FINE TUNING BASED AUTOMATIC FACE RECOGNITION USING ALEXNET

[1]D.Bharadwaja, [2]P.Sreenivasullu,
[1]M.Tech (Digital Electronics and Communication Systems), [2]Professor,
Electronics and Communications Engineering, NEC, Gudur.

Abstract: Face recognition has become a fascinating field for researchers. The motivation behind the enormous interest in the topic is the need to improve the accuracy of many real-time applications such as identity authentication, access control surveillance. The complexity of the human face and the changes due to different effects make it more challenging to design as well as implement a powerful computational system for human face recognition. Face recognition is an important biometric that is used to uniquely identify an individual. However, the presence of various disguises in the face will diminish the performance of any facial recognition system. Disguised face recognition problem becomes extremely challenging when important facial features get covered up. Because of the increase in popularity and success of deep learning in computer vision problems, features learned by Convolutional Neural Networks (CNN) can be used for face recognition. Recognition is done by using a pre-trained Convolutional Neural Network (Alex-Net Model) for facial feature extraction. A new ORL dataset of covered faces has also been introduced for training the deep networks, this indeed is very useful for law enforcement and other organizations like security, bio-metric as it would help to identify criminals, Protestants.

Index Term: Face Recognition, Convolutional Neural Network, AlexNet.

I. INTRODUCTION

A face is always considered as an important biometric that uniquely identifies an individual. The Face recognition technique identifies an individual by comparing the input image to the stored record of images. Such systems usually analyze the characteristics of individual's face and based on the analysis it will recognize the person. It is based on the fact that different individuals have a different facial feature which makes them unique. In comparison to other existing biometric systems like fingerprint recognition, handwriting recognition, etc. face recognition can attain a higher performance in security systems. Numerous algorithms and techniques have been developed for improving the performance of face recognition. These systems are usually susceptible to several challenges including illumination, image quality, expression, pose, aging, disguise, etc. Among these challenges, recognition of faces with disguise is a major challenge and has only been recently addressed by few researchers.

Disguise is an important and crucial face recognition challenge. There can be both intentional and unintentional changes through which a person becomes very difficult to be identified. Intentional disguises are those in which a person knowingly changes his identity by wearing a wig, changing hairstyle, wearing glasses, etc. Now a days it has become very difficult for the cops to identify protestants and criminals as they often find it easier to hide from the law by disguising themselves.

The existing face recognition systems are often found to be a failure in case of identifying people who cover their faces using scarves with only the eye portion visible. This is because the recognition system could not perform well as the lower facial portion gets covered up. Recently Deep learning has been highly explored in the field of computer vision. Many face recognition systems based on deep learning have been developed and it has been seen that they outperformed all the existing systems. Deep learning methods make use of deep neural networks such as Convolution Neural Networks (CNN).

Since training a deep model to require large dataset and huge computing power, designers often perform transfer learning. Transfer learning is the method by which a model developed for one task can be reused for a different related task. Since the pre-trained network has already learned several features, it can be used for a new classification task by just fine-tuning the network. Another advantage is that the number of images required for training and testing, the training time is reduced. Here, a pre-trained CNN AlexNet model is used for extracting features from facial images. The extracted features are then classified by training a multiclass SVM,

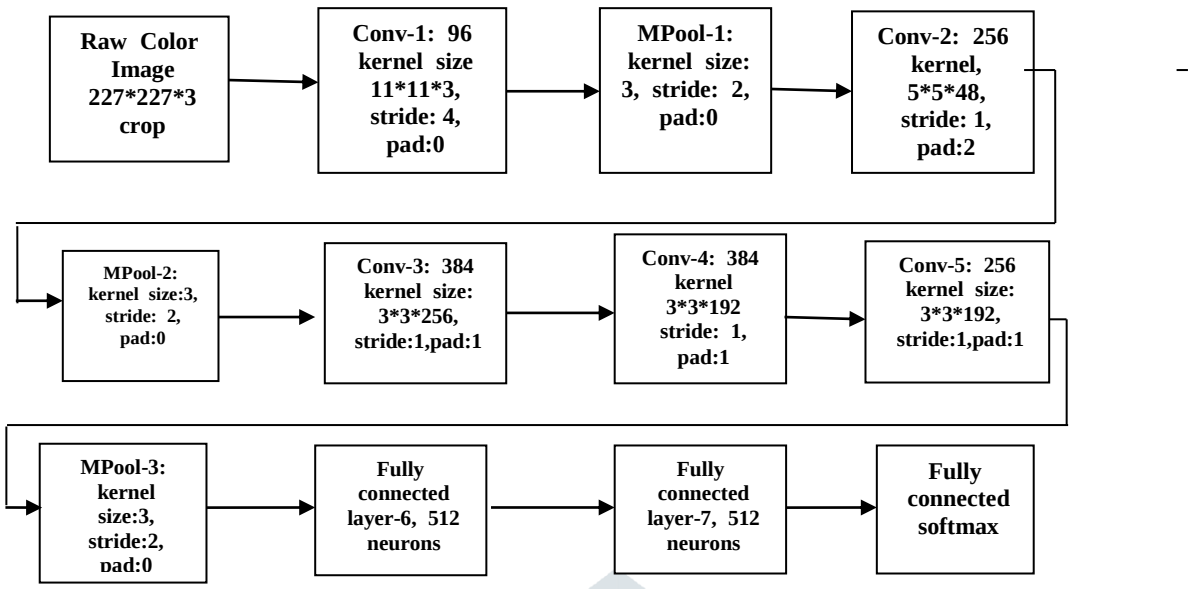

Figure 1:- Block diagram of Face recognition using Alexnet

II. EXISTING METHOD

Face recognition has become a fascinating field for researchers. The motivation behind the enormous interest in the topic is the need to improve the accuracy of many real-time applications. The complexity of the human face and the changes due to different effects make it more challenging to design as well as implement a powerful computational system for human face recognition. It is presented with an enhanced approach to improve human face recognition using a back-propagation neural network (BPNN) and features extraction based on the correlation between the training images. A key contribution of this paper is the generation of a new set called the T-Dataset from the original training data set, which is used to train the BPNN. It generated the T-Dataset using the correlation between the training images without using a common technique of image density. The correlated T-Dataset provides a high distinction layer between the training images, which helps the BPNN to converge faster and achieve better accuracy. Data and features reduction is essential in the face recognition process, and researchers have recently focused on the modern neural network.

