Face Recognition Based on Shallow Convolutional Neural Network Classifier

Roshan Shrestha
Department of Computer Engineering
Pokhara University
Kaski, Nepal
roshanshrestha01@gmail.com

Sanjeeb Prasad Panday (PhD)
Department of Electronics and Computer Engineering,
Pulchowk Campus, Institute of Engineering.
Tribhuvan University
Kathmandu, Nepal
sanjeeb@ioe.edu.np

ABSTRACT

Fast and accurate user identification and verification is always desirable. Face recognition, which is machine recognization of person face by analysing patterns on facial features is becoming important for security and validation. Less interaction from user contributes high enrolment as well as easily applicable for current technology further adds its importance. In this regard, we propose a Convolutional Neural Network (CNN) based face recognition technique previously done with eigenfaces[8] but CNN has better accuracy. Entire process is divided into four phases: capturing the image, features extraction, classification and matching. At&t faces dataset is used in this paper. Input images are first fed for face detection. Face detection in input images are performed using Viola Jones algorithm. Convolutional Neural Network (CNN) is applied for feature extraction and classification. The result obtained in this paper shows that - recall is 0.992, precision is 99.4, f1 score is 99.1 and f1 beta score is 0.992 for 70-30 split of dataset, i.e, 70% dataset used as training dataset and 30% as testing dataset.

CCS CONCEPTS

• **Computing methodologies** → **Image and video acquisition**; **Machine learning approaches**; **Computer graphics**

KEYWORDS

Face detection; Viola Jones; Face recognition; Convolutional Nerural Network; Face recognition application

IVSP 2020, March 20–22, 2020, Singapore, Singapore
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-7695-2/20/03…$15.00
DOI: https://doi.org/10.1145/3388818.3388825

1 INTRODUCTION

Human face has a variety of features which help to create a unique identification. These features not only help to recognize but to read and understand a person. Face recognition is coming to lime light as machine learning is growing so much in the past few years. Currently, simple face recognition can be found in examples like unlocking a mobile device with face, but has great potential in validation and identification of person in terms of security. Some of the facial features can be distance between eyes, width of nose, depth of eye socket, cheekbones, jaw line etc.
Face recognition includes face identification and verification. On one hand, face verification is where face is validated either accepting or rejecting on identity claimed with one-to-one matching. On the other hand, face identification is to identify a person based on digital face images included in dataset. The entire process is carried out in four steps: 1. Image acquisition 2. Image processing/Feature extraction 3. Classification 4. Matching and non-matching. Face recognition uses existing image acquisition equipment. It is easy to use as less involvement of user to verify, search against static digital image, inexpensive and can be carried out in mass to check security.
Some of its application:

- Retailors can generate data on customers, tracking shopping habits and target store ads.
- Attendance can be monitored in colleges and schools.
- Face recognition in streets to locate and identify missing persons, wanted criminal, etc.

CONVOLUTIONAL NEURAL NETWORK

Convolutional Neural Network (CNN) is feed forward neural network that is generally used to analyse visual image by processing data in grid like topology. CNN is also known as *"ConvNet"*. CNN has three layers which are input layer, hidden layer and output layer. The input layer connected to hidden layer in called local receptor field as shown in Figure 1
CNN hidden layer is futher divided into:

- Convolutional Layer

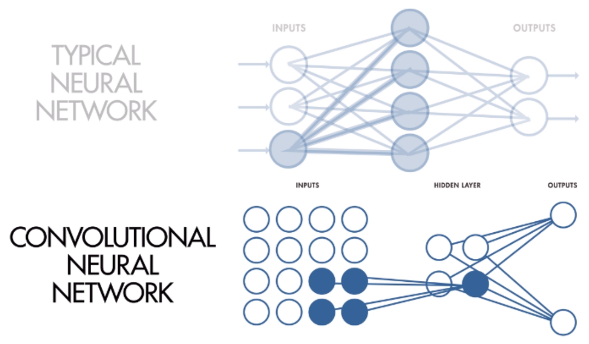


Figure 1. CNN classifier topology

- ReLU
- Pooling Layer
- Dropout Layer
- Fully connected Layer

2 LITERATURE REVIEW

R. Dellana and K. Roy [4] evaluated performance between Relevance Vector Machine (RVM) and Support Vector Machine (SVM) with conclusion that RVM is faster than SVM in testing phase, but predictive behaviour is similar. Their report also suggest that feature extraction before hand results in better accuracy. Feature extraction is important part in image processing. In report purpose by Karthik H S and Manikandan J [3], face recognition is carried out in real-time with Histogram of Oriented Gradients (HoG) as feature extraction and RVM classifier with promising results.
There are many possible classifier to choose for classification but CNN has better accuracy for image based classification which K. Simonyan and A. Zisserman [7] demonstrated at ImageNet Challenge 2014. Further report presented by R. Dellana and K. Roy [1] supports CNN as better classifier. They used two datasets and found that CNN has accuracy higher than Local Binary Pattern Histogram (LBPH) and Eigenface but tends to overfit in less training datasets.
In this paper, we propose face recognition using shallow CNN classifier. This proposed method using CNN shows higher accuracy and precision results rather than methods described by Karthik H S and Manikandan J [3] in same 70-30 split dataset.

