Detecting Pneumonia from X-Ray Images using Deep Learning

Deep Piyushkumar Patel

Last Year B. Tech, Computer Science and Engineering, Parul University, Parul Institute of Engineering and Technology, Vadodara, Gujarat, India
pateldeep7799[at]gmail.com

Abstract: ***The pneumonia disease is a major health problem for many, especially for developing nations where billions face poverty and rely on polluting forms of energy. Pneumonia is a common disease we have fought against for thousands of years. It's about time we put an end to this. Automating this detection task would greatly improve the efficiency of radiologists. Consequently, there is an urgent requirement for early location and treatment of such ailments. The information which is gathered by data analysis of hospitals is utilizing by applying different blends of calculations and algorithms for the early-stage prediction of pneumonia ailments. Deep Learning is one of the slanting innovations utilized in numerous circles far and wide including the medicinal services application for predicting illnesses. In this research, we compared the accuracy of deep learning algorithms that could be used for predicting the overall risks. The proposed experiment is based on standard deep learning algorithm such as Convolutional Neural Network (CNN), and VGG19 Model in Keras.***

Keywords: Lung diseases, Deep Learning, Image classification

1. Introduction

Pneumonia is an infection in the lungs that can be caused by bacteria, viruses, or fungi. According, to World Health Organization more than 808,000 children under the age of 5 in 2017 has been died which accounts 15% of all deaths of children under 5 years[1]. Chest X-rays are currently the best method for diagnosing pneumonia. However, there is still a lack of access with almost two-thirds of the world's population lacking access to radiology diagnostics [2]. It is even more difficult to make an X-ray of the chest than other methods such as CT or MRI. This leads to negative results. Thus, a feasible and accurate prediction of pneumonia diseases is very important. Laborite, from all around the world collects data on various health-related issues. This data can be exploited using various deep learning techniques to gain useful insights. Thus, these algorithms have become very useful, in recent times, to predict the presence or absence of pneumonia disease in lungs. To begin with, the work we are using different types of techniques and algorithms. In this paper, the image classification techniques and algorithms are used to increase the accuracy rate. In deep learning, classification algorithms In Machine learning, classification algorithms are supervised learning approach in which the computer learns from the input data and learn from it. This data collection can basically be bi-class (such as recognizing that the person is an animal or a human or whether the e-mail is spam or not spam) or it may be in many categories. Here are the names of classification algorithms which we are going to implement and compare the accuracy in this research: *1) Convolutional Neural Network (CNN.)*

2. Background of study

The lung is most important organ of the human body. The lungs are the center of the respiratory (breathing) system. Every cell of the body needs oxygen to stay alive and healthy. The air we breathe contains oxygen and other gases. When our air are in the lungs, oxygen is carried into your bloodstream and passed on to your body. Your body also needs to remove carbon dioxide. This gas is a waste product produced by cells during their normal, daily activities. The dataset which is used for image classification is available on the Kaggle website. The classification goal of this study is to predict whether the patient has a risk of pneumonia disease or not. The lungs dataset consist of more than 5000 images for training set and 600 images for testing set. The data analysis is carried out in Python programming by using Google colab Lab which is a more flexible and powerful data science application software.

3. Methodology

3.1 Workflow of building Deep Learning Problem

Figure 1 indicates the steps forward in order to build model.

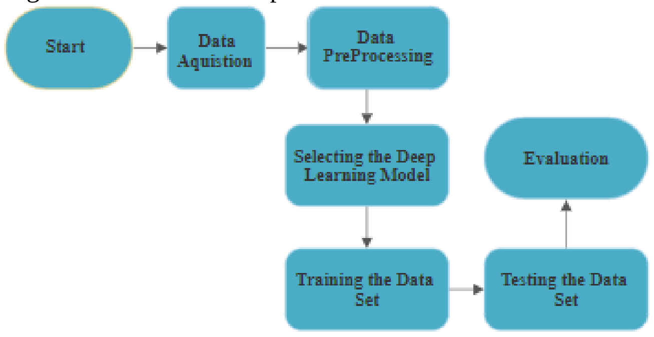


Figure 1: Workflow of Building Deep Learning Model.

3.2 Data Acquisition

The data has been collected from the Kaggle website.

