Challenge 2 : Glaucoma diagnosis and segmentation

Group 50 : Nicolas BOINAY, Mathieu OLIVIER, Maxime MICHEL, Matteo NOTTARIS

Project Objective

- Glaucoma is a disease that impacts the eyes and can cause the loss of sight.
- The issue is that once you get the symptoms it is already too late.
- Our goal :
  - Based on work already done, find a simple way to detect glaucoma withtout the help of experts but simply machine learning.
  - We worked on two different methods : features exploitation and CNN on area of interest.

Project steps

- Segmentation of the OD and OC
  - The original pipeline was not changed, the segmentation maps made with the U-net were used.
- Feature engineering and selection
  - The first method is to create and exploit new features
- Neural nets
  - The second method consists in using several CNN on the optic disc area.
- Performance conclusion

Feature *Engineering*

Feature engineering

- Creation of new features to train and use our classifiers and improve accuracy. This features are caracteristics of the segmentation map of the optical disc and optical cup:
  - For example : eccentricity, perimeter or orientation
- 26 new features combined in the same array. A DataFrame was built to ease the readability and analysis.
- A classical pre-processing step was added (skikit Imputer)

	VCDR	Contrast	SumCup	SumDisc	RationSum	X_Gap	y_Gap
0	0.636364	0.025032	560.0	1384.0	2.471429	-0.087294	-0.779490
1	0.772727	0.026834	814.0	1212.0	1.488943	0.600897	0.681259
2	0.735294	0.026927	537.0	922.0	1.716946	0.267306	-0.736572
3	0.656250	0.030969	332.0	837.0	2.521084	0.314217	0.622713
4	0.431818	0.038205	315.0	1336.0	4.241270	-0.181715	3.287841

	Cup_Ecc	Disc_Ecc	Major_Axis_Cup	...	Tension_disk	Courbure_cup
0	0.433612	0.501863	28.131979	...	0.5	0.072430
1	0.574482	0.631584	35.597504	...	0.5	0.090226
2	0.252888	0.221654	26.593462	...	0.5	0.091667
3	0.392519	0.332405	21.430751	...	0.5	0.084337
4	0.263552	0.568388	20.395773	...	0.5	0.096386

	Courbure_disk	av_Cup	av_Disk	std_cup	std_Disk	gradient_cup
0	0.081361	0.007546	0.016003	0.081933	0.112190	0.158220
1	0.088050	0.008749	0.011012	0.080025	0.085007	0.160826
2	0.077338	0.006709	0.009997	0.074617	0.086143	0.145434
3	0.098684	0.004398	0.009050	0.062030	0.081552	0.114903
4	0.092814	0.003598	0.010806	0.052490	0.078652	0.104501

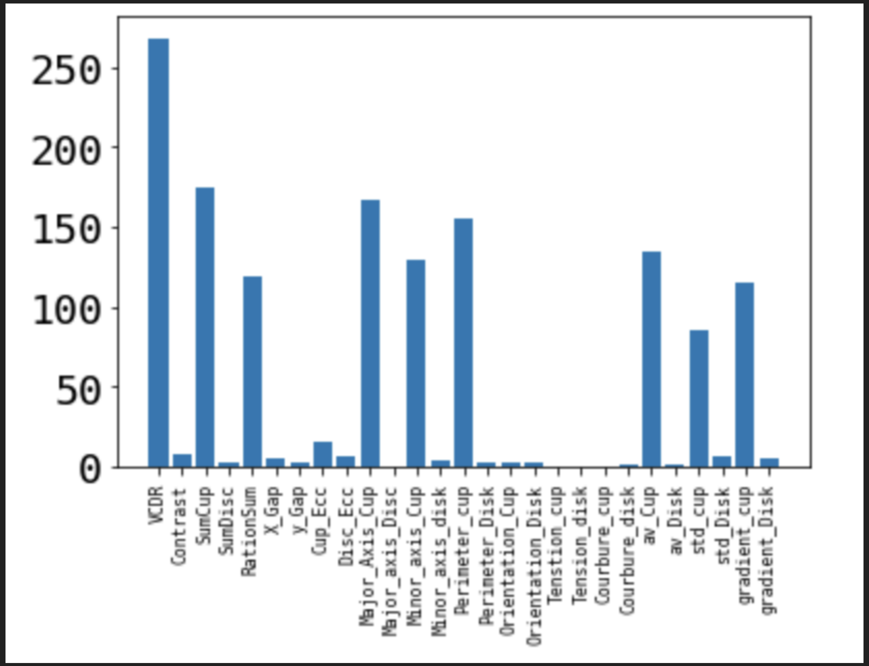
	gradient_Disk	Label
0	0.234691	1.0
1	0.180574	1.0
2	0.175111	1.0
3	0.165219	1.0
4	0.177928	1.0

