PREDICTING DISEASE TYPE BASED ON CHEST X-RAY SCANS

<u>GROUP MEMBERS:</u>

● **ADITYA JAIN**	**01FB15ECS018**
● **ABHUDAY VIBHASNHU**	**01FB15ECS011**
● **ABDULLAH ANWAR**	**01FB15ECS004**
● **ANUJ BHUSHAN**	**01FB15ECS044**

CONTENTS

PROBLEM STATEMENT

The project deals with the idea of classifying some 14 classes of chest diseases based on the X Ray Scans of the Chest portion. The Model learns with the training examples and then on test of the model, it responds to the target classes of the diseases upon similarities to the learned data the model has, making the disease to be classified among one of the 14 disease types to be classified.

The project is based on improving the medical analysis of the disease examination and how machine learning can improve and help the medical domain in a better and accurate way of treatment of diseases.

Machine Learning Techniques

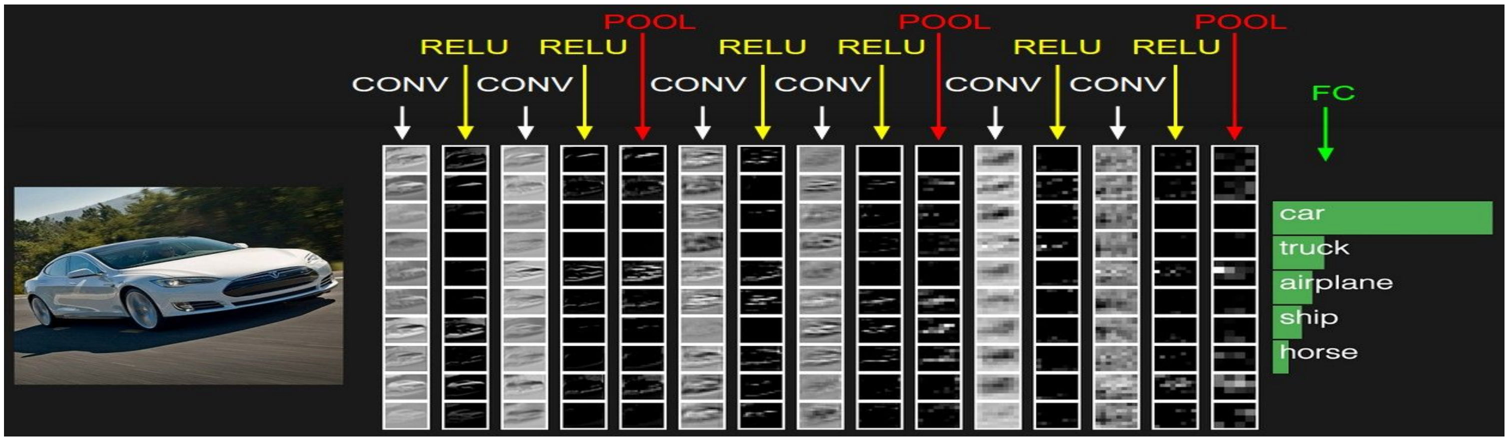
The techniques that will be incorporated for the project will be Unsupervised Learning (K-Means) and computational and model build by the help of CNN and ANN, to train the model from X-Ray Images of each patient. The K-Means can easily classify all the disease types based on the instance based learning, so incorporating K-Means and CNN can make a good model for our project.

To justify the selection of the following techniques can be that the dataset given to us has a target variable and learning based on that has to be unsupervised learning, also K-Means can be really helpful in predicting the Chest Disease type based on the classes of Diseases available and learnt.
whereas, CNN is a class of deep, feed-forward artificial neural networks that has successfully been applied to analyzing visual imagery. Since our dataset also deals with images of Chest scans for each patient the model also needs to be trained for it, hence we can use CNN for Image related training and the perceptron and size of the Neural Network can be determined based on learning from the dataset.

A- Convolutional Neural Network:

Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity. The whole network still expresses a single differentiable score function: from the raw image pixels on one end to class scores at the other. And they still have a loss function (e.g. Softmax) on the last (fully-connected) layer and all the tips/tricks we developed for learning regular Neural Networks still apply.
A simple ConvNet is a sequence of layers, and every layer of a ConvNet transforms one volume of activations to another through a differentiable function.

We use three main types of layers to build ConvNet architectures: **Convolutional Layer**, **Pooling Layer**, and **Fully-Connected Layer** (exactly as seen in regular Neural Networks). We will stack these layers to form a full ConvNet **architecture**



B- K Means :

Given a set of observations (x_1, x_2, …, x_n), where each observation is a *d*-dimensional real vector, *k*-means clustering aims to partition the *n* observations into *k* (≤ *n*) sets S = {S_1, S_2, …, S_k} so as to minimize the within-cluster sum of squares (WCSS) (i.e. variance). Formally, the objective is to find:

$$\underset{\mathbf{S}}{\arg\min} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2 = \underset{\mathbf{S}}{\arg\min} \sum_{i=1}^{k} |S_i| \operatorname{Var} S_i$$

The Equivalence can be deduced from identity . Because the total variance is constant, this is also equivalent to maximizing the squared deviations between points in *different* clusters

$$\sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2 = \sum_{\mathbf{x} \neq \mathbf{y} \in S_i} (\mathbf{x} - \boldsymbol{\mu}_i)(\boldsymbol{\mu}_i - \mathbf{y})$$

About the Data-Set

<u>Data-Set Details</u>:
Source of data generation available freely to world by:

National Institute of Health-Clinical Centre is one of the leading hospitals in the United States

As a part of a research study to explore deep learning techniques, NIH has recently open-sourced their dataset of frontal chest X-ray images of patients.

<u>Description about the dataset:</u>

We have been provided with two separate files to download (images and csv files). Data Set available as a package for HackerEarth competition. The train data has information for 18577 patients and test data has information for 12386 patients. The target variable has 14 types of thorax disease.