In existing system applies five distance measurement algorithms and then combined them to obtain the T-Dataset, which is fed into the BPNN, it achieved higher face recognition accuracy with less computational cost compared with the current approach by using reduced image features. It tests the framework on two small data sets, the YALE and AT&T data sets, as the ground truth. Furthermore, it evaluates our method on one of the state-of-the-art benchmark data sets, it produce a competitive face recognition performance.

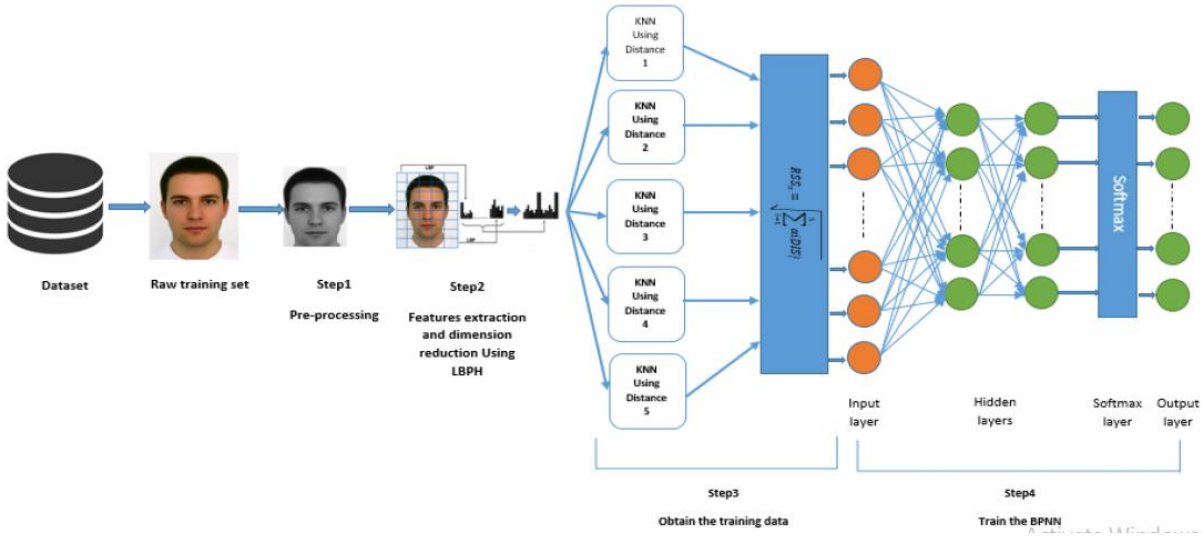

Figure 2: Existing recognition system using LBPH descriptors, multi-KNN, and BPNN neural network.

In this enhanced human face recognition using LBPH descriptors, BPNN, and multi-KNN. The figure above shows the proposed framework in detail. The main contribution is based on the fact that obtaining a robust T-Dataset will help the BPNN to converge quickly with improved accuracy. It gathered a robust T-Dataset relying on the correlation between the training images, not the density of images. Our method is divided into five steps. In step one; it has applied some of the pre-processing methods on the raw training images, including resizing and cropping using Haar-cascade detection, to eliminate the face background effect. Noise and illumination were reduced by converting the images to greyscale images and using histogram equalization to build a robust face

recognition system; it is extracted the most important local features from each image using the LBP descriptors and combined them into a global description using the histogram method.

Here the images are divided into 25 small cells after tried different grid sizes, it found that the 5x5 grid gives better performance at a reasonable time. Smaller grid sizes such as 4x4 provide fewer features 4* 4* 59 = 944 compared to 5*5*59 = 1475 features, which leads to less accuracy and perhaps to an under setting problem with the neural network training. A larger grid size provides more features however; it increases the computing time with a slight improvement inaccuracy, it applies the LBP method on image pixels by thresholding the 3*3 neighbourhood of each pixel with the centre value and considering the result as a binary number.

Finally, it applied the histogram method to concatenate the description of the new cell and obtain a new representation of 25 cell *59 dimension = 1475 for each training image, which helps to reduce the computation time. Step 3 was added as an extra step to obtain a robust T-Dataset, which is used as an input to our BPNN instead of using the LBPH descriptor of each training image. As mentioned earlier, the T-Dataset is gathered based on the correlation between the new representations of all training images. Based on the LBPH presentation of each image, which is obtained from step 2, it calculates the distance between each training image with all other training images using five distance methods are Correlation, Euclidean, Canberra, Manhattan, and Mahalanobis. It tried different scenarios to achieve higher accuracy.

First, it trained the BPNN using each distance method separately, and it achieved variant accuracy in another scenario, combined the five distances using the square-root of the sum of the squares to provide a robust distinction T-Dataset in a reduced dimension. In Step 4, the BPNN parameters are set up and then the training begins. In BPNN architecture contains an input layer followed by two fully connected hidden layers, followed by a soft-max classification layer.

In existing method, face recognition is done by using Local Binary Pattern Histogram, multi-KNN, and Back-Propagation Neural Networks are used to recognize, but it takes more time to achieve processing of faces recognition and also give less accuracy, this drawback can be overcome by using a proposed method.

III. PROPOSED METHOD

The proposed technique using a Deep learning technique is used to solve the face recognition problem, concerning reducing data consumption, and increase the recognition speed with help of pre-trained networks. Deep learning has been highly explored in the field of computer vision. Deep learning methods make use of deep neural networks such as Convolution Neural Networks (CNN) and Alexnet.

Since training a deep model to require large dataset and huge computing power, designers often perform transfer learning. Transfer learning is the method by which a model developed for one task can be reused for a different related task. Since the pre-trained network has already learned several features, it can be used for a new classification task by just fine-tuning the network. Another advantage is that the number of images required for training and testing, the training time is reduced. Here, a pre-trained CNN Alexnet model is used for extracting features from facial images. The extracted features are then classified by training a multiclass SVM,

A. BASIC DEEP LEARNING PROCESSES

To understand the activities going on in each stage of a deep neural network, below it explains the processes based on some mathematical principles. In this process deep learning neural networks help us to get more accuracy in face recognition, generally deep learning neural network only works in GPU's but by using Alexnet, it works in general computer and gives 98% accuracy.

