Report Challenge IMA 205 - Cardiac Pathology Prediction

Gaël Marcheville, 2nd year student at Télécom Paris

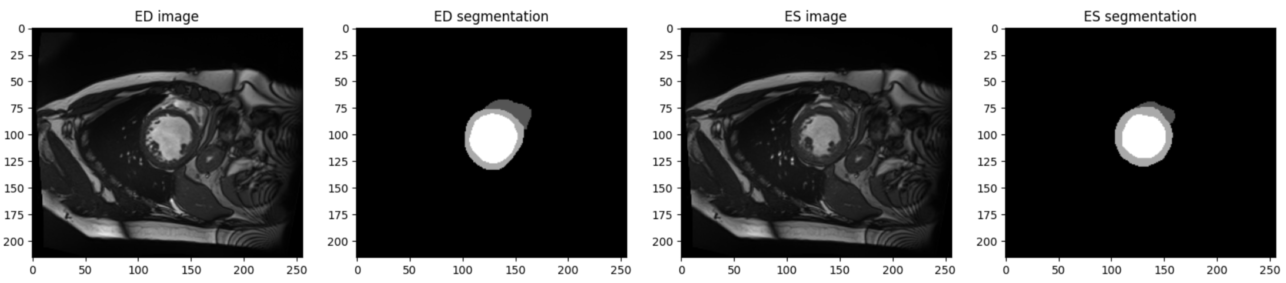
I. Introduction

1. Context

This report presents a solution to the "Cardiac Pathology Prediction" classification problem. The dataset consists of 100 cases for training and 50 for testing. For each case, we have images at end diastole (ED) and end systole (ES), along with segmentations of the myocardium, left ventricle, and right ventricle. The segmentations of the left ventricle are missing for the test data. Therefore, I performed this segmentation and proposed a classification method.

2. The dataset

The dataset is composed of 150 cases, each case being composed of 4 images (2 images at end diastole and 2 images at end systole) and 3 segmentations (myocardium, left ventricle, and right ventricle). Both images and segmentations are 3D volumes. Segmentation have values 0 for the background, 1 for the right ventricle, 2 for the myocardium, and 3 for the left ventricle (missing for test). Here is an example of a case:



3. The task

The goal of this challenge is to classify MRI images of the heart among five different diagnostic classes:

- 0 - Healthy control (HC)
- 1 - Myocardial infarction (MINF)
- 2 - Dilated cardiomyopathy (DCM)
- 3 - Hypertrophic cardiomyopathy (HCM)
- 4 - Abnormal right ventricle (RVA)

