iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 8 **Issue:** V **Month of publication:** May 2020

DOI: http://doi.org/10.22214/ijraset.2020.5168

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 8 Issue V May 2020- Available at www.ijraset.com

Accelerated Diagnosis and Reporting of Patients using Analysis of Bulk Chest X-ray Images to Aid Impacted Healthcare System during Covid19

Mohammed Azam Sayeed[1], Sakeena Fiza[2], Noor Ayesha[3], Mohammed Akram Sayeed[4]
[1]*Department of Computer sciences, SET Jain University Bangalore -562112*
[2]*UG Student at UC Irvine, California*

Abstract: ***This paper proposes Machine Learning based Methodology to assist Health staff to perform bulk reporting of patients on Chest X-ray images into Normal or Pneumonia diseased clusters which will be of assistance to currently overburdened health workers and possibly detect potential covid19 infected patients as pneumonia is known symptom of covid19. Also this paper demonstrates creating high accuracy models trained on existing clustered data capable of accurately predicting pneumonia in patients.***

Keywords*: Covid19, overburdened, Health workers, X-ray images, Normal, Pneumonia, Kmeans, PCA, Pickle, Bulk reporting, SVM, CNN, Tensorflow*

I. OVERBURDENED MEDICAL STAFF DURING CORONAVIRUS PANDEMIC

The World Health Organization (WHO) categorized the outbreak of Coronavirus Disease 2019 (COVID-19) as a pandemic on March 11, 2020. The disease "caused by severe acute respiratory syndrome-coronavirus 2 (SARS-CoV-2) has clear potential for a long-lasting global pandemic, high fatality rates, and incapacitated health systems."(Ferretti et al.). The popular symptoms are fever, cough, breathlessness, fatigue, cough, sore throat, breathlessness, and malaise (Singhal281). Though mild in most people, the disease can lead to "pneumonia, acute respiratory distress syndrome(ARDS) and multi-organ dysfunction" (Singhal 281). Similar pulmonary syndromes were commonplace in other strains of the coronavirus: the Middle East respiratory syndrome (MERS) and severe acute respiratory syndrome (SARS).

A review of the imaging studies employed during previous outbreaks will help determine the importance of radiologists in combatting the outbreak of COVID-19 (Hosseiny 1078). The SARS virus has been successfully contained with no human infection reported since 2003. Hence, it is vital to learn from such past outbreaks to manage and contain the escalating spread of COVID-19. Hosseiny et al. believe "imaging is a critical component of the diagnostic work-up, monitoring of disease progression, and follow-up in coronavirus-related pulmonary syndromes" which would be beneficial considering the overlap of clinical and radiological features of SARS, MERS, and COVID-19(1078).

Individuals are diagnosed with COVID-19 based on symptoms of pneumonia, recent travel history, or exposure to a known patient. "Chest imaging plays an important role in both assessments of disease extent and follow up" which can help distinguish between a mild and severe manifestation of the symptoms (Hosseiny 1080).

"A patient with severe disease from COVID-19 require a mean of approximately 13 days of respiratory support", such lengthy times will further stress the resources of a healthcare system already stretched beyond capacity (Cavallo). "Case growth rates of 25% to 35%" are expected in affected regions, this entails deadly cues in the intensive care units, lack of ventilators, and sufficient drug supplies, depleting reserves of personal protective equipment, resulting in a healthcare system ill-equipped for a pandemic (Cavallo).

In response to the highly impacted healthcare system fighting the COVID-19 outbreak, this paper proposes a system that will produce bulk X-rays categorizing patients as healthy or having pneumonia, which is a definitive symptom of COVID-19. This will reduce the burden of hospitals as it will prove to be a more effective approach. Such intensive testing will help mitigate the spread as shown by South Korea, Taiwan, and Singapore who employed contact tracing and extreme vigilance to curb the outbreak. This proposed method of testing will assist hospitals to use X-ray machines to analyze the health of a patient's lungs quickly assess the overall patients conditions and assist in doctor for prescription and potentially Identify and isolate the coronavirus victims and hence slow the spread of the virus (Rosebrock).

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 8 Issue V May 2020- Available at www.ijraset.com

II. PROPOSED METHODOLOGY

The Generalized Methodology can be briefly depicted as Fig 1. We are using Kmeans clustering on Bulk Chest x-ray Images after performing image pre processing , we obtain clustered labels for the data for no of clusters as 2. The Obtained clustered labels is then used for bulk reporting of patient that are normal/healthy lungs and ones which are pneumonia diseased patients. We also can use the clustered datapoints to model SVN and Tensorflow by reviewing by doctors to further improving accuracy and for predicting future bulk patient x-ray images. Steps of the proposed methodology is given as follows:

