KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY (KIIT)
Deemed to be University U/S 3 of the UGC Act, 1956
School of Computer Engineering

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

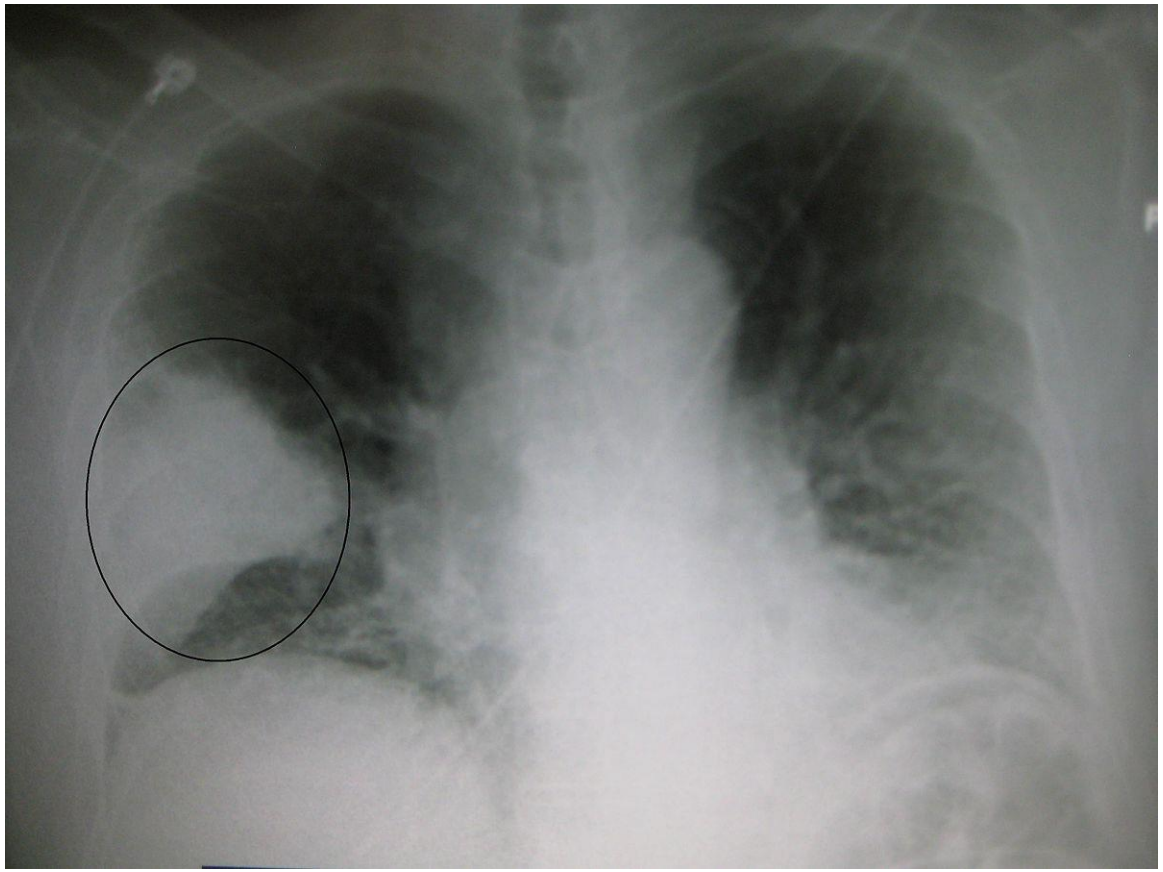
Submitted by:
Riddhiman Kumar (1729052)
Debasrito Lahiri (1729122)
Hritam Jaiswal (1729130)
Satvik Tiwari (1729155)

Under Guidance from:
Prof. Rajdeep Chatterjee

# Contents:

# Introduction

- Pneumonia condition
- High severity
- X-Ray, CT Scan and Sputum analysis diagnosis
- Manual verification of test results
- Costs of diagnosis(both time and monetary) very high
- Digitalization of X-rays
- Automation of analysis and verification of X-Rays

