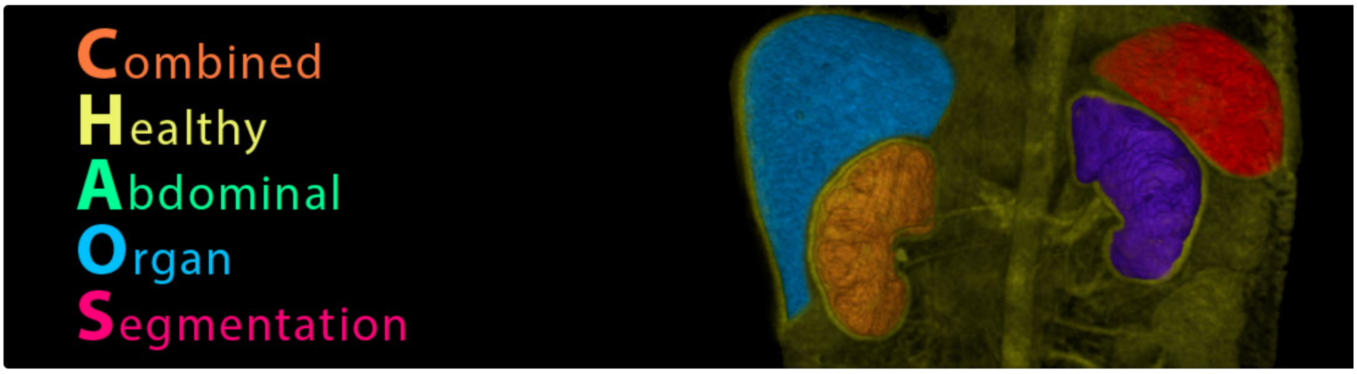
FINAL PROJECT: CHAOS

Liver Segmentation



Group 2 - Deep Learning Methods for Medical Image Analysis

Mariona Carrasco and Gerard Castells

1. Introduction

In this course project, the aim is to put our acquired knowledge into action by developing a precise segmentation algorithm. The focus is on tackling the liver segmentation task within the CHAOS (Combined Healthy Abdominal Organ Segmentation) challenge, using CT (computed tomography) images. The objective is to construct a reliable segmentation model, incorporating the methods gleaned from deep learning and medical image analysis courses.

2. Design and Results

2.1 Data Processing

In this challenge, we have a dataset consisting of medical imaging data from 20 patients. Each patient's data includes 16-bit DICOM images with a resolution of 512x512 pixels. These images have a spacing between 0.7-0.8 mm in the x-y direction and an inter-slice distance ranging from 3 to 3.2 mm, representing different CT slices. Additionally, for each slice, there is a corresponding mask in .jpg format.

The CT images are stored in DICOM files, while the masks are in .jpg format. The first part of our code is responsible for loading these images and masks, and then organizing them into a dictionary named "patient_data." In this dictionary, the keys represent patient identifiers, and the values consist of lists containing pairs of (DICOM image, mask) converted into NumPy arrays for each patient. This approach ensures that slices from the same patient are not included in both the training and validation datasets.

To create the training, validation, and testing sets, we use dictionaries. We randomly shuffle the patient identifiers and allocate them into sets: 15 for training, 4 for validation, and 1 for testing. Finally, we generate lists of image and mask pairs from these three dictionaries, which are subsequently utilized for various image segmentation tasks.

Ensuring the success of this process relies on arranging patient data sequentially, with each set representing slices from a CT scan. Using the 'sorted()' function is crucial to maintain consistency, aligning both images and masks in the exact sequence. This ensures that the slices are correctly ordered, a crucial factor for accurately calculating the dice score for each patient at a later stage.

2.2 Model and training

For this challenge, we employed a pre-existing U-Net model whose parameters were fine-tuned through a trial-and-error approach to achieve the optimal learning curve. Once the model was implemented, we compiled it using the Adam optimizer and assessed its performance using metrics like binary accuracy, precision, recall, and the dice coefficient. About the loss we have trained the model with three different losses: Binary cross-entropy, Dice loss and Intersection over Union (IoU) loss. Training a model with various loss functions allows it to learn different aspects of the data and optimize towards different objectives.

Each of these loss functions emphasizes different aspects of the learning task. For instance, while binary cross-entropy focuses on correct classification probabilities, Dice and IoU losses specifically target accurate segmentation and spatial overlap between predicted and true regions in an image.

The main achievement goal and the task requirements will determine which loss function is used. In this instance, the aim is to improve the model so that it produces masks that closely match the real mask. Therefore, the model's performance is measured by its Dice score, which compares the generated mask to the true mask to determine how well it performed.

As a consequence, we used three separate losses with the same parameters to train the model. The average dice scores across all patients were as follows: IoU loss = 0.933, dice loss = 0.928, and binary cross-entropy = 0.908. These results led us to conclude that the IoU loss is most in line with our goals and is most likely to produce better outcomes.

