Report Challenge 2025

Luciana Munhos
IM05 Challenge 2025
Loic Le Folgoc, Pietro Gori

May 2025

Abstract

This project addresses the classification of cardiac conditions from cardiac magnetic resonance imaging (CMRI) data, focusing on five diagnostic classes: healthy controls, myocardial infarction, dilated cardiomyopathy, hypertrophic cardiomyopathy and abnormal right ventricle. The approach involves extracting key anatomical features from the CMRI images, followed by the application of a Random Forest classifier. Hyperparameter optimization was performed using cross-validation and feature importance was leveraged to select the most relevant features. The model achieved a classification accuracy of 88% across the full dataset, with initial experiments showing higher accuracy (1.00) in a smaller subset of test cases. Additionally, an optional left ventricle segmentation was implemented. The segmentation was inspired by the methodology outlined in *Automatic Left Ventricle Segmentation Using Iterative Thresholding and an Active Contour Model With Adaptation on Short-Axis Cardiac MRI* [1]. This work demonstrates the potential of automated methods for supporting early detection of cardiac pathologies, which can be critical for timely diagnosis and intervention in clinical settings.

Contents

1 Introduction

This challenge focuses on the automated classification of cardiac conditions using cardiac magnetic resonance imaging (CMRI). The dataset consists of MRI scans from 150 subjects, provided in NIfTI (.nii) format, with each subject labeled into one of five diagnostic categories:

- Label '0' - Healthy controls
- Label '1' - Myocardial infarction
- Label '2' - Dilated cardiomyopathy
- Label '3' - Hypertrophic cardiomyopathy
- Label '4' - Abnormal right ventricle

Initially, we are provided for each image in the training-validation set with a corresponding segmentation of the cardiac anatomy. Each segmentation map consists in a 3D multi-label mask, where each label stands for the following structures:

- 0 - Background
- 1 - Right ventricle cavity
- 2 - Myocardium
- 3 - Left ventricle cavity

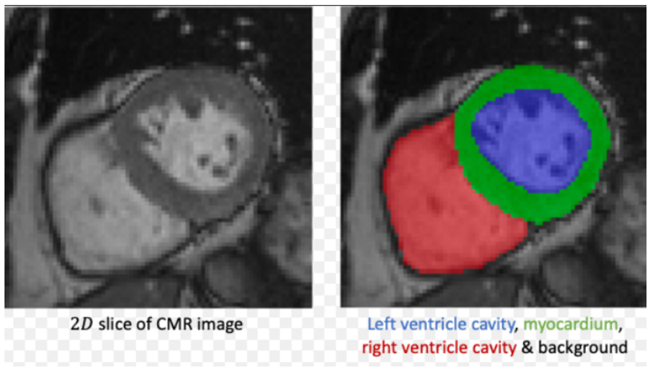
2D slice of CMR image
Left ventricle cavity, myocardium,
right ventricle cavity & background

Figure 1: **Example of cardiac segmentation showing the different structures in the MRI. The labels represent the background, right ventricle cavity, myocardium and left ventricle cavity.**

Handling this dataset presented the challenge of processing complex 3D MRI images and extracting relevant anatomical features critical for accurate classification. The task involved understanding how to work with NIfTI image formats, developing techniques for feature extraction and calculating specific cardiac metrics such as volume considering voxels, myocardial thickness and ejection fraction.

The objective of the challenge was to develop a machine learning model capable of classifying the cardiac images into the predefined categories based on these features. Additionally, a segmentation of the left ventricle was implemented to further refine the analysis.

This report will explain the feature selection process, detailing which features were chosen and how the final set was selected. It will also describe the different classification tests performed, including the use of Random Forest, XGBoost, PCA and cross-validation. Finally, the methodology for the left ventricle segmentation will be presented.

2 Related Work

This project builds upon three works that have explored the classification and segmentation of cardiac cine-MRI for disease diagnosis.

For feature extraction and classification, two articles from the ACDC challenge proceedings were particularly influential. In *Automatic Cardiac Disease Assessment on cine-MRI via Time-Series Segmentation and Domain Specific Features* [2], the authors propose a pipeline that relies on domain-specific features—such as volumes of the left and right ventricles, ejection fraction and myocardial wall thickness—extracted across the cardiac cycle. These informed the choice to compute physiological features from the available annotations and base the classification approach on them.