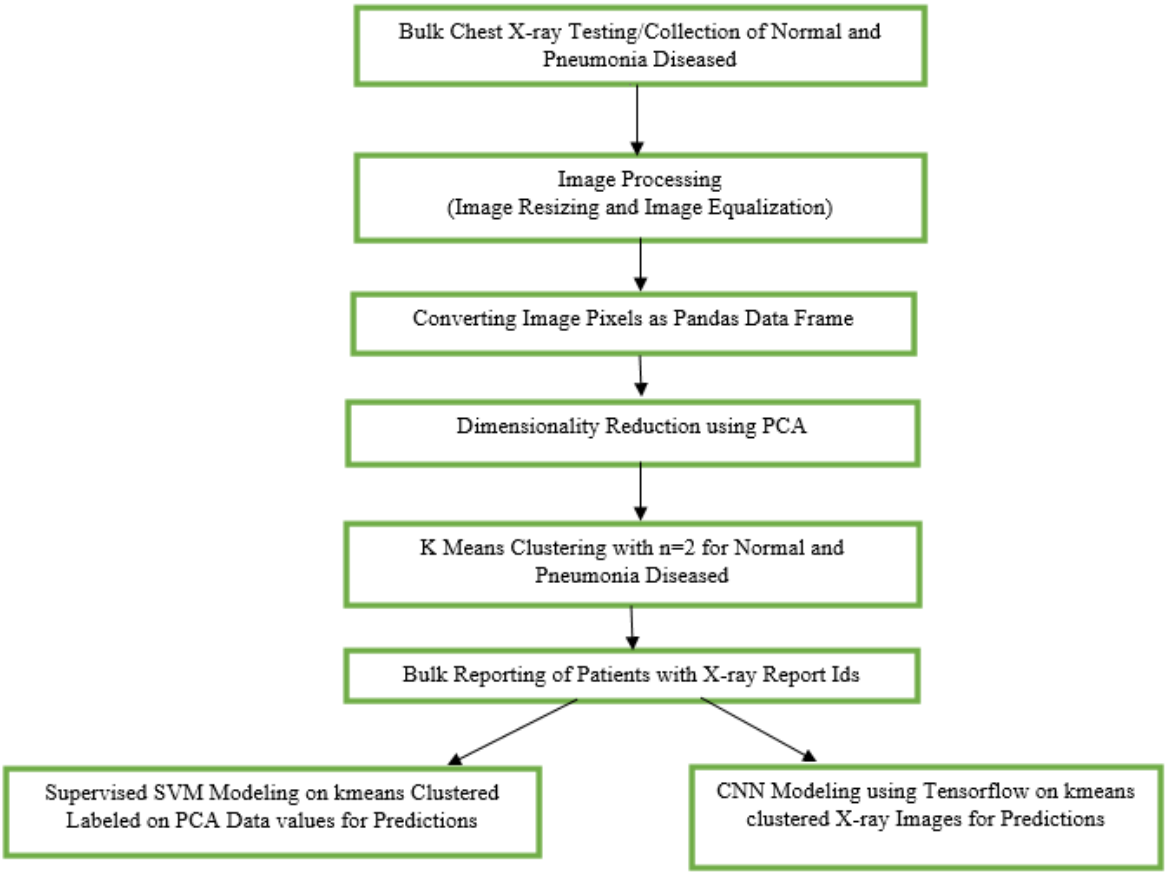


Fig. 1 Generalized Methodology for Bulk Reporting and Modelling

A. Bulk Chest X-ray Images of Normal and Pneumonia Patients

To stimulate Real time data for Proposed Algorithm for Bulk reporting and Modeling of Patients as Normal or Pneumonia infected we are using the Infamous kaggle dataset https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia , This Dataset is widely used in research purposes due its abundance of Images and for high resolution of images. For this Paper we are using 25 images of Healthy X-ray Patient images and 30 Pneumonia infected X-ray Images.

Fig. 2 Sample X-ray Images of Healthy(left) and Pneumonia infected(right)

We have imported various packages for the proposed Methodology mainly skimage used for image reading and for image processing, we have default used packages included such as matplotlib, numpy, pandas etc. we are also using sklearn for metrics, kmeans clustering , PCA dimensionality reduction and SVM Modeling. Also Random and Pickle are also imcluded which is used in algorithm.

```
from time import time
import numpy as np
from matplotlib import pyplot as plt
from sklearn import metrics
from sklearn.decomposition import PCA
import seaborn as sns
import skimage.io
from skimage.transform import rescale, resize
import pandas as pd
from skimage import exposure
from sklearn.cluster import KMeans
from sklearn.metrics import accuracy_score
from sklearn import svm
import random
import pickle
```

Fig. 3 Importing various packages for Proposed Methodology

B. Image Processing

1) *Reading Images from File System:* We are using sklearn.io.imread_collection for bulk reading of images from filesystem, this makes it convenient as images need not to be read individually. As given in Fig 4 , img holds the collection of images fetched based on specified filesystem and individual image from the collection is accessed through indexing such as imgs[0] for first Normal patient X-ray image present in file system.

```
imgs=skimage.io.imread_collection("chest-xray-Project/Normal/*.jpeg")
imgs2=skimage.io.imread_collection("chest-xray-Project/Pneumonia/*.jpeg")
```

```
fig, axes = plt.subplots(nrows=1, ncols=2)
ax = axes.ravel()
ax[0].imshow(imgs[0], cmap='gray')
ax[0].set_title("Normal Xray image")

ax[1].imshow(imgs2[0], cmap='gray')
ax[1].set_title("Pneumonia Xray image")
plt.tight_layout()
plt.show()
```

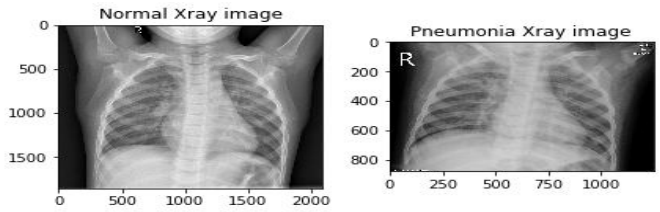


Fig. 4 Bulk Reading of Images using sklearn.io.imread_collection

2) *Image Resizing:* It is observed that the images have varying sizes, which makes it important to resize to a standard size in order for consistency. Pixel size of 250x250 is chosen considering there is not a significant loss of pixel data in resized images.

```
image_resized = resize(imgs[0], (250,250),
                       anti_aliasing=True)

fig, axes = plt.subplots(nrows=1, ncols=2)
ax = axes.ravel()
ax[0].imshow(imgs[0], cmap='gray')
ax[0].set_title("Original image")

ax[1].imshow(image_resized, cmap='gray')
ax[1].set_title("resized image")
plt.tight_layout()
plt.show()
```

```
D:\softwares\Anaconda\lib\site-packages\skimage\transform\_warps.py:105: UserWarning: The default mode, 'constant', will be cha
nged to 'reflect' in skimage 0.15.
  warn("The default mode, 'constant', will be changed to 'reflect' in "
```

