MULTI-ORGAN SEGMENTATION IN HEAD AND NECK CANCER

Muhammad Osama Khan, Muhammad Shujaat Mirza, Muhammad Muneeb Afzal
mok232@nyu.edu, msm622@nyu.edu, mma525@nyu.edu

1 ABSTRACT

Head and neck cancer is one of the most common cancers around the world (Siegel et al. (2014)). The main treatment strategy used for this cancer is radiation therapy (RT), which relies on delineating the organs-at-risk (OARs) (Han et al. (2008)) from a Computed Tomography (CT) scan. Since there can be hundreds of slices in a CT scan, it is not only time consuming to manually segment OARs but it can also lead to errors. Therefore, automatic (without any assistance from medical staff) OARs segmentation is imperative as it will decrease segmentation time and possible errors and finally help in radiation therapy planning. Since the advent of deep learning, U-Net (Han et al. (2008)) has been a seminal DL network in medical imaging segmentation and has inspired a plethora of other networks (Ronneberger et al. (2015); Çiçek et al. (2016); Zhang et al. (2007); Pednekar et al. (2018); Wachinger et al. (2015); Commowick et al. (2008); Fritscher et al. (2014); Wang et al. (2017)). In this project, we tackle the challenging task of multi-organ segmentation in head and neck cancer by dealing with challenges such as imbalanced classes, large memory footprint and small datasets. For each of these challenges, we tried multiple approaches and make recommendations for the practitioners in the community to better tackle these issues.

2 INTRODUCTION

In this work, we are invested in exploring approaches that could tackle the following three challenges faced by the medical imaging community:

2.1 IMBALANCED CLASSES

In biomedical segmentation, we usually deal with segmentation masks where objects of interest (such as tumors, organs-at-risk, lesions) occupy a small percentage of the mask. Concretely, Figure 1 below shows a typical mask for binary segmentation. We note that tumors (objects of interest) only occupy a small portion of the input image. This implies that even if we output the third image (all black), we will achieve a whopping accuracy of 98%! To this end, we need to find a way by which all pixel classifications in segmentation are not given equal weights. To solve this problem, we will analyze and compare different evaluation methods (F-score, DICE Coefficient, Jaccard, Sensitivity, Forbes coefficient) and loss functions (such as Cross entropy (CE), Weighted cross entropy (CE), Tversky loss, Generalized Dice loss) that have been employed in the biomedical domain. At the end, we will choose the metric and loss function that yields the best performance.

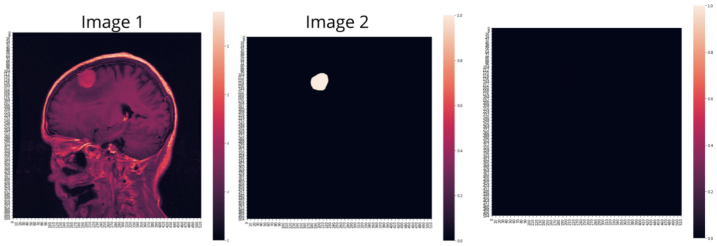


Figure 1: Shows an example of imbalanced classes.

2.2 Memory Footprint

Medical imaging datasets, which require high resolution 3D images are much more memory intensive compared to other DL applications. 3D data, coupled with large 3D models, leads to a large memory footprint. This, in turn, means that we can only fit a certain amount of data in our GPU memory, which becomes a major bottleneck in training. This bottleneck requires careful training that is not too slow and can be completed with minimal computing resources. To ameliorate this issue, we will analyze different strategies such as effective batch training, model parallelism and distributed learning.

2.3 Small Datasets

Since it is difficult to procure and anonymize healthcare data, one of the main problems in the biomedical domain is the lack of large datasets. This means that techniques like data augmentation become more important. However, data augmentation for medical images is not straightforward since we cannot randomly mirror, rotate or perturb the data (human anatomy is not inherently invariant under these transformations). One of our goals of this project will be to find and experiment with suitable data augmentation techniques that are compatible with biomedical images.