The second article, *Densely Connected Fully Convolutional Network for Short-Axis Cardiac Cine MR Image Segmentation and Heart Diagnosis Using Random Forest* [3], combines a CNN-based segmentation with a Random Forest classifier for diagnosis. While deep segmentation was not implemented in this work, the classification strategy using Random Forest, feature importance analysis and cross-validation served as reference points in building and evaluating these models.

For the segmentation part, I followed the method proposed in [1], where the authors use a combination of iterative thresholding and an active contour model with adaptation to segment the left ventricle from short-axis cine-MRI. This classical yet effective method does not rely on deep learning, making it a suitable choice for this project. It provided a practical and interpretable way to segment the LV by leveraging anatomical structure and image intensity, particularly in the end-diastolic and end-systolic frames.

3 Methods

3.1 Data Access and Execution Environment

To ensure reproducibility and reduce execution time, the dataset (Train and Test) was uploaded to a public Google Drive folder. This folder includes both the original `.nii` images and the segmentation results. Since generating all segmentation masks can take over 1.5 hours for the full test set, pre-computed segmentation masks were saved as `.rii` files.

Two segmentation versions were used:

- **Simple segmentation**: Fills the myocardium region and labels its interior as the left ventricle (class 3). Results are saved as `_simple_seg.rii` and can be regenerated locally in approximately 30 seconds.
- **Complex segmentation**: Produces another possible segmentation and is saved as `_complete_seg.rii`.

The notebook is designed to run either locally or in Google Colab. If running locally without downloading all precomputed masks, the Dataset folder includes enough partial segmentations to reproduce the simpler version. All required `.py` files and CSVs are stored in the same directory as the notebook to ensure smooth execution. Inside the notebook, there is a function called "plot_image_and_segmentation" which plots the image and the selected segmentation.

Initially, all development was done locally. However, due to a hardware failure—my SSD was no longer recognized by the BIOS and seems to be severely damaged—I had to migrate the entire workflow to Google Colab. I first adapted the code for personal use and then made the necessary adjustments to ensure it could be executed by anyone. Unfortunately, this unexpected issue with my computer caused a delay and I was unable to submit my predictions with the complex segmentation to Kaggle as planned.

3.2 Data Preprocessing

The original dataset did not provide left ventricle (LV) segmentations for the test set, which are essential for feature extraction and classification tasks. To solve this, a preprocessing step was implemented to generate basic LV masks based on the myocardium contours (label 2) already present in the segmentation images.

To create the LV segmentations, I started with the myocardium mask for each slice (already provided). First, I took the myocardium and filled the entire interior with label 3, representing the left ventricle (LV). Then, I removed the myocardium by changing the areas that had label 2 to label 0, effectively excluding the myocardium from the mask. The final result was a mask that only represented the LV, without any overlapping or conflicting labels.

This method was applied to each slice of all the test images, generating a simple LV segmentation mask for the entire volume. Since the myocardium segmentation is considered accurate and the LV is known to reside within it, this simple LV segmentation acts as a reliable ground truth for the LV segmentation. The segmentation was performed slice by slice and the final result are 3D masks representing the LVs. They are saved as files with the suffix `_simple_seg.nii` and they were used for the majority of the predictions, including the Kaggle submission. This preprocessing step was critical for enabling feature extraction from the images, providing a consistent and reliable method for identifying the LV.

Later, another LV segmentation method was implemented and tested to further refine the LV boundaries, based on the approach described in the literature. These segmentations were stored with the suffix `_complete_seg.nii`.

3.3 Feature extraction

In this project, a variety of cardiac features were extracted from the CMRI images to serve as input for the classification model. These features were derived based on the segmentations given (for the Train data) but also generated during the preprocessing step (for the Test data) and were selected based on their relevance to heart condition classification and their presence in the literature.

The following is a detailed description of how each of the features used in this project were calculated:

3.3.1 Ejection Fraction of Left and Right Ventricles

Abbreviations in the code: EF_LV, EF_RV

The ejection fraction (EF) of the left ventricle (LV) and right ventricle (RV) is calculated using the formula:

$$EF = \frac{V_{ED} - V_{ES}}{V_{ED}}$$

where V_{ED} is the end-diastolic volume (EDV) and V_{ES} is the end-systolic volume (ESV). This fraction is crucial for evaluating the function of both ventricles.

