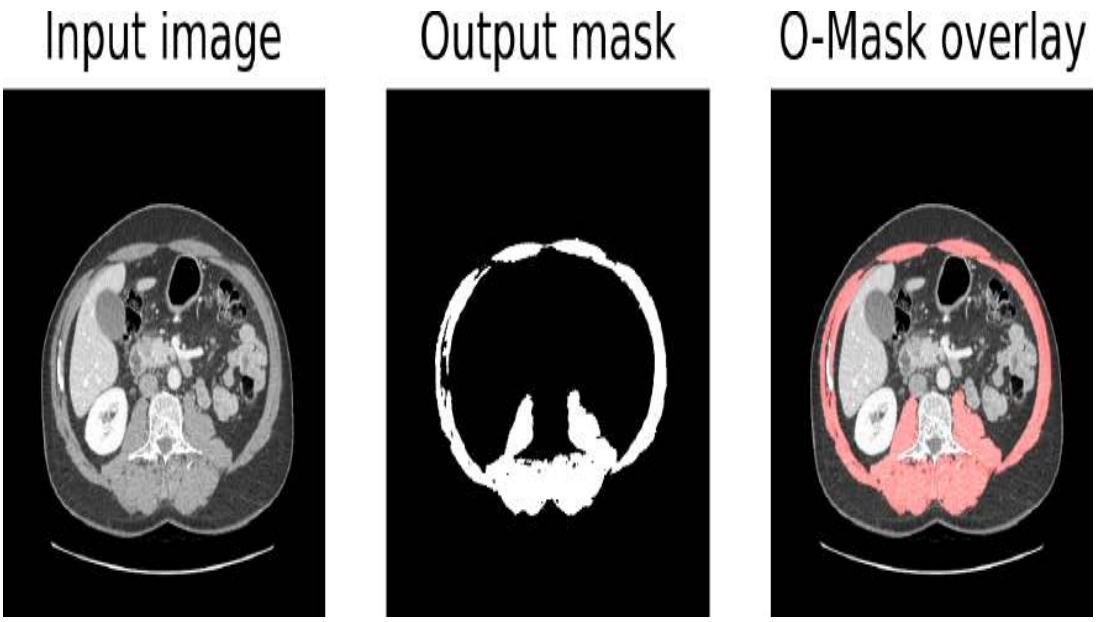
Segmentation and measurements of skeletal muscle cross-sectional area on computed tomography (CT) scans

Estimation of sarcopenia on different Lumbar Vertebrae levels (L3-L2-L1).



Aissam Djahnine

Summary

1 Segmentation and measurements of skeletal muscle cross-sectional area on computed tomography (CT) scans **3**

2 Dataset **4**
2.1 Evolution of L3 dataset 4
2.2 Evolution of L3-L2-L1 dataset 4

3 Training details **5**
3.1 Architecture 5
3.2 Training 6
3.2.1 Preprocessing 6

4 Improvements **6**
4.1 Data augmentation 6
4.1.1 Laplacian Filter : Hard Filter 6
4.1.2 Rotation 7
4.1.3 Artificial added hands 8
4.2 Post-Processing 8
4.2.1 Ribs removal 9

5 Continual Image Adding : Database Augmentation **11**

6 Experiments and results **11**

7 Conclusion **13**

2 Dataset

- We deal with 512 x 512 axial slices dataset at L3, L2 and L1 vertebrae level. Table 1 summarizes the evolution of the dataset after all the data augmentations and experiments. Details of this evolution are found in section 5 and sub-section 4.1.3.
- "Initial" column describes the images we had in the beginning at every level. Note that we didn't dispose of L2/L1 images-masks pairs while we had them at L3.
  A first segmentation quality evaluation was done before starting the project on the available masks and were classified into ACCEPTED/ERROR samples and we wanted to add as much as possible of errors images to accepted ones.
- "Final training" column describes the final images-masks pairs ,"Error remaining" column shows possible images to add in the future and "Augmentation" column for the images added with artificial hands (see 4.1.3).

	Initial	Error found	Augmentation	Total (with error)
L3	1684	519	0	2203
L2	2598	0	0	2598
L1	3266	0	0	3266
Total	7548	519	0	8067
	final training	Error remaining	Augmentation	Total
L3	1786	417	88	2291
L2	1559	1039	0	2598
L1	1039	2227	0	3266
Total	4472	3683	88	8155

Table 1. Dataset description. The table represents the total images before and after the different data augmentation techniques progressively applied.

2.1 Evolution of L3 dataset

At L3 level, we were focused on improving the segmentation results following two data augmentation techniques shown in figure 2 and which we will explain in details in next sections.

- Let "Removing artifact" denote the process of removing unwanted objects from the samples and considered "accepted" to be added after a training process.
- We use "Artificial hands" to denote process of adding artificial hands to some samples.

2.2 Evolution of L3-L2-L1 dataset

At this stage, we augment the size of our dataset using the models trained on L3 by generating masks for L2/L1 images. This process called : "Continual adding" (refer to section 5 is iterative based on repetitive training-evaluation on L1/L2 samples. Figure 2 shows in details the evolution of the dataset starting with L3 only samples.