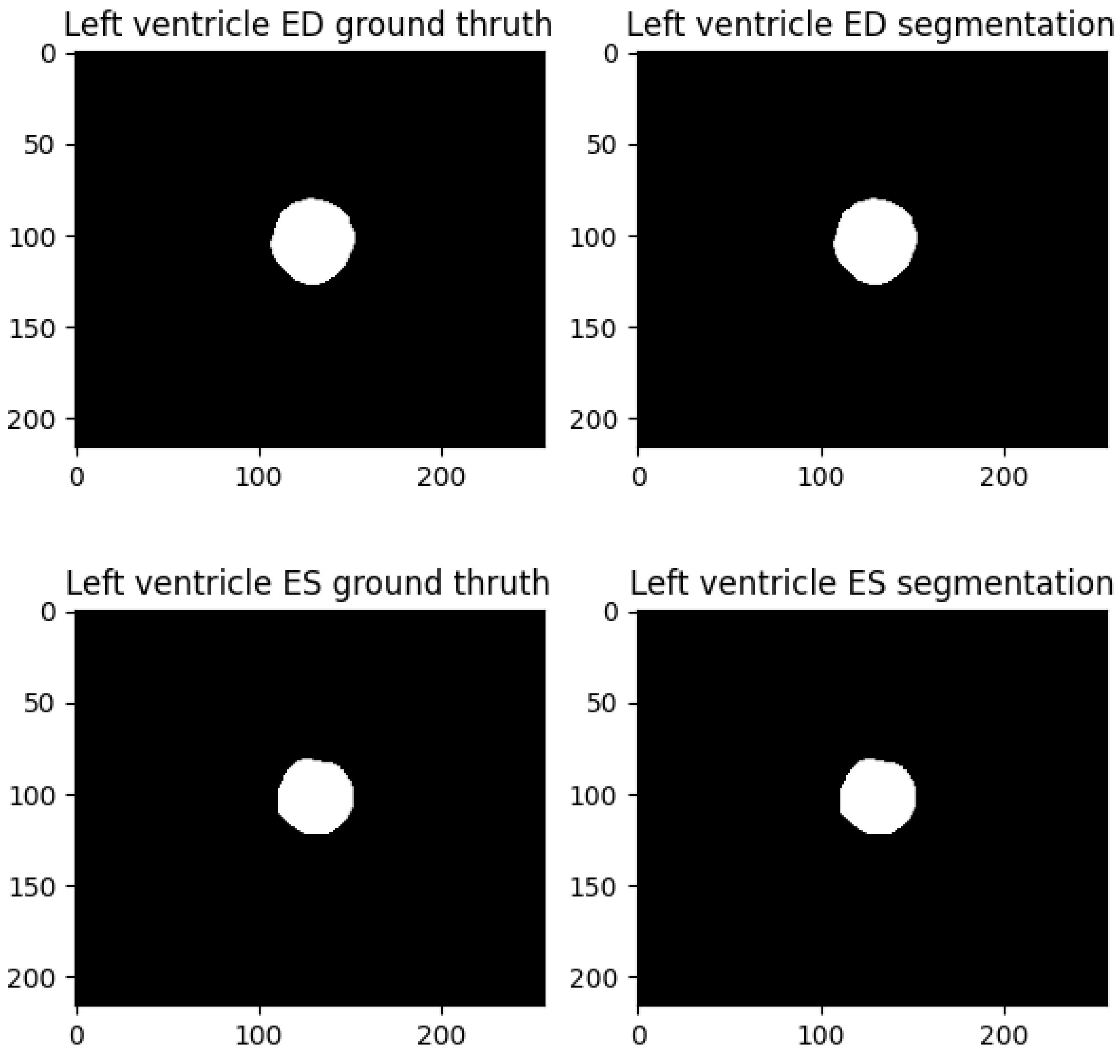
We have the label for the training set, but not for the test set. The evaluation metric is the categorization accuracy, which is defined as:

$$Acc = \frac{1}{N}\sum_{i=1}^{n} 1(f_i = y_i)$$

II. Segmentation of the left ventricle

Different methods are proposed in the articles to perform the segmentation of the left ventricle, such as using region growth with iterative thresholding by detecting the effusion into the surrounding myocardium and tissues [1]. I tried to implement some of these methods before abandoning them for a simpler but more efficient solution, directly exploiting the provided dataset.

Indeed, I discovered that by filling the inside of the myocardium segmentation, I obtained a segmentation of the left ventricle. To do this, I used the binary_fill_holes function from the skimage library. Here is an example of the result obtained:


Left ventricle ED ground thruth
Left ventricle ED segmentation
Left ventricle ES ground thruth
Left ventricle ES segmentation
0
50
100
150
200
0
100
200

Using this algorithm, I achieved a segmentation of the left ventricle with a similarity ranging between 99.99% and 100% compared to the left ventricle segmentation provided in the training data. Therefore, I consider this segmentation to be highly effective and have not been able to obtain better results by implementing those from previous research articles.