In- detail features:

- Attribute Count : 6
- Attribute List:[row_id , age , gender , view position , image_name , detected]
- Class Count : 14
- Class List: [class_1 to class_14]
- "Effusion": "class_5", "Fibrosis": "class_1", "Infiltration": "class_3"
- "Edema": "class_8", "Consolidation": "class_11", "Emphysema": "class_14"
- "Atelectasis": "class_7", "Pleural_Thickening": "class_13"
- "Nodule": "class_12", "Hernia": "class_9", "Mass": "class_4"
- "Pneumothorax": "class_6", "Pneumonia": "class_10"
- "Cardiomegaly": "class_2"

Final result format:

row_id,	**detected**
Id_100,	**class_1**

Id_10002, class_6

- **Total Instance Count**: The train data has information for 18577 patients and test data has information for 12386 patients. Making the total size of the dataset to be around : 30,963 patient data.

Data Preprocessing and Data Cleaning:

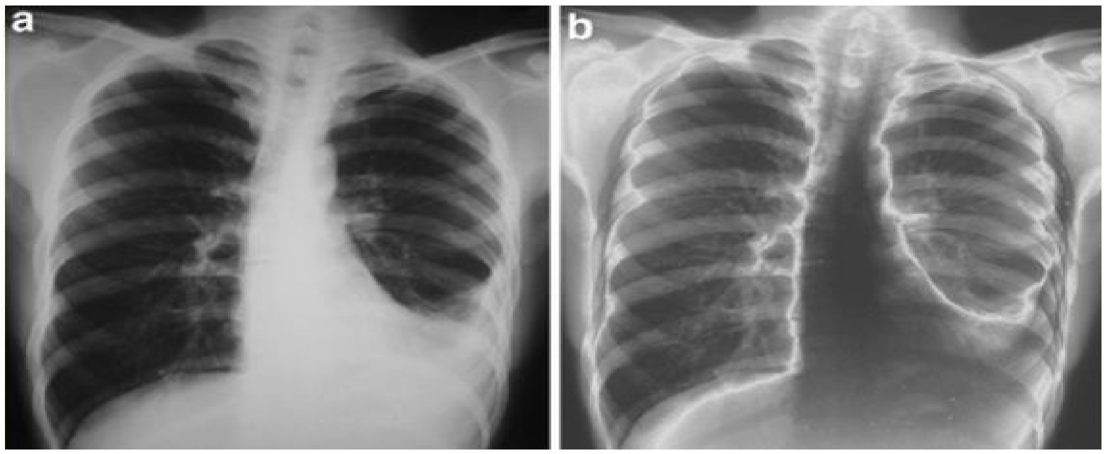
Digital radiographic image enhancement techniques are used to accentuate and sharpen image features for enhanced visualization and better quality images. Image enhancement does not try to quash the artifacts it just makes it easier to interpret by improving visualization.

In radiographic imaging, such as digital x-rays, CT scan or mammogram, images of different organs and tissues are produced. There are many sources of interference in the production of radiographic images, such as insufficient performance, noise of imaging device and the movement of patient. The quality of many images is poor in their contrast, and to improve their quality for better visualization, to see clearly enough critical details and reduce the noise for diagnostic purposes, methods of enhancement are used. The purpose of image enhancement is thus to improve the/ digital image quality and to support the human perception.
Image Contrast is the difference in appearance of two or more parts of an image seen simultaneously. An image must have good brightness contrast for proper vision. In a low contrast image we can't distinguish clearly between different objects. Increasing the contrast makes the light areas become lighter and dark areas become darker. Contrast enhancement increases the visual perception of difference between different parts of an image.

The images which we have has a resolution of 1024 x 1024(number of pixels in rows and columns). While we can load all the images in the same way, processing 18000 images of size 1024x1024 require enormous computation power. Instead, we resize them to a more appropriate sizes. After doing all types of contrast enhancements we resize the resolution to 128 x 128. This may result in some of loss of information, but this was necessary because of limited computational power.

HISTOGRAM BIAS CORRECTION

The histogram equalization processing gives an approach to force the image histogram to become uniform. To make the linear transformation suitable to various chest radiographic images, we may first obtain the image of uniform histogram by means of the histogram equalization. The idea in equalizing a histogram is to stretch and/or redistribute the original histogram using the entire range of discrete levels of the image, in a way that an enhancement of image contrast is achieved. Figure below shows an illustrating example of using HE for image contrast enhancement.



Left: Original normal chest radiographic image. Right: Processed image.

DATA AUGMENTATION

To take advantage of our large dataset, we implemented basic data augmentation techniques to prevent overfitting in our model while still making use of all the data available. Each training image, before being input into a neural network, was flipped 0, 90 degrees. Additionally, each image was either flipped left to right or not. Lastly, each image had some random small amount of gaussian noise added to each pixel value.

Design and Architecture of Model

ConvNet architectures make the explicit assumption that the inputs are images, which allows us to encode certain properties into the architecture. These then make the forward function more efficient to implement and vastly reduce the amount of parameters in the network.

Each hidden layer is made up of a set of neurons, where each neuron is fully connected to all neurons in the previous layer, and where neurons in a single layer function completely independently and do not share any connections.
We use three main types of layers to build ConvNet architectures: Convolutional Layer, Pooling Layer, and Fully-Connected Layer (exactly as seen in regular Neural Networks). We will stack these layers to form a full ConvNet architecture.

We will go into more details below, but a simple ConvNet for Chest X-Ray Disease classification could have the architecture [INPUT - CONV - RELU - POOL - Fully Connected(FC)]

A- **INPUT** [128x128x3] will hold the raw pixel values of the image, in this case an image of width 128, height 128, and with three color channels R,G,B.

B- **CONV** layer will compute the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input volume. This may result in volume such as [128x128x12] if we decided to use 12 filters.

C- **RELU** layer will apply an elementwise activation function, such as the max(0,x) thresholding at zero. This leaves the size of the volume unchanged ([128x128x12]).

D- **POOL** layer will perform a downsampling operation along the spatial dimensions (width, height), resulting in volume such as [16x16x12].

E**- FC (fully-connected) layer** will compute the class of diseases, resulting in volume of size [1x1x14], where each of the 14 types correspond to a classes of diseases, such as among the 14 categories of Chest X-Ray. As with ordinary Neural Networks and as the name implies, each neuron in this layer will be connected to all the numbers in the previous volume.