3 PROPOSED TECHNIQUE

Face recognition is carried out in four phases. In first phase, image is captured by any image capturing devices such as webcam, smart phone camera etc. For second phase, features of captured images are extracted. CNN [5] is used as classifier in third phase to identify face of a person. At last, for fourth phase, the output is given as matched or un-matched face where False Rejection Rate (FRR) and False Acceptance Rate (FAR) along with performance and accuracy is calculated. Figure 2 further breaks down the process in steps:

- The ORL Database of Faces is collected for training.
- Face detection or image capturing from any image capturing device.
- Feature extraction of captured image with CNN.
- Identify face from classification using CNN.
- New data for testing model.
- Face recognition for new face input to the model.

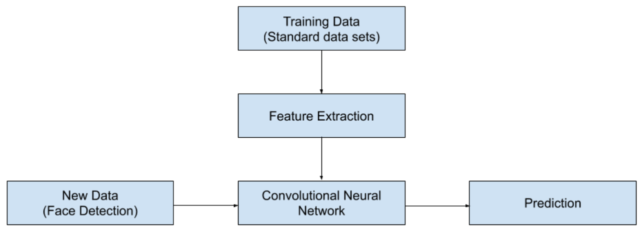


Figure 2. Face recognition methodology based on CNN classifier

DATA SETS

The ORL Database of Faces[6] is considered as standard images. These images consist of 400 faces with 40 individual subjects and 10 distinct face images. These set of faces were taken between April 1992 and April 1994 at the lab. Images are taken at different times, varying the lighting, facial expression and facial details (glasses/no glasses). All the sample images are taken at dark homogeneous background with the subjects in an upright, frontal position. Sample of image can be seen in figure 3. All sample images are in PGM format.

TRAINING THE MODEL

In this paper, data sets are divided as 60% of data for training and 40% for testing. Dataset will be divided in two groups training and testing dataset where model will be trained only in training dataset keeping test data set untouched and evaluated in test dataset. Model will be run in testing dataset for better results. Second experiment is conducted with 70% dataset for training and 30% for testing

DATA PREPROCESSING

Images in standard datasets are in grayscale. If the image from a source are not in gray scale, those image are converted in grayscale. After conversion of image into grayscale, Region of Interest (ROI) of image is extracted with haar cascade and pipe for further stages.

BUILDING THE NETWORK

After all data are loaded and preprocessing is done. Object detection in this domain: face detection is carried out with

Figure 3: The ORL Database of Faces.

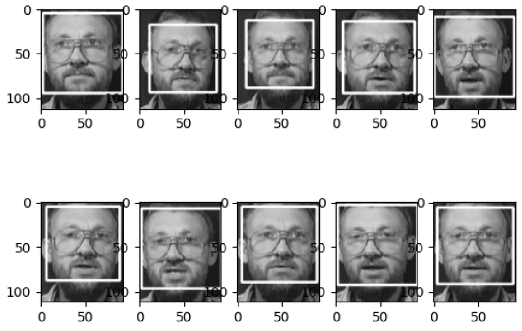


Figure 4. Haar face detection

haar feature-based cascade classifier proposed by Paul Viola and Michael Jones in [9]. Convolutional Neural Network (CNN) is defined. Firsty, all convolutional layer are defined. Network consists of three convolutional layers. The first convolutional layer takes input image and output stack of 16 feature maps. Each convolutional layer doubles the depth of output until it reach depth of 64. Here we move depth from 1 to 16 to 32 to 64. Each layer uses convolutional kernel size of 3x3 and padding of 1. Then, we define max pool layer which down samples x and y by 2. Dropout layer with probability of 0.2 is added to prevent overfitting. Network also has two fully connected layers. First fully connected layer takes fully downsized stack of feature maps. Eg 48x48 sized image is downsize two time by max pool. So, last convolutional layer output 6x6 sized image with 64 depth. Hence, first fully connected layer input is 6x6x64 results to 2304 inputs which outputs 1024 outputs. Now, these 1024 outputs is fed to final classification layer as 1024 inputs and produce 40 class scores. Here, 40 is the number of sample to be predicted.
We apply the network by sequence of convolutional layer and pooling layers. Input image is given to first convolutional layer. We apply relu activation function then pooling layer. This process is also followed for second and third convolutional layer. Final result of convolutional layer is flattened to vector shape. Dropout layer is added between flattened output and first fully connected layer. After dropout layer, feature vector is passed to first fully connected layer with input of size 2304 after which relu activation function is applied. After that one more dropout layer and inputs is given to final fully connected layer which results to 40 class scores.

4 RESULT & DISCUSSION

Experiment I

ORL database of faces image is given as input to CNN as discussed in section 3 with output 40 class scores. Data samples are shuffled to each epoch to reduce overfitting. The model is trained in 16GB RAM, Quad core personal CPU.

