CS 230 Final Report

George Liu – SUNETID gliu2

Automated segmentation of CT temporal bone structures

Category: Computer Vision

Introduction

CardinalSim (https://cardinalsim.stanford.edu/) is a surgical simulator that uses 3D glasses, computerized rendering of temporal bone anatomy from loaded CT scans of the head, and a haptic feedback device to help train otolaryngology residents in surgery before they do it for real patients (Figures 1 and 2). One challenge is that the 3D rendering requires manual segmentation of relevant temporal bone anatomy, which may take up to hours, for each individualized CT scan surgical rehearsal.

We are developing a deep learning application to automate segmentation of temporal bone structures on CT images. This will streamline surgical simulation by integrating labeling of surgical anatomy with CardinalSim, and reduce the reliance on external software for manual segmentation.

Our approach to automated segmentation relies on fully convolutional neural networks (FCNs), a class of deep learning models specialized for computer vision and in particular semantic segmentation (i.e. class labeling) [1]. We are extending previously described FCN models, such as 3D U-Net [2] and V-Net [3], to our task of volumetric CT image segmentation of temporal bone structures. This is possible because of our simulation database of labeled scans, and our computational resources which allow for training FCN models on our database. We are further supported by our collaborators with computer vision expertise.

Here I describe initial work to set up and begin training a 3D U-Net model to automatically segment 5 temporal bone structures: the carotid artery, facial nerve, sigmoid sinus, ossicles, and cochlea (Figure 3).

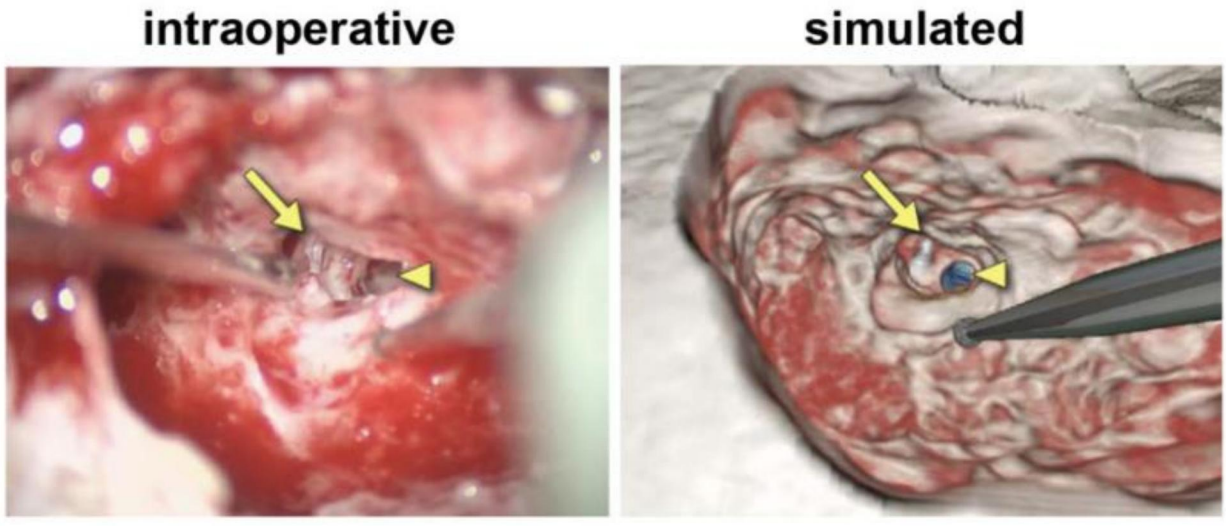


Figure 1. Intraoperative view (left) and surgical simulation (right) in CardinalSim of a tympanomastoid surgery (drilling behind the ear to get better access to the ear drum and middle ear).