3 Methods

3.1 Datasets

In this work, we proposed to segment 9 organs-at-risk that are of pivotal importance in radiation therapy for head and neck cancer. The 9 organs are: *brain stem, chiasm, mandible, optic nerve left, optic nerve right, parotid left, parotid right, submandibular left and submandibular right*. Since dataset sizes are usually quite limited due to the extensive annotation effort involved to segment the 3D scans, we used multiple datasets in this work:

1. MICCAI Head and Neck Auto Segmentation Challenge 2015 (Raudaschl et al. (2017))
2. Head-Neck Cetuximab (The Cancer Imaging Archive) (Clark et al. (2013))
3. Dataset from (Vallieres et al. (2017))

Note that the datasets above consist of CT scans since they are the preferred scan type for determining organs-at-risk segmentation in head and neck cancer.

3.2 Baseline Architecture

As a baseline architecture, we used 3D-Unet (Çiçek et al. (2016)), which has been shown to deliver promising results on several medical segmentation tasks. The original 3D-Unet paper uses concatenation to combine feature maps from the synthesis path with the upsampled feature maps from the synthesis path.

3.3 Approach

For each of the three challenges identified in the Introduction section, we try out multiple approaches and experiments. Specifically, we conduct the following experiments: 1) Imbalanced Classes (Loss functions) 2) Memory Footprint (Effective Batch Size, Data Parallelism, Effect of Input Size and Effect of Model Size) and 3) Small datasets (Data Augmentations, Self-Supervised Pre-training).

3.4 Implementation Details: Programming Framework

We used Python and PyTorch for the project. For the medical imaging specific data loaders and utilities, we used the MONAI and TorchIO packages. Furthermore, we leveraged open source implementations of 3D-Unet (Wolny) and medical imaging specific loss functions (JunMa11) to aid in faster experimentation. We leveraged the PyTorch Lightning library for distributed training. Lastly, we used Tensorboard Visualization Toolkit to visualize the results of different experimental runs.

3.5 Implementation Details: Hardware Requirement

For the hardware, we used a combination of NYU HPC and Google Cloud Platform (GCP). We were limited to the use of Nvidia V100 alone because the Unet model needed roughly 30 GB of memory for batch size of 1 3D CT scan.

4 Results

4.1 Imbalanced Classes

4.1.1 Loss Functions

One of the main strategies used to mitigate the problem of class imbalance is using appropriate loss functions that are robust to class imbalance. Different loss functions and their categories are discussed in the subsections below.

Cross Entropy: One of most frequently used loss function in deep learning is cross entropy. However, it is not optimal for class balance since it depends on optimizing pixel-wise errors which means that larger objects in the image get more representation. This, in turn, implies that smaller objects such as small organs will be given less weight leading to their poor segmentation results. The formula for categorical cross entropy loss is as follows:

$$L_{CCE} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c} * log(p_{i,c})$$

$y_{i,c}$ are hot-encoded labes, $p_{i,c}$ is a matrix of predictions, c and i repsentation class and pixel indices.

Focal Entropy: One of ways to improve cross entropy loss is reduce the weightage of easy examples (background) so that it can focus on hard examples (small orgrans). The formula for focal loss is as follow:

$$L_{CF} = \alpha(1 - (p_{t,c}))^{\gamma} * L_{CCE}$$

where γ represents a weight vector for each class, $p_{t,c}$ represents label probabilities for every class, and L_{CCE} is given in the equation above. The equation shows that the background class will have a smaller coefficient for L_{CCE} allowing us to focus on harder examples.

Dice Loss & its variants: For Dice loss, firstly, we define Dice Coefficient Similarity (DSC), a measure that we want to maximize to get the best results:

$$DSC = \frac{2TP}{2TP + FP + FN}$$

Where TP, FP, FN are True Positive, False Positives and False Negative respectively. It is a measure of how much or predictions overlaps with the true labels. Using DSC, we can simply define dice loss function as:

$$L_{DSC} = \sum_{c=1}^{C}(1 - DSC)$$

Where C is the number of classes. It must be noted that this loss function is effective because it only considers segmentation classes as opposed to the background class. In other words, it cannot overshadow small segmentation classes such as small organs.

Apart from vanilla dice loss, we experimented with its variants: Generalized Dice Loss and Masked Dice Loss. The results of all the loss functions are shown in Table 1.

