Convolutional Neural Network for Iris Recognition

Wael Alnahari (✉ master@wjn.sa)

University of Bisha <https://orcid.org/0000-0001-5247-3713>

Short Report

Keywords: Biometrics, Biometric Systems, Machine Learning, Deep Learning, Recognition, IRIS

Posted Date: February 16th, 2021

DOI: <https://doi.org/10.21203/rs.3.rs-244624/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Convolutional Neural Network for Iris Recognition

Wael Alnahari

College of Computers and Information Technologies, 24 Lab 1, University of Bisha,
8338 Riyadh - Al Nakhil Dist. Unit No. 4 - 5693,

Bisha,

67716,

Asir,

Saudi Arabia, Tel.: +966506202244,

Fax: +966126700946

Email: master@wjn.sa

(Corresponding author)

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 the test pictures' category (aka person name) with a 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.

Index Terms—Biometrics, Biometric Systems, Machine Learning, Deep Learning, Recognition, IRIS

I. NOMENCLATURE

CNN: Convolutional Neural Network

Iris: The circular, colored curtain of the eye. The opening of the iris forms the pupil. The iris helps regulate the amount of light that enters the eye.

II. INTRODUCTION

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. 737521:17512588

III. DISCUSSION

A. 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.

IV. IMPLEMENTATION OF CNN FOR BIOMETRIC RECOGNITION

A. Data Set: Description of the IITD Iris image database version 1.0 737521:17512589

A sample of the dataset pictures:

B. Layers

A CNN is composed of different types of layers as follows: 737521:17512588

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]);
```

Sample of Dataset Images

Fig. 1. A sample of the dataset pictures

```
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];
```

C. 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');
```

V. RESULTS

A. 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),'BiasLearnRateFactor',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),'BiasLearnRateFactor',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%

VI. CONCLUSION

A. Final Accuracy

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 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),'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');

```

B. 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:

VII. AVAILABILITY OF DATA AND MATERIALS

A. Methods

Matlab

Fig. 2. An accuracy rate of 54.56%

Fig. 3. An accuracy rate of 92.83%

Fig. 4. An accuracy rate of 96.41%

Fig. 5. An accuracy rate of 86.70%

Fig. 6. An accuracy rate of 15.55%

Fig. 7. Final accuracy rate of 97.46% and the time elapsed was 10 minutes and 30 seconds

Fig. 8. successful matches

Fig. 9. mismatches

B. Supplemental Material

Online [1]

VIII. FUNDING

WJN LLC

IX. COMPETING INTERESTS

The authors declare no competing interests.

ACKNOWLEDGEMENTS

I wish to thank my parents for their support and encouragement throughout my studies. Special thanks to Assistant Professor in Computer Science Department Dr. Mostafa Ahmad.

REFERENCES

- [1] W. Alnahari and Wesam Alnahri, "Convolutional Neural Network for Iris Recognition," May 2020. [Online]. Available: 10.24433/CO.9842635.v1; <https://doi.org/10.24433/CO.9842635.v1>

Figures

Sample of Dataset Images

Figure 1

A sample of the dataset pictures

Figure 2

An accuracy rate of 54.56%

Figure 3

An accuracy rate of 92.83%

Figure 4

An accuracy rate of 96.41%

Figure 5

An accuracy rate of 86.70%

Figure 6

An accuracy rate of 15.55%

Figure 7

Final accuracy rate of 97.46% and the time elapsed was 10 minutes and 30 seconds

Sample of Correct Incidents

Figure 8

successful matches

ALL mismatch incidents

Figure 9

mismatches