KiTS19 Challenge Segmentation

Yi-Chin Tsai, Yung-Nien Sun

Department of Computer Science and Information Engineering,
National Cheng Kung University, Tainan, Taiwan

Abstract. 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 this challenge is to accelerate the development of reliable kidney and kidney tumor semantic segmentation methodologies. We 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. Recently, fully convolutional neural networks (FCNs), including 2D and 3D FCNs, serve as the backbone in many volumetric image segmentation. However, 2D convolutions cannot fully leverage the spatial information along the third dimension while 3D convolutions suffer from high computational cost and GPU memory consumption. Our method consists of 2.5D convolutions for efficiently extracting g intra-slice and inter-slice features and densely supervised, data augmentation for generate better segmentation.

Keywords: CT, Kidney Tumor Segmentation, Deep Learning.

1 Method

Since the KiTS19 dataset is a set of complete CT scan volumes. Each case may include non-kidney region such as the chest or the foot. Therefore, first, we train a ResUNet to get a coarse segmentation of the kidney with the region of interest (ROI) in the volume. The extracted kidney then into DenseUNet for kidney and tumor segmentation. Finally, post-processing the results.

1.1 ResUNet

The model we use both long-range UNet and short-range ResNet skip connections (residue connections), as shown in Fig. 1. The residual connections help promote information propagation both forward and backward through the network, and improve model convergence and performance. But we design the model to work in 2.5D instead of 3D. In addition, because the purpose of this phase is to find the ROI position of the kidney in the volume. Therefore, we merge kidney and tumor of the target to one class.

The model input is a stack 5 slice of adjacent axial, providing large image content in the axial plane and extra contextual information in the orthogonal direction. The model output is a segmentation map corresponding to the center slice of the stack. In addition to larger input size, more layers and a much larger number of feature channels can be used in each layer than a 3D model. All convolutional layers use a filter size of 3×3 and use the Parametric Rectified Linear Unit (PReLU) as the nonlinear activation function. The spatial size and the number of channels of the output feature maps of each convolutional layer are shown in Fig. 1.

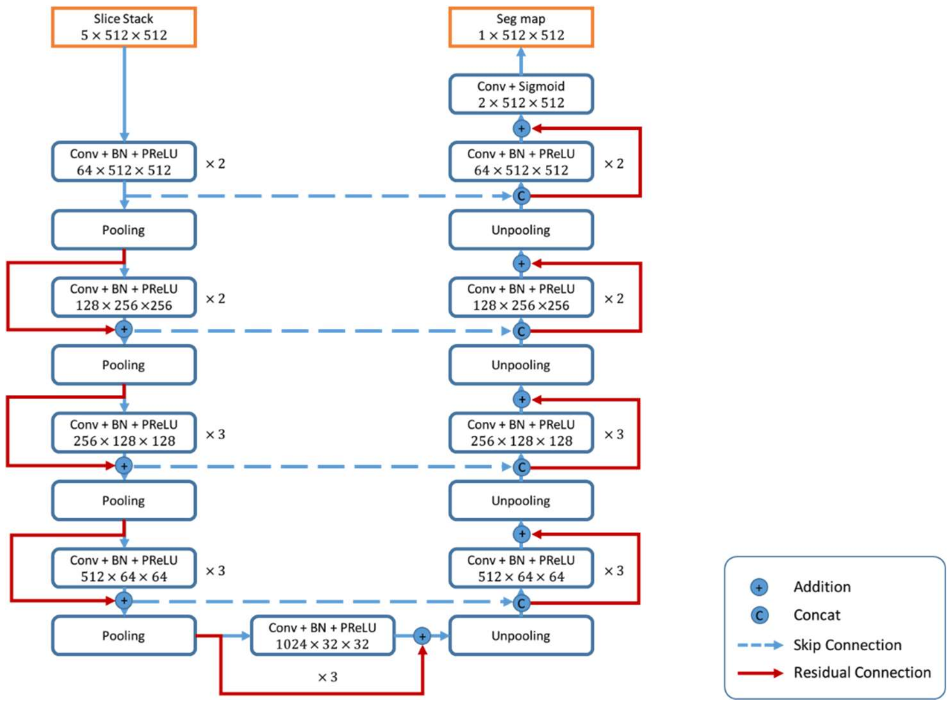

Fig. 1. ResUNet architecture for kidney coarse segmentation

1.2 DenseUNet

The DenseUNet follows the structure of DenseNet-161, which is composed of repetitive densely connected building blocks with different output dimensions. In each densely connected building block, there are direct connections from any layer to all subsequent layers. One advantage of the dense connectivity between layers is that it has fewer output dimensions than traditional networks, avoiding learning redundant features. Moreover, the densely connected path ensures the maximum information flow between layers, which improves the gradient flow, and thus alleviates the burden in searching for the optimal solution in a very deep neural network.

