A Facial Expression Recognition System using comparison with handcrafted and CNN features

Thesis Submitted to the Faculty of Engineering & Technology of Jadavpur University in partial fulfillment of the requirement for the Degree of

Master of Engineering in Software Engineering.

SUBMITTED BY

SUMIT BARAI

CLASS ROLL NUMBER: **001711002007**

EXAMINATION ROLL NUMBER: **M4SWE19011**

REGISTRATION NUMBER: **140967** OF 2017-2018

UNDER THE SUPERVISION OF

Dr. SAIYED UMER

ASSISTANT PROFESSOR
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
ALIAH UNIVERSITY
&

UNDER THE GUIDENCE OF

Dr. BIBHAS CHANDRA DHARA

PROFESSOR

DEPARTMENT OF INFORMATION TECHNOLOGY

JADAVPUR UNIVERSITY

MAY 2019

CERTIFICATE OF SUBMISSION

I hereby recommended the thesis, entitled ***"A Facial Expression Recognition System Using comparison with handcrafted and CNN feature",*** prepared under my supervision by Sumit Barai be accepted in partial fulfillment of the requirements for the degree of Master of Software Engineering from the Department of Information Technology under Jadavpur University.

Signature of Supervisor

Dr. SaiyedUmer

Assistant Professor

Department of Computer Science Engineering

Aliah University

Signature of Guide

Prof. Bibhas Chandra Dhara

Professor

Dept. of Information Technology

Jadavpur University

Countersigned by:

Head of the Department

Dr. BhaskarSardar

Dept. of Information Technology

Jadavpur University

Dean

Faculty of Engineering & Technology

Jadavpur University

DEPARTMENT OF INFORMATION TECHNOLOGY

JADAVPUR UNIVERSITY

CERTIFICATE OF APPROVAL

The thesis at instance is hereby approved as a creditable study of an engineering subject carried out and presented in a manner of satisfactory to warrant its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned does not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein, but approve this thesis for the purpose for which it is submitted.

_____

Signature of Supervisor

Dr. SaiyedUmer

Assistant Professor

Department of Computer Science Engineering

Aliah University

_____ _____

Signature of Examiner

Signature of Guide

Dr. Bibhas Chandra Dhara

Professor

Dept .of IT

Jadavpur University

Declaration of Originality and Compliance of Academic Ethics

I hereby declare that this thesis contains literature survey and original research work by me, as part of my **Master ofEngineering** in **Software Engineering** during academic session 2017-2019.

All information in this document have been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

_____

Name: **Sumit Barai**

Class Roll No.: **001711002007**

Exam Roll No.: M4SWE19011

Registration No.: 140967 OF 2017-2018

M.E. in Software Engineering

Jadavpur University

Thesis Title: ***A Facial Expression Recognition System Using comparison with handcrafted and CNN feature***

Acknowledgement

I would like to express my sincere gratitude to Jadavpur University for providing a conductive environment which helped me to pursue my study in the field of Master of Software Engineering. I am much obliged to be a student of School of Information Technology Department.

I would like to thank distinguished Dr. Bibhas Chandra Dhara for his valuable guidance and technical support in the accomplishment of my thesis. I am indebted to him for giving his precious time and motivation throughout the project.

Also, I would like to express my gratitude to my supervisor Dr. Saiyed Umer for giving me great intellectual freedom to pursue my topic of interest, for his immense patient and understanding and for guiding me each and every step of the process with knowledge and wisdom.

In addition I would like to thank all my seniors and friends for their resourceful advice during these days. Finally, I would also like to thank all teaching and non-teaching staffs who contributed, without any discrimination, to make this environment an excellent place for academic purposes.

Our journey begins by the path shown by our parents. I will be in debt to my parents who sacrificed and dedicated all these times to keep me in the path of honesty and hard work, and also shown faith in me.

_____

(**SumitBarai**)

Class roll no.: **001711002007**

Exam roll no.: M4SWE19011

Registration no.: 140967 OF 2017-2018

M.E. in Software Engineering

Jadavpur University

Kolkata

Abstract

Image classification has gained immense popularity in recent decade. It is widely used in smartphones, cyber security, robotics and many more. Its main objective is to classify images accurately.High accuracy shows how much better the classification model is. Research in this field to get higher accuracy on different datasets is very active. ImageNet is very popular large visual database. It has a huge repository of images (about 14 million) with over 20000 image tags. This is maintained by the computer vision lab at Stanford University.VGG16 and ResNet50 are famous classification models pre-trained on ImageNet dataset. Hence this paper presents an approach of fine-tuning different famous models like VGG16 and ResNet50 with facial expression dataset (KDEF) to get higher accuracy on facial expression dataset since facial expression classification is difficult due to the complexity and variety of facial expressions.

CONTENTS

Introduction

1.1Computer vision

Computer vision is an interdisciplinary scientific field that deals with how computers can be made to gain high-level understanding from digital images or videos. It tries to automate the tasks that the human visual system can do. Just like human can see, process, analyze and recognizes the objects and other beings around them, similarly Computer Vision tasks include methods like acquiring, processing, analyzing and understanding digital images by extracting useful features from images.[1] All this is done by converting digital images into matrix so that computers can process them and gain understanding. The ImageNet challenge or the Large Scale Visual Recognition Challenge (LSVRC) is an annual contest that has various sub challenges such as object classification, object detection and object localization. The LSVRC, especially the object classification challenge, started gaining a lot of attention from the year 2012 when Alex Krizhevsky implemented the famous AlexNet, which has shown great result by reducing the error rate on images to 15.7% (which at that time was never achieved). Now, looking at the latest results, Microsoft's ResNet has achieved an error rate of 3.57% and Google's Inception-v3 has achieved 3.46% and Inception-v4 has gone even further. Thus image classification/recognition has been gaining a lot of attention because of its broad spectrum of applications in different fields like:

- Virtual Reality
- Google smart glasses
- Face recognition in Smartphones
- Cyber security

Computer vision has various sub-fields, few of them briefly discussed in this paper are:

- Image classification
- Object detection

- Face recognition

1.2 Image classification

Image classification involves labelling an image based on the content of the image. Generally data has fixed set of labels and your model will have to predict the label that best fits the image. Computer Vision researchers have come up with many algorithms/models to do so[2].All this is done by extracting useful information from image into feature vectors. These are n-dimensional arrays having all info about image that is needed.

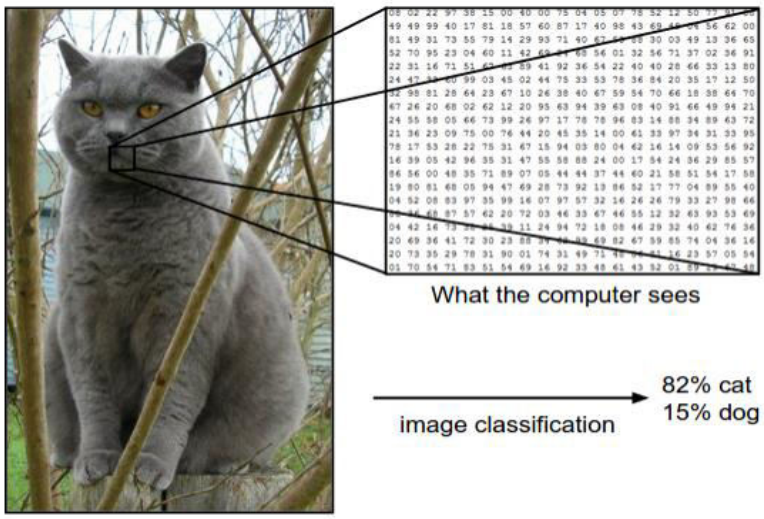

Fig 1.1: Image as a stream of numbers

1.3 Object detection

While object detection in an image involves identifying an object in that image and drawing a bounding box around each recognized object in that image. Also based on your model, it can detect what object it is.This is slightly more complicated to solve as compared to classification. Now a days it's one of the most famous application can be seen in CCTV cameras.

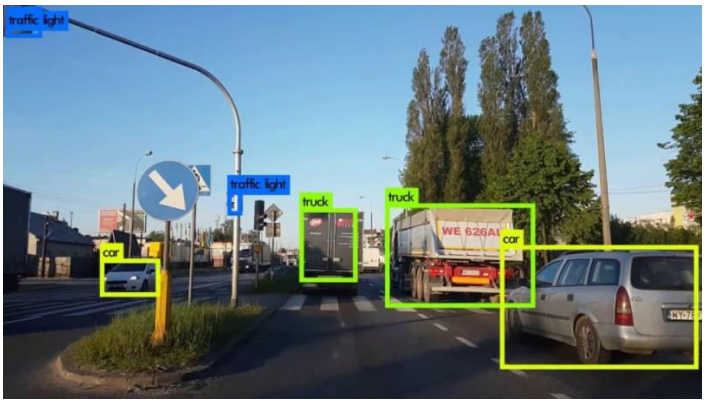

Fig 1.2: Object detection

1.4 Face recognition

Face recognition is also a kind of object detection in which our model has to detect human faces in a given image. But when the model has to recognize a particular individual then it is little bit more complex as it has to be trained with the many images of that individual from different angles. But individual face recognition has gained immense popularity in the last decade. One of the reason issmartphones have come up with this feature called face unlock which is implemented by face recognition.

