MAULANA AZAD NATIONAL INSTITUTE OF TECHNOLOGY

(A Deemed University)

Bhopal-462003

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MINOR PROJECT REPORT ON

"PERIOCULAR BIOMETRICS"

SUBMITTED IN PARTIAL FULFILLMENT FOR THE DEGREE OF

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Presented by :

Aman Jain

Kasis Kumar Shrestha

Mausam Jain

Kunal Gaurhar

Guided by :

Prof. Sweta Jain

SESSION: 2017-18

MAULANA AZAD NATIONAL INSTITUTE OF TECHNOLOGY

BHOPAL-462003

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We hereby declare that the work which is being presented in the project report titled "**Periocular Biometrics** " in partial fulfilment of the requirements of the degree of technology in computer science & engineering is an authentic record of our own work carried out under the guidance of **Prof. Sweta Jain .** The work has been carried out in **Maulana Azad National Institute of Technology, Bhopal** and has not been submitted for any other degree of diploma. This is to certify that the above declaration is correct to the best of our knowledge.

Aman Jain
151112211

Kasis Kumar Shrestha
151112204

Mausam jain
151112309

Kunal Gaurhar
151112236

MAULANA AZAD NATIONAL INSTITUTE OF TECHNOLOGY

Bhopal-462003

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that **Aman Jain , Kasis Kumar Shrestha , Mausam Jain and Kunal Gaurhar** students of third year, Bachelor Of Technology (COMPUTER SCIENCE AND ENGINEERING) have successfully completed their minor project on **Periocular Biometrics** in the partial fulfilment of the requirement of their Bachelor Of Technology in Computer Science and Engineering.

Guide :

Prof. Sweta Jain

Project Co-Ordinator :

Dr. Rajesh Wadhvani

ACKNOWLEDGEMENT

With due respect, we express our deep sense of gratitude to our respected mentor **Prof. Sweta Jain** for her valuable support. We are thankful for the encouragement she has given to us in completing this project. Needless to mention the help and support of our respected **HOD Dr**. **Meenu Chawla** and respected professors in allowing us to use the departmental labs and other services.

We are also grateful to our respected director **Dr. Narendra Singh Raghuwanshi** for permitting us to utilize all the necessary facilities of the college. We are also thankful to other staff members of our department for their kind co-operation and help. Last but not the least, we would like to express our deep appreciation to our family members and classmates for providing the much needed support and encouragement.

NAME :	**SCHOLAR NO.** :	**SIGNATURE :**
Aman Jain	151112211	
Kasis Kumar Shrestha	151112204	
Mausam Jain	151112309	
Kunal Gaurhar	151112236	

CONTENTS

ABSTRACT

The periocular region has recently emerged as a promising trait for unconstrained biometric recognition, specially on cases where neither the iris and a full facial picture can be obtained. Previous studies concluded that the regions in the vicinity of the human eye - the periocular region- have surprisingly high discriminating ability between individuals, are relatively permanent and easily acquired at large distances. Hence, growing attention has been paid to periocular recognition methods, on the performance levels they are able to achieve, and on the correlation of the responses given by other.

Periocular recognition has been an active research area in the last few years. In spite of the advancements made in this area, cross-spectral matching and recognition have largely remained unexplored. The objective of this project is to develop advanced algorithms for periocular recognition and to improve the state-of-the-art techniques in this area. Extracting soft biometric information such as gender from periocular images will also be explored in this project. As part of the project, a database of periocular images will be collected for performance calculation.

INTRODUCTION

Biometrics is the science of establishing human identity based on the physical or behavioral traits of an individual . Several biometric traits such as face, iris, hand geometry, and fingerprint have been extensively studied in the literature and have been incorporated in both government and civilian identity management applications. Recent research in biometrics has explored the use of other human characteristics such as gait , conjunctival vasculature , knuckle joints , etc., as supplementary biometric evidence to enhance the performance of classical biometric systems.

Periocular biometrics has been shown as one of the most discriminative regions of the face, gaining attention as an independent method for recognition or a complement to face and iris modalities under non-ideal conditions . The typical elements of the periocular region are labeled in Fig. 1 , left. This region can be acquired largely relaxing the conditions, in contraposition to the more carefully controlled conditions usually needed in face or iris modalities, making it suitable for unconstrained and uncooperative scenarios. Another advantage is that the problem of iris segmentation is automatically avoided, which can be an issue in difficult images .

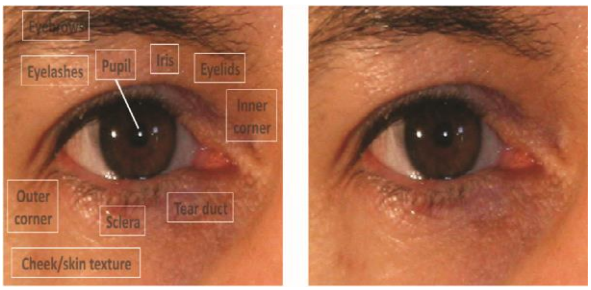


Fig .1(left) Fig.1(right)

Biometrics refers to the identification of individuals based on their characteristics. These systems are based on physiological or behavioral characteristics of human beings. A lot of research efforts have been focused on advancing the field of

biometrics due to its high impact on practical security applications. In the last few decades, biometrics has gained a lot of popularity and is widely deployed to minimize thefts and other crimes

Periocular region corresponds to the region of the eye as well as the area surrounding it. This has several potential applications including crime scene investigation by the law enforcement agencies, where offenders use a mask or other face covering devices to cover their face leaving only the periocular region visible . A number of research works are going on in the field of heterogeneous biometrics which includes face and iris traits.

The proposed periocular recognition process consists of a sequence of operations: image alignment (for the global matcher described in the next section), feature extraction, and matching. We adopted two different approaches to the problem: one based on global information and the other based on local information. The two approaches use different methods for feature extraction and matching.

LITERATURE REVIEW

1 Related Work

The work of the image recognition and processing has been going from long back since 1900's. It has been emerged through various projects and technologies. Many companies like google, Facebook believes in the better world security through the use of these biometrics. The field of the image processing in vast in itself that it contains the various sector like image segmentation, improvement of the image etc.

For a long time most of the government organizations, public and private companies were looking for a highly responsible security system and attendance maintenance organism without significantly increasing their budget for these. To meet their increasing demand, biometric system appeared to the scene. Finding its super sensitive efficiency in preventing identity theft and maintaining attendance records, everybody greeted it and installed it at their premises. It received high appreciation from various sectors.

Not only did it meet their expectations, but also exceeded their expectation in a great extent. Since then till date, various types of biometric identification and verification systems have come to serve more than meeting these two purposes. Keeping pace with the complex needs of the day, it has become cleverer, more complex and supersensitive. Now it can take a significant role in successful financial transactions.
As the name suggests, biometrics works on biological characteristics of a human being. It measures and verifies biological characteristics of an individual. According to the methods used in it, biometrics is of two types. One type that works on verification method is called behavioral biometrics. Another type is called physical biometrics. It works on both identification and verification method. Former type has extensive uses in offices, business

centers and various educational institutions. Later type is used by banks and other investigative departments for secure transaction and criminal investigation.

Behavioral Biometrics is again divided into two types: **Keystroke Recognition** and **Speaker Identification**.

