Kingdom of Saudi Arabia

Ministry of Education

Bisha University

Faculty of Computers and IT

Information System Department

Master in Cybersecurity

Biometric Systems

SYS 616

A Short Technical Report in

A CONVOLUTIONAL NEURAL NETWORK FOR IRIS RECOGNITION

Supervisor(s)

Dr. Mostafa Ahmad

Assistant Professor in Computer Science Department

Author(s)

Wael Ghazi Ahmed Alnahari

Final Practical Report

April 2020

TABLE OF CONTENTS

ABSTRACT

In this paper, I proposed an iris recognition system by using deep learning via convolutional neural networks (CNN). Although CNN is used for machine learning, the recognition is achieved by building a non-trained CNN network with multiple layers. The main objective of the code is recognizing test pictures' category (aka person name) with high accuracy rate after having extracted enough features from training pictures of the same category which are obtained from a dataset that I added to the code. I used IITD iris dataset which included 10 iris pictures for 223 people. The pictures were divided into 3 pictures for testing and 7 pictures for training. The categories are from 001 to 223. Since the number of pictures for training is low, in order to enhance the recognition accuracy of the network, I used five sets of layers and altered nine parameters of *sgdm* training option. The code concluded an accuracy rate of 97.46% and the time elapsed was 10 minutes and 30 seconds.

INTRODUCTION

1.1 Introduction to CNN

Convolutional neural networks (CNN) is a deep learning algorithm used to help machines categorize objects. CNN is used as a computer vision detection on images. CNN is designed to mimic human brain hence the name neural. CNN is used to treat visual inputs and determine what the input represents based on the inputs features. CNN uses a set of multiple layers to analyze the visual inputs and to determine what are the inputs' distinguishing features based on probability of repetition. CNN is useful for Artificial Intelligence (AI) such as self-driving cars to help the cars determine what the objects appearing in their cameras represent. In general, CNN is two parts: convolution layer and fully connected layer. The convolution layer is a single layer used to extract a specific feature from the visual input which can be repeated as many times as desired. The fully connected layer connects all convolution layers to one layer or representation to combine the features.[1]

1.2 CNN Architecture

In order to create a CNN system, one needs a dataset to be used for training and another dataset to be used for testing. The training dataset is categorized according to the desirable categories. The set of data for training is usually very large compared to the test data but it is not a requirement; however, the bigger the number of data for training the better the quality of the result. The machine uses the training images or train dataset to find distinguishing features of each category. The features are extracted with the help of layers. A layer in this case represents a specific process which transforms the image into another shape, size, color, or look which is done in some pixels that are extremely smaller than the actual image. Many layers are applied for images of the same category which are then stored in the network of the CNN. The stored features are based on the probability of repetition of a number of features among a category from the training dataset. Afterward, when an image is tested, the same layers will be applied to it and then are processed to determine which category is most similar to its features.

IMPLEMENTATION OF CNN FOR BIOMETRIC RECOGNITION

2.1 Data Set

Description of the IITD Iris Image Database version 1.0

==========

This iris image database mainly consists of the iris images collected from the students and staff at IIT Delhi, India. This database has been acquired in the Biometrics Research Laboratory during January - July 2007 using JIRIS, JPC1000, digital CMOS camera. The acquired images were saved in bitmap format.
The database of 2240 images is acquired from 224 different users and made available freely to the researchers. All the subjects in the database are in the age group 14-55 years comprising of 176 males and 48 females. The resolution of these images is 320 x 240 pixels and all these images were acquired in the indoor environment. All the images in the database were acquired from the volunteers who were not paid or provided any honorarium. The images were acquired using an automated program that requires users to present their eyes in a sequence until ten images are registered.

Organization of Database

==========

The acquired database is saved in 224 folders, each corresponding to 224 subjects. Majority of images were acquired from the left eyes while the rest images were acquired from right eye. Now the database has a label 'L' or 'R' which designates left or right eye. There are 1288 images from 224 subject that are from left eyes while the rest images from 211 subjects are from right eyes.Except folders 1-13, 27, 55 and 65 all other folders have five left and 5 right eye images. (**appended on 20-04-2016**).

