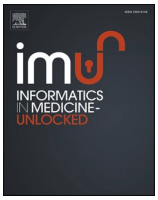
Informatics in Medicine Unlocked 24 (2021) 100585

Contents lists available at ScienceDirect

Informatics in Medicine Unlocked

journal homepage: www.elsevier.com/locate/imu



Loss, post-processing and standard architecture improvements of liver deep learning segmentation from Computed Tomography and magnetic resonance

Pedro Furtado

DEI/CISUC, Universidade de Coimbra, Portugal

ARTICLE INFO

Keywords:
Computed tomography
Liver
Deep learning
Segmentation

ABSTRACT

As deep learning is increasingly applied to segmentation of organs from Computed Tomography (CT) and Magnetic Resonance Imaging (MRI) sequences, we should understand the importance of certain operations that can improve the quality of results. For segmentation of the liver from those sequences, we quantify the improvement achieved with segmentation network, loss function and post-processing steps. Our results on a publicly available dataset show an improvement of 11% points (pp) by using DeepLabV3 instead of UNet or FCN, 4 pp by applying post-processing operations and 2pp using the top-performing loss function. The conclusions of this work help researchers and practitioners choosing the network and loss function and implementing effective post-processing operations.

1. Introduction

To enrich the level of understanding prior or during complicated medical procedures, physicians use advanced tools such as three-dimensional visualization and printing, which require extraction of the object(s) of interest from images. The precise segmentation of abdominal organs such as the liver is useful in those procedures, which motivates ongoing research to achieve better segmentation results and overcoming challenges originating from both highly flexible anatomical properties of abdomen and limitations of modalities reflected in image characteristics. In this context, image segmentation is the process of partitioning a digital image into multiple segments, where a segment is a contiguous set of pixels. It must identify the location and the precise boundaries of each object of interest, assigning a label to each pixel such that pixels with the same label belong to the same segmented object. Typical segmentation algorithms search for communalities in colour, intensity and texture to separate the image into regions. They face difficulties when there is lack of contrast with neighbouring regions and similarity between regions that are supposed to be segmented into different objects. For instance, in segmentation of liver from abdominal CT or MRI sequences, there is often lack of contrast between the liver and neighbour regions, and there are other organs with similar colour and texture. Some preprocessing operations can also be applied to enhance contrast or equalize histograms prior to segmentation.

CT is an imaging technique frequently used in medical diagnosis. The non-invasive procedure uses computer-processed combinations of many X-ray measurements taken from different angles to produce cross-sectional (tomographic) images (virtual "slices") of specific areas of a body part. The CT output is post-processed to reveal body structures based on their ability to absorb the X-ray beam. After acquisition the images of slices obtained by CT can be further processed by automatic computerized procedures to reveal, measure and analyze specific details. Already in the seventies, Stephens et al. [1] was reporting experiences with analysis of the liver using CT, describing the appearance of the normal liver and various hepatic abnormalities, the conduct of the examination itself and some of the problems that reduce the technical quality of the examination. Also, around the same time, Alfidi et al. [2] was examining the scope and accuracy of CT in detection of tumors, abscesses, cysts, and parenchymal disorders of the liver. Fast-forwarding to recent times, Chang et al. [3] analyzed computer-aided diagnosis of liver tumors on CT images, and Fuller et al. [4] studied the tumor growth rate in metastatic adrenocortical carcinoma using two-dimensional CT scans.

Traditionally, atlas-based segmentation separates the organs from background based on shape, edges and texture. Deep learning (DL) revolutionized the segmentation procedure. Instead of hand-coded and fixed organ extracting code, DL segmentation networks learn how to extract the organs and other objects of interest directly from a set of

E-mail address: pnf@dei.uc.pt.

https://doi.org/10.1016/j.imu.2021.100585
Received 27 December 2020; Received in revised form 22 April 2021; Accepted 22 April 2021
Available online 4 May 2021

images and corresponding segmentation groundtruths. The network iteratively modifies its own inner coefficients based on the amount of error (loss) along a large number of improvement iterations (back-propagation learning), until it optimizes the quality of segmentation. Typically, top performing segmentation networks are very accurate, scoring in the range of 0.85–0.95 Jaccard Index (JI) segmenting the liver. But the fraction of misclassified pixels is still quite visible, as errors are mostly concentrated near border regions, and sometimes there are also some incorrect small regions away from the main volume. The amount of errors also depends on factors such as acquisition conditions, contrast, variability in morphology and tissue specifics. How much could we expect to gain from using alternative networks, loss functions and from defining adequate post-processing steps in the context of segmentation of an organ such as the liver from CT and MRI scans? We concentrate on those major issues: describing networks, loss and post-processing mechanisms, and quantifying the improvement of specific choices. To achieve our goal, in this work we define and quantify the effect of each.

1.1. Backpropagation, loss, post-processing and networks

Deep learning uses convolution neural networks (CNN) to classify or segment (i.e. classify each pixel of) images. In its essence, a CNN is made of a sequence of layers, which compute feature maps. Feature maps are matrices of coefficients, the first one being the image itself with pixels as coefficients. Given those matrices (X), convolutions are mathematical product-sum operations with square filters (W) that move over the feature map and compute, at each point, the value $r = b_0 + w_0 x_0 + w_1 x_1 + \ldots + w_n x_n$, where w_i are the weights of the filters W, x_i are matrix coefficients, and b_0 is a bias factor. This output is then submitted to an activation function (A), and there are also pooling layers to down sample feature maps. Backpropagation is at the heart of weight learning to learn how to represent any data. Given a forward pass of the CNN through a batch of images, an error or loss is computed by comparing the output of the final layer with groundtruth data. Backpropagation propagates the error backwards from the last layer to the first, slightly adjusting network weights and bias at each iteration (using a learning rate). This process is repeated thousands of times with multiple training images and through multiple epochs (passes through all images) to successively reduce the error (loss).

In this process the loss function is the function that computes the error. At each iteration, given a set of groundtruth segmentations (GND) and the current segmentations (SEG), the loss is a measure of the difference between the two. This loss is used to determine a delta change that is backpropagated to incrementally change weights. Cross entropy (crossE) is most frequent, but dice (a.k.a. DSC) and intersect-over-the-union (IoU, a. k.a jaccard index) and variations of it are also popular choices. Both are also metrics used for evaluation of approaches, but accurate evaluation should consider those metrics over individual organs instead of over all image pixels (because more than 90% of all pixels are background). This detail is also important in the loss function, because it should balance the classes properly to avoid over-representing the background. Class balancing is achieved by multiplying influence of each pixel by inverse class frequency, or by computing the loss of each class and then averaging over all classes with equal weights for each class.

Post-processing is actually a very relevant and often overlooked tool to improve segmentation quality. It concerns image processing algorithms that apply properties and invariants of objects to improve a possibly noisy segmentation. Given an output of segmentation S and some knowledge K about the domain, post-processing computes an improved output $P = f(S,K)$. The mechanisms of post-processing are not well described and its contribution to quality of segmentation is not well quantified in prior work. Yet, we show that it contributes to a sensible improvement of the quality of segmentation in the case of organs. The properties used to implement post-processing in our proposal include the following: (1) an organ should be a single volume (3D) or area (2D); (2) it should be the maximum sized label region; (3) noisy pixels on borders or further away from the main volume can be isolated by eroding operations and removed; (4) edges/surfaces of organ are smooth (smoothing operator), and non-anatomical holes inside organs segments should be removed.

The segmentation network characteristics is also a relevant quality factor. A segmentation network has two main parts, the encoder and the decoder. The encoder is a CNN that extracts and successively compresses features from the image until it arrives at a compressed latent space using successive convolutions, activations and pooling layers. The decoder reinstates the full image size using an inverse operation, deconvolution. We focus on three main architectures, U-Net [5], FCN [6] and DeepLabV3 [7], because these represent the most popular networks and allow us to compare different architecture details. U-Net is a very popular network in medical imaging, made of symmetrical conv-deconv blocks of layers and a few direct forwarding connections in-between symmetrical layers. While U-Net trains deconvolution layers symmetric to the convolution layers, FCN has a similar encoder but uses simple non-trainable interpolation as decoder. The encoder used in U-Net is VGG-16, a dense CNN, while DeepLabV3 uses Resnet-18. The advantage of Resnet is that its residual blocks allow layers to feed into the next layer and directly into the layers about 2–3 hops away simultaneously. This operation helps avoid the vanishing gradients problem that occurs in deep networks such as these ones (vanishing means that, as more layers are added to neural networks, the gradients of the loss function tend to approach zero, making the network hard to train). DeepLabV3 also adds Atrous Spatial Pyramid Pooling (ASPP), a semantic segmentation module for resampling a given feature layer at multiple rates prior to convolution. This way the original image if probed with multiple filters that have complementary effective fields of view, thus capturing objects as well as useful image context at multiple scales. We compare the performance achieved by these three distinct network architectures (U-Net and FCN with VGG-16 and DeepLabV3 with Resenet-18 as encoder) in the context of organ segmentation from the CT and MRI images.

