Final Project Report
Multimedia Computing

COVID-19 testing using Sobel filtering and SVM classifier

Submitted by:
U1610146 Mirzashomol Karshiev
Inha University in Tashkent
2020

Abstract:

In this report, you will understand the steps of creating intelligent service using a machine learning algorithm. To be specific this report tells how to develop software, which tests patients for the COVID-19 outbreak. How tests can be achieved? First, a person should have an X-ray image of his/her lung that is fed into the SVC (Support Vector Classifier) machine learning algorithm that will output the result based on a trained model. To train a classification model we need to have a dataset of patients. A dataset has two types of images: normal and pneumonia. Before the training model with data, we have to preprocess the image like denoising the image using GaussianBlur, edge detection using Sobel filter, and feature extraction with HOG(Histogram of Oriented Gradients). After processing the images, the collection of images is fed into SVC.

Introduction:

The goal of this report is to show with knowledge of Image Processing, Computer Vision, and Machine Learning you can quickly build software that is capable of providing intelligent services like COVID-19 testing for end-users. The reason why it is quick because we live in the era of ICT convergence and every IT- technologies entered into every aspect of our life starting from your work, learning, social, spiritual, and health. Already everything is digitized that leads to an increased potential of computer technologies. For that reason, this report will investigate the solutions to COVID-19 using computer software technologies. Today we all facing the same problem which is a coronavirus name COVID-19. It emerged in China Wuhan province and it spread all over the world within a month. Humans even did not realize how this coronavirus can be fatal for all people. Many people have died due to this virus, and humanity has never faced such a type of virus. All countries are fighting against it by applying the rules of quarantine. The easiest way to deal with this coronavirus issue is to hold many test cases for COVID-19 fast. That can be achieved only with the help of IT technology. That is the reason for choosing this theme that somehow shows loyalty for the current situation. The investigation will behold on X-ray images of patients and test them for COVID-19 by using open-source libraries like OpenCV (Open Computer Vision) and Machine Learning algorithms like SVM (Support Vector Machine) supervised machine learning algorithm based on regressions. The patient's X-ray picture of their lung will be used for the test and feed into a machine learning model that tells whether you have a virus or not. This report will tell you all steps starting from a dataset ending with a machine learning algorithm.

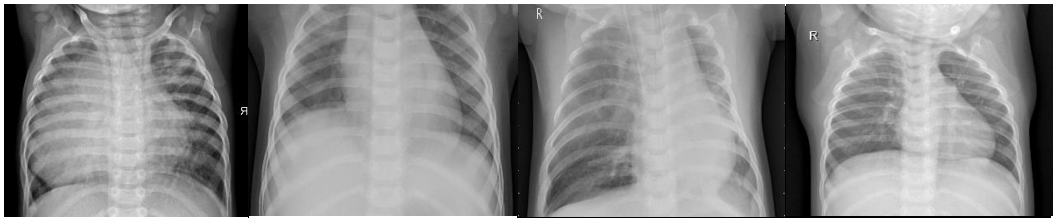
Method:

The process of developing software that tests for Covid-19 consist of 3 main steps:

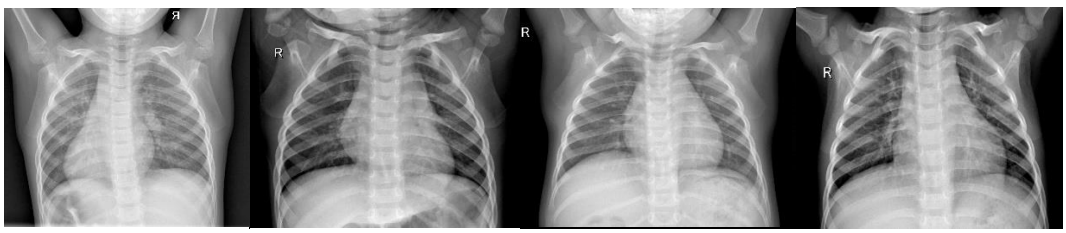
1. Loading a dataset.
2. Preparing a dataset for machine learning (Image Processing)
3. Training and testing machine-learning model.

1. Loading a dataset:

First of all, for our software to predict the test X-ray image of a person, we need to have a big dataset that consists of thousands of images or data that will be useful for a machine learning to be more precise on the output. For that reason we have to find the right dataset for a model. Currently, there are not a lot of datasets available on the internet for the COVID-19 virus. But the dataset for COVID-19 is available on the kaggle.com site. This dataset contains around 5 thousand images of people who have lung in normal and pneumonia conditions.



People with virus COVID-19.



People who don’t have virus COVID-19.

