VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

MACHINE LEARNING (CSE-4020)

REVIEW – 3

SLOT–L25+L26

TOPIC : Chest X-Ray Computer Aided Diagnosis Using Neural Networks

Team Members

NAME	REG NO
DEEPAK MALPANI	17BCE0305
UDIT SINGHANIA	17BCE2060

Google Drive Link for video:
https://drive.google.com/drive/folders/1Y6nbW-JhmfmdR_4E6BeplRL6nEP0nhxE?usp=sharing

TABLES OF CONENT

ABSTRACT

A major health sector issue in India is lack of Diagnosis Support Systems and doctors to serve the large number of patients in rural areas. Indian Hosiptals in rural areas also lack Radiologist. 1000 of cases are usually handled by single doctor. Our aim is to create an AI based Computer-Aided Diagnosis Tool, which can classify 14 abnormalities- consolidation, filtration, edema, pneuomthorax, etc. To assist the doctors in arriving at quick diagnosis.

We develop an algorithm that can detect pneumonia from chest X-rays at a level exceeding practicing radiologists. Our algorithm, CheXNet, is a convolutional neural network trained on ChestX-ray14, currently the largest publicly available chest Xray dataset, containing over 100,000 frontalview X-ray images with 14 diseases. Four practicing academic radiologists annotate a test set, on which we compare the performance of CheXNet to that of radiologists. We find that CheXNet exceeds average radiologist performance on the F1 metric. We extend CheXNet to detect all 14 diseases in ChestX-ray14 and achieve state of the art results on all 14 diseases.

1. MOTIVATION

A major health sector issue in India is lack of Diagnosis Support Systems and doctors to serve the large number of patients in rural areas. Indian Hospitals in rural areas also lack Radiologist. 1000 of cases are usually handled by single doctor. Our aim is to create an AI based Computer-Aided Diagnosis Tool, which can classify 14 abnormalities- consolidation, filtration, edema, pneuomthorax, etc. To assist the doctors in arriving at quick diagnosis.

2. OBJECTIVES

• To create a machine learning automated system in order to achieve far more

efficiency than it is present in the hospitals.

• Use of Machine and deep learning techniques, which help in automation

of pre-processing of X-Ray image and producing the output.

• Use of Convolutional neural networks which have revolutionized the field of

image recognition. And also making use of backpropagation techniques.

• This project's main objective is to simplify the work of pathologists and doctors, so that they can work more faster with reliable AI technology as their support.

3. INTRODUCTION

Chest X-rays are currently the best available method for diagnosing pneumonia (WHO, 2001), playing a crucial role in clinical care (Franquet, 2001) and epidemiological studies (Cherian et al., 2005). However, detecting pneumonia in chest X-rays is a challenging task that relies on the availability of expert radiologists.

Our model, ChexNet,is a convolutional neural network that inputs a chestX-ray image and outputs the probability of pneumonia along with a heatmap localizing the areas of the image most indicative of pneumonia. We train CheXNet on the recently released ChestX-ray14 dataset (Wang et al., 2017), which contains 112,120 frontal-view chest X-ray images individually labeled with upto 14 different thoracic diseases, including pneumonia. We use dense connections (Huang et al., 2016) and batch normalization (Ioffe & Szegedy, 2015) to make the optimization of such a deep network tractable. Detecting pneumonia in chest radiography can be difficult for radiologists. The appearance of pneumonia in X-ray images is often vague, can overlap with other diagnoses, and can mimic many other benign abnormalities. These discrepancies cause considerable variability among radiologists in the diagnosis of pneumonia (Neuman et al., 2012; Davies et al., 1996; Hopstaken et al., 2004). To estimate radiologist performance, we collect annotations from four practicing academic radiologists on a subset of 420 images from ChestX-ray14. On these 420 images, we measure performance of individual radiologists and themodel.

We find that the model exceeds the average radiologist performance on the pneumonia detection task. To compare CheXNet against previous work using ChestX-ray14, we make simple modifications to CheXNet to detect all 14 diseases in ChestX-ray14, and find that we outperform best published results on all 14 diseases. Automated detection of diseases from chest X-rays at the level of expert radiologists would not only have tremendous benefit in clinical settings, it would also be invaluable in delivery of health care to populations with inadequate access to diagnostic imaging specialists.

4. LITERATURE SURVEY

Recent advancements in deep learning and large datasets have enabled algorithms to surpass the performance of medical professionals in a wide variety of medical imaging tasks, including diabetic retinopathy detection (Gulshan et al., 2016), skin cancer classification (Esteva et al., 2017), arrhythmia detection (Rajpurkar et al., 2017), and hemorrhage identification (Grewal et al., 2017). Automated diagnosis from chest radiographs has received increasing attention with algorithms for pulmonary tuberculosis classification (Lakhani & Sundaram, 2017) and lung nodule detection (Huang et al., 2017). Islam et al. (2017) studied the performance of various convolutional architectures on different abnormalities using the publicly available OpenI dataset (Demner-Fushman et al., 2015). Wang et al. (2017) released ChestX-ray-14, an

