PRML MINOR PROJECT

IIT JODHPUR

Mask and No Mask Detection

Made by:
Mayank Singh Rajput(B19CSE054)
Mohit Ahirwar(B19CSE055)

I. Introduction

In this project, we are expected to design a machine learning mode which can classify whether a person is wearing a mask or not. The dataset that we have used is already provided. We have to first apply at least three classifiers to get the performance results and then compare them.

II. Observing the Dataset

First, we analysed the two datasets which were given. On the basis of that, we sorted that dataset into two classes - Masked and Unmasked. This dataset now contains pictures of people either wearing a mask or not. In all images, the face of the person is in focus and clearly visible. Then finally to run our classifiers, we read the dataset using OpenCv.

III. Image Processing

We processed the images to apply classifiers on them like :

A. Labelled Images into ‘1’ and ‘0’

We first labelled all the images as either ‘0’ for Masked and ‘1’ for unmasked.

B. Image Grayscale conversion

We converted all images into grayscale. Grayscale images made classification of images into masked and unmasked easier for our machine learning algorithm.

Code : gray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

C. Image Resizing

To remove any discrepancies for the algorithm to work, we converted all the images to a common pixel size, i.e 100x100.

Code : resized=cv2.resize(gray,(img_size,img_size))#img_size = 100

D. Appending the final images

After image processing, we appended all the images into an already created list (dataset) named ‘target’.

Code : target.append(label_dict[category])

IV. Applied Models

A). CNN

a) Building CNN Model

i) The first CNN layer

1) Code :

```
model.add(Conv2D(200,(3,3),input_shape = X.shape[1:]))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
```

ii) The second CNN layer

1) Code :

```
model.add(Conv2D(100,(3,3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
```

iii) Flatten Layer to stack the output convolution from convolution

1) Code :

```
model.add(Flatten())
```

iv) Performing dropout to avoid overfitting

1) Code :

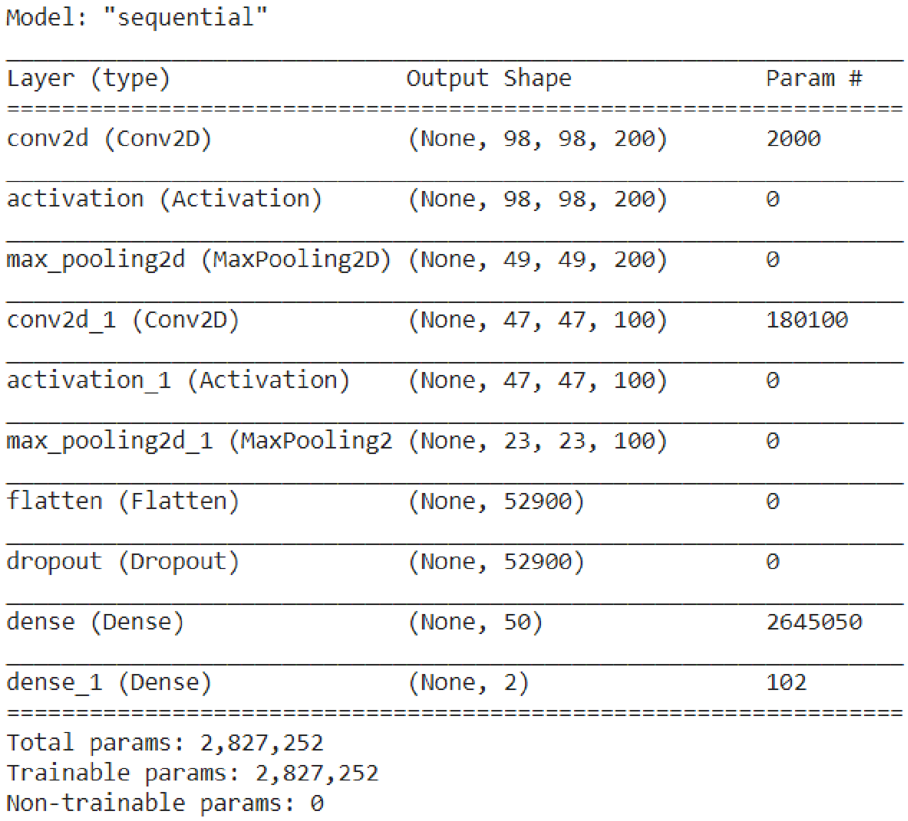
```
model.add(Dropout(0.5))
```

v) Compiling the model

1) Code :

```
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accura
cy'])
```

