Iris Feature Extraction Using Convolutional Neural Networks

Research seminar

Image Based Biometry 2020/21, Faculty of Computer and Information Science, University of Ljubljana

Andrej Hafner

***Abstract*—Iris recognition refers to the automated process of individual recognition based on the patterns in their irises. Due to its uniqueness it's a common modality used in biometric recognition. With a technique pioneered by Daugman, it was shown that it enables recognition with very low false match rates. We adapt the pipeline defined by Daugman and introduce convolutional neural networks (CNN) to function as feature extractors. We train the CNNs on a part of CASIA-Iris-Thousand dataset for closed set prediction. Trained models are then used as feature extractor, which are used in recognition. With DenseNet-201 we achieve 97.3% recognition accuracy in closed set recognition and 98.5% recognition accuracy in open set recognition.**

I. Introduction

Iris recognition is a process, in which an individual is identified with the use of patterns in their irises. Uniqueness of the iris is crucial, since this enables us to uniquely identify an individual. In this work we adapt the traditional pipeline invented by Daugman [1]. We use convolutional neural network (CNN) for closed and open set iris recognition. In closed set recognition CNN are trained and then used to predict the identity of the user. Since this approach is impractical in real life system, we focus on open set recognition, where templates are easily added to the database, without the need for model retraining. In open set recognition we use CNNs, trained on closed set, as feature extractors. We recognize subjects by calculating cosine similarity between the feature vectors. Models are trained and evaluated on the CASIA-Iris-Thousand database. We start by an overview of related works and then continue to the methodology, where we describe the dataset, define the recognition pipeline and set up the experimental protocol. In results we first describe the metrics used for evaluation and then discuss the results of closed and open set recognition.

II. Related works

We can attribute the success of iris recognition to the development of efficient feature descriptors. The first descriptors, developed by Daugman [1], achieved really good results and are still in use today in many systems worldwide. He used Gabor phase-quadrant feature descriptor on a segmented and normalized image of the iris. He matched the descriptors using Hamming distance.

Other handcrafted feature descriptors were also used, such as Discrete Fourier transforms (DFT) [2] and class-specific weight maps, learned from the images of the same class [3]. Authors of [4] used off-the-shelf CNNs to extract feature vectors, which were then used to train support vector machine (SVM). They extracted features from different layers throughout the neural network and achieved best performance using outputs from the layers in the middle. In [5] authors compare the approaches of feature extraction using CNNs for iris recognition. They compared the performance of using existing models, training them from scratch or finetuning. Feature dimensionality was reduced using principal component analysis (PCA) and the outputs were fed to a one-versus-rest SVM for classification. They achieved best results using finetuned models, the approach we also use in this research.

III. Methodology

In this chapter we first describe the iris dataset used. We continue with the description of CNNs used and their function as feature extractors. Next, we define the recognition pipeline and explain the difference between open and closed set recognition. We finish the chapter with the definition of the experimental protocol applied.

A. Dataset

CASIA-Iris-Thousand is an iris dataset collected by the Chinese Academy of Sciences Institute of Automation (CASIA) [6]. It contains 20000 iris images from 1000 subjects, collected by an IKEMB-100 camera in near-infrared spectrum (NIR). There are 10 left and 10 right eye images for each subject, some containing eyeglasses and specular reflections, which increases the intra-class variation. Since each iris is considered to be unique (of the same person), we define the image of a specific eye as it's own class, thus acquiring 2000 classes from the dataset. Size of the images is 640×480. We can see two examples from the dataset in figure (1).

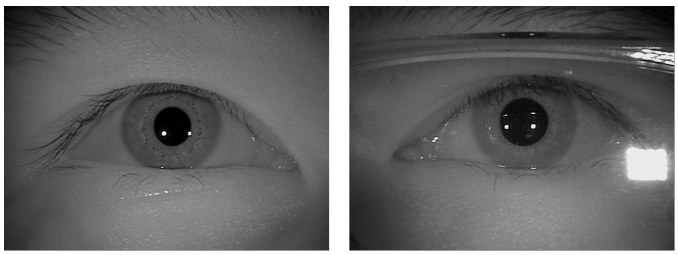

Figure 1. Example images from the CASIA-Iris-Thousand dataset. Images are from left and right eye of two different subjects. The subject on the right is wearing glasses.