Our DataFrame with the features created

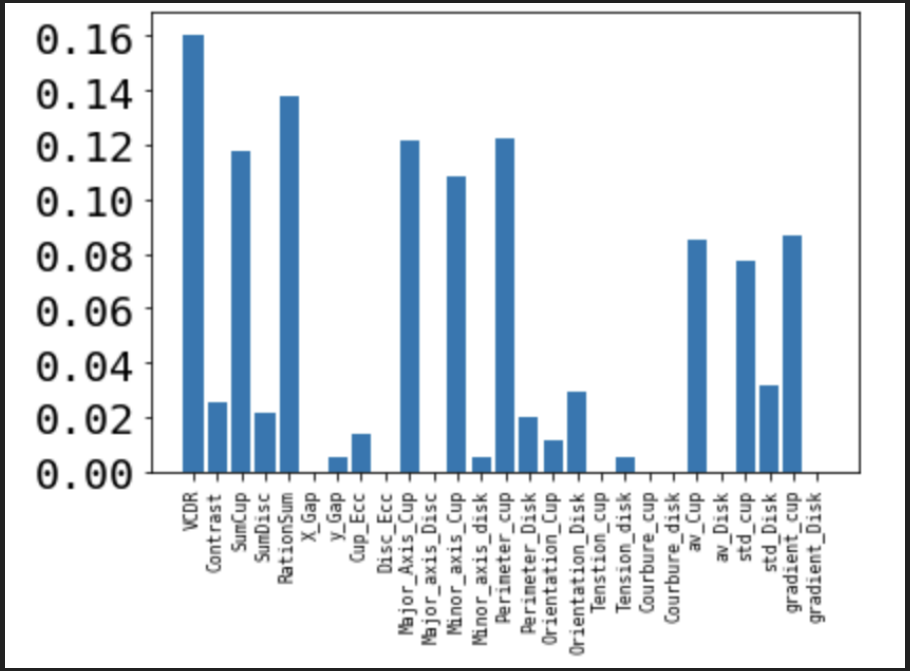
Feature selection

We have numerical input and categorical output. This is a classification predictive modeling problem with numerical input variables. The most common techniques are correlation based, although in this case, they must take the categorical target into account. Because of this we decided to use ANOVA correlation coefficient (linear) and the mutual information.

Anova -f from sklearn



Mutual information using mutual_info_class() from sklearn



So these two graphs show us that the most relevant features are **VCDRs, SumCup, Major_Axis_Cup, Minor_axis_Cup, Perimeter_Cup, av_cup, std_cup, gradient_cup.** The fact that both give the same results reinforce the trust we have in this choice.

CNN method

In this second method we decided to focus on the zone of interest and apply directly a CNN on it without first exploiting features. The results of both methods will be discussed at the end.

- Glaucoma displays its main clinical symptoms in the optic disc region, in order to exploit that there is a first part of pre-processing :

  1. **Crop** the regions of interest to a 60x60x3 focused on the OD to exclude irrelevant background contexts.
  2. Use Histogram equalization, **CLAHE**, for contrast enhancement and normalization.

- The images are therefore enhanced, focused on the disc region and way smaller so they can be processed by a CNN directly.

- Before trying the neural nets, we decided to go through two other steps of pre-processing to improve the training:

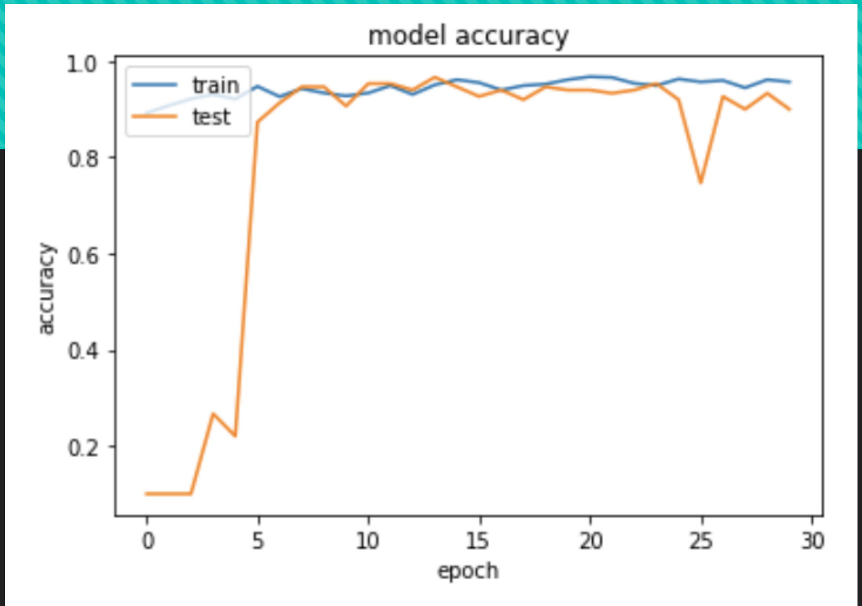
  1. Use a part of the validation set to train the models. It gives us 650 images of training and 150 of validation. This step is really important to avoid an overfitting caused by too few data when training.
  2. Use Data Augmentation : ImageGenerator tool from keras that makes our model more robust and again avoid an overfitting. It also prepares the right format of data set for our keras models.

- We then tried two neural nets and compared them. We decided to go with one fully trained and one pretrained.

Homemade CNN

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_4 (Conv2D)            (None, 60, 60, 64)        1792
_________________________________________________________________
batch_normalization_6 (Batch (None, 60, 60, 64)        256
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 30, 30, 64)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 30, 30, 128)       73856
_________________________________________________________________
batch_normalization_7 (Batch (None, 30, 30, 128)       512
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 15, 15, 128)       0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 15, 15, 256)       295168
_________________________________________________________________
batch_normalization_8 (Batch (None, 15, 15, 256)       1024
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 7, 7, 256)         0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 7, 7, 512)         1180160
_________________________________________________________________
batch_normalization_9 (Batch (None, 7, 7, 512)         2048
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 3, 3, 512)         0
_________________________________________________________________
flatten_3 (Flatten)          (None, 4608)              0
_________________________________________________________________
dense_8 (Dense)              (None, 1024)              4719616
_________________________________________________________________
dense_9 (Dense)              (None, 1024)              1049600
_________________________________________________________________
dense_10 (Dense)             (None, 1)                 1025
=================================================================
Total params: 7,325,057
Trainable params: 7,323,137
Non-trainable params: 1,920
_________________________________________________________________
```

CNN summary



This model gives us a very good accuracy. What we noticed is that we had to find a balance in the complexity of the model. Too much layers created a very important overfitting phenomenon and too few was giving a much worst accuracy.
We also used a lot of batch normalization to avoid overfitting.

The most important metric being AUC and not accuracy in our cases we based our final comparison with this metric. (See Performance Slide)

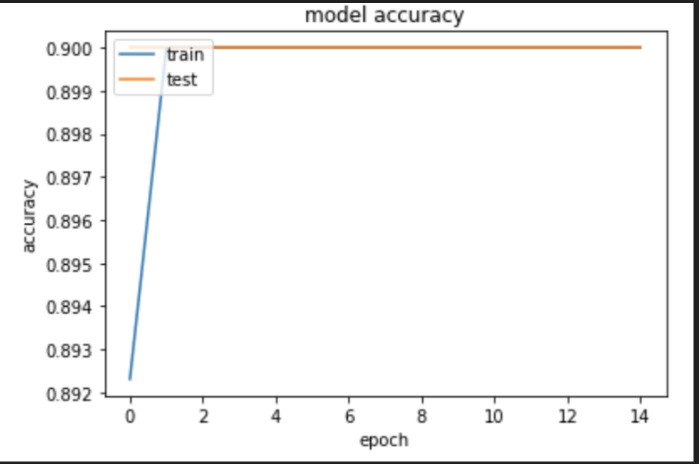
ResNet 18

```
ResNet18, preprocess_input = Classifiers.get('resnet18')
modelResNet18 = ResNet18((60, 60, 3), weights='imagenet')

for layer in modelResNet18.layers[:]:
    layer.trainable = False

modelRes = Sequential()
modelRes.add(modelResNet18)
modelRes.add(layers.BatchNormalization())
modelRes.add(layers.Flatten())
modelRes.add(layers.Dense(units=1024,activation="relu"))
modelRes.add(layers.Dense(units=1024,activation="relu"))
modelRes.add(layers.Dense(units=1, activation="sigmoid"))

modelRes.compile(loss='binary_crossentropy', optimizer=optimizers.Adam(lr=1e-6),metrics=["accuracy"])
```

We decided for the second net to use a pretrained one. We chose ResNet 18 because it is an 18 layers deep model, so the process time is short, and according to what we read is very suited for image classification like the one we do here.

We used imagenet weights as a base, our read of some papers all indicated it was the best way of using that net.

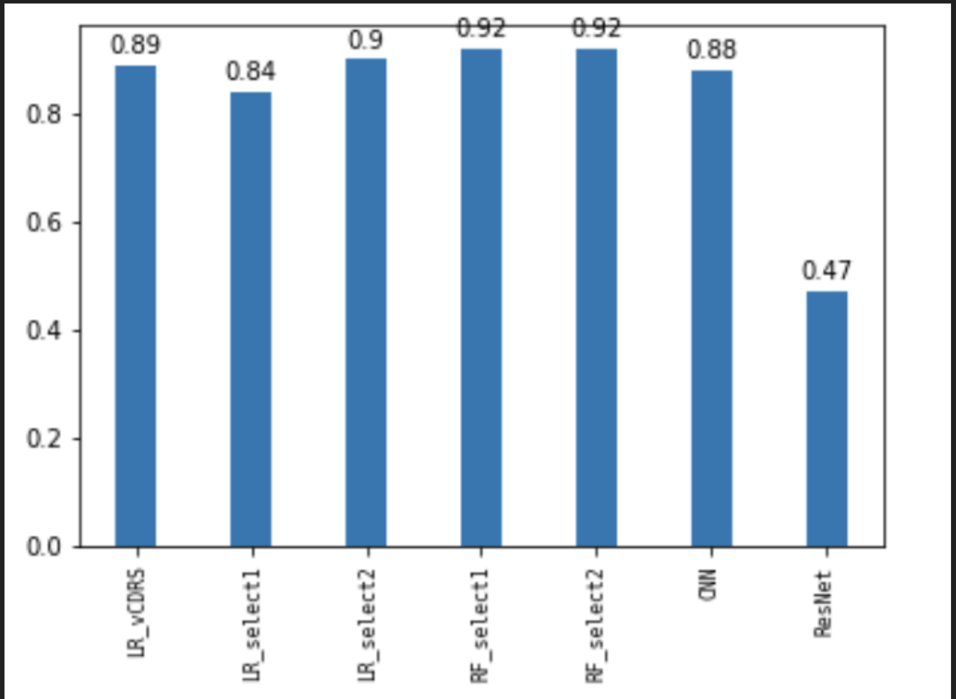
Then we added and trained the last layers to adapt to our objective.



As the graphs shows the accuracy did not evolve during training and stayed at 0.9.
What we happened here is a consequent overfitting, that seems to explain why the val_accuracy doesn't change. To prevent that we tried adding batchNormalization, then a Dense layer and used the imageGenerator tool but we haven't been able to avoid this problem. This issue is why we went from resNet 50 to resNet 18, we thought that with less layers the model would avoid overfitting.
We didn't manage to find a solution but we think that maybe some configuration on the model it self could help, or using something to generate more data.

Performance

AUC scores for each models on Validation



Select1 : VCDR, sum_cup, ratio_sum, major_axis_cup, minor_axis_cup, perimeter_cup

Select2 : all the features selected from our feature selection step

We decided to train 2 types of classifiers on several combinaison of features and 2 neural net for the second method.

Models :
- Logistic regression (on features)
- Random Forest (on features)
- CNN/ResNet (on cropped images)

The metric used is AUC which is the most pertinent in this study.
The Random Forest models give the best results for both combination of features. We therefor used RF on select2 for our final submission.
Both classifier are used with their default parameters, we decided to focus more on the feature preparation rather than classifier optimization.

It is also noticable that using a CNN directly on cropped images has a clear potential, without a good knowledge of those structures we succeded in having a very promissing score.

Conclusion

- We discovered that there were two ways to detect glaucoma :

  - Creating features and exploiting them. This method is the more transparent, the features created are the product of very simple computation on the masks. The process of analysing them and choosing them is very simple. This methods gave great results as we arrived at a AUC score of 0.97. We think that to have a better accuracy the first thing is to improve the creation of the segmentation mask on which is based all the features, maybe by using a high pass filter added to the U-net.

  - Applying a convolutional network directly on the area of interest. In our case, the optical disc area. This method seems to have a lot of potential but need a real work of configuration of the neural net and could lead to very heavy computation for high resolution picture. Using a 60x60 picture, which fasten the computation, is probably the main reason why the score is not so high. The use of a pre-train network could be the solution but it needs to be worked on to avoid overfitting, maybe by creating more data with a GAN.