G- **Network Architecture:** Three different architectures were used to build the model. Each architecture varies in terms of complexities and number of epochs they were trained for. All CNN models were trained on google cloud platform. Each CNN model performance was assessed using a held out test set of 10% of our data. Overfitting was assessed by comparing the cross entropy loss and accuracy on training vs test data. The data were split into training and test sets using a 70-30 split. Each CNN model performance was assessed using a held out test set of 30% of our data.

MODEL 1

This model relies on the concept of the inception module, whose invention aimed to alleviate two main problems:

1. A large network is more accurate and better at classification but also more prone to overfitting

2. Increasing a network in size dramatically increases the computational power necessary to train that network.

This architecture consists of 3 layers with each layer consisting of multiple convolutional layers. It also implements an additional dimensionality reduction step in each module to limit the number of dimensions being input to each module to save computational power.The architecture consists of different sized filters that are each passed over the same input image whose outputs are all concatenated together, along with a max pooling on the original image, to produce the final module output. Towards the end we add two fully connected layer each consisting of 4096 neurons. Neurons in a fully connected layer have full connections to all activations in the previous layer, as seen in regular Neural Networks. Their activations can hence be computed with a matrix multiplication followed by a bias offset.

We are using adaptive optimization (ADAM) that computes adaptive learning rates for each parameter. The reason for using adaptive optimization is because researchers say that in deep networks adam is best suited as the test accuracy achieved by this is very high but this comes at the cost of training accuracy. This model is prone to overfitting so to avoid overfitting we use two techniques namely early stopping and dropouts. In early stopping we stop the training If the training accuracy does not improve over certain number of epochs. This number is referred to as patience in keras module.

Dropout is a regularization technique for reducing overfitting in neural networks by preventing complex co-adaptations on training data. It is a very efficient way of performing model averaging with neural networks. The term "dropout" refers to dropping out units (both hidden and visible) in a neural network.

The given statistics shows us the number of trainable parameters.

```
Total params: 19,047,278
Trainable params: 19,047,278
Non-trainable params: 0
```

MODEL 2

This model uses the same concept as used in model 1. The difference comes in terms of the model complexity. This architecture is little less complex when compared to our first model.The architecture consists of three layers with each layer. Each layer has two convolution layer and to the end of each layer we add a max pooling layer to reduce the dimensions. Towards the end we add a two fully connected layer each consisting of 128 and 256 neurons. Neurons in a fully connected layer have full connections to all activations in the previous layer, as seen in regular Neural Networks. Their activations can hence be computed with a matrix multiplication followed by a bias offset. The optimization function is same as model 1 which is adaptive optimization.

The given statistics shows us the number of trainable parameters.

```
Total params: 3,535,022
Trainable params: 3,535,022
Non-trainable params: 0
```

MODEL 3

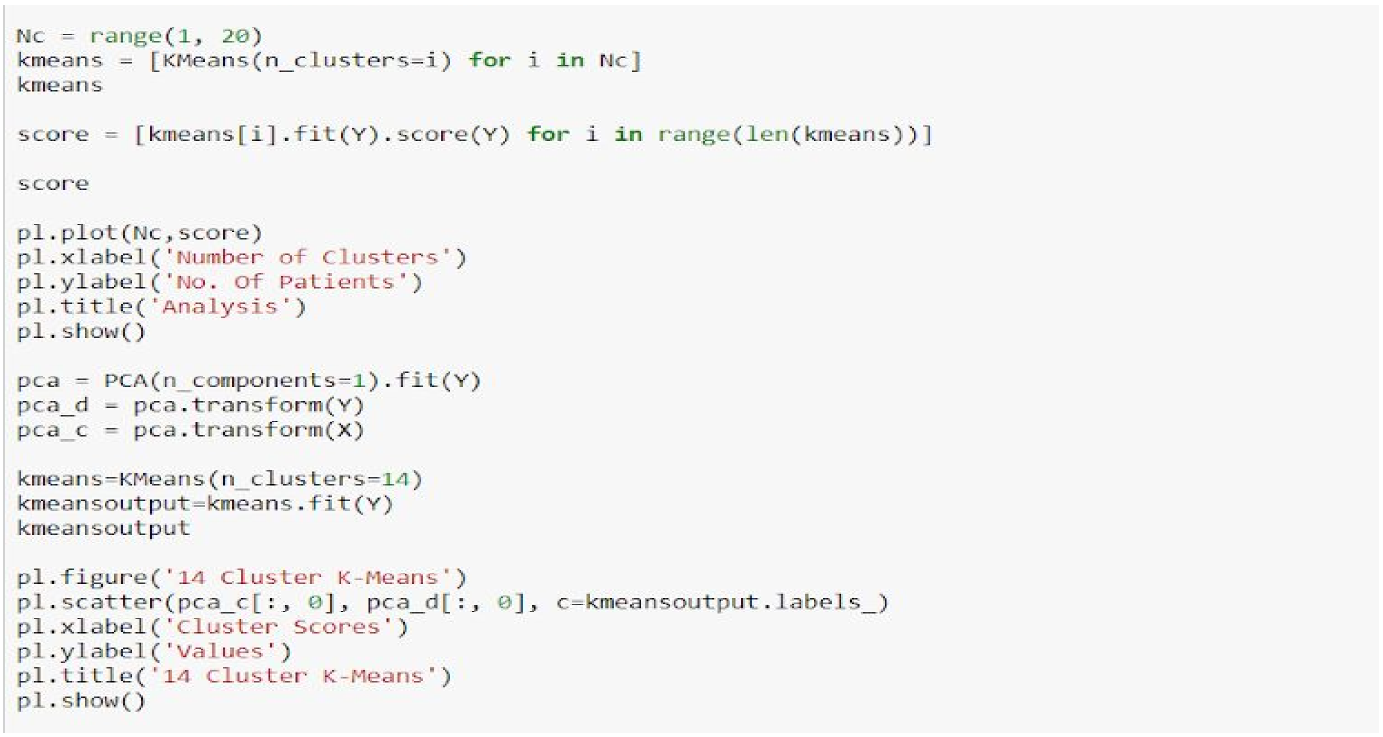
The model architecture is same as model 2. Deep networks need large amount of training data to achieve good performance. To build a powerful image classifier using very little training data, **image augmentation** is usually required to boost the performance of deep networks. Image augmentation artificially creates training images through different ways of processing or combination of multiple processing, such as **random rotation, shifts, shear and flips, etc.** Each training image, before being input into a neural network, was **flipped** 90 degrees. Objects in our images may not be centered in the frame. They may be off-center in a variety of different ways. Keras supports separate **horizontal and vertical random shifting** of training data by the width_shift_range and height_shift_range arguments. **zoom_range** specifies range for random zoom. The optimizer used in this model was traditional stochastic gradient descent (sgd). Different articles were referred and they say that sgd under fits even after training for 300 epochs but the advantage with this is that it is much more reliant on a robust initialization and annealing schedule. Consequently, if you care about fast convergence and train a deep or complex neural network, you should choose one of the adaptive learning rate methods.