Processing steps:

1. Preparing the Dataset,
2. Loading the Dataset,
3. Generating Eigen faces and recognizing faces.

1. Preparing the Data-set:

While preparing a dataset of 40 peoples. Each of these people has 10 images with different poses. That means in total there are 40 x 10 = 400 images. For every individual there are separate folders is created to store images and it will reduce confusion. The way to prepare the datasets is shown in bellow figure Fig 2.

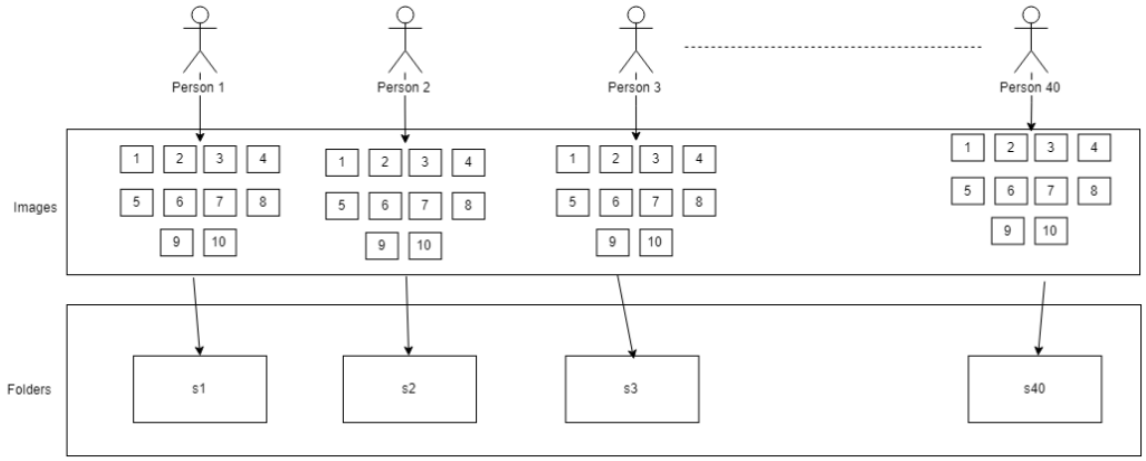

Figure 3: Preparing Dataset for Face Recognition using MATLAB

In fig 3 in the top row, there are 40 people marked as 1, 2, 3 to 40. Each of them has 10 images. These images are on a gray-scale. All of these images must have to be of the same dimension and resolution. Finally, all images of every individual are stored into a separate folder. In figure 3, it represents "s1, s2, s3 ….. s40" as folders and the structure of the dataset.

The steps involved in preparing the dataset.

- 10 images for every person
- 1 folder for each person
- Images must be in grayscale
- Images must have to be of the same resolution and dimension. The resolution of the image is 92 x 112 pixels.
- The name of the image must have to be numeric such as 1, 2, 3….,
- And the images must have to have the same extension such as BMP, PGM and so on. Do not mix up different extensions of image, after preparing the data-set, and the next step is to load the data sets.

2. Loading data-set:

After preparing the dataset, the next task is loading the dataset. It will implement a function in MATLAB to load the dataset. Let's name it 'load database' it declares that the name of the function 'load_database()' It doesn't take any input. That is why it has empty parenthesis. But it returns numeric form of images. The images to be returned will be stored in a variable named 'ouput_value'. After that, it takes two more variables named 'loaded' and 'numeric_Images'. These variables are persistent types.

"The persistent variables are local to the function in which they are declared, yet their values are retained in memory between calls to the function. Persistent variables are similar to global variables because MATLAB creates permanent storage for both. They differ from global variables in that persistent variables are known only to the function in which they are declared. This prevents persistent variables from being changed by other functions or from the MATLAB command line. After that, in the 'if' condition it is empty only then load the dataset. The persistent variables permanently store the data. And it needs to load the dataset only once. That is why it is important to check if the variable is empty or not at the beginning".

Now in database consist of 40 images. Each of them has 92 x 112 = 10304 pixels. Later the pixel values of the images will replace these 'zeros'. Then inside the 'for loop' using strcat function, it concatenating the names of the folders as s1, s2, s3 and so on, names of the images and the extension of the images (.pgm images).

After that using 'imread' function the images are loaded. It is necessary to reshape the images. After loading the images it used 'reshape' function to convert the images into a single column matrix, and also used 'size' function which gives the size of the rows and columns of the images. After that, it converted the images into 8-bit unsigned integer to reduce the memory usage. Using the 'uint8' function, it can directly convert data into 8-bit unsigned integers in MATLAB. It is to prevent the function from further loading the dataset.

3. Face Recognition using Alexnet:

To recognize the faces first load the dataset. After that using random function, it generates a random index. Using the sequence of a random index, the image is loaded which will be recognized later. The rest of the images are also loaded into a separate variable. After that, the loaded data set is given to Convolution neural network for classifying the images concerning their biases and weights of every image.

Finally, the difference between current image signatures with the pre-trained signature is compared; it will predict the recognized face. The accuracy of this method is around 98% with the help of CNN and Alexnet. This program will automatically load an image and then will find the image of the same person from the image data-set.

