Computer Vision Project : Expression Recognition

Arijit Ganguly
Computer Science and
Engineering Department
University of Texas at Arlington
Arlington, United States of
America
axg1460@mavs.uta.edu

***Abstract*— Expression Recognition using CNN.**

The establishment of social media platforms all over the world has changed technologies to adapt across the globe. One of them being, the introduction of front camera in mobile phones which was realized by the industry after most users started using it to take "selfies". People have taken selfies all over the world to share with families, friends or just to prove that they were at the top of a mountain. Selfies have become an intensive part of social media. An average selfie lover spends 4-5 hours per week taking selfies. This in turn, becomes approximately 260 hours a year just for taking selfies. Across the globe, there are 1 million selfies being uploaded each day. With the advent of social networking sites as a new platform for expression, expressing oneself effectively and accurately is crucial for virtual communication. Social networking giants such as Facebook and Snapchat understand the importance of effective virtual expression and have hence rolled out "How you are feeling" feature (Facebook) and mood filters (Snapchat) to depict one's mood. My project aims to recognize emotions from an camera in real time using implemented CNN modules.

1. INTRODUCTION (*HEADING 1*)

The scope of this project is to use Yale Face expression database to train a model to recognize emotions in an image and use it to predict the expressions in the live camera.

I accomplish this by using multiple machine learning methods. The project uses Local Binary Patterns to extract facial features from a set of images. Then extract histograms from the feature images. For Support Vector Machines, the code uses the histograms for each image as a list for the training data. Alternatively, for Convolutional Neural Network, I am using the Local Binary Pattern Images as the training data. The code is built to store the CNN modules and use the best model available which is marked by the developer.

The live camera emotion recognitions requires the face to be detected to eliminate the noise and receive good predictions from the model. To accomplish this, I use Haar cascades to detect a face in the image and then run the model predictions on detected faces in the frame.

2. ALGORITHMS USED

A. Local Binary Patterns

One of the most common algorithm used for facial feature extraction is the Local Binary pattern.

Local binary patterns are texture descriptors which help in facial feature extraction and getting the feature vector. The main description about Local Binary pattern is written in the famous paper Multiresolution Grayscale and Rotation Invariant Texture Classification with Local Binary Patterns .

These are not Harr like texture features that give a texture based on the Gray Level Co- Occurrence Matrix, LBPS give a local representation of the texture.

The most basic and the first step in LBP is to convert the given image in grayscale. Then, for each of the pixel, a neighborhood of size a is selected for the given center pixel. An LBP is the calculated for the given image and is stored as an 2D array in the output database with given feature vector set.

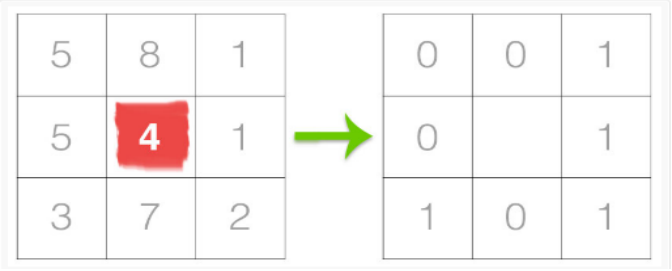


Figure 1: First step in LBP calculation

In the above image example, we can see that an 3x3 neighborhood is chosen. We can then threshold the surrounding pixels with that of the center pixel. If the any of the 8 surrounding pixels is greater than the center pixel then the intensity is set to 1 and if it is less than 1 then the surrounding pixel is set to 0.

The results of this binary encoding is stored in an 8-bit array which can then convert to decimal as shown in the image below:

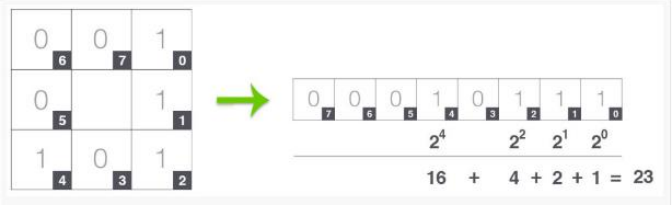


Figure 2: Second step in LBP calculation

In the above example we start at the top rightest corner of the matrix and go clockwise taking the binary string along the way and converting into the decimal equivalent in the end; in the above case, 23.
The above steps are then repeated every time for each pixel.

