Chest X-Ray Images (Pneumonia)

Abdelrahman Mohamed Abdallah

I. Introduction

. Chest X-ray images (Pneumonia) are radiographs of the chest that are used to view areas such as the lungs, heart, tissues, and bones that affects approximately 450 million people every year all over the world. This inflammation is caused by either virus or bacteria that infect the alveoli, sometimes other microorganisms cause infection too. They can be used to diagnose infections like pneumonia or to detect other conditions like cancer or heart disease. An x-ray shows abnormalities by appearing different from a normal image on the film; it may be denser, lighter, or contain dark spots or lines representing infection or inflammation. By using Chest X- Ray Images (Pneumonia), doctors can quickly identify and treat any potential health complications more effectively.

II. Related Work

Researchers have been working on medical and biomedical topics since a very long time ,however, Pneumonia has seen an upward trend lately due to the novel Coronavirus pandemic which is also caused by virus in the alveoli. So progress in detecting Pneumonia means progress in putting an end or at least controlling.

McDonald et al., “Automated Detection of Pediatric Pneumonia in Chest X-ray Images Using Deep Convolutional Neural Networks”, IEEE Transactions on Biomedical Engineering, 2018.

III. Methodology

III.I dAta

Data acquisition: The first step in developing a methodology for Chest X-ray images for Pneumonia detection is to acquire a large dataset of Chest X-ray images with associated labels that identify whether an image contains pneumonia or not. This can include both normal and abnormal chest X-rays.

The dataset used is available on Kaggle and the dataset is organized into 3 folders (train, test, val) and contains subfolders for each image category (Pneumonia/Normal). There are 5,863 X-Ray images (JPEG) and 2 categories (Pneumonia/Normal). Images of chest x-rays with different dimensions that belong to either of the two classes: normal and infected. The class of each image is indicated in the image file’s name itself as opposed to better and easier methods like indexing the images and referencing them in a CSV file.

Preprocessing: Once the data has been acquired, it should be preprocessed so that it is suitable for analysis and can be used to create an algorithm to detect pneumonia from the images given. This includes resizing/reshaping images, normalizing values, and removing any low quality/noisy data points from the dataset.

III.II Features

Feature extraction: After preprocessing the data, feature extraction needs to be done in order to bring out interesting characteristics from each image that could be used for classification purposes later on. This could include extracting features such as shapes, edges, textures, etc., which can be used as inputs into a machine learning algorithm. Selecting the most meaningful features is a crucial step in the process of classification problems especially in handwriting identification So I select this feature which I implement code by it.*(* mean, median ,std ,var , skew, entropy , prewitt_h , prewitt_v , sobelx , sobely *)*

