Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Iterative Semi-Automatic Segmentation using Volumetric Scribbles with a 3D CNN

Master's Thesis

Hlynur Jökull Skúlason

Advisors: Gustav Bredell & Dr. Christine Tanner
Supervisor: Prof. Dr. Ender Konukoglu
Computer Vision Laboratory
Department of Information Technology and Electrical Engineering, ETH Zürich

October 3, 2019

Abstract

Medical image segmentation is frequently performed with 2D networks that segment slice-wise. To utilize the spatial information within a volumetric image, we propose a framework that segments prostate magnetic resonance images of multiple classes in 3D, both automatically and semi-automatically. We implement a robot-user that marks discrepancies between the automatic segmentation and the ground-truth and creates scribbles in 2D or 3D. The semi-automatic model learns from the automatic segmentation network and uses 3D scribbles to increase the prediction accuracy by 16%, from 79.14% to 94.97%. We obtain a method that produces more accurate results than our 2D predecessor for up to 385 user interactions, which is a more realistic scenario for a human user.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Image segmentation is one of the biggest and most challenging tasks in medical image analysis to aid human experts in detecting and diagnosing diseases in the human body [1, 2]. Developing precise and reliable algorithms is critical and advancements in machine learning and computer vision have further increased these algorithms' accuracy and dependability.

Image segmentation can be classified into three types: Manual, semi-automatic and fully-automatic. Manual segmentation is a tedious task that requires a human to manually label every pixel. It yields the most accurate results but is time-consuming to implement into clinical practice. Fully-automatic segmentation requires no user interaction and can be effective, especially using machine learning, but has flaws such as overfitting or failing to generalize over multiple datasets. Semi-automatic, or interactive, segmentation allows a user to interact with a model's prediction to correct inaccuracies and improve the segmentation [3]. The user will see the current predicted segmentation and can select wrongfully classified pixels, which is fed back to the model to adjust its prediction. Interactive segmentation models can then be further improved with an iterative process. In each iteration, the user adds corrections to the segmentation to refine the model's prediction. By iterating over the same image multiple times, the model will achieve higher accuracy scores. The drawback of this method is the requirement of additional user input. Finding the balance between the amount of user input required and segmentation accuracy is imperative for employing such algorithms to clinical use.

One of the more common medical imaging modalities is magnetic resonance imaging (MRI), due to its versatility and effectiveness. MRI captures 2D slices of the subject and stores them in a volumetric image. Automatic segmentation algorithms are popular for both 2D and 3D images, working either on individual slices or segmenting the whole volume, and convolutional neural networks (CNNs) have become the most prevalent network architectures by achieving state-of-the-art performance on medical images [4, 5]. 2D networks are generally faster to segment and more cost-effective concerning computing power, while 3D networks can utilize the spatial information present in the volume to create more accurate segmentation masks. However, they are notorious for their high computational cost, and a limited amount of GPU memory remains a challenge when training 3D networks.

1.2 Focus of the Thesis

The focus of this thesis is to:

- Create a semi-automatic segmentation framework consisting of models based on the U-Net architecture.
- Segment 3D medical images in a multiclass setting, first automatically and then iteratively with inputs from a user in the form of clicks.
- Generate 3D Gaussian scribbles from the clicks, created by a robot-user, to mark inaccuracies in the predicted segmentation mask.
- Improve the prediction iteratively for further refinement.
- Explore different scribble configurations, such as creating larger scribbles or placing scribbles in the largest incorrect predictions.
- Predict more accurate segmentation masks during test time than previous methods with fewer interactions from the user.

1.3 Outline of the Thesis

The following lists each chapter and briefly describes their content:

Chapter 2: Related Work – Explores related work in the fields of image segmentation and interactive image segmentation with a focus on methods that utilize medical images as their data.

Chapter 3: Materials and Methods – This chapter describes the two datasets, introduces the architectures of our models, explains some limitations of 3D semantic segmentation, presents the evaluation metrics, the scribble generation process, and details the training procedure.

Chapter 4: Experiments and Results – In this chapter, we present how our experiments were conducted and their results with the two datasets.

Chapter 5: Discussion – This chapter discusses the results and proposes how our work can be expanded upon.

Chapter 6: Conclusion – Concluding the thesis, summarizing the key points of the framework.

Chapter 2

Related Work

In this chapter, we present literature related to the work of semantic segmentation. We first look at commonly used classification and segmentation networks and then examine segmentation networks with interactive components.

2.1 Image Segmentation Methods

Medical image segmentation has benefited greatly by the recent advancements in machine learning. Convolutional neural networks have achieved performance on par with human annotators and some have even surpassed human-level performance [6]. Medical images, unfortunately, lack consistency as they can vary between different scanners and acquisition techniques [4]. The need for robust segmentation algorithms is, therefore, essential for clinical use.

The U-Net network from 2015, built on the FCN architecture, has become a well-established model in the medical image segmentation field [7]. It is a fully-automatic segmentation model that showed impressive performance on 2D images. It was expanded to 3D a year later and is the model from which we build on [8]. Both architectures are explained further in Section 3.3.1.

The V-Net is an fully-convolutional architecture inspired by the U-Net. It uses de-convolutional layers, introduces a loss function based on the Dice score and integrates residual functions to the convolutional stages, ensuring faster convergence [9].

The Deep Contour-Aware Network (DCAN) is a fully-convolutional network that won the 2015 MICCAI Gland Segmentation Challenge by integrating multi-level contextual features to accurately segment glands from histology images. It creates accurate segmentation masks by combining the predicted segmentation with the predicted contours of the glands [10].

These fully-automatic segmentation networks, however, are not as reliable as semi-automatic networks, since they lack the supervision from a user. Semi-automatic networks can adapt more effortlessly when faced with new and unseen data, which is a common event in clinical practise [4].

2.2 Interactive Image Segmentation Methods

Interactive segmentation algorithms is an ever-growing research field with early algorithms such as GrabCut, Ilastik and Random Walker [11–13]. However, these long-established methods often require a considerable amount of user interaction as one user input improves the segmentation in one area of the image but might negatively affect the segmentation in other areas. Interactive segmentation algorithms utilizing deep learning have since become the prevalent approach with popular models such as DeepIGeoS, BIFSeg and UI-Net. DeepIGeoS and BIFSeg can segment either 2D or 3D images, while UI-Net only supports 2D segmentation.

The DeepIGeoS framework first automatically segments the input image. The user can then decide if the image has been correctly segmented. If not, the user clicks or draws lines on the image where the segmentation was incorrect and that is fed to another network along with the original segmentation to refine. The downside of the framework is that the segmentations cannot be edited iteratively, as the models are trained on batches of scribbles per iteration [14].

With BIFSeg, the user draws a bounding box around the region of interest and the model then segments the image. The user can then add foreground and background scribbles to improve the predicted segmentation. The model, however, needs a bounding box with each input image at the start of the training procedure to specify where the relevant information in the image is, instead of allowing the model to learn by itself. That requires more user input, as the model cannot start the training process without user input [15].

The UI-Net model is based on the U-Net architecture [7] and uses a robot-user to create scribbles (clicks, lines or shapes) that identify incorrect predictions from the U-Net model. Again, this framework relies on user input at every stage as there is not an initial prediction from a segmentation created automatically [16].

These aforementioned frameworks, however, all focus on binary segmentation, while medical images are often multiclass segmentation tasks [17]. Multiclass segmentation is a laborious task to approach as some classes may be undersampled in the ground truth, making it difficult for a network to successfully identify a few pixels of a specific class.

Our project is based on the work by Bredell et al. [18] They developed the framework that includes the AutoCNN and InterCNN architectures which we aim to expand to 3D, both in terms of the network and the user interaction method. The concept of their framework is as follows: a base segmentation algorithm (autoCNN) automatically segments a 2D image to get an initial prediction. The initial prediction is fed to a robot-user along with the ground-truth. The robot-user creates scribbles based on the difference between the predictions and the ground-truth. The scribbles are fed to the interactive network (interCNN) with the image and prediction. This process is iterated over K interactions to train interCNN.

Chapter 3

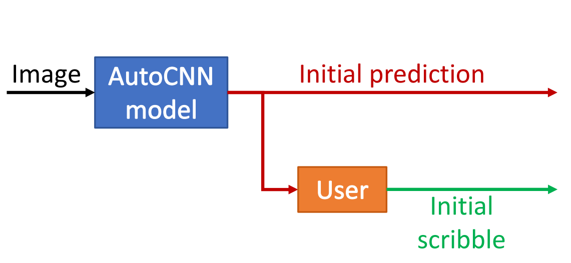
Materials and Methods

In this chapter, we introduce the dataset, framework, scribbles, network architectures, evaluation metrics and training procedure that we use in this project. Section 3.1 describes the structure of framework, Section 3.2 explains how the scribbles are generated, Section 3.3 introduces the U-Net model and our implementation, Section 3.4 details the evaluation metrics chosen to assess our model's performance, Section 3.5 discusses the dataset that is used as input to our network and Section 3.6 presents the training procedure of our modified network architectures.

