

A PROJECT REPORT

On

“Pneumonia Detection from Chest X-Ray using Image Classification with Deep Learning”

Submitted to

KIIT Deemed to be University

In Partial Fulfilment of the Requirements for the Award of

BACHELOR’S DEGREE IN

COMPUTER SCIENCE AND COMMUNICATION ENGINEERING

BY

RIDDHIMAN KUMAR (1729052)

DEBASRITO LAHIRI (1729122)

HRITAM JAISWAL (1729130)

SATVIK TIWARI (1729155)

UNDER THE GUIDANCE OF

PROF. RAJDEEP CHATTERJEE



KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY (KIIT)

Deemed to be University U/S 3 of the UGC Act, 1956

SCHOOL OF COMPUTER ENGINEERING

KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY

BHUBANESWAR, ODISHA – 751024

April 2021



KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY (KIIT)

Deemed to be University U/S 3 of the UGC Act, 1956

CERTIFICATE

This is to certify that the project entitled

“Pneumonia Detection from Chest X-Ray using Image Classification with Deep Learning”

Submitted by:

RIDDHIMAN KUMAR (1729052)

DEBASRITO LAHIRI (1729122)

HRITAM JAISWAL (1729130)

SATVIK TIWARI (1729155)

In partial fulfilment of the requirements for the award of the **Degree of Bachelor of Technology** in **Computer Science and Communication Engineering** is a bonafide record of the work carried out under my guidance and supervision at School of Computer Engineering, Kalinga Institute of Industrial Technology, Deemed to be University.

PROF. RAJDEEP CHATTERJEE

.....

The project was evaluated by us on:

EXAMINER’S NAME:

SIGNATURE:

ACKNOWLEDGEMENT

We take this opportunity to express our deep gratitude to all those helping hands without whom this project would have not been what it is. We take immense pleasure to express our thankfulness to our mentor, Prof. Rajdeep Chatterjee for his constant motivation and timely suggestions which helped us sail smooth through the odds we faced in our project. I would like to thank our Dean, Prof. Amulya Ratna Swain for giving us this opportunity to enhance our skills by giving a chance for this project and for motivating us throughout the B. Tech. course and also thank the complete faculty of Computer Science and Engineering who have provided us the knowledge to successfully complete the project. Last but not the least we would like to thank our friends who provided their helping hands in our project as and when required and by providing us with valuable feedback from time to time.

RIDDHIMAN KUMAR (1729052)

DEBASRITO LAHIRI (1729122)

HRITAM JAISWAL (1729130)

SATVIK TIWARI (1729155)

ABSTRACT

Pneumonia is a very serious condition of the lungs affecting the alveoli. It is caused by certain bacteriae and micro-organisms and although many vaccines and medicines have been developed for a lot of the organisms responsible, other organisms are still present in communities which cause the condition even now and results in affecting 450 million people every year of which about 4 million people die every year. Sometimes this condition is associated with other serious diseases too aggravating the damage caused by these diseases.

The method of detecting the condition is examination of symptoms related to it and for confirmation a lab test. In majority of the cases the most common and first choice lab test is a Chest X-Ray. It can be identified by the whitish patches in the alveolar region of the X-Ray.

Other lab tests that are sometimes advised are CT scan and/or sputum analysis. However, due to complicated process of these tests and the huge expenses associated with these tests, these are generally not the first choice of doctors and are generally undertaken only when an X-Ray is inconclusive.

As of now, all X-Rays are verified manually by a doctor or a lab pathologist which increases chances of human error. Also, in cases of minor pneumonia there is a chance that it might go undetected. If such cases can be detected effectively then the damage can be reduced by providing early and timely treatment. Also, the costs of needing a CT scan or other tests can also be reduced. Further, if it is automated the time for detecting the condition and generation of reports can be reduced. Normally, X-Ray reports are provided after about 24 hours. Considering that doctors may not be available for consultation the next day or for that matter in the same week of doing the test, the delay in getting treatment increases to about 3 to 7 days on average. However, this condition can create a huge damage in a matter of 2-3 days even.

Our projects will try to find a solution to this problem through the capabilities of computer science.

CONTENT

1. Introduction.....	6-7
1.1. Organization of study.....	6
1.2. Problem Statement.....	6
1.3. Objectives.....	6
1.4. Proposed project overview.....	7
2. Basic Concepts/Libraries Used.....	7-8
2.1. Numpy.....	7
2.2. Pandas.....	7
2.3. Matplotlib.....	7
2.4. OpenCV.....	7
2.5. Tensorflow.....	7
2.6. Keras.....	8
2.7. Convolution Neural Networks.....	8
2.8. Performance metrics.....	8
3. Project breakdown.....	9
4. Environment Setup.....	9
5. Dataset.....	9
6. Implementation and related code.....	10-14
7. Test results of the project.....	15-17
8. Conclusion.....	18
9. References.....	19

1. Introduction

1.1. Organization of study:

1. Chapter 1 introduces us to the need of the project, it's uses and intended users. It also gives a brief overview of the project.
2. Chapter 2 gives ideas of the libraries, concepts and various methods used to do our project.
3. Chapter 3 gives a step by step breakdown of the project implementation.
4. Chapter 4 shows the environment setups needed.
5. Chapter 5 gives some details about the dataset used for the project.
6. Chapter 6 shows the implementation of the mentioned concepts to build the proposed models and the related codebase.
7. Chapter 7 shows a test run of the model and it's output and comparison between the models.
8. Chapter 8 gives a final note on our project work.
9. Chapter 9 gives a reference of all the materials that we used during the creation of this project.

1.2. Problem Statement:

Modern X-Rays are digital and can be easily processed by computers. As such, it makes huge sense to automate the analysis of such X-Rays using ML for detection of this condition.

Accordingly, our project aims to train a deep learning model that will be able to classify between chest X-Rays possibly showing signs of pneumonia and chest X-Rays with normal conditions and effectively detect the possible presence of the condition in a patient.

