Volumetric Medical Image Segmentation with Deep Learning Pipelines

A THESIS PRESENTED TO THE FACULTY OF GRADUATE SCHOOL – UNIVERSITY OF MISSOURI-COLUMBIA

In partial fulfillment of
the requirements for the Degree
Master of Science

By:
IMAD EDDINE TOUBAL
Supervisor: Ye Duan

May, 2020

The undersigned, appointed by the dean of the Graduate School, have examined the thesis entitled:

VOLUMETRIC MEDICAL IMAGE SEGMENTATION WITH DEEP LEARNING PIPELINES

presented by: Imad Eddine Toubal,

a candidate for the degree of Master of Science,

and hereby verify that, in their opinion, it is worthy of acceptance.

Dr. Ye Duan, Associate Professor

Dr. Kannappan Palaniappan, Professor

Dr. Filiz Bunyak Ersoy, Assistant Research Professor

Acknowledgements

This thesis is dedicated to my mother, Ourdia Berouaken. Without her endless support and encouragement, I would have never been able to complete my graduate studies. I will forever be grateful for the sacrifices you have made to nurture my academic success.

This thesis is also dedicated to Dr. Duan whose assistance and guidance made this work possible.

Finally, I would like to thank my lab mate, Yuyan Li for sharing her expertise and being there for me as a friend. And with that, all the friends and connections I have made during my time at the University of Missouri.

Table of contents

List of Figures

List of Tables

Abstract

Automated semantic segmentation in the domain of medical imaging can enable a faster, more reliable, and more affordable clinical workflow. Fully convolutional networks (FCNs) have been heavily used in this area due to the level of success that they have achieved. In this work, we first leverage recent architectural innovations to make an initial segmentation: (i) spatial and channel-wise squeeze and excitation mechanism; (ii) a 3D U-Net++ network and deep supervision. Second, we use classical methods for refining the initial segmentation: (i) spatial normalization and (ii) local 3D refinement network applied to patches. Finally, we put our methods together in a novel segmentation pipeline. We train and evaluate our models and pipelines on a dataset of a 120 abdominal magnetic resonance – volumetric – images (MRIs). The goal is to segment five different organs of interest (ORI): liver, kidneys, stomach, duodenum, and large bowel. Our experiments show that we can generate full resolution segmentation of comparable quality to the state-of-the-art methods without adding computational cost.

Chapter 1: Introduction

1.1 Problem Statement

A fully automated segmentation of organs in abdominal medical images can enable a fast and efficient clinical workflow from diagnostics to treatment. For computer-assisted diagnostics and treatment, organ segmentation is a crucial first step [1].

Stereotactic MRI-guided online adaptive radiotherapy (SMART) [2, 3] is an effective treatment for the pancreas and other upper abdominal cancers. SMART allows precise delivery of escalated prescription dose to the abdominal tumor targets while avoiding the complications of radiation toxicity to the mobile gastrointestinal (GI) organs surrounding the tumor target. In the clinical workflow of SMART, manual segmentation of the GI orangs at risk (OARs) is one of the most important but also the most labor-intensive steps. Manual segmentation takes 10 minutes on average but ranges from 5 to 22 minutes [4]. The slow and costly manual segmentation step directly decreases the accessibility and affordability of online SMART and indirectly reduces the effectiveness of SMART due to intra-fractional body and organ movement of the patients.

Deep learning has given the rise to computer vision fields such as semantic segmentation using Convolutional Neural Networks (CNNs). These CNN-based segmentation methods have mainly focused on stable organs such as the brain and liver [5, 6, 7, 8]. Some digestive organs (stomach, bowel, and duodenum) present more of a challenge [9] due to their day-to-day instability (depending on different food consumed as well as the digestion process).

Advancements made in convolutional neural networks for semantic segmentations present an opportunity to readdress this challenge.

1.2 Contribution of the Thesis

In this study we explore a deep learning pipeline for semantic segmentation of 3D MRI images of the abdominal area. To sum up the main contributions of this work:

1. We leverage state-of-the-art CNN techniques to improve on fully convolutional networks and their segmentation quality.
2. We exploit voxel properties beyond the intensity value for a superior local segmentation quality.
3. We explore different pipelines that use the developed networks alongside data normalization and up-scaling techniques to achieve a high-resolution segmentation.

Chapter 2: Related Work

2.1 Medical Image Segmentation

Due to the essential role of medical image segmentation in computer-aided diagnosis system, it has pushed forward the research and development of new computer vision and image processing techniques [10].

Advancements in medical imaging such as microscopy, dermoscopy, X-ray, ultrasound, computed tomography (CT), magnetic resonance imaging (MRI), and positron emission tomography (PET) inspires researchers to employ an image segmentation pipeline to automatically extract regions of interest (ROI). Based on the application, ROI extraction can be organ/tissue segmentation, tumor/mass segmentation (within organs).

Both classical methods of medical image segmentation [11] as well as deep-learning-driven methods [12] are deployed in today's segmentation pipeline. However, it is inevitable that convolutional neural networks (CNNs) are what is driving the progress in the field.

2.2 Convolutional Neural Networks for Semantic Segmentation

Semantic segmentation in image processing is the task of assigning each pixel of an image (or each voxel in a volumetric image) a class. Most CNNs used in the field are based on the two main deep learning architectures for medical image segmentation: fully convolutional networks (FCN) [13] and U-Net [14].

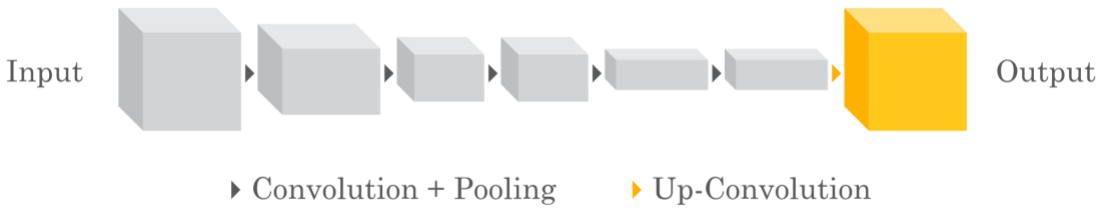


Figure 2.1 Fully convolutional network (FCN) architecture for semantic segmentation

FCN proposed single-step up-sampling (transpose-convolving) the output of multiple stacked convolutional layers. U-Net, on the other hand, proposed an encoder-decoder architecture that enabled multi-step up-sampling. Both of these methods included skip connections within the intermediate feature maps in order to improve prediction.

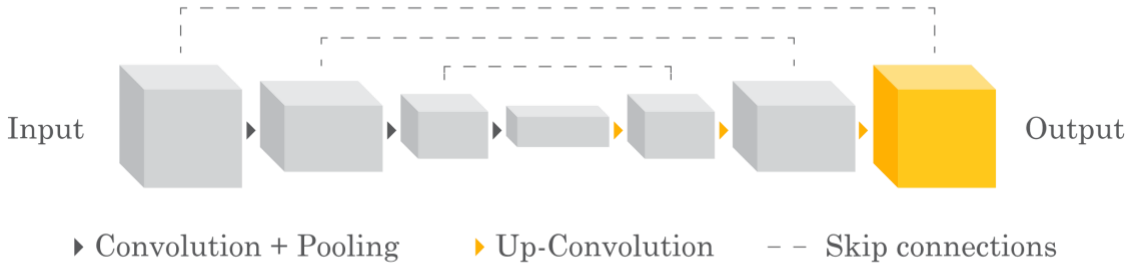


Figure 2.2: A simplified illustration of U-Net architecture with skip connections.

These two main methods have seen many iterations and revision in order to solve semantic segmentation problems in a variety of application. An extension of U-Net, for instance, is U-Net++ [15] where they introduced a core complex architecture that enables deep supervision (as can be seen in Figure 2.3).

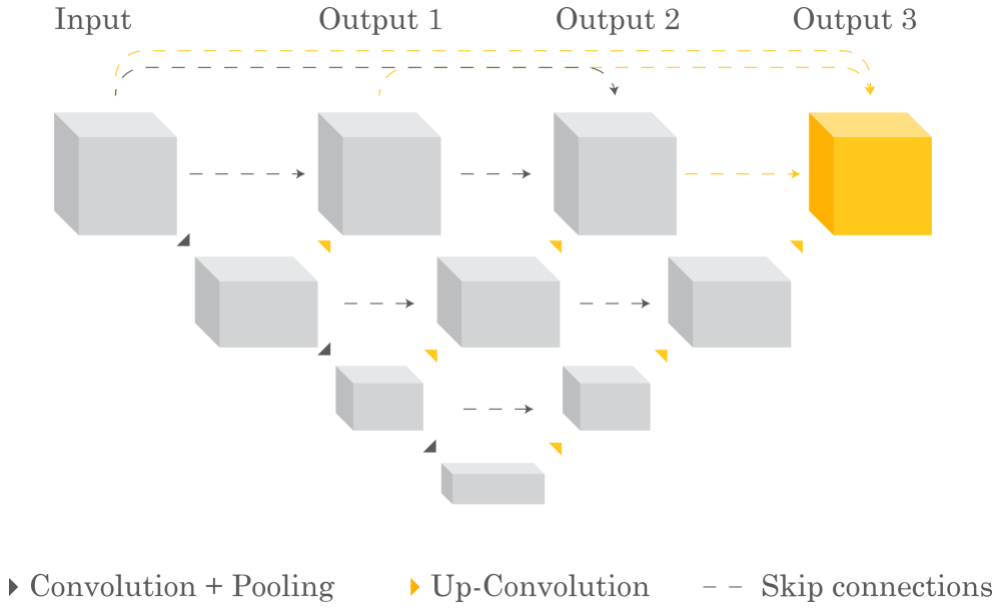


Figure 2.3 A simplified illustration of U-Net++

The re-designed architecture aims at reducing the semantic gap between the feature maps of the encoder and decoder sub-networks.

2.3 CNN-based Correction Network

For the dataset that is used in this work, the state-of-the-art work by Fu, Y. et. Al. [9] uses a "CNN-based correction network"; also referred to as "CNN+Correction Network" later in this thesis. Their proposed DL model contains a voxel-wise label prediction CNN and a correction network which consists of two sub-networks. The prediction CNN and sub-networks in the correction network each have a similar architecture, although their parameters are independent. The sub networks include a single deep "dense block" that consists of twelve densely connected convolutional layers

The correction network was designed to improve the voxel-wise labeling accuracy of a CNN by learning and enforcing implicit anatomical constraints in the segmentation process. Figure 2.1 Shows the general architecture and flow of data for CNN+Correction Network.

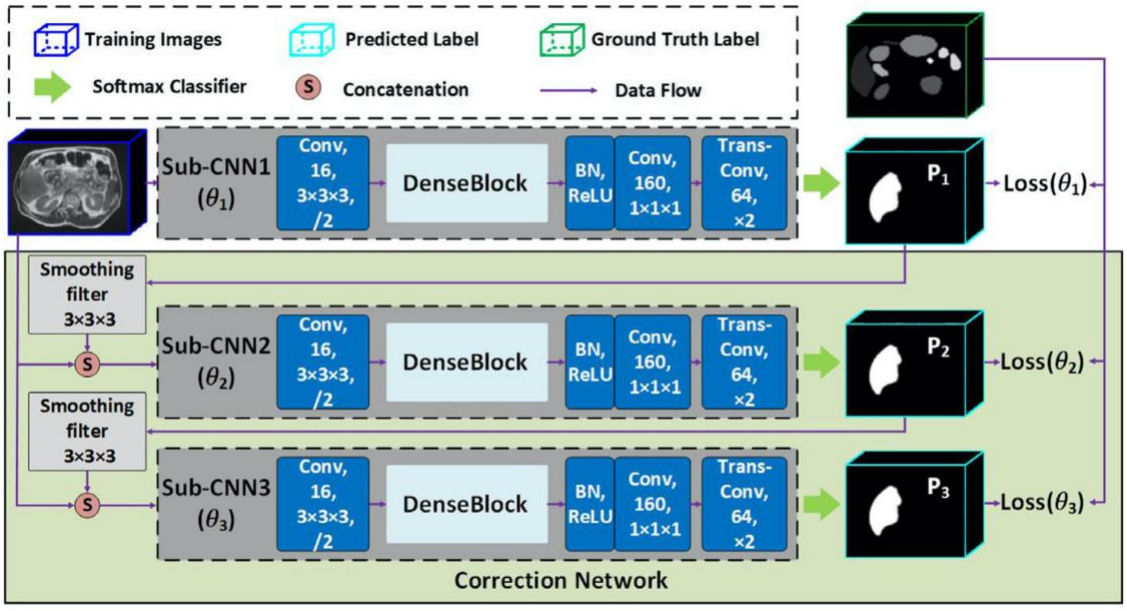


Figure 2.4 CNN+Correction Network architecture.

Figure 2.5 shows the architecture of the "dense block" used in this work.

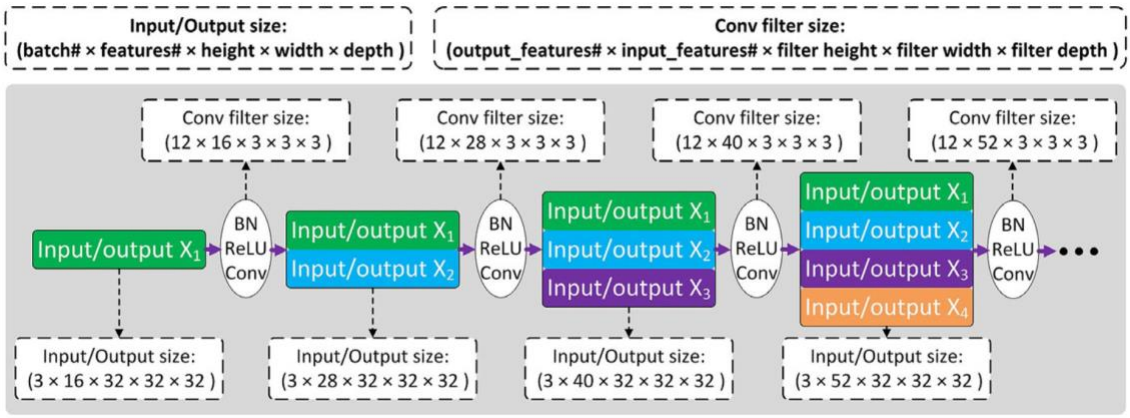


Figure 2.5 An example of a dense block of 4 layers.

Its sub-networks learn to fix the erroneous classification of its previous network by taking as input both the original images and the Softmax probability maps generated from its previous sub-network.

2.4 Attention and Gating Mechanisms

Attention idea in deep learning has been one of the most influential ideas in the past years. The concept was originally developed for neural machine translation applications [16], but it then spread to image processing applications such as image analysis, natural language processing (NLP) for image captioning [17], classification [18, 19], and semantic segmentation [20].

The integration of attention mechanisms in deep neural networks improved the quality of semantic segmentation by enhancing the networks' representational capabilities. Many advancements have been made for medical images [20], as well as natural images [21].