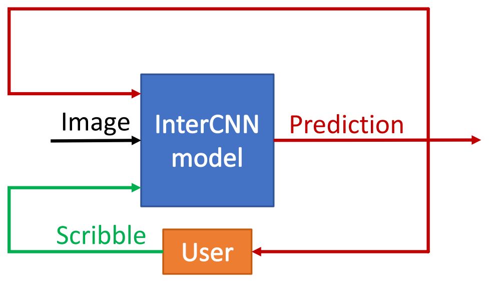
3.1 Framework

The framework we have created consists of two models that are based on the U-Net architecture but modified to fit our dataset and improve stability. Section 3.3.1 describes the original U-Net architecture and Section 3.3.2 details the modified architecture.

The first model, referred to as **AutoCNN**, is a fully-automatic segmentation algorithm that is used to generate the initial predictions from the input images. These predictions are used together with the ground-truth as input to a robot-user that compares them to the ground-truth and creates initial scribbles which are placed in the incorrect predictions. The second model, referred to as **InterCNN**, is fed three different inputs; the original image, the prediction from the AutoCNN model and the scribbles from the robot-user. Its task is to improve the AutoCNN prediction in an iterative process. In the first iteration, the robot-user creates a scribble for each class in the ground-truth based on the AutoCNN prediction and in the following iterations, the robot-user uses the InterCNN prediction to create the scribbles. The newer scribbles are then added to the ones from previous iterations, thus creating a scribble memory. Figure 3.1 illustrates the framework for the AutoCNN and InterCNN procedures.



(a) The AutoCNN procedure.



(b) The InterCNN procedure.

Figure 3.1: The framework of our automatic and semi-automatic segmentation models, illustrated by Bredell et al. (a) The AutoCNN model creates the initial prediction which the user sees and clicks on the incorrectly predicted pixels which are converted into scribbles. (b) The image, initial AutoCNN prediction and initial AutoCNN scribbles are fed into the InterCNN model, which creates an updated prediction. The user sees the new prediction and creates new scribbles (if needed) which are fed back into the InterCNN model along with the image and the updated prediction.

3.2 Scribble Generation

Creating the scribbles is an essential part of the InterCNN model. The robot-user compares the ground-truth and the AutoCNN prediction to find the incorrectly labeled pixels for each class. It then randomly selects a pixel for each class, simulating a click by an actual user. Each click is then converted to a scribble via a Gaussian function, either in 2D or 3D. One round of the robot-user creating scribbles for each class label is referred to as an iteration.

A 2D Gaussian scribble $G(x, y)$ and a 3D Gaussian scribble $G(x, y, z)$ are created with the following Gaussian functions:

$$G(x,y) = \exp\left(-4\log(2)\cdot\frac{(x-x_0)^2+(y-y_0)^2}{\sigma^2}\right) \tag{3.1}$$

$$G(x,y,z) = \exp\left(-4\log(2)\cdot\frac{(x-x_0)^2+(y-y_0)^2+(z-z_0)^2}{\sigma^2}\right) \tag{3.2}$$

where (x, y, z) are the height, width, and depth of the image volume, (x_0, y_0, z_0) are the center points in the volume and σ is a constant that controls the width of the functions. Figure 3.2 shows the process of creating a 2D scribble, but the process is identical for creating a 3D scribble. The robot user sees the ground-truth and the model prediction (top row) and marks the differences between them for each class label (middle row). It then selects a random pixel from the incorrect predictions which is turned into a Gaussian scribble (bottom row).

When creating the 2D scribbles, the robot-user places a Gaussian scribble for each label in each slice in the volume. The total number of user input U is, therefore, $U = I \times L \times N$, where I is the number of iterations, L is the number of unique labels and N is the number of slices in the volume. This number matches the number of scribbles from the work by Bredell et al. When creating the 3D scribbles, the robot-user places a Gaussian scribble for each label in the whole volume, reducing the number of scribbles by a factor N. To find the balance between model accuracy and user interaction, we explore placing multiple 3D scribbles for each class label in a single iteration, where M is the number of scribbles created for each label. We make sure that the total amount of user input is smaller than in the 2D method by constraining M such that $M < N$.

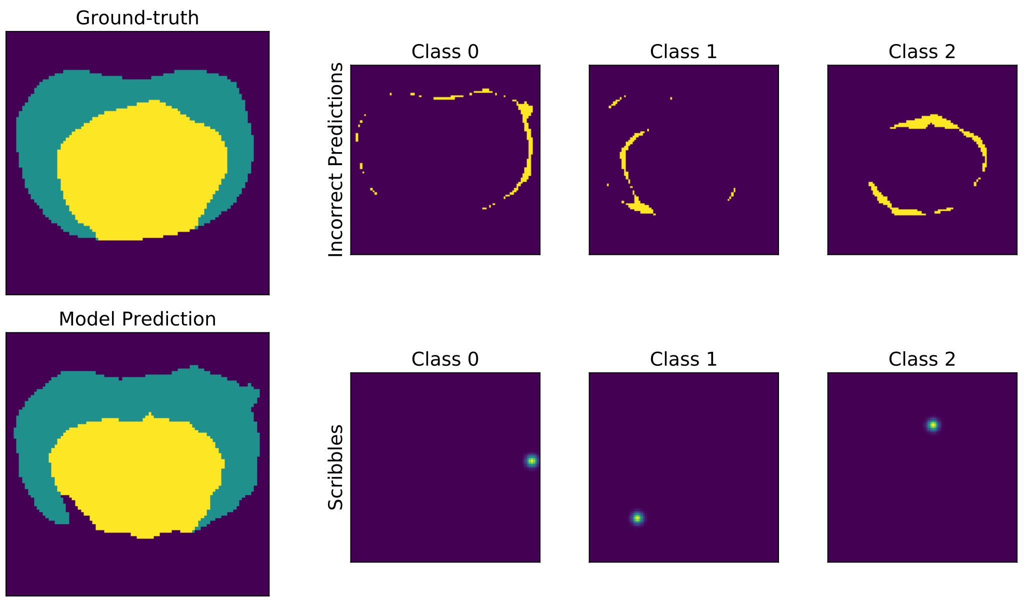


Figure 3.2: Creating the scribbles. The two images on the left show the ground-truth segmentation and the AutoCNN prediction. The upper row shows the incorrect prediction for each class label and the bottom row shows the scribbles created for each class label.

3.3 Model Architectures

3.3.1 Original U-Net Architecture

The U-Net architecture is a fully-connected CNN constructed for creating segmentation maps of input images. It is composed of a contracting encoder path and an expanding decoder path. The network has a depth of 5 and each block of layers in the encoder path is connected to the corresponding block in the decoder path via skip connections. It was originally designed for 2D image segmentation and then extended to 3D [7, 8].

In the 2D network architecture, a block of two 3×3 convolutional layers followed by a rectified linear unit (ReLU) activation function and a 2×2 max-pooling layer is repeated in the contracting path. In the expanding path, the same block is repeated but the max-pooling layer is removed and a 2×2 upsampling layer is placed at the start of the block.

The 3D network architecture reduces the depth of the network from 5 to 4, reduces the number of feature channels by half in the first convolutional layer of each block and it adds batch normalization to each convolution operation. Figure 3.3 shows the original 2D U-Net architecture and its 3D extension.

3.3.2 Our Modified Architecture

We use the 3D U-Net model with the following modifications:

- The Max pooling and transposed convolution layers have a 3D kernel of size $1 \times 2 \times 2$ for the depth, height, and width, respectively. This is done because the sub-sampled volumes have 15 slices, so not reducing the depth kernel size would result in a single-slice volume, effectively a 2D image, at the lowest layer in the network.

- A 20% dropout layer is added at the third level of the network. This is done to reduce the likelihood of overfitting the data.

- The number of feature channels in each convolution layer is halved. This is done to ensure that a sub-sampled volume with the full height and width of each image fits into the model.

Figure 3.4 shows our modified 3D U-Net model architecture where the two numbers in each block represent the input and output channels to the block. X and Y represent the number of input and output channels to the model.