The next step that we did was apply this IoU loss over a larger number of epochs in order to improve the training's accuracy and value. We then went ahead and trained the model for 400 epochs while keeping the identical parameters.

Subsequently, the "data generator" function is applied to the lists of training and validation data pairs generated earlier. This function generates batches of augmented images and masks, normalizes them, expands their dimensions, and applies data augmentation.

Finally, the model is trained using the previously obtained data pairs and evaluated using the "plotcurve" function, which generates two plots for learning curves (loss and accuracy) and one for evaluation metrics.

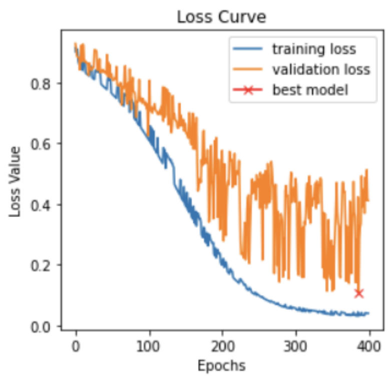


Figure 1: Loss curve

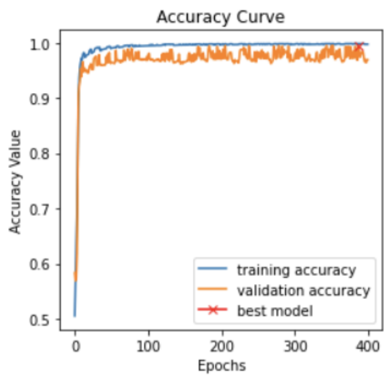


Figure 2: Accuracy curve

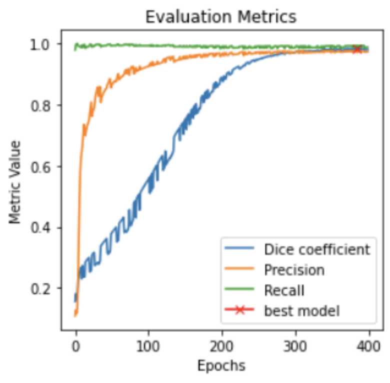


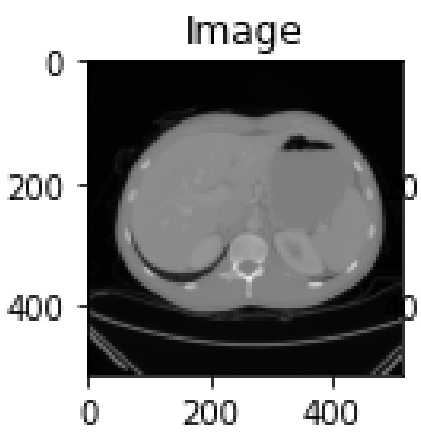
Figure 3: Evaluation metrics

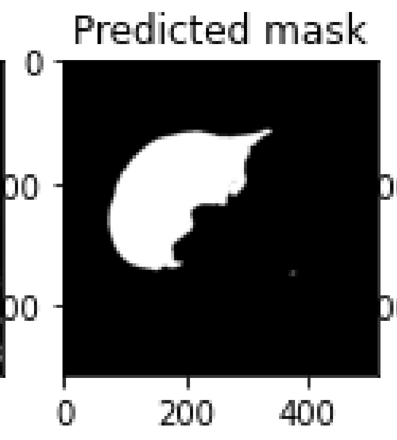
The values obtained from the loss and accuracy curves indicate that the model has been well trained, with low values around 0 for training and validation loss and close to 1 for accuracy. Also, analyzing the evaluation metrics Recall and Precision, which have obtained values really close to 1, indicate that the model has been well-trained because it captures relevant instances of the positive class and every positive prediction made by the model is correct. Finally, observing a Dice coefficient of practically 1, indicates a very high level of performance, considering that 1 represents perfection. In many cases, a value higher than 0.6 is generally considered satisfactory.

2.3 Predicted masks

To generate predicted segmentation masks using the pre-trained model, we will follow a series of steps. First, we'll save the trained model, and subsequently, we'll load it for inference. Then, we will take the dataset and divide it into two lists: the images and the corresponding ground truth masks.

The images will be fed into the model, slice by slice, to generate predicted masks for each of them. These predicted masks will be saved in a designated directory, along with the corresponding true masks for each slice. This way, we will have both the model's predictions and the actual masks readily accessible for further analysis.