Figure 4: The final result of face recognition using MATLAB

The method used in the proposed system is the convolution neural network with respect to Alexnet, Alexnet consists of eight layers and those layers are divided into two types as 5 Convolutional layer and 3 max-pooling layers. The figure shows the layers in Alexnet algorithm

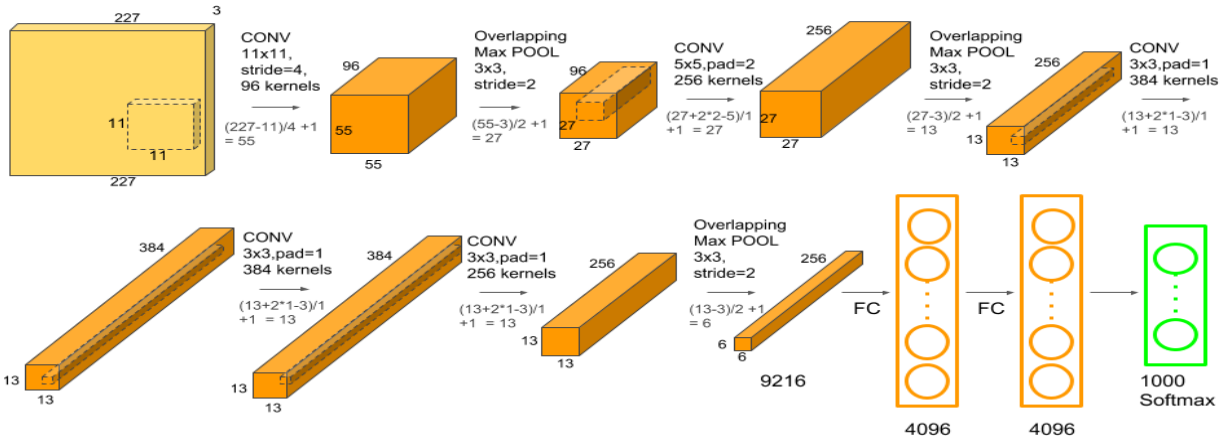

Figure 5: AlexNet architecture

AlexNet has the following layers:

1. **Input**: The AlexNet mentions the input size of 224×224.
2. **Conv-1**: The first convolutional layer consists of 96 kernels of size 11×11 applied with a stride of 4 and a padding of 0.
3. **MaxPool-1**: The max pool layer following Conv-1 consists of a pooling size of 3×3 and stride 2.
4. **Conv-2**: The second conv layer consists of 256 kernels of size 5×5 applied with a stride of 1 and padding of 2.
5. **MaxPool-2**: The max pool layer following Conv-2 consists of a pooling size of 3×3 and a stride of 2.
6. **Conv-3**: The third conv layer consists of 384 kernels of size 3×3 applied with a stride of 1 and a padding of 1.
7. **Conv-4**: The fourth conv layer has the same structure as the third conv layer. It consists of 384 kernels of size 3×3 applied with a stride of 1 and a padding of 1.
8. **Conv-5**: The fifth conv layer consists of 256 kernels of size 3×3 applied with a stride of 1 and a padding of 1.
9. **MaxPool-3**: The max pool layer following Conv-5 consists of a pooling size of 3×3 and a stride of 2.
10. **FC-1**: The first fully connected layer has 4096 neurons.
11. **FC-2**: The second fully connected layer has 4096 neurons.
12. **FC-3**: The third fully connected layer has 1000 neurons.

Alexnet architecture concerning parameters and size:

A. Calculating the tensor size at each stage

1. First the size of the Output Tensor of a Conv Layer

Let's define

O = Size of output image.
I = Size of input image.
K = Size of kernels used in the Conv Layer.
N = Number of kernels.
S = Stride of the convolution operation.
P = Padding.

The size (O) of the output image is given by

$$O = \frac{I - K + 2P}{S} + 1$$

The number of channels in the output image is equal to the number of kernels N.

2. Second the Size of Output Tensor of a Max Pool Layer

Let's define

O= Size of output image.
I = Size of input image.
S = Stride of the convolution operation.
Ps = Pool size.

The size (O) of the output image is given by

$$0 = \frac{I - Ps}{S} + 1$$

Note that this can be obtained using the formula for the convolution layer by making padding equal to zero and keeping Pool size the same as the kernel size. But unlike the convolution layer, the number of channels in the max pool layer's output is unchanged. Similarly the sizes of the outputs of MaxPool-2 and MaxPool-3 Third, the size of the output of a fully connected layer, A fully connected layer outputs a vector of length equal to the number of neurons in the layer**.**

In AlexNet, the input is an image of size 227x227x3. After Conv-1, the size of changes to 55x55x96 which is transformed to 27x27x96 after MaxPool-1, After Conv-2, and the size changes to 27x27x256 and following MaxPool-2 it changes to 13x13x256. Conv-3 transforms it to a size of 13x13x384, while Conv-4 preserves the size and Conv-5 changes the size back go 27x27x256. Finally, MaxPool-3 reduces the size to 6x6x256. This image feeds into FC-1 which transforms it into a vector of size 4096×1. The size remains unchanged through FC-2, and finally, the output of size 1000×1 after FC-3.
Next, calculate the number of parameters in each Conv Layer.

B. Calculating the total number of parameters in the network

1. Number of Parameters of a Conv Layer

On CNN, each layer has two kinds of parameters: weights and biases. The total number of parameters is just the sum of all weights and biases.

Let's define,

W_c= Number of weights of the Conv Layer.
B_c = Number of biases of the Conv Layer.
P_c = Number of parameters of the Conv Layer.
K = Size of kernels used in the Conv Layer.
N = Number of kernels.
C = Number of channels of the input image.

$$W_c = K^2 * C * N$$
$$B_c = N$$
$$P_c = W_c + B_c$$

In a Conv Layer, the depth of every kernel is always equal to the number of channels in the input image. So every kernel has K^2 x C parameters, and there are N such kernels.

2. Number of Parameters of a Max Pool Layer

There are no parameters associated with a Max Pool layer. The pool size, stride, and padding are hyper parameters. Number of Parameters of a Fully Connected (FC) Layer, There are two kinds of fully connected layers in CNN. The first FC layer is connected to the last Conv Layer, while later FC layers are connected to other FC layers.
Let's consider each case separately.