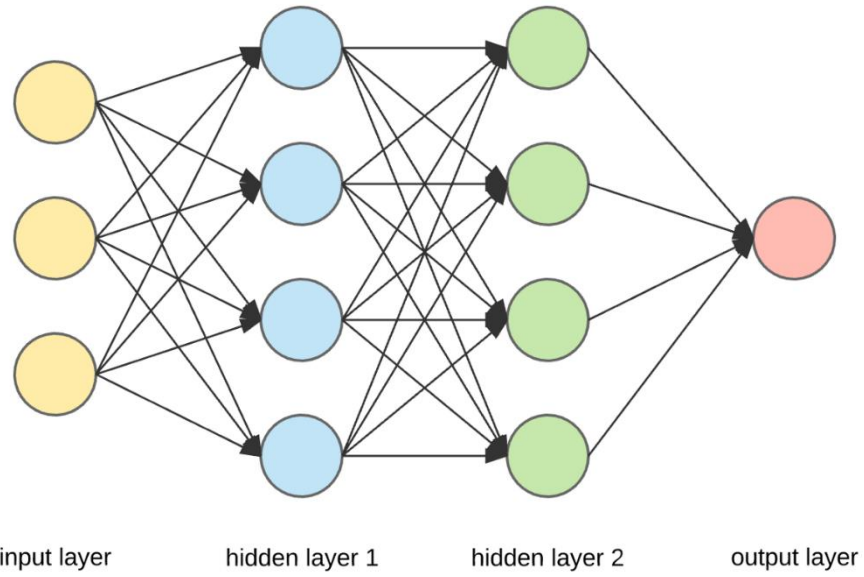


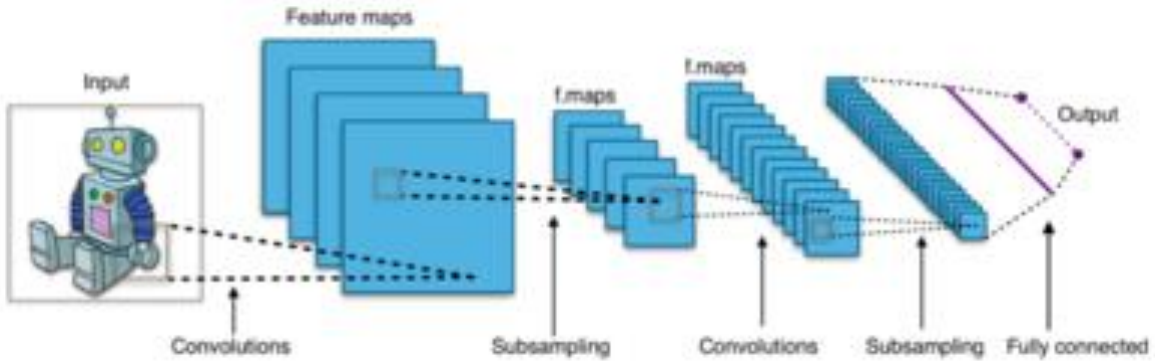
# Objectives

- Create a deep learning model using CNN
- Train it to predict the possible presence of pneumonia in chest X-ray images
- Verify the different metrics of the performance of the model on a test set of images
- Predict the possible presence of pneumonia on a single chest X-Ray image of known type to verify the model

# Convolution Neural Network

- Deep learning model
- Based on deep neural networks
- Uses convolution in each layer of network to extract feature from feature map
- Consists of input layer, output layer, convolution layers and fully connected layer.
- 5 convolution layers
- 16 filters in first layer and doubled in every consecutive layer
- Relu activation function
- Adam optimizer
- Validation using accuracy metric
- 10 epochs



