ADVANCED CHEST RADIOGRAPH ANALYSER(ACRA) USING CNN AND COMPARISION WITH TRADIONAL METHODS

Project ID:2020MP091

Review -III

Group Members

RA1611003030203 Shivam Kalra

RA1611003030205 Nitin Srivastava

RA1611003030218 Shakti Jain

RA1611003030251 Yashwanth Kumar

Supervised By:

Dr. Manikandan R.

Professor, Department of Computer Science & Engineering

Table of Contents I

Table of Contents II

Table of Contents III

Table of Contents IV

Abstract

- The uprising problems in the healthcare domain remain a challenge over the years. The unprecedented use of Machine Learning & it's underlying statistics can be used to deal with such problems.
- Use of CNN for classification purposes is recommended due to its embedded feature extraction.
- Ensembling of other feature extraction functions to provide input to native ML algorithms is done and results are compared with CNN model.
- Scope of further improvements through statistical significance and its impact on healthcare is realized.

Literature Survey

- Computed Tomography was introduced to resolve problems in healthcare.[5] [18] [10] [17] [19] [8]
- The model was built to increase detection power and process speed in medical imaging. [14]
- The detection process as per the model requirements needs a set of optimal features.
- To extract image features(textural, structural, etc) models have a wide range of performance. [7] [1] [6] [11]
- Various models are trained to make a substantial performance difference.
- Native ML classifiers and their variants' performance are measured through their statistical output.
- Use of ensemble functions for better feature extraction to get the optimal set of features.
- CNN performance outperforms new variants. [13]
- The underlying architecture of CNN is studied to provide future scope.

Identification of Research Gap and Problem

- The efficient classification of images(whether X-Ray or any kind) requires extensive domain knowledge. [12]
- The accuracy of the predictions in healthcare depends profoundly on the classifier adopted. [14]
- Classifiers here include Neural Network(NN), SVM, Random Forest, and their adopted variants.
- Feature extraction and set of optimal features are key to build an efficient model. [3]
- The obvious choice is CNN outperforms other native Classifiers. [13]
- The existing variants of native classifiers serve no purpose in terms of performance and space.
- That includes ensembling algorithms, methods like bootstrapping, and parameter tuning.
- CNN embedded feature extraction studied thoroughly for its statistical significance. [13] [3]

Expected Impact on Academics/ Industry

- The more accurate modes will reduce the existing aid process time.
- The variants of other classifiers will encourage scholars to explore depths of mathematics.
- The generic development of functions and methods could be fruitful for other domains.
- To further scope may lie in reducing the processing power and training space of the models.
- The real-time implementation of these accurate models will increase the average throughput in healthcare.
- Thriving for improvement it'll encourage others for their part of service.

Major Inputs (Infrastructure) Required

- All the experiments are carried out in python language using Keras and Scikit-learn librariesAll the experiments are carried out in python language using Keras and Scikit-learn libraries
- Keras is deployed with Tensorflow-GPU backend to build and train the CNN model.
- Scikit-learn[15] is used to build and train traditional Machine Learning models.
- PC configuration - Nvidia GTX 1060 GPU unit, CUDA Toolkit 10.0, cuDNN v7.6.5.

Dataset

- The images used are part of original dataset available on Kaggle.com
- Dataset name - “Chest X-Ray Images (Pneumonia)”.
- Consists of three main folders(i.e., training, testing, and validation folders).
- Two subfolders containing pneumonia (P) and normal (N) chest X-ray images.
- A total of 5,856 X-ray images of anterior-posterior chests were carefully chosen.

Sample Pneumonia Images

Figure: Sample Pneumonia Negative Images

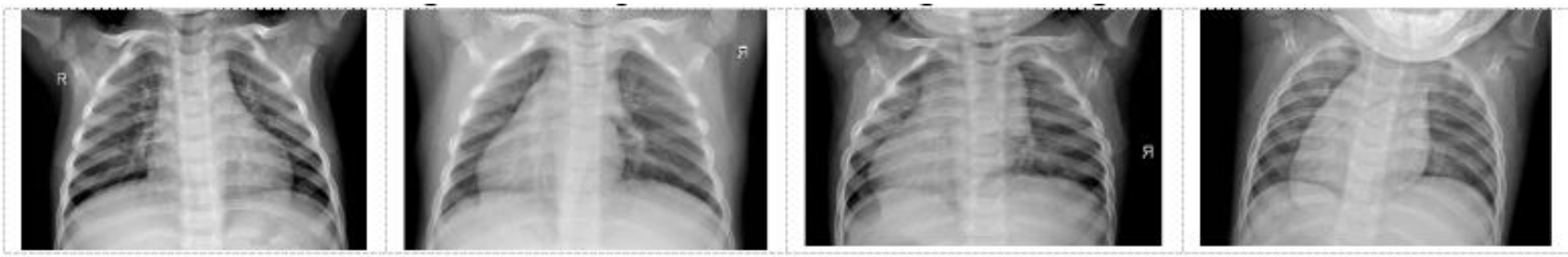
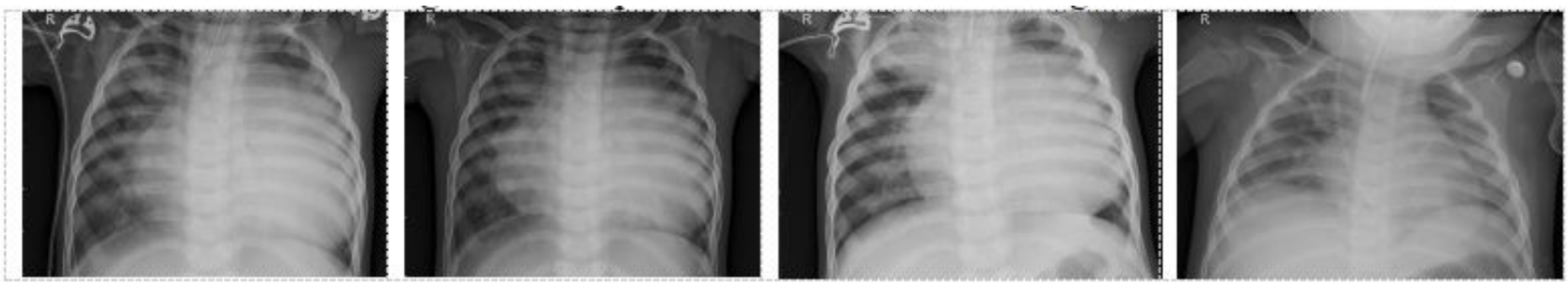


Figure: Sample Pneumonia Positive Images



Pre-processing and Augmentation

- Input images are adjusted to have size 64*64 pixels.
- CNN training requires large set of images that too in every possible scale.
- Several data augmentation methods to artificially increase the size and quality of the dataset.
- Image data so as to provide more images is modified by applying operations like rescaling, sheering, flipping.
- We have used the ImageDataGenerator class of Keras library and passed the parameters "rescale = 1./255, shear_range = 0.2, zoom_range = 0.2, horizontal_flip = True"

Method	Setting
rescale	1./255
shear_range	0.2
zoom_range	0.2
horizontal_flip	True

Feature Extraction

- CNN uses embedded feature extraction and there is no need for manual feature extraction.
- ML models like SVM, Random Forest, and KNN, manual feature extraction is the most crucial.
- X-ray images used are mostly gray level images so there is no need to extract the color features of an image
- We made use of three feature extraction methods for getting the most appropriate knowledge representation features of images, Haralick Textural feature extraction, HOG(Histogram of Oriented Gradients) and Hu-Moments.