The similarity measurement is the average of the values of similar pixels between the provided left ventricle segmentation and the left ventricle segmentation obtained by filling the holes in the myocardial segmentation: $\frac{1}{n}\sum_{i=1}^{n}\frac{|A_i \cap B_i|}{|A_i|}$, with $|A_i| = |B_i|$.

```
In [ ]: scoreED, scoreES = [], []

        for k in range (0,XTrain.shape[0]):
            dataED, dataES = XTrainImg[1,k].get_fdata(), XTrainImg[3,k].get_fdata() # get data
            segED, segES = mask_contour(dataED == 2), mask_contour(dataES == 2) # segmentation
            scoreED.append(np.mean(segED == (dataED == 3))) # give the difference between the pr
            scoreES.append(np.mean(segES == (dataES == 3))) # give the difference between the pr

        print('Percentage of accuracy on segmenation for ED : ', np.mean(scoreED))
        print('Percentage of accuracy on segmenation for ES : ', np.mean(scoreES))
```

```
Percentage of accuracy on segmenation for ED :  0.9999631441732135
Percentage of accuracy on segmenation for ES :  0.9999400653276702
```

III. Feature extraction

To implement features, I followed a strategy of implementing as many features as possible, based on different articles, and then selecting the most effective ones. The features included patient information (weight, height), volume measurements, circularity, thickness, and circumference of the heart and ventricles, as well as volume ratios between different organs. In addition, dynamic features were used, such as volume statistics and ejection fraction. I also segmented the left ventricle of the test set. Since this corresponds almost to the inside of the myocardium, I simply filled layer by layer the inside of the myocardium with label 3 to obtain segmentation. The heart of the problem is to have enough features to classify effectively but not too many to avoid overfitting.

I followed the strategy from [2] by having Instant features and Dynamic volume features. Instant features are features calculated on a single image, and Dynamic volume features are features calculated on images at two different times. Therefore, I implemented the following features:

Instant features:

- Volume of the myocardium, left ventricle and right ventricle
- Thickness of the myocardium (mean, standard deviation, max, min)
- Circularity and circumference of the myocardium (mean, max, min)
- Ratios of the volume of the right ventricle / left ventricle and the myocardium / left ventricle

Dynamic volume features:

- Ejection fraction of the left ventricle and right ventricle
- Median, standard deviation, kurtosis, and skewness of the myocardium volume between the two times

1. Instant features

a. Compute the Volume of the myocardium, the left ventricle, and the right ventricle

To compute the volume of the left ventricle, right ventricle, and myocardium, I drew a function based on the image segmentation: we summed the number of pixels of each label and multiplied by the volume of a pixel: $V = V_{pixel} \sum_{i=1}^{n}(f_i = y_i)$

b. Compute the thickness, circularity, and circumference of the myocardium

To compute the thickness, circularity, and circumference of the myocardium, I drew three functions based on the image segmentation: we computed the thickness by counting the number of pixels between the myocardium and the background, the circularity by computing the ratio between the area and the perimeter of the myocardium, and the circumference by computing the perimeter of the myocardium.

For the thickness, I used the distanceTransform function from CV2 to compute the distance between the myocardium and the background. I multiplied the result by the volume of a pixel to obtain the real distance. The thickness is the difference between the maximum and the minimum distance :
$Thickness = V_{pixel}(\max(distanceTransform) - \min(distanceTransform))$.

For the circularity, I use the area and the perimeter of the myocardium computed by the function findContours from CV2. The circularity is the ratio between the area and the perimeter :
$Circularity = \frac{Area}{Perimeter}$.

For the circumference, I use the perimeter of the myocardium computed by the function findContours from CV2. The circumference corresponds to the perimeter.

c. Compute the ratios

To compute the ratios, I simply divide the volumes of the volume of the myocardium and the right ventricle by the volume of the left ventricle :
$Ratio_{RV/LV} = \frac{V_{RV}}{V_{LV}}$, $Ratio_{Myo/LV} = \frac{V_{Myo}}{V_{LV}}$

2. Dynamic volume features

a. Compute the ejection fraction

To compute the ejection fraction, I use the volumes of the left ventricle at the two instants. The ejection fraction is the ratio between the difference of the volumes and the volume at the first instant :
$EF_{LV} = \frac{V_{LV_{ED}} - V_{LV_{ES}}}{V_{LV_{ED}}}$.