Usage of Database

==========

This database is only available for research and noncommercial purposes. Commercial distribution or any act related to commercial use of this database is strictly prohibited. Kindly acknowledge all the publicly available publications/work employing this database with the following acknowledgment:

"Portions of the work tested on the IITD Iris Database version 1.0"
A citation to "IIT Delhi Iris Database version 1.0,
http://web.iitd.ac.in/~biometrics/Database_Iris.htm

Related Publication:

==========

Ajay Kumar and Arun Passi, "Comparison and combination of iris matchers for reliable personal identification," Proc. CVPR 2008, Anchorage, Alaska, pp. 21-27 Jun. 2008

Contact Information:

==========

Ajay Kumar

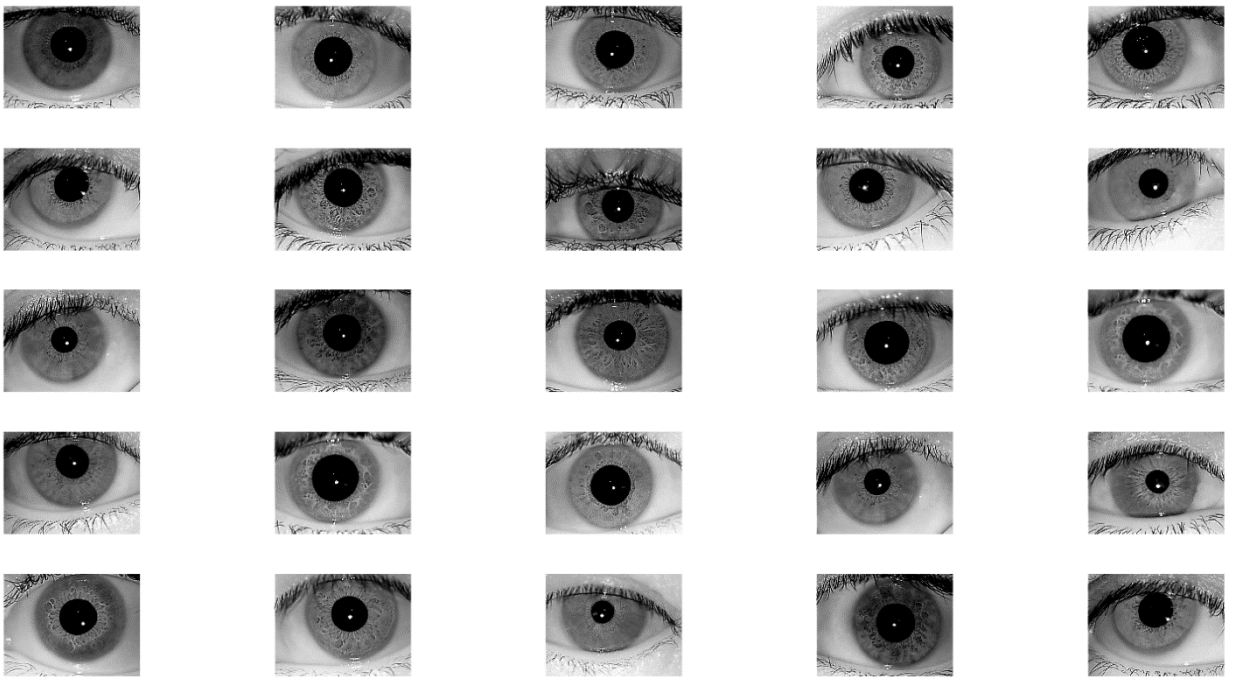
Biometrics Research Laboratory

Indian Institute of Technology Delhi

New Delhi, India

E-mail: ajaykr@ieee.org

A sample of the dataset pictures:



Sample of Dataset Images

2.2 Layers

A CNN is composed of different types of layers as follows: [1]

Convolutional layer: in this layer the computer scans the visual input few pixels at a time to determine a feature for each few pixels then continue to the next few pixels until the whole visual input is finalized.

Pooling layer (downsampling): in this layer the features of the previous layer is downsampled after having stored the most important features.

Fully connected layer: in this layer all the previous layers are connected together to build a conclusion of all important features.

The used layers of our CNN are:

```
  layers = [

imageInputLayer([x y z]);

convolution2dLayer(3,8,'Padding','same')
batchNormalizationLayer
reluLayer();

maxPooling2dLayer(5,'Stride',2)
```

```
convolution2dLayer(3,16,'Padding','same')
batchNormalizationLayer
reluLayer();

    averagePooling2dLayer(5,'Stride',2)
convolution2dLayer(3,16,'Padding','same')
batchNormalizationLayer
reluLayer();

maxPooling2dLayer(5,'Stride',2)
convolution2dLayer(3,32,'Padding','same')
batchNormalizationLayer
reluLayer();

    averagePooling2dLayer(5,'Stride',2)
convolution2dLayer(3,32,'Padding','same')
batchNormalizationLayer
reluLayer();

fullyConnectedLayer(223,'BiasLearnRateFactor',2);
softmaxLayer
classificationLayer];
```