3.3 Data PreProcessing

To build more accurate deep learning model, data preprocessing is required. Data preprocessing is the process of cleaning data and keeping the data which is needed.

3.4 Select Deep Learning Model

After preprocessing of data we will be using keras machine learning library based on Custom Deep Convolutional Neural Network for training and testing dataset.

3.5 Analyzing of X-Rays

I started by analyzing a couple of x-rays through a CNN. Compared to ancient neural networks, Convolutional Neural Networks ar loads additional economical at process image knowledge.

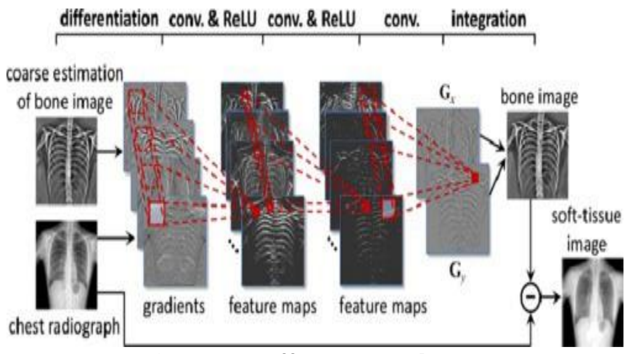


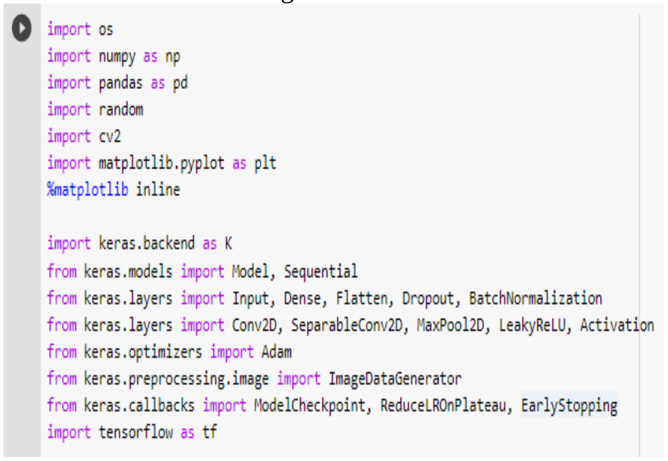
Figure 2: Different convolutions

In convolutions, rather than going pixel by pixel, we have a tendency to use “filters” to research parts of a picture. this can be such a robust model, that it will scale back the amount of operations from an easy network within the many millions to but 10 million. exploitation CNN's I classified all of my information[3].

4. Implementation of Deep Learning Model

4.1 Importing the Libraries

Here I have loaded the required libraries into Google colab to build a machine learning model.