2.5 Squeeze-and-Excitation Networks

As a way to enhance the quality of special encoding through the CNN feature hierarchy, "Squeeze-and-Excitation" (SE) blocks [22] were first introduced in

the literature in 2017. The channel-wise SE block introduced can be seen in Figure 2.6.

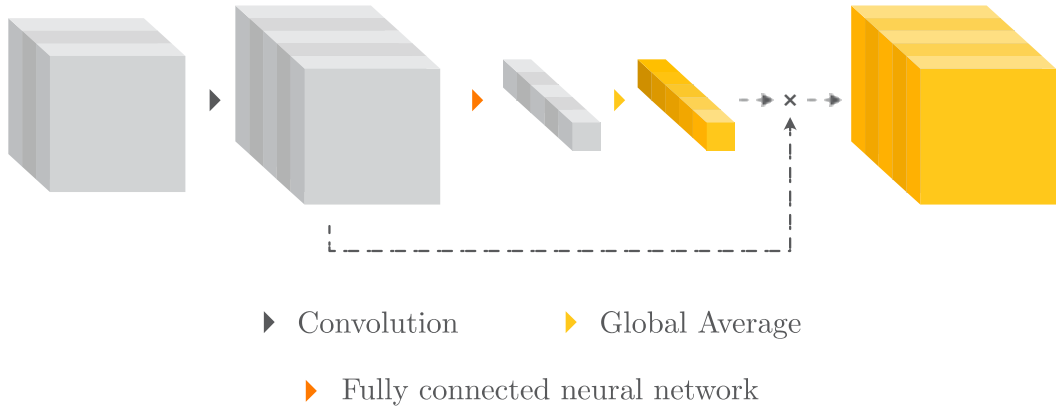


Figure 2.6 Squeeze and Excite Block

The goal is to explicitly model interdependencies between channels, with the aim to recalibrate channel-wise features in the feature maps. In a normal convolutional layer scenario, each of the learned kernels operates with a local receptive field, and therefore each pixel/voxel in the output feature map represents only the contextual information inside the local region defined by the kernel size. This issue is tackled by concurrently squeezing global special information into a channel descriptor using average pooling. After which, a gating mechanism with sigmoid activation is applied to the channel descriptor, and finally, multiplied by the convolution output to rescale it (channel-wise). SE has demonstrated its utility across multiple image processing tasks like classification, object detection and scene classification.

Concurrent Spatial and Channel Squeeze-and-Excitation [23] introduced other variants of SE to tackle the task of semantic segmentation in fully convolutional neural networks. As an extension to channel-wise SE (cSE), space-wise SE (sSE) was suggested along with concurrent spatial and channel squeeze-and-excitation (scSE) as can be seen in Figure 2.7. This work drew inspiration from previously mentioned SE to use the same concepts in spatial,

as well as spatial + channel wise squeezing. This work was evaluated on brain segmentation on MRI brain scans and organ segmentation on whole body CT scans. This novel method showed a consistent improvement over the classic architectures like DenseNet [24] and U-Net.

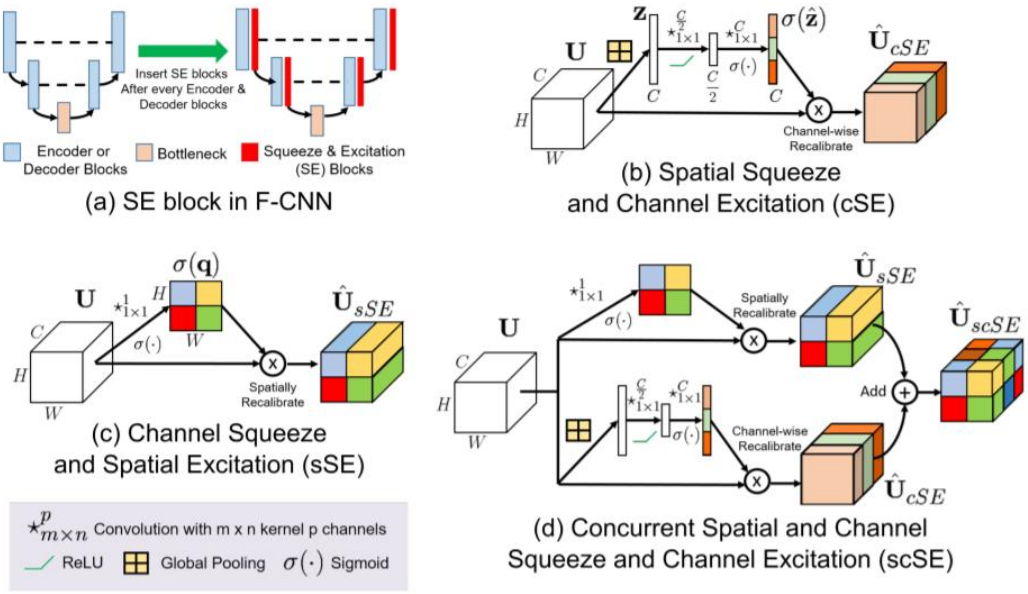


Figure 2.7 Concurrent Spatial and Channel Squeeze-and-Excite [23]

Chapter 3: Methods

This chapter first describes the dataset [9] used to train and evaluate our models. The deep learning models implemented and tested on this dataset are then discussed in detail.

3.1 Dataset

3.1.1 Dataset description

Dataset used for this project is composed of 120 volumetric MRI images [9]. The in-plane resolution of the images was 1.5 × 1.5 mm2 and a slice thickness of 3 mm. The organs were manually annotated by drawing an outline (contouring) the liver, kidneys, stomach, duodenum, and bowel for a total of five organs. The contours went through a thorough quality assurance by multiple trained professionals in order to ensure a high quality. The contouring process was highly time consuming but essential to achieve an accurate segmentation, and eventually, to train a high-quality network.

The final volumetric images after preprocessing are 256 × 256 × 64 voxels. The preprocessing essentially consisted of cropping to only keep the relevant region containing the organs of interest.

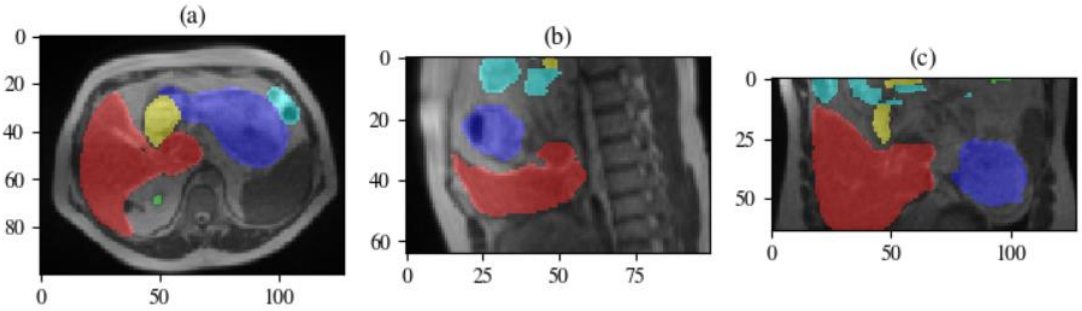


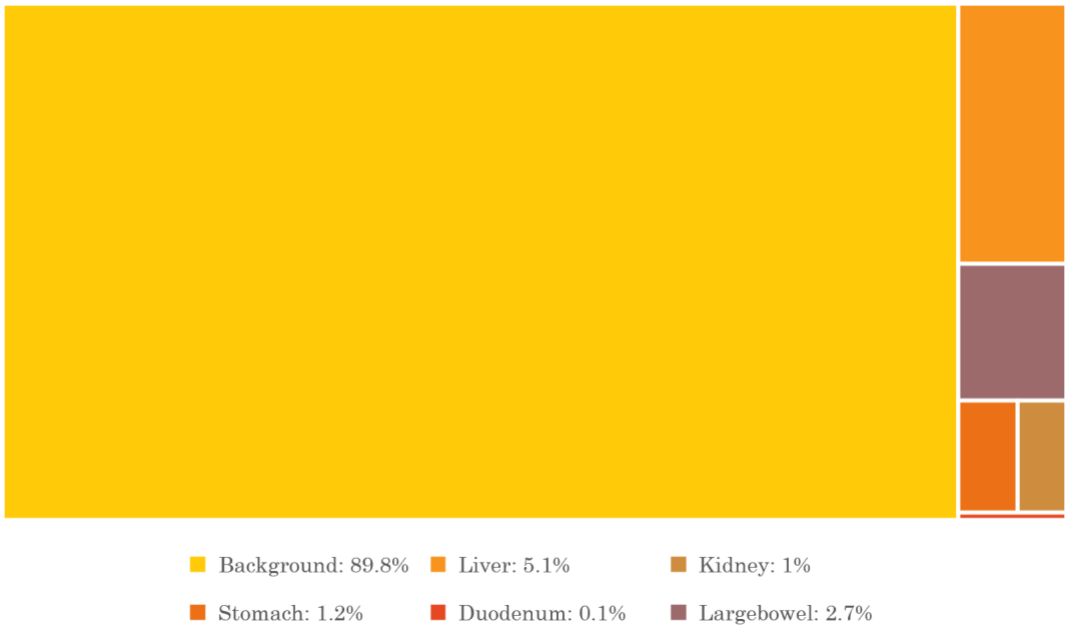
Figure 3.1: Orthogonal views of a volumetric MRI image. A slice view along the z, y, and x dimension is shown in (a), (b), and (c) respectively. The different organs/classes are highlighted in different colors superposed on the greyscale MRI image.

3.1.2 Training, validation, and testing

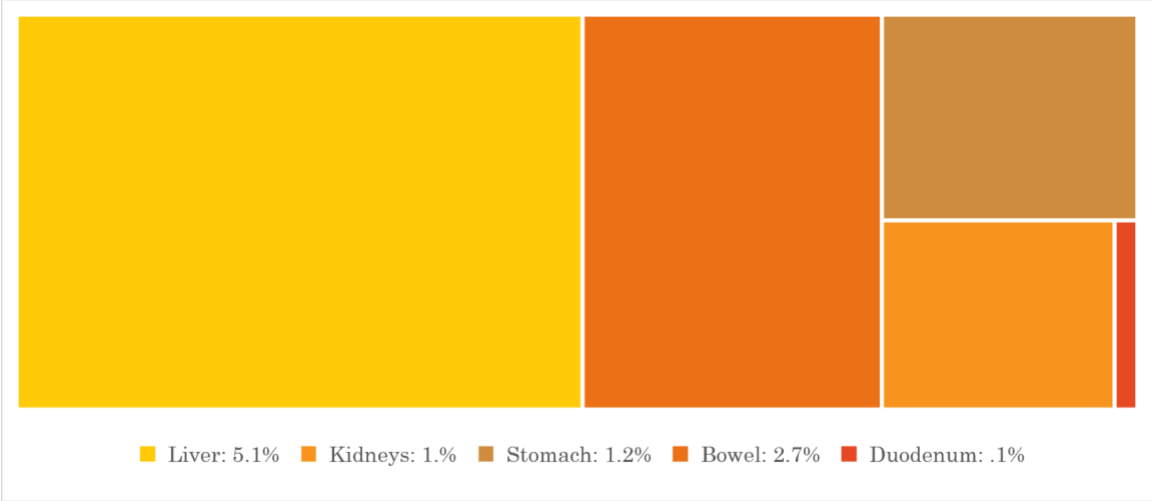
The dataset is split into subsets of 100, 10, and 10 images that were used for training, validation, and testing respectively. The results on the testing data is reported.

3.1.3 Dataset distribution

Given the different anatomical structures of the different abdominal organs (size, shape, complexity…etc.), the final volumetric images contain an unbalanced representation of the different classes as can be observed in Figure 3.2.



(a)

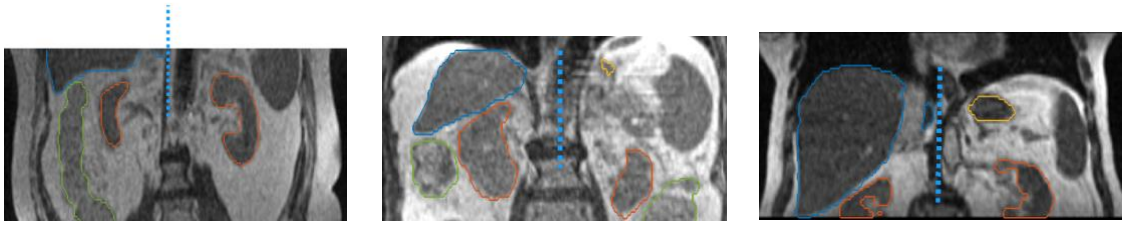


(b)

Figure 3.2: Class (organ) representation in the dataset (measured by the number of voxels belonging to which class). (a) Taking into consideration the background voxels. (b) Disregarding the background class for a better comparison between organ sizes.

3.2 Spatial Normalization

The advantage there is to the used MRI dataset is the uniformity that it provides; the relative positions and the general shape of the organs of interest are always going to be similar for different data samples. This helps the training process as the network can remember the general landmarks in the abdominal MRI image. However, as can be seen in Figure 3.3, the acquired dataset may have a slight shift in position or scale across samples.



(a) (b) (c)

Figure 3.3 illustrates how to manually and roughly estimate the distance (shown in blue dotted lines) from the axial plane of the origin (the middle point between two kidneys) to the top slice of the liver. The top of the liver is not visible in (a), the distance is therefore roughly estimated. The bottom of the right kidney (at the left side) is not visible in (c), therefore the origin point is estimated.

In order to further exploit this property of the data, a spatial normalization [25] procedure has been used. This process involves building a coordinate system that ensures uniformity across data samples.

Spatial normalization across data samples involves:

1. Identifying the origin o^p, a reference point in each image volume that are stable and can be easily identified for any patient p.
    - We use the middle point of the two kidneys as the cross-patient reference point.

