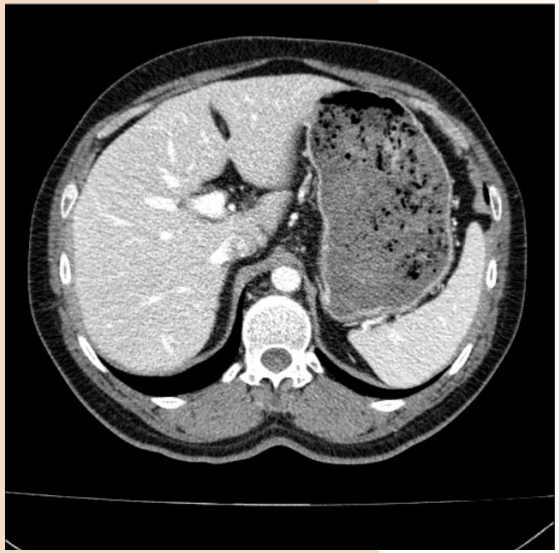
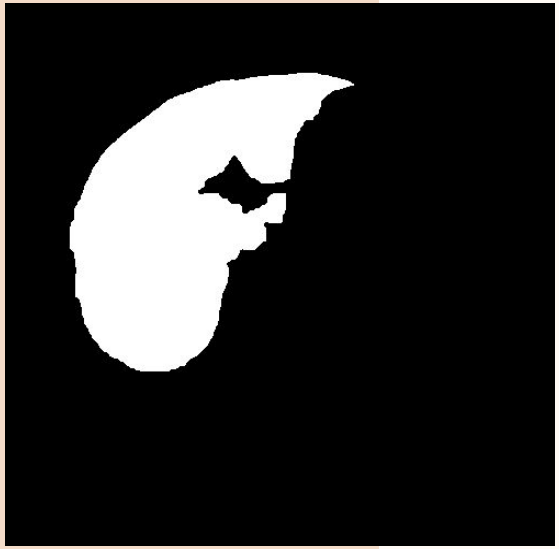
FINAL PROJECT: CHAOS

Liver Segmentation

Group 2 - Deep Learning Methods for Medical Image Analysis
Mariona Carrasco and Gerard Castells





Steps followed to solve the challenge

1. Analyzing the data to work with
2. Data loading
3. Data generator function
4. Training the model
5. Obtaining the predicted masks
6. Calculating the dice scores

Data structure

- Data
  - Train_sets
    - CT
      - 1
        - **DICOM_anon:** *.dcm files* containing the CT images for all the slices
        - **Ground:** *.jpg files* containing the masks for each slice
      - 2
        - DICOM_anon
        - Ground
      - 3
        - …
        - …
      - 4
      - …
    - MR
  - Test_sets

- **20 patients in Train_sets**
  - **15 train, 4 validation, 1 test**
- **Around 90 slices per patient.**

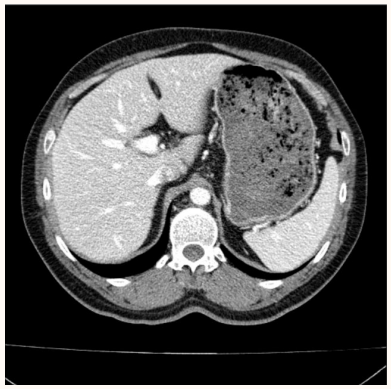


Figure 1: Dicom_anon file

Figure 2: Ground file

Data loading

Images: *.dcm files*

1. dcmread
2. pixel_array
3. apply_modality_LUT

np.unique(dicom_array)
array([-1024., -1023., -1022., ..., 1183., 1187., 1211.])