Haralick Features I

- Haralick Texture is used to quantify images based on textural properties of blocks of image data.[16]
- Haralick texture features make use of GLCM(Gray Level Co-occurrence Matrix)
- The gray level(pixel values) of the images are evaluated based on their number of co-occurrences in an image.
- Total 14 features are presented in [16] but in this study only 13 features are implemented. The last (14th) feature is normally considered to be unstable as mentioned in [2].
- The textural features that are computed are shown in figure below.

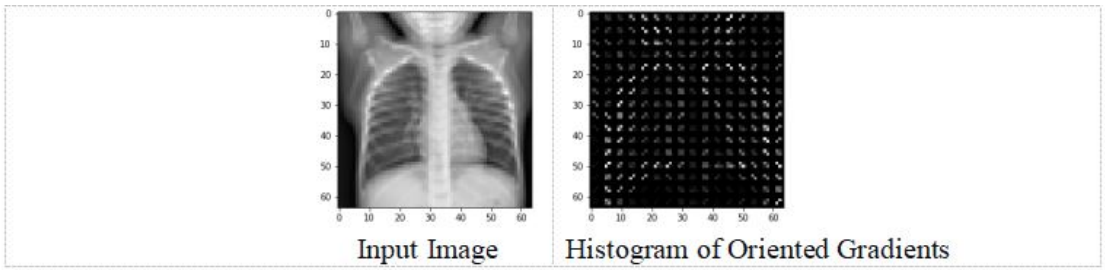
Haralick Features II

Angular Second Moment	$\sum_i \sum_j p(i,j)^2$
Contrast	$\sum_{n=0}^{N_g-1} n^2 \{\sum_{i=1}^{N_g} \sum_{j=1}^{N_g} p(i,j)\}, \lvert i-j \rvert = n$
Correlation	$\frac{\sum_i \sum_j (ij) p(i,j) - \mu_x \mu_y}{\sigma_x \sigma_y}$ where μ_x , μ_y , σ_x , and σ_y are the means and std. deviations of p_x and p_y , the partial probability density functions
Sum of Squares: Variance	$\sum_i \sum_j (i-\mu)^2 p(i,j)$
Inverse Difference Moment	$\sum_i \sum_j \frac{1}{1+(i-j)^2} p(i,j)$
Sum Average	$\sum_{i=2}^{2N_g} i p_{x+y}(i)$ where x and y are the coordinates (row and column) of an entry in the co-occurrence matrix, and $p_{x+y}(i)$ is the probability of co-occurrence matrix coordinates summing to $x+y$
Sum Variance	$\sum_{i=2}^{2N_g} (i - f_8)^2 p_{x+y}(i)$
Sum Entropy	$-\sum_{i=2}^{2N_g} p_{x+y}(i) \log\{p_{x+y}(i)\} = f_8$
Entropy	$-\sum_i \sum_j p(i,j) log(p(i,j))$
Difference Variance	$\sum_{i=0}^{N_g-1} i^2 p_{x-y}(i)$
Difference Entropy	$-\sum_{i=0}^{N_g-1} p_{x-y}(i) \log\{p_{x-y}(i)\}$
Info. Measure of Correlation 1	$\frac{HXY - HXY1}{\max\{HX, HY\}}$
Info. Measure of Correlation 2	$(1 - \exp[-2(HXY2 - HXY)])^{\frac{1}{2}}$ where $HXY = -\sum_i \sum_j p(i,j) \log(p(i,j))$, HX , HY are the entropies of p_x and p_y , $HXY1 = -\sum_i \sum_j p(i,j) \log\{p_x(i) p_y(j)\}$ $HXY2 = -\sum_i \sum_j p_x(i) p_y(j) \log\{p_x(i) p_y(j)\}$

HOG(Histogram of Oriented Gradients)

- Feature Descriptor is a simplified representation of the image that contains only the most important information about the image
- HOG feature descriptor counts the occurrences of gradient orientation in localized portions of an imageto form a histogram and provide a set of features from images.
- HOG features proposed by Navneet Dalal and Bill Triggs[4] are very efficient in extracting structural (shape and edges) information from images.
- A total of 8100 HOG features were extracted from each image in this work.

Figure: Sample HOG Input/output



Process and Parameters used for HOG

1. Input of Pre-processed images that were resized to 64x64 resolution was passed as input one by one.
2. Gradients are the small change in the x and y directions which were calculated separately for each image.
3. Magnitude and gradients were calculated using formulas $\sqrt[2]{(G_x)^2 + (G_y)^2}$ and $\theta = atan(G_y/G_x)$ resp.
4. Histogram of Gradients was created for the image on 9 orientations where "orientations" is referred to the number of buckets we created for the histogram. The size of cells on which the Histograms were created was 4x4 pixels.
5. Normalisation of gradients was carried out on 2x2 cells per block.
6. Final features for the complete image were returned.

HU moments

- Hu moments as proposed by Ming-Kuei Hu [9] is a feature extraction technique.
- Focuses on the recognition of visual patterns and characters independent of position, size, and orientation in the visual field.
- Image moments capture information about the shape of a blob(Binary Large OBject) because they contain information about the intensity I(x,y), as well as position x and y of the pixels.
- Image moments are a weighted average of image pixel intensities
- Central moments are the image moments about the center.
- Hu Moments (or rather Hu moment invariants) are a set of 7 numbers calculated using central moments that are invariant to image transformation.
- The first 6 moments have been proved to be invariant to translation, scale, and rotation, and reflection.
- While the 7th moment's sign changes for image reflection.