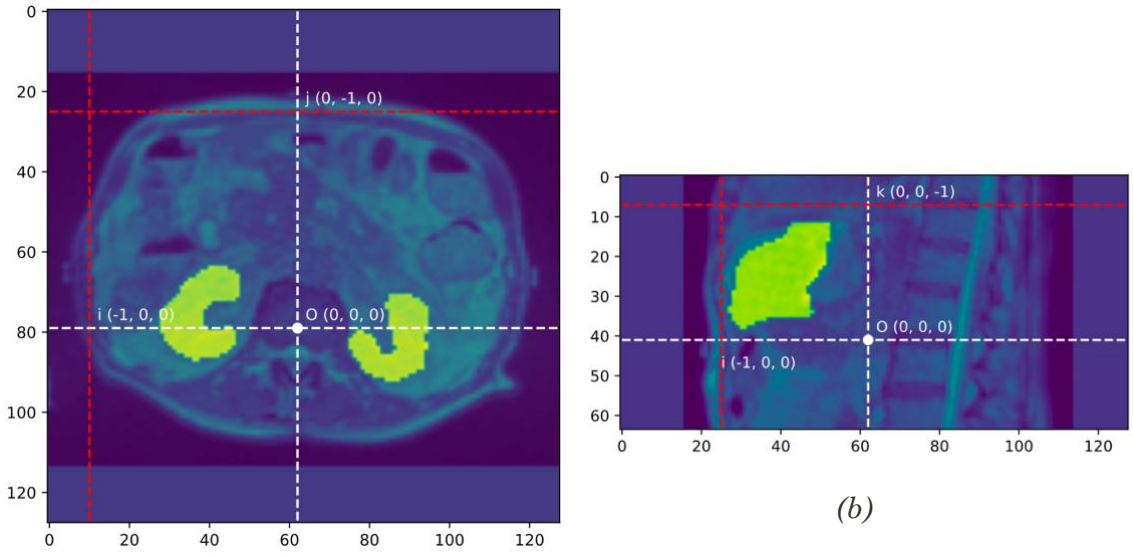
$$\overrightarrow{o^p}\begin{pmatrix} o_x^i \\ o_y^i \\ o_z^i \end{pmatrix} = \frac{\overrightarrow{c_1^i} + \overrightarrow{c_2^i}}{2}$$

      where: $\overrightarrow{c_1^i}$, and $\overrightarrow{c_2^i}$ are the centroids of the first and second kidneys. This is visually described in Figure 3.4.
    - If the kidneys are defined, the origin coordinates can be automatically calculated. However, for data samples with a single kidney, the x component of the origin o_x^i is moved to the mean x component of all healthy data samples (with both kidneys)

$$o_x^i = \underset{j}{\text{mean}}\left(o_x^j\right)$$

      Where j belongs to the list of indices indicating healthy data samples.

2. Defining the patient body cavity size on the axial slice at the origin.
3. Defining the patient vertical size, from the patient origin to the top slice of the liver.



(a) *(b)*

Figure 3.4 An illustration of the constructed coordinate system in a axial views of a sample image. (a) highlight the x-y component of the coordinate system (kidneys are highlighted for reference); whereas (b) highlights the x-z component (liver is highlighted for reference)

Finally, for each image voxel (with intensity $v_{i,j,k}$), we compute the $x_{i,j,k}$, $y_{i,j,k}$ and $z_{i,j,k}$ coordinates, with 0 value at the origin, and values normalized to the use height and body cavity size.

If we assume the slicing of a volume image $X = [v_{1,1,1}, v_{1,1,2}, \dots, v_{i,j,k}, \dots, v_{W,H,D}]$, a normalized image $Z \in \mathbb{R}^{W \times H \times D \times 4}$ is an augmented version of X where each component is concatenated with the voxel's position in the new coordinate system:

$$Z = \left[\begin{bmatrix} v_{1,1,1} \\ x_{1,1,1} \\ y_{1,1,1} \\ z_{1,1,1} \end{bmatrix}, \begin{bmatrix} v_{1,1,2} \\ x_{1,1,2} \\ y_{1,1,2} \\ z_{1,1,2} \end{bmatrix}, \quad \dots \quad , \begin{bmatrix} v_{W,H,D} \\ x_{W,H,D} \\ y_{W,H,D} \\ z_{W,H,D} \end{bmatrix} \right]$$

3.3 U-Net

As a backbone architecture, we have used a U-Net architecture [14] that consists of an encoder-decoder with skip connections as described in Figure 3.5. In this research, exploring U-Net was favored over fully convolutional network

architectures (FCNs) as it provides more spatial resolutions, thus, suggesting a better multi-scale feature extraction.

The architecture provides five (5) spatial resolutions. The encoder part of the network follows a traditional CNN; it is composed of five convolutional (dense) blocks in five different resolutions. Each convolutional block consisting of two densely connected convolutional layers as shown in Figure 3.6. After each convolutional block of the encoder, the resolution of feature map is reduced by factor of 2 in all dimensions (width, height and depth) using max-pooling {1, 1/2, 1/4, 1/8, 1/16}; whereas the feature channels count is doubled {16, 32, 64, 128, 256}.

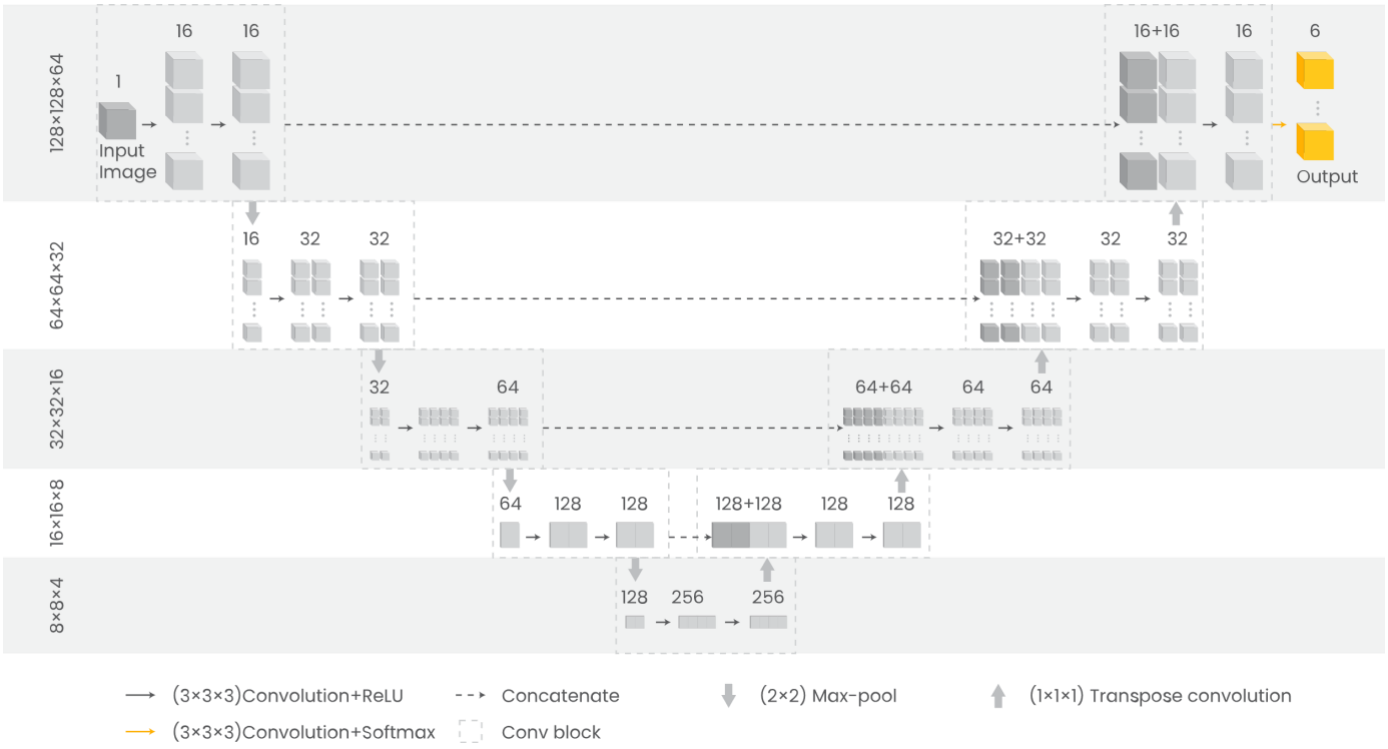


Figure 3.5 U-Net Architecture for volumetric images. Boxes indicated with dotted lines represent "convolutional blocks". The output of each spatial level in the encoder is concatenated to the input of the corresponding spatial scale in the decoder part.

In the decoder part of the network, the blocks are similar to the encoder part. The difference being in doubling resolution after each block using a transpose convolution of size $2 \times 2 \times 2$. Additionally, as previously mentioned, the output

of the encoder of the same spatial resolution is concatenated to the input of the decoder convolutional block.

The transpose convolution used for up-sampling in the network is sometimes replaced by simple up-sampling in other parts of this research in order to:

1. Decrease the number of learned parameters, and
2. Avoid the checkerboarding effect [26] at the output map that is usually caused by the transpose convolution operations.

The output of the final convolutional block in this network is then fed into a final convolution operation with a Softmax activation function to produce the final segmentation output Y. The final output of this network is a single volumetric image of 6 channels, each channel representing an organ of interest: {Background, Liver, Kidneys, Stomach, Duodenum, Large bowel}.

3.4 Dense Blocks

Let's consider an input feature map of channel size C, or $X \in \mathbb{R}^{W\times H\times D\times C}$. This input is passed through a convolutional layer h_1 to produce the feature map $H_1 = h_1(X)$ where $H_1 \in \mathbb{R}^{W\times H\times D\times 2C}$. H_1 is concatenated to the input again and then passed through a second convolutional layer h_2 to produce a feature map $H_2 = h_2([X, H_1])$. The final convolution takes the concatenation of $X, H_1,$ and H_2 as an input as pass it through a third convolutional layer h_3. The final output is then:

$$\hat{X} = h_3([X, H_1, H_2]) = h_3([X, h_1(X), h_2([X, h_1(X)])]) \quad (1)$$

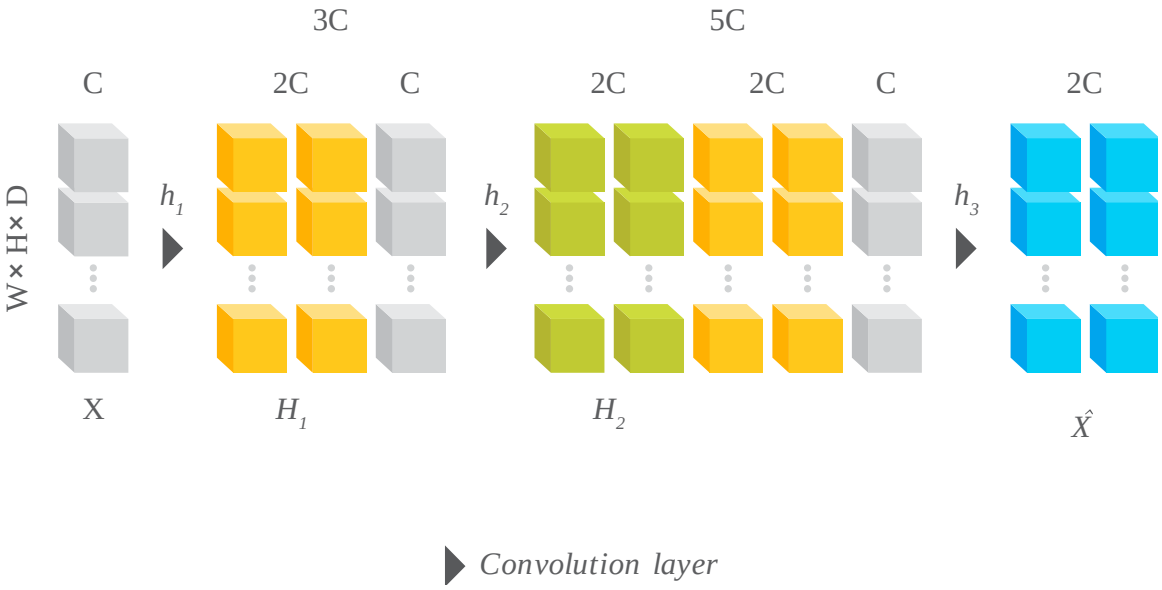


Figure 3.6 Dense Convolutional Block: The input as well as the outputs of each convolution operation is a tensor of rank 4 ($W \times H \times D \times C$). For the sake of simplicity, it is divided into C 3D tensors (cubes), distributed in a grid fashion and color coded for clarity. The number of channels is displayed on top of each tensor. The name of each tensor is annotated at the bottom.

Each convolution operation in the block is a 3D convolution W_i of size $C \times 3 \times 3 \times 3 \times 2C$ and padding of $1 \times 1 \times 1$ to match the input feature map size; followed by a rectify linear unit (ReLU) [27] referred to mathematically as the function δ.

$$h_i(X) = \delta(W_i * X) \tag{2}$$

The output of the dense block $\hat{X}$ is feature map of double the input channel count. $\hat{X} \in \mathbb{R}^{W \times H \times D \times 2C}$. A visual representation of the dense block is shown in Figure 3.6.

3.5 Squeeze and Excitation

3.5.1 Spatial Squeeze and Channel Excitation Block (cSE)

If we assume the slicing of the feature map $X = [X_1, X_2, \ldots, X_C]$ where $X_i \in \mathbb{R}^{W \times H \times D}$ is the volumetric map associated to the i^{th} channel. Spatial squeeze is essentially performing a global average on each volumetric image X_i for $i \in$

$\{1, 2, \dots, C\}$. The output of this operation is a vector $\mathrm{z} = [z_1, z_2, \dots, z_C]$ with each component $z_i \in \mathbb{R}$ is calculated using the following formula:

$$z_i = \frac{1}{W \times H \times D} \times \sum_{i}^{W} \sum_{j}^{H} \sum_{k}^{D} X_i(i, j, k) \tag{3}$$

The vector z is then fed through a two-layer multiplayer perceptron (MLP) [28] with sigmoid activation function σ applied after each layer. The output of the MLP is calculated as follows:

$$\hat{\mathrm{z}} = \sigma\big(W_2 \bullet \delta(W_1 \bullet \mathrm{z})\big) \tag{4}$$

Where $W_1 \in \mathbb{R}^{C \times \frac{C}{2}}$ and $W_2 \in \mathbb{R}^{\frac{C}{2} \times C}$ are the weights associated with the MLP. The vector $\hat{\mathrm{z}} \in \mathbb{R}$ encodes the channel-wise dependencies. Since the final activation of the MLP is a sigmoid function, we ensure that the values $\hat{z}_i$ are between 0 and 1. The encoded vector $\hat{\mathrm{z}}$ is then used to weigh the feature map X. The output of the cSE block is calculated as follows:

$$\hat{X}_{cSE} = f_{cSE}(X) = [\hat{z}_1 X_1, \hat{z}_2 X_2, \dots, \hat{z}_C X_C] \tag{5}$$

An illustration of the cSE block architecture is shown in detail in Figure 3.7.

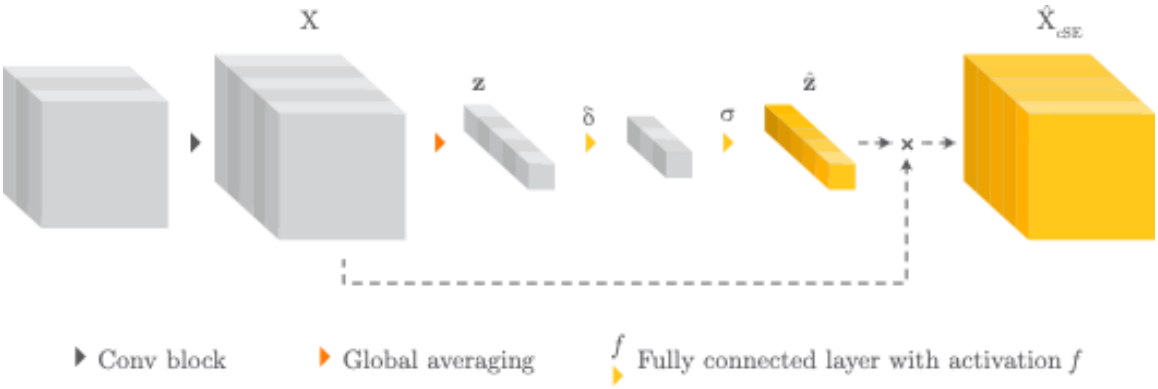


Figure 3.7 Spatial squeeze and channel excitation block (cSE)

The network learns the activations $\hat{\mathrm{z}}$ adaptively to in order to give weights to each channel of the feature map with respect to the channel's importance.