3.3.2 Left and Right Ventricles and Myocardial End-Diastolic Volume normalized by Body Surface Area

Abbreviations in the code: LV_EDV_BSA, RV_EDV_BSA, MYO_ED_BSA

The end-diastolic volume of the left ventricle (LV_EDV) and right ventricle (RV_EDV) is calculated and normalized by the body surface area (BSA). This normalization allows for comparison across individuals with different body sizes.

3.3.3 Myocardial Thickness mean and standard deviation

Abbreviations in the code: MYO_Thickness_Mean, MYO_Thickness_Std

The average myocardial thickness is calculated by measuring the distance between the inner and outer borders of the myocardium at each image slice. The mean thickness is computed by averaging these measurements across all slices.

The standard deviation of myocardial thickness is calculated to measure the variability in thickness across the myocardial region.

3.3.4 Thickness between Left Ventricle and Right Ventricle Cavities mean and standard deviation

Abbreviations in the code: LVM_Thickness_Between_LVC_RVC_Mean, LVM_Thickness_Between_LVC_RVC_Std

The average thickness between the left ventricle (LV) and right ventricle (RV) cavities is calculated by measuring the distance between their inner surfaces at each slice. The mean and standard deviation of this thickness are calculated to measure the overall thickness and its variability.

3.3.5 Maximum and Minimum Myocardial Thickness

Abbreviations in the code: MYO_Thickness_Max, MYO_Thickness_Min

The maximum and minimum myocardial thickness are calculated by selecting the highest and lowest thickness values, respectively, among all the measurements obtained at each slice.

3.3.6 End-Systolic Volume for Left Ventricle, Right Ventricle and Myocardium

Abbreviations in the code: LV_ESV, RV_ESV, MYO_ES

The end-systolic volume (ESV) for the left ventricle (LV), right ventricle (RV) and myocardium is calculated by summing the voxel volumes at the systolic phase. These values are important for assessing the heart's function at the end of systole.

3.3.7 Area at the Apex for Left and Right Ventricles

Abbreviations in the code: RVC_Apex_Area, LVC_Apex_Area

The area of the left ventricle (LV) and right ventricle (RV) at the apex is calculated by extracting the area from the transverse plane of the image at the apex region. These areas are critical for understanding the geometrical characteristics of the ventricles at the heart's apex.

3.3.8 Estimated Myocardial Mass

Abbreviation in the code: Mass

The myocardial mass is estimated by multiplying the myocardial volume by the density of myocardial tissue. This provides an estimate of the total muscle mass of the heart, which is important for evaluating myocardial enlargement (hypertrophy).

3.3.9 Ratio between RV and LV areas at the apex

Abbreviation in the code: RVC_LVC_Apex_Ratio

This feature is calculated as the ratio between the area of the right ventricle (RV) and the area of the left ventricle (LV) at the heart apex. To calculate it, the areas of both ventricles in that region are first extracted and then the ratio is computed:

$$RVC\_LVC\_Apex\_Ratio = \frac{Area_{RV\_Apex}}{Area_{LV\_Apex}}$$

3.3.10 Maximum Circumference of the Myocardium

Abbreviation in the code: MYO_Circ_Max

The maximum myocardial circumference is calculated by measuring the contour of the myocardium at each slice and selecting the maximum value from all the measurements.

3.3.11 Maximum Circumference of the RV

Abbreviation in the code: RV_Circ_Max

Similarly, the maximum circumference of the right ventricle (RV) is calculated by measuring the contour and selecting the maximum value among all slices.

3.3.12 Mean Circumference of the Myocardium

Abbreviation in the code: MYO_Circ_Mean

The mean myocardial circumference is calculated by averaging the contour measurements of the myocardium at each slice.

3.3.13 Mean Circumference of the RV

Abbreviation in the code: RV_Circ_Mean

The mean circumference of the right ventricle is calculated in the same way as for the myocardium, by averaging the contour measurements at each slice.

As I computed a large number of features, I reduced the feature set to avoid overfitting and to not consider redundant data. To achieve this, I performed a feature importance analysis based on a Random Forest Classifier. After training the model, I computed the feature importances and sorted them to identify which features contributed most to the classification. The most important ones, ranked by their importance, were selected for the classification.

Initially, I selected the top 14 most important features. However, I found that the ranking of features was unstable and some features were highly correlated with one another, leading to redundancy. To improve the model's stability and efficiency, I decided to remove these redundant features (for example I had a volume and the same volume divided by BSA as two different features).