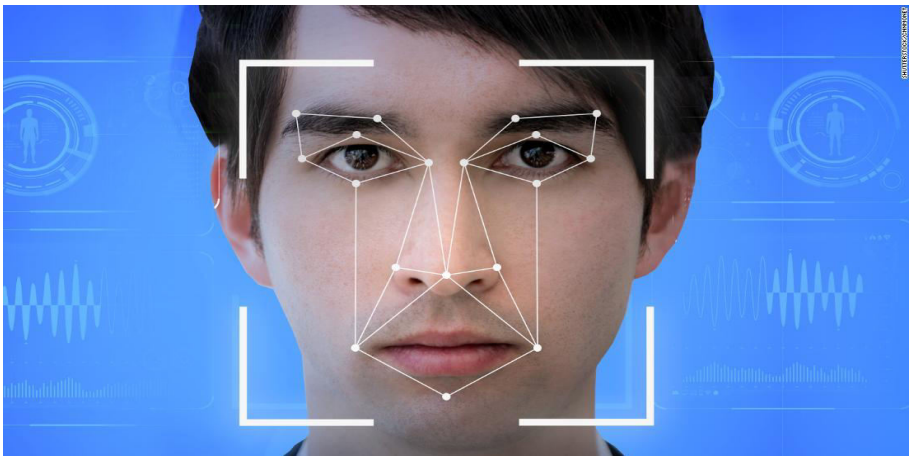

Fig 1.3: Face recognition

1.5 Facial expression

Facial expression can be defined as one or more motions orpositions of the muscles beneath the skin of the face. These movements may convey the emotional state of an individual to the observers. This can also be observed as a form of non-verbal communication. These were the primary means of communication among humans when language was yet to be discovered and even now, expressions form a huge part of our communication.

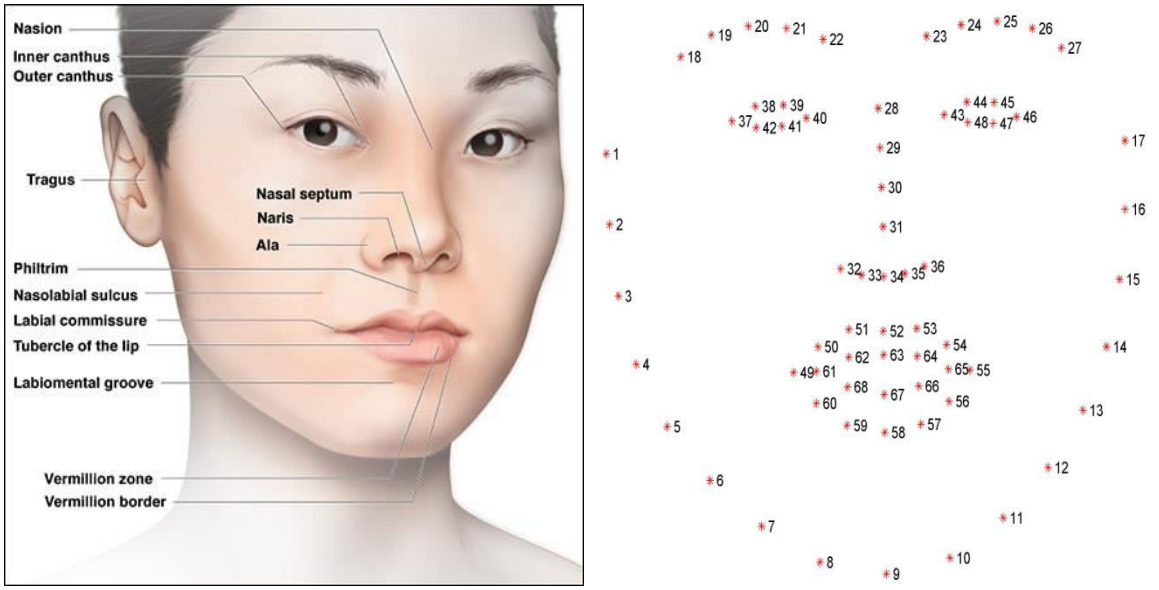

Fig 1.4: Facial landmarks

Facial recognition can also be said to recognize emotions as human emotions are vastly related to the the facial expressions they make under specific circumstances. Emotion recognition is a very important topic and there are a number of applications for this technology. Some of these fields of application include:

- Medicine
- E-learning
- Monitoring
- Marketing
- Entertainment

- Law

One of the example includes old age monitoring remoting and counseling & determining client's medical state. During healthcare, we can even determine patients feeling and comfort level about the treatment. In case of autism, struggling to interpret expressions. In case of e-learning, study the emotions and adjust the learning technique and presentation according to the style of learner. Determining fatigue in case of driving and alerting in advance. When a person is scared and withdrawing money then an ATM not dispensing money (as he might be forced by goons to do so against his will). In case of call-center, determining anger and stress levels in the voice and prioritize angry calls.

We can train models using our large set of training data which also says which facial expression corresponds to what expression. After training our model, we can implement the same for identifying the given facial expression. This is the current technique being used.

The problem seems simple but the accuracy of current models in not good enough to be applicable commercially. This is because we want very good accuracy of identification. Face recognition works excellent giving more than 99% accuracy but since our problem of facial expression is very difficult, therefore accuracy is low. The difficulty arises because we have same persons face and we have to identify the slightest variation on the same persons face to classify it to different expression classes. This happens because geometry of the same person don't change and the change in geometry due to expression is very low. So, this problem of facial expression classification becomes difficult.

Literature review

Many researches are done in the past and several researchers have devised many algorithms for successful high classification accuracy on different databases. Following are some of the significant work done.

Food image recognition using deep CNN [10]. Food recognition is relatively harder problem than conventional image recognition. This paper tried to solve this by the best combination of DCNN related techniques such as pre-training with the large-scale ImageNet dataset, fine-tuning and activation features extracted from the pre-trained DCNN. DCNN was pre-trained with 2000 categories in the ImageNet including 1000 food-related categories, which achieved 78.77% as the top-1 accuracy for UECFOOD100 and 67.57% for UEC-FOOD256;these are the best results on this dataset so far.

Fine-tuning deep learning models for plant disease identification[11] used deep learning with fine tuning on plant images to classify between healthy and diseased plants.They also used different architectures like VGG 16, Inception V4, ResNet with different layers. Data used for the experiment had 38 classes having diseased and healthy images of leafs of 14 plants from plantVillage. Their testing accuracy score was 99.75%. they used Keras with Theano backend to perform the training of the architectures.

A weighted mixture deep neural network (WMDNN) [12] is proposed to automatically extract the features that are effective for FER tasks Expression-related features of facial grayscale images are extracted by fine-tuning a partial VGG16 network, Features of LBP facial images are extracted by a shallow convolutional neural network (CNN) then outputs of both channels are fused in a weighted manner. The result of final recognition is calculated using softmax classification. The average recognition accuracies for benchmarking data sets ``CKC,'' ``JAFFE,'' and ``Oulu-CASIA'' are 0.970, 0.922, and 0.923, respectively.

Facial Expression Recognition with Convolutional Neural Networks [13] has used CNN to classify facial expressions. Authors used two different approaches to get the best results. First they used CNN to implemented three different classifiers: a baseline classifier with one convolutional layer, a CNN with a fixed size of five convolutional layers, and a deeper CNN then they used fine tuning using two existing models VGG16 which is trained on ImageNet and VGGFace which was trained on a facial recognition data set. Thus they used fractional max-pooling and fine tuning, achieving an accuracy of 0.48 in a seven class classification eventually.

Finetuning Deep Model for Object Detection [16] studied and investigated factors that influence the performance in fine tuning for object detection. Their analysis has shown that classes with more samples have higher impact on the feature learning & also it is better to make the sample number more uniform across classes. Their model proposed a hierarchical feature learning scheme. In this scheme, the knowledge from the group with large number of classes is transferred for learning features in its subgroups. Then they Finetuned on the GoogLeNet model, experimental results shown was 4.7% absolute mAP improvement on the ImageNet object detection dataset without increasing much computational cost.

Food Image Recognition Using Very Deep Convolutional Networks [17] used Deep CNN to recognize food images. The model architecture used was Google's image recognition architecture Inception. The architecture is a deep convolutional neural network (DCNN) having a depth of 54 layers. They fine-tuned this architecture for classifying food images from three well known food image datasets: ETH Food-101, UEC FOOD 100, and UEC FOOD 256. On these datasets they achieved, respectively, 88:28%, 81:45%, and 76:17% as top-1 accuracy and 96:88%, 97:27%, and 92:58% as top-5 accuracy. To the best of their knowledge these results were significant improvement than the best published results obtained on the same datasets.

Convolutional Neural Networks for Histopathology Image Classification [18] also used CNN for classification within a medical image data-set. Their experiments are done on Kimia Path24 dataset which consists of relatively large data of 27,055 histopathology training patches in 24 tissue texture classes along with 1,325 test patches for evaluation. Their result shown that pre-trained networks are quite competitive against training from scratch. Also fine-tuning does not seem to add any improvement for VGG16 while they observed considerable improvement in retrieval and classification accuracy when they fine-tuned the Inception structure.