For experimentation, we take publicly available CT and MRI segmentation datasets with liver segmentation data, set training optimizations and data augmentation, to evaluate the effect of applying the defined post-processing algorithms, alternative loss functions and alternative network architectures. This allowed us to conclude that post-processing improves segmentation quality by 2–5 pp, depending on the CT or MRI sequences, DeepLabV3, with its Resnet-18 encoder and ASPP multiscale capabilities surpasses FCN and UNet by 11% points (pp), and a best choice of loss function improved segmentation quality by 2 pp.

1.2. State-of-the-art in segmentation of CT and MRI

Prior to the use of DL, segmentation would be based on more traditional image processing approaches exploring organs and abdomen characteristics, e.g. multi-atlas approaches. For instance, the approach proposed by Bereciartua et al. [8] uses 3D models of the liver and probability maps. Le's proposal [9] is based on histograms to segment the liver, followed by active contours for refinement, while Huynh et al. [10] applies watershed together with active contours. Statistics concerning the volumes and locations of organs are used in most such approaches. With the advent of deep learning (DL), convolution neural networks (CNN) and segmentation networks, those traditional techniques were replaced by segmentation networks that learn based on training images and corresponding segmentation groundtruths. Zhou et al. [11] used fully convolutional networks (FCN), achieving top scores for the problem of segmenting abdominal organs from CT sequences. The approach takes 3-D CT images and applies a majority voting scheme on the output of segmentation of 2D slices taken from different image orientations, plus localization of organs. The steps are: (1) localize individual organs, i.e. determine bounding boxes for target organs using

sliding windows and pattern matching over Haar-like and LBP features; (2) calibrate and correct results using Hough voting for organ locations. Following that work, Bobo et al. [12] applied it to abdomen segmentation from MRI sequences. The work shows significant improvement for the segmentation compared to multi-atlas methods. In another work, Larsson et al. [13] presented SeepSeg, which segments abdominal organs using 3 steps: 1) Localize region of interest with a multi-atlas technique; 2) Use a CNN for pixel binary classification; 3) Post-process using thresholding, then remove positive samples except those of the largest connected region. Network architecture, ensembles and voting are some of the recent trends in research [14]. proposes ALAMO (Auto deep Learning based Abdominal Multi-Organ segmentation), a multi-slice 2D neural network. Groza [15] presents an ensemble of DL networks with voting to achieve improved segmentation scores for MRI scans.

Modifications of the loss function was used in Cai et al. [16] for improved segmentation for the pancreas in CT and MRI images. The authors used a "direct" loss function. They propose a Jaccard Loss (JACLoss) for training neural network image segmentation model. As explained by the authors, "to optimize the Jackard Index (JI) (a main segmentation metric) directly in network training makes the learning and inference procedures consistent. It empirically works better than the cross-entropy loss or the class-balanced cross-entropy loss when segmenting small objects, such as pancreas in CT/MRI images". Conze et al. [17] also replaces cross-entropy by dice. Finally, Salehi et al. [28] proposed a generalized loss function, based on differently weighting false positives and negatives. Loss is a metric, and there is a limitation with many metrics used in evaluation of segmentation quality, mentioned for instance in Zhang et al. [18]: "many scores are artificially high simply because the background is huge, hence the term TN (true negatives) is also huge, making specificity, ROC and AUC inviable as scores".

In spite of all the work in advanced architectures, literature is still missing a quantitative evaluation of the impact of loss and post-processing. In this work we take standard network architectures, but including DeepLabv3, and focus on defining and evaluating the contribution of the network, loss and post-processing improvements on the quality of segmentation of liver in CT images. Table 1 summarizes the scores achieved by recent related work on segmentation of the liver and other abdominal organs. Those works use different datasets from ours and explore advanced techniques such as multi-views, ensembles of networks and voting. All of them use some base network (most frequently UNet) that is then extended to integrate their advances. Hu et al. [23] and Wang et al. [24] obtained the best scores for CT in those works. Our best scores in this work (DeepLabV3, dice, post-processed), which we include here for comparison, were slightly lower but still competitive, even though we did not use ensembles of networks or multi-views.

Table 1
Summary of scores by other authors.

JI = IoU	Liver	spleen	R Kidney	L kidney
[25]	0.85	–		
[11]	0.88	0.77		
[23]	0.92	0.89		
[24]	0.96	0.94	0.96	0.94
[26]	0.9	–	0.84	0.80
[15]				
F-net	0.86	0.79	0.79	0.80
BRIEF	0.74	0.60	0.60	0.60
U-Net	0.89	0.80	0.77	0.78
[13]	0.90	0.87	0.76	0.84
[12]	0.84	0.87	0.64	
[29]	0.90(LiverNet)	–	–	–
[30]	0.91	–	0.87	

1.3. Structure of the document

The remainder of the work is organized as follows: section 2 discusses materials and methods. First, section 2.1 describes our methodology and the data processing sequence of our setup. Section 2.2 introduces the datasets used for our experimental work, including CT and MRI data. Then sections 2.3, 2.4 and 2.5 describe the relevant details of the loss, post-processing and architecture features that we define and experiment with, and section 2.6 describes the setup for our experimental work. Section 3 contains experimental results and analysis and section 4 concludes the paper.

2. Materials and methods

2.1. Methodology and data processing

Fig. 1 shows the methodology we follow to define and evaluate post-processing, loss and architectures. We first define the post-processing algorithms (1), then loss (2) and architectural alternatives (3). After that we resort to experimentation to find the best combination of those features. To do that we choose CT and MRI datasets segmenting the liver, define experimental setup and options and run experiments. The results of those experiments, analyzed based on a set of metrics, allows us to conclude regarding improvement of segmentation quality depending on the choices and alternatives defined.

Figs. 2 and 3 show a data processing view of the systems that we implemented. The training setup in Fig. 2 is shows the training images as inputs, the network that is trained (with alternative structures and features) and the loss function (with defined alternatives) that is used in backpropagation learning. Fig. 3 then shows the pipeline used with an already trained network to process new images and to evaluate quality, where the trained network is applied on new images and the post-processing operations improve the segmentation output after the network has processed the images.

2.2. The Computed Tomography and magnetic resonance datasets used

CT upper abdomen sequences from 40 different patients are used in this study (CT dataset). The images were acquired using equipments Philips SecuraCT, 16 detectors, Philips Mx8000, 64 detectors, Toshiba AquilionOne, 320 detectors (equipped with spiral CT). Subjects were all healthy (livers did not exhibit lesions or disease). A contrast agent was used, the abdomen sequences obtained at hepatic phase, i.e. 70-80 s pelvic incidence (p.i.) or 50–60 s after bolus tracking. In this phase the liver parenchyma enhances through blood supply by the portal vein, resulting in some potential enhancement of the hepatic veins. The resulting 2874 slices have a resolution of 512 × 512, XY spacing 0.7–0.8 mm and inter slice distance 3–3.2 mm. For our experiments the acquisitions were divided 80/20 into independent train and test datasets using 5-fold cross-validation.

Slices are cross-section images taken from the CT sequences. Fig. 4 shows a 3D model of the liver (a) and also a 3D model created by stacking successive slices of a CT sequence.

Fig. 5 shows CT slices and their segmentation results. In each one we can see the groundtruth on the left, with the CT slice and a superimposed region representing the liver on the right. In these examples the first slice was pretty well segmented, with few differences between the actual groundtruth and the segmentation outcome, while the second slice has some imperfections when compared to the groundtruth.