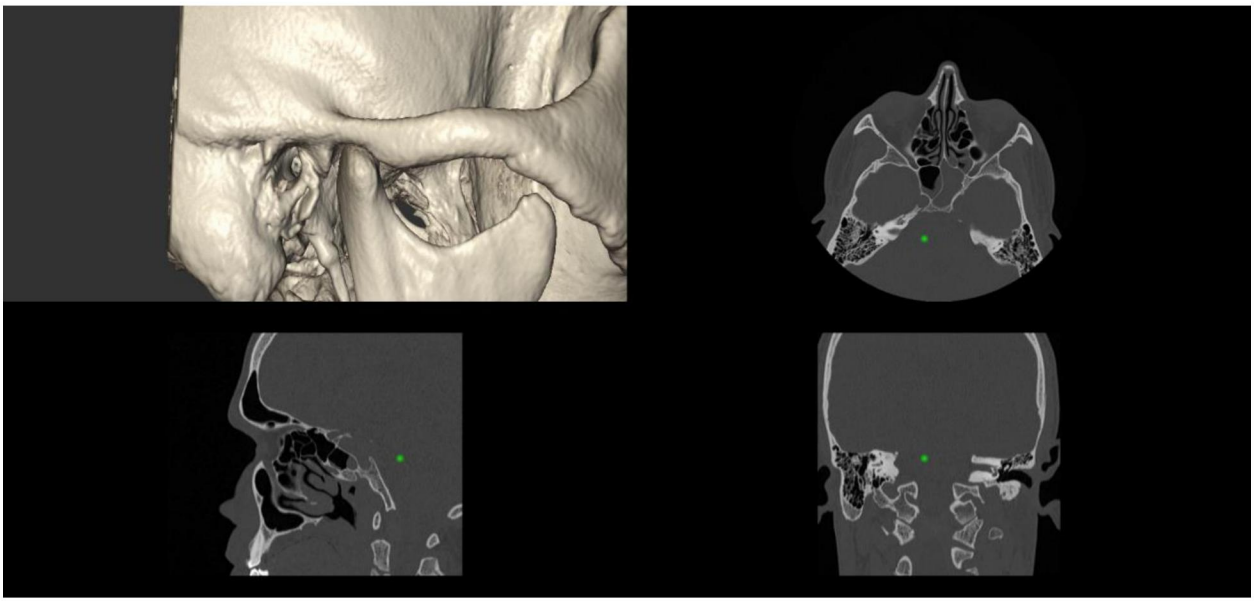
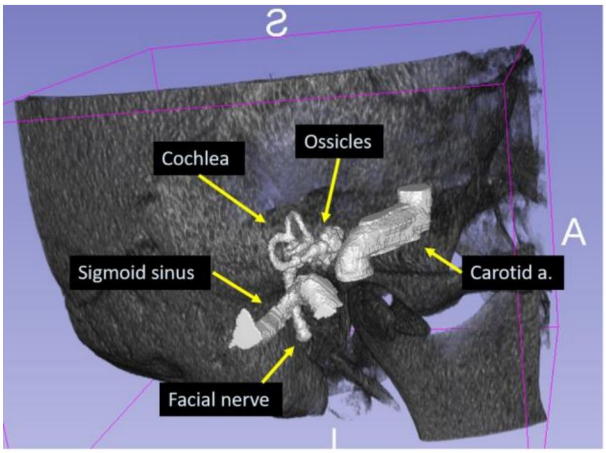


Figure 2. Realistic 3D rendering of lateral skull base anatomy in CardinalSim (top-left corner). Intensity-based thresholding of original CT scans in axial (top-right), sagittal (bottom-left), and coronal (bottom-right) views are used to produce 3D structure of bone.

Dataset:

Our dataset is comprised of 72 original CT scans and their ground-truth manual segmentations for 5 structures: the internal carotid artery, sigmoid sinus, facial nerve, ossicles, and cochlea (Figure 3).

Each scan was of a unique ear of a patient. 22 manually segmented temporal bone CT scans were obtained from patients at Stanford Hospital. 50 additional CT scans were obtained from patients seen at University of Western Ontario Medical Center.