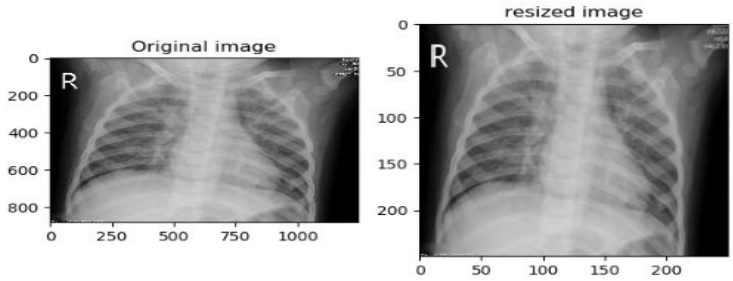


Fig. 5 Image Resizing

3) *Image Equalization:* Resized Images are required to be enhances an image as are having low contrast, methodology uses a method called Equalization, which "spreads out the most frequent intensity values" in an image 1. The equalized image has a roughly linear cumulative distribution function. Exposure from skimage provides function of Equalization in image processing.

```
plt.rcParams['figure.figsize'] = (16,9)
# Contrast stretching
p2, p98 = np.percentile(image_resized, (2, 98))
pic1 = exposure.rescale_intensity(image_resized, in_range=(p2, p98))
# Equalization
img_eq = exposure.equalize_hist(image_resized)
# Adaptive Equalization
img_adapteq = exposure.equalize_adapthist(image_resized, clip_limit=0.03)
fig, axes = plt.subplots(nrows=1, ncols=4)
ax = axes.ravel()
ax[0].imshow(image_resized, cmap='gray')
ax[0].set_title("Original image")
ax[1].imshow(pic1, cmap='gray')
ax[1].set_title("Contrast image")
ax[2].imshow(img_eq, cmap='gray')
ax[2].set_title("Equalization image")
ax[3].imshow(img_adapteq, cmap='gray')
ax[3].set_title("Adaptive image")
plt.tight_layout()
plt.show()
```

```
D:\softwares\Anaconda\lib\site-packages\skimage\util\dtype.py:141: UserWarning: Possible precision loss when converting from fl
oat64 to uint16
  .format(dtypeobj_in, dtypeobj_out))
```

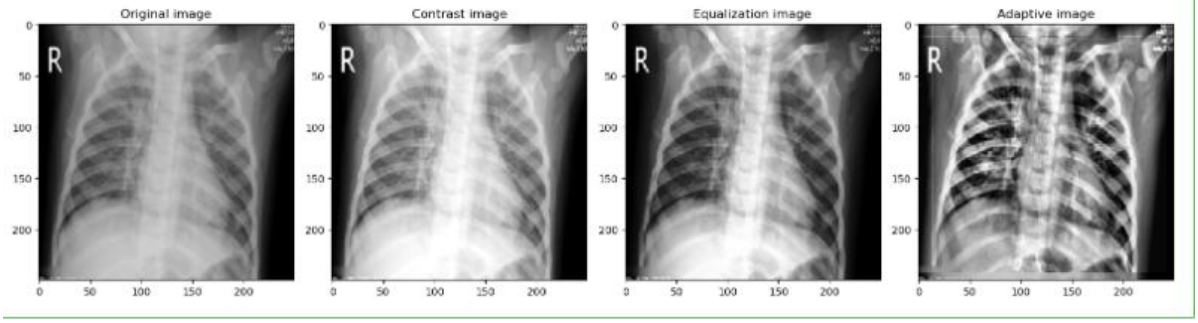


Fig. 6 Image Equalization

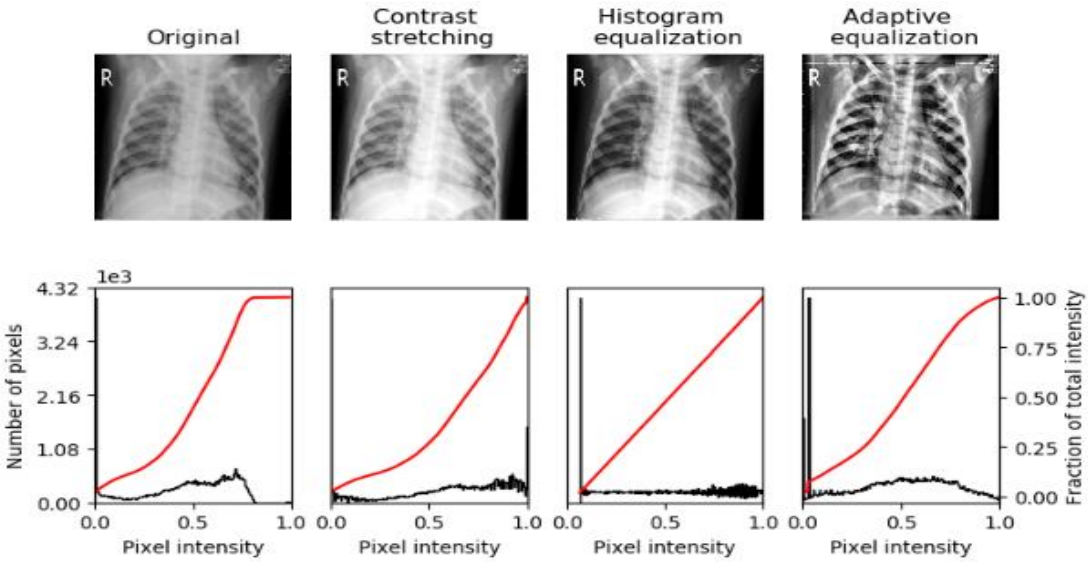


Fig. 7 Plotting Image and corresponding Histogram

Fig 6 and 7 shows comparision of Constrast stretching, Histogram equalization and adaptive equalization , We are using Adaptive Equalization for the proposed methodology as regions of ribs are quite distinctive from the inflammations and Organs which will assist in distinguishing Normal Healthy patient from Pneumonia Diseased patient.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 8 Issue V May 2020- Available at www.ijraset.com

C. Mapping Image Pixels as Pandas Data Frame

The below is the consolidated code for reading collection of images , resizing (250x250 = 62500) and adaptive equalization .the output is as list of arrays stacked up using np.vstack of each image pixel value in 1D using np.ravel(). This stacked array is to be done for normal and pneumonia patients images collection namely img and img2 as discused in Fig 4.

```
#first Iteration
image_resized = resize(imgs[0], (250,250), anti_aliasing=True)
img_adapteq = exposure.equalize_adapthist(image_resized, clip_limit=0.03)
arr1=img_adapteq.ravel()
#hstacking
for i in range(1,25):
    image_resized = resize(imgs[i], (250,250),
                anti_aliasing=True)
    # Adaptive Equalization
    img_adapteq = exposure.equalize_adapthist(image_resized, clip_limit=0.03)
    arr2=img_adapteq.ravel()
    arr1=np.vstack((arr1,arr2))

#similarly for pneumonia images img2
image_resized = resize(imgs2[0], (250,250),
                anti_aliasing=True)
    # Adaptive Equalization
img_adapteq = exposure.equalize_adapthist(image_resized, clip_limit=0.03)
arrt=img_adapteq.ravel()

for i in range(29):
    image_resized = resize(imgs2[i], (250,250),
                anti_aliasing=True)
    # Adaptive Equalization
    img_adapteq = exposure.equalize_adapthist(image_resized, clip_limit=0.03)
    arr3=img_adapteq.ravel()
    arrt=np.vstack((arrt,arr3))
```