Masks: *.png files*

1. Image.open
2. Array

np.unique(ground_array)
array([False, True])

patient_data = {'patient id' : [image,mask] for all the slices, 'patient id' : [image,mask], ...}

- 15 patients to train_data
- 4 patients to val_data
- 1 patient to test _data

Dictionary to list

- train_pairs
- val_pairs
- test_pairs

Data generator function

- Data augmentation

  Parameters:
  ImageDataGenerator(rotation_range=5, fill_mode='nearest')

- Image normalization
  - Dividing each image array by 1400
    - To have all values in the range [-1,+1]
- Mask normalization
  - Make sure masks are all boolean (True/False)

Training the model

- ➢ Implement U-Net Model
- ➢ Compile Model
  - Adam optimizer
  - Binary cross-entropy loss
  - Evaluation metrics: binary accuracy, precision, recall and dice coefficient
- ➢ Training
  - Data obtained with data generator function (training / validation)
  - num_epochs = 150
  - batch_size = 8

Parameters used:

base = 16
img_h = 512
img_w = 512
img_ch = 1
learning_rate = 1e-5
dropout = True
dr = 0.2
batch_norm = True
img_size= 512

Plot curves

Loss curve

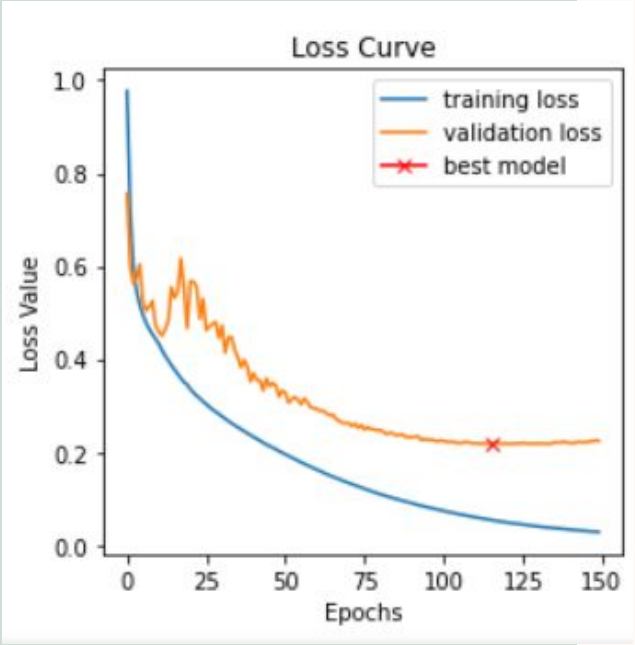


Figure 3: Graph of loss curve

Accuracy curve

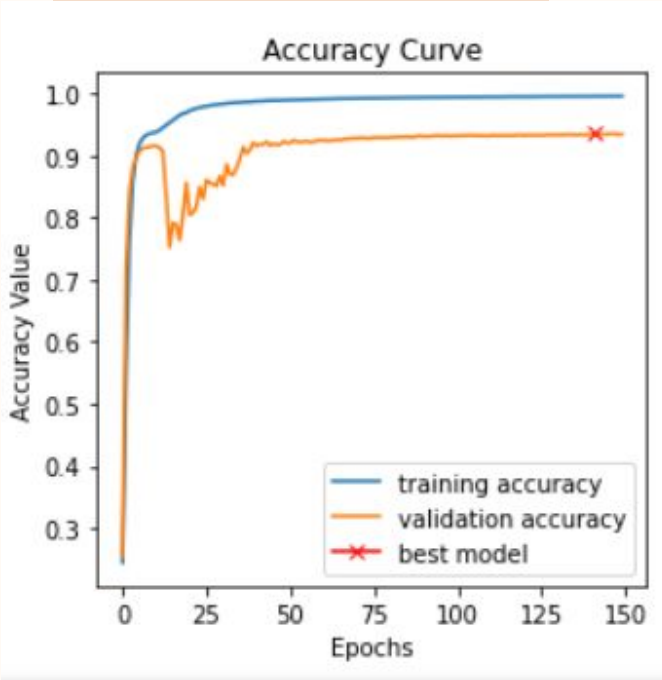


Figure 3: Graph of accuracy curve

Evaluation metrics

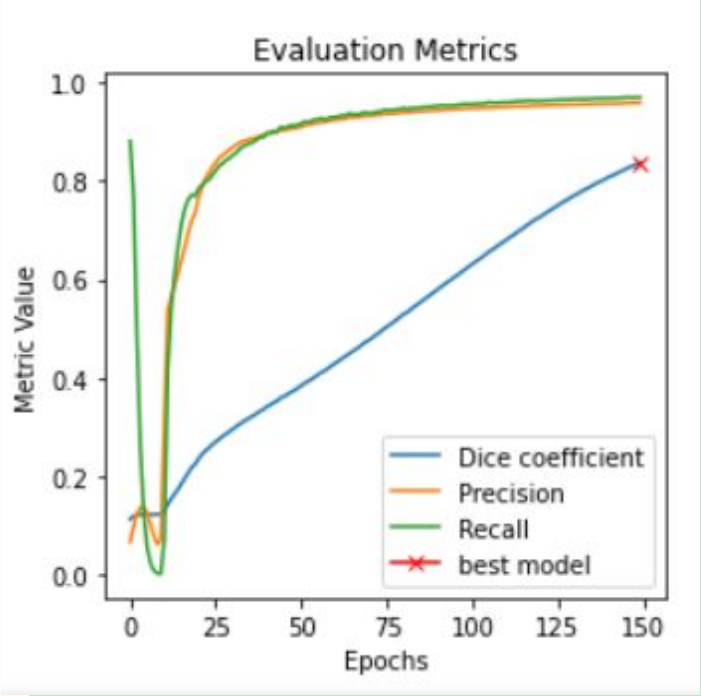


Figure 3: Graph with metrics

Obtaining the predicted masks

test_pairs =*{'patient id' : [image,mask]...}*

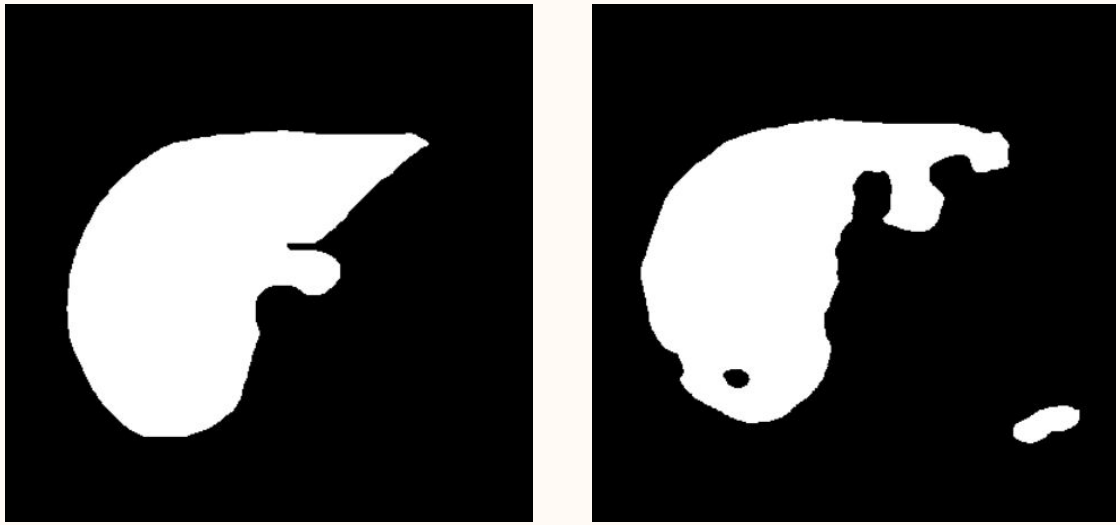
Dictionary to list

- test_images
- test_masks

test_images → model.predict → Binary → Predicted mask

threshold=128

True Masks VS Predicted Masks



Calculating the dice scores

Function → calculate_dice_coefficient(true_mask, predicted_mask_binary)

patient_dice_scores ={*'patient id' : [dice_score]...}*

average_patient_dice_scores=*{'patient id' : [average_score]...}*

Patient id	30	28	24	27	8	22	23	21	10
Average dice score	0.285	0.588	0.351	0.355	0.576	0.339	0.239	0.496	0.512

Total average= 0.43

6	2	14	19	18	1	16	25	5	26	29
0.485	0.42	0.549	0.595	0.468	0.56	0.18	0.552	0.418	0.326	0.567

Thanks!

Do you have any questions?