While our main experimentation in this work concerns the CT dataset just described, for added experimental variation we also include MRI scenario (MRI dataset) from Ref. [27]. It consists of 120 MRI sequences capturing abdominal organs obtained using T1-DUAL fat suppression protocol. The dataset contains both images and organs groundtruths, from which we selected liver groundtruth data. The sequences were acquired by a 1.5T Philips MRI, which produces 12-bit DICOM images

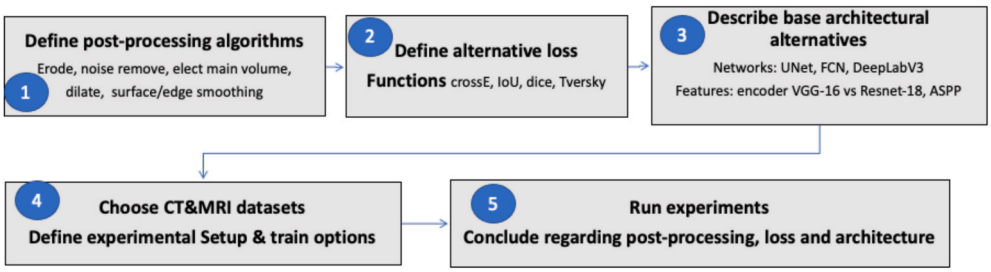


Fig. 1. Methodology followed in this work.

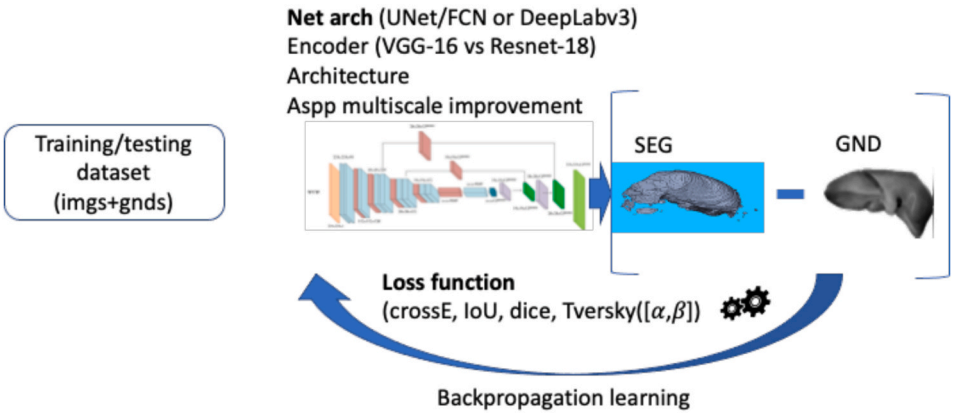


Fig. 2. Training and testing - data processing sequence.

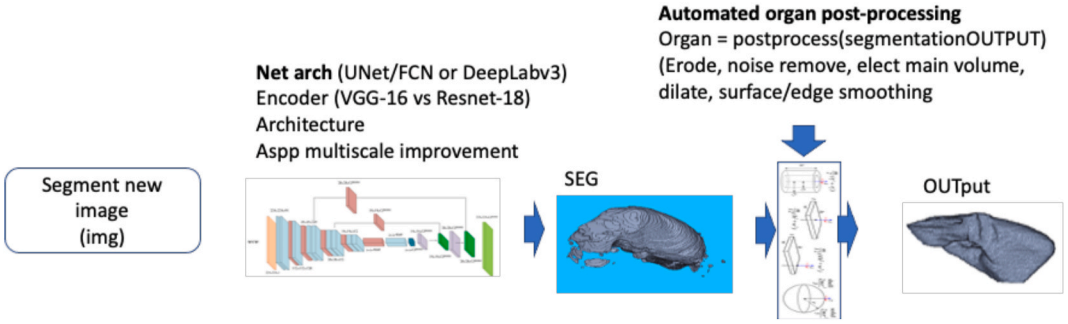


Fig. 3. Segmenting a new image - data processing sequence.

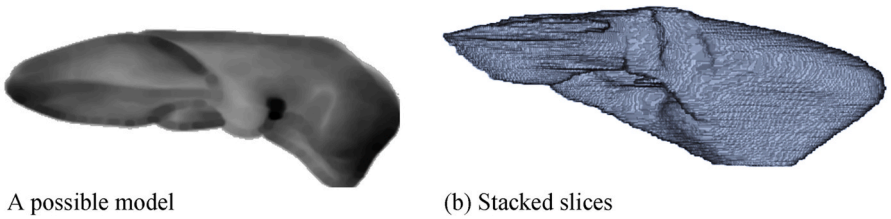


Fig. 4. Model and stacked slices of the liver. (a) a typical model of the liver; (b) a segmentation output organized as stacked slices on top of each other, resulting in 3D model that compares with (a).

with a resolution of 256 × 256. The inter-slice distance was 5.5–9 mm (average 7.84 mm), x-y spacing is between 1.36 and 1.89 mm (average 1.61 mm) and the number of slices is between 26 and 50 (average 36). In total there are 1594 slices (532 slice per sequence) used for training and testing, with the testing sequences being chosen randomly to include 20% of all sequences. The same data augmentation and training options were applied in this case.

2.3. Post-processing algorithms

Consider a CT sequence ims. The sequence (ims) is a ($m \times n \times s$) volume made of s stacked slices (im), where each slice is a 2D image (im) with ($m \times n$) pixels. Segmentation of a slice creates a ($m \times n$) labelmap (lmap) where each pixel contains a label corresponding to the segment the pixel belongs to, either background (0) or liver (1). The sequence of stacked labelmaps ($m \times n \times s$) creates labelmap sequence (lmaps) con-

taining the segmented organs.

Given both each labelmap (lmap) and the lmaps volume, post-processing removes imperfections, noisy pixels and in corrections by image processing operations. It uses simple morphological operations (some popular references on image processing and morphological operations include [19–22]) to remove errors resulting from the segmentation process. Errors are identified based on simple invariants, such as the fact that the liver is a single big volume and borders are smooth. The following operations are done in the 3D volume:

Algorithm 1. Post-process improvements 3D

Algorithm 1: Post-process improvements 3D	**Comments**
Input: 3D segmentation labelmap sequence - **lmaps** ($m \times n \times s$)	
Output: Cleaned 3D segmentation labelmap sequence - **lmaps** ($m \times n \times s$)	
1. lmaps = erode3D(lmaps)	uses imerode [20]
2. lmaps=bwlabel(lmaps)	obtains connected regions [20]
3. compute $[l_i, n_i]$ = nr of occur ni of each label li in lmaps	a histogram
4. $lmax: ni = max_{i\ in\ labels}(n_i)$	find label with max number of occurs
5. $lmaps(lmaps \neq l_{max}) = 0$	remove all labels except largest region
6. lmaps=dilate3D(lmaps)	uses imdilate [20]
7. End	

Algorithm 1 removes all except the largest region, which is the liver itself, but since the border regions often have significant erroneous pixel classifications (noise), it is preceded by an erosion to disconnect most of those erroneous pixels from the main volume representing the liver. Fig. 6 shows how Algorithm 1 removes segmentation errors. In (a) the organ volume is initially surrounded by some wrong classifications that are represented in the figure by some extra regions; in (a) to (b) imerode isolates those previously connected protruding regions; in (b) to (c) only the largest region is kept; in (c) to (d) imdilate restores the volume, but now without the erroneous parts.

The 2D algorithm (algorithm 2 shown next), applies first the erode-keepLargest-dilate operations as well, but this time on sequences of 2D slices obtained considering each one of the three main 3D axis, and then it also applies an operation to fill holes and an operation to smooth edges. Algorithms 1 and 2 apply those operations in a certain sequence to clean the estimations and obtain a single liver region: (1) first they apply an erosion operator (imerode [20]) to decouple small noises from the main volume; (2) 2D or 3D connected components are identified (bwlabel function [20]) and their sizes calculated as the number of times each identifier appears in the labeled mask; (3) only the largest remaining region is kept (by zeroing all except the label of the largest region found); (4) Next, imdilate [20] is applied to dilate the previously eroded image; (5) Finally, in the case of algorithm 2, edge smoothing (detailed in Algorithm 3) is applied. Given the binary labeled mask (a mask where liver is 1 and not liver is 0), a 2-D convolution with a uniform square filter is used to blur the image**.** From blurred image, pixels with value lower than threshold (default = 0.5) are zeroed, those with threshold larger or equal to 0.5 are liver. This procedure results in a much smoother surface.

