Learning Based Security System

A Project Report Submitted
in Partial Fulfillment of the Requirements
for the Course
EE617
Industrial Automation Control

by

ANKISH BANSAL 17204004

BHARAT GUPTA 17204009

to

Prof. Nishchal K Verma

DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY,KANPUR

NOVEMBER 2018

Contents

List of Figures

1 Introduction

1.1 Motivation

In the traditional neural network, it require millions of examples of one class to extract(learn) the feature of the data-set and collecting such a large data-set is a challenging problem. So learning from few examples is an exciting area to work during this project. We will building deep network model on the top of arduino hardware, which excite us to deal with human like ability to learn and adapt fast for any underlying distribution of data..

In the last decades, there has been very promising results using CNN based model over techniques used in computer vision from the past 50 years. This is another motivation to choose the leaning based model in this project.

1.2 Organization of the Report

First we explain the idea behind few shot learning, which is followed by the Literature Review in face recognition. We explain the approach has been followed in the deep-learning community. Then we explain our approach and model architecture. Then we show little demo of our data-set and software which we used in expriment. After that we show our experiments results, followed by limitation and failure reason. Then we eplained interfacing of Arduino with python. In last we end up with conclusion.

2 Literature Review

As our objective, in this problem to use learning based approach for the security system. To learn from the images, we use Convolutional Neural Network Based Model.

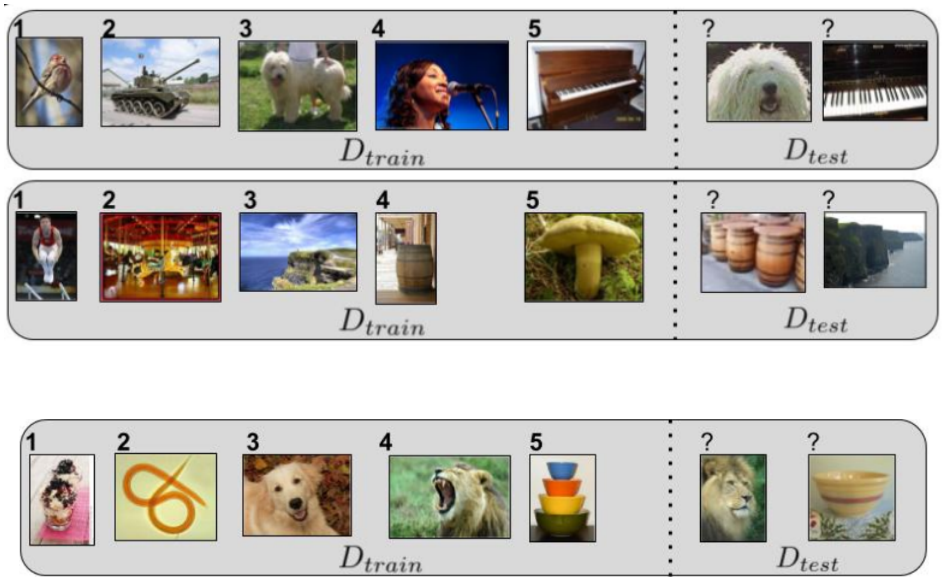
The problem, we are tacking comes under the area of **Few Shot learning**.

2.1 Few Shot Learning

As the name suggest, learn from the few examples. We(humans) have ability to learn from the very few examples. But if our NN have such ability then we can solve many problem in real world . For example in medical field, we cann't collect million of image of disease or bacteria. In such field, having human like ability, can be very helpful.

2.2 Demo of Few shot learning

We are pretty good in identified the image by matching and comparing some feature of two image. As shown in **??** we can pretty quickly analyze that first test image match with third and second test image match with forth. Like-wise we can identified in the second demo example.



