Iris segmentation and recognition

Raneim Nabil[1,2,3], Cylia Ouadah[1,2,3], Francisco Aarón Tovar Sáez[1,2,3], Jana Vujadinovic[1,2,3]

[1] *Università degli studi di Cassino e del Lazio Meridionale, Italy*
[2] *Erasmus Mundus Joint Master Degree in Medical Imaging and Applications*
[3] *Universitat de Girona, Spain*

Iris recognition has seen rapid growth in interest during the last few decades as it is one of the most reliable and stable biometric identifiers for wide applications in security. However, accurate algorithms for segmentation and recognition are required to differentiate between individuals based on the unique features their irises possess. Many machine learning (ML), deep learning (DL), and image processing (IP) algorithms have been proposed to accurately segment and recognize iris. These algorithms achieved good results when the iris was segmented properly, and the algorithm was robust for all cases. In this paper, we propose two different approaches for iris segmentation and recognition. The first approach is an end-to-end DL pipeline using optimized U-Net segmentation model followed by an optimized encoder-FCN classification model. The second approach is based on image processing the iris in polar coordinates, extracting features using Gabor filters, and applying a number of machine learning classifiers. Experimental results from the IP pipeline showed that SVM gave the best performance according to accuracy, percision, AUC,and F1score.On the other hand, logistic regression was better with deep learning polar images and deep learning recognition path features.

Keywords: Segmentation; Recognition; Deep Learning; Machine Learning; U-Net; FCN; Gabor Filters; SVM; KNN; Logistic Regression; Naive Bayes; Random Forest

1. IMAGE PROCESSING AND MACHINE LEARNING PIPELINE

1.1. Segmentation

In this project, a three-step non-iterative method is proposed. First, the pupil is segmented using adaptive thresholding, morphological operations and connected component analysis. Then, the resulting image is transformed from Cartesian to polar coordinates to finally segment the outer boundary of the iris.

1.1.1. Pupil Segmentation

Static thresholding for pupil segmentation presents some challenges. First, the illumination conditions may not be uniform along the image. Secondly, the eyelids and eyelashes may degrade pupil segmentation performance. Finally, any noise present in the image may be included in the segmentation using a static thresholding method.

Adaptive thresholding

Some of the images, as shown in Fig. 1a, present a non-uniform illumination background. A Gaussian filter with a standard deviation of σ is used to estimate the background illumination and remove noise. The choice of σ should be much smaller than the pupil radius and bigger than the eyelashes to reduce its degradation effect on the final segmentation. For this project, $\sigma = 5$ was selected as the best value to obtained the filtered image. The difference image between the original image and the Gaussian filtered image is shown in Fig. 1c with a uniform background. This image provides a high-pass filter that preserves the border of both the pupil and the iris outer boundary contours. The difference image is thresholded (Fig. 1d) at 0. This value was found to be optimal to keep the connected pupil structure and isolated from the neighbour components.

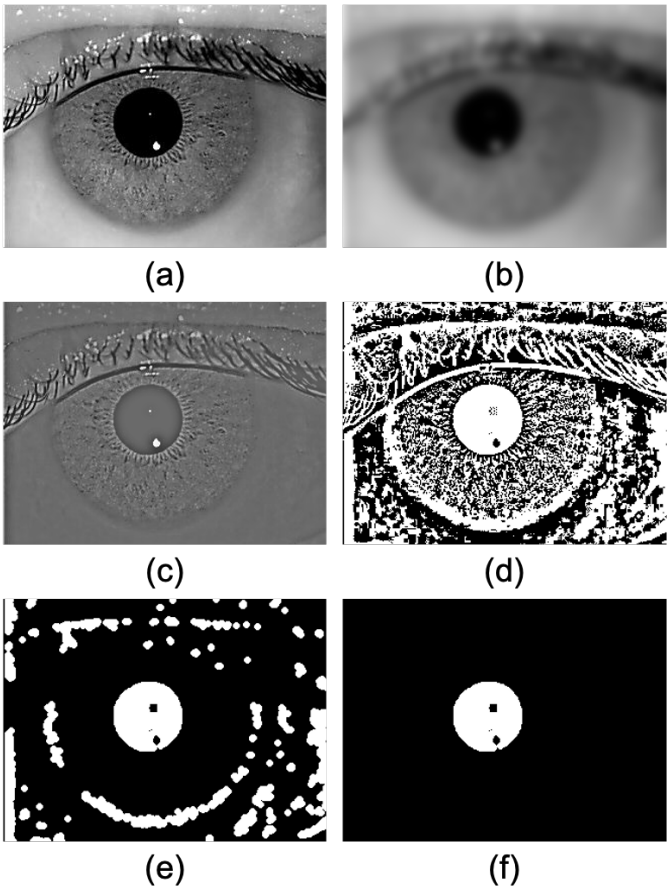


Figure 1: **a** Original image. **b** Smoothed image after applying a Gaussian filter to **a**. **c** Difference image between **a** and **b**. **d** Sign image of **c**. **e** Image **d** after applying morphological opening. **f** Resulting component of pupil after applying connected component analysis using area and eccentricity criteria.

Morphological Processing

After adaptive thresholding, the resulting image is a binary image with several objects. Each separated object is assigned a label and, for each label, every object is analyzed to perform connected component analysis. In particular for this project, an eccentricity criteria was established to get the connected component belonging to the pupil. The eccentricity of a component is determined as:

$$Ecc = \sqrt{\frac{Maj^2 - Min^2}{Maj^2}}$$

where Maj and Min are the major and minor axes of each component. For a perfect circular object, eccentricity is 0, while for elliptical objects it is between 0 and 1, and for almost circular objects it is almost 0. To avoid other very small and circular objects to be detected as pupil, before applying eccentricity criteria, an area criteria is applied to remove very small objects (smaller than 500 pixels of area). Then, the component associated with the pupil is selected as the one with the smallest eccentricity.

After pupil localization, another step to be done before transformation to polar coordinates, is pupil center location. Using a bounding box, the radius of the pupil is located using the X dimension of the bounding box as:

$$R_{pupil} = \frac{max(x) - min(x)}{2}$$

Then, X and Y coordinates of the center of the pupil can be found as:

$$C_x = \frac{max(x) + min(x)}{2}$$

$$C_y = max(y) + R_{pupil}$$

The radius is calculated using the X dimension information of the bounding box since the occlusions due to eyelid will affect the pupil vertically, not horizontally. The center is then firstly localized in the X dimension for the same reason. Finally, it is located in the Y dimension taking as reference the maximum y coordinate of the bounding box as a reference, as the occlusions due to eyelid affect the upper contour of the pupil, as shown in Fig. 2. The whole process of pupil center localization is shown in Fig. 2.

1.1.2. Cartesian to polar coordinates

Unlike many papers in the literature of iris segmentation, who rely on segmenting both pupil and outer boundary contour before transformation to polar coordinates, in this project the order is reversed, firstly transforming the image to polar coordinates before outer boundary segmentation.

For iris recognition, normalized (or polar) image has been used extensively. This conversion is done using a center and two concentric circles. However, when the segmentation of both pupil and outer boundary contours is performed, tow circles must be fitted to convert the image to polar coordinates. This process is introducing some error in the segmentation, as the new fitted circles do not correspond with the original contours. In this project, the outer boundary is segmented using the promise that it should be a circle sharing the same center as the pupil to avoid error addition due to the Cartesian to polar transformation.

Once the radius and center of the pupil have been determined, the image is transformed to polar coordinates using the determined center as the center of the transformation, no minimum radius, angular size of 360

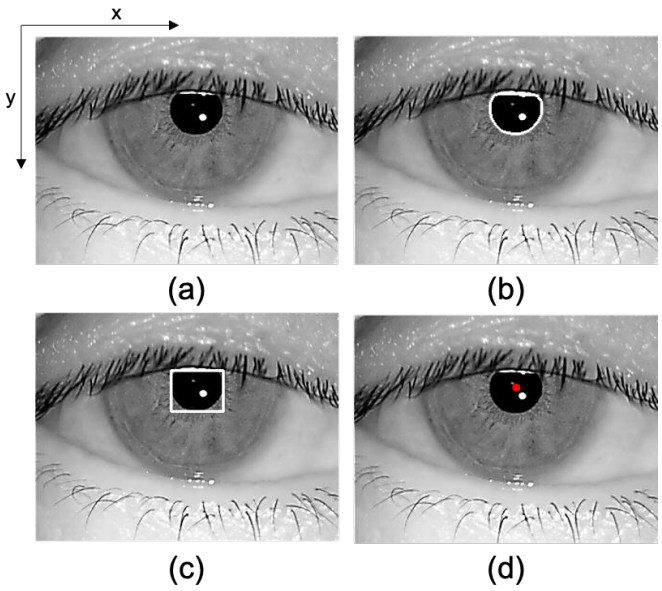


Figure 2: **a** Original image. **b** Original image with highlighted contour of the component corresponding to pupil. **c** Bounding box of contour **b**. **d** Localized pupil center.

degrees and maximum radius as 7 times the pupil radius. The outer radius is set as a high limit to ensure the outer boundary contour is within this range. As a drawback, this choice will introduce a considerable amount of pixels with 0 value that are not part of the original image as a consequence of the transformation.