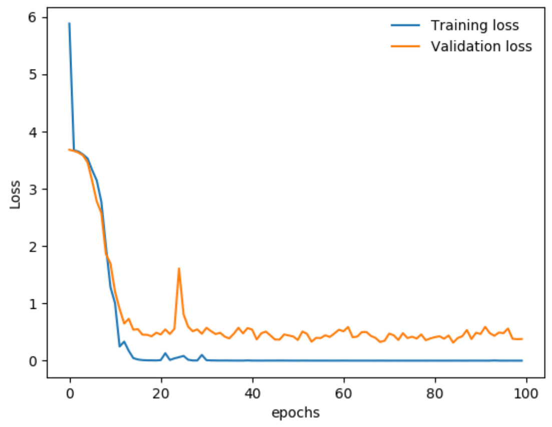


Figure 5. Training and Train Loss

Criterion and Optimizer are initialized, Cross Entroypy loss is calculated for criterion which perform LogSoftmax and Negative Log Loss also call NLLLoss. Cross Stochastic gradient descent (SGD) is set as an optimizer which is responsible for updating weights in network. Both criterion and optimizer model are initialized. 100 epochs is set for training. In each epoch, batch size of 10 batch is passed. In every epoch, training loss, test loss and accuracy is calculated.
In each epoch, image feature is given as input to model that return class score which is predicted output. Criterion initialized earlier compares predictive output and true labels in each epoch. Hence, Cross Entropy loss is given from predictive output and labeled. Loss is then backpropagated followed by SGD optimization to update network weights. Training loss in added in every batch and after finishing an epoch, average training loss is calculated by dividing accumulated taining loss with length of training dataset.
Network accuracy is also tested after finishing each epoch by loading sample from test dataset. Model trained in each epoch is given image feature from test dataset which returns class probability. Highest probability is returned as class score which corresponds to subject. Criterion of model prediction

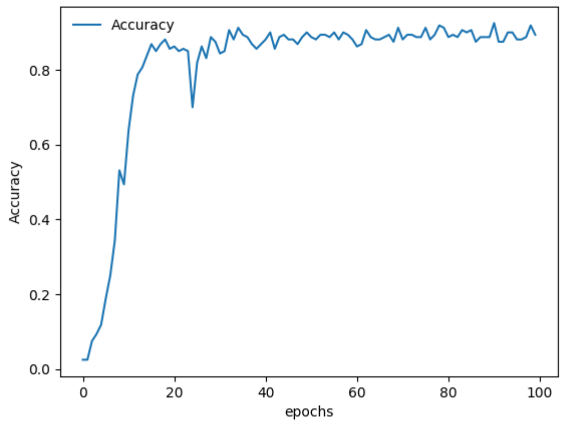


Figure 6. Accuracy at each epoch

of test dataset is computed to track validation loss. Training and validation loss is shown in figure 5.

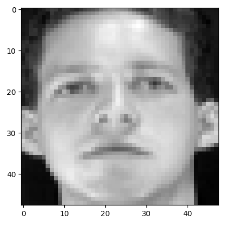


Figure 7. S1 subject

Here, sample s1 of ORL face database in figure 7 gives class score as output in 10, 65 and 97 epochs. This is shown in figure 8, 9 and 10 respectively. The output shows accuracy increasing as training of model progresses. Now we have list of match and unmatch. Therefore accuracy is calculated by taking mean. Accumulated validate loss and accuracy in each batch is computed and average taken by dividing by length of test datasets. Accuracy of each epoch is given in figure 6

Validation

The validation is done with confusion matrix of all 40 class scores. Predicted labels and truth values are used to calculate recall, precision, f1 score and f1 beta score. These are calculated for every epoch whose validation loss is below the previous one. Here, we save lowest validation loss in a variable and if any next validation loss is lower than saved then recall, precision, f1 score and f1 beta score are calculated. Figure 11 gives the confusion matrix of all 40 subject in ORL