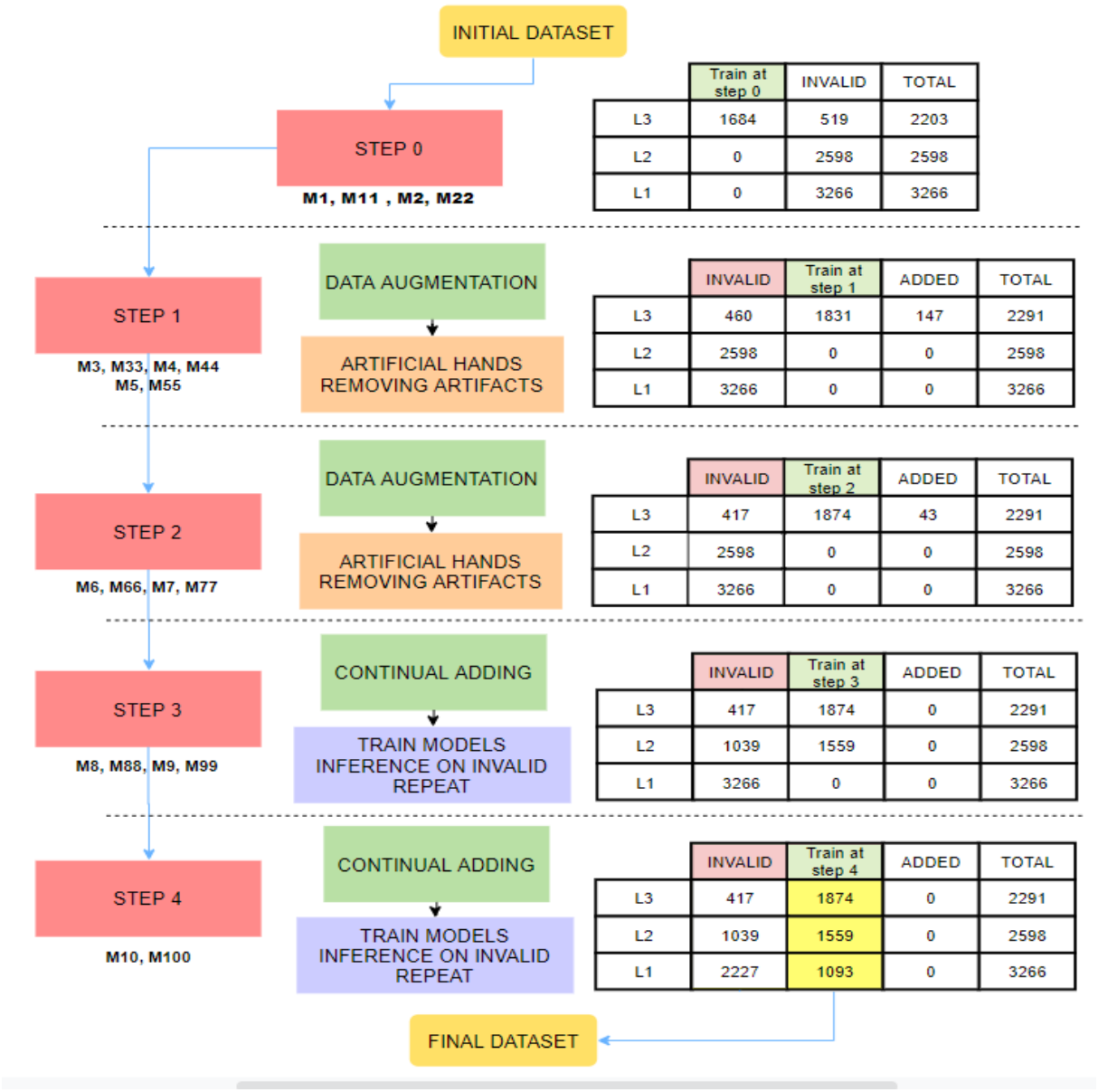


Figure 2. Dataset evolution diagram for L3,L2 and L1 via all data augmentation applied and continual adding. Under each step, we denote by M1,M11,M2,M22..etc top-2 final models for the experience using the actual dataset, refer to section 6 for details.

3 Training details

This section summarizes the technical details about building the network, hyperparameters selection and data preprocessing.

3.1 Architecture

- The network architecture used is a classic UNet with input images of 512 x 512 size. It consists of a encoder of 6 stages (from 16 feature maps to 512) and a decoder. The encoder follows the typical architecture of a convolutional network. It consists of the repeated application of two 3x3 convolutions, each followed by a ReLU activation and a 2x2 max pooling operation with stride 2 for downsampling while we double the

number of feature channels.

- Every step in the decoder consists of an upsampling of the feature map followed by a 2x2 de-convolution that halves the number of feature channels, a concatenation with the correspondingly cropped feature map from the encoder, and two 3x3 convolutions, each followed by a ReLU. [1]

3.2 Training

- We train the network for 30 epochs with a train:valid split ratio of 90:10 and for [2,4,6,8,16,32] batch size values. The network weights are optimized by RMSProp optimizer, with an initial learning rate of 0.001 (annealed down to zero following a Plateau schedule without restart), a momentum of 0.9, and a weight decay of 10^{-5}. The entire training process depends on the number of data used and batch size but requires between 3 and 5 hours on a single NVIDIA TITAN RTX GPU.

It's important to mention that no hyperparameters fine-tuning was done during all experiments.

3.2.1 Preprocessing

- The images are pre-processed before passing them to the model. We clip the 16 bits images in range of $[0,2000]HU$ and then normalize them between [-1,1]. Applying the Laplacian filter may change this range of values to between [-5,5].