2.3 Training Options

```
options = trainingOptions('sgdm', ...
    'InitialLearnRate', 0.0001, ...
    'ValidationData',imdsValidation, ...
    'ValidationFrequency',30, ...
    'Shuffle','every-epoch', ...
    'MaxEpochs', 10, ...
    'MiniBatchSize', 8, ...
    'ValidationFrequency',50, ...
    'LearnRateSchedule','piecewise', ...
    'LearnRateDropFactor',0.05, ...
    'LearnRateDropPeriod',60, ...
    'ExecutionEnvironment','parallel',...
    'Verbose', true, 'Plots','training-progress');
```

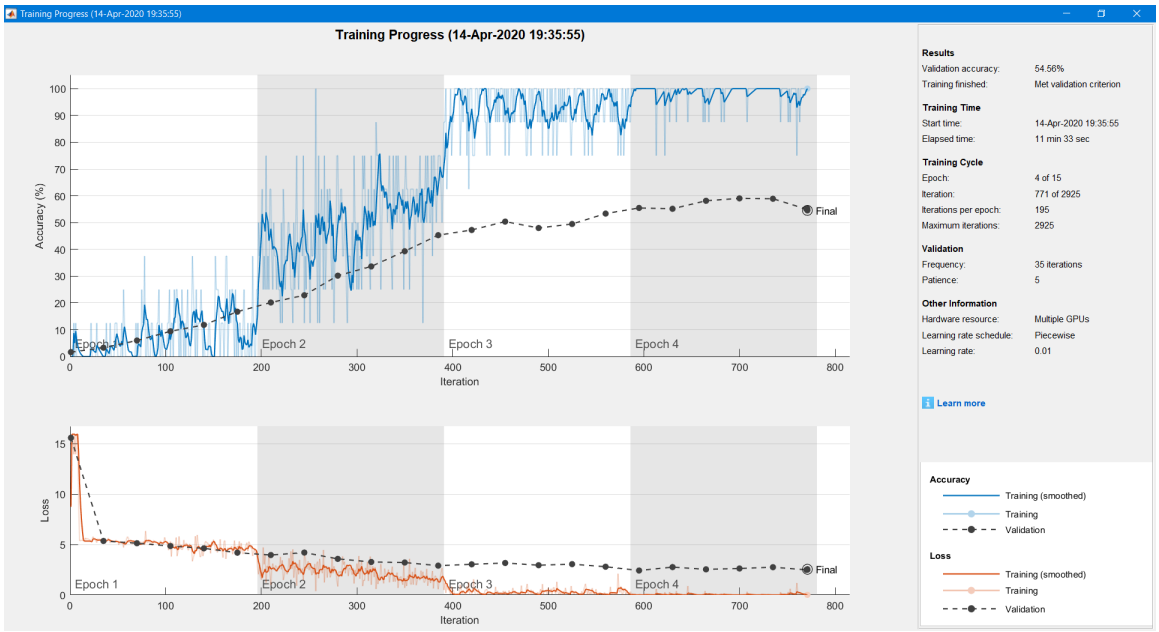
RESULTS

3.1 Training Trails

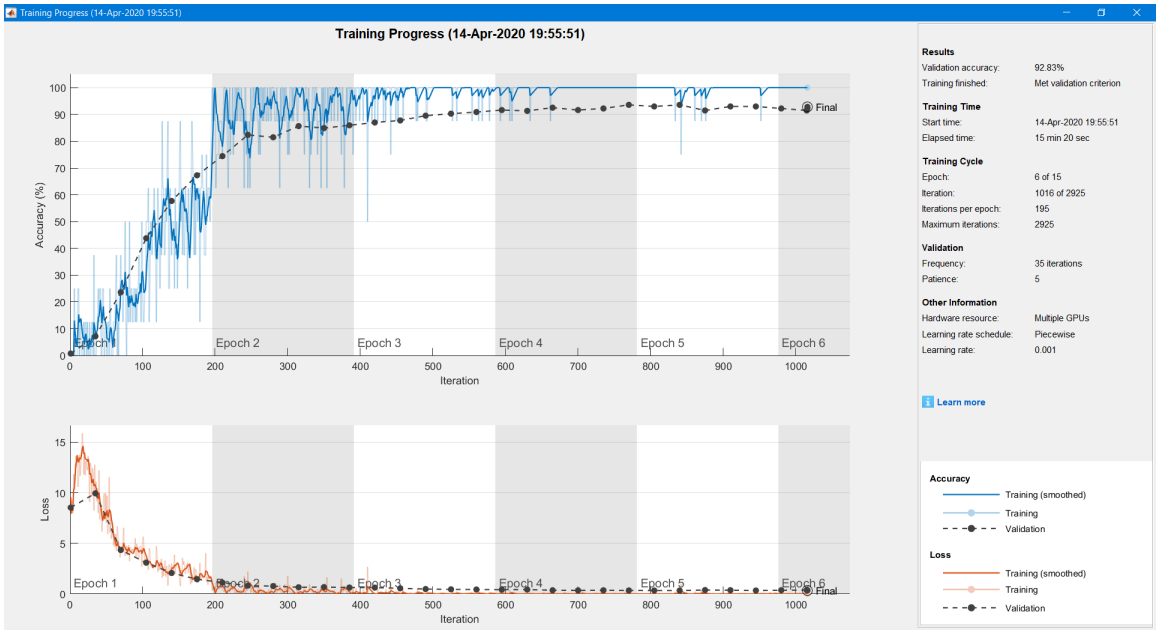
I got an accuracy of 54.56% using the following specifications for layers and options:

```
%% Define Network Architecture

layers = [


imageInputLayer([x y z]);


 convolution2dLayer(3,8,'Padding','same')

batchNormalizationLayer

reluLayer

maxPooling2dLayer(2,'Stride',2)

convolution2dLayer(3,16,'Padding','same')

batchNormalizationLayer

reluLayer

maxPooling2dLayer(2,'Stride',2)

convolution2dLayer(3,32,'Padding','same')

batchNormalizationLayer

reluLayer
```

```
fullyConnectedLayer(size(categories,2),'BiasLearnRateFa
ctor',2);
    softmaxLayer
    classificationLayer];

