U-NET FOR CARDIAC MRI SEGMENTATION
ACDC Challenge

IMA206
Project Report

ABRIK Samia
AKIL Zakaria
EL BOUHALI Adnane
Supervisor : Loïc Le Folgoc

June 2024

1 Introduction

1.1 Problematic

Cardiovascular diseases (CVDs) remain the leading cause of death globally, accounting for approximately 31% of all deaths each year, according to the World Health Organization (WHO). Early diagnosis and accurate assessment of cardiac function are crucial in managing and treating these conditions. Magnetic Resonance Imaging (MRI) has emerged as a vital non-invasive imaging technique for visualizing the heart's structure and function, offering unparalleled detail and clarity. However, the manual analysis of cardiac MRI images is labor-intensive, time-consuming, and subject to inter-observer variability. The need for automated and precise segmentation of cardiac structures from MRI images is therefore critical in improving diagnostic accuracy, treatment planning, and patient outcomes.

1.2 Objective

The goal of this project is to develop and evaluate a deep learning model, specifically a U-Net architecture, to automatically segment cardiac structures from MRI images using the Automated Cardiac Diagnosis Challenge (ACDC) dataset. The U-Net model, with its encoder-decoder structure, is well-suited for image segmentation tasks due to its ability to capture fine-grained details and contextual information. This project aims to achieve the following objectives:

- Data Preprocessing: Implement preprocessing steps to prepare the ACDC dataset for training and validation, including normalization, cropping, and slicing of MRI images.
- Model Development: Train a U-Net model to accurately segment key cardiac structures such as the left ventricle, right ventricle, and myocardium.
- Data Augmentation: Apply various data augmentation techniques to enhance the model's robustness and generalizability.
- Evaluation: Assess the model's performance using metrics such as Dice coefficient and Cross Entropy loss.
- Visualization: Provide visual representations of the segmentation results to demonstrate the model's effectiveness.

1.3 Description of the Dataset

The ACDC dataset is a well-known benchmark dataset used for the automated analysis of cardiac MRI images. It was introduced as part of the MICCAI 2017 challenge, aimed at advancing the state-of-the-art in cardiac image analysis. The dataset contains MR images of 150 patients, covering a wide range of pathological cases that include:

- Normal (NOR): Patients with no significant abnormalities.

- Myocardial Infarction (MI): Patients with previous myocardial infarction, showing alterations in the myocardial wall.
- Dilated Cardiomyopathy (DCM): Patients with dilated cardiomyopathy, characterized by dilation and impaired contraction of the left ventricle.
- Hypertrophic Cardiomyopathy (HCM): Patients with hypertrophic cardiomyopathy, showing thickened myocardial walls.
- Abnormal Right Ventricle (ARV): Patients with right ventricular abnormalities.

Each patient in the dataset is represented by a series of short-axis MR images acquired at end-diastolic (ED) and end-systolic (ES) phases of the cardiac cycle. The images are annotated by expert cardiologists, providing ground truth labels for three primary cardiac structures:

- Left Ventricle (LV): The chamber of the heart responsible for pumping oxygenated blood to tissues all over the body.
- Right Ventricle (RV): The chamber that pumps deoxygenated blood to the lungs for oxygenation.
- Myocardium (Myo): The muscular tissue of the heart.

The dataset is divided into training and testing sets, with 100 patients in the training set and 50 patients in the testing set. Each patient's data includes MR images and corresponding ground truth segmentations for both the ED and ES phases.

2 Methodology

2.1 Loading and Visualizing Data

In this section, we describe the processes of loading and visualizing the cardiac MRI data from the ACDC dataset. The ACDC dataset contains MRI images and corresponding segmentation masks for different phases of the cardiac cycle, specifically the end-diastolic (ED) and end-systolic (ES) phases. To effectively train the U-Net model, it is crucial to correctly load, preprocess, and visualize the data.
The data is organized into two main folders: 'training' and 'testing'. Each folder contains subfolders named after patient IDs in the format patientXXX, where XXX is a zero-padded number (e.g., patient001). Each patient's folder contains MRI image files and corresponding ground truth segmentation files in the NIfTI format with the .nii.gz extension.

The loading process involves the following steps:

- Determining Data Folders: Based on the patient ID, we determine whether the data belongs to the training set (IDs 1-100) or the testing set (IDs 101-150). This classification helps in organizing the data for the model.

