Challenge - IMA 205

Caini Pan
Nickname: littleleaf

1 Introduction

Cardiac function analysis plays a crucial role in the field of clinical cardiology, serving purposes such as disease diagnosis, patient management, and treatment plan development. Consequently, as artificial intelligence continues to advance, numerous methods have emerged for automated diagnosis based on cardiac magnetic resonance imaging (CMRI), specifically in the form of non-invasive computer-aided diagnosis (CAD).

The subject matter of this study pertains to four cardiac diseases that may not be readily identifiable initially but can potentially give rise to significant complications, such as heart failure and cardiac arrest. Thus, accurately identifying these four conditions at an early stage holds paramount importance for both patients and physicians.

This project seeks to automate the classification of patient types based on various medical metrics. Specifically, it involves a multiple-class classification task.

This report is organized as follows: Section 2 introduces the specific methods used in this challenge and the comparison between the different methods. Section 3 illustrates the experimental part of this challenge, including data pre-processing, experimental results and other steps. Finally, Section 4 is the conclusion of this challenge.

2 Methods

As suggested in the paper [1], some specific medical metrics are needed for the classification task which include the volume of the left atrium, volume of the right ventricle, and other relevant parameters. Calculating these metrics necessitates identifying and labeling the corresponding regions in the 3D MRI images of the heart. However, the left ventricular label is missing for the test set. Therefore, to accurately predict the patient types in the test set, obtaining the left ventricular labels is crucial.

As a result, the method for this challenge in general is divided into two main parts: segmenting the left ventricular portion and classifying the patient categories.

2.1 Segmentation

When it comes to the segmentation of medical images, one widely recognized method is UNet [2]. Therefore, in this experiment, the UNet model is directly considered and modified.

The main network structure of UNet can be divided into encoder and decoder, which are as follows:

1) Encoder: The encoder component consists of a series of convolutional and pooling layers that progressively reduce the size of the input image while extracting high-level feature information. The encoder is characterized by its ability to enhance feature extraction through an increased number of convolutional layers and channels. Typically, it adopts a structure of convolutional layers with ReLU activation and batch normalization layers.

2) Decoder: The decoder component consists of a series of deconvolution and convolution layers that progressively recover the size and boundary information of the original image. The decoder is characterized by its ability to improve image recovery through an increased number of deconvolution layers and channels. Additionally, it utilizes jump connections to directly access low-level feature information from the encoder, thereby enhancing segmentation accuracy.

In UNet, the encoder and decoder are symmetric, which means that each convolutional and pooling layer of the encoder has a corresponding deconvolutional and convolutional layer. Also, UNet has a central part, which is the connection between the encoder and decoder, that further extracts feature information and helps to preserve the boundary information of the image.

Desired results can be attained by employing specific input and output data or by implementing modifications to the network structure. The experimental section will provide a comprehensive explanation of the practical process involved in these attempts, including detailed information about the data format. The present section primarily focuses on the conceptual analysis of why these attempts may or may not yield the desired results.

Three attempts have been recorded as follows:

2.1.1 Modification 1

In the general case of neural networks, it is common practice to input an image and obtain the corresponding segmentation result directly. In scenarios where the image size is not very large and the segmented area exhibits conventional characteristics (such as being approximately round), the input is typically regarded as slices of the original data (without labels), while the labels comprise a binary mask that solely contains the left ventricle label. To enable the network to output the binary segmentation result, the final layer of the network is modified. Specifically, the original network's last layer, which has a channel number of 2, undergoes an additional convolution using a (1x1) convolutional kernel to integrate the channel number to 1. However, in the new output, the value of each pixel corresponds to the sum of the original two channel values. It is important to note that by using the sigmoid activation function for the network layer with a channel number of 2, most of the feature information will be lost.

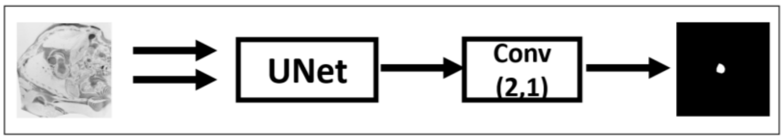


Figure 1 Network Structure (1)

2.1.2 Modification 2

The input remains unchanged, with the slices of the original data serving as the input. However, in this case, the label is transformed into a one-hot encoded format. Subsequently, the network is trained based on the output of the original network, enabling us to obtain the probability of each pixel representing either the background or the left ventricle (the output channel is 2). The channel that corresponds to the probability of each pixel representing the left ventricle is considered as the output, and the resulting output is then binarized using the probabilities obtained.

