Kidney Semantic Segmentation

William Han
University of California, Irvine

May 14, 2022

1 Methods

The data used in this project consists of kidney tumor CT exams extracted from the Kidney Tumor Segmentation Challenge (KiTS). 321 images and its respective mask were allocated to the training set, while 81 were used for validation. I created a 2D U-Net, 3D U-Net, and custom, Squeeze and Excite U-Net architecture to accomplish the segmentation task. First, the 2D U-Net has 56 layers, which is including the input, normalization, regularization, concatenation, transpose layers. It also has 230,498 trainable and 1,072 non-trainable parameters. Although the images passed into the 2D U-Net are 2 dimensional, I used 3D convolutional operations due to the fact that the full matrix was still a 3D tensor, where in (z, y, x), z is equal to 1. Second, the 3D U-Net has 57 layers, 689,140 trainable parameters and 1,072 non-trainable parameters. In this network, images of size 96 x 96 x 96 were passed, therefore 3D convolutional operations were also used. Lastly, in the custom, Squeeze and Excite U-Net, there were 76 layers, 234,580 trainable parameters, and 1,072 non-trainable parameters. Squeeze and Excitation operations were applied after every contracting layer. The batch sizes for the 2D U-Net and Squeeze and Excite U-Net were 16. Initially, the batch size for the 3D U-Net architecture was 2, but I decided to change it to 16 for consistency. I trained each model for 10 epochs and all three converged during this set iteration. The optimizer used was Adam and the learning rate stayed at 2e-4 throughout. A sparse categorical cross entropy loss function was used during training and for interpretation, the dice score metric was applied. I planned to evaluate the model by collecting the mean, median,

25th percentile, and 75th percentile for the test sets during training and validation.

2 Results

I evaluated the model by creating a NumPy array of dice scores for the test set during training and validation. I distributed these dice scores into a Pandas data frame. With these dice scores and through Pandas operations, I found the 25th and 75th percentile, as well as the mean and median. It should be known that these scores were rounded to 3 decimal points. These statistical measures gave me an accurate performance measure of each of my models. The highest dice score for each model in its respective statistical measure category is denoted in bold.

Models	Mean	Median	25th percentile	75th percentile
2D	0.954	0.964	0.950	0.973
3D	**0.964**	**0.970**	**0.961**	**0.975**
Custom	0.950	0.964	0.946	0.971

Table 1: Statistics for Training Test Set

Models	Mean	Median	25th percentile	75th percentile
2D	0.923	**0.956**	0.912	**0.969**
3D	**0.933**	**0.956**	**0.936**	0.967
Custom	0.931	**0.956**	0.926	**0.969**

Table 2: Statistics for Validation Test Set

Note that during validation, the median for all three models were synonymous. Additionally, the 75th percentile for the 2D U-Net model and the custom U-Net model were the same. Overall, it seems that that 3D U-Net model generally outperforms the others.

3 Discussion

The results were not expected because I hypothesized that the custom model would be best in performance. The biggest difference between the three

different algorithms is the input dimensionality. The Squeeze and Excite model was given 2 dimensional data (1 x 96 x 96), therefore we cannot say how well it would perform on 96 x 96 x 96 images. I chose the Squeeze and Excite model due to its scalability in terms of feature maps