- Reading Configuration Files: Each patient's folder contains a configuration file (Info.cfg) that specifies the frame numbers corresponding to the ED and ES phases. Reading this file allows us to accurately locate and load the specific frames for each phase.
- Loading Images: The MRI images and segmentation masks are loaded using specialized medical imaging libraries (such as MONAI), which handle the NIfTI format effectively.

By systematically organizing and loading the data in this manner, we ensure that the MRI images and their corresponding segmentation masks are correctly paired and ready for subsequent steps such as preprocessing and augmentation.
Visualization is a crucial step in verifying the integrity and correctness of the loaded data. It involves displaying the MRI slices along with their corresponding segmentation overlays to ensure they are properly aligned and accurately reflect the anatomical structures of interest.

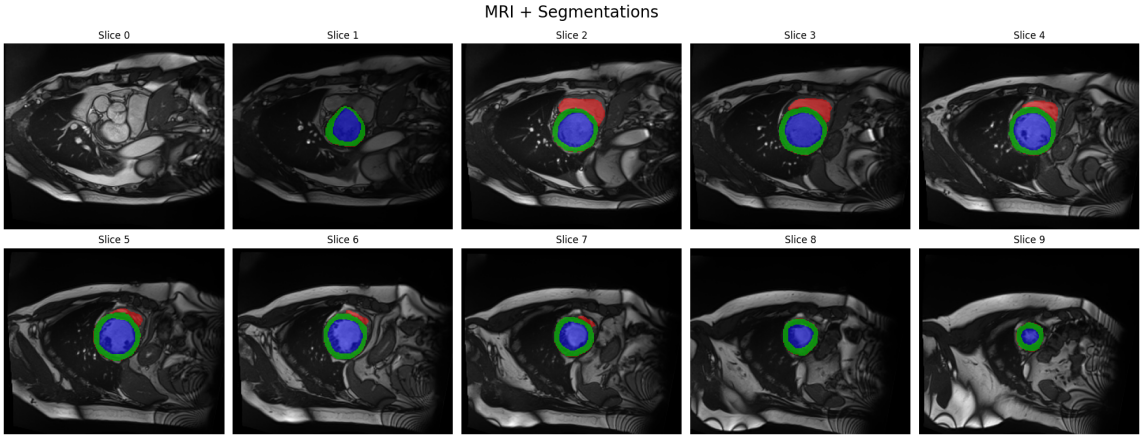

Figure 1: Example visualization of a patient's 2D slices with their corresponding segmentations

2.2 Pre-processing

Preprocessing is a critical step in preparing the cardiac MRI images and their corresponding segmentation labels for input into the U-Net model. The goal of preprocessing is to ensure that the data is in a suitable format and condition for model training and evaluation, which improves the overall performance and accuracy of the model. In this project, the preprocessing steps were focused on transforming the images into tensors and cropping the images to focus on the region of interest.

2.2.1 Transforming Images into Tensors

The MRI images and segmentation masks were initially loaded using the MONAI library's LoadImage function. This function reads the medical images and converts them into a format that can be easily manipulated. After loading, the images were transformed into tensors, which are the primary data structure used by PyTorch. This conversion enables

efficient processing and leverages PyTorch's capabilities for tensor operations and GPU acceleration.

2.2.2 Cropping

To focus on the region of interest, we implemented a function to crop the MRI images and segmentation masks around the heart. A margin of 20 pixels was added to ensure the heart is fully included within the crop. The cropping is made along volumes in order to have a uniform cropping of slices in the same volume. Additionally, the crop was made square to prevent deformation and distortions of the heart shape. This was achieved by calculating the maximum length of the crop in the x and y dimensions and centering the crop accordingly.

2.2.3 Resizing

After cropping, the images and segmentation masks were resized to a uniform size of 128x128 pixels. This resizing process ensures consistency in the dimensions of the input data while preserving the essential features of the heart region. Standardizing the input size is crucial for the U-Net model, as it expects inputs of a fixed size.

2.2.4 Slicing

The cardiac MRI images are volumetric data, consisting of multiple slices. To feed these images into the U-Net model, we transformed the 3D volumes into distinct 2D slices. Each slice was treated as an individual image, with its corresponding segmentation label. This approach allows the model to learn from 2D representations of the 3D volumes and simplifies the training process.