III.III CODE

```
import pandas as pd
import numpy as np
import os
import glob as gb
import cv2
import tensorflow as tf
import csv

from scipy.stats import entropy
from skimage.filters.rank import entropy
from skimage.morphology import disk
from skimage.filters import prewitt_h,prewitt_v

code = {'NORMAL':0 ,'PNEUMONIA':1}
def getcode(n) :
    for x , y in code.items() :
        if n == y :
            return x
#file path
trainpath = 'dataset/train/'
testpath = 'dataset/test/'
valpath = 'dataset/val/'

for folder in  os.listdir(trainpath) :
    files = gb.glob(pathname= str( trainpath +'//' + folder +
'/*.jpeg'))
    print(f'For training data , found {len(files)} in folder
{folder}')

for folder in  os.listdir(testpath) :
    files = gb.glob(pathname= str( testpath +'//' + folder +
'/*.jpeg'))
    print(f'For val data , found {len(files)} in folder {folder}')

for folder in  os.listdir(valpath) :
    files = gb.glob(pathname= str( valpath +'//' + folder +
'/*.jpeg'))
    print(f'For testing data , found {len(files)} in folder
{folder}')
```

```
s = 28

X_train = []
y_train = []
for folder in  os.listdir(trainpath) :
    files = gb.glob(pathname= str( trainpath +'//' + folder +
'/*.jpeg'))
    for file in files:
        image = cv2.imread(file, 0)
        image_array = cv2.resize(image , (s,s))
#         img = image_array/.255
        equalized_img = cv2.equalizeHist(image_array)
        mean_img=equalized_img.mean()
        median_img=np.median(equalized_img)
        std_img=equalized_img.std()
        var_img=equalized_img.var()
        skew_img=(3*(mean_img-median_img))/std_img
        entropy_img=entropy(equalized_img, disk(5)).mean()
        prewitt_h_img=prewitt_h(equalized_img).mean()
        prewitt_v_img=prewitt_v(equalized_img).mean()
        sobelx_img =
cv2.Sobel(equalized_img,cv2.CV_64F,1,0,ksize=5).mean()
        sobely_img =
cv2.Sobel(equalized_img,cv2.CV_64F,0,1,ksize=5).mean()
#
app=[mean_img,median_img,std_img,var_img,skew_img ,
code[folder]]

app=[mean_img,median_img,std_img,var_img,skew_img,e
ntropy_img,prewitt_h_img,prewitt_v_img,sobelx_img,sobel
y_img ,code[folder]]
        X_train.append(list(app))
#         y_train.append(code[folder])

print(f'we have {len(X_train)} items in X_train')


X_test = []
y_test = []
for folder in  os.listdir(testpath) :
    files = gb.glob(pathname= str(testpath + '//' + folder +
'/*.jpeg'))
    for file in files:
        image = cv2.imread(file ,0)
        image_array = cv2.resize(image , (s,s))
        equalized_img = cv2.equalizeHist(image_array)
        mean_img=equalized_img.mean()
        median_img=np.median(equalized_img)
        std_img=equalized_img.std()
        var_img=equalized_img.var()
        skew_img=(3*(mean_img-median_img))/std_img
        entropy_img=entropy(equalized_img, disk(5)).mean()
        prewitt_h_img=prewitt_h(equalized_img).mean()
        prewitt_v_img=prewitt_v(equalized_img).mean()
        sobelx_img =
cv2.Sobel(equalized_img,cv2.CV_64F,1,0,ksize=5).mean()
        sobely_img =
cv2.Sobel(equalized_img,cv2.CV_64F,0,1,ksize=5).mean()
#
app=[mean_img,median_img,std_img,var_img,skew_img
,code[folder] ]
app=[mean_img,median_img,std_img,var_img,skew_img,e
ntropy_img,prewitt_h_img,prewitt_v_img,sobelx_img,sobel
y_img  ,code[folder] ]
        X_test.append(list(app))
#         y_test.append(code[folder])

print(f'we have {len(X_test)} items in X_test')


X_val = []
files = gb.glob(pathname= str(valpath + '//' + folder +
'/*.jpeg'))
for file in files:
    image = cv2.imread(file ,0)
    image_array = cv2.resize(image , (s,s))
    equalized_img = cv2.equalizeHist(image_array)
    mean_img=equalized_img.mean()
    median_img=np.median(equalized_img)
    std_img=equalized_img.std()
    var_img=equalized_img.var()
    skew_img=(3*(mean_img-median_img))/std_img
    entropy_img=entropy(equalized_img, disk(5))
    prewitt_h_img=prewitt_h(equalized_img).mean()
    prewitt_v_img=prewitt_v(equalized_img).mean()
    sobelx_img =
cv2.Sobel(equalized_img,cv2.CV_64F,1,0,ksize=5).mean()
    sobely_img =
cv2.Sobel(equalized_img,cv2.CV_64F,0,1,ksize=5).mean()
#
app=[mean_img,median_img,std_img,var_img,skew_img,
code[folder] ]

app=[mean_img,median_img,std_img,var_img,skew_img,e
ntropy_img,prewitt_h_img,prewitt_v_img,sobelx_img,sobel
y_img , code[folder]]
    X_val.append(list(app))
print(f'we have {len(X_val)} items in X_val')


col_names =
['mean','median','std','var','skew','entropy','prewitt_h','prewitt
_v','sobelx','sobely','label']

f= open('Chest_xray.csv', 'a', newline='')
w= csv.writer(f)
w.writerows(X_train)
w.writerows(X_test)

f.close()
pima = pd.read_csv("chest_xray.csv", header=None,
names=col_names)

pima.head()



feature_cols =
['mean','median','std','var','skew','entropy','prewitt_h','prewitt
_v','sobelx','sobely']
X = pima[feature_cols]
Y = pima.label # Target variable
```

```
from sklearn.model_selection import train_test_split


# set aside 75% of train and test data for evaluation
X_train, X_test, y_train, y_test = train_test_split(X,
Y,test_size=0.25, random_state = 2, shuffle=True)

print(f'we have {len(X_train)} items in X_train')
print(f'we have {len(X_test)} items in X_test')


from sklearn.tree import DecisionTreeClassifier # Import
Decision Tree Classifier
from sklearn import metrics #Import scikit-learn metrics
module for accuracy calculation

clf = DecisionTreeClassifier()
clf = clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))


from pandas import read_csv
from matplotlib import pyplot
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import
LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC

models = []
models.append(('LR', LogisticRegression(solver='liblinear',
multi_class='ovr')))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC(gamma='auto')))

# evaluate each model in turn
results = []
names = []

for name, model in models:
#     kfold = StratifiedKFold(n_splits=10, random_state=1,
shuffle=True)
#     cv_results = cross_val_score(model, X_train, y_train,
cv=kfold, scoring='accuracy')
    model.fit(X_train,y_train)
    cv_results= metrics.accuracy_score(y_test,
model.predict(X_test))
    results.append(cv_results)
    names.append(name)
    print('%s: %f ' % (name, cv_results.mean()))
```

