IMA205, 2023 Kaggle challenge : cardiac MRI classification

MANON HEFFERNAN, Institut Polytechnique de Paris, France

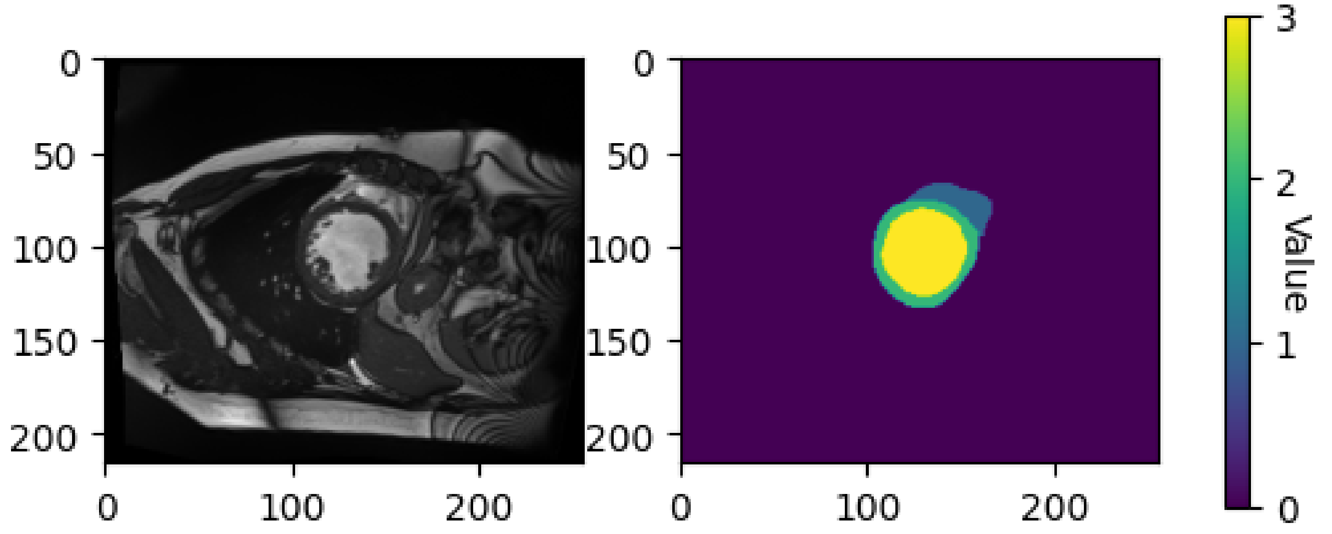


Fig. 1. MRI from the subject 001 from the training set, along with the end-diastolic segmentation

This report presents my work for the 2023 Kaggle challenge, as part of the IMA205 class. My submission is under the pseudonym **Coléoptère facétieux**.

1 CHALLENGE PRESENTATION

The goal of this challenge is the classification of cardiac MRI into 5 classes :

(1) Healthy controls
(2) Myocardial infarction
(3) Dilated cardiomyopathy
(4) Hypertrophic cardiomyopathy
(5) Abnormal right ventricle

Students could use any classification algorithm and apply all pre and postprocessing methods that may be deemed relevant. Students were also in charge of extracting all the necessary features.

2 PREPROCESSING : LEFT VENTRICLE SEGMENTATION

All subjects in the training set were provided with full end-systolic and end-diastolic segmentations. The segmented parts were the myocardium, the left ventricle, the right ventricle and the background.

The test set segmentations were missing the left ventricle segmentation. The latter being essential to feature extraction, it was necessary to pre-process the test set in order to properly extract the left ventricle and get full segmentations.

The articles linked in the resource section of the Kaggle challenge described the methods used by the different researchers to segment the different regions of interest (ROI), mostly using machine learning methods.

