Project Report for IDC409

"Say Cheese"

Yogendra Kumar **MS19113**

Indian Institute of Science Education and Research, Mohali

March 1, 2023

Abstract

Smile detection is a specific facial expression analysis job having uses in patient monitoring, user experience analysis, and photo selection. A grin is one of the most significant and revealing facial expressions since it communicates the underlying emotional state, including joy, contentment, and satisfaction. We propose a real-time, accurate, and robust smile detection system. The support vector classifier (SVC) is used to train the classifier. The proposed smile detector is tested with different feature descriptors on publicly available databases including real-world face images. The comparisons against benchmark classifiers including Linear Regression and Random Forest Classifier suggest that the proposed Support Vector Machine based smile detector in general performs better and is very efficient where we get approx 80% accuracy. Compared to state-of-the-art smile detectors, the proposed method achieves competitive results without preprocessing and manual registration.

Contents

1 Objectives

Identify the smiling expression on a face with reasonably high accuracy using methodology and algorithms discussed in class.

2 Introduction

Emotions play an important role in our lives, and recognising emotions is something with widespread applications in corporate sector, administration and law. In our project, we have attempted to solve the problem of classifying images into categories of "smiling" and "non-smiling". We used an applied supervised learning-based technique, Support Vector Classification on a provided set of images containing smiling and non-smiling faces. One such set was used for training the model via SVM and another set of images was used for the prediction and measurement of model accuracy. The dataset was obtained from kaggle.com.

2.1 SVM

Support Vector Machines (or SVMs) are supervised machine learning models that analyse data for classification and regression analysis. The algorithm for SVM performs classifications by identifying a hyperplane separating two different sets of data points. In two dimensional data this means identifying a line in a two-dimensional plane containing the data points which best separates the data. This means that the margin between this line and the data points has to be maximised.

Let data-points are denoted by (X_i, Y_i) with (Y_i) being either 1 or -1 depending on the group of data point it belongs to. We are required to find a hyperplane separating points on the basis of their Y_i values. Equation of the required hyperplane:

$$W^T * X_i - a \geq 0$$

where W is the vector normal to the hyperplane and $a/||W||$ is offset of the hyperplane. We need to maximise the value $a/||W||$. For our uses, a has value of 2.

2.2 Model effectiveness measurement

Confusion matrix

Confusion matrix is a feedback tool used to verify and check results obtained from an ML model as well as gauge its performance. It is usually used for supervised learning techniques.

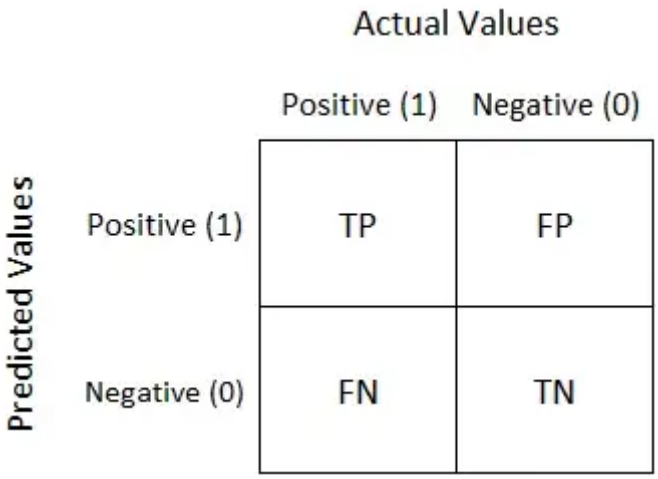

Figure 1: Confusion Matrix

True Positive: Prediction is positive but true.
True Negative: Prediction is negative but true.
False Positive: Prediction is positive but false (Type 1 error).
False Negative: Prediction is negative but false (Type 2 error).

Entries are used to calculate following measurements:
Recall: percentage of correct predictions

$$\textbf{Recall} = \frac{TP}{TP + FN} \tag{1}$$

Precision: From all the classes we have predicted as positive, how many are actually positive.

$$\textbf{Precision} = \frac{TP}{TP + FP} \tag{2}$$

3 Method & Code

```
import os
import random
from imutils import paths
import cv2
import numpy as np
from pylab import *
```

We import some relevant packages here.

```
train_folder = os.path.join(os.getcwd(), "train_folder")
test_folder = os.path.join(os.getcwd(), "test_folder")
```

We save the paths of two folders that contain the test and training set data. The figure below indicates how we have arranged our data:

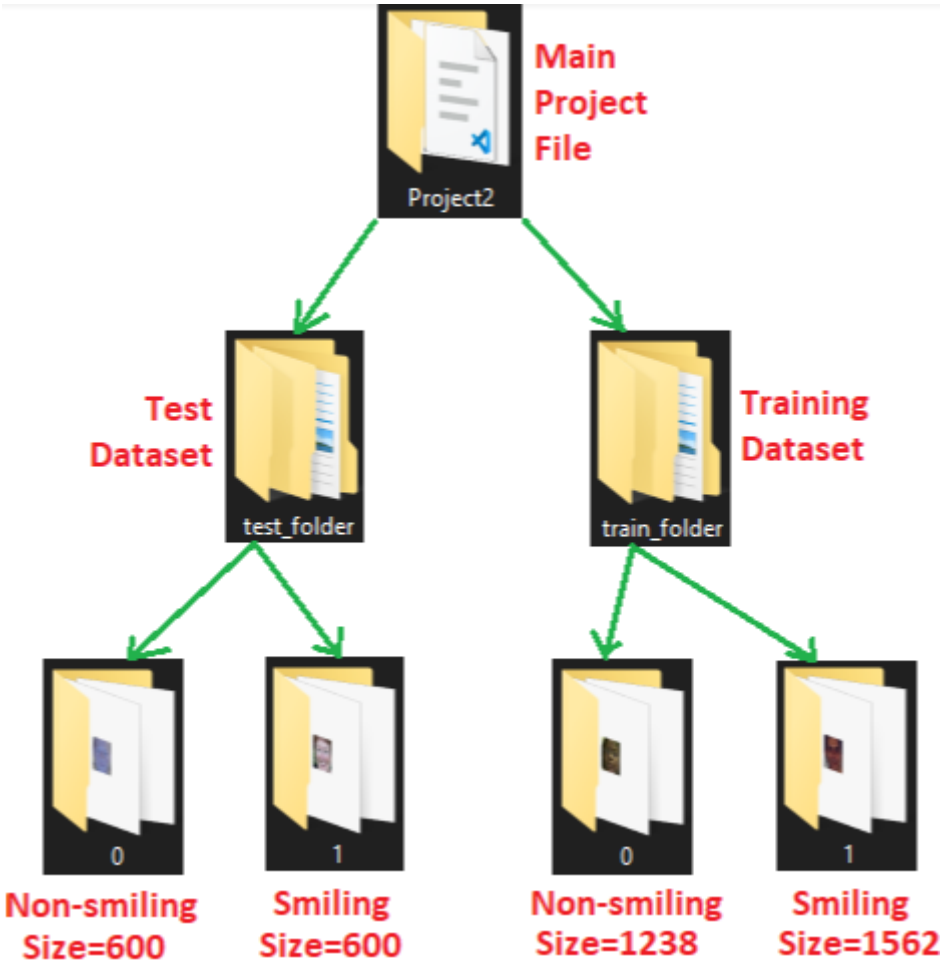

Figure 2: Hierarichal arrangement of data folders

```
def dataextractor(path):
  img_height=32
  img_width=32
```

```
  data=[]
  labels = []
  imagepaths = list(paths.list_images(path))
  for imagepath in imagepaths:
    image = cv2.imread(imagepath)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    image = cv2.resize(image,(img_height,img_width),interpolation=cv2.
   INTER_AREA)
    image = np.asarray(image, dtype=dtype)
    label = imagepath.split(os.sep)[-2] # To find out image labels
    label = int(label)
    labels.append(label)
    data.append(image)
  return np.array(data,dtype='float')/255.0, np.array(labels)
```

This data-extractor function arranges the image data into a rudimentary format on which SVM can be applied. This function acts on the second-level folder, i.e, that folder may contain test or training data. We define the height and width of the image that will be used for processing as 32×32 and we save these height and width data to the img_height and img_width variables.

The path.list_images(path) function lists the paths of all the images that lie any number of levels below the folder path specified in path. Hence, this call gives us the paths of all the images of the folder and we convert it to a list and store it in imagepaths variable.

Running a loop over this imagepaths variable elements, we first image-read the image contained in the path, and store it into a certain image variable. Then we convert this image data to Grayscale from the present RBG format by using the cv2.cvtColor(image, cv2.COLOR_BGR2GRAY). Then we resize this image by passing the defined img_height and img_width constants for all images and we perform resizing by shrinking.

The np.asarray converts the image into the array format, where we have specified the initial and final data types to be same, since we are okay with working with the grayscale-value format of data for each pixel. Next, we are attempting to extract the sample type splitting the path of the image using the split function with the path separator for the operating system, and moving up one level which contains the label of the sample (0 for not-smiling and 1 for smiling). We type-cast this variable into the integer data type and store it into a label variable, later appending it to the labels array list that intends to store the sample types of image data. We also store all the image-array data points into a data list. Next, we return a merged image-array generated from all the images using the np.array function, then scaled by the maximum grayscale pixel range of 255; and return the labels.

```
# splitting the data into train and test
train_X,train_y = dataextractor(train_folder)
test_X,test_y = dataextractor(test_folder)
```

Next, we run this data extractor to load the image data of the training and the test set onto some train_X, train_y variables.

