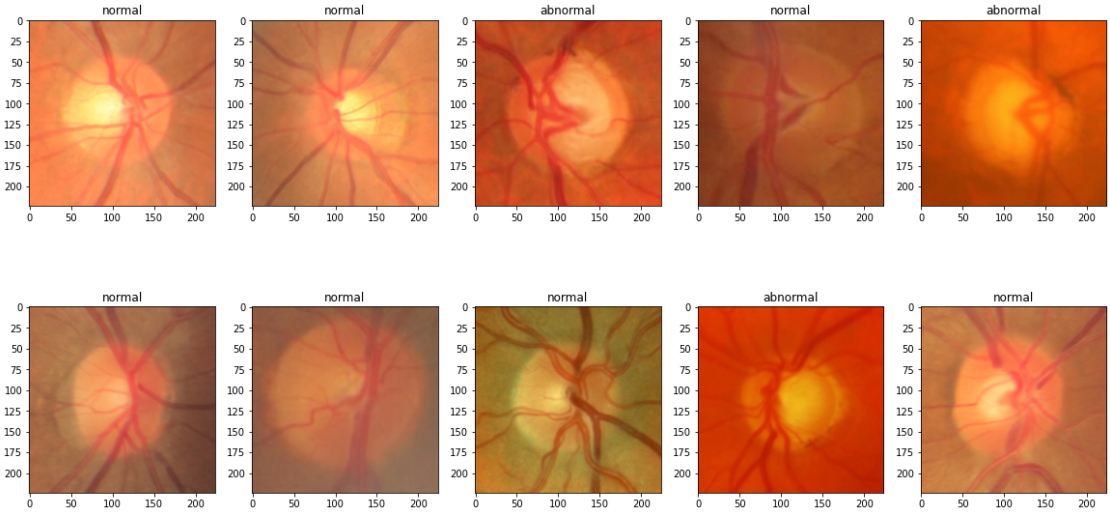
Neural classifier for the detection of glaucoma

Lorenzo Venieri

1) Exploratory analysis of data:

The data available consists of 1707 fundus images (photographies of the rear of an eye), each labeled as normal or abnormal.

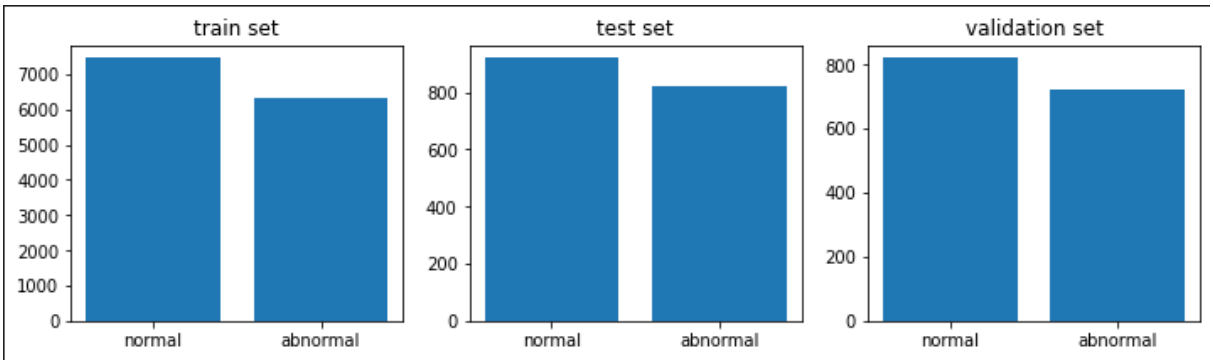


Some images of the dataset, with their label

We have 10 folds, each one containing all the images, divided into three sets: train (1379 images), validation (154 images) and test (174 images). To create these folds the images were presumably shuffled and split into 10 sets to build, for each fold, a training set out of 8 of them and the other two as validation and test sets.

Each fold in the set X_train contains 1379 images of size 224x224, with 3 color channels.
Each fold in the set y_train contains 1379 instances with 2 labels each.
Each fold in the set X_test contains 174 images of size 224x224, with 3 color channels.
Each fold in the set y_test contains 174 instances with 2 labels each.
Each fold in the set X_valid contains 154 images of size 224x224, with 3 color channels.
Each fold in the set y_valid contains 154 instances with 2 labels each.