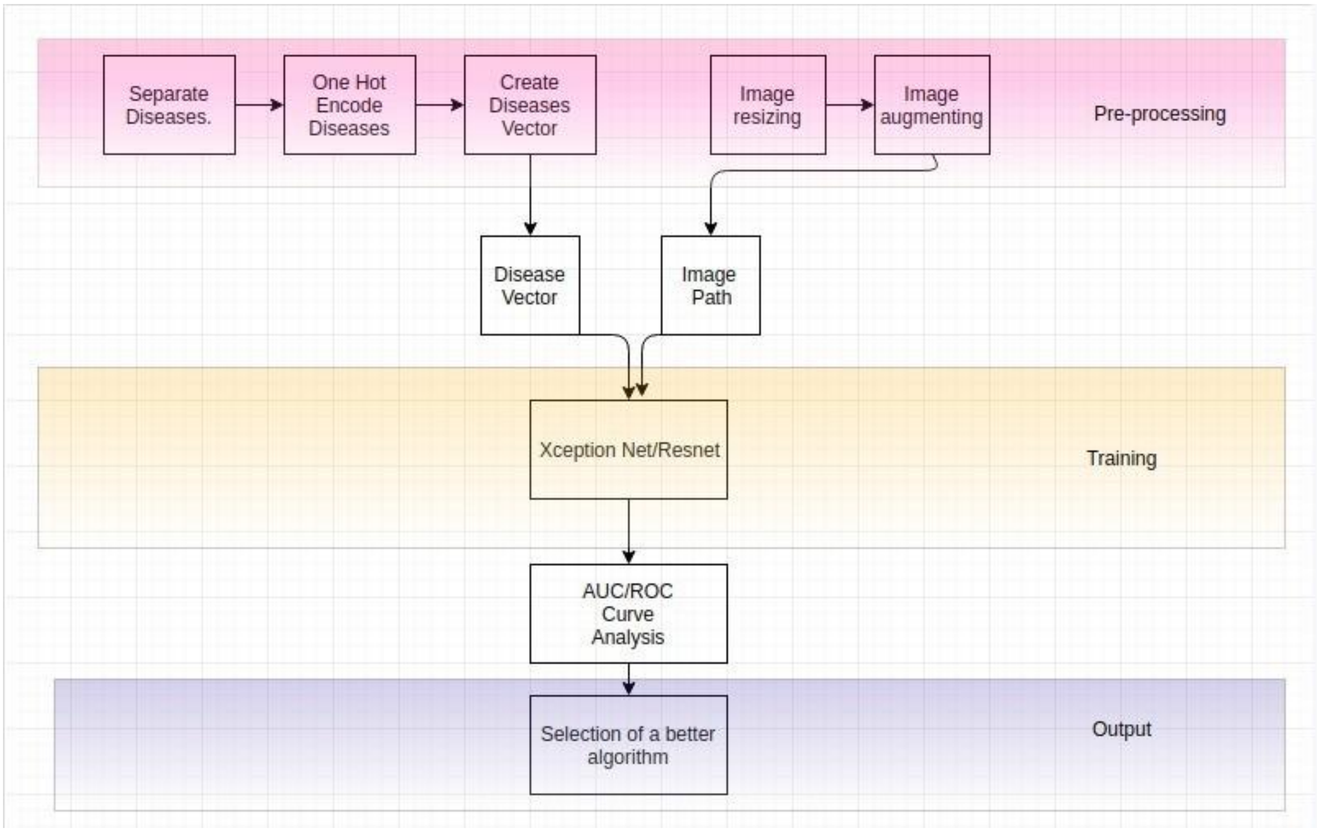
order of magnitude larger than previous datasets of its kind, and also benchmarked different convolutional neural network architectures pre-trained on ImageNet. Recently Yao et al. (2017) exploited statistical dependencies between labels in order make more accurate predictions, outperforming Wang et al. (2017) on 13 of 14 classes.

5. PROPOSED METHODOLOGY

a. DATASET DESCRIPTION

We use the ChestX-ray14 dataset released by Wang et al. (2017) which contains 112,120 frontal-view X-ray images of 30,805 unique patients. Wang et al. (2017) annotate each image with up to 14 different thoracic pathology labels using automatic extraction methods on radiology reports. We label images that have pneumonia as one of the annotated pathologies as positive examples and label all other images as negative examples.Forthe pneumonia detection task, we randomly split the dataset into training (28744 patients, 98637 images),validation(1672patients, 6351 images), and test (389 patients, 420 images). There is no patient overlap between the sets. Before inputting the images into the network, we downscale the images to 224×224 and normalize based on the mean and standard deviation of images in the ImageNet training set. We also augment the training data with random horizontalflipping.

b. ARCHITECTURE DIAGRAM



c. EXPERIMENTAL SETUP

We collected a test set of 420 frontal chest X-rays. Annotations were obtained independently from four practicing radiologists at Stanford University, who were asked to label all 14 pathologies in Wang et al. (2017). The radiologists had 4, 7, 25, and 28 years of experience, and one of the radiologists is a sub-specialty fellowship trained thoracic radiologist. Radiologists did not have access to any patient information or knowledge of disease prevalence in the data. Labels were entered into a standardized data entry program.