What we observe with this model is that the training accuracy increases very slowly with each epoch but a better test accuracy is observed.

```
Total params: 3,535,022
Trainable params: 3,535,022
Non-trainable params: 0
```

Pseudo Code and Algorithms

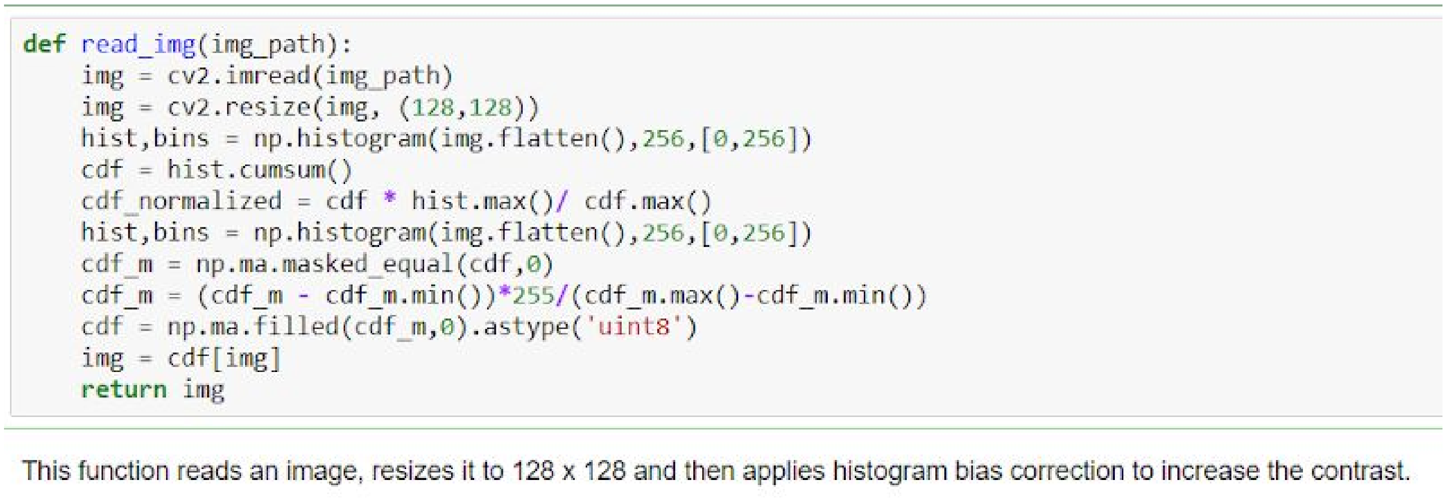
K-Means Classification:



```
Nc = range(1, 20)
kmeans = [KMeans(n_clusters=i) for i in Nc]
kmeans

score = [kmeans[i].fit(Y).score(Y) for i in range(len(kmeans))]

score

pl.plot(Nc,score)
pl.xlabel('Number of Clusters')
pl.ylabel('No. Of Patients')
pl.title('Analysis')
pl.show()

pca = PCA(n_components=1).fit(Y)
pca_d = pca.transform(Y)
pca_c = pca.transform(X)

kmeans=KMeans(n_clusters=14)
kmeansoutput=kmeans.fit(Y)
kmeansoutput

pl.figure('14 Cluster K-Means')
pl.scatter(pca_c[:, 0], pca_d[:, 0], c=kmeansoutput.labels_)
pl.xlabel('Cluster Scores')
pl.ylabel('Values')
pl.title('14 Cluster K-Means')
pl.show()
```

Convolutional Neural Networks

A- Data Preprocessing



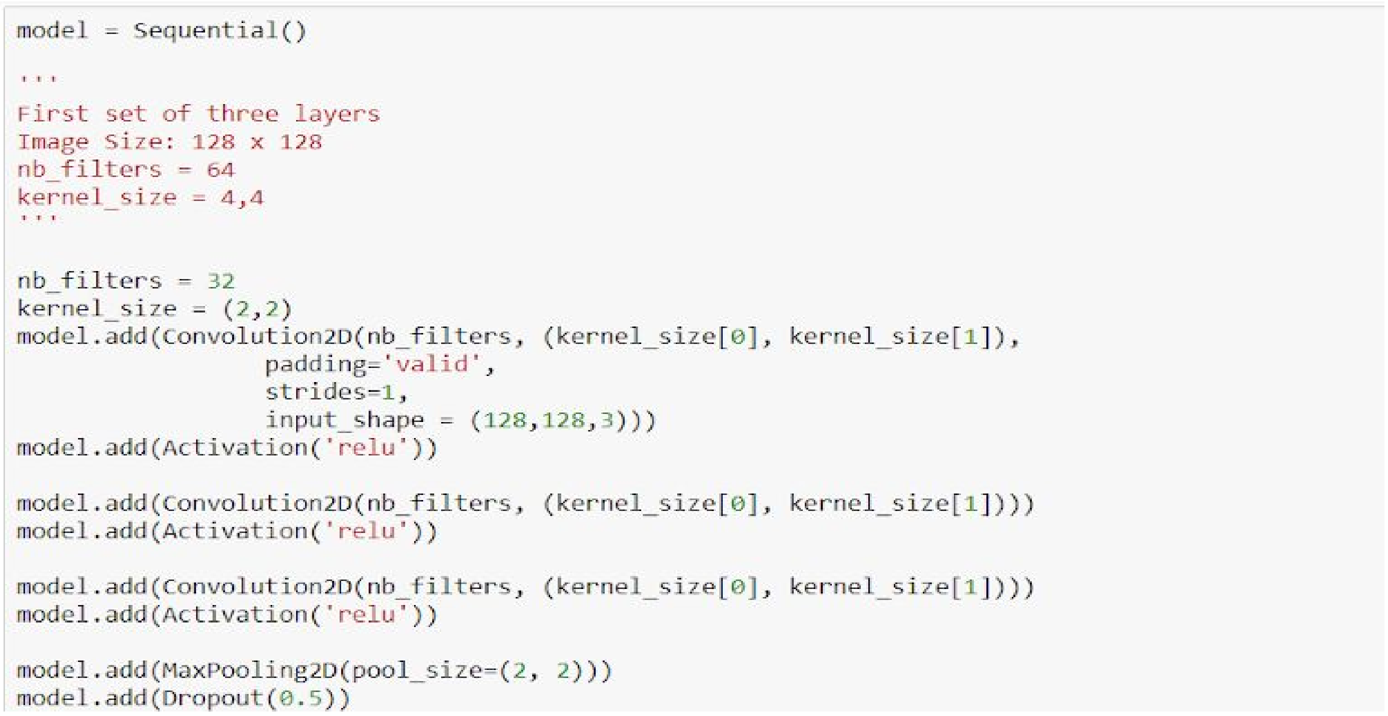
```
def read_img(img_path):
    img = cv2.imread(img_path)
    img = cv2.resize(img, (128,128))
    hist,bins = np.histogram(img.flatten(),256,[0,256])
    cdf = hist.cumsum()
    cdf_normalized = cdf * hist.max()/ cdf.max()
    hist,bins = np.histogram(img.flatten(),256,[0,256])
    cdf_m = np.ma.masked_equal(cdf,0)
    cdf_m = (cdf_m - cdf_m.min())*255/(cdf_m.max()-cdf_m.min())
    cdf = np.ma.filled(cdf_m,0).astype('uint8')
    img = cdf[img]
    return img
```