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

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
```

Figure 3: Libraries

4.2 Reading the DataSet

I have loaded some normal and pneumonia images to just a look at how much different they look.

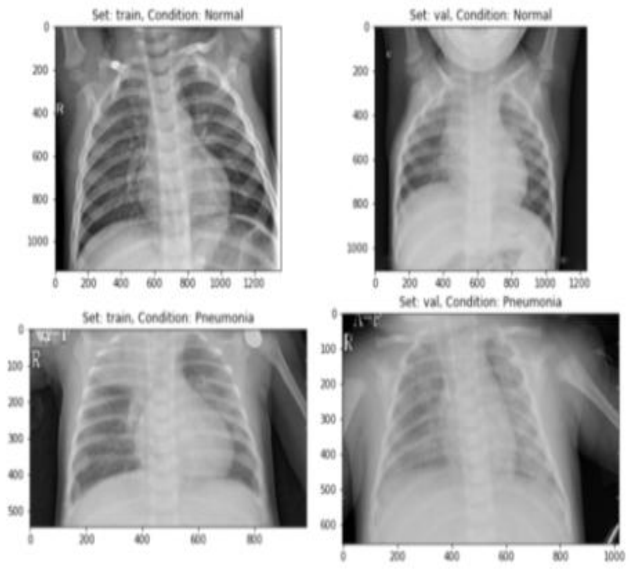

Set: train, Condition: Normal
Set: val, Condition: Normal
Set: train, Condition: Pneumonia
Set: val, Condition: Pneumonia

Figure 4: Pneumonia or Not

4.3 Organizing our Data

Detecting respiratory disorder in Associate in Nursing x-ray scan could be a easy binary classification problem: either we have a tendency to observe respiratory disorder, or we have a tendency to don’t. I provided this illustration with numbers! zero meant traditional and one meant respiratory disorder. Using Python, I created a knowledge frame for every image, tagged with zero or one betting on its folder, and shuffled all together. And I also split data sets into three main parts i.e train, test and validation.

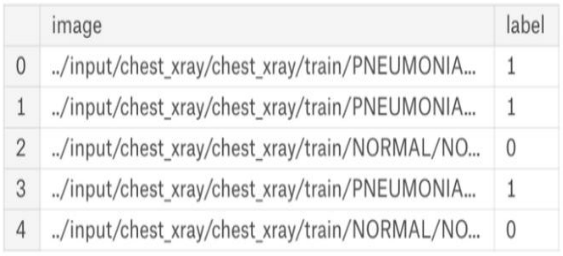


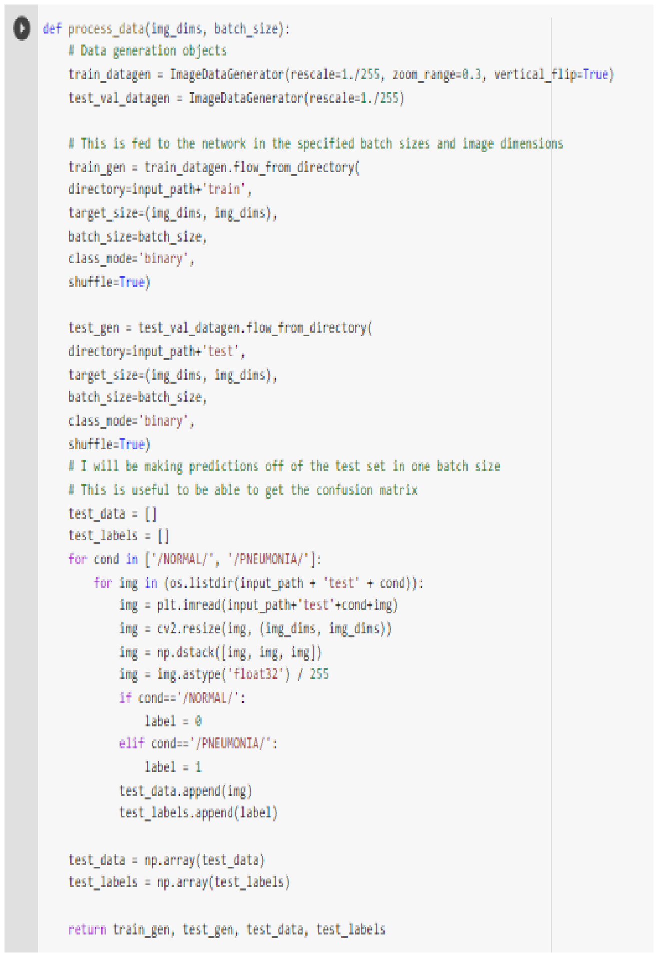
	image	label
0	../input/chest_xray/chest_xray/train/PNEUMONIA...	1
1	../input/chest_xray/chest_xray/train/PNEUMONIA...	1
2	../input/chest_xray/chest_xray/train/NORMAL/NO...	0
3	../input/chest_xray/chest_xray/train/PNEUMONIA...	1
4	../input/chest_xray/chest_xray/train/NORMAL/NO...	0

Figure 5: Labeling Data

4.4 Implementing CNN

The Convolutional Neural Network (ConvNet / CNN) is a Deep Learning algorithm that can capture imagery, give value (readable and discriminatory instruments) to the various elements / elements in the image and be able to distinguish one from the other.

In my tests, I usually set a batch_size = 64. Usually a number between 32 and 128 should work well. You should usually increase / decrease the batch size depending on the computational resources and model performance.



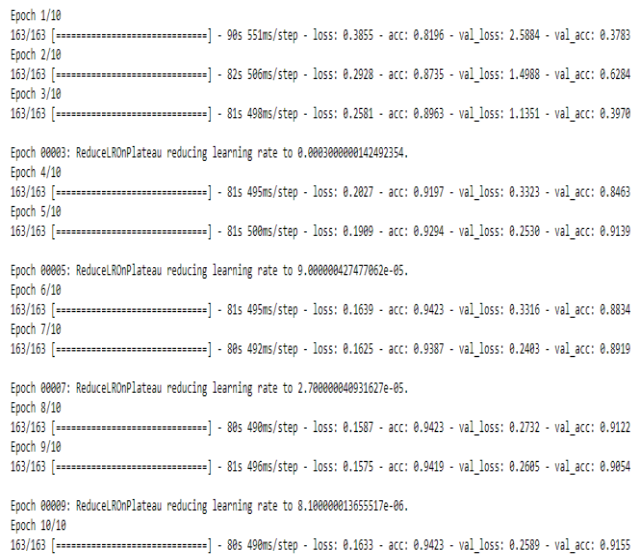
```
def process_data(img_dims, batch_size):
    # Data generation objects
    train_datagen = ImageDataGenerator(rescale=1./255, zoom_range=0.3, vertical_flip=True)
    test_val_datagen = ImageDataGenerator(rescale=1./255)