Rotation Invariants

Figure: Rotation Invariants

$$I_1 = \eta_{20} + \eta_{02}$$

$$I_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2$$

$$I_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2$$

$$I_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2$$

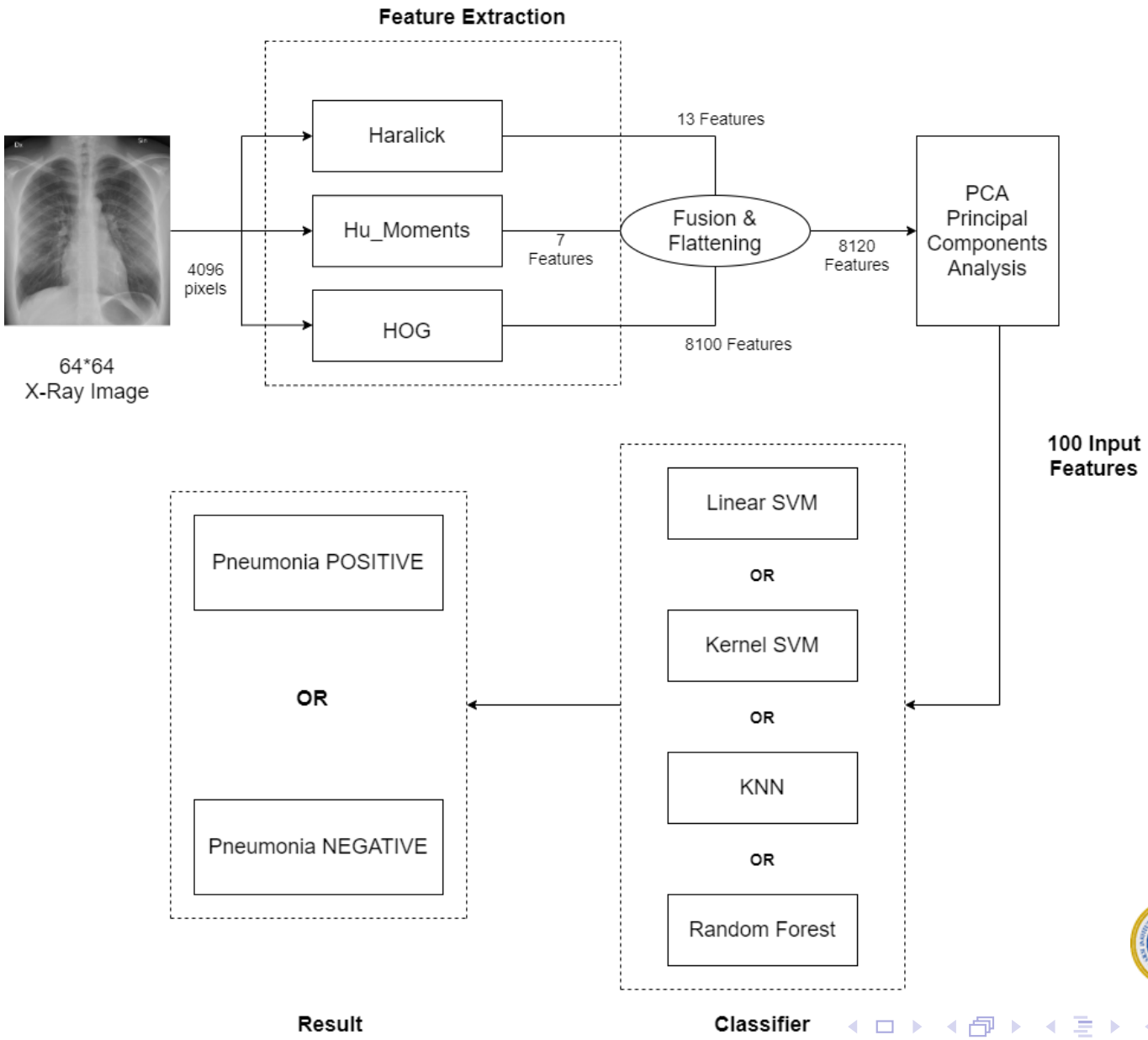
$$I_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$$

$$I_6 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})$$

$$I_7 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] - (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2].$$

- Total of 8120 input features were extracted, 13 of which are Haralick features, 8100 of HOG and 7 of hu_moments
- To avoid overfitting principal component analysis (PCA) was applied to reduce the dimensionality of input data
- 100 principal components were extracted as the final input features.

Classification using Traditional ML Methods(Architecture)



Traditional ML Methods Used

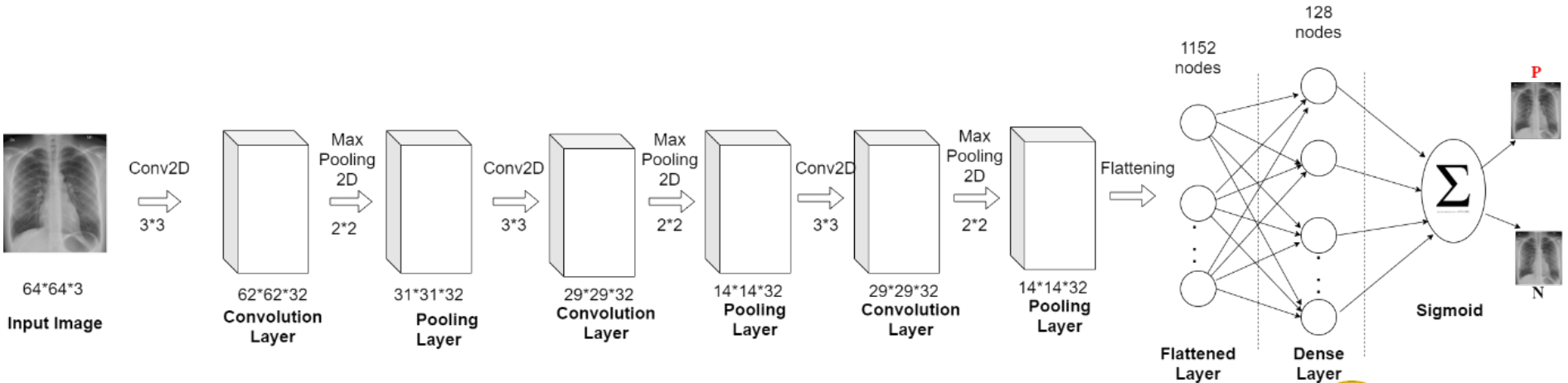
- Traditional machine learning methods that are brought into consideration are Support Vector Machine (both Linear and Kernel SVM), K-Nearest Neighbour (KNN) and Random Forest.
- All these classifiers are designed as binary classifiers and the classification result is either 'Pneumonia Positive' or ' Pneumonia Negative'.
- SVMs (Support Vector Machines): SVMs that are used in this study are linear SVM and Kernel SVM. In both the SVMs the parameter values were C=2 and Gamma='scale'. and the Gaussian RBF Kernel is used in Kernel SVM.

Traditional ML Methods Used(continued ...)