Algorithm 2. Post-process improvements on 2D

Algorithm 2: Post-process improvements on 2D	**Comments**
Inputs: 3D segmentation labelmap sequence - **lmaps** ($m \times n \times s$); process axis ax (e.g. [1 2 3])	
Output: Cleaned 3D segmentation labelmap sequence - **lmaps**	($m \times n \times s$)
1. For each axis ax_i in ax,	
2. For each slice $lmap_i$ along axis ax_i,	$lmap_i$ is binarized
3. $lmap_i$ = erode2D($lmap_i$)	uses imerode [20]
4. $lmap_i$ =bwlabel($lmap_i$)	obtain connected regions [20]
5. compute $[l_i, n_i]$ = nr occur ni of each label l_i in $lmap_i$	a histogram
6. $lmax: ni = max_{i\ in\ labels}(n_i)$	find label with maximum number of occurrences
7. $lmap_i(lmap_i \neq l_{max}) = 0$	remove all labels except largest region
8. $lmap_i$=dilate2D($lmap_i$)	uses imdilate [20]
9. $lmap_i$=fillholes($lmap_i$)	uses imfill [20]
10. $lmap_i$=Smoothedges($lmap_i$)	
11. End	
12. End	

Algorithm 3. Smoothedges($lmap_i$)

Algorithm 3: Smoothedges($lmap_i$)
Input: binary mask $lmap_i$ $(m \times n)$
Output: Smoothed mask - $lmap_i$ $(m \times n)$
1. $lmap_i$ = conv2D($lmap_i$, kernel=uniform) 2. $lmap_i(lmap_i < 0.5)$=0 3. $lmap_i(lmap_i \geq 0.5)$=1 4. End

Table 2 shows the parameters used in our experimental work for these algorithms.

2.4. Loss functions

Besides the default loss function (cross entropy), we also experiment with iou, dice and variations that include a generalized loss function.

Cross entropy (crossE, the default to compare with): cross-entropy is well-known and the default loss function. Given the set of probabilities p of a single pixel of the segmentation output to be of each possible class, and the real probabilities (one-hot encoding of the class), cross entropy measures dissimilarity between p and q. If ti and si are the groundtruth and the CNN score of each pixel for each class i respectively,

$$crossE = -\sum_{i}^{C} t_i \log s_i \tag{1}$$

By applying a class frequency inverse weight to the value for each pixel we obtain class-weighted cross-entropy, which is the variant we use and denote as "crossE".

Intersect over the Union (IoU): IoU is a convenient measure of the degree of overlap or match between segmentation-obtained regions and the corresponding groundtruth regions. Given the number of true positives (TP), false positives (FP) and false negatives (FN) in the classification of pixels, loss is (1-IoU),

$$IoU(loss) = 1 - IoU = 1 - \frac{TP}{TP + FP + FN} \tag{2}$$

But since this IoU averages over all pixels and we identified the problem with that measurement, IoU averaged over the classes is used instead,

$$IoU(loss) = 1 - \frac{\sum_{I=1}^{C} IoU_i}{C}, IoU_i = 1 - \frac{TP_i}{TP_i + FP_i + FN_i} \tag{3}$$

Dice (dice): The dice or Dice Similarity Coefficient (DSC) is a metric that is highly correlated and can be obtained from IoU directly. The loss formula for the dice is:

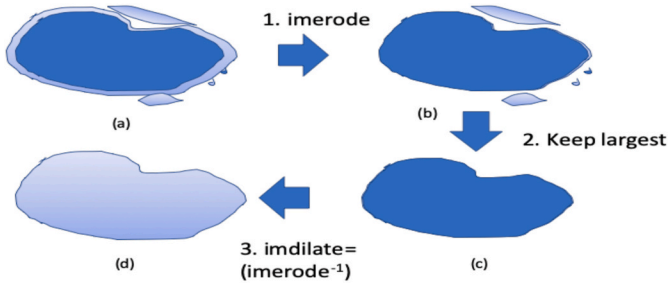


Fig. 6. Post-processing to remove noise.

$$dice(loss) = 1 - DSC = 1 - \frac{2TP}{2TP + FP + FN} \tag{4}$$

As with IoU we use an average over classes,

$$dice(loss) = 1 - \frac{\sum_{I=1}^{C} dice_i}{C}, dice_i = 1 - \frac{2TP_i}{2TP_i + FP_i + FN_i} \tag{5}$$

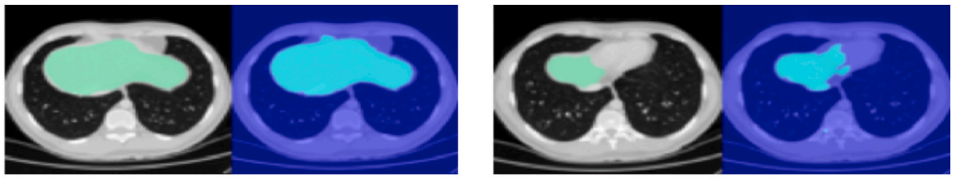
Tversky loss (Tloss): Tversky loss is a generalization of IoU loss where more weight can be given to either false positives or to false negatives by including a multiplier factor:

$$T(loss) = 1 - \frac{TP}{TP + \alpha FP + \beta FN}, \ 0 \leq \alpha, \beta \leq 2, \ \alpha + \beta = 2, \tag{6}$$

Loss without considering the background (dice noBK): Since the background is easier to segment than the remaining classes and is also huge, dice noBK is an alternative that removes the background from the loss formula (i.e. it averages loss over all classes except the background). The objective is to try to emphasize the need to segment the other classes well. An experimental approach is necessary to evaluate if this alternative improves the outcome.

2.5. Segmentation networks

Figs. 7–9 shows block diagrams of the segmentation networks U-Net,



(a) (b)

Fig. 5. Segmentation of liver slices. (a) a well segmented slice with a great liver area; (b) a segmented slice of a smaller liver area near the liver extremity, shows a significant overflow into neighbour regions.

Table 2
Post-processing parameters used.

Operation	Matlab function	parameters
Erode (2D and 3D)	J = imerode (labelmap, SE) SE = strel ('disk',r) (2D) SE = strel ('sphere',r) (3D)	r = 3
Bwlabel	L = bwlabel (labelmap, conn)	conn = 8
dilate (2D and 3D)	J = imdilate (labelmao, SE) SE = strel ('disk',r) (2D) SE = strel ('sphere',r) (3D)	r = 3
Fill holes	imfill	imfill (labelmap,'holes')

Smoothedges conv2D and threshold	windowSize = 5;
	kernel = ones(w)/w^2;
	blurryImage = conv 2 (single (binaryLabelmap), kernel, 'same');
	binaryImage = blurryImage > 0.5; % Rethreshold	

FCN and DeepLabV3 that we build and experiment with. The legend of Figs. 7–9 also summarizes the contents of the block of layers that are shown. At the same time Tables 3, 4 and 5 list the layers, divided into three main regions corresponding to encoder, decoder and input/output layers, plus bridge layers (UNet) and ASPP layers (DeepLabV3). In Tables 3,4 and 5 U-Net has 58 layers, FCN has 51 layers and DeepLabV3 has 100 layers. U-Net and FCN (Figs. 7 and 8) are both using VGG-16 encoder, while DeepLabV3 (Fig. 9) uses a Resnet-18. Comparing FCN to UNet, FCN uses interpolation layers to decode, while U-Net uses a sequence of de-convolution layers that is symmetric to the encoder layers. From Table 3 we see that layers 1 to 22 of UNet are VGG-16 encoder, layers 23 to 27 are bridge layers and layers 28 to 55 are symmetric decoder layers. In Table 4 we see that, in FCN, layers 2 to 31 are VGG-16 encoder layers, layers 40 to 45 are the decoder layers and the remaining layers are input, bridge and output layers. DeepLabV3 is

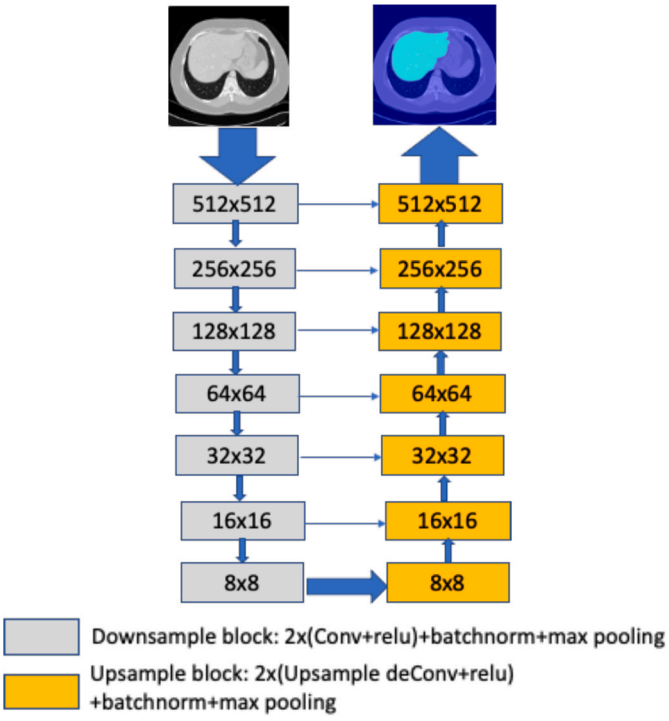


Fig. 7. UNet segmentation network.