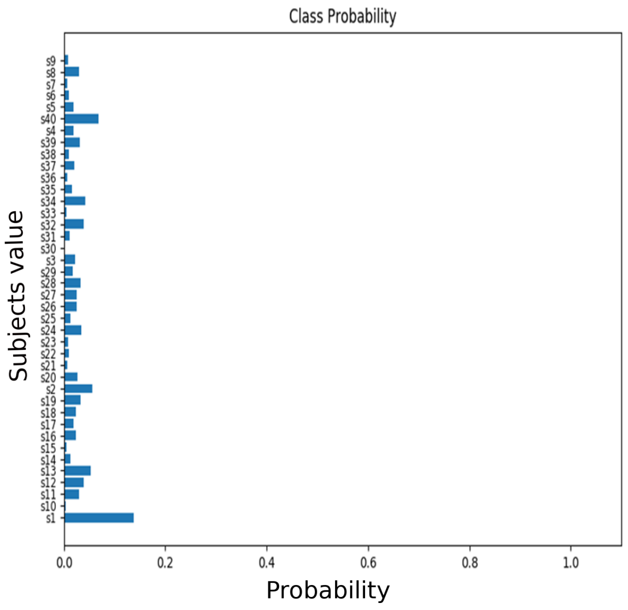


Figure 8: Softmax at 10 epoch

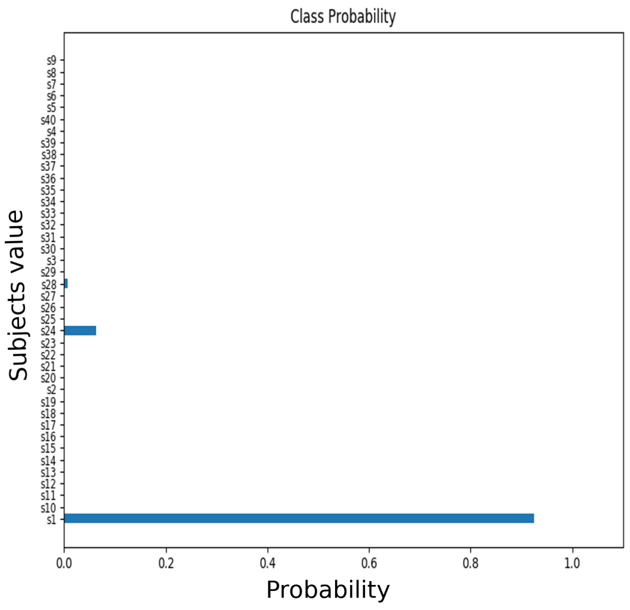


Figure 9. Softmax at 65 epoch

database. Equation 1, 2, 3 and 4 are formulae for calculating Recall, Precision, F1 score and F1 beta score respectively.

$$Recall(r) = \frac{TruePositives}{TruePositives + FalseNegative} \quad (1)$$

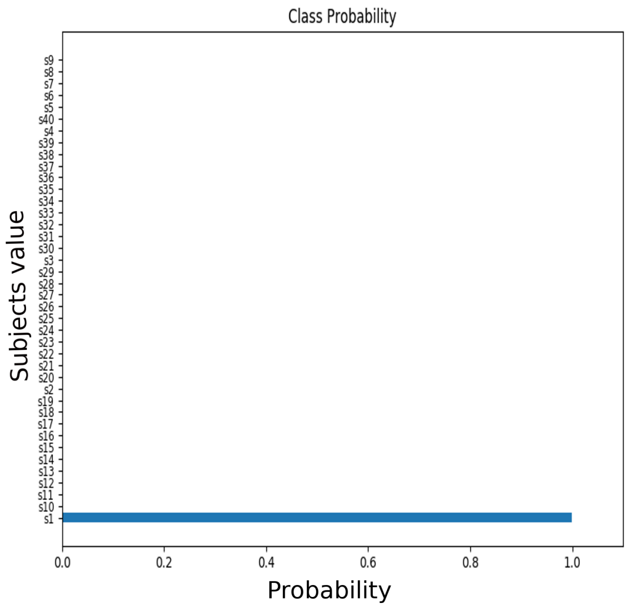


Figure 10. Softmax at 97 epoch

$$Precision(p) = \frac{TruePositives}{TruePositives + FalsePositive} \tag{2}$$

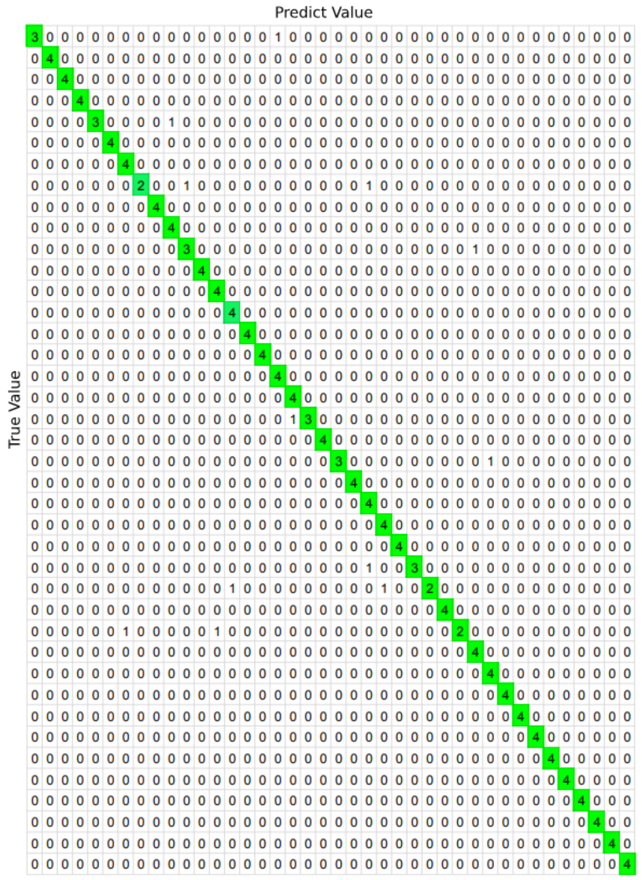
$$F1score = \frac{2 * p * r}{p + r} \tag{3}$$

$$F1_{\text{beta}}score = (1 + \beta)\frac{p * r}{\beta * p + r} \tag{4}$$

The beta parameter determines the weight of precision in the combined score. beta < 1 lends more weight to precision, while beta > 1 favors recall (beta -> 0 considers only precision, beta -> inf only recall).