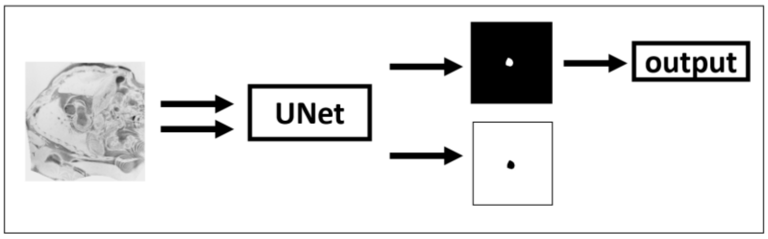


Figure 2 Network Structure(2)

2.1.3 Modification 3

Upon careful examination of the left ventricle's position within the complete slice, it becomes apparent that there exists a surrounding muscle layer, with the myocardial region being accurately labeled. Consequently, we propose replacing the input with data slices containing labels for both the myocardial region and the left ventricle. Even in this scenario, we continue to train the network based on the output of the original network, allowing us to derive the probability of each pixel representing either the background or the left ventricle. The resulting output represents the probability of each pixel being the left ventricle, which is subsequently binarized using the probabilities obtained.

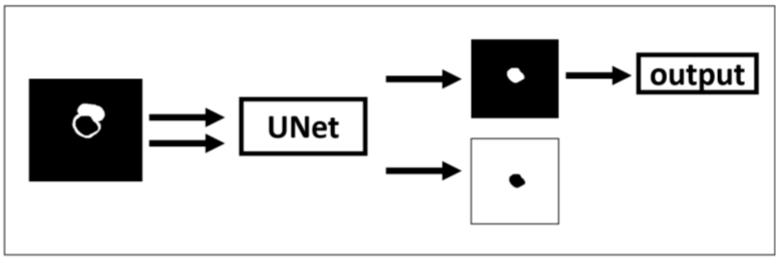


Figure 3 Network Structure (3)

2.2 Classification

A set of medical feature metrics are utilized as extracted features for patient classification. As described in the paper [1], each patient is characterized by both patient-related and image-related features. The patient-related features include the patient's weight (in kg) and height (in cm). Conversely, the image-related features are obtained from automatically acquired segmentations, encompassing volumes (in ml) of the left ventricle, right ventricle, and myocardium for both end-diastole (ED) and end-systole (ES). Additionally, features such as ejection fraction (EF) for the left and right ventricles, the ratio of left ventricular volume to left ventricular volume for ED and ES, and the ratio of myocardial volume to left ventricular volume for ED and ES are calculated. In total, 14 features are utilized, comprising 2 patient-based features and 12 image-based features. With the provided data containing labels, we can compute these metrics and employ them as network inputs for training purposes.

For the test set data, by utilizing the provided myocardial and right ventricular labels in conjunction with the segmented left ventricular label, we can similarly derive the aforementioned features and consequently obtain the necessary features for classifying the test set. Following the suggestion of the paper, I performed two operations:

2.2.1 Random Forest Classifier

A random forest classifier consisting of 100 decision trees was constructed. The classifier utilizes the concept of bagging to classify multiple decision trees through training and voting. During the construction of each decision tree, the random forest classifier randomly selects a specific percentage of data from the original dataset for training and also randomly selects a subset of features for node splitting. This approach helps reduce the variance of the decision tree and enhances the generalization capability of the random forest classifier. As a result, it is well-suited for handling large-scale datasets and high-dimensional data.

2.2.2 Multi-Layer Perceptron

An MLP (Multi-Layer Perceptron) as constructed using the dense net architectures [3]. The dense net is a neural network model composed of multiple fully connected layers, where each neuron in a layer is connected to all neurons in the previous layer. This characteristic of the dense network enables it to exhibit high expressive power. The training process of the dense network involves minimizing the loss function using the backpropagation algorithm and updating the weights and biases of neurons to optimize the model. After conducting multiple iterations, a five-layer network architecture was selected. In this architecture, the input comprises patient characteristics, and the output represents the probability of the patient belonging to each disease type.

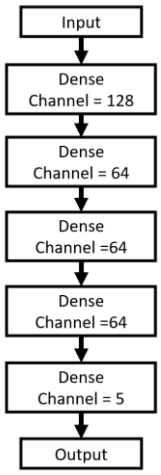


Figure 4

2.3 Loss function

Initially, we considered using the dice loss function (as suggested in the paper). However, the convergence issues were encountered with the loss function due to errors in the code implementation. Consequently, we opted to switch to binary cross-entropy, a common loss function for dichotomous classification tasks. Binary cross-entropy is specifically designed for binary classification problems, enabling the measurement of the discrepancy between the model's predicted output and the true label. It is calculated based on the concept of cross-entropy.

$$BinaryCrossentropy = -\frac{1}{N}\sum_{i=1}^{N} y_{true,i}\log y_i + \left(1 - y_{true,i}\right)\log\left(1 - y_i\right) \tag{1}$$

3 Code development and verification

3.1 Data preprocessing(dataprocess_seg.ipynb)

The data we are working with is originally in a three-dimensional (3D) format, whereas our neural network requires two-dimensional (2D) input. To address this, by using the *nibabel* library, we extract the dimensions (x, y, z) of the 3D data. Subsequently, we slice the data along the z-axis to obtain the axial plane, which is then stored in the corresponding folder.