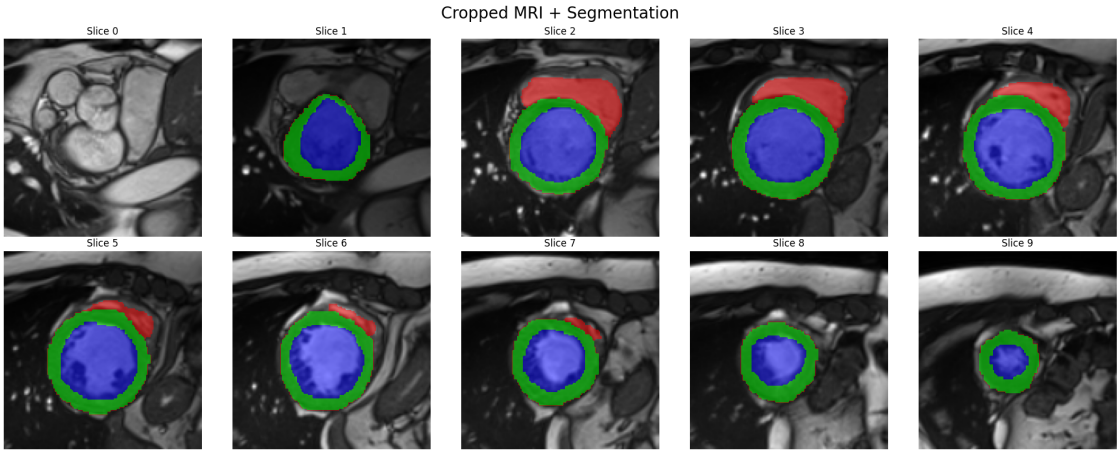

Figure 2: Result of cropping and resizing on the previous example

2.3 Data Augmentation

Initially, we trained our U-Net model on the cardiac MRI images without any data augmentation. This approach yielded good and promising results. However, to make the project

more robust and interesting, we decided to incorporate data augmentation techniques. Data augmentation is a powerful strategy in machine learning and deep learning that involves creating new training samples from the existing data. It helps to:

- Increase the size of the training set: By generating new variations of the training data, data augmentation effectively increases the size of the dataset, which can lead to better generalization of the model.
- Improve model robustness: Augmented data introduces variations making the model more robust to different inputs.
- Prevent overfitting: By exposing the model to diverse examples, data augmentation reduces the risk of overfitting, where the model performs well on the training data but poorly on unseen data.

To achieve these benefits, we applied several data augmentation techniques using the MONAI library. The transformations we used are:

Transformation	**Description**	**Purpose**
RandRotate90d	Randomly rotates the image and segmentation mask by 90 degrees.	This transformation introduces rotational variations in the data, helping the model learn rotational invariance.
RandFlipd (axis 0)	Randomly flips the image and segmentation mask along the x-axis.	This transformation helps the model become invariant to horizontal flips, simulating different orientations that the heart might appear in.
RandFlipd (axis 1)	Randomly flips the image and segmentation mask along the y-axis.	Similar to the previous flip, this transformation helps the model learn to recognize features regardless of their vertical orientation.
RandScaleIntensityd	Randomly scales the intensity of the image.	This transformation adjusts the brightness and contrast of the images, helping the model learn to be invariant to differences in image intensity that might result from variations in imaging conditions.
RandAdjustContrastd	Randomly adjusts the contrast of the image based on a gamma value.	This transformation further adjusts the image contrast, allowing the model to better handle variations in contrast and ensuring that it focuses on essential features regardless of contrast differences.

2.4 U-net Architecture

The U-Net architecture, as described in the original paper [2] and implemented in the milesial/Pytorch-UNet repository [1], features a distinctive design aimed at semantic segmentation tasks, such as segmenting cardiac MRI images. The U-Net architecture can be divided into two main parts: the encoder and decoder. The contracting path, encoder, captures the context and reduces the spatial dimensions of the input, while the expanding path, decoder, recovers, though skip connections, the spatial information and generates the segmentation map.