Case 1: Number of Parameters of a Fully Connected (FC) Layer connected to a Conv Layer

Let's define,

W_{CF}= Number of weights of an FC Layer which is connected to a Conv Layer.
B_{CF} = Number of biases of an FC Layer which is connected to a Conv Layer.
O = Size of the output image of the previous Conv Layer.
N = Number of kernels in the previous Conv Layer.
F = Number of neurons in the FC Layer.

$$W_{CF} = O^2 \times N \times F$$
$$B_{CF} = F$$
$$P_{CF} = W_{CF} + B_{CF}$$

Case 2: Number of Parameters of a Fully Connected (FC) Layer connected to an FC Layer

Let's define,

W_{ff} = Number of weights of an FC Layer which is connected to an FC Layer.
B_{ff} = Number of biases of an FC Layer which is connected to an FC Layer.
P_{ff} = Number of parameters of an FC Layer which is connected to an FC Layer.
F = Number of neurons in the FC Layer.
F_{-1} = Number of neurons in the previous FC Layer.

$$W_{ff} = F_{-1} \times F$$
$$B_{ff} = F$$
$$P_{ff} = W_{ff} + B_{ff}$$

In the above equation, F_{-1} x F is the total number of connection weights from neurons of the previous FC Layer to the neurons of the current FC Layer. The total number of biases is the same as the number of neurons (F). Similarly verifying FC2 as above equation, the total number of parameters for FC-2 in AlexNet is 16,781,312.
Number of Parameters and Tensor Sizes in AlexNet

The total number of parameters in AlexNet is the sum of all parameters in the 5 Conv Layers + 3 FC Layers. It comes out to a whopping 62,378,344, the table below provides a summary.

Table 1: Total number of parameters

Layer Name	Tensor Size	Weights	Biases	Parameters
Input Image	227x227x3	0	0	0
Conv-1	55x55x96	34,848	96	34,944
MaxPool-1	27x27x96	0	0	0
Conv-2	27x27x256	614,400	256	614,656
MaxPool-2	13x13x256	0	0	0
Conv-3	13x13x384	884,736	384	885,120
Conv-4	13x13x384	1,327,104	384	1,327,488
Conv-5	13x13x256	884,736	256	884,992
MaxPool-3	6x6x256	0	0	0
FC-1	4096×1	37,748,736	4,096	37,752,832
FC-2	4096×1	16,777,216	4,096	16,781,312
FC-3	1000×1	4,096,000	1,000	4,097,000
Output	1000×1	0	0	0
Total				62,378,344

Finally Alexnet compares every image\face with respect to trained network and gives perfect accuracy by using parameters and weights of every person faces\images.

IV. CNN overview

The overall architecture of CNNs consists of two main parts: feature extractors and a classifier. In the feature extraction layers, each layer of the network receives the output from its immediate previous layer as its input and passes its output as the input to the next layer. The CNN architecture consists of a combination of three types of layers: convolution, max-pooling, and classification.

There are two types of layers in the low and middle-level of the network: convolutional layers and max-pooling layers. The even numbered layers are for convolutions and the odd numbered layers are for max-pooling operations. The output nodes of the convolution and max-pooling layers are grouped into a 2D plane called feature mapping. Each plane of a layer is usually derived from the combination of one or more planes of previous layers. The nodes of a plane are connected to a small region of each connected plane of the previous layer. Each node of the convolution layer extracts the features from the input images by convolution operations on the input nodes.

Higher-level features are derived from features propagated from lower level layers. As the features propagate to the highest layer or level, the dimensions of features are reduced depending on the size of the kernel for the convolutional and max-pooling operations respectively. However, the number of feature maps usually increased for representing better features of the input images for ensuring classification accuracy. The output of the last layer of the CNN is used as the input to a fully connected network which is called the classification layer. Feed-forward neural networks have been used as the classification layer as they have better performance.

In the classification layer, the desired number of features is selected as inputs concerning the dimension of the weight matrix of the final neural network. However, the fully connected layers are expensive in terms of network or learning parameters. Nowadays, there are several new techniques including average pooling and global average pooling that are used as an alternative of fully-connected networks. The score of the respective class is calculated in the top classification layer using a soft-max layer. Based on the highest score, the classifier gives output for the corresponding classes.

1) Convolution Layer

In this layer, feature maps from previous layers are convolved with learnable kernels. The output of the kernels go through a linear or non-linear activation function such as a sigmoid, hyperbolic tangent, Softmax, rectified linear, and identity functions to form the output feature maps. Each of the output feature maps can be combined with more than one input feature map. $x_{jl}=f(\Sigma x_{il-1 i\epsilon Mj} * k_{ijl} + b_{jl})$

Where x_{jl} is the output of the current layer, x_{il-1} is the previous layer output, k_{ijl} is the kernel for the present layer, and b_{jl} are biases for the current layer. M_j represents a selection of input maps. For each output map, an additive bias b is given. However, the input maps will be convolved with distinct kernels to generate the corresponding output maps. The output maps finally go through a linear or non-linear activation function such as sigmoid, hyperbolic tangent, Softmax, rectified linear, or identity functions.

2) Sub-sampling Layer

The sub-sampling layer performs the down sampled operation on the input maps. This is commonly known as the pooling layer. In this layer, the number of input and output feature maps does not change. For example, if there are N input maps, then there will be exactly N output maps. Due to the down sampling operation, the size of each dimension of the output maps will be reduced, depending on the size of the down sampling mask. For example: if a 2×2 down sampling kernel is used, then each output dimension will be half of the corresponding input dimension for all the images. This operation can be formulated as

$$x_{jl}=\text{down}(x_{jl-1})$$

Where down (.) represents a sub-sampling function. Two types of operations are mostly performed in this layer: average pooling or max-pooling. In the case of the average pooling approach, the function usually sums up over N×N patches of the feature maps from the previous layer and selects the average value. On the other hand, in the case of max-pooling, the highest value is selected from the N×N patches of the feature maps. Therefore, the output map dimensions are reduced by n times. In some special cases, each output map is multiplied with a scalar. Some alternative sub-sampling layers have been proposed, such as fractional max-pooling layer and sub-sampling with convolution.

.

3) Classification Layer

This is the fully connected layer which computes the score of each class from the extracted features from a convolutional layer in the preceding steps. The final layer feature maps are represented as vectors with scalar values that are passed to the fully connected layers. The fully connected feed-forward neural layers are used as a soft-max classification layer. There are no strict rules on the number of layers that are incorporated in the network model. However, in most cases, two to four layers have been observed in different architectures including LeNet, AlexNet, and VGG Net.

As the fully connected layers are expensive in terms of computation, alternative approaches have been proposed during the last few years. These include the global average pooling layer and the average pooling layer which help to reduce the number of parameters in the network significantly.

In the backward propagation through the CNNs, the fully connected layers update following the general approach of fully connected neural networks (FCNN). The filters of the convolutional layers are updated by performing the full convolutional operation on the feature maps between the convolutional layer and its immediate previous layer. Fig. 4 shows the basic operations in the convolution and sub-sampling of an input image.

