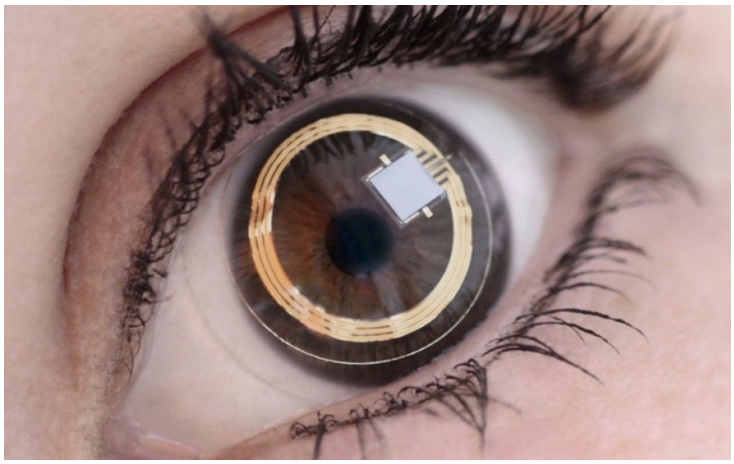
An iris recognition framework in presence of colored and transparent lenses

Chiara Giacanelli, Leonardo Paolucci
Computer Science Departement, Sapienza University of Rome
Biometric Systems Project



Abstract

This report presents our project for the Biometric Systems' course. Starting from a baseline common iris recognition framework, we built a system able to do a closed-set identification and verification with irises without lens, with transparent lens and with colored lens.

1 Introduction

Iris recognition is becoming one of the most reliable modalities for biometric recognition. It consists of finding unique iris muscular patterns in order to identify or verify the identity of an individual.
The iris has the great mathematical advantage that its pattern variability among different persons is enormous. In addition, as an internal (yet externally visible) organ of the eye, the iris is well protected from the environment and stable over time [1].
Despite this, the fast deployment of iris recognition technologies in real applications has increased the concerns about its security, along with the exponential growth of people who wear contact lenses instead of glasses to deal with health reasons. Contact Lenses are, however, increasingly being used for cosmetic reasons also where texture and color of the iris region is superimposed with thin textured lens. In addition, recent studies state that colored lens can be useful to reduce headaches; a survey presented on the 10th May of 2023, in fact, found that the top five issues that were helped by the colored lenses are: light sensitivity at 91%; eye strain at 82%; reading problems and print distortions at 73%; headaches at 65%; and fatigue at 62% [5].
The usage of colored, as well as transparent lens, changes the texture of the iris and it can lead to a decrease of the overall performance of the classical iris recognition models.
With our project our aim was to study the behaviour of a basic Iris Recognition model in presence of irises without lenses, with transparent lenses and with colored lenses, to show the differences with the benchmark model and to improve it to match the necessities of such irises' images.
We built an iris recognition system with the following characteristics: it's supposed to work in a **controlled setting** and with **cooperative users**; it performs **closed-set identification** and **verification**; it can be given as input both right and left eye and irises captured by different infrared sensors.

2 Related Works

Revisiting Iris Recognition with Color Cosmetic Contact Lenses [2]: this study, conducted by the IIIT-Delhi University, presents an in-depth analysis of the effect of contact lenses on the iris recognition algorithms' performance. They studied the performance by:

- Varying the gallery probe combinations, to see the effect of different types of lenses on iris recognition;
- Varying the acquisition device (performed different experiments based on two different sensors, Cogent and Vista);

They used the VeryEye iris recognition technology to assess the performance and they concluded that color cosmetic lens significantly increases the false rejection at a fixed false acceptance rate. Also, the experiments on four existing lens detection algorithms suggested that incorporating lens detection helps in maintaining the iris recognition performance.

Observing the Effects of Colored Lenses on Iris Recognition using Feature Extraction Technique [3]: in this paper, the authors discuss the presence of a contact lens, particularly a colored lens in iris recognition frameworks, and the challenges it brings as it alters the natural iris patterns. They present a feature extraction method to remove the effect of colored contact lenses and make it more secure. This has proven to be very efficient and also less space consuming.

This feature extraction method is composed of 6 modules followed during the authentication process:

1. Image conversion
2. Edge detection
3. Pupil detection
4. Normalization
5. Feature extraction
6. Matching

Matching is done by comparing the similarity between the feature vectors, so they didn't develop any machine learning model.

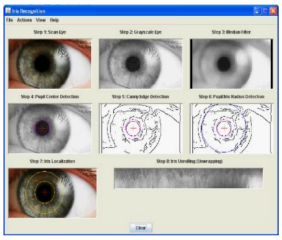
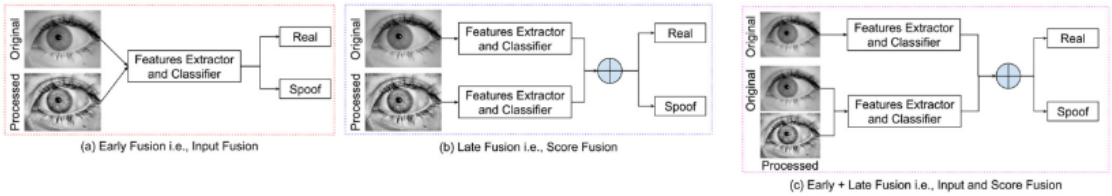


Fig 2 The different phases of the recognition

Image comparison is success:

Generalized Contact Lens Iris Presentation Attack Detection [4]: in this very recent study (2022) the authors described how iris recognition systems are subjected to presentation attacks (a presentation attack occurs when a bad actor, or fraud perpetrator, uses someone else's physical characteristics or biometric data, commonly known as "spoofs," to impersonate someone else). The iris presentation attacks described in the paper are: iris printout, a replay of iris video, synthetic irises, prosthetic eyes, and 3D contact lenses. The authors developed a novel generalized 3D contact lens iris PA detection system based on the concept of multiple stages of fusion: a deep CNN model is trained on two different input information and later fused using weighted score fusion.



(a) Early Fusion i.e., Input Fusion

(b) Late Fusion i.e., Score Fusion

(c) Early + Late Fusion i.e., Input and Score Fusion

3 Dataset Description

We started our project by collecting the available datasets for the topic. We selected two of them in particular: the UBIRIS.v1 and IIIT-D Contact Lens Iris Database.

3.1 UBIRIS.v1

UBIRIS.v1 database is composed of 1877 images collected from 241 persons during September, 2004 in two distinct sessions. Its most relevant characteristic is to incorporate images with several noise factors, simulating less constrained image acquisition environments. This enables the evaluation of the robustness of iris recognition methods.
In the first session, they collected images trying to minimize noise factors, especially those relative to reflections, luminosity and contrast.
In the second session they changed the capture place in order to introduce natural luminosity factor. Images collected at this stage simulate the ones captured by a vision system without or with minimal active participation from the subjects, adding several noise problems and emulating real life conditions. These images will be on the recognition stage compared to the ones collected during the first session.

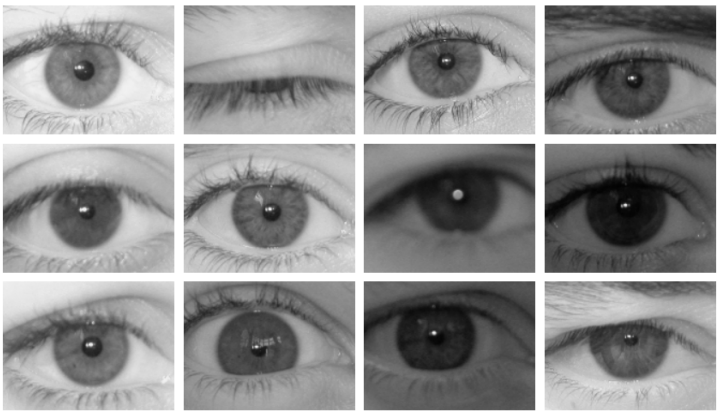


Figure 1: Images from the two sessions of the UBIRIS.v1 dataset converted into grayscale

We used this dataset in order to train the original model in the first place. This dataset doesn't include irises with contact lens.

3.2 IIIT-D Contact Lens Iris Database

The IIIT-D Contact Lens Iris Database is a dataset from the IIIT - Delhi, containing over 6500 images representing the iris of 101 users.
These images are captured using two sensors: Vista and Cogent.
For every user, the iris is captured in three ways:

- Natural: the images are of the original iris, without any presentation change;
- Colored: the iris is covered by a textured colored lenses of 4 possibile colors: hazel, blue, green and gray;
- Transparent: the iris is covered by a transparent lens;

All the images are already converted into black and white but the dataset contains also a table in which, for every person's ID, there's the correspondent color of the lens they're wearing, whether if they are male of female, and if they are wearning Bausch & Lomb lens (parameters which we didn't care about, as we considered them simply along with the colored ones).

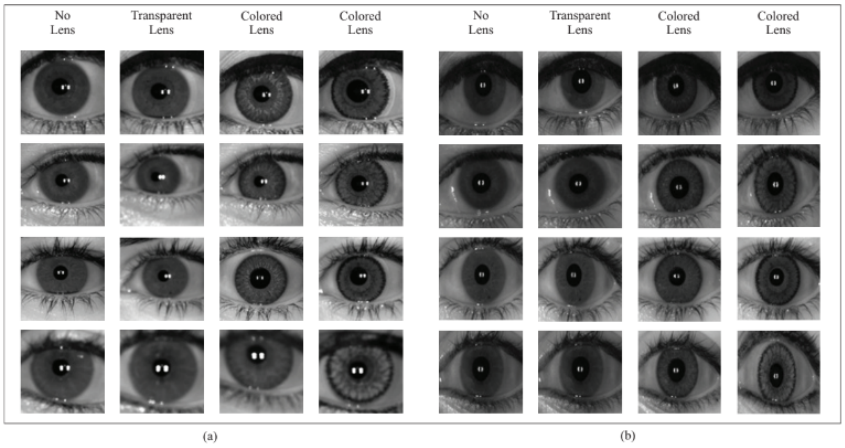


Figure 2: Iris images in IIIT-D Contact Lens Iris Database (a) images captured using Cogent iris sensor and (b) images captured using Vista iris sensor. The two color lenses used here are from CIBA Vision in third column and Bausch & Lomb in fourth column.

Number of subjects (classes)	101 (202)
Types of contact lens	Without lens, transparent, and colored
Lens manufactures	CIBA Vision and Bausch & Lomb
Lens colors	Blue, Gray, Hazel and Green
Number of subjects per colored lens type	Blue (20), Gray (29), Green (30) and Hazel (22)
Iris sensors used for acquisition	Cogent dual iris sensor and VistaFA2E iris sensor
Number of images per subject per lens type	Minimum 5 images per eye class, per lens type
Total number of image in the database	6570

Figure 3: Details of the IIT-D Database

We used the ***"IIIT-D Contact Lens Database"*** for the sake of our project.
The UBIRIS.v1 dataset is the one adopted in the benchmark model and we used it to perform comparisons regarding the image processing phase and the evaluation phase.

4 Objectives

The objective of our work was to study the behaviour of a benchmark iris recognition model in presence of a gallery in which users are enrolled with three types of irises: colored, transparent and normal.
With respect to the presented papers, after collecting information on the lack points of such method, we wanted to create an unique model able to discriminate users whether they are wearing contact lenses or not.
Being it a complex task, we conducted a deep ablation study on the preprocessing phase and model selection phase, which is accurately described below, and performed several experiments that lead us to find good approaches dealing with irises of different nature. For the sake of the project, we didn't implement a contact-lens detector, for the fact that we wanted the model to be as generalizable as possible and we didin't want to treat irises with contact lens as "impostor" images.

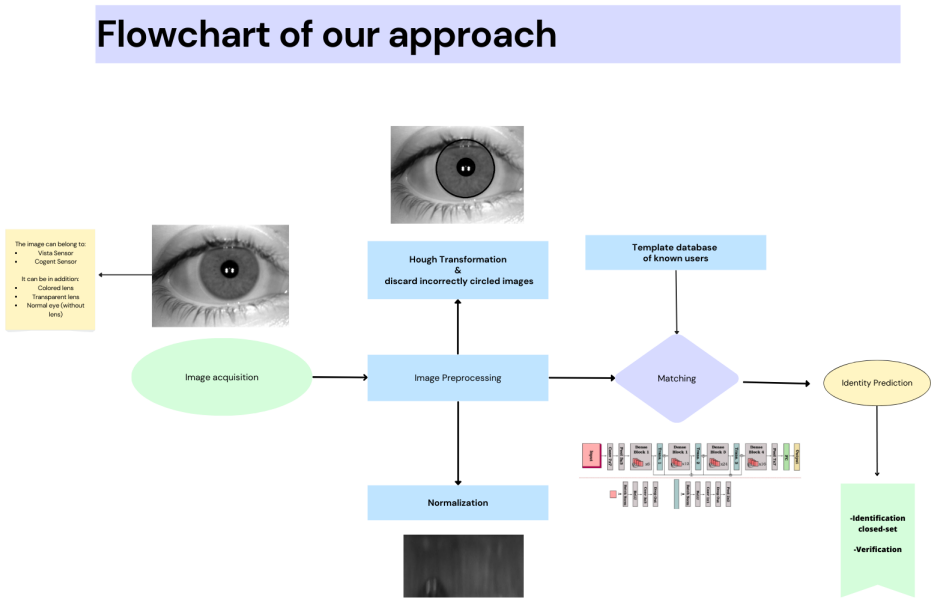


Figure 4: Flowchart of our approach

5 Proposed Method

The first step of our project has been the study of the benchmark model, available at *this link*. This project is built up in Python3, Pytorch and OpenCv and proposes a biometric system with a database of registered/unknown users and two main functions: identification open/closed-set and verification. The system uses a pretrained ResNet50 network and it preprocesses iris' images in order to give the network a normalized probe to process.
Our work was split in several steps, summarized below:

1. **Environment Setup**
2. **Dataset and Gallery organization**, along with a proper train-validation-test splitting;
3. **Preprocessing** and normalization of irises;
4. **Data augmentation**
5. **Ablation study**
6. **Final Implementation**
7. **Evaluation**

5.1 Environment Setup

The first step we had to do was setting up our work environment. We used Google Colab for our project. This environment allowed us to save our project, gallery and model in Google Drive and to access to them without any problem.
Colab permitted us to execute our Pytorch scripts and train our models with its T4 GPU and 15 GB of RAM, even though with its storage and time limitations that prevented us to train heavier models.

5.2 Dataset and Gallery Organization

The starting Dataset, IIIT-D Contact Lens Iris Database, was divided into two groups, Cogent and Vista, representing the sensors used to get the images. Inside the groups there was increasing numbered folders, one for each user, in which are contained other three folders, one for each kind of lens setting and the irises from both eyes (left and right). Inside them the actual images, saved in bitmap format.
We then organized the the dataset, following the file organizer given with the starting project, into known and unknown users and taking Cogent and Vista images, giving them names which to record user number and type of lens setting.
The renamed image would be in this format:

$$\{user\ id\}\_\{type\ of\ image\}\_\{number\ of\ image\}.bmp$$

For example:

$$99\_Colored\_19.bmp$$

We didn't consider the type of sensor relevant to keep in the name of the user.
Eventually, for each user, we put all the images in one single folder, as the aim of the project is to identify them regardless of the lenses. For the purposes of our project we then just needed known users, since we developed a biometric system in a closed-set environment.
After the preprocessing of images, we splitted the images in train and validation sets, each one with the same users and with a split of 80% for the train set and 20% for the validation set.

5.3 Preprocessing

The preprocessing phase has several stages:

- Image preparation
- Iris and pupil location
- Discard of unuseful images
- Normalization

All the images in the gallery are in the *bitmap format* and we decided to keep them in this way and not to convert them in another format, which would have probably made them loose quality and consequently some important data information necessary to process.
We first performed image preparation, applying image conversion and refinement techniques with the OpenCv library. The input images were already in grayscale, so we didn't need to convert them in the monochromatic format (as it was necessary with the old dataset).

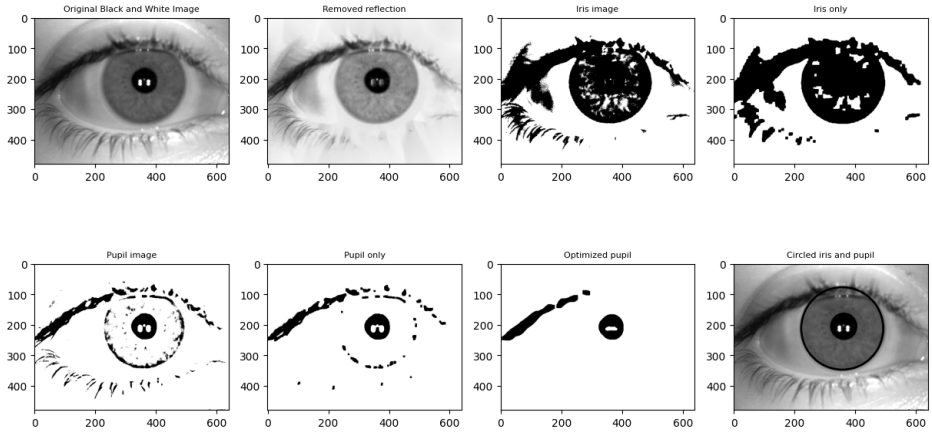


Figure 5: Preprocessing benchmark for an iris with transparent lens

The next step we conducted was the removal of any type of reflection caused by the sensor capturing the image, which could lead to a mismatch during the iris and pupil location phase, because there would have been white spots inside them that could have caused noise. This phase wasn't present in the baseline project, and has greatly improved the process of iris and pupil discovery.

We removed the reflection by:

1. **Image thresholding:** for every pixel, the same threshold value is applied. If the pixel value is smaller than the threshold, it is set to 0, otherwise it is set to a maximum value. In this case, we applied a binary thresholding, and we filter out pixels with too large values: in this case, pixels over the value of 150 are assigned the value 255;

2. **Dilation:** we apply to the thresholded image a kernel of dimension 2x2 to de-noise the resulting image;

3. **Inpainting:** the function reconstructs the selected image area from the pixel near the area boundary[1]. We use the output of dilation as the mask to inpaint the greyscale image,using the region neighborhood, resulting in the same image without areas of reflection.

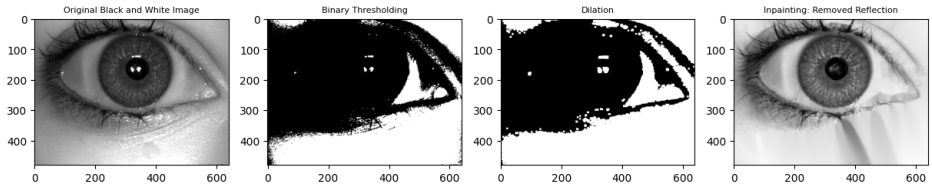


Figure 6: Steps of reflection removal for an iris with colored lens

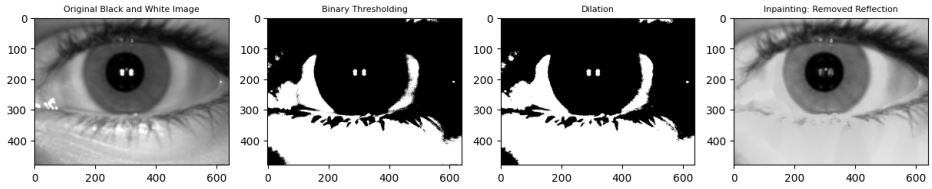
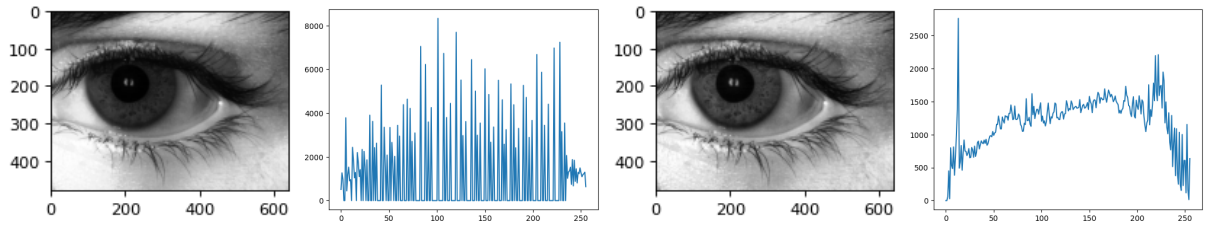


Figure 7: Steps of reflection removal for an iris without lens

As we can see from the pictures above, colored lenses tend to add more texture to the iris.
The third step we followed is the image binarization. This was necessary to detect the location of the iris. We applied contrast to the image without reflection and then computed Adaptive Histogram Equalization (CLAHE). CLAHE is a basic image processing technique that can improve an image's overall contrast without boosting noise at the same time.



(a) CLAHE Histogram Equalization to contrasted iris image without lens

Eventually, the iris image has been binarized as we can see in the "Iris Image" section in Figure 5.
In order to accurately find the iris circle, we applied other filters:

- Closing, in order to cover small holes inside the foreground and small black points on the iris;
- Erosion, in order to remove excessive noise;
- Morphological Gradient, to measure the difference between dilation and erosion of an image;

[1]From the OpenCV documentation, "OpenCV: Inpainting"

After these final steps, we were able to find the circle surrounding the area of the iris, through the Hough Transform technique, clarly explained in the following sub-section. In order to do this, we assumed the center of the iris to be the center of the image (even though it isn't always like this, we assumed our system to be in a **controlled setting**, since we would discard wrongly circled irises in the following steps).
After finding the coordinates of the location of the iris, we proceeded to localize the pupil, adding more filters to the image.
The steps followed were similar to the ones for the location of the iris, so we wouldn't explain them in detail since we did it before. Instead, we will deepen the *pupil optimization technique* we developed in the baseline project, since we realized that in most of the irises with lens, the pupil would end up coinciding with the iris circle.

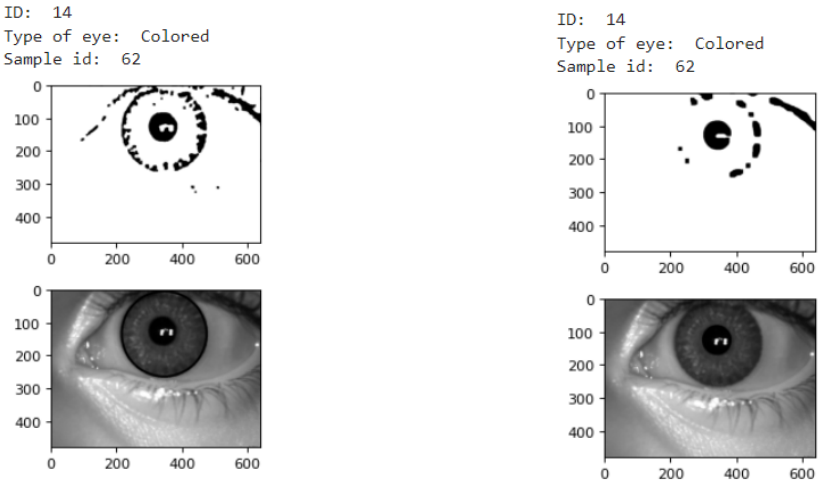


Figure 9: Pupil location without (left) and with optimization (right) for the same iris with colored lens

As we can see in the image above, the algorithm without the implemented optimization would recognize the border of the pupil as the one of the iris. This would be very frequent in irises with lens (especially the colored ones), since they appear to be **more textured** and the binarization isn't able to discriminate well between the borders of the two different circles.
With these assumptions we added the following steps to the image:

- Median blur: this operation is similar to the other averaging methods. Here, the central element of the image is replaced by the median of all the pixels in the kernel area. This operation processes the edges while removing the noise, that is what we needed in order to make the important elements of the picture (which is essentially the pupil) more compact and distinct.
- Closing, in order to cover the holes left.

These two simple operations were enough to optimize our pre-processing a lot. Finally, we returned the circled iris and pupil image and give it as input to the normalization algorithm.

5.3.1 Hough Circle Transform

We used the Hough Circle Transform technique to find the circles of the iris and the pupil. For both the elements, we pass the function the binarized image, the coordinates of the center of the image and the function returns an array of circles with their coordinates and the radius.
From this array, we take the circle coordinates that has the smallest distance from the center of the image.

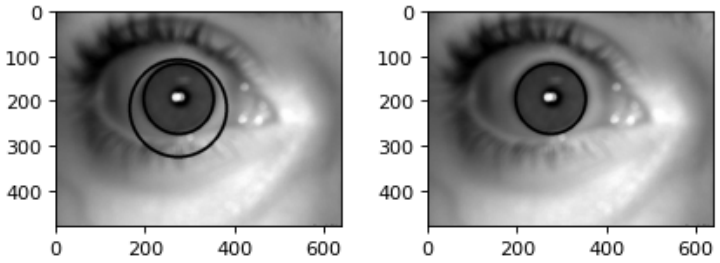


Figure 10: Hough Circle Transform to detect the location of an iris with transparent lens

5.3.2 Normalization

The last step of our preprocessing has been the normalization. Before doing this, we selected the images which weren't correctly circled that are the case in which:

- The pupil is found outside the iris;
- No circles are found;

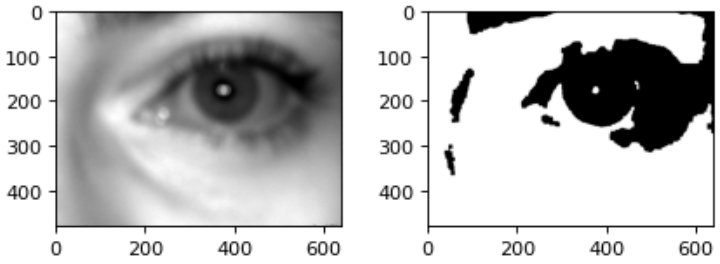


Figure 11: Example of a discarded iris image with transparent lens because no circles are found

We discarded these images and proceeded the normalization with the ones judged positively by the previous steps.
We normalized the iris region by unwrapping the circular crown region, set betwen the borders of pupil and iris, into a rectangular block of constant dimensions. We excluded the values at the boundary of the pupil iris border, and the iris scelra border as these may not correspond to areas in the iris region and will introduce noise We gave the circled image as input, along with the output vertical and horizontal dimensions of the rectangular image we wanted to return.

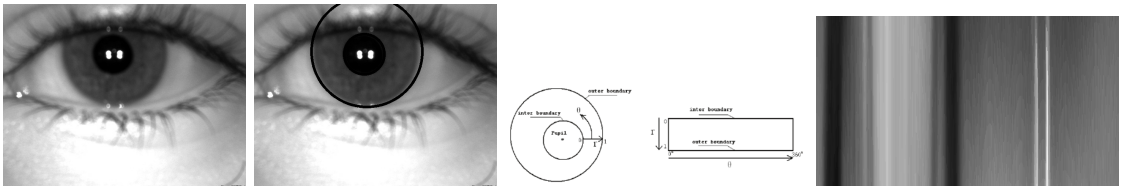
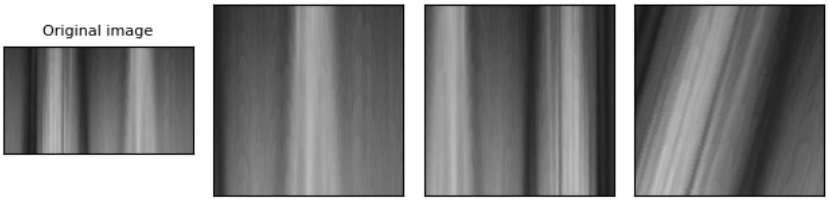


Figure 12: Normalization phase for an image with transparent lens

5.4 Data augmentation

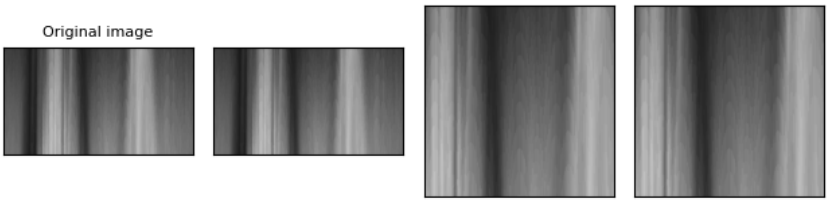
Before feeding the model the data for training, we performed some data augmentation in order to make the model generalize better and not to overfit since our dataset has various images belonging to the same users which have different features (being of lens of different colors and without lens). We performed different data augmentation techniques for the train and validation set. For the training set we performed:

1. Random Resize Cropping
2. Random Horizontal Flipping
3. Random Rotation of 90 degrees



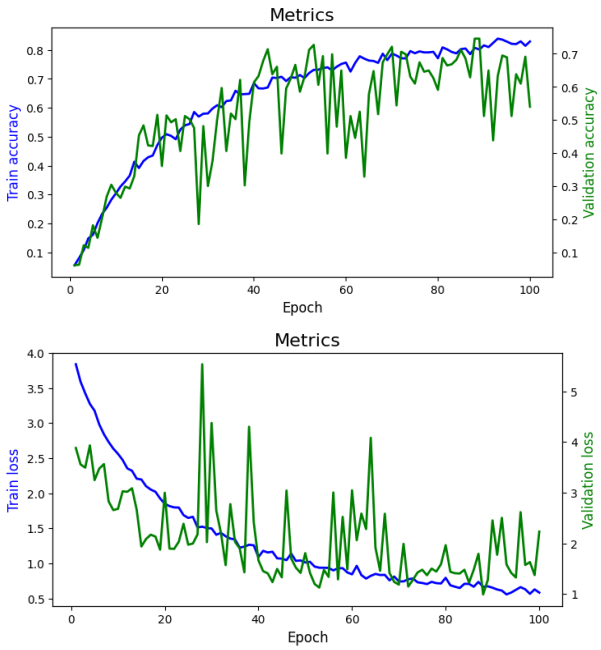
For the validation set we performed:

1. Resizing
2. Center Cropping
3. Color Jittering



5.5 Ablation study

Our experiments started by testing the baseline model with our new dataset. The baseline model used a *ResNet50* network with *learning rate= 0.0002*, *Adam* optimizer and *batch size* of 16.
As we can see from the plots below the model was (clearly) overly overfitting.



Overfitting occurs when the machine learning model isn't able to generalize well on unseen data, because it "learns the noise" provided by the training set too much.
To avoid this problem, we started doing some analysis in terms of:

- Data provided
- Hyperparameters
- Model used

5.5.1 Models

Residual Neural Networks We started by testing different pre-trained versions of the ResNet model with less layers: the ResNet18 and ResNet34. The difference between the two is in the number of layers of the network: the first hase 18 layers, and the latter has 34.
Substantially, ResNets are Deep Convolutional Neural Networks with Skip Connections, which avoid the vanishing gradient problem by connecting activations of a layer to further layers by skipping some layers in between.
In the baseline code we had a ResNet with 50 layers, and we tried to test less if having less layers would improve the performances, decreasing the weight of our model (which doesn't have a lot of data to train with).
The model better behaving was the Resnet with 18 layers (the lightest one), but the two were still too much overfitting, so we choose to try to proceed to other architectures.

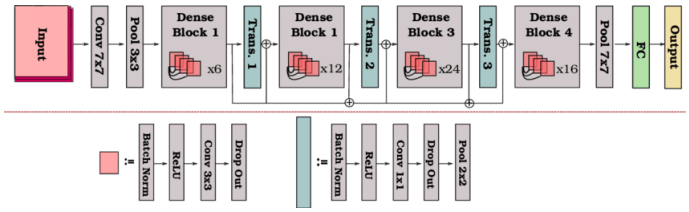
Siamese Network We then tried to build a triplet siamese network.
A triplet Siamese Network is a type of neural network that contains three identical subnetworks used to generate feature vectors for each input and compare them. We built a triplet siamese network which would compare three inputs:

- The anchor, which would be the input image;
- The positive sample, which would be a random chosen image belonging to the same class of the anchor;

- The negative sample, which would be a random chosen image belonging to another different class;

We fed these three inputs to three different densenet121 networks, and finally computed their semantic similarity. We then used the result as an input to another Sequential Neural Network which would do the classification.
Being a heavier network for the amount of data we had it would take a lot of time to converge, and during training it resulted in very poor results (40% of accuracy at the 50th epoch). Since Google Colab presented us problems of runtime timeout, we ended up by concluding that it was a too complex model for our task.

Dense Convolutional Network We finally experimented with another pre-trained network: the densenet121. In DenseNet, each layer obtains additional inputs from all preceding layers and passes on its own feature-maps to all subsequent layers. Concatenation is used, and each layer is receiving a "collective knowledge" from all preceding layers.



This collective knowledge turned out to be very impactful and important for the sake of our project, since the model was able to discriminate more because it strengthens feature propagations and encourages feature reuse. In fact, we collected good results with the first trial of training and we decided to keep this network as our benchmark model.

5.5.2 Hyperparameters

We then proceeded doing experiments on the hyperparameters on the network, since our Densenet121 was still overfitting a bit.
In particular, we studied the effect of the learning rate and batch size on the performance on the network.

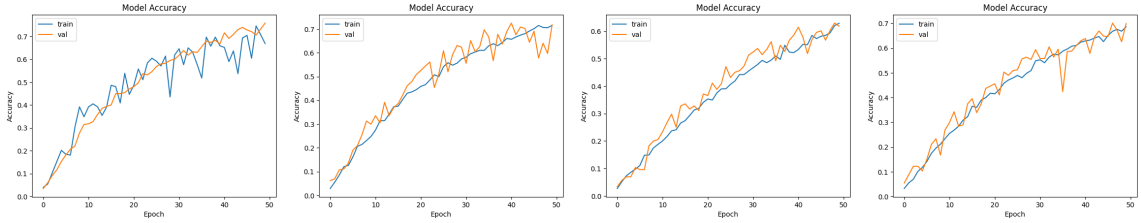


Figure 14: From left to right: densenet121 trained different learning rates and batch size (in the first plot there's a mistake in the legend: train and val must be switched colors)

We tested the network with:

- lr = 0.0002 and batch size = 16 and 32
- lr = 0.0001 and batch size = 16 and 32
- lr = 0.00005 and batch size = 16
- lr = 0.00008 and batch size = 16

We decreased the learning rate because we noticed our model was converging too fast. We eventually conveyed that 0.0001 of learning rate and batch size of 16 was the best choice for our model.

5.5.3 Data Provided

Our model was still quite unbalanced so we studied how impactful was the data we provided to it. Initially, to select the train and validation subsets, we would shuffle the dataset and take randomly 80% of it for the training and 20% of it for the validation set.
The problem was that, in our dataset, we had three macro-categories to take in consideration: irises with colored lens, irises without lens and irises with transparent lens. This could mean that, shuffling randomly the entire dataset and taking a subpart of it, we might have unbalanced data.
For this reason, we decided to make sure that, for both the sets, we had all the types of images: to select the training set, we first divided the entire dataset into three parts: the one with all the samples belonging to the "normal" images, the one with the "colored" and the one with the "transparent". All these sets had these images shuffled randomly. Then, we built the training and the validation subsets by splitting (80/20) and merging the samples.
We finally shuffled again the subsets.

```
for user_id, user_samples in samples.items():
    #We take all the indices belonging to the dataset
    indices = list(range(len(user_samples)))
    random.shuffle(indices)

    #We create the three subsets with probes belonging to the Normal, Colored, Transparent irises
    normal = [user_samples[i] for i in indices if "Normal" in user_samples[i]]
    colored = [user_samples[i] for i in indices if "Colored" in user_samples[i]]
    transparent = [user_samples[i] for i in indices if "Transparent" in user_samples[i]]

    #We initialize, for the three lists, the split variable (test_size = 0.2)
    split_normal = int(len(normal) * (1.0 - test_size))
    split_colored = int(len(colored) *(1.0-test_size))
    split_transparent = int(len(colored) *(1.0-test_size))

    #We create the train set with the names of the samples of the three lists (splitted with the indices), merged together
    train = [i for i in normal[:split_normal]] + [i for i in colored[:split_colored]] + [i for i in transparent[:split_transparent]]
    #We shuffle the train set
    random.shuffle(train)
    #We do the same for the validation set
    val = [i for i in normal[split_normal:]] + [i for i in colored[split_colored:]] + [i for i in transparent[split_transparent:]]
    random.shuffle(val)
```

Figure 15: Balanced train-val splitting

This was the key to improve our model and to reach a final implementation.

5.6 Final Implementation

To recap, our final model has the following characteristics:

- Pre-trained *densenet121* as the main network;
- Learning rate = *0.0001*;
- *Adam* optimizer;
- *16 batch size*;
- *CrossEntropy* loss;
- Trained for *50 epochs*;
- *80-20* train-val split.

6 Evaluation

To evaluate the model performances we used different metrics and graphs according to the type of task we wanted to describe:

- Accuracy and Loss, to show overall performance of our model in terms of good and bad predictions;
- For the task of **verification**: ROC curve, AUC, EER and a custom metric that we called TCR;
- For the task of **identification closed-set**: the CMC curve;

We conducted this phase with a gallery of 50 users, each one enrolled with different types of irises.

Accuracy and Loss

As a standard and simple way to understand if the model is working well, we computed the accuracy of it for each epoch of training and evaluation, that helped us selecting the best feature for the model among the ones we tried. In the end, after 50 Epochs of training, the model gets a 0.78 of accuracy in training and 0.72 in evaluation, which is a huge increment from the starting values with resnet50 of 0.63 in training and an oscillating evaluation accuracy.
As it's shown in images below, the accuracy for validation starts to get a bit lower then the train one after 35/40 epochs, but still their values a pretty similar and still the val accuracy continue to grow. We decided then to tolerate this light overfitting.

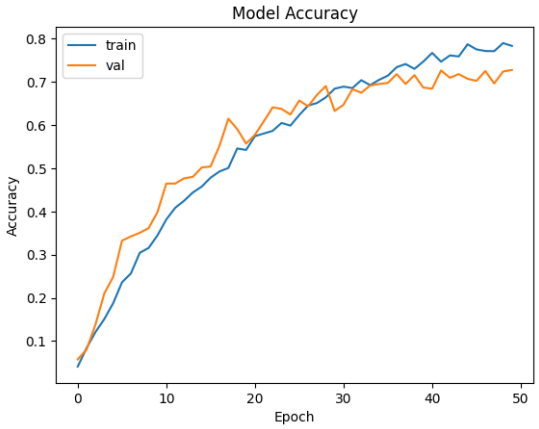


Figure 16: Accuracy curve for train and validation over the epochs

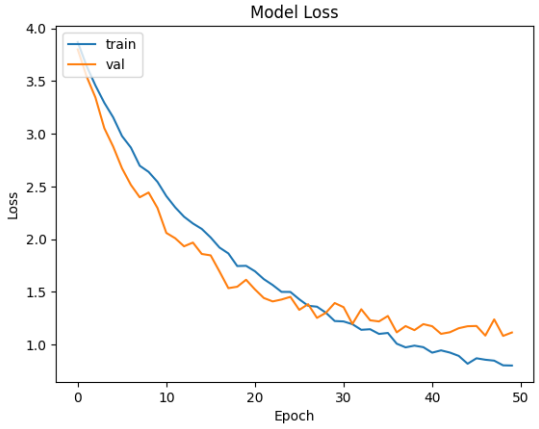


Figure 17: Loss curve for train and validation over the epochs

6.1 Verification

6.1.1 ROC Curve, AUC and EER

As a multiclass classification problem, we couldn't compute the standard binary Receiver Operating Characteristic curve, so we decided to use the OneVsAll paradigm for Multiclass ROC computation.
To compute it, we:

- elaborated the probability for each probe belonging to each gallery class;
- evaluated **False Acceptance Rate** and **False Rejection Rate** for each class, against all the other gallery samples;
- used decreasing thresholds to compute them;
- averaging all the results pairing rates calculated with the threshold at the same position in the threshold list;

As in the image, our ROC curve is pretty solid, with an AUC of 0.98 and an EER of 0.07, showing that for high thresholds the model can almost always find all the right classes, while most of the other ones continue to be correctly predicted as false.

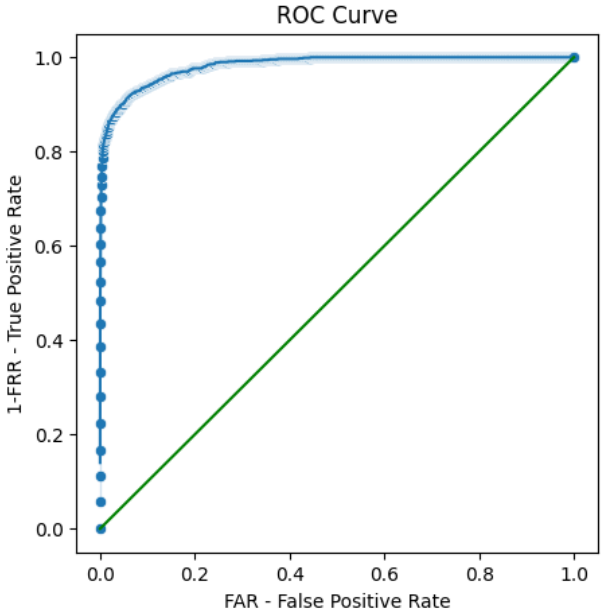


Figure 18: Multiclass OneVsAll ROC curve

```
AUC:  0.9831848354018382
EER:  0.07331008774638231
```

Figure 19: AUC and EER values

As we can see from the images above, the results are really good. This is because, even if the probability of being the right label isn't the highest, for each right probe the probability assigned is still pretty high, showing that the model does a good job at classifying.
In the following page we can see two examples of this behaviour: the first example depicts a misclassification for an iris with transparent lens in which the correct label has been assigned the second highest probability (29% against the 32% of the correct label).

```
Registered users:   4%|▌          | 40/1000 [01:26<09:30,  1.68it/s]
User id:  36
Sorted labels:  ['40', '36', '78', '26', '99', '51', '55', '38', '43', '30', '22', '50', '75', '18', '54', '68', '90', '12', '11', '13', '57', '56', '95
Sorted probabilities:  [0.3250287175178528, 0.2957902252674103, 0.284669429063797, 0.0267594326287508, 0.01300710067152977, 0.011917446739971638, 0.00768
User Path:  ../data_iiitd/system_database/registered_users/36_Transparent_32.bmp
```

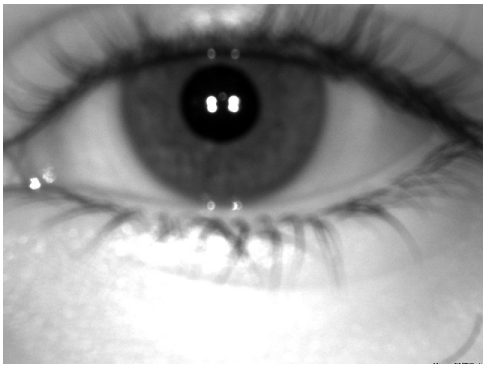


Figure 20: Misclassification for an iris with Transparent lens

In this example we can see a misclassified iris with colored lens in which the true prediction lies in the second place as well. In this case, the probability, even if the second highest one, is not too high due to the fact that the iris (as we can see in the Figure **21**) contains a lot of noise caused by the eyelashes, covering a good portion of the iris, and also the opening of the eye which isn't too wide.

```
Registered users:  10%|▉          | 105/1000 [01:59<06:23,  2.34it/s]
User id:  95
Sorted labels:  ['50', '95', '22', '3', '83', '18', '40', '51', '47', '12', '55', '81', '49', '38', '35', '27', '68', '57', '91', '11', '5', '30', '94',
Sorted probabilities:  [0.7327106595039368, 0.11671915650367737, 0.03532462567090988, 0.020505422726273537, 0.015508116222918034, 0.014226367697119713,
User Path:  ../data_iiitd/system_database/registered_users/95_Colored_20.bmp
```

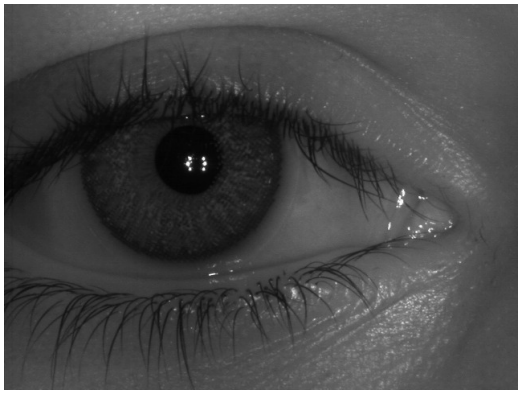


Figure 21: Misclassified image with Colored lens

6.1.2 Typed Correctness Rates - TCR

To understand how our model performs on every type of image, we evaluate their accuracy rates separately and plotted tham with histograms. We called it **Typed Correctness Rates**. As shown in the figure and in the data below, the model performs really good on transparent lenses and no lenses, getting near 0.9 for both for both, while it has much problems with colored lenses, getting a total rate of 0.45. This can explain why the final accuracy of the model is not that high, but still this gives a road to follow for an improval of the project.

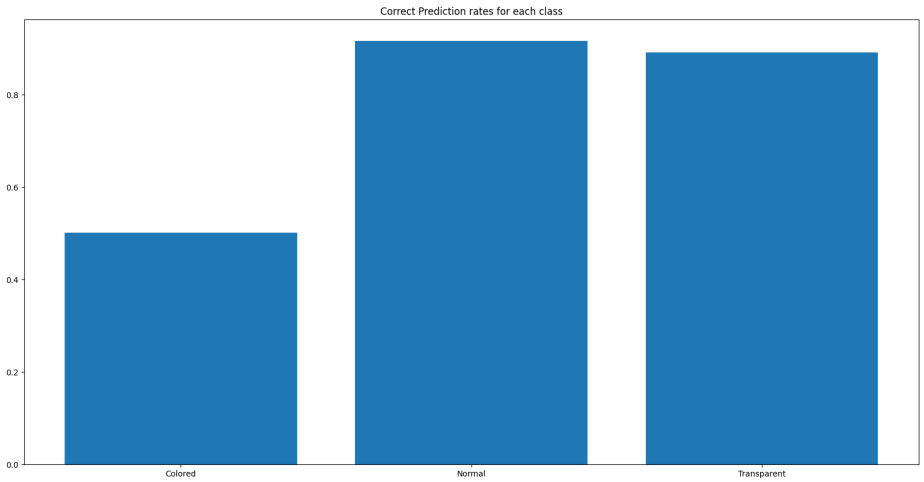
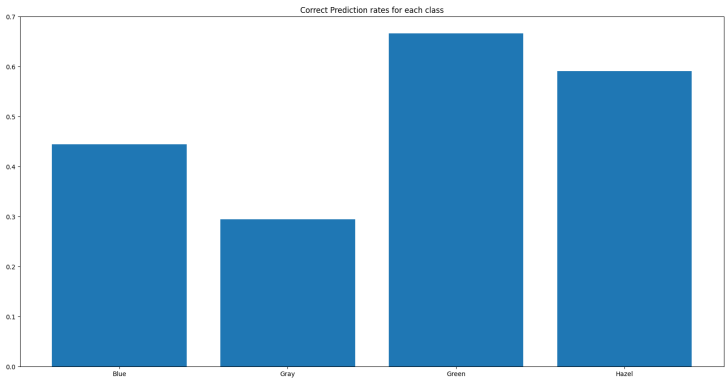


Figure 22: TCR histograms for Colored lenses, No lenses and Transparent lenses

	Correct Count	Total Count
Colored	146	291
Normal	265	289
Transparent	281	315

Figure 23: Data for TCR showing the strongly better performance for No lenses and Trasparent lenses

Our final assumption is that, as we could observe in the plots of ROC and TCR, the most mismatched images are the Colored lenses' ones, but that the classification still gives the right class a high probability most of the times compared to the ones given to the other 49 labels.
After this assumption, we thought that it was appropriate to deepen the "Colored" side of the dataset, plotting, for each image with colored lenses given as input to the model, the TCR of each color, to study which one gets mismatched the most.



	Correct count	Total count
Blue	28	63
Gray	23	78
Green	54	81
Hazel	42	71

As we can see, over a total of 293 "Colored" probes, the color which gets wrong classified most is gray. Instead, the one that gets classified the most is green. This gray underperformance could be a derivation of the dataset composition, given that all original images were captured in black and white.
It would have been useful to know the natural color of the iris of the users, to see if there was a huge difference caused by the colors of the lenses.

6.2 Identification closed-set

6.2.1 Cumulative Match Characteristic curve

We calculated, as evaluation metric for identification closed-set task, the CMC curve. The CMC curve works as follows:

- Each probe is compared against all gallery classes;
- The resulting scores are reverserly sorted and ranked;
- We determine the rank at which a True Match occurs;

Having 50 classes to compare our predictions with, our ranks are in the range of (0,50).
Already at rank 4 we can go over 0.8 and at rank 10 we superpass 0.9, which shows that, even if the probability for the right class is not always the highest one, it is in the highest ranking places most of the times.

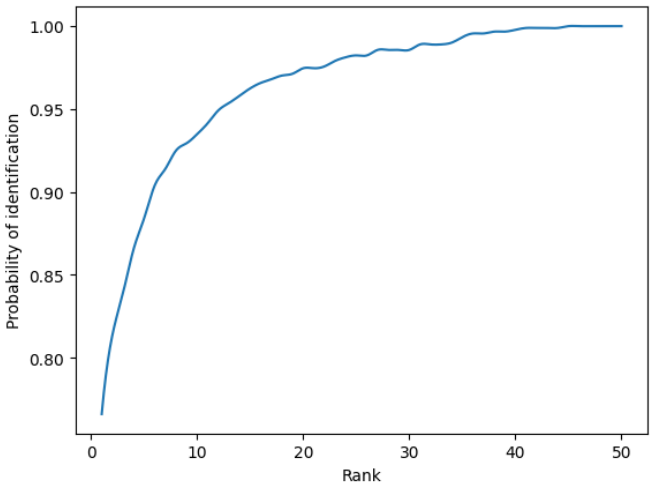


Figure 25: CMC curve at the various ranks

With these assumptions, we printed out the **True Positive Identification Rate (TPIR)** which is the probability of observing the correct identity within the top given K ranks. In particular, we computed it for ranks going from 1 to 30, with a step of 5 ranks each time.

```
True Positive Identification Rate at rank  1 :  0.7660044150110376
True Positive Identification Rate at rank  5 :  0.8841059602649006
True Positive Identification Rate at rank  10 :  0.934878587196468
True Positive Identification Rate at rank  15 :  0.9624724061810155
True Positive Identification Rate at rank  20 :  0.9746136865342163
True Positive Identification Rate at rank  25 :  0.9823399558498896
True Positive Identification Rate at rank  30 :  0.9856512141280354
```

Figure 26: TPIR at ranks from 0 to 30

7 Conclusions

The raising usage of contact lenses is facing new challenges for iris recognition and with this project we tried to setup a starting point to find a good solution for the problem. We came with this idea when thinking about events in which people cosplay their favourite characters (such as, the Romics or Lucca Comics or even bigger international events for example San Diego Comicon). In these kind of events people try to look as much as possible like these characters, even wearing contact lenses to change the color of their eyes. For security resons, in these places, there might be the necessity to verify the identity of people joining, and iris recognition could be a good solution for both the organizers and the users.
We started with the assumption that contact lenses could have a high impact on biometric systems' performances in identifying people. We discovered how impactful is the texture of the colored lenses on the recognition task, and also that transparent lenses don't give much problems to the classifier, as shown in our TCR results, in which the accuracy of the colored lenses is just the half of the other two.
We are still satisfied with our results, achieving nearly 70% of model accuracy, because we demonstrated that a more generalizable model is possible to be developed.
Future works may try to major the total accuracy of the model or focusing on the worse performance of it using more images of irises with colored lenses or adding a lens detector to the network that would be useful to discriminate easy-to-be-classified-wrong images.
In both cases it would be great for future security systems based on iris recognition.

References

[1] John Daugman. "How Iris Recognition Works". en. In: *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY* 14.1 (2004), p. 1.

[2] Naman Kohli Daksha Yadav Mayank Vatsa Richa Singh. "Revisiting Iris Recognition with Color Cosmetic Contact Lenses". en. In: (2013).

[3] Adityavardhan Chavali Dr. C. Nalini. "Observing the Effects of Colored Lenses on Iris Recognition using Feature Extraction Technique". en. In: *International Journal of Innovative Research in Science, Engineering and Technology* 4 (2015).

[4] Akshay Agarwal Mayank Vatsa. "Generalized Contact Lens Iris Presentation Attack Detection". en. In: *IEEE TRANSACTIONS ON BIOMETRICS, BEHAVIOR, AND IDENTITY SCIENCE,* 4.3 (2022).

[5] GlobeNewswire Press. "New Study Suggests Colored Lenses Can Reduce Headaches". In: *://www.einpresswire.com/article/633004997/new-study-suggests-colored-lenses-can-reduce-headaches* (2023).