    %% Specify Training Options
    options = trainingOptions('sgdm', ...
    'InitialLearnRate', 0.01, ...
    'ValidationData',imdsValidation, ...
    'ValidationFrequency',35, ...
    'Shuffle','every-epoch', ...
    'MaxEpochs', 15, ...
    'MiniBatchSize', 8, ...
    'LearnRateSchedule','piecewise', ...
    'LearnRateDropFactor',0.05, ...
    'LearnRateDropPeriod',60, ...
    'ExecutionEnvironment','parallel', ...
    'Verbose', true, 'Plots','training-progress');
```



When I adjusted `'InitialLearnRate'` to `0.001` instead of `0.01` I got an accuracy rate of 92.83%

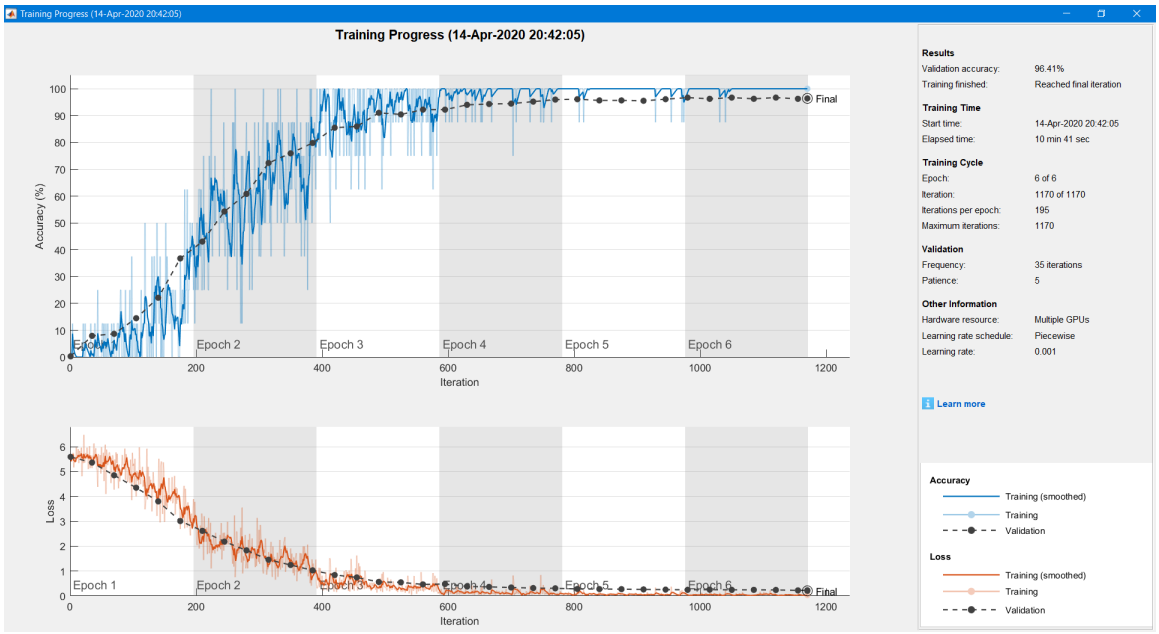


When I used the same training options as above but the maxEpoch to 6 in order to compare the results with the previous trial and changed the layers I managed to reduce the elapsed time to 00:10:41 with 96.41% accuracy

```
%% Define Network Architecture
layers = [

imageInputLayer([x y z]);

convolution2dLayer(3,8,'Padding','same')
batchNormalizationLayer
reluLayer();

maxPooling2dLayer(5,'Stride',2)
convolution2dLayer(3,16,'Padding','same')
batchNormalizationLayer
reluLayer();

     averagePooling2dLayer(5,'Stride',2)
convolution2dLayer(3,16,'Padding','same')
batchNormalizationLayer
reluLayer();
```