This function reads an image, resizes it to 128 x 128 and then applies histogram bias correction to increase the contrast.

B- Model Construction and learning

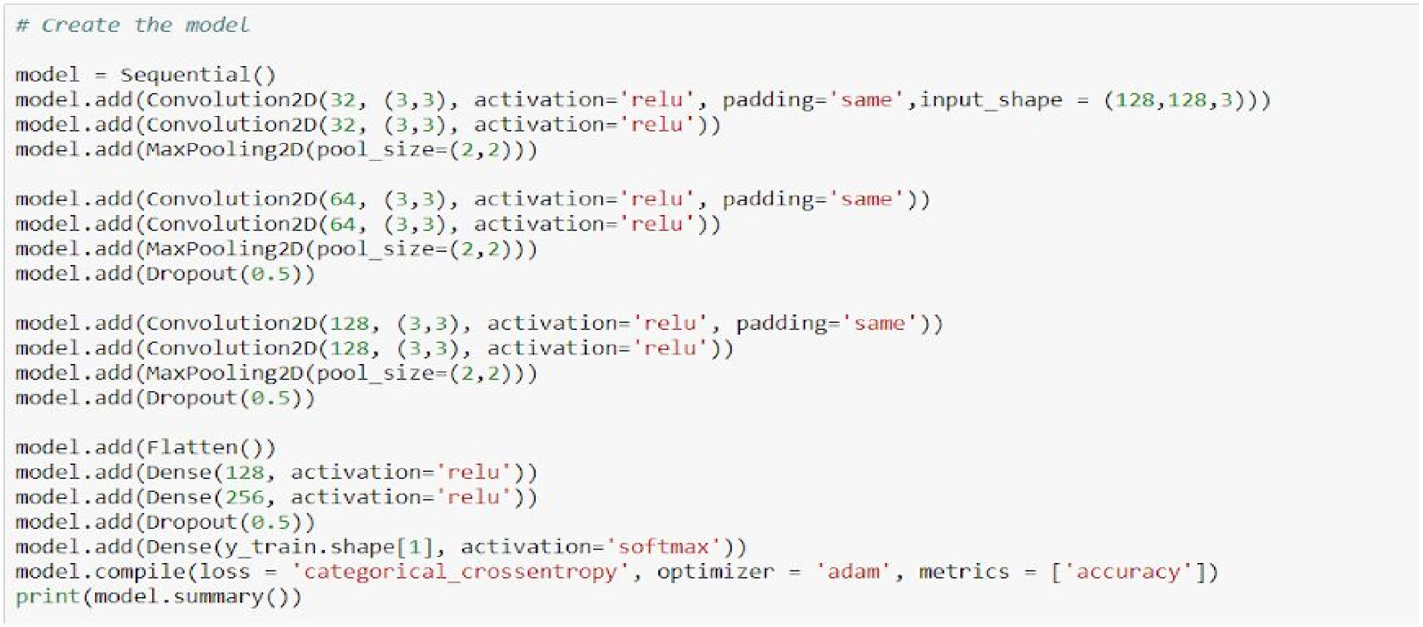
We will use keras module for this problem. Keras is a high-level API written in Python. It runs on top of Tensorflow, CNTK or Theano. It is relatively easy to use and is very fast.

MODEL #1



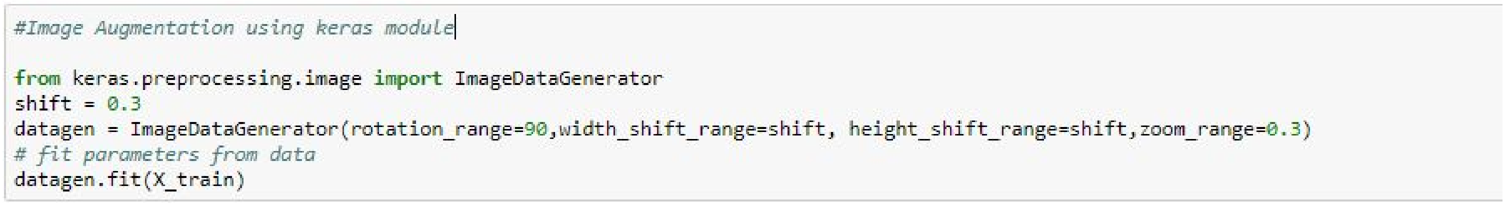
```
model = Sequential()

'''
First set of three layers
Image Size: 128 x 128
nb_filters = 64
kernel_size = 4,4
'''

nb_filters = 32
kernel_size = (2,2)
model.add(Convolution2D(nb_filters, (kernel_size[0], kernel_size[1]),
                 padding='valid',
                 strides=1,
                 input_shape = (128,128,3)))
model.add(Activation('relu'))

model.add(Convolution2D(nb_filters, (kernel_size[0], kernel_size[1])))
model.add(Activation('relu'))

model.add(Convolution2D(nb_filters, (kernel_size[0], kernel_size[1])))
model.add(Activation('relu'))

model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.5))
```