Tversky Loss:We define Tversky Index which is a generalization of dice coefficient as:

Loss Function	Mean Dice Score
Dice CE Loss	**0.737**
Dice Loss	0.701
Generalized Dice Loss	0.250
Masked Dice Loss	0.680
Dice Focal Loss	0.695
Focal Loss	0.632
Tversky Loss	0.634

Table 1: Mean Dice Scores for different loss functions used in the experiments.

$$TI = \frac{TP}{TP + \alpha FN + \beta FP}$$

In this case, we change the values of α and β to tweak the weightage of FNs and FPs. For Tversky coefficient, α+β=1. Note when α=β=0.5, Tversky index is equivalent to dice coefficient. For our experiments, we used the value of α=0.7 and β=0.3 so that we can punish FNs more. Using TI, we define Tverskly Loss as follows:

$$L_{Tversky} = \sum_{c=1}^{C}(1 - TI)$$

Loss Functions Results: The results for all the experiments are shown in the Table 1 and Figure **??** below.

Dice CE (Cross Entropy) Loss is made up of a linear combination of Dice and CE loss with appropriately chosen weight coefficients. Similarly, Dice Focal Loss is a linear combination of Dice and Focal Loss. The results above show that the best results were obtained for *Dice CE Loss* since this loss function was able to focus on both large objects via cross entropy and small objects via Dice loss.

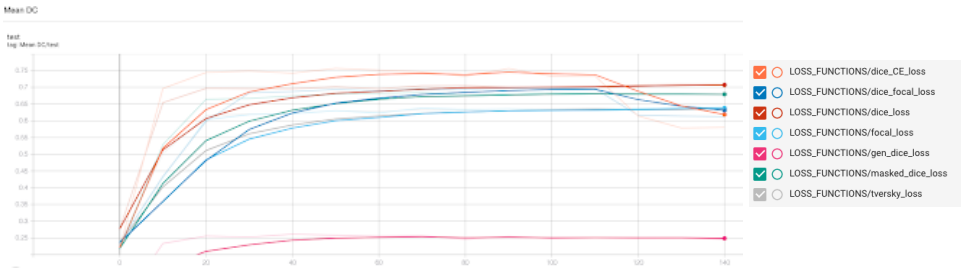
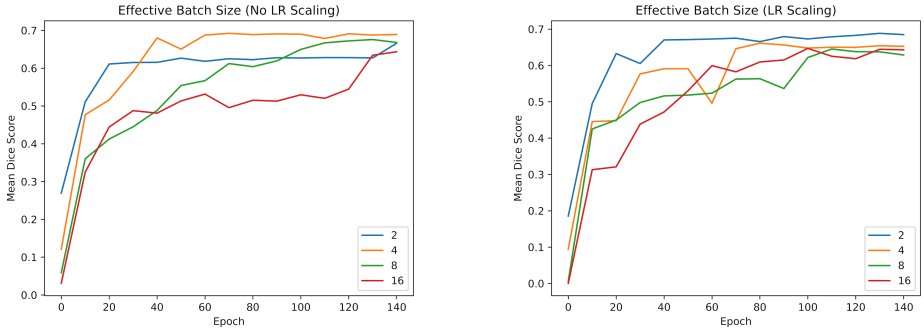


Figure 2: Shows Mean Dice Score for each of the loss function

4.2 Memory Footprint

4.2.1 Effective Batch Size

The concept of effective batch sizes, whereby we accumulate the gradients for a few examples before updating the parameters, was experimented with to deal with the issue of memory footprint. Since the gradients acquired through multiple examples yield a better gradient estimate, this could allow us to scale the learning rate as we increase the effective batch size, resulting in faster convergence. However, in the experiments we conducted (Figure 3), we observed that in general for larger batch sizes of 4-16, it was preferable to not use learning rate scaling. This is probably because the base learning rate for the chosen optimizer is already high and increasing the rate further has the effect of skipping minima during training. The best performance was obtained with an effective batch size of 4 with no learning rate scaling.



