```
!pip install kaggle
```

```
!ls -a
```

```
!mkdir .kaggle

import json
token = {"username":"sivaganesh1988","key":"778f842999ef35c2050806923bf27d8d"}
with open('/content/.kaggle/kaggle.json', 'w') as file:
    json.dump(token, file)


!chmod 600 /root/.kaggle/kaggle.json

# may run into problems, see updated version below
!cp /content/.kaggle/kaggle.json ~/.kaggle/kaggle.json

!kaggle config set -n path -v{/content}
```

```
!kaggle datasets download -d nih-chest-xrays/data -p /content
```

```
!ls
```

```
!unzip data.zip -d input
```

```
import pandas as pd
data = pd.read_csv("/content/input/Data_Entry_2017.csv")
data.head()
```

```
data.columns = ['Image_Index', 'Finding_Labels', 'Follow_Up_#', 'Patient_ID', 'Patient_Age
                'View_Position', 'Original_Image_Width', 'Original_Image_Height',
                'Original_Image_Pixel_Spacing_X', 'Original_Image_Pixel_Spacing_Y', 'Unnam
```

```
data.drop('Unnamed', axis=1, inplace=True)
```

```
data.head()
```

```
data.Finding_Labels.value_counts().head(20)
```

```
data.drop(['Original_Image_Pixel_Spacing_X', 'Original_Image_Pixel_Spacing_Y'], axis = 1,
data.drop(['Original_Image_Width', 'Original_Image_Height'], axis = 1, inplace = True)
data.head()
```

	Image_Index	Finding_Labels	Follow_Up_#	Patient_ID	Patient_Age	Pat
0	00000001_000.png	Cardiomegaly	0	1	58	
1	00000001_001.png	Cardiomegaly|Emphysema	1	1	58	
2	00000001_002.png	Cardiomegaly|Effusion	2	1	58	
3	00000002_000.png	No Finding	0	2	81	
4	00000003_000.png	Hernia	0	3	81	

```
total_visits = data.groupby('Patient_ID')['Image_Index'].count().reset_index()
total_visits.sort_values(by="Image_Index", ascending=False).head(10)
```

	Patient_ID	Image_Index
10006	10007	184
13669	13670	173
15529	15530	158
12833	12834	157
13992	13993	143
1835	1836	137
19123	19124	130
20212	20213	119
17137	17138	117
11236	11237	116

**Resize **

```
import cv2
import os
import time


def create_directory(directory):
    """
    Creates a new folder in the specified directory if folder doesn't exist.
    INPUT
        directory: Folder to be created, called as "folder/".
    OUTPUT
        New folder in the current directory.
    """
    if not os.path.exists(directory):
```

```
        os.makedirs(directory)


def crop_and_resize_images(path, new_path, img_size):
    """
    Crops, resizes, and stores all images from a directory in a new directory.
    INPUT
        path: Path where the current, unscaled images are contained.
        new_path: Path to save the resized images.
        img_size: New size for the rescaled images.
    OUTPUT
        All images cropped, resized, and saved to the new folder.
    """
    create_directory(new_path)
    dirs = [l for l in os.listdir(path) if l != '.DS_Store']
    # total = 0

    for item in dirs:
        # Read in all images as grayscale
        img = cv2.imread(path + item, cv2.IMREAD_GRAYSCALE)
        img = cv2.resize(img, (img_size, img_size))
        cv2.imwrite(str(new_path + item), img)
        # total += 1
        # print("Saving: ", item, total)


if __name__ == '__main__':
    start_time = time.time()
    crop_and_resize_images(path='/content/input/images_001/images/', new_path='/content/in
  #  crop_and_resize_images(path='/content/input/images_002/images/', new_path='/content/i
  #  crop_and_resize_images(path='/content/input/images_003/images/', new_path='/content/i
  #  crop_and_resize_images(path='/content/input/images_004/images/', new_path='/content/i
  #  crop_and_resize_images(path='/content/input/images_005/images/', new_path='/content/i
  #  crop_and_resize_images(path='/content/input/images_006/images/', new_path='/content/i
  #  crop_and_resize_images(path='/content/input/images_007/images/', new_path='/content/i
  #  crop_and_resize_images(path='/content/input/images_008/images/', new_path='/content/i
  #  crop_and_resize_images(path='/content/input/images_009/images/', new_path='/content/i
   # crop_and_resize_images(path='/content/input/images_010/images/', new_path='/content/i
 #   crop_and_resize_images(path='/content/input/images_011/images/', new_path='/content/i
 #   crop_and_resize_images(path='/content/input/images_012/images/', new_path='/content/i

    print("Seconds: ", time.time() - start_time)
```

```
Seconds:  72.4887957572937
```

reconcile_label

```
import os

import pandas as pd


def get_lst_images(file_path):
    """
```

```
    Reads in all files from file path into a list.