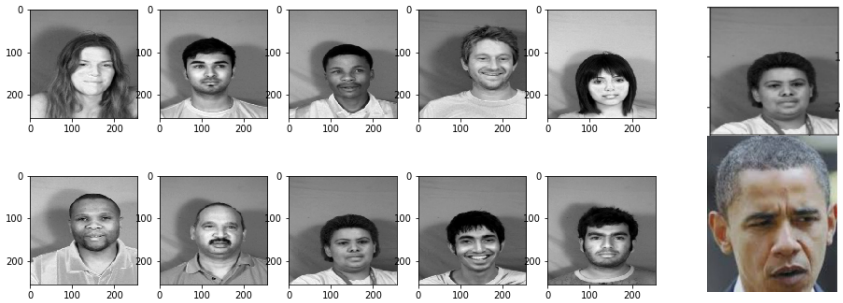


Figure 1: Demo

But this task is pretty difficult for the CNN. Learning such model from $5-10$ image is challenge. It always suffer from **overfitting** in such cases.

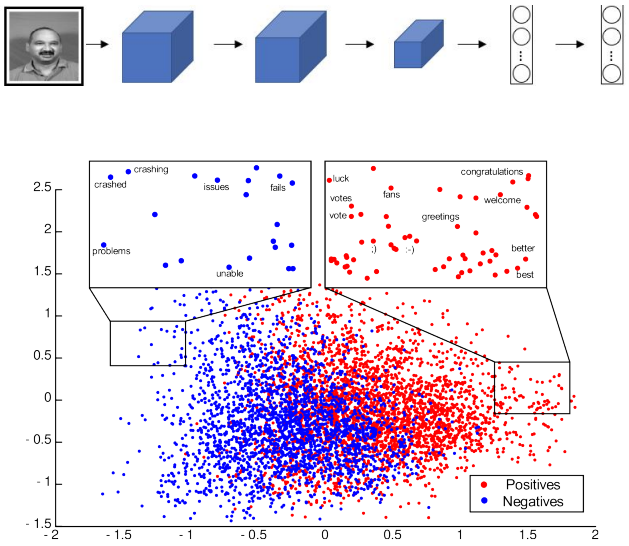
This problem has interested to many researchers and working from the last 4 years. And many researchers has shown powerful research in this domain.

There are two approach for face recognition in literature. using deep learning model. The first is **FaceNet** and second is **DeepFace**. Both are very successful approaches in deep learning field.

The basic idea behind the model is to find the embedding of faces(images). Then using learning objective function, we learn the parameter of the model.

2.3 What is Embedding space

Embedding space represent the hidden feature of the images. If we have image of 100×100 and passing through this input through the model, its output is let's suppose 100×1. This is learned features with dimension $R^{100\times1}$.



2.4 FaceNet

In this approach, the learning objective is to compare the embedding space of the one image with positive and negative sample at the same time and update the weight parameters such that it follow the following condition.

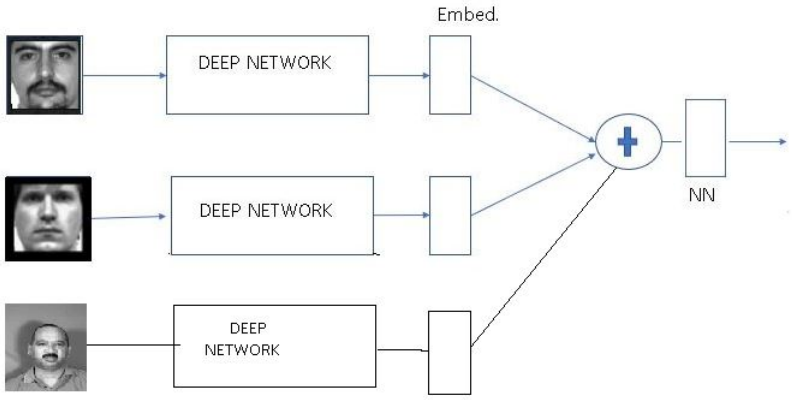


Figure 2: FaceNet Architecture

2.4.1 Notation

A $\rightarrow$ *Anchor* (*Any sample*)
$P \rightarrow$ *Positive w.r.t. Anchor*
$N \rightarrow$ *Negative w.r.t. Anchor*