# Performance metrics

- Accuracy: It is how many of the predictions done are correct.
- Accuracy=(TP+TN)/Total population
- Precision: It is how many of the positively identified cases were actually positive
- Precision=TP/(TP+FP)
- Recall: It is the ability of a model to correctly predict all the positive cases in a sample as positive
- Recall=TP/(TP+FN)
- F1-Score: It is the harmonic mean of precision and recall and gives an idea of a balance between the two
- F1-Score=2*(Precision*Recall)/(Precision+Recall)
- Loss: It is the amount of error/penalty for error in a prediction
- Here we use binary cross entropy as the loss function
- Confusion matrix: It is a matrix representing the TN,TP,FN,FP values
- Confusion matrix=

| | |
|---|---|
| True +ve | False +ve |
| False -ve | True -ve |

# Project breakdown: Part 1

- Importing of libraries and setting seed
- Importing of dataset and extracting the data from it
- Verification of dataset
- Defining data pre-processing function for labelling the testing dataset and data augmentation of training and validation datasets using ImageDataGenerator of keras

# Project breakdown: Part 2

- Setting the batch size, epochs, image dimension variables and using the function created in pervious step for data pre-processing
- Setting up the CNN layers and compiling them to create a model along with generation of callbacks
- Training the model using training data and validation data for tuning the model
- Plotting of graph showing the model losses and accuracy changes over the different epochs of the training

# Project breakdown: Part 3

- Performing predictions using tuned model on test dataset
- Checking of different performance metrics for the test dataset predictions
- Importing of another single image to perform predictions on
- Pre-processing of the image
- Performing prediction on the image

# Implementation

Chest_X_Ray_final.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing

```python
# General libraries
import os
import numpy as np
import pandas as pd
import random
import cv2
import matplotlib.pyplot as plt
%matplotlib inline

# Deep learning libraries
import keras.backend as K
from keras.models import Model, Sequential
from keras.layers import Input, Dense, Flatten, Dropout, BatchNormalization
from keras.layers import Conv2D, SeparableConv2D, MaxPool2D, LeakyReLU, Activation
from keras.optimizers import Adam
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import ModelCheckpoint, ReduceLROnPlateau, EarlyStopping
import tensorflow as tf

# Setting seeds for reproducibility
seed = 232
np.random.seed(seed)
tf.compat.v1.set_random_seed(seed)
```

[2]

```python
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[3]

```
ls
```

```
drive/  sample_data/
```

[4]

```
!unzip -uq "/content/drive/MyDrive/input.zip" -d "/content/drive/MyDrive/"
```

0s completed at 1:34 AM

# Implementation

Chest_X_Ray_final.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

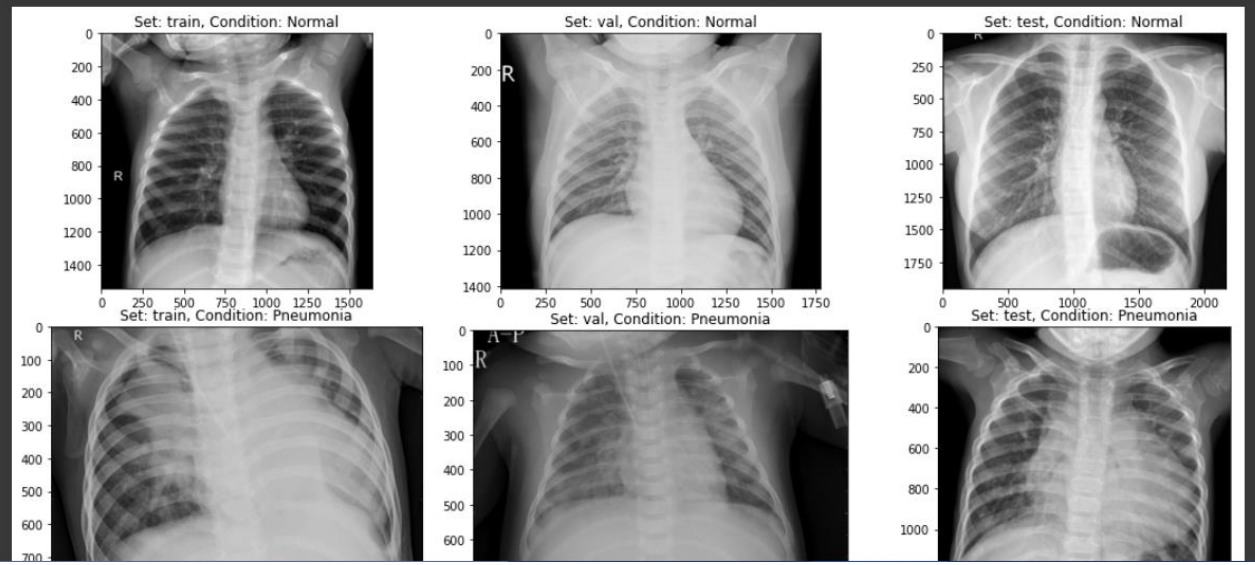
+ Code + Text

RAM Disk Editing