4 Improvements

4.1 Data augmentation

4.1.1 Laplacian Filter : Hard Filter

- Since our dataset contains hard and soft filter images, we wanted that the model learns this change of texture by randomly applying a hard filter (Laplacian filter) to our input images. We wanted to apply this filter only to non-noisy images (i.e soft filter images). Therefore, we proceed by : first estimating the local noise variance in a given image, select them wrt a given threshold and then apply the filter on low noise variance examples.
- Local noise variance can be estimated in different ways, we chose a simple method (*IMMERKÆR, J. (1996)* that requires only the use of a 3x3 mask followed by a summation over the image or a local neighborhood. Equation 1 shows the method followed :

$$\sigma_n = \sqrt{\frac{\pi}{2}} \frac{1}{6(W-2)(H-2)} \sum_{imageI} |I(x,y) * N| \tag{1}$$

- After we estimate the local noise variance for all training images, we select the non-noisy images and apply the Laplacian filter on them following this equation :

$$I_{output} = I_{input} + k.I_{input} * F \tag{2}$$

with : F the filter kernel used, K a coefficient in $[0.8,1.2]$ range and I_{input} is the input Image. Figure 4 shows examples after applying the filter.

1. Olaf Ronneberger, Philipp Fischer et Thomas Brox.U-Net:Convolutional Networks for Biomedical Image Segmentation. 2015. arXiv :1505.04597

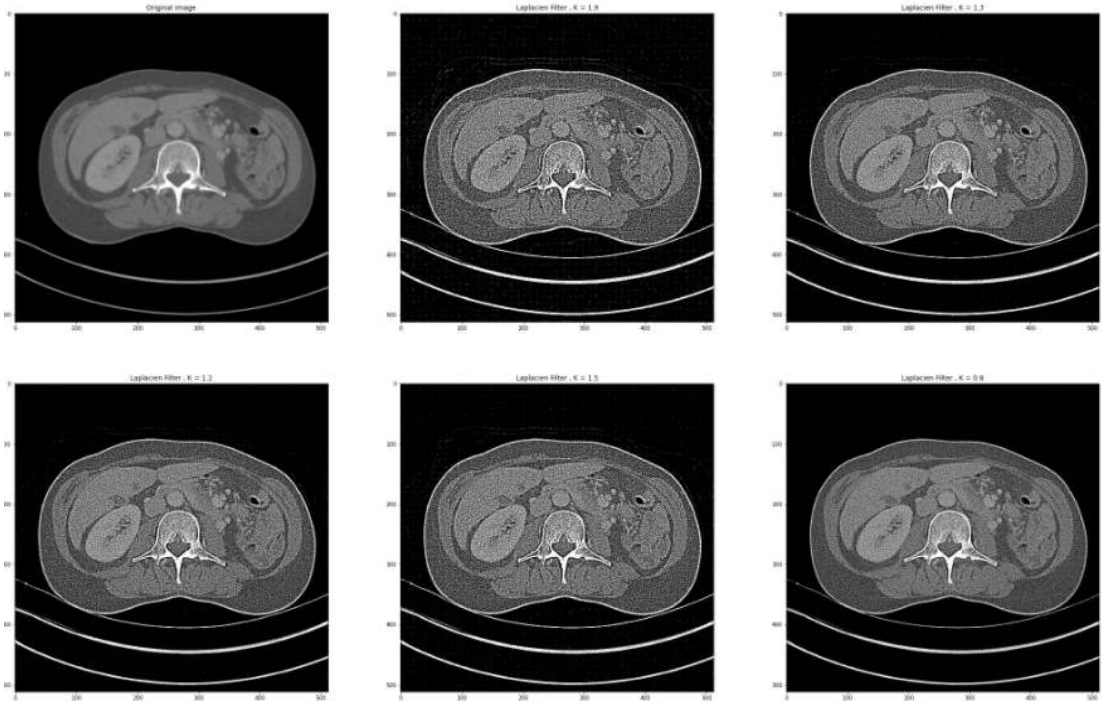


Figure 3. Hard filter samples with different coefficient k. Top-right presents a normal image with no filter applied while other samples shows the effect of Laplacien filter wrt to coefficient k.

4.1.2 Rotation

- Rotation is a common augmentation method. Since the patients in the scan exams can have a small rotation than usual. We decided to apply some random rotations in range of $[-30,30]$ degrees so that we cover all possible orientations.