$$||f(A) - f(P)||_2^2 \leq ||f(A) - f(N)||_2^2 \tag{1}$$

The learning objective for this approach is

$$L(A.P, N) = max(||f(A) - f(P)||_2^2 - ||f(A) - f(N)||_2^2 + \alpha, 0) \tag{2}$$

$$J = \sum_{i=0}^{N} L(A^i.P^i, N^i) \tag{3}$$

where α is margin, such to ensure that boundary between the positive and negative should be far then some positive parameter. In short

$$\mathcal{L} = max(d(a, p) - d(a, n) + margin, 0)$$

2.4.2 Intution behind the model

From figure, it can be easily observe that first the distance between the Anchor and Negative sample is less than the distance between the Anchor and Positive sample. After learning or updating the parameter using the learned objective function, the embedding space has changed what we want.

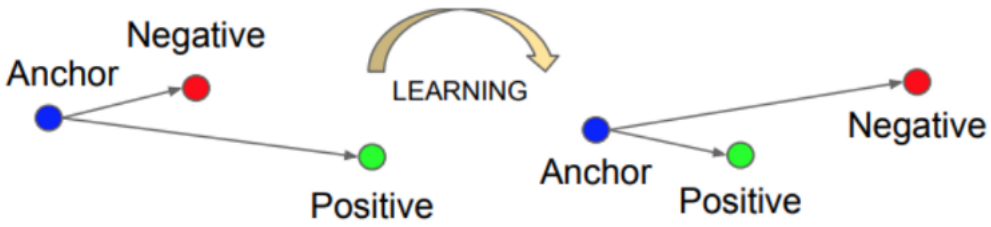


Figure 3: Intution Behind the Learning the Embedding Space

2.5 DeepFace

This approach is much simpler then the previous. This is easy to train and also converges fast. The idea behind this approach is to compare the embedding space using simple euclidean distance between two sample. But a batch consist of half positive sample and half negative samples. The label for the positive samples are 1 and 0 for the negative sample.

We are following this approach in our project. The embedding space work similar as discussed in **FaceNet**. But the problem formulations has changed. Here we compare only two samples at a time.

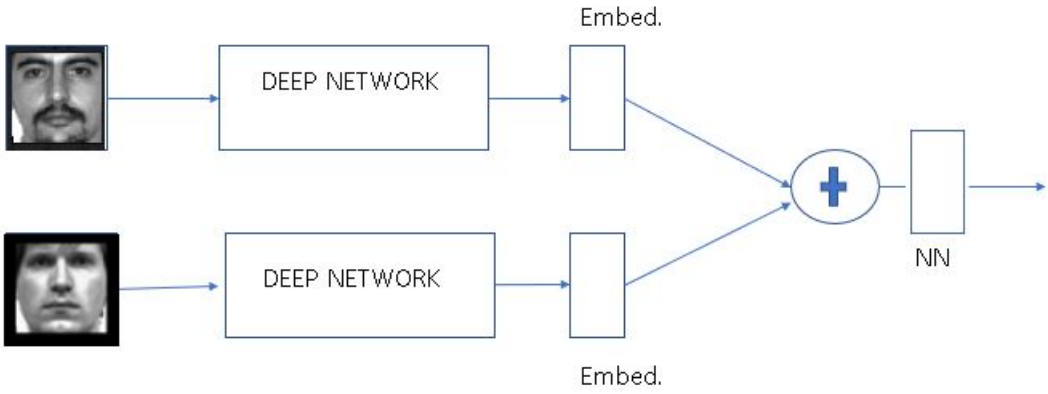


Figure 4: Intution Behind the Learning the Embedding Space

In the above model, we have black box, that is deep network. It can be any network. Following image show that, it can be simple model as Le-net, VGG , RES-Net or it can be as complex as Inception-Resnet model.

