Machine Learning Project

Group 16

Divit Sheth	2019A3PS0353H
Gaurav Dash	2019AAPS0274H
Akshat Agrawal	2019AAPS0264H
Jahanvi Gurdasani	2019AAPS0556H
Shambhavi Vishnu Sabhahit	2019A8PS0486H

Paper Implemented:

A robust method for face recognition and face emotion detection system using support vector machines

Github Link for Code Implementation:

https://github.com/Mehr29/Face-and-Facial-Expression-Recognition

Facial Expression Recognition Using Machine Learning Algorithms

Abstract

Facial recognition is a technology that is capable of recognizing a human face from a digital image or video frame. It employs machine learning algorithms which find, capture, store and analyze facial features in order to match them with images of individuals in a pre-existing database. Facial recognition plays a major role in the detection and prevention of crime, particularly identity theft. It also provides businesses a fast and accurate method of identifying individuals in a way that is less intrusive and requires no contact.

Facial expression recognition is a technology which uses biometric markers to detect emotions in human faces. This technology is a sentiment analysis tool and is able to automatically detect the six basic or universal expressions conveying nonverbal communication cues that play an important role in interpersonal relations.

Algorithms Used

Principal Component Analysis (PCA)

For dimensionality reduction, Principal Component Analysis is an unsupervised learning approach. The concept of correlation between the feature vectors is used to reduce the dimensionality of the data. It converts a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables termed principal components via an orthogonal transformation.

Choosing the components with the highest correlation in the data is what PCA is all about. It is carried out by computing the covariance matrix of the data's features and then determining the covariance matrix's eigenvalues and eigenvectors.

Covariance Matrix can be computed as :-

It is of the form

For Population

$$Cov(x,y) = \frac{\Sigma (x_i - \bar{x}) * (y_i - \bar{y})}{N}$$

For Sample

$$Cov(x,y) = \frac{\Sigma (x_i - \bar{x}) * (y_i - \bar{y})}{(N-1)}$$

	f1	f2	f3	f4
f1	var(f1)	cov(f1,f2)	cov(f1,f3)	cov(f1,f4)
f2	cov(f2,f1)	var(f2)	cov(f2,f3)	cov(f2,f4)
f3	cov(f3,f1)	cov(f3,f2)	var(f3)	cov(f3,f4)
f4	cov(f4,f1)	cov(f4,f2)	cov(f4,f3)	var(f4)

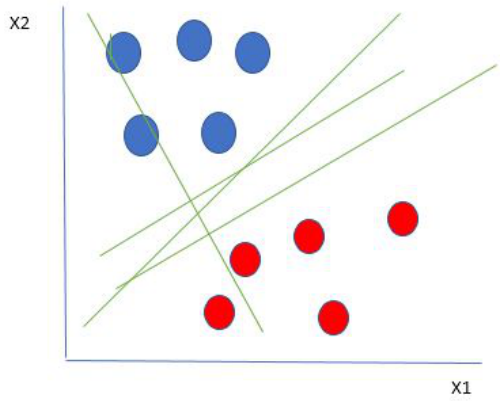
Eigenvalues and eigenvectors are calculated by solving the characteristic equation of the matrix

$$A.X = \lambda.X$$

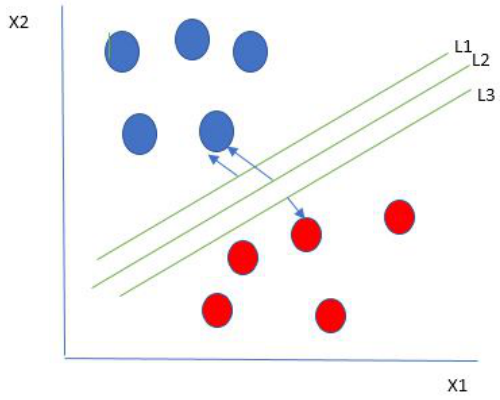
Thus, we get the k Principal components by taking k eigenvectors which correspond to the largest k eigenvalues.

Support Vector Machine (SVM)

SVM is a supervised machine learning technique that may be used for both classification and regression. It's most suited for classification, yet we also say regression difficulties. The SVM algorithm's goal is to find a hyperplane in an N-dimensional space that distinguishes between data points. The hyperplane's size is determined by the number of features.
Consider two independent variables, x1 and x2, as well as one dependent variable, either a blue or a red circle:-



From the figure above it's very clear that there are multiple lines (our hyperplane here is a line because we are considering only two input features x1, x2) that segregates our data points or does a classification between red and blue circles. So to choose the best line or in general the best hyperplane that segregates our data points, we take the best hyperplane as the one that represents the largest separation or margin between the two classes.