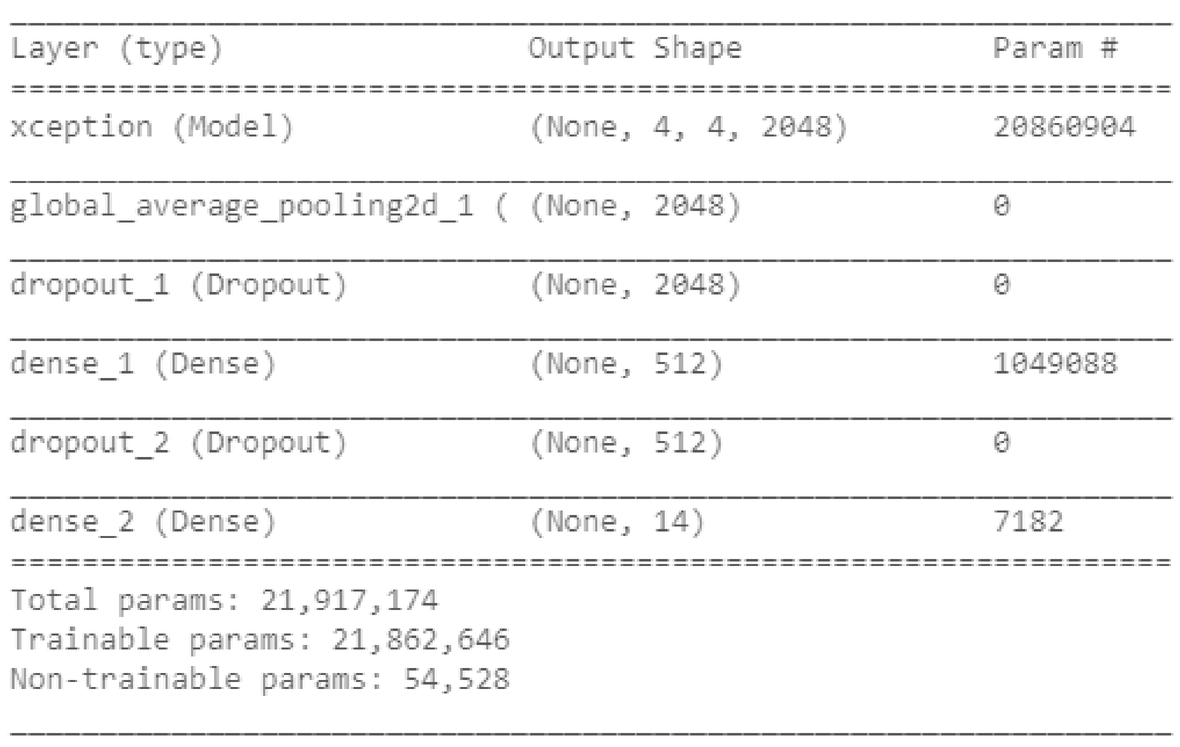
i. MODEL USED–XCEPTION

Xception is a novel deep convolutional neural network architecture, where Inception modules have been replaced with depthwise separable convolutions. With a similar parameter count, Xception significantly outperforms Inception V3 on a larger image classification dataset called JFT.

This code and the pre-trained weights only can be used for research purposes.

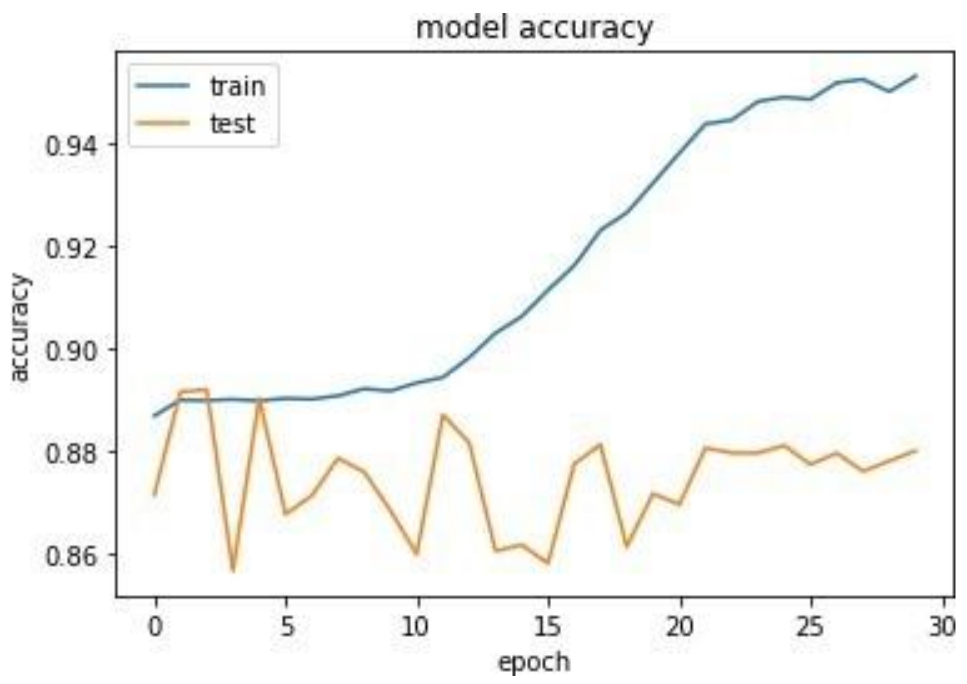
The canonical input image size for this Xception is 299x299, each pixel value should in range [-1,1] (RGB order), and the input preprocessing function is the same as Inception V3. According to the doc in Keras, this model gets to a top-1 validation accuracy of 0.790 and a top-5 validation accuracy of 0.945 on ImageNet.

The code is tested under Tensorflow 1.5, Python 3.5, Ubuntu 16.04.