In each fold there are 788 abnormal instances and 919 normal instances.
If we sum up all the images in the different folds, we can have an idea about how the classes are distributed between train, validation and test sets:
In the train sets there are 7452 normal instances and 6338 abnormal instances.
In the test sets there are 919 normal instances and 821 abnormal instances.
In the validation sets there are 819 normal instances and 721 abnormal instances.



But the distribution in each fold may vary, for instance the distributions in fold0 and fold1 are:
Distribution of normal/abnormal instances in Fold0:
In the train set there are 754 normal instances and 625 abnormal instances.
In the test set there are 82 normal instances and 92 abnormal instances.
In the validation set there are 83 normal instances and 71 abnormal instances.

Distribution of normal/abnormal instances in Fold1:
In the train set there are 740 normal instances and 639 abnormal instances.
In the test set there are 91 normal instances and 83 abnormal instances.
In the validation set there are 88 normal instances and 66 abnormal instances.

Overall the classes are just slightly imbalanced: there are about 15% more normal instances than abnormal ones.

2) Search of best model and hyperparameters:
The search for the best model and hyperparameters was done just on the Fold0, before training the model on each fold to validate the results using cross-validation.
5 models were built, they will be discussed below.

Model 1:
The first model is built on an EfficientNetB0 feature extractor, pretrained on the ImageNet dataset. On top of the feature extractor, that was set as untrainable during the training, was built a classification head that consists of 4 layers: GlobalAveragePolling2D, BatchNormalization, dropout (0.2), and at the end a FC layer.
If we look at the model summary the number of parameters are:
Total params: 4,057,253
Trainable params: 5,122
Non-trainable params: 4,052,131

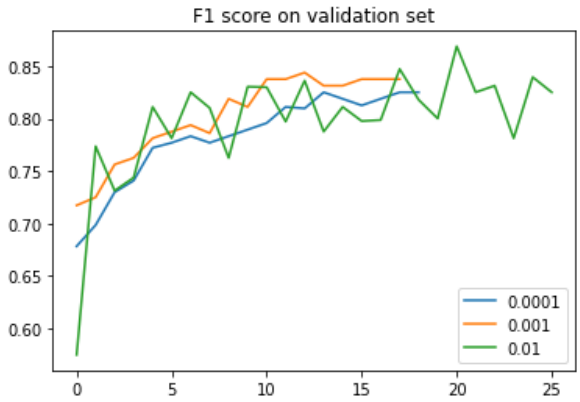
We don't do standardization nor normalization as preprocessing: in EfficientNet documentation is written that "This model takes input images of shape (224, 224, 3), and the input data should range [0, 255]. Normalization is included as part of the model."
Image augmentation could be done but we reached good results even without using it.
It was used Adam as optimizer, which has proven to be a good optimizer that needs little hyperparameters optimization: we just need to find a good starting learning rate and Adam will automatically find optimal momentum etc. The starting learning rates tried were 0.0001, 0.001 and 0.01.
The batch size for all the models trained in this work is 32.
The loss function used is binary cross entropy, since we are dealing with a binary classification task.
The metric monitored was f1-score (average = micro: TP, FP and FN are computed globally) that is better than simple accuracy as it takes more directly into account recall, that for this task is particularly important since we would like to detect as much abnormal cases as possible, even if it comes at the expense of detecting more false positives (less precision), to start curing the disease as early as possible.
For this reason it could be used a metric that puts more emphasis on recall as the Fb-score, with b>1: this metric gives to recall b times the importance given to precision.
If we want to focus on precision and recall of the abnormal class we could use as metric only the f1-score on the abnormal class:

```
def f1_class1(y_true, y_pred): #F1 score for class 1
f1 = tfa.metrics.F1Score(num_classes=2)
return f1(y_true, y_pred)[1]
```

Another metric that could be useful to monitor is the recall on the abnormal class, this would be at the expense of all the other metrics, like precision on the abnormal class and precision and recall on the normal class. Depending by how much we want to avoid missing an abnormal case, we need to choose the right metric to maximize between these ones.

Model1 as described was trained with the different learning rates for a maximum of 30 epochs. To avoid overfitting and using too much computing resources the training was done using early stopping with patience = 5 and f1_score on the validation set as the metric to maximize. The training goes on until there are no improvements of the performance on the validation set for the last 5 epochs. When the training is done the best weights are restored and saved.