4) Network parameters and required memory for CNN

The number of computational parameters is an important metric to measure the complexity of a deep learning model. The size of the output feature maps can be formulated as follows:

$$M=_{(N-F)S}+1$$

Where N refers to the dimensions of the input feature maps, F refers to the dimensions of the filters or the receptive field, M refers to the dimensions of output feature maps, and S stands for the stride length. Padding is typically applied during the convolution operations to ensure the input and output feature map have the same dimensions. The amount of padding depends on the size of the kernel. The equation is used for determining the number of rows and columns for padding.

$$P=(F-1)/2$$

Here P is the amount of padding and F refers to the dimension of the kernels. Several criteria are considered for comparing the models. However, in most of the cases, the number of network parameters and the total amount of memory are considered. The number of parameters ($Parm_l$) of l_{th} layer is calculated based on the following equation:

$$Parm_l=(F\times F\times FM_{l-1})\times FM_l$$

If bias is added with the weights, then the above equation can be written as follows:

$$Parm_l=(F\times(F+1)\times FM_{l-1})\times FM_l$$

Here the total number of parameters of l_{th} layer can be represented with P_l, FM_l is for the total number of output feature maps, and FM_{l-1} is the total number of input feature maps or channels. For example, let's assume the l_{th} layer has $M_{l-1}=32$ input features maps, $FM_l=64$ output feature maps, and the filter size is $F=5$. In this case, the total number of parameters with a bias for this layer is $Parm_l=(5\times5\times33)\times64=528{,}000$

Thus, the amount of memory (Mem_l) needs for the operations of the l_{th} layer can be expressed as

$$Mem_l=(N_l\times N_l\times FM_l)$$

Applications of CNNs

Most of the techniques that have been discussed above are evaluated on computer vision and image processing tasks. Here are some recently published papers that have been discussed, which are applied for different modalities of computer vision and image processing.

1) CNNs for solving Graph problem: Learning a graph data structure is a common problem with various different applications in data mining and machine learning tasks. DL techniques have made a bridge in between the machine learning and data mining groups. An efficient CNN for arbitrary graph processing was proposed in 2016.

2) Image processing and computer vision: Most of the models, then it apply in different application domains including image classification, detection, segmentation, localization, captioning, video classification and many more. There is a good survey on deep learning approaches for image processing and computer vision-related tasks Single image super-resolution using CNN methods. Image de-noising using block-matching CNN. CNN methods are also applied for speech processing: speech enhancement using multimodal deep CNN, and audio tagging using Convolutional Gated Recurrent Network (CGRN).

3) CNN for medical imaging: A good survey on DL for medical imaging for classification, detection, and segmentation tasks, there are some papers published after this survey, which was developed for medical diagnosis with images and corresponding text descriptions, Cardiac Segmentation using short-axis MRI, Segmentation of optic disc and retinal vasculature using CNN. Brain tumour segmentation using random forests with features learned with fully convolutional neural network (FCNN),

V. RESULTS AND DISCUSSION

The proposed method of Convolutional neural network (ALEXNET) in MATLAB R20108a is used for recognition of faces with respect to different facial expressions; here Alexnet is used to train the data set with respect to get better results in face recognition. By using Alexnet the accuracy will increase to 98.6% with respect to less storage of data set, and it improves the operation speed of face recognition,

The input to Alexnet is an RGB image of size 256×256. This means all images in the training set and all test images need to be of size 256×256. If the input image is not 256×256, it needs to be converted to 256×256 before using it for training the network. To achieve this, the smaller dimension is resized to 256 and then the resulting image is cropped to obtain a 256×256 image.

If the input image is gray-scale, it is converted to an RGB image by replicating the single channel to obtain a 3-channel RGB image. Random crops of size 224×224 were generated from inside the 256×256 images to feed the first layer of Alexnet.

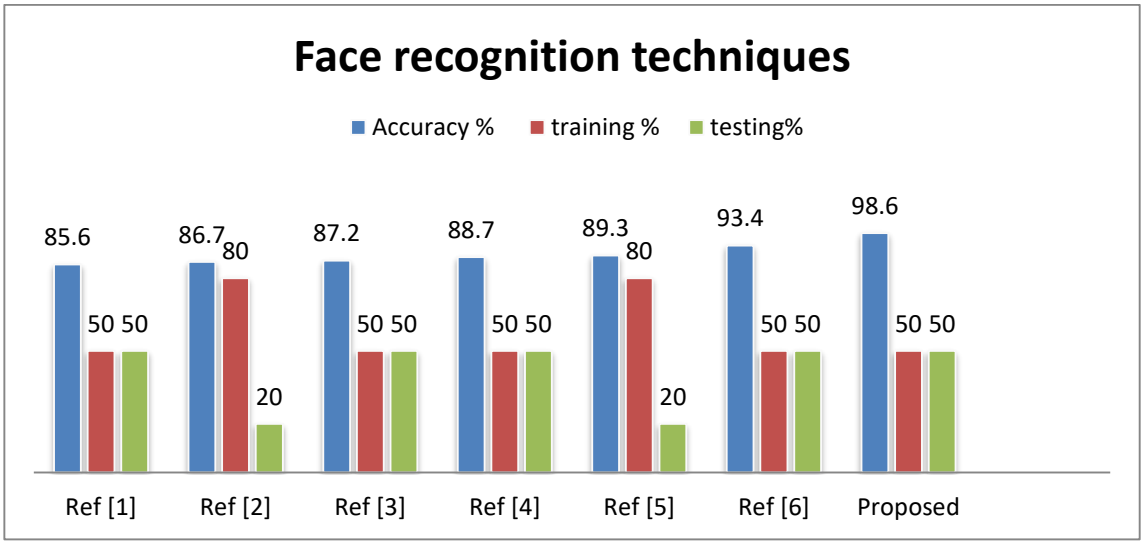

Figure 6: Analysis of Different types of face recognition techniques

Figure 6 Shows the Comparison of different types of algorithms in past years to solve face recognition problems, the proposed method Alexnet gives more accuracy with respect to previous methods. In existing systems uses LBPH and BPNN for face recognition, 50% of data is used to train the data set and remaining data set for testing; it produces 93.4% percent accuracy. Coming to proposed technique training of data set is taken as same with respect to overcome the drawback of exiting system, similarly 50% of training and remaining for testing is given to Alexnet for classification, finally it get 98.6% of accuracy.