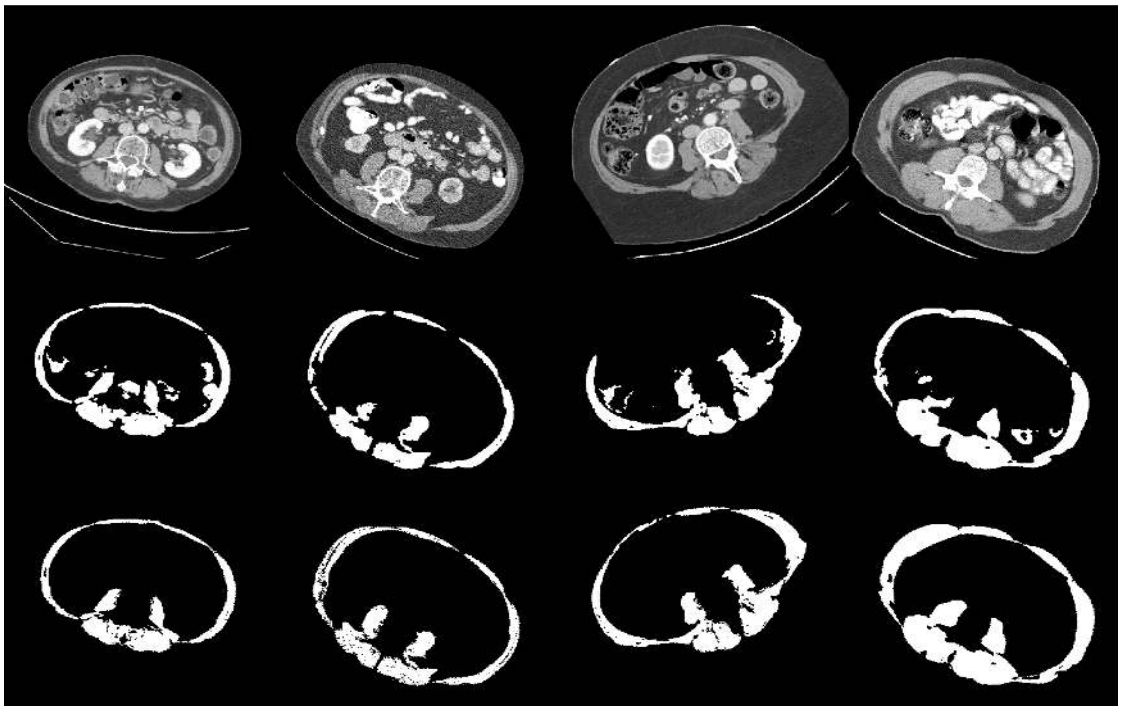


Figure 4. Rotation examples. Row (1) illustrates images while training, row(2) and row(3) show prediction and ground truth respectively.

4.1.3 Artificial added hands

- In order to augment our training dataset, we try to automatically correct error masks (apparent hands and objects in the scan). The model seems to be confused in the presence of them and it tends to segment them. One proposed solution is to remove these parts by hand. The solution is quite applicable but still time-consuming for just few easy cases.
- The other proposed solution is based on data augmentation mechanism by adding some artificial hands and objects to existed training images in a way that it reflects real cases. These artificial hands are extracted from the dataset itself and added in regions where they don't overlap with the image. We wanted to fully automate the process and add them dynamically while training but that would require and additional verification step so we decided to proceed in a static way. It means that we add them outside the training process in a random way and then check all the outputs and select the most satisfying ones then finally add them to the train set. Figure 5 illustrates the process we followed to perform this artificial hand adding.

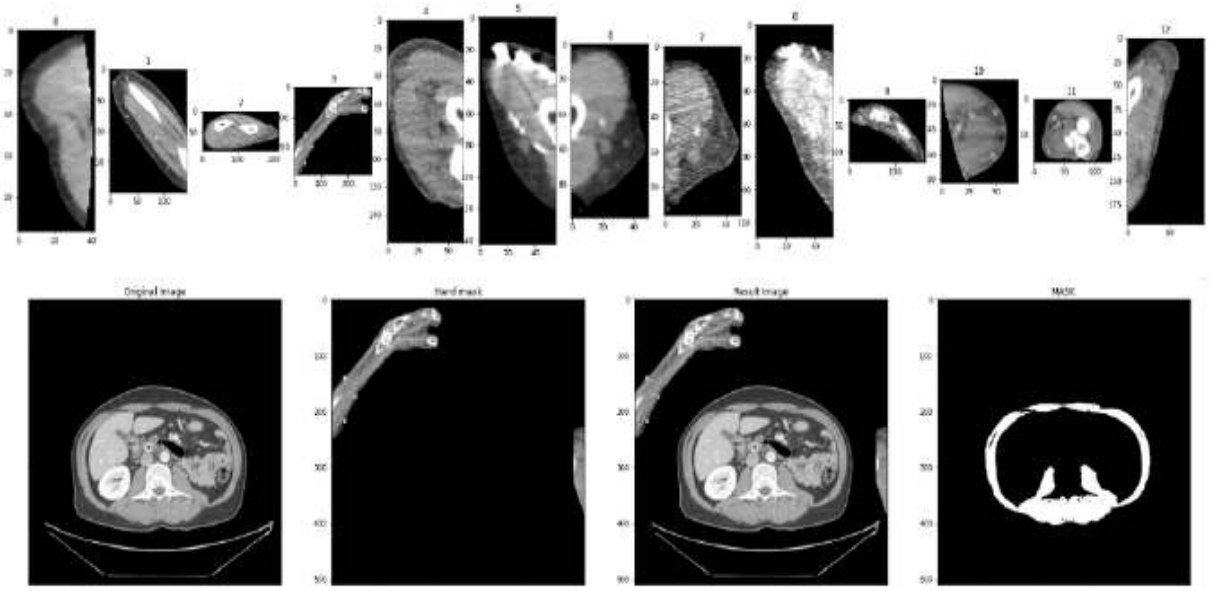


Figure 5. Adding artificial hands : row(1) shows hand masks extracted from dataset. row(2) shows an example of adding these hand masks to images.

4.2 Post-Processing

When training on L3 only dataset, we encountered several limitations such as : selecting ribs regions (mostly inside them) even though we deal with non ribs axial slices. These few observations made us anticipate the future issues we might face when dealing with L2 or L1 level since they are not ribs-free. Figure 6 shows some L2-L1 examples where the ribs were selected as part of the final segmentation.