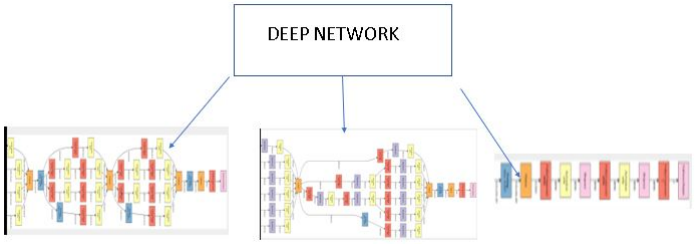


Figure 5: Intution Behind the Learning the Embedding Space

The model, we use in our experiment is simple VGG model, but few changes in the model.

To reduce the number of parameter in the architecture, we focus on Network in Network method. Ituse 1×1 filter followed by 3×3 filte which reduce the number of parameters by multiple of tens times depend on architecture.

3 Arduino(AtMega Microcontroller)

Arduino is an open-source electronics design platform. The Arduino board is specially designed for programming and prototyping with Atmel microcontrollers. An arduino interacts with physical world via sensors. Using arduino electric equipments can be designed to respond to change in physical elements like temperature, humidity, heat or even light. This is the automation process.. There several types of arduino boards.

The open-source Arduino environment allows one to write code and load it onto the Arduino board's memory. The development environment is written in python and based on Processing, AVR-GCC, and other open source software.

Figure 6: Arudino(AtMega Board)

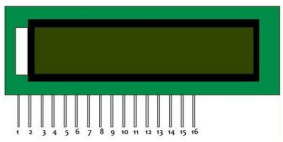


Figure 7: LCD Display

3.1 LCD(Liquid Crystal Display)

Liquid Crystal Display (LCD) screen is an electronic display module. An LCD has a wide range of applications in electronics. The most basic and commonly used LCD in circuits is the 16x2display. LCDs are commonly preferred in display because they are cheap, easy to programmeand can display a wide range of characters and animations.

A 16x2 LCD have two display lines each capable of displaying 16 characters. This LCD has Command and Data registers. The command register stores command instructions given to the LCD while the Data register stores the data to be displayed by the LCD.

3.2 Arduino with LCD Interfacing

A Liquid Crystal Display commonly abbreviated as LCD is basically a display unit built using Liquid Crystal technology. When we build real life/real world electronics based projects, we need a medium/device to display output values and messages. The most basic form of electronic display available is 7 Segment display which has its own limitations. The next best available option is Liquid Crystal Displays which comes in different size specifications. Out of all available LCD modules in market, the most commonly used one is 162 LCD Module which can display 32 ASCII characters in 2 lines (16 characters in 1 line). **now we are going to learn how to interface lcd to arduino**