MODEL #2

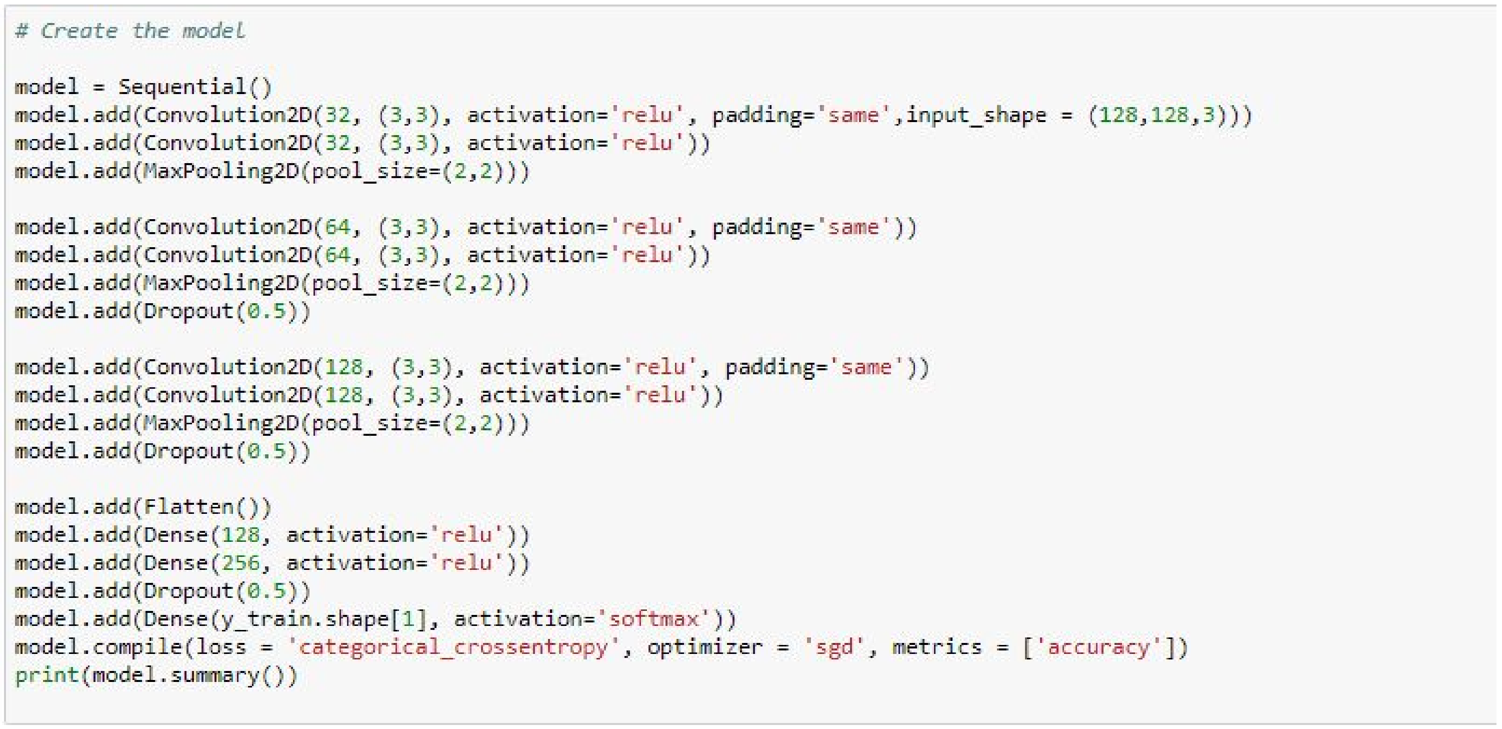


```
# Create the model

model = Sequential()
model.add(Convolution2D(32, (3,3), activation='relu', padding='same',input_shape = (128,128,3)))
model.add(Convolution2D(32, (3,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Convolution2D(64, (3,3), activation='relu', padding='same'))
model.add(Convolution2D(64, (3,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.5))

model.add(Convolution2D(128, (3,3), activation='relu', padding='same'))
model.add(Convolution2D(128, (3,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.5))

model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(y_train.shape[1], activation='softmax'))
model.compile(loss = 'categorical_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
print(model.summary())
```

MODEL #3



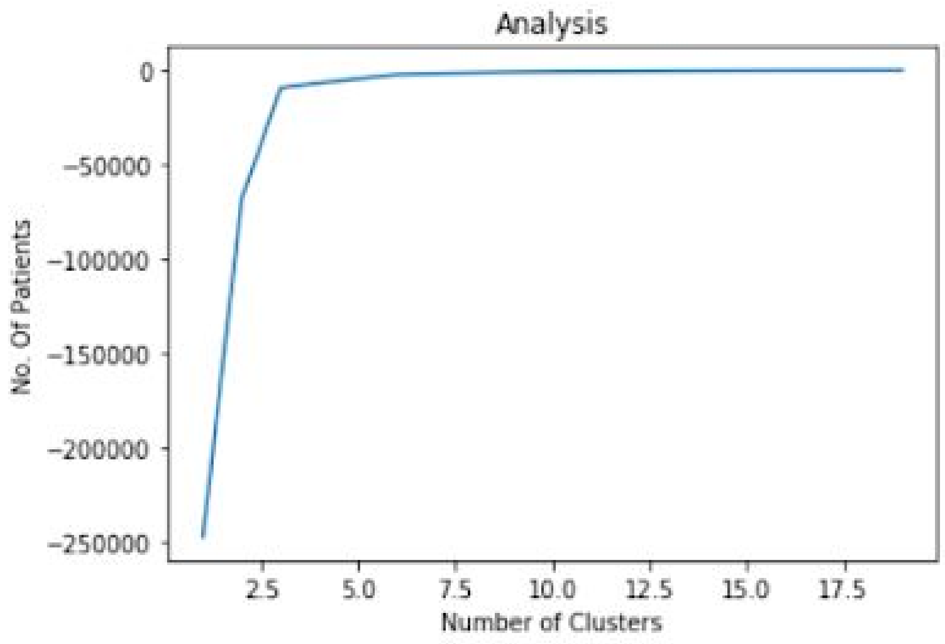
```
#Image Augmentation using keras module

from keras.preprocessing.image import ImageDataGenerator
shift = 0.3
datagen = ImageDataGenerator(rotation_range=90,width_shift_range=shift, height_shift_range=shift,zoom_range=0.3)
# fit parameters from data
datagen.fit(X_train)
```