Plot of the f1-score on the validation set during the training of models 1 for different learning rates:

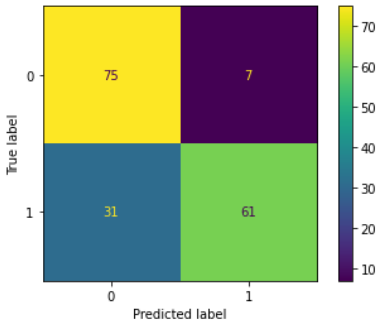


The best performance on the validation set was reached by the model with lr = 0.01 (on epoch 21).

Performance of the models on the test set:

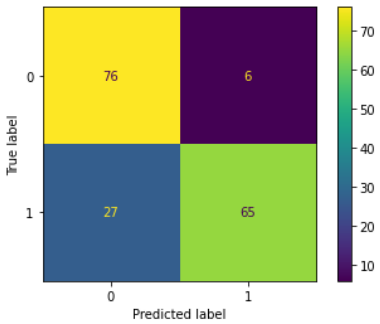
- lr = 0.0001

	precision	recall	f1-score	support
0	0.71	0.91	0.80	82
1	0.90	0.66	0.76	92
micro avg	0.78	0.78	0.78	174



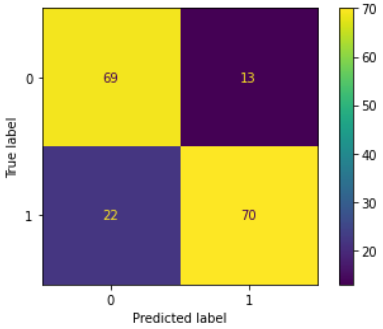
- lr = 0.001

	precision	recall	f1-score	support
0	0.74	0.93	0.82	82
1	0.92	0.71	0.80	92
micro avg	0.81	0.81	0.81	174



- lr = 0.01

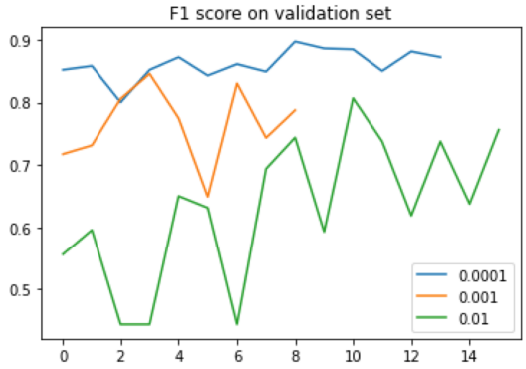
	precision	recall	f1-score	support
0	0.76	0.84	0.80	82
1	0.84	0.76	0.80	92
micro avg	0.80	0.80	0.80	174



We need high F1-score and high recall on the abnormal set: since we want to spot glaucoma when is present, we need to minimize the amount of False Negatives for the abnormal class (1).
Considering this, the best performing model on the test set is the Model1_lr0.01, we will use this one as a base to train the next model.

Model 2:
This model is built unfreezing the last 24 layers of the best performing Model1 that are not BatchNormalization layers (4 layers of the classification head + 20 layers of the pretrained feature extractor part of EfficientNetB0).
The best performance on the validation set was reached by the model with lr = 0.0001 (on epoch 9).

Plot of the f1-score on the validation set during the training of the models 2 for different learning rates:

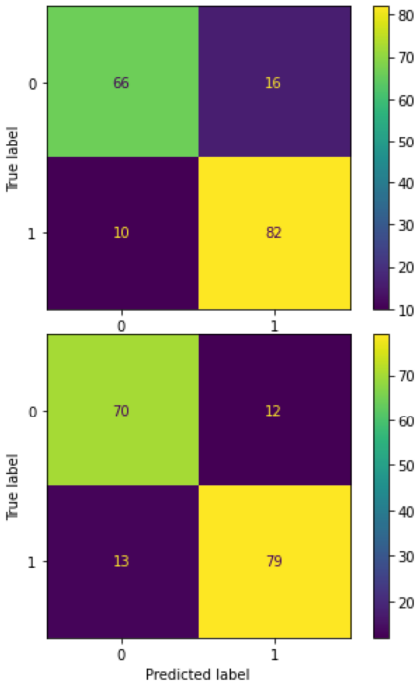


Performance of the models on the test set:
- lr = 0.0001

	precision	recall	f1-score	support
0	0.87	0.80	0.84	82
1	0.84	0.89	0.86	92
micro avg	0.85	0.85	0.85	174