    # 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+'test',
    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
```

The next step was to build a model. This can be explained in the next 5 steps.

1) I used five convolutional blocks that contain a layer of convolutional, max-pooling and batch-normalization.
2) On top of that I used a flat layer and followed it with four layers that were fully connected.
3) And in the middle I used the dropouts to reduce the excess balance (over-fitting).
4) The function of activation was Relu throughout the last layer where it was Sigmoid as this is a problem of binary separation.
5) I used Adam as an optimizer and cross-entropy as a loss.

5. Training a Model

Next I trained the model with 10 epochs with 32 batch size. Please note that usually the higher batch size gives better results but is paid for by a higher calculation load. Another study also states that there is a good batch size of positive results that can be obtained by investing sometime in the hyper-parameter setting [4]. Figure shows a training accuracy of model.

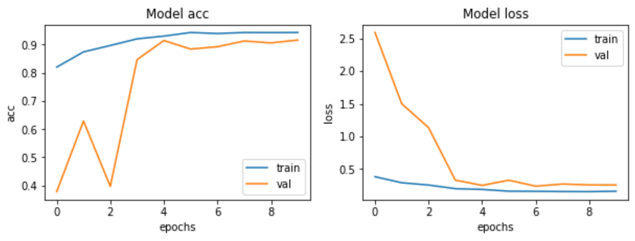


```
Epoch 1/10
163/163 [==============================] - 90s 551ms/step - loss: 0.3855 - acc: 0.8196 - val_loss: 2.5884 - val_acc: 0.3783
Epoch 2/10
163/163 [==============================] - 82s 506ms/step - loss: 0.2928 - acc: 0.8735 - val_loss: 1.4988 - val_acc: 0.6284
Epoch 3/10
163/163 [==============================] - 81s 498ms/step - loss: 0.2581 - acc: 0.8963 - val_loss: 1.1351 - val_acc: 0.3970

Epoch 00003: ReduceLROnPlateau reducing learning rate to 0.0003000000142492354.
Epoch 4/10
163/163 [==============================] - 81s 495ms/step - loss: 0.2027 - acc: 0.9197 - val_loss: 0.3323 - val_acc: 0.8463
Epoch 5/10
163/163 [==============================] - 81s 500ms/step - loss: 0.1909 - acc: 0.9294 - val_loss: 0.2530 - val_acc: 0.9139

Epoch 00005: ReduceLROnPlateau reducing learning rate to 9.000000427477062e-05.
Epoch 6/10
163/163 [==============================] - 81s 495ms/step - loss: 0.1639 - acc: 0.9423 - val_loss: 0.3316 - val_acc: 0.8834
Epoch 7/10
163/163 [==============================] - 80s 492ms/step - loss: 0.1625 - acc: 0.9387 - val_loss: 0.2403 - val_acc: 0.8919