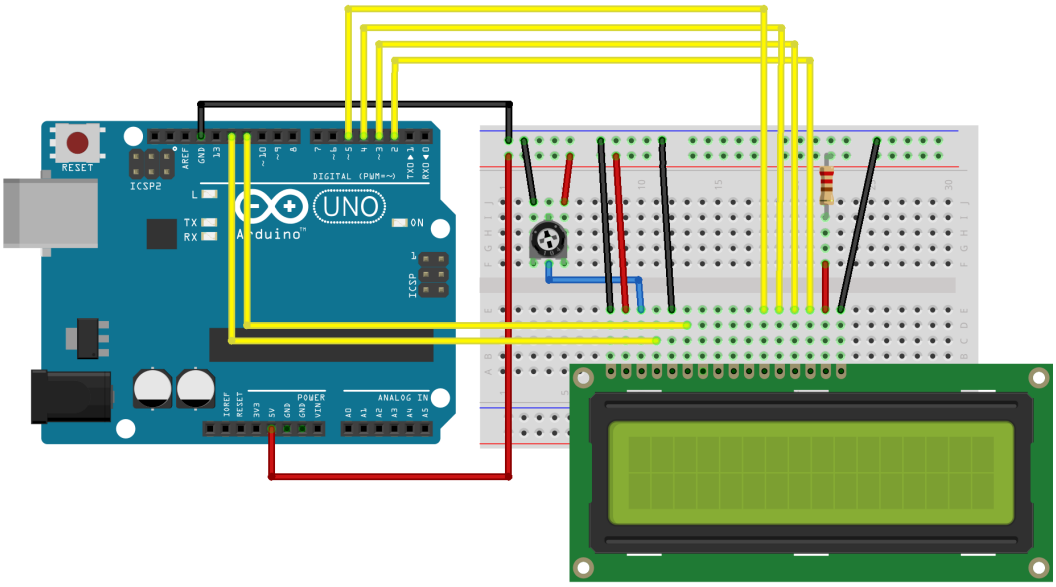


Figure 8: BLOCK DIAGRAM LCD INTERFACING WITH ARUDINO

Pin	Connections
• Digital pin 2	LCD D7 pin
• Digital pin 3	LCD D6 pin
• Digital pin 4	LCD D5 pin
• Digital pin 5	LCD D4 pin
• Digital pin 11	LCD Enable
• Digital pin 12	LCD RS pin
• Digital pin 7	Connection to Water Pump
• Digital pin 8	LED Pin indicating Soggy soil
• Digital pin 9	LED Pin indicating Moist soil
• Digital pin 10	LED Pin indicating Dry soil
• Analog Pin 4	Connection to Soil Moisture Sensor
• VCC	5VDC
• GND	Ground

Figure 9: BLOCK DIAGRAM LCD INTERFACING WITH ARUDINO PIN NO.

3.3 Arduino Interfacing With Python (by Pyserial)

There are several ways to approach Ardunio USB communication, but in this case we will be using Python on the computer side to send and receive information. As such, this instructable expects that you have some prior knowledge of Ardunio, and of Python (or other similar scripting language).

Why Python?

Python is a versatile, easy to learn, and easy to use scripting language. Its power, and huge library of user-created modules (everything from keyboard emulation to game programming) makes it an ideal language for a wide verity of computer side tasks. You could easy parse network information and make an Arduino visualizer, create a game controller, or make a keypad computer login system. Arduino with Python opens up a word of possibilities.

3.3.1 PYTHON CONNECTED TO PYSERIAL

To initiate a connection with the Arduino from Python, we first have to figure out which COM Port the Arduino is on. This task is luckily made simple by the Ardunio programming environment. Simply look in the bottom right corner of your Arduino IDE, and you will see some text containing the COM Port number. We will use this to initiate our Python serial connection, like so:

```
arduino = serial.Serial('COM1', 115200, timeout=.1)
```