Additionally, weight and height were initially included as features. However, after evaluating their importance, it became clear that they had little impact on the model's performance. Consequently, they were removed from the final feature set.

The final set of features used for classification included metrics such as LV volume, ejection fraction, myocardial thickness and others that were found to be stable and informative. These features were critical in classifying heart conditions, as they captured both anatomical and functional aspects of the heart. The function process_dataset is the most relevant one because it extracts the features. It reads metadata, loads segmentation files of the heart from given paths and computes the features. Additionally, it normalizes some values by body surface area (BSA) and computes BMI. The function also allows for optional deletion of certain features. After processing, it returns the metadata as a NumPy array.

3.4 Classification Algorithms

For this study, I chose Random Forest, XGBoost, AdaBoost and PCA combined with Cross-Validation to address the complexities of medical image classification. These algorithms were chosen for their ability to handle high-dimensional data effectively. Random Forest was the primary choice due to its strong performance and resilience against overfitting. It is widely acknowledged for its accuracy and reliability, often outperforming other algorithms in machine learning challenges.

XGBoost was tested for its efficiency in processing large datasets. I was interested in its boosting mechanism, which iteratively corrects errors from previous models and its regularization features, which help avoid overfitting—critical for maintaining generalization in medical imaging tasks.

AdaBoost was explored due to its method of combining weak classifiers to create a stronger one. This technique appealed because of its ability to improve classification accuracy, especially in noisy data, a common challenge in medical images.

I also experimented with PCA, which I applied to reduce dimensionality and focus on the most relevant features. However, it provided an accuracy of only 80%, significantly lower than the other algorithms, indicating that it was less effective for this particular classification task.

Cross-validation was used to fine-tune the models, ensuring the best possible performance by selecting optimal hyperparameters. The results of these classifications models will be presented in the section Results.

3.5 Evaluation Metrics

In this project, Categorization Accuracy was used as the primary metric to evaluate the performance of the classification models. This metric reflects the percentage of correctly classified instances out of all test instances, offering a clear and intuitive understanding of model performance. At first, stratified sampling was applied to split the data into training and testing sets, ensuring a balanced distribution of classes. However, since the dataset was already balanced, this step was not necessary.

Random Forest was evaluated using cross-validation to tune the model's hyperparameters effectively. This approach helped ensure a robust evaluation by assessing the model's performance across different subsets of the data, mitigating overfitting and providing a better estimate of its generalization to unseen data. Once the best parameters were calculated, the Random Forest model was trained and evaluated accordingly.

Categorization accuracy, in conjunction with other important metrics such as the confusion matrix, classification report and ROC AUC score, was used to evaluate and display the results. This enabled a deeper understanding of how well the model handled different classes and provided a detailed performance assessment.

Optional Segmentation of the Left Ventricle (LV)

The approach in the paper I considered for my segmentation is based on region growing with an initial seed and using thresholds derived from the mean and standard deviation of the LV intensity. To obtain the initial seed, the paper uses a Hough-based search method, then performs region growing, iterating until the segmentation is stable. The thresholds are calculated from the mean and standard deviation of the LV intensity, which helps separate the heart regions.

In my implementation, I followed this general idea but made key adjustments to address issues that appeared while executing and observing my resulting segmentations. First, I took the seed from the myocardium ground truth mask (myocardium mask) because the Hough method didn't give good results. In many cases, Hough localized the seed in the wrong places, like in the right ventricle (RV) instead of the desired left ventricle (LV) region and often produced incorrect circles. So, the myocardium mask information as the starting point helped a lot.

As step 2, there is the calculation of the mean and standard deviation of blood signal estimation. An edge-based region-growing technique is used, starting from the previous initial seed, to segment the LV region that is predominantly composed of blood. The region-growing process expands iteratively, adding neighboring pixels that are within a certain intensity threshold (defined by `max_diff`) of the current mean intensity. The process continues until no more pixels can be added based on the threshold. Once the region is fully grown, the mean (`LVmean`) and standard deviation (`LVstd`) of the intensity values within this LV region are computed. All pixels that were visited during this growing step are considered as part of the full-blood region for subsequent calculations.