Epoch 00007: ReduceLROnPlateau reducing learning rate to 2.700000040931627e-05.
Epoch 8/10
163/163 [==============================] - 80s 490ms/step - loss: 0.1587 - acc: 0.9423 - val_loss: 0.2732 - val_acc: 0.9122
Epoch 9/10
163/163 [==============================] - 81s 496ms/step - loss: 0.1575 - acc: 0.9419 - val_loss: 0.2605 - val_acc: 0.9054

Epoch 00009: ReduceLROnPlateau reducing learning rate to 8.100000013655517e-06.
Epoch 10/10
163/163 [==============================] - 80s 490ms/step - loss: 0.1633 - acc: 0.9423 - val_loss: 0.2589 - val_acc: 0.9155
```

Figure 6: Training Efficiency

5.2 Visualizing the loss and accuracy plots

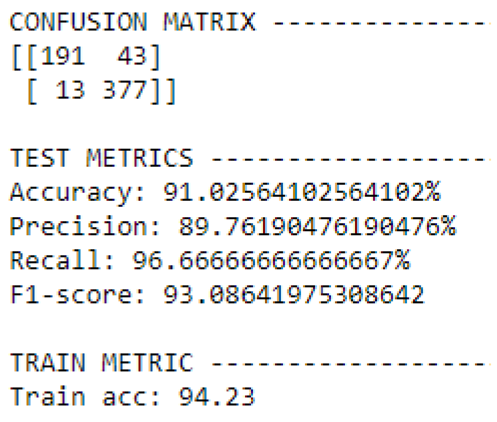
Figure shows good model because the model is converging which can be observed from the decreases in loss and validation loss with epoch. Moreover, we have just used 10 epochs and accuracy of the model is 90%.



Accuracy vs Epoch | Loss vs Epoch

5.3 Accuracy using Confusion Matrix

Basically, confusion matrix is a matrix which is used to measure Recall, Precision, Accuracy, and AUC-ROC curve it is also known as an error matrix. For this project, I have used confusion matrix to check Test and Train matrix accuracy which is nearly 92% for test dataset and 95% for train dataset.



```
CONFUSION MATRIX --------------
[[191  43]
 [ 13 377]]

TEST METRICS ------------------
Accuracy: 91.02564102564102%
Precision: 89.76190476190476%
Recall: 96.66666666666667%
F1-score: 93.08641975308642

TRAIN METRIC ------------------
Train acc: 94.23
```

6. Conclusion

The main purpose of this research is to compare the accuracy of CNN algorithm on lung data sets and build a model which is useful for future where we will not need a x-ray machine to check that we have pneumonia or not we can do that by just using our smartphones. Although, by just using 10 epoch we are getting nearly 95% accuracy. Moreover, this project is in complete completion but it is amazing to see the success of an in-depth learning model.

7. Future Scope

Nowadays, most of the people have smart devices such as smartphones and laptop. Through deep learning models we can develop such app or website which can detect and classify X-Rays image consisting of lung cancer and pneumonia. It will be a huge achievement that we will not need a laboratory to check our x-rays and we can do it by our own. We have seen in recent time we are need to this type of technology and our next approach should be tackle this type of problem.

References

[1] S. Pawshe, N. Prasad, V. Phondekar, Prof. R. Shintre "Detecting Pneumonia from Chest X-Ray Images using Committee Machine", International Research Journal of Engineering and Technology, pp. 2395-0072, volume 07 Issue 03, March 2020.

[2] WHO"https://www.unicef.org/health/childhood diseases" a survey.

[3] S. Seema, Suhas Goutham, Smaranita Vasudev, Rakshith R Putane, "Surveillance video analysis using deep learning techniques for traffic and crowd management", International Journal of latest trends in Engineering and Technology, Volume 14 Issue 3, September 2019.

[4] J. Hernandez-Ortega, J. Galbally, J. Fierrez, R. Haraksim, "FaceQnet: Quality Assessment for Face Recognition based on Deep Learning", 2019 International Conference on Biometrics (ICB).

Author Profile

Deep Piyushkumar Patel is a last year Computer Science student pursing Bachelor of Technology in Parul Institute of Engineering and Technology, Parul University. He is a Kaggle expert. His research Interset is in Machine Learning, Deep Learning, Artificial Intelligence, Virtual reality, Computer Vision, Augmented Reality, Gestural Interaction, Automation, Natural Language Processing.