It measures the characteristics of a person's typing patterns. This is used for identifying the persona's creating inappropriate email and unethical typing. It is highly useful for reducing fraudulent activity on the Internet. Keystroke recognition software within a computer recognizes it and sends a message about it to the person taking care of surveillance system.

Speaker Identification

Speaker identification and recognition is used to identify an unknown speaker's identity. It verifies and tallies a person's voice and its patterns, pitch and speech style. . Behavioral pattern, intensity and pitch of a voice vary from man to man.

Physical Biometrics Systems may be of various types as given below:

Fingerprint identification

Fingerprint identification is the oldest and cheapest method used widely. It works on a same principle that says fingerprints of different individuals are not the same. It identifies and verifies an individual's fingerprint with the data stored in it.

Palm Vein Authentication

Hand geometry is verisimilitude to fingerprint analysis. It takes and analyses palm prints and other physical characteristics to uniquely identify an individual's palm. It measures and compares the different physical characteristics of various human hands.

Retina Scan

It analyzes and identifies unique blood vessel patterns of an individual. It works on the capillary blood vessels existing in the back of human eye. It throws a low-intensity light to a person's eye to get an image pattern formed by the blood vessels.

Face Recognition

It identifies and analyses the face of a person. It uses an analyzing method that easily distinguishes the difference between the two faces of two different persons

A timeline of Image Recognition

Having become more familiar with the concept of Image Recognition, let us get into the detail of how this specific technology emerged and evolved. Here below, we drew up a brief timeline of the main stages it went through:

2001 : Paul Viola and Michael Jones invented a simultaneous face detection algorithm allowing for human figures to be identified through their facial traits.

2005 : Navneet Dalal and Bill Triggs published Histograms of Oriented Gradients (HOG), theorizing a feature detector for the recognition of pedestrians in security system circuits.

2012: Alex Krizhevsky, Ilya Sutskever and Geoffrey Hinton hit it big with a new object recognition algorithm ensuring an 85% level of accuracy.

2015: the Convolutional Neural Network (CNN) developed IR tools whose level of accuracy in facial recognition exceeded 95%.

Today: Google, Amazon and even some car manufacturers are channeling their efforts as well as their R&D investments into the development of new technologies that integrate

ImageRecognition.

Amazon, for instance, recently launched Amazon Recognition, a new product based on Image Recognition functions and able to scan photos, guess at emotions through face recognition and classify objects or animals.

1.1 Classification Of various algorithms for image classifications :

The image classification pipeline. We've seen that the task in Image Classification is to take an array of pixels that represents a single image and assign a label to it. Our complete pipeline can be formalized as follows:

Input: Our input consists of a set of N images, each labeled with one of K different classes. We refer to this data as the training set.

Learning: Our task is to use the training set to learn what every one of the classes looks like. We refer to this step as training a classifier, or learning a model.

Evaluation: In the end, we evaluate the quality of the classifier by asking it to predict labels for a new set of images that it has never seen before. We will then compare the true labels of these images to the ones predicted by the classifier. Intuitively, we're hoping that a lot of the predictions match up with the true answers (which we call the ground truth).

1.1.1 Nearest Neighbor :

This classifier has nothing to do with Convolutional Neural Networks and it is very rarely used in practice, but it will allow us to get an idea about the basic approach to an image classification problem.

Example image classification dataset: CIFAR-10. One popular toy image classification dataset is the CIFAR-10 dataset. This dataset consists of 60,000 tiny images that are 32 pixels high and wide. Each image is labeled with one of 10 classes (for example "airplane, automobile, bird, etc"). These 60,000 images are partitioned into a training set of 50,000

images and a test set of 10,000 images. In the image below you can see 10 random example images from each one of the 10 classes:

For example given the CIFAR-10 training set of 50,000 images (5,000 images for every one of the labels), and we wish to label the remaining 10,000. The nearest neighbor classifier will take a test image, compare it to every single one of the training images, and predict the label of the closest training image.

The images are taken in the form of just two blocks of 32 x 32 x 3. One of the simplest possibilities is to compare the images pixel by pixel and add up all the differences.

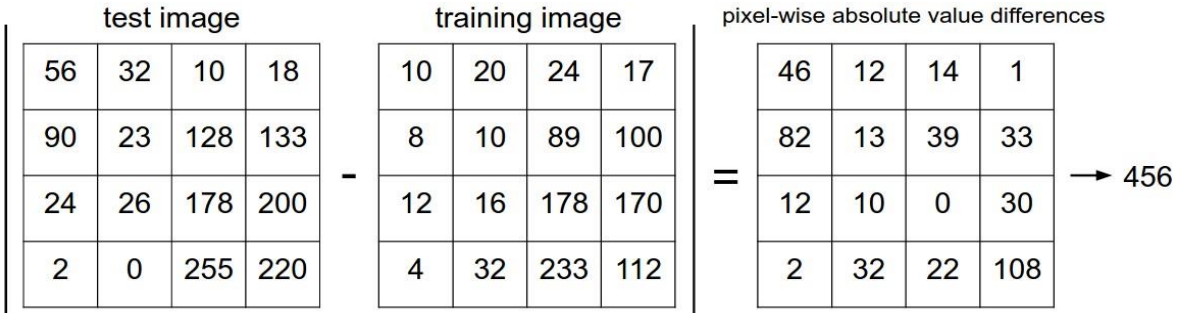


Fig.[2] Ref.[1]

we have a train(X,y) function that takes the data and the labels to learn from. Internally, the class should build some kind of model of the labels and how they can be predicted from the data. And then there is a predict(X) function, which takes new data and predicts the labels.

The choice of distance. There are many other ways of computing distances between vectors. Another common choice could be to instead use the L2 distance, which has the geometric interpretation of computing the euclidean distance between two vectors. The distance takes the form:

If we run the Nearest Neighbor classifier on CIFAR-10 with this distance, we would obtain 35.4% accuracy (slightly lower than our L1 distance result).

L1 vs. L2. It is interesting to consider differences between the two metrics. In particular, the L2 distance is much more unforgiving than the L1 distance when it comes to differences between two vectors. That is, the L2 distance prefers many medium disagreements to one big one.

1.1.2 K-nearest neighbor classifier:

We have noticed that it is strange to only use the label of the nearest image when we wish to make a prediction. Indeed, it is almost always the case that one can do better by using what's called a **k-Nearest Neighbor Classifier**. The idea is very simple: instead of finding the single closest image in the training set, we will find the top **k** closest images, and have them vote on the label of the test image. In particular, when *k = 1*, we recover the Nearest Neighbor classifier. Intuitively, higher values of **k** have a smoothing effect that makes the classifier more resistant to outliers:

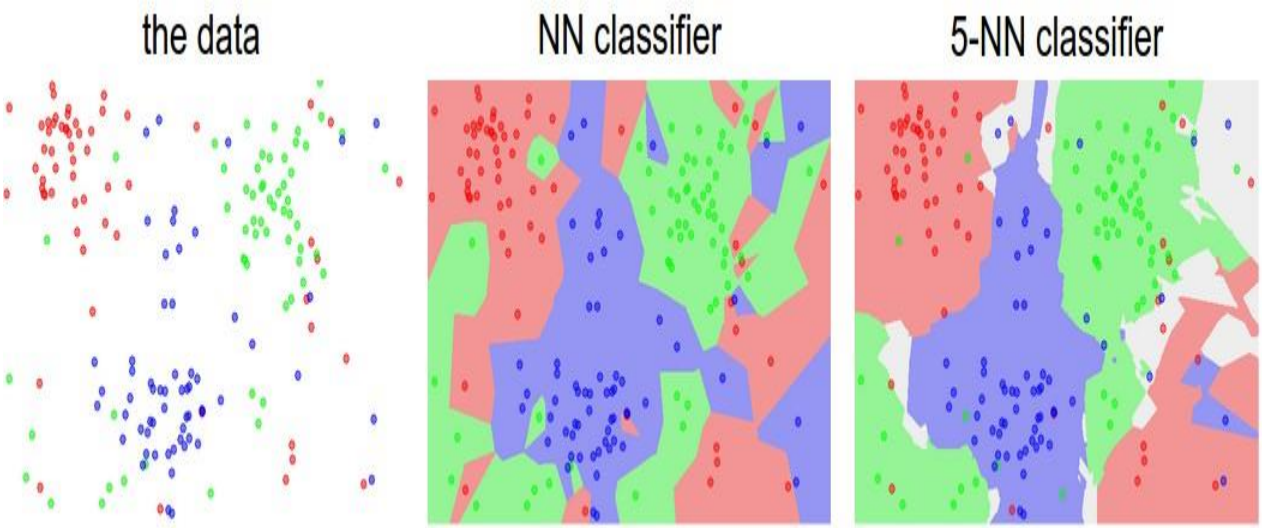


Fig.3 [1]

An example of the difference between Nearest Neighbor and a 5-Nearest Neighbor classifier, using 2-dimensional points and 3 classes (red, blue, green). The colored regions show the decision boundaries induced by the classifier with an L2 distance. The white

regions show points that are ambiguously classified (i.e. class votes are tied for at least two classes). Notice that in the case of a NN classifier, outlier datapoints (e.g. green point in the middle of a cloud of blue points) create small islands of likely incorrect predictions, while the 5-NN classifier smooths over these irregularities, likely leading to better generalization on the test data (not shown). Also note that the gray regions in the 5-NN image are caused by ties in the votes among the nearest neighbors (e.g. 2 neighbors are red, next two neighbors are blue, last neighbor is green).In practice, you will almost always want to use k-Nearest Neighbor. But what value of *k* should you use? We turn to this problem next.

Fig.4 [1]

Pros and Cons of Nearest Neighbor classifier.

It is worth considering some advantages and drawbacks of the Nearest Neighbor classifier. Clearly, one advantage is that it is very simple to implement and understand. Additionally, the classifier takes no time to train, since all that is required is to store and possibly index the training data. However, we pay that computational cost at test time, since classifying a test example requires a comparison to every single training example. This is backwards, since in practice we often care about the test time efficiency much more than the efficiency at training time. In fact, the deep neural networks we will develop later in this class shift this tradeoff to the other extreme: They are very expensive to train, but once the training is finished it is very cheap to classify a new test example. This mode of operation is much more desirable in practice.

1.1.3 Linear Classification

Linear classifier. In this module we will start out with arguably the simplest possible function, a linear mapping:

$$f(xi,W,b)=Wxi+bf(xi,W,b)=Wxi+b$$

In the above equation, we are assuming that the image xixi has all of its pixels flattened out to a single column vector of shape [D x 1]. The matrix W (of size [K x D]), and the vector b (of size [K x 1]) are the parameters of the function. In CIFAR-10, xixi contains all pixels in the i-th image flattened into a single [3072 x 1] column, W is [10 x 3072] and b is [10 x 1], so 3072 numbers come into the function (the raw pixel values) and 10 numbers come out (the class scores). The parameters in W are often called the weights, and b is called the bias vector because it influences the output scores, but without interacting with the actual data xi.

We notice that a linear classifier computes the score of a class as a weighted sum of all of its pixel values across all 3 of its color channels. Depending on precisely what values we set for these weights, the function has the capacity to like or dislike (depending on the sign of each weight) certain colors at certain positions in the image. For instance, you can imagine that the "ship" class might be more likely if there is a lot of blue on the sides of an image (which could likely correspond to water). You might expect that the "ship" classifier would then have a lot of positive weights across its blue channel weights (presence of blue increases score of ship), and negative weights in the red/green channels (presence of red/green decreases the score of ship).

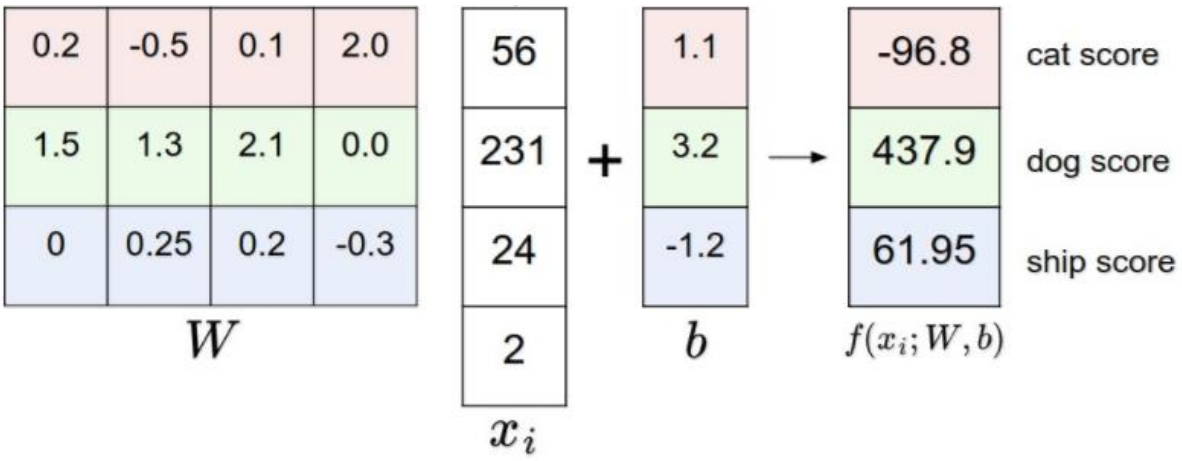


Fig.5 [1]

An example of mapping an image to class scores. For the sake of visualization, we assume the image only has 4 pixels (4 monochrome pixels, we are not considering color channels in this example for brevity), and that we have 3 classes (red (cat), green (dog), blue (ship) class). (Clarification: in particular, the colors here simply indicate 3 classes and are not related to the RGB channels.) We stretch the image pixels into a column and perform matrix multiplication to get the scores for each class. Note that this particular set of weights W is not good at all: the weights assign our cat image a very low cat score. In particular, this set of weights seems convinced that it's looking at a dog.

Analogy of images as high-dimensional points. Since the images are stretched into high-dimensional column vectors, we can interpret each image as a single point in this space (e.g. each image in CIFAR-10 is a point in 3072-dimensional space of 32x32x3 pixels). Analogously, the entire dataset is a (labeled) set of points.

Since we defined the score of each class as a weighted sum of all image pixels, each class score is a linear function over this space. We cannot visualize 3072-dimensional spaces, but if we imagine squashing all those dimensions into only two dimensions, then we can try to visualize what the classifier might be doing:

As we saw above, every row of WW is a classifier for one of the classes. The geometric interpretation of these numbers is that as we change one of the rows of WW, the

corresponding line in the pixel space will rotate in different directions. The biases bb, on the other hand, allow our classifiers to translate the lines. In particular, note that without the bias terms, plugging in xi=0xi=0 would always give score of zero regardless of the weights, so all lines would be forced to cross the origin.