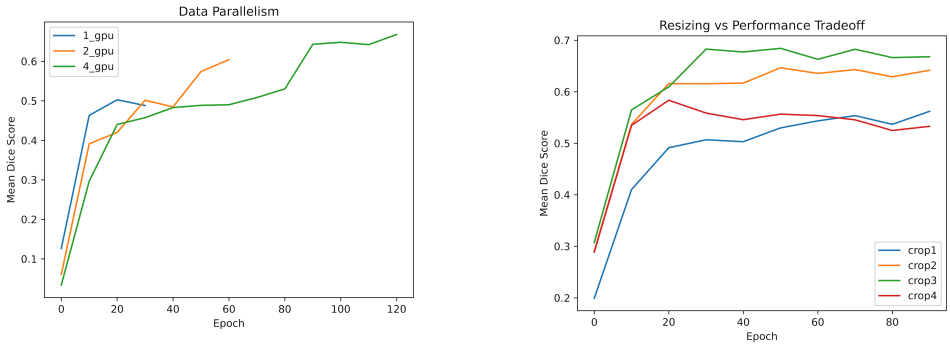
(a) Without learning rate scaling

(b) With learning rate scaling

Figure 3: Effect of effective batch size (2,4,8,16) on Dice Score.

4.2.2 Data Parallelism

Another approach that we tried to deal with the memory footprint issue was to use data parallelism, which simulates using larger batch sizes by dividing the data across multiple GPUs. Based on the findings of the previous task (effective batch size), the same learning rate was used in all experiments with 1, 2 and 4 GPUs. However, the dataset we are working with is peculiar in the sense that all the CT scans are not of the same size. Hence, we needed to resize the CT scans in the dataset to 78*206*164, which is the average size of CT scans in the training set. In order to ensure a fair comparison between the different experiments, we compared the performance of the 3 setups after the same number of iterations. As illustrated in the Figure 4a, data parallelism with 4 GPUs yielded the best results. It is important to note that these results are not directly comparable with those of Figure 3 since the scans in this experiment were resized whereas the scans in the effective minibatch experiment used the original size.



(a) Effect of data parallelism (1,2,4) GPUs on Dice Score.

(b) Effect of input size of CT Scans on Dice Score.

Figure 4: Effect of Data Parallelism and Input Size on Memory Footprint

4.2.3 Input Size

Yet another strategy that we tried to deal with the memory footprint issue was to resize the input CT scan. With the original CT scan and a Unet network, we can only fit 1 example on a Tesla V100 with 32 GB memory. Hence, resizing the input to a smaller size has the benefit that the same experiments can be performed on other hardware with lower memory capacities. Furthermore, since for this particular dataset, each CT scan has a different size, in order to use batch sizes greater than 1 on the same GPU, it is imperative to resize the CT scans to the same size. Four experiments were performed in this case. In each experiment, the CT scan was resized to different sizes. *Crop1* was the largest size followed by *crop2*, *crop3* and *crop4*. Interestingly, we observed in Figure 4b

that decreasing the CT scan size from *crop1* down to *crop3* improved the performance. This was possibly the case because the network was not deep enough to effectively learn from the large CT scan. More on this on the next slide. However, reducing the CT scan size further significantly reduces the performance since the network is no longer able to distinguish the different structures in the head and neck CT scan. Overall, resizing to a size of *crop3* yielded the best results for this choice of dataset and architecture.

4.2.4 Model Size

As we observed in the previous set of experiments, the model was not deep enough for effective learning from the large CT scans. This idea was also reinforced by an analysis of the dice scores of the different organs. In almost all experiments, at least one organ had a dice score of 0, meaning that the network was unable to learn about that organ at all. This can be seen from the graphs in the Figure 5, where some classes always have a dice score of 0 throughout the experiments. This suggested that the network perhaps did not have enough capacity to effectively learn about all the organs.

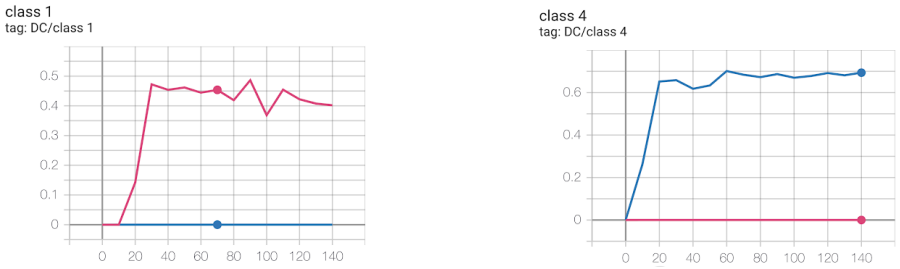


Figure 5: Analysis of the dice scores of the different organs. In almost all experiments, at least one organ had a dice score of 0.