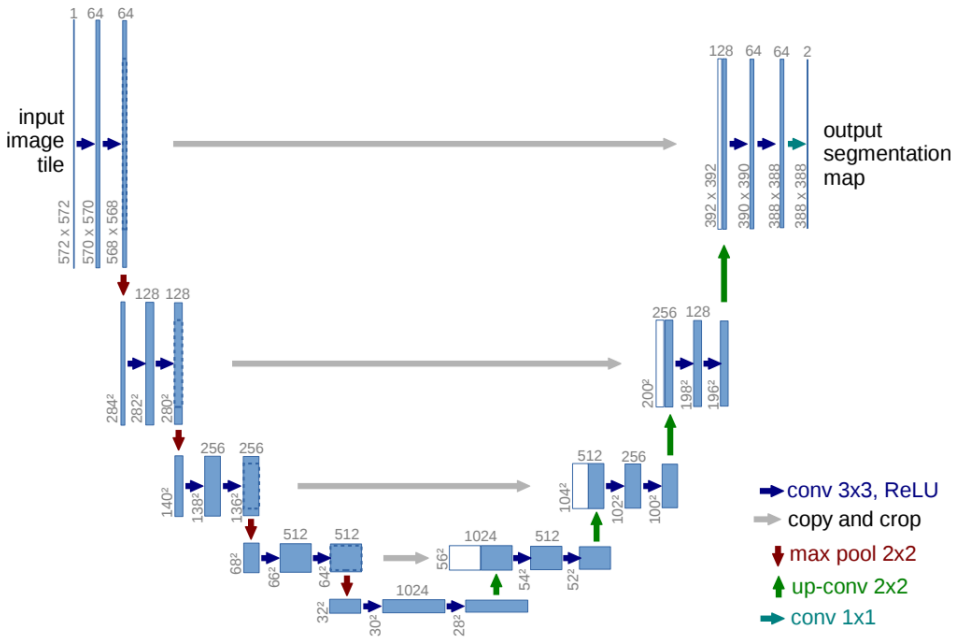

Figure 3: U-net architecture

Contracting Path:
The contracting path follows the typical architecture of a convolutional network. It consists of the repeated application of two 3x3 convolutions (unpadded convolutions), each followed by a rectified linear unit (ReLU) and a 2x2 max pooling operation with stride 2 for downsampling. At each downsampling step we double the number of feature channels.

Expansive Path:
Every step in the expansive path consists of an upsampling of thefeature map followed by a 2x2 convolution ("up-convolution") that halves the number of feature channels, a concatenation with the correspondingly cropped feature map from the contracting path, and two 3x3 convolutions, each followed by a ReLU.

Final Layer:
At the final layer of the network, a 1x1 convolutional layer is employed to map the 64-component feature vector to the desired number of output classes, which in the case of cardiac MRI segmentation is equal to 4.

Four essential classes are used to collectively build the U-Net architecture:

1. DoubleConv: defines a double convolutional block consisting of two sequential convolutional layers followed by batch normalization (BN) and ReLU activation. This architecture (convolution $\rightarrow$ BN $\rightarrow$ ReLU) * 2 helps the model learn hierarchical features by applying two sets of convolution operations.

2. Down: performs downscaling using max pooling followed by a DoubleConv block. The max pooling reduces the spatial dimensions of the input feature map by a factor of 2, while DoubleConv captures and refines the features at each downsampling step.

3. Up: performs upscaling of the input feature map followed by a DoubleConv block to handle skip connections in the expansive path of U-Net. It uses either bilinear upsampling or transposed convolution depending on the bilinear flag. The concatenation of the upsampled feature map with the cropped feature map from the contracting path ensures that the model preserves spatial information while refining the segmentation.

4. OutConv: defines the final convolutional layer that maps the features to the desired number of output channels. It uses a 1x1 convolution layer to perform this mapping.

Together, these classes form the core components of the U-Net architecture, facilitating effective feature extraction, spatial localization, and segmentation output generation for complex medical image analysis tasks like cardiac MRI segmentation.

2.5 Training

2.5.1 Loss Functions

In our training process, we utilized two different loss functions: Cross-Entropy Loss and Dice-Cross-Entropy Loss.

1. Cross-Entropy Loss:
Cross-Entropy Loss is commonly used for classification problems. It measures the performance of a classification model whose output is a probability value between 0 and 1. The Cross-Entropy Loss increases as the predicted probability diverges from the actual label.

Mathematically, the Cross-Entropy Loss is defined as:

$$\text{CE Loss} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c}\log(\hat{y}_{i,c})$$

where:

- N is the numbers of samples.
- C is the number of classes.
- $y_{i,c}$ is the actual label for sample i and class c.
- $\hat{y}_{i,c}$ is the predicted probability for sample i and class c.