Table 2: Comparison of existing methods with proposed method

DATA SET CONSIST OF 400 IMAGES				
S.no	**Author**	**Model**	**No of training image**	**%Accuracy**
1	M. A. Lone, S. M. Zakariya, and R. Ali [1]	CASNN, FFNN	200	86,88
2	C.-L. Fan, X.-T. Chen, and N.-D. Jin [2]	WT+PCA	320	87
3	T. Ayyavoo and J. S. Jayasudha [3]	FKNN	200	89
4	F. A. Bhat and M. A. Wani [4]	PCA,LDA,DCT,ICA	200	85,88,91,87
5	W. Ge, W. Quan, and C. Han [5]	LDA	320	89.5
6	Ausif Mahmood [6]	LBPH, Multi-KNN, BPNN	200	95.5
7	Proposed	CNN, Alexnet	200	98.5

The above table shows the following existing systems regarding to different techniques

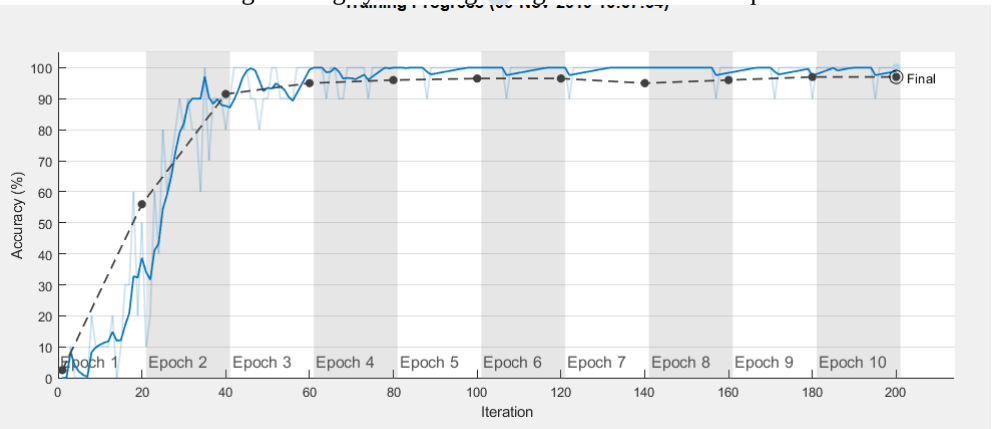

Figure 7: Result Accuracy

In the above figure 7 shows about accuracy of the proposed method with respect to training and testing, training dataset of 50% is given regarding to previous existing method, to check and comparing the accuracy with respect to proposed technique, here the data set consist of 400 images with 40 different faces/persons, 200 images are given for training and remaining for testing with this 98.3% accuracy is gained.

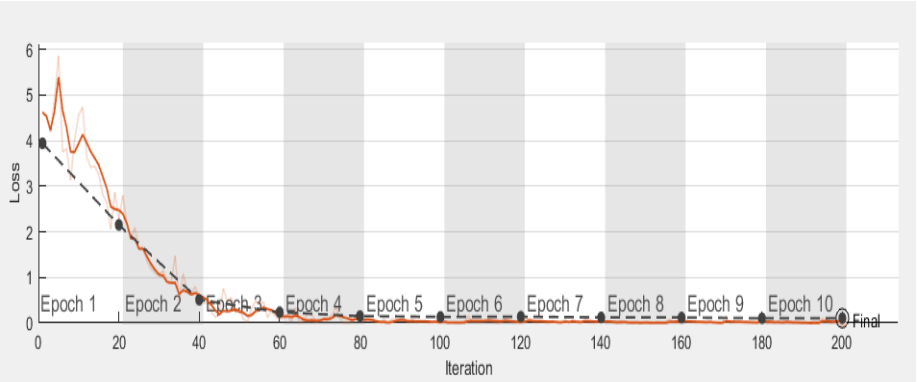

Figure 8: Loss of accuracy

Loss accuracy of the given tested data is acquired 1.4% out-off given 200 images, and bellow figure shows different training set to get different accuracy values, here calculating the data sets with respect to 100, 200,250,300, and 350 images as training.

Similarly, calculating of different values in training and testing gives more accuracy,

1. Accuracy is 95.7% when the training of images is 100.
2. Accuracy is 96% when the training of images is 150.
3. Accuracy is 98.6% when the training of images is 200.
4. Accuracy is 98.75% when the training of images is 250.
5. Accuracy is 99.8% when the training of images is 300.
6. Accuracy is 100% when the training of images is 350.

S.no	Training	Testing	Accuracy%	Loss%
1.	100	300	95.7	4.3
2.	150	250	96	4
3.	200	200	98.6	1.4
4.	250	150	98.75	1.25
5.	300	100	99.8	0.2
6.	350	50	100	0