3.5.2 Channel Squeeze and Spatial Excitation (sSE)

The channel squeeze and spatial excitation channel (sSE) block squeezes the feature map channel-wise, then excites spatially. If we consider again the same slicing of the feature map $X = [X_1, X_2, \dots, X_C]$ where $X_i \in \mathbb{R}^{W \times H \times D}$. The squeeze operation produces a single channel feature map q through a convolutional layer with sigmoid activation h_{sq}.

$$\mathrm{q} = h_{sq} = \sigma(W_{sq} * X) \tag{6}$$

Where $W_{sq} \in \mathbb{R}^{C \times 3 \times 3 \times 3 \times 1}$ is the learned convolution kernel. For the excitation part of this operation, the feature map q is element-wise multiplied with each channel of the input feature map X as follows:

$$\hat{X}_{sSE} = f_{sSE}(X) = [\mathrm{q} \circ X_1, \mathrm{q} \circ X_2, \dots, \mathrm{q} \circ X_C] \tag{7}$$

Channel squeeze and spatial excitation block (sSE) will learn during the training to highlight the most relevant spatial locations in a feature map. An illustration of the sSE block is shown in Figure 3.8.

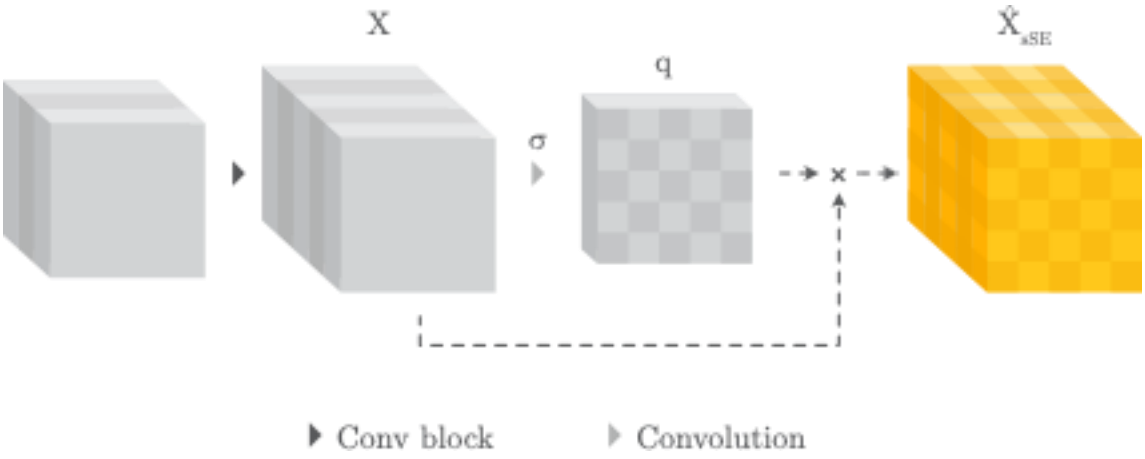


Figure 3.8 Channel squeeze and spatial excitation block (sSE)

3.5.3 Spatial and Channel Squeeze and Excitation (scSE)

Finally, the full spatial and channel squeeze and excitation module is illustrated in Figure 3.9. This block uses a combination of sSE and cSE block by adding the output feature maps of both operations:

$$\hat{X}_{scSE} = \hat{X}_{cSE} + \hat{X}_{sSE} \tag{8}$$

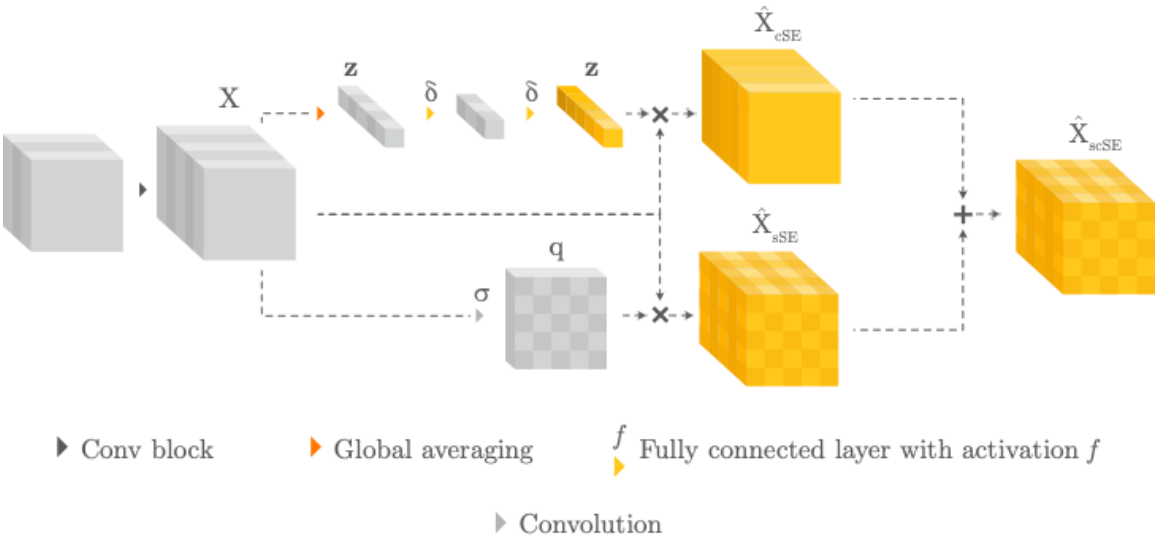


Figure 3.9 Spatial and Channel Squeeze and Excitation (scSE)

A the activation of voxel in location (i, j, k, c) of the input feature map X is higher when it gets high importance from both squeeze and excitation blocks. This recalibration encourages the network to learn more meaningful feature maps, that are relevant both spatially and channel-wise.

3.6 U-Net++ and Deep Supervision

Similar to U-Net, U-Net++ is an encoder-decoder style architecture. However, it is augmented with multiple nested convolutional blocks that are connected as seen in Figure 3.10. The main encoder-decoder is referred to as the "backbone"; instead of concatenating the output of each encoder to the input of its equivalent decoder convolutional block, a number of intermediate convolutional blocks are introduced.

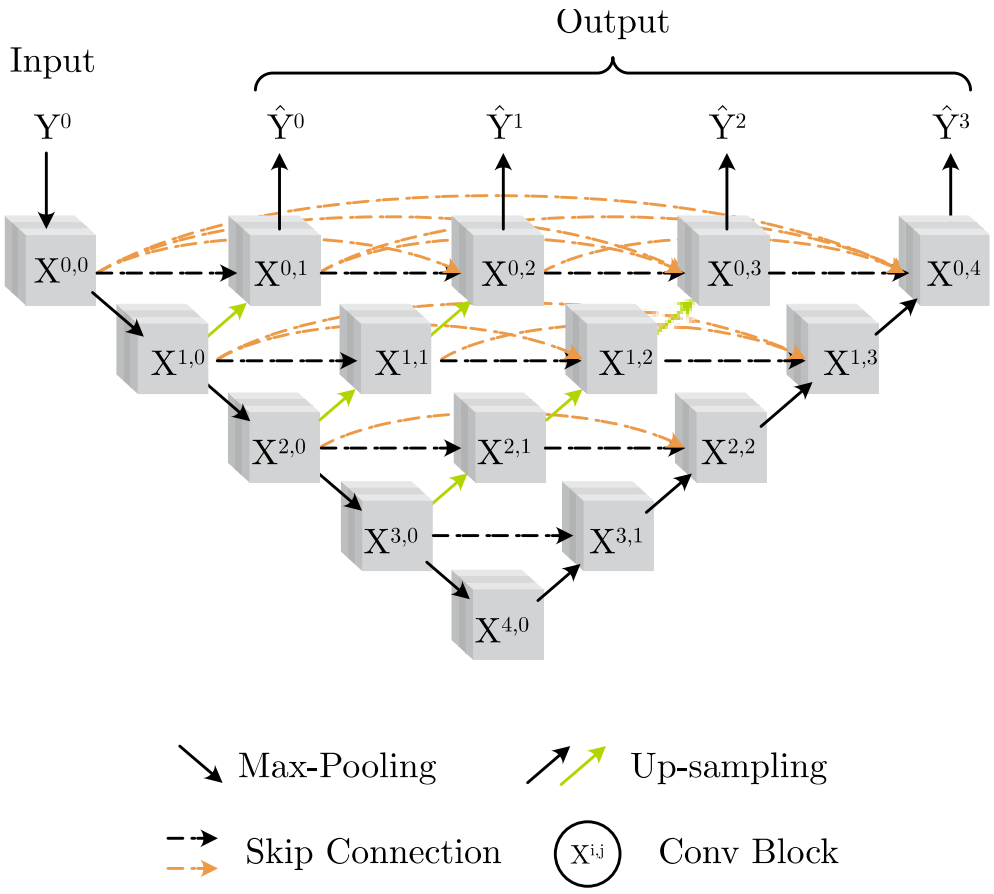


Figure 3.10 An overview of the used U-Net++ architecture. Each convolutional block is represented by a circle. The input image X is fed to the first convolutional block $X^{0,0}$; $\hat{Y}^i$ for $i \in \{1, 2, 3, 4\}$ are the outputs of this architecture.

The intermediate blocks take as input the output of encoder block of the same spatial resolution, concatenated with the output of the previous intermediate block, and the output of the next smaller spatial resolution (scaled-up). This can be seen in Figure 3.11.

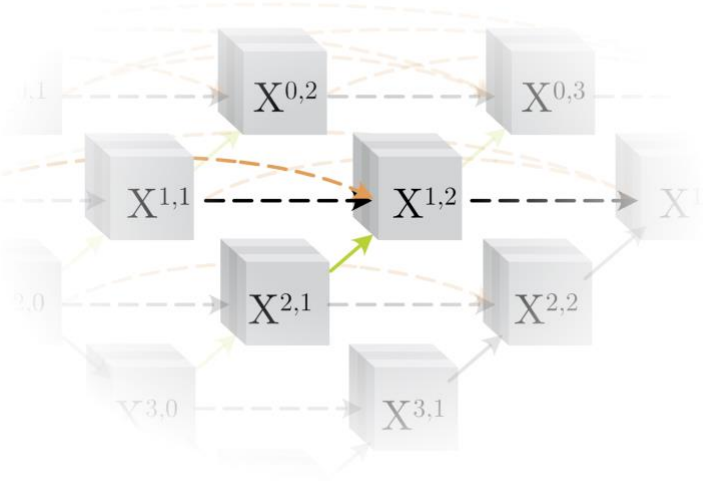


Figure 3.11 Intermediate convolutional block in U-Net++

The total loss for this network is calculated through adding the losses associated with each output. The total loss equation then becomes:

$$\mathcal{L} = \sum_{i}^{m} \mathcal{L}_f\left(Y, \hat{Y}^i\right) \tag{9}$$

Where:

- $\mathcal{L}_f$: The loss function.
- $\hat{Y}^i$: The i^{th} output segmentation of the network.
- Y: The ground truth segmentation.

3.7 GlobalSegNet: Dense U-Net++ with Squeeze and Excitation

As a way to improve the accuracy of the high performing U-Net++, a Spatial and Channel Squeeze and Excitation (scSE) module was added as an extra layer to each convolutional block (except for the bottleneck) as can be seen in Figure 3.12.

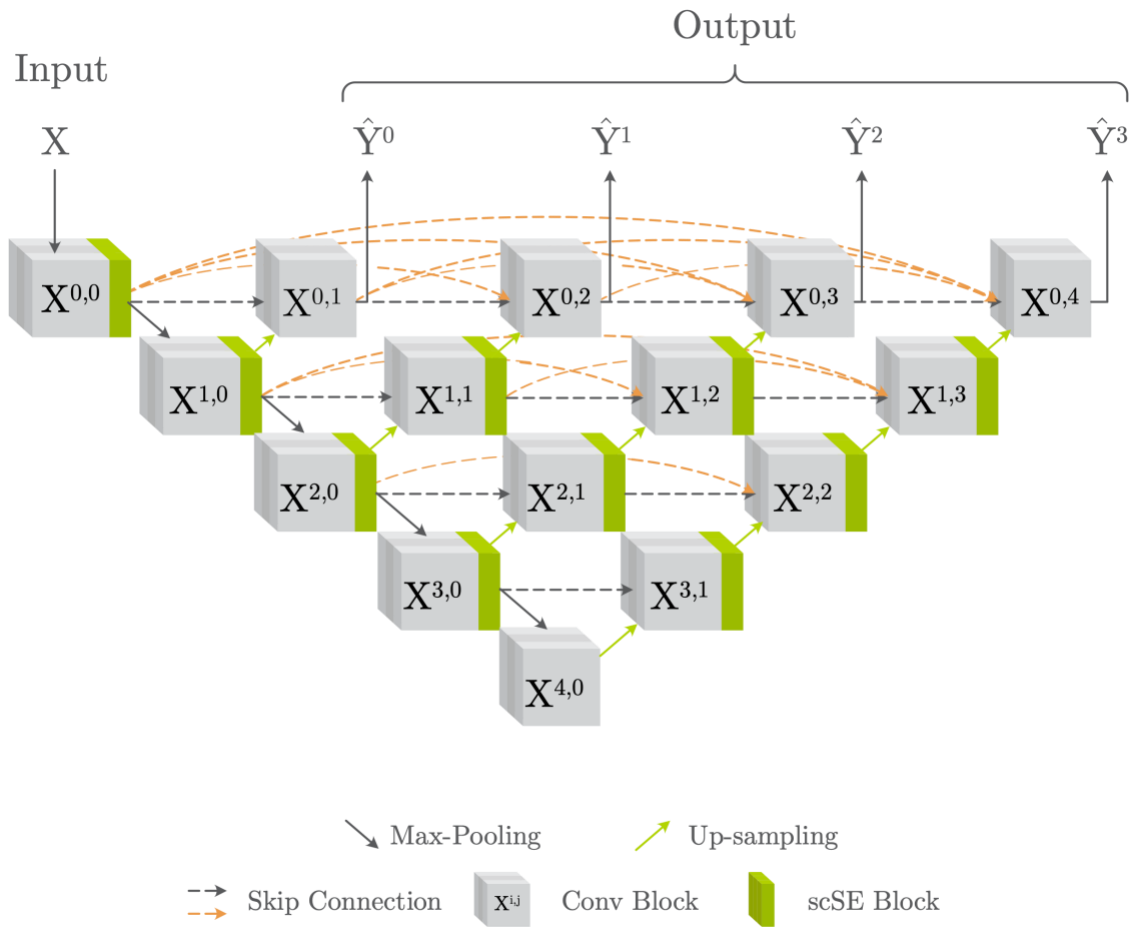


Figure 3.12 Dense U-Net++ with Squeeze and Excitation (U-Net++ w/ scSE)

The conducted experiments – as well as other work on squeeze and excitation mechanisms in the literature – pointed that a better training performance is achieved by leaving the bottleneck without a scSE block. As previously mentioned in U-Net++, this model is trained by minimizing the loss function described in Equation (9).