2. Dice-Cross-Entropy Loss:
Dice Loss is used to measure the overlap between two samples. It is particularly useful in segmentation tasks where the objective is to maximize the overlap between the predicted and ground truth masks. The Dice-Cross-Entropy Loss combines both Dice Loss and Cross-Entropy Loss to leverage the advantages of both.

Mathematically, the Dice Loss is defined as:

$$\text{Dice Loss} = 1 - \frac{2\sum_{i=1}^{N} y_i\hat{y}_i}{\sum_{i=1}^{N} y_i + \sum_{i=1}^{N}\hat{y}_i}$$

The combined Dice-Cross-Entropy Loss can be expressed as:

$$\text{DiceCE Loss} = \text{Dice Loss} + \lambda \text{ CE Loss}$$

where λ is a weighting factor that can control the relative importance of each loss during training, ensuring an optimal balance between classification accuracy and spatial coherence. We used $\lambda = 0.1$ experimantally.

2.5.2 Training setup

- **Data Splitting:** The dataset was split into training and validation sets. Specifically, 70% of the data was used for training, and the remaining 30% was reserved for validation.
- **Batch Size:** A batch size of 32 was used to ensure efficient training and stable gradient updates.
- **Epochs:** The model was trained for a total of 50 epochs, which provided sufficient iterations for convergence.
- **Optimizer:** We used the Adam optimizer with a learning rate of 10^{-3}. This optimizer is known for its ability to handle sparse gradients and adaptively adjust learning rates.

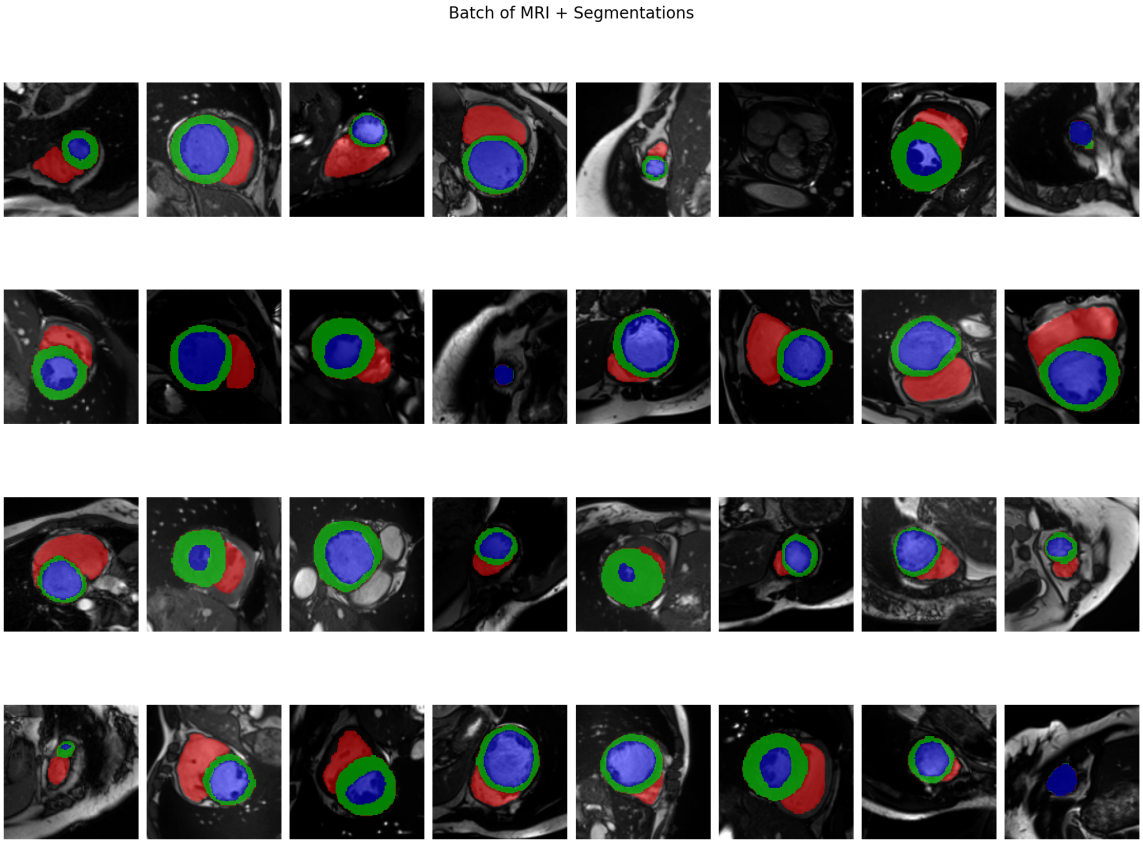

Figure 4: Batch of 32 shuffeled slices

2.6 Results

2.6.1 Results without data augmentation

We initially trained the model using only the provided dataset, without any data augmentation, and with just the CrossEntropyLoss as criterion to optimize, to establish a baseline for comparison with results obtained after applying data augmentation techniques.