Since the volumes are computed in the same way as the volumes of the left ventricle, the ejection fraction of the left ventricle is computed in the same way.

b. Compute the volume statistics

To compute the volume statistics, I use the volumes of the myocardium at the two instants. I compute the median, the standard deviation, the kurtosis and the skewness of the volumes of the myocardium between the two instants. I use the function median, std, kurtosis and skew from numpy to compute these statistics.
The kurtosis and the skewness, corresponding to the fourth and the third moment, are computed with the following formulas :
$Kurtosis = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i-\bar{x})^4}{\sigma^4}$, $Skewness = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i-\bar{x})^3}{\sigma^3}$

IV. Classification

1. First approach: Multi-layer perceptron

In the article [2], it's explain that they suceed to classify the patients with a multi-layer perceptron, whith 93% accuracy. Therefore, I tried to implement a multi-layer perceptron to classify the patients. I used the MLPClassifier from sklearn. I used the parameters of the MLPClassifier given in the paper : `An ensemble of 50 multilayer perceptrons (MLP) [...]. The MLP’s architecture consists of four hidden layers, each containing 32 units, followed by batch normalization, leaky ReLU nonlinearity and a Gaussian noise layer (σ=0.1). Each MLP was trained on a random subset of 75% of the training data, while the remaining 25% were used for epoch selection. Further regularization was provoked by only presenting a random subset of 2/3 of the features to each MLP. We trained all MLPs for 400 epochs (with a patience of 40 epochs) [...] with an initial learning rate of 5·10−4, decayed by 0.97 per epoch. An epoch was defined as a set of 50 batches containing 20 patients each. During testing, the softmax outputs of all MLPs were averaged to obtain an overall MLP score`. I used the features described in the previous section, and also try subsets of the features following the advice of the different articles. I used the train set to train the model and the validation set to validate the model, putting the submission on kaggle to see the efficiency. I never obtained a score higher than 0.5 on the validation set. I tried to change the parameters of the MLPClassifier but I didn't obtain better results.If I have different results than the paper, it's maybe because I didn't use exactly the same features (they use extra metaData like mass), or because I didn't use the same preprocessing (for exemple if they use a normalization of the data). Therefore, I decided to try another approach.

2. Second approach: Random Forest

a. Method

In the articles [2], [3] and [4] they used a random forest to classify the patients. Therefore, I tried to implement a random forest to classify the patients. I used the RandomForestClassifier from sklearn. I used the default parameters of the RandomForestClassifier. One thing I saw was that besides the features and the number of trees, the other parameters didn't change the score. I tried many GridSearchCV to find the best parameters, but I didn't obtain better results. For the number of tree, 100 seems to be a good number if you want fast but reliable results : the score is stable most of time and the computation time is not too long. If you want to have more stable results without overfitting, 1000 seems to be a good number : both resuls on train and test are goods and stable.

