**Deep Learning**

**Framework**

**Hardware**

For the computation, a personal laptop has been used. It has an Intel i7 CPU with 6 cores and 16GB RAM and an NVIDIA RTX 2070 Max-Q GPU with 8GB of VRAM.

**Software**

The framework in which this part of the project was developed was TensorFlow GPU. This library takes advantage of the CUDA framework for NVIDIA GPUs. Getting CUDA to work was a challenge but the computational advantage far outweighed the effort.

**Data management**

For this part, generators have been used because of memory usage considerations. By using generators, only the current batch of training examples is loaded into RAM.

**Other considerations**

All architectures presented in this part were entirely developed by our team and are the final result of hundreds of iterations.

**Segmentation**

**U-Net**

For the segmentation task, a U-Net-like architecture has been used. The architecture was developed by our team, not imported. It consists of a contractive part (using *Max Pooling*) and an expansive path (using *Transpose Convolution*). The intuition behind this architecture is that the model can learn how to generate segmentations of the iris based on the information that has been synthetized in the contractive part and guided by the targets being presented at the end of the expansive path. U-Net is also a great architecture for this task because it manages to stay true to the geometry of the object-to-be-segmented by introducing information about it at each reconstruction stage.

The model has been trained for 200 epochs with an early stopping callback. The optimization algorithm was Adam and the loss function used is Binary Crossentropy.

The F1 score achieved by the model was: 96.76%.

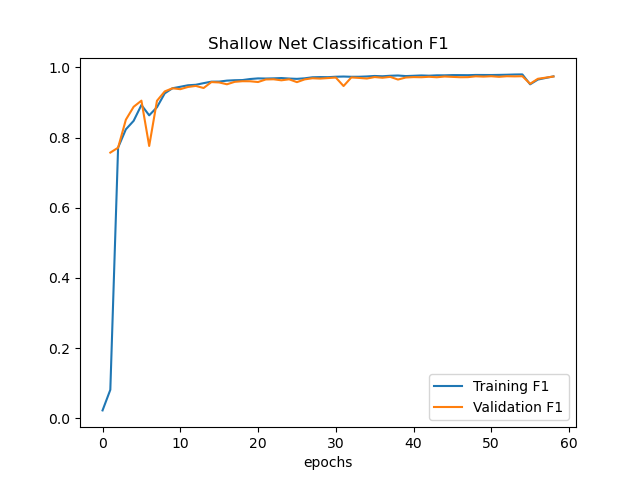


Figure 1. U-Net Segmentation F1 Score

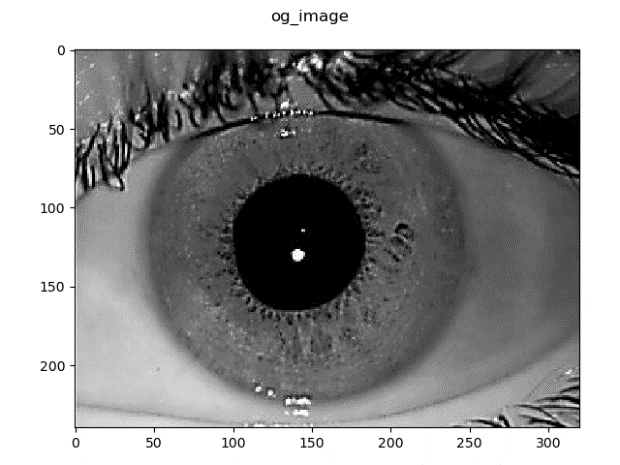
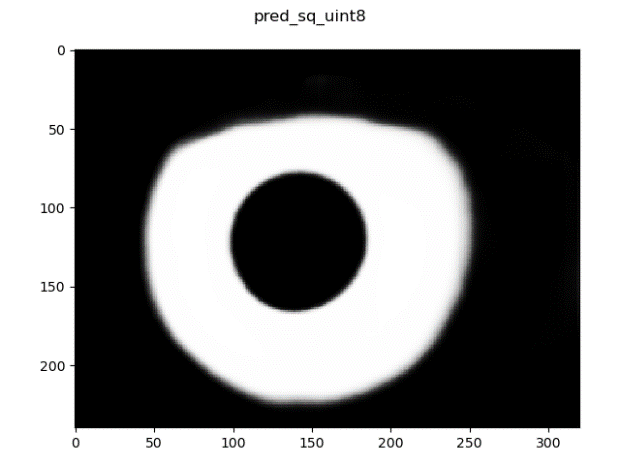
****