So we choose the hyperplane whose distance from it to the nearest data point on each side is maximized. If such a hyperplane exists it is known as the maximum-margin hyperplane/hard margin. So from the above figure, we choose L2. The data points nearest to the boundary on either side are called **support vectors**.
To find the margin, we get the distance of any point from the decision boundary as :-

$$\frac{t_n y(\mathbf{x}_n)}{\|\mathbf{w}\|} = \frac{t_n(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n) + b)}{\|\mathbf{w}\|}.$$

Thus, the maximum margin solution can be formulated as :-

$$\arg\max_{\mathbf{w},b} \left\{ \frac{1}{\|\mathbf{w}\|} \min_n \left[t_n \left(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n) + b \right) \right] \right\}$$

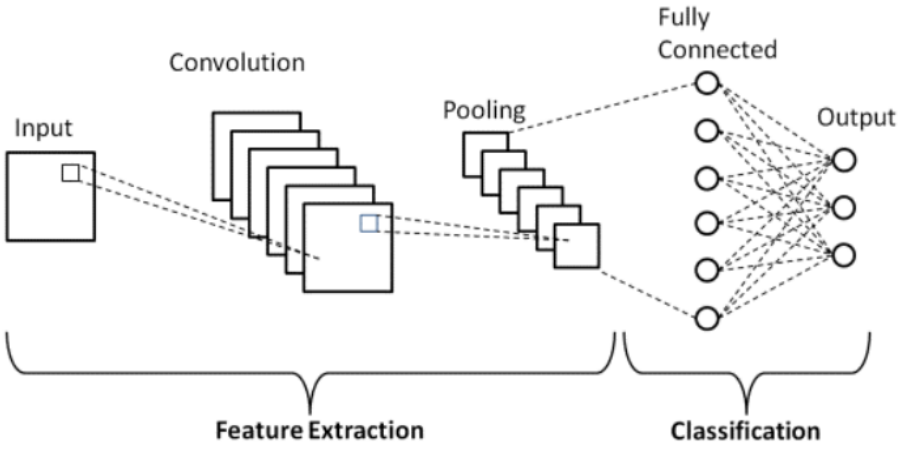
This is a very complex optimization problem and is solved with the help of Lagrange multipliers, which gives the function :-

$$L(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{n=1}^{N} a_n \left\{ t_n(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n) + b) - 1 \right\}$$

In order to classify new data points using the trained model, we evaluate the sign of y(x) defined by (7.1). This can be expressed in terms of the parameters {an} and the kernel function by substituting for w using (7.8) to give

$$y(\mathbf{x}) = \sum_{n=1}^{N} a_n t_n k(\mathbf{x}, \mathbf{x}_n) + b.$$

Convolutional Neural Network (CNN):

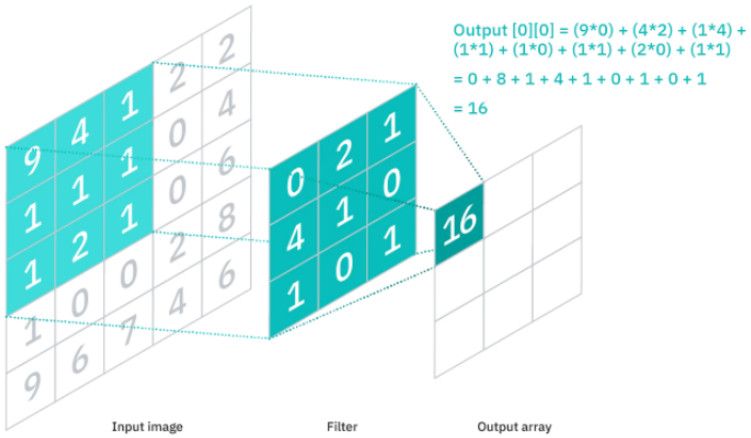


A convolutional neural network (CNN) is a form of artificial neural network that is specifically intended to process pixel input and is used in image recognition and processing. A CNN employs a technology similar to a multilayer perceptron that is optimized for low processing requirements. An input layer, an output layer, and a hidden layer with several convolutional layers, pooling layers, fully connected layers, and normalizing layers make up CNN's layers. The removal of constraints and improvements in image processing efficiency result in a system that is significantly more effective and easier to train for image processing and natural language processing.

A convolutional network's first layer is the convolutional layer. While further convolutional layers or pooling layers can be added after convolutional layers, the fully-connected layer is the last layer. The CNN becomes more complicated with each layer, detecting larger areas of the image. Earlier layers concentrate on basic elements like colors and borders. As the visual data

travels through the CNN layers, it begins to distinguish larger elements or features of the item, eventually identifying the target object.