3.8 Models' Computational Complexity

3.8.1 Conv Block complexity

If we consider the convolutional block (shown in Figure 3.6). The number of parameters for such block is calculated as follows:

$$\#\mathrm{P}_{encoder} = (C+1)\times 3\times 3\times 3\times 2C + (3C+1)\times 3\times 3\times 3\times 2C + (5C+1)\times 3\times 3\times 3\times 2C \quad (10)$$

In the case of a decoder block, since the output of the corresponding encoder block is also used as in input, the equation becomes:

$$\#\mathrm{P}_{decoder} = (\mathbf{2C}+1)\times 3\times 3\times 3\times 2C + (3C+1)\times 3\times 3\times 3\times 2C + (5C+1)\times 3\times 3\times 3\times 2C \quad (11)$$

3.8.2 scSE Block complexity

Let us consider the spatial squeeze and channel-wise excitation first (cSE). The spatial squeeze is an average pooling process; therefore, no learned parameters are associated with this operation. The equation for the number of parameters of two-layer MLP in terms of the number of channels C is given as follows:

$$\#\mathrm{P}_{cSE} = (C+1)\times\frac{C}{2} + \left(\frac{C}{2}+1\right)\times C \quad (12)$$

Next, let us consider the channel-wise squeeze and spatial excitation (sSE). The channel-wise squeeze is a $(1\times 1\times 1)$ convolution operation with an output channel size of 1. The kernel of this convolution operation is the only learnable parameter of sSE block; therefore, the number of parameters association with sSE block is given by the equation below:

$$\#\mathrm{P}_{sSE} = C+1 \quad (13)$$

3.8.3 Architecture summaries of the used models

A comparison of the models used in this research in terms of the number of parameters as well as the number of operations is described in Table 3.1.

	Model		**Model + scSE Blocks**	
	#P	FLOPs	#P	FLOPs
U-Net	$8.82\times 10_6$	$234.13\times 10_9$	$8.93\times 10_6$	$234.22\times 10_9$
U-Net++	$10.85\times 10_6$	$750.04\times 10_9$	$10{,}97\times 10_6$	$750.17\times 10_9$

Table 3.1 A complexity comparison of the two main networks (U-Net and U-Net++) with and without added scSE blocks. The table highlights the number of free parameters (denoted #P) as well as the number of multiply-adds (FLOPS [29])

It is apparent that adding scSE component to convolutional blocks does not add a significant complexity to the neural network. However, while the difference between U-Net and U-Net++ is relatively small in terms of the number of parameters, the number of multiply-adds (FLOPS) is significantly higher (more than thrice as many operations). Therefore, adding squeeze and excitation components does not significantly add to the complexity of the network as much as changing the architecture would.

3D convolution is often much more computationally expensive than 2D convolution, however, it is undoubtedly better at capturing 3D context [30]. Given the architecture of U-Net++, the data needs to downscale by half in the x and y-axis in order to preserve memory.

3.9 Local Refinement Network RefNet

The refinement network (RefNet) is a U-Net architecture that is designed improve local segmentation accuracy. The inputs to this network is a 3D patch $P \in \mathbb{R}^{64\times64\times16\times10}$ of the normalized image Z and the segmented organ probability map $\hat{Y}$ from the initial segmentation network in the current patch. The output is a locally refined organ segmentation $\hat{Y}_P$ contained in this 3D patch.

The training data will be the pairs of 3D patches $(64 \times 64 \times 16)$ randomly sampled around the neighborhoods of the segmented organ contours by GlobalSegNet. 4000/400/400 of the training data patches, produced from the cohort of 100 training samples will be used for training/validation/testing.

To apply the RefNet (after GlobalSegNet) on a new image, patches will be selected in the same way but will be uniformly sampled with half overlapping in order to ensure smooth transition between adjacent patches. The final segmentation $\tilde{Y}$ is formed by reconstructing the patches. For a voxel with multiple overlapping patches, we take the mean value for that voxel across patches. The refinement process is visually displayed in Figure 3.13.

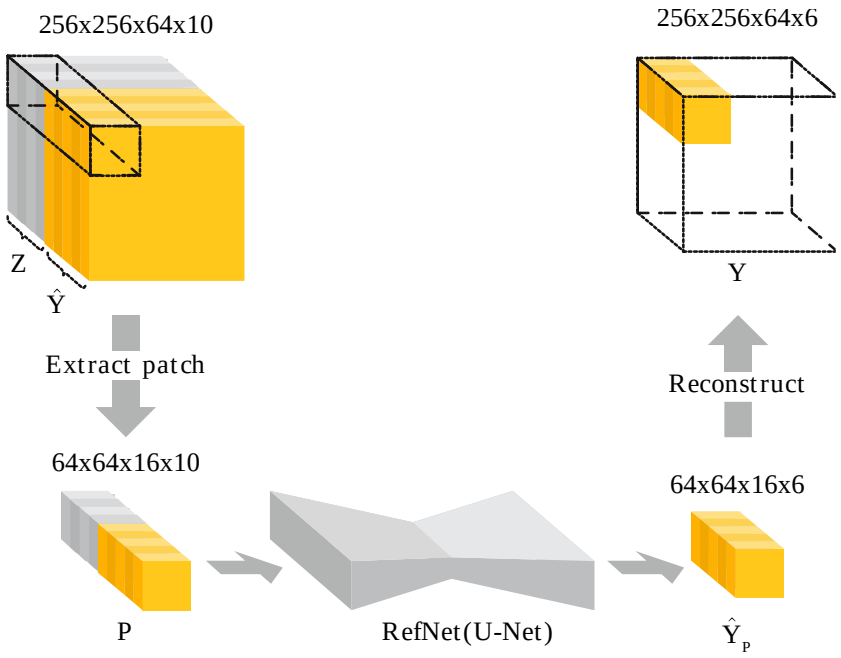


Figure 3.13 RefNet:

Comparing with the global auto-segmentation network GlobalSegNet, RefNet 1) focuses on a much smaller 3D region and 2) uses a voxel size = 1.5 mm3, halving the voxel size (≈ 3.5×3.5×3 mm3) in GlobalSegNet. Thus, we expect RefNet to improve the local segmentation accuracy.

3.10 Segmentation Pipelines

In this section we discuss a number of different pipelines for segmentation and how they compare to each other as well as to simply using end-to-end convolutional neural network architecture (U-Net and U-Net++).

Figure 3.14 shows the different pipelines implemented in this research; their performance is then compared in Experimental Results chapter.

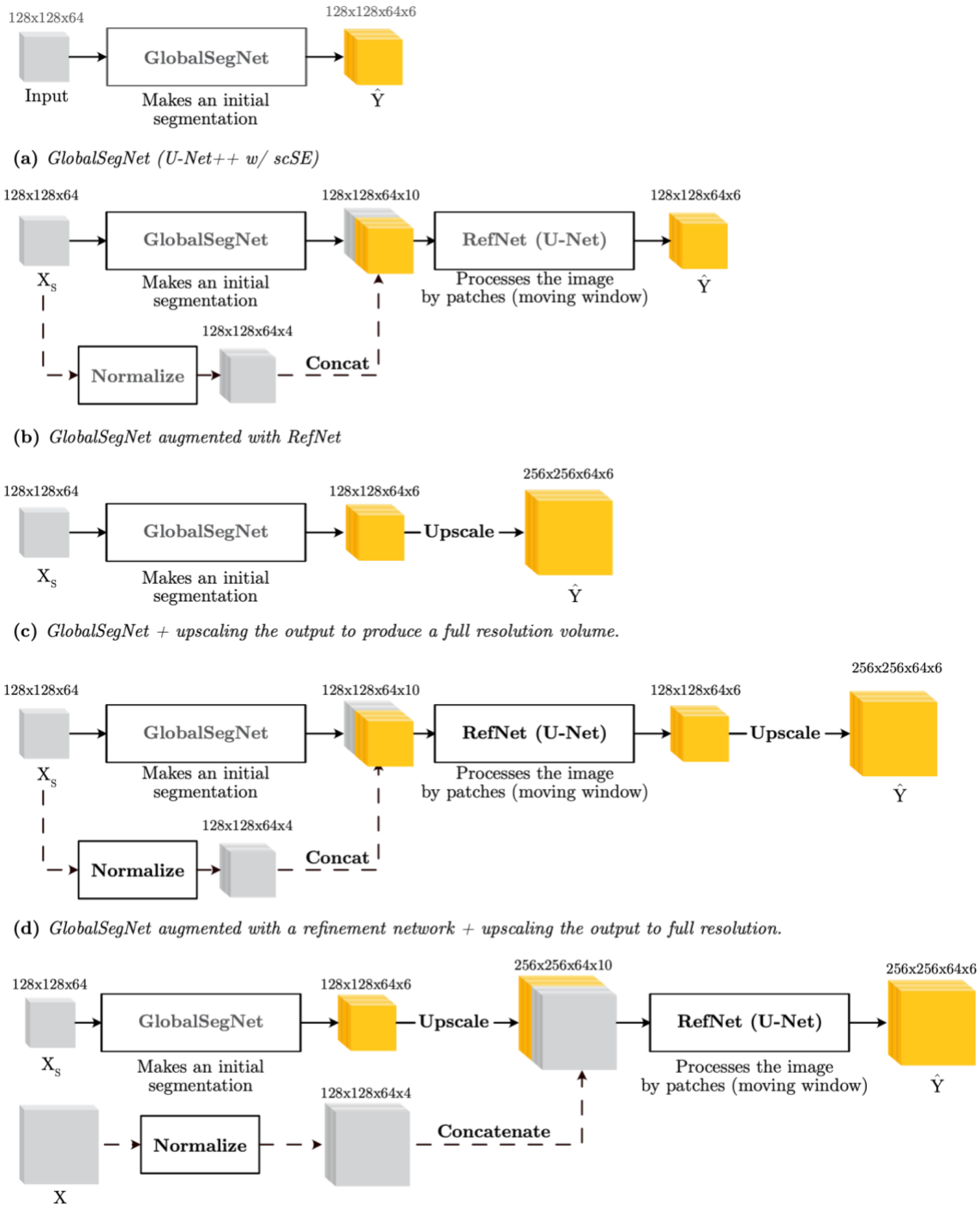


Figure 3.14 Different pipelines used for segmentation. Pipelines (a) and (b) produce a half resolution segmentation, whereas (c), (d), (e) produce full resolution output.

Pipelines (a) and (b) are applied to the half-resolution dataset in order to compare with the original work [9] on this dataset. In (a) we simply use the

GlobalSegNet model to make a half resolution segmentation. In pipeline (b), we attempt to refine the segmentation using RefNet, and produce a segmentation of the same size. For (c) and (d), we simply augment (a) and (b) by adding an up-sampling step on the output probability maps. The final pipeline (e) uses GlobalSegNet to make an initial segmentation on half resolution, which is then upscaled and concatenated with the full resolution spatially normalized data Z. The 10-channel tensor is then processed with RefNet to produce a final segmentation.

3.11 Implementation Configuration and Hardware

The networks discussed in this thesis are implemented using TensorFlow and Keras [31] in Python. The networks were trained by minimizing the loss function using Adam optimizer [32] with learning rate $\alpha = 1 \times 10^{-4}$. The trained parameters of the networks were initialized randomly using Xavier [33] uniform initializer. The training ran for a maximum of 1000 epochs, with an early stopping condition in the case where loss does not improve for 50 epochs.

The training took advantage of GPU acceleration. An NVIDIA TITAN X with 11 Gb of RAM and

Chapter 4: Experimental Results and Discussion

In order to evaluate the quality of the models and pipelines used for this segmentation task, we use the previously discussed Methods on the testing set composed of 10 volume images.

The empirical performance of the different methods is the DICE coefficient [34] as computed in the following equation:

$$\mathrm{DICE}\left(Y_i, \widehat{Y}_i\right) = 2 \times \frac{\left|Y_i \cap \widehat{Y}_i\right|}{|Y_i| + \left|\widehat{Y}_i\right|} \tag{14}$$

With Y_i being the ground truth 3D binary mask of organ i and $\widehat{Y}_i$ is the segmentation output mask of the same organ.

The mean DICE coefficient is then calculated as the unweighted average of DICE coefficient associated to each organ:

$$\mathrm{DICE}_{mean} = \underset{i \in \{organs\}}{\mathrm{mean}} \left(\mathrm{DICE}\left(Y_i, \widehat{Y}_i\right)\right) \tag{15}$$

Both the DICE score of the individual organs, and the average DICE score of all organs are summarized in Figure 4.1, Figure 4.4, and Figure 4.6.

In order to compare with the original work on this dataset [9], the evaluation was performed on the half-resolution dataset (of 128×128×64 voxels). Additionally, some methods took advantage of the full resolution dataset (of 256×256×64 voxels) produced a full resolution segmentation. Due to hardware limitation, we were not able to use the full resolution data to train U-Net++.

4.1 CNN Architectures

We first evaluate the individual models directly on the dataset (original full resolution as well as downsized half resolution)

4.1.1 On half-resolution

In this part, we discuss the performance of the U-Net family of architectures as well as the U-Net++ family on the down-sampled dataset ($X_S \in \mathbb{R}^{128\times128\times64}$).