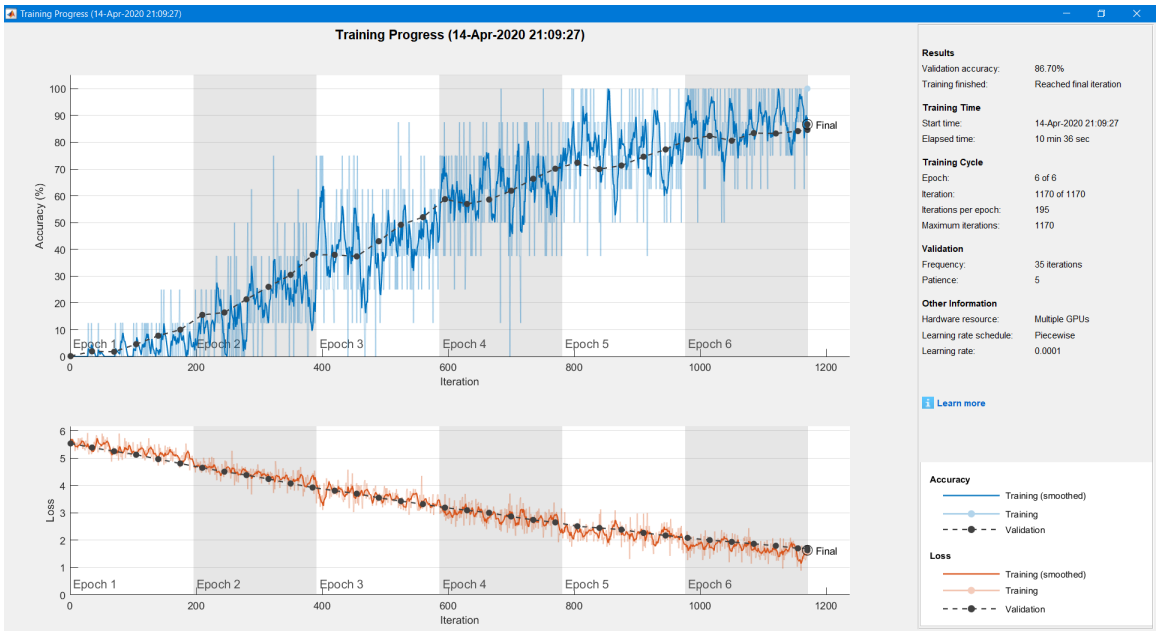
```
    maxPooling2dLayer(5,'Stride',2)
    convolution2dLayer(3,32,'Padding','same')
    batchNormalizationLayer
    reluLayer();

        averagePooling2dLayer(5,'Stride',2)
    convolution2dLayer(3,32,'Padding','same')
    batchNormalizationLayer
    reluLayer();



fullyConnectedLayer(size(categories,2),'BiasLearnRateFa
ctor',2);
    softmaxLayer
    classificationLayer];

    %% Specify Training Options
    options = trainingOptions('sgdm', ...
    'InitialLearnRate', 0.001, ...
    'ValidationData',imdsValidation, ...
    'ValidationFrequency',35, ...
```

```
'Shuffle','every-epoch', ...
'MaxEpochs', 6, ...
'MiniBatchSize', 8, ...
'LearnRateSchedule','piecewise', ...
'LearnRateDropFactor',0.05, ...
'LearnRateDropPeriod',60, ...
'ExecutionEnvironment','parallel', ...
'Verbose', true, 'Plots','training-progress');
```

