Iris Recognition in Deep Learning

- **Implementing CNN in post-preprocessing**

Yi-Yang Lin

Outline

Introduction

This report explores the application of deep learning, specifically convolutional neural networks (CNNs), to iris image recognition.

The process of iris recognition can be dissected into several critical steps, each integral to the overall effectiveness of the system. The theoretical and practical frameworks established by pioneers such as John Daugman and Libor Masek have set a solid foundation for this field.

Daugman introduced the essential methodologies, while Masek enhanced the accessibility of these technologies through practical implementations and open-source projects. The systematic approach to biometric identification generally involves several stages: Image Acquisition, Preprocessing, Iris Localization, Feature Extraction, Feature Encoding, Matching, Decision Making, Verification/Identification, and Access or Authentication.

In the paper "DeepIris: Iris Recognition Using a Deep Learning Approach" by Shervin Minaee and Amirali Abdolrashidi. Minaee and Abdolrashidi employ a direct application of deep learning techniques to raw iris images. This method leverages the powerful feature-learning capabilities of deep neural networks to automatically extract and learn the most relevant features from the iris images without the need for explicit feature extraction and encoding steps.

However, I proposed to combine the approach of preprocessing and feature extraction with application of deep leaning. After preprocessing, use the approach by Minaee and Abdolrashidi to apply a deep learning model, ResNet, directly to these preprocessed images. The CNN can learn additional complex features from the preprocessed data, which might not be captured through traditional feature extraction techniques.

This report aims to focus on applying ResNet to preprocessed images, which includes in those stages in CNN after Normalization and Encoding. By training the CNN model on these

processed images, specifically, the polar rectangular images, we aim to validate the efficacy of the matching process.

How CNNs Work

Input Layer: Takes the raw pixel values of the image.

Convolutional Layer
Activation Function (usually ReLU)
Pooling Layer
Normalization Layer
Fully Connected Layer
Output Layer
Softmax/Logistic Function

Hidden Layers: These include multiple convolutional, activation, and pooling layers. CNN architectures can vary significantly here.

feature extraction, feature maps, filter kernels, and the role of each layer in extracting and transforming input data.

Output Layer: Uses a softmax or sigmoid activation function to make predictions.

CNN Steps

Data Preparation: Gather and prepare the dataset for training and testing the CNN. Ensure that the dataset is labeled correctly, balanced (if applicable), and split into training, validation, and testing sets.

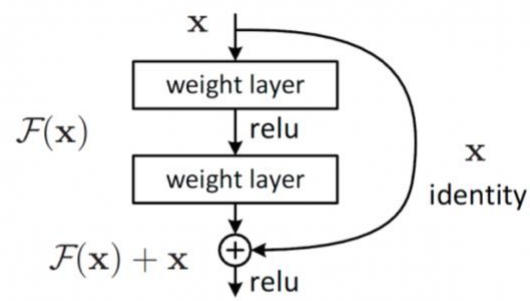
Model Design: Design CNN architecture based on the problem is solving. Decide on the number of layers, their types (convolutional, pooling, fully connected), activation functions, and other hyperparameters. Consider existing CNN architectures (e.g., LeNet, AlexNet, VGG, ResNet) as references.

Training: Train your CNN using the prepared dataset and the chosen framework. Experiment with different optimization algorithms (e.g., Adam, SGD), learning rates, batch sizes, and regularization techniques (e.g., dropout) to improve model performance.

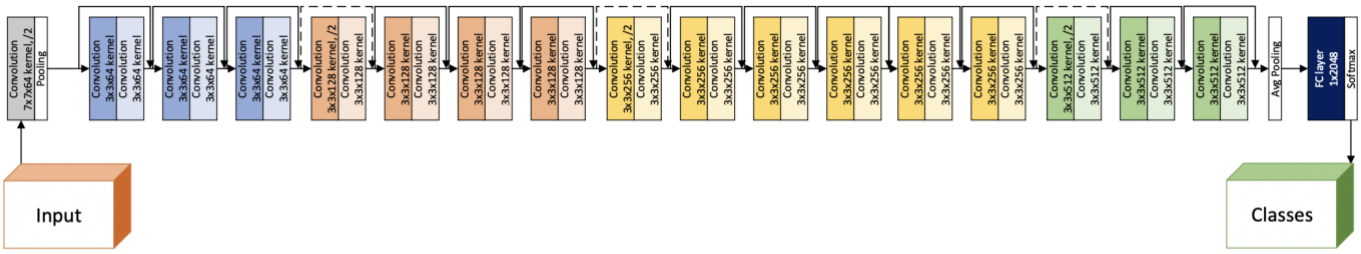
Evaluation: Evaluate the trained CNN using the validation set to assess its performance metrics (e.g., accuracy, precision, recall, F1-score). Fine-tune the model and hyperparameters based on validation results to improve performance further.