- KNN (K-Nearest Neighbour): For similarity measure 'Miskowski Distance' is used in this study with p=2 that makes it equivalent to standard Euclidian metric and five neighbors are used to classify each data point(i.e n_neighbors= 5).
- Random Forests: It is based on the concept of ensemble learning and is a collection of multiple decision trees.In this experiment, total 10 Decision trees were ensembled to form the Random Forest Classifier model and "Entropy" is used to measure the quality of split to ensure lower homogeneity in splitting.

Convolutional Neural Network (CNN) I

- CNN is a deep learning technique that was first introduced by Fukushima et al.
- The embedded feature extraction performed by convolving feature detectors over the digital image.
- Architecture of the CNN used in the experiment is shown in Figure below.



Convolutional Neural Network (CNN) II

- The pre-processing was performed as already discussed in Pre-processing and Augmentation section.
- Finally the images in 64*64 format were fed to CNN.
- The training network consists of two parts: the feature extractor and classifier.
- The feature extractor comprises of 2 convolutions (Conv2d) layers using 32 numbers of 3x3 filters (feature detectors) and 2 max-pooling layers of size 2×2, and a RELU activator between them.
- The outputs from the convolution and max pooling operations are in the form of 2D planes called feature maps, and we obtained 62x62x32 and 29x29x32 sizes of feature maps respectively, for the convolution operations accompanying 31x31x32 and 14x14x32 sizes of feature maps from the pooling operations respectively.
- After the last max-pooling layer, classifier phase begins.

Convolutional Neural Network (CNN) III

- The 2D plane output from the pooling layer is converted into flat features that are fed to a simple neural network with a dense layer and a Sigmoid function acting as an output node.
- The summary of the CNN model that was trained is shown in table shown below.

Convolutional Neural Network (CNN) IV

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 62, 62, 32)	896
max_pooling2d_1 (MaxPooling2	(None, 31, 31, 32)	0
conv2d_2 (Conv2D)	(None, 29, 29, 32)	9248
max_pooling2d_2 (MaxPooling2	(None, 14, 14, 32)	0
conv2d_3 (Conv2D)	(None, 12, 12, 32)	9248
max_pooling2d_3 (MaxPooling2	(None, 6, 6, 32)	0
flatten_1 (Flatten)	(None, 1152)	0
dense_1 (Dense)	(None, 128)	147584
dense_2 (Dense)	(None, 1)	129

Total params: 167,105
Trainable params: 167,105
Non-trainable params: 0

Results Obtained I

- Accuracy metric is used to measure the correct classification rate.
  Accuracy = (TP+TN) /(TP+TN+FN+FP)
  where TP = true positive, FN = false negative, FP = false positive, and TN = true negative.
- Confusion matrices are calculated to study the behaviour of the classifiers on the Chest X-ray data.

	Class 1 (Positive) Predicted	Class 2 (Negative) Predicted
Class 1 Actual	TP	FN
Class 2 Actual	FP	TN

Results Obtained II

Traditional Machine Learning methods :

After manual feature extraction by Haralick Features, Hu_moments and HOG, the best accuracy results are:-

- Linear SVM: -

Training Accuracy = 72.81%

Validation Accuracy = 62.5%

- Kernel SVM: -

Training Accuracy = 94.99%

Validation Accuracy = 62.5%

- KNN: -

Training Accuracy = 91.73%

Validation Accuracy = 58.01%

- Random Forest: -

Training Accuracy = 97.77%

Validation Accuracy = 61.37%

Results Obtained III

Table: Resulting Confusion Matrices from each classifer

	Training Confusion Matrix		Validation Confusion Matrix	
Linear SVM	0	1341	0	234
	0	3592	0	390
Kernel SVM	1094	247	0	234
	0	3592	0	390
KNN	1006	335	35	199
	73	3519	63	327
Random Forest	1248	93	30	204
	17	3575	37	353

Convolutional Neural Network (CNN) :