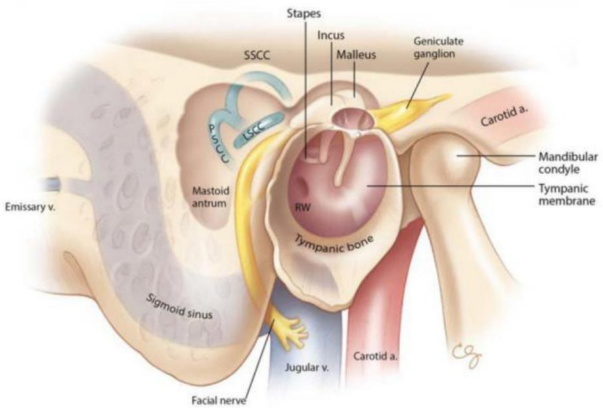


Figure 3. Example of labeled training data (left). The 3D rendering of an original CT scan is visualized in 3D Slicer (left) to show the anatomy of the right ear in one patient. Manual segmentations of five structures of interest (cochlea, ossicles, cochlea, sigmoid, sinus, and facial nerve) are and overlaid as white 3D-rendered structures within the original CT scan, and labeled (yellow arrows). Schematic of the anatomy of the right ear is shown for comparison (right).

Approach:

Implementation of training a 3D U-net model for semantic segmentation was done in PyTorch (Python 3.5). Input tiles of size 64x 64x56 voxels were used to train 3D-Unet model. There are 576 tiles in each image, because the image size is 512x512x481 voxels.

Initial training was performed on CPU on a Windows 10 laptop. I used a cross entropy loss function. Because of memory and computation constraints, I trained the model on only one image, feeding one tile at a time to train by stochastic gradient descent. This first attempt at training produced the learning curve shown in Figure 4.

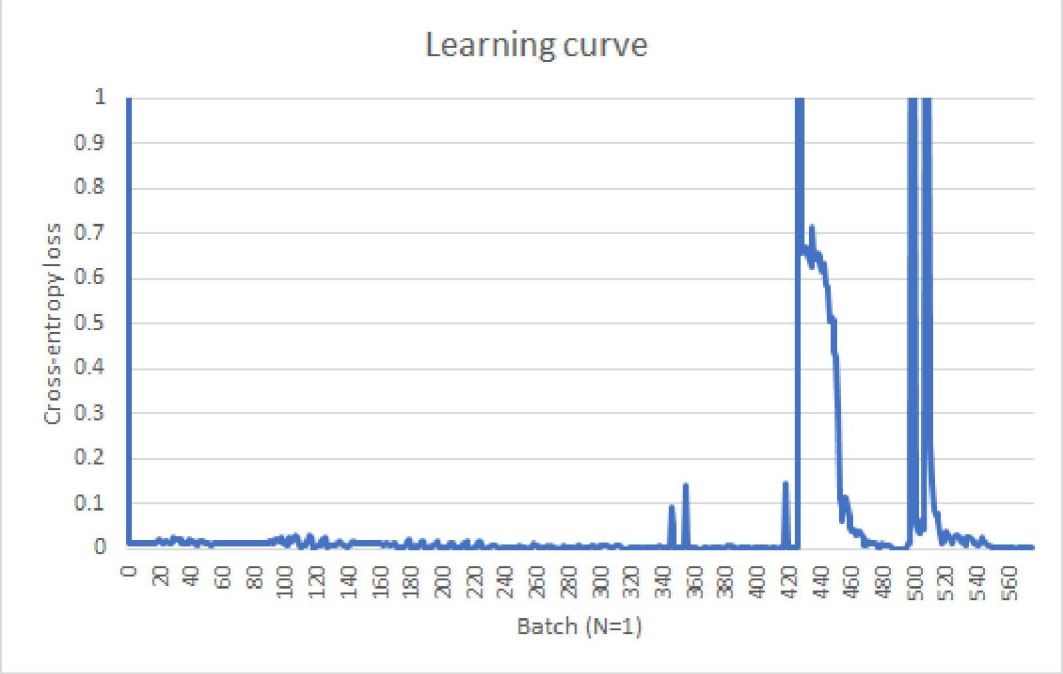


Figure 4. Learning curve after training 3D-Unet using stochastic gradient descent for all 576 tiles with carotid artery segmentation labels only in one CT scan image.