ResNet

ResNet, or Residual Network, addresses the vanishing gradient problem in deep neural networks by introducing skip connections that allow gradients to flow directly through multiple layers. These connections, which perform identity mapping, skip one or more layers and add the input directly to the output of a block, thereby simplifying the network's learning objectives. As a result, ResNet can effectively train much deeper networks, with configurations like ResNet-50, ResNet-101, and ResNet-152 showing improved performance on various tasks. The architecture has set new benchmarks in large-scale image recognition competitions such as ImageNet. ResNet's ability to maintain performance with increased depth has made it a fundamental model in advancing the capabilities of deep learning across a wide range of applications.



The residual block used in ResNet Model



The architecture of ResNet50 neural network [13], and how it is transferred for iris recognition. The last layer is changed to match the number of classes in our dataset.

In the **Code > ResNet_MNIST** Folder, it provides training code of ResNet demonstration.

ResNet_MNIST.html

ResNet_MNIST.ipynb

The example of how to use a ResNet model to perform image classification on the MNIST dataset, which consists of handwritten digits.

ResNet-50, a popular variant of the ResNet architecture, with the Keras library in TensorFlow.

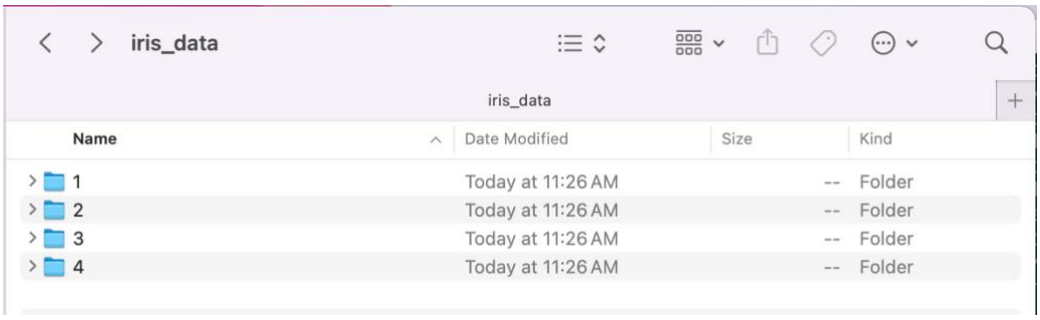
This example includes data preparation, model setup, training, and accuracy testing, using a basic form of the ResNet block for learning from the 28x28 pixel images.

While MNIST is relatively straightforward and does not require very deep networks, using ResNet illustrates how it might approach more complex image recognition tasks. The accuracy achieves will depend on factors such as the number of epochs trained for and how it configured the network, but it should be expected that high performance given the simplicity of the MNIST dataset.

In Practice

Data Selection

In this work, randomly chose 4 people's iris images in the dataset from *CASIA Iris Image Database*.



Preprocessing

For the data preparation of polar images for the CNN, this process involves capturing the iris between the inner and outer circles of the eye. Drawing on techniques from Libor Masek's study, "Recognition of Human Iris Patterns for Biometric Identification," this process utilizes these methods:

Image Acquisition

Obtain high-resolution eye images using suitable imaging equipment.

Edge Detection

Apply the Canny edge detector to identify potential boundaries of the iris and pupil in the eye images.

Circle Detection via Hough Transform

Use the circular Hough transform to detect the circular shapes of the iris and pupil based on the edges identified.

Accumulate votes for potential circle parameters (center coordinates and radii) to identify the most likely candidates for the iris and pupil edges.