The smoothed image is used for polar transformation to avoid noise interference and eyelashes segmentation degradation. The polar image is derived from the Cartesian image as:

$$I(r,\theta) = I(x(r,\theta), y(r,\theta)),$$

where $x(r,\theta) = rcos\theta, y(r,\theta) = rsin\theta, R_{pupil} < r < 7 \cdot R_{pupil}$ and $0° < \theta < 360°$.

1.1.3. Outer Boundary segmentation

After polar image transformation (Fig. 3c), a mask is created to localize the new "false" pixels created in the transformation (Fig. 3d). In the polar image, both pupil and outer boundary contour are present as horizontal edges. Therefore, a vertical Sobel filter with size $k = 5$ pixels is used to locate these horizontal edges. The resulting image is multiplied by the mask (to remove any values that do not belong to the original image) (Fig. 3e) and then summed along the X dimension to create a vector. The maximum of this vector will correspond to the most prominent horizontal edge in the polar image (the pupil) and the second maximum will correspond to the outer boundary contours. The second maximum is assumed not to be in a distance smaller than 0.6 times the radius of the pupil. There are some cases in which

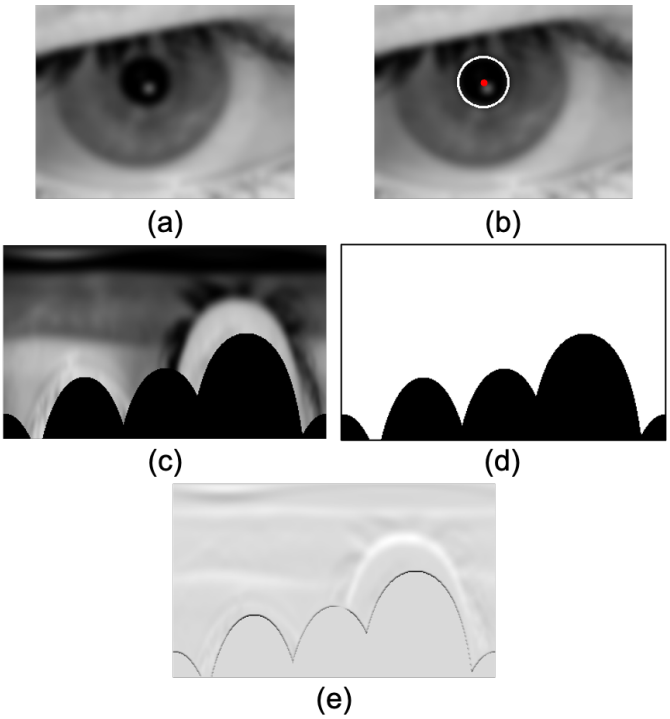


Figure 3: **a** Smoothed image. **b** Smoothed image with localized center and radius of the pupil. **c** Polar transformed image **a** using center and radius in **b**. **d** Mask of "true" pixels present in Cartesian image after polar transformation. **e** Result of vertical Sobel derivative applied in image **c** multiplied by mask **c**.

the outer boundary is within a row with some "false" pixels due to the polar transformation. In this cases, the second maximum will not correspond to the true outer boundary since this row will contain many 0s (from the "false" pixels) (Fig. 4a,b). To compensate for this effect, a second vector with the same shape as the first vector is created. This second vector contains the percentage or original pixels present in each row of the polar image, ranging from 0 to 1. The values lower than 0.85 are set to 1 to avoid division by 0 (Fig. 4c). Finally, the sum vector of horizontal contours is divided by this new vector to create a weighted average over the "true" pixels and detect the correct outer boundary (Fig. 4d,e).

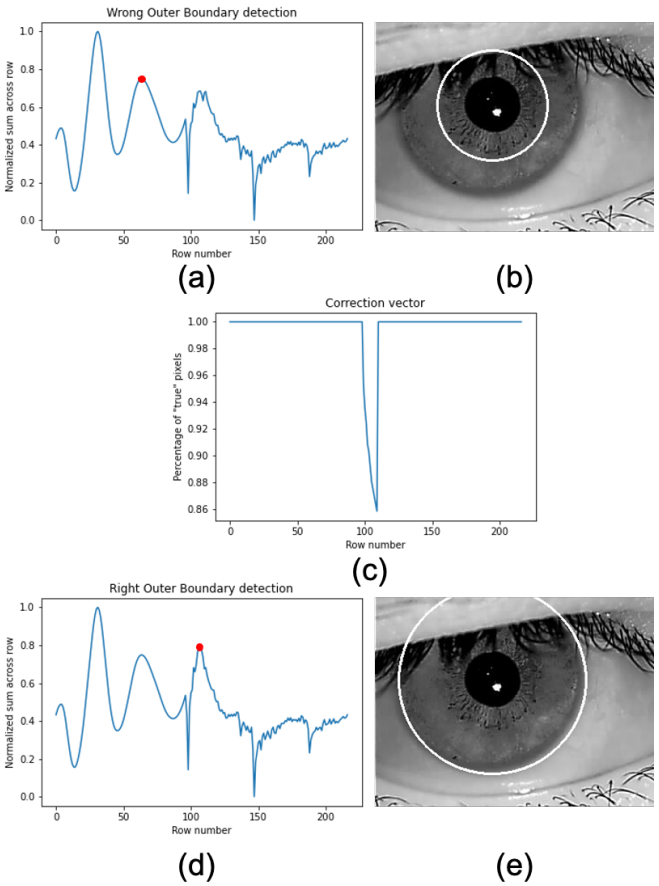


Figure 4: **a** Sum of horizontal contours along the X direction without correction. **b** The incorrect outer boundary detection using vector from **a**. **c** Vector used to correct for the 0 pixel values in the "false" regions of the polar image. **d** Sum of horizontal contours along the X direction with correction. **e** The correct outer boundary detection using vector from **d**.

1.1.4. Final mask creation

Once the pupil center, radius of the pupil and radius of the outer boundary are determined, two masks are created with circles centered at the pupil center with radius of pupil and outer boundary, respectively. The final mask is created as the intersection between the two masks. This process is shown in Fig. 5.

1.1.5. Evaluation of the Image Processing segmentations

The explained Image Processing Iris segmentation process is evaluated using the Dice Coefficient metrics against the ground truth masks:

$$DSC(X,Y) = \frac{|X \cap Y|}{|X| + |Y|}$$

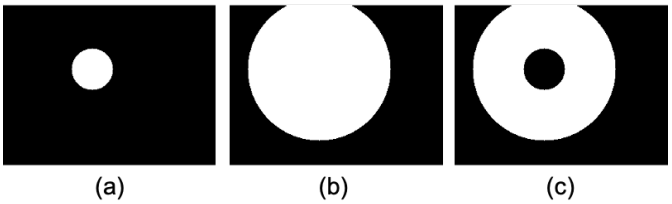


Figure 5: **a** Mask of the pupil. **b** Mask of the outer boundary contour. **c** Iris segmentation mask obtained from the intersection of **a** and **b**.

Fig. 6 shows the DSC metric for every image in the dataset. The mean DSC value in the dataset is 0.8645.

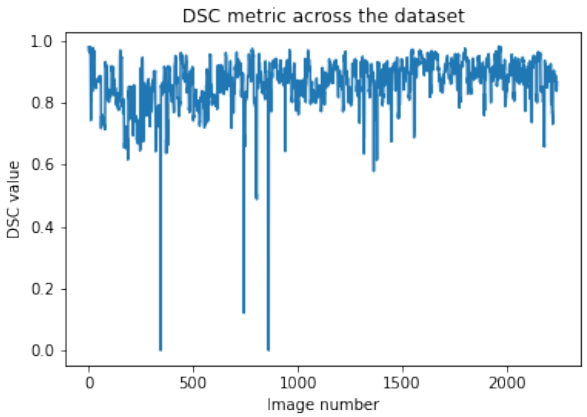


Figure 6: Dice Score Coefficient for every image on the dataset.

1.2. Polar transformation

Resulting masks using the Image Processing pipeline are transformed to polar coordinates prior recognition, as shown in literature extensively. This transformation is done according to the following formula:

$$I(r,\theta) = I(x(r,\theta), y(r,\theta)),$$

where $x(r,\theta) = rcos\theta, y(r,\theta) = rsin\theta, R_{pupil} < r < R_{outbound}$ and $0° < \theta < 360°$.

Fig. 7 shows the segmented pupil and outer boundary contours and the resulting polar image used for recognition.

Figure 7: **a** Iris pupil and outer boundary contours segmented. **b** Transformed polar image.