Figure 2. (left) Original Eye Image; (right) Segmentation

**Deep Feature Extraction**

In order to perform the classification necessary for the recognition task, it was necessary to find a representation for the information present in the iris.

**Polar Transformation**

A first stage was to obtain a rectified version of the iris. To do so, the original image was multiplied with the segmentation mask to cut everything but the iris from the image, then a polar transform was applied to the iris image so that the image lost its circular structure. The intuition behind the latter step was that inference power can be wasted by the classification network learning that the iris is round, the aim is to concentrate the analytical power of the network on the features that differentiate between different patient’s irises.

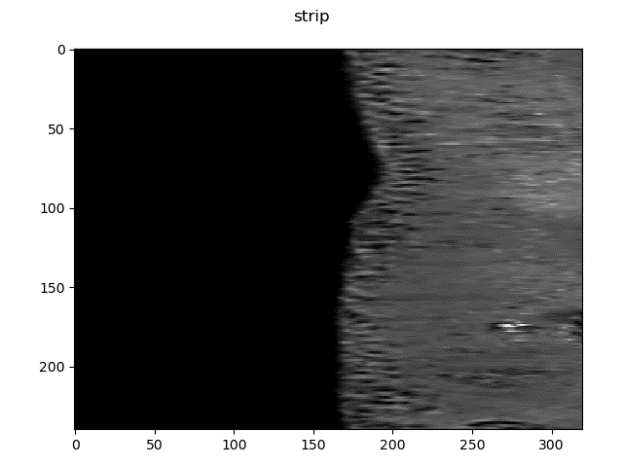


Figure 3. Rectified image of the segmented iris

**Autoencoder**

For the main feature extraction step, an autoencoder network was used. The intuition behind this choice is that the encoder part of the network can be used to extract the essential aspects of the rectified iris image in the shape of a vector that can later be used as a feature vector for the classification algorithms.

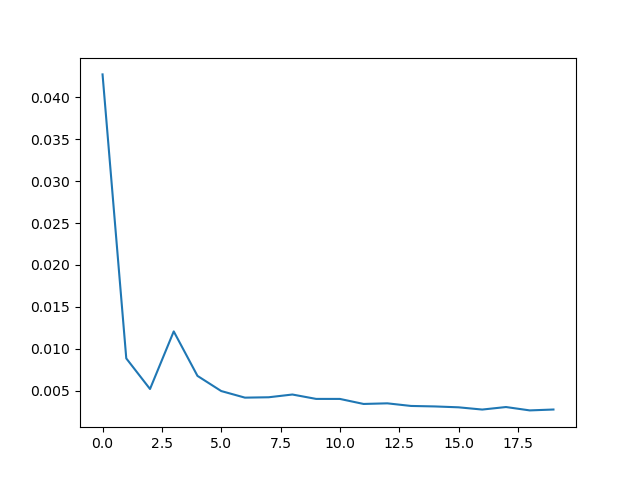


Figure 5. Autoencoder Loss vs Epochs

As a specific measure for this task, in the beginning, the reconstructions was evaluated but later the autoencoder was optimized according to the classification performance. An interesting approach would be to use an encoder-classifier mixed architecture but unfortunately, due to time constraints, this hypothesis could not be tested.