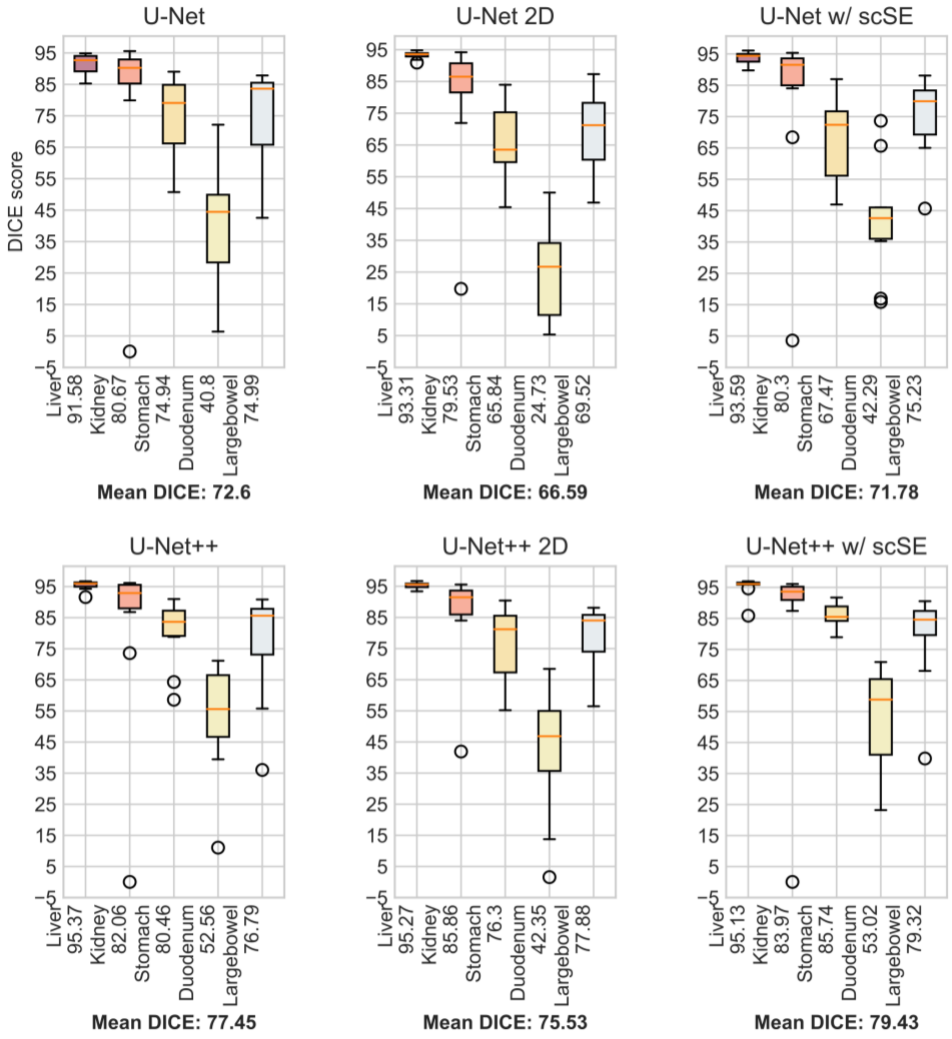


Figure 4.1 DICE Scores of different used CNN Architectures on half resolution data.

For the sake of comparison, a 2D variation of each network has been implemented. The 2D networks use $3 \times 3 \times 1$ kernels for all convolution operations. This kind of convolution cuts on the computational cost of the model; however, it comes with a slight sacrifice on the quality of segmentation as it does not take advantage of the depth context. This can be directly seen in

Figure 4.1. Another thing to note is that U-Net++ outperforms simple U-Net in general.

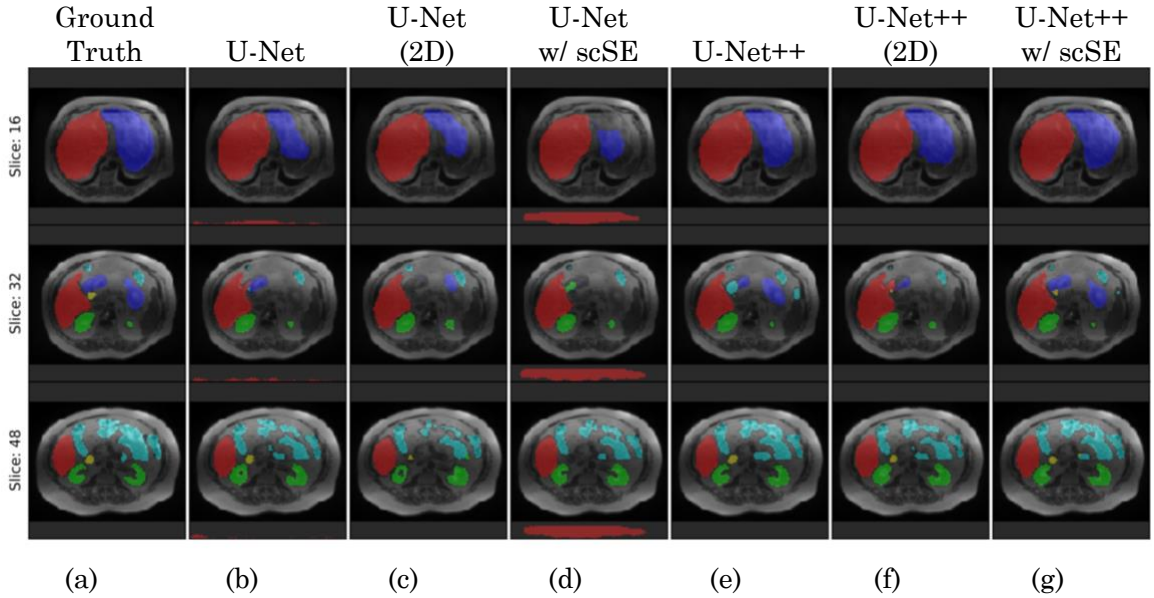


Figure 4.2 shows that the global quality of the segmentation is improved from U-Net to U-Net++; additionally, scSE blocks seem to improve the segmentation quality in a local level for U-Net++.

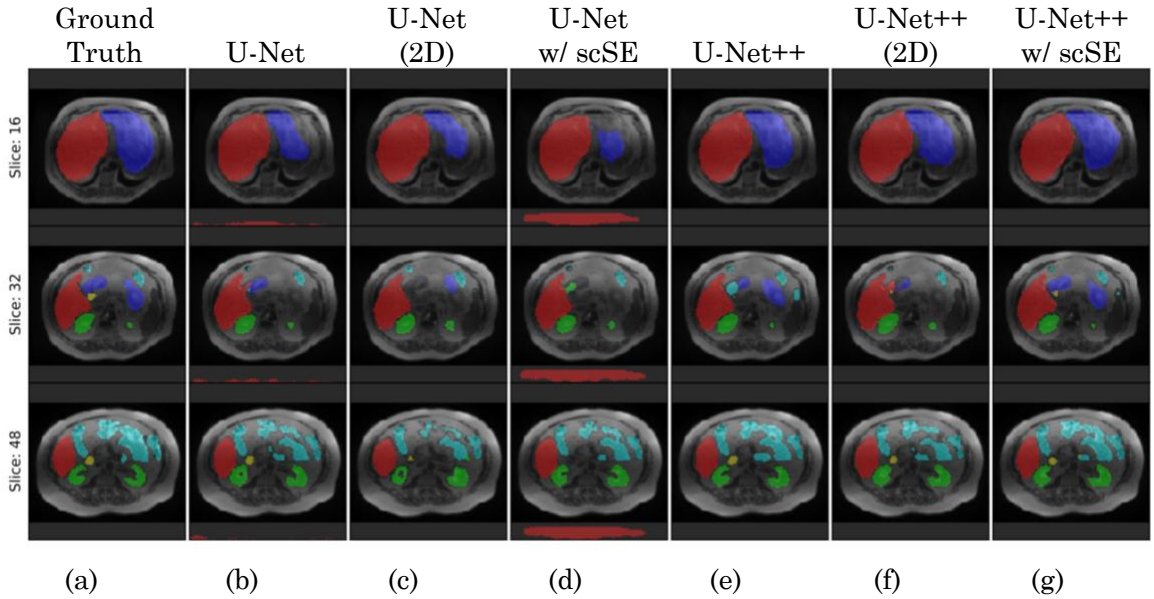


Figure 4.2 Top axial view of a sample output volume image from the testing set. Rows represent the slice used in the z-axis whereas the columns represent the different architectures attempted.

Some artifacts can be observed for the segmentations produced by U-Net architecture (both with and without the squeeze and excitation blocks). These artifacts represented in columns (b) and (d) in

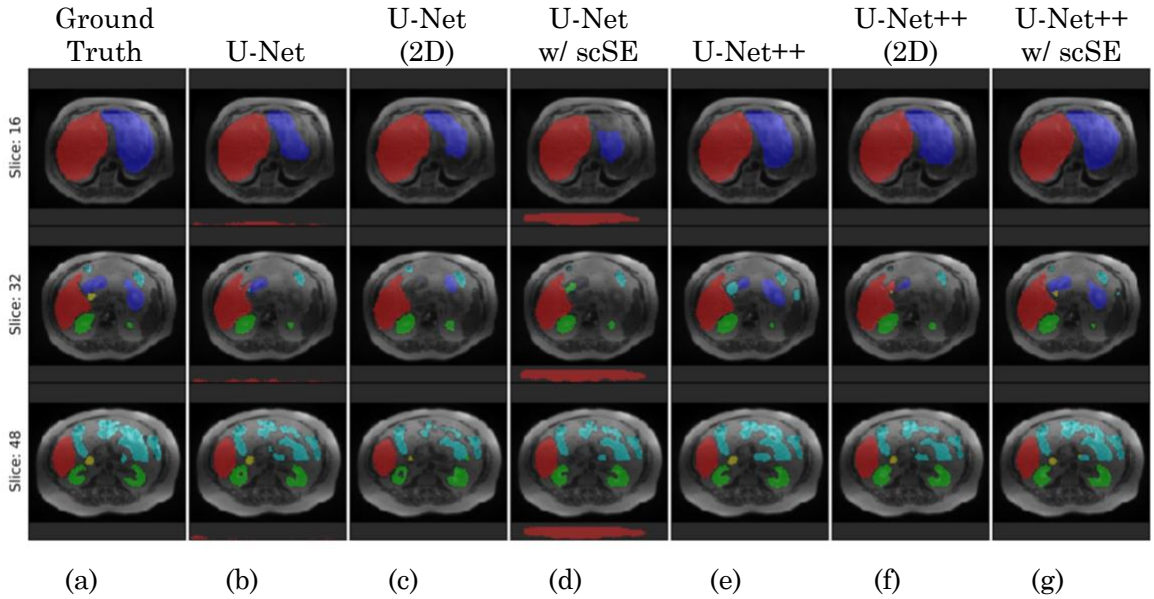
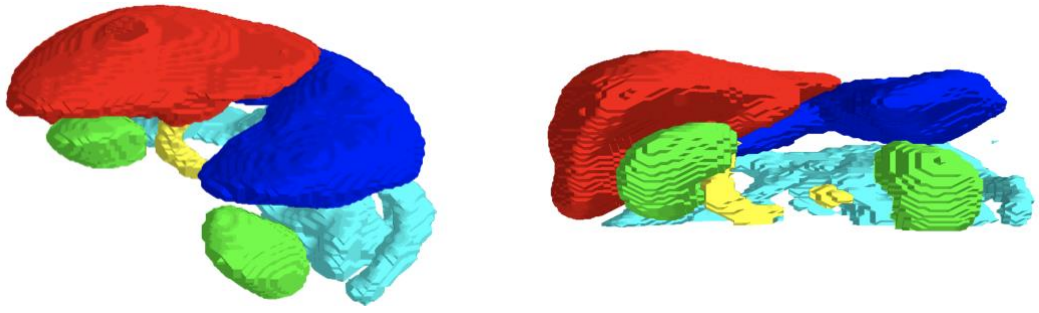


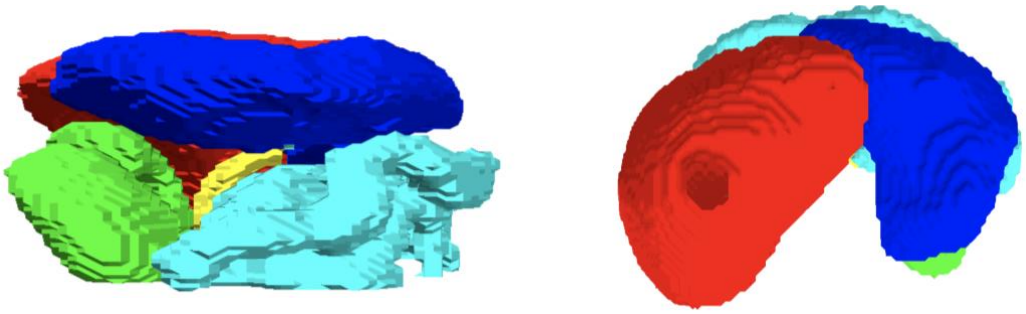
Figure 4.2 where it shows empty areas misclassified as Liver (red).

Artifacts appear on a smaller scale for other U-Net / U-Net++ architectures. This is one of the reasons why we introduced RefNet and the different pipelines (which are discussed in "Pipelines"). The aim is to capture those artifacts and eliminate them to finally produce a cleaner segmentation.

In order to visualize the 3D structure of the output segmentations. We have implemented a Vis3D, a tool built with the python library Mayavi [35].



(a) *(b)*



(c) *(d)*

Figure 4.3 3D view of the segmentation output of GlobalSegNet (using the developed tool Vol3D).

4.1.2 On full resolution

In this part, we discuss the performance of the U-Net architectures on the full resolution dataset ($X \in \mathbb{R}^{256\times256\times64}$).

Similarly to applying these networks on the down-sampled images, we can see from Figure 4.5.d that the U-Net architecture augmented with squeeze and excitation blocks (scSE) introduces artifacts to the segmentation. However, the quality of segmentation appears to be superior than U-Net and U-Net 2D. This time, the empty area on the bottom of the axial view is misclassified as "large bowel" this time.

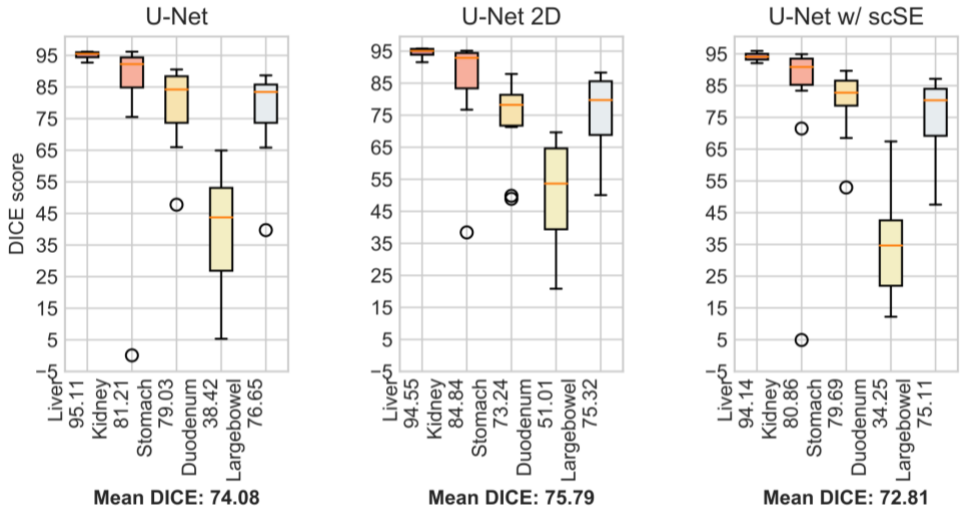


Figure 4.4 DICE Scores of different used CNN Architectures on full resolution data.