For Machine Learning and Deep Learning tasks, the data should have the same dimension. Since every polar image has an X dimension of 360 (angular size) and a varying y dimension (depending on the radius of the outer boundary contour and pupil), they need to be resized to a common shape. The final Y dimension was set as the mean of all the individual Y dimensions of every polar image.

1.3. Recognition

For Iris recognition features had to be extracted from the polar images obtained through the image processing pipeline as well as deep learning pipeline. Once obtained, images were convolved with specific Gabor filters and features concerning mean and standard deviation were considered. Furthermore the effectiveness of Gabor filters was monitored with respect to prediction accuracy, f1-score, ROC obtained using KNN and SVM.

Once the optimization of Gabor features has been achieved, additional machine learning algorithms were used to find the optimal recognition pipeline.

1.3.1. Gabor Filters

For the feature extraction with respect to texture analysis Gabor filters were used. For the choice of the specific Gabor filters used initial visual observation was made to find the optimal parameters required by the filter. The aforementioned parameters are following:

θ - affects orientation of the Gabor filter. When it equals to 0, the position of Gabor filter will be vertical.

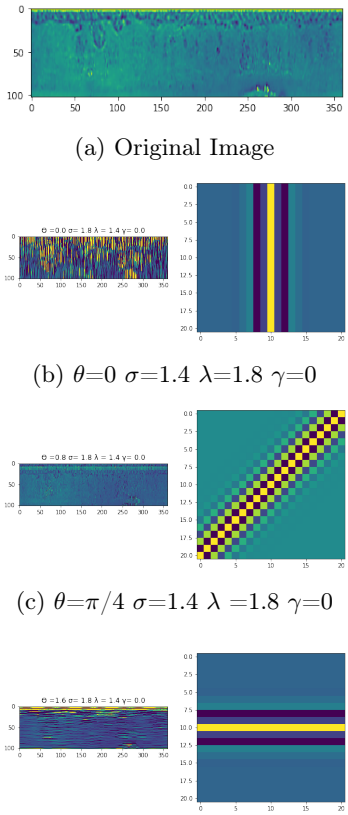
λ - affects wavelength of the sinusoidal component, hence affects the width of the strips of Gabor function

γ - controls height of Gabor function

σ - controls std. deviation of the Gaussian envelope. The higher the value, the higher the bandwidth and hence more strips present

ψ - affects the phase or offset of the sinusoid

However, only the first four parameters have been optimized with the offset set to 0 at all times. On the Figure 8 an example of effects of filter with changes with respect to the direction can be seen. What was observed overall is that filters visually don't seem to be very similar with respect to output when convolved with the image of interest. This is done so to initially set a base point which will be further used for optimization of filters. Furthermore, only the real part of Gabor filter has been used.



(a) Original Image

(b) θ=0 σ=1.4 λ=1.8 γ=0

(c) θ=π/4 σ=1.4 λ =1.8 γ=0

(d) θ=π/2 σ=1.4 λ=1.8 γ=0

Figure 8: Effects of filters with different parameters on a single sample of the image.

The initial configuration of Gabor filters contained λ = [1.4, 1.7, 2.0, 2.3], σ = [1.4, 1.8], γ = [0, 1] and θ = [0, π/4, π/2], totaling in 48 different filters. Furthermore, the size of the kernel initially applied was 25x25 and once the image was convolved with the filter the resulting image was divided initially in 36 sub-images and from each sub-image two information has been extracted, mean and std. deviation. With the resulting feature vector in the first case being 1120x3456 in case of training data set and 1118x3456 in case of testing data set. However, the size of the feature vector varied depending on the number of filters as well as the number of sub-images used for attaining the results.

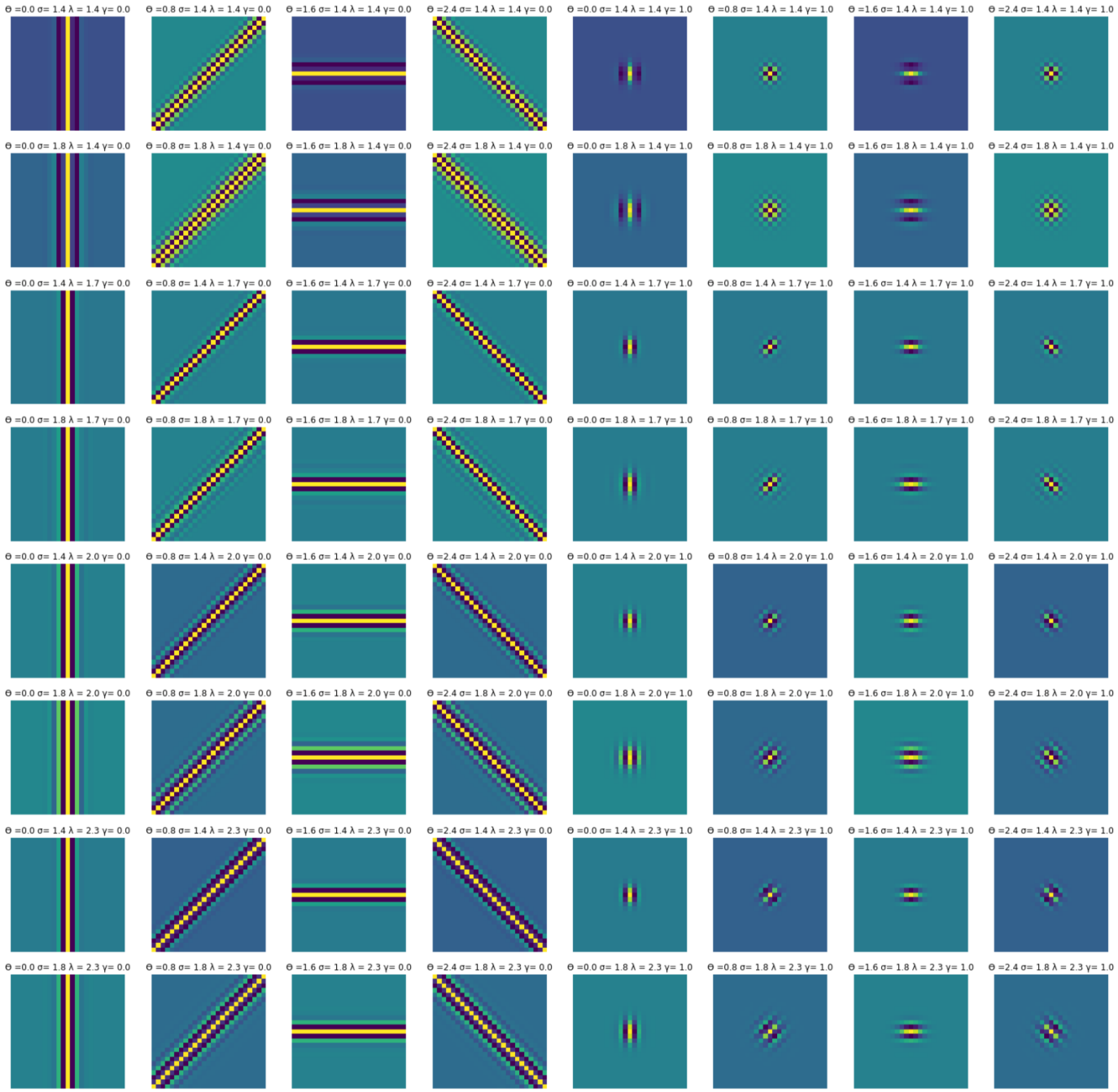


Figure 9: Set of 64 different filters that showcase the changes in parameters we considered with respect to attaining the Gabor filters. The following values were used $\lambda = [1.4, 1.7, 2.0, 2.3]$, $\sigma = [1.4, 1.8]$, $\gamma = [0, 1]$ and $\theta = [0, \pi/4, \pi/2, 3\pi/4]$ for finding the optimal parameters.

Iterative procedure of optimizing the the kernel size, number of sub-images to consider as well as parameters required for Gabor filters to give optimal results will be considered in the following text.

Gabor Filters - Kernel Size Estimation

When observing the effects of kernel size on the images, 5 differently sized kernels were applied 5x5, 10x10, 15x15,

,

Method	K size	PA	Precision	F1 score	AUC
KNN [n = 1]	5	88.19	90.11	87.87	94.07
KNN [n = 2]	5	78.53	82.68	77.2	89.22
SVM	5	89.71	90.59	89.31	94.83
KNN [n = 1]	10	91.5	93.12	91.21	95.73
KNN [n = 2]	10	82.38	85.66	81.1	91.15
SVM	10	92.4	93.81	92.23	96.18
KNN [n = 1]	15	91.5	93.09	91.19	95.73
KNN [n = 2]	15	82.65	86.4	81.53	91.29
SVM	15	91.68	92.72	91.33	95.82
KNN [n = 1]	20	91.41	92.52	90.86	95.69
KNN [n = 2]	20	83.09	86.03	81.83	91.51
SVM	20	92.22	93.17	91.71	96.09
KNN [n = 1]	25	91.68	93.97	91.42	95.82
KNN [n = 2]	25	86.63	86.46	82.33	91.78
SVM	25	92.49	93.41	92	96.23

Table 1. Effects of kernel size on the results of the classifiers. It can be seen from these results that the best results were achieved when setting kernel size to 25x25 based on PA as well as AUC of SVM in that case.