b) Model Summary



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 98, 98, 200)       2000
_________________________________________________________________
activation (Activation)      (None, 98, 98, 200)       0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 49, 49, 200)       0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 47, 47, 100)       180100
_________________________________________________________________
activation_1 (Activation)    (None, 47, 47, 100)       0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 23, 23, 100)       0
_________________________________________________________________
flatten (Flatten)            (None, 52900)             0
_________________________________________________________________
dropout (Dropout)            (None, 52900)             0
_________________________________________________________________
dense (Dense)                (None, 50)                2645050
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 102
=================================================================
Total params: 2,827,252
Trainable params: 2,827,252
Non-trainable params: 0
```

i)

c) Splitting the dataset : **training 50% and testing 50%**

i) Code :

```
train_X,test_X,train_y,test_y = train_test_split(X,y_cat,test_size=0.5)
```

d) Model Fitting

i) Code :

```
CNN = model.fit(train_X,train_y,epochs=20,validation_split=0.2)
```

e) Accuracy Calculations

i) Code :

```
accuracy = model.evaluate(x=test_X,y=test_y,batch_size=32)
print("Accuracy - CNN : ",accuracy[1]*100,"%")
```

ii) Accuracy:

1) Accuracy - CNN : 98.84231686592102 %

f) Checking Performance of our model

i) Code :

```
print(classification_report(test_y, prediction))
```

B). SVM - Linear and Gausian

a) Creating the Classifier

i) Code :

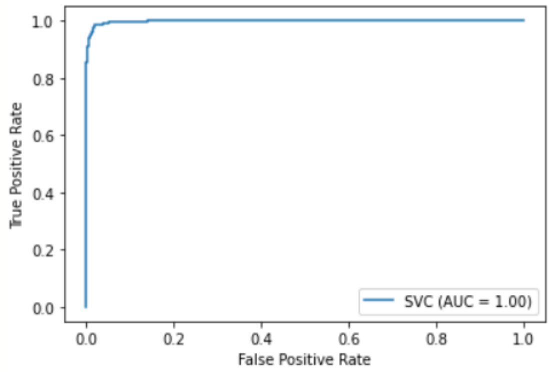
```
clf = svm.SVC(kernel='linear')
clf = svm.SVC(kernel='rbf')
```

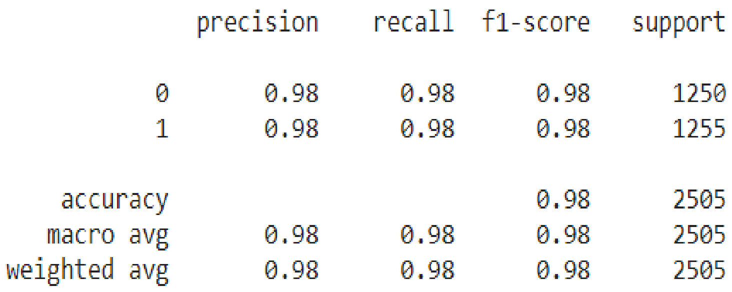
ii) Accuracy :

Accuracy(linear)- SVM (Linear): 93.0938123752495 %

Accuracy(gaussian)- SVM (gaussian): 98.16367265469061 %

iii) Plotting ROC Curve





```
              precision    recall  f1-score   support

           0       0.98      0.98      0.98      1250
           1       0.98      0.98      0.98      1255

    accuracy                           0.98      2505
   macro avg       0.98      0.98      0.98      2505
weighted avg       0.98      0.98      0.98      2505
```

C). KNN

a) Creating KNN Classifier

```
knn = KNeighborsClassifier(n_neighbors=2)
```

b) Train model using training sets