```
input_path = '../content/drive/MyDrive/input/chest-xray-pneumonia/chest_xray/chest_xray/'

fig, ax = plt.subplots(2, 3, figsize=(15, 7))
ax = ax.ravel()
plt.tight_layout()

for i, _set in enumerate(['train', 'val', 'test']):
    set_path = input_path+_set
    ax[i].imshow(plt.imread(set_path+'/NORMAL/'+os.listdir(set_path+'/NORMAL')[0]), cmap='gray')
    ax[i].set_title('Set: {}, Condition: Normal'.format(_set))
    ax[i+3].imshow(plt.imread(set_path+'/PNEUMONIA/'+os.listdir(set_path+'/PNEUMONIA')[0]), cmap='gray')
    ax[i+3].set_title('Set: {}, Condition: Pneumonia'.format(_set))
```



# Implementation

Chest_X_Ray_final.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing

[6]

```python
# Distribution of our datasets
for _set in ['train', 'val', 'test']:
    n_normal = len(os.listdir(input_path + _set + '/NORMAL'))
    n_infect = len(os.listdir(input_path + _set + '/PNEUMONIA'))
    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
```

```python
# input_path = '../input/chest_xray/chest_xray/'
#input_path = '../input/chest-xray-pneumonia//chest_xray/chest_xray/'
input_path = '../content/drive/MyDrive/input/chest-xray-pneumonia/chest_xray/chest_xray/'

def process_data(img_dims, batch_size):
    # Data generation objects
    train_datagen = ImageDataGenerator(rescale=1./255, zoom_range=0.3, vertical_flip=True)
    test_val_datagen = ImageDataGenerator(rescale=1./255)

    # This is fed to the network in the specified batch sizes and image dimensions
    train_gen = train_datagen.flow_from_directory(
    directory=input_path+'train',
    target_size=(img_dims, img_dims),
    batch_size=batch_size,
    class_mode='binary',
    shuffle=True)

    test_gen = test_val_datagen.flow_from_directory(
    directory=input_path+'val',
    target_size=(img_dims, img_dims),
    batch_size=batch_size,
    class_mode='binary',
    shuffle=True)

    # I will be making predictions off of the test set in one batch size
    # This is useful to be able to get the confusion matrix
    test_data = []
```

# Implementation

Chest_X_Ray_final.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing

```python
# This is fed to the network in the specified batch sizes and image dimensions
train_gen = train_datagen.flow_from_directory(
directory=input_path+'train',
target_size=(img_dims, img_dims),
batch_size=batch_size,
class_mode='binary',
shuffle=True)

test_gen = test_val_datagen.flow_from_directory(
directory=input_path+'val',
target_size=(img_dims, img_dims),
batch_size=batch_size,
class_mode='binary',
shuffle=True)