Next, a MSI Trident X 9SD-021US computer (PC Intel Core i7-9700K RTX 2070 8GB GDDR6 16GB 512GB PCIe NVME SSD) was purchased by our lab. This allowed us to use one 8 GB NVIDIA RTX 2070 GPU to train our 3D-Unet model using CUDA in PyTorch. With further GPU memory, I increased the mini-batch size from 1 to 4 tiles of 64x 64x56 pixels, and used sliding tiles to form each mini-batch so that not the same input was repeatedly used for training each iteration. For 1000 iterations, this input tiles

from about 7 CT scans. After 1000 iterations of training, we saw a learning curve that was a flat line after the two iterations at a value of 0.6, with an occasional bump (data not shown).

Before saving the results of training loss during 1000 iterations, I repeated training for 2000 iterations to increase the amount of training data used for training the 3D-Unet model. However during this training, our computer made a loud crack sound and the computer's power went off. Evaluation by Stanford Medicine's Information Resource and Technology staff technician indicated a likely hardware issue involving the power supply and/or mother board. I shipped the computer back to MSI for warranty repair and evaluation. The computer is expected to be repaired and returned no earlier than 1 week from today (3/19/2019).

In the meantime, I explored resources on AWS EC2 P3 instances and Google Compute Engine to move this project forward using cloud computation. However both accounts did not have GPU quota limits greater than 0, and so I could not move forward on this front as well.

Insights/Discussion

My next steps, once my AWS/GCE GPU limit increase requests are granted or the MSI computer is repaired and returned, is to assess whether training can be enhanced to show a downward-sloping learning curve. Based on assessment of my training data, I believe I have an imbalanced dataset where the negative samples (background pixels) overwhelmingly outnumber the number of foreground pixels (internal carotid artery, ossicles, cochlea, facial nerve, sigmoid sinus). I can address the issue by evenly sampling positive and negative samples, by creating each mini-batch of 4 tiles such that 2 tiles are foreground and 2 background.

In addition, I believe that structures in CT scans are stereotyped in terms of their absolute coordinates in a CT scan. This is because each patient's head is fixed in a CT scanner in a similar way. One way to take advantage of this global position data for segmentation is to input the whole image into the 3D-Unet (rather than just tiles). However, this will require more GPU memory. An alternative is to input the position coordinates of the image into a separate fully-connected neural network, and then skip-forward-feed that network to the final layer of the original 3D-Unet architecture for calculation of each pixel's class probabilities.

While I plan to begin with training 3D-Unet for segmentation of 1 class label, I will later train to segment foreground (any of 5 class labels) and then multiclass training (simultaneous training with 5 class labels).

My PI is also interested in purchasing additional hardware to help move this project forward. I will certainly look into this possibility as well.

References:

1. Long J, Shelhamer E, Darrell T. Fully Convolutional Networks for Semantic Segmentation ppt. CVPR 2015 Proc IEEE Conf Comput Vis Pattern Recognit. 2015;39:3431–40.

2. Çiçek Ö, Abdulkadir A, Lienkamp SS, Brox T, Ronneberger O. 3D U-net: Learning dense volumetric segmentation from sparse annotation. Lect Notes Comput Sci (including Subser Lect Notes Artif Intell Lect Notes Bioinformatics). 2016;9901 LNCS:424–32.

3. Milletari F, Navab N, Ahmadi SA. V-Net: Fully convolutional neural networks for volumetric medical

image segmentation. Proc - 2016 4th Int Conf 3D Vision, 3DV 2016. 2016;:565–71.

Code:

https://github.com/gliu2/CSim