The main point to take out of those articles was that the authors only had the MRI images available, whereas for this challenge, we were already provided nearly complete segmentations. And since the myocardium surrounds the left ventricle, I believe it sufficient to consider the left ventricle to be the zone enclosed by the myocardium.

To segment the left ventricle, I created a mask for the myocardium. I then used the *binary_fill_holes* function from the *scipy.ndimage* library which returns the indices of the filled binary elements extracted from the mask : so both the myocardium and the left ventricle in this case. I used this information to only take into account the part that wasn't already labeled as the myocardium, and label it as the left ventricle.

This solution was potentially meant to be changed, if the results obtained weren't satisfying enough. This choice of segmentation turned out to be sufficient, and a lot less complex, both on a conceptual level than on an implementation and computational level. Therefore, I did not feel the need to implement machine learning methods to segment the left ventricle.

3 FEATURE EXTRACTION

The feature extraction is one of the main challenges of this classification exercice. The goal is to choose and properly compute relevant and meaningful features in order to have the proper tools to distinguish between the different classes and achieve good results.

This is the part that I actually started working on, as it also helps to get familiar with the data format and get an overview of the type and values of computed features.

Author's address: Manon Heffernan, Institut Polytechnique de Paris, Palaiseau, France.

The challenge description on Kaggle included articles which described the types of features used in order to classify. After roaming through them I figured those used in [1] were pretty common and constituted a good starting point. Those features are the features (2) to (15) in the following list. I've decided to add the myocardium thickness, I will detail the reasons in the following subsection.

The final features I used are the following :

(1) myocardium thickness,
(2) EDV(myocardium),
(3) ESV (myocardium),
(4) EDV (left ventricle),
(5) ESV (left ventricle),
(6) EDV (right ventricle),
(7) ESV (right ventricle),
(8) Ejection fraction (left ventricle),
(9) Ejection fraction (right ventricle),
(10) ratio between EDV left/EDV right,
(11) ratio between ESV left/ESV right,
(12) ratio between EDV myocardium/EDV left,
(13) ratio between ESV myocardium/ESV left,
(14) weight,
(15) height.

where EDV stand for End-Diastolic Volume and ESV for End-Systolic Volume.

3.1 myocardium thickness

The myocardium thickness appeared to be a relevant feature, since its value varies depending on the diagnosis. An hypertrophic cardiomyopathy is linked to a thickening of the myocardium, while a dilated cardiomyopathy is linked to a progressive thinning of the cardiac walls, starting with the left ventricle and eventually impacting the right ventricle wall too. Article [2] indicates an average end-diastolic thickness of 8.6±2.1 mm in healthy subjects, while of 15.8±4.2 mm in patients with hypertrophic cardiomyopathy.

To compute the thickness, I computed the myocardium mask for each of the end-diastolic slices of the segmentation. Then on each of them, I computed the distance map, which basically finds for all non-zero values the closest zero-value. Then I chose for each slice, the maximum value, and for the final thickness the maximum on all the slices. It is noteworthy that the value (in millimeters) returned by my function actually corresponds to half the real thickness. Indeed, the maximum value on a slice would be reached by the points on the mid-line of the myocardium, and this value only corresponds to half the thickness. Therefore, it would be necessary to multiply it by 2 to interpret it medically. I only realised this fact when I read through my code after the challenge submission deadline, while writing this report. Had I realised sooner I would've adjusted that in the function in order to use the "real" measures as features, though I have doubts it would change much, since the features are normalised later on for the classification.