Inception-v3 for Flower Classification [19] also fined tuned imageNet dataset with Inception-v3 model of TensorFlow platform, and used the transfer learning technology to retrain the flower category datasets, which greatly improved the accuracy of flower classification. Datasets used are the Oxford- 17 flower dataset and the Oxford-I 02 flower dataset. Transfer learning method is basically fine tuning which keep the parameters of the previous layer and remove the last layer of the Inception-v3 model, then retrain last layer. The number of output nodes in the last layer is equal to the number of categories in the dataset. For example, ImageNet dataset has 1000 classes, so the last layer has 1000 output nodes in the original Inception-v3 model. The classification accuracy of their model are 95% on Oxford-I7 flower dataset and 94% on Oxford- 102 flower dataset, which is higher than other method.

Training ResNet-50 on ImageNet [20] demonstrated that training ResNet-50 on ImageNet can be achieved within 15 minutes. For such purpose they used 90 epochs with 1024 Tesla P100 GPUs. They made it possible by using a large minibatch size of 32k. They used several techniques such as RMSprop warm-up, batch normalization and a slow-start learning rate schedule. They managed to train the model in 15 minutes and maintained an accuracy of 74.9%.

ImageNet/ResNet-50 Training in a Flash [] tried doing the same as above and achieved better result. Their model addressed primary issues such as instability of a large mini-batch training and the gradient synchronization overhead. Their model eventually managed to reduce the training time to 122 seconds with the validation accuracy of 75.29% using 3456 Tesla V100 GPUs.

Fast R-CNN [22] proposed a new CNN named Fast Region-based Convolutional Network method (Fast R-CNN) for object detection. It is a modified version based on previous version of R-CNN.Fast R-CNN trains the very deep VGG16 network 9× faster than R-CNN, is 213× faster at test-time. Compared to SPPnet, Fast R-CNN trains VGG16 3× faster, tests 10× faster, and is more accurate.

SIFT is a very famous feature to classify images and used by [26] in their paper. Batik is a traditional fabric of Indonesian cultural heritage. Automatic batik image classification is required to preserve the wealth of traditional art of Indonesia. In such classification, a method to extract unique characteristics of batik image is important. Combination of Bag of Features (BOF) extracted using Scale-Invariant Feature Transform (SIFT) and Support Vector Machine (SVM) classifier which had been successfully implemented in various classification tasks such as hand gesture, natural images, vehicle images, is applied to batik image classification in this study. The experimental results show that average accuracy of this method reaches 97.67%, 95.47% and 79% in normal image, rotated image and scaled image, respectively.

Hand crafted features

2.1 Handcrafted features

2.1.1 Feature Descriptor

A feature descriptor is a representation of an image or an image patch that simplifies the image by extracting useful information and throwing away other not so useful information. A feature descriptor converts an image of size width x height x 3 (channels) to a feature vector / array of length n.

2.1.2 HOG Feature Descriptor

Histogram of Oriented Gradient (HOG) is also a kind of feature descriptor and has been quite successful in detecting pedestrian because of its small size of feature vector.

HOG descriptor of an image can be calculated for many image sizes but for the sake of understandinglet's take the example in [4] as it has explained it well, taking a patch of image having size of 64 x 128 x 3 and giving output feature vector of length 3780. The feature vector is not useful for the purpose of viewing the image. But, it is very useful for tasks like image recognition, classification and object detection.

In the HOG feature descriptor, the distribution (histograms) of directions of gradients (oriented gradients) are used as features. Gradients (x and y derivatives) of an image are useful because the magnitude of gradients is large around edges and corners (regions of abrupt intensity changes) and we know that edges and corners pack in a lot more information about object shape than flat regions.

HOG feature descriptor used for pedestrian detection is calculated on a 64×128 patch of an image.In this case, the patches need to have an aspect ratio of 1:2. For example, they can be 100×200 or 256×512.

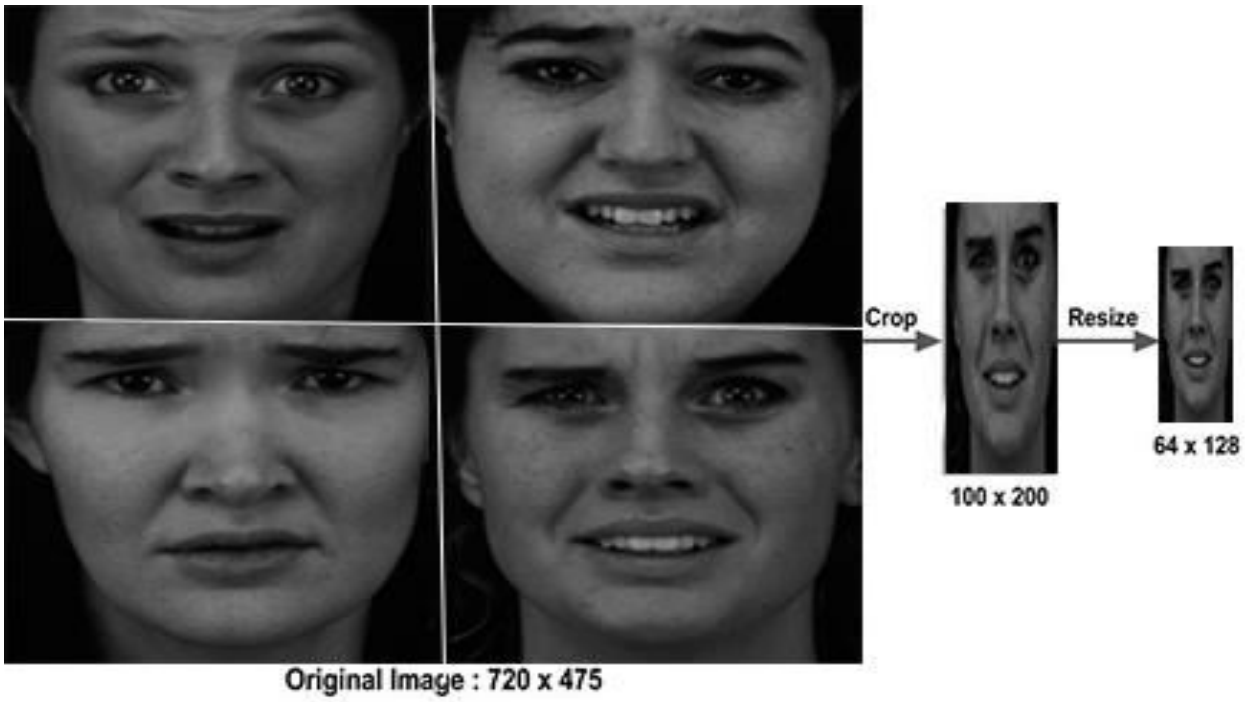

Fig 2.1: Crop the required image

In the image above, large image of size 720×475 is taken. A patch of size 100×200 has been selected for calculating HOG feature descriptor. This patch is cropped out of an image and resized to 64×128. Now it is ready to calculate the HOG descriptor for this image patch.

To calculate a HOG descriptor, we need to first calculate the horizontal and vertical gradients; after all, we want to calculate the histogram of gradients. This is easily achieved by filtering the image with the following kernels.

-1	0	1

-1
0
1

Next, we can find the magnitude and direction of gradient using the following formula

$$g = \sqrt{g_x^2 + g_y^2}$$

$$\theta = \arctan \frac{g_y}{g_x}$$

Figure below shows the gradients:

Fig 2.2: Figure with gradient

The x-gradient fires on vertical lines and the y-gradient fires on horizontal lines. The magnitude of gradient fires where ever there is a sharp change in intensity. None of them fire as the region is smooth.

The patch of image selected earlier is divided into 8×8 cells and a histogram of gradients is calculated for each 8×8 cells. Feature descriptor are very important and one of the main reason it is usedas to describe a patch of an image is that it provides a compact representation. An 8×8 image patch contains 8x8x3 = 192 pixel values. The gradient of this patch contains 2 values (magnitude and direction) per pixel which adds up to 8x8x2 = 128 numbers. By the end of this section we will see how these 128 numbers are represented using a 9-bin histogram which can be stored as an array of 9 numbers. Not

only is the representation more compact, calculating a histogram over a patch makes this representation more robust to noise.

The histogram in HOG descriptor is essentially a vector (or an array) of 9 bins (numbers) corresponding to angles 0, 20, 40, 60 … 160. The image below shows one 8×8 patch in the image and how the gradients look.