The non-uniform size of the slices is apparent. To achieve consistency, all the data is resized to dimensions of (256, 256). Moreover, the data values are rescaled to a range of 0 to 255, addressing the occurrence of a WARNING when saving the data using the original values alone. These preprocessing steps prepare the data as input for the network.

The training data is stored in the *Train_img* folder, comprising two file types: those labeled with the "seg" suffix and those without it. Files labeled with the "seg" suffix represent the original data that has been sliced with corresponding labels, while files without the suffix correspond to the original data that has been sliced without labels. Test data is saved in the *Test_img* folder. According to the slices, two types are obtained:

1) The slices containing raw data, which reveal the detailed structure within the heart, are stored in the *Train_img* folder for the training data and the *Test_img* folder for the test data.

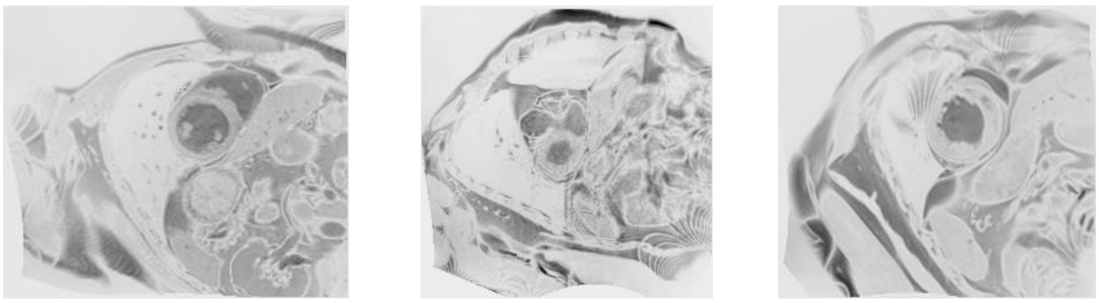


Figure 5 slice without labels in *Train_img*

2) The slices with labels have been thoroughly tested, revealing that the value 253 represents the left ventricle, 254 represents the myocardium, and 255 represents the right ventricle.

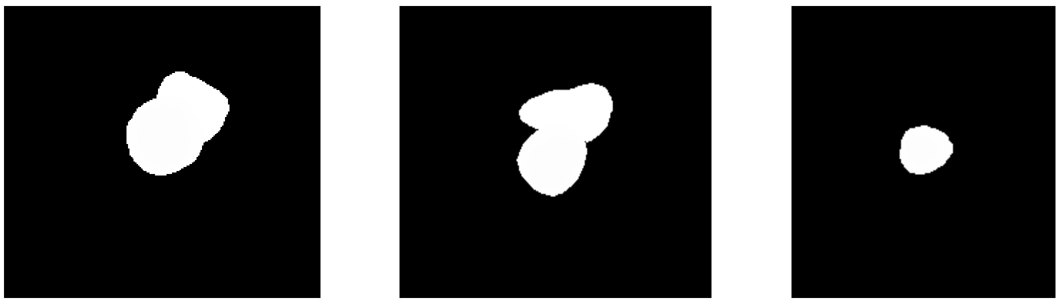


Figure 6 slice with labels in *Train_img*

To prepare the true label for the next segmentation, we simply use the segmentation result of the left ventricle while excluding the regions labeled as the right ventricle and myocardium. This is achieved by setting the pixels with specific values to 0, resulting in the desired data for the segmentation process.

In order to prepare the input data for segmentation, we remove the left ventricular part from the labeled slice by setting those corresponding pixels to 0, while leaving the other regions unchanged. The resulting processed data is saved in the *Mask_noLV* folder.

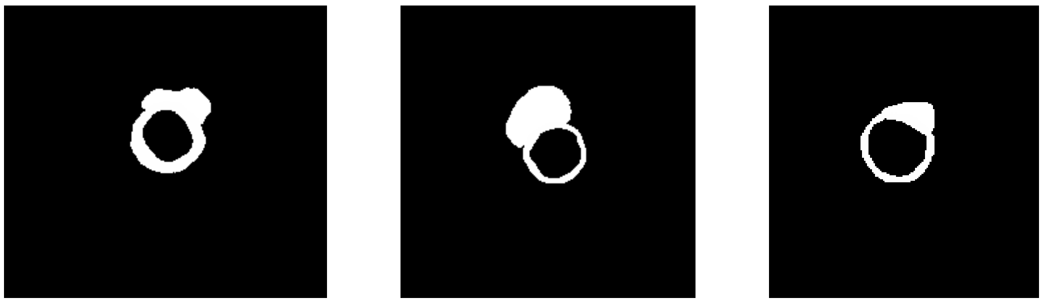


Figure 7 slice in *Mask_noLV*

To obtain the segmentation label specifically for the left ventricular part, we set the pixels corresponding to label 3 to 255, while setting all other pixels to 0. This processing step allows us to create the desired label for the segmentation task. The resulting label is then saved in the *Mask_LV* folder. It is important to note that we do not perform segmentation on the whole test dataset for one time. Instead, we directly read the test dataset for making a single prediction in the subsequent testing period.