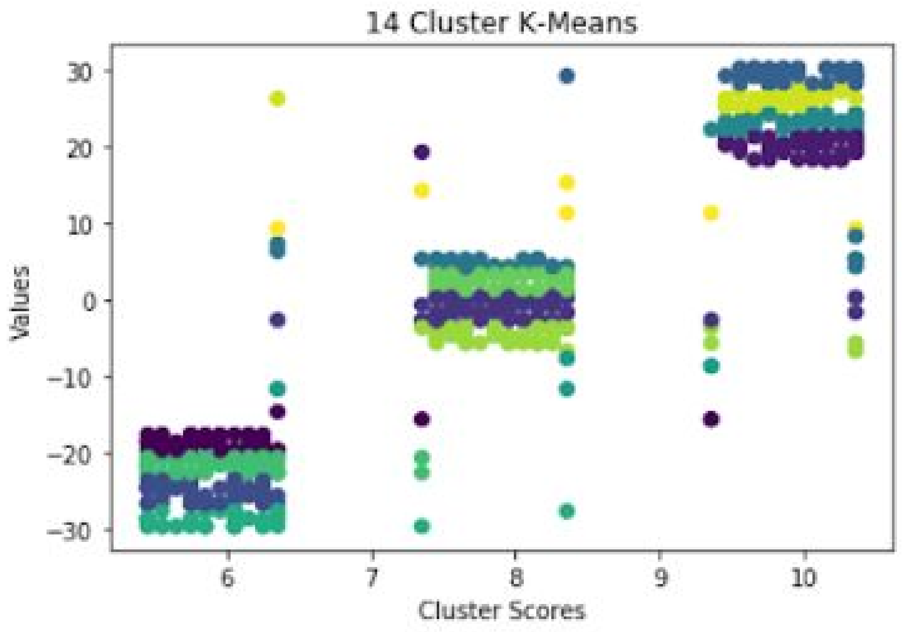


```
# Create the model

model = Sequential()
model.add(Convolution2D(32, (3,3), activation='relu', padding='same',input_shape = (128,128,3)))
model.add(Convolution2D(32, (3,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Convolution2D(64, (3,3), activation='relu', padding='same'))
model.add(Convolution2D(64, (3,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.5))

model.add(Convolution2D(128, (3,3), activation='relu', padding='same'))
model.add(Convolution2D(128, (3,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.5))

model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(y_train.shape[1], activation='softmax'))
model.compile(loss = 'categorical_crossentropy', optimizer = 'sgd', metrics = ['accuracy'])
print(model.summary())
```

Results

Cluster Formation and Classification:





Convolutional Neural Networks:

As shown above different models were trained to arrive at the best model with best accuracy. Each model was varied in terms of complexity of architecture, loss functions, optimizers in terms of **SGD** and **ADAM.** Every model was trained for different number of epochs, with epochs varied based on the model complexity, optimizers and accuracy. We used Google cloud services to run our model.

Performance

The data were split into training and test sets using a 70-30 split. Each CNN model performance was assessed using a held out test set of 30% of our data.

MODEL #1

This model was the most complex model we built. Adaptive optimization(ADAM) was used here as many recent papers say that when we have a complex and a deeper architecture we should make use of adam as it converges fast with high training accuracy and minimum loss. But this is prone to overfitting.On training for 10 epochs we got training accuracy of 32 %, but the test accuracy was very low. We got a test accuracy of 29.92%. This clearly shows that we overfitted.

Batch Size: 128
No. of epochs: 10
Optimizer: Adam
Training accuray: 32 %
Test accuracy: 29.92 %

MODEL #2

The model was relaxed here. We still used Adaptive optimization(ADAM). Our model overfitted, On training for 25 epochs we got training accuracy of 61%, but the test accuracy was very low. We got a test accuracy of 30.27%.

Batch Size: 128
No. of epochs: 25
Optimizer: Adam
Training accuray: 61 %
Test accuracy: 30.27 %

MODEL #3

The architecture of our model remained the same as the previous same. The modification was done in terms of the optimizer. We used the traditional stochastic optimizer. Different articles were referred and they say that sgd under fits even after training for 300 epochs but the advantage with this is that it is much more reliant on a robust initialization and annealing schedule.