```
knn.fit(train_X,train_y)
```

c) Predicting response from dataset

```
y_pred_knn = knn.predict(test_X)
```

d) Accuracy - KNN : 93.01397205588823 %

D). Logistic Regression

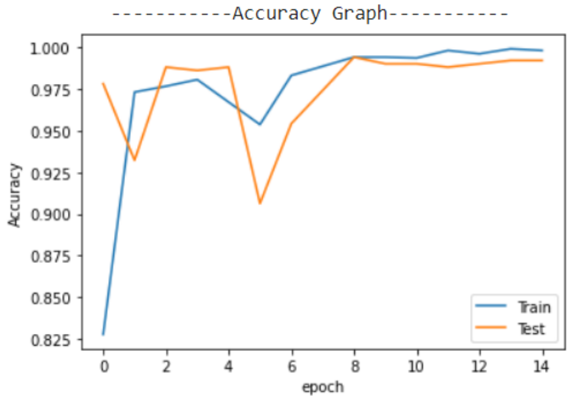
a) Implementation :

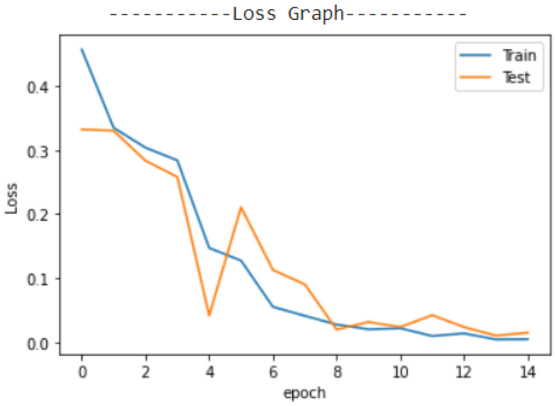
```
clf_lr = LogisticRegression(max_iter=3000).fit(train_X,train_y)
y_pred_lr = clf_lr.predict(test_X)
```

b) Accuracy - Logistic Regression : 94.41117764471058 %

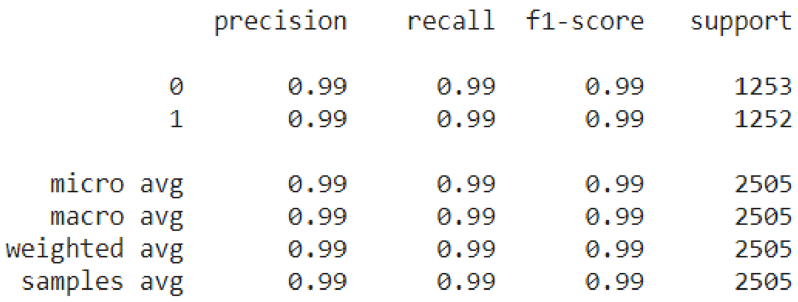
VII. RESULTS AND COMPARISON

- Accuracy and Loss Graphs for Convolution Neural Network (CNN)





- Confusion Matrix for CNN

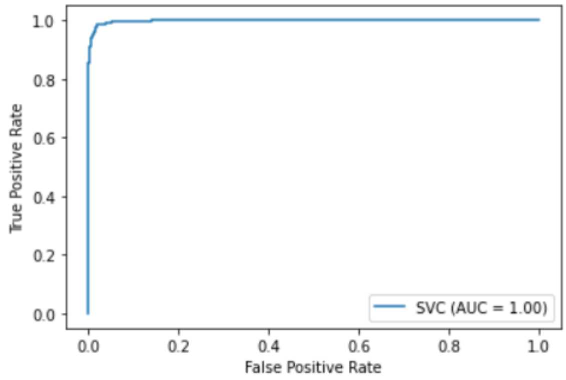


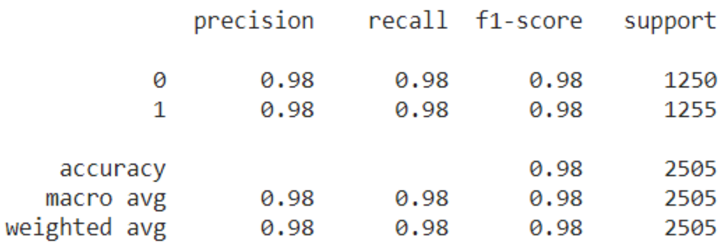
```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      1253
           1       0.99      0.99      0.99      1252