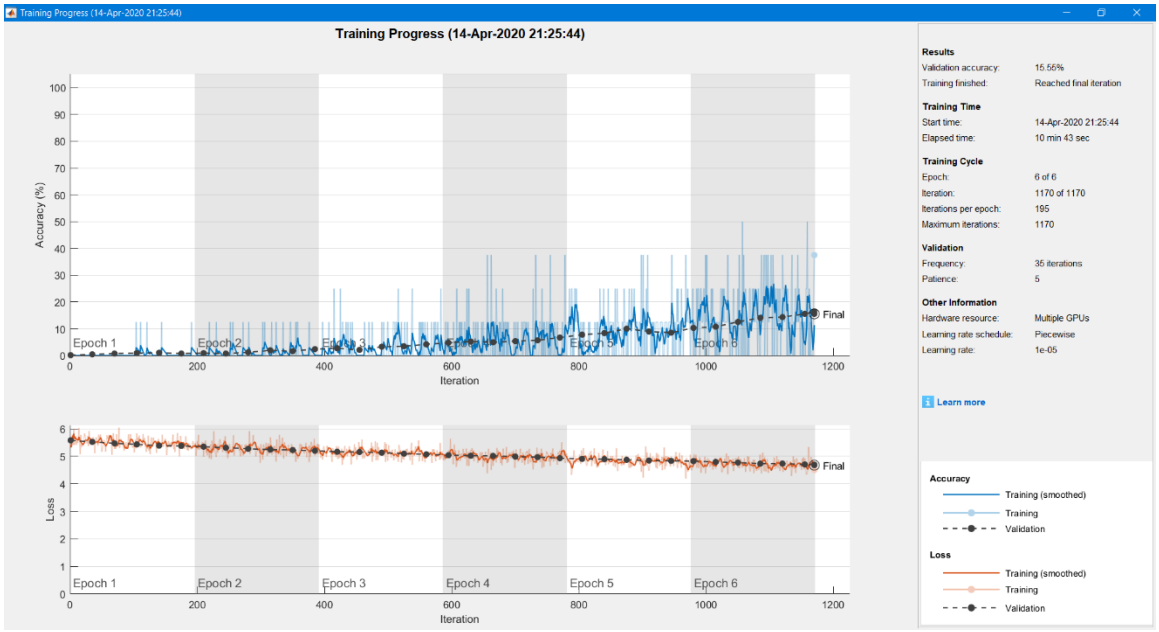


When I changed `'InitialLearnRate'` to `0.0001` and

`'ValidationFrequency'` to `35` I ended up with a lower accuracy 86.70%



When I changed `'InitialLearnRate'` to `0.00001` I ended up with an extremely low accuracy of 15.55%



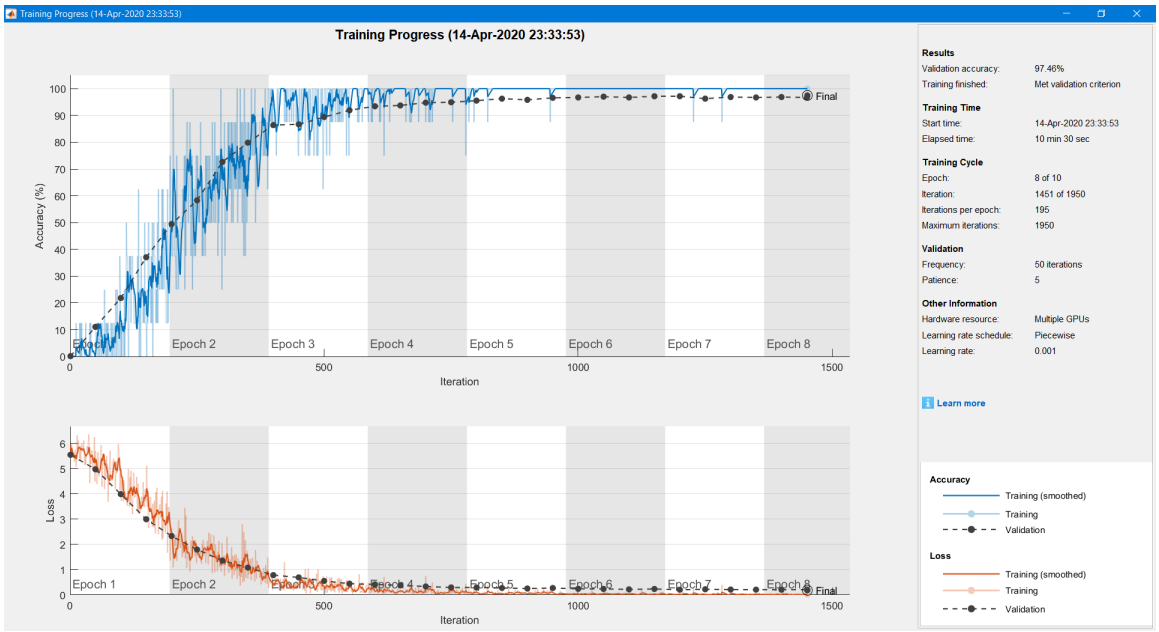
3.2 Final Accuracy

Using the following layer and training options:

```
layers = [

imageInputLayer([x y z]);

convolution2dLayer(3,8,'Padding','same')
batchNormalizationLayer
reluLayer();

maxPooling2dLayer(5,'Stride',2)
convolution2dLayer(3,16,'Padding','same')
batchNormalizationLayer
reluLayer();

     averagePooling2dLayer(5,'Stride',2)
convolution2dLayer(3,16,'Padding','same')
batchNormalizationLayer
reluLayer();

maxPooling2dLayer(5,'Stride',2)
convolution2dLayer(3,32,'Padding','same')
```