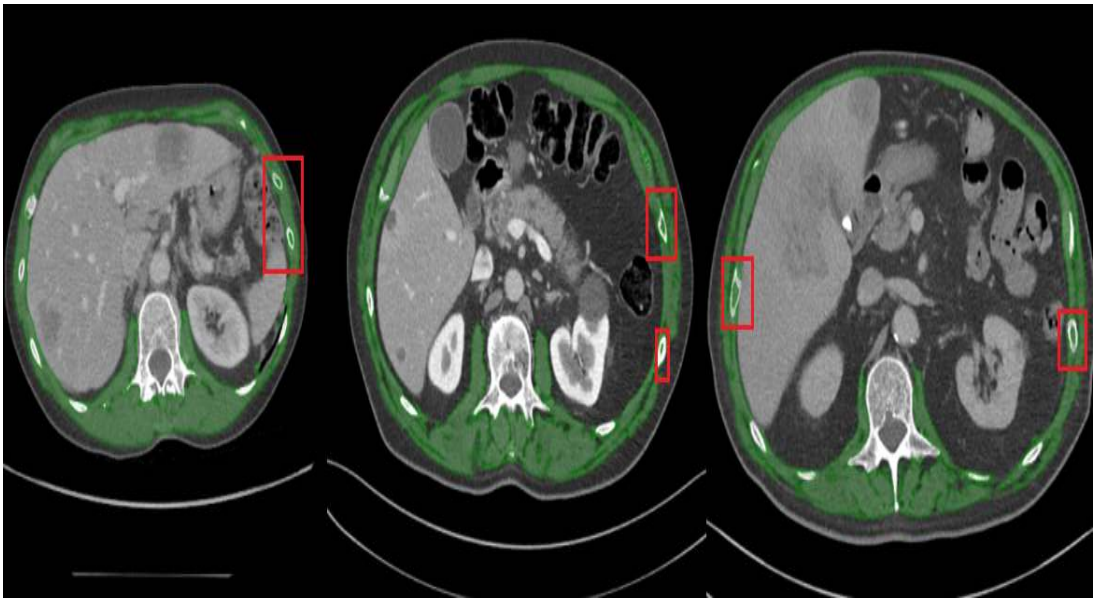


Figure 6. Examples of L1 overlay masks showing selected inside-ribs regions (red boxes).

For this matter, we suggested a ribs removal post-processing step in our pipeline. The next section describes in details the followed procedures.

4.2.1 Ribs removal

The two methods explained below serve only as final post-processing. One can note that final ribs-removed masks are re-used to train intermediate models (Section 5).

- **First method** : Using 3D-slicer software as shown in figure 7 with hand-select thresholds to only get ribs. We obtain at the end a binary ribs mask, we do a closing operation to close the holes and we subtract it from the generated mask and we repeat the process over all training images. This method is so time and effort consuming to be followed regarding the size of the samples we needed to process. A second method was proposed to face this problem instead.