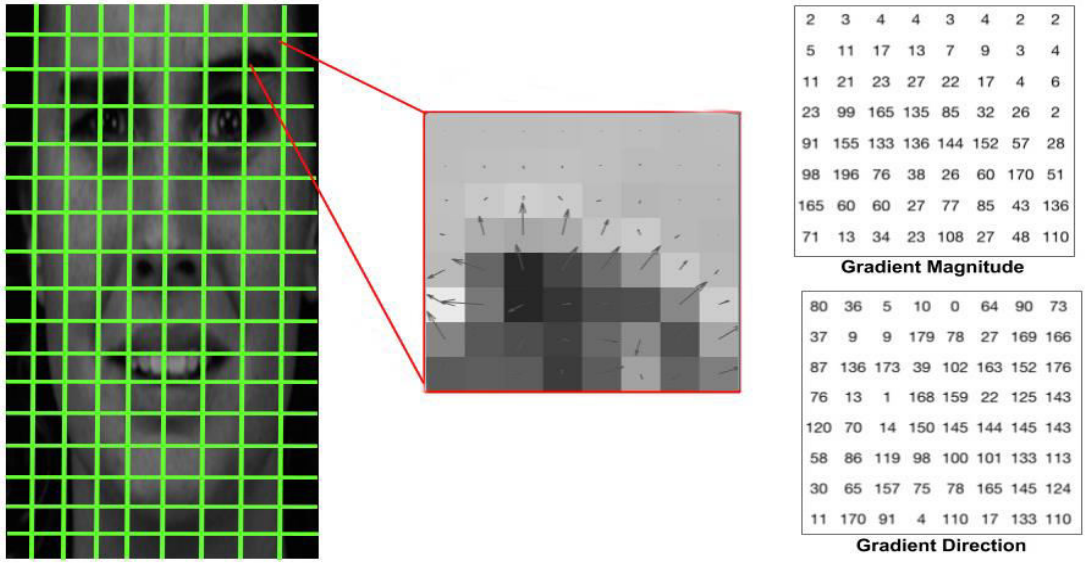

Fig 2.3: Gradient magnitude and direction

Image in the center is very informative. The arrow shows the direction of gradient and its length shows the magnitude.

On the right side of image above, raw numbers representing the gradients in the 8×8 cells with one minor difference — the angles are between 0 and 180 degrees instead of 0 to 360 degrees. These are called “unsigned” gradients because a gradient and it’s negative are represented by the same numbers. In other words, a gradient arrow and the one 180 degrees opposite to it are considered the same. But, why not use the 0 – 360 degrees ? Empirically it has been shown that unsigned gradients work better than signed gradients.

The next step is to create a histogram of gradients in these 8×8 cells. The histogram contains 9 bins corresponding to angles 0, 20, 40 … 160.

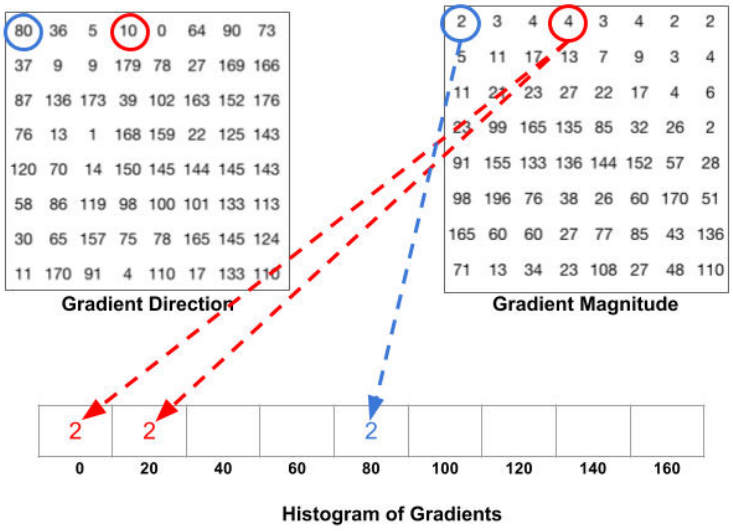

Fig 2.4: Binning (using 9 bins)

The above figure illustrates the process. We are looking at magnitude and direction of the gradient of the same 8×8 patch as in the previous figure. A bin is selected based on the direction, and the vote (the value that goes into the bin) is selected based on the magnitude. Let's first focus on the pixel encircled in blue. It has an angle (direction) of 80 degrees and magnitude of 2. So it adds 2 to the 5th bin. The gradient at the pixel encircled using red has an angle of 10 degrees and magnitude of 4. Since 10 degrees is half way between 0 and 20, the vote by the pixel splits evenly into the two bins.

There is one more detail to be aware of. If the angle is greater than 160 degrees, it is between 160 and 180, and we know the angle wraps around making 0 and 180 equivalent. So in the example below, the pixel with angle 165 degrees contributes proportionally to the 0 degree bin and the 160 degree bin.

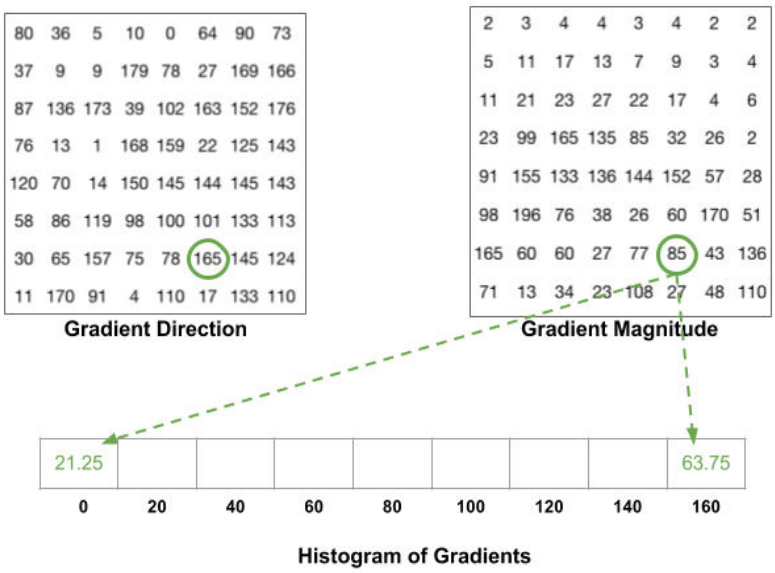

Fig 2.5: Wrap around bin

The contributions of all the pixels in the 8×8 cells are added up to create the 9-bin histogram. For the patch above, it looks like this

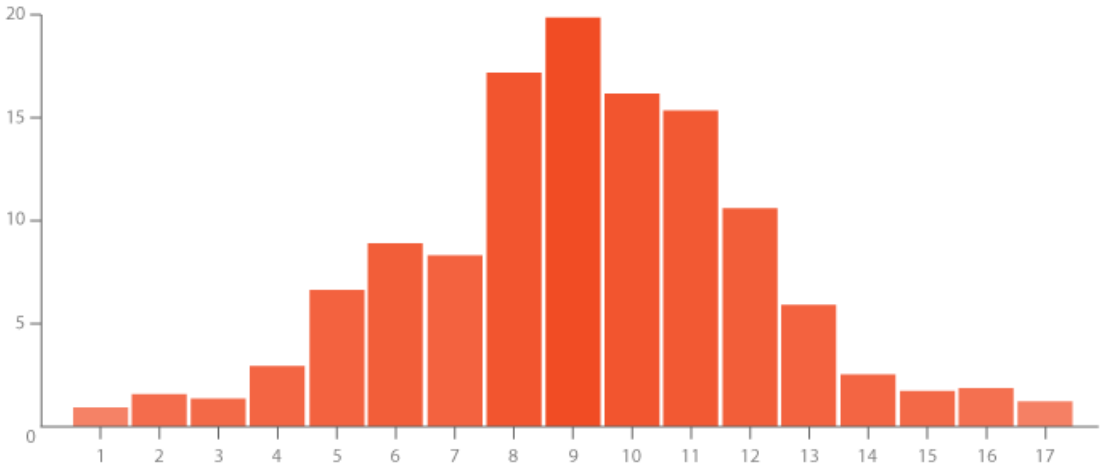

Fig 2.6: 9-bin Histogram

Next step is Normalization. In the previous step, we created a histogram based on the gradient of the image. Gradients of an image are sensitive to overall lighting. If you make the image darker by dividing all pixel values by 2, the gradient magnitude will change by half, and therefore the histogram values will change by half. Ideally, we want

our descriptor to be independent of lighting variations. In other words, we would like to "normalize" the histogram so they are not affected by lighting variations.

One can simply normalize the 9×1 histogram but a better idea is to normalize over a bigger sized block of 16×16. A 16×16 block has 4 histograms which can be concatenated to form a 36 x 1 element vector. The window is then moved by 8 pixels (seebelow picture) and a normalized 36×1 vector is calculated over this window and the process is repeated.

To calculate the final feature vector for the entire image patch, the 36×1 vectors are concatenated into one giant vector. There are 7 horizontal and 15 vertical positions for 16x16 block making a total of 7 x 15 = 105 positions. Each 16×16 block is represented by a 36×1 vector. So when we concatenate them all into one giant vector we obtain a 36×105 = 3780 dimensional vector.

2.1.3 LBP Feature Descriptor

Local Binary Pattern (LBP) [24] is a simple yet very efficient texture operator which labels the pixels of an image by thresholding the neighborhood of each pixel and considers the result as a binary number.

It uses 4 Parameters:

1. Radius: the radius is used to build the circular local binary pattern and represents the radius around the central pixel. It is usually set to 1.
2. Neighbors: the number of sample points to build the circular local binary pattern. Keep in mind: the more sample points you include, the higher the computational cost. It is usually set to 8.

3. Grid X: the number of cells in the horizontal direction. The more cells, the finer the grid, the higher the dimensionality of the resulting feature vector. It is usually set to 8.
4. Grid Y: the number of cells in the vertical direction. The more cells, the finer the grid, the higher the dimensionality of the resulting feature vector. It is usually set to 8.

The first computational step of the LBPH is to create an intermediate image that describes the original image in a better way, by highlighting the facial characteristics.