20x20, 25x25. The initial aforementioned configuration of filters was used in this case.

After convolving images with the Gabor filters and extracting the Gabor features which considered standard deviation and mean of sub-sections of an image, PCA which considers 99% variance was applied so to decrease the number of features.

From the table 1 it can be seen that the best results have been achieved using the kernel size of 25. Hence for the future classifications and optimization this kernel size has been used. In the following table PA stands for Prediction accuracy

Gabor Filters - Optimal Sub-image number

By setting the kernel size to 25x25 the filters were applied to images but std. deviation and mean were calculated on differently sized sub-images. Once filter was applied the images were split in 4, 8, 16, 24, 40 and for each sub-section mean and standard deviation were calculated.

However, due to original size of the image being 102x360 and 102 not being optimal to dividing the image due to division, the images were cropped by 2 pixels at the bottom uttermost part.

The following list showcases the size of each box used depending on the number of sub-images the image has been divided into

- 4 sub-images - 50x180
- 8 sub-images - 50x90
- 16 sub-images - 25x90
- 24 sub-images - 25x60
- 40 sub-images - 20x45

Method	SubImage	PCA %	PA	Precision	F1 Score	AUC
KNN	4	99 [27]	80.32	83.22	79.99	90.12
SVM	4	99 [27]	85.15	87.37	84.81	92.54
KNN	8	99 [55]	89	91.11	88.69	94.47
SVM	8	99 [55]	90.79	92.56	90.48	95.47
KNN	16	99 [110]	91.86	93.81	91.63	95.91
SVM	16	99 [110]	92.4	94.07	91.99	96.18
KNN	24	99 [159]	92.93	94.74	92.77	96.45
SVM	24	99 [159]	94.45	95.49	94.26	97.21
KNN	24	90 [29]	91.32	92.54	90.83	95.64
SVM	24	90 [29]	92.04	93.14	91.68	96
KNN	40	99 [258]	92.58	94.52	92.38	96.27
SVM	40	99 [258]	93.74	94.86	93.39	96.86
KNN	40	90 [45]	92.93	94.29	92.54	96.45
SVM	40	90 [45]	92.93	93.92	92.53	96.45

Table 2. Effects of number of sub-images on the results of the classifiers. From the table it can be observed that best results are achieved when dividing the image in 24 parts we can obtain best recognition results.

In table 2 effects of changing the number of sub-images we are considering can be seen. The best results were achieved when diving the image in 24 sub-images with each sub-image having dimension 25x60. There results were used for following analysis of effects of different gabor filter parameters on results.

Gabor Filters - Parameters

After optimizing the kernel size as well as the number of sub-images that are considered the next step was to optimize the Gabor filters by methodologically changing specific parameters and observing how they affect the Recognition result.

For each change in filters upon obtaining the feature vector, which contains information on standard deviation and mean of sub-images, PCA has been applied to reduce the number of features with respect to keeping the variance at 99 and 90%.

The first step was to increase the number of angles considered and observe the effects of $\theta = [0, \pi/4, \pi/2, 3\pi/4]$. In fact, all the filters observed on Figure 9 are the ones that have been used for this iteration of Gabor Filter parameter optimization. The values for other parameters were set constant. This concluded in creation of feature vectors which were of size 1120x3072 in case of training and 1118x3072 in case of testing.

The Table 3 showcases result in case of using the aforementioned parameters, whereas Table 4 and 5 showcase results in case of using the aforementioned parameters however setting $\gamma = 0$ and 1 respectively. Furthermore in both of the latter mention cases, when reducing γ to a single value the features vectors obtained when applying gabor filters were of size 1120x1536 and 1118x1536 for train and test.

When observing results which were obtained in Table 3, 4 and 5 it can be concluded that the best results have

Method	PCA	Prediction Accuracy	Precision	F1 Score	AUC
KNN	0.99 [166]	93.38	95.09	93.24	96.68
SVM	0.99 [166]	94.72	95.74	94.54	97.35
KNN	0.90 [28]	91.68	92.76	91.25	95.82
SVM	0.90 [28]	92.75	93.58	92.35	96.36

Table 3. Recognition results achieved using the 64 aforementioned filters and having γ = [0, 1].

Method	PCA	Prediction Accuracy	Precision	F1 Score	AUC
KNN	0.99 [147]	92.75	94.55	92.61	96.36
SVM	0.99 [147]	94.9	95.9	94.74	97.44
KNN	0.90 [27]	91.95	93.01	91.52	95.96
SVM	0.90 [27]	92.4	93.76	92.08	96.18

Table 4. Recognition results achieved using the 64 aforementioned filters and having γ = [0]

Method	PCA	Prediction Accuracy	Precision	F1 Score	AUC
KNN	0.99 [91]	92.75	93.89	92.37	96.36
SVM	0.99 [91]	93.92	94.69	93.53	96.95
KNN	0.90 [23]	89.27	90.58	88.52	94.61
SVM	0.90 [23]	89.53	91.01	89.01	94.74

Table 5. Recognition results achieved using the 64 aforementioned filters and having γ = [1]

Method	PCA	PA	Precision	F1 Score	AUC
KNN	0.99 [91]	92.75	93.89	92.37	96.36
SVM	0.99 [91]	93.92	94.69	93.53	96.95
KNN	0.90 [23]	89.27	90.58	88.52	94.61
SVM	0.90 [23]	89.53	91.01	89.01	94.74

Table 6. Recognition results achieved using θ = [0, π/2], = [1.4, 1.7, 2.0, 2.3], σ = [1.4, 1.8] and γ = [0,1]

Method	PCA	PA	Precision	F1 Score	AUC
KNN	0.99 [143]	93.65	95.31	93.46	96.81
SVM	0.99 [143]	94.28	95.45	93.99	97.12
KNN	0.90 [31]	92.13	93.09	91.6	96.05
SVM	0.90 [31]	92.84	93.57	92.45	96.41

Table 7. Recognition results achieved using θ = [0, π/2] and = [1.4, 1.7], σ = [1.4, 1.8] and γ = [0,1]

been achieved using the following parameters: λ = [1.4, 1.7, 2.0, 2.3] σ = [1.4, 1.8] γ = [0] θ = [0, π/4, π/2, 3π/4].

The following step considered using a smaller number of angles to see how that will affect the results.

Firstly we kept λ, σ and γ the same as initially but reduced the θ to be 0 and π/2.

Furthermore for these two values of the angles the values considered for λ were in one case [1.4, 1.7, 2.0 and 2.3] and in the second [1.4, 1.7] as seen in tables Tables 6 and 7 respectively. This choice of reducing the number of λ considered was based on the initial visual observation of the effects filters. The 2.0 and 2.3 values for λ has visually very good effect when observing θ which was not vertical or horizontal, hence the choice to see their effect on the result.

Method	PCA	PA	Precision	F1 Score	AUC
KNN	0.99 [52]	92.31	93.42	92.03	96.14
SVM	0.99 [52]	92.75	94.08	92.42	96.36
KNN	0.90 [12]	82.83	85.12	82.11	91.37
SVM	0.90 [12]	84.79	86.96	84.14	92.36

Table 8. Recognition results achieved using θ = [π/4, 3π/4], λ = [1.4, 1.7], σ = [2.0, 2.3] and γ = [0,1]

Method	PCA	PA	Precision	F1 Score	AUC
KNN	0.99 [82]	92.84	94.16	92.42	96.41
SVM	0.99 [82]	93.11	94.28	92.78	96.54
KNN	0.90 [18]	88.28	89.81	87.57	94.12
SVM	0.90 [18]	90.07	92.27	89.57	95.01

Table 9. Recognition results achieved using θ = [0] ,λ = [1.4, 1.7], σ = [2.0, 2.3] and γ = [0,1]

Method	PCA	PA	Precision	F1 Score	AUC
KNN	0.99 [82]	92.84	94.16	92.42	96.41
SVM	0.99 [82]	93.11	94.28	92.78	96.54
KNN	0.90 [18]	88.28	89.81	87.57	94.12
SVM	0.90 [18]	90.07	92.27	89.57	95.01

Table 10. Recognition results achieved using θ = [π/4], λ = [1.4, 1.7], σ = [2.0, 2.3] and γ = [0,1]

Method	PCA	PA	Precision	F1 Score	AUC
KNN	0.99 [44]	92.67	93.91	92.54	96.32
SVM	0.99 [44]	93.47	94.29	93.2	96.72
KNN	0.90 [12]	81.93	84.73	81.27	90.93
SVM	0.90 [12]	83.99	86.02	83.48	91.96

Table 11. Recognition results achieved using θ = [π/2], λ = [1.4, 1.7], σ = [2.0, 2.3] and γ = [0,1]

From the Table 6 and Table 7 it is visible that in fact when we reduce the number of 's we achieve better results which can be due to reduction of redundant data.

