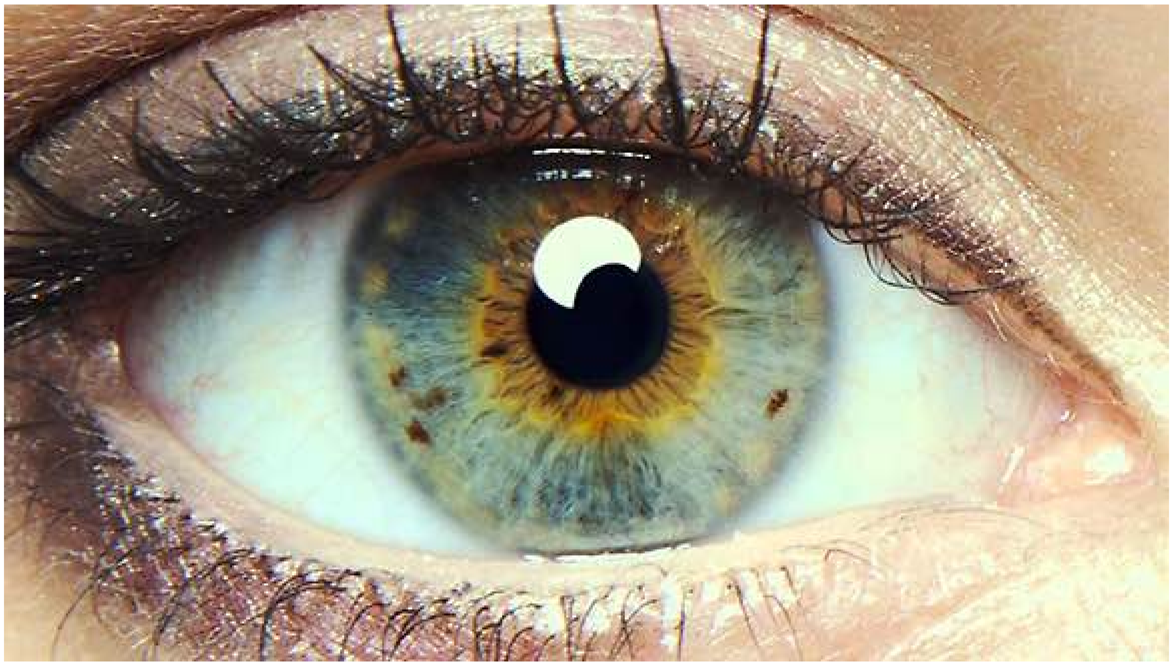
GROUP 17

EURECOM
Sophia Antipolis

AML Challenge 2 :

Glaucoma Diagnosis and Segmentation

Vincent Labarre, Ankita Sahoo, Yassine Elkheir, Lamia Salhi

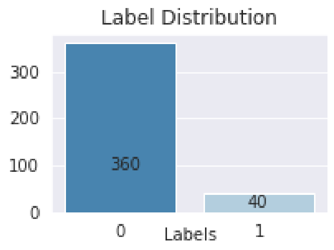

Crédits images : © Koba Films

I. Introduction :

During this challenge, we tried to diagnose glaucoma from images of the inside back surface of the eye. Our motivation is to develop a deep learning system that can detect glaucoma disease. We aim to work with multiple convolutional neural network models UNet, VGG, and AlexNet and test different kinds of data augmentation. Later we can choose the best performing model.

II. Data Pre-processing :

The training dataset is composed of 400 images.This number is quite low, therefore we tried to implement data augmentation to increase the amount of data for improving the model results. Moreover, from the shown label distribution on the right, we can say that this distribution is skewed. To overcome the imbalancement, and also allow our models to cope with a larger amount of data, we tried different methods of data augmentation and upsampling that will be presented in this part.



a. Upsampling:

To remedy the skewed distribution of labels, we upsampled the label 1 with different percentages to see what happened.

label0 % - label1 % (label0 nb - label1 nb)	36%-64% (360-640)	45%-55% (360-440)	50%-50% (360-360)	55%-45% (360-294)	60%-40% (360-240)	78%-22% (360-90)
Losses (train / val)	0.0004 / 0.0153	0.0044 / 0.0160	0.0161 / 0.0218	0.0018 / 0.0091	0.0456 / 0.0466	0.0094 / 0.0177
AUC (train / val)	0.9414 / 0.9248	0.9298 / 0.8651	0.9439 / 0.9257	0.9183 / 0.9230	0.9534 / 0.9291	0.9375 / 0.9158
Submission	0.86407	0.80725	0.90462	0.87521	0.92832	0.91890

Table 1 : Result Using Upsampling on the baseline model ("UNet")

We note that the results are better when we keep the predominance of the label 0, because we imagined that there is this predominance too in the testing set. Thus we choose to upsample the label 1 as 90 or 240 and keep the label 0 same as before that is 360 and we noted that respectively (360-90) and (360-240). This highlights how, more or less, upsampling can improve our different models and prevent overfitting.

b. Data Augmentation:

We tested different kinds of data augmentation.