The image below shows this procedure:

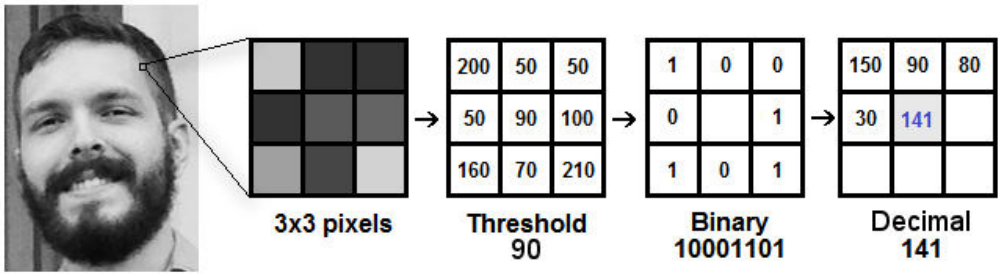

Fig 2.7: LBP on face

Based on the image above, let's break it into several small steps so we can understand it easily:

1. Suppose we have a facial image in grayscale.
2. We can get part of this image as a window of 3x3 pixels.
3. It can also be represented as a 3x3 matrix containing the intensity of each pixel (0~255).
4. Then, we need to take the central value of the matrix to be used as the threshold.
5. This value will be used to define the new values from the 8 neighbors.

6. For each neighbor of the central value (threshold), we set a new binary value. We set 1 for values equal or higher than the threshold and 0 for values lower than the threshold.
7. Now, the matrix will contain only binary values (ignoring the central value). We need to concatenate each binary value from each position from the matrix line by line into a new binary value (e.g. 10001101). Note: some authors use other approaches to concatenate the binary values (e.g. clockwise direction), but the final result will be the same.
8. Then, we convert this binary value to a decimal value and set it to the central value of the matrix, which is actually a pixel from the original image.
9. At the end of this procedure (LBP procedure), we have a new image which represents better the characteristics of the original image.
10. Now, using the image generated in the last step, we can use the Grid X and Grid Y parameters to divide the image into multiple grids, as can be seen in the following image:

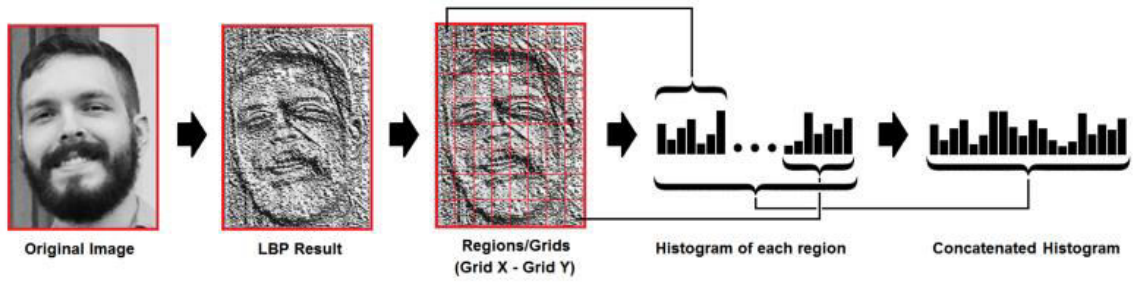

Fig 2.8: LBP result

Based on the image above, we can extract the histogram of each region as follows:

- As we have an image in grayscale, each histogram (from each grid) will contain only 256 positions (0~255) representing the occurrences of each pixel intensity.

- Then, we need to concatenate each histogram to create a new and bigger histogram. Supposing we have 8x8 grids, we will have 8x8x256=16384 positions in the final histogram. The final histogram represents the characteristics of the image original image.

2.1.4 SIFT Feature Descriptor

The scale-invariant feature transform (SIFT) is a feature detection algorithm in computer vision to detect and describe local features in images. The novel image feature extraction was described at 1999 by David G. Lowe who is researches from University of British Colombia. SIFT is a method that is invariant to scale, rotation, and illumination condition.

Matching features across different images in a common problem in computer vision. When all images are similar in nature (same scale, orientation, etc) simple corner detectors can work. But when you have images of different scales and rotations, you need to use the Scale Invariant Feature Transform. Thus SIFT is useful in such cases.

SIFT isn't just scale invariant. You can change the following, and still get good results:

- Scale (duh)
- Rotation
- Illumination
- Viewpoint

SIFT is quite an involved algorithm and described well in [28]. It has a lot going on and can become confusing. Here's brief steps of the algorithm of what happens in SIFT.

1. Constructing a scale space: This is the initial preparation. You create internal representations of the original image to ensure scale invariance. This is done by generating a "scale space".
2. LoG Approximation: The Laplacian of Gaussian is great for finding interesting points (or key points) in an image. But it's computationally expensive. So we cheat and approximate it using the representation created earlier.
3. Finding keypoints: With the super-fast approximation, we now try to find key points. These are maxima and minima in the Difference of Gaussian image we calculate in step 2
4. Get rid of bad key points: Edges and low contrast regions are bad keypoints. Eliminating these makes the algorithm efficient and robust. A technique similar to the Harris Corner Detector is used here
5. Assigning an orientation to the keypoints: An orientation is calculated for each key point. Any further calculations are done relative to this orientation. This effectively cancels out the effect of orientation, making it rotation invariant.
6. Generate SIFT features: Finally, with scale and rotation invariance in place, one more representation is generated. This helps uniquely identify features.

Classification algorithms used

3.1 SVM

"Support Vector Machine" (SVM) [25] is a supervised machine learning algorithm which can be used for both classification and regression challenges. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples. In two dimensional space this hyperplane is a line dividing a plane in two parts where in each class lay in either side. However, it is mostly used in classification problems. Support Vector Machine is a frontier which best segregates the two classes (hyper-plane/ line).

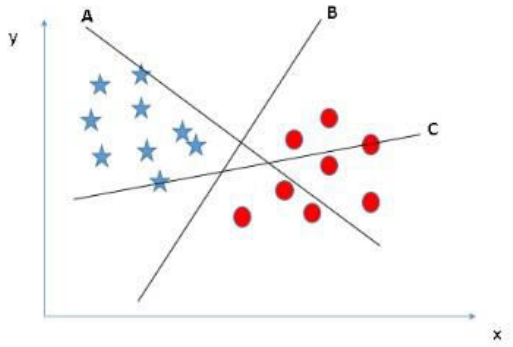

Fig 3.1 SVM classification

In the diagram above, we have three hyper-planes (A, B and C). Now, identify the right hyper-plane to classify star and circle. A thumb rule to identify the right hyper-plane: "Select the hyper-plane which segregates the two classes better". In this scenario, hyper-plane "B" has excellently performed this job.

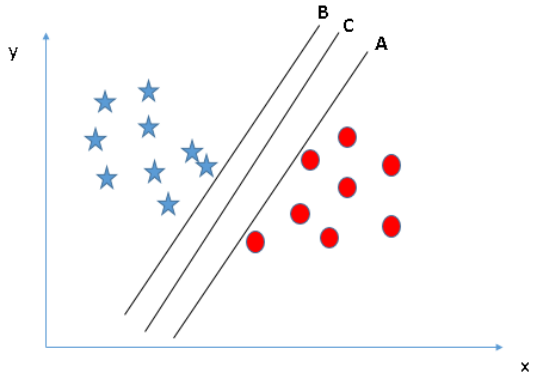

Fig 3.2: Determining Hyperplane

Here, we have three hyper-planes (A, B and C) and all are segregating the classes well. Now, how can we identify the right hyper-plane?

Here, maximizing the distances between nearest data point (either class) and hyper-plane will help us to decide the right hyper-plane. This distance is called as Margin. As shown in the diagram below.

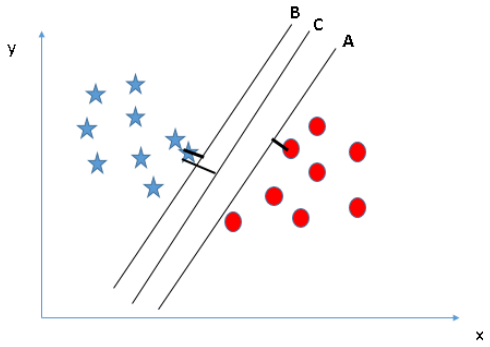

Fig 3.3 Hyperplane margins

Above, you can see that the margin for hyper-plane C is high as compared to both A and B. Hence, we name the right hyper-plane as C. Another lightning reason for selecting the hyper-plane with higher margin is robustness.

But In the case below, we can't have linear hyper-plane between the two classes. Till now, we have only looked at the linear hyper-plane.