Training Accuracy = 95.51%

Validation Accuracy = 92.95%

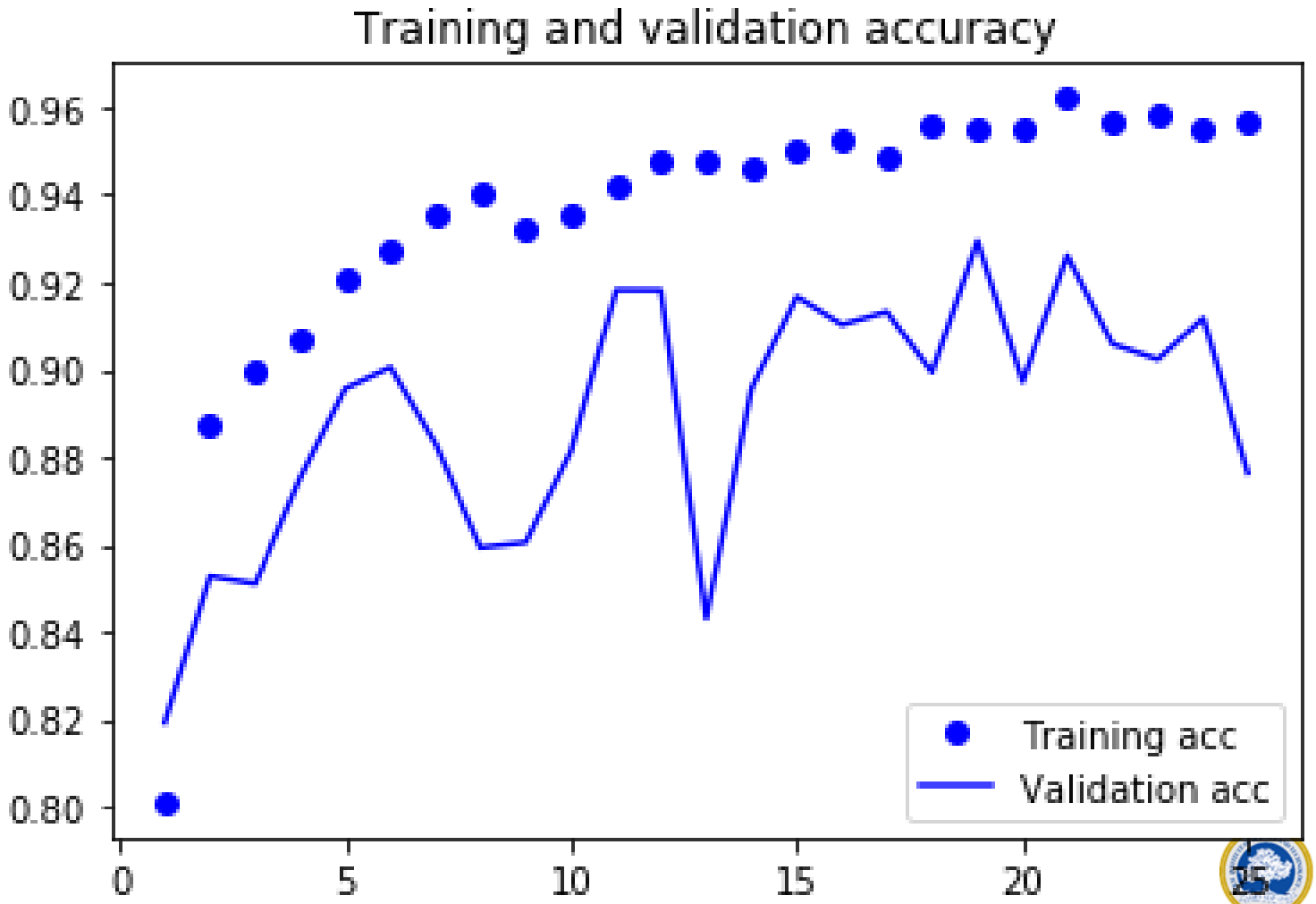
Results Obtained IV

Figure: Output Stats of 25 epoch CNN training process

```
Epoch 1/25
163/163 [==============================] - 335s 2s/step - loss: 0.4392 - ac
curacy: 0.8009 - val_loss: 0.3064 - val_accuracy: 0.8192
Epoch 2/25
163/163 [==============================] - 233s 1s/step - loss: 0.2568 - ac
curacy: 0.8874 - val_loss: 0.1909 - val_accuracy: 0.8526
                                    :
                                    :
                                    :
Epoch 19/25
163/163 [==============================] - 231s 1s/step - loss: 0.1232 - ac
curacy: 0.9551 - val_loss: 0.2360 - val_accuracy: 0.9295
                                    :
                                    :
Epoch 25/25
163/163 [==============================] - 232s 1s/step - loss: 0.1177 - ac
curacy: 0.9566 - val_loss: 0.0730 - val_accuracy: 0.8764
```