1.3. Objectives:

The main objective of this project is to create and train a deep learning model that can detect the possible presence of pneumonia in chest X-ray images.

1.4. Proposed project overview:

Our project will include creating and training a deep learning model using CNN to predict the possible presence of pneumonia in chest X-ray pics and verifying the different metrics of the performance of the model. Then we will use it to predict the possible presence of pneumonia on a single chest X-Ray of known type to verify the model. We will be using a chest X-ray dataset obtained from kaggle for this purpose. The dataset was uploaded by Paul Mooney.

2. Basic concepts/libraries used:

2.1. Numpy:

Numpy is a python library that is used for numerical computations in python. It stands for Numerical python. It allows us to use large multidimensional arrays and do complex mathematical operations.

2.2. Pandas:

Pandas is also a python library that has various data structures and related operations for operating on numerical data and time series.

2.3. Matplotlib:

Matplotlib is an open-source low level graph plotting library in python that serves as a visualization utility.

2.4. OpenCV:

OpenCV is a Python open-source library, which is used for computer vision and image manipulations in Artificial intelligence, Machine Learning, face recognition, etc.

2.5. Tensorflow:

TensorFlow is an open source machine learning framework. It is used for implementing machine learning and deep learning applications.

2.6. Keras:

Keras is an open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library.

2.7. Convolution Neural Network:

Convolution neural networks are a deep learning model based on deep neural networks that use convolution in each layer of the neural network to form the feature map and extract features from it. We have used a basic CNN architecture having 5 convolution layers of which the first one uses Conv2d while the rest use SeparableConv2d. The first layer consists of 16 filters and is doubled in every consecutive layer. We have used Max Pooling to decrease the size of the feature maps. We have given a dropout of 0.2 in the last 2 convolution layers and used relu as our activation function. For our final output we use sigmoid activation function. We have used adam optimizer and binary_crossentropy for our loss function. Validation during training time is performed using accuracy metric. The number of epochs is 10. We also perform batch normalization in the layers.

2.8. Performance metrics:

Here we will use 6 performance metrics. Those are:

- i. Accuracy
- ii. Precision
- iii. Recall
- iv. F1-score
- v. Loss
- vi. Confusion matrix

3. Project breakdown:

Our project consists of the following steps:

- i. Importing of libraries and setting seed
- ii. Importing of dataset and extracting the data from it
- iii. Verification of dataset
- iv. Defining data pre-processing function for labelling the testing dataset and data augmentation of training and validation datasets using ImageDataGenerator of keras
- v. Setting the batch size, epochs, image dimension variables and using the function created in pervious step for data pre-processing
- vi. Setting up the CNN layers and compiling them to create a model along with generation of callbacks
- vii. Training the model using training data and validation data for tuning the model
- viii. Plotting of graph showing the model losses and accuracy changes over the different epochs of the training
- ix. Performing predictions using tuned model on test dataset
- x. Checking of different performance metrics for the test dataset predictions
- xi. Importing of another single image to perform predictions on
- xii. Pre-processing of the image
- xiii. Performing prediction on the image

4. Environment Setup:

Here, we have used google colaboratory for our project. As, all the libraries were preinstalled, there was no additional setup needed.

5. Dataset:

Here, we are using the dataset “Chest X-Ray Images (Pneumonia)” from kaggle provided by Paul Mooney. It has 5863 images of 2 categories.

6. Implementation and related code:

Steps 1 and 2:

```
Chest_X_Ray_final.ipynb ☆
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text
1 # General libraries
2 import os
3 import numpy as np
4 import pandas as pd
5 import random
6 import cv2
7 import matplotlib.pyplot as plt
8 %matplotlib inline
9
10 # Deep learning libraries
11 import keras.backend as K
12 from keras.models import Model, Sequential
13 from keras.layers import Input, Dense, Flatten, Dropout, BatchNormalization
14 from keras.layers import Conv2D, SeparableConv2D, MaxPool2D, LeakyReLU, Activation
15 from keras.optimizers import Adam
16 from keras.preprocessing.image import ImageDataGenerator
17 from keras.callbacks import ModelCheckpoint, ReduceLROnPlateau, EarlyStopping
18 import tensorflow as tf
19
20 # Setting seeds for reproducibility
21 seed = 232
22 np.random.seed(seed)
23 tf.compat.v1.set_random_seed(seed)

[2] 1 from google.colab import drive
2 drive.mount('/content/drive')

Mounted at /content/drive

[3] 1 ls

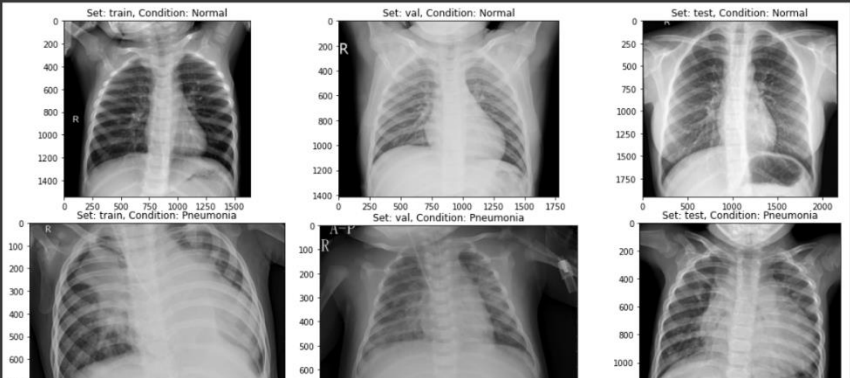
drive/ sample_data/

[4] 1 unzip -uq "/content/drive/MyDrive/input.zip" -d "/content/drive/MyDrive/"
0s completed at 1:34 AM
```

