# Cascaded Semantic Segmentation for Kidney and Tumor

Xiaoshuai Hou[1] and Chunmei Xie[1] and Fengyi Li[1] and Yang Nan[1]

[1] PingAn Technology Co., Ltd, Shanghai, CHINA

**Abstract.** Automated detection and segmentation of kidney tumors from 3D CT images is very useful for doctors to make diagnosis and treatment plan. In this paper, we described a multi-stage semantic segmentation pipeline for kidney and tumor segmentation from 3D CT images based on 3D U-Net architecture. The current method can achieve 0.9XX, 0.8XX average dice for kidney and tumor in the KiTS19 challenge.

**Keywords:** Automated detection and segmentation, CT, Semantic, U-Net.

## 1 Introduction

There are more than 400,000 new cases of kidney cancer each year, and surgery is its most common treatment. Due to the wide variety in kidney and kidney tumor morphology, there is currently great interest in how tumor morphology relates to surgical outcomes, as well as in developing advanced surgical planning techniques. Automatic semantic segmentation is a promising tool for these efforts, but morphological heterogeneity makes it a difficult problem.

The goal of KiTS19 challenge [1] is to accelerate the development of reliable kidney and kidney tumor semantic segmentation methodologies. The challenge organizers have produced ground truth semantic segmentations for arterial phase abdominal CT scans of 300 unique kidney cancer patients who underwent partial or radical nephrectomy at our institution. 210 of these have been released for model training and validation, and the remaining 90 will be held out for objective model evaluation.

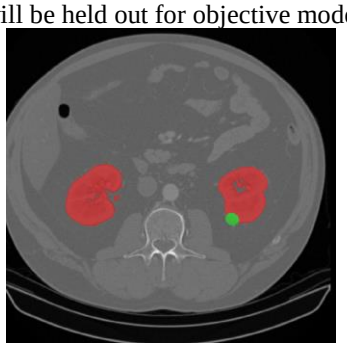

**Fig. 1.** An example of 2D axial slice of 3D CT images. The kidney class is shown in red and the tumor is shown in green.

Automated detection and segmentation of 3D kidney tumors can help doctors quickly locate the tumors and provide accurate reproducible results for further quantification analysis. Semantic segmentation CNNS with encoder-decoder architecture have been widely used for multimodal brain tumor segmentation challenge, liver tumor segmentation challenge, etc. In the work, motivated by the nnUNet [2], we propose a three-stage neural network to locate and segment the kidney and tumor from 3D volumetric CT images. We describe our pipeline in the following section.

## 2 Methods

Our three-stage semantic segmentation pipeline consists of three steps, firstly get the coarse location of kidney and tumor based on a lightweight low-resolution 3D U-Net from 3D CT images with low resolution, and then crop the left/right VOI and get the accurate kidney and tumor location based on a high-resolution 3D U-Net with the same architecture(treat kidney and tumor as only one class), and thirdly classify the kidney and tumor region based on third segmentation model, and finally adopt some post-processing method to fill the holes inside the tumor and remove some false positives. All the models are trained from scratch with 5-fold cross-validation.

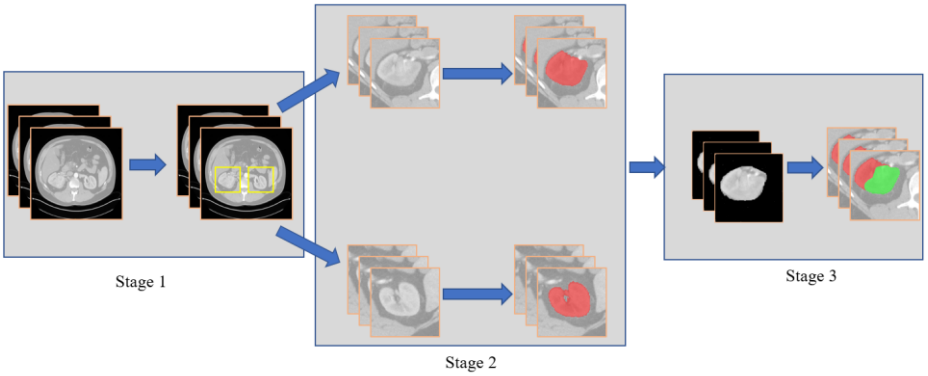


**Fig. 2.** The architecture of our three-stage segmentation pipeline.

### 2.1 Stage 1 and Stage 2

The stage 1 and stage 2 is based on the nnUNet, nnUNet supports 2D, 3D, 3D Cascade mode and we use the 3D cascade architecture as our stage 1 and stage 2 after some modifications. The first stage preprocesses the training 3D CT images to spacing 1.71548519x1.71548519x3.41427984 through down sampling and train the low-resolution U-Net model with a patch size of 128x128x96. The second stage preprocesses 3D CT images to spacing 0.781625x0.781625x0.781625 through up sampling and crops the VOI of kidney regions as the training dataset and train the high-resolution U-Net model with a patch size of 192x160x56.

### 2.2 Stage 3 – Tumor 3D U-Net

The stage 3 is used for segmenting the tumor foreground from kidney background, its encoder and decoder structure are as follows. We set the voxels intensities outside the kidney regions to zero during training procedure.