Figure 10: Com Port Defined

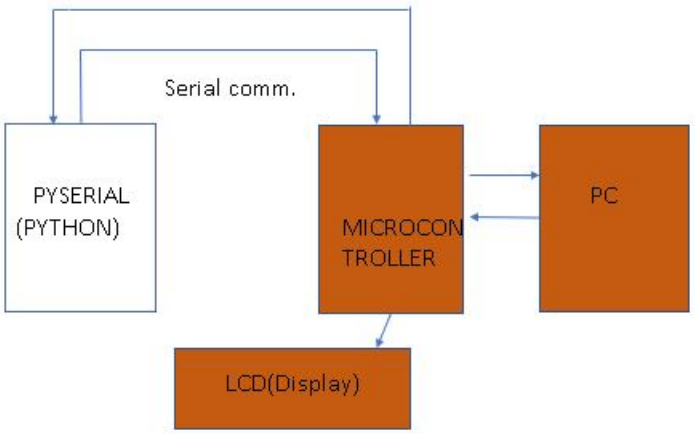
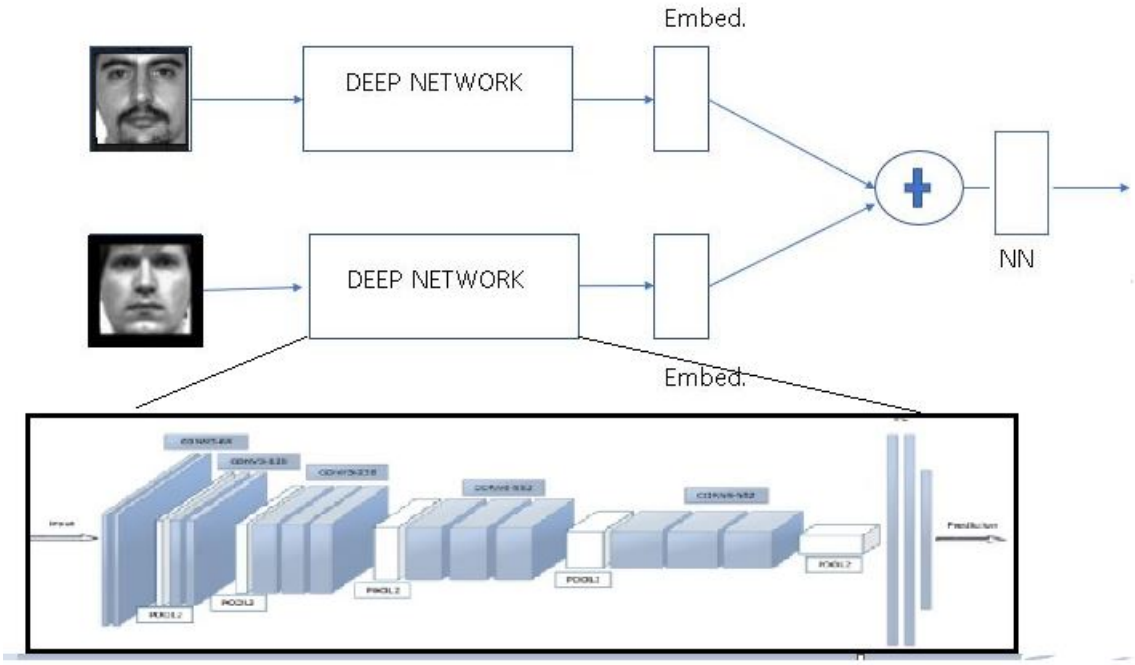


Figure 11: Com Port Defined

The above code will create a new serial object called "ardunio" on "COM1" with a "115200" baud-rate and a .1 second timeout. It is extremely important that you keep the chosen baud-rate on hand, as it must match exactly with the baud-rate on the Ardiuno side of things.

4 Our Approach

As discussed in section ii , we are following **DeepFace** model learning objective to train the model. The model architecture, we followed, in our experiment is shown in following block diagram.



We are using a little complex model, with a total of 14 layers, where we have 10 convolutional layer and 4 dense layer. In total, we have approximated $450,000$ parameters to be learnable in our problem. The complete architecture, we have described in the following page.

4.1 The learning objective function

$$L(x_i, x_j) = a(W * (||f(x_i) - f(x_j)||_2^2) + b) \tag{4}$$

$$J = \sum_{i,j} L(x_i, x_j) \tag{5}$$

where a is activation function, W is weight vector and b is the bias for the dense layer.

Finally, we use permutation and data-augmentation to train the model on all the possible samples

For **Data-Augmentation**, we used Random Rotation, ZCA whitening, Horizontally Flips, Shearing Angle Variation and intensity variation

.

4.2 Model Architecture

We show the number of parameters in each layer and parameter to be trainable and non-trainable in our model's architecture.