We load this datasets using python code. We have two folders that have positive and negative cases of COVID-19 virus.

```
DATADIR="C:/Users/HP/Desktop/MC Project/images"
categories=["negative","positive"]
```

For loading these negative and positive images into n-dimensional numpy array we have to use os library.

```
    #<--------------Loading Images for Training----------------->
    def __populateTrainMatrix__(self):

        training_data=[]

        for category in categories:
            path= os.path.join(DATADIR,category)
            for img in os.listdir(path):
                #reading image from folder
                img_array=cv2.imread(os.path.join(path,img),cv2.IMREAD_GRAYSCALE)
                #denoising image
                denoised_img=self.__imageProcessing__(img_array)
                #edge detection using Sobel
                img_edge=self.__edgeDetection__(denoised_img)
                #resizing to 1xN vector
                resized_img=self.__resizeTo1xN__(img_edge)
                #collecting all vectorized images for training
                training_data.append(resized_img)

        return training_data
```

2. Preparing a dataset for machine learning:

The above examples of images need to be fed to machine learning algorithm but before straightly feeding these images we have to convert them into machine understandable format which is number. Moreover, these images can have noises like outliers, which really affects to the correctness of a model. For that reason, we have to read the image convert it into gray scale format then denoise image using Gaussian Blur algorithm, which by the help of kernel averages the pixels of an image. After which outliers are eliminated.

```
        #image denoising using GausianBlur
        blur_image=cv2.GaussianBlur(image,(5,5),0)
```

After applying noise reduction, we use edge detection algorithm that Sobel filtering.

```
    #<--------------Sobel Filtering----------------->
    def __edgeDetection__(self,img):
        #Calcution of Sobelx
        sobelx = cv2.Sobel(img,cv2.CV_32F,1,0,ksize=5)
        #absolute of sobelx gradient
        abs_sobelx = cv2.convertScaleAbs(sobelx)

        #Calculation of Sobely
        sobely = cv2.Sobel(img,cv2.CV_32F,0,1,ksize=5)

        #absolute of sobely gradient
        abs_sobely = cv2.convertScaleAbs(sobely)

        #joining together sobelx and sobely
        cv2.addWeighted(abs_sobelx,0.5,abs_sobely,0.5,0,img)

        return img
```

After applying it we have to vectorize every image in format 1xN. After that all images have to be collected into one matrix and should be labeled accordingly. As soon as our model is a classification model we have to label with 1 or 0. In summary, train matrix has features and labels has values.

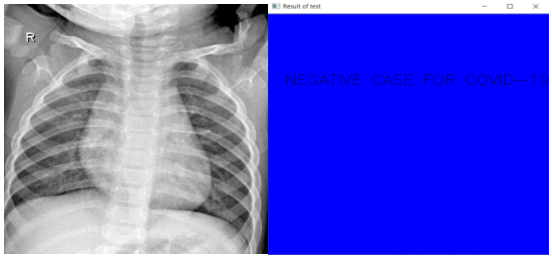
3. Training and testing machine-learning model:

As I mentioned above the model we use is SVM. The SVM (Support Vector Machine) is supervise machine learning algorithm. We will use it for classification of images. We will use kernel type ‘linear’. There are 5 or 6 types of regressions for SVM.

```
    #<--------------SVM Model Initialization----------------->
    def __modelSVM__(self):
        #create SVM machine learning classifier
        svm = cv2.ml.SVM_create()
        #SVC
        svm.setType(cv2.ml.SVM_C_SVC)
        #kernel type Linear means linear regression
        svm.setKernel(cv2.ml.SVM_LINEAR)
        #criteria
        svm.setTermCriteria((cv2.TERM_CRITERIA_MAX_ITER, 100, 1e-6))

        return svm

    #<--------------Training SVC Model----------------->
    def __modelFitData__(self,x_train,y):
        #calling model
        model=self.__modelSVM__()
        #fitting training set
        model.train(x_train, cv2.ml.ROW_SAMPLE, y)

        return model
```

Experiment Results:

The experiment is done on sample test pictures to see whether an image belongs to normal or pneumonia. For example, let's see we have 2 images one for normal and one for pneumonia, we will test 2 images for the correctness of the model. For that reason, if the result window's background color is blue that means it is a negative case for COVID-19. If the result window's background color is red that means the test is positive for coronavirus.



Normal Result

Pneumonia Result

We have to understand if you train the model with less data then the parameters are not optimized well and the model can predict wrongly sometimes.

References:

https://jakevdp.github.io/PythonDataScienceHandbook/05.07-support-vector-machines.html
https://www.datacamp.com/community/tutorials/svm-classification-scikit-learn-python
https://docs.opencv.org/3.4/d1/d73/tutorial_introduction_to_svm.html