**EMOTION DETECTION USING MACHINE LEARNING**

Emotion Detection from facial expressions has played a significant role for a long time in the field of artificial intelligence and machine learning. Emotion Detection means recognizing human/facial emotions by recognizing certain landmark features from the face and predicting the correct behavior. There is a continuous research how machines can detect the emotions accurately felt by face and signal by our brain. There are numerous algorithms which can detect the emotions of a person and this report will use a few of these available algorithms to implement the best strategy to achieve accurate emotion recognition. Emotion detection is widely used for analyzing security threats, recording suspicious behavior from surveillance, understanding the human behavior patterns to improve the feasibility of applications, patient check in process in health care industry and so on. The other important application of emotion detection is seen in market promotions where clients need complete validation of person’s emotion.

**Methodology**

The main aim of this project is to build a hybrid model with the help of ensemble techniques such as majority voting and principal component analysis. The proposed ensemble technique utilizes multiple models as base models and then integrate them another subsequent model to produce more accurate predictions/results. *I have experimented with the dataset* ***FER 2013*** *as well as* ***CK+48****. Training on the FER dataset took enormous amount of time and the results were not accurate since a lot of images lacked clarity and inaccurate emotion*. The dataset used in our experiment is **CK+48** from Kaggle containing 750 images in total belonging to 5 categories of emotions. These categories include *Anger*, *Fear*, *Happy*, *Surprise* and *Sadness*. Images are of type .png and .jpg format are utilized in the dataset. The data set can be classified into 135 images that correspond to Anger, 75 images corresponding to Fear, 207 images corresponding to Happy, 84 images corresponding to Sadness and 249 images corresponding to Surprise. The dataset is divided into 70 percent for training and the remaining 30 percent for testing and validation. Three algorithms like CNN, VGG19 and Resnet50 were taken into consideration for the purpose of applying ensemble mechanism. These three models are the base classifiers for Ensemble technique. To increase the usefulness of the model, Majority voting method is used to predict the estimator with highest number of votes among three odd classifiers. Necessary libraries like keras and tensorflow for effective Image Processing and the programming language used is python.

**Data Augmentation**

The data is labelled into five emotions such as *Anger*, *Fear*, *Happy*, *Surprise* and *Sadness*. The goal is to reduce the background noise and maximize the feature detection and recognition in the emotion from these images. The key features that help the model identify the emotion can be obtained from primary regions like Eyes, Mouth and cheek boundaries etc.

****

In order to identify these key features, *canny-edge* detection algorithm is used. To achieve this, initially the images are converted to Grayscale. Later, I have applied Gaussian blur to remove the noise. With the help of gradient intensity, I was able to determine the key features from the images. Python’s scikit package provides inbuilt canny edge detections at many levels. By providing multiple levels of canny detection, with a value for sigma set at 3, The above figure to the right shows how the key features are highlighted using 2 levels of sigma for canny detection. We also utilized the flip and rotate properties provided in *ImageDataGenerator()* on our canny images to check if they help assist in better predicting our images. The following depiction is one such sample.



It is also necessary that these images be converted to various dimensional formats for passing them as training inputs for our ensemble mechanism. Principal Component Analysis was also applied/tested on a lower level in or to standardize the data. The reason why standardization is very much needed is, if there are large differences between the scales (ranges) of the features, then those with larger scales will dominate over those with the small scales. PCA makes maximum variability in the dataset more visible by rotating the axes. PCA identifies a list of the principal axes to describe the underlying dataset before ranking them according to the amount of variance captured by each.

**Evaluation**

After augmenting our data into inputs for our models, we begin to split our datasets for training and test/validations. A train size of 0.7 is used to split our data. These split datasets are then forwarded to out models where we started with VGG19, Resnet50 and CNN. Each of these models are trained over 50 epochs. The standard input to the network is 224 x 224 size and consists of 19 layers. The image is streamed through a heap of 16 convolutional layers and 3 fully connected layers. The network contains five sets of convolutional layers with different size filters followed by max pooling in between the convolution layers to bring out the maximum values from the image. The resultant training and testing accuracy are 74% and 65% respectively. The below graph shows how the results of training and testing are obtained using VGG19.



Resnet means Residual Networks. Resnet50 has 48 convolutional layers along with one max pool and one average pool layer. The Resnet50 model achieved an accuracy of 85% after 50 epochs for training whereas it achieved 82% in the testing/validation phase.



A Convolutional Neural Network is a powerful neural network that uses filters to extract features from images. It also does so in such a way that position information of pixels are retained. We have implemented our own CNN models from scratch to train on our datasets. Our CNN model includes 24 layers containing 6 convolutions, 7 batch normalizations, 3 max-poolings, 4 dropouts and 1 dense layer. After training our train/test data, we were able to achieve an accuracy close to 92%. The accuracy and loss graphs look as follows.

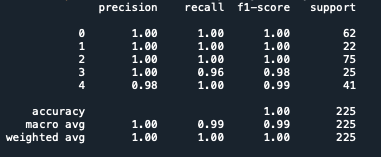


Since deep learning neural networks are nonlinear methods, they offer increased flexibility and can scale in proportion to the amount of training data available. A successful approach to reducing the variance of neural network models is to train multiple models instead of a single model and to combine the predictions from these models. This is called ensemble learning and not only reduces the variance of predictions but also can result in predictions that are better than any single model. The resultant model is then used to further improve our scores by instantiating a hybrid model approach with another model such as ADABoost or XGBoost. The resultant model is used to extract features using feature extraction strategy on trained models. This process of further training can help achieve better predictions than our current results. Our final model consists of the XGBoost classifier which will be used to calculate the predictions and fitness.

Our final model was able to achieve a training accuracy of 100% and testing accuracy of 99%. In our total of 750 images, since the training size was at 70% and the validation size was 30%, our images were split into 525 images for training and 225 images for testing. A confusion matrix plot for the resultant model was looking as following



In the above matrix, the labels 0-4 correspond to Happy, Fear, Surprise, Sadness and Anger. The confusion matrix was able to show us that there was an image which was not predicted accurately. However, most test images were predicted accurately. We can also report the classification report which was achieved as following:



**Other discarded trials**

Initial experiments include using pre-trained VGG16 model on imagenet to extract the generic features from the images like edges, roundness etc. using convolutional filters and feeding the extracted features to the Random Forest model for training. Before feeding the input to the Random Forest, the last step done in feature extractor is flattening the features into single dimension. As machine learning models goes from bottom to top and the top layer used as a classifier to classify images. Then I have combined with custom dense layer and output layers to predict images. After a single epoch run, the accuracy is shown as achieving up to 78% with VGG16 along with Random Forest. These values might be updated in the final report after few training runs. But I have experimented with SVM as well, Using SVM the model seems to be overfitting.

Results