The feature detector in a CNN is a two-dimensional (2-D) weighted array that represents a portion of the image. The filter size, which can vary in size, affects the size of the receptive field. After that, the filter is applied to a portion of the image, and a dot product between the input pixels and the filter is calculated. After that, the dot product is loaded into an output array. The filter then shifts by a stride, and the procedure is repeated until the kernel has swept across the entire image. A feature map, activation map, or convolved feature is the ultimate output of a series of dot products from the input and the filter. An example of the working of the same is presented below:



Results on Implementation

SVM:Accuracy: 49.27%

```
[[322   0  17 300 235 217 216]
 [ 41   0 186  23  95 180 111]
 [ 68   0   5 256  21 118 101]
 [244   8 222 321  37  94 295]
 [ 10   0  42 197 344 327 337]
 [322   0 203 323  74 123 128]
 [331   0 325 117 127 315 135]]
```

PCA with SVM:

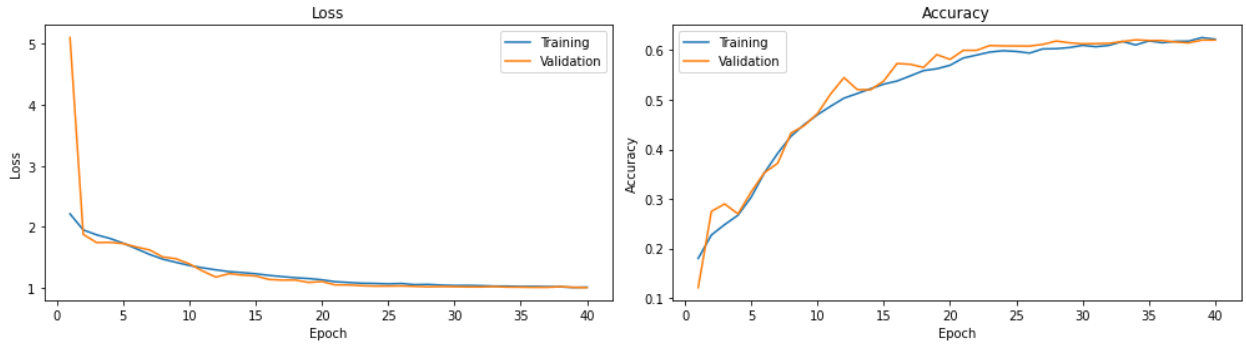
Accuracy : 55.31%

```
array([[ 246,    0,   92,  353,  239,   44,  194],
       [  14,    5,   13,   35,   23,    9,   16],
       [ 107,    0,  255,  292,  268,  120,  174],
       [  73,    0,   75, 1637,  218,   42,  160],
       [  87,    0,   90,  384,  537,   44,  282],
       [  33,    0,   61,  171,   80,  493,  109],
       [  81,    0,   83,  417,  261,   52,  644]])
```

Improvement

CNN:

Accuracy: 65.76%




```
Epoch 20/20
90/90 [==============================] - 9s 99ms/step - loss: 1.0149 - accuracy: 0.6222 - val_loss: 1.0073 - val_accuracy: 0.6203
CPU times: user 2min 2s, sys: 2.43 s, total: 2min 5s
Wall time: 3min 22s
```

```
              precision    recall  f1-score   support

           0       0.55      0.50      0.52       787
           1       0.47      0.19      0.27        72
           2       0.48      0.29      0.36       826
           3       0.80      0.89      0.84      1472
           4       0.45      0.57      0.50       939
           5       0.76      0.78      0.77       623
           6       0.58      0.59      0.59      1023

    accuracy                           0.62      5742
   macro avg       0.58      0.55      0.55      5742
weighted avg       0.62      0.62      0.61      5742

[[ 396    8   49   59  164   15   96]
 [  36   14    6    2   11    3    0]
 [ 130    7  238   38  242   88   83]
 [  21    0   23 1307   38   26   57]
 [  80    1   69   64  538   11  176]
 [  15    0   58   40    7  488   15]
 [  47    0   48  117  201   10  600]]
```

Experiments Conducted So Far:

- We collected the dataset with 28709 labeled images and performed a Support Vector Classifier on it.
- We observed that applying PCA simplifies the complexity of the Support Vector Classifier model and can give better results.
- We tried to implement the Support vector Classifier using different kernels available.
- We then applied Convolutional Neural Networks (CNN) to the images so that it can capture specific features in the images wherever it may be.
- We started with a basic convolutional network of 3 layers and then introduced Dropout and Batch Normalization and also increased the number of layers.
- We got significantly better results by applying the convolutional neural network.

Main Findings and Accomplishments/Conclusions:

Facial emotion recognition was done on the dataset using Principal Component Analysis (PCA) and Support Vector Machine (SVM) as implemented in the paper chosen. Another model, based

on convolutional neural networks (CNN) was used to improve the accuracy of classification. A model based purely on SVM was also used to train the dataset to compare the efficiency of the three models. The accuracies obtained for the algorithms used have been presented below in a tabular format:

Algorithm implemented	Accuracy obtained (%)
SVM	49.27%
PCA along with SVM	55.31%
CNN	65.76%

Contributions:

Divit Sheth	Code Implementation and Report
Gaurav Dash	Code Implementation and Report
Akshat Agrawal	Code Implementation, Improvement and Report
Jahanvi Gurdasani	Code Implementation and Report
Shambhavi Vishnu Sabhahit	Code Implementation and Report