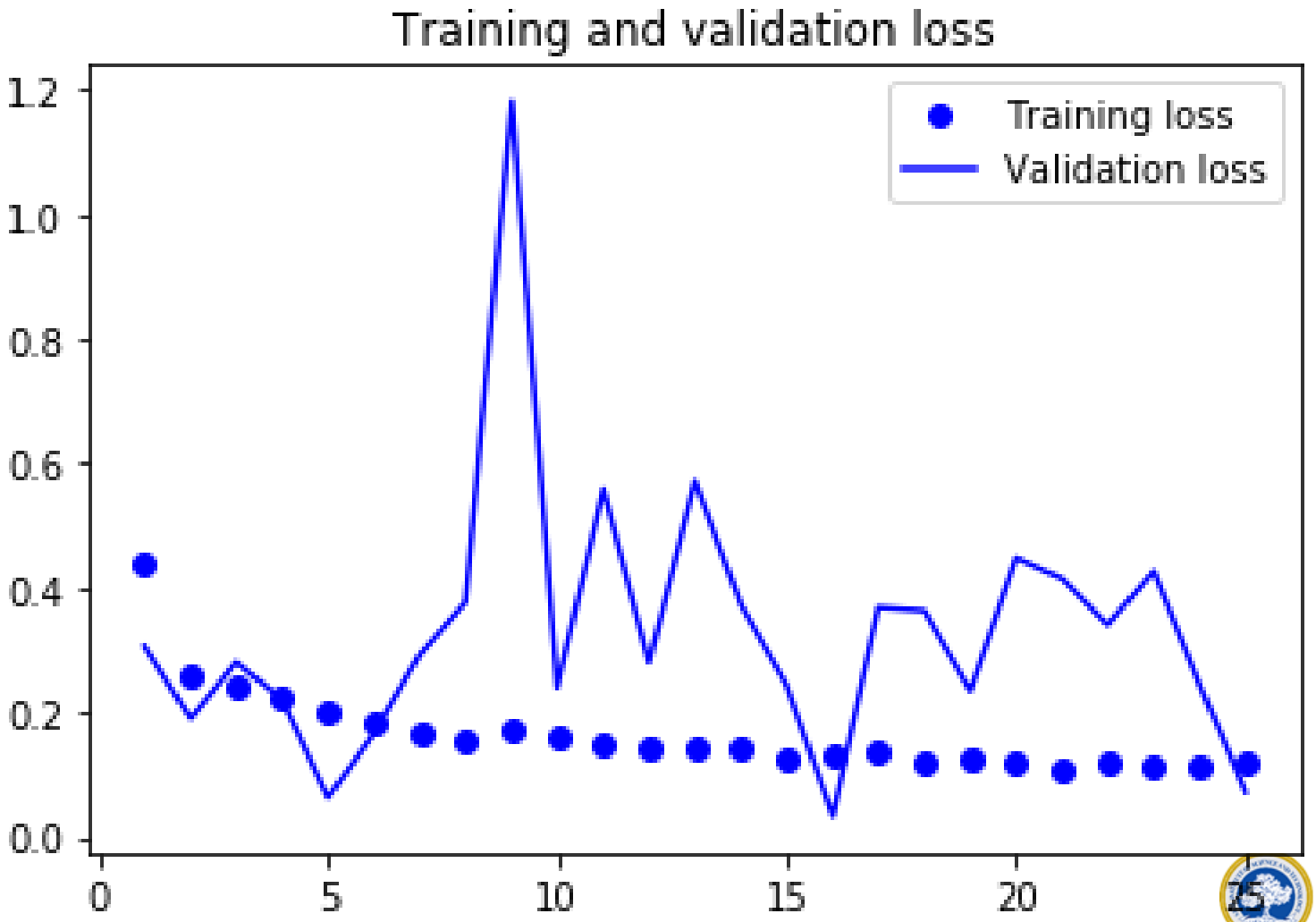
Results Obtained V

Figure: Graphical Representation of Accuracy variation in different epochs



Results Obtained VI

Figure: Graphical Representation of loss variation in different epochs

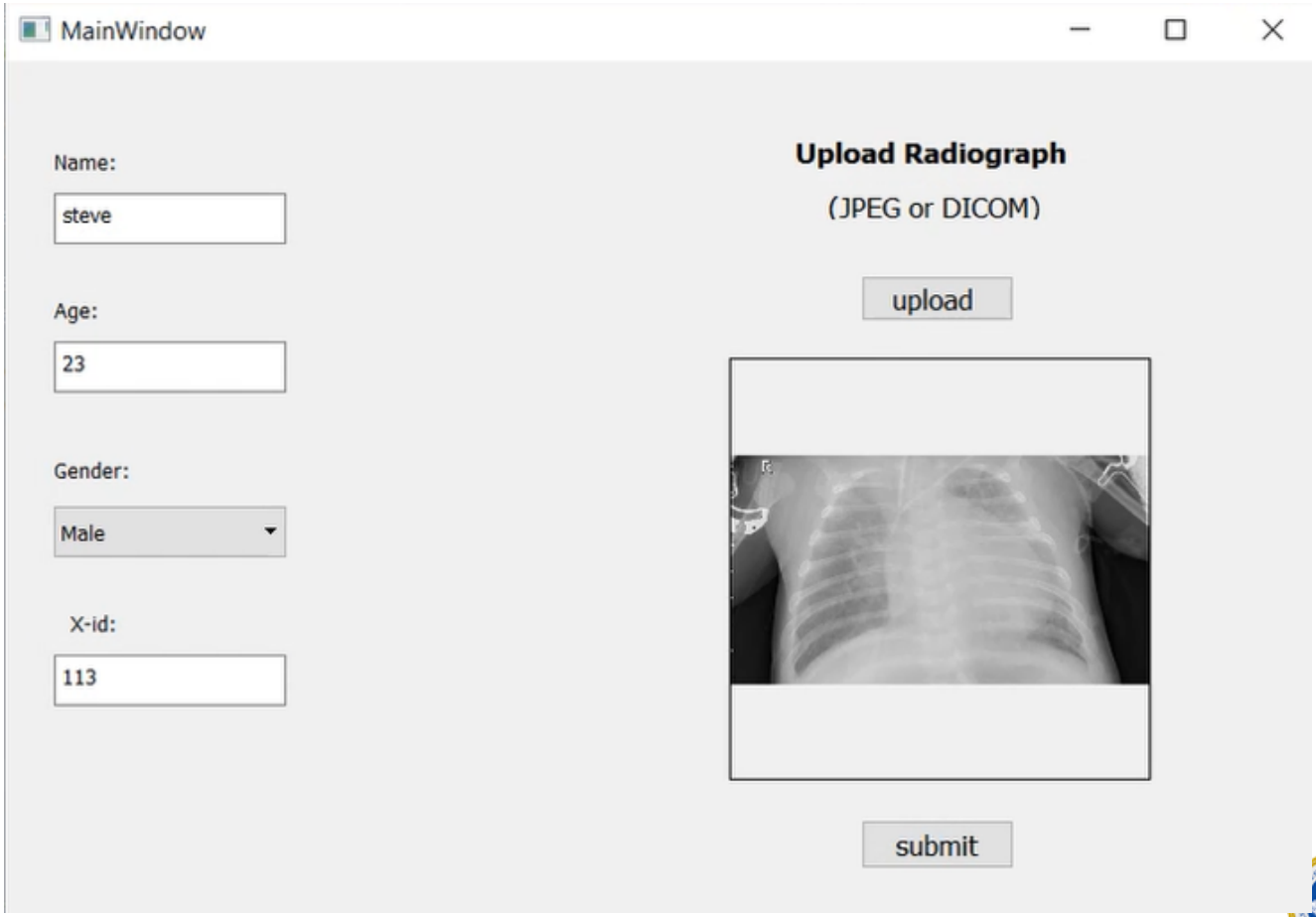


Result VII

- CNN outstands all the other traditional Machine Learning techniques used in this study to detect Pneumonia from Chest X-ray images.
- To affirm the performance of the model, we repeated the training process of the models several times, each time obtaining the same results.
- A User Interface is also created using PyQT Framework of python for Chest X-ray classification
- The Screenshots of the User Interface are shown in following slides.

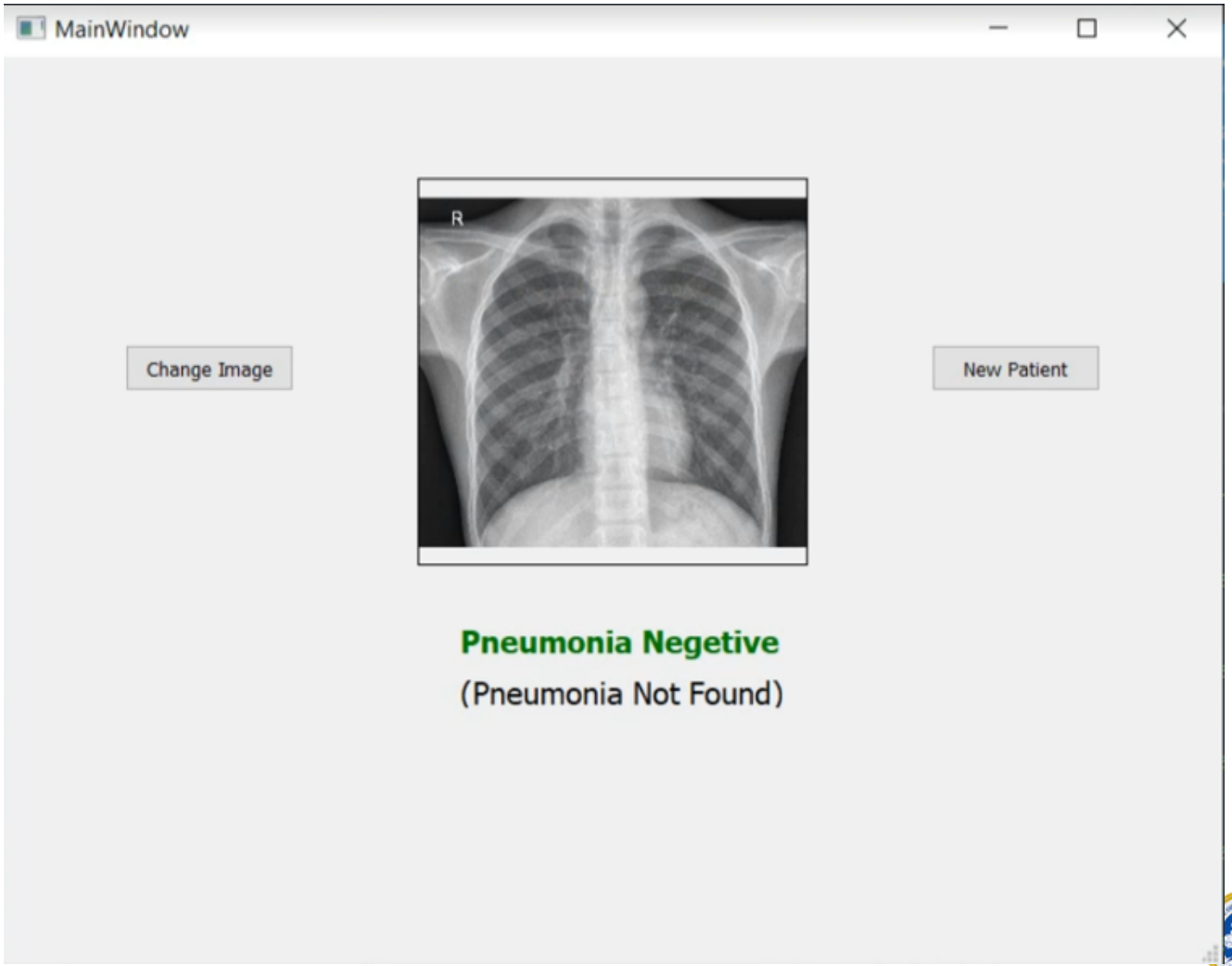
User Interface I

Figure: Screen 1 of User Interface



User Interface II

Figure: Screen 2 of User Interface



Implementation/Coding I

Figures below shows the libraries used for training and testing CNN and traditional machine learning models respectively.

```
# Importing Keras libraries and packages
from keras.models import Sequential
from keras.layers import Convolution2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
from keras.layers import Dense
from keras.preprocessing.image import ImageDataGenerator
```