   micro avg       0.99      0.99      0.99      2505
   macro avg       0.99      0.99      0.99      2505
weighted avg       0.99      0.99      0.99      2505
 samples avg       0.99      0.99      0.99      2505
```

- ROC and Confusion Matrix for SVM (Gausian)



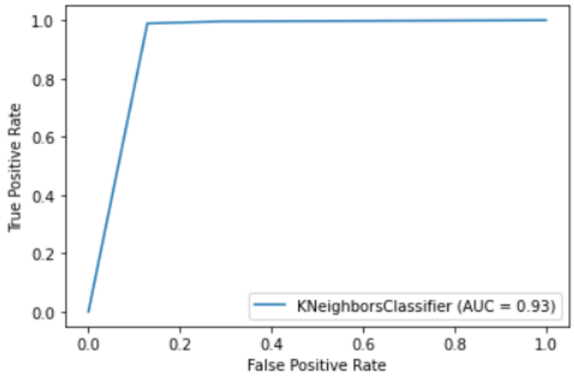


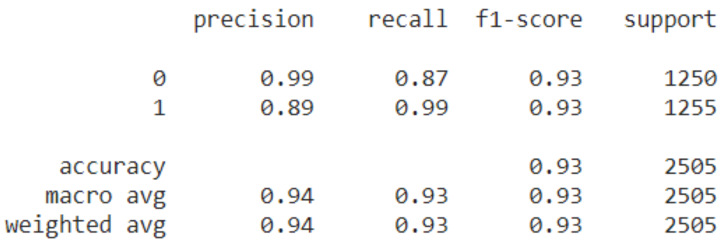
```
              precision    recall  f1-score   support

           0       0.98      0.98      0.98      1250
           1       0.98      0.98      0.98      1255

    accuracy                           0.98      2505
   macro avg       0.98      0.98      0.98      2505
weighted avg       0.98      0.98      0.98      2505
```

- ROC and Confusion Matrix for KNN



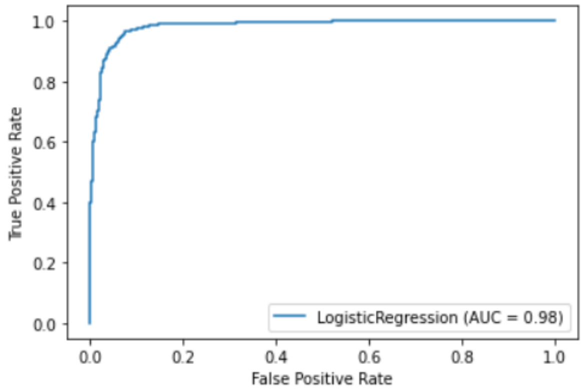


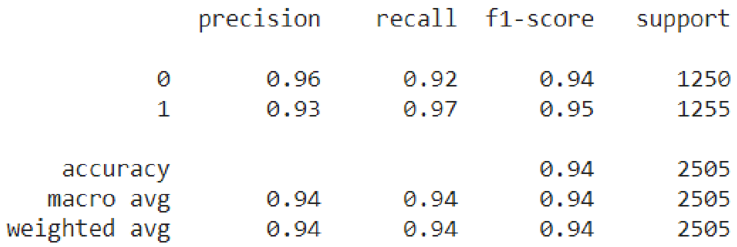
```
              precision    recall  f1-score   support

           0       0.99      0.87      0.93      1250
           1       0.89      0.99      0.93      1255

    accuracy                           0.93      2505
   macro avg       0.94      0.93      0.93      2505
weighted avg       0.94      0.93      0.93      2505
```

- ROC and Confusion Matrix for Logistic Regression





```
              precision    recall  f1-score   support

           0       0.96      0.92      0.94      1250
           1       0.93      0.97      0.95      1255

    accuracy                           0.94      2505
   macro avg       0.94      0.94      0.94      2505
weighted avg       0.94      0.94      0.94      2505
```

Accuracy Table :

Classifier	Accuracy
CNN	**98.84 %**
SVM - Gaussian	98.16 %
Logistic Regression	94.41 %
KNN	93.01%

VIII.CONTRIBUTIONS

For Coding Part :

1. Mohit :- CNN model, SVM (linear + gaussian) ,open cv2, ROC plots
2. Mayank :- KNN,Logistic Regression,data preprocessing,confusion matrices.

IX.Conclusions

From the performance measurement, it is clear that CNN performed the best among the bunch, followed by SVM(gaussian), logistic regression and KNN.

X. Acknowledgment

.We have successfully completed the face with mask recognition by using various classifiers and comparing their performances.We also learnt how to use machine learning pipeline in a project from preprocessing,training a model to classification and finding performance.We got to learn a lot from this project.We express our gratitude to Dr Richa for giving us this opportunity to work on this project.

XI.References

[1] https://www.tensorflow.org/tutorials/images/cnn
[2] https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_tutorials.html
[3] https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
[4] https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
[5] https://machinelearningmastery.com/logistic-regression-for-machine-learning/