Fig. 8 vstack and ravel on numpy array having image pixel values

From above code we get numpy array having pixel values arr1 as (25, 62500) representing 25 rows for each image of total 25 normal x-ray images, and 62500 columns representing each image pixel values of resized image of 250x250 dimensions. Similarly art has size (30,62500) .We can easily convert this in the Pandas DataFrame using the below command

```
1 data=pd.DataFrame(np.vstack((arr1,arrt)))
2 data.head(2)
```

	0	1	2	3	4	5	6	7	8	9	...	62490	62491	62492	62493	62494	
0	0.108344	0.108344	0.079534	0.079534	0.052068	0.052068	0.308613	0.342916	0.366233	0.379967	...	0.36782	0.370994	0.370994	0.375938	0.379112	0.
1	0.108344	0.108344	0.079534	0.079534	0.052068	0.052068	0.308613	0.342916	0.366233	0.379967	...	0.36782	0.370994	0.370994	0.375938	0.379112	0.

2 rows × 62500 columns

Fig. 9 DataFrame constructed having pixel values as Columns and rows as corresponding image

D. Dimensionality Reduction using PCA

Since there are a 62500 columns each representing the pixel value of image, we need to reduce the dimensions of the image. We use PCA to reduce the columns to 29 columns using pca.explained_variance_ratio_.sum() value covering 85% variance.

```
pca = PCA(n_components=29)
pca.fit(data)
```

```
PCA(copy=True, iterated_power='auto', n_components=29, random_state=None,
  svd_solver='auto', tol=0.0, whiten=False)
```

```
existing_2d = pca.transform(data)
existing_df_2d = pd.DataFrame(existing_2d)
existing_df_2d.head()
```

5	6	7	8	9	...	19	20	21	22	23	24	25	26	27	28
47	-5.962239	-13.285103	5.285634	3.071883	...	-8.036150	-1.271624	-3.119868	-5.099877	-15.203178	-1.209128	-12.074186	-2.346665	-0.724259	0.431441
00	2.961727	-6.636785	-11.098674	8.553159	...	-6.280867	-4.124956	0.304885	-0.308340	4.486024	-3.326474	-5.137371	-4.410998	5.201771	-3.953996
64	4.555577	-6.555822	9.214549	5.534222	...	15.999237	4.222030	-1.280850	-8.256420	1.382633	0.464706	6.224798	-1.579035	2.709269	-8.944333
73	8.990788	-13.641924	-0.888676	-0.892554	...	-4.671292	-0.738754	-5.905048	2.105506	1.361419	-4.470977	1.156634	12.673905	-3.030331	1.171083
09	-2.863926	5.891147	-11.020804	2.204706	...	-3.738111	-1.921217	2.389178	1.618817	11.804820	-1.047968	0.208842	-0.972234	2.693345	3.425651

Fig. 10 PCA with n_components=29

We need to save the PCA variable in to the pickle file in order to use the same generated PCA variable transform on new data points having existing behaviour to trained data points, this will be used in SVM modelling.

In [194]:
```
pickle_out = open("dict.pickle","wb")
pickle.dump(pca, pickle_out)
pickle_out.close()
```

In [195]:
```
pickle_in = open("dict.pickle","rb")
example_dict = pickle.load(pickle_in)
print(example_dict)
```

```
PCA(copy=True, iterated_power='auto', n_components=29, random_state=None,
  svd_solver='auto', tol=0.0, whiten=False)
```

Fig. 11 Saving PCA variable in pickle File

E. Kmeans Clustering on DataFrame

The DataFrame obtained by Fig 10 after PCA dimensionality reduction to 29 columns is clustered using sklearn kmeans clustering as given below . The clustered labels are added as column to the dataframe.

```
# Using the KMeans from sklearn
kmeans = KMeans(n_clusters=2)
kmeans.fit(existing_df_2d)
```

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=2, n_init=10, n_jobs=None, precompute_distances='auto',
    random_state=None, tol=0.0001, verbose=0)
```

```
kmeans.cluster_centers_
```

```
array([[-13.35937152,   1.72302244,  -0.24767855,   1.89164907,
         -0.88870968,   0.70340526,  -0.04754013,  -0.49333006,
          0.09721476,  -1.88295379,   0.16595179,   0.31659792,
          0.53191301,   1.14372363,  -0.06275946,   0.11561861,
         -0.43862949,   0.03320853,  -0.03381129,   0.20778987,
         -0.43571837,   0.22543995,   0.35107015,  -1.0413495 ,
          0.16630417,  -0.08261815,   0.39524561,  -0.14593293,
         -0.12951805],
       [ 13.85416306,  -1.78683808,   0.25685183,  -1.96171015,
          0.92162486,  -0.72945731,   0.04930088,   0.51160154,
         -0.10081531,   1.95269282,  -0.17209815,  -0.32832377,
         -0.5516135 ,  -1.18608377,   0.06508388,  -0.11990078,
          0.45487502,  -0.03443847,   0.03506356,  -0.21548579,
          0.45185609,  -0.23378958,  -0.36407274,   1.079918  ,
         -0.17246359,   0.08567808,  -0.40988434,   0.15133785,
          0.13431502]])
```

```
final =existing_df_2d
final['Actual Label']=kmeans.labels_
```

Fig. 12 Kmeans Clustering after PCA

Also we add additional column ImagesNames which will contain the names of Images (having Report ID of patient) , list of images can be retrieved in command prompt using cmd ls –lrt which will list the images in file system based on added date. ImagesNames will be used in bulk reporting step for increasing readability of bulk reports to Medical staff.