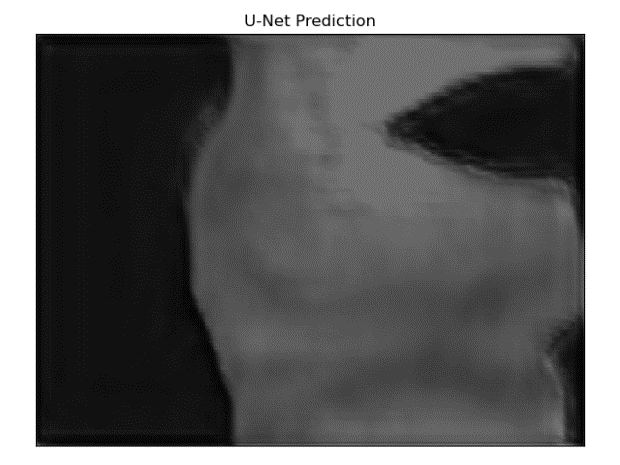
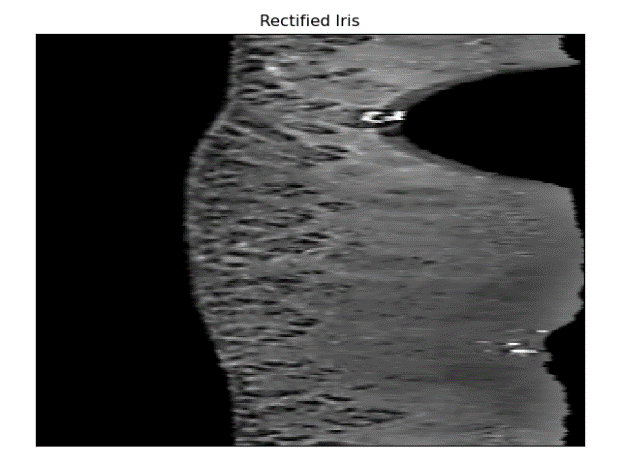


Figure 6. (left) Rectified Iris; (right) Autoencoder Reconstruction

**Classification**

For the classification task, two scenarios were considered. In the first one, deep features have been used, those are the output of the encoder when rectified iris images were inputted. This represents the standard way to do recognition. However, it has also been of interest how would classification work directly on the rectified iris images. In this section, both cases will be presented.

**Using Deep Features**

After extracting the features using the encoder, the feature vector is used as an input to a neural network developed for classification. Since only machine learning classification methods were used for the image processing pipeline, it has been considered appropriate to use deep learning methods for this pipeline for two reasons: to explore recognition with deep learning and to keep this a pure deep learning approach.

The Area Under the Curve for this approach was 0. 9935, the Equal Error Rate was 0.0221 and the Decidability Index was 0.2098.