Secondly we kept λ, σ and γ the same as initially but changed the value of θ to be π/4 and 3π/4.

Furthermore for these two values of the angles the values considered for λ were in this case [2.0, 2.3] as it was also visually observed that they have most effects on the images.

The results achieved are not as good as the ones when observing solely the horizontal and vertical with consideration of only two values of λ.

The next step was to consider the angles separately. In this case the λ was equal to [1.4, 1.7, 2.0, 2.3], γ was 0 and 1 and σ was 1.4 and 1.8. Each one of the following cases contained overall 16 features with features matrices produce containing 1120x768 elements in case of training and 1118x768 elements in case of testing.

From the Tables 9, 10 and 11 it can be seen that in fact vertical axis carries most information which could be visually concluded as well and the horizontal one leads to least amount of information with respect to the results we have attained.

Based on all of the corresponding results in the end the final combination of filters to get feature vectors was set with following parameters:

- Kernel Size = 25x25
- λ = [1.4, 1.7, 2.0, 2.3]
- σ = [1.4, 1.8]
- γ = [0]
- θ = [0, $\pi/4$, $\pi/2$, $3\pi/4$]
- Number of sub-images = 24

1.3.2. Machine Learning

This part is dedicated to the choice of the best machine learning model for iris recognition using the resulting polar images from both the image processing segmentation path and the deep learning segmentation path. 50% of each of the two datasets were used for training the models (with respect to 5 inputs for each subject), and the remaining 50% were kept for models evaluation and decision making. Features were extracted based on the optimal Gabor kernels described in the previous section.
After this phase, Principal components analysis was used to reduce the features dimensionality, trials led to keep 99% of the features variability for both cases. The following machine learning techniques were applied on the train data and tested : K nearest neighbours, Naives Bayes classifier, Support vector machine, Logistic regression and random forestclassifier. The following two sections are dedicated to the results obtained for both polar datasets.

With Image processing polar images :

Features are extracted from the images according to the optimal kernels parameters (ie 32 filters * 24 sub images * 2 values (mean +variability) equal to 1536 features). PCA is applied with 99% to get 147 features for each sample.

- For SVM, Random forest and logistic regression hyper-parameters were optimized using a GridSearch.
- For KNN , K=1 gives the best results for the test (due to the high similarity between train and test data).

The following table summarizes the results obtained for the previously mentioned models with SVM giving the best results:

With Deep learning polar images :

The same things as the previous section were applied to the polar images (features extraction using optimal Gabor kernels + PCA with 99%).In addition, optimization of the hyperparameters was made the same way. The results are summarized in the following table and logistic regression gqve the best performance:

Method	Accuracy	Precision	F1score	AUC
KNN(k=1)	92.75	94.55	92.71	96.36
SVM	94.9	95.9	94.74	97.44
LG	94.63	95.6	94.46	97.30
Naive Bayes	66.10	89.38	70.69	82.97
R-forest	86.05	88.11	85.21	92.99

Table 12. Best results - polar images

Method	Accuracy	Precision	F1score	AUC
KNN(k=1)	88.75	90.68	88.44	94.35
SVM	90.18	91.81	89.80	95.07
LG	90.62	91.86	90.18	95.29
Naive Bayes	53.30	83.28	59.53	76.55
R-forest	80.00	84.33	79.35	89.96

Table 13. Best results - DL images

With the DL recognition path CNN as Features extractor:

As what preceded, in this section multiple machine learning algorithms were applied for iris recognition, but the features used were extracted from the Deep learning recognition path (flattened feature maps before the fully connected layers). Each image has a feature vector of length 2240, Principal components analysis was applied in order to reduce this number before feeding the machine learning algorithms. We tried different percentages for PCA, and the best results were for 90% of the variability or 159 features. As it's the case in the previous section, Grid Search was used to optimize hyper-parameters and different K were used for KNN algorithm at first, to conclude that the best model is for K=1.
Results are shown in the following table :

Method	Accuracy	Precision	F1score	AUC
KNN(k=1)	79.64	83.99	79.17	89.78
SVM	83.04	86.18	82.37	91.48
LG	83.12	85.29	82.19	91.52
Naive Bayes	64.91	82.74	67.44	82.38
R-forest	68.66	75.33	67.42	84.26

Table 14. Best results - CNN features

2. DEEP LEARNING

2.1. Segmentation

2.1.1. Model definition

In order to segment the Iris images using a Deep Learning approach, the extensively used U-Net model has been

used in this project. U-Net is formed by 2 main parts. An encoder, whose main function is to extract high-order abstract features from the data while reducing the spatial size of the images using double convolutional blocks; and a decoder, that progressively recovers the original matrix size of the input image by a series of concatenations and double convolutional blocks.

Every double convolutional block is formed by a convolutional layer using a 3x3 kernel, followed by a ReLu activation function, another 3x3 convolutional layer and ReLu activation function. In the encoder, every double convolutional block is followed by a Max Pooling layer, to reduce the matrix size by half. On the contrary, the number of filters in the next double convolutional block is doubled to compensate for the loss in spatial information and to encode this spatial information into high-level feature information. The last double convolutional block of the model is called the bottleneck layer, after which the decoder is defined. To recover the original matrix size, a transposed deconvolution with a stride of 2x2 is applied to the bottleneck to increase the matrix size by 2. At the same time, the number of filters in the upsampled layer, is reduced by half. After the transposed convolution layer, a concatenation layer is applied with the corresponding filters in the encoder. Finally, a double convolutional block is applied before the next transposed convolution until the original matrix size of the input is recovered. Finally, to the last layer, a sigmoid activation function is applied, since the purpose of this task is binary pixel segmentation.

2.1.2. Model optimization

There are many parameters in the U-Net model that should be chosen carefully and whose decision may impact drastically on the model performance. Among all the parameters, the ones optimized in the project will be the following. The number of double convolutional blocks in the encoder and decoder, the number of initial filters in the first double convolutional block in the encoder and the effect of adding dropout to the layers in the encoder. To choose the optimal value for each parameter, the rest were fixed to a reasonable value and one parameter was tested at a time. The optimal parameter is chosen as the one leading the the highest metrics performance of the model. Then, this parameter is fixed as the optimum one and the rest are tested until all of them are optimized.

For each training, the model was trained for 200 maximum epochs, using Adam optimizer, with a initial learning rate of 0.001, binary cross-entropy cost function, early stopping after 25 epochs (if validation error does not decrease) and decreasing learning rate by 0.1 after 10 epochs (if validation error does not decrease).

For every case, 5 images were set as training data and 5 images as test data randomly with a fixed random seed to allow reproducibility of the results. Moreover, 15% of the training data was set as validation data to allow early stopping and decreased learning rate callbacks while training. Input images were min-max normalized before training.

Table 15 shows the optimization process of the parameters. In italics it is shown the parameter being tested. In bold, the optimal parameter value leading to the highest Dice Score Coefficient on the test set. The optimal parameters of U-Net for the specific problem present in the project are 4 encoder/decoder double convolutional blocks, 4 number of initial filters in the first double convolutional block in the encoder and no dropout. These optimal parameters lead to a Dice Coefficient Score of 0.9735 in the test dataset. However, one of the main drawbacks of U-Net is that it performs pixelwise segmentation which, in some images may lead to small segmentations in unwanted regions of the image (Fig. 10b) because the local pixel values resemble the distribution of the object to be segmented. To solve this issue, connected component analysis is performed on the output U-Net segmentation to keep the component with the largest area. This post-processing step improves the DSC in the test dataset with a value of 0.9736.

Figure 10: **a** Original image. **b** U-Net segmentation output. **c** Output U-Net segmentation after connected component analysis.

The optimal U-Net model for this project is shown in Fig. 11. The training progress of the U-Net optimized model is shown in Fig. 12. The red dot in the plot represents the optimal model training using early stopping technique. After this point, the training error continues decreasing while the validation error starts to increase progressively due to memorization of the training data or overfitting.

Dropout is a technique to inactivate randomly some neurons present in the model to avoid overfitting of the model to the training and increase generalization capability of the model. However, in this specific project, the dataset is so homogeneous (low variability on the 10 images of a case, same angle and illumination for all the acquisitions, same eye location in every image...) that

Table 15. Optimization of the U-Net parameters.