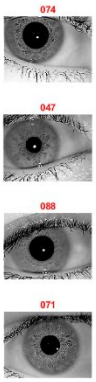
```
    batchNormalizationLayer
    reluLayer();

        averagePooling2dLayer(5,'Stride',2)
    convolution2dLayer(3,32,'Padding','same')
    batchNormalizationLayer
    reluLayer();



fullyConnectedLayer(size(categories,2),'BiasLearnRateFa
ctor',2);
    softmaxLayer
    classificationLayer];

    %% Specify Training Options
    options = trainingOptions('sgdm', ...
    'InitialLearnRate', 0.001, ...
    'ValidationData',imdsValidation, ...
    'ValidationFrequency',50, ...
    'Shuffle','every-epoch', ...
    'MaxEpochs', 10, ...
```

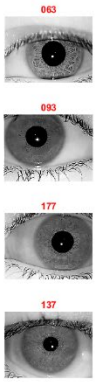
```
'MiniBatchSize', 8, ...
'LearnRateSchedule','piecewise', ...
'LearnRateDropFactor',0.5, ...
'LearnRateDropPeriod',50, ...
'ExecutionEnvironment','parallel', ...
'Verbose', true, 'Plots','training-progress');
```



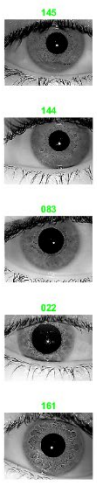
3.3 Samples of Identification Results

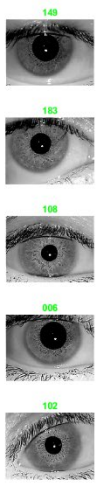
The for loop for samples was adjusted to display all mismatches then a sample of successful matches were displayed with the same numbers of mismatches as follows:

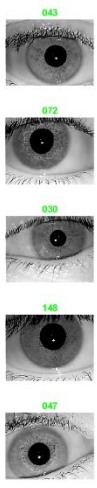




ALL mismatch incidents







Sample of Correct Incidents

APPENDIX: MATLAB CODE

CNN Code for Biometric Recognition

```
%   Biometric Systems
%   CYS616
%   Wael Ghazi Ahmed Alnahari
%   440804845
%   Dr. Mostafa Abdel-Halim Mostafa Ahmad
%   A CONVOLUTIONAL NEURAL NETWORK FOR IRIS RECOGNITION
clc;    % Clear the command window.
close all;  % Close all figures (except those of imtool.)
clear;  % Erase all existing variables. Or clearvars if you
want.
workspace;  % Make sure the workspace panel is showing.
reset(gpuDevice);  % Reset GPU memory

%% Load train Data
categories =
{'001','002','003','004','005','006','007','008','009','010
','011','012','013','014','015','016','017','018','019','02
0', ...
```

```
'021','022','023','024','025','026','027','028','029','030'
,'031','032','033','034','035','036','037','038','039','040
', ...

'041','042','043','044','045','046','047','048','049','050'
,'051','052','053','054','055','056','057','058','059','060
', ...

'061','062','063','064','065','066','067','068','069','070'
,'071','072','073','074','075','076','077','078','079','080
', ...

'081','082','083','084','085','086','087','088','089','090'
,'091','092','093','094','095','096','097','098','099','100
', ...

'101','102','103','104','105','106','107','108','109','110'
,'111','112','113','114','115','116','117','118','119','120
', ...

'121','122','123','124','125','126','127','128','129','130'
```

```
,'131','132','133','134','135','136','137','138','139','140
', ...

'141','142','143','144','145','146','147','148','149','150'
,'151','152','153','154','155','156','157','158','159','160
', ...

'161','162','163','164','165','166','167','168','169','170'
,'171','172','173','174','175','176','177','178','179','180
', ...

'181','182','183','184','185','186','187','188','189','190'
,'191','192','193','194','195','196','197','198','199','200
', ...

'201','202','203','204','205','206','207','208','209','210'
,'211','212','213','214','215','216','217','218','219','220
', ...
              '221','222','223'};

    imdsTrain =
imageDatastore(fullfile(pwd,"Dataset\TrainData",
```

```
categories),'IncludeSubfolders',true,'FileExtensions','.bmp
','LabelSource','foldernames');
num_train = size(imdsTrain.Files,1);

%% Size of first image in dataset
    img = readimage(imdsTrain,1);
    [x , y , z] = size(img);

%% Load Test Data
    imdsValidation =
imageDatastore(fullfile(pwd,"Dataset\TestData",
categories),'IncludeSubfolders',true,'FileExtensions','.bmp
','LabelSource','foldernames');
num_test = size(imdsValidation.Files,1);
%% Calculate the number of images in each category.
    labelCount = countEachLabel(imdsTrain);

    %% Define Network Architecture
    layers = [

    imageInputLayer([x y z]);
```

```
convolution2dLayer(3,8,'Padding','same')
batchNormalizationLayer
reluLayer();

maxPooling2dLayer(5,'Stride',2)
convolution2dLayer(3,16,'Padding','same')
batchNormalizationLayer
reluLayer();

    averagePooling2dLayer(5,'Stride',2)
convolution2dLayer(3,16,'Padding','same')
batchNormalizationLayer
reluLayer();

maxPooling2dLayer(5,'Stride',2)
convolution2dLayer(3,32,'Padding','same')
batchNormalizationLayer
reluLayer();

    averagePooling2dLayer(5,'Stride',2)
convolution2dLayer(3,32,'Padding','same')
batchNormalizationLayer
```

```
    reluLayer();



fullyConnectedLayer(size(categories,2),'BiasLearnRateFactor
',2);
    softmaxLayer
    classificationLayer];

    %% Specify Training Options
    options = trainingOptions('sgdm', ...
    'InitialLearnRate', 0.001, ...
    'ValidationData',imdsValidation, ...
    'ValidationFrequency',50, ...
    'Shuffle','every-epoch', ...
    'MaxEpochs', 10, ...
    'MiniBatchSize', 8, ...
    'LearnRateSchedule','piecewise', ...
    'LearnRateDropFactor',0.5, ...
    'LearnRateDropPeriod',50, ...
    'ExecutionEnvironment','parallel', ...
    'Verbose', true, 'Plots','training-progress');
```

```
%% Train Network Using Training Data
    [net_Wael, info] =
trainNetwork(imdsTrain,layers,options);
    save net_Wael

    %% Classify validation
    labels = classify(net_Wael,imdsValidation);

        %% *Test one at a time*
true=0;
false=0;
    for i = 1:size(imdsValidation.Files,1)
        im = imread(imdsValidation.Files{i});
            if labels(i) == imdsValidation.Labels(i)
               true=true+1;
            else
                false=false+1;
            end
    end
```

```
    % Find the square root of the false matches to
determine the subplots

    f0=floor(sqrt(false));
    if f0 == sqrt(false)
        f1=f0;
    else
        f1=f0+1;
    end

    % Recalbirating true and false
    true=0;
    false=0;

    %start plotting mismatches
    figure('Name','All Mismatch Pictures')
    for i = 1:size(imdsValidation.Files,1)
        im = imread(imdsValidation.Files{i});
        if labels(i) ~= imdsValidation.Labels(i)

            colorText = 'r';
            false=false+1;
```

```
        if false > f0*f1
                break
        else
        subplot(f1,f0,false);
        imshow(im);
        title(char(labels(i)),'Color',colorText);
        end
    end
end
        xlabel('ALL mismatch incidents')

%start plotting sample of correct match
figure('Name','Sample of Correct Match')
true=0;
false=0;
    for i = 1:size(imdsValidation.Files,1)
        ii = randi(size(imdsValidation.Files,1));
        im = imread(imdsValidation.Files{ii});
        if labels(ii) == imdsValidation.Labels(ii)
           colorText = 'g';
           true=true+1;
           if true > f0*f1
```

```
                    break
                else
                        subplot(f1,f0,true);
                        imshow(im);

title(char(labels(ii)),'Color',colorText);
                end
            end
        end
            xlabel('Sample of Correct Incidents')

    %% Compute Accuracy
    YValidation = imdsValidation.Labels;
    accuracy02 = sum(labels ==
YValidation)/numel(YValidation);
    display(accuracy02);
```

ACKNOLEDGEMENT

I wish to thank my parents for their support and encouragement throughout my studies.

REFERENCES

Online Sources

1. **Online Article:** Convolutional Neural Network Architecture: Forging Pathways to the Future: https://missinglink.ai/guides/convolutional-neural-networks/convolutional-neural-network-architecture-forging-pathways-future/

2. **Dataset:** IITD Iris Image Database https://www4.comp.polyu.edu.hk/~csajaykr/IITD/Database_Iris.htm