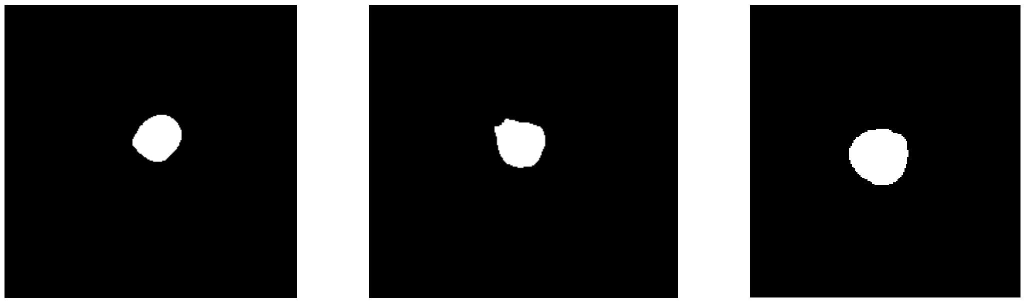


Figure 8 slice in *Mask_LV*

According to the data preprocessing we can get 1902 images, which is enough for a 256x256 segmentation task [4], so no data augmentation is used.

3.2 Segmentation(segmentation.ipynb)

First, we start by reading the slices of raw data (without labels) from the *Train_img* folder. These slices are then resized to (256, 256) and subjected to bitwise inversion. The resulting array has dimensions of (1902, 256, 256). Here is the image after the bitwise inversion:

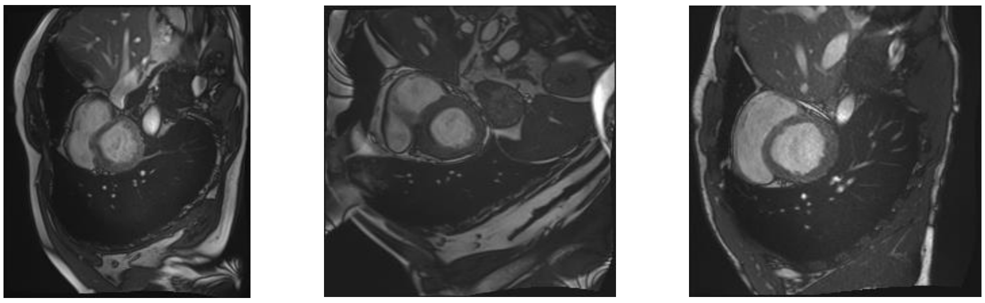


Figure 9 slice of no-label after bitwise inversion

Next, we read the images from the *Mask_noLV* folder, which contain the labels with the left ventricle set to 0. We resize these images to (256, 256), resulting in a final array of dimensions (1902, 256, 256).

Finally, we read the images from the *Mask_LV* folder, which contain the labels with only the left ventricle. These images are resized to (256, 256), resulting in a final array of dimensions (1902, 256, 256).

For the segmentation, the UNet network is chosen as the main segmentation network, and modified it somewhat. For this, I tried three different modifications. A detailed description of each of these changes will be provided by me in the following text.

3.2.1 Modification 1

The input consists of slices from the original data, resulting in a shape of (1902, 256, 256, 1), representing 2D slices of the data. The activation function in the last layer is changed to sigmoid, which is also applied after the last layer of the UNet. In other words, we apply an additional convolutional operation to the output of the original network, resulting in a direct binary mask image with dimensions (256, 256, 1). The last layer of the original network produces probabilities for the background and left ventricle. Integrating the number of channels to 1 would result in the sum of these probabilities, which is not desirable. The obtained results were unsatisfactory due to the feature loss caused by applying a sigmoid activation function during the additional convolution. Due to the unsatisfactory results, no intensive training was conducted.

3.2.2 Modification 2

In modification 2, we incorporated the architecture of the original UNet and introduced dropout with a rate of 0.2 [5]. However, to address potential overfitting, we adjusted the dropout rate to 0.5. Instead, we used the one-hot encoded format with dimensions (1902, 256, 256, 2). As a result, the predicted output had dimensions of (256, 256, 2), with each dimension representing the probability of the pixel being the background/target. To identify the probability of each pixel being classified as the left ventricle, we considered the values from the second dimension of the output data. Finally, employing a probability threshold of 0.5, we applied threshold segmentation to obtain the binarized result.