3.2 end-systolic and end-diastolic volumes

The diastole is the relaxed phase in the cardiac cycle, and the systole is the contraction phase. The volumes of the different regions of interest at the end of those specific phases are important features in the disease diagnosis. The same way the myocardium thickens or thins depending on the situation, the ventricles' volumes can change significantly depending on the diagnosis.
The reference values I got from Wikipedia [3] are the following :
End-diastolic volumes :

- left ventricle : 142 ml (± 21 ml)
- right ventricle : 144 ml (± 23 ml)

End-systolic volumes:

- left ventricle : 47 ml (± 10 ml)
- right ventricle : 50 ml (± 14 ml)

From a diagnosis perspective, those values can help indicate the stage of the disease. For example, [4] indicates that in the case of dilated cardiomyopathy, both ventricular volumes increase, but that the left ventricle is the first one to be impacted. Therefore, a patient with an enlarged left ventricle but within range right ventricle could be at an earlier stage of the disease than someone with both ventricles enlarged. The myocardial volumes can indicate thinning or thickening, interpreted conjointly with other features.

To compute those values, I needed two pieces of information : the number of pixels corresponding to each volume, and the corresponding volume of one pixel (voxel).

Initially, I had only counted the pixels but when I saw the very high values output and the fact that I couldn't make any sense of them from a medical point of view, I thought it would be better to render the features in an intelligible way, meaning in ml. To achieve that, I needed to find the value of one voxel. It turns out .nib images have that information in the header, so I was able to extract that value and use it to compute the final volumes. That's when I realised that only counting the pixels might have not been a good idea, since the voxel volume might not be the same for all the images. I did not check if it was the case or not but as a matter of precaution and intelligibility, I used the voxel information systematically (the latter point doesn't make much difference for the automated classification, I believe).

3.3 Ejection fraction

The ejection fraction corresponds to the ratio between the stroke volume and the end-diastolic volume. The stroke volume is the difference between the end-diastolic volume and end-systolic volume. That is :

$$EF = \frac{(EDV - ESV)}{EDV}$$

The reference values [5] are the following :

- left ventricle : 0.66 (± 6%)
- right ventricle : 0.67 (± 4.6%)

A low ratio is associated with heart failure, diseases like myocardial infarction or hypertrophic cardiomyopathy.

The left ejection fraction is usually the one which is the most computed, as the left ventricle is the one that assumes most of the pumping (the right ventricle is in charge of pumping blood to the lungs only). But since one of the classes we are aiming for is "abnormal right ventricle", it is important to include features characterising this ROI, especially since it sounds like a rather broad term and could include issues in contraction, or abnormal volumes (hence including the end-systolic and end-diastolic volumes, among other features).

This feature is very straightforward to compute and only requires the EDV and ESV to have been previously computed.

3.4 Ratios of diastolic and systolic volumes of ROI

The article [1] mentions using the ratios between the end-diastolic and end-systolic volumes between the left and right ventricles, and the myocardium and the left ventricle.

The right ventricle/left ventricle volume ratios are relevant to characterize the relative volume of the ventricles. A high ratio could indicate an enlarged right ventricle or a smaller left ventricle volume, both being worth further investigating. In the computation, I mistakenly used the inverse (left/right instead of right/left). As for the myocardium thickness, had I realised sooner, I would've corrected that.

The myocardium/left ventricle volume ratios help diagnose enlarged cardiomyopathy or dilated cardiomyopathy among others.

For example, if the ratio is elevated during diastole (when the heart is relaxed and filling with blood), it may indicate that the myocardium has become thickened or stiff, which can impair the heart's ability to fill with blood properly. On the other hand, if the ratio is elevated during systole (when the heart is contracting to pump blood), it may suggest that the heart is working harder than normal to pump blood, which can be a sign of heart failure or other conditions that affect cardiac function.

Here again, the features are straightforward to compute, once end-diastolic and end-systolic volumes have been computed.

3.5 Patient features

The height and weight of the patients were also used as features. In themselves, they don't say much about the cardiac condition of a person. But it can influence the diagnosis, as the exact same features may not mean quite the same when the subject is 1.40m high versus 2m high, same when weighing 35kg versus 95kg.

3.6 Computation and saving the features

In order to be more efficient, I organised the feature computation such that those features were saved in a file, so that it may be possible to reuse the features instead of recomputing them, or trying with different batches of features without having multiple variables. This is also useful to have an overall view of the values.