```
zz=final
zz.head()
```

5	6	7	8	9	...	22	23	24	25	26	27	28	Actual Label	ImagesNames
1.538147	-5.962239	-13.285103	5.285634	3.071883	...	-5.099877	-15.203176	-1.209128	-12.074186	-2.346665	-0.724259	0.431441	0	IM-0156-0001.jpeg
9.853300	2.961727	-6.636785	-11.098674	8.553159	...	-0.308340	4.486024	-3.326474	-5.137371	-4.410998	5.201771	-3.953996	0	IM-0158-0001.jpeg
1.739764	4.555577	-6.555822	9.214549	5.534222	...	-8.256420	1.382633	0.464706	6.224798	-1.579035	2.709269	-8.944333	0	IM-0160-0001.jpeg
4.728073	8.990788	-13.641924	-0.888676	-0.892554	...	2.105506	1.361419	-4.470977	1.156634	12.673905	-3.030331	1.171083	0	IM-0162-0001.jpeg
3.741609	-2.863926	5.891147	-11.020804	2.204706	...	1.618817	11.804820	-1.047968	0.208842	-0.972234	2.693345	3.425651	0	IM-0115-0001.jpeg

Fig. 13 Kmeans clustering labels column and corresponding Image Column

F. Intuition of Kmeans Clustering

K-Means Clustering – This algorithm works step-by-step where the main goal is to achieve clusters which have labels to identify them. The algorithm creates clusters of different data points which are as homogenous as possible by calculating the centroid of the cluster and making sure that the distance between this centroid and the new data point is as less as possible. The smallest distance between the data point and the centroid determines which cluster it belongs to while making sure the clusters do not interlay with each other. The centroid acts like the heart of the cluster. This ultimately gives us the cluster which can be labelled as needed.

This Unsupervised Learning Technique is ideal for the problem the paper addresses when medical staff especially during covid 19 crisis are overwhelmed with patients, there are lots of x-ray images generated for influx of patients which currently the medical system are not prepared for, since pneumonia is also one of the symptom of covid19 which makes it absolutely necessary to detect these. Hence Kmeans is used to cluster x-ray images into 2 clusters and generate cluster labels for each x-ray image ideally into normal and diseased patients in our case having pneumonia.

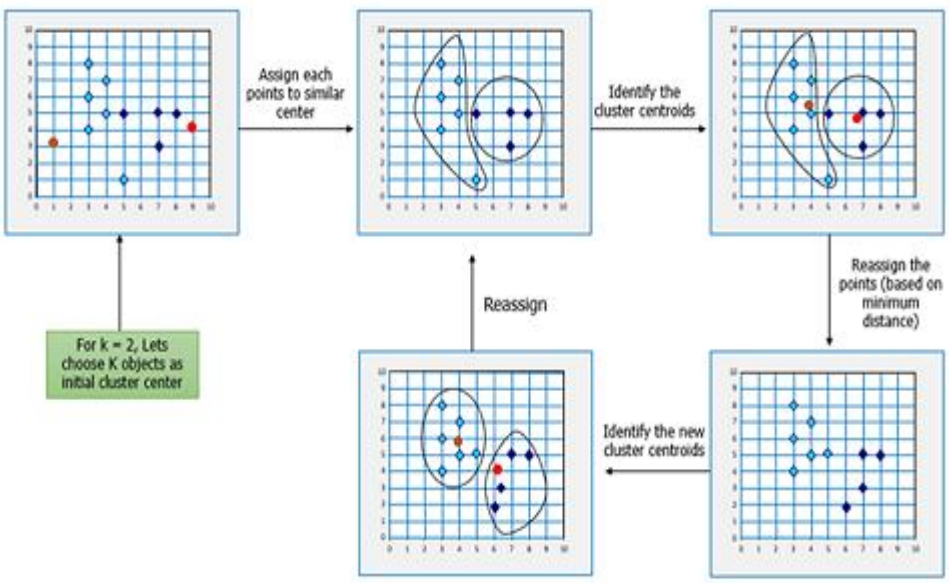
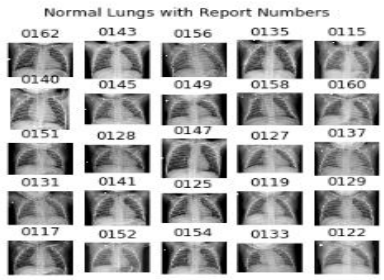


Fig. 14 Kmeans clustering algorithm

G. Bulk Reporting of Patients Reports

Our assessments after Kmeans clustering is compared to clustering label to the known label we find that kmeans clustering is able to cluster x-ray images into normal and pneumonia patients to 85% accuracy , which is a very large number and dependable one. The incorrectly labelled data is most of the normal patients clustered in pneumonia category this can be furthered improved by taking larger datasets and an quick review by medical staff only for normal patients once. Since in this problem statement we already acknowledge the presence of large datasets as images , kmeans clustering can significantly be improved to approx. 90-92% accuracy.

```
1  total_normal_images=list(zz[zz['Actual Label']==0]['ImagesNames'])
2  numberofReports =25
3  sample_normal = random.sample(total_normal_images,numberofReports)
4  f,ax = plt.subplots(5,5,figsize=(5,5))
5
6  for i in range(0,numberofReports):
7      im = cv2.imread('chest-xray-Project/Normal/'+sample_normal[i])
8      ax[i//5,i%5].imshow(im)
9      ax[i//5,i%5].axis('off')
10     ax[i//5,i%5].set_title(sample_normal[i][3:7])
11 f.suptitle('Normal Lungs with Report Numbers')
12 plt.show()
```

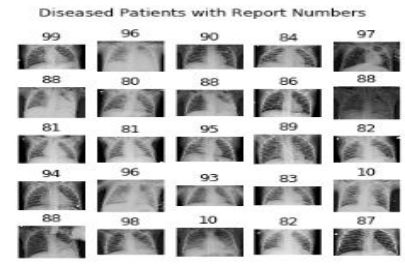



```
1  total_normal_images[0]
'IM-0156-0001.jpeg'
```

Fig. 15 Kmeans clustering labels column and corresponding Image Column for Normal patients

The Fig 15 shows the Quicker and more visual representation of bulk reporting of Normal patients using dataframe image row corresponding kmeans clustered column label “Actual label” equal to corresponding cluster label Number of Normal patients and corresponding x-ray Image using “ImagesNames” as explained in Fig 13. We are using matplotlib to display images with corresponding image title as Patient report id extracted from locally saved image names in file system. For Demonstartion we are displaying randomly 25 patients from the cluster , for displaying more patients or all normal patients , the lines 2,4,8,9,10 should be adjusted accordingly.