# I will be making predictions off of the test set in one batch size
# This is useful to be able to get the confusion matrix
test_data = []
test_labels = []

for cond in ['/NORMAL/', '/PNEUMONIA/']:
    for img in (os.listdir(input_path + 'test' + cond)):
        img = plt.imread(input_path+'test'+cond+img)
        img = cv2.resize(img, (img_dims, img_dims))
        img = np.dstack([img, img, img])
        img = img.astype('float32') / 255
        if cond=='/NORMAL/':
            label = 0
        elif cond=='/PNEUMONIA/':
            label = 1
        test_data.append(img)
        test_labels.append(label)

test_data = np.array(test_data)
test_labels = np.array(test_labels)

return train_gen, test_gen, test_data, test_labels
```

# Implementation

Chest_X_Ray_final.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share D

+ Code + Text

RAM Disk Editing

[8]

```python
# Hyperparameters
img_dims = 150
epochs = 10
batch_size = 32

# Getting the data
train_gen, test_gen, test_data, test_labels = process_data(img_dims, batch_size)
```

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

```python
# Input layer
inputs = Input(shape=(img_dims, img_dims, 3))

# First conv block
x = Conv2D(filters=16, kernel_size=(3, 3), activation='relu', padding='same')(inputs)
x = Conv2D(filters=16, kernel_size=(3, 3), activation='relu', padding='same')(x)
x = MaxPool2D(pool_size=(2, 2))(x)

# Second conv block
x = SeparableConv2D(filters=32, kernel_size=(3, 3), activation='relu', padding='same')(x)
x = SeparableConv2D(filters=32, kernel_size=(3, 3), activation='relu', padding='same')(x)
x = BatchNormalization()(x)
x = MaxPool2D(pool_size=(2, 2))(x)

# Third conv block
x = SeparableConv2D(filters=64, kernel_size=(3, 3), activation='relu', padding='same')(x)
x = SeparableConv2D(filters=64, kernel_size=(3, 3), activation='relu', padding='same')(x)
x = BatchNormalization()(x)
x = MaxPool2D(pool_size=(2, 2))(x)

# Fourth conv block
x = SeparableConv2D(filters=128, kernel_size=(3, 3), activation='relu', padding='same')(x)
x = SeparableConv2D(filters=128, kernel_size=(3, 3), activation='relu', padding='same')(x)
x = BatchNormalization()(x)
x = MaxPool2D(pool_size=(2, 2))(x)
x = Dropout(rate=0.2)(x)
```

# Implementation

Chest_X_Ray_final.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share D

+ Code + Text

RAM Disk Editing

```python
x = MaxPool2D(pool_size=(2, 2))(x)

# Fourth conv block
x = SeparableConv2D(filters=128, kernel_size=(3, 3), activation='relu', padding='same')(x)
x = SeparableConv2D(filters=128, kernel_size=(3, 3), activation='relu', padding='same')(x)
x = BatchNormalization()(x)
x = MaxPool2D(pool_size=(2, 2))(x)
x = Dropout(rate=0.2)(x)

# Fifth conv block
x = SeparableConv2D(filters=256, kernel_size=(3, 3), activation='relu', padding='same')(x)
x = SeparableConv2D(filters=256, kernel_size=(3, 3), activation='relu', padding='same')(x)
x = BatchNormalization()(x)
x = MaxPool2D(pool_size=(2, 2))(x)
x = Dropout(rate=0.2)(x)

# FC layer
x = Flatten()(x)
x = Dense(units=512, activation='relu')(x)
x = Dropout(rate=0.7)(x)
x = Dense(units=128, activation='relu')(x)
x = Dropout(rate=0.5)(x)
x = Dense(units=64, activation='relu')(x)
x = Dropout(rate=0.3)(x)

# Output layer
output = Dense(units=1, activation='sigmoid')(x)

# Creating model and compiling
model = Model(inputs=inputs, outputs=output)
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Callbacks
checkpoint = ModelCheckpoint(filepath='best_weights.hdf5', save_best_only=True, save_weights_only=True)
lr_reduce = ReduceLROnPlateau(monitor='val_loss', factor=0.3, patience=2, verbose=2, mode='max')
early_stop = EarlyStopping(monitor='val_loss', min_delta=0.1, patience=1, mode='min')
```

# Implementation

Chest_X_Ray_final.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing

[9]
```
early_stop = EarlyStopping(monitor='val_loss', min_delta=0.1, patience=1, mode='min')
```