Test number	Number of encoder blocks	Number of initial filters	Dropout	Test Dice Coefficient Score
1	*1*	8	0.0	0.2808
2	*2*	8	0.0	0.6872
3	*3*	8	0.0	0.9560
4	***4***	8	0.0	**0.9611**
4	4	*8*	0.0	0.9611
5	4	*16*	0.0	0.9605
6	4	*32*	0.0	0.7508
7	4	***4***	0.0	**0.9735**
8	4	*2*	0.0	0.9563
8	4	4	***0.0***	**0.9735**
9	4	4	*0.3*	0.5926
10	4	4	*0.5*	0.4558
11	4	4	*0.8*	0.3790

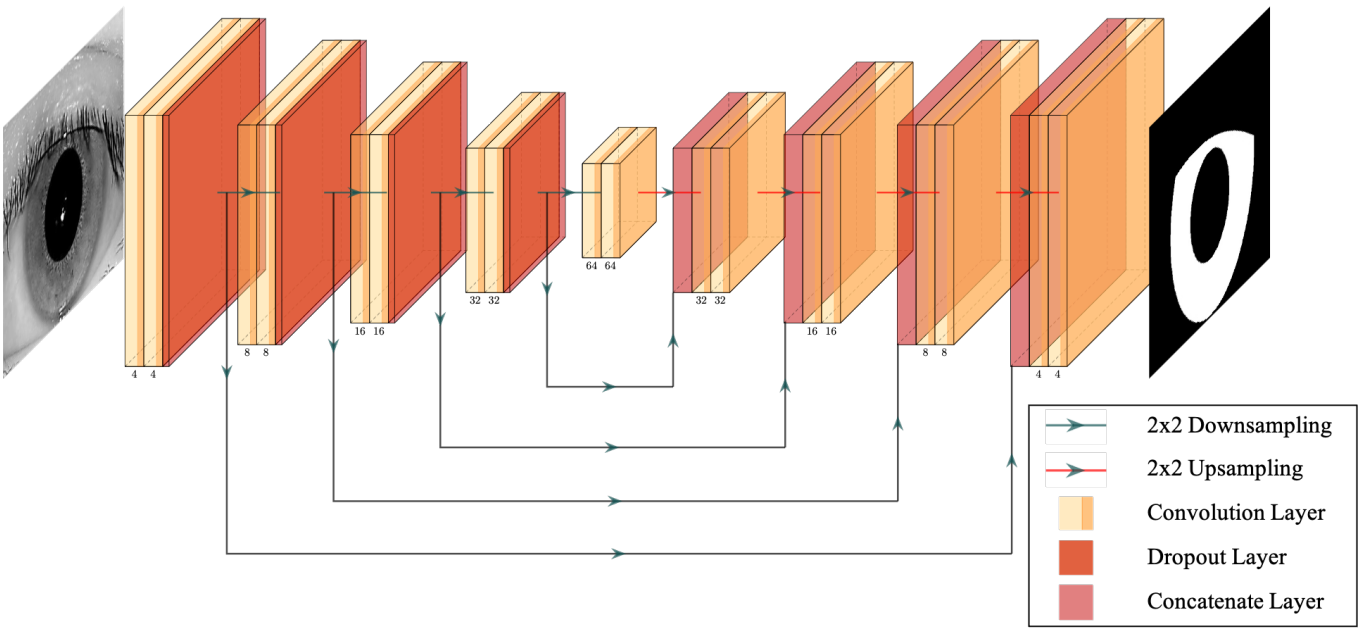


Figure 11: Optimized U-Net model

introducing any dropout to the training, highly degrades the model performance.

2.1.3. Hard attention

As suggested in (Lian et al., 2018), a hard attention mechanism can be added to the base model U-Net to improve significantly the segmentation of the Iris. This U-Net modified hard attention model can be shown in Fig. 13.

The modified model presents two main modifications. Firstly, in the bottleneck, the features from this layer are downsampled using a max pooling 2x2 layer, followed by a flatten layer and a Fully Connected layer with 4 output neurons. These 4 output numbers will be the coordinates of the bounding box of the segmentation. Finally, the coordinates of the bounding box are used to create a mask and multiply it by the last layer before the output segmentation. In this way, backpropagation outside the bounding box will prevent learning in neurons looking at unwanted parts in the image (eyelids, eyelashes...) and all the learning capabilities of the model will be used in the Region Of Interest determined by the bounding box.

The hard attention modified U-Net is trained in two steps. Firstly, the decoder layer is frozen during training and the only trained part in the model is the encoder

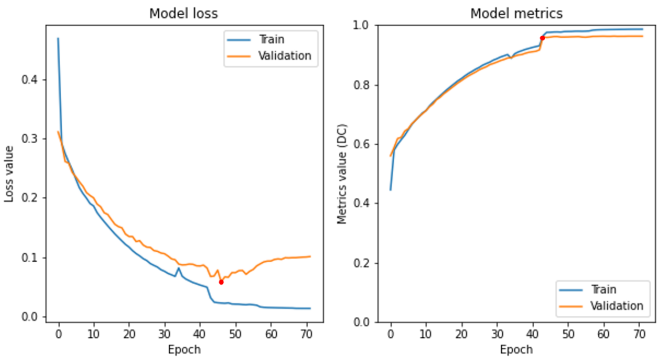


Figure 12: Training progress of the loss function value (left) and DSC metric (right) of training data (blue) and validation data (orange)

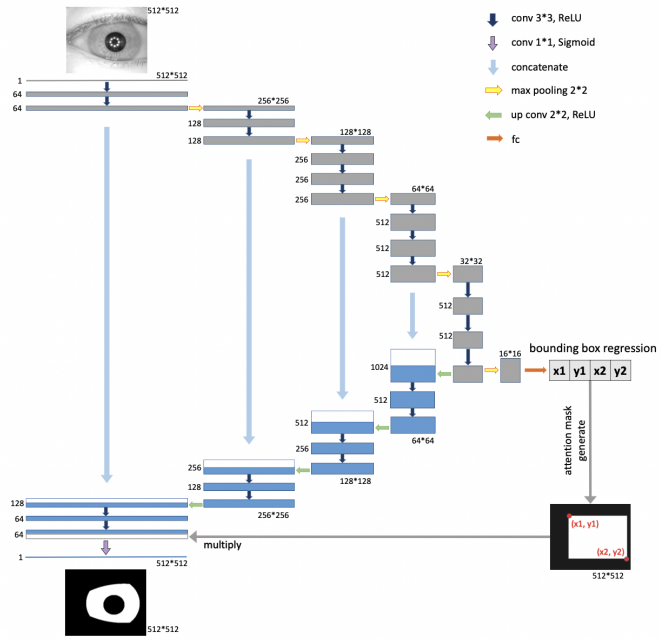


Figure 13: U-Net modified hard attention model. Source: (Lian et al., 2018)

and the FCN layer to predict the coordinates of the bounding box. Secondly, the encoder and FCN parts of the model are frozen, and the decoder is trained to actually perform the final segmentation. Both training steps are performed using the same training and data splitting previously explained in the optimized U-Net training. The only difference is that, in the first step of the training, the cost function is not binary cross-entropy but mean squared error.

The optimized parameters of U-Net found previously are kept in this hard-attention modified model. However, in each of the two step training processes, there is one parameter to optimize. In the first step training, the downsampled feature maps before flattening and FCN can be extracted from the bottleneck or any of the double convolutional blocks in the encoder. For this purpose, experiments are performed extracting this feature map from the bottleneck, the 4th, 3rd and 2nd encoder blocks. Results shown in Table 16 show that the optimal choice is extracting the feature map from the 4th encoder block. In the second step training process, the generated mask from the bounding box coordinates has pixel value of 1 inside the bounding box and δ outside. Therefore, the value of δ should be chosen as well. Experiments were performed using values of 0.0, 0.2, 0.4, 0.6, 0.8 and 1.0. The optimal value as shown in Table 17 is found to be 0.6, coinciding with the papers results in the CASIA dataset, which is very similar to the dataset used in the project. For δ values of 0.0 and 1.0, the model performs the worst due to no learning at all or no mask information, respectively.

Table 16. Optimization of the hard attention modified U-Net parameters in the first step training process.

Test number	Block	Test Error (MSE)
1	*Bottleneck*	125.3797
2	***Encoder (4th)***	**84.7164**
3	*Encoder (3rd)*	172.2547
4	*Encoder (2nd)*	5834.2827

Table 17. Optimization of the hard attention modified U-Net parameters in the second step training process.

Test number	Delta	Test DSC
1	*0.0*	0.9074
2	*0.2*	0.9447
3	*0.4*	0.9449
4	***0.6***	**0.9498**
5	*0.8*	0.9480
6	*1.0*	0.9449

After hard attention modified U-Net optimization and training, the Dice Coefficient Score in the test dataset (0.9498) is still lower than baseline U-Net optimized (0.9672). The reason is the same as why dropout does not introduce any improvement on the model performance. hard attention mechanism is a generalization improvement technique for the model that can segment extremely well irises with varying locations and sizes in the image. The dataset used in the project is completely homogeneous, so any

generalization improvement technique, will degrade the model performance instead of improving it.

2.2. Polar image transformation

For comparison with Image Processing masks in the Image Processing recognition part, Deep Learning segmentations were transformed to polar images. Contrary to IP masks, DL masks were not segmented with the premise that both the pupil and outer boundary contours should be concentric. Instead, they are highly variable and they adapt notoriously to the ground truth masks. Therefore, fitting two circles in the DL masks is not so evident and may introduce some error.