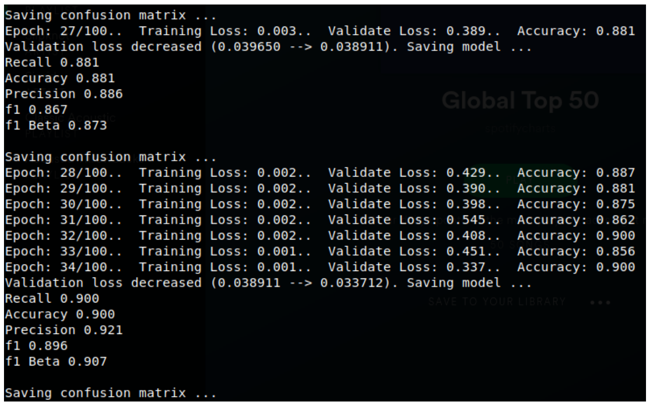
In each 100 epoch; best model i.e when validation loss is lowest is saved with value of recall, precision, f1 score and f1 beta score. In figure 12 validation loss decreases at epoch 27 where validation has value 0.389, after which validation loss doesn't decrease than 0.389 so those model recall, precision, f1 and f1 beta are not calculated. Also model is not saved because it is not necessary as model validation has not decreased. In epoch 34 validation loss decreases to 0.33 so here model is saved again and recall, precision, f1 and f1 beta score are calcuated.

Epoch continues, validation loss decrease to lowest as seen in figure 13 at epoch 67 where validation loss is 0.335. At this poinst, recall is 0.906, precision 0.928, f1 score 0.900 and f1 beta score 0.911


Predict Value
True Value

3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	3	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4

Figure 11. Confusion Matrix for 4 test data

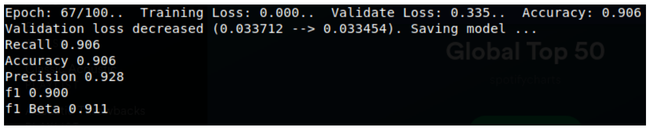


```
Saving confusion matrix ...
Epoch: 27/100..  Training Loss: 0.003..  Validate Loss: 0.389..  Accuracy: 0.881
Validation loss decreased (0.039650 --> 0.038911). Saving model ...
Recall 0.881
Accuracy 0.881
Precision 0.886
f1 0.867
f1 Beta 0.873

Saving confusion matrix ...
Epoch: 28/100..  Training Loss: 0.002..  Validate Loss: 0.429..  Accuracy: 0.887
Epoch: 29/100..  Training Loss: 0.002..  Validate Loss: 0.390..  Accuracy: 0.881
Epoch: 30/100..  Training Loss: 0.002..  Validate Loss: 0.398..  Accuracy: 0.875
Epoch: 31/100..  Training Loss: 0.002..  Validate Loss: 0.545..  Accuracy: 0.862
Epoch: 32/100..  Training Loss: 0.002..  Validate Loss: 0.408..  Accuracy: 0.900
Epoch: 33/100..  Training Loss: 0.001..  Validate Loss: 0.451..  Accuracy: 0.856
Epoch: 34/100..  Training Loss: 0.001..  Validate Loss: 0.337..  Accuracy: 0.900
Validation loss decreased (0.038911 --> 0.033712). Saving model ...
Recall 0.900
Accuracy 0.900
Precision 0.921
f1 0.896
f1 Beta 0.907

Saving confusion matrix ...
```

Figure 12. Saving model when validation loss is lower

Experiment II

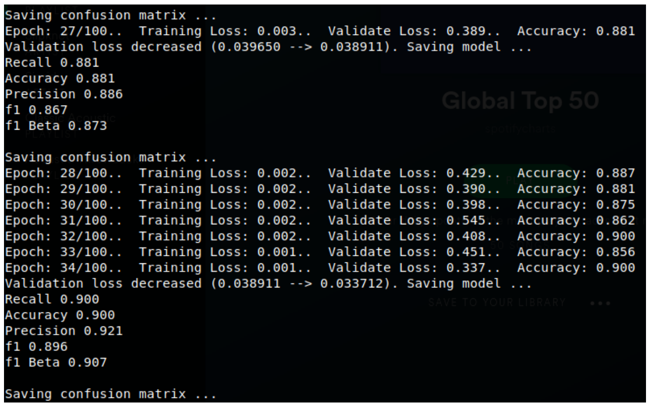
In first experiment dataset is divided into 60-40 ratio. In Experiment II, dataset is divided into 70-30 split. Using prepared script, dataset is easily separated - Separated data is used for traning and testing. Recall, precision, f1 score and f1 beta



```
Epoch: 67/100..  Training Loss: 0.000..  Validate Loss: 0.335..  Accuracy: 0.906
Validation loss decreased (0.033712 --> 0.033454). Saving model ...
Recall 0.906
Accuracy 0.906
Precision 0.928
f1 0.900
f1 Beta 0.911
```