III.VI CLASSIFICATION MODEL

Model Selection: Once feature extraction has been done, it is time to select a suitable model to classify whether each image belongs to either the ‘normal’ or ‘pneumonia’ class. It is important to choose a model that is accurate and fast enough as this will determine how well the system works overall. Examples include Support Vector Machines (SVMs), Random Forests and Decision Trees, among others.

Model Training: After selecting a suitable model for the task at hand, training needs to be done on the extracted features using labeled data so that the model can learn how to differentiate between normal chest X-rays and those with pneumonia present within them accurately and reliably. This can also provide insight into what parameters are important in detecting pneumonia from Chest X-ray images and help fine tune performance of the future system further by tuning hyperparameters such as regularization strength etc..

IV. ANALYSIS AND RESULTS

Evaluation & Testing: Finally after all of these steps have been completed , evaluation & testing must take place in order to determine how well our system works . Here we evaluate its accuracy against unseen test set samples containing both normal and abnormal chest x ray images& calculate precision & recall scores accordingly . The higher these scores , more accurate & reliable our system would be otherwise we would have keep adjusting hyperparameter values or use different models altogether in order optimize our performance

Output:

LR: 0.836986
LDA: 0.848630
KNN: 0.736986
CART: 0.798630
NB: 0.708219
SVM: 0.710959

This means Logistic Regression Model is best in our project

V. CONCLUSION

The model that has been developed for Chest X-Ray images (Pneumonia) processing demonstrates excellent performance with high accuracy scores on both the test and validation datasets. In conclusion, this model is highly suitable for use in detection of Pneumonia from chest x-rays with minimal false positive rates and no significant decrease in overall performance. This model can be further improved in terms of its generalizability and robustness by using more advanced neural network architectures as well as integration of other computer vision techniques such as image enhancement and segmentation.

VI. REFERENCES

“Pneumonia”, Wikipedia, https://en.wikipedia.org/wiki/Pneumonia#Epidemiology

Hammoudi, Karim & Benhabiles, Halim & Melkemi, Mahmoud & Dornaika, Fadi & Arganda-Carreras, Ignacio & Collard, Dominique & Scherpereel, Arnaud. (2020). Deep Learning on Chest X-ray Images to Detect and Evaluate Pneumonia Cases at the Era of COVID-19, https://arxiv.org/pdf/2004.03399.pdf

Rajpurkar, Pranav, et al. "ChexNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning." arXiv preprint arXiv:1711.05225 (2017).

Yao, Liang, et al. "Learning to diagnose from scratch by exploiting dependencies among labels." The British Machine Vision Conference 2016. BMVA Press, 2016.