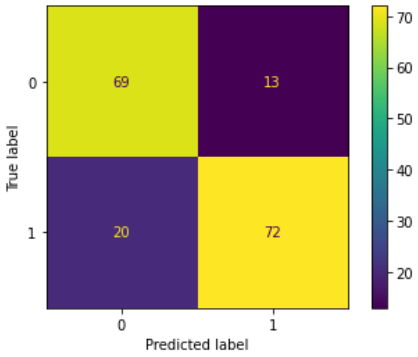
- lr = 0.001

	precision	recall	f1-score	support
0	0.84	0.85	0.85	82
1	0.87	0.86	0.86	92
micro avg	0.86	0.86	0.86	174



- lr = 0.01

	precision	recall	f1-score	support
0	0.78	0.84	0.81	82
1	0.85	0.78	0.81	92
micro avg	0.81	0.81	0.81	174



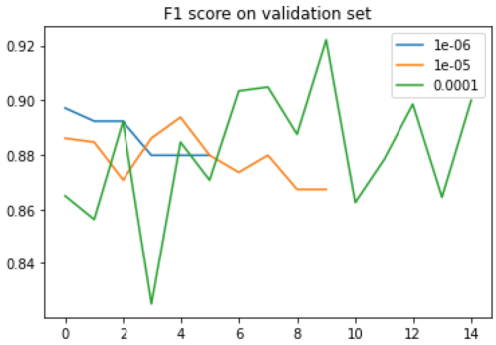
Although Model2_lr0.001 has the best overall performance, the recall of the Model2_lr0.0001 on the abnormal class is much better. At the expense of just 1% in f1-score, we have less false negative for the abnormal class. So we pick this Model to build Model3.

Model 3:

This model is built unfreezing all the layers of the best performing Model2 that are not BatchNormalization layers. When fine-tuning all the layers it is recommended to lower the learning rate to not destroy the pretrained weights, so we will analyze the performance of lr between 0.000001, 0.00001, 0.0001.

The best performance on the validation set was reached by the model with lr = 0.0001 (on epoch 10).

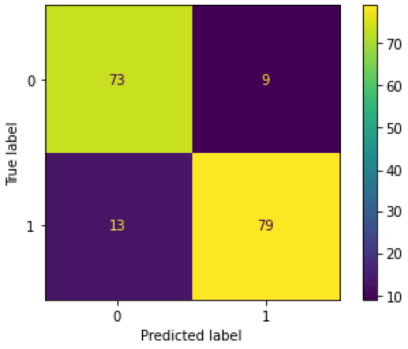
Plot of the f1-score on the validation set during the training of the models 3 for different learning rates:



Performance of the models on the test set:

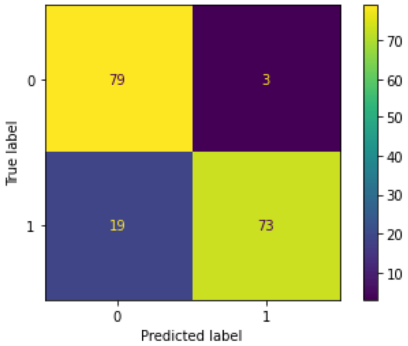
- lr = 1e-06

	precision	recall	f1-score	support
0	0.85	0.89	0.87	82
1	0.90	0.86	0.88	92
micro avg	0.87	0.87	0.87	174



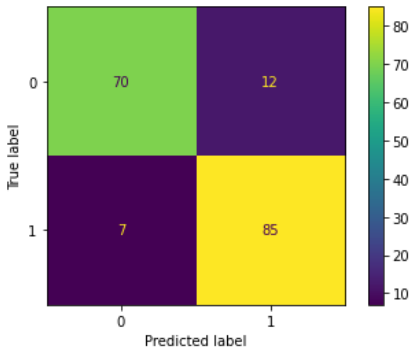
- lr = 1e-05

	precision	recall	f1-score	support
0	0.81	0.96	0.88	82
1	0.96	0.79	0.87	92
micro avg	0.87	0.87	0.87	174