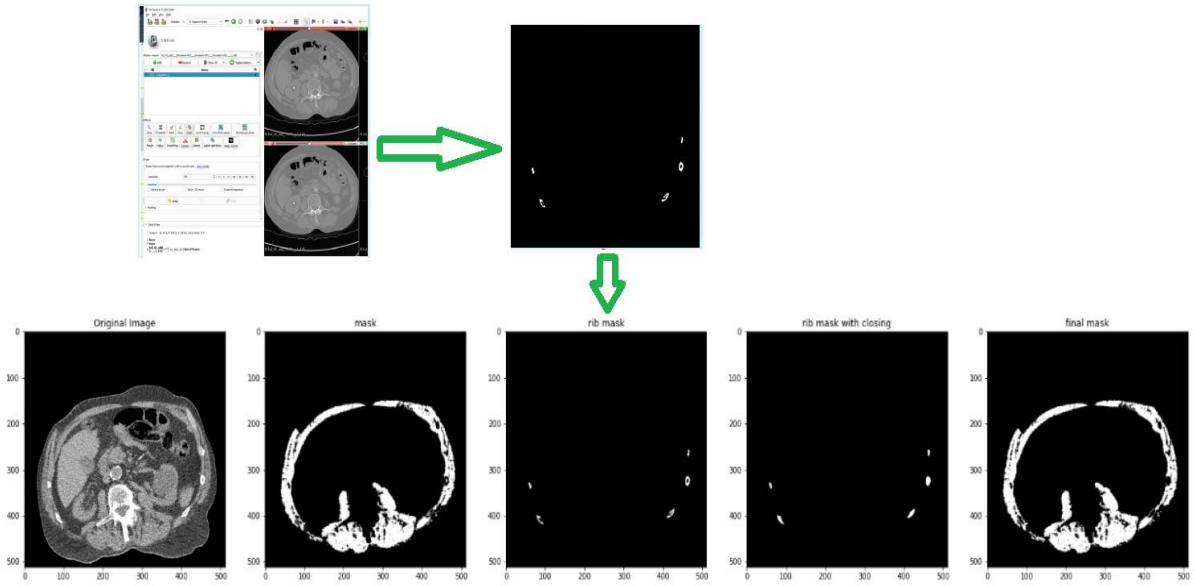
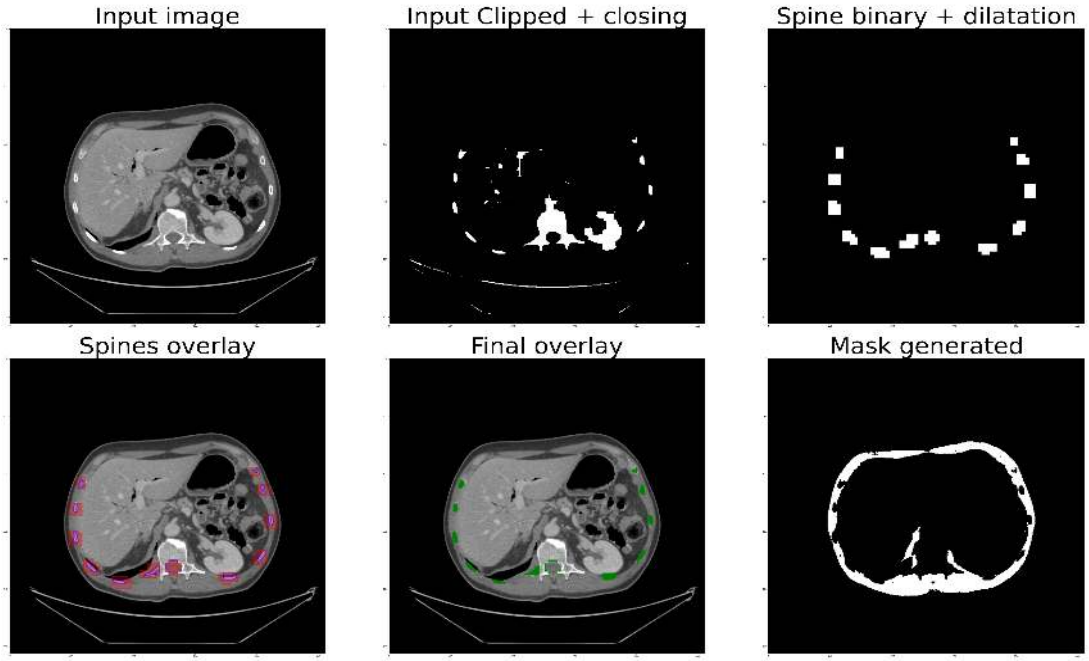
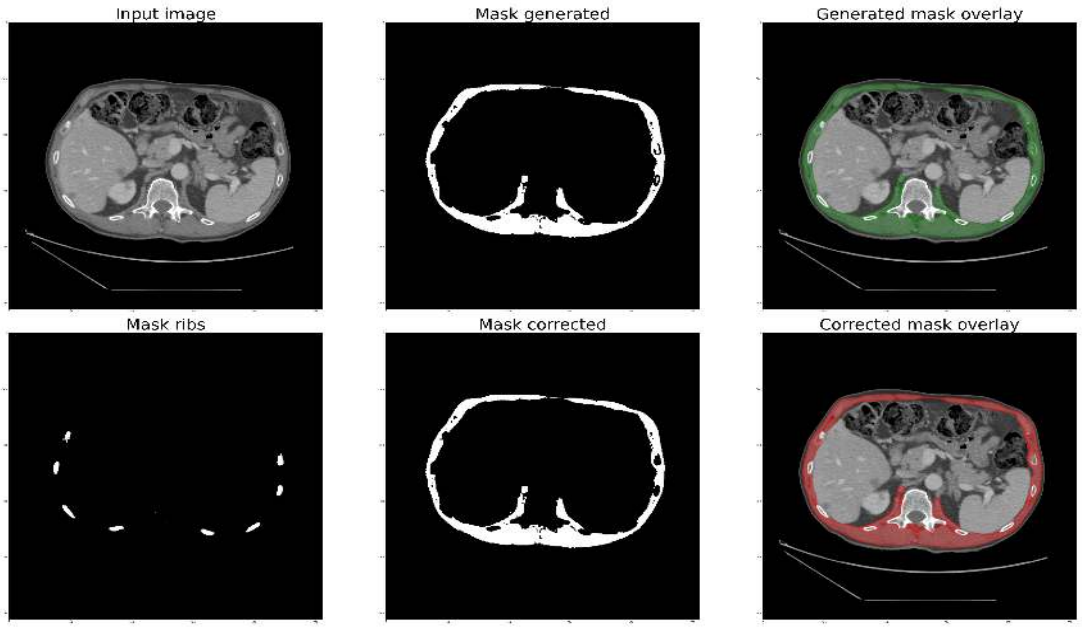


Figure 7. Method 1 : Ribs removal Hand-selected method.

- **Second method** : Using the coarse spine information provided by Medisys team [2] , we could remove the ribs from the generated masks. We proceed as follows : we set a threshold to select only bones, we see that we select the ribs and parts of the spine as well. We apply a closing operation to close holes in ribs borders. Besides, we use the dilated version of coarse spine segmentation to locate the ribs and then We take the intersection between the coarse segmentation and threshold-based segmentation to select the final rib mask. We get the final mask by rectifying the generated mask by subtracting the ribs mask. The method seems to work just fine on L1-L2 images.



(a) Steps followed to get the ribs mask to correct the generated mask.



(b) Ribs removal example. Green overlay represents the generated mask before removal while the red overlay is the final corrected mask after removing the ribs mask.

Figure 8. Method 2 : Automatic Ribs removal method.