Starting from the DL mask, Hough circular transform is used to locate the pupil contour and pupil center thus, determining the radius of the pupil. To calculate the radius of the outer boundary, a horizontal intensity profile is analyzed along the center of the pupil. ΔR is calculated as the mean of the length of both segments were the horizontal intensity profile passes through the mask. Then, the outer boundary contour is calculated as:

$$R_{outbound} = R_{pupil} + \Delta R$$

Fig. 14 illustrates the complete process. Once the pupil center, pupil radius and outer boundary radius have been determined, polar image is obtained in the same way as Image Processing segmentations polar transformations.

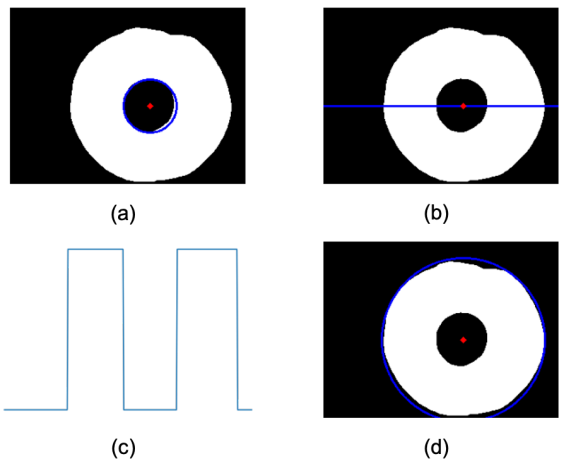


Figure 14: **a** Detected pupil circle and center using Hough Circular Transform. **b** Horizontal line across the center detected in **a**. **c** Intensity profile along line in **b**. **d** Final detected outer boundary

2.3. Recognition

For recognition task in the Deep Learning pipeline, the outputs from optimized U-Net with connected components post-processing is used.

2.3.1. Model definition

In this project, a encoder followed by a Fully Connected Network is used for classification similar to VGG16. The encoder is used to extract high-order abstract features from the data while reducing the spatial size of the images using double convolutional blocks. The FCN takes the flattened features passed them through a series of hidden fully connected layers and gives the classification using a final softmax layer with as many neurons as classes in the dataset (224 in this case).

Every double convolutional block is formed by a convolutional layer using a 3x3 kernel, followed by a ReLu activation function, another 3x3 convolutional layer and ReLu activation function. In the encoder, every double convolutional block is followed by a Max Pooling layer, to reduce the matrix size by half. On the contrary, the number of filters in the next double convolutional block is doubled to compensate for the loss in spatial information and to encode this spatial information into high-level feature information.

There are several parameters in the encoder and FCN to be optimized. Before optimizing these parameters, there is an extra parameter that should be optimized before these ones. DL recognition model takes as input the DL segmented images. These segmented images are obtained multiplying the original image by the DL mask. The DL mask is 1.0 in the mask and δ outside the mask. δ values can range between 0.0 and 1.0. A value of 0.0 (as commented previously in the hard attention section) implies no learning in many regions of the image during backpropagation, while a value of 1.0 implies that the input image for recognition is the original image and there is no use of the DL segmentation output. To choose the optimum value for δ, several experiments are performed keeping the parameters in the encoder and FCN fixed. These parameters are: 3 double convolutional blocks in the encoder, 8 initial filters in the first double convolutional block in the encoder and 1 hidden layer in the FCN with 512 neurons.

For each experiment, the model is trained using Adam optimizer with learning rate of 0.001, categorical cross-entropy as cost function, with no validation data, no callbacks and 30 epochs. The reason why no validation data is used to select the best model parameters is the low number of images available per case for training data (only 5). In segmentation, the data is quite homogeneous in terms of iris localization, illumination conditions, iris

size, etc. Each image present in the training data is seen as an independent image. This is, no matter the labels for image segmentation and every single image helps the training model to learn the map between input image and segmented image. However, in classification every single image is extremely necessary as there are only 5 images per case. This is the reason why no validation data is used. Moreover, the 5 images per case present in the training dataset are extremely homogeneous, which makes the model to overfit completely after 15 epochs with 100% accuracy in training data. This makes the training process to stop completely after 15 epochs, so the weights at this moment will be the final ones to evaluate the model performance on the test dataset. For this specific task and project, the model selection is crucial, rather than the learning process. When the model overfits the training data, the performance on the test dataset explains how well the selected model parameters can explain the underlying data distribution. In Fig 15, three different training processes with different model parameters have been plotted. For visualization purposes, the validation data was set as the test data. In all three cases in can be observed that, when the model overfits completely the training data, the model no longer improves and the performance on the validation dataset is kept constant. Moreover, for different parameter selection, the final validation accuracy is different once the model overfits.

Table 18 shows the results of these experiments. The optimal value of δ is 0.6. Fig. 16 shows the original image and the segmented image using the optimal value of delta. This image will be the input of the DL recognition model.

Table 18. Optimization of δ value for input image in the Deep Learning recognition model.

Test number	Delta	Test accuracy
1	*0.0*	0.6196
2	*0.2*	0.7518
3	*0.4*	0.7598
4	***0.6***	**0.8027**
5	*0.8*	0.7839
6	*1.0*	0.7321

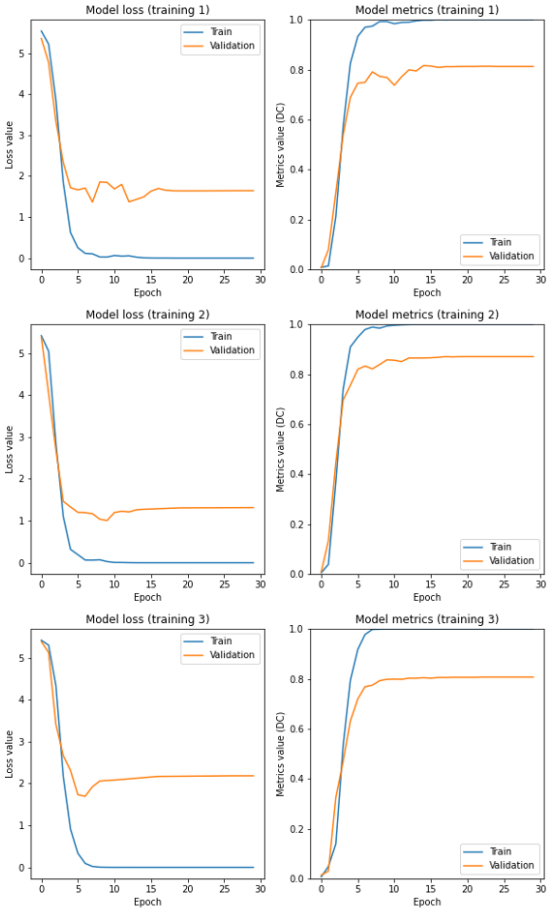


Figure 15: Different training processes (left: cost function value, right: accuracy) using different model parameters in the training and validation datasets

Figure 16: **a** Original input image. **b** Original image multiplied by the Deep Learning obtained mask using $\delta = 0.6$

2.3.2. Model optimization

The Deep Learning model for recognition of this project has many parameters to be optimized for best model performance. Regarding the encoder part of the model, the parameters to optimize are the number of double convolutional blocks and the number of initial filters in the first double convolutional block. For the FCN part, the parameters to optimize are the number of hidden layers and the number of neurons per layer. To choose

Table 19. Optimization of the Deep Learning recognition model parameters.

Test number	# encoder blocks	# initial filters	# hidden FCN layers	# neurons	Test Accuracy
1	*2*	8	1	[512]	0.7277
2	*3*	8	1	[512]	0.6196
3	*4*	8	1	[512]	0.8393
4	***5***	8	1	[512]	**0.8625**
5	*6*	8	1	[512]	0.8455
4	5	*8*	1	[512]	0.8625
6	*5*	***2***	1	[512]	**0.8804**
7	5	*4*	1	[512]	0.8625
8	5	*16*	1	[512]	0.0045
6	5	2	*1*	*[512]*	0.8804
7	5	2	*1*	*[224]*	0.8643
8	5	2	*1*	*[50]*	0.7696
9	5	2	*1*	*[1024]*	0.8732
10	5	2	***2***	***[1024, 512]***	**0.8955**
11	5	2	*2*	*[512, 1024]*	0.8536
12	5	2	*2*	*[512, 512]*	0.6973
13	5	2	*2*	*[1024, 224]*	0.8321
14	5	2	*2*	*[512, 224]*	0.8491
15	5	2	*2*	*[224, 1024]*	0.8616
16	5	2	*2*	*[224, 512]*	0.8696
17	5	2	*2*	*[1024, 1024]*	0.8366
18	5	2	*3*	*[1024, 512, 224]*	0.8536
19	5	2	*3*	*[224, 512, 1024]*	0.7500
20	5	2	*3*	*[1024, 512, 1024]*	0.8473
21	5	2	*3*	*[1024, 224, 1024]*	0.8336
22	5	2	*3*	*[512, 1024, 512]*	0.8054
23	5	2	*3*	*[224, 512, 224]*	0.8080

the optimal value for each parameter, the rest were fixed to a reasonable value and one parameter was tested at a time. The optimal parameter is chosen as the one leading the the highest metrics performance of the model. Then, this parameter is fixed as the optimum one and the rest are tested until all of them are optimized.