1. Rotation Augmentation Technique :

We implemented four functions to increase the number of images, making a precise rotation (45, 90, 180, or 270 degrees) of a given image (and its segmentation). This technique is very powerful as it creates artificial variations in existing images to expand an existing image data set. This creates new and different images from the existing image data set that represents a **comprehensive** set of **possible** images.

label 0 % - label1 % (label0 nb - label1 nb)	36%-64% (460-640)	45%-55% (460-440)	50%-50% (360-360)	60%-40% (360-240)	78%-22% (360-90)
Losses (train / val)	0.0611 / 0.4857	0.0832 / 0.3125	0.0438 / 1.5408	0.0329 / 0.3177	0.0383 / 0.1633
AUC (train / val)	0.9460 / 0.8308	0.9548 / 0.8761	0.9562 / 0.8267	0.9382 / 0.9050	0.9432 / 0.8586

Table 2 :Result Using Rotation Upsampling on the baseline model ("UNet")

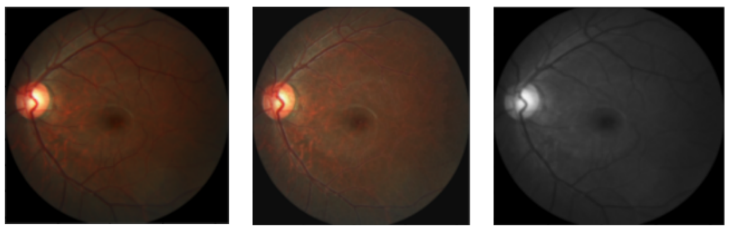
As we can see on this table, the results are better when we keep the predominance of the label 0, because this imbalancement actually exists in the train, the validation and the test set. This method makes our model more robust and thus can overcome overfitting.

2. Lightning conditions :

The different lightning conditions and intensity variations among images across various databases were circumvented by performing normalization of the histogram using ***Contrast Limited Adaptive Histogram Equalization (CLAHE).***
For this, we implemented a function that allows us to improve the contrast of images. It takes as input an image, and generates an image with a better contrast. Instead of using Adaptive Histogram Equalization (AHE) which over amplifies noise in relatively homogeneous regions of an image, we used CLAHE, a variant of AHE to prevent this by limiting the amplification.

3. RGB to Grayscale :

We also tried to remove the last dimension of every input, which responds to the color channel, so instead of working with a model that takes in an image with 3 channels (RGB), we transformed images to GrayScale and trained our base model. We believe that it makes the model more robust to different devices, clinics, and data collection processes.



4. CLAHE and RGB grayscale on the Baseline Model :

In this part we introduce and discuss the results of CLAHE and RGB to Grayscale data augmentation.
We darkened the images by a random factor belonging to the interval [0.4 ; 0.8] and here is a chart resuming our results.

Best scores achieved Augmentation- Class 1	0 augmented images	400 augmented images	800 augmented images	1200 augmented images
Loss (train / val)	0.0300 / 0.0344	0.0331 / 0.0491	0.0181 / 0.0705	0.1231 / 0.1312
AUC (train / val)	0.9495 / 0.9137	0.5576 / 0.6136	0.4702 / 0.6159	0.4885 / 0.6633
Submission	0.88378	0.63592	0.50314	0.64677

Table 3 : Result Using Lightning conditions Upsampling on the baseline model (“UNet”)

As a consequence, adding augmented images entailed poor performances. It is because there is no problem of brightness in our images (in fact, the optic nerve is very shiny).

Performance Data (360/40)	Intensity Conditions (CLAHE LIGHTNING)	From RGB to GrayScale	Intensity Conditions (CLAHE + GrayScale
Train (Loss, Score)	0.0680, 94.62%	0.3254, 91.54%	0.0721, 93.68%
Validation (Loss, Score)	0.2581, 92.34%	0.4254, 89.65%	0.3027, 90,81%

Table 4 : Result Using Lightning conditions and GrayScale images with the baseline model (“UNet”)

Using the CLAHE function, we can see that the score improves slightly. In addition, we can see the absence of the problem of overfitting. We can say that standardization of the contrast of the images gives good results.
Concerning GrayScale augmentation, although the standard model performance decreased, we believe that it makes the model more robust to different devices, clinics, and data collection processes.

To conclude, for the preprocessing of the two models we used vanilla upsampling for VGG and CLAHE with rotational upsampling for AlexNET. RGB to grayscale data augmentation wasn’t chosen because it didn’t improve our results.