2. Automatic segmentation of spine and ribs done on L2/L1 level axial slices.

5 Continual Image Adding : Database Augmentation

- Our initial training dataset contained only correct L3 level Images-masks pairs. But since the goal is to increase the size of this dataset and add new image-mask pairs (for L3,L2 and L1 levels), we had to continually add corrected samples (for L3) and new ones for L2-L1 by following close-to active learning approach (even though we believe that this procedure may add some bias). We train the model on the initial dataset, select top-2 models and then choose the best segmentation given for an image or we can reject them both and test them later with the next models then we repeat this process several times. This task of selecting/adding to dataset can be effort consuming if done in a ineffective way. Team medisys proposed a simple tool to make this task easier and both less time and effort consuming.
- As mentioned above in sub-section 2.2, we increased the size of our dataset following the continual adding. Table 2 summarizes the experiences we conducted with associated dataset samples number at the end of each one.

	Initial	Added	Artificial hands	total
L3	1684	102	88	1874
L2	0	1559	0	1559
L1	0	1039	0	1039
total	1684	2700	88	4472

Table 2. Final dataset statistics after continual adding. "Added" Column in red, show the total number of samples added in each level.

6 Experiments and results

In the past months, we conducted different experiences in order to increase the size of the final dataset. We tested various data augmentation and post-processing techniques. Table 3 shows details of each experience (data augmentation applied and others etc..). Figures 9, 10 show performance results on L3 only and L3-L2-L1 respectively.
These boxplots show the performance of our top-2 models from each experience on validation set. We denote (M=1 , M=11..) : two models from the same experience.

> One can obviously note that we can't compare, in a direct way, the results between the different experiences since the number of samples in valid and train set changes when we add new image-masks pair after each training.

Experience	Description
L3 Only	
M1, M11	`Hard Filter only`
M2, M22	`Hard Filter + Rotation`
M3, M33	`Hard Filter + Hand Masks(88)`
M4, M44	`Hard Filter + Hand Masks(88) + Error Added (59)`
M5, M55	`Hard Filter + Rotation + Error Added (59)`
M6, M66	`Hard Filter + Rotation + Hand Masks(88) + Error Added (102)`
M7, M77	`Hard Filter + Rotation + Hand Masks(88) + Error Added (102) + GN`^{3}
L3-L2-L3	
M8, M88	`Same as M7, M77 + L2 Added (1093)`
M9, M99	`Same as M7, M77 + L2 Added (1559)`
M10, M100	`Same as M7, M77 + L2 Added (1559) + L1 Added (1039)`

Table 3. Experiences Description and details. We report chronologically the experiences and the changes made with data augmentation techniques, number of samples in the dataset and which level(L3, L2 or/and) we train on.