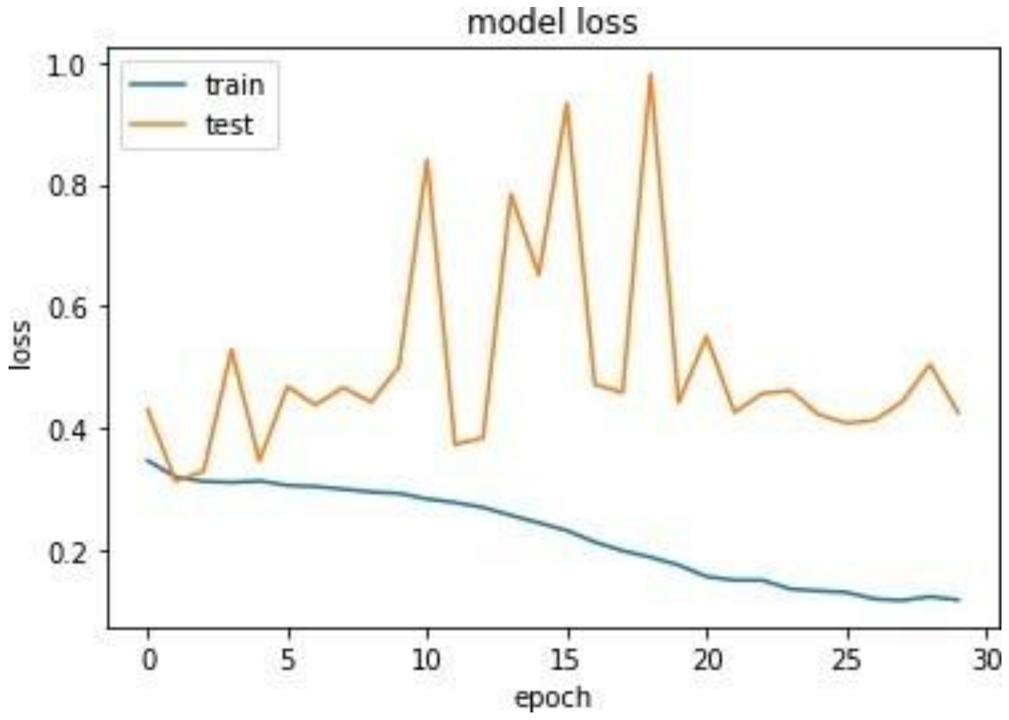


```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
xception (Model)             (None, 4, 4, 2048)        20860904
_________________________________________________________________
global_average_pooling2d_1 ( (None, 2048)              0
_________________________________________________________________
dropout_1 (Dropout)          (None, 2048)              0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               1049088
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 14)                7182
=================================================================
Total params: 21,917,174
Trainable params: 21,862,646
Non-trainable params: 54,528
_________________________________________________________________
```

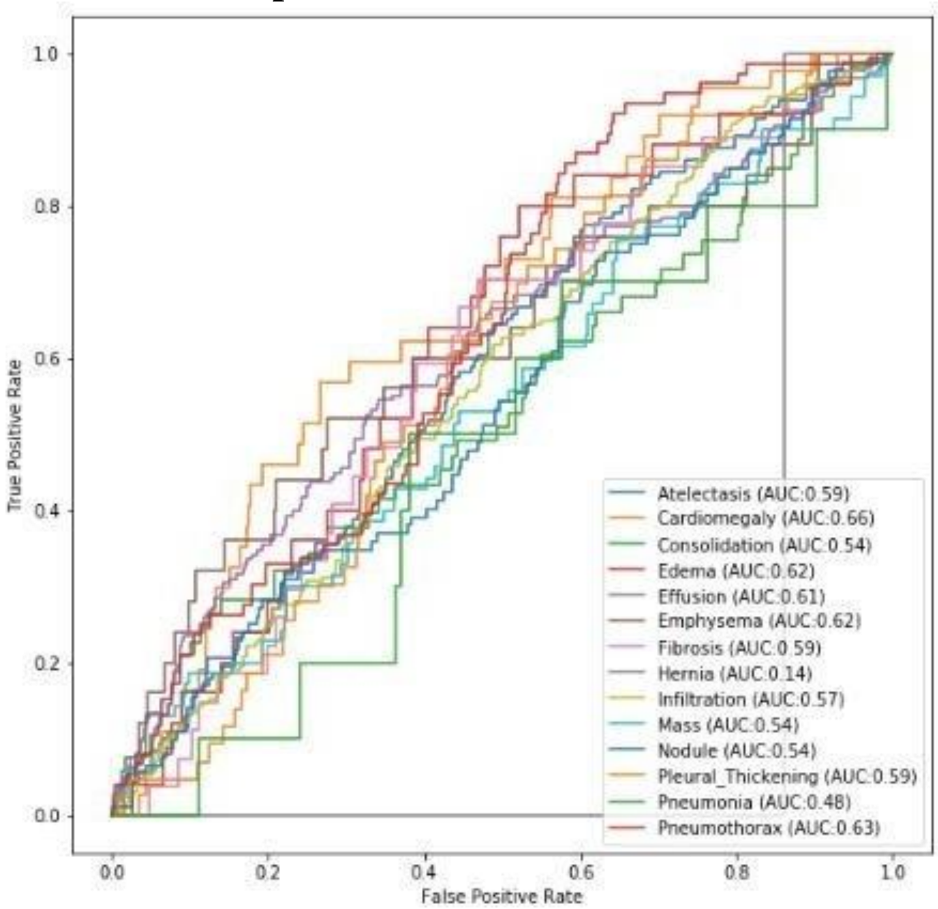
Model Accuracy of train data and test data