Figure 13. Best model with lowest validation loss

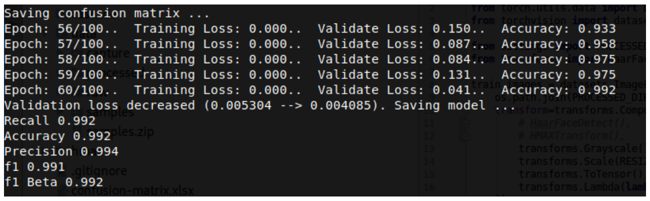
score is calculated. Figure 16 show the confusion matrx for generated model.



```
Saving confusion matrix ...
Epoch: 27/100..  Training Loss: 0.003..  Validate Loss: 0.389..  Accuracy: 0.881
Validation loss decreased (0.039650 --> 0.038911). Saving model ...
Recall 0.881
Accuracy 0.881
Precision 0.886
f1 0.867
f1 Beta 0.873

Saving confusion matrix ...
Epoch: 28/100..  Training Loss: 0.002..  Validate Loss: 0.429..  Accuracy: 0.887
Epoch: 29/100..  Training Loss: 0.002..  Validate Loss: 0.390..  Accuracy: 0.881
Epoch: 30/100..  Training Loss: 0.002..  Validate Loss: 0.398..  Accuracy: 0.875
Epoch: 31/100..  Training Loss: 0.002..  Validate Loss: 0.545..  Accuracy: 0.862
Epoch: 32/100..  Training Loss: 0.002..  Validate Loss: 0.408..  Accuracy: 0.900
Epoch: 33/100..  Training Loss: 0.001..  Validate Loss: 0.451..  Accuracy: 0.856
Epoch: 34/100..  Training Loss: 0.001..  Validate Loss: 0.337..  Accuracy: 0.900
Validation loss decreased (0.038911 --> 0.033712). Saving model ...
Recall 0.900
Accuracy 0.900
Precision 0.921
f1 0.896
f1 Beta 0.907

Saving confusion matrix ...
```

Figure 14. Saving model when validation loss is lower

In each 100 epoch; best model is saved determined by lower validation loss. Figure 14 shows decrease in validation loss as compared to epoch 25 and 30.
As we proceed, lowest validation loss is seen at epoch 60, which is shown in figure 15 In this case accuracy is 99.2, recall is .992, precision is .994, f1 score is .991 and f1 beta score is 0.992.

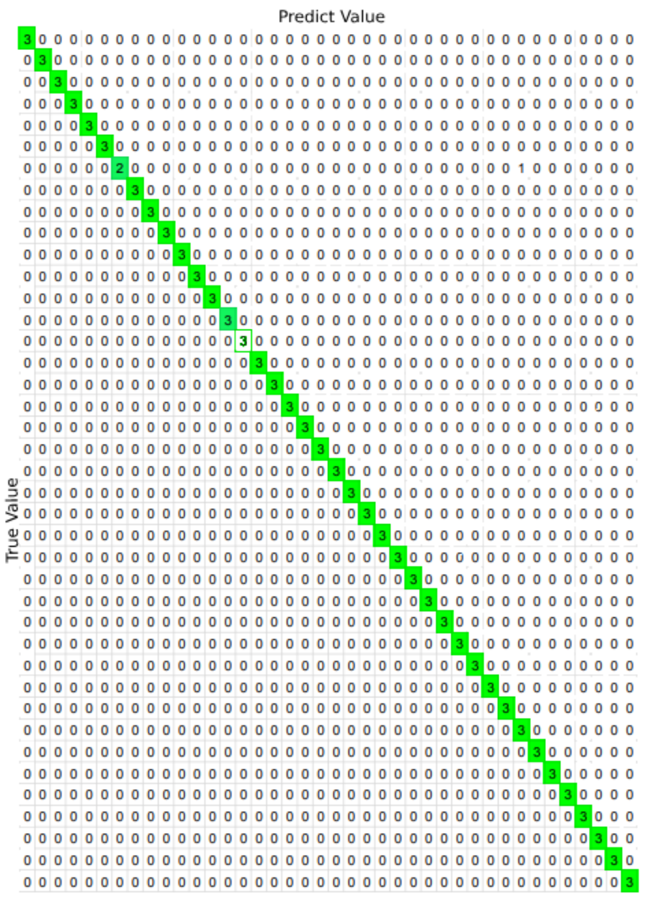


```
Saving confusion matrix ...
Epoch: 56/100..  Training Loss: 0.000..  Validate Loss: 0.150..  Accuracy: 0.933
Epoch: 57/100..  Training Loss: 0.000..  Validate Loss: 0.087..  Accuracy: 0.958
Epoch: 58/100..  Training Loss: 0.000..  Validate Loss: 0.084..  Accuracy: 0.975
Epoch: 59/100..  Training Loss: 0.000..  Validate Loss: 0.131..  Accuracy: 0.975
Epoch: 60/100..  Training Loss: 0.000..  Validate Loss: 0.041..  Accuracy: 0.992
Validation loss decreased (0.005304 --> 0.004085). Saving model ...
Recall 0.992
Accuracy 0.992
Precision 0.994
f1 0.991
f1 Beta 0.992
```

Figure 15. Best model with lowest validation loss

5 IMPLEMENTATION WITH GTK+ APPLICATION

Gtk+ desktop application takes new data from an image capturing input device and stores. After this stored image is transferred to pretrained network from validation. Figure 17 shows initial window of application when launched.
This figure shows button for three major steps; capture image, training model which is used to transfer learning from previously saved model and start predicting video.