Layer (type)	Output Shape	Param #
FeatureNet_ImageInput (Input	(None, 256, 256, 3)	0
conv2d_1 (Conv2D)	(None, 256, 256, 8)	32
conv2d_2 (Conv2D)	(None, 252, 252, 8)	1608
activation_1 (Activation)	(None, 252, 252, 8)	0
max_pooling2d_1 (MaxPooling2	(None, 126, 126, 8)	0
conv2d_3 (Conv2D)	(None, 126, 126, 16)	144
conv2d_4 (Conv2D)	(None, 124, 124, 16)	2320
activation_2 (Activation)	(None, 124, 124, 16)	0
max_pooling2d_2 (MaxPooling2	(None, 62, 62, 16)	0
conv2d_5 (Conv2D)	(None, 62, 62, 32)	544
conv2d_6 (Conv2D)	(None, 60, 60, 32)	9248
activation_3 (Activation)	(None, 60, 60, 32)	0
max_pooling2d_3 (MaxPooling2	(None, 30, 30, 32)	0
conv2d_7 (Conv2D)	(None, 30, 30, 64)	2112
conv2d_8 (Conv2D)	(None, 28, 28, 64)	36928
activation_4 (Activation)	(None, 28, 28, 64)	0
max_pooling2d_4 (MaxPooling2	(None, 14, 14, 64)	0
conv2d_9 (Conv2D)	(None, 14, 14, 128)	8320
conv2d_10 (Conv2D)	(None, 12, 12, 128)	147584
activation_5 (Activation)	(None, 12, 12, 128)	0
max_pooling2d_5 (MaxPooling2	(None, 6, 6, 128)	0
global_max_pooling2d_1 (Glob	(None, 128)	0

Total params: 208,840
Trainable params: 208,840
Non-trainable params: 0

Layer (type)	Output Shape	Param #	Connected to
ImageA_Input (InputLayer)	(None, 256, 256, 3)	0	
ImageB_Input (InputLayer)	(None, 256, 256, 3)	0	
FeatureGenerationModel (Model)	(None, 128)	208840	ImageA_Input[0][0] ImageB_Input[0][0]
merge_features (Concatenate)	(None, 256)	0	FeatureGenerationModel[1][0] FeatureGenerationModel[2][0]
dense_1 (Dense)	(None, 32)	8224	merge_features[0][0]
batch_normalization_1 (BatchNor	(None, 32)	128	dense_1[0][0]
activation_6 (Activation)	(None, 32)	0	batch_normalization_1[0][0]
dense_2 (Dense)	(None, 8)	264	activation_6[0][0]
batch_normalization_2 (BatchNor	(None, 8)	32	dense_2[0][0]
activation_7 (Activation)	(None, 8)	0	batch_normalization_2[0][0]
dense_3 (Dense)	(None, 4)	36	activation_7[0][0]
batch_normalization_3 (BatchNor	(None, 4)	16	dense_3[0][0]
activation_8 (Activation)	(None, 4)	0	batch_normalization_3[0][0]
dense_4 (Dense)	(None, 1)	5	activation_8[0][0]

Total params: 217,545
Trainable params: 217,457
Non-trainable params: 88

5 Tool for Experiment

5.1 Dataset used

The dataset, we have collected from internet, have 10 different person and each person has 10 different image with variation in the intensity, shearing angle, etc.