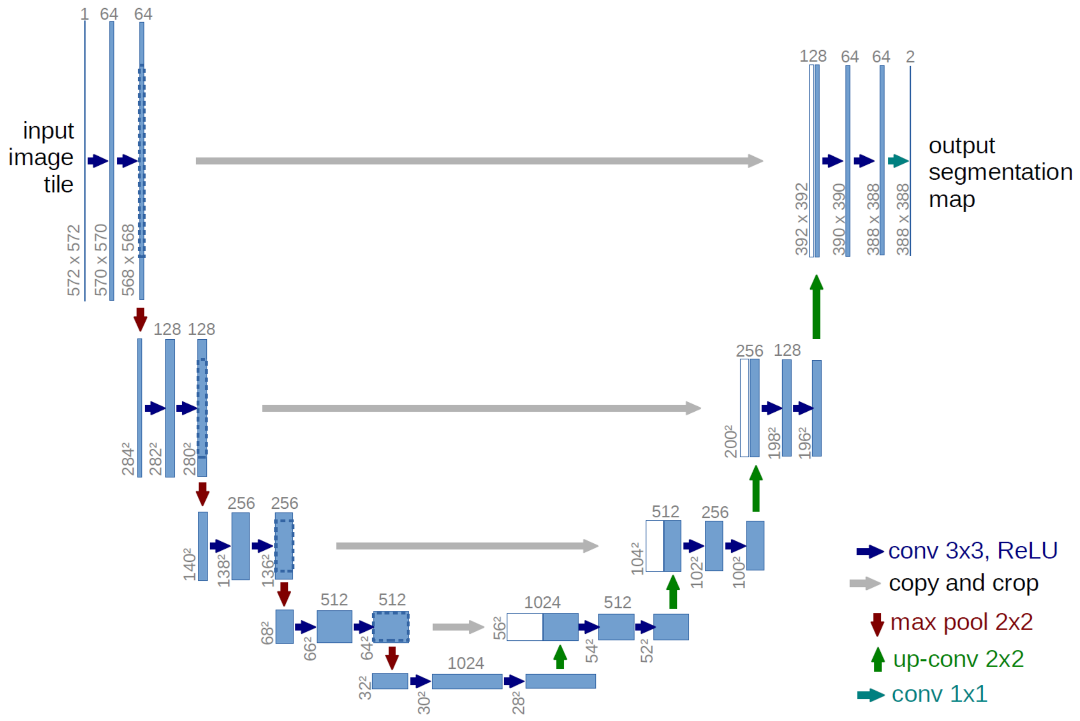
3.4 Evaluation Metrics

To evaluate the segmentation masks quantitatively, we use four metrics. The first two are the Dice similarity coefficient (DSC), also known as the Dice score or F1 score, calculated in either a 2D or 3D setting. The DSC between a target T and a prediction P is defined as [19]:

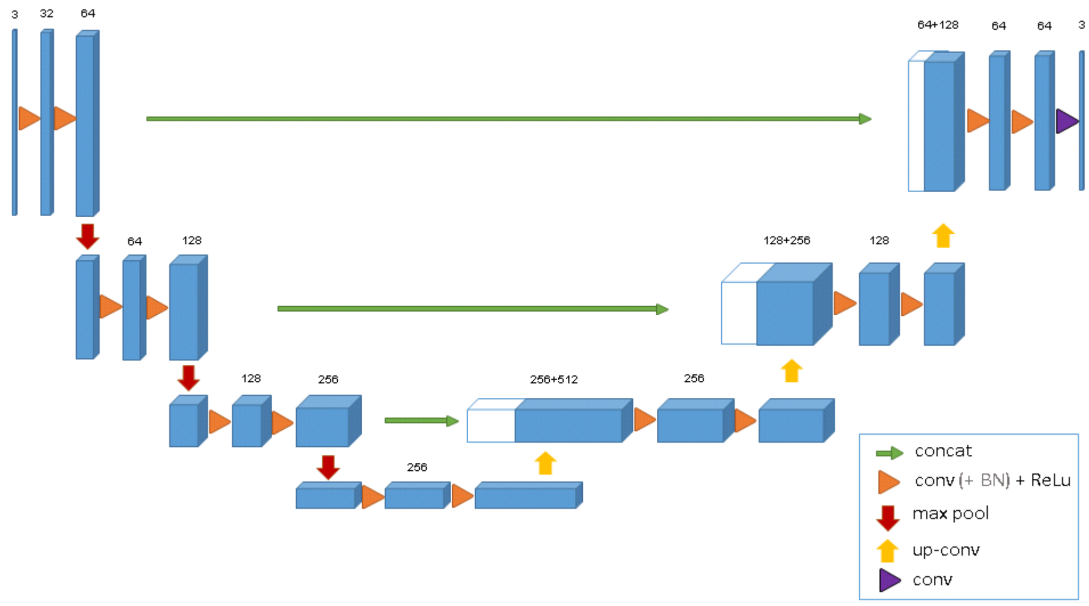
$$\text{DSC} = \frac{2|T \cap P|}{|T| + |P|} \tag{3.3}$$

where $|T|$ is the number of elements in set T and $|T \cap P|$ is the number of elements common in both sets. The DSC ranges between 0 and 1, with 1 meaning that the two sets are identical.

We use the DSC to measure our predictions in two ways; evaluating in 2D, meaning every slice in the volume gets a separate Dice score and the scores are then averaged, and evaluating in 3D, where the predicted volume's accuracy is measured as a whole. The key difference between 2D and 3D Dice score evaluation is that some slices may not include a specific label in the ground-truth, resulting in that slice having a Dice score of 1.0 if our model predicts no pixels with that particular label. That can boost the 2D scores compared to the 3D scores.



(a) The 2D U-Net architecture [7].



(b) The 3D U-Net architecture [8].

Figure 3.3: The 2D and 3D U-Net architectures by Ronneberger et al. and Çiçek et al.

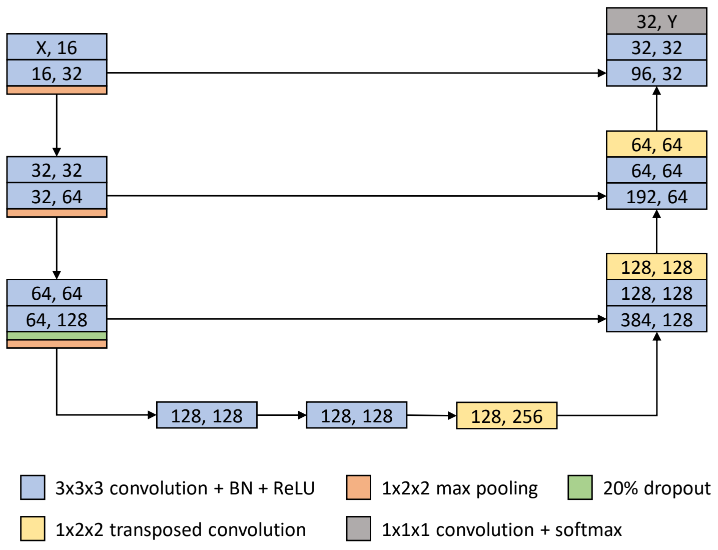


Figure 3.4: Our Modified 3D U-Net Architecture. The two numbers in each convolution block represent the number of input and output channels. X is the input channel (1 for the prostate dataset and Y is the output channel (3 for the prostate dataset).

The third metric is the Hausdorff distance (HD) and the fourth metric is the 95th percentile of the Hausdorff distance (HD95). The Hausdorff distance between two sets X and Y is defined as [20]:

$$\mathrm{HD}(X,Y) = \max\left(\mathrm{hd}(X,Y), \mathrm{hd}(Y,X)\right), \tag{3.4}$$

where $\mathrm{hd}(X,Y)$ and $\mathrm{hd}(Y,X)$ are the directed Hausdorff distances and are given by:

$$\mathrm{hd}(X,Y) = \max_{x\in X}\min_{y\in Y}\|x-y\|_2 \qquad \mathrm{hd}(Y,X) = \max_{y\in Y}\min_{x\in X}\|x-y\|_2 \tag{3.5}$$

where $\|x-y\|_2$ is the Euclidean distance between points $x \in X$ and $y \in Y$.

3.5 NCI-ISBI 2013 Challenge Dataset

The National Cancer Institute collaborated with the International Society for Biomedical Imaging to create the NCI-ISBI 2013 Challenge dataset that contains prostate gland magnetic resonance imaging (MRI) data [17]. The images were acquired as T2-weighted MR axial pulse sequences. Of the 60 patient cases, only 29 have multi-label segmentation masks where each mask consists of three different labels: background (0), peripheral zone tumor (1) and central zone tumor (2). The dataset is evaluated for each of the two tumor labels, ignoring the accuracy of the background prediction. The number of slices in each volume range between 15 and 24 and every slice has a resolution of 320×320. To ensure a consistent size of the input, the volumes are sub-sampled to $15 \times 320 \times 320$. The dataset is highly imbalanced, where labels $(0, 1, 2)$ compose 97.39%, 0.53% and 2.08% of the ground-truth. An example of the images and the corresponding segmentation mask can be seen in Figure 3.5 with a single MRI slice on the left side and its corresponding ground-truth segmentation on the right side.