    INPUT
        file_path: specified file path containing the images.
    OUTPUT
        List of image strings
    """
    return [i for i in os.listdir(file_path) if i != '.DS_Store']


if __name__ == '__main__':
    data = pd.read_csv("/content/input/Data_Entry_2017.csv")
    sample = os.listdir('/content/input/resized-256/')

    sample = pd.DataFrame({'Image Index': sample})

    sample = pd.merge(sample, data, how='left', on='Image Index')

    sample.columns = ['Image_Index', 'Finding_Labels', 'Follow_Up_#', 'Patient_ID',
                      'Patient_Age', 'Patient_Gender', 'View_Position',
                      'Original_Image_Width', 'Original_Image_Height',
                      'Original_Image_Pixel_Spacing_X',
                      'Original_Image_Pixel_Spacing_Y', 'Unnamed']

    sample['Finding_Labels'] = sample['Finding_Labels'].apply(lambda x: x.split('|')[0])

    sample.drop(['Original_Image_Pixel_Spacing_X', 'Original_Image_Pixel_Spacing_Y', 'Unna
    sample.drop(['Original_Image_Width', 'Original_Image_Height'], axis=1, inplace=True)

    print("Writing CSV")
    sample.to_csv('/content/input/sample_labels.csv', index=False, header=True)
```

```
Writing CSV
```

Images to Array

```
import time

import cv2
import numpy as np
import pandas as pd


def convert_images_to_arrays(file_path, df):
    """
    Converts each image to an array, and appends each array to a new NumPy
    array, based on the image column equaling the image file name.
    INPUT
        file_path: Specified file path for resized test and train images.
        df: Pandas DataFrame being used to assist file imports.
    OUTPUT
        NumPy array of image arrays.
    """

    lst_imgs = [l for l in df['Image_Index']]
```

```
    return np.array([np.array(cv2.imread(file_path + img, cv2.IMREAD_GRAYSCALE)) for img i


def save_to_array(arr_name, arr_object):
    """
    Saves data object as a NumPy file. Used for saving train and test arrays.
    INPUT
        arr_name: The name of the file you want to save.
            This input takes a directory string.
        arr_object: NumPy array of arrays. This object is saved as a NumPy file.
    """
    return np.save(arr_name, arr_object)


if __name__ == '__main__':
    start_time = time.time()

    labels = pd.read_csv("/content/input/sample_labels.csv")

    print("Writing Train Array")
    X_train = convert_images_to_arrays('/content/input/resized-256/', labels)

    print(X_train.shape)

    print("Saving Train Array")
    save_to_array('/content/input/X_sample.npy', X_train)

    print("Seconds: ", round(time.time() - start_time), 2)
```

```
Writing Train Array
(4999, 256, 256)
Saving Train Array
Seconds:  5 2
```

```
from __future__ import absolute_import, division, print_function, unicode_literals

try:
  # %tensorflow_version only exists in Colab.
  %tensorflow_version 2.x
except Exception:
  pass
import tensorflow as tf
print("Num GPUs Available: ", len(tf.config.experimental.list_physical_devices('GPU')))
```

Model

```
import numpy as np
import pandas as pd
from keras.callbacks import EarlyStopping
from keras.callbacks import TensorBoard
```

```
from keras.layers import Dense, Activation, Flatten, Dropout
from keras.layers import MaxPooling2D
from keras.layers.convolutional import Conv2D
from keras.models import Sequential
from keras.utils import np_utils
from keras.utils import multi_gpu_model
from sklearn.metrics import precision_score, recall_score, f1_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

def split_data(X, y, test_data_size):
    '''
    Split data into test and training datasets.
    INPUT
        X: NumPy array of arrays
        y: Pandas series, which are the labels for input array X
        test_data_size: size of test/train split. Value from 0 to 1
    OUPUT
        Four arrays: X_train, X_test, y_train, and y_test
    '''
    return train_test_split(X, y, test_size=test_data_size, random_state=42)



def reshape_data(arr, img_rows, img_cols, channels):
    '''
    Reshapes the data into format for CNN.
    INPUT
        arr: Array of NumPy arrays.
        img_rows: Image height
        img_cols: Image width
        channels: Specify if the image is grayscale (1) or RGB (3)
    OUTPUT
        Reshaped array of NumPy arrays.