After that, step 3 addresses coil-sensitivity distortion in MR images, which can affect segmentation and volume calculations. MR scanners exhibit intensity variations based on coil position, causing distortions. To correct this, the code fits a plane to the full-blood voxels (identified in step 2) using a least-squares method. This plane models the intensity variations caused by the coil's sensitivity. Once the plane is fitted, it is subtracted from the original image to remove the distortion, producing a corrected image. The `fit_plane_least_squares` function extracts the coordinates and intensity values of full-blood voxels and fits a plane equation $z = ax + by + c$ to the data. The least squares method finds the optimal coefficients of the plane. The `correct_coil_bias` function then subtracts the plane from the original image to remove the coil bias. This process compensates for coil sensitivity, improving the accuracy of segmentation and analysis in MR images.

Looking at the values of the corrected image, it needs to be normalized to make sure its values are within a standardized range, enhancing the stability of subsequent analysis (to avoid negative values).

For Step 4, I focused on estimating the myocardium's signal intensity (`Myoc_mean`) using a threshold-based region-growing technique, similar to Step 2. I started by calculating the mean and standard deviation of the left ventricle (`LVmean` and `LVstd`) in the corrected image, which I obtained in Step 3. I then defined a series of decreasing thresholds based on these values and the seed intensity. As I applied these thresholds, I monitored the growth of the region and tracked how the volume changed. If the volume increased suddenly, I knew the region was starting to include surrounding structures like the right ventricle or fat. This discontinuity in volume growth helped me identify the threshold where the region-effusion began. I also tracked the growth ratio between thresholds to pinpoint the exact moment the region expanded too far.

Once I identified the correct threshold, I refined the myocardium mask by dilating it to ensure I captured the full myocardium and excluded surrounding areas. Finally, I calculated the mean and standard deviation of the intensity in the myocardium mask, giving me the `Myoc_mean` and `Myoc_std`, which are key for accurate estimation of the myocardium's signal intensity.

In the final step of LV segmentation (step 5), I encountered a challenge when using the standard deviation of the LV region (`LVstd`) as a threshold. Initially, I attempted to use it to define a threshold for the region-growing process, but I found that the LV standard deviation was quite variable across different slices. This variability made it unstable for consistent thresholding and as a result, it wasn't a reliable criterion for segmenting the LV region. The fluctuations in LV intensity made it hard to predict a proper threshold value, leading to inconsistencies in the segmented LV mask.

To address this issue, I decided to try a different approach. Instead of relying on the LV standard deviation, I used a more stable threshold based on the mean signal intensity of the myocardium (`myoc_mean`) and the seed value. Specifically, I calculated a threshold as the average of the `myoc_mean` and the seed value:

$$\texttt{secure\_final\_threshold\_step5} = \frac{\texttt{myoc\_mean} + \texttt{seed\_val}}{2}$$

This provided a more robust and stable threshold that didn't vary as much between slices. I then applied the region-growing algorithm with this new threshold and iterated over different threshold values, adjusting based on the segmented LV volume. If the volume was too large (over 4000), I increased the threshold and if it was acceptable, I stopped iterating. This method allowed me to segment the LV more reliably, with the final mask representing the correct LV region. This approach ensures that the segmentation stays within the LV region and prevents the segmentation from "escaping" the LV. Additionally, each time the segmentation volume exceeds 4000, the threshold becomes more restrictive, helping to maintain accurate and stable segmentation.

4 Results

4.1 Classification results

In this section, the classification models' performances are compared based on their cross-validation accuracy. Here there are the results of various models tested, including a basic Random Forest model, XGBoost, PCA with Random Forest and AdaBoost . The overall accuracy for each model is summarized in Table 1.

Model	**Accuracy (CV)**
Basic Random Forest (Split)	0.90
Random Forest with best parameters	0.90
XGBoost	0.89
PCA + Random Forest	0.80
Random Forest (Top 14 Features)	0.93
Random Forest (Dropping Redundant Features, Submitted)	0.92
AdaBoost	0.79
Segmentation-based Random Forest	0.92

Table 1: Table 1: Overall accuracy (CV) for each model.

Table 2 presents the precision scores per class for two models: the Basic Random Forest (Split) and the Random Forest (Cross-Validation with Top 14 Features). These precision values indicate the model's ability to correctly identify each class, offering a deeper understanding of its performance across individual categories.

Model	**Class 0**	**Class 1**	**Class 2**	**Class 3**	**Class 4**
Basic Random Forest (Split)	0.80	1.00	0.80	1.00	1.00
Random Forest (CV with Top 14 Features)	0.80	1.00	0.80	1.00	1.00

Table 2: Table 2: Precision per class for Basic Random Forest (Split) and Random Forest (Cross-Validation with Top 14 Features).