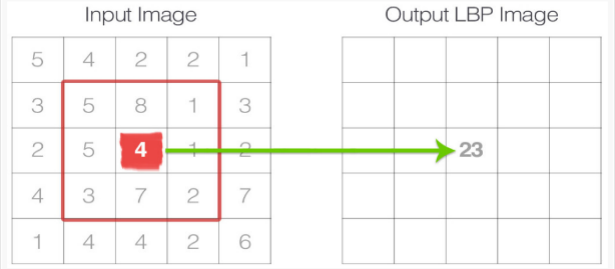


Figure 3: Third step in LBP calculation

The best part about using LBP is that it can capture details at the finest of details however that is also the biggest drawback since we cannot capture at different scales but only 3x3.
To take care of this issue, two parameters were introduced:

1. p is the number of points in a circularly symmetric neighborhood

2. To account for various scales, the radius of the circle r is considered.

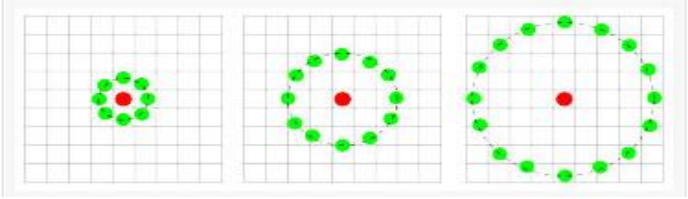


Figure 4: Varying p and r in LBP

B. Haar Feature Cascading

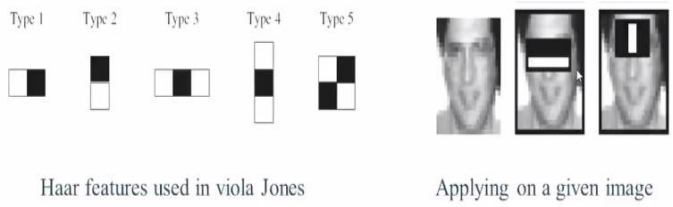


Figure 5: Haar features

Haar features are similar to convolution kernels as described above. In this image shown below, on the left hand side, it is something similar with a black region replaced by +1 and light region is -1. so if we apply these different types of mask to image as shown, then the mask is applied across the whole image and the white region values are added and subtracted from the black region and through we get the total values which help for getting features.

For example, if we take a look in the above diagram, the inversion of Type 3 helps us to identify in the rightmost image that it is a nose and thus applying different masks similar to this helps us in identifying different features on a face.

Identify applicable funding agency here. If none, delete this text box.

Viola-jones 24x24 sub-windows and calculates the values throughout the image till u reach the bottom right corner of the image. So if u calculate with different sizes of the mask on different positions throughout the image, you end up calculating around 160,000+ features in this window.

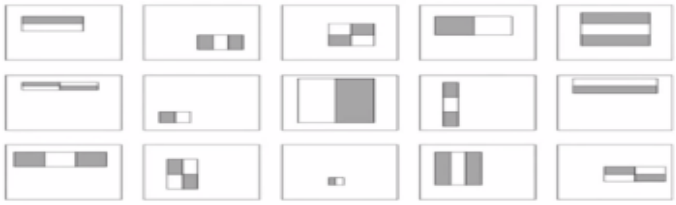


Figure 6: 160,000+ features

For real time face detection this 160,000+ features are too much to go through and recognize the features and we need to narrow it down. This is done by Adaboost which eliminates the redundant features and narrow it down to thousands of features.

Instead of calculating the 2500 in the 24x24 window over the whole image every time, cascades are used in hierarchy i.e. for example, out of 2500 features, first 10 features are kept in one classifier and next 20 features are kept in another classifier and so on. So in the first stage itself u can identify whether it is a face or not and the image can be rejected early on. Therefore, it gives a lot of advantages in real time.