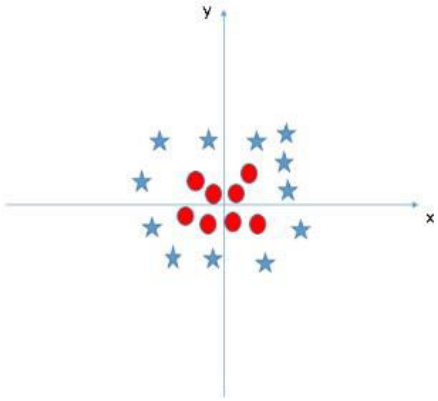

Fig 3.4: Encircling points

SVM can solve this problem. It solves this problem by introducing additional feature. Here, we will add a new feature z=x^2+y^2. Now, let's plot the data points on axis x and z:

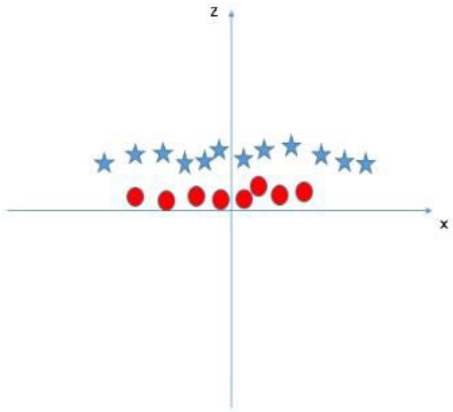

Fig 3.5: Feature engineered figure

In SVM, it is easy to have a linear hyper-plane between these two classes. But, another burning question which arises is, should we need to add this feature manually to have a hyper-plane. No, SVM has a technique called the kernel trick. These are functions which takes low dimensional input space and transform it to a higher dimensional space i.e. it converts not separable problem to separable problem, these functions are called kernels. It is mostly useful in non-linear separation problem.

Steps taken in SVM image classification are:

Fig 3.6: SVM image classification steps

A hyperplane is a line that splits the input variable space. In SVM, a hyperplane is selected to best separate the points in the input variable space by their class, either class 0 or class 1 as described above. In two-dimensions you can visualize this as a line and let's assume that all of our input points can be completely separated by this line. For example:

$$f(x_i, W, b) = Wx_i + b)$$

wherex_i is an image's pixel data flattened to a K × 1 vector, W is a C×K weight matrix, and b is a C×1 bias vector. The output of the function is a C × 1 vector of class scores, where C is the number of classes. As the score for a class is the weighted sum of an image's pixel values, we can interpret the linear classifier as how much an image matches the "label" for a class.

CNN

The most popular model for image classification is Convolutional Neural Networks (CNNs). CNN's are best known for their ability to recognize patterns present in images.CNN is where you feed the network images and the network classifies the data. CNNs tend to start with an input "scanner" which isn't intended to parse all the training data at once. For example, to input an image of 100 x 100 pixels, you wouldn't want a layer with 10,000 nodes. Rather, you create a scanning input layer of say 10 x 10 which you feed the first 10 x 10 pixels of the image. Once you passed that input, you feed it the next 10 x 10 pixels by moving the scanner one pixel to the right. This technique is known as sliding windows.

4.1 How CNN works

CNN's make use of filters (kernels) [23], to detect what features, such as edges, are present throughout an image. A filter is just a matrix of values, called weights, that are trained to detect specific features. The filter moves over each part of the image to check if the feature it is meant to detect is present. To provide a value representing how confident it is that a specific feature is present, the filter carries out a convolution operation, which is an element-wise product and sum between two matrices.

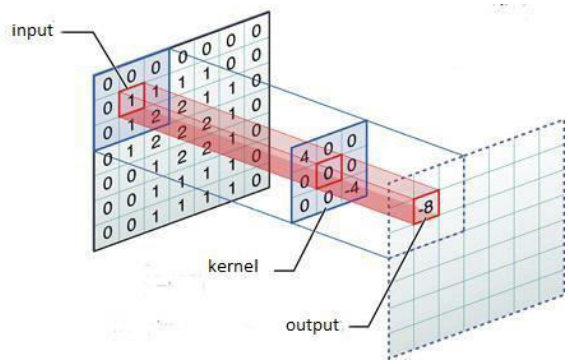

Fig 4.1: CNN filter

When the feature is present in part of an image, the convolution operation between the filter and that part of the image results in a real number with a high value. If the feature is not present, the resulting value is low. Let's understand this with an example.

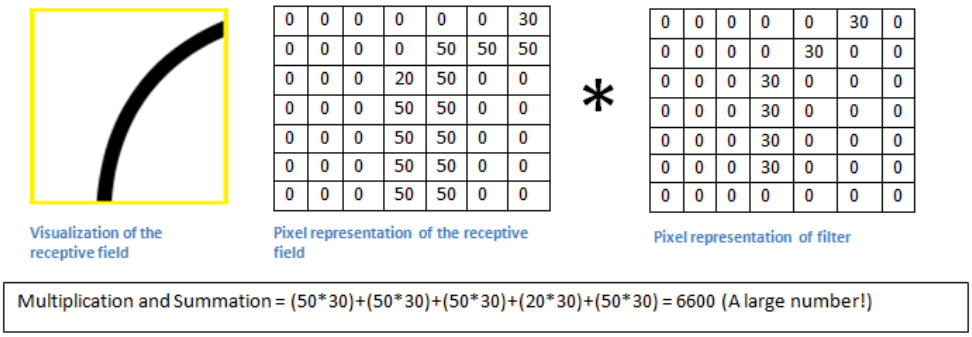

Fig 4.2: Multiplying matrices

In the above picture, a filter that is in charge of checking for right-hand curves is passed over a part of the image. Since that part of the image contains the same curve that the filter is looking for, the result of the convolution operation is a large number i.e. 6600.

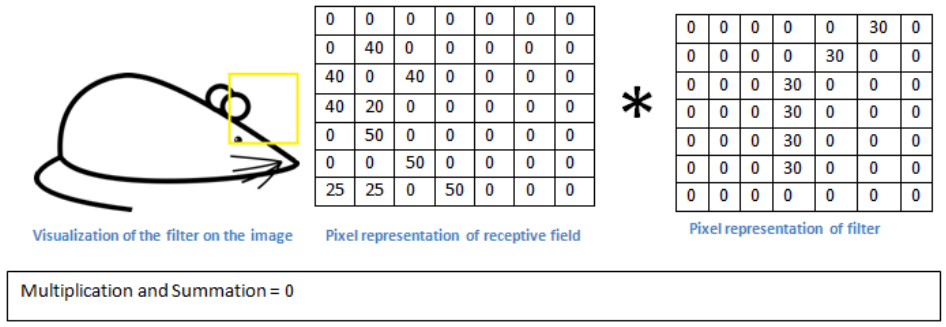

Fig 4.3: Summation after multiplication

While that same filter is passed over a part of the image with a considerably different set of edges, the convolution's output is small, meaning that there was no strong presence of a right hand curve.

The result of passing this filter over the entire image is an output matrix that stores the convolutions of this filter over various parts of the image. The filter must have the same number of channels as the input image so that the element-wise multiplication can

take place. For instance, if the input image contains three channels (like RGB), then the filter must contain three channels as well.

Additionally, a filter can be slid over the input image at varying intervals, using a stride value. The stride value dictates by how much the filter should move at each step. The output dimensions of a strided convolution can be calculated using the following equation:

$$n_{out} = floor\left(\frac{n_{in} - f}{s}\right) + 1$$

Where n_in = dimension of the input image

f = window size

s = stride

We do this so that the Convolutional Neural Network can learn the values for a filter that detect features present in the input data, the filter must be passed through a non-linear mapping. The output of the convolution operation between the filter and the input image is summed with a bias term and passed through a non-linear activation function. The purpose of the activation function is to introduce non-linearity into our network. Since our input data is non-linear. Now the activation function used here is Rectified Linear Unit (ReLU) activation function.

ReLU function is not relatively complex. As shown is diagram, values that are less than or equal to zero become zero and all positive values remain the same.

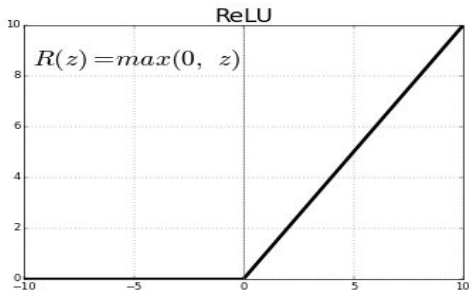

Fig 4.4: ReLU

After one or two convolutional layers, it is common to reduce the size of the representation produced by the convolutional layer. This reduction in the representation's size is known as **downsampling**.

To speed up the training process and reduce the amount of memory consumed by the network, we try to reduce the redundancy present in the input feature. There are a couple of ways we can downsample an image, but for this post, we will look at the most common one**: max pooling**.

In max pooling, a window passes over an image according to a set stride. At each step, the maximum value within the window is pooled into an output matrix, hence the name max pooling.

In the diagram, a window of size f=2 passes over an image with a stride of 2. f denotes the dimensions of the max pooling window (red box) and s denotes the number of units the window moves in the x and y-direction. At each step, the maximum value within the window is chosen.

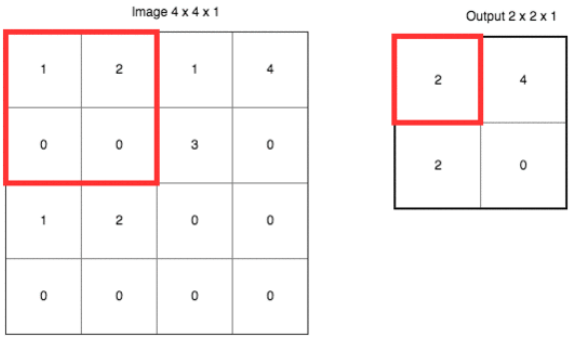

Fig 4.5: Max-pooling

Max pooling significantly reduces the representation size, in turn reducing the amount of memory required and the number of operations performed later in the network.