Table 3: Classification of dataset

Finally the output waveform shows Accuracy and Loss,

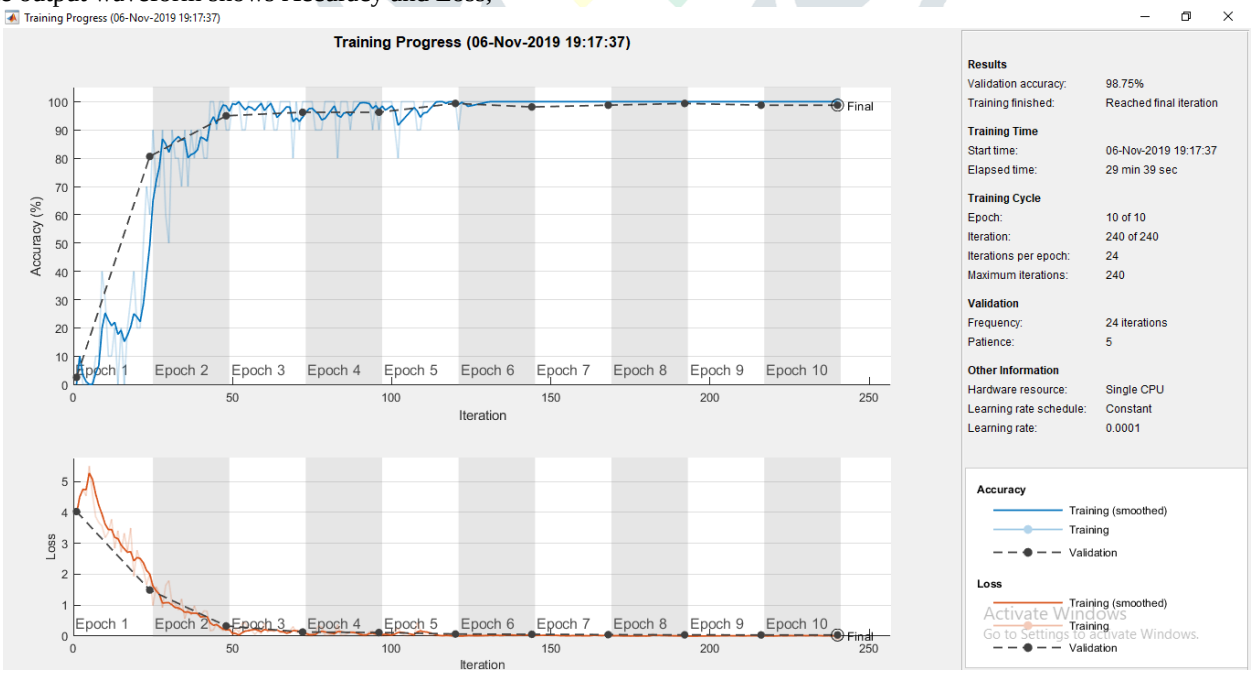

Figure 9: Final output in face recognition

The bellow figure shows the recognition of different faces with respect to different persons, and displays like figure recognition completed

CONCLUSION:

Disguised face recognition is an interesting and challenging task. Not much work has been done in recognizing people who hide their identity by covering their face with scarves using deep learning methods. Now it evaluated the extracted learned features from a pre-trained Convolutional Neural Network (Alex-Net) followed by ORL data sets. The detected and pre-processed face image is fed as an input to the CNN (Alex-Net). The proposed system is tested on a newly introduced disguise dataset and a high accuracy rate is achieved through MATLAB R2018a. Accuracy can be further improved by increasing the number of images in the dataset or by using a much deeper convolutional neural network.

Reference:

[1] M. A. Lone, S. M. Zakariya, and R. Ali, ``Automatic face recognition system by combining four individual algorithms," in Proc. Int. Conf. Comput. Intell. Commun. Netw., Gwalior, India, 2011,
[2] C.-L. Fan, X.-T. Chen, and N.-D. Jin, ``Research of face recognition based on wavelet transform and principal component analysis," in *Proc. 8th Int. Conf. Natural Comput.*, Chongqing, China, 2012,
[3] T.Ayyavoo and J.S.Jayasudha, ``Face recognition using enhanced energy of discrete wavelet transform," in Proc. Int. Conf. Control Commun. Comput. (ICCC), Thiruvananthapuram, India, Dec. 2013,
[4] F.A.Bhat and M.A.Wani, ``Performance comparison of major classical face recognition techniques," in *Proc. 13th Int. Conf. Mach. Learn. Appl.*,Detroit, MI, USA, 2014,
[5] W.Ge, W.Quan, and C.Han, ``Face description and identi_cation using histogram sequence of local binary pattern," in *Proc. 7th Int. Conf. Adv. Comput. Intell. (ICACI)*, Wuyi, China, Mar. 2015,
[6] Ausif Mahmood "Enhanced Human Face Recognition Using LBPH Descriptor, Multi-KNN, and Back-Propagation Neural Network" in March 15, 2018, accepted April 7, 2018, date of publication April 10, 2018,
[7] G. Righi, J. J. Peissig, and M. J. Tarr. "Recognizing disguised faces", Visual Cognition, 20(2):143–169, 2012.
[8] T. I. Dhamecha, R. Singh, M. Vatsa, andA. Kumar. "Recognizing disguised faces: Human and machine evaluation"(2014),
[9] Parama Bagchi, Debotosh Bhattacharjee and Mita Nasipuri "Robust 3D face recognition in presence of pose and partial occlusions or missing parts" International Journal in Foundations of Computer Science & Technology (IJFCST), Vol.4, No.4, July 2014
[10] Karl Weiss, Taghi M. Khoshgoftaar and DingDing Wang, "A survey of transfer learning", Journal of Big data (2016)3:9
[11] Patterson, K. E., & Baddeley, A. D. (1977). "When face recognition fails". Journal of Experimental Psychology: Human Learning and Memory, 3, 406 417.
[12] Ramanathan, N.; Chowdhury, A. & Chellappa, R. (2004). "Facial similarity across age, disguise, illumination and pose", Proceedings of International Conference on Image Processing, Vol. 3, pp. 1999–2002
[13] Martinez, A. & Benavente, R. (1998). The AR face database. Computer Vision Center, Technical Report.
[14] Alexander, J. & Smith, J. (2003). "Engineering privacy in public: Confounding face recognition, privacy enhancing technologies", Proceedings of International Workshop on Privacy Enhancing Technologies, pp. 88–106
[15] Cox, I.J.; Ghosn, J. & Yianilos, P.N. (1996). "Feature-based face recognition using mixture distance", Proceedings of International Conference on Computer Vision and Pattern Recognition, pp. 209-216
[16] P. Viola and M. J. Jones, "Robust real-time face detection," International journal of computer vision, vol.57, no. 2, pp. 137-154, 2004.
[17] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, 2005, vol. 1, pp. 886-893: IEEE.
[18] S.S.Farfade, M. J. Saberian, and L.-J. Li, "Multi-view face detection using deep convolutional neural networks," in Proceedings of the 5th ACM on International Conference on Multimedia Retrieval, 2015, pp. 643-650:ACM.
[19] K.Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint face detection and alignment using multitask cascaded convolutional networks," IEEE Signal Processing Letters, vol. 23, no. 10, pp. 1499-1503, 2016.