4 METHODS

Once the features extracted, the next goal is to classify the subjects. I have built a multi-layer perceptron, a random forest classifier and a k-nearest neighbours classifier out of curiosity.

4.1 Multi-layer perceptron (MLP)

This challenge helped me become more comfortable with machine learning architectures. Still, the process was quite long and therefore the architecture of the final multi-layer perceptron is simpler than the one described in the article it's inspired from [7].

I have kept the layer size of 32 indicated by [7]. Other than that, I have stacked a few basic relu functions and finished with a softmax function, more adapted to multi-class classification problems. The whole idea was to try reusing the structures we had been presented in the labs to try to get somewhere. I added a bit of noise to the features to try to get a more robust network.

I used the Adam optimizer, because I had used it previously in class and because it is used in [7]. The professor told us in class that it is one of the most common and most used optimizers, so I figured I couldn't really go wrong by using it. I used the cross entropy for the loss function, it being very classical in multi-class classification problems.

I created two modes in my program, linked to the train-validation Boolean. If True, then I separate the initial training set in a training-validation set. I implemented it to be able to track the loss and accuracy of the networks over the epochs, to be able to adapt to the result, for example by reducing the number of epochs for the training part to avoid over-fitting. Once the parameters tuned, I set it to False and trained the network on the full 100-subject training set.

Initially, I had put a high number of epochs, around 600, just to be able to see what the loss and accuracy functions would look like. Then I iteratively narrowed down the number to eventually reach 35, by trying to choose the number at which the accuracy reaches its plateau. I also had to put a low learning rate of 0.001, otherwise the network reached an accuracy of 0.98 (or more) very fast and I couldn't limit the over-fitting.

I tried a few variations of the final network that I have implemented, by increasing the size of the layers (from 32 to 64), by adding an extra linear layer. Those didn't seem to improve the network, the accuracy being the same regardless, so I decided to go for the smaller and simpler network.

4.2 Random Forest and k-Nearest Neighbours

A few articles from the challenge ressource page [1],[6],[7] mentioned using a random forest classifier, on its own or with other classification tools (MLP for example).

Since we had implemented a Random Forest Classification algorithm in a lab, I had initially started with this approach and reused the architecture from the lab. I was expecting good results, since it seemed to be giving good results in the articles and with a simple implementation : article [1] basically described using 1000 trees to run the classification, and that pretty much was it.

But I got really bad results with this method (0.2 public accuracy, 0.09 on the full test set after the deadline) and I couldn't make sense of why so I eventually turned to the MLP, which ended up giving the results I used for the final submissions.

I only figured out the issue when working on the MLP (which didn't give much better results, maximum of 0.4 of accuracy, 0.28 on the whole test set after the deadline). In fact, to build the features and label list, I was iterating over the folders in the Train folder, and I had a *for* loop iterating through the metaDataTrain.csv file and adding the weight and height to the features, but also building the label list on the side. The thing is, this approach implicitely supposed that the files were being read in numerical order (001, 002,...099), which isn't the case. The *os.path.join* function creates a list with all the folders but they are probably added based on the inode or something, which have no reason to be orderly matching the names of the folders. So all that time, I had been trying to classify the features, but with the wrong labels attached... After fixing this issue, the accuracy with MLP went from 0.4 to 0.6 then 0.86, which is a lot better.

In the end, the accuracy with Random Forest was of 0.8 on the 30% test set on Kaggle. I was pleased but still a bit surprised that such a simple architecture could achieve good results like that.

Out of curiosity, I wanted to know just how easy it would be to cluster the data. Since it's hard to visualise the data (given the dimension of the feature space), I resorted to using knn to at least get an idea of how clusterable the data is. I proceeded to implement an elementary parameter optimisation of knn with *GridSearchCV*, as seen in the labs, and run a k-nearest neighbours classification with the best parameters. I really didn't know what to expect, and I was quite surprised to get an accuracy as good as with Random Forest. I compared the classifications output on the test set by those two methods and had around a 20% difference in the values.

