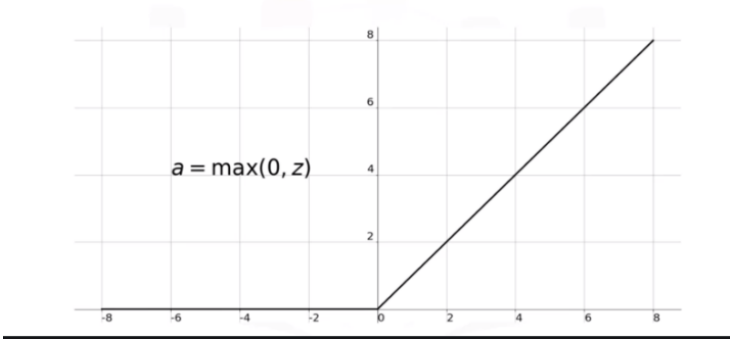



fig.19: ReLU function

**Dense :** These are the fully connected layers, basically they are used as the last layer in a network. The main parameters of this layer are neuron_size and activation_function.

$$output = ActivationFunction(dot(input, kernel) + bias)$$

### 1.13.3 DATA PREPARATION

After creating the model and including all the layers wanted now it's time to load the data and preprocess it. The image data is loaded with the instance of ImageDataGenerator class from Keras and loaded into the Google colab server from the local folder.

### 1.13.4 MODEL TRAINING

The Neural Network and dataset is ready, it's the time to proceed towards training the model. Tensorboard is used to plot the metrics while training. The EarlyStopping and ModelCheckpoint used to stop the training when there is no significant improvement in the performance and to save the model training parameters at each epoch respectively. fit() method is called to start the training. training data, test data , number of epochs to train the model and batch size are the main attributes for the fit() method. Three models have been trained and every model has been trained for 50 epochs and batch size is taken as 32. Before training the model the model must be compiled and loss function should be decided. The loss function is useful because based on the loss function only backpropagation will take place. Choosing the loss function is critical, for classification problems we choose cross entropy loss.If the dataset is multilabel dataset then CategoricalCrossEntropy loss should be chosen.

Optimizers are another very important parameter to be selected before training of the model. Optimizers help the model to converge to global minima. This has a hyperparameter called learning rate(lr), this hyperparameter decides how fast the training converges depending on the learning rate specified. In this case Adam optimizer is used.

### 1.13.5 METRICS

There has to be something to observe the performance of the model. To track the performance of model training different metrics are used. The choice of metrics is very problem specific. If the problem is a balanced binary class classification problem then accuracy is the potential option, However, it can be clubbed with other metrics such as auc score too. **In this project accuracy and auc score is clubbed together.**

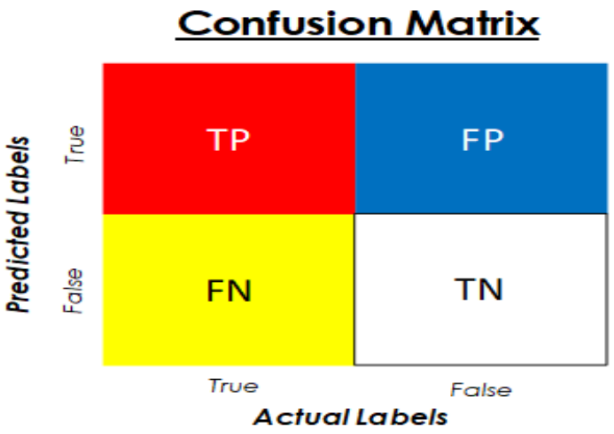



fig.20 : confusion Matrix

The AUC score is the area under the curve of the plotting of the FPR (False positive rate) and TPR (True positive rate) where each of the mentioned terms can be described as below.

$$TPR = TP / (TP + FN), FPR = FP / (FP + TN)$$

$$Accuracy = (TP + TN) / (TP + TN + FP + FN)$$

$$precision = TP / (TP + FP), Recall = TP / (TP + FN)$$

Accuracy is a reliable metric when the dataset is balanced among classes. However, if the classes are not balanced then it is not all advisable to use accuracy as performance metrics for the model. For unbalanced dataset accuracy will be misleading and in most of the cases the trained model is having a bias towards the majority class. For example let's consider a

dummy dataset with 100 data points, 95 data points belong to one category and the 5 to the other one. To achieve a 95% accuracy on training dataset there is no need of training any model, for every query data point if we simply return the former class as our classification we can achieve an accuracy of 95% which at glance looks great but in reality it is misleading. Then to solve this problem we use a confusion matrix and AUC score. In this project we are dealing with a dataset which has an unbalanced distribution of classes. Minority classes share the space with 34 % of the overall data. This verifies the using accuracy here as the metric is nor advisable and use AUC is kind of compulsory here.

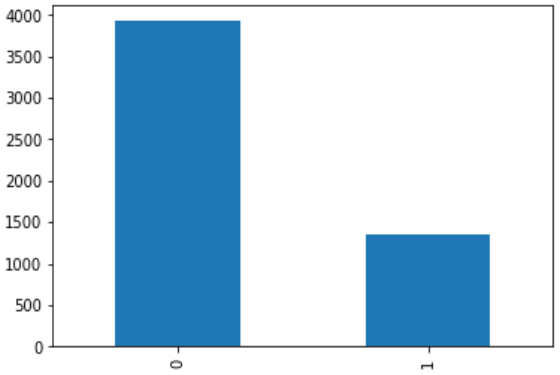



fig. 21 : Class distribution

## 1.14 Research Question

The problem statement is to do classification followed by explanation of the results. There could be many research question based on the performance and model we choose some of them are as followed:

1. Why is the inceptionV2 model chosen even though the InceptionResnetV3 has the best performance ?
2. What could be the reasons behind poor performance of the model.What measures could be taken to improve upon them ?
3. Why is AUC score chosen over Accuracy score ?
4. What is the pre-developed model/work on the topic ?