Bias trick. Before moving on we want to mention a common simplifying trick to representing the two parameters W,bW,b as one. Recall that we defined the score function as:

f(xi,W,b)=Wxi+bf(xi,W,b)=Wxi+b

As we proceed through the material it is a little cumbersome to keep track of two sets of parameters (the biases bband weights WW) separately. A commonly used trick is to combine the two sets of parameters into a single matrix that holds both of them by extending the vector xixi with one additional dimension that always holds the constant 11- a default bias dimension. With the extra dimension, the new score function will simplify to a single matrix multiply:

f(xi,W)=Wxi

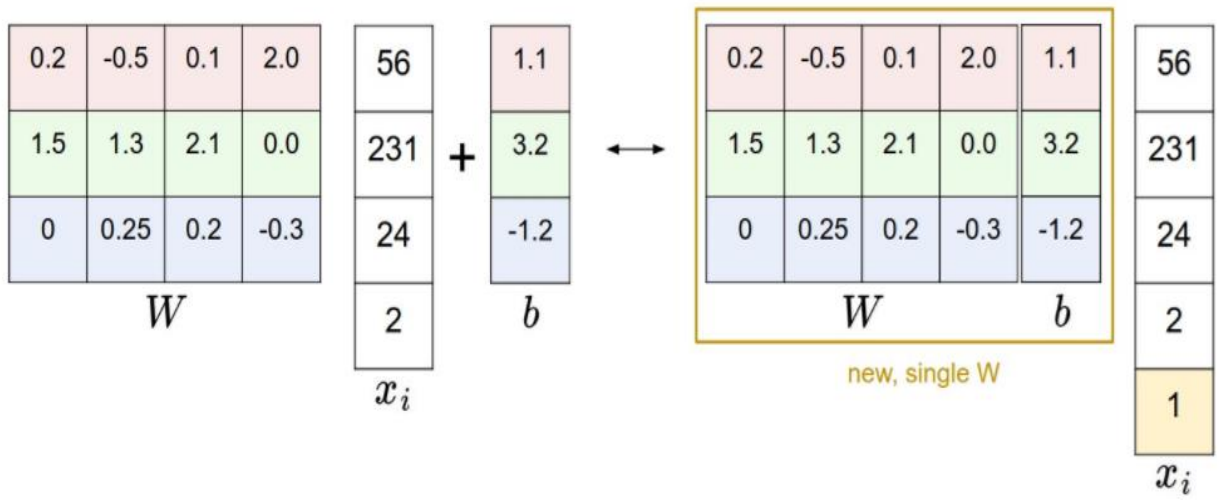


Fig.6 [1]

1.2 Convolutional Neural Network :

Convolutional Neural Networks are very similar to ordinary Neural Networks that are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity. The whole network still expresses a single differentiable score function: from the raw image pixels on one end to class scores at the other. And they still have a loss function (e.g. SVM/Softmax) on the last (fully-connected) layer and all the tips/tricks we developed for learning regular Neural Networks still apply.

ConvNet architectures make the explicit assumption that the inputs are images, which allows us to encode certain properties into the architecture. These then make the forward function more efficient to implement and vastly reduce the amount of parameters in the network.

Architecture Overview:

Recall: Regular Neural Nets. As we saw in the previous chapter, Neural Networks receive an input (a single vector), and transform it through a series of *hidden layers*. Each hidden layer is made up of a set of neurons, where each neuron is fully connected to all neurons in the previous layer, and where neurons in a single layer function completely independently and do not share any connections. The last fully-connected layer is called the "output layer" and in classification settings it represents the class scores.

Regular Neural Nets don't scale well to full images. In CIFAR-10, images are only of size 32x32x3 (32 wide, 32 high, 3 color channels), so a single fully-connected neuron in a first hidden layer of a regular Neural Network would have 32*32*3 = 3072 weights. This amount still seems manageable, but clearly this fully-connected structure does not scale to larger images. For example, an image of more respectable size, e.g. 200x200x3, would lead to neurons that have 200*200*3 = 120,000 weights. Moreover, we would almost certainly

want to have several such neurons, so the parameters would add up quickly! Clearly, this full connectivity is wasteful and the huge number of parameters would quickly lead to overfitting.

3D volumes of neurons.

Convolutional Neural Networks take advantage of the fact that the input consists of images and they constrain the architecture in a more sensible way. In particular, unlike a regular Neural Network, the layers of a ConvNet have neurons arranged in 3 dimensions: **width, height, depth**. (Note that the word *depth*here refers to the third dimension of an activation volume, not to the depth of a full Neural Network, which can refer to the total number of layers in a network.) For example, the input images in CIFAR-10 are an input volume of activations, and the volume has dimensions 32x32x3 (width, height, depth respectively). As we will soon see, the neurons in a layer will only be connected to a small region of the layer before it, instead of all of the neurons in a fully-connected manner. Moreover, the final output layer would for CIFAR-10 have dimensions 1x1x10, because by the end of the ConvNet architecture we will reduce the full image into a single vector of class scores, arranged along the depth dimension. Here is a visualization:

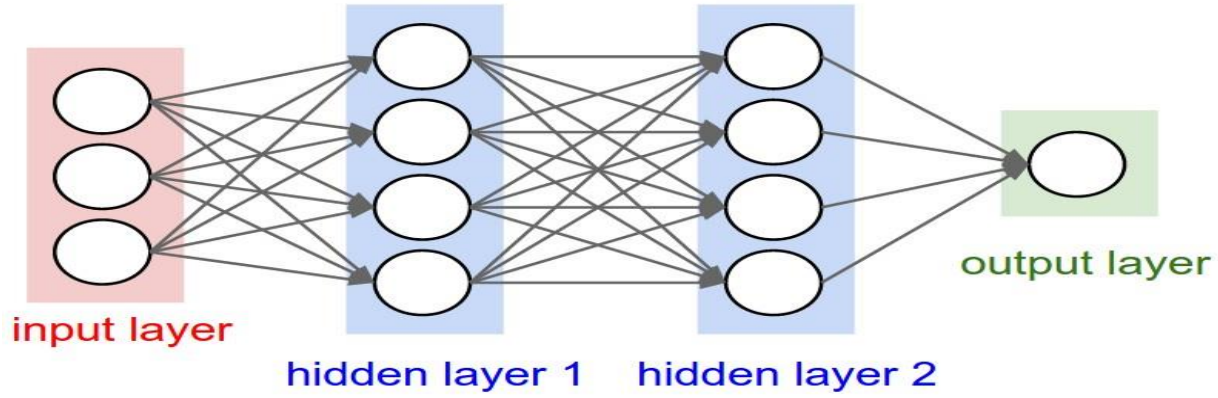


Fig.7 [2]

Left: A regular 3-layer Neural Network. Right: A ConvNet arranges its neurons in three dimensions (width, height, depth), as visualized in one of the layers. Every layer of a ConvNet transforms the 3D input volume to a 3D output volume of neuron activations. In

this example, the red input layer holds the image, so its width and height would be the dimensions of the image, and the depth would be 3 (Red, Green, Blue channels).