Steps 3 and 4:

```
Chest_X_Ray_final.ipynb ☆
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text
3 input_path = '../content/drive/MyDrive/input/chest-xray-pneumonia/chest_xray/chest_xray/'
4
5 fig, ax = plt.subplots(2, 3, figsize=(15, 7))
6 ax = ax.ravel()
7 plt.tight_layout()
8
9 for i, _set in enumerate(['train', 'val', 'test']):
10     set_path = input_path+_set
11     ax[i].imshow(plt.imread(set_path+'/NORMAL/'+os.listdir(set_path+'/NORMAL')[0]), cmap='gray')
12     ax[i].set_title('Set: {}, Condition: Normal'.format(_set))
13     ax[i+3].imshow(plt.imread(set_path+'/PNEUMONIA/'+os.listdir(set_path+'/PNEUMONIA')[0]), cmap='gray')
14     ax[i+3].set_title('Set: {}, Condition: Pneumonia'.format(_set))
```



```
Chest_X_Ray_final.ipynb ☆
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text
RAM Disk Editing

1 # Distribution of our datasets
2 for _set in ['train', 'val', 'test']:
3     n_normal = len(os.listdir(input_path + _set + '/NORMAL'))
4     n_infect = len(os.listdir(input_path + _set + '/PNEUMONIA'))
5     print('Set: {}, normal images: {}, pneumonia images: {}'.format(_set, n_normal, n_infect))

Set: train, normal images: 1341, pneumonia images: 3875
Set: val, normal images: 8, pneumonia images: 8
Set: test, normal images: 234, pneumonia images: 390

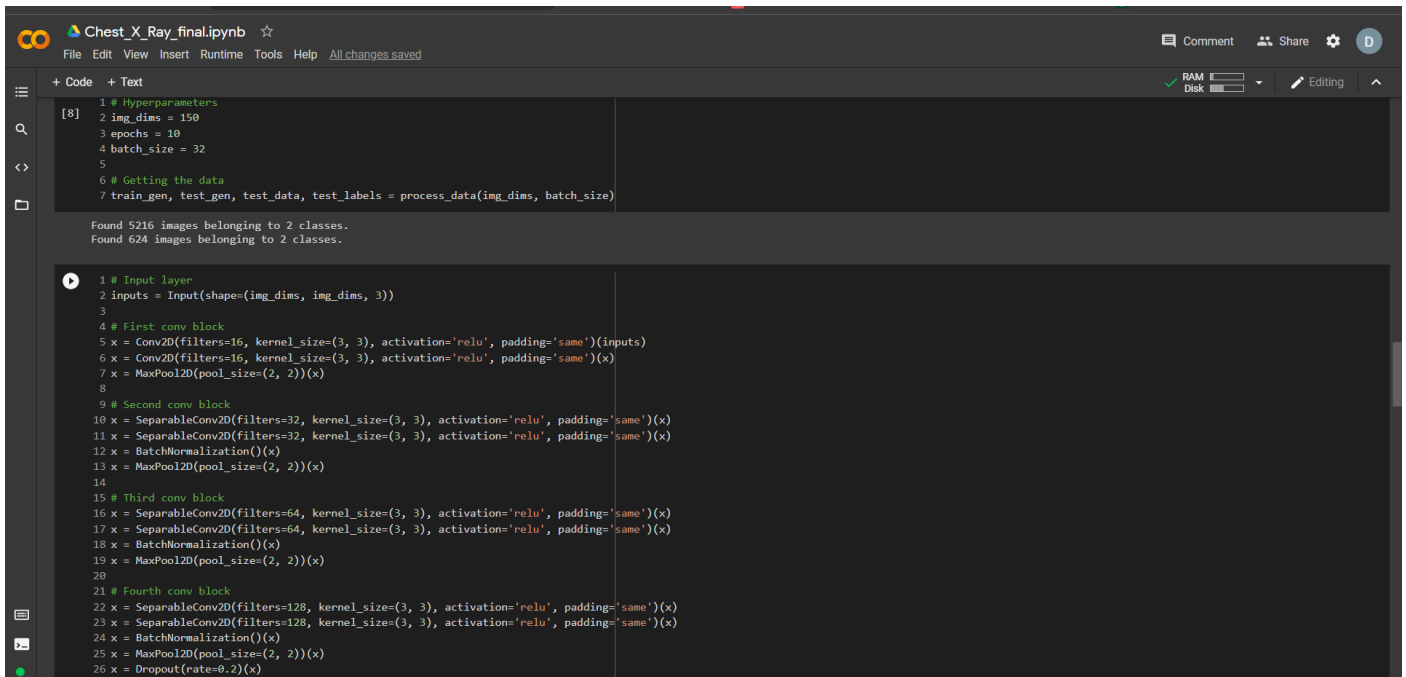
1 # input_path = '../input/chest_xray/chest_xray/'
2 #input_path = '../input/chest-xray-pneumonia/chest_xray/chest_xray/'
3 input_path = '../content/drive/MyDrive/input/chest-xray-pneumonia/chest_xray/chest_xray/'
4
5 def process_data(img_dims, batch_size):
6     # Data generation objects
7     train_datagen = ImageDataGenerator(rescale=1./255, zoom_range=0.3, vertical_flip=True)
8     test_val_datagen = ImageDataGenerator(rescale=1./255)
9
10    # This is fed to the network in the specified batch sizes and image dimensions
11    train_gen = train_datagen.flow_from_directory(
12        directory=input_path+'train',
13        target_size=(img_dims, img_dims),
14        batch_size=batch_size,
15        class_mode='binary',
16        shuffle=True)
17
18    test_gen = test_val_datagen.flow_from_directory(
19        directory=input_path+'val',
20        target_size=(img_dims, img_dims),
21        batch_size=batch_size,
22        class_mode='binary',
23        shuffle=True)
24
25    # I will be making predictions off of the test set in one batch size
26    # This is useful to be able to get the confusion matrix
27    test_data = []
```

```
Chest_X_Ray_final.ipynb ☆
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text
RAM Disk Editing

11 train_gen = train_datagen.flow_from_directory(
12     directory=input_path+'train',
13     target_size=(img_dims, img_dims),
14     batch_size=batch_size,
15     class_mode='binary',
16     shuffle=True)
17
18 test_gen = test_val_datagen.flow_from_directory(
19     directory=input_path+'val',
20     target_size=(img_dims, img_dims),
21     batch_size=batch_size,
22     class_mode='binary',
23     shuffle=True)
24
25 # I will be making predictions off of the test set in one batch size
26 # This is useful to be able to get the confusion matrix
27 test_data = []
28 test_labels = []
29
30 for cond in ['/NORMAL/', '/PNEUMONIA/']:
31     for img in os.listdir(input_path + 'test' + cond):
32         img = plt.imread(input_path+'test'+cond+img)
33         img = cv2.resize(img, (img_dims, img_dims))
34         img = np.dstack([img, img, img])
35         img = img.astype('float32') / 255
36         if cond=='/NORMAL/':
37             label = 0
38         elif cond=='/PNEUMONIA/':
39             label = 1
40         test_data.append(img)
41         test_labels.append(label)
42
43 test_data = np.array(test_data)
44 test_labels = np.array(test_labels)
45
46 return train_gen, test_gen, test_data, test_labels
```