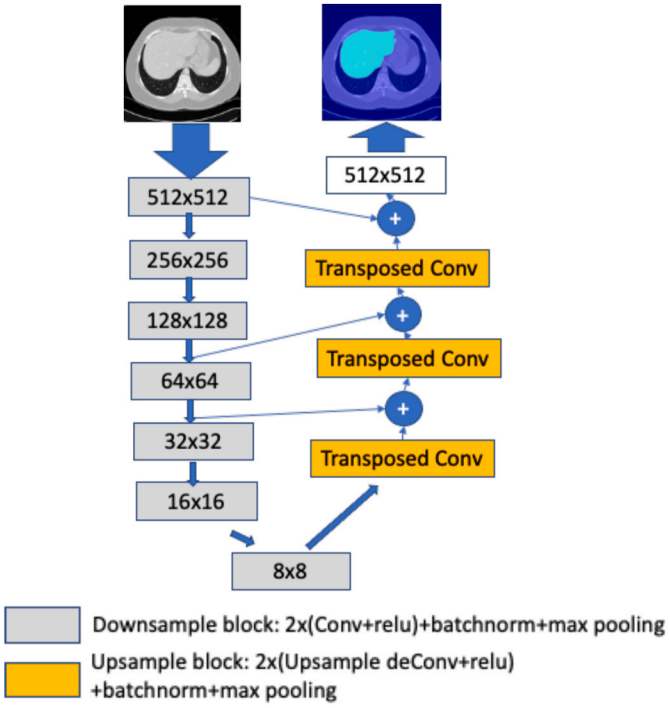


Fig. 8. FCN segmentation network.

deeper, with layers 2 to 64 being Resnet-18 encoder layers (Table 5). The residue blocks of Resnet-18 encoder, also depicted in Fig. 9, are important to avoid the vanishing gradients problem of deep networks. The next sequence of layers of DeepLabV3 after Resnet-18 layers, also depicted in Fig. 9 and listed in Table 5, is the "ASPP" layers. ASPP resamples a feature layer at multiple rates prior to convolution, so that the original image if probed with multiple filters that have complementary effective fields of view, thus capturing objects as well as useful image context at multiple scales. In this case 4 atrous convolutions are applied in parallel with 4 different rates (dilation factors) of 1, 6, 12 and 18, as shown in Fig. 9. In what concerns the decoder, UNet's decoder has many layers (layers 28 to 55) that are symmetric to the corresponding encoder layers, while FCN uses a small set of interpolation layers (layers 40 to 45) shown in Table 4. Although not detailed, all the three networks include forwarding connections from certain encoder to certain decoder layers.

2.6. *Data augmentation, training options and metrics*

Data augmentation is a standard mechanism to enrich the dataset with further diversity and also quantity, by adding randomly transformed versions of existing training images. We defined transformations that translate the organ, scale up and down and rotate slightly as well. Those consisted in translations of up to 10 pixels and random rotations up to 10°, both in any direction, as well as shearing up to 10 pixels and scaling to 10% up and down as well. For the experiments we used a PC with an intel core i5 at 3.4 GHz, with 16 GB RAM, an NVIDEA GForce GTX 1070 GPU Pascal architecture with 1920 cores, 8 GB GDDR5, 8 Gbps memory speed, and an SSD disk of 1 TB. The PC runs windows and Matlab 2018b was used for the coding and experimentation. The experiments themselves involved training each of the networks (FCN, DeepLabV3, U-Net) and then using the networks to classify independent test images. The dataset was split into 80% train and 20% test sequences. Training was configured to 500 training epochs. Table 6 has a summary of training parameters.

For the experiments, we report first training runtimes and training evolution. Then we compare segmentation networks and alternative loss

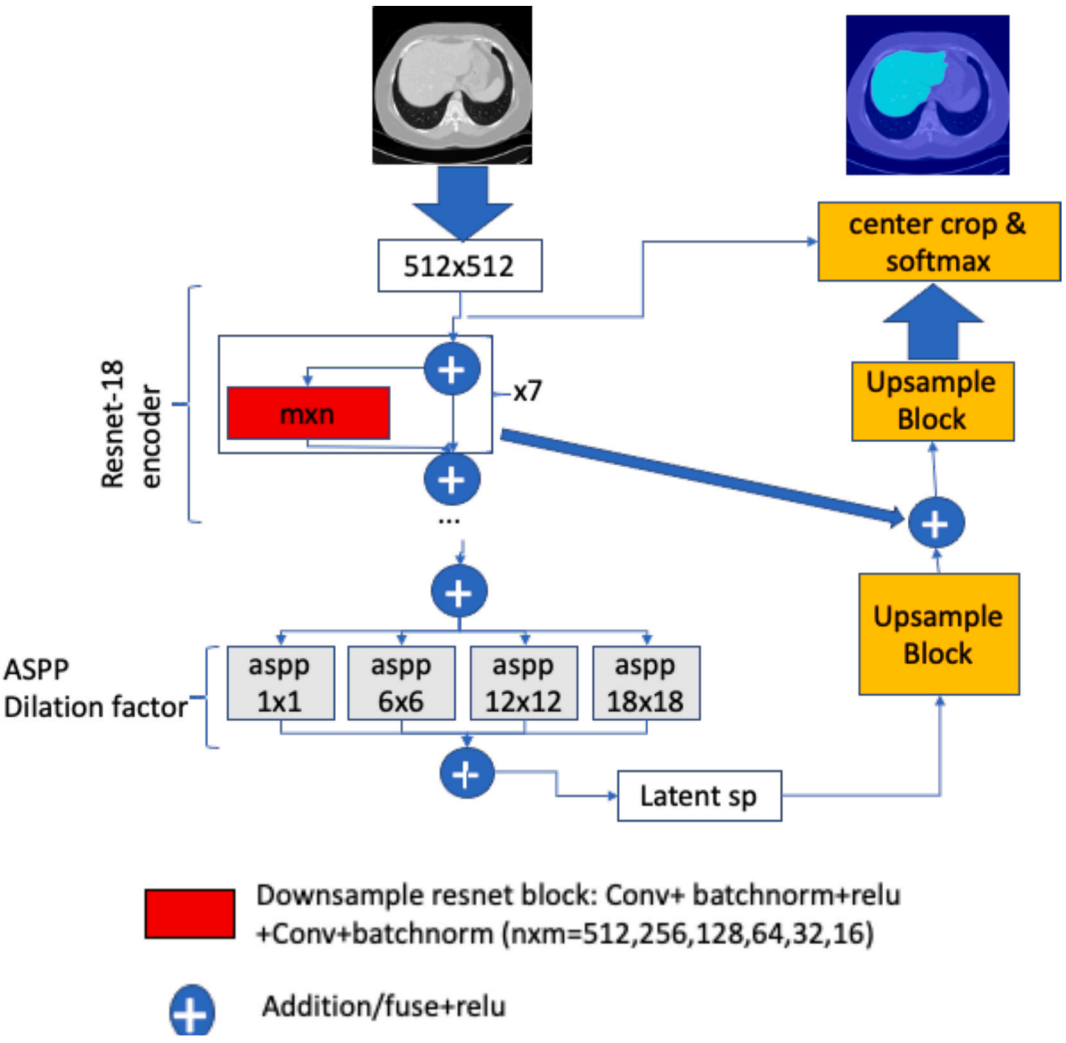


Fig. 9. DeepLabV3 segmentation network.

Table 3
UNet layers (VGG-16 encoder; In, Bridge and last layers; Symmetrical decoder).