Implementation/Coding II

```
import os

import matplotlib as mpl
import matplotlib.pyplot as plt

import cv2
import pandas as pd
import numpy as np
import mahotas

from PIL import Image,ImageOps

from skimage.feature import hog
from skimage.color import rgb2grey

from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.svm import SVC

from sklearn.metrics import roc_curve, auc
```

Below Figures shows the code used for creating and compiling the Convolutional Neural Network.

Implementation/Coding III

Initialising the CNN

```
In [2]: classifier = Sequential()
```

Step01 - Convolution

```
In [3]: classifier.add(Convolution2D(32, (3, 3), input_shape=(64,64,3), activation='relu'))
```

Step02 - Pooling

```
In [4]: classifier.add(MaxPooling2D(pool_size=(2,2)))
```

```
WARNING:tensorflow:From D:\Users\shiva\Anaconda3\envs\tensorflow\lib\site-packages\keras\backend\tensorflow_backend.py:4070: Th
e name tf.nn.max_pool is deprecated. Please use tf.nn.max_pool2d instead.
```

Adding more Convolutional Layer

```
In [5]: classifier.add(Convolution2D(32, (3, 3), activation='relu'))
        classifier.add(MaxPooling2D(pool_size=(2,2)))
        classifier.add(Convolution2D(32, (3, 3), activation='relu'))
        classifier.add(MaxPooling2D(pool_size=(2,2)))
```

Step03 - Flattening

```
In [7]: classifier.add(Flatten())
```

Step04 - Full Connection

```
In [8]: classifier.add(Dense(output_dim=128, activation='relu'))
        classifier.add(Dense(output_dim=1, activation='sigmoid'))
```

Compiling the CNN

```
In [9]: classifier.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

SRM INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be University u/s 3 of UGC Act, 1956

Implementation/Coding IV

Figure below shows how the image augmentation is done and the preprocessed and augmented images are provided in 64*64 size.

Implementation/Coding V

Part02 - Fitting the CNN to the images

In [10]:
```
from keras.preprocessing.image import ImageDataGenerator

#Below pixels are rescaled to have value b/w 0 and 1 and different modifications are performed
train_datagen = ImageDataGenerator(
        rescale=1./255,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True)

#Here only rescaling is done
test_datagen = ImageDataGenerator(rescale=1./255)
```

Loading Data

In [11]:
```
train_generator = train_datagen.flow_from_directory('train',
                                                    target_size=(64, 64),
                                                    batch_size=32,
                                                    class_mode='binary')

validation_generator = test_datagen.flow_from_directory('test',
                                                        target_size=(64, 64),
                                                        batch_size=32,
                                                        class_mode='binary')
```

```
Found 4933 images belonging to 2 classes.
Found 624 images belonging to 2 classes.
```

Configuring Checkpointer

In [12]:
```
checkpointer = ModelCheckpoint(filepath="best_weights.hdf5",
                               monitor = 'val_acc',
                               verbose=1,
                               save_best_only=True)
```

Fitting the CNN to data

In [13]:
```
history = classifier.fit_generator(train_generator,
                        samples_per_epoch=5216,
                        nb_epoch=25,
                        validation_data=validation_generator,
                        nb_val_samples=624)
```

SRM INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be University u/s 3 of UGC Act, 1956

Implementation/Coding VI

Follwing figures depicts the image preprocessing, feature extraction and other stages involved in training and testing the tradional machine learning methods.

Feature Extraction Functions

In [2]:
```
def fd_hu_moments(image):
    feature = cv2.HuMoments(cv2.moments(image)).flatten()
    return feature

def fd_haralick(image):
    # compute the haralick texture feature vector
    haralick = mahotas.features.haralick(image).mean(axis=0)
    return haralick

def create_features(img):
    # flatten three channel color image
    #color_features = img.flatten()
    # convert image to greyscale
    grey_image = rgb2grey(img)
    # get HOG features from greyscale image
    hog_features,hog_imgs = hog(grey_image, orientations=9, pixels_per_cell=(4, 4),cells_per_block=(2, 2),visualize=True)  #got i
    # combine all features into a single array
    hu_features = fd_hu_moments(img)  #Total 7 features
    haralick_features = fd_haralick(img)  #Total 13 features
    #print('no. of features => hog=',len(hog_features),', hu=',len(hu_features),', haralick=',len(haralick_features))
    flat_features = np.hstack([hog_features,hu_features,haralick_features])
    return flat_features
```

Implementation/Coding VII

In [16]:
```
# Doing same operations to get train2(Pneumonia part)
train2 = getDF('.\\train\\PNEUMONIA\\')
pca = PCA(n_components=100)
train2 = pca.fit_transform(train2)
ss = StandardScaler()
train2 = ss.fit_transform(train2)
```

In [17]:
```
train2.shape
```

Out[17]:
```
(3592, 100)
```

In [18]:
```
label2 = np.array(['pneumonia']*3592).reshape(-1,1) #3592 is no. of rows in train2
train2 = np.concatenate((train2,label2),axis=1)
```

In [19]:
```
train2.shape
```

Out[19]:
```
(3592, 101)
```

In [20]:
```
train_set = np.concatenate((train1,train2),axis=0)
```

In [21]:
```
train_set
```

Out[21]:
```
array([['0.6864043909191616', '-1.1578634753077777',
        '-0.14978184392678884', ..., '-0.266531406013027',
        '-0.09254903651618077', 'normal'],
       ['0.01234082264659507', '0.8260274065765886',
```

Implementation/Coding VIII

Figure: Training and Testing the Kernel SVM.

Training a Kernel SVM model

In [134]:
```
classifier = SVC(C=2,kernel = 'rbf',verbose=True)
classifier.fit(X_train, Y_train)
```