Steps 5 and 6:

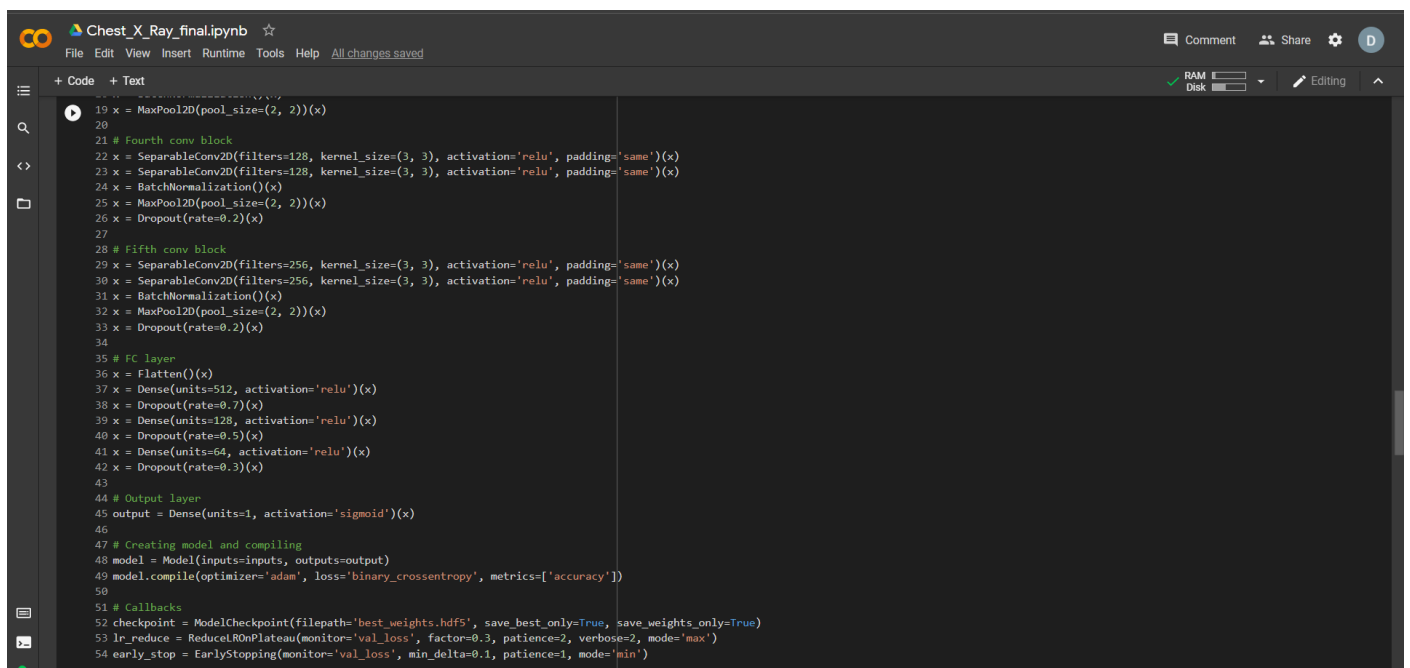


The screenshot shows a Jupyter Notebook titled 'Chest_X_Ray_final.ipynb'. The code defines hyperparameters and processes the data. It then defines the first part of a neural network architecture, including an input layer, two convolutional blocks, and a third convolutional block.

```
1 # Hyperparameters
2 img_dims = 150
3 epochs = 10
4 batch_size = 32
5
6 # Getting the data
7 train_gen, test_gen, test_labels = process_data(img_dims, batch_size)

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

1 # Input layer
2 inputs = Input(shape=(img_dims, img_dims, 3))
3
4 # First conv block
5 x = Conv2D(filters=16, kernel_size=(3, 3), activation='relu', padding='same')(inputs)
6 x = Conv2D(filters=16, kernel_size=(3, 3), activation='relu', padding='same')(x)
7 x = MaxPool2D(pool_size=(2, 2))(x)
8
9 # Second conv block
10 x = SeparableConv2D(filters=32, kernel_size=(3, 3), activation='relu', padding='same')(x)
11 x = SeparableConv2D(filters=32, kernel_size=(3, 3), activation='relu', padding='same')(x)
12 x = BatchNormalization()(x)
13 x = MaxPool2D(pool_size=(2, 2))(x)
14
15 # Third conv block
16 x = SeparableConv2D(filters=64, kernel_size=(3, 3), activation='relu', padding='same')(x)
17 x = SeparableConv2D(filters=64, kernel_size=(3, 3), activation='relu', padding='same')(x)
18 x = BatchNormalization()(x)
19 x = MaxPool2D(pool_size=(2, 2))(x)
20
21 # Fourth conv block
22 x = SeparableConv2D(filters=128, kernel_size=(3, 3), activation='relu', padding='same')(x)
23 x = SeparableConv2D(filters=128, kernel_size=(3, 3), activation='relu', padding='same')(x)
24 x = BatchNormalization()(x)
25 x = MaxPool2D(pool_size=(2, 2))(x)
26 x = Dropout(rate=0.2)(x)
```