## 1.15 Structure of the Work

The work is started with a search of dataset, as dataset is the fuel for any AI problem. Finding a suitable dataset is the most important part of the solution. After doing much brainstorming, finally managed to find one on Kaggle with 5895 total CXR images.

Once the dataset is available, model exploration is the next part in the work where experimentation with below hyperparameters were done:

- Best base models to compare (VGG16 or ResNet50 or InceptionResnetV3)
- Input size
- Total trainable parameters..
- Kernel size and filters in each Conv2D layers
- Dropout size
- Number of dense layers
- Units of neurons in dense layers

After designing the model architecture, compiling the model has to be done. While compiling the model a couple of things have to be taken care of such as:

- Optimizer (Adam in this case)
- Learning rate

Model compile has been done now, after this training is carried out. For training the model there are various parameters to be chosen after experimentation with different values of them and choosing the best parameters which gives the model the best performance.

- Number of epochs
- Batch size
- Validation split

The performance of VGG16 model, ResNet50 model and InceptionResnetV3 model architectures are compared and the model architecture with best performance is selected for further processes. When the model has been trained and the best model has been picked the next step would be model inference. Where the trained model is tested with a fresh set of images.

Now the time has come for the results explanation. There are two methods of result explanation, SILHOUETTE SEPARATION based explanation and GRADIENT based explanation. There can be a disagreement between these two explanations but the user can get the overall idea of the damaged lung part of the user.

Finally, UI design for demonstration. After training the model and when once the model is ready for use a simple UI is developed using PyCharm IDE as the development environment. For UI development HTML/CSS is used for web page building. Flask is used for serving the API of the project. Flask is the lightweight framework for python for building software APIs. For communicating with the HTML tags Jinja2 template is used.

# 2. State of the Art

**IMAGE SEGMENTATION**

Image segmentation is the state of the art in this field which shows the exact region of the chest X-Ray which is having Pneumonia. It is a powerful algorithm which can segment the group of pixels in an image based on the requirements. Image segmentation can be subdivided into two major categories which are Semantics segmentation and Instance segmentation. In instance segmentation if many objects belonging to a similar class are there then each object would be in different color whereas in semantic segmentation each class would be shown in one single color irrespective of the number of instances of that class in the image.

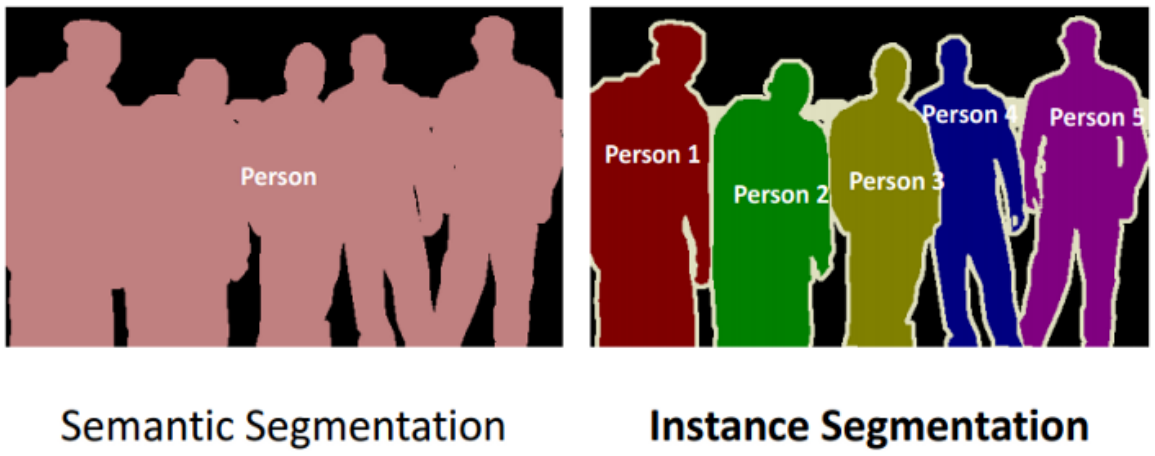



fig.22 : Instance vs semantic

In the fig 22, the clear difference between the instance and semantic segmentation is shown.

Now the question is how this idea can be implemented in this case of classification followed by explanation. Well, if the well labelled data is fed into the models of image segmentation such as Detectron2 by Facebook AI Research(FAIR) then a very precise segmentation can be achieved. Below in the fig. 22, the image shown is the segmented image on the similar dataset but with labelling. Just 164 images were labelled and Detectron2 trained on it. The results were quite promising and this is the state of the art kind of explanation which aimed to be achieved by avoiding some cumbersome manual work such as image labelling.

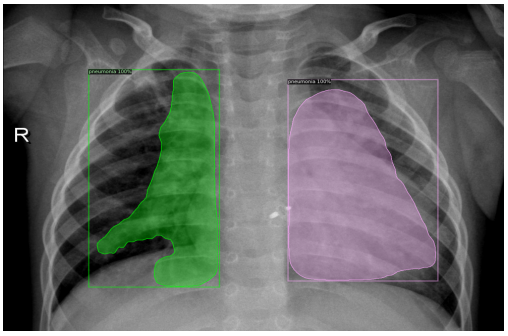



fig. 23 : Segmenting the affected region

One can clearly notice how perfectly the affected region in the chest X-Ray has been segmented. This kind of explanation, if achieved, would be an ideal result for this project.

# 3. Pneumonia Classification Explanation

### 3.1 DATASET

In the last segment it is discussed that in the state of the art method of explanation the image segmentation model is trained on the image data but one has to go through the pain of labelling the data manually. For a smaller dataset this might not sound as cumbersome as it is but in reality the dataset size is normally large. So manually labelling a large image dataset is a lot of work. In this project we are not using any manual labelling of the data. The biggest and most crucial step in any AI project is the data collection. There is a saying that "you are what you eat", same thing follows in AI too. One gets the output according to the data fed into the model. The performance of any AI model is dependent on the quality of data.