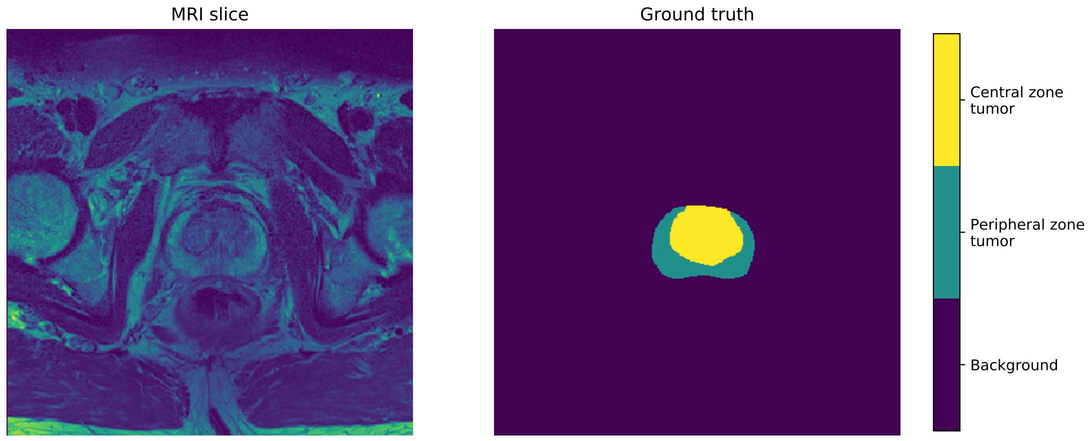


Figure 3.5: Example of the NCI-ISBI 2013 dataset.

3.6 Training Procedure

3.6.1 Data Splitting and Jackknifing

Each dataset is split into four different groups: AutoCNN training data, AutoCNN validation data, InterCNN validation data, and testing data. The InterCNN model is trained on the combined sets of the AutoCNN training and validation data. We select around 50% and 25% of the volumes for training and validating the AutoCNN model, respectively, to improve its robustness and to feed more unseen data to the InterCNN model.

The 29 volumes in the NCI-ISBI prostate dataset are split in the following way: 15 volumes for AutoCNN training, 8 volumes for AutoCNN validating, 23 combined volumes for InterCNN training, 1 volume for InterCNN validating and 5 volumes for testing both models.

Furthermore, the dataset is jackknifed. The subjects are randomly sampled three to five times to create different data splits. Our models are trained independently on the data splits and the models' accuracy scores are then averaged to achieve the final accuracy score.

3.6.2 Data Pre-Processing and Augmentation

We normalize each volume to have zero mean and unit variance. As mentioned in Section 3.5, we sub-sample the prostate volumes to $15 \times 320 \times 320$.

We employ the following data augmentation techniques: random horizontal flips, random rotations ($\pm 5°$), random scaling ($\pm 10\%$), random elastic deformations and random intensity shifts (± 0.1). Each technique has a 50% probability of being applied.

The elastic deformations, described in [7] and implemented by [21], are created in the following way: two (3×3) deformation fields are created by randomly sampling from a normal distribution with a standard deviation of 10. They are then expanded to the height and width of the volume using bicubic interpolation and applied to the image.

Figure 3.6 shows how an input image is data augmented with the five individual data augmentation techniques along with the deformation fields used in the elastic deformation.

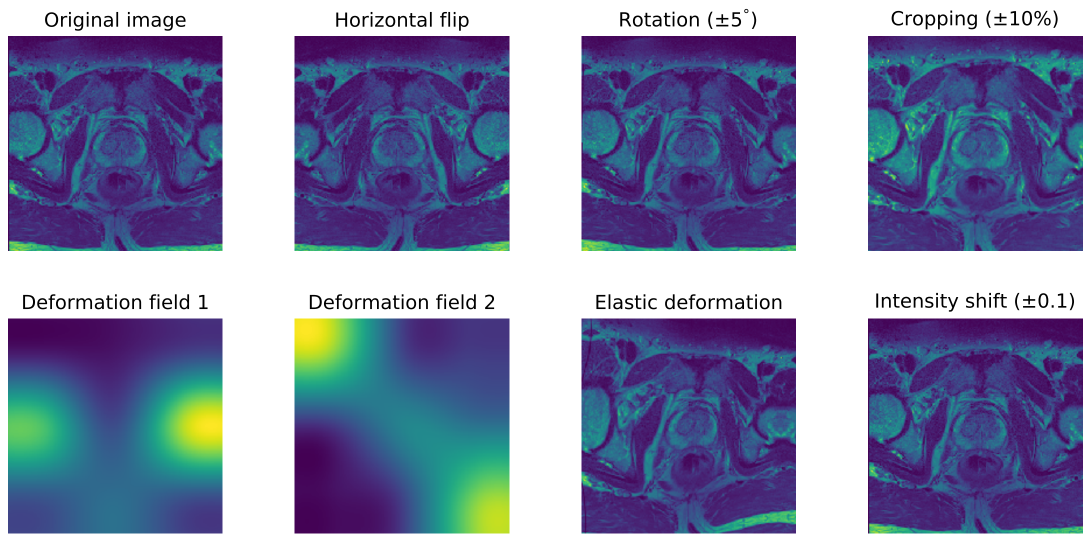


Figure 3.6: The data augmentation techniques used in our experiments.

3.6.3 Hyper-Parameters

The modified U-Net model detailed in 3.3.2 is implemented with the Kaiming weight initialization introduced by [22], as we are using the ReLU activation function in our model. We train both the AutoCNN and InterCNN networks using the Adam optimizer with an initial learning rate of $3.0\text{E}{-4}$ and a weight decay of $1.0\text{E}{-4}$. Since the optimizer uses an adaptive learning rate, we choose to reduce the initial learning rate by a small fraction, decreasing it by 1% if the validation loss hasn't decreased after every 100 epochs for AutoCNN or 10 epochs for InterCNN. The batch size is 1 and the loss function is the cross-entropy loss. We saved the model parameters that achieved the highest Dice score on the validation data.

The AutoCNN model is trained for 500 epochs and the InterCNN model is trained for 50 epochs with either 5 or 10 training iterations. The InterCNN model is then tested with 20 iterations.

Chapter 4

Experiments and Results

The code is developed in Python 3.6 with PyTorch 1.1.0 as the deep learning library. We use Pydicom and pynrrd to manage the prostate dataset. Scikit-learn and MedPy are used to calculate the Dice and Hausdorff scores. The data augmentation techniques are implemented with SciPy and OpenCV. All experiments are run in a Sun Grid Engine cluster with Intel Xeon E5 processors. Both the AutoCNN and InterCNN model are trained on an NVIDIA Titan Xp graphics card with 12 gigabytes of video memory.

In a multiclass segmentation task, we use the original labels of the dataset, segmenting to 3 unique classes in the prostate dataset, while a binary segmentation task signifies combining all tumor labels and segment the images in a background/foreground setting.

The models are evaluated with the four metrics detailed in Chapter 3.4. We do not include the accuracy of the background prediction, but only look at the accuracy of predicting the different tumor labels.

4.1 Using the NCI-ISBI Prostate Dataset

The prostate dataset, as described in Section 3.5, is split according to Section 3.6.1. Training the AutoCNN model for 500 epochs takes 20 hours and training the InterCNN model for 50 epochs takes 41 hours on average.

4.1.1 AutoCNN Model

We first train the AutoCNN model for the two segmentation tasks, a multiclass and a binary task. Table 4.1 shows the average accuracy of the AutoCNN model for the two tasks. The binary and multiclass scores are calculated by averaging over 15 and 40 distinct scores (1 or 2 classes $\times$ 5 test samples $\times$ 3 or 4 data splits), respectively.

We can see in Table 4.1 that the AutoCNN model is more effective at segmenting the prostate images in a binary setting than a multitask setting, reaching a Dice score of 86.49% on average

Table 4.1: Average accuracy of the AutoCNN model with the prostate dataset.

	Dice score [%]		Hausdorff [mm]	
Segmentation task	2D	3D	HD	HD95
Binary	81.92 ± 6.13	91.06 ± 2.18	33.51 ± 35.33	2.24 ± 0.88
Multiclass	77.81 ± 9.08	79.14 ± 16.76	26.41 ± 24.70	4.32 ± 5.93

versus 78.48%. The model performs much better in the binary task as it is generally an easier problem to solve due to the imbalanced nature of the dataset. Nevertheless, both of these scores create a strong foundation for the InterCNN to build on. An accurate AutoCNN is pivotal for a successful InterCNN network as the AutoCNN predictions are fed into the InterCNN model.

4.1.2 InterCNN Model with 2D Scribbles

We then train the InterCNN model for 10 iterations using 2D scribbles for the multiclass segmentation task, where we create one scribble per class in every slice. Table 4.2 shows the average accuracy of the InterCNN model over 20 testing iterations. The scores are calculated by averaging over 800 distinct InterCNN scores. We also include the Dice scores from Bredell et al. They used 2D scribbles as well with one scribble per class at every slice in the volume. Figure 4.1 shows the accuracy of our InterCNN model using 2D scribbles compared to their model.