Isolation of the Iris Region

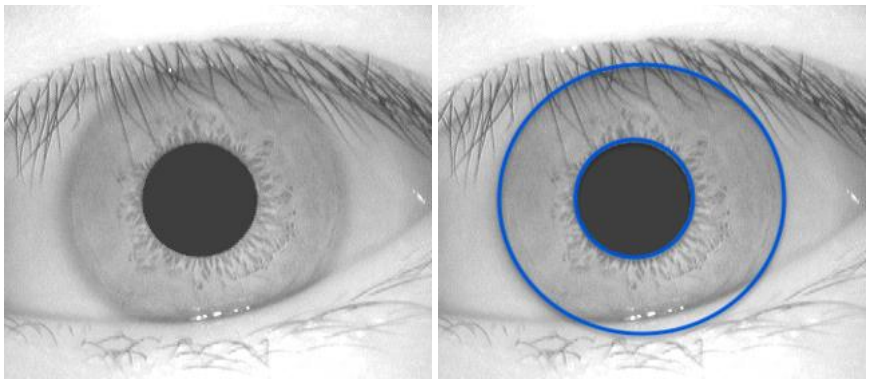
Extract the iris region by using the detected circle parameters to mask out non-iris parts such as the pupil and the sclera (the white part of the eye).

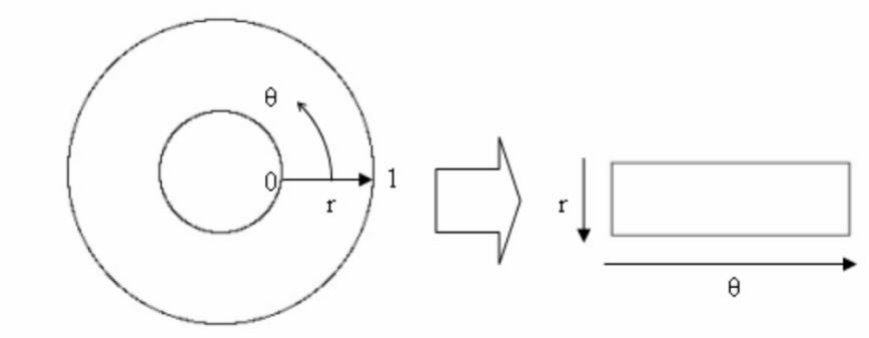
Normalization

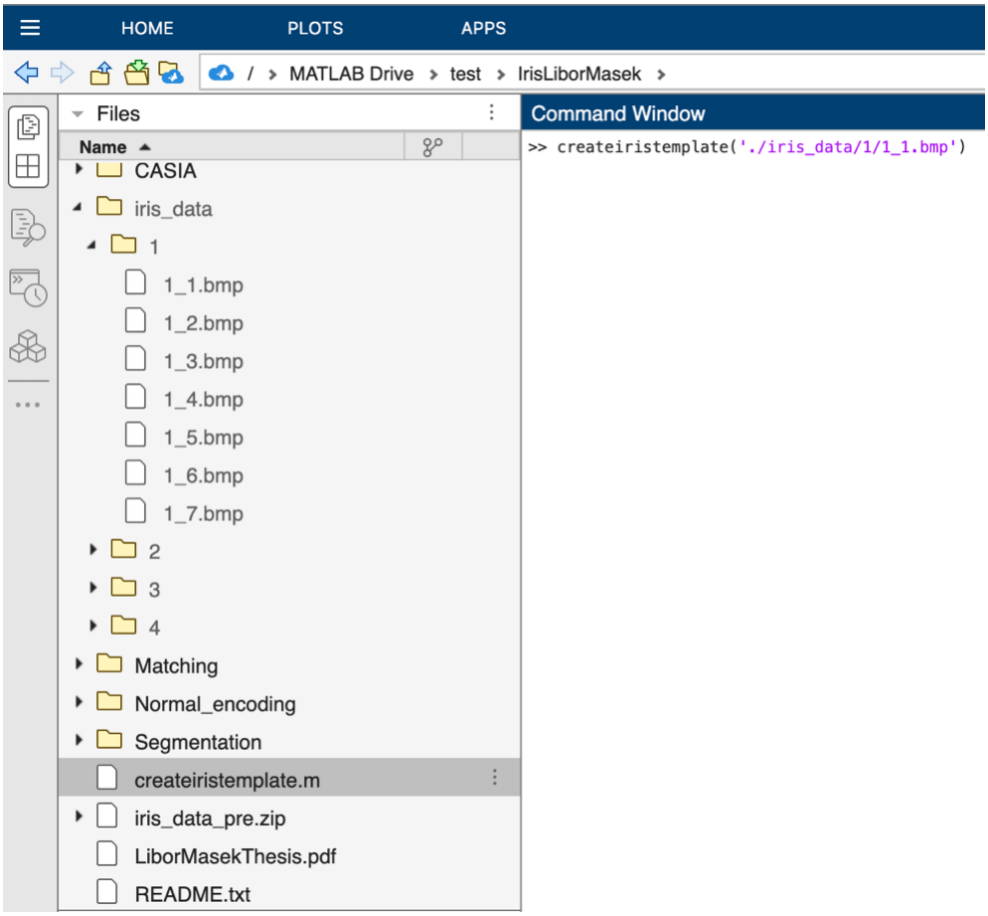
Normalize the extracted iris region to a rectangular form to standardize the size and dimensions, making it suitable for feature extraction and comparison.