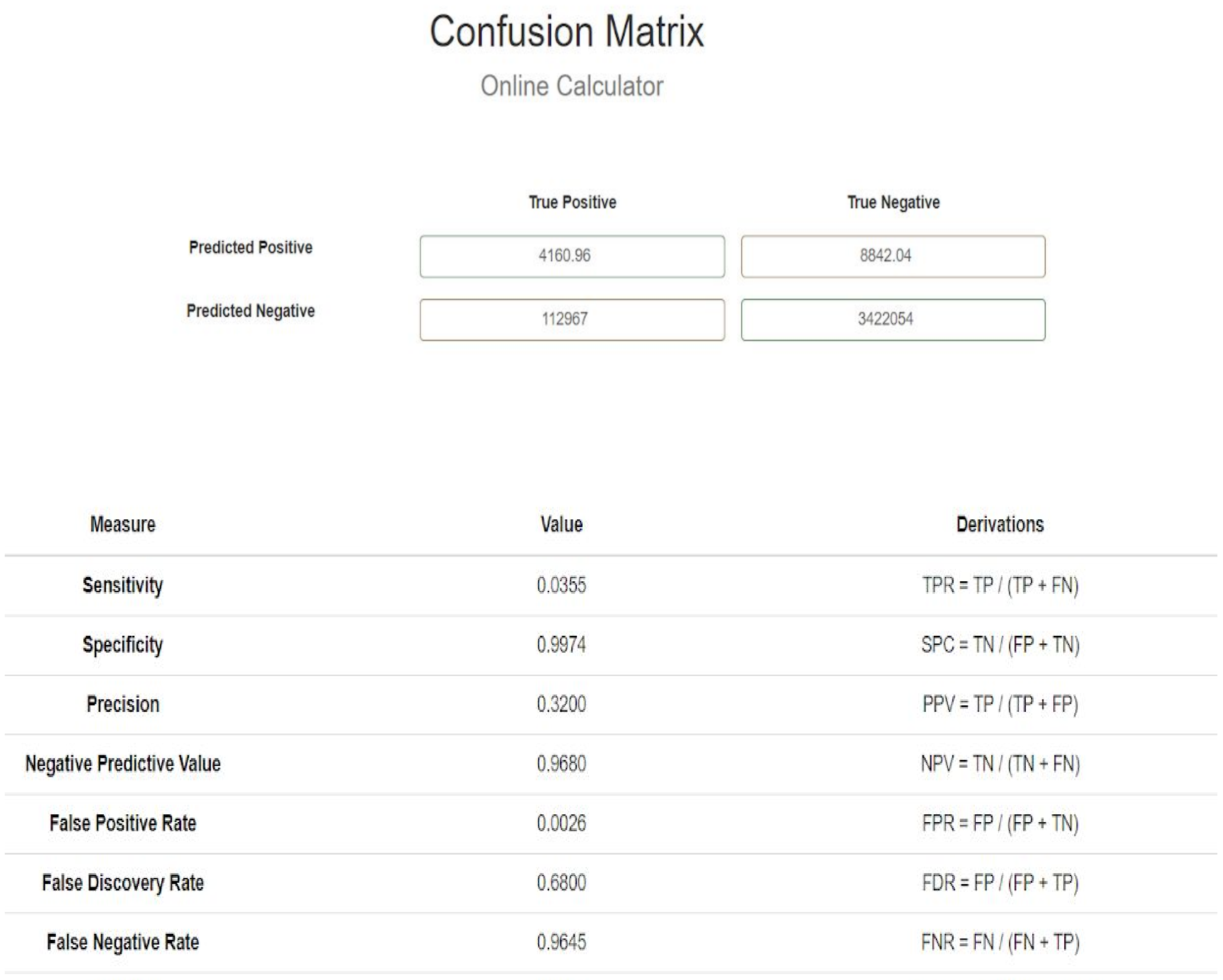
Before feeding the train images to the network we performed image augmentation. Each training image, before being input into a neural network, was **flipped** 90 degrees. Objects in our images may not be centered in the frame. They may be off-center in a variety of different ways. Keras supports separate **horizontal and vertical random shifting** of training data by the width_shift_range and height_shift_range arguments. **zoom_range** specifies range for random zoom.

Using these techniques really helped as our accuracy improved, with no overfitting.

Batch Size: 128
No. of epochs: 30
Optimizer: Sgd
Training accuray: 33.23 %
Test accuracy: 33.72 %

Confusion matrix:

The Matrix will be analyzed based on the 13003 test samples of the 18577 among the batch size of 3 total dataset.
The test dataset is randomly sampled for the model to learn and the matrix generated is as follow:



Confusion Matrix

Online Calculator

	True Positive	True Negative
Predicted Positive	4160.96	8842.04
Predicted Negative	112967	3422054

Measure	Value	Derivations
Sensitivity	0.0355	TPR = TP / (TP + FN)
Specificity	0.9974	SPC = TN / (FP + TN)
Precision	0.3200	PPV = TP / (TP + FP)
Negative Predictive Value	0.9680	NPV = TN / (TN + FN)
False Positive Rate	0.0026	FPR = FP / (FP + TN)
False Discovery Rate	0.6800	FDR = FP / (FP + TP)
False Negative Rate	0.9645	FNR = FN / (FN + TP)

Conclusion

The model is robust to the two image sizes tested(224X224 and 512X512 pixels) and the neural network architecture used. Furthermore, network visualization demonstrates that macroscopic features are learned effectively by the model. In particular, symmetry appears to be a salient feature of normal CXR images detected by the model. As deeper network architectures did not change performance, we expect the same feature to be prominent in InceptionV3 and Residual Network architectures.

The above model may be improved in a number of ways:
1. Increased preprocessing such as cropping lungs from images or cropping edges of the CXR images to highlight the lung regions

2. Weighting of examples such that sensitivity rather than specificity is emphasized, to orient toward clinical usage

3. Integration of level of uncertainty in physician ground truth diagnosis

4. Integration of a segmentation component such that network can learn small, specific features rather than just macroscopic features.

5. Inclusion of more abnormal examples, which are by nature very variant and also less common.

Contrast enhancement using histogram equalization is an effective method, four techniques of histogram processing has been applied on a large number of digital radiographic images.

Reference and citations

[1] Image Processing and Image Enhancement, Dr. William L. Joyner, East Tennessee State University Johnson City, Texas, 1996

[2] Medical Image Analysis Methods, Edited By “Lena Costaridou” Published By “Taylor and Francis Group”

[3] Adaptive Mammographic Image Enhancement Using First Derivative and Local Statistics, "Jong Kook Kim, JeongMi Park, KounSik Song, and Hyun Wook Park", 'IEEE TRANSACTIONS ON MEDICAL IMAGING, VOL. 16, NO. 5, OCTOBER 1997'

[4] Deep Learning for abnormality detection in Chest X-Ray images. Christine Tataru, Darvin Yi Archana Shenoyas, Anthony Ma.

[5] Digital radiographic image enhancement for improved visualization. Nisar Ahmed1, Ahmed2 Sheikh m.Arshad

[6] http://ruder.io/optimizing-gradient-descent/index.html#conclusion

[7] https://machinelearningmastery.com/image-augmentation-deep-learning-keras/