Model Loss of train data and test data



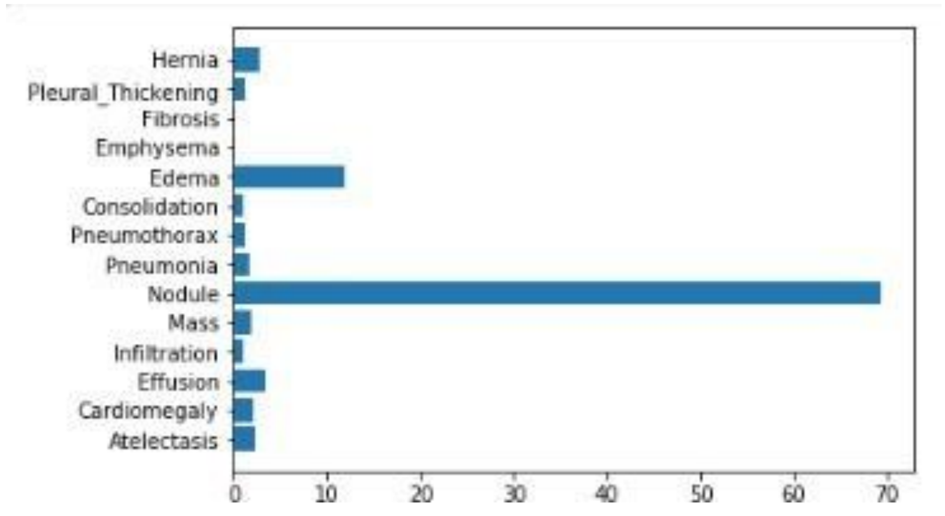
True positive rate v/s False positive rate of all 14 diseases



Result after testing on a X-Ray image

```
{'Atelectasis':            2.2382933646440506,            'Cardiomegaly':
2.143384888768196, 'Effusion':
3.316996619105339,' I n f i l t r a t i o n ' : 0.9539476595818996, 'Mass':
                         1.9700216129422188,                        'Nodule':
 69.39355731010437,              'Pneumonia':            1.648751087486744,
 'Pneumothorax':              1.143045723438263,          'Consolidation':
 1.0709989815950394,       'Edema':     11.799414455890656,      'Emphysema':
 0.08636426064185798,           'Fibrosis':          0.24755566846579313,
'Pleural_Thickening':             1.2561391107738018,              'Hernia':
2.7315298095345497}
```

Graphical Representation of the result

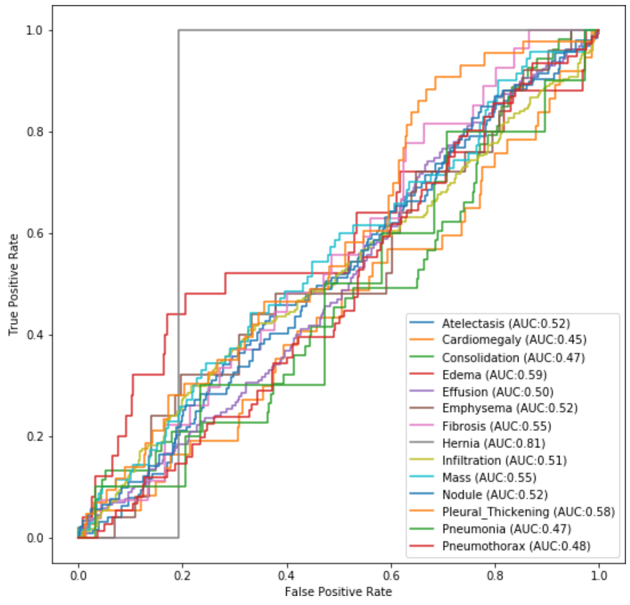


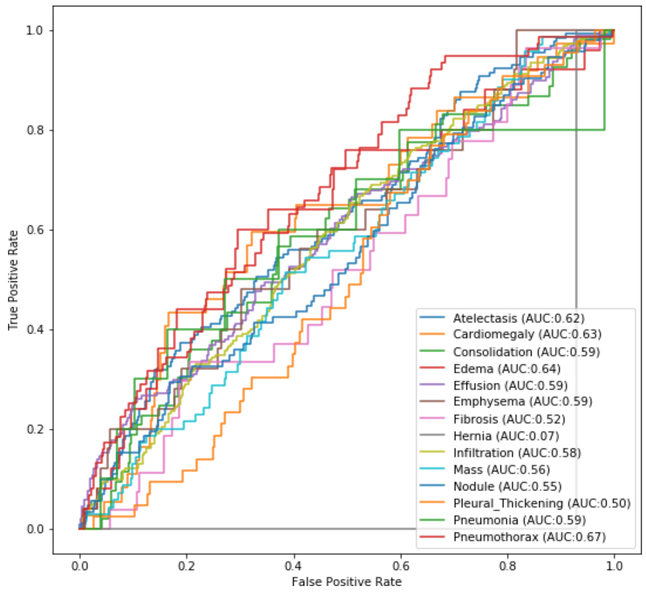
Comparative Study

We trained the same dataset using a resnet model and obtained the following result-