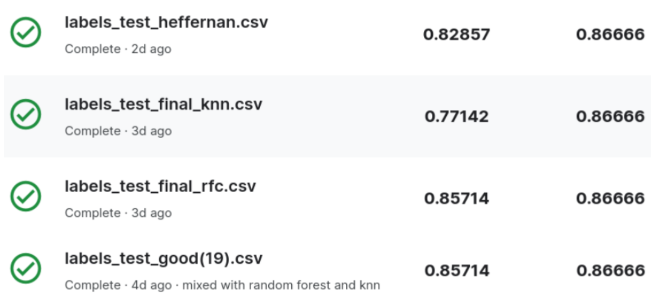
5 RESULTS

In this section, I will only discuss the results I got once I solved the mislabeling that occurred when building the features list.

The majority of the results I submitted were the results of small tweaks to the main structure : removing a layer, increasing the layer size to 64 instead of 32, changing the learning rate, the noise level... Too much noise decreased the training accuracy so I stuck to a small value, and too many epochs resulted in the network over-fitting the training data. I decided to go for the smaller and simpler version of the network, since increasing the number of layers or their size didn't seem to impact the results significantly enough to justify the complexity increase.

I also tried mixing up the results from the MLP, Random Forest and KNN : I trained two MLPs, ran 2 Random Forest classifications and 1 knn. I then proceeded to choose for each subject the class that was the most present, out of all 5 possible classifications. The results output by the MLP were so similar (if not identical) that I didn't push the test any further, same for the different results for the Random Forest. Though, it is indicated in [7] that they trained many MLP, so perhaps training a few dozens MLP as in the article, and taking the most likely class for each subject could improve the accuracy and robustness of the final result.

Also, it was difficult to evaluate the improvement, since the accuracy given by Kaggle was always of 0.86. Now that the challenge is over, I can see that the results output by Random Forest were the among the best ones, even better than some of the results output by the MLP. Indeed, the best accuracy I have reached on the private leaderboard is 0.85714, which has been reached by classifications obtained with Random Forest, MLP and also one of the mix between all three methods, as I've described just above. This means that the mistakes were not on the 30% the labels were being tested on, but elsewhere (and the mistakes on those 30% were identical or there were as many on each try).



✓ labels_test_heffernan.csv Complete · 2d ago	0.82857	0.86666
✓ labels_test_final_knn.csv Complete · 3d ago	0.77142	0.86666
✓ labels_test_final_rfc.csv Complete · 3d ago	0.85714	0.86666
✓ labels_test_good(19).csv Complete · 4d ago · mixed with random forest and knn	0.85714	0.86666

Fig. 2. Public and private results on MLP, KNN, RFC and mix results

REFERENCES

[1] 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

[2] Dong SJ, MacGregor JH, Crawley AP, McVeigh E, Belenkie I, Smith ER, Tyberg JV, Beyar R. Left ventricular wall thickness and regional systolic function in patients with hypertrophic cardiomyopathy. A three-dimensional tagged magnetic resonance imaging study. Circulation. 1994 Sep;90(3):1200-9. doi: 10.1161/01.cir.90.3.1200. PMID: 8087929; PMCID: PMC2396316. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2396316/

[3] Wikipedia contributors. "End-systolic volume." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 28 Nov. 2022. https://en.wikipedia.org/w/index.php?title=End-systolic_volume&oldid=1124405560://

[4] American Heart Association's official website. https://www.heart.org/en/health-topics/cardiomyopathy/what-is-cardiomyopathy-in-adults/dilated-cardiomyopathy-dcm

[5] Wikipedia contributors. "Ejection fraction." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 18 Apr. 2023. https://en.wikipedia.org/wiki/Ejection_fraction

[6] 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

[7] 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