After multiple convolutional layers and downsampling operations, the image representation is converted into a feature vector that is passed into a Multi-Layer Perceptron, which simply is a neural network with at least three layers. This is referred to as a Fully-Connected Layer. In the fully-connected operation of a neural network, the input

representation is flattened into a feature vector and passed through a network of neurons to predict the output probabilities. The following image describes the flattening operation:

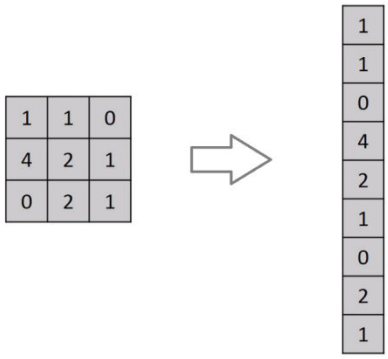

Fig 4.6: converting 2D to 1D matrix

The rows are concatenated to form a long feature vector The feature vector is then passed through multiple dense layers. At each dense layer, the feature vector is multiplied by the layer's weights, summed with its biases, and passed through a non-linearity.Following image visualizes the fully connected operation

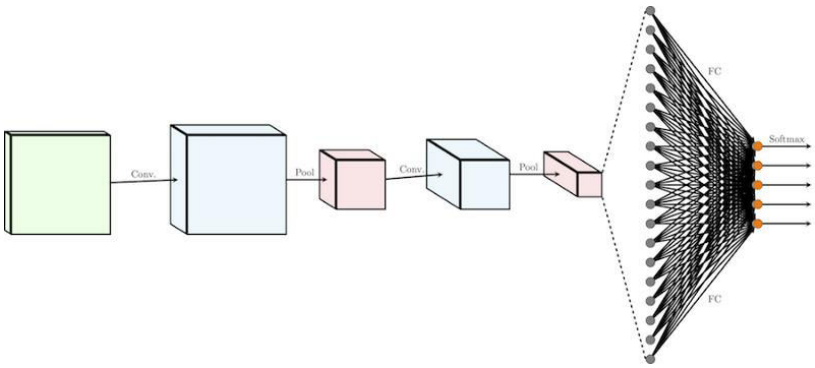

Fig 4.7: CNN model

The output layer of a CNN is for producing the probability of each class given the input image. To obtain these probabilities, we initialize the final Dense layer to contain the same number of neurons as there are classes. The output of this dense layer then passes through the Softmax activation function, which maps all the final dense layer outputs to a vector whose elements sum up to 1.

To measure how accurate our network was in predicting the handwritten digit from the input image, we make use of a loss function. The loss function assigns a real-valued number to define the model's accuracy when predicting the output digit. A common loss function to use when predicting multiple output classes is the Categorical Cross-Entropy Loss function, defined as follows:

$$H(y,\hat{y}) = \sum_{i}^{n} y_i log \frac{1}{\hat{y}} = -\sum_{i}^{n} y_i \log \hat{y}$$

$\hat{y}$ = CNN's prediction, y = desired output label

Deep convolutional neural networks (CNNs) became a revolution in computer vision by their ability to understand visual data, through the ability to learn "big models" with very large number of images. Producing massively annotated training data for new categories or tasks is unrealistic. Fortunately, when trained on a large enough, very diverse set of data (e.g., ImageNet), CNN features appear to transfer across a broad range of tasks. However, an open question is how to best adapt a pre-trained CNN for novel categories/tasks. Fine-tuning: Fine-tuning is by far the dominant strategy for transfer learning with neural networks. Also known as transfer learning.

4.2 Transfer learning

Transfer learning is the idea of overcoming the isolated learning paradigm and utilizing knowledge acquired for one task to solve related ones [27].

There is a stark difference between the traditional approach of building and training machine learning models, and using a methodology following transfer learning principles.

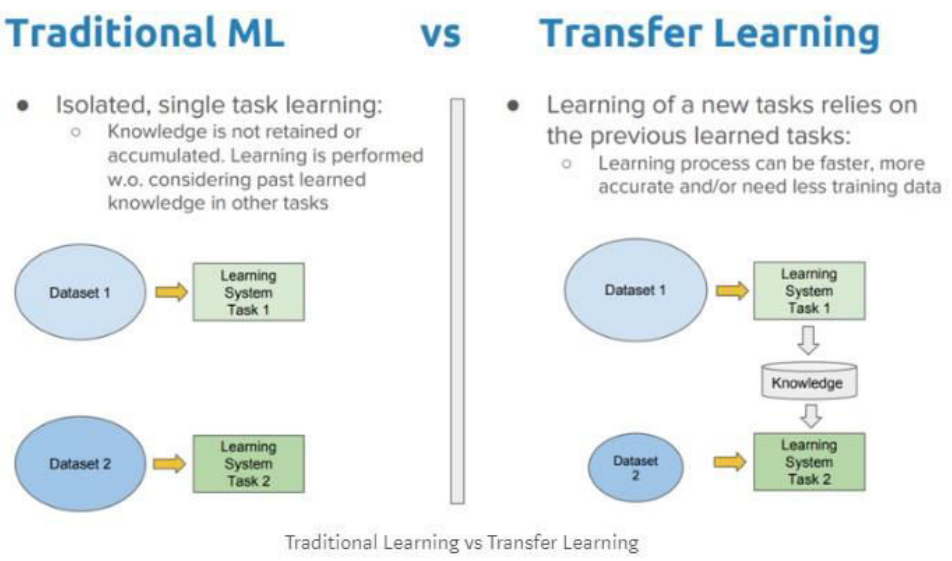

Fig 4.8: Transfer learning vs Traditional method

Thus, we can use pre-trained models to fine-tune/transfer learning. Pre-trained models are available for everyone to use through different means. The famous deep learning Python library, keras, provides an interface to download some popular models. You can also access pre-trained models from the web [29] since most of them have been open-sourced.

For computer vision, you can leverage some popular models:

- VGG 16
- ResNet 50

4.2.1 VGG16

The VGG-16 model is a 16-layer (convolution and fully connected) network built on the ImageNet database, which is built for the purpose of image recognition and classification. This model was built by Karen Simonyan and Andrew Zisserman.

The architecture of the VGG-16 model is depicted in the following figure.

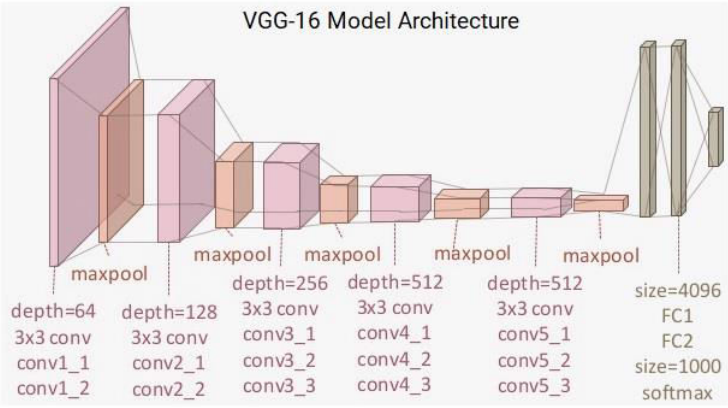

Fig 4.9: VGG16 model

You can clearly see that we have a total of 13 convolution layers using 3 x 3 convolution filters along with max pooling layers for downsampling and a total of two fully connected hidden layers of 4096 units in each layer followed by a dense layer of 1000 units, where each unit represents one of the image categories in the ImageNet database [30]

A stack of convolutional layers is followed by three Fully-Connected (FC) layers: the first two have 4096 channels each, the third performs 7-way ILSVRC classification and thus contains seven channels (one for each class). The final layer is the softmax layer. The configuration of the fully connected layers is the same in all networks. All hidden layers are equipped with the rectification (ReLU) nonlinearity.

To conclude, VGG-16 consists of 16 weight layers that include 13 convolutional layers with filter size of 3x3 and 3 fully-connected layers. The stride and padding of all convolutional layers are fixed to 1 pixel. All convolutional layers are divided into 5 groups and each group is followed by a max-pooling layer (Figure 1). Max-pooling is carried out over a 2x2 window with stride 2. The number of filters of convolutional layer group starts from 64 in the first group and then increases by a factor of 2 after each max-pooling layer, until it reaches 512.

4.2.2 ResNet50

ResNet50 is another current state of the art convolutional neural network architecture. It is similar in architecture to networks such as VGG-16 but with the additional identity mapping capability.

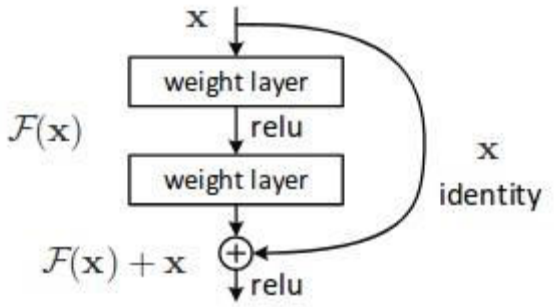

Fig 4.10: ResNet50

Rather than fitting the latent weights to predict the final emotion at each layer, ResNet models fit a residual mapping to predict the delta needed to reach the final prediction from one layer to the next. The identity mapping enables the model to bypass a typical CNN weight layer if the current layer is not necessary. This further helps the model to avoid overfitting to the training set. From an overall architecture and performance perspective, ResNet allows for much deeper networks while training much faster than other CNNs. In the case of ResNet50, there are 50 layers.

