Mini Project

Semester 7

17BIT048 • Rutvik Patel
17BIT051 • Parth Shah

Overview

Title

Machine Learning based Model for Pneumonia Detection
(Using Chest X-Ray Radiographs)

Tools used

Python3 IDE (Google Colab),
Tensorflow, OpenCV, Shell Commands

Expected delivery

December 2020

Abstract

Use Image Processing and Convolutional Neural Networks to predict whether the person is suffering from Pneumonia, using his/her chest Radiograph.

Motivation

Some of the most popular methods for Pneumonia detection, currently, include Sputum tests, Blood tests and CT Scan. Methods like Blood tests take longer time to give out results. Whereas, methods like Sputum test is comparatively inaccurate and lastly, methods like CT Scan are very expensive for a common man. These factors inspired us to consider this project where a patient can get a rough idea about his health, thus decreasing costs for common man and number of tests required.

Objective

To Classify whether the given(input) image belongs to “Pneumonia” class or “Normal” class.

Literature Survey

In recent time, exploration of Machine learning (ML) algorithms in detecting thoracic diseases has gained attention in research area of medical image classification. There have been several approaches which make use of U-Net, SegNet, Cardiac Net, Alex Net and Google Net. Due to computational constraints, we decided to create our own Deep CNN learning model.

Brief Methodology

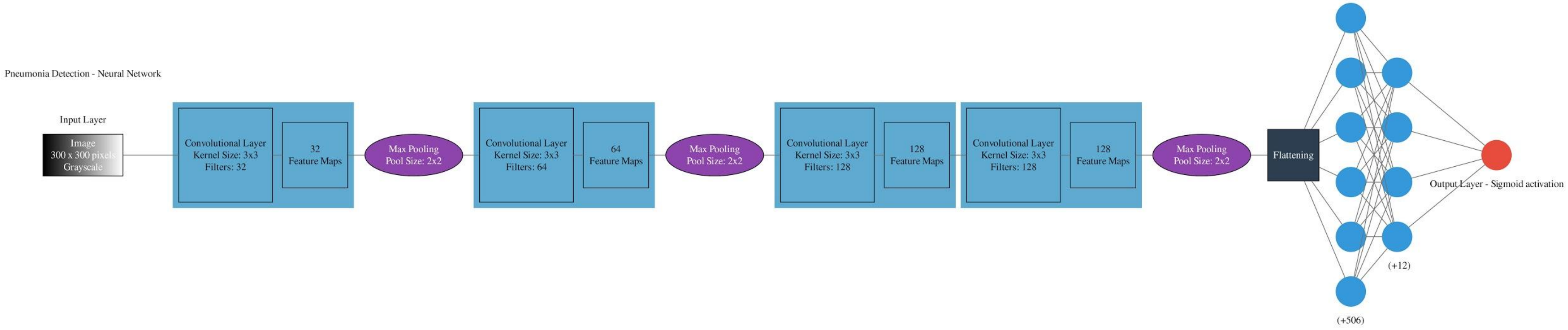
Use various Image processing techniques to pre-process the images, split into train, validation and test set; train a CNN based model for the classification task and then predict whether the Radiograph presented belongs to a person suffering from Pneumonia or not.

Approach

- Downloading the dataset
- Importing the libraries
- Pre-processing the dataset:
  - Shell commands for removing unnecessary files
  - Defining a function to resize the images
  - Defining a function to get the matrix of the images
  - Changing the color space of images to grayscale
- Dividing the dataset into train, validation and test set
- Building a CNN model
- Training the model on the train dataset
- Testing the model on the validation dataset
- Examining the accuracy and loss values graphically
- Tweaking the hyperparameters
- Testing the model on the test dataset
- Classifying the images into 2 classes:
  - Normal
  - Pneumonia

Implementation and Design

The proposed pneumonia detection system using the ‘Densely Connected Deep Convolutional Neural Network’ is described in the Figure. The architecture of the proposed model has been divided into two main stages - data preprocessing and augmentation, and classification model.



Pneumonia Detection - Neural Network
Input Layer
Image
300 x 300 pixels
Grayscale
Convolutional Layer
Kernel Size: 3x3
Filters: 32
32
Feature Maps
Max Pooling
Pool Size: 2x2
Convolutional Layer
Kernel Size: 3x3
Filters: 64
64
Feature Maps
Max Pooling
Pool Size: 2x2
Convolutional Layer
Kernel Size: 3x3
Filters: 128
128
Feature Maps
Convolutional Layer
Kernel Size: 3x3
Filters: 128
128
Feature Maps
Max Pooling
Pool Size: 2x2
Flattening
(+506)
(+12)
Output Layer - Sigmoid activation