The screenshot shows the continuation of the Keras model definition. It includes a fifth convolutional block, a fully connected layer, and the final output layer. The model is then compiled with Adam optimizer and binary crossentropy loss, and callbacks are defined for saving the best model and early stopping.

```
27
28 # Fifth conv block
29 x = SeparableConv2D(filters=256, kernel_size=(3, 3), activation='relu', padding='same')(x)
30 x = SeparableConv2D(filters=256, kernel_size=(3, 3), activation='relu', padding='same')(x)
31 x = BatchNormalization()(x)
32 x = MaxPool2D(pool_size=(2, 2))(x)
33 x = Dropout(rate=0.2)(x)
34
35 # FC layer
36 x = Flatten()(x)
37 x = Dense(units=512, activation='relu')(x)
38 x = Dropout(rate=0.7)(x)
39 x = Dense(units=128, activation='relu')(x)
40 x = Dropout(rate=0.5)(x)
41 x = Dense(units=64, activation='relu')(x)
42 x = Dropout(rate=0.3)(x)
43
44 # Output layer
45 output = Dense(units=1, activation='sigmoid')(x)
46
47 # Creating model and compiling
48 model = Model(inputs=inputs, outputs=output)
49 model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
50
51 # Callbacks
52 checkpoint = ModelCheckpoint(filepath='best_weights.hdf5', save_best_only=True, save_weights_only=True)
53 lr_reduce = ReduceLROnPlateau(monitor='val_loss', factor=0.3, patience=2, verbose=2, mode='max')
54 early_stop = EarlyStopping(monitor='val_loss', min_delta=0.1, patience=1, mode='min')
```

Step 7:

```
Chest_X_Ray_final.ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text
[9] 54 early_stop = EarlyStopping(monitor='val_loss', min_delta=0.1, patience=1, mode='min')

1 # Fitting the model
2 hist = model.fit_generator(
3     train_gen, steps_per_epoch=train_gen.samples // batch_size,
4     epochs=epochs, validation_data=test_gen,
5     validation_steps=test_gen.samples // batch_size, callbacks=[checkpoint, lr_reduce])

/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/training.py:1844: UserWarning: 'Model.fit_generator' is deprecated and will be removed in a future version. Please use 'Model.fit', which
warnings.warn("'Model.fit_generator' is deprecated and '

Epoch 1/10
163/163 [=====] - 1970s 12s/step - loss: 0.4854 - accuracy: 0.7858 - val_loss: 0.6858 - val_accuracy: 0.6234
Epoch 2/10
163/163 [=====] - 340s 2s/step - loss: 0.2731 - accuracy: 0.8922 - val_loss: 0.9553 - val_accuracy: 0.6299
Epoch 3/10
163/163 [=====] - 337s 2s/step - loss: 0.2245 - accuracy: 0.9115 - val_loss: 1.4857 - val_accuracy: 0.6250
Epoch 4/10
163/163 [=====] - 339s 2s/step - loss: 0.2399 - accuracy: 0.9109 - val_loss: 0.4260 - val_accuracy: 0.8553
Epoch 5/10
163/163 [=====] - 334s 2s/step - loss: 0.2060 - accuracy: 0.9184 - val_loss: 0.3800 - val_accuracy: 0.8405

Epoch 00005: ReduceLROnPlateau reducing learning rate to 0.0003000000142492354.
Epoch 6/10
163/163 [=====] - 334s 2s/step - loss: 0.1847 - accuracy: 0.9396 - val_loss: 0.2750 - val_accuracy: 0.8947
Epoch 7/10
163/163 [=====] - 332s 2s/step - loss: 0.1473 - accuracy: 0.9483 - val_loss: 0.4036 - val_accuracy: 0.8191

Epoch 00007: ReduceLROnPlateau reducing learning rate to 9.000000427477062e-05.
Epoch 8/10
163/163 [=====] - 333s 2s/step - loss: 0.1318 - accuracy: 0.9501 - val_loss: 0.2381 - val_accuracy: 0.9161
Epoch 9/10
163/163 [=====] - 335s 2s/step - loss: 0.1385 - accuracy: 0.9551 - val_loss: 0.2475 - val_accuracy: 0.9161

Epoch 00009: ReduceLROnPlateau reducing learning rate to 2.700000040931627e-05.
Epoch 10/10
163/163 [=====] - 333s 2s/step - loss: 0.1374 - accuracy: 0.9516 - val_loss: 0.2589 - val_accuracy: 0.9161

[11] 1 fig, ax = plt.subplots(1, 2, figsize=(10, 3))
      2 ax = ax.ravel()
      3
      4 for i, met in enumerate(['accuracy', 'loss']):
      5     ax[i].plot(hist.history[met])
      6     ax[i].plot(hist.history['val_' + met])
      7     ax[i].set_title('Model {}'.format(met))
      8     ax[i].set_xlabel('epochs')
      9     ax[i].set_ylabel(met)
     10     ax[i].legend(['train', 'val'])
```