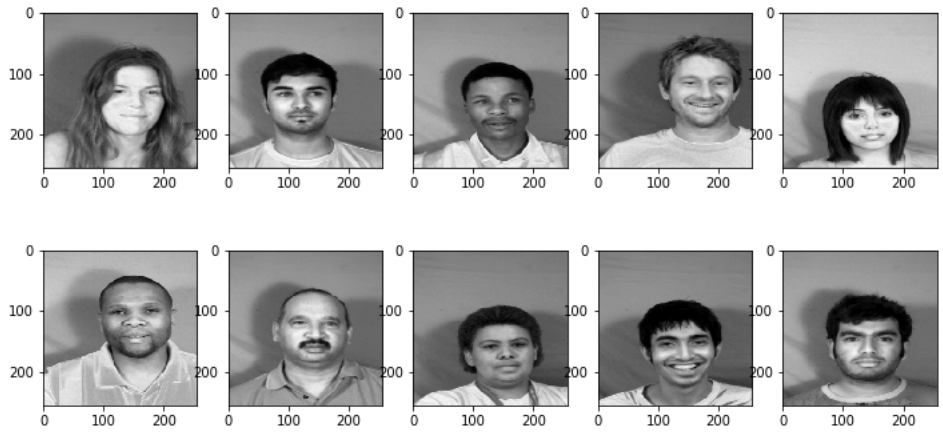


Figure 12: Data-Set Images

5.2 Software Used

The software used in our experiment are as follows

Figure 13: Tensorflow

Figure 14: Python

Figure 15: Arduino

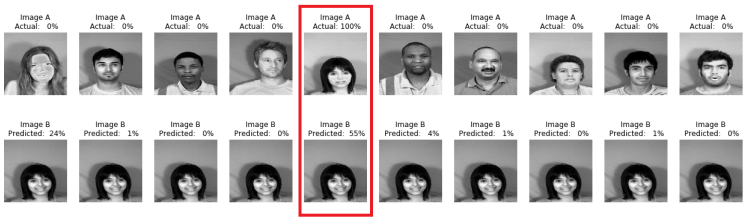
Figure 16: Software Used in Experiment

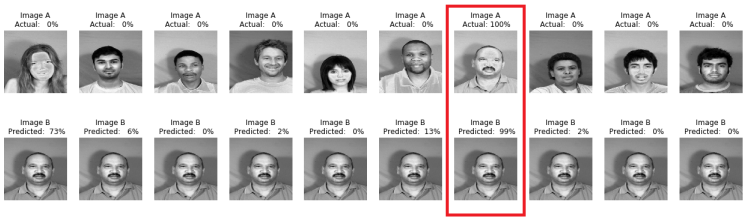
6 Expriments and Results

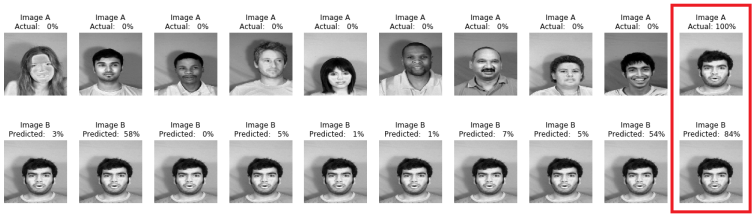
During testing, we send our test image along with all the different faces present in the data-set. Our model will compare the test image with the all class which means 10 images in our cases and find the similarity between the pair.

For our problem setting, we get **accuracy of** 91%. Here are some of the results, we get during testing.

Note: For an test image, the score of more than 50% gives us true result.







6.1 Arduino Interfacing With CNN model

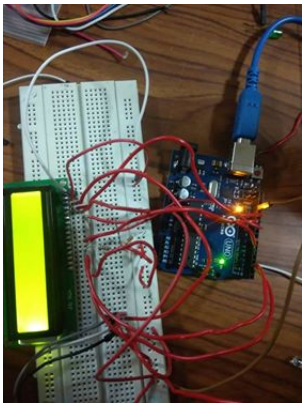


Figure 17: Arduino Inrefacing

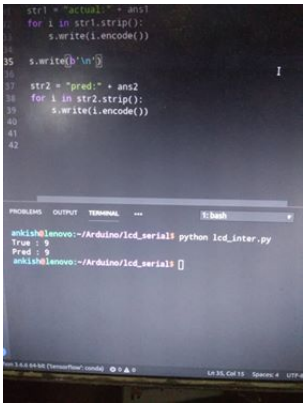


Figure 18: Results On window

7 Our Contribution

i To reduce the number of parameter in the architecture, we used **Network in Network** approach in network arch. This methods helps the number of parameters from 138M to .45M parameters using our architecture.

ii **We are able to train our model with very few examples using technique mentioned above.**

8 Limitation of model

As our model has around $425,000$ parameter to learn, but dataset is very small. To learn from such small data, we used permutation technique to generate the dataset. Using permutation we generatedv 4500 samples from 100 images. We also used image augmentation technique to generate the data to make our model robust for little variation in term of intensity, color, size, shearing angle and contrast variation in images. This way we generate enough data to make it learn.

9 Conclusion

We have studied Few Shot Learning problem formulation and approach has been followed in the literature. Then we experiment with real dataset with ten different person faces. We found that our model has learned with only 100 image. After discussion with deep-learning practitions, we found that our model suffer from the overfit because of small data-set.But the application, we are dealing with, there will never be any problem from that. Finally with interfaced our CNN model with arduino and displayed the result on the LCD.