Table 4.2: Average accuracy of the InterCNN model with the prostate dataset using 2D scribbles.

	Dice score [%]		Hausdorff [mm]	
	2D	3D	HD	HD95
Our model	89.19 ± 8.69	90.57 ± 8.17	11.91 ± 17.42	1.03 ± 2.62
Bredell et al.	94.87	92.58	–	–

Looking at Table 4.2 and Figure 4.1, we see that our InterCNN with 2D scribbles is underperforming compared to the 2D network of Bredell et al., with average Dice scores of 89.88% and 93.72%, respectively. Our AutoCNN is more accurate for both Dice scores, but our InterCNN accelerated slower than the 2D counterpart. We ultimately reached a 3D Dice score of 94.14%, 2.79% lower than what the 2D network achieved.

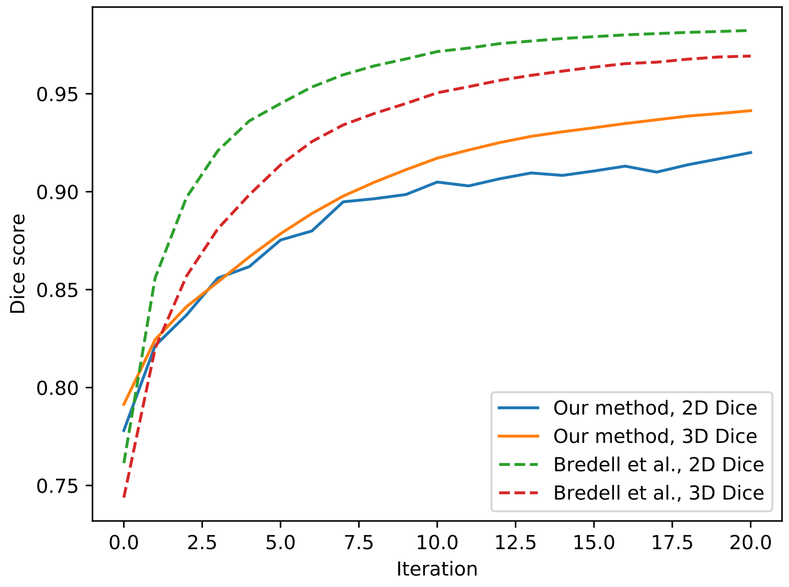


Figure 4.1: Accuracy of the InterCNN model with the prostate dataset using 2D scribbles. Comparing the 2D and 3D Dice scores of our model with the 2D model by Bredell et al.

4.1.3 InterCNN Model with 3D Scribbles

Finally, we train the InterCNN model using 3D scribbles for the two segmentation tasks. We use 10 training iterations and look at different ways of creating the scribbles. First, we use a vanilla setting (1 scribble per class in the whole volume) as a baseline for 3D scribbles. We want to determine if the model's accuracy is dependent on the size of the scribble, so we triple the size of the Gaussian curve by increasing σ in Equation 3.2 by a factor of 3. We also investigate if placing the scribbles intelligently, by finding the largest connected component (LCC) of the incorrect prediction and placing the scribbles there, affects the performance. Lastly, we explore how much an increased amount of user input improves the performance. We look at placing 5, 10, or 15 scribbles for each class in the volume over a single iteration. All configurations place scribbles at random locations, either in the volume or in the largest connected component of the incorrect prediction Table 4.3 shows the average accuracy of the InterCNN over 20 testing iterations for these six scribble configurations and the two segmentation tasks. The 2D scribble results by Bredell et al. from Table 4.2 are included as well. The binary and multiclass scores are calculated by averaging over 300 and 800 distinct scores (1 or 2 classes $\times$ 20 test iterations $\times$ 5 test samples $\times$ 3 or 4 data splits), respectively.

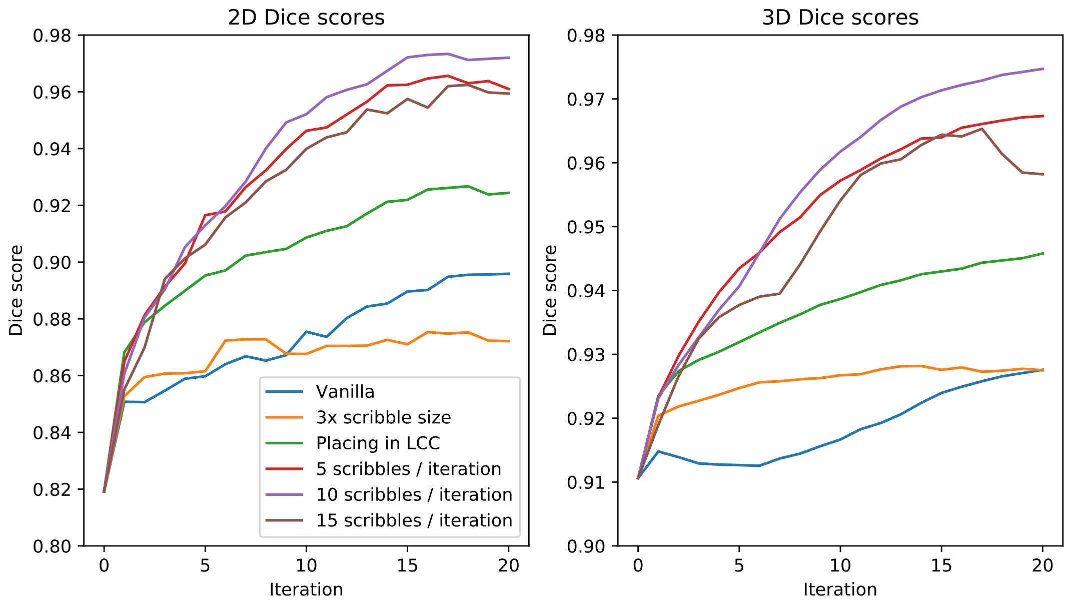
Figure 4.2 shows the InterCNN accuracy of each scribble configuration in a binary setting. Figure 4.2a shows the 2D and 3D Dice scores as a function of the 20 test iterations with iteration 0 as the AutoCNN score. Figure 4.2b shows the 2D and 3D Dice scores as a function of the total user interaction. Each iteration consists of 2 scribbles (for the background/foreground labels) multiplied by the number of scribbles placed in the volume. After 20 iterations, the first three scribble configurations have accumulated 40 user interactions. The final three configurations, which place additional scribbles in each volume, accumulate 200, 400, and 600 user interactions, respectively.

Figure 4.3 shows the InterCNN accuracy of the scribble configurations in a multiclass setting. Figure 4.3a shows the 2D and 3D Dice scores as a function of the 20 test iterations and Figure 4.3b shows the 2D and 3D Dice scores as a function of the total user interaction. Each iteration consists of 3 scribbles multiplied by the number of scribbles placed in the volume. After 20 iterations, the first three scribble configurations have accumulated 60 user interactions and the final three configurations have accumulated 300, 600 and 900 user interactions, respectively. We also compare these multiclass results with the previous scores from Bredell et al.

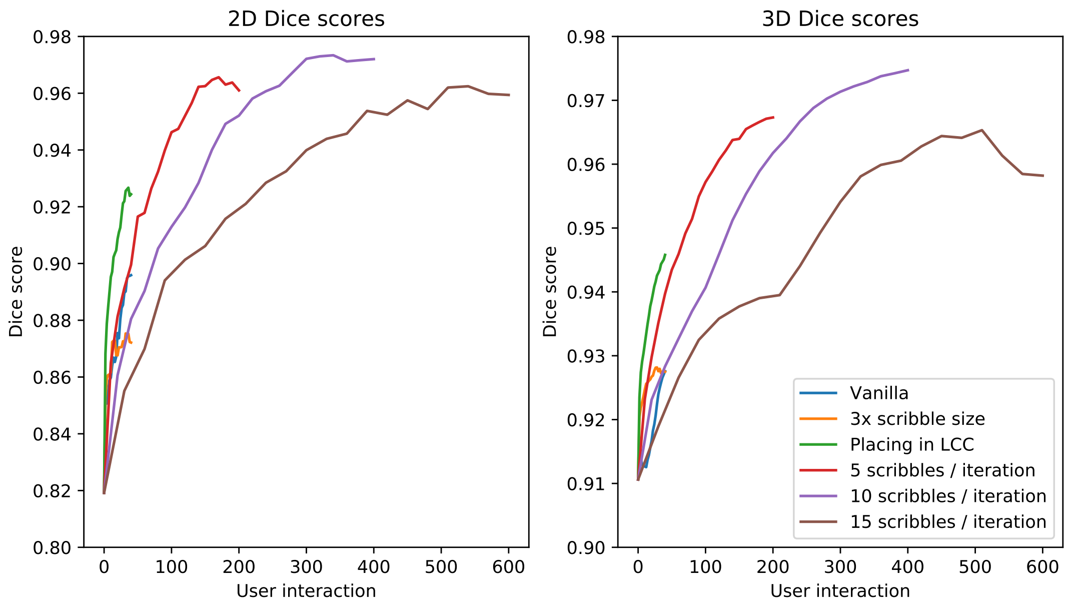
Table 4.3: Average accuracy of the InterCNN model with the prostate dataset using 3D scribbles.