    '''
    return arr.reshape(arr.shape[0], img_rows, img_cols, channels)

def cnn_model(X_train, y_train, kernel_size, nb_filters, channels, nb_epoch, batch_size, n
    '''
    Define and run the Convolutional Neural Network
    INPUT
        X_train: Array of NumPy arrays
        X_test: Array of NumPy arrays
        y_train: Array of labels
        y_test: Array of labels
        kernel_size: Initial size of kernel
        nb_filters: Initial number of filters
        channels: Specify if the image is grayscale (1) or RGB (3)
        nb_epoch: Number of epochs
        batch_size: Batch size for the model
        nb_classes: Number of classes for classification
    OUTPUT
        Fitted CNN model
    '''

    model = Sequential()
```

```
'''
First set of three layers
Image size: 256 x 256
nb_filters = 32
kernel_size = (2,2)
'''

model.add(Conv2D(nb_filters, (kernel_size[0], kernel_size[1]),
                 padding='valid',
                 strides=1,
                 input_shape=(img_rows, img_cols, channels)))
model.add(Activation('relu'))

model.add(Conv2D(nb_filters, (kernel_size[0], kernel_size[1])))
model.add(Activation('relu'))

model.add(Conv2D(nb_filters, (kernel_size[0], kernel_size[1])))
model.add(Activation('relu'))

model.add(MaxPooling2D(pool_size=(2, 2)))

'''
Second set of three layers
Image Size: 128 x 128
nb_filters = 64
kernel_size = 4,4
'''

nb_filters = 64
kernel_size = (4, 4)

model.add(Conv2D(nb_filters, (kernel_size[0], kernel_size[1])))
model.add(Activation('relu'))

model.add(Conv2D(nb_filters, (kernel_size[0], kernel_size[1])))
model.add(Activation('relu'))

# model.add(Conv2D(nb_filters, (kernel_size[0], kernel_size[1])))
# model.add(Activation('relu'))

model.add(MaxPooling2D(pool_size=(2, 2)))

'''
Third set of three layers
Image Size: 64 x 64
nb_filters = 128
kernel_size = 8,8
'''

nb_filters = 128
kernel_size = (8, 8)

model.add(Conv2D(nb_filters, (kernel_size[0], kernel_size[1])))
model.add(Activation('relu'))
```

```
        model.add(Conv2D(nb_filters, (kernel_size[0], kernel_size[1])))
        model.add(Activation('relu'))

        # model.add(Conv2D(nb_filters, (kernel_size[0], kernel_size[1])))
        # model.add(Activation('relu'))

        model.add(MaxPooling2D(pool_size=(12, 12)))

        model.add(Flatten())
        print("Model flattened out to: ", model.output_shape)

        model.add(Dense(4096))
        model.add(Activation('relu'))
        model.add(Dropout(0.2))

        model.add(Dense(4096))
        model.add(Activation("relu"))
        model.add(Dropout(0.2))

        model.add(Dense(nb_classes))
        model.add(Activation('softmax'))

        #model = multi_gpu_model(model)
        model.compile(loss='categorical_crossentropy',
                      optimizer='adam',
                      metrics=['accuracy'])

        print(model.summary())

        stop = EarlyStopping(monitor='acc',
                             min_delta=0.001,
                             patience=2,
                             verbose=0,
                             mode='auto')

        tensor_board = TensorBoard(log_dir='./Graph', histogram_freq=0, write_graph=True, writ

        model.fit(X_train, y_train, batch_size=batch_size, epochs=nb_epoch,
                  verbose=1,
                  validation_split=0.2,
                  class_weight='auto',
                  callbacks=[stop, tensor_board]
                  )
        return model

    if __name__ == '__main__':

        batch_size = 100
        nb_classes = 15
        nb_epoch = 20
        #nb_gpus = 2
        img_rows, img_cols = 256,256
        channels = 1
        nb_filters = 32
```

```
kernel_size = (2, 2)

# Import data
labels = pd.read_csv("/content/input/sample_labels.csv")
X = np.load("/content/input/X_sample.npy")

y = labels.Finding_Labels
# y = np.array(pd.get_dummies(y))
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(y)
y = y.reshape(-1, 1)

print("Splitting data into test/ train datasets")
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

print("Reshaping Data")
X_train = X_train.reshape(X_train.shape[0], img_rows, img_cols, channels)
X_test = X_test.reshape(X_test.shape[0], img_rows, img_cols, channels)

print("X_train Shape: ", X_train.shape)
print("X_test Shape: ", X_test.shape)

input_shape = (img_rows, img_cols, channels)

print("Normalizing Data")
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')

X_train /= 255
X_test /= 255

y_train = np_utils.to_categorical(y_train, nb_classes)
y_test = np_utils.to_categorical(y_test, nb_classes)
print("y_train Shape: ", y_train.shape)
print("y_test Shape: ", y_test.shape)


model = cnn_model(X_train, y_train, kernel_size, nb_filters, channels, nb_epoch, batch

print("Predicting")
y_pred = model.predict(X_test)

y_test = np.argmax(y_test, axis=1)
y_pred = np.argmax(y_pred, axis=1)

precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average="weighted")
print("Precision: ", precision)
print("Recall: ", recall)
print("F1: ", f1)
```

```