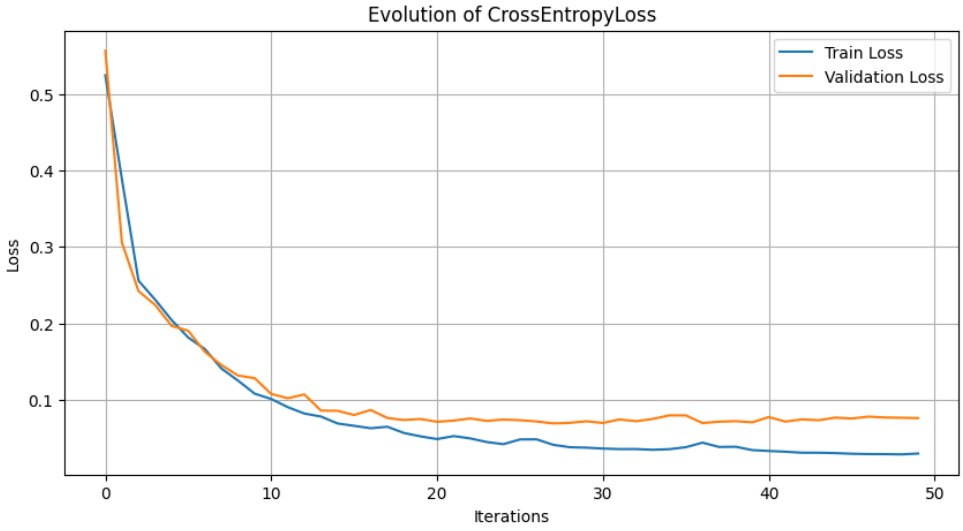

Figure 5: Evolution of the train and validation CrossEntropyLoss during training

- Consistent Learning: The training process exhibited a consistent decrease in both training and validation loss, achieving a low and stable final loss value. This indicates that the model effectively learned from the dataset and did not overfit.
- Stability: The loss curves demonstrate the stability of the training process. Both training and validation losses converge smoothly, suggesting that the model's parameters are being optimized effectively without significant fluctuations.

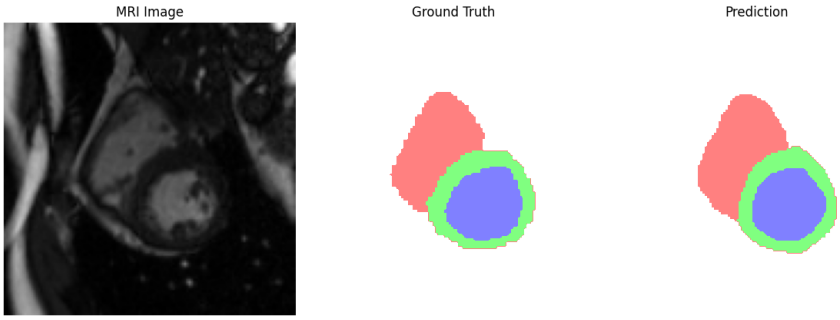

Figure 6: Example of segmentation prediction on the train set

- High Accuracy: The visual predictions of the model are highly accurate when compared to the ground truth. The segmented heart regions are clearly defined, and the model successfully identifies the different anatomical structures within the heart.

- Promising Results: These results are very promising, indicating that even without data augmentation, the model is capable of producing high-quality segmentations. The clear distinction between the predicted regions and the ground truth demonstrates the model's ability to generalize from the training data.

2.6.2 Results with data augmentation

To further enhance the model's robustness and generalization capabilities, we incorporated data augmentation techniques during training. This involved randomly rotating, flipping, scaling, and adjusting the intensity and contrast of the training images, as previously described. We used as criterion to optimize both CrossEntropyLoss and DiceCEloss to understand the impact of these techniques on training and validation performance.