Steps 8 to 10:

```
Chest_X_Ray_final.ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text
[9] 54 early_stop = EarlyStopping(monitor='val_loss', min_delta=0.1, patience=1, mode='min')

1 # Fitting the model
2 hist = model.fit_generator(
3     train_gen, steps_per_epoch=train_gen.samples // batch_size,
4     epochs=epochs, validation_data=test_gen,
5     validation_steps=test_gen.samples // batch_size, callbacks=[checkpoint, lr_reduce])

/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/training.py:1844: UserWarning: 'Model.fit_generator' is deprecated and will be removed in a future version. Please use 'Model.fit', which
warnings.warn("'Model.fit_generator' is deprecated and '

Epoch 1/10
163/163 [=====] - 1970s 12s/step - loss: 0.4854 - accuracy: 0.7858 - val_loss: 0.6858 - val_accuracy: 0.6234
Epoch 2/10
163/163 [=====] - 340s 2s/step - loss: 0.2731 - accuracy: 0.8922 - val_loss: 0.9553 - val_accuracy: 0.6299
Epoch 3/10
163/163 [=====] - 337s 2s/step - loss: 0.2245 - accuracy: 0.9115 - val_loss: 1.4857 - val_accuracy: 0.6250
Epoch 4/10
163/163 [=====] - 339s 2s/step - loss: 0.2399 - accuracy: 0.9109 - val_loss: 0.4260 - val_accuracy: 0.8553
Epoch 5/10
163/163 [=====] - 334s 2s/step - loss: 0.2060 - accuracy: 0.9184 - val_loss: 0.3800 - val_accuracy: 0.8405

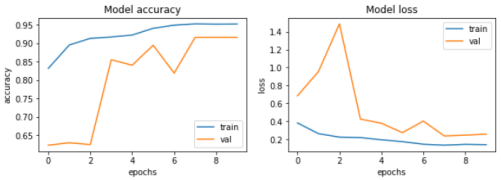
Epoch 00005: ReduceLROnPlateau reducing learning rate to 0.0003000000142492354.
Epoch 6/10
163/163 [=====] - 334s 2s/step - loss: 0.1847 - accuracy: 0.9396 - val_loss: 0.2750 - val_accuracy: 0.8947
Epoch 7/10
163/163 [=====] - 332s 2s/step - loss: 0.1473 - accuracy: 0.9483 - val_loss: 0.4036 - val_accuracy: 0.8191

Epoch 00007: ReduceLROnPlateau reducing learning rate to 9.000000427477062e-05.
Epoch 8/10
163/163 [=====] - 333s 2s/step - loss: 0.1318 - accuracy: 0.9501 - val_loss: 0.2381 - val_accuracy: 0.9161
Epoch 9/10
163/163 [=====] - 335s 2s/step - loss: 0.1385 - accuracy: 0.9551 - val_loss: 0.2475 - val_accuracy: 0.9161

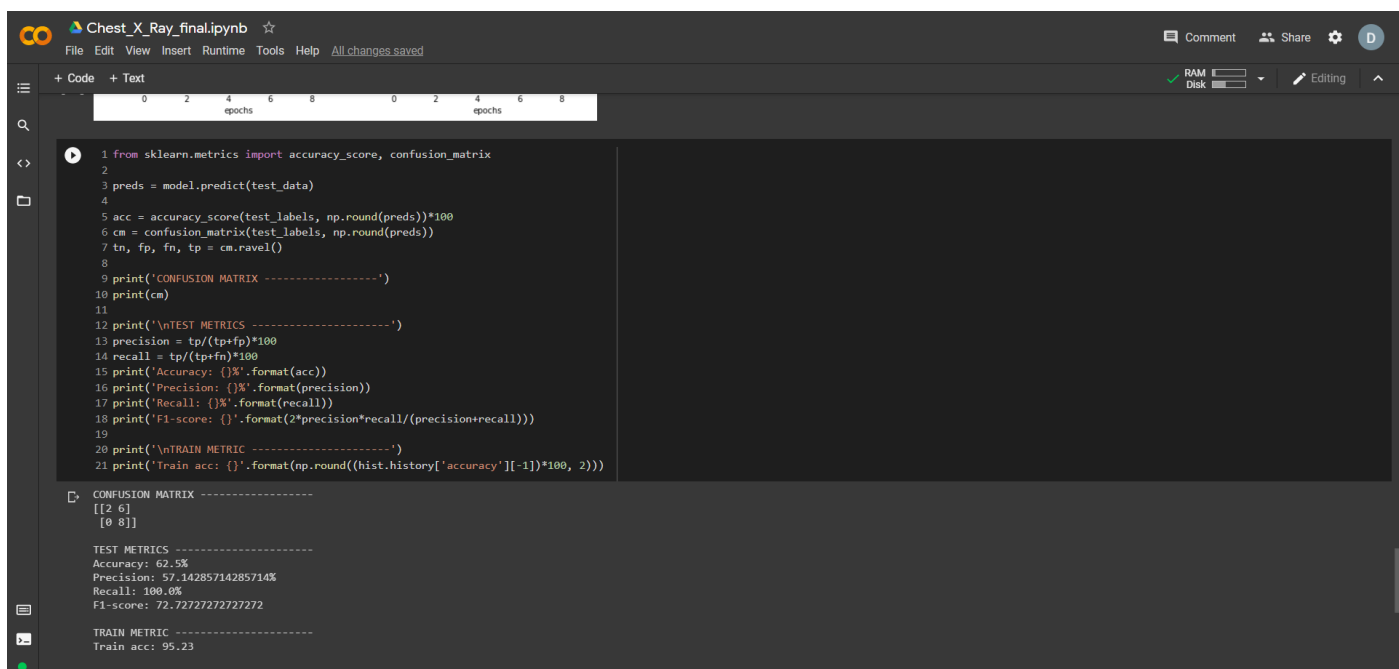
Epoch 00009: ReduceLROnPlateau reducing learning rate to 2.700000040931627e-05.
Epoch 10/10
163/163 [=====] - 333s 2s/step - loss: 0.1374 - accuracy: 0.9516 - val_loss: 0.2589 - val_accuracy: 0.9161

[11] 1 fig, ax = plt.subplots(1, 2, figsize=(10, 3))
      2 ax = ax.ravel()
      3
      4 for i, met in enumerate(['accuracy', 'loss']):
      5     ax[i].plot(hist.history[met])
      6     ax[i].plot(hist.history['val_' + met])
      7     ax[i].set_title('Model {}'.format(met))
      8     ax[i].set_xlabel('epochs')
      9     ax[i].set_ylabel(met)
     10     ax[i].legend(['train', 'val'])

Model accuracy
Model loss
```



```
[12] 1 from sklearn.metrics import accuracy_score, confusion_matrix
      2
      3 preds = model.predict(test_data)
      4
      5 acc = accuracy_score(test_labels, np.round(preds))*100
      6 cm = confusion_matrix(test_labels, np.round(preds))
      7 tn, fp, fn, tp = cm.ravel()
      8
      9 print('CONFUSION MATRIX -----')
     10 print(cm)
     11
     12 print('\nTEST METRICS -----')
```