The models were evaluated using cross-validation, and the results show that the Random Forest with dropping redundant features achieved one of the best accuracy of 0.92, as well as the Random Forest (Top 14 Features) at 0.93 and the segmentation-based Random Forest. It is important to consider that those values only consider the Training labeled data. Then, the PCA + Random Forest model, while offering an accuracy of 0.80, it performed significantly lower than the other models. Additionally, the AdaBoost model showed the least performance with an accuracy of 0.79.

The Random Forest models with cross-validation consistently performed well, with a solid accuracy of 0.90 for both the basic Random Forest (Split) and Random Forest (Cross-Validation) models, while XGBoost had an accuracy of 0.89.

For my submission, I chose the prediction made by the Random Forest with dropping of redundant features. I identified an issue with my previous model, Random Forest (Top 14 Features), where a few of the selected features were highly correlated and did not contribute additional valuable information. To address this, I decided to remove those redundant features. Given the relatively small number of features I had, I revisited the previously lower-ranked features that were outside the top 14 in terms of feature importance, and considered them again. By incorporating these, I was able to enhance my model's performance. As a result, before the "14 best features" model, as I referred to it, achieved an accuracy of 0.93 on Kaggle. After further refining the feature set, the new model reached a score of 1.00.

For the classification model, I chose the version with a cross-validation score of 0.92, as I believed it would generalize better to unseen data, despite a slightly lower performance compared to the 0.93 achieved using the 14 best features. This approach yielded a strong result with an accuracy of 88% on the hidden cases, which I consider to be a remarkable performance.

4.2 Segmentation results

In this section, three different cases are presented to illustrate the good performance of the segmentation algorithm. The figures demonstrate how the algorithm progressively refines the segmentation, correcting previous errors at each step. It is worth mentioning that in the notebook, it is possible to access the various segmentations through the function plot_image_and_segmentation to visualize the improvements at each stage. And the code for the segmentation is in segmentation.py.

The first case corresponds to a standard left ventricle (LV) segmentation. The second case illustrates a myocardium that appears to be open, which makes segmentation more challenging, as the LV lies within the MYO and this boundary or change of luminosity is used to define the limits of the LV. Despite the complexity, the segmentation result is very accurate. The third case presents a particularly difficult scenario, where the LV is quite small. In this case, the initial seed can touch the myocardium, causing the first region-growing process to explode, because the intensity of the seed can be very close to the one of the myocardium. However, as the algorithm progresses, the segmentation correctly identifies the LV.

The following figures show the progression of the segmentation in each case, from the initial image to the final segmentation, highlighting the effectiveness of the algorithm.

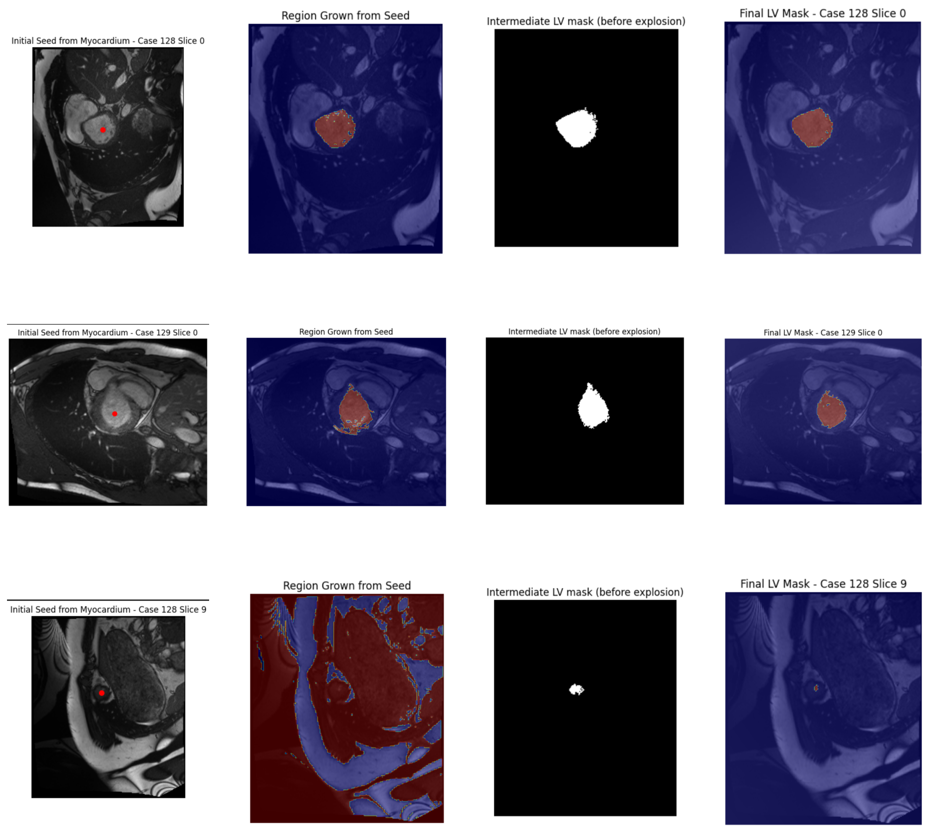
Initial Seed from Myocardium - Case 128 Slice 0
Region Grown from Seed
Intermediate LV mask (before explosion)
Final LV Mask - Case 128 Slice 0
Initial Seed from Myocardium - Case 129 Slice 0
Region Grown from Seed
Intermediate LV mask (before explosion)
Final LV Mask - Case 129 Slice 0
Initial Seed from Myocardium - Case 128 Slice 9
Region Grown from Seed
Intermediate LV mask (before explosion)
Final LV Mask - Case 128 Slice 9