After much brainstorming and asking a few doctors for the CXR data of the patient we finally found one dataset on Kaggle, this dataset has 3 classes Normal, bacterial pneumonia and viral pneumonia. For the scope of this project only a binary class dataset was needed.

After some preprocessing and adjustments on dataset 3 classes were merger into two i.e. Normal and Pneumonia.

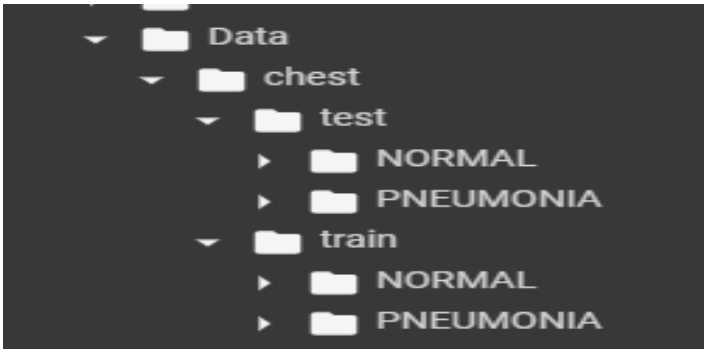



fig.24: data hierarchy

So taking a closer look at fig.24 will explain the data organization. There were a total 5897 CXR images 619 images were kept as test data and the rest as training data.

The actual image size is $1240 \times 840 \times 3$ but due computational limitations it had to be downsized to $128 \times 128 \times 3$. Downsizing the image leads to information loss which will lead to bad performance of the model. But to process $1240 \times 840 \times 3$ This size of image huge computational resources are needed. To balance between computational resources requirement and performance of the model, downsizing the images to $128 \times 128 \times 3$ worked best in this case.

### 3.2 SOLUTION APPROACH

#### 3.2.1 Best Model selection :

In this project, four state of the art CNN pre-trained models have been chosen for training and on top of them few Conv2D and Dense layers were mounted for better performance. Out of these four model architectures the best one with best performance accuracy and auc score combined have been picked for inference and classification. The summary and performance of both the model architectures are shown

- **ResNet50**

  In the fig.25 we can see that the accuracy and the AUC score has been plotted on each epoch. The scores on validation data are converging somewhere around 0.85 and train data is converging somewhere around 0.95. So it can be said that the

model is overfitting the data as there is a significant difference between the validation and train scores.

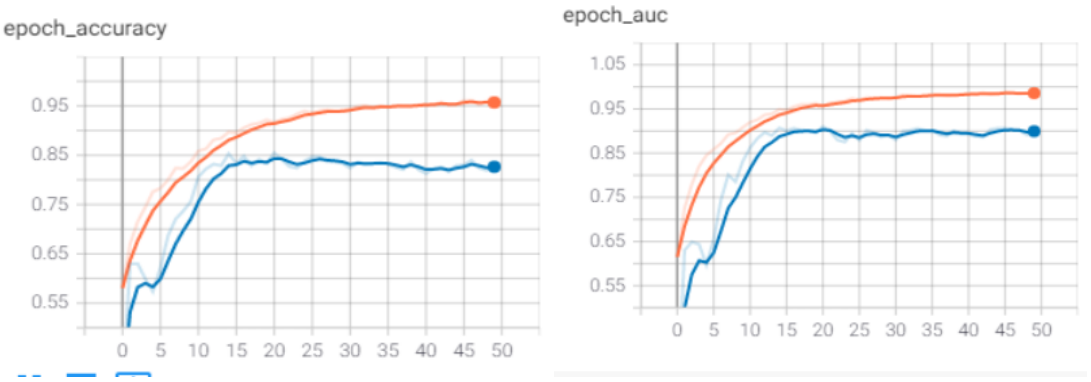



fig.25: ResNet50 training and performance

- **Vgg16**

In the fig. 26 we can see that the accuracy and the AUC score has been plotted on each epoch. The scores on validation data are converging somewhere around 0.8 and train data is converging somewhere around 0.98. So it can be said that the model is overfitting the data as there is a significant difference between the validation and train scores. The difference in performance is more than as it is in case of ResNet50.

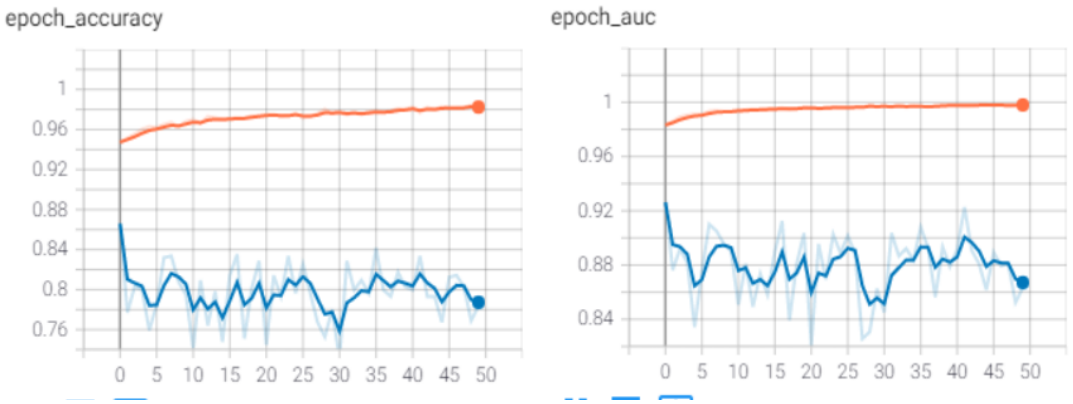



fig.26: VGG16 training and performance

If noticed on the plot of accuracy and AUC score above we can clearly see a big gap between the train performance scores. There could be many reasons for this

behaviour. Along with overfitting the model is underfitting as well. The performance on the validation data is effectively decreasing and the train performance is increasing.