**Table 1.** Encoder structure, where IN stands for instance normalization, Conv-3x3x3 convolution with a stride size of 1x1x1, Conv stride 2x2x1 with a stride size of 2x2x2.

| Name | Ops | Output size |
|---|---|---|
| Input | | 1x160x160x56 |
| InitConv | Conv, IN, LeakyReLU | 30x160x160x56 |
| EncoderBlock0 | Conv, IN, LeakyReLU | |
| EncoderDown1 | Conv stride 2x2x1, IN, LeakyReLU | 60*80x80x56 |
| EncoderBlock1 | Conv, IN, LeakyReLU | |
| EncoderDown2 | Conv stride 2x2x1, IN, LeakyReLU | 120*40x40x56 |
| EncoderBlock2 | Conv, IN, LeakyReLU | |
| EncoderDown3 | Conv stride 2x2x2, IN, LeakyReLU | 240*20x20x28 |
| EncoderBlock3 | Conv, IN, LeakyReLU | |
| EncoderDown4 | Conv stride 2x2x2, IN, LeakyReLU | 320*10x10x14 |
| EncoderBlock4 | Conv, IN, LeakyReLU | |
| EncoderDown5 | Conv stride 2x2x2, IN, LeakyReLU | 320*5x5x7 |
| EncoderBlock5 | Conv, IN, LeakyReLU | |

**Table 2.** Decoder structure, ⊗ stands for feature concatenation of decoder up and skin connection from encoder, where IN stands for instance normalization, Conv-3x3x3 convolution with a stride size of 1x1x1, Conv1-1x1x1 convolution.

| Name | Ops | Output size |
|---|---|---|
| DecoderUp4 | ConvTranspose3d | 320x10x10x14 |
| DecoderBlock4 | ⊗EncoderBlock4, Conv1, IN, LeakyReLU, Conv, IN, LeakyReLU | |
| DecoderUp3 | ConvTranspose3d | 240x20x20x28 |
| DecoderBlock3 | ⊗EncoderBlock2, Conv1, IN, LeakyReLU, Conv, IN, LeakyReLU | |
| DecoderUp2 | ConvTranspose3d | 120x40x40x56 |
| DecoderBlock2 | ⊗EncoderBlock2, Conv1, IN, LeakyReLU, Conv, IN, LeakyReLU | |
| DecoderUp1 | ConvTranspose3d | 60x80x80x56 |
| DecoderBlock1 | ⊗EncoderBlock1, Conv1, IN, LeakyReLU, Conv, IN, LeakyReLU | |
| DecoderUp0 | ConvTranspose3d | 120x160x160x56 |
| DecoderBlock0 | ⊗EncoderBlock0, Conv1, IN, LeakyReLU, Conv, IN, LeakyReLU | |
| DecoderEnd | Conv1, Softmax | 1x160x160x56 |

### 2.3 Data preprocessing and augmentation

Because the intensity distribution of CT scans is very different, we normalize all input images to zero mean and unit std (based on foreground voxels only). The data augmentation methods include elastic deformation, rotation transform, gamma transformation, random cropping, etc.

### 2.4 Loss and Optimization

We train the model with the combination of dice loss and cross entropy loss and use Adam optimizer with initial learning rate of 1e-4. During training, we keep an

exponential moving average of the validation and training losses. Whenever training loss did not improve by at least $5 * 10^3$within the last 30 epochs, the learning rate was reduced by factor 5. The training was terminated automatically if validation loss did not improve by more than $5 * 10^3$within the last 50 epochs.

## 3 Results

We report the preliminary results using the test data provided by KiTS19 challenge. The test dataset contains 90 cases without annotations. We uploaded our segmentation results to the KiTS19 server for evaluation of per class dice. An example of our prediction results is depicted in Fig. 3.

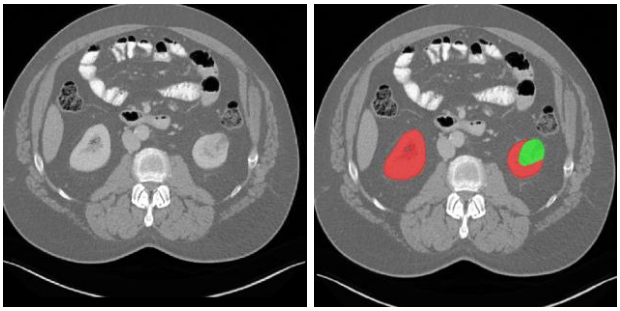

**Fig. 3.** An example of prediction results of case 220. The kidney class is shown in red and the tumor is shown in green.

We implemented out network in PyTorch and trained it on NVIDIA Tesla V100 GPU. Table 1 shows the results of our model on the KiTS19 challenge test dataset.

**Table 3.** Mean dice of the proposed three-stage semantic segmentation pipeline on KiTS19 test dataset.

| Model | Dice | |
|---|---|---|
| | Kidney | Tumor |
| | 0.9xx | 0.8xx |

## 4 Conclusion

We described a three-stage semantic segmentation pipeline for kidney and tumor segmentation from 3D CT images. Preliminary results on KiTS19 challenge test results are 0.9xx, 0.8xx average dice for kidney and tumor respectively.

## References

1. https://kits19.grand-challenge.org/home/
2. Isensee, Fabian, et al. "nnU-Net: Breaking the Spell on Successful Medical Image Segmentation." arXiv preprint arXiv:1904.08128 (2019).