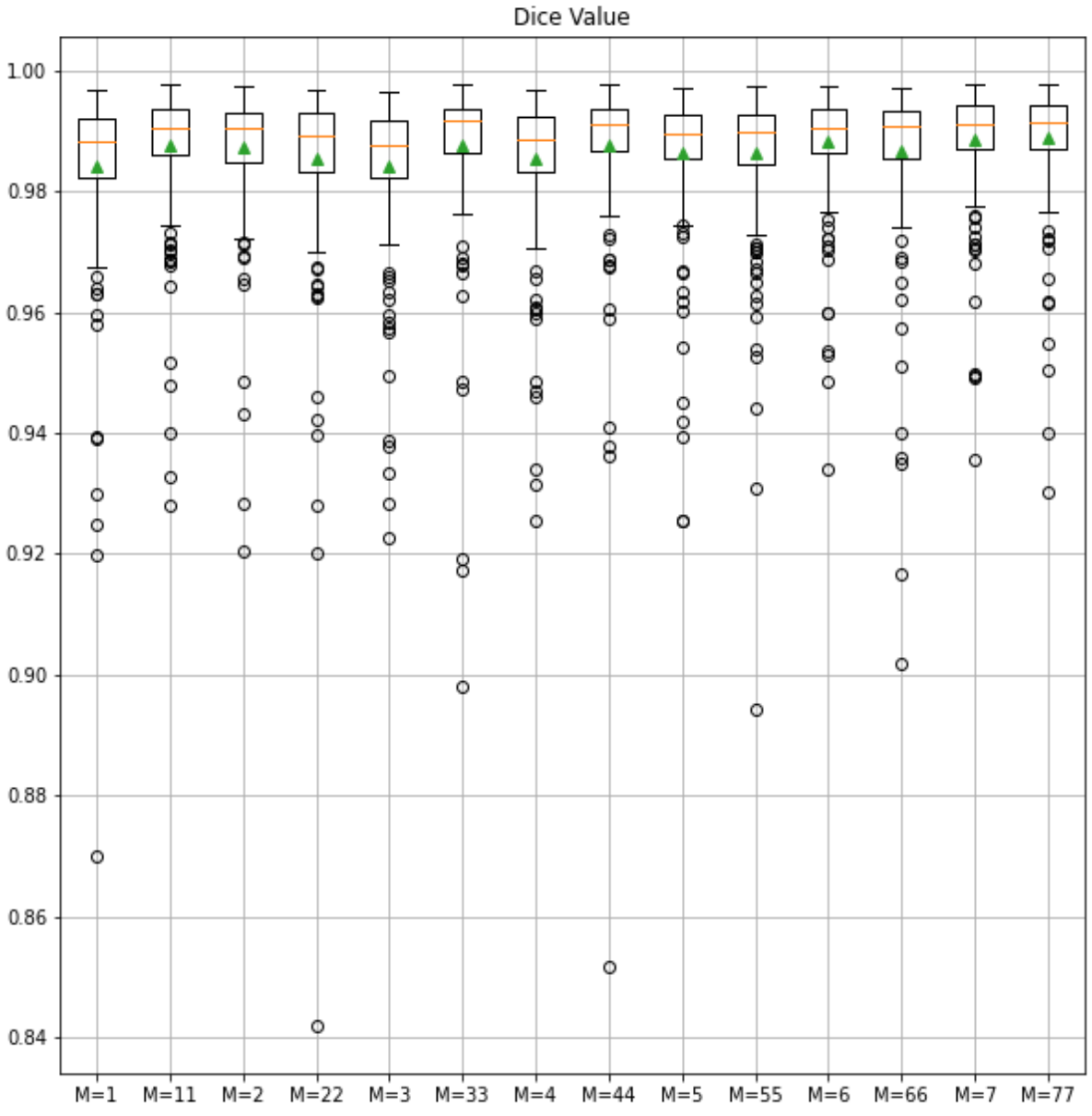


Figure 9. Average dice coefficient boxplot on valid set of L3-only dataset.

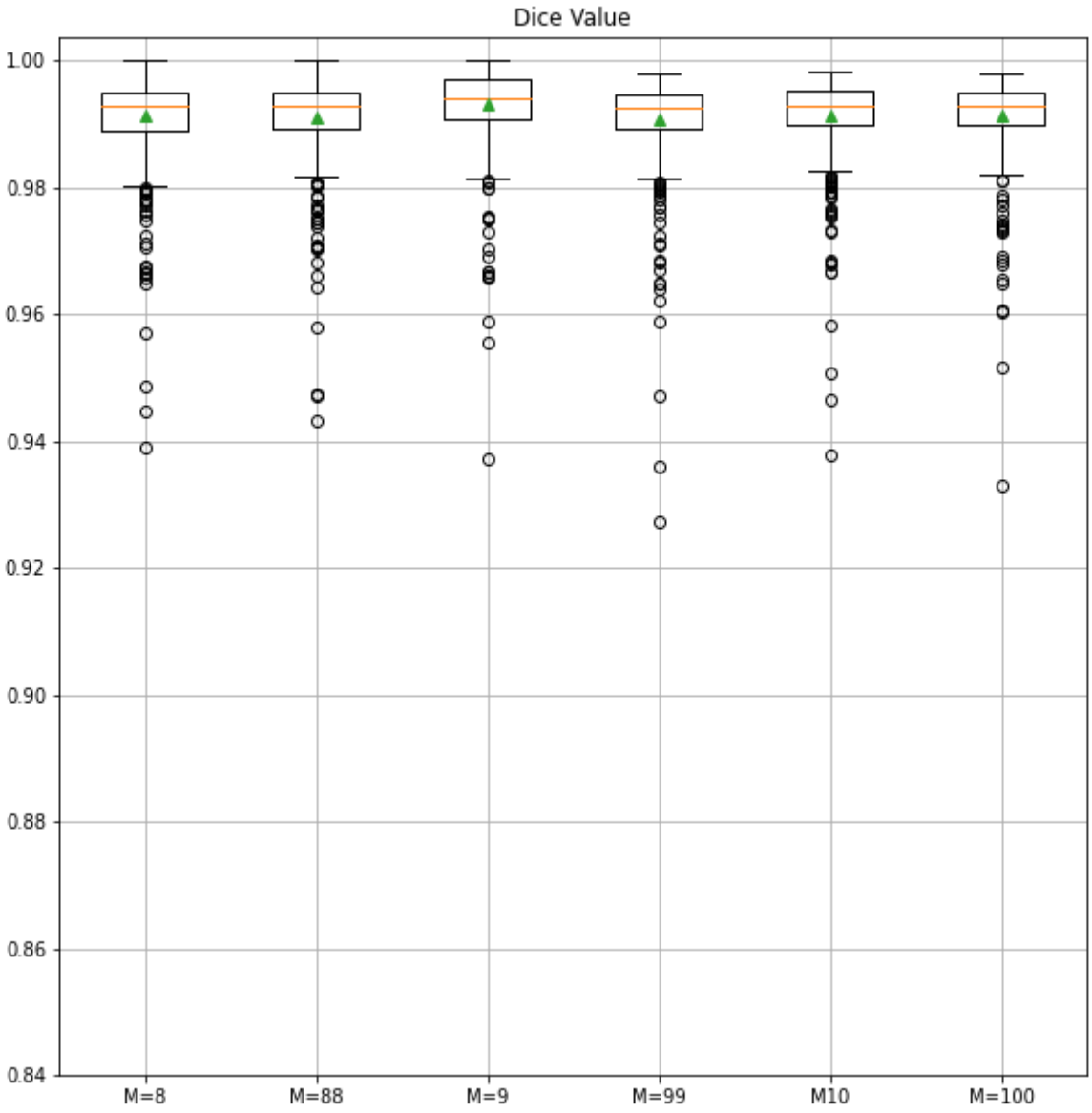


Figure 10. Average dice coefficient boxplot on valid set of L3/L2 and L1 dataset. Refer to table 3 for experience details.

- The final models (M=7,M=77) and (M=10,M=100) performs well on the final L3 only and L3-L2-L1 dataset respectively.

7 Conclusion

To sum up from all experiments performed, The results obtained within these last months are both satisfying and promising. Even though we mainly succeed the segmentation on different lumbar vertebrae levels (L3-L2 and L1) but there are still some challenges to encounter : noisy samples, Non-texture variability images.

The next step would be the full fine-tuning process of models' hyper-parameters to enhance performance and improve segmentations.