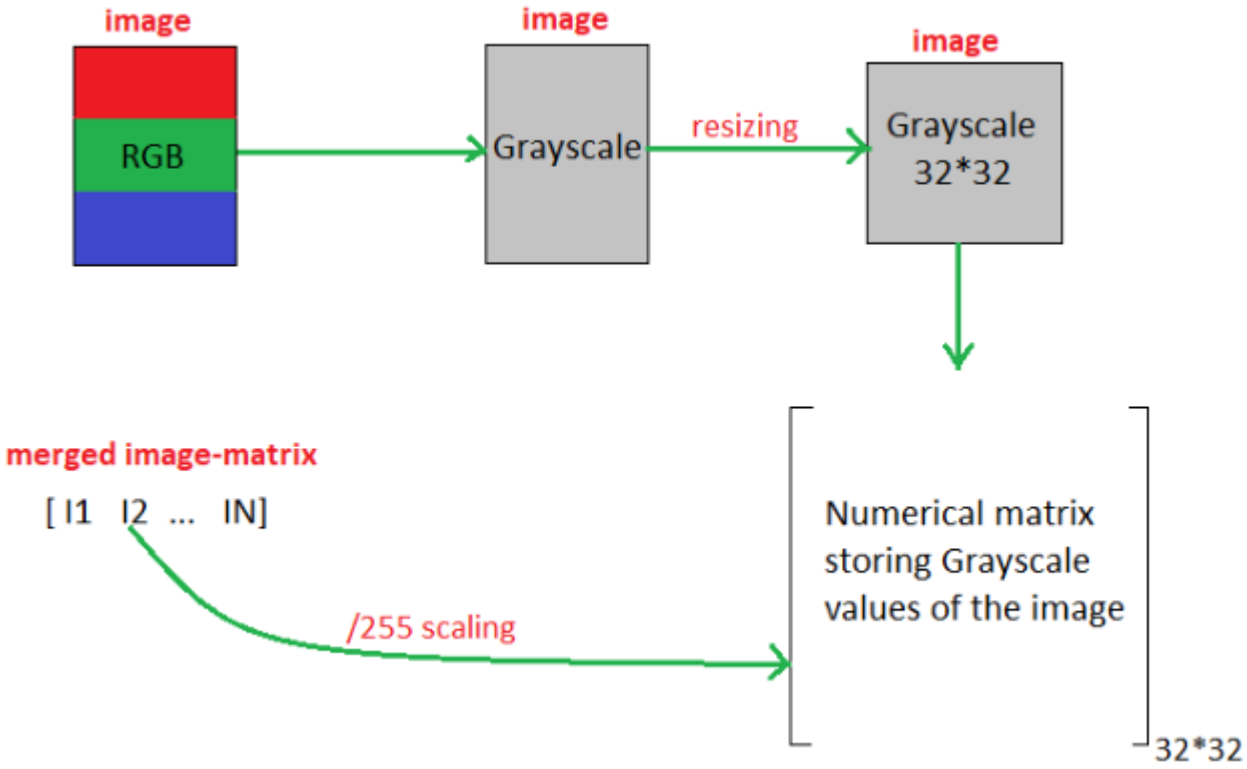

Figure 3: Required image processing, conversion to Matrix, and then arrangement

```
train_X = train_X.reshape(2800,1024)
test_X = test_X.reshape(1200,1024)
```

Next, we reshape the merged-image matrix elements so that they form an $\mathbb{R}$ type data set where $n = 32 \times 32 = 1024$ for both the merged-image matrices.