III. MODELS :

In this part we introduce the results for two of our models: VGG and AlexNet.

a. Metrics :

For all the models we tried we used AUC and confusion matrix as we’re dealing with a classification problem. The confusion matrix helped us to see that often one model learns better than the class 0 but struggles to classify the label 1. For the confusion matrix, we used the probabilities that the trained model gave us on the validation set and discretized them to be 1 or 0 according to a threshold = 0.5. This threshold seemed more convenient here to compare the different models.

b. VGG Network:

1. Model, preprocessing and regularization

VGG is composed of a series of convolutional networks in which max pooling is used and it is also composed of three fully-connected: the first two have 4096 channels and the third one 2 channels corresponding to each class (0 and 1).
VGG takes input images of size 224 x 224. Thus we had to resize the images. We also used upsampling to improve our results as shown in the first part.
We used some dropouts to avoid overfitting and regularize the network. Finally, the cross entropy loss function was chosen because it was the most adapted function in our situation and it was giving us satisfying results. In the rest of this part we will compare different tunings that we’ve made.
The Jupyter notebook allows you to test different VGG (VGG11, VGG13, VGG16, VGG19). We first tried VGG19 and tuned our model as explained below

(label0 nb - label1 nb)	Dropout (Do) tuning **(360-90 / 360-240)**			Batchsize (Bs) tuning **(360-90 / 360-240)**			Optimizer tuning (dropout=0,72,Bz=4) **(360-240)**	
model	Do=0.5	Do=0.72	Do=0.9	Bs=4	Bs=8	Bs=16	Adam	SGD momentum=0.9
Training (Loss - AUC)	0.7777 / 0.9373	0.9651 / 0.9672	0.7777 / 0.9373	0.9634 / 0.9351	0.9651 / 0.9672	0.9975 / 0.9926	0.9431	0,9998
Validation (Loss - AUC)	0.8093 / 0.9926	0.856 / 0.8337	0.8133 / 0.9026	0.8541 / 0.9431	0.856 / 0.8337	0.6899 / 0.8439	0.9431	0,7924

Table 5 : Result Using different tunings on VGG19, lr=1e-4, cross entropy

2. Dropout

We noted after a few tests that the VGG is prone to overfitting. A way to avoid this is to perform Dropout, which randomly subsamples (with a probability p) the outputs of the dropped-out layer to reduce the capacity or thinning the network during training. Here, we tested for p=0.5, p=0.72 and p=0.9. In fact, p=0.72 is the best compromise between overfitting (with p=0.5, too little dropped-out connections) and underfitting (with p=0.9, too many dropped-out connections).

3. Batchsize

We also tuned the batchsize to avoid overfitting. Mini Batching allows the model to learn the model by batches of samples and thus makes the learning computationally more efficient and can allow a more robust convergence because of more frequent updates. Here we tested VGG19 for three different batch sizes. As we can see the higher the batch size is, the more the model overfits. Thus, by aiming for the highest batch size on which our model doesn't overfit we chose Bs=4 which gave us a training AUC =0.9351 and a validation AUC of 0.9431.

4. Optimizer tuning

We used an ADAM optimizer which is one of the best performing optimisers. It is also easy to tune it as its default parameters work better.There are various claims which state that ADAM does not converge better and gets stuck in local minimums and in this scenario SGD+momentum optimizer works better. Thus, we also experimented the model with the SGD+Momentum optimizer with a momentum of 0.9 to confirm that our model does not converge at local minima.After all the experimentation we found ADAM working better than SGD+momentum optimiser.Thus we chose ADAM.

5. Neural network (number of layers)

We also tested different numbers of layers and compared VGG16 and VGG19 with the same tunings. We can see that with VGG16 the model slightly overfits. Moreover, by testing VGG13 and VGG11 we also found that the model was overfitting a lot.
In fact, we tuned our parameters for VGG19 and they are not adapted for the other models. But even after testing multiple tunings for the other models we didn't find a model better than the VGG19 with dropout=0.71 and batch_size=4.

model	VGG 16 (Adam, dropout=0.7, batch_size=4)	VGG 19 (Adam, dropout=0.71, batch_size=4)
Training (AUC)	0.9652	0.9351
Validation (AUC)	0.9207	0.9431

Table 6 : Result of VGG19 and VGG16, lr=1e-4, cross entropy, adam optimizer, dropout=0,71, batch_size=4

To conclude, our best model is the VGG19, with batch size equal to 4, with dropout of 0,71 and adam optimizer trained with cross entropy loss.

c. AlexNet:

1. Model, preprocessing and regularization

AlexNet contained eight layers; the first five were convolutional layers.Some of them followed max-pooling layers (the second, the fourth and the fifth layers), and the last three were fully connected layers. It used the non-saturating ReLU activation function. AlexNet takes input images of size 256 x 256. We used some dropouts to avoid overfitting and regularize the network, and we normalized contrast images using the CLAHE function as it gives good results as presented above.
ALexNET was tuned the same way as VGG, in the next parts we introduce two other tunings we've made on AlexNET. Also we tried a different way of training the model by using cropped images on the disc area. But we did not proceed to use this process as it was overfitting the model.