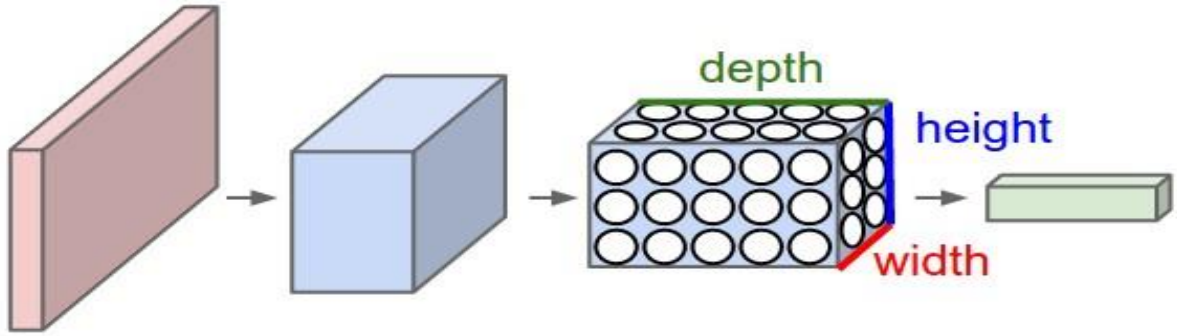


Fig.8 [2]

A ConvNet is made up of Layers. Every Layer has a simple API: It transforms an input 3D volume to an output 3D volume with some differentiable function that may or may not have parameters.

1.2.1 Layers used to build ConvNets

As we described above, a simple ConvNet is a sequence of layers, and every layer of a ConvNet transforms one volume of activations to another through a differentiable function. We use three main types of layers to build ConvNet architectures: **Convolutional Layer**, **Pooling Layer**, and **Fully-Connected Layer** (exactly as seen in regular Neural Networks). We will stack these layers to form a full ConvNet **architecture**.

Example Architecture: Overview. We will go into more details below, but a simple ConvNet for CIFAR-10 classification could have the architecture [INPUT - CONV - RELU - POOL - FC]. In more detail:

INPUT [32x32x3] will hold the raw pixel values of the image, in this case an image of width 32, height 32, and with three color channels R,G,B.

CONV layer will compute the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are

connected to in the input volume. This may result in volume such as [32x32x12] if we decided to use 12 filters.

RELU layer will apply an elementwise activation function, such as the max(0,x)max(0,x) thresholding at zero. This leaves the size of the volume unchanged ([32x32x12]).

POOL layer will perform a downsampling operation along the spatial dimensions (width, height), resulting in volume such as [16x16x12].

FC (i.e. fully-connected) layer will compute the class scores, resulting in volume of size [1x1x10], where each of the 10 numbers correspond to a class score, such as among the 10 categories of CIFAR-10. As with ordinary Neural Networks and as the name implies, each neuron in this layer will be connected to all the numbers in the previous volume.

In this way, ConvNets transform the original image layer by layer from the original pixel values to the final class scores. Note that some layers contain parameters and other don't. In particular, the CONV/FC layers perform transformations that are a function of not only the activations in the input volume, but also of the parameters (the weights and biases of the neurons). On the other hand, the RELU/POOL layers will implement a fixed function. The parameters in the CONV/FC layers will be trained with gradient descent so that the class scores that the ConvNet computes are consistent with the labels in the training set for each image.

1.2.2 Backpropagation :

Backpropagation is a common method for training a neural network. The goal of backpropagation is to optimize the weights so that the neural network can learn how to correctly map arbitrary inputs to outputs.

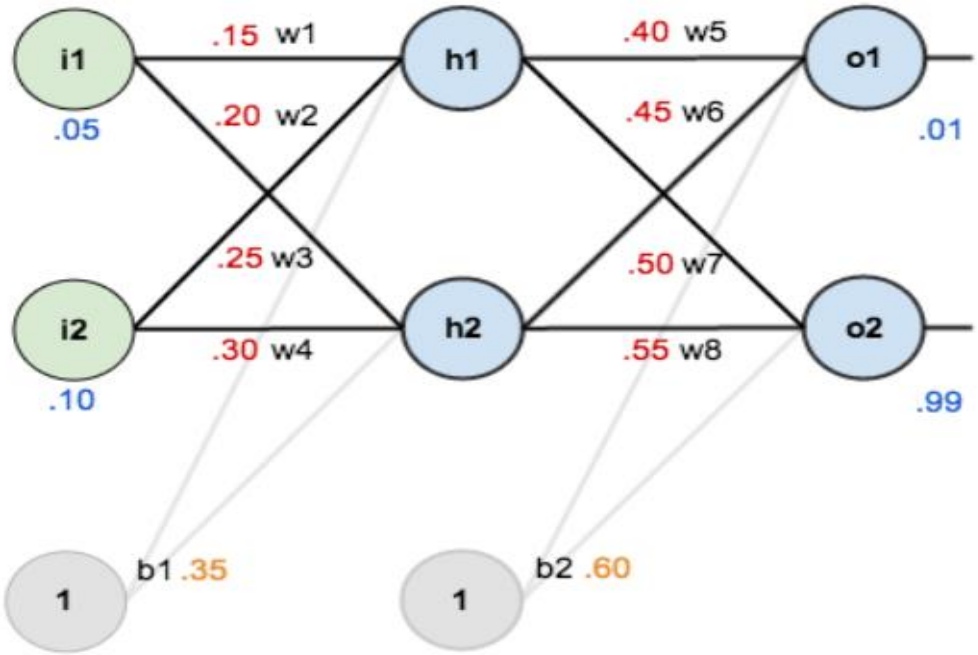


Fig.9 [3]

Forward pass :

The total net input to each hidden layer neuron, squash the total net input using an activation function (here we use the logistic function), then repeat the process with the output layer neurons.

$$net_{h1} = w_1 * i_1 + w_2 * i_2 + b_1 * 1$$

$$out_{h1} = \frac{1}{1+e^{-net_{h1}}} :$$

$$net_{o1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$$

$$out_{o1} = \frac{1}{1+e^{-net_{o1}}} :$$

Calculating the Total Error

We can now calculate the error for each output neuron using the squared error function and sum them to get the total error:

$$E_{total} = \sum \frac{1}{2}(target - output)^2$$

The Backwards Pass

Our goal with backpropagation is to update each of the weights in the network so that they cause the actual output to be closer the target output, thereby minimizing the error for each output neuron and the network as a whole

Output Layer

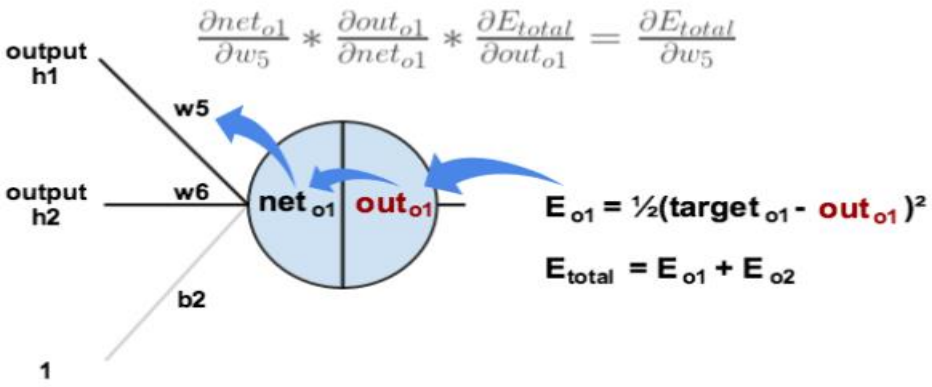


Fig.[10] Ref.[3]