```
1  total_normal_images=list(zz[zz['Actual Label']==1]['ImagesNames'])
2  sample_normal = random.sample(total_normal_images,25)
3  f,ax = plt.subplots(5,5,figsize=(5,5))
4
5  for i in range(0,25):
6      im = cv2.imread('chest-xray-Project/Pneumonia/'+sample_normal[i])
7      ax[i//5,i%5].imshow(im)
8      ax[i//5,i%5].axis('off')
9      ax[i//5,i%5].set_title(sample_normal[i][6:8])
10 f.suptitle('Diseased Patients with Report Numbers')
11 plt.show()
```




```
1  total_normal_images[1]
'person96_virus_178.jpeg'
```

Fig. 16 Kmeans clustering labels column and corresponding Image Column for pneumonia patients

Similarly Fig 16 shows the bulk reporting of Pneumonia patients using dataframe image row corresponding kmeans clustered column label “Actual label” equal to corresponding cluster label Number of Pneumonia patients and corresponding x-ray Image using “ImagesNames” as explained in Fig 13. We are using matplotlib to display images with corresponding image title as Patient report id extracted from locally saved image names in file system. For Demonstartion we are displaying randomly 25 patients from the cluster , for displaying more patients or all normal patients , indexing should be adjusted accordingly.

H. Modeling Using K means Cluster Labels

In order To futher improve the accuracy and predict the category of the xray images , we can train models using the base kmeans clustered labels using larger datasets to produce about 90-92% accuracy and further 8% correction of labels by medical staff.

1) *SVM Model:* For Demonstration We are using the same dataFrame generated in Fig 13 having kmeans clustered labels as Actual Label for the reduced dimenstionality by PCA to 29 columns for 55 images. We then split the dataframe in train and test as 70-30 split and train SVM model on train data with kernel as linear. We are getting high accuracy about 88% for the dataset for trained model despite using kmeans labels clustering accuracy of 85%, implying this is a reliable approach as model would do even better with larger dataset and corrections of few deviances in kmeans labels.

```
tdata=zz.iloc[:,:30]
tdata.head()
```

5	6	7	8	9	...	20	21	22	23	24	25	26	27	28	Actual Label
147	-5.962239	-13.285103	5.285634	3.071883	...	-1.271624	-3.119868	-5.099877	-15.203176	-1.209128	-12.074186	-2.346665	-0.724259	0.431441	0
300	2.961727	-6.636785	-11.098674	8.553159	...	-4.124956	0.304885	-0.308340	4.486024	-3.326474	-5.137371	-4.410998	5.201771	-3.953996	0
764	4.555577	-6.555822	9.214549	5.534222	...	4.222030	-1.280850	-8.256420	1.382633	0.464706	6.224798	-1.579035	2.709269	-8.944333	0
073	8.990788	-13.641924	-0.888676	-0.892554	...	-0.738754	-5.905048	2.105506	1.361419	-4.470977	1.156634	12.673905	-3.030331	1.171083	0
609	-2.863926	5.891147	-11.020804	2.204706	...	-1.921217	2.389178	1.618817	11.804820	-1.047968	0.208842	-0.972234	2.693345	3.425651	0

```
train,test=train_test_split(tdata,test_size=0.3,random_state=4)
print(train.shape)
print(test.shape)
```

```
(38, 30)
(17, 30)
```

```
train_x=train.iloc[:,:29]
train_y=train.iloc[:,29]
test_x=test.iloc[:,:29]
test_y=test.iloc[:,29]
```

```
model=svm.SVC(kernel='linear')
model.fit(train_x,train_y)
predicted=model.predict(test_x)
print(metrics.accuracy_score(test_y,predicted))
```

```
0.8823529411764706
```

Fig. 17 Kmeans clustering labels column and corresponding Image Column for pneumonia patients

In order to predict the class of new bulk set of Xray images to Normal or Pneumonia diseased patients using the trained SVM model , we need to preform the same sequence if instructions as explained above in Fig 6,8,9,11,17 and 15,16. The flow includes similar steps as discussed above Imgae Resizing and Image Equalization, Coverting Image Pixels as Pandas Data Frame, Loading the saved PCA variable used in kmeans clustering labeling used in trained SVM model and fitting the dataFrame to reduce to expected 29 columns for trained SVM model and finally perform Bulk reporting based on the corresponding ImageName column and Classified Column for the each row representing images.

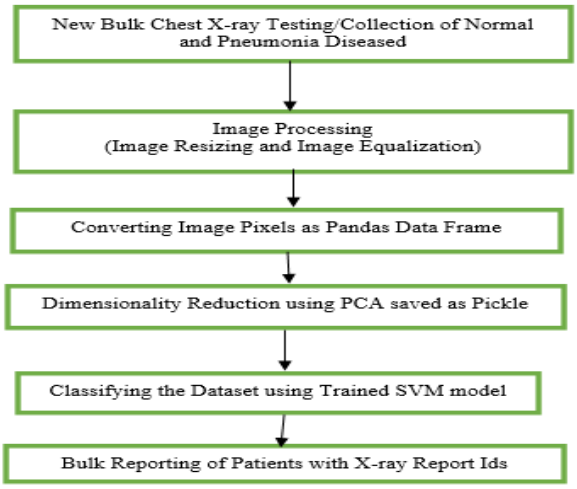


Fig. 18 Flow for Classifying new Bulk Xray Images using trained SVM Model

2) *CNN Model using Tensorflow:* If we need to Model using kmeans clustering obtained in Fig 13 but we want to avoid performing repetitive steps of the Image processing , Converting Image pixels as Pandas DataFrame, Dimensionality Reduction using PCA saved as Picke , in above Fig 18. Then we can opt for CNN Model instead which saves the overhead especially of image processing.

For the Below Demonstration we are using the Entire Data available in kaggle dataset
https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia
Saved in file system assuming these images were clustered using Kmeans clustering in Proposed Methodology, and subsequently collecting the images of obtained kmeans clustering label to corresponding imageName as Normal and Pneumonia. Collection of images are furthered assumed to be divided into train , test and val to assess the performance of CNN model.

a) Importing Packages: The Import Tensorflow and the Keras classes needed to construct our model.

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D,Dense,Flatten,Dropout,MaxPooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import ReduceLROnPlateau
```