2. Tuning : data augmentation

In general we apply binary Cross Entropy as the loss function, so we tuned our parameters, and we have tested it on our data with rotational upsampling and without it.
On the table we can see that the model learns better the label one with upsampling. Without it no diseased person was classified even though the training and validation AUC are satisfying. With upsampling, the losses are lower, and glaucoma is detected but there is a slight overfitting. Thus, we tried to overcome overfitting through a different loss function.

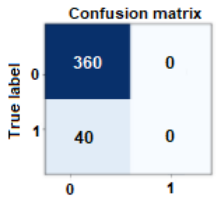
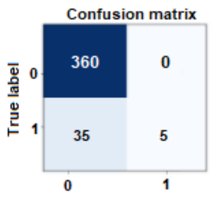
Performance	Lightning Normalization using CLAHE Without Upsampling (360 - 40)	Lightning Normalization using CLAHE with rotational Upsampling (360 - 240)
Training (Loss - AUC)	0.6104 - 0.9554	0.5241 - 0.9904
Validation (Loss - AUC)	0.6622 - 0.9217	0.5917 - 0.8838
Confusion Matrix (ON Validation SET)		

Table 7: AlexNET with CLAHE, lr=1e05, cross entropy, adam optimizer, dropout=0,71, batch_size=4, with and without upsampling

3. Tuning : loss function

As can be seen, the Cross Entropy shifts towards predicting all images as healthy to improve the accuracy, that is what causes overfitting. We have implemented another loss function, called the exponential weighted cross entropy :

$$ExpWCrossEntropy = W1 * (e^{y*log(\hat{y})} - 1) - W2 * (1 - y) * log(1 - \hat{y})$$

In practice, we have put **W1 = 1** and **W2 = 0.1,** so that we can enforce gradients from data with label 1 to be bigger than label 0 when the prediction is wrong.

Performance	Lightning Normalization Without Upsampling (360 - 40)	Lightning Normalization with rotational Upsampling (360 - 240)
Training (AUC)	0.9900	1.0000
Validation (AUC)	0.9283	0.9210

*Table 8 : AlexNET with CLAHE, lr=1e05, **ExpWCrossEntropy**, adam optimizer, dropout=0,71, batch_size=4, with and without upsampling*

We see an improvement using the new loss function compared to the cross entropy function, there is still the overfitting problem, but we see that the model is doing slightly better on our validation set. By changing the parameters, we see an improvement of our results, the best result with the corresponding parameters will be shown in the next session. To conclude, we chose an exponentially weighted cross entropy function without upsampling as it allows us to predict more glaucoma. Later we played with the other tuning in order to overcome the overfitting and you can find our best model result in the next table.

IV. Model Selected and Performance :

Although VGG and Unet were also giving us satisfying results,we chose AlexNet as we could tune the model more appropriately and get a better validation AUC. At first, We expected the VGG19 to have better results than the ALexNET because for the ImageNet Large-Scale Visual Recognition Challenge, VGG was an improvement to AlexNet. However, we saw the Alexnet fetched us a better result as it is a simple model with fewer layers and training it for a small dataset was more efficient.We also note, from the confusion matrix shown before, that the label 0 is better learned than the label 1 which is due to the imbalancement of the dataset that we have to keep even with upsampling as explained in the first part.

Performance	ALexNet Best Model ((360/40), dropout = 0.72, lr = 1e-05, wd = 1e-05, batch_size = 8, num_of_workers = 32)	Unet (upsampling 50)	VGG19 Best Model ((360/240), dropout=0,72,lr = 1e-04, batchsize=4, Adam optimizer)
Training (AUC Accuracy)	0.9598	0.93842	0.9351
Validation (AUC Accuracy)	0.9582	0.93760	0.9431
Testing on 30% (AUC Accuracy)	0.97487	0.93603	0.92175

Table 9 : Comparison of AlexNET, Unet and VGG19 best models

V. Conclusion :

Thus through this project, we learnt how to create a model to classify image datasets and do its segmentation. For the scenario of classifying Glaucoma, we first preprocessed the dataset by upsampling and data augmentation methods and then fed the processed dataset through convolutional neural networks. By comparing VGG, Unet and Alexnet we found that the Alexnet model is performing better on the provided image dataset. The hyperparameters were tuned to avoid overfitting issues as well as enhanced the Test AUC of the model. If we would have more time to experiment, we would probably have experimented with the Dense Net model and finished implementing our own ResNET model that we didn't complete due to time limitation.