During the training of this network, the original dataset is divided into a training set, validation set, and test set with proportions of 0.6, 0.3, and 0.1, respectively.

The Adam optimizer was utilized with a learning rate of 0.00005, with the BinaryCrossentropy loss function and a batch size of 16. The training process lasted for 89m57.3s over 100 epochs. The following graphs illustrate the changes in the loss function and accuracy:

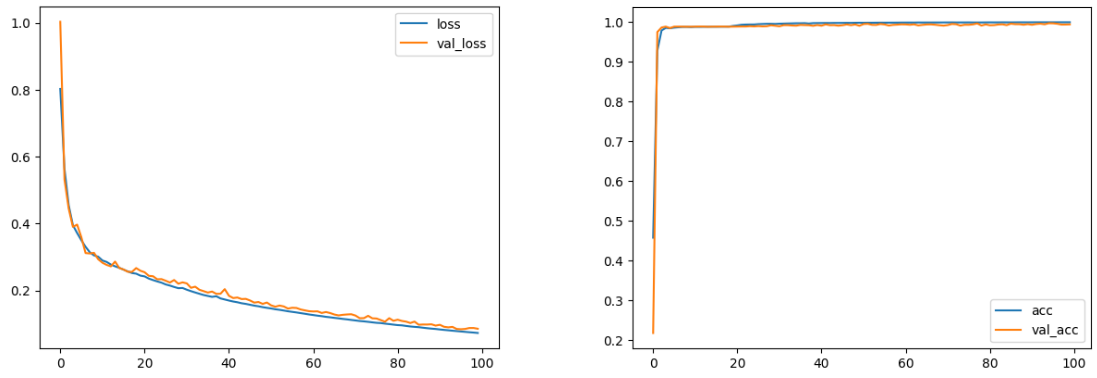
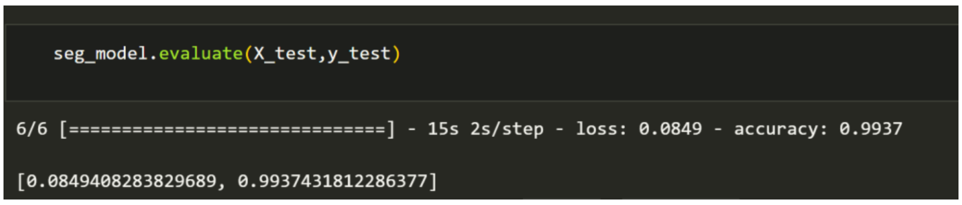


Figure 10 loss value and accuracy of modification 2

This approach yields favorable results on the evaluation of the test set.



```
seg_model.evaluate(X_test,y_test)

6/6 [==============================] - 15s 2s/step - loss: 0.0849 - accuracy: 0.9937

[0.0849408283829689, 0.9937431812286377]
```

Figure 11 test result of modification 2

However, certain errors are noticeable to the human eye, as shown in the results below. The performance heavily relies on the classification network when utilizing such data for classification. Additionally, the segmentation results are not satisfactory. This could potentially be attributed to the complexity of the original data structure, which contains numerous intricate details.