		Dice score [%]		Hausdorff [mm]	
Task	Scribble configuration	2D	3D	HD	HD95
Binary					
	Vanilla	87.50 ± 3.75	91.88 ± 2.46	8.27 ± 3.03	2.11 ± 0.83
	3× scribble size	86.87 ± 4.87	92.60 ± 2.02	6.58 ± 3.04	1.78 ± 0.60
	Placing in LCC	90.72 ± 4.55	93.77 ± 1.59	6.69 ± 2.99	1.55 ± 0.56
	5 scribbles / iteration	93.58 ± 4.13	95.36 ± 1.75	5.14 ± 2.36	$\mathbf{1.20} \pm \mathbf{0.38}$
	10 scribbles / iteration	$\mathbf{94.11} \pm \mathbf{4.45}$	$\mathbf{95.72} \pm \mathbf{2.75}$	4.85 ± 1.99	1.27 ± 0.61
	15 scribbles / iteration	93.08 ± 7.13	94.95 ± 4.68	$\mathbf{4.56} \pm \mathbf{2.59}$	1.37 ± 0.74
Multiclass					
	Vanilla	82.26 ± 7.84	81.62 ± 13.21	13.82 ± 12.78	3.32 ± 3.04
	3× scribble size	80.77 ± 9.19	81.79 ± 13.11	16.70 ± 18.44	3.71 ± 4.01
	Placing in LCC	82.11 ± 9.43	83.09 ± 13.00	13.48 ± 8.68	3.22 ± 3.22
	5 scribbles / iteration	87.41 ± 7.18	87.31 ± 11.37	10.54 ± 8.50	2.07 ± 2.43
	10 scribbles / iteration	90.61 ± 6.66	90.97 ± 8.10	$\mathbf{8.00} \pm \mathbf{8.27}$	1.76 ± 3.57
	15 scribbles / iteration	$\mathbf{91.25} \pm \mathbf{7.94}$	$\mathbf{91.89} \pm \mathbf{7.75}$	9.19 ± 14.48	$\mathbf{1.45} \pm \mathbf{1.91}$
	Bredell et al.	94.87	92.58	–	–

Table 4.3 shows that InterCNN improves the AutoCNN predictions in the binary setting, with a peak increase of the 2D and 3D Dice scores by 12.19% and 4.66%. The highest scores are achieved by placing 10 scribbles in a volume in each iteration, reaching an average 3D Dice score of 95.72%. Figure 4.2a shows that placing 5 scribbles in each iteration achieves the most accurate and consis-



(a) 2D and 3D Dice scores as a function of test iterations.



(b) 2D and 3D Dice scores as a function of user interactions.

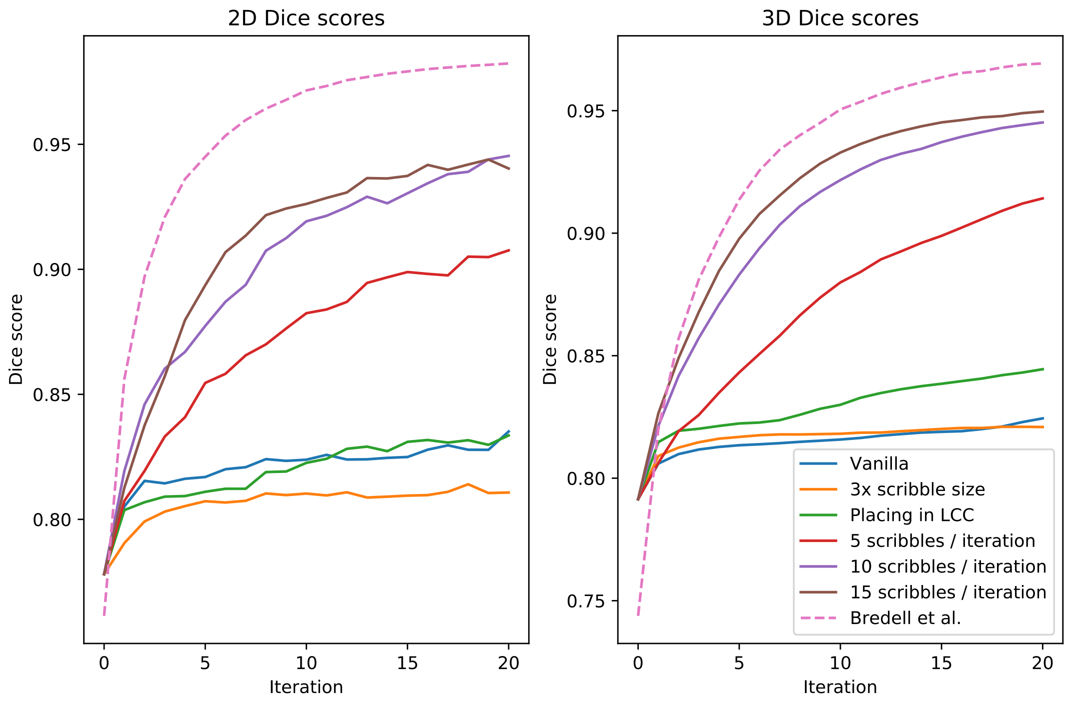
Figure 4.2: Accuracy of the InterCNN model with the prostate dataset using 3D scribbles in a binary setting.

tent 3D Dice results at the start. After 6 iterations, however, placing 10 scribbles proves more valuable and reaches 97.47% after 20 iterations, while placing 5 scribbles reaches 96.73%. Looking at Figure 4.2b, we see that placing the scribbles in the largest connected component reaches 94.58% accuracy after only 40 user interactions. Placing 5 scribbles reaches its maximum accuracy after 200 interactions and placing 10 scribbles reaches the highest score after 400 interactions.

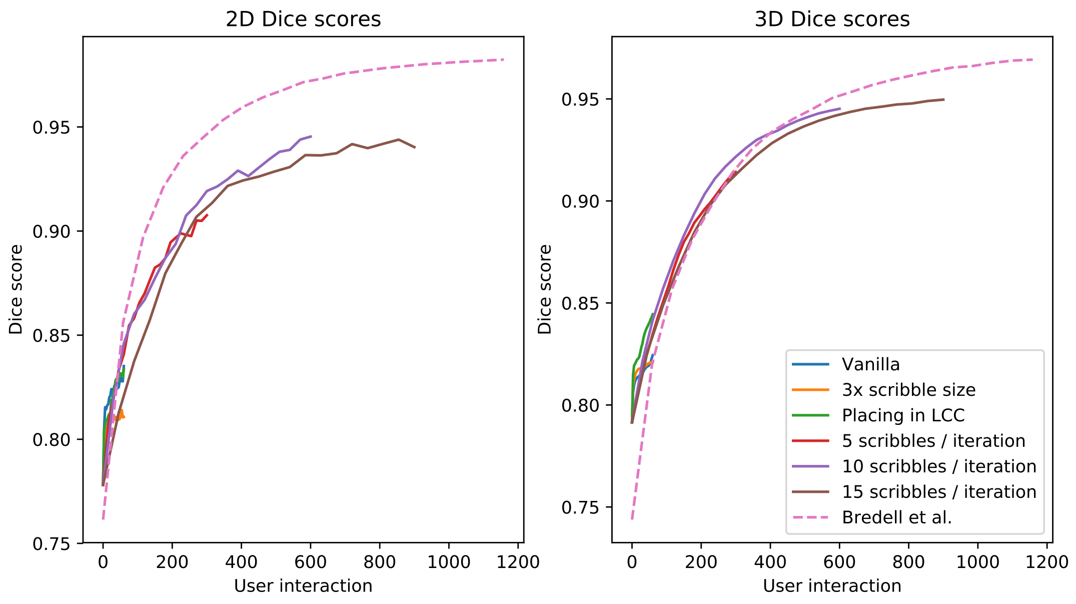
In the multiclass setting, the AutoCNN predictions are again improved with the InterCNN model, with a maximum increase of the 2D and 3D Dice scores by 13.44% and 12.75%. Placing 15 scribbles in each iteration attains the highest Dice scores out of the six configurations, with an average Dice score of 91.57%. Figure 4.3a shows that placing 15 scribbles in each iteration achieves the most accurate 3D Dice scores throughout the testing phase, reaching 94.97% accuracy after 20 iterations. All of the 3D scribble configurations, however, are lower than the 2D interCNN accuracy scores of Bredell et al. When the scores are plotted as a function of the number of user interactions in Figure 4.3b, the 3D scribbles benefit from requiring fewer user interactions than the 2D interCNN model.