Even though the segmentation produced by U-Net w/ scSE appears the closest to the ground truth (according to Figure 4.5), the network's overall dice score is the lowest of the U-Net family when trained on full resolution. This is due to the previously mentioned artifacts. Artifacts appear on

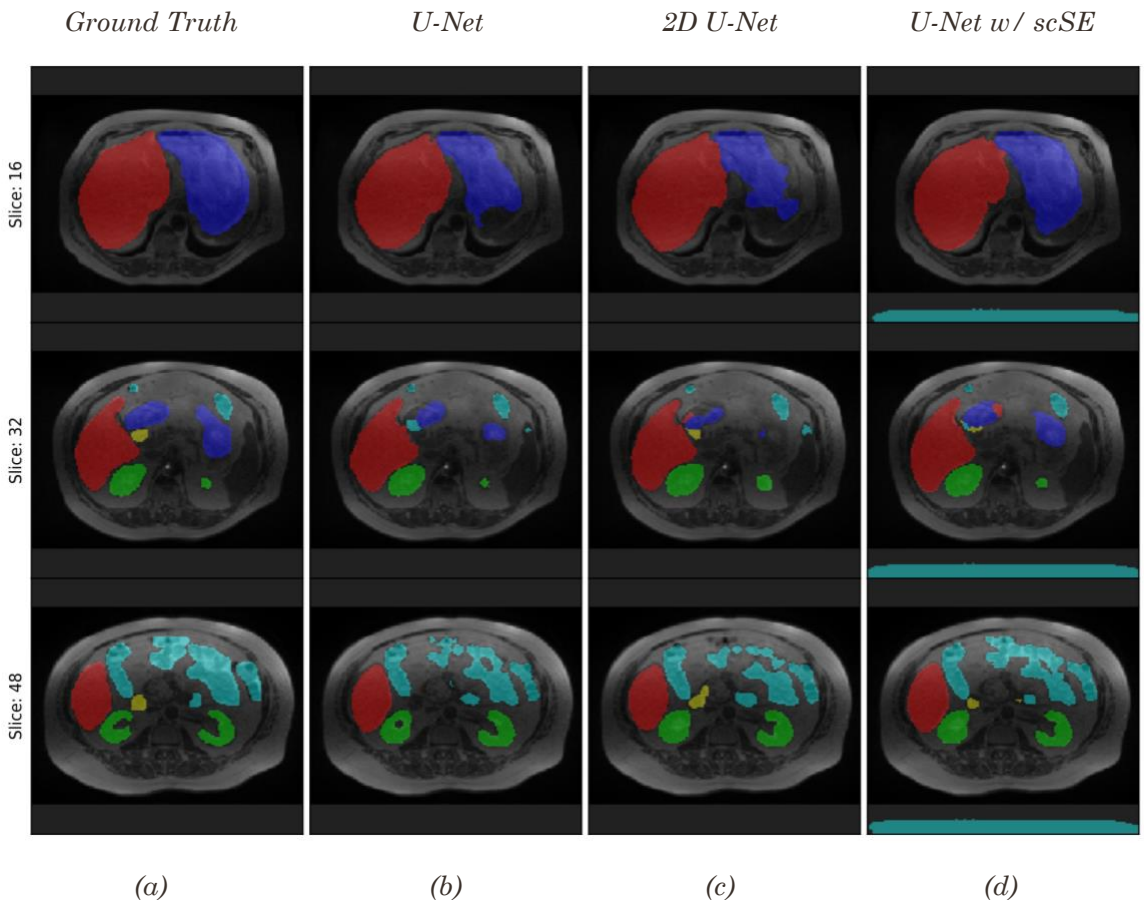


Figure 4.5 Top axial view of a sample output volume image from the testing set.

4.2 Pipelines

To evaluate the performance of the different used pipelines, we first evaluate the dice score of the produced segmentations from each pipeline. DICE scores are summarized in Figure 4.6. (a), and (b) are pipelines that produce half resolution segmentation; whereas (d), (e), and (f) produce full resolution segmentation.

In general, it is easier to achieve a better average DICE score in low resolution. This is true for end-to-end networks, but also for the pipelines used.

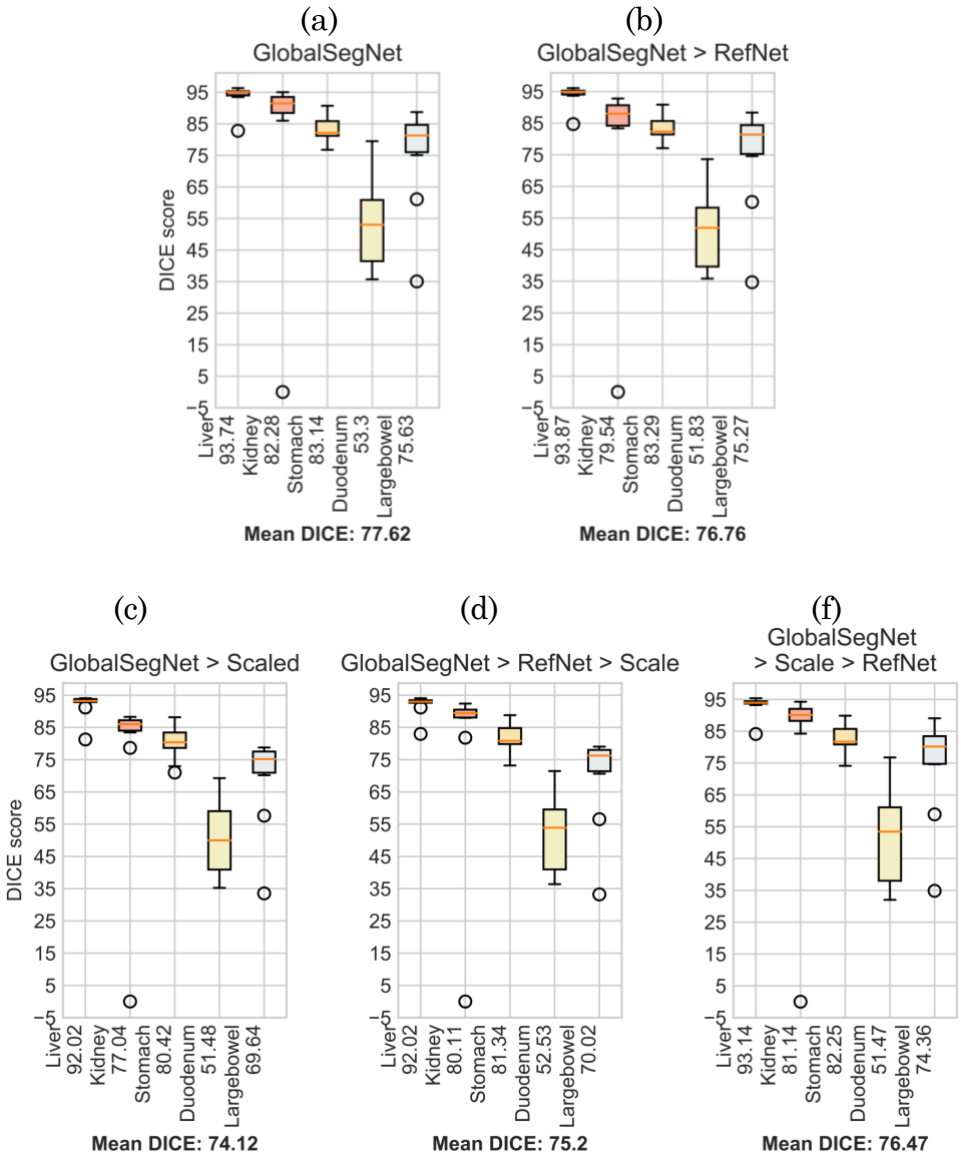


Figure 4.6 DICE Scores of different used pipelines

Figure 4.6 shows that up-scaling the low-resolution output to produce a full resolution segmentation leads to a lower DICE score. This is due to the fact that up-scaling causes a blocky low-detail segmentation (as shown in Figure 4.8.

Additionally, it is noted that applying refinement network before up-scaling reduces the DICE score; however, it generates a more robust segmentation. This can be explained by (c) and (d) in Figure 4.6; up-scaling the output of

GlobalSegNet leads to a lower DICE score than up-scaling GlobalSegNet+RefNet output.

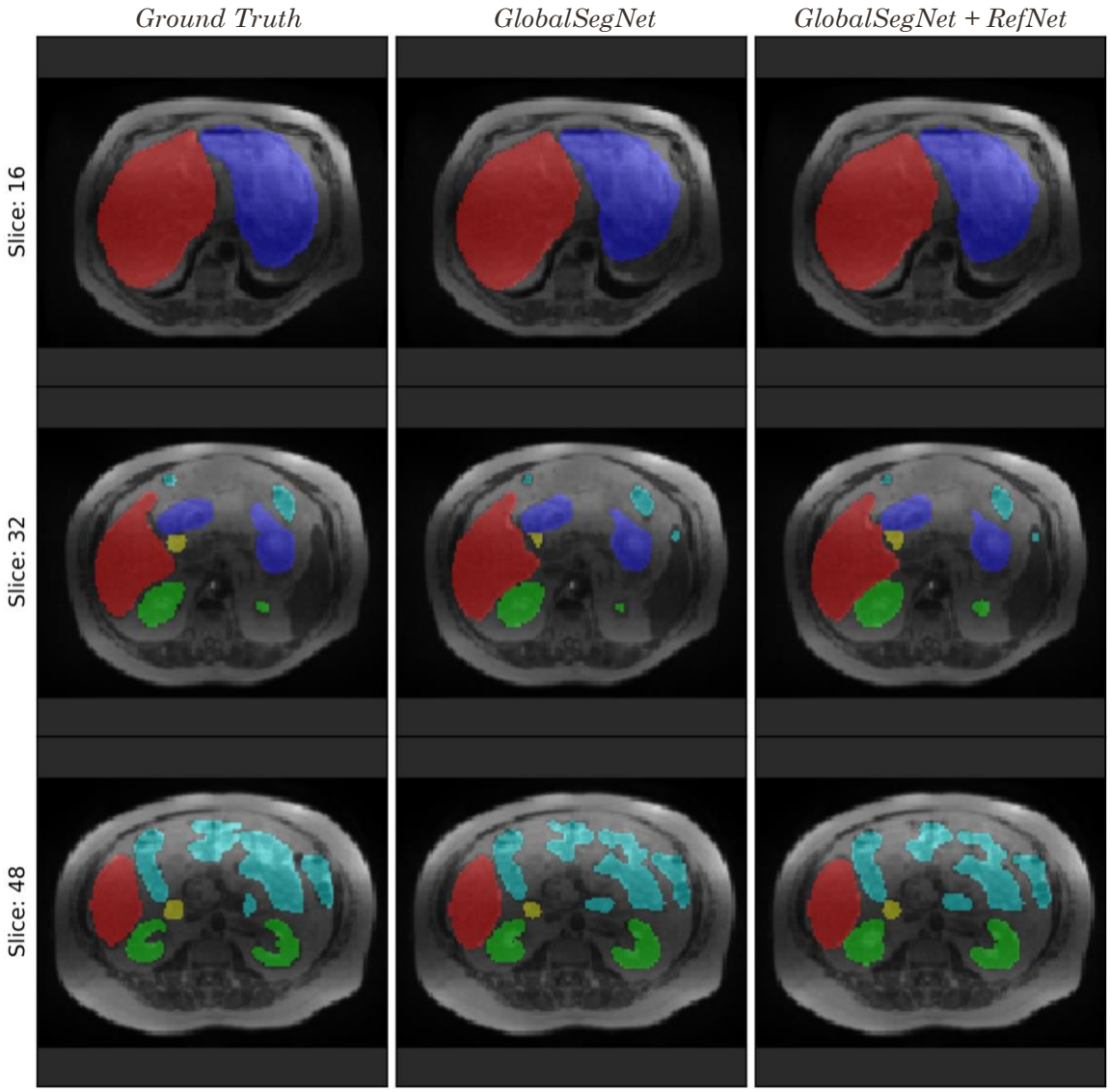


Figure 4.7 A qualitative comparison of the different pipelines with a half resolution output.