**Comparison**

| attributes | VGG16 | RESNET50 | InceptionResnet V3 |
|---|---|---|---|
| Trainable parameters | **9,747,010** | **38,752,130** | **65,037,282** |
| validation Accuracy | **0.7835** | **0.8304** | **0.8061** |
| Validation AUC | **0.8631** | **0.9064** | **0.8761** |
| Epochs of training | **50** | **200** | **20** |

table 1: Model comparison

- **Model selection:**

Considering computational resources the obvious answer would be VGG16 but if the performance plots of both the models are referred, it is very much evident that the VGG16 model is overfitting the data since the difference between training auc score and validation auc score is continuously rising after each epoch. While the model ResNet50 performance plot is better and it is converging to a point which was not the case in VGG16. **So the ResNet50 has been selected even though it requires more computational resources and time for training.**

- Explanation **:** The main task of this project is the explanation given to the user about the CXR classification. Two methods are chosen for this.

### 3.3 SILHOUETTE EXPLANATION

In fig.27, CXR with NORMAL and PNEUMONIA has been shown. The main distinctive feature in a CXR is the silhouette separation. **Silhouette sign(source)** is somewhat of a misnomer and in the true sense actually denotes the loss of a silhouette, thus, it is sometimes also known as **loss of silhouette sign.** In loss of silhouette the CXR actually loses the border of separation among different parts of the chest.

To detect the loss of silhouette sign an algorithm has been developed. To mitigate the noise from row 20 to row 110 out of 128 rows of the image pixels have been considered. The image is now biparted and the sum of all the pixels from each half is compared then the most damaged half of the chest would be the one which has the maximum summation value. Loss of silhouette sign leads to loss of borders and makes the image more whitish. Now to find the most damaged quadrant of the chest the same thing is repeated but now in the most damaged half.

$$r, c = 128$$

$$left\ half = \sum_{i=0}^{r}\sum_{j=0}^{c/2} CXR(i,j),\ right\ half = \sum_{i=0}^{r}\sum_{j=c/2}^{c} CXR(i,j)$$

$$mask\ list = [left\ half\ masked,\ right\ half\ masked]$$

$$most\ affected\ half = masklist\ [argmax(masklist)\ ]$$

$$if\ argmax(masklist) = 0:$$

$$upper\ quad = \sum_{i=0}^{r/2}\sum_{j=0}^{c/2} right\ masked\ CXR(i,j),\ lower\ quad = \sum_{i=r/2}^{r}\sum_{j=0}^{c/2} right\ masked\ CXR(i,j)$$

$$if\ argmax(masklist) = 1:$$

$$upper\ quad = \sum_{i=r/2}^{r}\sum_{j=c/2}^{c} leftt\ masked\ CXR(i,j),\ lower\ quad = \sum_{i=r/2}^{r}\sum_{j=c/2}^{c} leftt\ masked\ CXR(i,j)$$

The above algorithm is to find the correct lobe in the CXR which is most dominant in classification. The bounding box is just an numpy array with last 2 elements from each end as values 0 and rest values as the corresponding CXR image values.

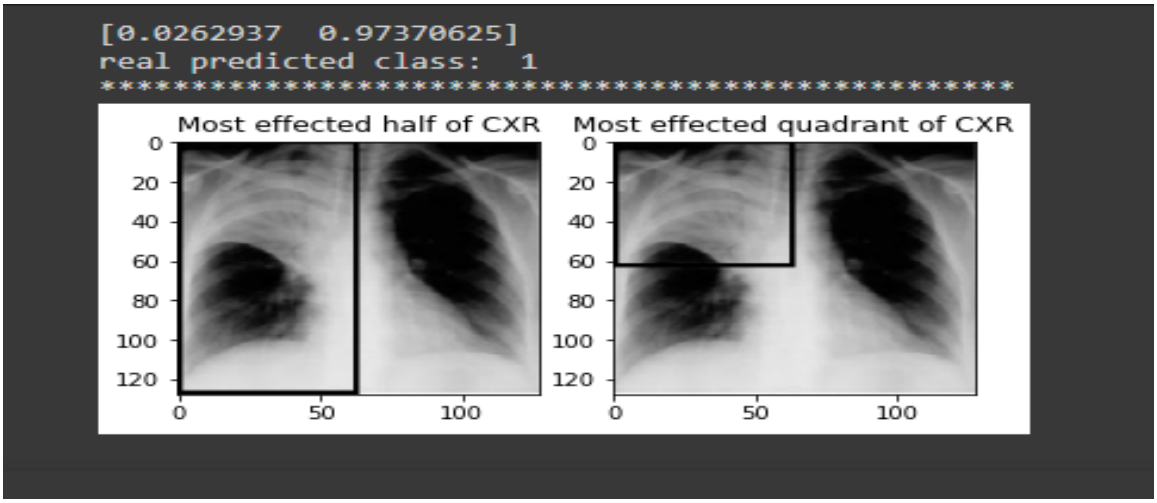



fig.27: Silhouette based result

### 3.4 GRADIENT BASED EXPLANATION

As discussed what gradient based explanation is. A separate CNN model with 2 Conv2D and 1 Dense layer has been trained for extracting the weights of each pixel. The model has been shown in fig.29. The AUC score on the validation data and train data has been plotted. We can see both train and validation performance seem to converge at a similar point. This shows that the model is robust and not overfitting over the data. The AUC score after the training of the model is somewhere close to 0.93 an validation data and similar to train data. Although the model is shallow the performance has come out to be too good to be true. The main purpose of this model is to get the weight matrix of the same size as the image.

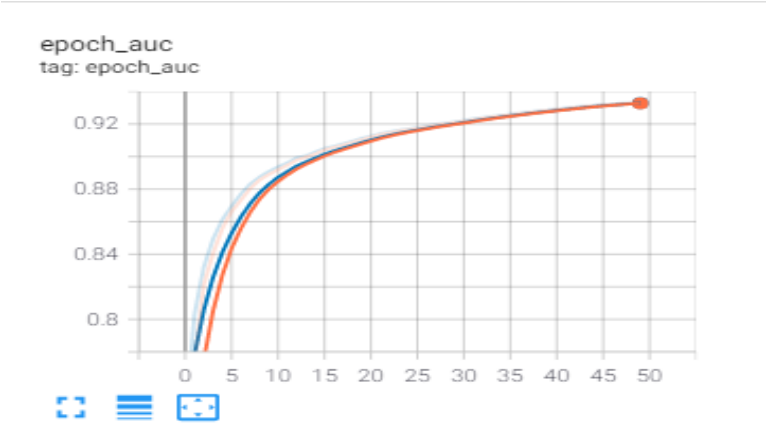


fig.28 : Grad Model Performance

The layer conv2d (Conv2D) has weights dimensions (128,128,3) which represents each of the pixel importance in the classification. These weights are extracted and shown as an image and the pixels which have the highest weightages in making decisions are the brightest. However, to analyse better a threshold of 0.7 has been chosen. So the pixels which have more than 70% contribution in classification are shown.

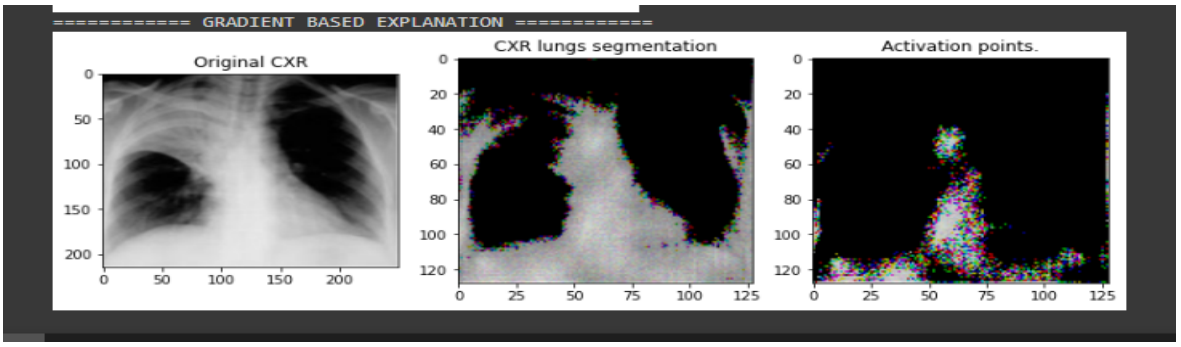



fig.29 : Gradient Based Explanation result

The image shown in the middle and last in fig.29 are the direct plot of conv2d (Conv2D) layer output with threshold of 0.5 and 0.7. In the last image the pixels are plotted whose contribution in the classification is more than 70%. This could be very useful , as one may think that this algorithm will show only the white color pixel values as the most contributing pixels in classification. But, if it is noticed carefully one can see in the fig. 29 that in the right upper lobe (RUL) of the chest there is a complete white or in other words we can say that RUL is completely washed away. But in the 3rd image it is clearly seen that not all of the pixels in RUL are shown as the most dominant pixels but a cluster of pixels are shown which shows that our model is learning something out of the dataset. The algorithm follows basic mathematical operation on the original input image and the gradients of the pixels. First, the gradients are extracted from the trained grad model and multiplied with the input image. By doing this we get the importance of each pixel then the pixel values are normalized between 0 and 1. Finally, we choose a threshold value between 0.5 and 1 to refine the gradient plotting . Higher the threshold value smaller the number of pixels we get. Then after all these processes we get to see the pixels plotted as the explanation for the classification of the CXR as shown in the fig.29.

# 4. Results and Evaluation

The metrics used for this classification problem is

- Accuracy
- AUC score

The accuracy and AUC scores for the final selected models are shown in the table. The metrics values are on the test dataset which contains 619 CXR images.

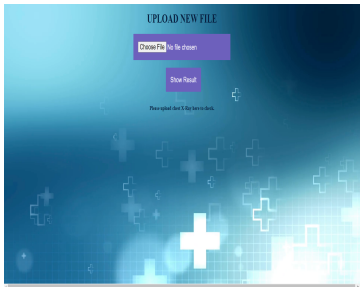
| Model | Accuracy | AUC |
|---|---|---|
| InceptionResnetV3 | 0.8061 | 0.8761 |
| Grad Model | 0.75 | 0.9334 |

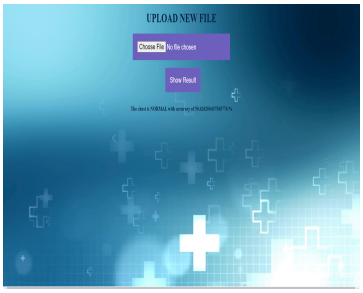
table 3: results

AUC and accuracy have come out to be as 80% and 87.6% for inceptionResnetV3, 75% 93.34% from the grad model respectively. Even though in the grad model we have AUC score 93.34% but the model is slightly biased. The main reasons are that the architecture of the model is shallow, we have a very small dataset to train the model. These two reasons are enough to make instead of the fact that the image input for inference in NORMAL the model will show the user it is having PNEUMONIA and explain some random pixels and lobe.

The result is shown in a UI where a user can upload the image from the internet and can click show result to see the classification and result analysis. The UI is kept simple and straightforward. No other functionalities other than I/O are added as of now and would be considered an upgrade in the near future. The output consists of classification classes with probability. In addition to above details 3 images are also displayed. One image is the original image, the second image is the image with a bounding box around the correct lobe and the final image is the most dominant pixels plotted. One thing to note is that when the image is classified as NORMAL then the image with bounding box and image with most dominant pixels are not plotted. This is because if the image is normal the user is only interested in knowing what is the likelihood of this classification. In the normal CXR image no lobe is affected and it does not make sense to make any random bounding box on the CXR. Considering these facts we have chosen to show image analysis only when the classification is in PNEUMONIA.

**FINAL OUTPUT**



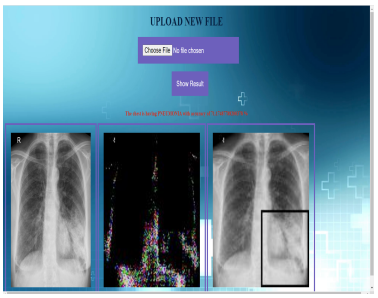




fig.30: UI landing page fig.31 : NORMAL CXR fig.32 : PNEUMONIA CXR

# 5. Conclusion and Future Work

The objective of explaining to the user why has his/her CXR been classified as NORMAL or PNEUMONIA has been achieved. The SILHOUETTE sign shows the most damaged quadrant of the CXR and to complement this overall explanation is provided by the GRADIENT based explanation method. There are some cases where these two methods contradict a little. In fig.33, the contradicting results have been shown.

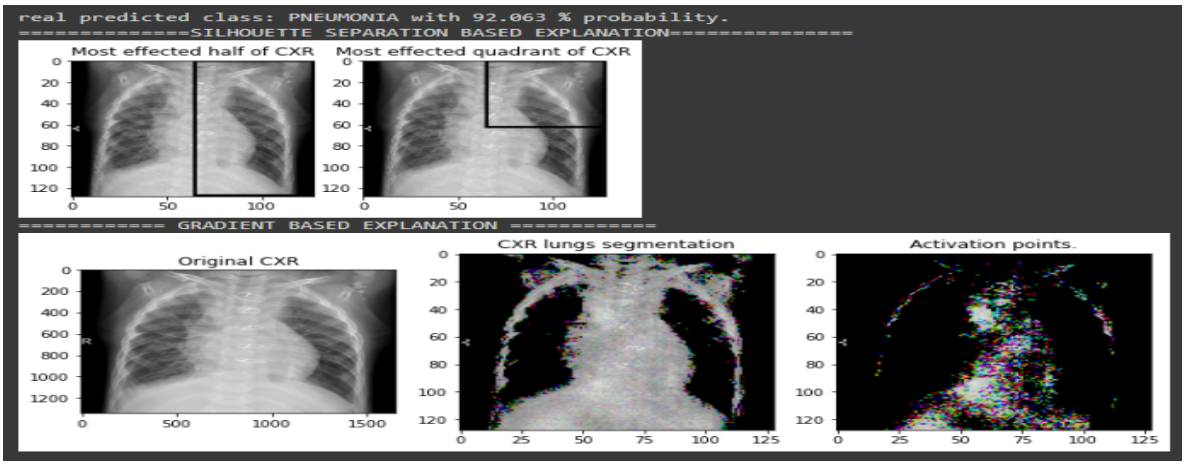



fig.33: Contradicting results

The classification is working great here, but the pneumonia seems to be present in the right side of the patient's chest. However, the SILHOUETTE method showed the left half of the chest as having pneumonia. But when the gradient explanation is looked upon this shows the more correct explanation as wherever there is loss of border is detected the weights of those pixels are lighting up. So to pick one among these two methods, the GRADIENT based explanation method would be the choice. However when the Pneumonia is widely spread in the chest then the SILHOUETTE method can do wonders as shown in fig.27.

Possible reasons for this contradiction:

- There are very minor differences between the densities of chest parts which leads to the very vague border shared among them.
- Low dimension of shape of images taken.
- Shallow neural network
- Size of the dataset is not enough.

### 5.1 FUTURE WORK (or outlook)

The future of this project work revolves around fine tuning the model and achieving better performance. The performance of the present model is great when there is clear Pneumonia in the CXR when the CXR is having small or very slight Pneumonia then the model struggles to perform to it's best. If the classification does not happen properly then all the following actions will fail.

There is another problem in estimating the bounding box around the correct CXR lobe. Basically, the estimation of the bounding box is based upon the density difference of the different parts in the CXR. So if you notice there is some white noise in every CXR image. These noises affect the calculation of the correct lobe.

Now, to predict the most dominant pixels of the CXR which are leading to the classification of the image then we need a robust model trained on the data as it is very easy to confuse the model in this use case. Generally, the white noise is having the maximum pixel values since they are in white color to minimise the participation of noise in the calculation we need to consider exact parts of the CXR which contains the lungs.

To solve the above mentioned shortcoming of the project some procedures can be followed as mentioned:

- Experiment with a bigger dataset.
- The computation power is limited to (128,128) size of the image. Images with more pixels can be used.
- More robust model with more layers can be experimented with.

- On the explanation side, as of now we are only checking with 4 bounding boxes on the CXR to estimate the correct lobe. But to be more precise we need to have more bounding boxes proposals, say 16.

## 5.2 SCOPE

- Improve the performance of the silhouette explanation.

  As we know the working of finding the correct lobe algorithm where it uses the density based separation which is also known as Silhouette separation to get the most dominant lobe in the image. For this algorithm we were only using the 4 boxes around each lobe in the CXR image to predict the right lobe. However we have completely ignored the right middle and left middle lobe for coping up with the resource limitations. One major solution to this problem would be to use smaller boxes i.e one-eighth of the image as one dominant region and six lobe. Doing this we will have a total of 24 images for each image. If the original dataset is of 3GB then for this solution the dataset would be of 72GB. However these memory constraints can be avoided by using the TRAIN THEN MASK approach. Where the model will be first trained on the original dataset then one-eighth masked image will be fed to the model for classification. Here the data loss is 12.5 % from the original then we should expect correct classification from the model and if the classification is true then other following explanations can be made correctly.

- Use of lungs segmentation and using only the lungs part to predict the most dominating pixels by avoiding white noise.

  We have seen the state of the art solution where manual image labelling is done before training the model. As seen before, when the pixel value is less than 0.5 then we can see clear separation of lungs from other parts in the chest as shown in the fig 34 below. The left image is the original image and the right image is the image after selecting pixels with gradient values less than 0.5. These gradients could be used for separating lungs from the entire image to remove the noise and do pixel analysis and get most dominating pixels only from the lungs part which will be more precise to the explanation.

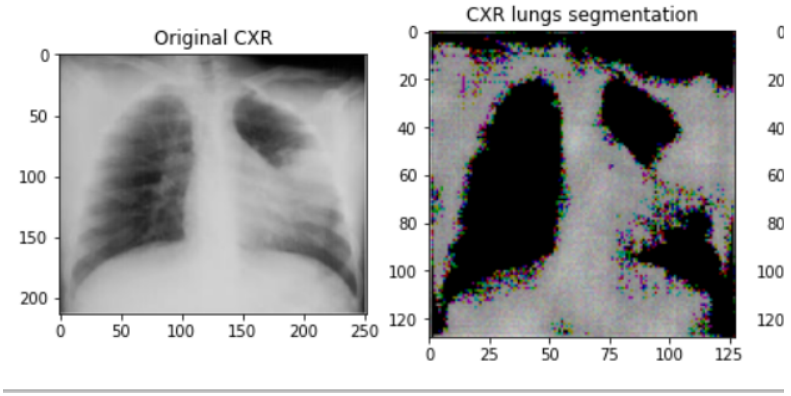



fig.34 : Lung separation

- Final step in this project would be to show the results in an interactive UI.

  Although we have a simple UI for showing the image analysis and classification, there are very few functionalities for the user. I/O operations are only allowed on this UI.

# 6. Try & Failures

### 6.1 MASKING AND TRAINING

In this approach, the algorithm consists of training the model with masked images. The masking was not random, the masking was done systematically on each CXR image twice. Half of the image was masked and the other was left open for the model to learn similarly the other half masked and the former masked half was open for the model to learn. This dataset was twice the size of the original dataset as there were two half masked images for each image. The target values were changed accordingly let's say the CXR image is from PNEUMONIA and by human intelligence it is clear that right side of the CXR is having Pneumonia then right side masked image is labeled as 0 as the half having Pneumonia is masked and the other image is labeled as 1 where left part is masked and right part where Pneumonia exit is exposed to model. The above algorithm is clearly shown in the fig. 36. If the right side masked of the CXR is having Pneumonia then that masked image is labeled as 0 and the other as 1.

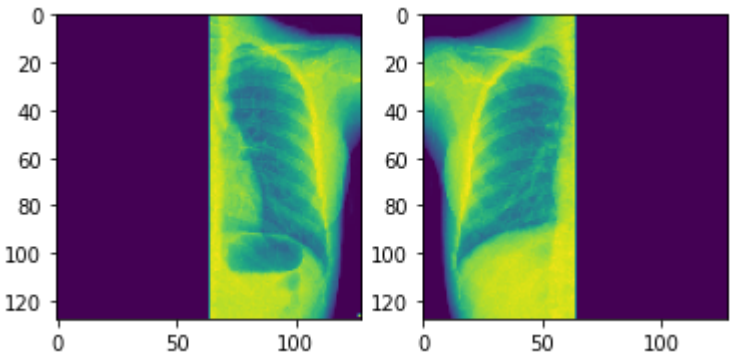



fig. 35 : Masking of type 1

After this there is a next step where the image is masked quarterly and we have 4 differently masked images for each CXR. In this approach we have 4 times the original dataset. The system was not able to handle this much data, so the original data set was clipped by 4 times. The main objective of this approach was to find the quadrant where Pneumonia exists. By human intelligence one quadrant where Pneumonia was dominant and not masled was labeled as class PNEUMONIA and the other masked quadrant was labeled as NORMAL. In this case too there is a data loss but half of the previous one but here we are only taking one-fourth of the original data.

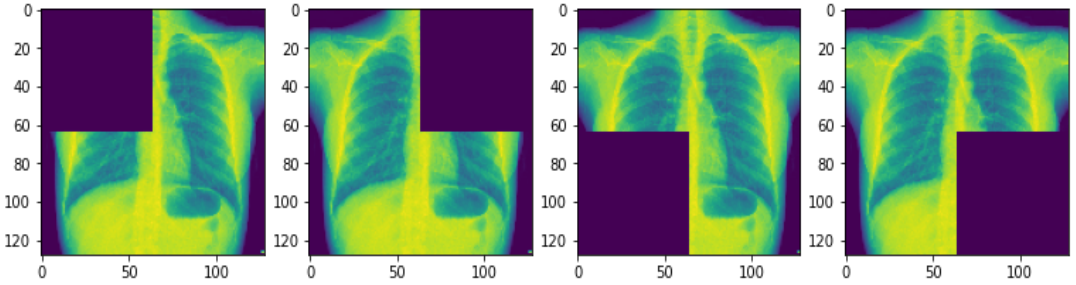



fig. 36 : Masking type 2

In the above algorithms, there is a lot of data loss. Only half and quadrant of the image is fed to the model and the rest part of the image is masked. Model learnt the relationship of NORMAL and PNEUMONIA class separately but in reality in a CXR few regions in the image are NORMAL and few regions have PNEUMONIA. So in this case NORMAL and PNEUMONIA co-exist but the model never saw that coexistence as both the classes were fed separately.

### 6.2 TRAINING THEN MASKING

In this method or algorithm, the model was first trained on the original data without any kind of masking. After training the model the CXR images were masked, first each half was masked and then we masked each of the quadrants of the images. To know which half of the image is having Pneumonia we fed both the half masked images to model to predict the class and note the classification probabilities. It may be possible that both the masked images would be classified in a similar class so to get the half of the image having Pneumonia the classification probabilities are compared and picked image of the maximum value of the probability.

Now to get the affected lobe we do the same thing with the images with their quadrant masked.

This method solves the problem of data overflowing the memory issue. As the training data remains the same and we are testing the model with the masked data only for the inferences. However, the model is not robust as the model is trained on normal images without mask data and during the inference the model is fed a masked image. There is a great chance that the model will predict the masked image opposite to the groundtruth.

### 6.3 LAYER EXPLANATION

In this, first the classification model is trained on the original data without masking. However, for this approach we need to make a custom model with a subclassing method. A subclassing model of model architecture is a method of defining model architecture where the person who is making the model has the freedom to play with the layers he is defining. This is possible by inheriting the Model class and Layer class from tensorflow.Keras .

The problem with this approach is that different layers learn different parts of the image. When the layers are plotted they give random output which is hard to interpret and even harder for the end user to interpret if he does not understand the working of layers. The output of the MaxPooling2D has been shown in the fig. 37 below for better understanding.

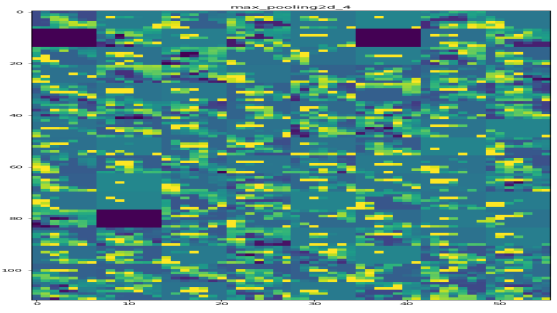



fig.37 : MaxPool layer output

This is the output of the one of the MaxPool layer deep inside the layer of the CNN architecture. This to an end user does not make any sense and could not be inferred from this. However, the starting layers may be useful to show the explanation as the starting layers learn the very basic features like vertical and horizontal edges of the image. But the starting layers would not be enough to understand the output as the features learnt by layers are very generic which can be applicable to both the classes NORMAL and PNEUMONIA in this case.

# 7. Bibliography

[1] Thakkar, S. (2020). *AI vs ML vs DL: What's the difference?*

[2] Computer Science Department Stanford University Stanford, CA 94305. (n.d.). *WHAT IS ARTIFICIAL INTELLIGENCE?*

[3] *A. M. Turing (1950)* Computing Machinery and Intelligence. Mind 49: 433-460.

[4] IBM. (n.d.). *What is artificial intelligence?*

[5] Barbaroux, H. (2020, 06 03). *SUPERVISED LEARNING VS UNSUPERVISED LEARNING*.

[6] *University of Strathclyde Glasgow, Bellekens, X., University of Strathclyde, University of Strathclyde, & University of Strathclyde. (2017, 01). INTRUSION DETECTION SYSTEM.*

[7] Agarwal, M. (n.d.). *Back Propagation in Convolutional Neural Networks*.

[8] *TRANSFER LEARNING IN DEEP LEARNING*. (2020, 11 08). https://www.analyticsinsight.net/forum/profile/analyticsinsight/

[9] Centre for Data Analytics, NUI Galway Aylien Ltd., Dublin. (n.d.). *An overview of gradient descent optimization algorithms*.

[10] Saket Thavanani. (2020, 05 11). *Comparative Performance of Deep Learning Optimization Algorithms Using Numpy*.

[11] Electrical Engineering, Indian Institute of Technology, Hyderabad. (n.d.). *Transfer learning using VGG-16 with Deep Convolutional Neural Network for Classifying Images*. 10.29322/IJSRP.9.10.2019.p9420.

[12] A. S. B. Reddy and D. S. Juliet. (n.d.). "Transfer Learning with ResNet-50 for Malaria Cell-Image Classification. *2019 International Conference on Communication and Signal Processing (ICCSP)*. 10.1109/ICCSP.2019.8697909

[13] ImageNet. (n.d.). http://www.image-net.org

[14] He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. (n.d.). "Deep residual learning for image recognition." *IEEE conference on computer vision and pattern recognition*, (pp. 770-778. 2016.).

[15] *Silhouette sign*. (n.d.). https://en.wikipedia.org/Silhouette_sign

[16] *IIIT Hyderabad, IIT Hyderabad, & Cisco Systems, Bangalore. (n.d.). Grad-CAM++: Generalized Gradient-based Visual Explanations for Deep Convolutional. https://arxiv.org/pdf/1710.11063v1.pdf.*

[17] *CNN layers*. (n.d.). http://d2l.ai/chapter_convolutional-neural-networks/conv-layer.html

[18] *Maxpooling layer image*. (n.d.). https://androidkt.com/explain-pooling-layers-max-pooling-average-pooling-global-average-pooling-and-global-max-pooling/

[19]Dropout image. (n.d.). https://www.researchgate.net/figure/Example-of-dropout-layer-probability-of-50-appears-on-the-right_fig8_333411007

[20](semantic vs instance image. (n.d.). Semantic Segmentation using Deep Lab V3 | Deep Learning Analytics

[21]confusion matrix image. (n.d.). Confusion Matrix for Multiple Classes in Python - Stack Overflow

[22]maxpool layer output image). (n.d.). Visualizing representations of Outputs/Activations of each CNN layer - GeeksforGeeks

# Appendix

https://github.com/GuptAmit725/Master_project

https://github.com/GuptAmit725/Master_project/blob/main/MSPROJECT_4.ipynb