- lr = 0.0001

	precision	recall	f1-score	support
0	0.91	0.85	0.88	82
1	0.88	0.92	0.90	92
micro avg	0.89	0.89	0.89	174

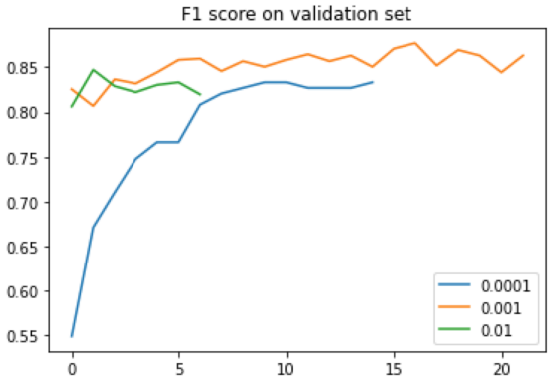


The best performing model on the test set is Model3_lr0.0001, both for overall f1-score and for recall on the abnormal class.

Model 4:

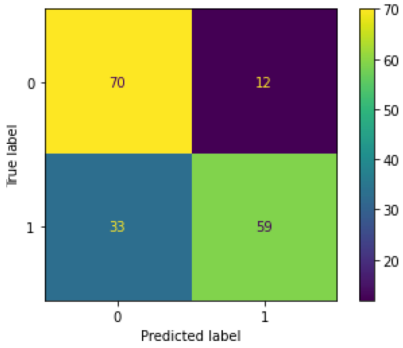
This model is built on a ResNet50 feature extractor, pretrained on the ImageNet dataset. On top of the feature extractor, that was set as untrainable during the training, was built a classification head that consists of 4 layers, analogously to what was done in model 1: GlobalAveragePolling2D, BatchNormalization, dropout (0.2), and at the end a FC layer.
This model has:
Total params: 23,600,002
Trainable params: 8,194
Non-trainable params: 23,591,808
The best performance on the validation set was reached by the model with lr = 0.001 (on epoch 17).

Plot of the f1-score on the validation set during the training of the models 4 for different learning rates:



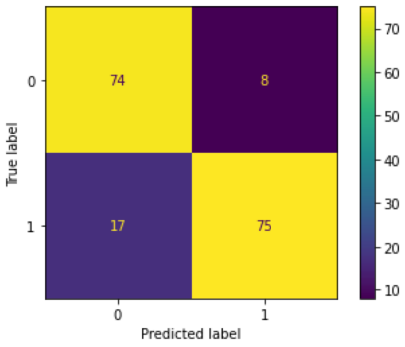
Performance of the models on the test set:
- lr = 0.0001

	precision	recall	f1-score	support
0	0.68	0.85	0.76	82
1	0.83	0.64	0.72	92
micro avg	0.74	0.74	0.74	174



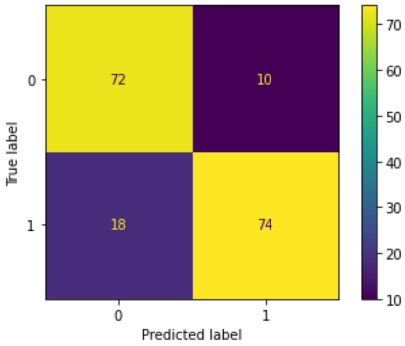
- lr = 0.001

	precision	recall	f1-score	support
0	0.81	0.90	0.86	82
1	0.90	0.82	0.86	92
micro avg	0.86	0.86	0.86	174



- lr = 0.01

	precision	recall	f1-score	support
0	0.80	0.88	0.84	82
1	0.88	0.80	0.84	92
micro avg	0.84	0.84	0.84	174

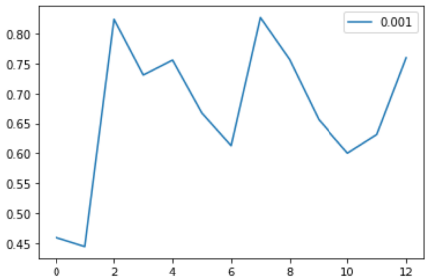


The best performing model on the test set is Model4_lr0.001, both for overall f1-score and for recall on the abnormal class.