Hidden Layer

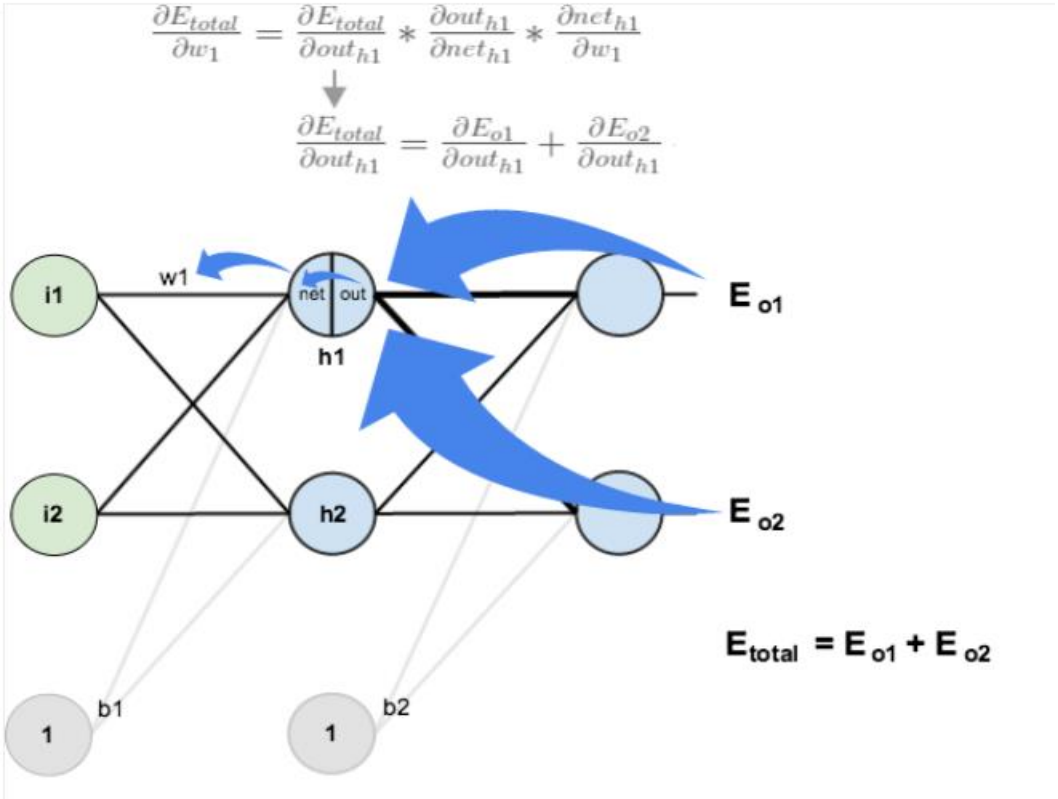


Fig.[11] Ref.[3]

1.2.3 General pooling.

In addition to max pooling, the pooling units can also perform other functions, such as *average pooling* or even *L2-norm pooling*. Average pooling was often used historically but has recently fallen out of favor compared to the max pooling operation, which has been shown to work better in practice.

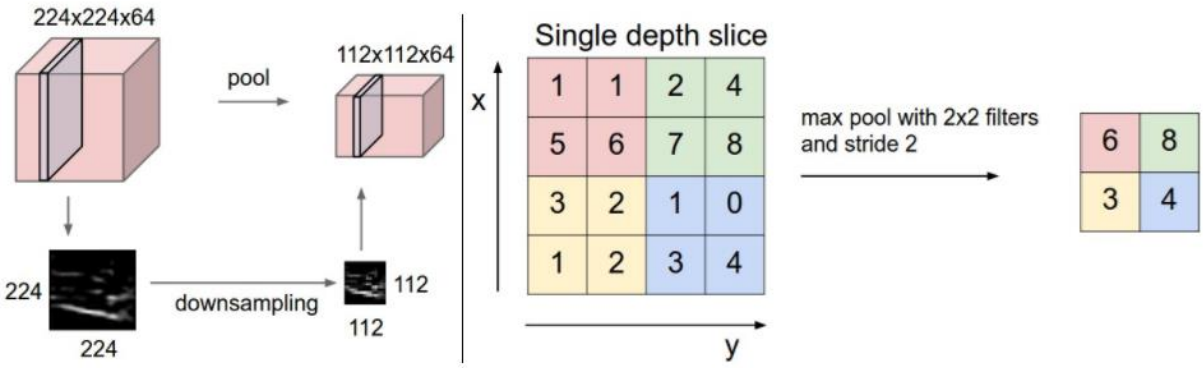


Fig.[12] Ref.[2]

Pooling layer downsamples the volume spatially, independently in each depth slice of the input volume. Left: In this example, the input volume of size [224x224x64] is pooled with filter size 2, stride 2 into output volume of size [112x112x64]. Notice that the volume depth is preserved. Right: The most common downsampling operation is max, giving rise to max pooling, here shown with a stride of 2. That is, each max is taken over 4 numbers (little 2x2 square).

1.2.4 Different architectures in CNN :

There are several architectures in the field of Convolutional Networks that have a name. The most common are:

LeNet. The first successful applications of Convolutional Networks were developed by Yann LeCun in 1990's. Of these, the best known is the LeNet architecture that was used to read zip codes, digits, etc.

AlexNet. The first work that popularized Convolutional Networks in Computer Vision was the AlexNet, developed by Alex Krizhevsky, Ilya Sutskever and Geoff Hinton. The AlexNet was submitted to the ImageNet ILSVRC challenge in 2012 and significantly outperformed the second runner-up (top 5 error of 16% compared to runner-up with 26% error). The Network had a very similar architecture to LeNet, but was deeper, bigger, and featured Convolutional Layers stacked on top of each other (previously it was common to only have a single CONV layer always immediately followed by a POOL layer).

ZF Net. The ILSVRC 2013 winner was a Convolutional Network from Matthew Zeiler and Rob Fergus. It became known as the ZFNet (short for Zeiler & Fergus Net). It was an improvement on AlexNet by tweaking the architecture hyperparameters, in particular by expanding the size of the middle convolutional layers and making the stride and filter size on the first layer smaller.

GoogLeNet. The ILSVRC 2014 winner was a Convolutional Network from Szegedy et al. from Google. Its main contribution was the development of an Inception Module that dramatically reduced the number of parameters in the network (4M, compared to AlexNet with 60M). Additionally, this paper uses Average Pooling instead of Fully Connected layers at the top of the ConvNet, eliminating a large amount of parameters that do not seem to

matter much. There are also several followup versions to the GoogLeNet, most recently Inception-v4.

VGGNet. The runner-up in ILSVRC 2014 was the network from Karen Simonyan and Andrew Zisserman that became known as the VGGNet. Its main contribution was in showing that the depth of the network is a critical component for good performance. Their final best network contains 16 CONV/FC layers and, appealingly, features an extremely homogeneous architecture that only performs 3x3 convolutions and 2x2 pooling from the beginning to the end. Their pretrained model is available for plug and play use in Caffe. A downside of the VGGNet is that it is more expensive to evaluate and uses a lot more memory and parameters (140M). Most of these parameters are in the first fully connected layer, and it was since found that these FC layers can be removed with no performance downgrade, significantly reducing the number of necessary parameters.