1	im input 512×512×1	16	2×2 max pool stride [2 2]	31	512 3×3 convs stride [1 1]	46	ReLU
2	64 3×3 convs stride [1 1]	17	512 3×3 convs stride [1 1]	32	ReLU	47	128 3×3 convs stride [1 1]
3	ReLU	18	ReLU	33	512 3×3 convs stride [1 1]	48	ReLU
4	64 3×3 convs stride [1 1]	19	512 3×3 convs stride [1 1]	34	ReLU	49	64 2×2 transp convs stride [2 2]
5	ReLU	20	ReLU	35	256 2×2 transp convs stride [2 2]	50	ReLU
6	2×2 max pool stride [2 2]	21	50% dropout	36	ReLU	51	Depth concat of 2 inputs
7	128 3×3 convs stride [1 1]	22	2×2 max pool stride [2 2]	37	Depth concat of 2 inputs	52	64 3×3 convs stride [1 1]
8	ReLU	23	1024 3×3 convs stride [1 1]	38	256 3×3 convs stride [1 1]	53	ReLU
9	128 3×3 convs stride [1 1]	24	ReLU	39	ReLU	54	64 3×3 convs stride [1 1]
10	ReLU	25	1024 3×3 convs stride [1 1]	40	256 3×3 convs stride [1 1]	55	ReLU
11	2×2 max pool stride [2 2]	26	ReLU	41	ReLU	56	2 1×1 convs stride [1 1]
12	256 3×3 convs stride [1 1]	27	50% dropout	42	128 2×2 transp convs stride [2 2]	57	softmax
13	ReLU	28	512 2×2 transp convs stride [2 2]	43	ReLU	58	Cross-entropy loss
14	256 3×3 convs stride [1 1]	29	ReLU	44	Depth concat of 2 inputs		
15	ReLU	30	Depth concat of 2 inputs	45	128 3×3 convs stride [1 1]		

functions. Our next experiment compares quantitatively and visually the results with and without post-processing. Then we also include results on MRi dataset for added variation. The most important metric we use is IoU of the liver, but other metrics are also reported that include global accuracy, mean accuracy, weighted IoU, mean IoU and meanBFScore, a measurement of the degree of match between boundaries of segments and groundtruths. One important detail in what concerns metrics is that we include percentage points increase (pp) as a way to evaluate the improvement brought by a certain approach. Given a metric expressed as percentage (0–100%), in our case we used IoU, pp is the difference between the value achieved by the metric in one scenario/configuration, minus the value achieved in the other scenario/configuration. An IoU of

Table 4
FCN layers (VGG-16 encoder; in, bridge layers; decoder; out layers).

1	Image Input	19	Conv 512 3x3x256 filters	37	ReLU + 50% dropout
2	Conv 64 3x3x3 filters	20	ReLU	39	Conv 5 1x1 filters
3	ReLU	21	Conv 512 3x3x512 filters	40	Transposed Conv 5 4x4x5
4	Conv 64 3x3x64 filters	22	ReLU	41	Addition
5	ReLU + max pool 2x2	23	Conv 512 3x3x512 filters	42	Transposed Conv 5 4x4x5
7	Conv 128 3x3x64 filters	24	ReLU + max pool 2x2	43	Addition
8	ReLU	26	Conv 512 3x3x512 filters	44	Transposed Conv 5 16x16x5
9	Conv 128 3x3x128 filters	27	ReLU	45	Crop 2D
10	ReLU + max pool 2x2	28	Conv 512 3x3x512 filters	46	Softmax
12	Conv 256 3x3x128 filters	29	ReLU	47	Pixel Classification Layer
13	ReLU	30	Conv 512 3x3x512 filters	48	Conv 5 1x1 filters
14	Conv 256 3x3x256 filters	31	ReLU + max pool 2x2	49	Crop 2D
15	ReLU	33	Conv 4096 7x7x512 filters	50	Conv 5 1x1 filters
16	Conv 256 3x3x256 filters	34	ReLU + 50% dropout	51	Crop 2D
17	ReLU + max pool 2x2	36	Conv 4096 1x1x4096 filters		

Table 5
DeepLabV3 layers (Resnet 18 encoder; ASPP layers; decoder; in/out layers).

1	Image Input	37	Batch Normalization + ReLU	72	aspp Conv 256 3x3 filters
2	Conv 64 7x7 filters	39	Conv 256 3x3 filters	73	aspp Batch Norm + ReLU
3	BatchNorm,relu,maxPool [3x3]	40	Batch Norm + Residue addition + ReLU	75	aspp Conv 256 3x3 filters
6	Conv 64 3x3 filters	43	Conv 256 1x1 filters	76	aspp Batch Norm + ReLU
7	Batch Normalization+ReLU	44	Batch Normalization	78	aspp Conv 256 3x3 filters
9	Conv 64 3x3 filters	45	Conv 256 3x3 filters	79	aspp Batch Norm + ReLU
10	Batch Norm + Res add + ReLU	46	Batch Normalization + ReLU	81	Conv 256 1x1 filters
13	Conv 64 3x3 filters	48	Conv 256 3x3 filters	82	Batch Normalization + ReLU
14	Batch Norm + ReLU	49	Batch Norm + Residue addition + ReLU	84	transp conv 256 8x8 filters
16	Conv 64 3x3 filters	52	Conv 512 3x3 filters	85	Crop 2D
17	Batch Norm + Res add + ReLU	53	Batch Normalization + ReLU	86	Conv 48 1x1 filters
20	Conv 128 3x3 filters	55	Conv 512 3x3 filters	87	Batch Normalization + ReLU
21	Batch Norm + ReLU	56	Batch Norm + addition + ReLU	89	Depth concatenation
23	Conv 128 3x3 filters	59	Conv 512 1x1 filters	90	Conv 256 3x3 filters
24	Batch Norm + Res add + ReLU	60	Batch Normalization	91	Batch Normalization + ReLU
27	Conv 128 1x1 filters	61	Conv 512 3x3 filters	93	Conv 256 3x3 filters
28	Batch Normalization	62	Batch Normalization + ReLU	94	Batch Normalization + ReLU
29	Conv 128 3x3 filters	64	Conv 512 3x3 filters	96	Conv 5 1x1 filters
30	Batch Norm + ReLU	65	Batch Norm. +addition + ReLU	97	Transp Convs 5 8x8 filters
32	Conv 128 3x3 filters	68	aspp Depth concatenation	98	Crop 2D
33	Batch Norm + Res add + ReLU	69	aspp Conv 256 1x1 filters	99	Softmax
36	Conv 256 3x3 filters	70	aspp Batch Normalization + ReLU	100	Pixel Classification

Table 6
Experimental configurations.

Configuration	Parameter	value
Data augmentation	Translation	rand 0 to 10 pixels
	Rotation	rand 0–10°
	Shearing	rand 0 to 10 pixels
	Scaling	rand 0–10% up or down
Train/test	Division of CT + MRI sequences	80% train, 20% test
	Epochs	500
	Learn rate schedule	piecewise
	LearnRateDropPeriod	10
	LearnRateDropFactor	0.8
	Momentum	0.9
	InitialLearnRate	0.0005
	MiniBatchSize	8
	Shuffle	'every-epoch'
	ValidationPatience	Inf
Datasets	Classes	Background + liver

Table 7
Times and memory.

Architecture	Training time (mins)	epochs (and iterations)	Final validation accuracy	Segmentation time (one slice, in secs)
DeepLabV3	159	5 (2600)	99.25%	0.093 (±0.005)
FCN	864	30 (16260)	93.92%	0.093 (±0.004)
U-Net	789	30 (16260)	92.81%	0.081 (±0.007)
Post-processing time				8.7 (±0.87)
GPU Memory used (DeepLabV3)				8014 MB

Table 8
Global metrics.

Approach	Global Accuracy	Mean Accuracy	Weighted IoU	Mean IoU
Simple	0.84	0.90	0.79	0.56
DeeplabV3	0.98	0.98	0.96	0.88
FCN	0.95	0.97	0.92	0.77
UNet	0.86	0.91	0.89	0.75

Table 9
Global metrics vs losses (CT data).

	Global Accuracy	Mean Accuracy	Weighted IoU	Mean BFScore	Mean IoU	IoU liver
crossE	0.99	0.97	0.97	0.88	0.90	0.82
iou	0.98	0.96	0.97	0.89	0.89	0.80
Tloss (α1.5, β0.5)	0.97	0.95	0.97	0.87	0.88	0.78
Tloss (α0.5, β1.5)	0.97	0.97	0.97	0.88	0.87	0.76
dice	0.99	0.97	0.98	0.90	0.91	0.84
dice noBK	0.98	0.92	0.97	0.84	0.89	0.79

Table 10
Summary of post-processing improvements (CT data, pp = percentage points).

Step	IoU liver	IoU Seq 1	IoU Seq 2	IoU Seq 3	IoU Seq 4
Segmentation Output	0.88	0.915	0.802	0.887	0.860
3D Postprocessing	0.91	0.928	0.845	0.898	0.895
2D Postprocessing	0.92	0.930	0.848	0.902	0.896
Total improvement	4 pp	1.5 pp	4.6 pp	1.5 pp	3.6 pp

0% means IoU = 0 and corresponds to zero quality of segmentation, while an IoU of 100% means IoU = 1 (IoU varies between 0 and 1), corresponding to perfect match between the segmentation output and the corresponding groundtruth. As an example, if IoU increases from 86% to 92%, then pp is 6, meaning that segmentation quality improved by 6% points.