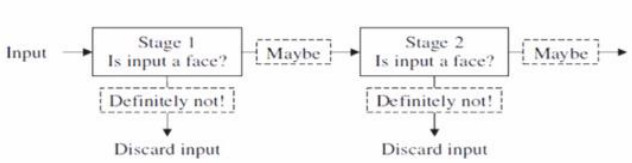


Figure 7: 160,000+ features

C. Convolution

When CNN is fed with a new image, it doesn't seem to know the exact location of these selected features. Instead, it tries to locate them everywhere in the fed image itself. The mathematical procedure of calculating/ determining the match of the selected features in the given picture is called convolution, hence the name Convolutional Neural Network. To come up with a match of the selected feature with a patch/piece of the image, simply multiply each pixel in the feature by the value of the corresponding pixel in the image. Then sum the products and divide it by the total count of pixels in the feature. For both white pixels, the product will be 1 whereas for black the product will be 1 again. In addition to this, any combination of mismatched pixels will result in a -1. All matched pixels result in 1 and all mismatched pixels for the feature will result in -1.

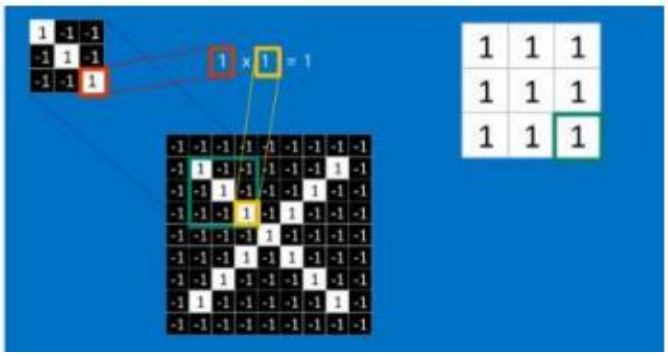


Figure 8: Convolution

This entire process is repeated again and again until the entire image is covered. We try every possible patch of the image. We use the answers resulting from every convolution step and create a new 2D array with a numeric value associated with it. Like the one depicted in the image. This is dependent on the location of the position of the image patch. This produces a map for the Filtered image. The map depicts exactly where the feature is found using the number value associated with the array. 1 is a strong match. -1 is a strong negative and 0 holds for no match.

D. Mutli-Task Cascaded Neural Network

MTCNN (Multi-task Cascaded Neural Network) detects faces and facial landmarks on images/videos. This method was proposed by Kaipeng Zhang et al. in their paper 'Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks', IEEE Signal Processing Letters, Volume: 23 Issue: 10.

The whole concept of MTCNN can be explained in three stages out of which, in the third stage, facial detection and facial landmarks are performed simultaneously. These stages consist of various CNN's with varying complexities.

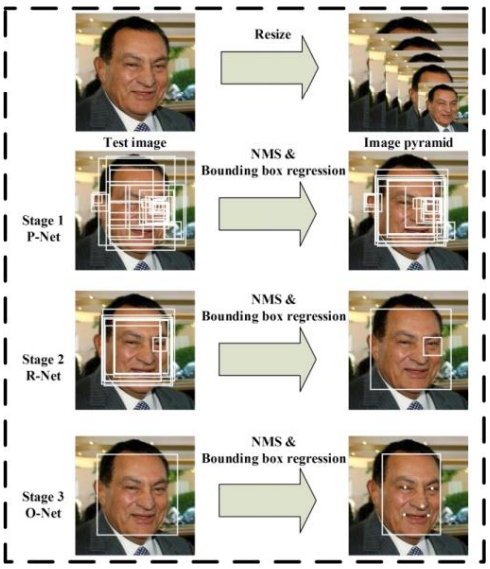


Figure 9: Pipleline for MTCNN

A simpler explanation of the three stages of MTCNN can be as follows :

In the first stage the MTCNN creates multiple frames which scans through the entire image starting from the top left corner and eventually progressing towards the bottom right corner. The information retrieval process is called P-Net(Proposal Net) which is a shallow, fully connected CNN.
In the second stage all the information from P-Net is used as an input for the next layer of CNN called as R-Net(Refinement Network), a fully connected, complex CNN which rejects a majority of the frames which do not contain faces.
In the third and final stage, a more powerful and complex CNN, known as O-Net(Output Network), which as the name suggests, outputs the facial landmark position detecting a face from the given image/video.

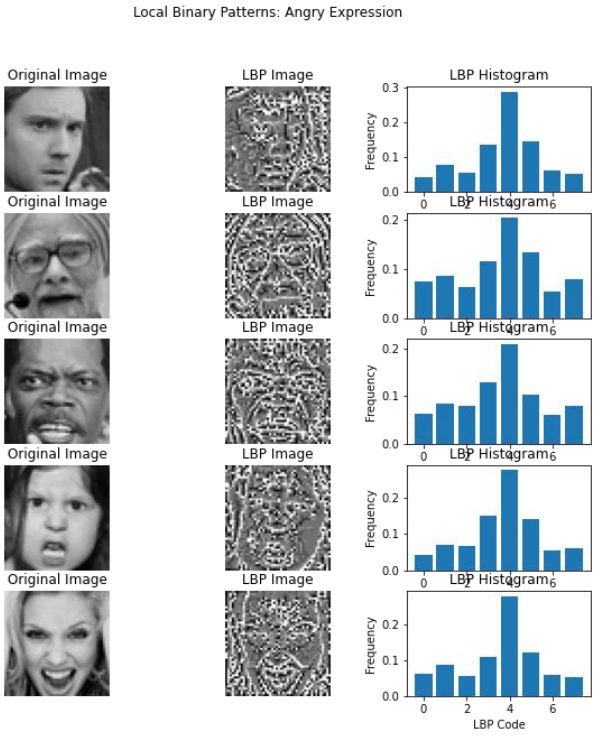
3. PROJECT SPECIFICATION

The project uses the Yale face dataset to build a model that is capable of detecting face expressions in real time. I build multiple machine learning models to achieve accurate results.

The project is divided into multiple phases. The first is the dataset handler. Since the dataset is large, we cannot input the entire dataset into a local system. Hence, a data handler is created to work with the dataset. It will look for the data set folder in the project and get the links to all the files available. The second is feature extraction using Local Binary patterns. For this project to get as much detail as possible from an image, we are using n = 1and p =8. The input to this can be changed to provide results in other options of LBP as well. Third is the last option to build the model from the extracted features. To accomplish good accuracy, I experimented with various models of support vector machine however only MTCNN was capable of providing great accuracy, which can be made better with even more parameter tuning.

A. Feature Extraction

Feature extraction using Local Binary patterns gives us a binarized images while maintaining texture information as well. This important as the human face has many textures and the changes in the textures denote certain expressions based on the emotion we show. Below is an example of an expression from a dataset.

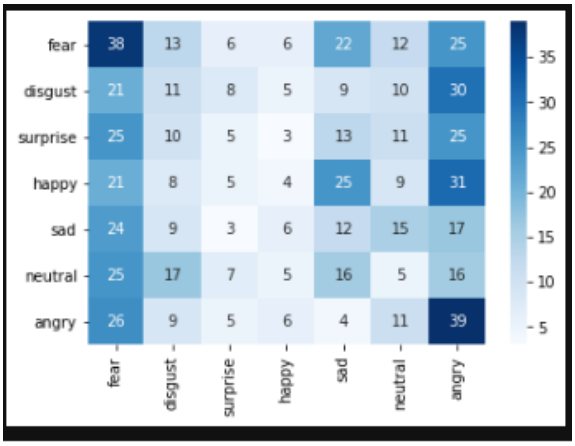


The above plot shows the features extracted from the image of angry expression images in the dataset. The two features extracted are the local binary pattern image and the histogram resulting from the LBP image.

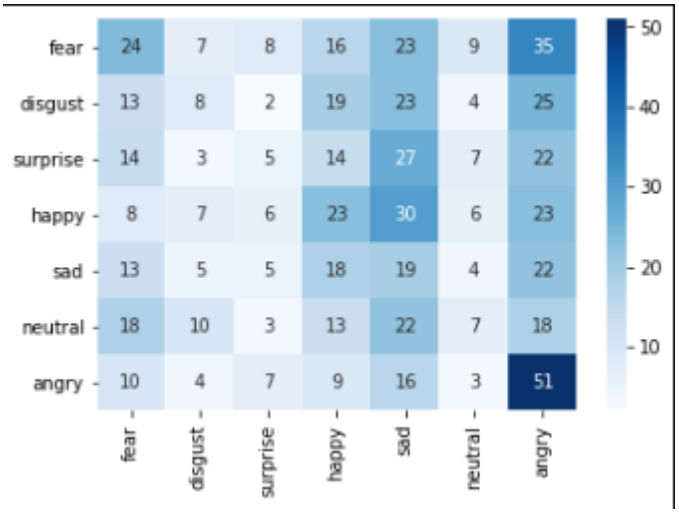
B. Machine Learning Models

Using Support Vector Machines gave close to bad accuracies over the same dataset.

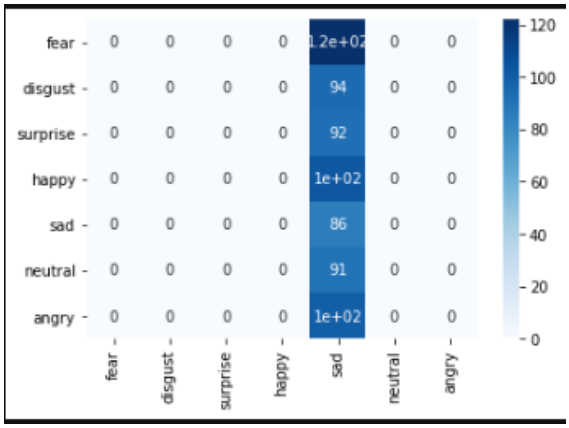
1. LINEAR SVC – 16.6%



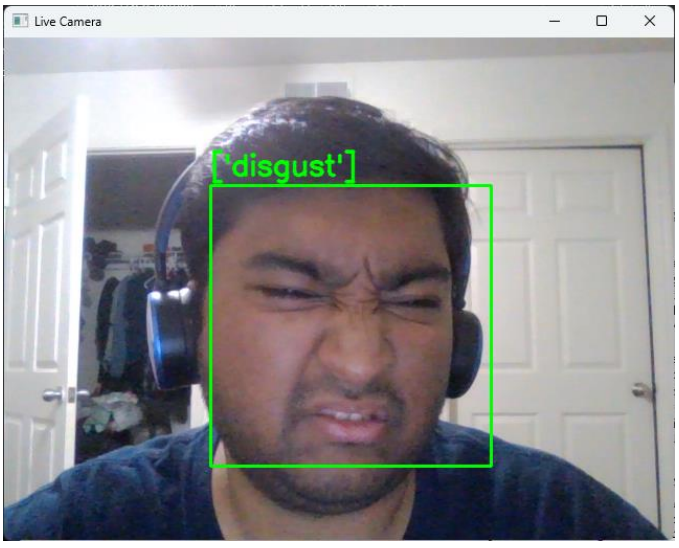
2. NU SVC – 19.9%



3. SVC – 12.5%



Best option was using MTCNN model, with model accuracy reaching 80%.
MTCNN model result on live camera.



REFERENCES

[1] Robust Real-Time Face Detection – Paul Viola and Michael Jones

[2] A_Comparative_Study_of_Multiple_Object_Detection_Using_Haar-Like Feature Selection and Local Binary Patterns in Several Platforms

[3] Multi-Task Cascaded Neural Network

[4] Dataset Link - https://mavsuta-my.sharepoint.com/:f:/g/personal/axg1460_mavs_uta_edu/Ejh9ZBM2TTtGm3UghPaq6uYBmrdcL1FmFg5MCKWGVebTwg?e=911soY

IEEE conference templates contain guidance text for composing and formatting conference papers. Please ensure that all template text is removed from your conference paper prior to submission to the conference. Failure to remove template text from your paper may result in your paper not being published.