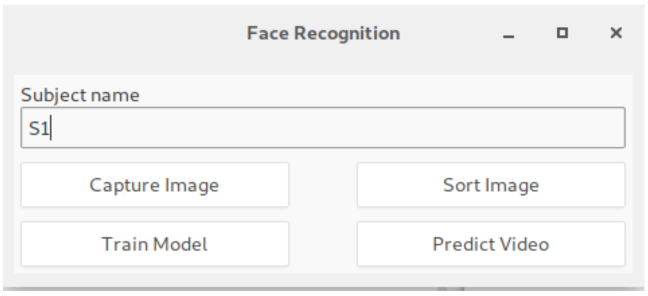


Predict Value

True Value

3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3

Figure 16. Confusion Matrix for 3 test data



Subject name
S1

Figure 17. Initial window on lauch

Capture Image

Before capturing image, we must declare a subject name. Every time when an image is captured, image is saved in capture directory which will be used for transfer learning as a data source in name of subject name. Figure 17 shows subject name is S1 which is to be replaced by the name of subject whose image is to be captured. After subject name is declared, capture image is clicked to open video window.

Advik webcam with f=3.85mm Megapixel webcam is used for inputing video source as laptop webcam resolution was not enough.
Capture image takes a webcam video input showing a detected face with bounded box of blue rectangle. Rectangular border face is saved in gray scale image in capture directory from which transfer learning data is taken. Bounded rectangle box is indication of face detection and start with "Keyboard 'C' key" is pressed to capture face bounded by rectangle and stored in subject name directory. At least 10 images of face in multiple viewing angle is taken to diversify image source of an individual. Figure 18 shows capture image window bounded by blue rectange in video but black in grayscale.

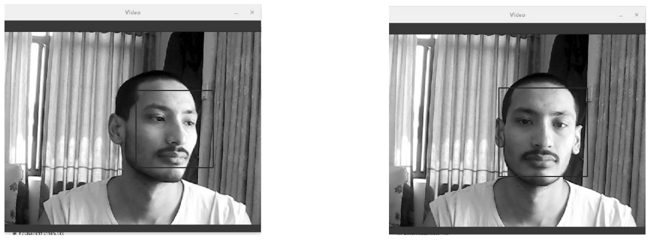


Figure 18. Face detection and capture

Pressing "C" key in keyboard capture bounded box image and save in subject name location. Subject name is set as "roshan" which is label for this image. Capture image example which are stored is shown in figure .

Figure 19. Face detection and capture

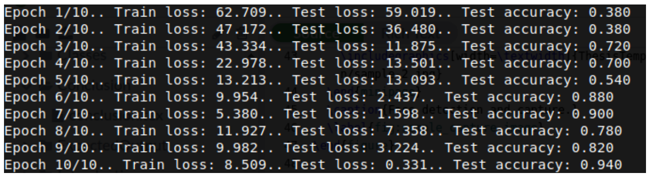
Images are located in capture directory which will be used as data source for transfer learning. Directory structure of capture image is shown in figure 20

Transfer learning

Network trained in ORL database is stored with name orl-database-faces.pt when "Train Model" button is clicked. Pretrained model is loaded and new faces from capture image is used to training model again for new faces. Similarly as previous training and validating model, criterion and optimizer is initialized, cross entropy loss is calculated to perform LogSoftmax. 10 epoch is set for transfer learning. Train loss and test loss is calculated in each epoch as shown in figure 21.
10 epoch is run in background and is visible only in shell. Once training is completed, gtk application is notified with an alert as shown in figure 22 After training is done with new data set from capture directory, predict video window can be launched.

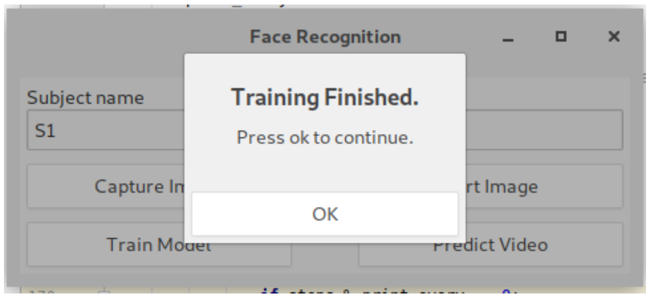
```
data
└─ capture
│  └─ roshan
│  │  └─ 1.jpg
│  │  └─ 2.jpg
│  │  └─ ...
│  │  └─ 10.jpg
│  └─ ankit
│  └─ subject-name
└─ raw
└─ processed
```

Figure 20: Capture dataset storage.



```
Epoch 1/10.. Train loss: 62.709.. Test loss: 59.019.. Test accuracy: 0.380
Epoch 2/10.. Train loss: 47.172.. Test loss: 36.480.. Test accuracy: 0.380
Epoch 3/10.. Train loss: 43.334.. Test loss: 11.875.. Test accuracy: 0.720
Epoch 4/10.. Train loss: 22.978.. Test loss: 13.501.. Test accuracy: 0.700
Epoch 5/10.. Train loss: 13.213.. Test loss: 13.093.. Test accuracy: 0.540
Epoch 6/10.. Train loss: 9.954.. Test loss: 2.437.. Test accuracy: 0.880
Epoch 7/10.. Train loss: 5.380.. Test loss: 1.598.. Test accuracy: 0.900
Epoch 8/10.. Train loss: 11.927.. Test loss: 7.358.. Test accuracy: 0.780
Epoch 9/10.. Train loss: 9.982.. Test loss: 3.224.. Test accuracy: 0.820
Epoch 10/10.. Train loss: 8.509.. Test loss: 0.331.. Test accuracy: 0.940
```