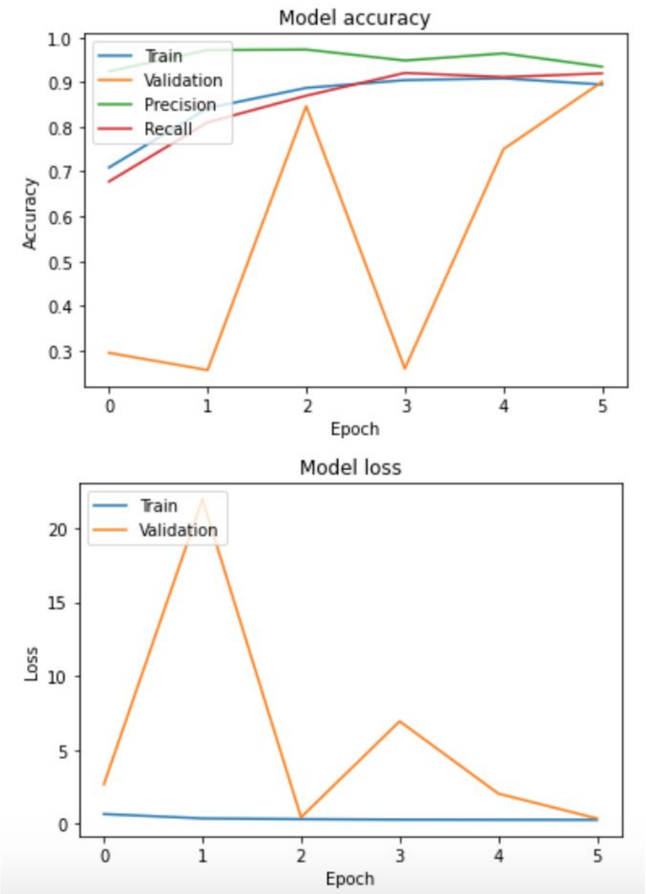
Experimentation

```
Epoch 1/6
32/32 [==============================] - ETA: 0s - loss: 0.2915 - accuracy: 0.8799 - precision: 0.9082 - recall: 0.9310
Epoch 00001: saving model to Checkpoints/saved-model-01-acc=0.88-recall=0.93-v_acc=0.89-v_recall=0.95.h5
32/32 [==============================] - 282s 9s/step - loss: 0.2915 - accuracy: 0.8799 - precision: 0.9082 - recall: 0.9310 - val_loss: 0.2460 - val_accuracy: 0.8910 - val_precision: 0.9057 - val_recall: 0.9526
Epoch 2/6
32/32 [==============================] - ETA: 0s - loss: 0.2859 - accuracy: 0.8867 - precision: 0.9211 - recall: 0.9259
Epoch 00002: saving model to Checkpoints/saved-model-02-acc=0.89-recall=0.93-v_acc=0.91-v_recall=0.95.h5
32/32 [==============================] - 280s 9s/step - loss: 0.2859 - accuracy: 0.8867 - precision: 0.9211 - recall: 0.9259 - val_loss: 0.2462 - val_accuracy: 0.9071 - val_precision: 0.9283 - val_recall: 0.9483
Epoch 3/6
32/32 [==============================] - ETA: 0s - loss: 0.2355 - accuracy: 0.9062 - precision: 0.9284 - recall: 0.9499
Epoch 00003: saving model to Checkpoints/saved-model-03-acc=0.91-recall=0.95-v_acc=0.90-v_recall=0.97.h5
32/32 [==============================] - 279s 9s/step - loss: 0.2355 - accuracy: 0.9062 - precision: 0.9284 - recall: 0.9499 - val_loss: 0.2450 - val_accuracy: 0.9006 - val_precision: 0.9036 - val_recall: 0.9698
Epoch 4/6
32/32 [==============================] - ETA: 0s - loss: 0.2283 - accuracy: 0.9102 - precision: 0.9357 - recall: 0.9455
Epoch 00004: saving model to Checkpoints/saved-model-04-acc=0.91-recall=0.95-v_acc=0.90-v_recall=0.94.h5
32/32 [==============================] - 280s 9s/step - loss: 0.2283 - accuracy: 0.9102 - precision: 0.9357 - recall: 0.9455 - val_loss: 0.2395 - val_accuracy: 0.9006 - val_precision: 0.9241 - val_recall: 0.9440
Epoch 5/6
32/32 [==============================] - ETA: 0s - loss: 0.2059 - accuracy: 0.9062 - precision: 0.9370 - recall: 0.9319
Epoch 00005: saving model to Checkpoints/saved-model-05-acc=0.91-recall=0.93-v_acc=0.92-v_recall=0.91.h5
32/32 [==============================] - 282s 9s/step - loss: 0.2059 - accuracy: 0.9062 - precision: 0.9370 - recall: 0.9319 - val_loss: 0.2126 - val_accuracy: 0.9167 - val_precision: 0.9769 - val_recall: 0.9095
Epoch 6/6
32/32 [==============================] - ETA: 0s - loss: 0.2400 - accuracy: 0.9102 - precision: 0.9399 - recall: 0.9423
Epoch 00006: saving model to Checkpoints/saved-model-06-acc=0.91-recall=0.94-v_acc=0.87-v_recall=0.96.h5
32/32 [==============================] - 277s 9s/step - loss: 0.2400 - accuracy: 0.9102 - precision: 0.9399 - recall: 0.9423 - val_loss: 0.2700 - val_accuracy: 0.8718 - val_precision: 0.8780 - val_recall: 0.9612
```

Results

To evaluate and validate the effectiveness of the proposed approach, we conducted the experiments many times, with several approaches. Parameter and hyperparameters were heavily turned to increase the performance of the model. Different results were obtained, but we decided to consider the result with optimum Accuracy and Recall values.

For our model, training loss = 0.2283, training accuracy = 0.9102, training recall=0.9455, validation loss=0.2395, validation accuracy=0.9006 and validation recall=0.944. For test set, loss=0.4396, accuracy=0.8429 and recall=0.9436.


Model accuracy
Train
Validation
Precision
Recall
Accuracy
Epoch
Model loss
Train
Validation
Loss
Epoch

Difficulties Faced

Not accustomed to the techniques of Deep Learning based Image Classification

Removing unnecessary files and then employing various image processing techniques.

Deciding on the values of hyper-parameters

Approach

DataSet

Pre-processing

Algorithm

Hyper-parameter

Deciding on which dataset to use for this problem

With little experience for the task at hand, it was quite challenging to design an algorithm

Discussion

For this task, we explored several different types of neural network architectures, before settling with a comparatively smaller architecture which was designed by us. The architectures we explored and tried using include VGG16, VGG19 and Xception.

We were unable to use these architectures due to computational constraints and very long training time.

Thank You