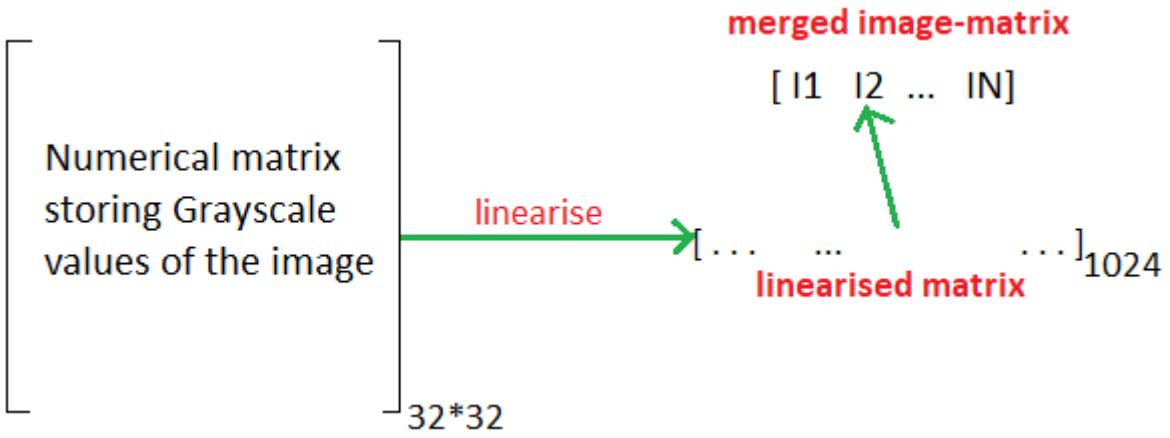

Figure 4: Square-matrix to vector conversion inside the arranged image-matrix elements

```
from sklearn.svm import SVC
model = SVC(kernel='linear')
```

Next, we import the Support Vector Machine package SVC. We set the model of learning to be SVM with a linear kernel. This is the first and default choice and we think about other kinds of Kernels only if this method shows unimpressive performance.

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

Finally, the Support Vector Machine learning is done here. The training set data was best-fit using the model.fit call.

4 Comparison with other ML Models

We train three different kinds of model with the same data and we got the following results:-

Support Vector Classifier: We got accuracy on training set: 89% and acccuracy on testing set: 78%. Following is the confusing matrix for the SVM model:

Total Sample Size = 1300	**Predicted Condition**	
	Positive	**Negative**
Positive	447	153
Negative	114	586

Logistic Regression: We got accuracy on training set: 86% and accuracy on testing set: 79%. Following is the confusion matrix for the Logistic Regression model:

Total Sample Size = 1300	**Predicted Condition**	
	Positive	**Negative**
Positive	462	138
Negative	105	495

Random Forest Classifier: We got accuracy on training set: 99% and accuracy on testing set: 76%. Following is the confusion matrix for the Random Forest Classifier model:

Total Sample Size = 1300	**Predicted Condition**	
	Positive	**Negative**
Positive	417	183
Negative	104	496

5 Results

Evaluating the accuracy of the model using SVM on the training set, we see that the accuracy comes out to be 0.893 or around 89%. Evaluating the same for the test set, the accuracy comes out to be 0.778 or around 78%.

The classification report generated using the **metrics** functionality of the **sklearn** library, gives us the following output:

Classified Value	Precision	Recall	f1-score	Support
0	0.8	0.74	0.77	600
1	0.76	0.81	0.78	600

Metric	Precision	Recall	f1-score	Support
Accuracy	-	-	0.78	1200
Macro Avg	0.78	0.78	0.78	1200
Weighted Avg	0.78	0.78	0.78	1200

The resultant confusion matrix comes out as:

Total	**Predicted Condition**	
Sample Size = 1300	**Positive**	**Negative**
Positive	447	153
Negative	114	586

In the above data we can see that the linear kernel is performing well enough. Upon trying out other kernels like Polynomial (78%) or Radial basis (77.5%) kernels, we see that there is negligible improvement, if at all.
Other algorithms such as Logistic Regression and Random Forest Classifier are also accurate but do not show any significant improvement in accuracy when compared to SVM.

6 Significance

This application of SVM can be expanded to more emotions, like anxiety, anger and surprise. Emotion recognition software is currently being used very widely in the industry for a host of different reasons:

- **For admissions/interviews (HR Assistance):** Models for emotion recognition can be used to observe how candidates feel during an interview and give insights regarding their reactions to individual questions.
- **Evaluating customer service:** Such models can also be used to determine if a customer is satisfied with the customer service. This is done by comparing their facial expression before and after they emerge from the service center.

- **Audience Engagement:** Enterprises in the industry are also using emotion recognition to determine their business outcomes in terms of the audience's emotional responses.
- **Healthcare:** Emotion recognition models are used to determine if a patient needs medicine or extra prescription. It can also be used as feedback for treatment procedures and therapies.

7 Conclusion

Based on the fact that the accuracy of the model on the test set is 78%, we can say that SVM (with linear kernel) is a reliable algorithm to help us classify the images into "smiling" and "non-smiling" categories. It has the best accuracy amongst SVM, Random Forest and Logistic Regression Algorithms making it the best classifier for this task. This approach is compatible with all feature descriptors. To improve results, additional preprocessing steps such as image illumination normalization may be used. The google colab link to the python code is attached here.