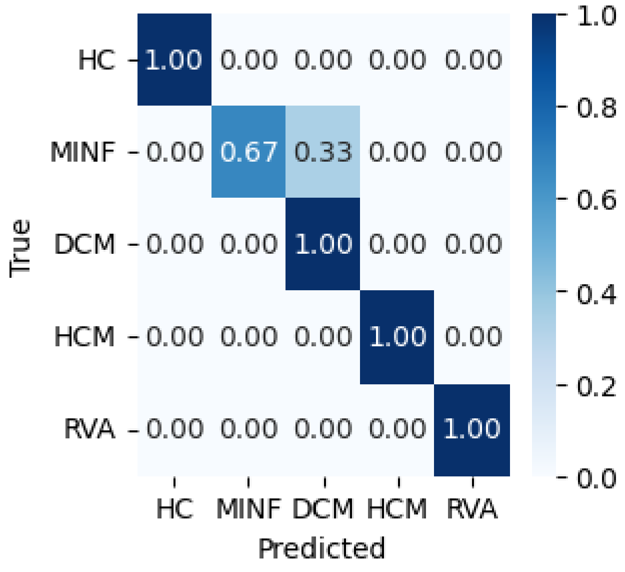
b. Features

The keypoint for the random forest is the features. First, I tried the whole set of features described in the previous section. I obtained a score of 0.73 and 0.8 on the validation set on kaggle, depending on the number of trees and random state. Then, I tried to limit the number of features, to avoid overfitting. I follow the features used in the article [2], given in a table in the article. I try to modify a little bit the subset, looking first the results with a train/test split and crossvalidation, and then the results on kaggle. I obtained the best results with the following features :

- right ventricle and left ventricle volume at end-diastole and end-systole
- myocardium volume at end-diastole
- mean and maximum thickness of the myocardium at end-diastole and end-systole
- ratio between the right ventricle volume and the left ventricle volume at end-diastole and end-systole
- ratio between the myocardium volume and the left ventricle volume at end-diastole and end-systole
- ejection fraction of the right ventricle

c. Limitations

In articles [2] and [3], they talk about missclassification between the patients MINF and DCM. They say that the random forest can missclassify the patients MINF and DCM, because they have similar features. I have done a train/test split (80%/20%), changing the random state and looking the confusion matrix. For the cases where there are missclassification, the confusion matrix was like the following :



3. Improve Random Forest with Gradient Boosting

a. Method

I will try to improve the results of the random forest with a gradient boosting algorithm. The goal is to classify the patients MINF and DCM. After random Forest, we apply a gradient boosting algorithm to classify the patients that were classified as MINF or DCM. I used the GradientBoostingClassifier from sklearn. I used the default parameters of the GradientBoostingClassifier.

b. Features

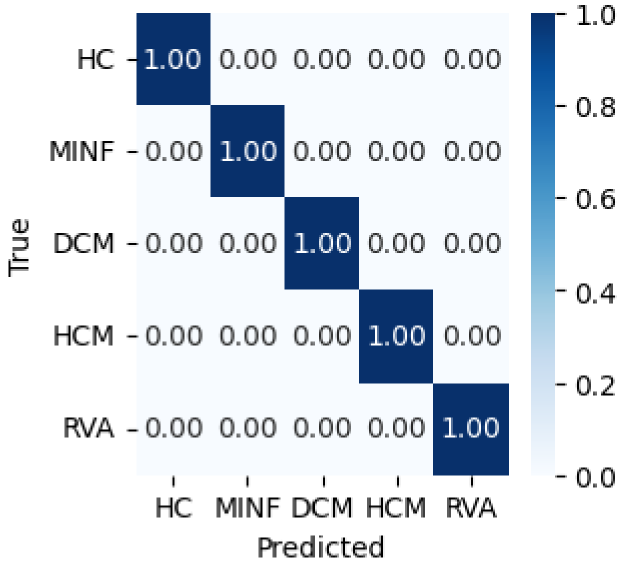
For the features, I learned that `The myocardial volume is relatively small compared with the LV volume in patients with DCM,` [3], so I decided to use the volumes of the myocardium, the right and the left ventricle at end-diastole and end-systole, and the ratio between the myocardium volume and the left ventricle volume at end-diastole and end-systole. I also add the right ventricle ejection fraction, and let some features of the random forest that I think are important to not change too much the classification that was done by the random forest.

Finally, the features are :

- myocardium, right ventricle and left ventricle volume at end-diastole and end-systole
- ratio between the myocardium volume and the left ventricle volume at end-diastole and end-systole
- ratio between the right ventricle volume and the left ventricle volume at end-diastole and end-systole
- ejection fraction of the right ventricle and left ventricle
- mean thickness of the myocardium at end-systole

c. Results

The confusion matrix after the gradient boosting algorithm as a second classification shows that the gradient boosting algorithm is able to classify the patients MINF and DCM. I obtained a score of 0.93 on the validation set on kaggle.