Figure 2: **The figures show the segmentation process for three cases: Case 1 shows normal LV segmentation, Case 2 demonstrates segmentation within the myocardium, and Case 3 shows a small LV segmentation. Each row shows the progression from initial slice to final segmentation.**

Although I was unable to submit my prediction with the segmentation to Kaggle (due to my SSD being damaged, and I have not been able to recover my computer yet), the segmentation achieved a 0.92 accuracy in cross-validation with the final set of features used for prediction. Based on intuition, it could be said that my segmentation would have achieved a result between 76% accuracy and 100% (although the latter is unlikely), since I observed that there are 6 class prediction differences compared to the prediction I submitted.

5 Conclusion

In this project, I applied machine learning techniques to classify cardiac diseases and segment the left ventricle (LV) from medical images. The classification results were promising, with a good performance in cross-validation. However, one of the major challenges I faced was the limited number of training cases, making it difficult to determine with certainty whether the model was overfitting.

For the segmentation task, the method worked well overall, with a few challenges that required attention. The most significant challenge was the subtle difference in intensity between the myocardium (MYO) and LV, which sometimes made it difficult for the algorithm to distinguish between the two. In some cases, the MYO appeared 'open', which caused the region growing algorithm, starting from within the LV, to 'explode' outside of the MYO, resulting in segmentation errors. Additionally, the initial seed point sometimes had an intensity value closer to that of the MYO rather than the LV, especially in cases where the LV contained dark spots. These spots confused the algorithm and made it difficult to set the appropriate threshold for the segmentation.

Despite these challenges, the segmentation method produced promising results, and it was possible to segment the LV accurately in most cases.

Through this project, I gained valuable insights into the application of machine learning in medical image analysis. I learned to navigate complex challenges related to both classification and segmentation, ultimately obtaining good results that validate the effectiveness of my approach.

6 Future Work

Although the current approach works well, there is still room for improvement. Future work could involve utilizing deep learning models for both classification and segmentation, as these methods may offer better generalization and more accurate results, particularly for complex cases. Additionally, improving feature extraction techniques and addressing challenges such as data imbalance would further enhance the robustness of the model for real-world applications.

Considering my current implementation, one potential improvement would be to avoid saving the segmentation files. Instead, I could focus solely on extracting the features after performing the segmentation, which would reduce computational resource usage and optimize the process a lot.

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," *IEEE Transactions on Biomedical Engineering*, vol. 57, no. 4, pp. 905–913, 2010.

[2] M. Khened, V. A. Kollerathu, and G. Krishnamurthi, "Automatic cardiac disease assessment on cine-mri via time-series segmentation and domain specific features," in *Statistical Atlases and Computational Models of the Heart. ACDC and MMWHS Challenges*, pp. 68–81, Springer, 2018.

[3] M. Khened, V. A. Kollerathu, and G. Krishnamurthi, "Densely connected fully convolutional network for short-axis cardiac cine mr image segmentation and heart diagnosis using random forest," in *Statistical Atlases and Computational Models of the Heart. ACDC and MMWHS Challenges*, pp. 140–151, Springer, 2018.