Fig. 19 Importing Keras classes and Tensorflow

b) Creating the Model

```
image_height = 150
image_width = 150
batch_size = 10
no_of_epochs  = 10
```

```
model = Sequential()
model.add(Conv2D(32,(3,3),input_shape=(image_height,image_width,3),activation='relu'))
model.add(Conv2D(32,(3,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.2))
model.add(Conv2D(64,(3,3),activation='relu'))
model.add(Conv2D(64,(3,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.2))
model.add(Conv2D(128,(3,3),activation='relu'))
model.add(Conv2D(128,(3,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.2))
model.add(Flatten())
model.add(Dense(units=128,activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(units=1,activation='sigmoid'))
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

Fig. 18 Sample Sequential Model used for Image classification

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 148, 148, 32)      896
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 146, 146, 32)      9248
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 73, 73, 32)        0
_________________________________________________________________
dropout (Dropout)            (None, 73, 73, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 71, 71, 64)        18496
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 69, 69, 64)        36928
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 34, 34, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 34, 34, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 32, 32, 128)       73856
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 30, 30, 128)       147584
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 15, 15, 128)       0
_________________________________________________________________
dropout_2 (Dropout)          (None, 15, 15, 128)       0
_________________________________________________________________
flatten (Flatten)            (None, 28800)             0
_________________________________________________________________
dense (Dense)                (None, 128)               3686528
_________________________________________________________________
dropout_3 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 129
=================================================================
Total params: 3,973,665
Trainable params: 3,973,665
Non-trainable params: 0
_________________________________________________________________
```

Fig. 18 CNN Model Architecture

c) *Data Preparation using Image Data Generator class*: We are considering values for Data preparation in Fig 18 as: batch_size = 10, epochs = 10, IMG_HEIGHT = 150, IMG_WIDTH = 150. We have to Format the images into appropriately pre-processed floating point tensors before feeding to the network:

i) Read images from the disk.

ii) Decode contents of these images and convert it into proper grid format as per their RGB content.

iii) Convert them into floating point tensors.

iv) Rescale the tensors from values between 0 and 255 to values between 0 and 1, as neural networks prefer to deal with small input values.

All these tasks can be done with the ImageDataGenerator class provided by tf.keras. It can read images from disk and preprocess them into proper tensors. It will also set up generators that convert these images into batches of tensors—helpful when training the network. Optionally you can apply horizontal flip, zoom augmentation and rotation to avoid overfitting problems in model if necessary.

```
train_datagen = ImageDataGenerator(rescale=1./255,
                                   rotation_range=15,
                                   shear_range=0.2,
                                   zoom_range=0.2
                                   )

test_datagen = ImageDataGenerator(rescale=1./255)
```

```
training_set = train_datagen.flow_from_directory('chest-xray-pneumonia/chest_xray/train',
                                                 target_size=(image_width, image_height),
                                                 batch_size=batch_size,
                                                 class_mode='binary')

test_set = test_datagen.flow_from_directory('chest-xray-pneumonia/chest_xray/test',
                                            target_size=(image_width, image_height),
                                            batch_size=batch_size,
                                            class_mode='binary')

# Updated part --->
val_set = test_datagen.flow_from_directory('chest-xray-pneumonia/chest_xray/val',
                                           target_size=(image_width, image_height),
                                           batch_size=1,
                                           shuffle=False,
                                           class_mode='binary')
```

```
Found 5216 images belonging to 2 classes.
Found 624 images belonging to 2 classes.
Found 16 images belonging to 2 classes.
```

Fig. 16 data preparations

d) *Training and Validating the Model:* Use the fit_generator method of the constructed CNN model class to train the network. As in Fig 18 we have achieved an High accuracy of : 0.90 , can also be observed in Fig 19 for the trained model.

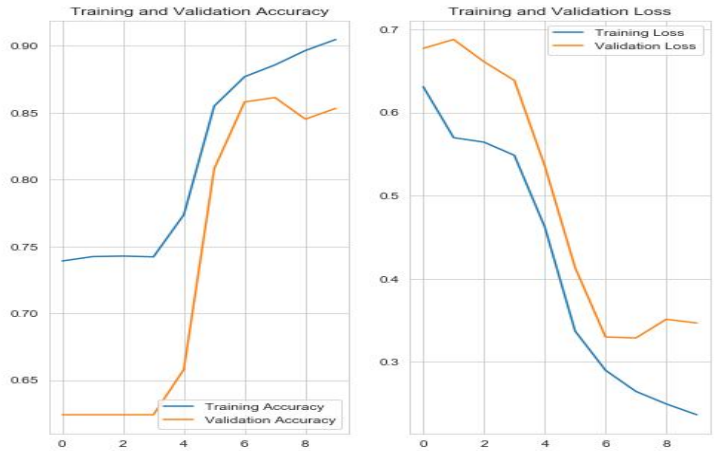


Fig. 19 Graphical representation of validation step

```
reduce_learning_rate = ReduceLROnPlateau(monitor='loss',
                                         factor=0.1,
                                         patience=2,
                                         cooldown=2,
                                         min_lr=0.00001,
                                         verbose=1)

callbacks = [reduce_learning_rate]
```

```
history = model.fit_generator(training_set,
                      steps_per_epoch=5216 //batch_size,
                      epochs=no_of_epochs,
                      validation_data=test_set,
                      validation_steps=624//batch_size,
                      callbacks=callbacks
                     )
```

```
WARNING:tensorflow:sample_weight modes were coerced from
  ...
    to
  ['...']
WARNING:tensorflow:sample_weight modes were coerced from
  ...
    to
  ['...']
Train for 521 steps, validate for 62 steps
Epoch 1/10
521/521 [==============================] - 1018s 2s/step - loss: 0.6310 - accuracy: 0.7391 - val_loss: 0.6771 - val_accuracy:
0.6242
Epoch 2/10
521/521 [==============================] - 948s 2s/step - loss: 0.5696 - accuracy: 0.7424 - val_loss: 0.6878 - val_accuracy: 0.
6242
Epoch 3/10
521/521 [==============================] - 952s 2s/step - loss: 0.5642 - accuracy: 0.7428 - val_loss: 0.6609 - val_accuracy: 0.
6242
Epoch 4/10
521/521 [==============================] - 933s 2s/step - loss: 0.5483 - accuracy: 0.7422 - val_loss: 0.6385 - val_accuracy: 0.
6242
Epoch 5/10
521/521 [==============================] - 936s 2s/step - loss: 0.4621 - accuracy: 0.7737 - val_loss: 0.5357 - val_accuracy: 0.
6581
Epoch 6/10
521/521 [==============================] - 1012s 2s/step - loss: 0.3369 - accuracy: 0.8550 - val_loss: 0.4133 - val_accuracy:
0.8081
Epoch 7/10
521/521 [==============================] - 928s 2s/step - loss: 0.2898 - accuracy: 0.8769 - val_loss: 0.3300 - val_accuracy: 0.
8581
Epoch 8/10
521/521 [==============================] - 953s 2s/step - loss: 0.2644 - accuracy: 0.8857 - val_loss: 0.3288 - val_accuracy: 0.
8613
Epoch 9/10
521/521 [==============================] - 1017s 2s/step - loss: 0.2493 - accuracy: 0.8965 - val_loss: 0.3512 - val_accuracy:
0.8452
Epoch 10/10
521/521 [==============================] - 941s 2s/step - loss: 0.2365 - accuracy: 0.9047 - val_loss: 0.3468 - val_accuracy: 0.
8532
```