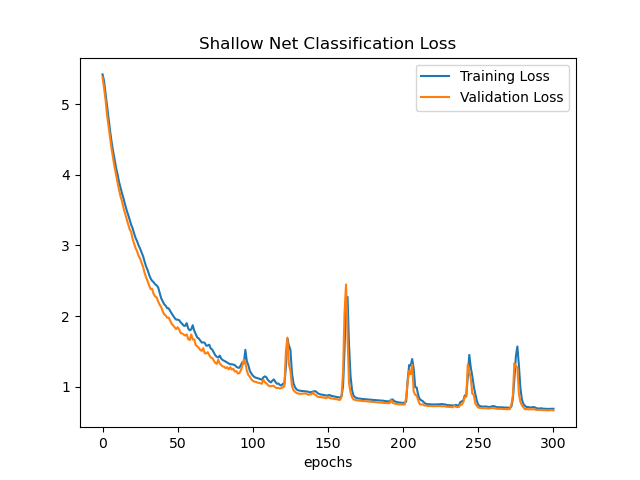


Figure 7. Classification Loss

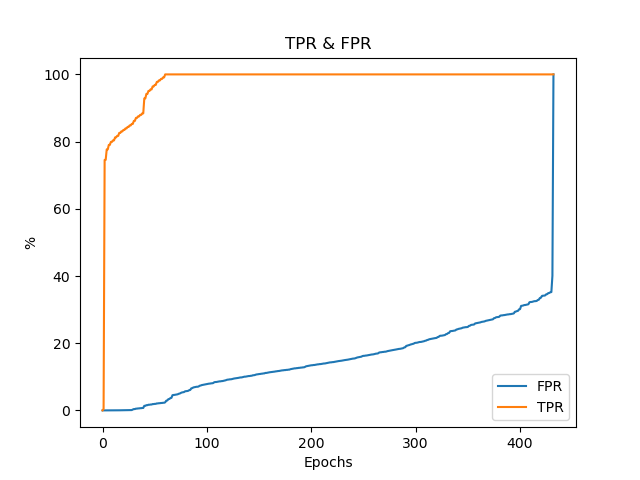


Figure 8. True Positive rate and False Positive Rate

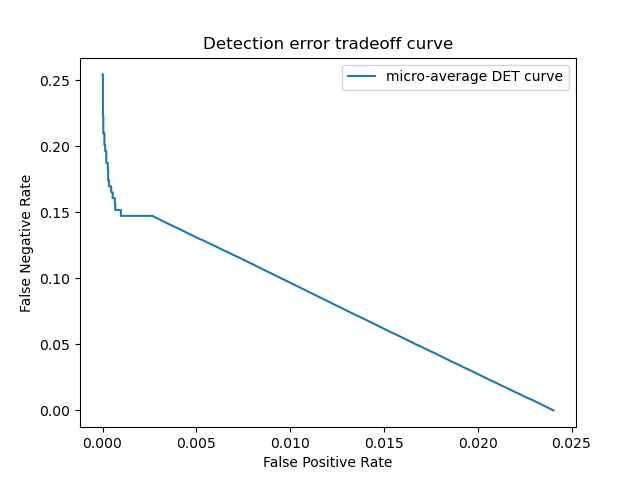


Figure 9. DET Curve (FNR vs FPR)

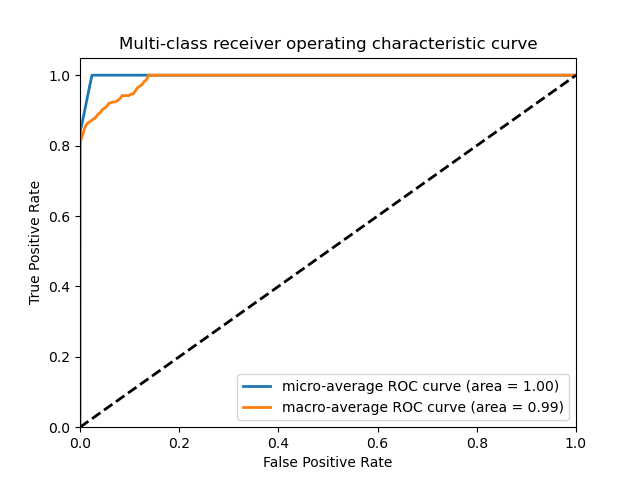


Figure 10. ROC curve (TPR vs FPR)

**Using rectified images only**

For this task, the model of choice was also a neural network. This method uses a model which consists of a series of MaxPooling operations followed by layers with an increasingly higher number of filters to be able to perform a form of feature extraction in this manner, before entering a dense layer that directly precedes the last layer with the *softmax* activation function for classification.

This method has a similar AUC as the previous one, although metrics such as the Decidability Index show a poorer performance. This is a sign that the feature extraction performed prior to the classification step, the one done using the autoencoder architecture, as a beneficial addition to the recognition pipeline.

The Area Under the Curve for this approach was 0. 9900, the Equal Error Rate was 0.0312 and the Decidability Index was 0.0973.