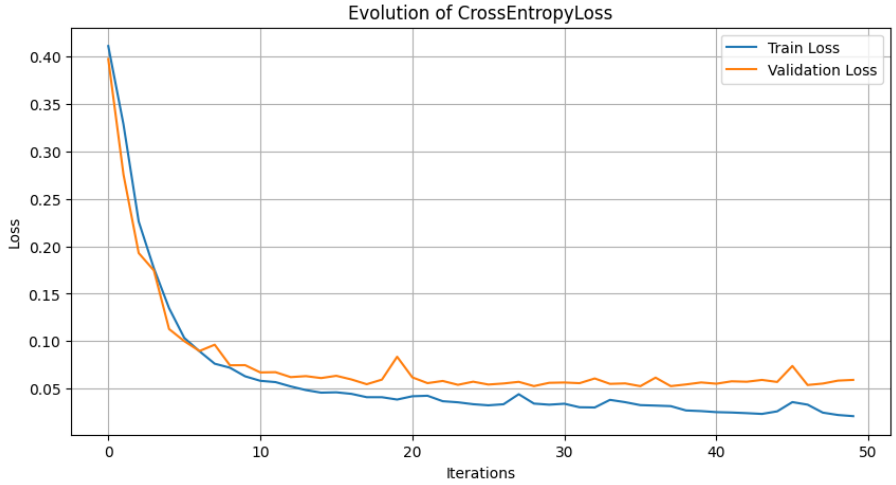

Figure 7: Evolution of the train and validation CrossEntropyLoss during training

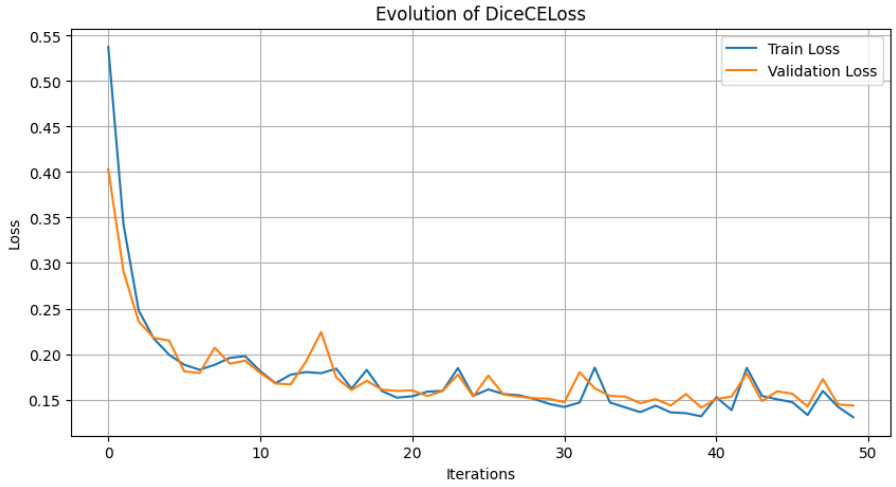

Figure 8: Evolution of the train and validation DiceCELoss during training

The training and validation loss curves with data augmentation exhibit more fluctuations compared to the initial results without data augmentation. This increased variability is expected as the data augmentation techniques introduce a broader range of variations in the training set, making it more challenging for the model to converge smoothly.

Despite the fluctuations, the loss curves indicate that the model is learning to generalize better across different variations of the input data. The data augmentation helps the model to adapt to a wider distribution, thereby enhancing its ability to handle diverse and unseen examples during inference.

The results with CrossEntropyLoss and DiceCEloss show that both loss functions can effectively guide the model training, though DiceCEloss, being a compound loss function, may offer better handling of class imbalances in segmentation tasks. The fluctuations in the curves reflect the model's adaptation process to the augmented dataset, and over time, the model stabilizes as it finds a balance within the augmented data's broader distribution.

Overall, data augmentation contributes to improving the model's robustness by exposing it to a variety of transformations, which is crucial for achieving reliable performance in real-world scenarios. This will be further be confirmed during testing phase where we evaluate the model using Dice score.