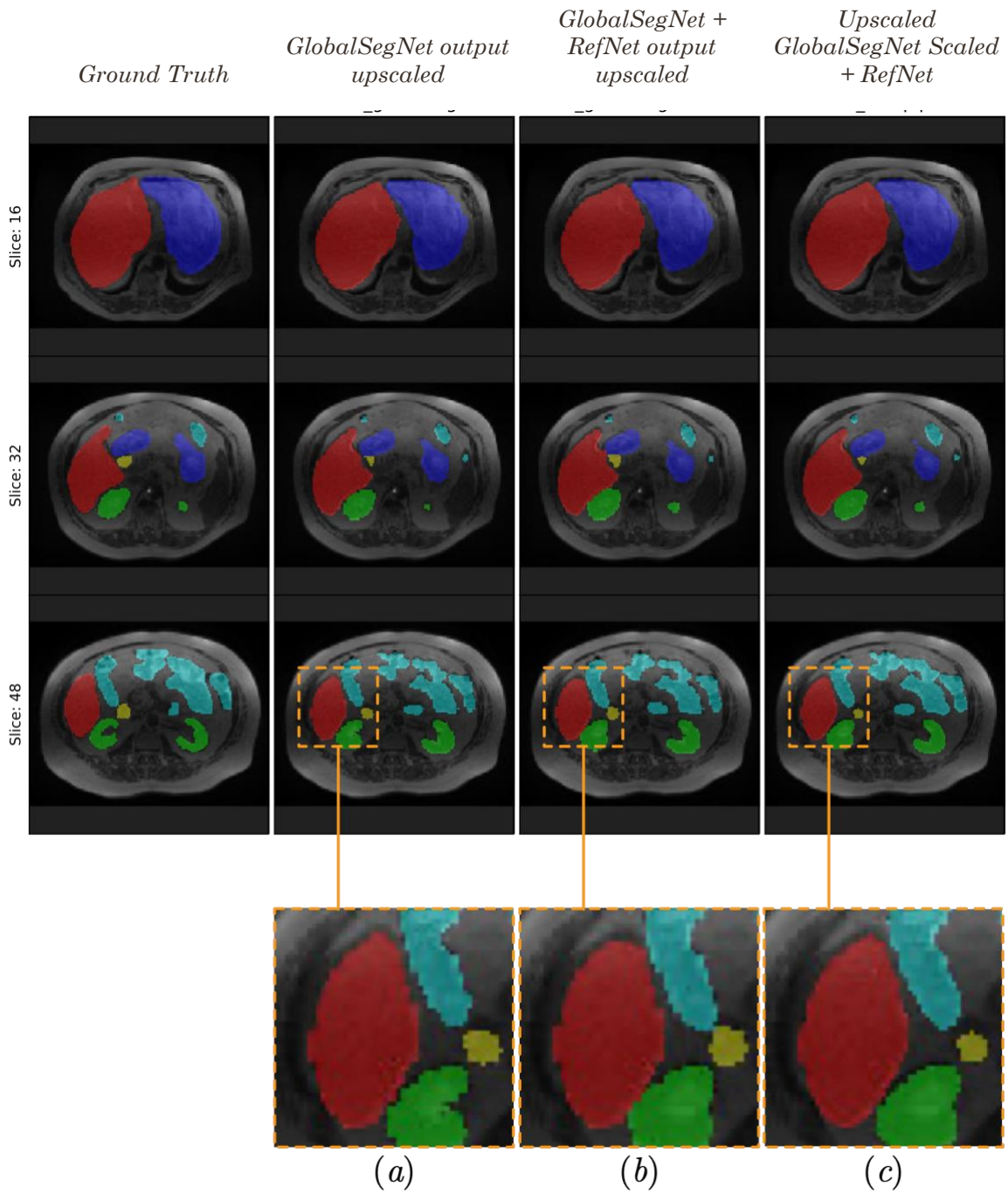
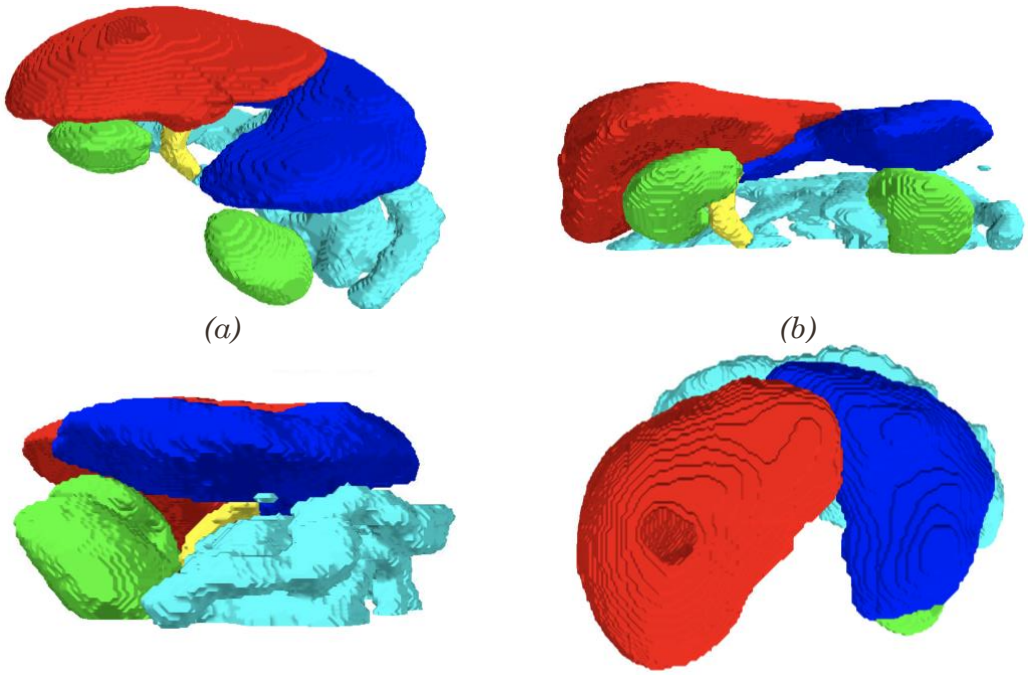


Figure 4.8 A comparison of the different pipelines with a full resolution output. (a), (b) and (c) shows a zoomed-in patch for clarity on the difference in detail.

From Figure 4.8, it is noticed that the segmentation made by the final pipeline provides the highest resolution; wheras up-scaling a low resolution output causes a blocky and low-detailed segmentation.



(a) *(b)* *(c)* *(d)*

Figure 4.9 3D view of the segmentation of the final pipeline. (a) shows a 3D view, (b) shows a front view, (c) shows a side view, and (d) shows a top view.

4.3 Results Summary

To summarize our experimental results, we report the accuracy of each method as well as its execution time as our computational complexity efficiency measure. In order to compare our results with the state-of-the-art methods, we compare our work to CNN+Correction Network explained in Related Work.

In order to evaluate CNN+Correction Network, the model has been retrained and tested without any added post-processing steps in order to compare the deep learning networks. The results reported in the original work are summarized in

Organ	**Liver**	**Kidneys**	**Stomach**	**Duodenum**	**Bowel**	**Mean**
DICE Score	95.3	93.1	85.0	86.6	65.5	85.1

Table 4.1 DICE score table for CNN+Correction Network as reported in the original paper [9].

4.3.1 Accuracy

The three first rows of Table 3.1 show the mean DICE scores associated with the output of the testing data. We report the DICE scores for individual organs as well as a mean score across organs.

	DICE Scores					
	Liver	Kidneys	Stomach	Duodenum	Bowel	Mean
CNN+Correction Network	93.9	79.1	82.4	52.8	75.5	76.74
GlobalSegNet	93.74	**82.28**	83.14	**53.30**	**75.63**	**77.63**
GlobalSegNet + RefNet	**93.87**	79.54	**83.29**	51.83	75.27	76.76
GlobalSegNet + Upscale	92.02	77.04	80.42	51.48	69.64	74.12
GlobalSegNet + RefNet + Upscale	92.02	80.11	81.34	**52.53**	70.02	75.20
GlobalSegNet + Upscale + RefNet	**93.14**	**81.14**	**82.25**	51.47	**74.36**	**76.47**

Table 4.2 Summary table of the accuracy of the different used methods

4.3.2 Computational Complexity

In order to evaluate our methods from a computational complexity perspective, we measure the inference time of each method. The inference time is computed through sequentially inputting 100 images and measuring the mean inference time. Similar to the accuracy table, the first three rows of Table 4.3 show the inference time on half resolution; whereas the last four rows on full resolution.

Network	**Inference time (seconds)**
CNN+Correction Network	6.333
GlobalSegNet	**0.466**
GlobalSegNet + RefNet	1.888
CNN+Correction Network + Up-scale	6.545
GlobalSegNet + Up-scale	**0.517**
GlobalSegNet + RefNet + Up-scale	2.077
GlobalSegNet + Upscale + RefNet	9.118

Table 4.3 Summary table of the inference time of the different used methods

The most efficient methods for half resolution output (top three rows) as well as full resolution output (bottom four rows) are highlighted in bold.

Conclusion

In this work, we explored different segmentation deep learning networks and methods to segment liver, kidneys, stomach, bowel and duodenum on MRI volumes. We have leveraged recent deep learning methods like deep supervision and squeeze and excitation mechanisms into the fully convolutional network; GlobalSegNet. Deep-learning-based segmentation pipelines have also been implemented through introducing up-scaling, spatial normalization components, as well as patch-based local refinement step (RefNet). We have evaluated our methods and pipelines against the state-of-the-art work on the MRI dataset. We have achieved comparable results to the original work while significantly improving the inference time and work around GPU memory limitation in order to create full resolution segmentations.

Future Work

While it is challenging to improve segmentation quality on this dataset, there is always room for improvement in terms of performance.

"2.5D" Convolution to improve computational cost: Due to current hardware limitations, 3D convolution is significantly more expensive than 2D convolution. However, using 2D convolution for a 3D dataset makes the network ignore important depth information. A workaround worth exploring is to use 2.5D convolution [36], in order to take full advantage of 3D geometry relations whilst still keeping computational cost low.

Semi-automated workflows: Including deep-learning-powered semi-automatic procedures that use user interactive contour correction will improve the accuracy of segmentation while adding an insignificant amount of work to physicians.

References

[1] B. Ginneken, C. Schaefer-Prokop and M. Prokop, "Computer-aided diagnosis: how to move from the laboratory to the clinic.," *PubMed,* vol. 216 (3), pp. 719-732, 2011.

[2] L. Henke, R. Kashani, C. Robinson, A. Curcuru, T. DeWees, J. Bradley, O. Green, J. Michalski, S. Mutic, P. Parikh and O. J3., "Phase I trial of stereotactic MR-guided online adaptive radiation therapy (SMART) for the treatment of oligometastatic or unresectable primary malignancies of the abdomen.," *Radiotherapy and oncology,* pp. 519-526, 2017.

[3] S. Rudra, N. Jiang, S. Rosenberg, J. Olsen, P. Parikh, M. Bassetti and P. Lee, "High Dose Adaptive MRI Guided Radiation Therapy Improves Overall Survival of Inoperable Pancreatic Cancer," *International Journal of Radiation Oncology • Biology • Physics,* vol. 99, no. 2, 2017.

[4] J. M. Lamb, M. Cao, A. U. Kishan, N. Agazaryan, D. H. Thomas, N. Shaverdian, Y. Yang, S. Ray, D. A. Low, A. C. Raldow, M. L. Steinberg and P. P. Lee, "Online Adaptive Radiation Therapy: Implementation of a New Process of Care," *Cureus,* vol. 9, 2017.

[5] C. Hao, Q. Dou, L. Yu, J. Qin and P.-A. Heng., "VoxResNet: Deep voxelwise residual networks for brain segmentation from 3D MR images," *NeuroImage,* pp. 446-455, 2018.

[6] P. Hu, F. Wu, J. Peng, P. Liang and D. Kong, "Automatic 3D liver segmentation based on deep learning and globally optimized surface evolution.," *Physics in medicine and biology,* vol. 61, no. 24, 2016.

[7] P. Moeskops, M. A. Viergever, A. M. Mendrik, L. S. de Vries, M. J. N. L. Benders and I. Isgum, "Automatic Segmentation of MR Brain Images With a Convolutional Neural Network," *IEEE Transactions on Medical Imaging,* vol. 35, no. 5, p. 1252–1261, 2016.

[8] P. F. Christ, M. E. A. Elshaer, F. Ettlinger, S. Tatavarty, M. Bickel, P. Bilic, M. Rempfler, M. Armbruster, F. Hofmann, M. D'Anastasi and e. al, "Automatic Liver and Lesion Segmentation in CT Using Cascaded Fully Convolutional Neural Networks and 3D Conditional Random Fields," *Lecture Notes in Computer Science,* p. 415–423, 2016.

[9] Y. Fu, T. R. Mazur and X. Wu, "A novel MRI segmentation method using CNN-based correction network for MRI-guided adaptive radiotherapy," *Medical Physics,* pp. 5129-5137, 2018.

[10] Y. Guo and A. S. Ashour, "Neutrosophic Set in Medical Image Analysis," in *Handbook of Medical Imaging*, 2019, pp. 229-243.

[11] T. Heimann and H.-P. Meinzer, "Statistical shape models for 3D medical image segmentation: A review," *Medical Image Analysis,* vol. 13, no. 4, pp. 543-563, 2009.

[12] T. Zhou, S. Ruan and Stéphane Canu, "A review: Deep learning for medical image segmentation using multi-modality fusion," *Array,* Vols. 3-4, 2019.

[13] J. Long, E. Shelhamer and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE Conference on Computer Vision and Pattern Recognition,* p. 3431–3440, 2015.

[14] O. Ronneberger, P. Fischer and T. Brox, "U-net: Convolutional networks for biomedical image segmentation".

[15] Z. Zhou, M. M. R. Siddiquee, N. Tajbakhsh and J. Liang, "Unet++: A nested u-net architecture for medical image segmentation.," *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support,* pp. 3-11, 2018.

[16] D. Bahdana, K. Ch and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," *arXive,* vol. 7, 2014.

[17] P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould and L. Zhang, "Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering," *CVPR,* vol. 3, 2017.

[18] S. Jetley, N. A. Lord, N. Lee and P. H. S. Torr, "Learn To Pay Attention," *International Conference on Learning Representations 2018,* vol. 2, 2018.

[19] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang and X. Tang, "Residual Attention Network for Image Classification," *CVPR 2017,* vol. 1, 2017.

[20] O. Oktay, J. Schlemper, L. L. Folgoc, M. Lee, M. Heinrich, K. Misawa, K. Mori, S. McDonagh, N. Y. Hammerla, B. Kainz, B. Glocker and D. Rueckert, "Attention U-Net: Learning Where to Look for the Pancreas," *MIDL,* vol. 3, 2017.

[21] Z. Zhong, Z. Q. Lin, R. Bidart, X. Hu, I. B. Daya, Z. Li, W.-S. Zheng, J. Li and A. Wong, "Squeeze-and-Attention Networks for Semantic Segmentation," *arXive,* 2019.

[22] J. Hu, L. Shen, S. Albanie, G. Sun and E. Wu, "Squeeze-and-Excitation Networks," *Computer Vision and Pattern Recognition,* vol. 4, 2017.

[23] A. G. Roy, N. Navab and C. Wachinger, "Concurrent Spatial and Channel Squeeze & Excitation in Fully Convolutional Networks," *Computer Vision and Pattern Recognition,* vol. 2, 2018.

[24] G. Huang, Z. Liu, L. v. d. Maaten and K. Q. Weinberger, "Densely Connected Convolutional Networks," *Computer Vision and Pattern Recognition,* vol. 5, 2016.

[25] J.-F. Mangin, J. Lebenberg, S. Lefranc, N. Labra, G. Auzias, M. Labit, M. Guevara, H. Mohlberg, P. Roca, P. Guevara, J. Dubois, F. Leroy, G. Dehaene-Lambertz, A. Cachia, T. Dickscheid, O. Coulon, C. Poupon, D. Rivière, K. Amunts and Z. Sun, "Spatial normalization of brain images and beyond," *Medical Image Analysis,* vol. 33, 2016.

[26] A. Odena, V. Dumoulin and C. Olah, "Deconvolution and Checkerboard Artifacts," Distill, 17 October 2016. [Online]. Available: http://distill.pub/2016/deconv-checkerboard. [Accessed 23 3 2020].

[27] V. Nair and G. E. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines," *International Conference on Machine Learning (ICML-10),* pp. 807-814, 2010.

[28] C. M. Bishop, Pattern Recognition and Machine Learning.

[29] X. Zhang, X. Zhou, M. Lin and J. Sun, "ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices," *arXive,* vol. 2, 2017.

[30] J. Yang, X. Huang, B. Ni, J. Xu, C. Yang and G. Xu, "Reinventing 2D Convolutions for 3D Images," *arXiv,* vol. 2, 2019.

[31] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A.

Harp, G. Irving, M. Isard, R. Jozefowicz, Y. Jia and L, *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems,* 2015.

[32] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv,* 2-14.

[33] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," *International Conference on Artificial Intelligence and Statistics,* vol. 9, pp. 249-256, 2010.

[34] C. E. Noel, F. Zhu, Lee, r. Y, H. Yanle and P. J. Parikh, "Segmentation precision of abdominal anatomy for MRI-based radiotherapy," *Medical Dosimetry,* vol. 39, pp. 212-217, 2014.

[35] P. Ramachandran and G. Varoquaux, "Mayavi: 3D Visualization of Scientific Data," *IEEE Computing in Science & Engineering,* vol. 2, no. 13, pp. 40-51, 2011.

[36] Y. Xing, J. Wang, X. Chen and G. Zeng, "2.5D Convolution for RGB-D Semantic Segmentation," *IEEE International Conference on Image Processing (ICIP),* pp. 1410-1414, 2019.