ResNet. Residual Network developed by Kaiming He et al. was the winner of ILSVRC 2015. It features special skip connections and a heavy use of batch normalization. The architecture is also missing fully connected layers at the end of the network. The reader is also referred to Kaiming's presentation (video, slides), and some recent experiments that reproduce these networks in Torch. ResNets are currently by far state of the art Convolutional Neural Network models and are the default choice for using ConvNets in practice (as of May 10, 2016). In particular, also see more recent developments that tweak the original architecture from Kaiming He et al. Identity Mappings in Deep Residual Networks (published March 2016).

Challenges in Image Recognition and classification :

- Viewpoint variation. A single instance of an object can be oriented in many ways with respect to the camera.
- Scale variation. Visual classes often exhibit variation in their size (size in the real world, not only in terms of their extent in the image).
- Deformation. Many objects of interest are not rigid bodies and can be deformed in extreme ways.
- Illumination conditions. The effects of illumination are drastic on the pixel level.
- Background clutter. The objects of interest may blend into their environment, making them hard to identify.
- Intra-class variation. The classes of interest can often be relatively broad, such as chair. There are many different types of these objects, each with their own appearance.
- Occlusion. The objects of interest can be occluded. Sometimes only a small portion of an object (as little as few pixels) could be visible

Implementation

1 Algorithms

1.1 Detection Algorithm(Viola – Jones Algorithm) :

The basic features are :

1. **Haar features :**

   They are similar to these convolutional kernels which are used to detect the presence of that features in the given image . Each features results in a single value which is calculated by subtracting the sum of pixels under white rectangle from the sum of pixels under black rectangle.

2. **Integral Image**

   Integral image allows for the calculation of sum of all pixels inside any given rectangle using only four values at the corner of the rectangle .

3. **Adaboost**

   There are approximately 160000+ features values within a detector at 24x24 baser resolution which need to be calculated . But it is to be understood that only few set of features will be useful among all these features to identify .It is a machine learning algorithm which helps in finding only the best features among all these 160000+. After these features are found a weighted combination of all these features in used in evaluating and deciding any given window has a face or not .

4. **Cascading**

   A cascade classifier is used which is composed of stages each containing a strong classifier. So all the features are grouped into several stages where each stage has certain number of features .

1.2 Training Algorithm

1) Input the folder containing the images and respective labels.

2) Creates the datastore for the input images and their corresponding label

3) Loop now to crop the image to the defined dimension and writes back in the datastore

4) Initialize the pretrained AlexNet network

5) Split the labels into two sets of training anf testing images and finds the number of total labels of different classes

6) Eliminate the last three layers of the AlexNet neural net and Fully connected layer

7) Train of the neural net

8) Classify the test images and finding the accuracy achieved

9) Save the Convolutional neural network formed for further testing purposes

1.3Testing Algorithm

1) Reading the image from the user

2) Extraction of the periocular region from the image

3) Classify the image using classify.

4) Show the image and classification result together

2 Flow Chart

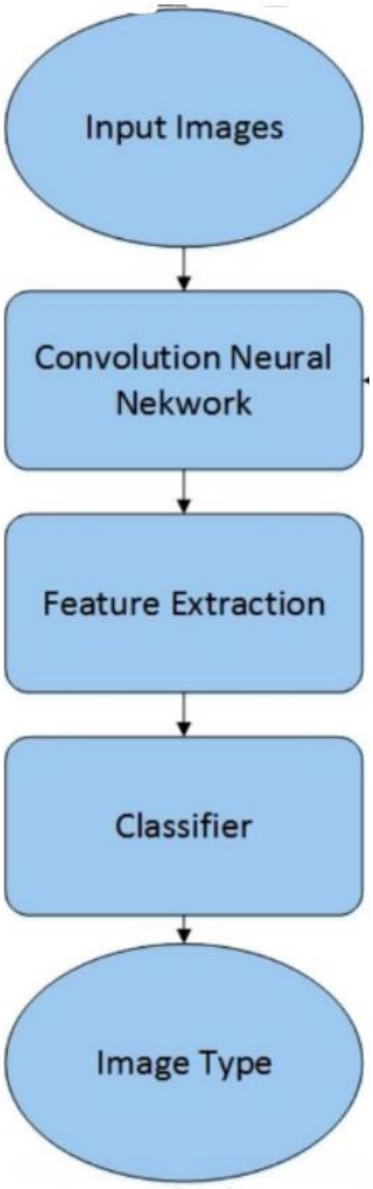


Fig.[13] Ref .[6]

2.1 Description of the Flowchart :

1. Input Image :

   We input the image by browsing through the specific path i.e using uigetfile( ). Then we read the image using the function imread ( ) . We make sure that we resize the original image into particular respective dimension for computation.

2. Feature Extraction :

   Convolutional Neural Network extract the different features through the different layer from the image like max pool, average pool etc.

3. Classifier :

   We predict the label of the input image with the help of the SVM and fully connected layer which has been added to the alexnet for increasing the accuracy and efficiency of the algorithm

4. Image Type :

   The classified image is shown and labeled .

3 Implementation details

1) Software -

   MATLAB 2017a was used for the project in training and testing purpose.

2) Language -

   MATLAB (*matrix laboratory*) is a multi-paradigm numerical computing environment.

3) Dataset used -

   Faces94 dataset consisting of 40 different persons.

4.Code :

4.1 Image training :

```
%To input the folder containing the images and respective labels
r=uigetdir();

%Creates the datastore for the input images and their corresponding label
images = imageDatastore(r,...
    'IncludeSubfolders',true,...
   'LabelSource','foldernames');
n=numel(images.Labels);
m=numel((countEachLabel(images)))/2;

%This loop crops the image to the defined dimension and writes back in the datastore
for i = 1:n
        a=imread((images.Files{i,1}));
        a=imcrop(a,bbox);
        a=imresize(a,[227 227]);
        imwrite(a,images.Files{i,1});
        pause(0.0010000);
end

%Initialises the pretrained AlexNet network
net=alexnet;

%This splits the labels into two sets of training anf testing images and finds the number of
total labels of different classes
[trainingImages,testImages] = splitEachLabel(images,0.7,'randomized');
numTrainImages = numel(trainingImages.Labels);
trainingLabels = trainingImages.Labels;
testLabels = testImages.Labels;

%Here we elemimnate the last three layers of the AlexNet neural net and Fully conneced
layer
layersTransfer=net.Layers(1:end-3);
layers = [layersTransfer,
    fullyConnectedLayer(m,'WeightLearnRateFactor',20,'BiasLearnRateFactor',20),
    softmaxLayer,
    classificationLayer]
```

```
options = trainingOptions('sgdm',...
    'MiniBatchSize',5,...
    'MaxEpochs',10,...
   'InitialLearnRate',0.0001);

%The Training of the neural net
netfinal = trainNetwork(trainingImages,layers,options);

%Classifying the test images and finding the accuracy achieved
predictedLabels = classify(netfinal,testImages);
accuracy = mean(predictedLabels == testLabels);

%saving the Convolutional neural network formed for further testing purposes
save netfinal;
```