Figure 21. Training model for new dataset



Training Finished.
Press ok to continue.
OK

Figure 22. Alert for completion of training

Predict Video

Predict video window is lauched when "Predict Video" button is clicked. This opens up a window whose input is from Advik webcam with f=3.85mm Megapixel webcam. When taking a video frame face detection is done by Haar Cascade and that image is predicted with trained model. Outcome is mapped to dataset value and result is added to bounded box of face as shown in figure 23

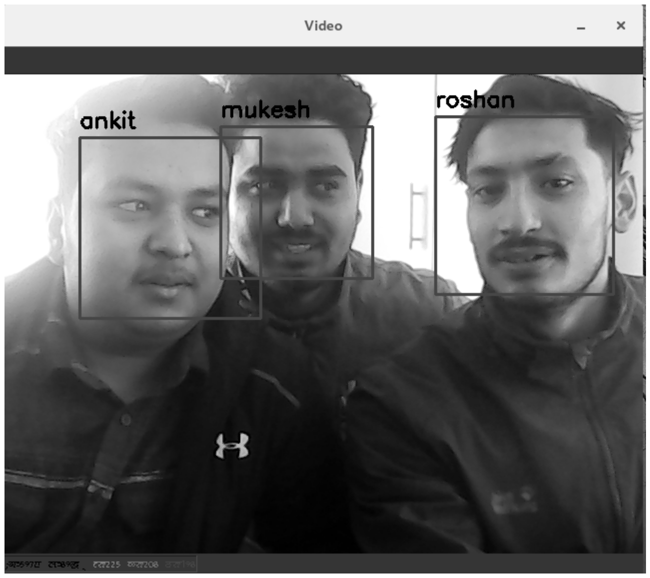


Figure 23. Predict face in video frame

6 COMPARISION

Karthik H S and Manikandan J [3] concludes its accuracy in 90.00%, 94.50% and 97.00% for One Against All, Hierarchical Tree and Half Against Half method respectively with ORL database and RVM as classifire where as our proposed method has higher accurracy up to 98% to 99% accuracy in 70-30 split of dataset.
This model performs better than network presented by Yan, Kewen and Huang, Shaohui and Song, Yaoxian and Liu, Wei and Fan, Neng [10]. In this paper data split of 70-30 gives us about 99.2% accuracy where as purpose method [10] gives 99.82% in 90-10 split which in compare has way less testing data than training data.
Also in paper presented in 2017 by Patrik KAMENCAY, Miroslav BENCO, Tomas MIZDOS, Roman RADIL [2] accuracy is about 98.3% in 80-20 datasplit which also in compare has less testing dataset. Method purposed in this paper has about 99% accuracy in 70-30 split.

7 CONCLUSION

In this paper, face recognition is carried out with CNN as classifier. Accuracy is above 90% as shown in figure 6. ORL database of faces is used for training purposes and validation whereas image capture from webcam is used for verifciation. Face recognition steps: capturing, extraction, classification and match and non-match is carried out. Recall, precission, f1 score and f1 beta score concludes classification is at average 90 as shown in figure 13. This model can be used for other classification problems as well.

Object detection i.e. face detection is done with Haar cascade classifier which- has limitation to detect face and this can be improved with higher object detecting network such as Faster RCNN, Masked RCNN. Better face detection can contribute in high feature extraction and better classification. CNN is used here in simple shallow network which limits classification accuracy. For better accuracy and increased performance, deeper CNN such as resnet, vggnet, densenet, alexnet etc can be used.

8. REFERENCE

[1] R. Dellana and K. Roy. “Data augmentation in CNN-based periocular authentication”. In: *2016 6th International Conference on Information Communication and Management (ICICM), Hatfield, United Kingdom* (2016).

[2] Patrik Kamencay et al. “A new method for face recognition using convolutional neural network”. In: (2017).

[3] HS Karthik and J Manikandan. “Evaluation of relevance vector machine classifier for a real-time face recognition system”. In: (2017).

[4] D.D. Nguyen and H.S. Le. “Kinect Gesture Recognition: SVM vs. RVM”. In: *015 Seventh International Conference on Knowledge and Systems Engineering (KSE), Ho Chi Minh City, Vietnam* (2015).

[5] Waseem Rawat and Zenghui Wang. “Deep convolutional neural networks for image classification: A comprehensive review”. In: *Neural computation* (2017).

[6] Sohini Roychowdhury and Michelle Emmons. “A survey of the trends in facial and expression recognition databases and methods”. In: *arXiv preprint arXiv:1511.02407* (2015).

[7] K. Simonyan and A. Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: (2014).

[8] Matthew A Turk and Alex P Pentland. “Face recognition using eigenfaces”. In: (1991).

[9] Paul Viola and Michael Jones. “Rapid Object Detection using a Boosted Cascade of Simple Features”. In: (2001).

[10] Kewen Yan et al. “Face recognition based on convolution neural network”. In: *2017 36th Chinese Control Conference (CCC)*. IEEE. 2017, pp. 4077–4081.