Table 19 shows the optimization process of the parameters. In italics it is shown the parameter being tested. In bold, the optimal parameter value leading to the highest Dice Score Coefficient on the test set. The optimal parameters of the Deep Learning recognition model for the specific problem present in the project are 5 encoder double convolutional blocks, 2 as number of initial filters in the first double convolutional block in the encoder. The optimal parameters for the Fully Connected Network part are 2 hidden layers with 1024 and 512 neurons, respectively. These optimal parameters lead to an accuracy Score of 0.8955 in the test dataset.

The optimal DL recognition model for this project is shown in Fig. 17.

2.3.3. Deep Learning Recognition model performance

Apart from accuracy, the DL recognition model performance is evaluated using different performance metrics.

2.3.4. ROC and AUC

ROC curve is a plot that shows the diagnostic ability of a binary classifier system as its discrimination threshold is varied. AUC provides a measure of performance across all possible classification thresholds. The AUC for a perfect classifier is 1.0, while for a random classifier is 0.5. For multi-class classification problems, for every class a binary classification task is used to perform the ROC curve (Fig. 18a). Then, for all the classes, the average is performed without taking into consideration class imbalance (micro-average) or performing a weighted average taking into account class imbalance (macro-average). These averages are used to compute the AUC for the multi-class classification problem (Fig. 18b) . In the project, since dataset is perfectly balanced, micro and

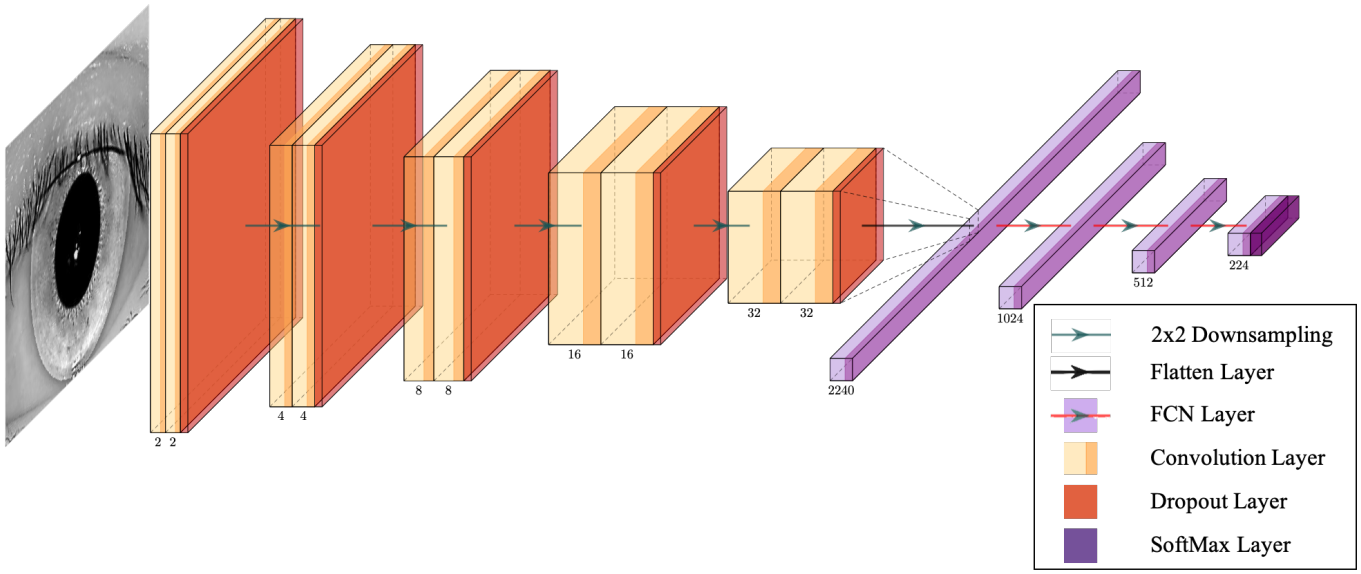


Figure 17: Optimized Deep Learning recognition model

macro averages lead to the same result and same AUC (0.99).

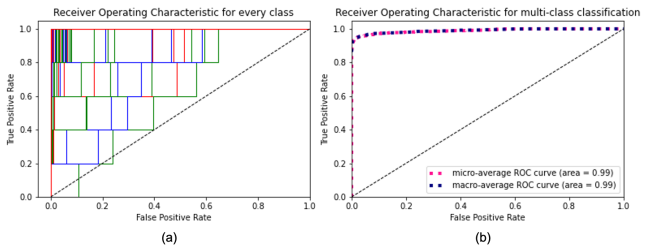


Figure 18: **a** ROC curves for every class. **b** ROC curve (micro and macro averages) for multi-class classification

2.3.5. Equal Error Rate (EER)

EER is a biometric performance measurement used to predetermine the threshold values for its false acceptance rate and its false rejection rate. When the rates are equal, the common value is referred to as the equal error rate this value is known as the EER. The value indicates that the proportion of false acceptances is equal to the proportion of false rejections. The EER for the DL recognition model is 4.255%.

2.3.6. Decidability Index (DI)

The DI (d) measures how well two distributions are separated. Using the mean and standard deviations of both distributions $(\mu_1, \mu_2, \sigma_1, \sigma_2)$, DI can be calculated as:

$$d = \frac{|\mu_1 - \mu_2|}{\sqrt{(\sigma_1^2 + \sigma_2^2)/2}}$$

The DI for the DL recognition model is 1.6975.

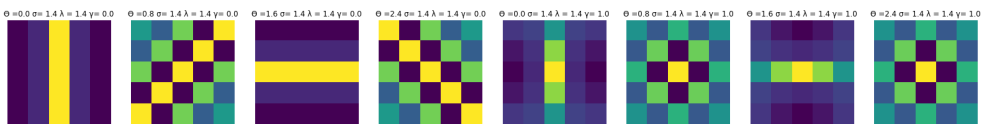
2.3.7. Performance metrics comparison

The usual DI index in iris recognition literature are typically higher than 10. In our case, the data is so homogeneous that the DI of the model is much smaller. The homogeneity of the data and the overfitting capability of the model can also be observed in terms of ROC and AUC (Fig. 18). Most of the cases, are perfectly classified (AUC = 1.0) with very few classes whose individual AUC are less than 1.0. Overall, this contributes to have an almost perfect classifier (AUC=0.99) and an EER considerably low (4.255%).

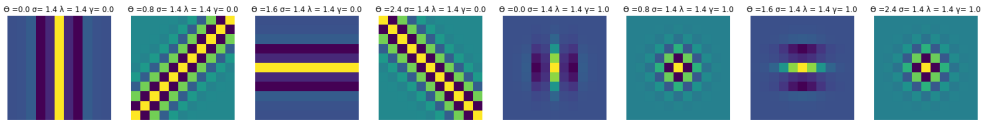
REFERENCES

Lian, Sheng, et al. "Attention guided U-Net for accurate iris segmentation." Journal of Visual Communication and Image Representation 56 (2018): 296-304.

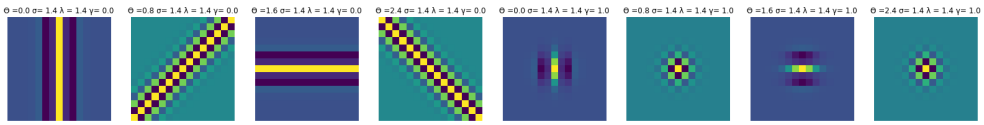
APPENDIX



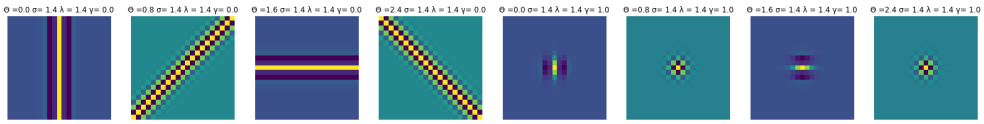
(a) 5x5



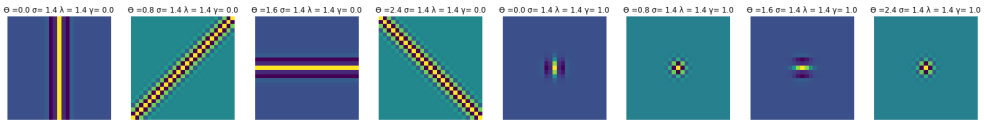
(b) 10x10



(c) 15x15



(d) 20x20



(e) 25x25

Figure A1: Sample of Gabor filter Kernels of different sizes