3. Results and analysis

This section shows the experimental results, which are analyzed in detail in the next section.

3.1. Timing and memory consumption

Table 7 shows the (minimum) training times for DeepLabV3, FCN and UNet to converge to a final validation accuracy (+-2%) on the CT dataset. It also shows the corresponding number of epochs, the final validation accuracy and the time taken to segment a new image (including also standard deviation). Additionally, Table 7 shows GPU memory used and post-processing time.

According to Table 7, segmentation training using training images and groundtruths took 159 min for the fastest approach (DeepLabV3). GPU memory consumption for DeepLabV3 (which was measured using Asus™ GPU TWEAK II tool™) was 7604 MB (average consumption; maximum consumption was 8014 MB, and minimum was 810 MB along the training session). After training, the time needed to segment a new

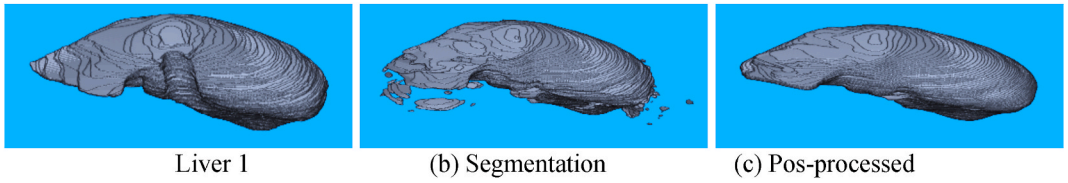


Fig. 10. Test liver 1 post-processing.

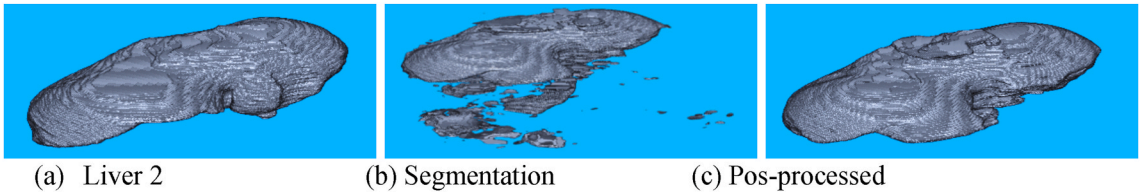


Fig. 11. Test liver 2 post-processing.

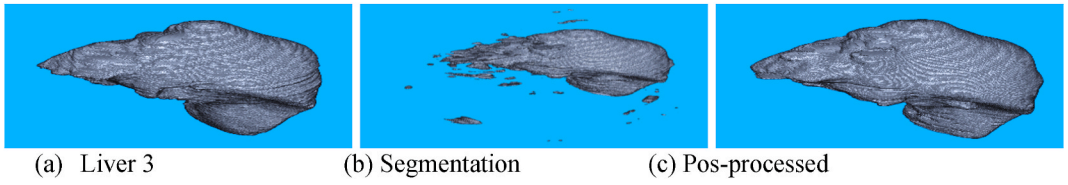


Fig. 12. Test liver 3 post-processing.

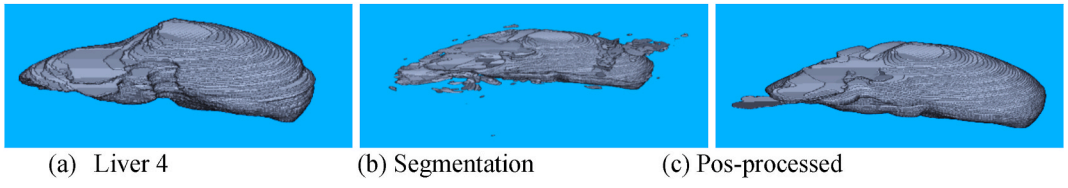


Fig. 13. Test liver 4 post-processing.

Table 11
Loss and post-processing improvements CT vs MRI data (pp = percentage points).

Loss	CT IoU liver	MRI IoU liver
crossE	0.88	0.86
Iou	0.9	0.88
Dice	0.91	0.88
Step	**CT IoU liver**	**MRI IoU liver**
Segmentation Output	0.88	0.86
3D Postprocessing	0.91	0.89
2D Postprocessing	0.92	0.91
Total improvement	4 pp	5 pp

image using DeepLabV3 was 0.093 (±0.005) secs. Post-processing time for an individual liver, which involves running the sequence of algorithms described in this work, was 8.7 (±0.9) secs.

3.2. Comparing networks

Table 8 shows the CT scores achieved by different networks using accuracy and IoU-based metrics.

DeepLabV3 converged much faster than the other networks, as shown in the converge times shown in Table 7, and also scored higher than the remaining network architectures, as shown in Table 8. In Table 8 metrics averaging over all pixels (global accuracy and weighted IoU) have very high scores for all networks, and the same happens with ROC and AUC of DeepLabV3 shown in Fig. 3. That is because more than 90% of all pixels are background, and a large fraction of the background is always well segmented because it includes all the parts of the image that are fairly constant. The conclusion is that global metrics should be avoided, instead we should analyze metrics that quantify the scores of the liver only. Also, we should not use accuracy, also shown in Table 8, because it does not consider false positives (background pixels classified as liver), a frequent occurrence.

Focusing on mean IoU in Table 8, we can see that DeepLabV3 is the top performing segmentation network (IoU score 88%). It achieves 28% points (pp) above Simple, which means that the advantage of using DeepLabV3 instead of the very basic one (Simple) is large. Most importantly also, FCN and UNet scored lower than DeepLabV3 (77% and 75%), but still 17 pp above Simple. The reason why DeepLabV3 is better is most probably related to its use of Resnet-18 as encoder instead of VGG-16 in the other two networks that we used, and also the innovations of ASPP and CRF that we discussed before in this work. As part of future work, we are currently investigating the details that lead to this difference.

3.3. Loss functions

Table 9 shows the scores of DeepLabV3 on CT data considering four different loss functions. Global metrics (metrics over all pixels) are reported together with mean metrics (average over the two classes liver and background) and also the IoU of the liver.

Table 9 shows the scores obtained by different loss functions. We can see again that global (pixel averaged) and accuracy metrics score very high usually, but as we explained those are reflecting quality of segmentation of the background mainly. Instead, we must focus on IoU of liver. We conclude from IoU of the liver that modification of the loss function to dice improved the liver score (84%) by 2pp when compared with the default cross entropy function (82%). The remaining loss functions, IoU, dice noBK and Tloss, did not improve compared to cross entropy (79 and 80% respectively).

3.4. Postprocessing

Table 10 shows the liver IoU achieved by each post-processing step. It also shows the improvement in four different test sequences (complete patient scans).

Table 10 shows the improvements that were obtained by running the post-processing steps, and Figs. 10–13 visualize those improvements. From the Figures it is very clear what post-processing does and why it contributed around 4% to improve the quality of segmentation. There are many imperfections new borders of the 3D models, the post-processing steps remove many of those imperfections by isolating the main volume effectively, also smoothing and filling holes. Table 10 shows that the total improvement was 4% points (pp) in average over all patient sequences, with the first 3D postprocessing improving 3 pp and 2D postprocessing improving an additional 1 pp. Since most errors and improvements are near borders, these improvements are very visible in the images. The results for four test liver scans in Table 10 and the corresponding images in Figs. 10–13 also show that some sequences are much better segmented than others (scores varied between 80% and 91.5%), consequently the post-processing operations also correct a lot more in some scans (4.6 pp and 3.6 pp) than in others (1.5 pp in two other liver scans).

Figs. 10–13 shows the corrections achieved by the post-processing algorithm for the same four test sequences visually, where the improvements are quite apparent.

Table 11 shows our result applying the same loss and post-processing steps to the MRI dataset. We reach similar conclusions on this data.

3.5. Experimental conclusions

The previous experiments have tested network architectures and modifications to improve quality of segmentation of liver CT scans. The improvements were the use of DeepLabV3 segmentation network, followed by testing different loss functions and post-processing operations to remove noise. These have improved IoU scores by 11 pp, 2 pp and 4 pp in average, respectively. DeepLabV3 was also able to converge much faster to a good solution.