In order to confirm this, we performed an experiment with different model sizes in order to observe the effect of model size on performance. Three models were used. Firstly, a relatively *shallow SegResNet* model with 1.19 M parameters. Secondly, the *same UNet* architecture that was used in the previous experiments and thirdly, a *deeper Unet* with roughly 4 times the number of parameters as the Unet used in the previous experiments. With the deeper Unet, each organ had a non-zero dice score, confirming our hypothesis that the lower network capacity was indeed the reason for the 0 dice score for one or more organs in the previous experiments. The results, as shown in the Figure 6, obtained with the deeper Unet model were also comparable to those reported in the literature. Note, however, that the goal was not to reproduce the results from the literature since those works use specific tricks like custom loss functions or custom augmentations.

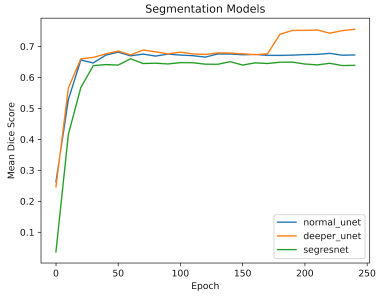
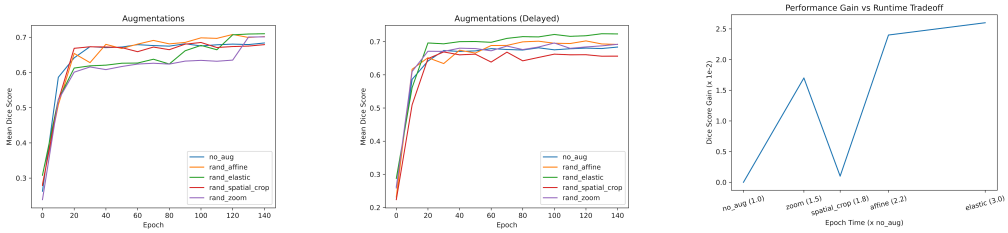


Figure 6: Effect of model size on Dice Score.

4.3 Small Datasets

4.3.1 Data Augmentations

As outlined in the introduction, another challenge associated with medical imaging is small datasets since it is relatively resource intensive to carefully and manually annotate 3D volumes such as CT scans or MRIs. Hence, we aimed to address that using data augmentations. We used 4 different spatial transforms to augment the relatively small dataset of 261 training and 10 test CT scans. Note that *random affine* applies affine transformations such as *rotation*, *translation* and *scaling*. *Elastic deformation*, on the other hand, applies non affine transformations that can squeeze and stretch different structures as can be seen from the black regions in the scan. Also, the outline of the CT scan is more elongated compared to the rounded version in the original scan. *Random crop* crops a region of the input scan and then resizes it back to the original dimensions. Lastly, *random zoom* either zooms in or out of the scan. Together, these augmentations allow us to simulate the effect of larger dataset size by augmenting the dataset with these transformed CT scans.



(a) Augmentations (b) Delayed Augmentations (c) Performance vs. Runtime

Figure 7: Data Augmentations

As seen in the Figure 7, random elastic performs best but also takes 3 times more epoch training time; random affine might be a good compromise between training time and performance gain.

4.3.2 Pretraining

To solve the issue of small datasets, we alse leverage self-supervised pretraining where the labels are generated from the input data. We used pretrained model from Models Genesis[1]. As seen in the Figure 8, the gains from self-supervised pretraining are appreciable for the first 60 epochs.

One potential issue with this approach is that the scans need to resized to ensure compatibility with model used to pretrain the Models Genesis network, for which the input sizes need to be divisible by 16 since the network involves 4 downsampling layers.

5 Conclusion

In this work, effective batch size and data parallelism were used to deal with the large memory footprint issues. Augmentations and self supervised pretraining were used to address small datasets issue. Experimenting with various loss functions enabled us to address class imbalance issue. With these insights, we hope that the practitioners working in medical imaging will be more effective in their work and will get a head start into dealing with these prevalent challenges.

References

Özgün Çiçek, Ahmed Abdulkadir, Soeren S Lienkamp, Thomas Brox, and Olaf Ronneberger. 3d u-net: learning dense volumetric segmentation from sparse annotation. In *International conference on medical image computing and computer-assisted intervention*, pp. 424–432. Springer, 2016.