Model Name	Accuracy Percentage
ResNet	89
XceptionNet	94

Lets look at the AUC-ROC curves of both nets:

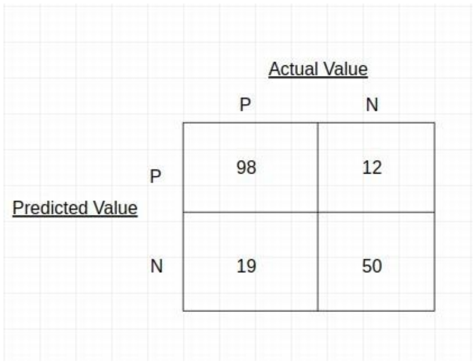




Resnet Xception Net

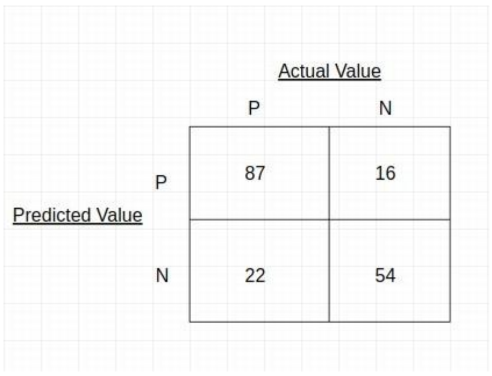
As we can see, y-axis stands for true positive rate and Xception Net shows more tendency to curve towards tpr, so Xception is better than ResNet.

Let's take a look at the confusion matrices for each net for generalized 183 trial samples.



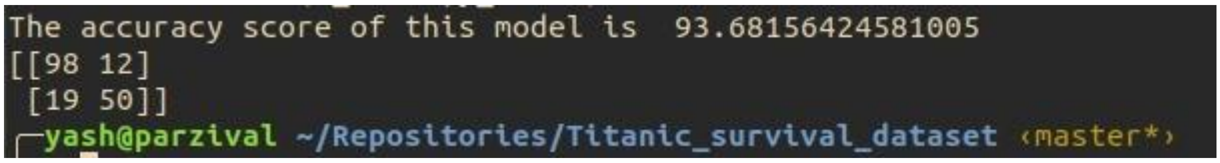
Predicted Value		Actual Value P	Actual Value N
	P	98	12
	N	19	50

Xception net

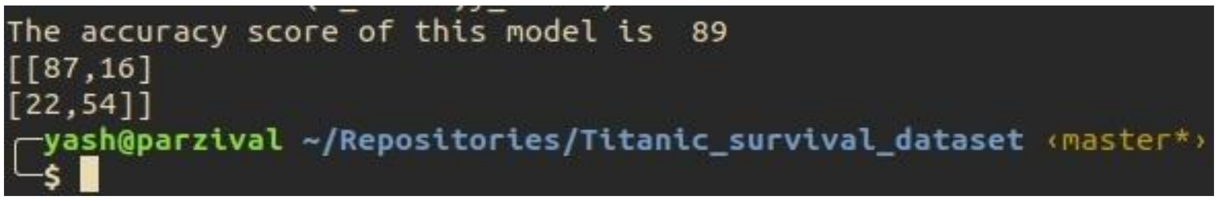


Predicted Value		Actual Value P	Actual Value N
	P	87	16
	N	22	54

Resnet



```
The accuracy score of this model is  93.68156424581005
[[98 12]
 [19 50]]
┌yash@parzival ~/Repositories/Titanic_survival_dataset ‹master*›
```



```
The accuracy score of this model is  89
[[87,16]
[22,54]]
┌yash@parzival ~/Repositories/Titanic_survival_dataset ‹master*›
└$ 
```

6. CONCLUSION

Pneumonia accountsfora significant proportion of patient morbidity and mortality (Gon¸calves-Pereira et al., 2013). Early diagnosis and treatment of pneumonia is critical to preventing complications including death (Aydogduetal., 2010). With approximately 2 billion procedures per year, chest X-rays are the most common imaging examination tool used in practice, criticalforscreening, diagnosis,and management of a variety of diseases including pneumonia (Raoof et al., 2012). However, two thirds of the global population lacks access to radiology diagnostics, according to an estimate by the World Health Organization (Mollura et al., 2010). There is a shortage of experts who can interpret X-rays, even when imaging equipment is available, leading to increased mortality from treatable diseases (Kesselman et al., 2016). We develop an algorithm which detects pneumonia from frontal-view chest X-ray images at a level exceeding practicing radiologists. We also show that a simple extension of our algorithm to detect multiple diseases outperforms previous state of the art on ChestX-ray14, the largest publicly available chest Xray dataset. With automation at the level of experts, we hope that this technology can improve healthcare delivery and increase access to medical imaging expertise in parts of the world where access to skilled radiologists is limited.