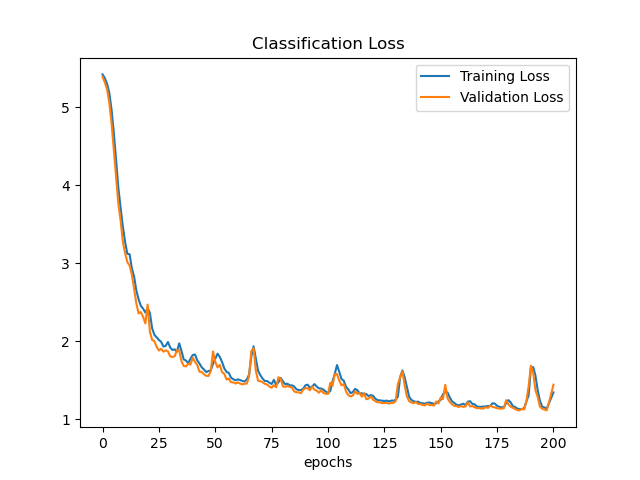


Figure 11. Classification Loss

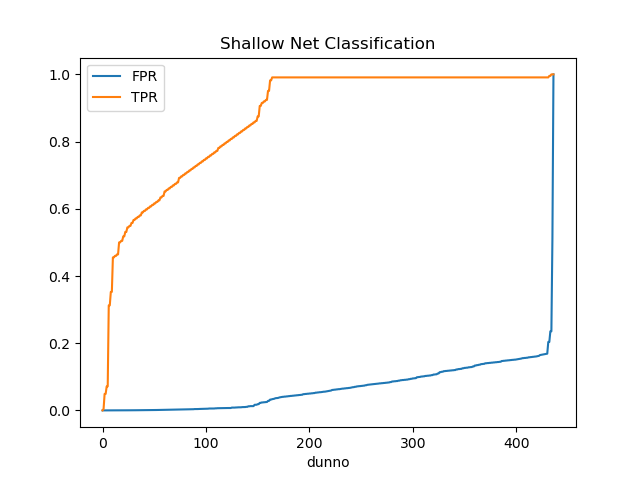


Figure 12. True Positive rate and False Positive Rate

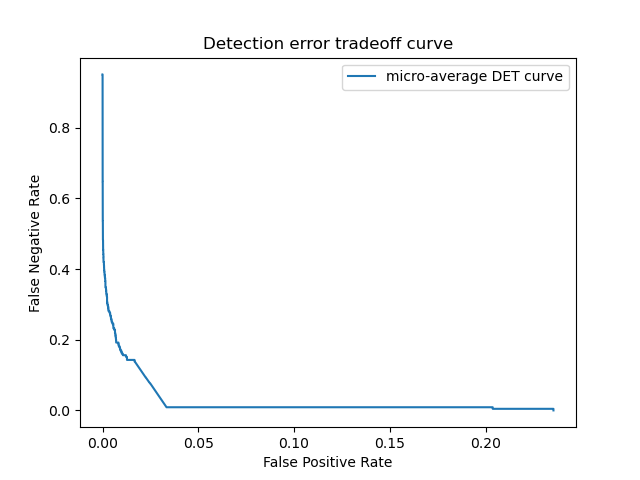


Figure 13. DET Curve (FNR vs FPR)

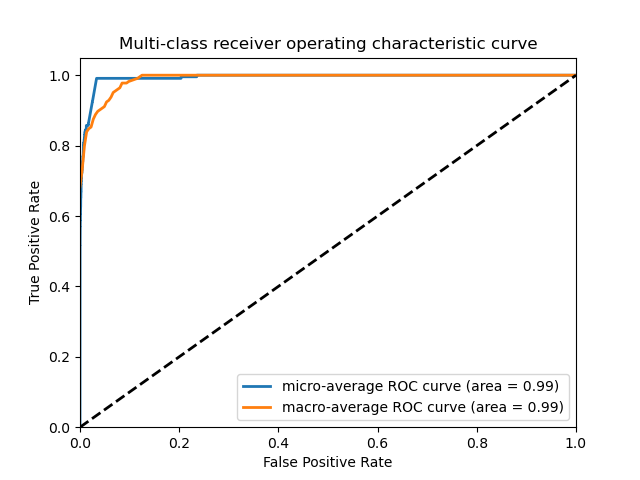


Figure 14. ROC curve (TPR vs FPR)