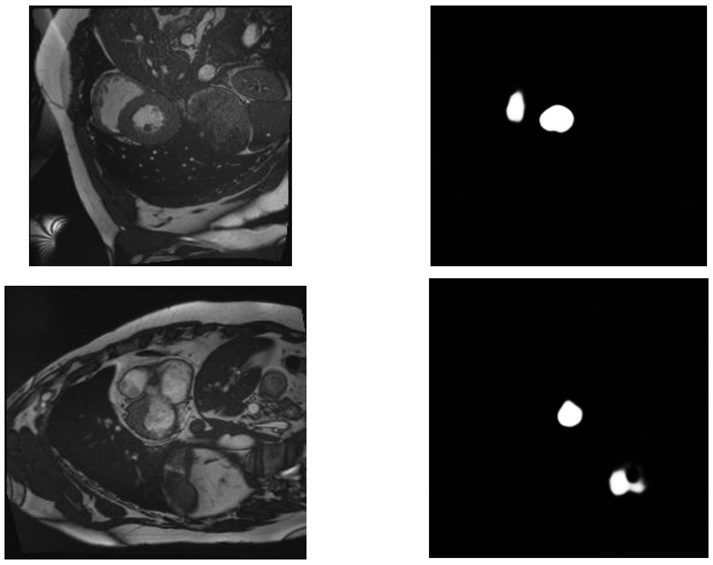


Figure 12 result of segmentation for modification 2

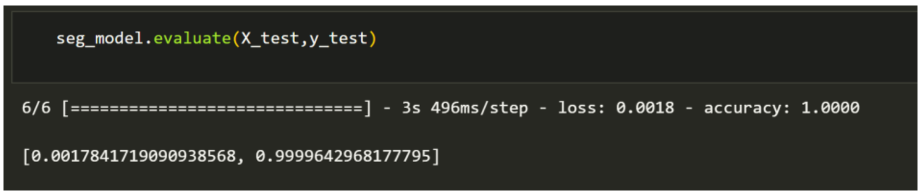
We observed that the segmentation results tested exhibit slightly blurred boundaries since threshold segmentation has not been applied yet. Currently, the probability of each pixel belonging to the left ventricle is being calculated. However, it is evident that certain regions are incorrectly identified with a high probability, appearing very bright.

3.2.3 Modification 3

The input for this method consists of images retrieved from the *Mask_noLV* folder, having dimensions of (1902, 256, 256, 1). The output follows the same format as the original network, and a threshold of 0.5 is applied for segmentation. The results demonstrate significant improvement compared to the original method (the second method). Notably, the occurrence of segmented regions merging together, referred to as "two patches on one map," has been successfully eliminated.

The Adam optimizer was employed with a learning rate of 0.00005, utilizing the BinaryCrossentropy loss function. The training process consisted of 100 generations, with a batch size of 16, and it lasted for a duration of 89m57.3s.

This approach yielded favorable results when evaluated on the test set.


```
seg_model.evaluate(X_test,y_test)
```

```
6/6 [==============================] - 3s 496ms/step - loss: 0.0018 - accuracy: 1.0000

[0.0017841719090938568, 0.9999642968177795]
```

Figure 13 test result of modification 2

The graphs below illustrate the changes in the loss function and accuracy (acc):

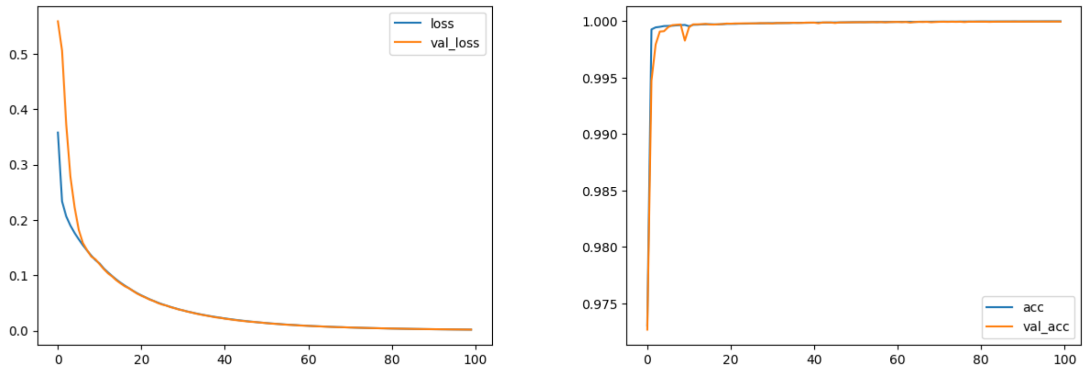


Figure 14 loss value and accuracy of modification 3

In method 2, the network is required to extract features and information from the intricate original structure in order to train effectively. However, in method 3, our input is significantly simplified compared to before, and the domain of labels is better defined. Consequently, the network finds it easier to learn the accurate means of identifying the left ventricle.

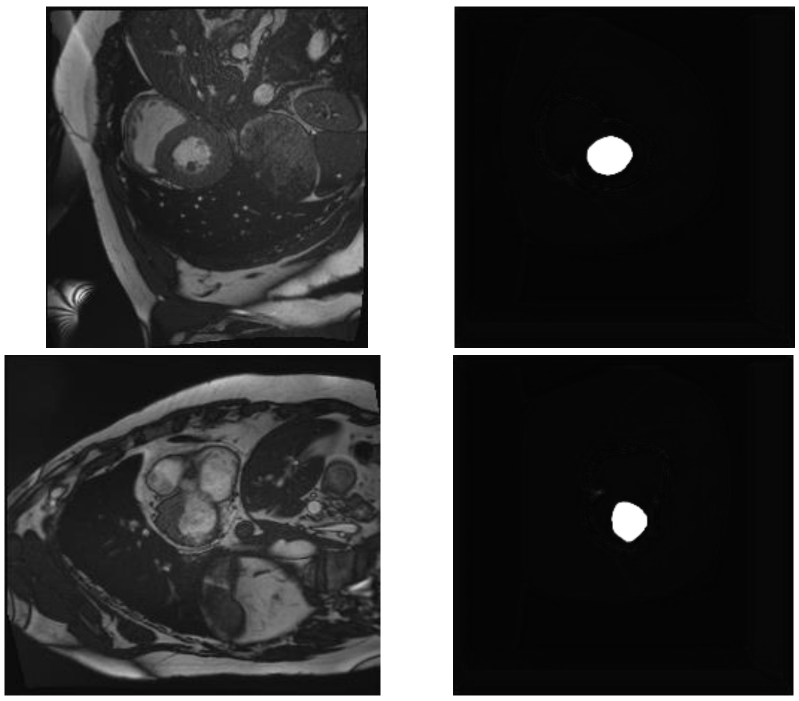


Figure 13 result of segmentation for modification 3

Here, it is evident that even without utilizing threshold segmentation, there are fewer instances of misidentification and the boundaries appear significantly clearer.