REFERENCES

1. https://arxiv.org/pdf/1711.05225.pdf

2. https://biomedical-engineering-online.biomedcentral.com/articles/10.1186/s12938-018-0544-y

3. https://arxiv.org/abs/1705.02315

4. X-ray(radiography)—chest.https://www.radiologyinfo.org/en/info.cfm?pg=chestrad. Accessed 10 June2018.

5. Simonyan K, Zisserman A. Very deep convolutional networks for large scale image recognition.

6. ArXiv preprint arXiv:14091556. 2014.http://arxiv.org/abs/1409.1556v6.

7. Szegedy C, LiuW,Jia Y, Sermanet P, Reed S, Anguelov D, et al. Going deeper with convolutions. In: 2015 IEEE conference on computer vision and pattern recogni tion (CVPR).Boston,MA,USA: IEEE.2015.p.1–9.http://dx.doi.org/10.1109/CVPR.2015.7298594.

8. CDC, 2017.URLhttps://www.cdc.gov/features/pneumonia/index.html

APPENDIX

```
#!/usr/bin/env python
# coding: utf-8

# In[17]:


# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load in

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the "../input/" directory.
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# Any results you write to the current directory are saved as output.


# In[18]:
```

```
import numpy as np
import pandas as pd
import os
from glob import iglob, glob
import matplotlib.pyplot as plt
from itertools import chain

get_ipython().run_line_magic('matplotlib', 'inline')


# In[ ]:





# In[19]:


dataframe = pd.read_csv("/kaggle/input/sample/sample_labels.csv")
all_image_paths = {os.path.basename(x): x for x in
                 glob(os.path.join('/kaggle/input/sample/*','images*', '*.png'))}
print('Scans found:', len(all_image_paths), ', Total Headers', dataframe.shape[0])
dataframe['path'] = dataframe['Image Index'].map(all_image_paths.get)
dataframe['Patient Age'] = dataframe['Patient Age'].map(lambda x: int(x[:-1]))


# In[20]:


dataframe = dataframe[dataframe['Finding Labels'] != 'No Finding']
all_labels = np.unique(list(chain(*dataframe['Finding Labels'].map(lambda x: x.split('|')).tolist())))
pathology_list = all_labels
dataframe['path'] = dataframe['Image Index'].map(all_image_paths.get)
dataframe = dataframe.drop(['Patient Age', 'Patient Gender', 'Follow-up #', 'Patient ID', 'View Position',
        'OriginalImageWidth', 'OriginalImageHeight',
'OriginalImagePixelSpacing_x','OriginalImagePixelSpacing_y'], axis=1)
for pathology in pathology_list :
    dataframe[pathology] = dataframe['Finding Labels'].apply(lambda x: 1 if pathology in x else 0)
dataframe = dataframe.drop(['Image Index', 'Finding Labels'], axis=1)


# In[21]:


dataframe.head()


# In[22]:


dataframe['disease_vec'] = dataframe.apply(lambda x: [x[all_labels].values], 1).map(lambda x: x[0])
```

```
# In[23]:


dataframe.head()


# In[24]:


from sklearn.model_selection import train_test_split

train_df, test_df = train_test_split(dataframe,
                                     test_size = 0.25,
                                     random_state = 2018)


# In[25]:


X_train = train_df['path'].values.tolist()
y_train = np.asarray(train_df['disease_vec'].values.tolist())
X_test = test_df['path'].values.tolist()
y_test = np.asarray(test_df['disease_vec'].values.tolist())


# In[26]:


print(X_train[0],y_train[0])


# In[27]:


X_test[0]


# In[28]:


from skimage.io import imread, imshow
print(imread(X_train[0]).shape)
images_train = np.zeros([len(X_train),128,128])
for i, x in enumerate(X_train):
    image = imread(x, as_gray=True)[::8,::8]
    images_train[i] = (image - image.min())/(image.max() - image.min())
images_test = np.zeros([len(X_test),128,128])
for i, x in enumerate(X_test):
    image = imread(x, as_gray=True)[::8,::8]
    images_test[i] = (image - image.min())/(image.max() - image.min())


# In[29]:
```

```
X_train = images_train.reshape(len(X_train), 128, 128, 1)
X_test = images_test.reshape(len(X_test), 128, 128, 1)
X_train.astype('float32')


# In[30]:


X_test[0].shape


# In[32]:


from keras.models import Sequential
from keras.layers import Dropout, GlobalAveragePooling2D, Dense, Dropout, Flatten
from keras.applications.resnet import ResNet50
base_model1 = ResNet50(input_shape = (128, 128, 1),
                        include_top = False, weights = None)
model1 = Sequential()
model1.add(base_model1)
model1.add(GlobalAveragePooling2D())
model1.add(Dropout(0.3))
model1.add(Dense(512))
model1.add(Dropout(0.3))
model1.add(Dense(len(all_labels), activation='softmax'))
model1.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
model1.summary()


# In[33]:


history1 = model1.fit(X_train, y_train, epochs = 30, verbose=1, validation_data=(X_test, y_test))


# In[34]:


from keras.models import Sequential
from keras.layers import Dropout, GlobalAveragePooling2D, Dense, Dropout, Flatten