Model 5:

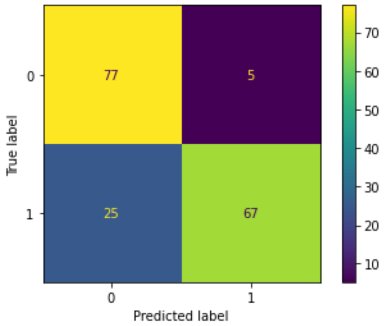
This model is built on a Xception feature extractor, pretrained on the ImageNet dataset. On top of the feature extractor was built a classification head that consists of 4 layers, analogously to what was done in model 1 and 4.
The model was only trained with lr = 0.01.
The training consists of 10 epochs with only the classification head trainable, followed by 20 epochs of training with the last 20 layers of the Xception feature extractor unfreezed.
During the last part of the training we have 7,320,394 trainable parameters out of 20,873,770.
The best performance on the validation set was reached on epoch 18 (10+8).

Plot of the f1-score on the validation set during the training of the model 5 for lr = 0.001:



Performance of the model on the test set:

	precision	recall	f1-score	support
0	0.75	0.94	0.84	82
1	0.93	0.73	0.82	92
micro avg	0.83	0.83	0.83	174



Election of the best model on Fold0:
Considering the f1-score and the recall on the abnormal class of the test set of Fold0, the best performing model is Model3_lr1e04, with f1-score = 0.89 and recall on the abnormal class = 0.92.
We will use this model to evaluate its performance and reliability on the whole dataset with cross validation.

3) Cross-validation:
This technique consists on dividing the dataset into k sets and use 1 of them as test set and the other k-1 to train the model (1 of these is usually used as validation set during training). If we repeat this procedure with all the k different choices of training and test sets we will have an evaluation of the ability of the model to perform on data not seen during training.
Doing this, each sample is used in the test set once and used to train the model k-1 times.

We already have the 10 folds ready to perform cross-validation. For each one of them we train the best model obtained on fold0 and train it on the train set of that fold, finding the best epoch of training using the validation set. Then we test the performance of this model on the test set of the given fold. Of course, for each fold the model trained on the previous fold is discarded and the training starts again from the best model elected on section 2. The f1-scores on the 10 different test sets are stored in a list and we will summarize the performances using the their mean and standard deviation (to have an idea of the variance).

The results obtained are a mean of 0.95975, with a standard deviation of 0.02274.

If we didn't have the 10 folds ready we could have built them either by hand or using the Kfold() scikit-learn class. As previously said, the procedure consists of partitioning the dataset into k folds and, for each step of the cross-validation loop, selecting k-2 of them to train the model, 1 of them as validation set during the training and the last 1 as test set. When doing this, it's important that each one of the k sets is representative of the entire dataset: the proportion of normal and abnormal cases in each set should be close to the proportion of the entire dataset. We also have to keep in mind that the original dataset order might not be random, so it is good practice to shuffle the dataset before partitioning it. If the data set is imbalanced, we could use stratified k-fold cross-validation to create partitions that contain a more equal mix of classes.

Final considerations:

Transfer learning from a pretrained model is a powerful tool. We started from EfficientNetBO, trained on the ImageNet dataset. This dataset contains millions of images, but very different from the fundus images that our task has to deal with. Nevertheless we were able, with not much computational resources nor a big dataset to begin with, to build a model that has a really good performance.
We proceeded by steps: first training the classification head of the model, then unfreezing the last layers of the model and only at the end fine-tuning the whole NN. This gave the learning process the ability to concentrate each time on the weights that needed to adapt more to the new task. If we proceeded without training the classification head on the frozen base first, initial epochs may overwrite the useful representations encoded in the pre-trained model.

It is important to consider that the good results obtained by the model on this dataset do not mean that our model will have such good performance on other datasets of fundus images that the model has not seen during training, obtained with different methods (as seen in [Diaz-Pinto, A., Morales, S., Naranjo, V. et al. CNNs for automatic glaucoma assessment using fundus images: an extensive validation. BioMed Eng OnLine 18, 29 (2019).] https://doi.org/10.1186/s12938-019-0649-y).
This problem comes from the fact that our model has not only never seen that data but the whole database has different expert labelers and image characteristics.