Experimental Setup

5.1 Data Overview

The Karolinska Directed Emotional Faces (KDEF) is a set of totally 4900 pictures of human facial expressions of emotion. The material was developed in 1998 by Daniel Lundqvist, Anders Flykt and Professor Arne Öhman at KarolinskaInstitutet, Department of Clinical Neuroscience, Section of Psychology, Stockholm, Sweden [8]. The set contains 70 individuals, each displaying 7 different emotional expressions, each expression being photographed (twice) from 5 different angles.

Data overview table:

Total images	2423
Train split	1213
Test split	1210
classes	7

Expressions to recognize are:

- Neutral

Fig 5.1: Neutral face

- Happy

Fig 5.2: Happy

- Angry

Fig 5.3: Angry face

- Afraid

Fig 5.4: Afraid face

- Disgust

Fig 5.5: Disgust face

- Sad

Fig 5.6: Sad face

- Surprised

Fig 5.7: Surprised face

Fig 5.1: Different facial expressions from our dataset

5.2 Hardware and Software Configuration

Hardware System Configuration:

Processor: Intel(R) Xeon (TM)

Number of cores: 6

Speed: CPU 3.40 GHz

RAM: 32GB

Hard Disk: 1000 GB

Software System Configuration:

Operating System: Windows 10 Pro

System Type: 64-bit Operating System

Processor: x64-based processor

Numerical Computing Environment: MATLAB, Python

Version: MATLAB R2017a, PYTHON 3.5.6

Presented Approach

Problem statement:

The proposed face recognition system has pre-processing, feature extraction and classification components. We are examining there different cases for feature extraction technique. Thus, seeing which gives the best results among handcrafted features. After which we fine tune DNN model to achieve better results.

Pre-processing:

All the images are provided in gray scale but every algorithm has different image size inputs. For handcrafted features image size was 200x200. For VGG16 the data needed to be resized into size 224x224 as it supports this specific size only. Similarly ResNet50 also needed same size as VGG16.

Method followed:

To achieve our goal of recognizing facial expression out of 7 expressions for each image we will first use a linear support vector machine (SVM) classifier baseline model. We first find local features of images with HOG, LBP, SIFT and then after getting feature vector of those images from HOG, LBP & SIFT we applied linear SVM classifier on those feature vectors.

Most image classification techniques nowadays are trained on ImageNet. Test images will be presented with no initial labels and algorithms will have to produce labels specifying the class to which the images belong [3]. So after SVM we will use some of the current state of the art architectures - VGG-16 and ResNet50. We chose to go with VGG-16 and ResNet50 because they won in the past the ImageNet challenge, achieved near state of the art results in terms of prediction accuracy, and follow a relatively standard CNN architecture as explained previously.

Results and Discussion

Result of KDEF Dataset:

	Accuracy
SVM (HOG)	61.9%
SVM (LBP)	40.4%
SVM (SIFT)	60.82%
VGG16	80.50%
ResNet50	85.52%

This study shows that fine tuning a pre-trained DNN model always gives better results compared to handcrafted features as they have to be made from the scratch while DNN models are already pre-trained on previous data.

The result for CNN models are apparently better than handcrafted features due to machine learning techniques being applied which makes use of multiple dense layers being trained on the training dataset. Even though the result for CNN models is good, but it is not yet exemplary. Our KDEF dataset is very challenging and its extremely difficult to identify a very small change in geometry on the same person's face. The slightest of facial expression changes can completely change to the result. Since, the model is being applied to detect multiple expressions on the same person, it becomes extremely difficult as compared to face recognition where we have different person's face and so geometry of the face widely differes.

So, even though face recognition gives a very high accuracy (over 99%), we still have 85% approximate accuracy in case of KDEF.

Conclusion and future Scope

We have done facial expression recognition with SVM using feature vector of HOG and LBP. We also explored the VGG-16 and ResNet50 architectures for recognizing facial emotions using deep learning. The results demonstrated that we were able to achieve acceptable results on KDEF dataset.

Future work that we would like to perform is to find another local feature extraction algorithm to test against. We would also like to farther optimization the implementation to gain additional performance advantages. We believe that we can farther improve accuracy using less time.

Also it has broad range in future applications. Like Facebook can recognize the expressions of users uploading their picture and recommend them counselling doctors or fun activity nearby if they are sad.

References

[1] https://en.wikipedia.org/wiki/Computer_vision

[2]https://medium.com/deep-dimension/an-analysis-on-computer-vision-problems-6c68d56030c3

[3]https://heartbeat.fritz.ai/the-5-computer-vision-techniques-that-will-change-how-you-see-the-world-1ee19334354b

[4]https://www.learnopencv.com/histogram-of-oriented-gradients/

[5] dataset: Lundqvist, D., Flykt, A., & Öhman, A. (1998). The Karolinska Directed Emotional Faces – KDEF, CD ROM from Department of Clinical Neuroscience, Psychology section, Karolinska Institutet, ISBN 91-630-7164-9.

[6] Pantic, Maja, and Leon JM Rothkrantz. "Automatic analysis of facial expressions: The state of the art." IEEE Transactions on Pattern Analysis & Machine Intelligence 12 (2000): 1424-1445.

[7]C. Shan, S. Gong, and P.W. McOwan, “Robust Facial Expression Recognition Using Local Binary Patterns,” Proc. IEEE Int’l Conf. Image Processing, pp. 914-917, 2005.

[8] http://www.emotionlab.se/kdef/

[9] Lundqvist, D., Flykt, A., & Öhman, A. (1998). The Karolinska Directed Emotional Faces – KDEF, CD ROM from Department of Clinical Neuroscience, Psychology section, Karolinska Institutet, ISBN 91-630-7164-9.

[10] Yanai, Keiji, and Yoshiyuki Kawano. "Food image recognition using deep convolutional network with pre-training and fine-tuning." 2015 IEEE International Conference on Multimedia & Expo Workshops (ICMEW). IEEE, 2015

[11]Too, Edna Chebet, et al. "A comparative study of fine-tuning deep learning models for plant disease identification." Computers and Electronics in Agriculture (2018).

[12] Yang, Biao, et al. "Facial expression recognition using weighted mixture deep neural network based on double-channel facial images." IEEE Access 6 (2018): 4630-4640.

[13] Raghuvanshi, Arushi, and Vivek Choksi. "Facial Expression Recognition with Convolutional Neural Networks." CS231n Course Projects (2016).

[14]https://www.google.com/search?biw=1366&bih=625&tbm=isch&sa=1&ei=ldrAXPOxAcG3ggeXvYVY&q=object+detection+&oq=object+detection+&gs_l=img.3...57665.57665..57786...0.0..0.0.0.....1....1..gws-wiz-img.OXRHEbczJzc#imgrc=yUOBxg-ydY8zuM:

[15]https://www.google.com/search?biw=1366&bih=576&tbm=isch&sa=1&ei=0NrAXPj1OKTn_QbTlonQBQ&q=face+detection+images&oq=face+detection&gs_l=img.1.1.0i67j0j0i67j0l7.521674.522559..524545...0.0..0.518.1716.3-2j1j1.....1....1..gws-wiz-img.....0i7i30.XWhyyN6CI3s#imgrc=auuy1LfLZU8LuM:

[16] Ouyang, Wanli, et al. "Factors in finetuning deep model for object detection with long-tail distribution." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[17] Hassannejad, Hamid, et al. "Food image recognition using very deep convolutional networks." Proceedings of the 2nd International Workshop on Multimedia Assisted Dietary Management. ACM, 2016.

[18] Kieffer, Brady, et al. "Convolutional neural networks for histopathology image classification: Training vs. using pre-trained networks." 2017 Seventh International Conference on Image Processing Theory, Tools and Applications (IPTA). IEEE, 2017.

[19] Xia, Xiaoling, Cui Xu, and Bing Nan. "Inception-v3 for flower classification." 2017 2nd International Conference on Image, Vision and Computing (ICIVC). IEEE, 2017.

[20] Akiba, Takuya, Shuji Suzuki, and Keisuke Fukuda. "Extremely large minibatch SGD: training resnet-50 on imagenet in 15 minutes." arXiv preprint arXiv:1711.04325 (2017).

[22] Girshick, Ross. "Fast r-cnn." Proceedings of the IEEE international conference on computer vision. 2015.

[23] https://towardsdatascience.com/convolutional-neural-networks-from-the-ground-up-c67bb41454e1

[24] https://towardsdatascience.com/face-recognition-how-lbph-works-90ec258c3d6b

[25] https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/

[26] Azhar, Ryfial, et al. "Batik image classification using sift feature extraction, bag of features and support vector machine." Procedia Computer Science 72 (2015): 24-30.

[27]https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a

[28] http://aishack.in/tutorials/sift-scale-invariant-feature-transform-introduction/

[29] https://keras.io/applications/

[30] Savoiu, Alexandru, and James Wong. "Recognizing Facial Expressions Using Deep Learning."