4. Conclusions

In this work we have defined and evaluated different post-processing, loss and network architectures for segmentation of the liver in CT and MRI. We defined the precise post-processing operations that improve the quality of segmentation of organs. We also defined alternative loss functions and network architectures. Through experimental work we were able to show that the network architecture achieved an improvement of 11 pp by using DeepLabV3 instead of the more common UNet and FCN, choice of dice loss function improved 2 pp and post-processing improved 4 pp. Many works on advanced segmentation network architectures, including ensembles, are using UNet or other VGG-based networks as their basic building blocks. We conclude that DeepLabV3 with its residue-based encoders and ASPP features should be used as base architectures for future improvements. Dice should be used as loss function (2 pp improvement), and the post-processing algorithms that we defined in this work should be applied after segmentation of organs to achieve a further (4pp) improvement.

Future work challenges include integrating the post-processing steps as part of the final deconvolution stages and using DeepLabV3 as the base architecture for more advanced ensemble-based solutions, and testing the advanced solutions using DeepLabV3 as base network, dice as loss function and the postprocessing operations we defined in this work.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

For this work we used the CHAOS challenge dataset [27]. We would like therefore to thank the CHAOS challenge organizers for sharing the dataset. I also acknowledge my employer, University of Coimbra.

References

[1] Stephens DH, Sheedy PF, Hattery R, MacCarty R. Computed tomography of the liver. Am J Roentgenol 1977;128(4):579–90. 1977.
[2] Alfidi RJ, Haaga JR, Havrilla TR, Pepe RG, Cook SA. Computed tomography of the liver. American Journal of Roentgenol- ogy 1976;127(1):69–74. 1976.
[3] Chang C, Chen H, Chang Y, Yang M, Lo C, Ko W, Lee Y, Liu K, Chang R. Computer-aided diagnosis of liver tumors on computed tomography images. Comput Methods Progr Biomed 2017;145:45–51. 2017.
[4] Fuller S, Shafiei A, Ilanchezhian M, Bagheri M, DelRivero J. Tumor growth rate in metastatic adrenocortical carcinoma using two- dimensional Computed Tomography scan. J Clin Oncol 2019;37(15 suppl):e16 125. e16 125, 2019.
[5] Ronneberger O, Fischer P, Brox T. U-Net: convolutional networks for biomedical image segmentation. In: International Conference on Medical image computing and computer-assisted intervention. Cham: Springer; 2015. p. 234–41.
[6] Long J, Shelhamer E, Darrell T. Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2015. p. 3431–40.
[7] Chen L, Papandreou G, Kokkinos I, Murphy K, Yuille A. Deeplab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE Trans Pattern Anal Mach Intell 2017;40(4):834–48.
[8] Bereciartua A, Picon A, Galdran A, Iriondo P. Automatic 3D model-based method for liver segmentation in MRI based on active contours and total variation minimization. Biomed Signal Process Contr 2015;20:71–7. https://doi.org/10.1016/j.bspc.2015.04.005. 2015.
[9] Le N, Bao P, Huynh H. Fully automatic scheme for measuring liver volume in 3D MR images. Bio Med Mater Eng 2015;26(s1):1361–9. https://doi.org/10.3233/BME-151434. 2015.
[10] Huynh H, Le N, Bao P, Oto A, Suzuki K. Fully automated MR liver volumetry using watershed segmentation coupled with active contouring (2018). International journal of computer assisted radiology and surgery 2017;12(2):235–43. https://doi.org/10.1007/s11548-016-1498-9 PMID: 27873147. 2017.
[11] Zhou X, Takayama R, Wang S, Zhou X, Hara T, Fujita H. Automated segmentation of 3D anatomical structures on CT images by using a deep convolutional network based on end-to-end learning approach. In: Medical imaging 2017: image processing, vol. 10133. International Society for Optics and Photonics; 2017. p. 1013324.
[12] Bobo M, Bao S, Huo Y, Yao Y, Virostko J, Plassard A, Landman B. Fully convolutional neural networks improve abdominal organ segmentation. In: Medical imaging 2018: image processing, vol. 10574. International Society for Optics and Photonics; 2018. 105742V.
[13] Larsson M, Zhang Y, Kahl F. Deepseg: abdominal organ segmentation using deep convolutional neural networks. In: Swedish symposium on image analysis 2016; 2016.
[14] Chen Y, Ruan D, Xiao J, Wang L, Sun B, Saouaf R, Yang W, Li D, Fan Z. Fully automated multi-organ segmentation in abdominal magnetic resonance imaging with deep neural networks. arXiv preprint arXiv:1912.11000; 2019.
[15] Groza V, Brosch T, Eschweiler D, Schulz H, Renisch S, Nickisch H. Comparison of deep learning-based techniques for organ segmentation in abdominal CT images. In: 1st conference on medical imaging with deep learning. Amsterdam, The Netherlands: MIDL 2018; 2018. p. 1–3. 2018.
[16] Cai J, Lu L, Zhang Z, Xing F, Yang L, Yin Q. Pancreas segmentation in MRI using graph-based decision fusion on convolutional neural networks. In: Ourselin S, Joskowicz L, Sabuncu MR, Unal G, Wells W, editors. MIC- CAI 2016, vol. 9901. Springer, Cham: LNCS; 2016. p. 442–50. https://doi.org/10.1007/.978-3-319-46723-851.
[17] Conze P, Kavur A, Gall E, Gezer N, Meur Y, Selver M, Rousseau F. Abdominal multi-organ segmentation with cascaded convolutional and adversarial deep networks. arXiv preprint arXiv:2001.09521; 2020.
[18] Zhang X, Thibault G, Decencière E, Marcotegui B, Laÿ B, Danno R, Chabouis A, et al. Exudate detection in color retinal images for mass screening of diabetic retinopathy. Med Image Anal 2014;18(7):1026–43.
[19] Soille P. Morphological image analysis: Principles and applications. Springer-Verlag; 1999. p. 173–4.
[20] Gonzalez RC, Woods RE, Eddins SL. Digital image processing using MATLAB. Gatesmark Publishing; 2009.
[21] Haralick Robert M, Shapiro Linda G. Computer and robot vision, vol. I. Addison-Wesley; 1992. p. 158–205.
[22] Boomgard V, Van Balen R. Methods for fast morphological image transforms using bitmapped images. Comput Vis Graph Image Process: Graph Model Image Process May 1992;54(Number 3):254–8.
[23] Hu P, Wu F, Peng J, Bao Y, Chen F, Kong D. Automatic abdominal multi-organ segmentation using deep convolutional neural network and time-implicit level sets. International journal of computer assisted radiology and surgery 2017;12(3):399–411. 2017.
[24] Wang Y, Zhou Y, Shen W, Park S, Fishman E, Yuille A. Abdominal multi-organ segmentation with organ-attention networks and statistical fusion. Med Image Anal 2019;55:88–102. 2019.
[25] Gibson E, Giganti F, Hu Y, Bonmati E, Bandula S, Gurusamy K, Davidson B, Pereira S, Clarkson M, Barratt D. Towards image-guided pancreas and biliary endoscopy: automatic multi-organ segmentation on abdominal ct with dense dilated networks. In: Miccai. Springer; 2017. p. 728–36. 2017.
[26] Kim J, Lee J. Deep-learning-based fast and fully automated segmentation on abdominal multiple organs from CT. In: International forum on medical imaging in asia 2019. vol. 11050. International Society for Optics and Photonics; 2019. 110500K.
[27] Kavur A, Sinem N, Barıs M, Conze P, Groza V, Pham D, Chatterjee S, Ernst P, Ozkan S, Baydar B, Lachinov D, Han S, Pauli J, Isensee F, Perkonigg M, Sathish R, Rajan R, Aslan S, Sheet D, Dovletov G, Speck O, Nurnberger A, Maier-Hein K, Akar B, Unal G, Dicle O, Selver M. CHAOS challenge - combined (CT-MR) healthy abdominal organ segmentation. In: arXiv pre-print; 2020. https://doi.org/10.5281/zenodo.3362845. https://arxiv.org/abs/2001.06535CHAOS. data a.
[28] Salehi SS, Erdogmus D, Gholipour A. Tversky loss function for image segmentation using 3D fully convolutional deep networks. In: International workshop on machine learning in medical imaging; 2017. Springer: Cham, Switzerland.
[29] Chlebus G, Meine H, Thoduka S, Abolmaali N, van Ginneken B, Hahn H, Schenk A. Reducing inter-observer variability and interaction time of MR liver volumetry by combining automatic CNN-based liver segmentation and manual corrections. PloS One 2019;14:e0217228.
[30] Fu Y, Mazur T, Wu X, Liu S, Chang X, Lu Y, Harold H, Kim H, Roach M, Henke L, et al. A novel MRI segmentation method using CNN-based correction network for MRI-guided adaptive radiotherapy. Med Phys 2018;45:5129–37.