```
# Fitting the model
hist = model.fit_generator(
        train_gen, steps_per_epoch=train_gen.samples // batch_size,
        epochs=epochs, validation_data=test_gen,
        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`, whic
  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]
```
fig, ax = plt.subplots(1, 2, figsize=(10, 3))
```

# Implementation

Chest_X_Ray_final.ipynb

File Edit View Insert Runtime Tools Help All changes saved

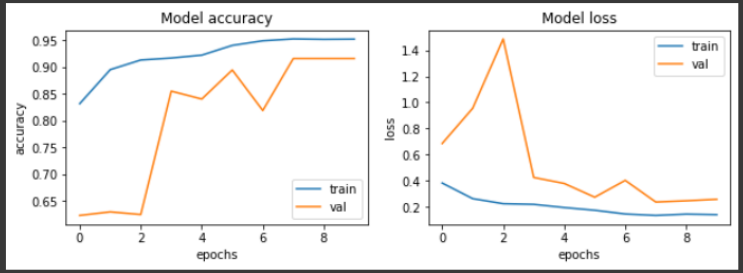
Comment Share

+ Code + Text

RAM Disk Editing

```
fig, ax = plt.subplots(1, 2, figsize=(10, 3))
ax = ax.ravel()

for i, met in enumerate(['accuracy', 'loss']):
    ax[i].plot(hist.history[met])
    ax[i].plot(hist.history['val_' + met])
    ax[i].set_title('Model {}'.format(met))
    ax[i].set_xlabel('epochs')
    ax[i].set_ylabel(met)
    ax[i].legend(['train', 'val'])
```




[12]

```
from sklearn.metrics import accuracy_score, confusion_matrix

preds = model.predict(test_data)

acc = accuracy_score(test_labels, np.round(preds))*100
cm = confusion_matrix(test_labels, np.round(preds))
tn, fp, fn, tp = cm.ravel()

print('CONFUSION MATRIX ------------------')
print(cm)

print('\nTEST METRICS ----------------------')
```

# Implementation

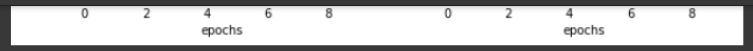
Chest_X_Ray_final.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing




```python
from sklearn.metrics import accuracy_score, confusion_matrix

preds = model.predict(test_data)

acc = accuracy_score(test_labels, np.round(preds))*100
cm = confusion_matrix(test_labels, np.round(preds))
tn, fp, fn, tp = cm.ravel()

print('CONFUSION MATRIX ------------------')
print(cm)

print('\nTEST METRICS ----------------------')
precision = tp/(tp+fp)*100
recall = tp/(tp+fn)*100
print('Accuracy: {}%'.format(acc))
print('Precision: {}%'.format(precision))
print('Recall: {}%'.format(recall))
print('F1-score: {}'.format(2*precision*recall/(precision+recall)))

print('\nTRAIN METRIC ----------------------')
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
```