3.3 Feature extraction (dataprocess_train/dataprocess_test)

To compute medical metrics, we must determine the volumes of the left ventricle, right ventricle, and myocardium. To achieve this, we acquire the respective mask slices based on the provided labels. By multiplying the area of each slice with the interslice distance, we sum these small "micro-elements" to approximate the volume. Since we are sectioning the original 3D data based on the z-axis coordinates, we can assume a uniform interslice distance of 1 unit.

Therefore, in both notebooks, we first obtain the mask images of the right ventricle and the myocardial part. It is worth noting that the mask image of the left ventricle has already been obtained in the segmentation section.

Following the above explanation, the individual metrics were calculated:

First, we calculate the volumes of the left ventricle (LV) during the end-diastole (ED) and end-systole (ES) periods. Similarly, we calculate the volumes of the right ventricle (RV) during the ED and ES periods, as well as the volume of the myocardial fraction during the ED and ES sections. Next, using the calculated volumes, we compute various ratios, including the RV/LV ratio during the ED and ES periods, the Mc/LV ratio, and the LV and RV ejection fractions during the ED and ES periods. We then write these characteristics, along with the provided height and weight data, into a CSV file. This results in a feature vector of dimension (100, 14).

Table 1 Medical metrics of Patient (Id = 9)

Parameters	Value
Height/cm	153
Weight/kg	61
Volume_ED_LV	14190
Volume_ES_LV	12062
Volume_ED_RV	4429
Volume_ES_RV	3315
Volume_ED_Mc	10065
Volume_ES_Mc	11136
EF_LV	0.149753
EF_RV	0.251524
Ratio_RV_LV_ED	0.312121
Ratio_RV_LV_ES	0.274762
Ratio_Mc_LV_ED	0.709302
Ratio_Mc_LV_ES	0.274762

3.4 Classification (classification.ipynb)

Here we also tried two approaches, which will be analyzed specifically at the experimental level below.

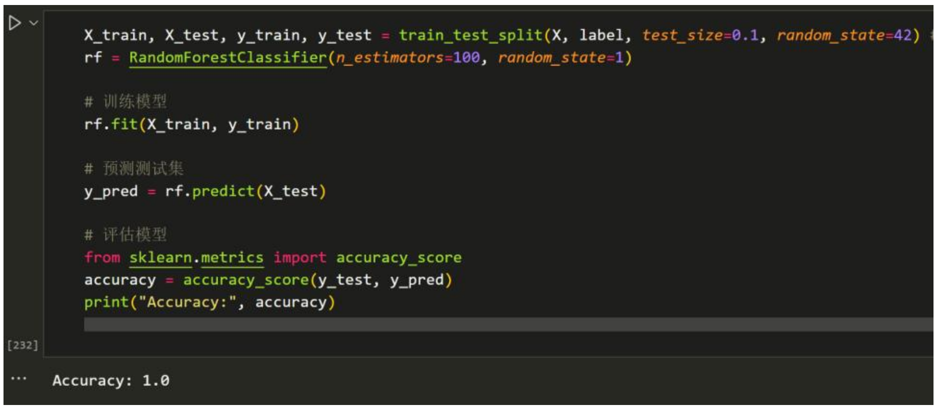
3.4.1 Random Forest Classifier

In this approach, we divided the train and test datasets in a 9:1 ratio. A Random Forest

Classifier with 100 decision trees was designed [1]. After training, the classifier achieved an accuracy of 1.0.

However, when evaluated on our test set, the accuracy achieved by the model is only 0.4666, which is below the desired threshold of 0.5. This low accuracy suggests that the random forest classifier, being better suited for larger datasets, is not an optimal choice for our current dataset of only 100 samples. Additionally, since our input data consists of calculated medical features rather than images, common data augmentation techniques used for images are not applicable in this context. Hence, the random forest classifier may not provide the best solution for our task.