Figure 4.4 gives a closer look at the 3D Dice scores. The top graph, plotting the accuracy for up to 600 user interactions, shows that placing 10 scribbles in each iteration (bold purple line) achieves a higher accuracy from the start, up to 385 user interactions. After that, the 2D interCNN model overtakes our 3D one. The bottom graphs show the accuracy of up to 150 user interactions, which is a more realistic number of interactions for a human user. We see that the same scribble configuration of 10 scribbles per iteration is $1 - 2\%$ higher than the 2D interCNN model between 60 (the number of interactions required for a single iteration in the 2D model) and 150 user interactions.

With 560 slices in the 29 volumes of the prostate dataset, a single volume has around 19 slices on average. The final three experiments of Table 4.3, placing 5, 10, or 15 scribbles per volume, require 74%, 48% and 22% fewer user interactions after 20 testing iterations, respectively. The highest 3D accuracy scores are 5.51%, 2.41% and 1.96% lower than the 2D model by Bredell et al. The ideal configuration is, therefore, placing 10 scribbles in each iteration, as it requires 48% fewer user interactions to reach a 3D Dice score that is only 2.41% lower.

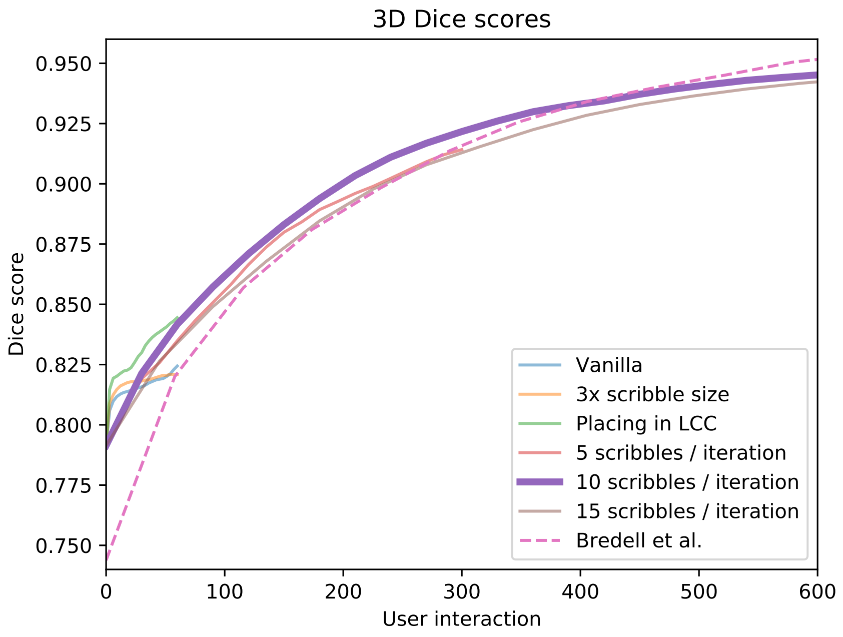


(a) 2D and 3D Dice scores as a function of test iterations.

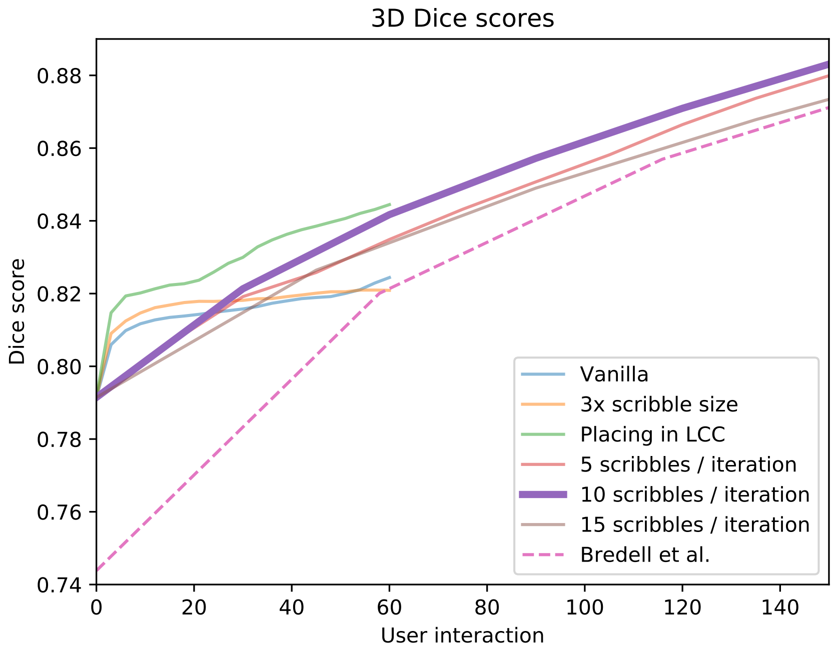


(b) 2D and 3D Dice scores as a function of user interactions.

Figure 4.3: Accuracy of the InterCNN model with the prostate dataset using 3D scribbles in a multiclass setting.



(a) Looking closer at Figure 4.3b, up to 600 user interactions.



(b) Looking closer at Figure 4.3b, up to 150 user interactions.

Figure 4.4: A closer view of the 3D Dice accuracy in Figure 4.3b.

Chapter 5

Discussion

We have demonstrated that our framework can accurately segment volumetric MR images of a prostate. We created a strong automatic segmentation basis with our AutoCNN model, reaching average 3D Dice scores of 91.06% and 79.14% for the binary and multiclass settings, respectively. Our 2D InterCNN model, though not as powerful as the model by Bredell et al., further improved the average multiclass segmentation accuracy by 11.43% using 2D scribbles and by 12.75% using 3D scribbles and placing 15 scribbles per class in each iteration. The most valuable configuration, however, involved placing 10 3D scribbles in each iteration, where we reached a peak 3D Dice score of 94.52% after 20 iterations. Our highest score was 2.41% lower than the network by Bredell et al. but we reduced the number of user interactions required by 48%. We were also able to predict more accurately when looking at fewer user interactions as shown in Figure 4.4. Our method was more accurate than our 2D predecessor up to 385 user interactions, thanks to a stronger AutoCNN foundation.

It is important to note that the binary results were averaged over 3 data splits and the multiclass results were averaged over 4 data splits. We originally trained the multiclass InterCNN model on 5 splits. However, there is an anomaly within the fifth data split, where the scores on two of the five test volumes drop significantly. The AutoCNN accuracy decreases by $20 - 40\%$ and the average InterCNN accuracy is decreased by $10 - 20\%$. We have yet to find the cause of this irregularity within the data split.

Looking at Table 4.3, it is interesting to see that both the vanilla task and placing 5 scribbles per iteration have a lower Dice score when measured in 3D than in 2D. That indicates that the model is more accurate in segmenting the volume slice-by-slice rather than a whole volume at a time.

We can also see from Tables 4.1 – 4.3 that the multiclass Dice scores, especially the 3D scores, have high standard deviations. The AutoCNN's abnormal deviation can be explained by the fact that the scores for class 1 in the dataset are more inconsistent than the second class, causing a bigger fluctuation in the averaged scores. The InterCNN scores create a particularly large deviation as they are averaged over a big range, either 300 or 800 distinct scores that range from 78% up to 98%.

The configuration of placing a scribble in the largest connected component is an intriguing experiment. It outperformed the vanilla and larger scribbles configurations in both the binary and multiclass settings, especially when looking at the scores as a function of user interaction. Unfortunately, it is less relevant in the real world as it would be difficult for a human user to locate the largest incorrect automatic prediction when creating the scribbles.

5.1 Limitations of 3D Semantic Segmentation

Managing the GPU memory is one of the biggest challenges with 3D data and 3D networks. Our AutoCNN model with the prostate data has 3,851,763 parameters and one forward/backward pass of a $15 \times 320 \times 320$ volume requires 10.38 gigabytes of GPU memory. The InterCNN, with a slightly bigger input size but the same model architecture, has 3,853,491 parameters and one forward/backward pass occupies 10.40 gigabytes of GPU memory. There are available networks that can reduce the memory consumption. One method, by Brügger et al., reduces the memory usage by 40% by using partially reversible architectures [21]. The downside is that it increases the training time by 50%, extending our InterCNN training time by 20 hours.

Batch size is a big factor when working with 3D networks. Feeding in a sub-sampled volume instead of a few slices at a time provides more spatial information but simultaneously decreases the total number of batches available. That reduces the number of forward passes of the input data through the model. Therefore, to increase the number of unique batches fed to the network and improve the model's reliability, data augmentation is a necessary step.