V. Algorithm and Results analysis

1. Results

It's a little bit difficult to analyse the results because we don't have the labels of the test set, and we succeed to have 100% on the train set. We can only analyse the results on the validation set on kaggle. I obtained 93% on the public score and 100% on the private, which seems to be a very good score (close to the score of the best articles). However, the results given by the Gradient Boosting Algorithm are not as good as the results given by the Random Forest. Indeed, they depend a lot on the random state. For the different try I have done on the validation set on kaggle, it can well classify the patients MINF / DCM, but it can also missclassify and give a score lower than without, depending on the random state. Therefore, I think that the results are not as good as they seem to be for this part of the project. For Random Forest, the results are stable, I think that the features are well chosen and that the algorithm is well implemented.

2. Feature importance

To understand the results of the Random Forest, I tried to analyse the importance of the features, which can give us some information about the classification, and the way we can detect the different diseases.To analyse the importance of the features, I used the function feature_importances_ from the RandomForestClassifier. I computed the mean of the feature importance for the different random state. I obtained the following results :

```
Feature ranking for the Random Forest Model:

14% circum_ED_max
13% thick_ED_min
12% thick_ED_mean
10% circum_ED_min
 7% rv_ED
 6% thick_ED_max
 6% r_rv_lv_ED
 6% myo_ED
 6% lv_ED
 5% circum_ED_mean
 5% thick_ED_std
 4% Weight
 3% circul_ED
 2% Height
```

We saw that the maximal and minimal circumference and the mean and the minimal thickness during the End Diastole are goods features to classify the patients. It's a little bit surprising because the thickness is not a feature used in all articles, and at the start of my search I have done a model without that give 80% of accuracy. The dataset is not very big, and the features are correlated, so it's difficult to know if the features are really important or if it is too linked to the dataset.

VI. References

[1] H. -Y. Lee, N. C. F. Codella, M. D. Cham, J. W. Weinsaft and Y. Wang, "Automatic Left Ventricle Segmentation Using Iterative Thresholding and an Active Contour Model With Adaptation on Short-Axis Cardiac MRI," in IEEE Transactions on Biomedical Engineering, vol. 57, no. 4, pp. 905-913, April 2010, doi: 10.1109/TBME.2009.2014545.

[2] Isensee, F., Jaeger, P.F., Full, P.M., Wolf, I., Engelhardt, S., Maier-Hein, K.H. (2018). Automatic Cardiac Disease Assessment on cine-MRI via Time-Series Segmentation and Domain Specific Features. In: , et al. Statistical Atlases and Computational Models of the Heart. ACDC and MMWHS Challenges. STACOM 2017. Lecture Notes in Computer Science(), vol 10663. Springer, Cham. https://doi.org/10.1007/978-3-319-75541-0_13

[3] Wolterink, J.M., Leiner, T., Viergever, M.A., Išgum, I. (2018). Automatic Segmentation and Disease Classification Using Cardiac Cine MR Images. In: , et al. Statistical Atlases and Computational Models of the Heart. ACDC and MMWHS Challenges. STACOM 2017. Lecture Notes in Computer Science(), vol 10663. Springer, Cham. https://doi.org/10.1007/978-3-319-75541-0_11

[4] Khened, M., Alex, V., Krishnamurthi, G. (2018). Densely Connected Fully Convolutional Network for Short-Axis Cardiac Cine MR Image Segmentation and Heart Diagnosis Using Random Forest. In: , et al. Statistical Atlases and Computational Models of the Heart. ACDC and MMWHS Challenges. STACOM 2017. Lecture Notes in Computer Science(), vol 10663. Springer, Cham. https://doi.org/10.1007/978-3-319-75541-0_15