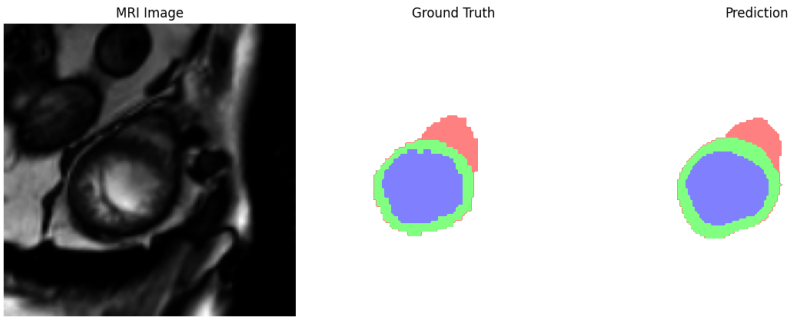

Figure 9: Another example of prediction on the train set

Qualitatively, we also get a very similar segmentation prediction compared to the ground truth.

2.7 Testing and Evaluation

For the testing phase, we applied the same preprocessing steps as used during training, ensuring consistency across the datasets. This involved transforming the 3D MRI volumes into 2D slices and converting the images into tensors. However, we did not apply any data augmentation techniques during testing to maintain the integrity of the original test data.

To evaluate the model's performance on the test set, we used the DiceMetric from the MONAI library, which is a widely recognized metric for evaluating segmentation tasks in medical imaging.

The Dice Score is a statistical measure used to gauge the similarity between two sets. It is particularly effective for evaluating segmentation tasks where it measures the overlap

between the predicted segmentation and the ground truth segmentation. The Dice Score ranges from 0 to 1, where:

- 0 indicates no overlap.
- 1 indicates perfect overlap.

The formula for the Dice Score is:

$$DICE = \frac{2|A \cap B|}{|A| + |B|}$$

where A is the set of predicted segmentation pixels, and B is the set of ground truth segmentation pixels.

The Dice Score is particularly robust in handling class imbalances common in medical imaging, where the background often dominates the image. It provides a direct measurement of the overlap between the predicted and ground truth segmentations, making it a natural choice for segmentation tasks. In medical imaging, precise overlap between predicted and actual segmentations is crucial for accurate diagnosis and treatment planning, making the Dice Score an essential metric for evaluating segmentation performance.

Here is a summary of the scores we got by evaluating the model on the test set:

Model	**Data Augmentation**	**Epochs**	**Loss Function**	**Test Loss**	**Test Mean Dice Score**
U-Net	Yes	50	Cross Entropy Loss	0.0871	0.8630
U-Net	No	50	Cross Entropy Loss	0.0715	0.8423
U-Net	Yes	50	Dice CE Loss	0.1597	0.8820

Table 1: Performance of U-Net Models

For a visual support, we predicted the segmentations on the test set and vizualized them along with the ground truth.

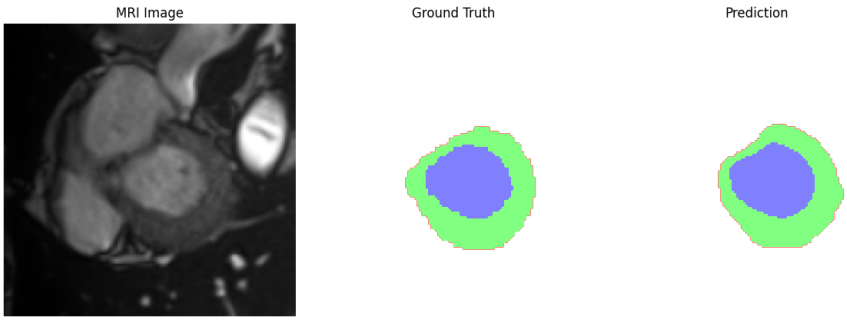

Figure 10: Example of segmentation prediction on the test set

3 Conclusion

In conclusion, the segmentation accuracy of the given method is approximately 88%. This result indicates a solid performance, yet there are several areas for future improvement. Efforts should be directed towards hyperparameter tuning, which involves experimenting with different learning rates, batch sizes, and optimizer configurations to optimize the training process. Additionally, enhancing the model architecture by integrating skip connections, attention mechanisms, or employing more advanced architectures like U-Net++ or Attention U-Net could further improve accuracy. In addition, we should be aware that in practice we don't have access to ground truth segmentations in the test set, so the cropping that we did would be impossible. Therefore, we should consider working on an approach to testing the performance of the model involving evaluating it on patches taken from the image we want to segment (patch sampling). Finally, the use of ensemble methods, which combine predictions from multiple models, can increase both robustness and overall segmentation performance.

References

[1] Milesial. Pytorch-unet, 2018. Accessed: 2024-06-21.

[2] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015.