Splitting data into test/ train datasets
Reshaping Data
X_train Shape:  (3999, 256, 256, 1)
X_test Shape:  (1000, 256, 256, 1)
Normalizing Data
y_train Shape:  (3999, 15)
y_test Shape:  (1000, 15)
Model flattened out to:  (None, 1152)
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/optimizers.py:79

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorfl

Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_36 (Conv2D)           (None, 255, 255, 32)      160
_________________________________________________________________
activation_51 (Activation)   (None, 255, 255, 32)      0
_________________________________________________________________
conv2d_37 (Conv2D)           (None, 254, 254, 32)      4128
_________________________________________________________________
activation_52 (Activation)   (None, 254, 254, 32)      0
_________________________________________________________________
conv2d_38 (Conv2D)           (None, 253, 253, 32)      4128
_________________________________________________________________
activation_53 (Activation)   (None, 253, 253, 32)      0
_________________________________________________________________
max_pooling2d_16 (MaxPooling (None, 126, 126, 32)      0
_________________________________________________________________
conv2d_39 (Conv2D)           (None, 123, 123, 64)      32832
_________________________________________________________________
activation_54 (Activation)   (None, 123, 123, 64)      0
_________________________________________________________________
conv2d_40 (Conv2D)           (None, 120, 120, 64)      65600
_________________________________________________________________
activation_55 (Activation)   (None, 120, 120, 64)      0
_________________________________________________________________
max_pooling2d_17 (MaxPooling (None, 60, 60, 64)        0
_________________________________________________________________
conv2d_41 (Conv2D)           (None, 53, 53, 128)       524416
_________________________________________________________________
activation_56 (Activation)   (None, 53, 53, 128)       0
_________________________________________________________________
conv2d_42 (Conv2D)           (None, 46, 46, 128)       1048704
_________________________________________________________________
activation_57 (Activation)   (None, 46, 46, 128)       0
_________________________________________________________________
max_pooling2d_18 (MaxPooling (None, 3, 3, 128)         0
_________________________________________________________________
flatten_6 (Flatten)          (None, 1152)              0
_________________________________________________________________
dense_16 (Dense)             (None, 4096)              4722688
_________________________________________________________________
activation_58 (Activation)   (None, 4096)              0
_________________________________________________________________
dropout_11 (Dropout)         (None, 4096)              0
_________________________________________________________________
dense_17 (Dense)             (None, 4096)              16781312
_________________________________________________________________
activation_59 (Activation)   (None, 4096)              0
```

```
_________________________________________________________________
dropout_12 (Dropout)         (None, 4096)              0         
_________________________________________________________________
dense_18 (Dense)             (None, 15)                61455     
_________________________________________________________________
activation_60 (Activation)   (None, 15)                0         
=================================================================
Total params: 23,245,423
Trainable params: 23,245,423
Non-trainable params: 0
_________________________________________________________________
None
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow_core/python
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorfl

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorfl

Train on 3199 samples, validate on 800 samples
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/callbacks.py:112

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/callbacks.py:112

Epoch 1/20
3199/3199 [==============================] - 64s 20ms/step - loss: 1.9626 - acc: 0.53
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/callbacks.py:126

Epoch 2/20
3199/3199 [==============================] - 42s 13ms/step - loss: 1.7048 - acc: 0.55
Epoch 3/20
3199/3199 [==============================] - 42s 13ms/step - loss: 1.6989 - acc: 0.55
Epoch 4/20
3199/3199 [==============================] - 42s 13ms/step - loss: 1.6961 - acc: 0.55
Predicting
Precision:  0.3025
Recall:  0.55
F1:  0.39032258064516134
/usr/local/lib/python3.6/dist-packages/sklearn/metrics/classification.py:1437: Undefi
  'precision', 'predicted', average, warn_for)
/usr/local/lib/python3.6/dist-packages/sklearn/metrics/classification.py:1437: Undefi
  'precision', 'predicted', average, warn_for)
```