```
[LibSVM]
```

Out[134]:
```
SVC(C=2, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=True)
```

Making prediction on test set

In [135]:
```
Y_pred = classifier.predict(X_test)
Y_predTrain = classifier.predict(X_train)
```

Evaluating Results

In [136]:
```
from sklearn.metrics import accuracy_score
accuracy_score(Y_train, Y_predTrain)  #Training Accuaracy
```

Out[136]:
```
0.9499290492600851
```

In [137]:
```
accuracy_score(Y_test, Y_pred)  #Testing Accuracy
```

Out[137]:
```
0.625
```

In [138]:
```
from sklearn.metrics import confusion_matrix
pd.DataFrame(confusion_matrix(Y_test, Y_pred))
```

Out[138]:

	0	1
0	0	234
1	0	390

In [139]:
```
pd.DataFrame(confusion_matrix(Y_train, Y_predTrain))
```

Implementation/Coding IX

Figure: Training and Testing the Linear SVM.

Training a linear SVM model

In [140]:
```
classifier = SVC(C=2,kernel = 'linear')
classifier.fit(X_train, Y_train)
```

Out[140]:
```
SVC(C=2, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

Making prediction on test set

In [141]:
```
Y_pred = classifier.predict(X_test)
Y_predTrain = classifier.predict(X_train)
```

Evaluating Results

In [142]:
```
from sklearn.metrics import accuracy_score
accuracy_score(Y_train, Y_predTrain)  #Training Accuaracy
```

Out[142]:
```
0.7281573079262113
```

In [143]:
```
accuracy_score(Y_test, Y_pred)  #Testing Accuracy
```

Out[143]:
```
0.625
```

In [145]:
```
from sklearn.metrics import confusion_matrix
pd.DataFrame(confusion_matrix(Y_test, Y_pred))
```

Out[145]:

	0	1
0	0	234
1	0	390

In [146]:
```
pd.DataFrame(confusion_matrix(Y_train, Y_predTrain))
```

Out[146]:

	0	1

Implementation/Coding X

Figure: Training and Testing the KNN.

Training a KNN model

In [147]:
```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
classifier.fit(X_train, Y_train)
```

Out[147]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                     weights='uniform')
```

Making prediction on test set

In [148]:
```
Y_pred = classifier.predict(X_test)
Y_predTrain = classifier.predict(X_train)
```

Evaluating Results

In [149]:
```
from sklearn.metrics import accuracy_score
accuracy_score(Y_train, Y_predTrain)  #Training Accuaracy
```

Out[149]:
```
0.9172917088992499
```

In [150]:
```
accuracy_score(Y_test, Y_pred)  #Testing Accuracy
```

Out[150]:
```
0.5801282051282052
```

In [151]:
```
from sklearn.metrics import confusion_matrix
pd.DataFrame(confusion_matrix(Y_test, Y_pred))
```

Out[151]:

	0	1
0	35	199
1	63	327

In [152]:
```
pd.DataFrame(confusion_matrix(Y_train, Y_predTrain))
```

Out[152]:

	0	1

Implementation/Coding XI

Figure: Training and Testing the Random Forest Classifier.

Training a Random Forest Model

In [28]:
```
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators = 7, criterion = 'entropy',verbose=1)
classifier.fit(X_train,Y_train)
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   7 out of   7 | elapsed:    0.5s finished
```

Out[28]:
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='entropy', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=7,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=1, warm_start=False)
```

Making prediction on test set

In [29]:
```
Y_pred = classifier.predict(X_test)
Y_predTrain = classifier.predict(X_train)
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   7 out of   7 | elapsed:    0.0s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   7 out of   7 | elapsed:    0.0s finished
```

Evaluating Results

In [30]:
```
accuracy_score(Y_train, Y_predTrain)
```

Out[30]:
```
0.9777011960267585
```

In [31]:
```
accuracy_score(Y_test, Y_pred)
```

Out[31]:
```
0.6137820512820513
```

In [32]:
```
from sklearn.metrics import confusion matrix
```

References I


R. Zainuddin A. Mueen, M. Sapian Baba.
multilevel feature extraction and x-ray image classification.
journal of applied science, 2007.


L.P. Coelho.
Mahotas: Open source software for scriptable computer vision.
Journal of Open Research Software, 2013.
http://dx.doi.org/10.5334/jors.ac.


S. Gopi C.V. Hari, J.V. Joseph, V.P. Felix, and J. Amudha.
Mid-point hough transform: A fast line detection method.
Annual IEEE India Conference, 2009.


Navneet Dalal and Bill Triggs.
Histogram oriented gradients for human detection.
IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005.

References II

I. Kokkinos H. Boussaid.
Fast and exact: Admmbased discriminative shape segmentation with loopy part models.
Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR), 2014.

A.T. Pinhas H. Greenspan.
medical image categorization and retrieval for pacs using gmmkl framework.
IEEE transaction on Information Technology in Biomedicine, 2007.

H. Ghasemian H. Pourghasem.
content-based medical image classification using a new hierarchical merging scheme.
Computerized Medical Imaging and Graphics.

References III

S. Hermann.
Evaluation of scan-line optimization for 3d medical image registration.
Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR), 2014.

Ming-Kuei Hu.
Visual pattern recognition by moment invariants.
IRE Transactions on Information Theory, 1962.

P. Maduskar J. Melendez, G. B. Van.
A novel multipleinstance learning-based approach to computer-aideddetection of tuberculosis on chest x-ray.
IEEE Transactions on Medical Imaging, 2015.

References IV


P. Bhattacharya M. mahmudur Rahman, B. C. Desai.
Medical image retrieval with probabilistic multiclass support vector machine classifiers and adaptive similarity fusion.
Computerized Medical Imaging and Graphics, 2007.


C. Krishna Mohan M. Srinivas, Debaditya Roy.
Discriminative feature extraction from x-ray images using deep convolutional neural networks.
2016.


Uchenna Joseph Maduh Okeke Stephen, Mangal Sain and Do-Un Jeong.
An efficient deep learning approach to pneumonia classification in healthcare.
2019.

References V

D.C. Park.
Image classification using partitioned-feature based classifier model.
Systems and Applications (AICCSA), IEEE/ACS, 2010.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay.
Scikit-learn: Machine learning in Python.
Journal of Machine Learning Research, 12:2825–2830, 2011.

K. Shnmugam Robert M. Haralick and Its'hak Dinsten.
Texural features for image classification.
IEEE Transactions on System, Man, and Cybernetics, 1973.

References VI


S. Candemir S. Jaeger, A. Karargyris.
Automatic tuberculosis screening using chest radio-graphs.
IEEE Transactions.


E. Konen et al. U. Avni, H. Greenspan.
X-ray categorization and retrieval on the organ and pathology level, using patch-based visual words.
Med Imaging, IEEE Transactions, 2011.


S. Candemir Z. Xue, D. You.
Chest x-ray image view classification.
Proceedings of the Computer-Based Medical Systems IEEE 28th International Symposium, 2015.