B. CNNs and feature extraction

CNNs have been dominating the field of computer vision. In this research we use ResNet-101 [7] and DenseNet-201 [8] architectures, both intended for image classification tasks. ResNet introduces the concept of skip or residual connections, where the outputs skip a layer and are feed into the next layer. This helps solve the problem of vanishing gradients. DenseNet utilizes the concept of densely connected layers. Each layer obtains additional inputs from all the preceding layers and each layer also passes it's own feature maps to subsequent layers. The inputs to the layers are concatenated with the passed feature maps. This enables the network to be more compact, increasing the computational and memory efficiency.

We use finetuning to train our models. In finetuning, we initialize our models with pretrained weights, trained on a

different dataset. This helps the models to learn faster and achieve better accuracy, since the earlier layers already contain filters, trained to detect edges and color blobs, which are present in most of the images. It is also a better approach, when we don't have a sufficient amount of data, which is true in our case (10 images per class) [9]. Both of our architectures were pretrained on ImageNet dataset [10]. Because of this, their final layers contain 1000 outputs, since the ImageNet dataset contains 1000 classes. We reshaped the final layers of both networks to match the number of classes in our closed dataset, which we describe in chapter (III-D).

Once trained on a closed set, CNNs can function as feature extractors. The output of each layer is a descriptor of the input to the network. The earlier layers retain coarser information and the later layers finer and more abstract information. In both architectures we flatten the output of the last average pooling layer, receiving a 2048-dimensional feature vector.

C. Recognition pipeline

We define two recognition pipelines, first one for closed set recognition and second one for open set recognition. Closed set recognition is when we recognize only the identities which were used in training for our models. Recognition is done by inputting the preprocessed iris image to the model, then the identity is found by finding the maximum output on the final layer. This creates a problem, since adding a new identity would require reshaping the final layer and retraining the network. This wouldn't be practical in a real life system, where many identities could be enrolled on a daily basis. To solve this problem, we use our trained models as feature extractors, which enables us to do open set recognition. We extract a feature vector from the input of the image, which is used as an identifier for the class and enrolled to a database. When doing recognition, we again pass the iris image to the model and extract the feature vector. We compute the similarity between this vector and all the vectors enrolled in our database. Recognition is done by finding the subject, which has the corresponding vector with the highest similarity to the vector we retrieved from the model. This enables us to easily enroll new identities to our database, without retraining the models.

First stage of recognition is shared by both pipelines, which we can see in the figure (2). We use a similar approach to traditional iris recognition, first defined by Daugman [1]. Modified method was defined by the authors of [11] and the code was taken from their GitHub repository. Iris is first segmented from the image using weighted adaptive Hough transform and ellipsopolar transform (*wahet*). We normalize the segmented iris to size 256×64. Next, we enhance the normalized image by increasing the contrast between the lighter and darker areas. This accentuates the features of the iris that make it unique. We then stack copies of the enhanced image on top of each other four times, getting an image of size 256×256. This enables us to get a square image, which is required by the model. Finally we resize the image to the input size of 224×224.

In the second stage of the pipeline we use the CNN for closed set recognition. Irises we want to identify are input to the CNN and the subject is recognized by looking for the maximum output on the final layer. For open set recognition we input the iris to the network and then take the flattened output of the last average pooling layer in both architectures as a 2048-dimensional feature vector. At enrollment phase images feature vectors are saved to a database. When trying to recognize

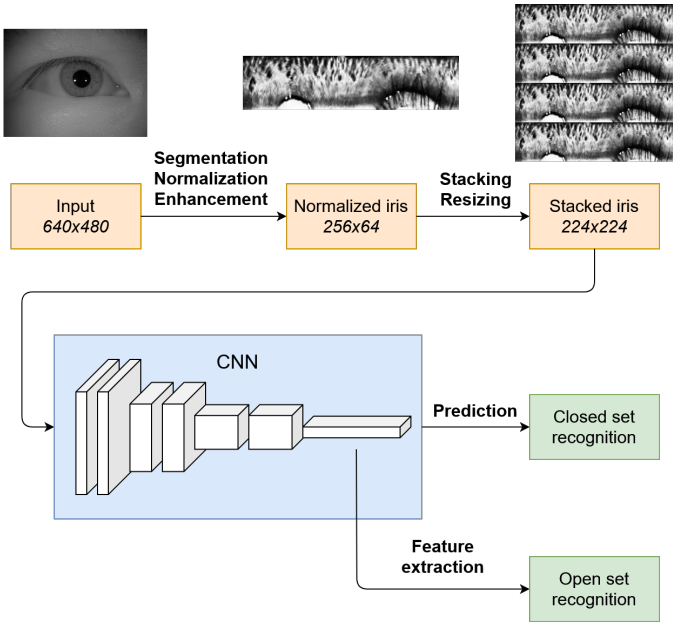

Figure 2. Iris recognition pipeline for open and closed sets. Iris is segmented from the image using weighted adaptive Hough transform and ellipsopolar transform, then normalized to a size of 256×64. We then increase the contrast in the image to accentuate the features of the iris. Output is then stacked on top of each other four times in order to get a square image, then resized to 224×224, which is the size required by the CNN. CNN is then used for prediction in closed set recognition and as a feature extractor for open set recognition.

an iris, we again feed it to the CNN and extract a feature vector. We then compute the cosine similarity (1) between the the latter feature vector and all of the feature vectors in the database.

$$cos(x, y) = \frac{x \cdot y}{||x|| \cdot ||y||} \tag{1}$$

Subject that has a corresponding feature vector with the highest similarity is matched.

D. Experimental protocol

For our experiments we split the dataset into two parts. The first part is used for closed set recognition and to train CNNs for feature extraction. It contains 1500 classes. For each class we choose 7 images for training, 2 for validation and 1 for testing. The second part consists of 500 classes and is used for open set recognition. We use 7 images per class for enrollment into the database and 3 images per class for testing.

We trained both ResNet-101 and DenseNet-201 using Adam optimizer and categorical cross-entropy loss function. ResNet-101 was trained with learning rate $2 \cdot 10^{-5}$ for 80 epochs and DenseNet-201 with learning rate 10^{-4} for 80 epochs. Both models were trained on Nvidia GTX 980 Ti GPU with 8GB VRAM. Both models and their weights were taken from the Pytorch library [12].

For open set evaluation we used both of the models trained for closed set recognition. For each class we enroll 7 images and test the performance using 3 images. We also evaluate open set recognition using 5-fold cross validation. In each fold we enroll 8 images and use 2 for testing.

IV. Results

In this chapter we first define the metrics used for recognition evaluation. We then present the results of the closed and open set recognition, followed by the discussion.

A. Metrics

We evaluate the performance of closed and open set recognition with rank-1 accuracy, rank-5 accuracy and with cumulative match-score curve (CMC). In closed set recognition we can apply softmax function to the outputs of the final linear layer. This gives a probability estimation for each of the classes, as the sum of the output equals 1. In open set recognition we receive cosine similarities between all the pairs and we select the highest similarity in each class (one out of seven). This way we get a similarity between the feature vector being evaluated and each of the classes. The value of cosine similarity is between 0 (vectors are orthogonal) and 1 (vectors are parallel), when vectors are normalized. Next, we order the probabilities in closed set recognition and similarities in open set recognition in descending order. Rank-1 accuracy is then defined by the number of the ground truth labels that are equal to the corresponding label of class with the highest probability or similarity, divided by the number of all test cases. We can define rang-n accuracy in the same way, but in this case we check for equality of labels in the first n highest probability or similarity predicted labels, again divided by the number of all test cases. This means that rank-k accuracy, where k is equal to the number of classes, is always equal to 100%. We plot a CMC curve by evaluating rank-n accuracy for ranks from 1 to number of all classes. Y axis is then the rank-n accuracy and X axis is the rank. A CMC curve tells us how the recognition accuracy is changing with increasing ranks.

B. Closed set evaluation

Results of closed set evaluation are presented in table (I).

Table I
Closed set recognition evaluation results

Model	Rank-1 accuracy	Rank-5 accuracy
DenseNet-201	**97.3%**	**99.3%**
ResNet-101	96.2%	98.7%

We can see that DenseNet-201 outperformed ResNet-101 in both rank-1 and rank-5 accuracy. Both of the models performed very well, reaching higher than 96% rank-1 accuracy. In figure (3) we can see, that rank-n accuracy is increasing fast up to $n = 6$, then it slows down. This means that most of the correct recognition's are in the first 6 highest probability predictions. At around rank-50 recognition accuracy comes really close to 100%. Incorrectly recognized test cases can be attributed to noisy data, which leads to mistakes in iris segmentation, as seen in figure (5).

C. Open set evaluation

In open set recognition evaluation DenseNet-201 again outperforms ResNet-101 in feature extraction function of CNN (table (II)).

In order to ensure that the test cases were not selected in such a way, that would increase the accuracy, we also cross validated both of our models (results are denoted with CV next to the model name). DenseNet-201 comes on top again,

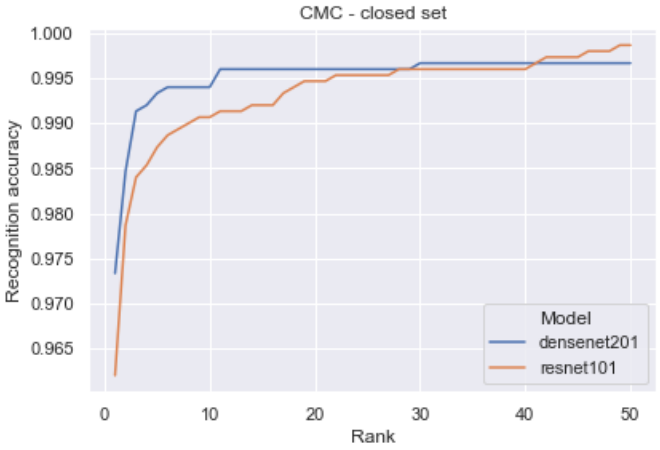

Figure 3. CMC curves from close set recognition evaluation. In the legend we can see which CMC curve belongs to the evaluation of which model. We plot CMC curves to rank 50, since the accuracy stops increasing until rank is equal to the number of all classes. It also makes the graph more readable.

Table II
Open set recognition evaluation results

Model	Rank-1 accuracy	Rank-5 accuracy
DenseNet-201	**98.7%**	**99.6%**
ResNet-101	98.4%	99.2%
DenseNet-201 CV	**98.5%** (±0.2%)	**99.2%** (±0.1%)
ResNet-101 CV	97.7% (±0.3%)	98.9% (±0.2%)

achieving almost the same accuracy as in normal testing. Standard deviation across the folds is below 0.3% for both of the models, meaning that no part of the images of a certain class stands out in better or poorer performance. Same holds true across different rank-n accuracies, as seen in CMC curves in figure (4), where the coloured area around the curves represents the standard deviation. The authors of [4] used Daugman's implementation as a baseline on the CASIA-Iris-Thousand database and achieved 90.7% recognition accuracy. We can compare our results to their baseline estimation, although we need to consider that they tested it on the whole database.

In figure (4) we can see that even at rank 50 we don't reach 100% recognition accuracy. The reason for this are probably the glasses, that the subjects are wearing in images. In figure (5), we can see that glasses create a reflection, which intersects with the iris. This prevents the *wahet* algorithm from good segmentation and normalization of the iris. In turn this leads to the CNNs extracting feature vectors from the stacked image, which are not good descriptors of the iris. We could solve this problem by increasing the robustness of the first stage in our recognition pipeline. In real life systems we could also rate the quality of segmented iris. If the quality would be too low, we would reject the recognition and ask the subject to try again.

V. Conclusion

In this research we approach the task of iris recognition with CNNs. Irises are segmented and normalized from the images using the *wahet* algorithm. We finetune DenseNet-201 and ResNet-101 on a subset of the CASIA-Iris-Thousand dataset for closed set recognition. Models are then reused as feature

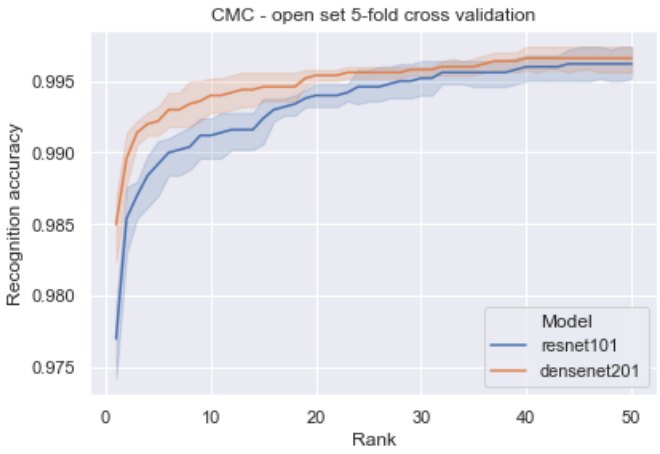

Figure 4. CMC curves from open set recognition cross validation. Curves belong to the models as described in the legend. Coloured area around the curve defines the standard deviation calculated between all the folds in cross validation. Values for specific rank-n accuracies are averaged over all the folds. We plot CMC curves to rank 50, since the accuracy stops increasing until rank is equal to the number of all classes. It also makes the graph more readable.

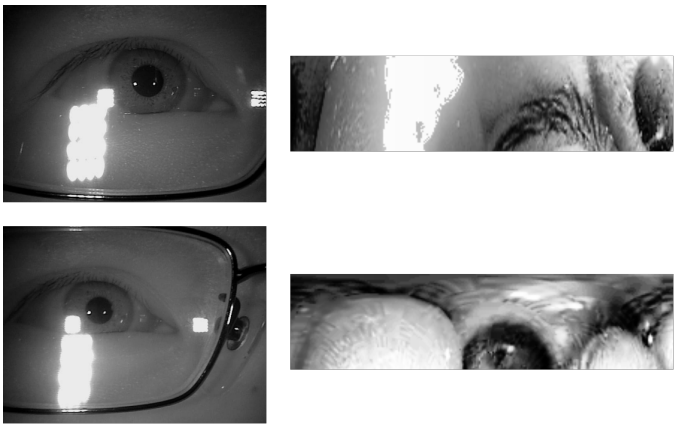

Figure 5. Examples of false matches in the open set recognition evaluation. Subjects in these two images are wearing glasses, which create a reflection over the iris. On the right we can see the result of segmentation, normalization and enhancement. There is almost no sign of iris structure, which makes each iris unique (we can see a good example in figure (2)). The lack of structure prevents the CNN to extract discriminative features, thus a false match is made.

extractors for open set recognition. We show that finetuned CNNs can produce discriminative features, with which we can achieve very high recognition accuracy. We also show that initializing CNNs with weights pretrained on other datasest can lead to a speedup in training. It also helps if the amount of data per class is small.

We make the implementation and the models publicly available on our GitHub page [13].

References

[1] J. Daugman, "How iris recognition works," in *The essential guide to image processing*. Elsevier, 2009, pp. 715–739.

[2] K. Miyazawa, K. Ito, T. Aoki, K. Kobayashi, and H. Nakajima, "An effective approach for iris recognition using phase-based image matching," *IEEE transactions on pattern analysis and machine intelligence*, vol. 30, no. 10, pp. 1741–1756, 2008.

[3] W. Dong, Z. Sun, and T. Tan, "Iris matching based on personalized weight map," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 9, pp. 1744–1757, 2010.

[4] K. Nguyen, C. Fookes, A. Ross, and S. Sridharan, "Iris recognition with off-the-shelf cnn features: A deep learning perspective," *IEEE Access*, vol. 6, pp. 18 848–18 855, 2017.

[5] A. Boyd, A. Czajka, and K. Bowyer, "Deep learning-based feature extraction in iris recognition: Use existing models, fine-tune or train from scratch?" *arXiv preprint arXiv:2002.08916*, 2020.

[6] Chinese Academy of Sciences Institute of Automation. (Jan. 2020), "Casia iris image database." [Online]. Available: http://biometrics.idealtest.org

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[8] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[9] R. Cadène, N. Thome, and M. Cord, "Master's thesis: Deep learning for visual recognition," *arXiv preprint arXiv:1610.05567*, 2016.

[10] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[11] C. Rathgeb, A. Uhl, P. Wild, and H. Hofbauer, "Design decisions for an iris recognition sdk," in *Handbook of Iris Recognition*, second edition ed., ser. Advances in Computer Vision and Pattern Recognition, K. Bowyer and M. J. Burge, Eds. Springer, 2016.

[12] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[13] "Publicly available implementation (Accessed: 17.1.2021)." [Online]. Available: https://github.com/AndrejHafner/iris-recognition-cnn