from keras.applications.xception import Xception
base_model = Xception(input_shape = (128, 128, 1),
                        include_top = False, weights = None)
model = Sequential()
model.add(base_model)
model.add(GlobalAveragePooling2D())
model.add(Dropout(0.3))
model.add(Dense(512))
model.add(Dropout(0.3))
model.add(Dense(len(all_labels), activation='softmax'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
model.summary()
```

```
# In[35]:


history = model.fit(X_train, y_train, epochs = 30, verbose=1, validation_data=(X_test, y_test))


# In[48]:


def history_plot(history):
    plt.plot(history.history['acc'])
    plt.plot(history.history['val_acc'])
    plt.title('model accuracy')
    plt.ylabel('accuracy')
    plt.xlabel('epoch')
    plt.legend(['train', 'test'], loc='upper left')
    plt.show()
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    plt.title('model loss')
    plt.ylabel('loss')
    plt.xlabel('epoch')
    plt.legend(['train', 'test'], loc='upper left')
    plt.show()


# In[50]:


predictions1 = model1.predict(X_test, batch_size = 32, verbose = True)


# In[39]:


predictions = model.predict(X_test, batch_size = 32, verbose = True)


# In[40]:


from sklearn.metrics import roc_curve, auc
fig, c_ax = plt.subplots(1,1, figsize = (9, 9))
for (idx, c_label) in enumerate(all_labels):
    fpr, tpr, thresholds = roc_curve(y_test[:,idx].astype(int), predictions[:,idx])
    c_ax.plot(fpr, tpr, label = '%s (AUC:%0.2f)'  % (c_label, auc(fpr, tpr)))
c_ax.legend()
c_ax.set_xlabel('False Positive Rate')
c_ax.set_ylabel('True Positive Rate')
fig.savefig('barely_trained_net.png')


# In[51]:
```

```
from sklearn.metrics import roc_curve, auc
fig, c_ax = plt.subplots(1,1, figsize = (9, 9))
for (idx, c_label) in enumerate(all_labels):
    fpr, tpr, thresholds = roc_curve(y_test[:,idx].astype(int), predictions1[:,idx])
    c_ax.plot(fpr, tpr, label = '%s (AUC:%0.2f)'  % (c_label, auc(fpr, tpr)))
c_ax.legend()
c_ax.set_xlabel('False Positive Rate')
c_ax.set_ylabel('True Positive Rate')
fig.savefig('barely_trained_net.png')


# In[41]:


sickest_idx = np.argsort(np.sum(y_test, 1)<1)
fig, m_axs = plt.subplots(4, 4, figsize = (16, 32))
for (idx, c_ax) in zip(sickest_idx, m_axs.flatten()):
    c_ax.imshow(X_test[idx, :,:,0], cmap = 'bone')
    stat_str = [n_class[:6] for n_class, n_score in zip(all_labels, y_test[idx]) if n_score>0.5]
    pred_str = ['%s:%2.0f%%' % (n_class[:4], p_score*100)  for n_class, n_score, p_score in
zip(all_labels, y_test[idx], predictions[idx]) if (n_score>0.5) or (p_score>0.5)]
    c_ax.set_title('Dx: '+', '.join(stat_str)+'\nPDx: '+', '.join(pred_str))
    c_ax.axis('off')
fig.savefig('trained_img_predictions.png')


# In[ ]:


sickest_idx = np.argsort(np.sum(y_test, 1)<1)
fig, m_axs = plt.subplots(4, 4, figsize = (16, 32))
for (idx, c_ax) in zip(sickest_idx, m_axs.flatten()):
    c_ax.imshow(X_test[idx, :,:,0], cmap = 'bone')
    stat_str = [n_class[:6] for n_class, n_score in zip(all_labels, y_test[idx]) if n_score>0.5]
    pred_str = ['%s:%2.0f%%' % (n_class[:4], p_score*100)  for n_class, n_score, p_score in
zip(all_labels, y_test[idx], predictions1[idx]) if (n_score>0.5) or (p_score>0.5)]
    c_ax.set_title('Dx: '+', '.join(stat_str)+'\nPDx: '+', '.join(pred_str))
    c_ax.axis('off')
fig.savefig('trained_img_predictions.png')


# In[ ]:


model.save('chest-xray.h5')


# In[ ]:


from sklearn.externals import joblib
joblib.dump(model, 'model.pkl')
print("Model dumped!")
```

```
# In[8]:


import warnings
warnings.filterwarnings("ignore")
from keras.models import load_model
new_model = load_model("/home/udit/Music/chest-xray.h5")


# In[9]:


def predictor(trial):
    w = [trial]
    import numpy as np
    from skimage.io import imread, imshow
    images_t = np.zeros([len(w),128,128])
    for i, x in enumerate(w):
        image = imread(x, as_gray=True)[::8,::8]
        images_t[i] = (image - image.min())/(image.max() - image.min())
    X_t = images_t.reshape(len(w), 128, 128, 1)
    p = new_model.predict(X_t)
    class_p = [ 'Atelectasis', 'Cardiomegaly', 'Effusion', 'Infiltration', 'Mass', 'Nodule',
'Pneumonia','Pneumothorax', 'Consolidation', 'Edema', 'Emphysema', 'Fibrosis', 'Pleural_Thickening',
'Hernia']
    d1={}
    for i in range (len(class_p)):
        d1[class_p[i]]=p[0][i]*100
    return d1;


# In[14]:


test_1 = '/home/udit/Music/xray_dataset/00025115_000.png'
result = predictor(test_1)


# In[15]:


print(result)


# In[16]:


#max(d1, key=d1.get)
import heapq
heapq.nlargest(5, result, key=result.get)


# In[17]:
```

```
import matplotlib.pyplot as plt
plt.barh(range(len(result)), list(result.values()), align='center', )
plt.yticks(range(len(result)), list(result.keys()))

plt.show()
```