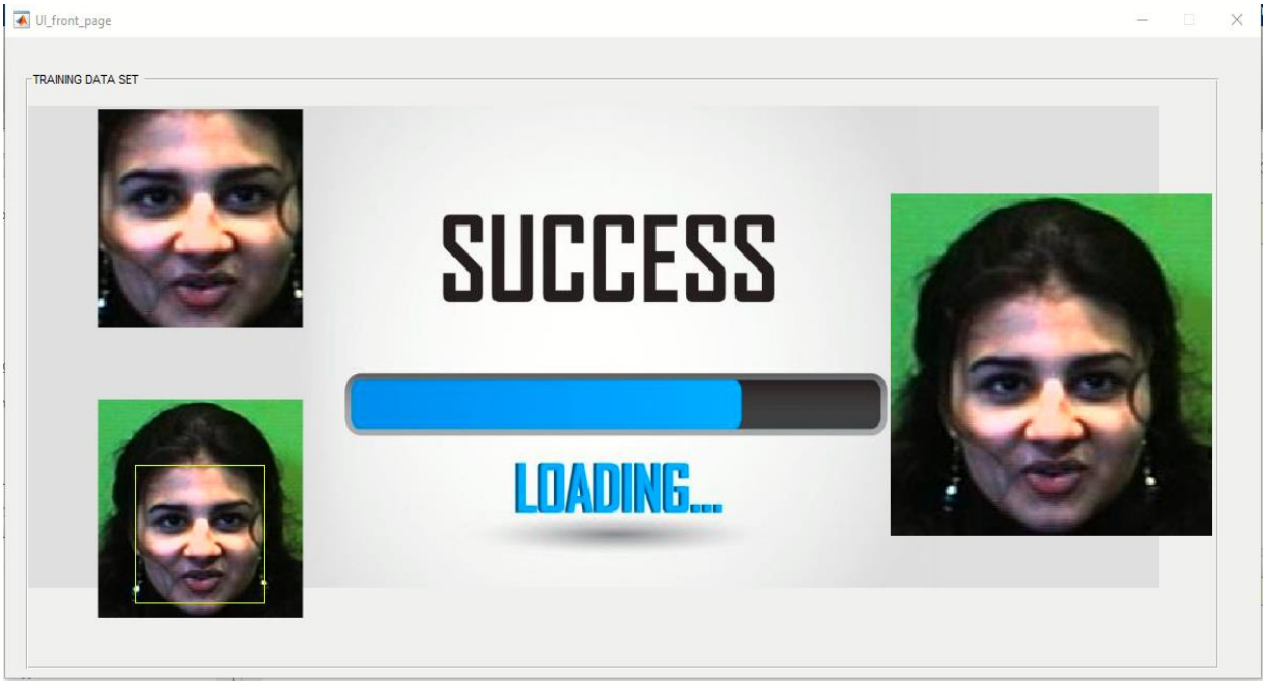
4.2 Image testing :

```
%Reading the image from the user
[filename pathname] = uigetfile({'*.jpg';'*.bmp'},'File Selector');
pic1 = strcat(pathname, filename);
pic1=imread(pic1);
pic1 = imresize(pic1,[227 227]);

%Extraction of the periocular region from the image
detector = vision.CascadeObjectDetector('EyePairBig');
detector.MergeThreshold = 1;
bbox = step(detector,pic1);
out = insertObjectAnnotation(pic1,'rectangle',bbox);
pic2=imcrop(pic1,bbox)

%Classify the image using classify.
predictedLabels = classify(netTransfer,pic2);

%Show the image and classification result together
disp(predictedLabels);
```

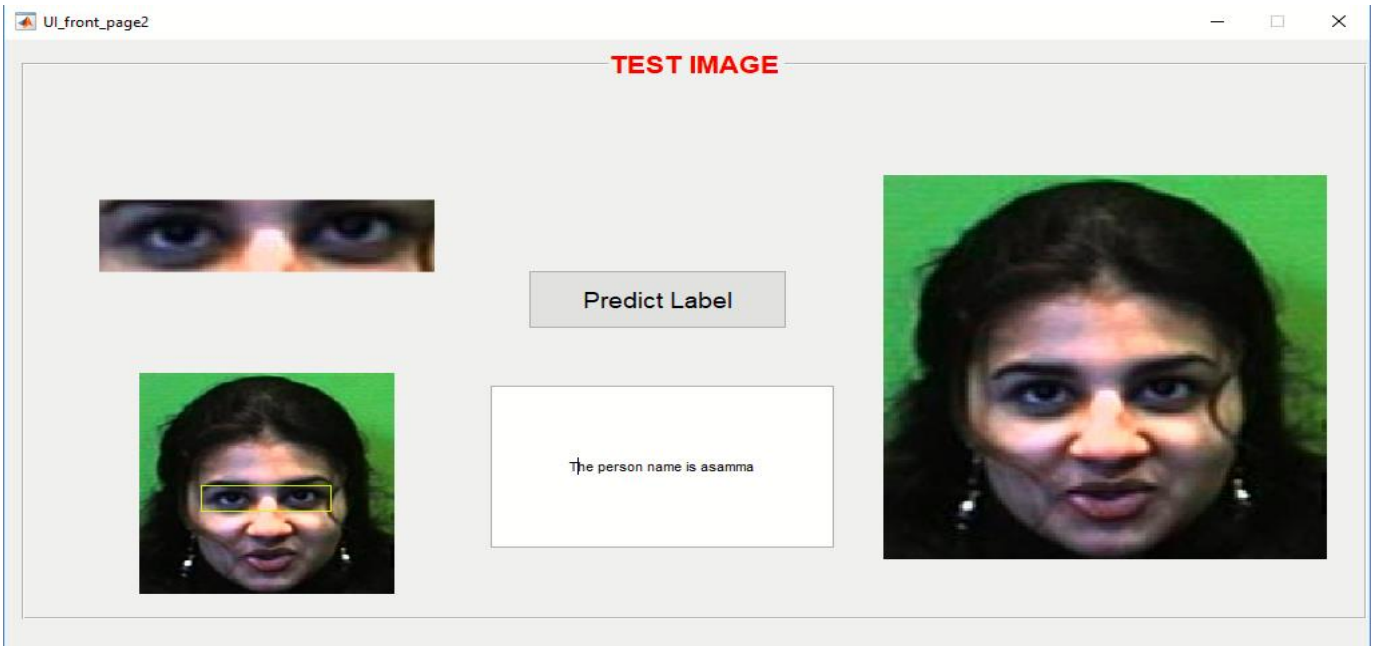
Screenshots

1) **Front Page :**

2) **Training :**



3) Testing



Results:

In this periocular biometrics image recognition project, we have achieved an accuracy of 94%.The training time took 6 hours on the Faces94 dataset.

Conclusion :

We have presented a face detector with a reasonably good accuracy and running time. However, many aspects of the design are tuned for the constrained scene conditions of the training images provided, hurting its robustness. This is not unfair given the scope and requirements of the project. Our algorithm is sensitive to the color information in the image and will not work for a gray scale image.

For multi face images, the face detection algo does not work properly to identify every image hence we have taken single face images for which the algorithm works the best with good accuracy. We have taken dataset with images having good brightness and face area shown so as to detect face easily which is not possible in real scenarios and images.

We have the limitations of good processors like GPU so the learning takes much of a time .If provided with proper computers with GLU's , the algorithm works even faster hence the efficiency of the algorithm increases.

References :

1. http://cs231n.github.io/classification/
2. http://cs231n.github.io/convolutional-networks/
   (Stanford University)
3. https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/
4. A survey on periocular biometrics research
   (Fernando Alonso-Fernandez ∗, Josef Bigun)
5. https://content.iospress.com/articles/ai-communications/aic739