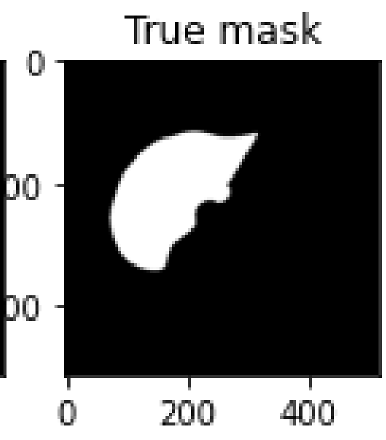


Figure 4: Image

Figure 5 : Predicted mask

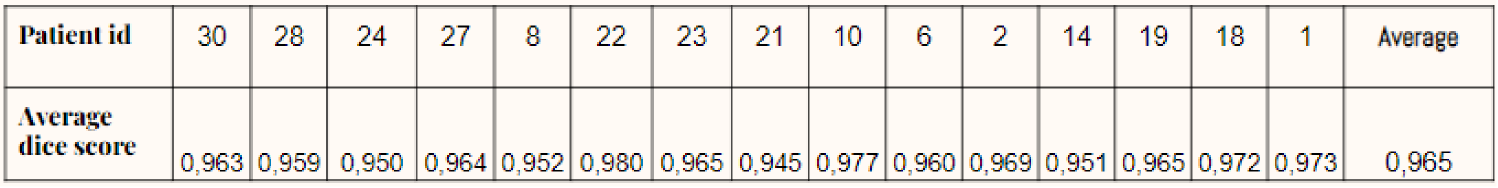
Figure 6: True mask

The comparative image showcases the segmentation image on the left (Figure 4), the predicted mask (Image 5), and the actual mask on the right (Image 6). It demonstrates liver segmentation utilizing a binary mask for an image slice from the test data patient. Upon close examination and comparison of the two masks, their strikingly close similarity is confirmed by a dice score of 0.957 for this specific mask.

2.4 Dice score

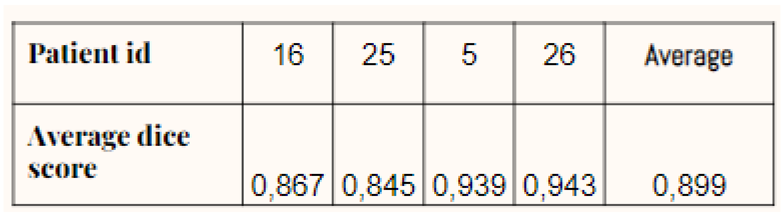
To compute the Dice score for each patient, we will create a dictionary named "patient_dice_scores." This dictionary will store the average Dice score for all the slices belonging to each patient. The Dice score is calculated by comparing the predicted mask generated by the model with the ground truth mask.

To arrive at the average Dice score for each patient, we will first calculate the Dice score for each slice within a patient's dataset. Subsequently, we will compute the mean of these individual slice scores and associate this average with the respective patient's identifier. This approach allows us to assess the overall segmentation performance for each patient in a clear and organized manner.



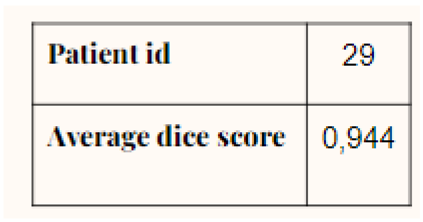
Patient id	30	28	24	27	8	22	23	21	10	6	2	14	19	18	1	Average
Average dice score	0,963	0,959	0,950	0,964	0,952	0,980	0,965	0,945	0,977	0,960	0,969	0,951	0,965	0,972	0,973	0,965

Table 1: Dice scores train data



Patient id	16	25	5	26	Average
Average dice score	0,867	0,845	0,939	0,943	0,899

Table 2: Dice scores validation data



Patient id	29
Average dice score	0,944

Table 3: Dice scores test data

The average of Dice scores for each patient is presented in Table 1,2 and 3. Upon calculating the median, an overall average of 0,936 is obtained, suggesting an elevated level of performance. It is important to note that a score of 1 represents perfection. The particular test patients attained an average score of 0.944 across all slices, signaling a well-trained model upon implementation with new data.

3. Conclusion

In conclusion, this project has served as a valuable application of the knowledge acquired throughout the course. It not only provided a real-world example through the CHAOS challenge but also presented an opportunity to navigate and learn from the encountered challenges. While the implemented U-Net model showcased commendable performance, there's acknowledgment of the potential for improvement. To enhance the predictive accuracy of masks, future endeavors could explore employing a high-performance model and expanding the training dataset with a larger volume of CT images.

In the context of liver segmentation, where precise organ localization is crucial due to its consistent anatomical location, the preference for IoU loss and its resulting higher Dice scores becomes even more apparent. IoU loss emphasizes accurate object localization by measuring the overlap between predicted and ground truth liver masks. Given the liver's stable and predictable position within the body, this focus on precise alignment incentivizes the model to precisely delineate the liver boundaries, aligning with the organ's anatomical structure. Additionally, IoU's sensitivity to region overlap proves advantageous for delineating the liver's boundaries accurately, particularly in scenarios with varying tissue intensities or complex shapes within the organ. These characteristics of IoU loss align well with the demands of liver segmentation, resulting in superior performance, as evidenced by the higher Dice scores attained.

Even with certain obstacles that could be solved with other operations or modifications, the final result is a significant achievement and we were successful in properly training a model. Nevertheless, despite the hurdles, we are deeply satisfied with the overall outcomes, appreciating the practical insights gained not only from the project itself but also from the entire course experience.