5.2 Future Work

In Chapter 4, we only present our results on the MRI prostate dataset. We experimented with the 2015 BraTS Challenge dataset but were not able to finish our tests to include them in this thesis. It would be beneficial to run our experiments using (or any other dataset), as achieving comparable results on another dataset would demonstrate that our framework is versatile and reliable.

For future development of the AutoCNN and InterCNN framework, an important step would be to solve the issues with the GPU memory, either by training the models on GPUs with more graphics memory or using memory-saving techniques. There are graphics cards on the market with double the memory of what we had available. That would enable creating larger networks or allowing larger input sizes, which could greatly benefit our models' capabilities. Another possibility is using the method of [21] and accept the extended training time in favor of the ability to construct larger models.

As mentioned in Section 3.3.2, the input images have a height and width of 320×320, with $15-24$ slices in each volume. We sub-sampled each volume to have 15 slices so each input volume is equal in size, but PyTorch allows variable-sized inputs. Therefore, the model parameters might vary for the different inputs. We chose to disregard this feature in PyTorch and keep the input sizes consistent, but enabling it equates not needing to sub-sample the volumes which reduces the training time of our models.

Further tuning of our models is possible, such as implementing a loss summation procedure or selecting a different loss function. Summing the loss entails saving the computation graph after each iteration (one forward pass through the network). After several iterations, the gradients would then be calculated and the model parameters updated. That might help the model identify which pixels are incorrectly classified, giving it only the latest scribbles. The current way of updating the model parameters after each forward pass might be detrimental to the learning phase, since new scribbles are added to the scribble memory. The graphs are substantial in size and are stored on the GPU memory. We were not able to fit our models and the graphs on our GPUs but a graphics card with more graphics memory would solve this problem. Furthermore, we chose the cross-entropy loss in our architectures. We experimented with the Dice loss and found no improvement but optimizing the loss functions, such as combining the two functions, might refine the training of our models.

Figure 4.3 shows that the 3D AutoCNN model is more accurate than its 2D counterpart, while the 2D InterCNN is superior to the 3D scribble configurations. Combining these two architectures might further improve the work by Bredell et al.

Numerous other convolutional neural networks are specifically tuned for medical image segmentation such as the V-Net, DCAN, or the P-Net from the DeepIGeoS framework [9,10,14]. They have achieved high accuracy scores on various challenge datasets and could be better suited as the building blocks for our AutoCNN and InterCNN models.

Finally, the user input is currently set to clicks that are fed to a 2D or 3D Gaussian function. There are other types of user input that might improve the accuracy of our InterCNN over each iteration, e.g. allowing the user to draw lines or shapes as was presented in [14, 15], or using other functions than a Gaussian. Exploring more configurations for the 3D Gaussian scribbles could prove beneficial, such as creating more than 15 scribbles in each iteration to match the exact amount of user interaction that was used by Bredell et al.

Chapter 6

Conclusion

In this thesis, we designed a framework with two 3D fully-convolutional networks for segmenting medical images, first automatically and then iteratively via scribbles generated by a robot-user. We introduced multiple ways of creating the scribbles in 3D, most of which could be applied by a human in a real-world scenario. Our models were able to accurately segment an MRI prostate volume, both in a binary and a multiclass setting. Our automatic segmentation algorithm achieved 79.14% during test time, which was further improved by our semi-automatic algorithm, with one scribble configuration reaching 94.97% accuracy while using 48% less user interaction than our predecessor. It produced more accurate segmentation masks with up to 385 user interactions. We believe our work could be expanded further by implementing more effective ways of creating scribbles or fixing issues with the GPU memory to create larger and/or deeper models.

Bibliography

[1] M. H. Hesamian, W. Jia, X. He, and P. Kennedy, "Deep Learning Techniques for Medical Image Segmentation: Achievements and Challenges," *Journal of Digital Imaging*, vol. 32, pp. 582–596, August 2019.

[2] D. Shen, G. Wu, and H.-I. Suk, "Deep Learning in Medical Image Analysis," *Annual Review of Biomedical Engineering*, vol. 19, pp. 221–248, June 2017. PMID: 28301734.

[3] T. Sakinis, F. Milletari, H. Roth, P. Korfiatis, P. M. Kostandy, K. Philbrick, Z. Akkus, Z. Xu, D. Xu, and B. J. Erickson, "Interactive segmentation of medical images through fully convolutional neural networks," March 2019.

[4] G. J. S. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. W. M. van der Laak, B. van Ginneken, and C. I. Sánchez, "A Survey on Deep Learning in Medical Image Analysis," *Medical Image Analysis*, vol. 42, pp. 60 – 88, July 2017.

[5] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: an overview and application in radiology," *Insights into Imaging*, vol. 9, pp. 611–629, August 2018.

[6] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F. Li, "ImageNet Large Scale Visual Recognition Challenge," *CoRR*, vol. abs/1409.0575, 2014.

[7] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015* (N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, eds.), (Cham), pp. 234–241, Springer International Publishing, November 2015.

[8] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, "3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016* (S. Ourselin, L. Joskowicz, M. R. Sabuncu, G. Unal, and W. Wells, eds.), (Cham), pp. 424–432, Springer International Publishing, October 2016.

[9] F. Milletari, N. Navab, and S. Ahmadi, "V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation," in *2016 Fourth International Conference on 3D Vision (3DV)*, pp. 565–571, October 2016.

[10] H. Chen, X. Qi, L. Yu, and P. Heng, "DCAN: Deep Contour-Aware Networks for Accurate Gland Segmentation," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2487–2496, June 2016.

[11] C. Rother, V. Kolmogorov, and A. Blake, "GrabCut: Interactive foreground extraction using iterated graph cuts," in *ACM transactions on graphics (TOG)*, vol. 23, pp. 309–314, ACM, 2004.

[12] C. Sommer, C. Straehle, U. Köthe, and F. A. Hamprecht, "Ilastik: Interactive learning and segmentation toolkit," in *2011 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, pp. 230–233, March 2011.

[13] L. Grady, T. Schiwietz, S. Aharon, and R. Westermann, "Random Walks for Interactive Organ Segmentation in Two and Three Dimensions: Implementation and Validation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2005* (J. S. Duncan and G. Gerig, eds.), (Berlin, Heidelberg), pp. 773–780, Springer Berlin Heidelberg, 2005.

[14] G. Wang, M. A. Zuluaga, W. Li, R. Pratt, P. A. Patel, M. Aertsen, T. Doel, A. L. David, J. Deprest, S. Ourselin, and T. Vercauteren, "DeepIGeoS: A Deep Interactive Geodesic Framework for Medical Image Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PP, July 2017.

[15] G. Wang, W. Li, M. A. Zuluaga, R. Pratt, P. A. Patel, M. Aertsen, T. Doel, A. L. David, J. Deprest, S. Ourselin, and T. Vercauteren, "Interactive Medical Image Segmentation Using Deep Learning With Image-Specific Fine Tuning," *IEEE Transactions on Medical Imaging*, vol. 37, pp. 1562–1573, July 2018.

[16] M. Amrehn, S. Gaube, M. Unberath, F. Schebesch, T. Horz, M. Strumia, S. Steidl, M. Kowarschik, and A. Maier, "UI-net: Interactive Artificial Neural Networks for Iterative Image Segmentation Based on a User Model," in *Proceedings of the Eurographics Workshop on Visual Computing for Biology and Medicine*, VCBM '17, (Goslar Germany, Germany), pp. 143–147, Eurographics Association, September 2017.

[17] N. Bloch, A. Madabhushi, H. Huisman, J. Freymann, J. Kirby, M. Grauer, A. Enquobahrie, C. Jaffe, L. Clarke, and K. Farahani, "NCI-ISBI 2013 Challenge: Automated Segmentation of Prostate Structures," *The Cancer Imaging Archive*, 2015.

[18] G. Bredell, C. Tanner, and E. Konukoglu, "Iterative Interaction Training and Memory Effects on Segmentation Editing," master's thesis, ETH Zürich, October 2018.

[19] L. R. Dice, “Measures of the Amount of Ecologic Association Between Species,” *Ecology*, vol. 26, no. 3, pp. 297–302, 1945.

[20] R. T. Rockafellar and R. J.-B. Wets, *Variational Analysis*, vol. 317 of *Grundlehren der mathematischen Wissenschaften*. Springer-Verlag Berlin Heidelberg, 1 ed., 2009.

[21] R. Brügger, “Partially Reversible U-Net for Memory-Efficient Volumetric Image Segmentation,” master’s thesis, ETH Zürich, April 2019.

[22] K. He, X. Zhang, S. Ren, and J. Sun, “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification,” in *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV ’15, (Washington, DC, USA), pp. 1026–1034, IEEE Computer Society, November 2015.