# Implementation

Chest_X_Ray_final.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share D

+ Code + Text

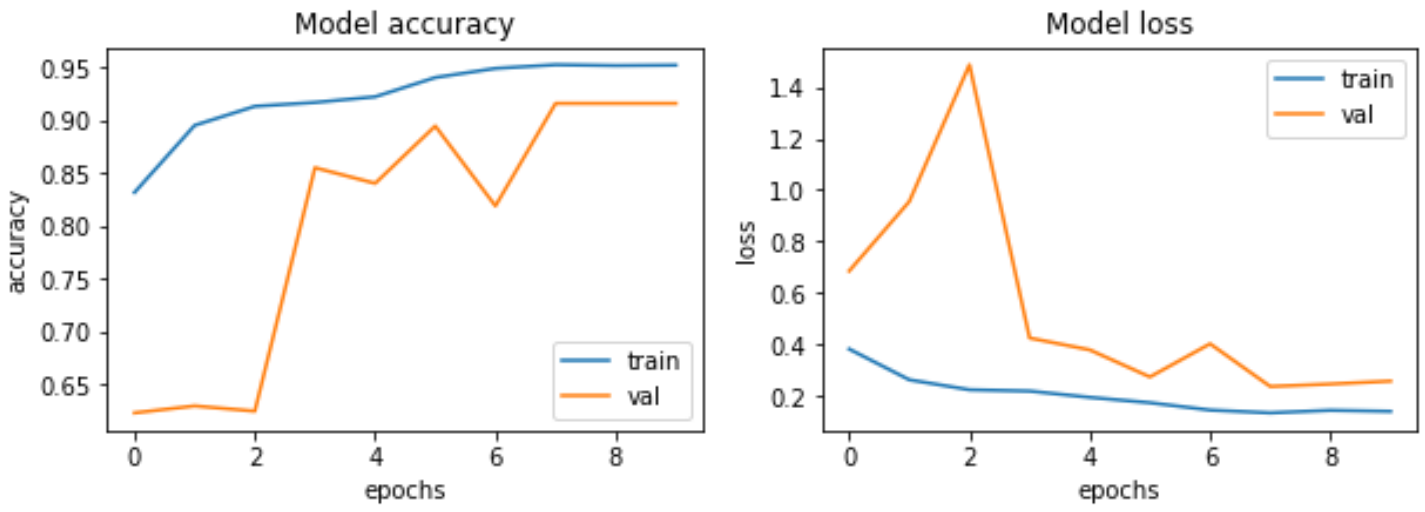
RAM Disk Editing

[13]
```
user_input='/content/test_case_pneumonia.jpeg'
user_img = cv2.imread(user_input,0)
user_img = cv2.resize(user_img, (150, 150))
user_img = np.dstack([user_img, user_img, user_img])
user_img = user_img.astype('float32') / 255
user_img_arr=[]
user_img_arr.append(user_img)
user_img_arr=np.array(user_img_arr)
#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]
```
model.predict(user_img_arr)
```

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

# Test Results




- Accuracy increases
- Loss decreases
- Saturation at $7^{th}$ epoch

# Test Results

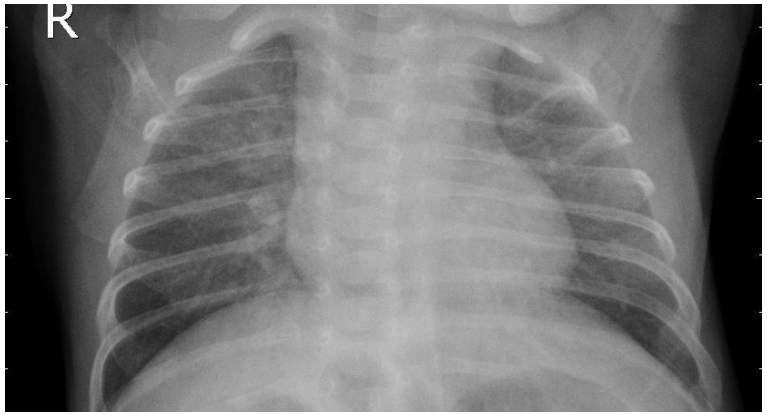
```
CONFUSION MATRIX ------------------
[[2 6]
 [0 8]]

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

TRAIN METRIC ----------------------
Train acc: 95.23
```

- Recall 100%
- Could predict all positive cases as positive
- 57.14% precision
- 57.14% of all predicted positive cases are actually positive
- Accuracy 62.5% and F1-Score 72.73% due to low precision

# Test Results




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

- Original image positive case
- Model predicted 99.98% chance of positive case
- Test result in correlation with actual observations

# 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.

# References


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

# Thank You!