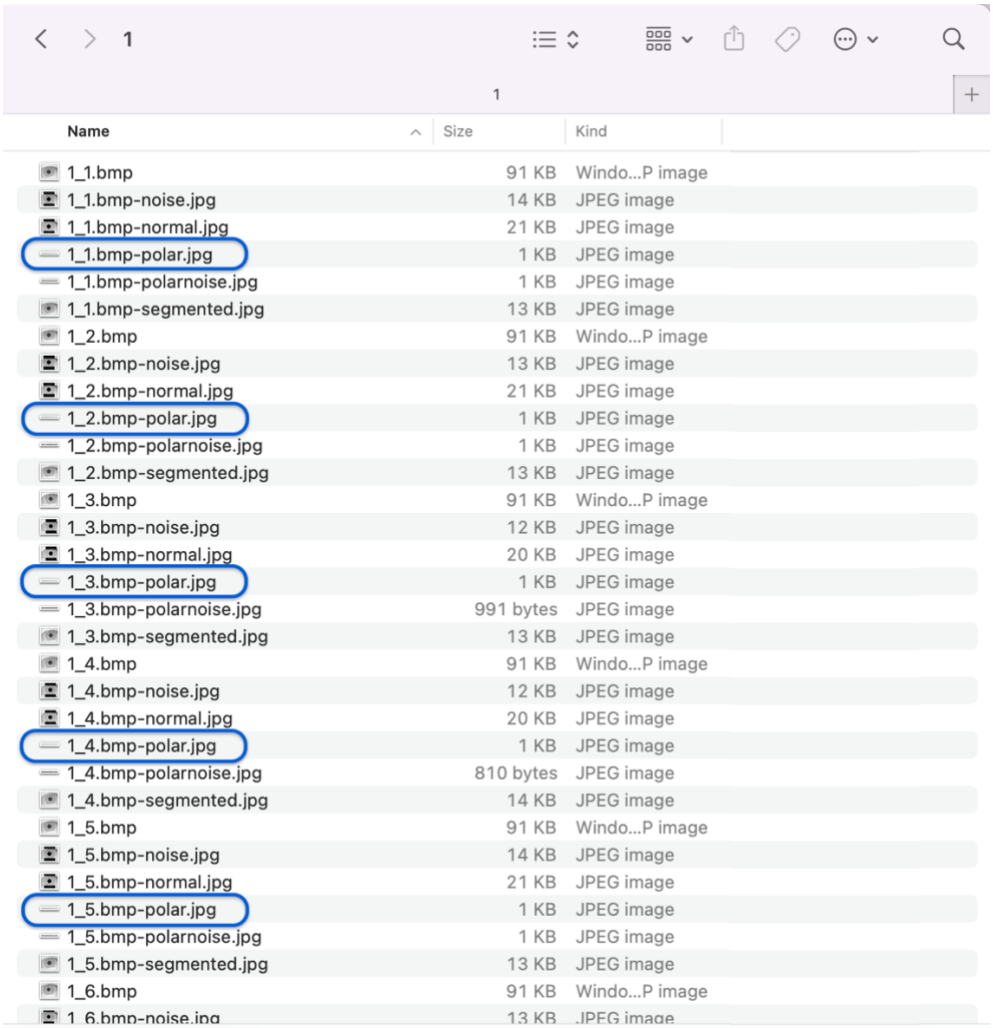
to enhance the clarity and detail of the iris images before they are input into the CNN.

This step ensures that the features relevant for biometric identification are effectively accessible.