The experimental result is shown in the following figure:



```
X_train, X_test, y_train, y_test = train_test_split(X, label, test_size=0.1, random_state=42)
rf = RandomForestClassifier(n_estimators=100, random_state=1)

# 训练模型
rf.fit(X_train, y_train)

# 预测测试集
y_pred = rf.predict(X_test)

# 评估模型
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 1.0
```

Figure 14 the result of Random Forest Classifier

3.4.2 Multi-Layer Perceptron

The Dense net architecture for constructing the MLP network is introduced. Reducing the number of channels is not always an optimal approach, especially when dealing with limited data. Significantly reducing the number of channels may result in the model's inability to capture sufficient features, leading to underfitting. Moreover, reducing the number of channels can potentially impact the model's performance and accuracy since a higher number of channels can provide more informative representations. However, an excessive number of channels can lead to an overly complex network, resulting in overfitting. To address this, a network consisting of four hidden layers followed by a single-layer output network is designed. The initial input layer integrates the number of channels to 128, which is then reduced to 64. Subsequently, three fully connected layers with 64 channels are connected, utilizing the rectified linear unit (ReLU) function as the activation. Finally, the last layer integrates the number of channels to 5 and employs the softmax activation function to output the probabilities of the patient belonging to each category.

Using the Adam optimizer with a learning rate of 0.0001, the loss function chosen as categorical_crossentropy, a batch size of 8, and 500 training epochs, we monitored the variations in the loss function and accuracy. The progress of the loss function and accuracy metric is presented below:

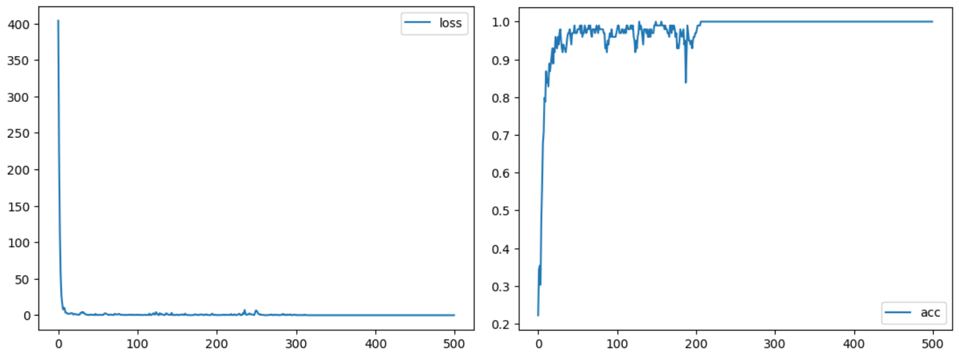


Figure 15 loss value and accuracy of this network

The test dataset tested in this case obtained a comparatively high accuracy on Kaggle. The prediction result is [2, 4, 0, 3, 4, 1, 0, 0, 4, 4, 3, 3, 4, 4, 0, 4, 2, 3, 1, 0, 4, 0, 3, 3, 1, 2, 1, 4, 2, 4, 1, 4, 3, 2, 0, 3, 1, 3, 1, 0, 1, 1, 1, 2, 4, 4, 0, 1, 1, 1].

Since there was no fixed random seed previously, the results of the prediction generated during further examination of the code were slightly different from the version submitted on the leaderboard on Kaggle, but these two predictions were verified by Kaggle to have the same accuracy.

4 Conclusion

In this segmentation task, we utilized the UNet architecture to perform the segmentation of the left ventricle, while patient classification was accomplished using MLP. The overall outcome yielded a high accuracy rate. However, it should be noted that the current classification network lacks robustness. The classification results exhibit significant variability when the classification network is altered. In conclusion, our results suggest that further improvements can be made to enhance the stability and reliability of the classification network. Future efforts may focus on addressing the limitations of our current approach and exploring alternative methods to achieve better performance.

References

[1] Wolterink, J. M., Leiner, T., Viergever, M. A., & Išgum, I. (2018). Automatic segmentation and disease classification using cardiac cine MR images. In Statistical Atlases and Computational Models of the Heart. ACDC and MMWHS Challenges: 8th International Workshop, STACOM 2017, Held in Conjunction with MICCAI 2017, Quebec City, Canada, September 10-14, 2017, Revised Selected Papers 8 (pp. 101-110). Springer International Publishing.

[2] Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical

image segmentation. In Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18 (pp. 234-241). Springer International Publishing.

[3] Huang, G., Chen, D., Li, T., Wu, F., Van Der Maaten, L., & Weinberger, K. Q. (2017). Multi-scale dense networks for resource efficient image classification. arXiv preprint arXiv:1703.09844.

[4] Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. Journal of big data, 6(1), 1-48.

[5] 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 Statistical Atlases and Computational Models of the Heart. ACDC and MMWHS Challenges: 8th International Workshop, STACOM 2017, Held in Conjunction with MICCAI 2017, Quebec City, Canada, September 10-14, 2017, Revised Selected Papers 8 (pp. 140-151). Springer International Publishing.