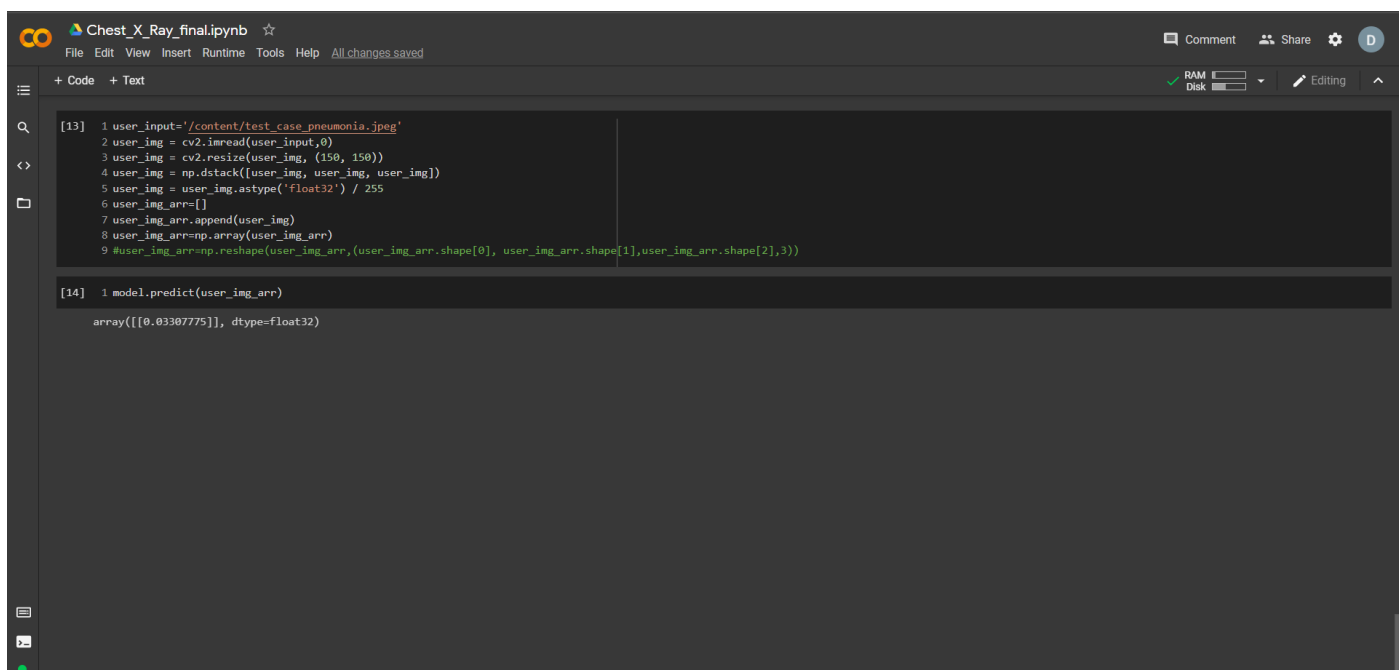
```
1 from sklearn.metrics import accuracy_score, confusion_matrix
2
3 preds = model.predict(test_data)
4
5 acc = accuracy_score(test_labels, np.round(preds))*100
6 cm = confusion_matrix(test_labels, np.round(preds))
7 tn, fp, fn, tp = cm.ravel()
8
9 print('CONFUSION MATRIX -----')
10 print(cm)
11
12 print('\nTEST METRICS -----')
13 precision = tp/(tp+fp)*100
14 recall = tp/(tp+fn)*100
15 print('Accuracy: {}'.format(acc))
16 print('Precision: {}'.format(precision))
17 print('Recall: {}'.format(recall))
18 print('F1-score: {}'.format(2*precision*recall/(precision+recall)))
19
20 print('\nTRAIN METRIC -----')
21 print('Train acc: {}'.format(np.round((hist.history['accuracy'][-1])*100, 2)))
```

CONFUSION MATRIX -----
[[2 6]
 [0 8]]

TEST METRICS -----
Accuracy: 62.5%
Precision: 57.14285714285714%
Recall: 100.0%
F1-score: 72.72727272727272

TRAIN METRIC -----
Train acc: 95.23

Steps 11 to 13:

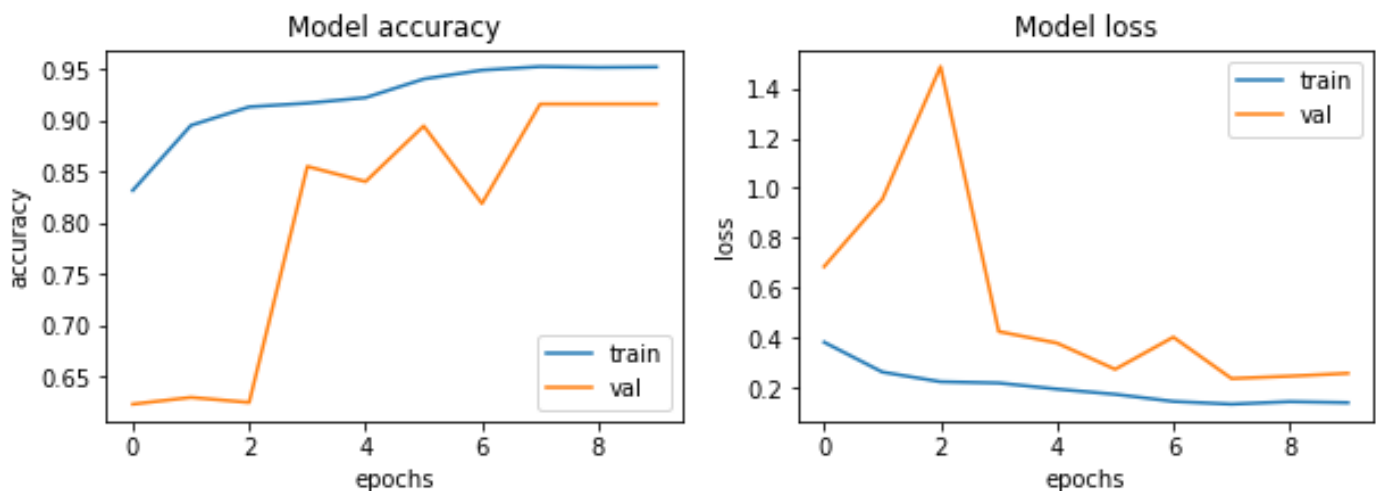


```
[13] 1 user_input='/content/test_case_pneumonia.jpeg'
2 user_img = cv2.imread(user_input,0)
3 user_img = cv2.resize(user_img, (150, 150))
4 user_img = np.dstack([user_img, user_img, user_img])
5 user_img = user_img.astype('float32') / 255
6 user_img_arr=[]
7 user_img_arr.append(user_img)
8 user_img_arr=np.array(user_img_arr)
9 #user_img_arr=np.reshape(user_img_arr,(user_img_arr.shape[0], user_img_arr.shape[1],user_img_arr.shape[2],3))

[14] 1 model.predict(user_img_arr)

array([[0.03307775]], dtype=float32)
```

7. Test results of the project:



The graphs above show the change in accuracy and loss of the model over the different training phases. The metrics have been calculated using both the training dataset and validation dataset. It is very clear from the graph that the model became more accurate as the training progressed with the model becoming more accurate in a very stable manner for the training dataset as compared to the validation dataset. This is very normal as the model is predicting on the dataset using which it was trained, i.e. the model already knows the classification of the data in the training dataset. However, for the validation dataset too the model becomes substantially accurate as the training progresses and towards the end of the training from about the 7th epoch we can see the graph for validation dataset becomes a horizontal straight line. This means the model has achieved maximum accuracy possible. As such, our choice of number of epochs seems right.

Similarly, we can see the loss function is also getting minimized as the training keeps progressing. Again considering that the model memorized the training dataset the overall low loss of the training dataset graph is very obvious. But, even for the validation dataset, the loss amount becomes very stable 7th epoch onwards. Thus, from these graphs it is clear that our choice of number of epochs was correct.

```
☞ CONFUSION MATRIX -----  
  [[2 6]  
   [0 8]]  
  
TEST METRICS -----  
Accuracy: 62.5%  
Precision: 57.14285714285714%  
Recall: 100.0%  
F1-score: 72.72727272727272  
  
TRAIN METRIC -----  
Train acc: 95.23
```

Our final accuracy for the training dataset was 95.23 which again was due to the memorization of the training dataset by the model. However, our accuracy for the test dataset was 62.5% which is not a really good value. But what is significant here is that our recall is 100%. It means that our model could actually identify all the positive cases properly. As such considering that our precision is 57.14%, we can say that our

model gave many false positives i.e. 57.14% of the total predicted positive cases were true positives while the rest were false positives. Also, our F1 score is 72.73% which is an acceptable value.



Finally we fed the above image to the model to do prediction on. This image was a positive pneumonia case.

```
array([[0.9998472]], dtype=float32)
```

The model predicted a label of 0.9998472 which meant that there was a 99.98% chance for it to have a label of 1 i.e. positive which was in correlation with our image type.

8. Conclusion:

- From the above test results it is clear that although our model may not be accurate enough, it is capable enough to properly predict positive cases of pneumonia.
- However, it may sometimes give false positives.
- As such, when detected positive by the model, a patient needs to have a manual inspection done by a doctor to verify if it is a true positive or not.
- Considering that our model had 100% recall, it is clear that the chances of false negatives are 0.
- Thus, a sample detected as pneumonia negative is surely a true negative case.
- Considering that the objective of this project was to develop a model which can detect possible positive cases of pneumonia from a set of sample images we can say that our model is successful in detecting the possible presence of the condition in the images.
- Thus, our project is successful.

9. References:

1. Convolutional neural network - Wikipedia:
https://en.wikipedia.org/wiki/Convolutional_neural_network
Published: 31/08/2013, Accessed: 26/02/2021
2. Pneumonia – Wikipedia:
<https://en.wikipedia.org/wiki/Pneumonia>
Published: 16/05/2002, Accessed: 26/02/2021
3. Chest X-Ray Images (Pneumonia) | Kaggle:
<https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia>
Published: 22/03/2018, Accessed: 23/02/2021
4. Image Classifier using CNN – GeeksforGeeks:
<https://www.geeksforgeeks.org/image-classifier-using-cnn/>
Published: 09/08/2019, Accessed: 02/03/2021
5. Basic classification: Classify images of clothing | TensorFlow Core:
<https://www.tensorflow.org/tutorials/keras/classification>
Published: 22/10/2019, Accessed: 03/03/2021