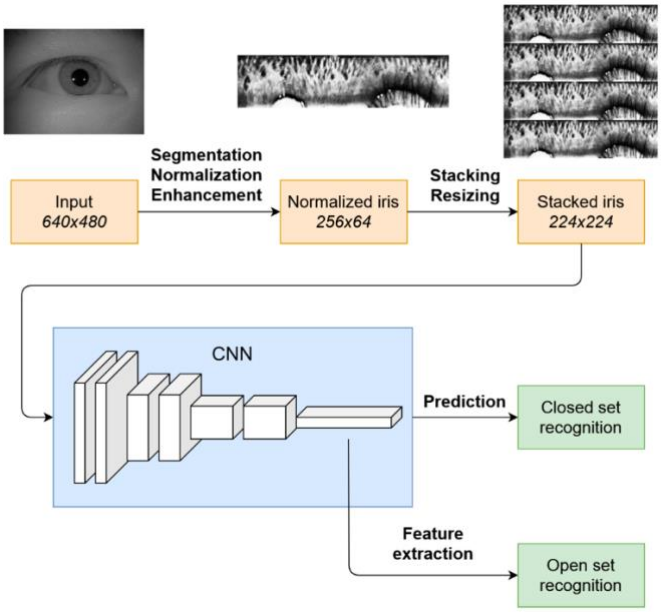



HOME
PLOTS
APPS
/ > MATLAB Drive > test > IrisLiborMasek >
Files
Name
CASIA
iris_data
1
1_1.bmp
1_2.bmp
1_3.bmp
1_4.bmp
1_5.bmp
1_6.bmp
1_7.bmp
2
3
4
Matching
Normal_encoding
Segmentation
createiristemplate.m
iris_data_pre.zip
LiborMasekThesis.pdf
README.txt
Command Window
>> createiristemplate('./iris_data/1/1_1.bmp')

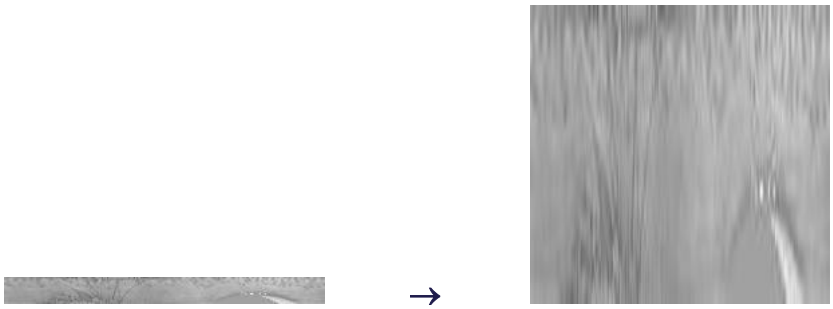


Stretching the Polar Iris Images to be Square for Training Input

While inspired by stacking the resized images from Andrej Hafner's Recognition pipeline, I propose resizing the image to be square.




```
from PIL import Image
image = Image.open("iris_image.jpg")
resized_image = image.resize((224, 224), Image.Resampling.LANCZOS)
resized_image.save(iris_image _stretch.jpg')
```



Summary

Inspired by Andrej Hafner's Recognition pipeline, which involves stacking and resizing iris images post-segmentation, I opted to resize the iris images by stretching them to a square format. Additionally, drawing from the work of Shervin Minaee and Amirali Abdolrashidi, who utilized ResNet in their end-to-end deep learning framework for iris recognition, I applied a similar approach using a residual convolutional neural network (CNN) by inputting segmented polar images. This method enhances the vision model's capability, allowing for more effective training. My training pipeline leverages segmented iris images, feeding them into ResNet to accurately identify iris biometry. This integration aims to optimize the accuracy and efficiency of the iris recognition process.

Code

ResNet_Iris.html

ResNet_Iris.ipynb

Challenges and Future Developments

When developing iris recognition systems using convolutional neural networks (CNNs), a significant challenge arises from the limited amount of input data available per person. This scarcity of data per class can hinder the model's ability to accurately learn and generalize, as deep learning models, particularly CNNs, require substantial datasets to train effectively. Such constraints often lead to models that are not adequately robust, making them prone to overfitting where they perform well on training data but fail to predict new, unseen data accurately.

To mitigate these issues, implementing preprocessing steps such as Region of Interest (ROI) checks and image enhancement becomes crucial. ROI checks help by isolating the iris from less informative regions of the eye, focusing the model's training on relevant features. Additionally, image enhancement techniques can improve the quality and consistency of the input images, making the limited data more effective for training. Despite these efforts, the inherent limitation of data per person remains a fundamental challenge, often necessitating further strategies like data augmentation or hybrid models that combine classical image processing techniques with deep learning to enhance performance and generalization capabilities of the system.