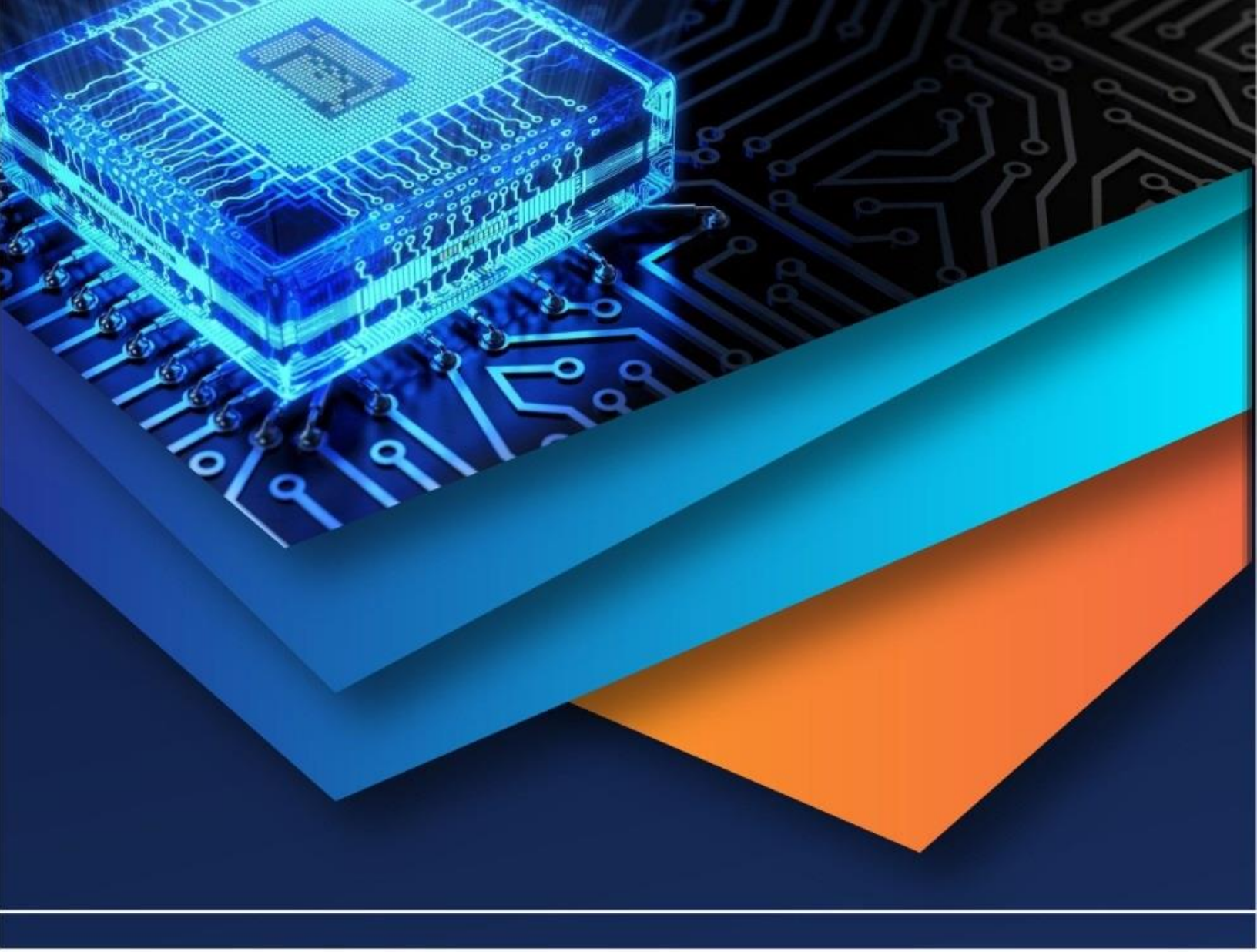
Fig. 19 Training the model for 521 steps and validate for 62 steps and 10 Epoch

III. APPLICATIONS OF METHODOLOGY

The proposed system is capable of assisting medical staff workers by providing Bulk reporting of patients on Chest X-ray images to Normal and Pneumonia diseased clusters especially during the covid19 times where medical staff is heavily burdened also since Pneumonia being covid19 symptom which makes this research more relevant during these uncertain times. Paper also successfully demonstrates creating high accuracy models for predicting new bulk Chest X-ray patients images based on previous clustered patients with the supervision of Health workers.

REFERENCES

[1] Singhal, Tanu. “A Review of Coronavirus Disease-2019 (COVID-19).” Indian journal of pediatrics vol. 87,4 (2020): 281-286. doi:10.1007/s12098-020-03263-6
[2] American Journal of Roentgenology. 2020;214: 1078-1082. 10.2214/AJR.20.22969 https://www.ajronline.org/doi/full/10.2214/AJR.20.22969
[3] Ferretti, Luca, et al. “Quantifying SARS-CoV-2 Transmission Suggests Epidemic Control with Digital Contact Tracing.” Science, American Association for the Advancement of Science, 8 May 2020, science.sciencemag.org/content/368/6491/eabb6936/tab-pdf.
[4] Cavallo, Joseph J. “Hospital Capacity and Operations in the Coronavirus Disease 2019(COVID-19) Pandemic-Planning for the Nth Patient.” JAMA Health Forum, American Medical Association, 17 Mar. 2020, jamanetwork.com/channels/health-forum/fullarticle/2763353.
[5] Mohammed Azam Sayeed, Noor Ayesha, Sakeena Fiza and Mohammed Akram Sayeed , “Analysis of Urine Samples to Classify as Hydrated or Dehydrated using Image Processing and XGboost Model” DOI: 10.22214/ijraset.2020.1053
[6] Mohammed Azam Sayeed, Sakeena Fiza, Noor Ayesha, Mohammed Akram Sayeed , “Detecting Malaria from Segmented Cell Images of Thin Blood Smear Dataset using Keras from Tensorflow” DOI: 10.22214/ijraset.2020.1109



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY