Ruprecht-Karls-Universität Heidelberg

Facial Expression Recognition

Fundamental of Machine Learning

Heidelberg Collaboratory for Image Processing (HCI)

Report Submitted By

Muhammad Umair

Hafiz Muhammad Fahad

Raheel Ahsan

23rd March 2018

Submitted to: Dr.Ullrich Köthe

Table of Contents

Abstract

By Muhammad Umair

As we are moving rapidly towards an era of Artificial Intelligence and Machine learning there is a massive increase in human computer interaction (HCI) by means of smart phones, self deriving cars, security systems and etc. There has been a significant improvement in HCI with the help of models which are effective in facial expression recognition. Facial expression recognition have always been a challenging task. In this mini project we have tried to build a classifier that is capable of determining human facial expressions from their images. For our study we tried two different models. The first model is a SIFT-SVM Classifier that detects the key point from an image using Scale Invariant Feature Transformation technique and creates a visual vocabulary with the help of these features after that the transformed features by Support Vector Machine to classify. The second model is a Convolutional Neural Network which has been proven to outperform different models for image classification. The data used in this study is a facial expression recognition (fer_2013) dataset used in kaggle competition.

Data

By Muhammad Umair

The data used for the analysis is facial expression recognition 2013 dataset used in kaggle competition. The data set consist of images with the corresponding expression attached to it. Each instance in the data is a 48 x 48 pixel gray scale image (two dimensional arrays). The labels attached to images are from 0 to 6 where 0,1,2,3,4,5,6 represents anger, disgust, fear, happy, sad surprise, and neutral respectively. During the data screening and preprocessing we found that there is a major class imbalance in the selected dataset that is the class "disgust" has only 547 images as compared to other classes which have more than 4500 images with class "Happy" with the highest number of images that are around 8000. Some examples of data are shown below.

Surprise

Happy

Sad

Neutral

Fear

Angry

Disgust

Data Augmentation

Deep Neural Networks need large amount of data for training to achieve good performance. Data Augmentation is a technique to boost Networks performance. Image Augmentation artificially creates training images through different ways of processing and combinations of processing for example by rotating, shifting or flipping the image. Previous work on image classification using Convolutional Neural Networks has demonstrated the effectiveness of data augmentation through improved performance of the network when applied. These are some of the examples of image augmentation that were applied on the dataset.

Introduction

By Raheel Ahsan

One of the most interesting and challenging task in Computer Vision these days is to recognize human expressions. It is a non-verbal way of communicating and necessary to understand during interaction with human. Therefore a Facial Expression Recognition (FER) algorithm can be very useful in Human-Computer Interface. It has many applications in real world, ranging from ‘Music as per Mood’ to ‘Social Humanoid Robot’. It can prove to be handy in Mental Health Estimators, like ‘Fatigue Detection’, or Law Enforcement, like ‘Lie Detector’. Its use is not just limited to the examples provided above.

Due to its importance in Computer Vision, there have been numerous advancements in making a model for Facial Expression Recognition. One the most successful one of these are using Convolutional Neural Network (CNN). Because of its success in image classification, it has been extended to FER as well. One of the shortcomings of using CNN for this issue is the limitation of Dataset. To overcome this issue, many researches use Data Augmentation to increase the number of samples and to make the model more generalized.

One other method that is often used is to do Feature Extraction and then applying some classification method. Some of the renowned Feature Extraction Algorithm are color histogram, FAST (Features from Accelerated Segment Test), SIFT (Scale Invariant Feature Transform), PCA-SIFT (Principal Component Analysis-SIFT), F-SIFT (fast-SIFT) and SURF (speeded up robust features). The classifier then classifies the features into one of the labels. Most commonly used classifiers are Support Vector Machine (SVM) and K-Nearest Neighbors (KNN).

For our report, we used the dataset ‘FER2013’ available on Kaggle. We first used two models of CNN, shallow and deep. We then used SIFT to extract features and classified those features using SVM.

Scale Invariant Feature Transform (SIFT)

By Muhammad Umair

Scale Invariant feature transformation is a technique to extract local features from an image. The features extracted using SIFT are invariant to image scale and rotations that is, if two images of the same scene are taken from different viewpoints or orientation both the images will have approximately the same local key points. There are four major stages to compute key features from an image using SIFT as mentioned in the most renowned paper in Computer Vision "Distinctive Image Features from Scale-Invariant key points" by David G. Lowe.

1. Scale-space extrema detection:

In the following stage we identify the scale and locations of the key points in the image which can be assigned to the different images of the same object with different view points. In order to detect the locations of key points that are invariant to scale change we try to search for those features that are consistent in number of possible scales. This is done by using a continuous function known as scale space.

$L(x, y, \sigma)$ is a scale space function of an image which is produced by the convolution of Gaussian Kernel on different scales of an input image.

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$$

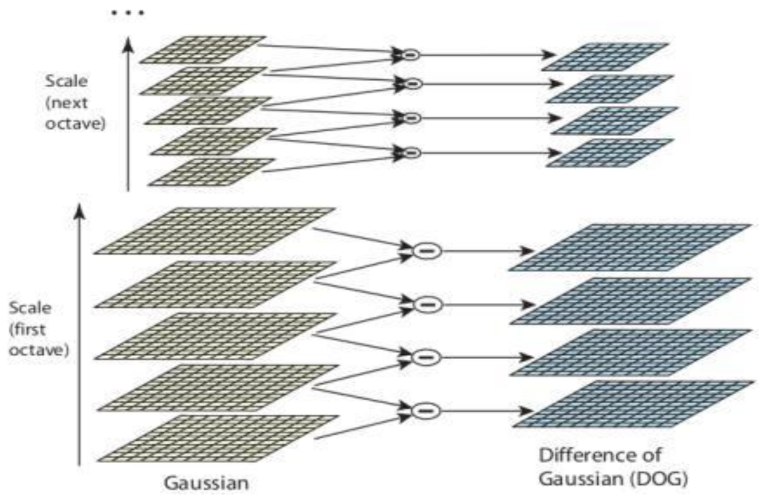
where * is the convolution operator in x, y and

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$

In order to efficiently detect the location of the stable key points the paper proposes to use scale-space extrema in difference-of-Gaussian function which is a close approximation of scale normalized Laplacian of Gaussian which is used in other work on scale invariance. The difference of Gaussian can be computed by the following formula:

$$D(x, y, \sigma) = \big(G(x, y, k\sigma) - G(x, y, \sigma)\big) * I(x, y)$$

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma)$$



The advantage of using Difference-of-Gaussian instead of Laplacian of Gaussian is that it is computationally inexpensive that is we replace the calculation of second order derivative by just simple subtraction.

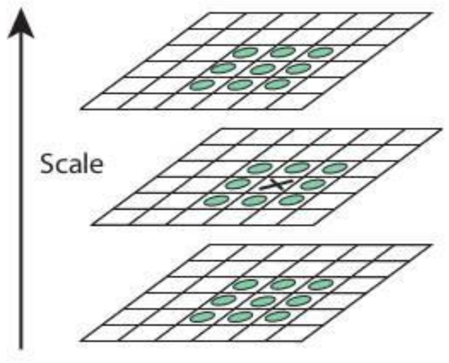
$$\frac{\partial G}{\partial \sigma} = \sigma \nabla^2 G \qquad (heat\ equation)$$

$$\sigma \nabla^2 G = \frac{\partial G}{\partial \sigma} \approx \frac{G(x, y, k\sigma) - G(x, y, \sigma)}{k\sigma - \sigma}$$

and therefore,

$$G(x, y, k\sigma) - G(x, y, \sigma) \approx (k-1)\sigma^2 \nabla^2 G$$

Now each sample point is compared to its eight neighboring points and nine neighboring points in the above and below scale, hence the sample point is only selected if it is greater or smaller than all its 26 neighboring points. Key points are not detected in the lower and upper most scale as there are not enough neighboring points for comparison.



2. Accurate Key point Localization:

After finding the key point candidate by comparing a pixel to its 26 neighboring pixel values, the next step is to perform a detailed fit to nearby data to determine location, scale and ratio of principle curvatures that is using the pixel data we calculate the sub pixel values this is performed with the help of Taylor expansion of the image around the approximate key points

$$D(\mathbb{X}) = D + \frac{\partial D^T}{\partial \mathbb{X}}\mathbb{X} + \frac{1}{2}\mathbb{X}^T \frac{\partial^2 D}{\partial \mathbb{X}^2}\mathbb{X}$$

where D and its derivatives are evaluated at the sample point and $\mathbb{X} = (x, y, \sigma)^T$T is the offset from this point. The location of the extreme points, $\widehat{\mathbb{X}}$, is determined by taking the derivative of this function with respect to x and setting it to zero, giving

$$\widehat{\mathbb{X}} = -\frac{\partial^2 D^{-1}}{\partial \mathbb{X}^2}\frac{\partial D}{\partial \mathbb{X}}$$

Some of the key points generated lie along the edge or have low contrast, in either case both these type of key points are not useful as features, we will try to get rid of these points. For low contrast features we will simply check there intensities. If the magnitude of the intensity at a current pixel in the Difference-of-Gaussian image is less than a certain value it is rejected.

It is not enough to only reject the points that have low contrast because we want to extract those features or key points of the image which are stable. After we have successfully rejected the key points with low contrast we will try to reduce the number of key points by rejecting the key points that lie along the edges.

In order to reject those points which lie along the edge we will calculate the Hessian along that key point.

$$\mathcal{H} = \begin{bmatrix} D_{xx} & D_{yx} \\ D_{xy} & D_{yy} \end{bmatrix}$$

Once the Hessian has been calculated now calculate the trace and determinant of the corresponding hessian matrix,

$$Tr(\mathcal{H}) = D_{xx} + D_{yy} = \alpha + \beta,$$

$$Det(\mathcal{H}) = D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta$$

Where α is the eigen value with the largest magnitude and β is the eigen value with the smallest value, we can compute the sum of eigen values with the trace of hessian and their product with the help of determinant.

Let r be the ratio of largest eigen value to the smaller one then,

$$\frac{Tr(\mathcal{H})^2}{Det(\mathcal{H})} = \frac{(\alpha + \beta)^2}{\alpha\beta} = \frac{(r\beta + \beta)^2}{r\beta^2} = \frac{(r+1)^2}{r}$$

As the ratio of eigen value increases so, to check that the ratio of principal curvatures is below some threshold, r, we only need to check,

$$\frac{Tr(\mathcal{H})^2}{Det(\mathcal{H})} < \frac{(r+1)^2}{r}$$

3. Orientation assignment

After we have done the accurate the accurate key point localization we will collect the gradient directions and magnitudes around each key point, and we will look for the most prominent orientations in the region around that key point and then assign this orientation to the key point.

The further calculation are done according to these orientations, this ensures rotation invariance.

The size of the "orientation collection region" around the key point depends on its scale. The bigger the scale, the bigger will be the collection region.

Gradient magnitudes and orientations are calculated using these formulae:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$

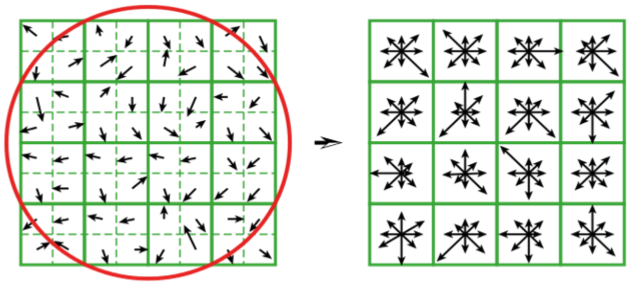
$$\theta(x,y) = \tan^{-1}\left(\left(L(x+1,y) - L(x-1,y)\right)/\left(L(x,y+1) - L(x,y-1)\right)\right)$$

An orientation histogram is formed from the gradient orientations of sample points within a region around the key point. The orientation histogram has 36 bins covering the 360 degree range of orientations. Each sample added to the histogram is weighted by its gradient magnitude and by a Gaussian-weighted circular window with a σ that is 1.5 times that of the scale of the keypoint.

After creating the orientation histogram, we will look for peaks in the histogram once we have detected the highest peak we will look for other local peaks that is within 80% of the highest peak is used to also create a key point with that orientation. Multiple key points will be created at the same location and scale but different orientations for locations with multiple peaks.

4. Local Image Descriptor:

On Application of previous operations location, scale, and orientation have been assigned to the key points of the image. We take 16 x 16 window around each key point, this window is broken into sixteen 4x4 windows as shown in the figure below. The key point descriptor is created by first computing the gradients magnitude and orientation within these 4x4 windows. These are then weighted by a Gaussian window, indicated by the overlaid circle in the figure below.



After performing the following operations we will end up getting 128 numbers for each key point.

Support Vector Machine(SVM)

By Raheel Ahsan

SVMs is one of the Supervised Learning Method. It is used for Classification and Regression. SVM maps all the input vectors on a space on which it finds a hyperplane to separate the classes. To get the maximal distance between the hyperplane and the points, it creates two parallel hyperplanes, equidistant and on either side of the separating hyperplane such that no point lies between them. Samples along these hyperplanes are called 'Support Vectors'. The area between these planes is called 'Margin', and SVM tries to find such a combination of these hyperplanes so that the 'Margin' is maximized. The separating hyperplane is therefore called 'Maximum Margin Hyperplane'. This is based on the assumption that the larger the 'Margin', the better the generalization error of the classifier will be.

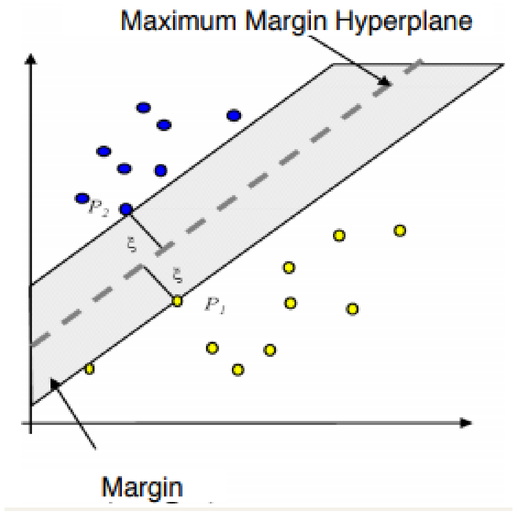


Figure 1

Thus SVM belongs to a family of generalized linear classification.

Modeling the decision boundary

- Consider training data $(\overrightarrow{x_i}, y_i), i = 1, \dots, N$ and let $\vec{x}_i$ be a d-dimensional attribute vector.
- Let class attribute be $y_i \in \{-1, +1\}$.
- Decision Boundary: $\vec{w}.\vec{x} + b = 0$; where $\vec{w}$ is orthogonal to the plane
- Classification of unknown objects $\vec{z}$

$$y = \begin{cases} +1, & \vec{w}.\vec{z} + b > 0 \\ -1, & \vec{w}.\vec{z} + b < 0 \end{cases}$$

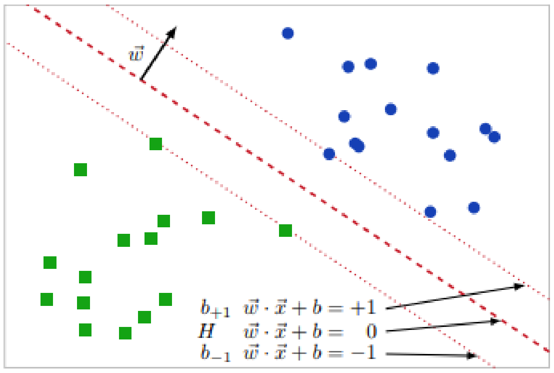

$\vec{w}$
b_{+1} $\vec{w}\cdot\vec{x}+b=+1$
H $\vec{w}\cdot\vec{x}+b=0$
b_{-1} $\vec{w}\cdot\vec{x}+b=-1$

Figure 2

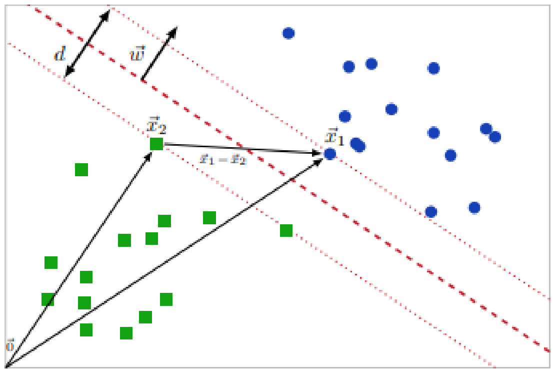


Figure 2

As shown in Figure 2, the equations of the parallel hyperplanes are:

- $b_{+1} = \vec{w}.\vec{x} + b = +1$ (Positive Margin) ①
- $b_{-1} = \vec{w}.\vec{x} + b = -1$ (Negative Margin) ②

Hence as shown in Figure 3:

- $\vec{x}_1$ in class +1 on b_{+1}
- $\vec{x}_2$ in class -1 on b_{-1}
- Simultaneously solving ① & ② we get

$$\vec{w}.(\vec{x}_1 - \vec{x}_2) = 2$$

$$d = \frac{2}{||\vec{w}||} \qquad ③$$

Computing $\vec{w}$ and b

Our constraint (in compact form) for the problem is:

$$y_i(\vec{w}.\vec{x}_i + b) \geq 1, \quad i = 1,2,\ldots.,N$$

The main aim here is to Maximize d in ③, thus minimizing $||\vec{w}||^2$

Hence the SVM model is learned by the following convex optimization problem:

$$minimize \quad \frac{1}{2}||\vec{w}||^2$$

$$subject\ to \quad y_i(\vec{w}.\vec{x}_i + b) \geq 1, \quad i = 1,2, \ldots., N$$

Lagrangian of the Optimization Problem

$$L = \frac{1}{2}\left|\left|\vec{w}\right|\right|^2 - \sum_{i=1}^{N} \lambda_i(y_i(\vec{w}.\vec{x}_i + b) - 1) \qquad where \quad \lambda_i \geq 0 \qquad \textcircled{4}$$

$\lambda_i \geq 0$ only when $y_i\overline{(w}.\vec{x}_i + b) = 1$ i.e. only for the 'Support Vectors'.

Once the λ_i's are known, we can wirte the function of Decision Boundry as:

$$\left(\sum_{i=1}^{N} \lambda_i y_i \vec{x}_i \,.\, \vec{x}\right) + b = 0 \quad where \quad \sum_{i=1}^{N} \lambda_i y_i \vec{x}_i = \vec{w}$$

When we substitute $\vec{w}$ in $\textcircled{4}$, we get dual optimization problem:

$$L_D = \sum_{i=1}^{n} \lambda_i - \frac{1}{2}\sum_{i,j=1}^{N} \lambda_i \lambda_j y_i y_j \vec{x}_i \,.\, \vec{x}_j$$

Hence the classification of an unknown object $\vec{z}$ will be done with:

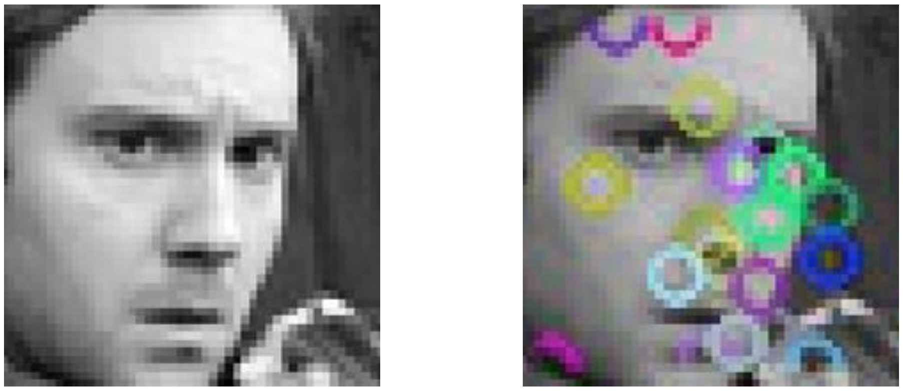
$$y = sign(\vec{w}.\vec{z} + b) = sign(\sum_{i=1}^{N} \lambda_i y_i \vec{x}_i \,.\, \vec{z} + b)$$

Method SIFT -SVM Classification

By Muhammad Umair

The first method applied in this project for classifying facial expression is feature detection through Scale Invariant Feature Transformation and classification through Support Vector Machines. The approach applied is as follow:

Our data consist of 48 x 48 pixel images with different expression labels attached to each image. Following is an example when SIFT was applied on one of the images in training data.



As seen from the above example the figure on the right represents the picture in which the keypoints are marked. There are 24 keypoints in the above image.

Each key point or important feature of the image is represented by a descriptor vector of length 128 values. Each descriptor defines the orientation of the neighbors around the keypoint.

Once key points or important features for every image in the dataset are extracted the next step is to use these features for classification. Unfortunately every image does not have the same number of key points, the only thing which is same in all key points is the length of descriptor vectors.

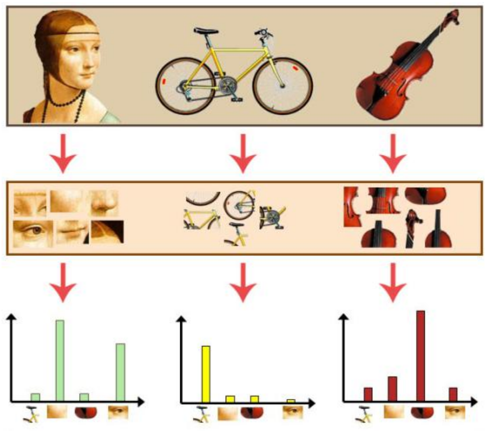
In order to use the key features extracted using SIFT for classification we have to create a Bag of features or the bag of visual words.

Image Representation with a Bag of Visual Words

The concept of visual vocabulary is a strategy which is inspired from text retrieval methods. This strategy has proven its success in object retrieval on large scale database. It enables efficient indexing for local image features.

The visual vocabulary helps us to build a compact summarization of all the image's words. All the key features in every image are used to construct a visual vocabulary this is done by performing clustering on all the key features which in the end will give us limited number of feature or visual words explaining each image.

For constructing the visual vocabulary different types of clustering techniques can be performed. In this project we have used K-mean lustering to build our bag of features.



K-Means Clustering

For the construction of visual vocabulary we have used k-means. K-means is a clustering technique which assign same group, label or cluster to the observation which are similar to each other. There are different ways to ,measure similarity between the point in our case feature. We have used euclidean distance formula to measure similarity. We performed clustering on all the key features of all the images, and initialized number of clusters to 2000.

After constructing the visual vocabulary we will use the key points of every image to construct the histogram of visual words. Now every image can be represented with the 2000 visual vocabulary that is each image is now a vector of length 2000.

Classification Through SVM

Now as we have represented each image as a histogram of visual words. We will use these vector of length 2000 for classification as every image is represented by same number of feature which was not previously the case. We have use Support Vector Machine for this multi class classification problem. As mention before that our data has class imbalance we tried to tackle the problem

Neural Network

By Hafiz Muhammad Fahad

Neural networks are a set of algorithms, modeled loosely after the human brain, that are designed to recognize patterns. They interpret sensory data through a kind of machine perception, labeling or clustering raw input. Neural network recognize the patterns numerically, contained in vectors, into which all real-world data like images, sound, text or time series, must be translated.

Neural networks help us cluster and classify. We can think of them as a clustering and classification layer on top of data you store and manage. They help to group unlabeled data according to similarities among the example inputs, and they classify data when they have a labeled dataset to train on. (To be more precise, neural networks extract features that are fed to other algorithms for clustering and classification; so you can think of deep neural networks as components of larger machine-learning applications involving algorithms for reinforcement learning, classification and regression.)

Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) have been shown to be extremely effective at complex image recognition problems. Convolutional neural networks are mode by simple neurons that have learn-able weights and biases. Each neuron receives inputs then performs a dot product..

Basic Elements of a CNN

i. Convolution layer

ii. Activation function

iii. Batch Normalization layer

iv. pooling layer

v. Fully connected layer

1. Convolutional Layer

Convolution layer is one of the basic layer to build a convolutional neural network. The parameters of the layers consist of learn-able filters. During forward pass every filter (or kernel) convolved with input image volume across width and height. Every filter take a dot product with the entries of the input image, add them all and produce the 2D Activation map of that filter. And number of the filters (K) indicates the depth of the output e.g if input is (n*n) size and we apply K number of filters then the output will be (n * n * K) .

1.1. Padding

When we are making deep neural network we need to use padding. Because when we apply a convolution the output size will be decrease. But we use to add padding to maintain original size of the image.

Types of Padding

"Valid": input (n*n) convolve with filter $(f * f) \rightarrow (n - f + 1 * n - f + 1)$

"Same": after padding output size is same as the input size (n + 2p – f +1 * n + 2p – f +1)

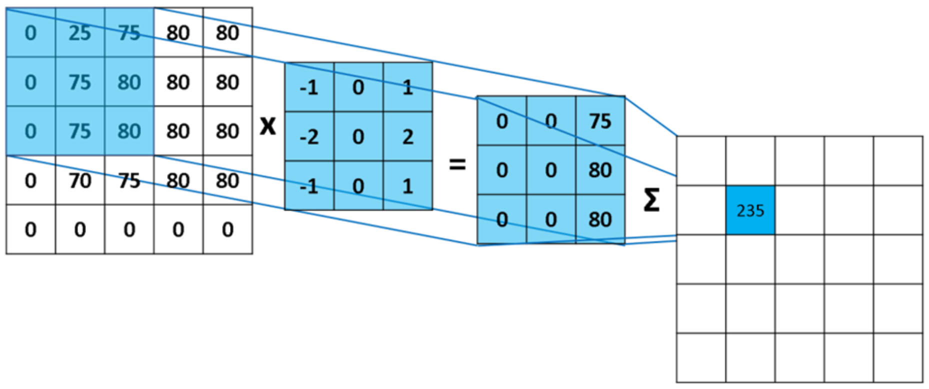
Here, $p = \frac{(f-1)}{2}$, f is usually odd

1.2. Strides

When we apply a filter to the image from left up corner after finding the value in this region we need to step away from left. So strides(S) decide how many cells skip during convolution. The size of the output after convolution also depends of the number of the strides.

(n * n) image (f * f * K) filter (p) padding (s) stride

$$output = \frac{(n + 2p - f)}{s} + 1 * \frac{(n + 2p - f)}{s} + 1 * K$$



2. Activation Function :Nonlinear Transformation

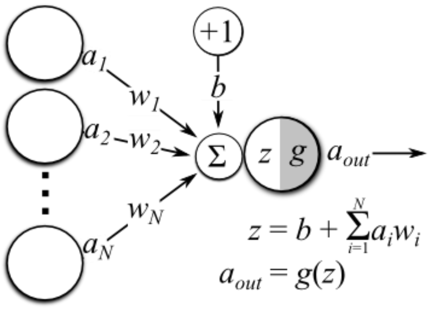
There are two types of activation function

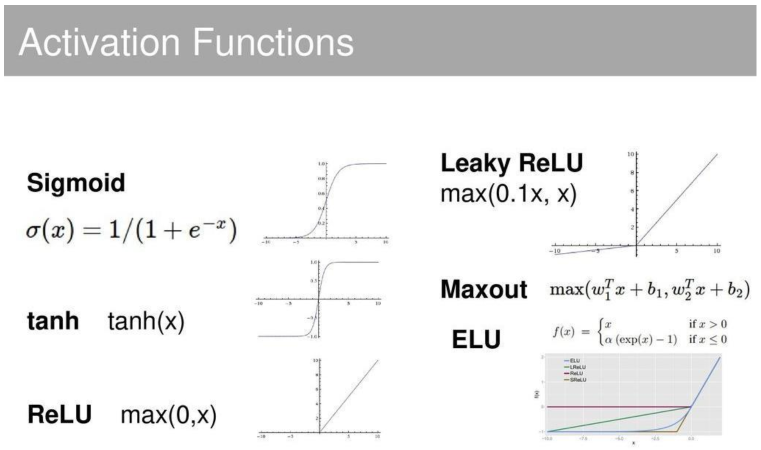
i. Linear activation function and

ii. Non-linear activation function

But we use non linear activation function in neural networks. There are many functions in non linear that we use e.g sigmoid $(1 + e^{-x})^{-1}$, tangent hyperbolic |tanh(x)| , ReLUs max(0,x) etc we use in our model is ReLU.

ReLU is abbreviation of Rectified Linear Units. This is mostly used in neural networks because it trains NN much faster than others without loose of accuracy. If we have a input (a) we apply (w) filter and add a biases (b) and after that apply a Activation function (g) to get new output activation layer $a^{out} = g (w * a + b)$





3. Normalization Layer

Normalization layer have been proposed in the regularization. In practice these layers ease the detection of high frequency (large homogeneous activations are damped). Normalization is good only for 1 layer like you have a input layer and apply a normalization then connect to single neuron, and get a result. But recently, normalization layers have been fallen out of favor, when you have some hidden layers then simple normalization doesn't work perfectly. And have been replace by other alternative regularization techniques (batch normalization etc). In mini batch-wise normalization we consider a subset [x] of training sample .consider only element wise normalization by using this formula

$$\hat{x} = \frac{x(i,j,k) - E\,[x(i,j,k)]}{\sqrt{Var[x(i,j,k)]}}$$

Here E is a mean value and Var is a variance.

Preserve expressiveness of network:

$$x \leftarrow a\,\hat{x} + B$$

Where a, B are learned along with batch normalization

4. Pooling Layer

Pooling is also one of the important layer in convolution neural network. Pooling has a aim to reduce the spatial resolution of the output (down sampling). Standard pooling: is a max pooling in which we apply a kernel to the input and take a maximum value of all in the receptive field.

4.1. Alternative pooling layer

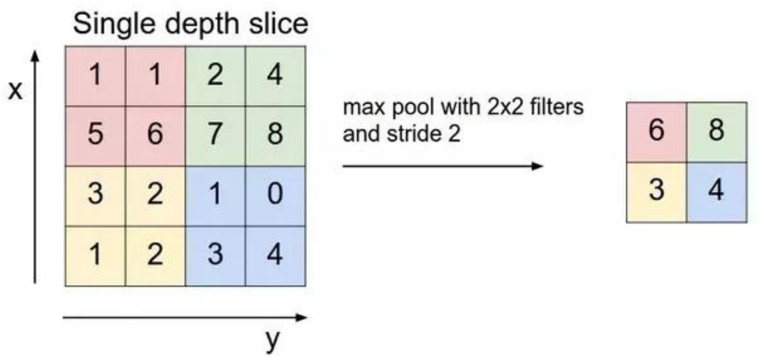
i. Averaging over all values in receptive field

ii. L2-norm over all values in R.F.

The most common form of pooling is 2*2 size kernel with 2 strides. The advantage of pooling is to reduce the number of parameters and this also control the over-fitting.

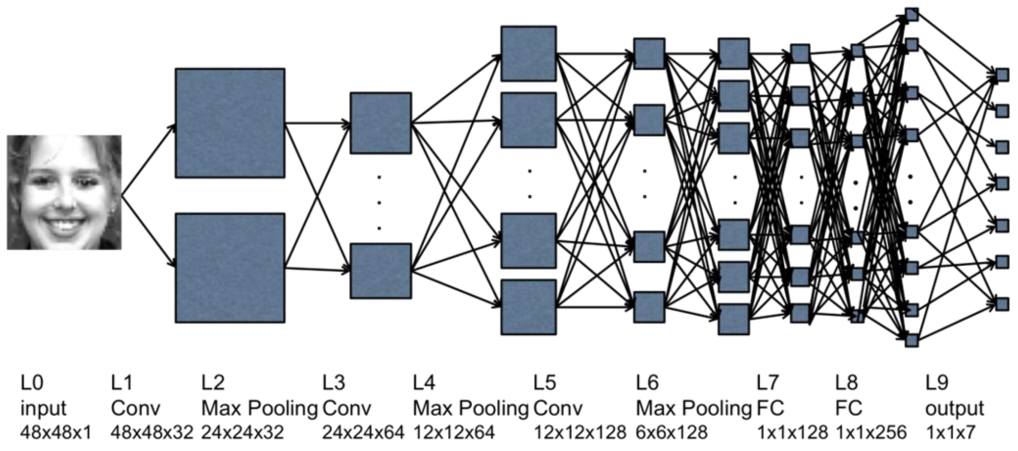
After applying pooling the output size will be

$$output = \frac{(n-f)}{s} + 1 * \frac{(n-f)}{s} + 1 * K$$



5. Fully connected layer

Finally after applying several layers like (convolution layer, max-pooling layer) we are going towards last layer called fully connected layer .in this layer every neurons from previous activation layer connect with all neurons of fully connected layer.



Softmax regression and logistic regression

By Hafiz Muhammad Fahad

Softmax regression also know as multinomial logistic or multi-class classification is a generalization of the logistic regression because here we want to handle multiple classes but in logistic regression we only suppose that labels are binary $y^{(i)} \in \{0,1\}$. We use logistic regression only to classify the 2 classes but in softmax $y^{(i)} \in \{1,2,\ldots.N\}$, where N is the number of different classes.

Our hypothesis functions in the form of:

$$h_\theta(x) = \frac{1}{1+\exp(-\theta^T x)}$$

And θ is the model parameters that were trained to minimize the cost function

But in softmax regression we have N number of different classes so the label y can take N different values.

Let we have given a test input image x and we want to estimate the probability $P(y = k|x)$ for each class $k = 1,2,\ldots\ldots N$. In softmax regression out hypothesis function will give the output N dimensional vector (whose sum is equal to 1) give us K estimated probabilities.

In softmax regression out hypothesis function will be:

$$h_\theta(x) = \begin{bmatrix} P(y=1|x;\theta) \\ P(y=2|x;\theta) \\ \vdots \\ P(y=N|x;\theta) \end{bmatrix} = \frac{1}{\sum_{j=1}^{N} \exp(\theta^{(j)T} x)} \begin{bmatrix} \exp(\theta^{(1)T} x) \\ \exp(\theta^{(2)T} x) \\ \vdots \\ \exp(\theta^{(N)T} x) \end{bmatrix}$$

Here is normalizes the distribution and the sum of all this term is equal to one.

Data Augmentation

In CNNs we need a lot of data to train the model that's why we need to use a data augmentation to increase the data by changing rearranging, reshape, rotating etc.

CNN Model

By Hafiz Muhammad Fahad

There are a lot of libraries in python to make CNN models e.g Tensorflow, keras, pytorch, and many others, first we tried in tensor-flow library and then convert it into keras library because keras is easier to understand as compare to tensor-flow for us and also using Numpy, scikit-learn libraries. We run our code in laptop (i7- 4 cores with 2gb GPU GT-740M) to make our implementation faster.

1. Hyper parameters

Due to less power of GPU we implemented different parameters only in 1 epoch to compare the results that which one is good Convolutional neural network.

i) First we tried to run with 4 convolution and max-pooling layer and 2 fully connected layer with Activation='Sigmoid' and optimizer='adam' we got these results

```
____________________________________________________
None
3589/3589 [==============================] - 19s 5ms/step
loss data: 1.789746799120899
Accuracy on a test data: 0.2513234884451943
```

ii) Then using the same number of layers but with Activation= 'softmax' and optimizer='adam' we got

```
____________________________________________________
None
3589/3589 [==============================] - 20s 6ms/step
loss data: 1.7783573758273312
Accuracy on a test data: 0.27222067428668184
```

iii) And finally we tried with same number of layers with Activation ='softmax' but using with optimizer='SGD'

```
None

3589/3589 [==============================] - 21s 6ms/step

loss data: 1.7918482530944093

Accuracy on a test data: 0.26776260797294543
```

2. Deep & Shallow Convolutional Neural Network

So after trying different parameters only in 1 epoch we got good results with Activation='softmax' and optimizer='adam' so we decided to run our deep and shallow CNNs code with these parameters.

Deep CNN Model

By Hafiz Muhammad Fahad (Running time approximate : 8:30 hours)

Layer (type)	Output Shape	Param #	Filter size	strides
conv2d_1 (Conv2D)	(None, 48, 48, 64)	640	3 x 3	1
activation_1 (ReLU)	(None, 48, 48, 64)	0		
conv2d_2 (Conv2D)	(None, 48, 48, 64)	102464	5 x 5	1
activation_2 (ReLU)	(None, 48, 48, 64)	0		
batch_normalization_1	(None, 48, 48, 64)	256		
max_pooling2d_1	(None, 24, 24, 64)	0	2 x 2	2
dropout_1 (0.25)	(None, 24, 24, 64)	0		
conv2d_3 (Conv2D)	(None, 24, 24, 128)	73856	3 x 3	1
activation_3 (ReLU)	(None, 24, 24, 128)	0		
conv2d_4 (Conv2D)	(None, 24, 24, 128)	409728	5 x 5	1
activation_4 (ReLU)	(None, 24, 24, 128)	0		
batch_normalization_2	(None, 24, 24, 128)	512		
max_pooling2d_2	(None, 12, 12, 128)	0	2 x 2	2
dropout_2 (0.25)	(None, 12, 12, 128)	0		
conv2d_5 (Conv2D)	(None, 12, 12, 256)	295168	3 x 3	1
activation_5 (ReLU)	(None, 12, 12, 256)	0		
conv2d_6 (Conv2D)	(None, 12, 12, 256)	1638656	5 x 5	1
activation_6 (ReLU)	(None, 12, 12, 256)	0		
batch_normalization_3	(None, 12, 12, 256)	1024		
max_pooling2d_3	(None, 6, 6, 256)	0	2 x 2	2
dropout_3 (0.25)	(None, 6, 6, 256)	0		
flatten_1 (Flatten)	(None, 9216)	0		
dense_1 (Dense)	(None, 512)	4719104		
batch_normalization_4	(None, 512)	2048		
activation_7 (ReLU)	(None, 512)	0		
dropout_4 (0.25)	(None, 512)	0		
dense_2 (Dense)	(None, 512)	262656		
batch_normalization_5	(None, 512)	2048		
activation_8 (reLU)	(None, 512)	0		
dropout_5 (0.25)	(None, 512)	0		
dense_3 (Softmax)	(None, 7)	3591		

Total params: 7,511,751
Trainable params: 7,508,807
Non-trainable params: 2,944

Shallow CNN

BY Hafiz Muhammad Fahad

(Running time approximate : 6 hours)

Layer (type)	Output Shape	Param #	filter size	strides
conv2d_2 (Conv2D)	(None, 48, 48, 64)	640	3 x 3	1
activation_1 (ReLU)	(None, 48, 48, 64)	0		
batch_normalization_1	(None, 48, 48, 64)	256		
max_pooling2d_1	(None, 24, 24, 64)	0	2 x 2	2
dropout_1 (0.25)	(None, 24, 24, 64)	0		
conv2d_3 (Conv2D)	(None, 24, 24, 128)	73856	3 x 3	1
activation_2 (ReLU)	(None, 24, 24, 128)	0		
batch_normalization_2	(None, 24, 24, 128)	512		
max_pooling2d_2	(None, 12, 12, 128)	0	2 x 2	2
dropout_2 (0.25)	(None, 12, 12, 128)	0		
conv2d_4 (Conv2D)	(None, 12, 12, 256)	295168	3 x 3	1
activation_3 (ReLU)	(None, 12, 12, 256)	0		
batch_normalization_3	(None, 12, 12, 256)	1024		
max_pooling2d_3	(None, 6, 6, 256)	0	2 x 2	2
dropout_3 (0.25)	(None, 6, 6, 256)	0		
flatten_1 (Flatten)	(None, 9216)	0		
dense_1 (Dense)	(None, 512)	4719104		
batch_normalization_4	(None, 512)	2048		
activation_4 (ReLU)	(None, 512)	0		
dropout_4 (Dropout)	(None, 512)	0		
dense_2 (Softmax)	(None, 7)	3591		

Total params: 5,096,199
Trainable params: 5,094,279
Non-trainable params: 1,920

Result and Evaluation

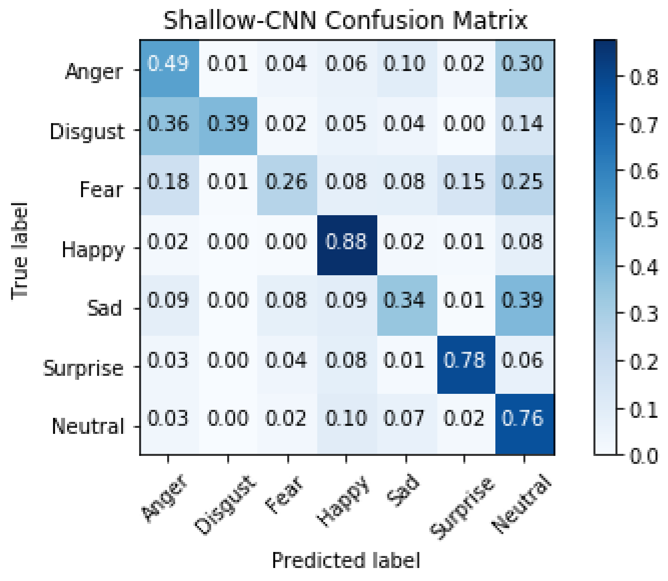
By Raheel Ahsan

Evaluation Metrics Used

- Normalized Confusion Matrix
  - It is a table showing the performance of a classification method on the data for which the labels are already known. It is normalized therefore the rows equals to 1 and the values in each block shows the proportion of the data with actual label in the row, got a prediction of label in the column
- Precision= $\frac{True\ Positive}{True\ Positive+False\ Positive}$
  - The precision calculates the ability of the classifier to not get False Positives. The closer it is to 1, the better.
- Recall= $\frac{True\ Positive}{True\ Positive+False\ Negative}$
  - The recall calculates the ability of the classifier to get all the Positives. The closer it is to 1, the better.
- F-score= 2.$\frac{Precision.Recall}{Precision+Recall}$
  - This is a harmonic mean of Precision and Recall. The closer it is to 1, the better.

Shallow CNN

Test Accuracy = 61.1%



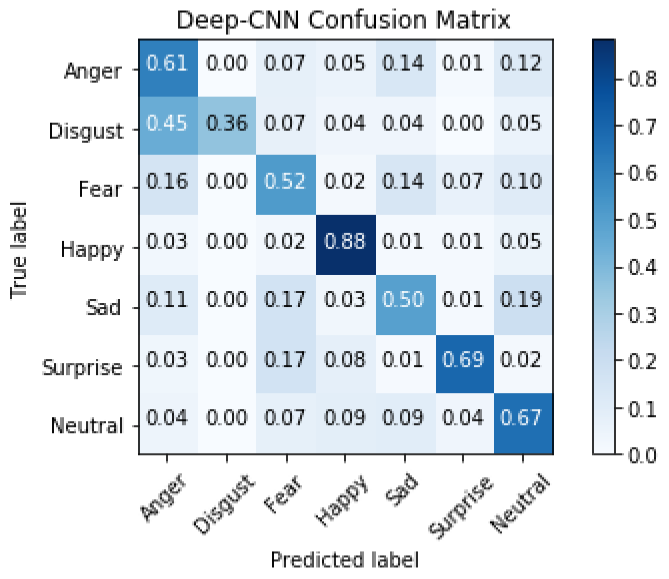
We can clearly see that label ‘Happy’ has the highest proportion as ‘True Positive’. This is because the label ‘Happy’ has the most data. For the similar reason, ‘Disgust’ has the least data and hence has the minimum of its proportion in ‘True Positive’. This is because of class imbalance. But even with this issue, the diagonal of the matrix above has very high proportion than their counterparts.

	'Anger'	'Disgust'	'Fear'	'Happy'	'Sad'	'Surprise'	'Neutral'
Precision	0.53	0.69	0.57	0.79	0.58	0.76	0.42
Recall	0.49	0.39	0.26	0.88	0.34	0.78	0.76
f-score	0.51	0.5	0.36	0.83	0.43	0.77	0.55

The effect of class imbalance can be seen in the table above as well. ‘Happy’ has the best Precision, Recall and F-score. Overall the model looks good other than a couple of low values.

Deep CNN

Test Accuracy = 66.26%.



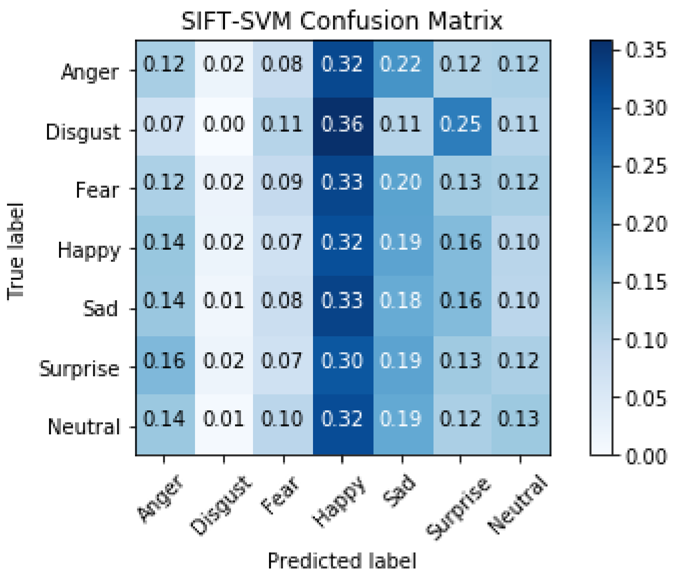
The matrix above is almost the same as the one in Shallow-CNN. Some of the True Positives have increased whereas a few of them got worse.

	'Anger'	'Disgust'	'Fear'	'Happy'	'Sad'	'Surprise'	'Neutral'
Precision	0.56	0.91	0.48	0.85	0.59	0.8	0.59
Recall	0.61	0.36	0.52	0.88	0.5	0.69	0.67
f-score	0.58	0.51	0.5	0.87	0.54	0.74	0.62

The table above also looks quite similar to the one in Shallow-CNN. With ‘Happy’ being classified the best and a couple of low values.

SIFT-SVM Classification

Test Accuracy = 17.84%



Our model of SIFT-SVM Classifier did not work well on this data. It is highly affected by the class imbalance. Most of the each label is classified as ‘Happy’ as it has the most samples.

	'Anger'	'Disgust'	'Fear'	'Happy'	'Sad'	'Surprise'	'Neutral'
Precision	0.12	0.	0.15	0.25	0.16	0.12	0.19
Recall	0.12	0.	0.09	0.32	0.18	0.13	0.13
f-score	0.12	0.	0.11	0.28	0.17	0.12	0.16

The overall performance of this model on every label is really bad. All the values in the above table are closer to 0.

Comparison and Conclusion

By Raheel Ahsan

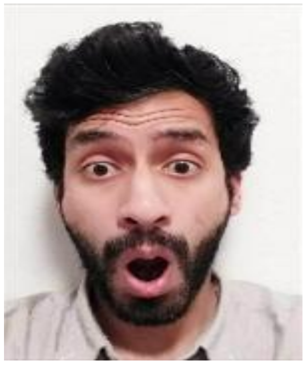
The best result out of the three models that we tried on FER2013 is from Deep-CNN. The recall precision and f-score of all the labels in Deep-CNN and Shallow-CNN are almost similar and so are the values of Normalized Confusion Matrix. The Test Accuracy of Deep-CNN is higher than that of Shallow-CNN. Although the difference is of just 5%, even after doubling the number of layers, but in this case it matters a lot.

The results of SIFT-SVM were not as we expected. The predicted labels are very much skewed towards the label with the highest number of samples (class imbalance). Out of all the papers we read, we did not see anyone use this method and we thought to try it out on this dataset.

We then used our model of Deep-CNN to classify our (authors') pictures just to get a little fame.

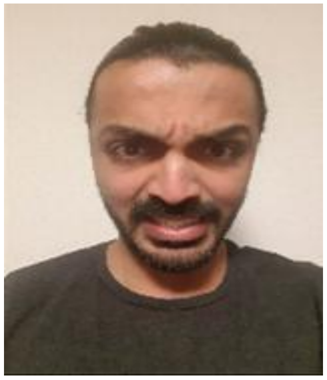
Results

```
['Anger',      'Disgust',     'Fear',      'Happy',     'Sad',      'Surprise',    'Neutral']
[[0.09325562    0.00069405    0.34183317    0.03272475    0.005677    0.5235809       0.00223439]]
Prediction: Surprise
```



Muhammad Umair

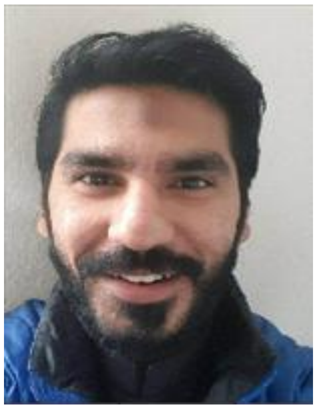
```
['Anger',          'Disgust'          , 'Fear',          'Happy',          'Sad',          'Surprise'     , 'Neutral']
[[0.37158233     0.00452868         0.04909673         0.3655689          0.07495393       0.0514528      0.08281657]]
Prediction: Anger
```



Raheel Ahsan

```
['Anger',          'Disgust',       'Fear',          'Happy',          'Sad',        'Surprise',       'Neutral']
[[7.0045598e-02   4.4072098e-05   2.4001657e-03   8.4454823e-01   9.2627732e-03   2.3653468e-03   7.1333736e-02]]

Prediction: Happy
```



Hafiz Muhammad Fahad

Bibliography

https://mlnotebook.github.io/post/CNN1/

https://keras.io/

https://docs.opencv.org/3.3.0/da/df5/tutorial_py_sift_intro.html

http://scikit-learn.org/stable/supervised_learning.html#supervised-learning

A.Nielsen, M. (2015). Neural Network and Deep Learning. Determination Press.

Batch Normalization From Scratch . (2017). Retrieved from GLUON: http://gluon.mxnet.io/chttp://ufldl.stanford.edu/tutorial/supervised/SoftmaxRegression/hapter04_convolutional-neural-networks/cnn-batch-norm-scratch.html

Challenges in Representation Learning: Facial Expression Recognition Challenge. (n.d.). Retrieved from kaggle.com: https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge

Convolutional Neural Networks-Basics. (2017, april). Retrieved from https://mlnotebook.github.io/post/CNN1/

Deep Learning Tutorial. (n.d.). Retrieved from ufldl.standford.edu: http://ufldl.stanford.edu/tutorial/supervised/SoftmaxRegression/

G.Lowe, D. (2004). Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision* .

Mundher Al-Shab, W. P. (2017). Facial Expression Recognation Using a Hybrid CNN -SIFT Aggregator. *MUlti disciplinary Trends in Artificial Inteligence* .

NG, A. (2018). *Neural Networks and Deep Learning*. Retrieved from coursera.com: https://www.coursera.org/learn/neural-networks-deep-learning/

Shima Alizadeh, A. F. (2017). Convolutional Neural Networks for Facial Expression Recognition.

Sinha, U. (n.d.). *SIFT Theory and Practice*. Retrieved from http://aishack.in/tutorials/sift-scale-invariant-feature-transform-features/

Stefano Berretti Boulbaba, B. A. (2011). 3D facial expression recognition using SIFT descriptors of automatically detected keypoints. *The Visual Computer* .

Zhenhai Liu, H. W. (2015). Effective Facial Expression Recognition via the Boosted Convolutional Neural Network. *CCF Chinese Conference on Computer Vision* .