Kenneth Clark, Bruce Vendt, Kirk Smith, John Freymann, Justin Kirby, Paul Koppel, Stephen Moore, Stanley Phillips, David Maffitt, Michael Pringle, et al. The cancer imaging archive (tcia):

[1] https://github.com/MrGiovanni/ModelsGenesis

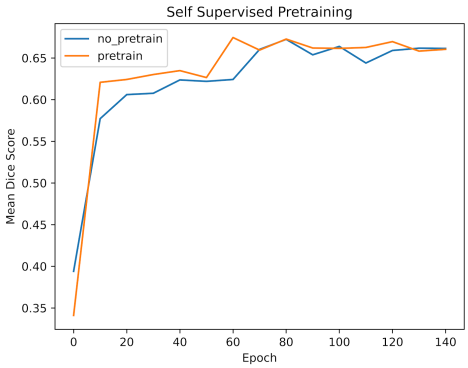


Figure 8: Self supervised Pretraining

maintaining and operating a public information repository. *Journal of digital imaging*, 26(6): 1045–1057, 2013.

Olivier Commowick, Vincent Grégoire, and Grégoire Malandain. Atlas-based delineation of lymph node levels in head and neck computed tomography images. *Radiotherapy and Oncology*, 87(2): 281–289, 2008.

Karl D Fritscher, Marta Peroni, Paolo Zaffino, Maria Francesca Spadea, Rainer Schubert, and Gregory Sharp. Automatic segmentation of head and neck ct images for radiotherapy treatment planning using multiple atlases, statistical appearance models, and geodesic active contours. *Medical physics*, 41(5):051910, 2014.

Xiao Han, Mischa S Hoogeman, Peter C Levendag, Lyndon S Hibbard, David N Teguh, Peter Voet, Andrew C Cowen, and Theresa K Wolf. Atlas-based auto-segmentation of head and neck ct images. In *International Conference on Medical Image Computing and Computer-assisted Intervention*, pp. 434–441. Springer, 2008.

JunMa11. Junma11/segloss. URL https://github.com/JunMa11/SegLoss.

Gargi V Pednekar, Jayaram K Udupa, David J McLaughlin, Xingyu Wu, Yubing Tong, Charles B Simone II, Joseph Camaratta, and Drew A Torigian. Image quality and segmentation. In *Medical Imaging 2018: Image-Guided Procedures, Robotic Interventions, and Modeling*, volume 10576, pp. 105762N. International Society for Optics and Photonics, 2018.

Patrik F Raudaschl, Paolo Zaffino, Gregory C Sharp, Maria Francesca Spadea, Antong Chen, Benoit M Dawant, Thomas Albrecht, Tobias Gass, Christoph Langguth, Marcel Lüthi, et al. Evaluation of segmentation methods on head and neck ct: auto-segmentation challenge 2015. *Medical physics*, 44(5):2020–2036, 2017.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer, 2015.

Rebecca Siegel, Jiemin Ma, Zhaohui Zou, and Ahmedin Jemal. Cancer statistics, 2014. *CA: a cancer journal for clinicians*, 64(1):9–29, 2014.

Martin Vallieres, Emily Kay-Rivest, Léo Jean Perrin, Xavier Liem, Christophe Furstoss, Hugo JWL Aerts, Nader Khaouam, Phuc Felix Nguyen-Tan, Chang-Shu Wang, Khalil Sultanem, et al. Radiomics strategies for risk assessment of tumour failure in head-and-neck cancer. *Scientific reports*, 7(1):1–14, 2017.

Christian Wachinger, Karl Fritscher, Greg Sharp, and Polina Golland. Contour-driven atlas-based segmentation. *IEEE transactions on medical imaging*, 34(12):2492–2505, 2015.

Zhensong Wang, Lifang Wei, Li Wang, Yaozong Gao, Wufan Chen, and Dinggang Shen. Hierarchical vertex regression-based segmentation of head and neck ct images for radiotherapy planning. *IEEE Transactions on Image Processing*, 27(2):923–937, 2017.

Wolny. wolny/pytorch-3dunet. URL https://github.com/wolny/pytorch-3dunet.

Tiezhi Zhang, Yuwei Chi, Elisa Meldolesi, and Di Yan. Automatic delineation of on-line head-and-neck computed tomography images: toward on-line adaptive radiotherapy. *International Journal of Radiation Oncology* Biology* Physics*, 68(2):522–530, 2007.