However, the original DenseNet-161 is designed for the object classification task while our problem belongs to the segmentation topics. Moreover, a deep FCN network for segmentation tasks actually contains several max-pooling and upsampling operations, which may lead to the information loss of low-level (i.e., high resolution) features. Given above two considerations, we develop a DenseUNet. The detailed structure is shown in the Table 1, which inherits both advantages of densely connected path and UNet-like connections. Specifically, the dense connection between layers is employed within each micro-block to ensure the maximum information flow while the UNet long range connection links the encoding part and the decoding part to preserve low-level information.

We also added densely supervised in DenseUNet. Specifically, each of the upsampling feature maps channel will be converted to 3 by 1×1 conv, and calculated loss with the reduced size of target image. The advantage of this is that each upsampling will be as similar to the target as possible.

Table 1. Architectures of the DenseUNet. The symbol k means kernel size, s means stride, p means padding and ch means output channels. "[ ] $\times\ d$" means this block is repeated for d times. Note that every "conv" is contain "Batch Normalization + ReLU + Conv" and upsampling is bilinear interpolation.

	Feature map ($h \times w$)	DenseUNet
input	512×512	-
convolution 1	256×256	$\text{conv}\ (\text{k}=7, \text{s}=2, \text{p}=3, \text{ch}=96)$
pooling	128×128	$\text{max pool}\ (\text{k}=3, \text{s}=2, \text{p}=1)$
dense block 1	128×128	$\begin{bmatrix} \text{conv}\ (\text{k}=1, \text{ch}=192) \\ \text{conv}\ (\text{k}=3, \text{p}=1, \text{ch}=48) \end{bmatrix} \times 6$
transition layer 1	64×64	$\text{conv}\ (\text{k}=1) + \text{average pool}\ (\text{k}=2)$
dense block 2	64×64	$\begin{bmatrix} \text{conv}\ (\text{k}=1, \text{ch}=192) \\ \text{conv}\ (\text{k}=3, \text{p}=1, \text{ch}=48) \end{bmatrix} \times 12$
transition layer 2	32×32	$\text{conv}\ (\text{k}=1) + \text{average pool}\ (\text{k}=2)$
dense block 3	32×32	$\begin{bmatrix} \text{conv}\ (\text{k}=1, \text{ch}=192) \\ \text{conv}\ (\text{k}=3, \text{p}=1, \text{ch}=48) \end{bmatrix} \times 36$
transition layer 3	16×16	$\text{conv}\ (\text{k}=1) + \text{average pool}\ (\text{k}=2)$
dense block 4	16×16	$\begin{bmatrix} \text{conv}\ (\text{k}=1, \text{ch}=192) \\ \text{conv}\ (\text{k}=3, \text{p}=1, \text{ch}=48) \end{bmatrix} \times 24$
upsampling layer 1	32×32	$\text{upsampling}\ (\text{k}=2)$ $+\ \text{skip connection}\ (\text{dense block 3})$ $+\text{conv}\ (\text{k}=3, \text{p}=1, \text{ch}=768)$
upsampling layer 2	64×64	$\text{upsampling}\ (\text{kernel}=2)$ $+\ \text{skip connection}\ (\text{dense block 2})$ $+\text{conv}\ (\text{k}=3, \text{p}=1, \text{ch}=384)$
upsampling layer 3	128×128	$\text{upsampling}\ (\text{k}=2)$ $+\ \text{skip connection}\ (\text{dense block 1})$ $+\text{conv}\ (\text{k}=3, \text{p}=1, \text{ch}=96)$
upsampling layer 4	256×256	$\text{upsampling}\ (\text{k}=2)$ $+\ \text{skip connection}\ (\text{convolution 1})$ $+\text{conv}\ (\text{k}=3, \text{p}=1, \text{ch}=96)$
upsampling layer 5	512×512	$\text{upsampling}\ (\text{k}=2)$ $+\text{conv}\ (\text{k}=3, \text{p}=1, \text{ch}=96)$
convolution 2	512×512	$\text{k}=3, \text{p}=1, \text{ch}=3$

1.3 Post-Processing

We use 3D connected components to remove some wrong segmentation result. ， Specifically, it is used 26-connected algorithm. 26-connected pixels are neighbors to every pixel that touches one of their faces, edges, or corners. These pixels are connected along either one, two, or all three of the primary axes. In addition to 18-connected pixels, each pixel with coordinates $(x \pm 1, y \pm 1, z \pm 1), (x \pm 1, y \pm 1, z \mp 1), (x \pm 1, y \mp 1, z \pm 1), (x \mp 1, y \pm 1, z \pm 1)$, is connected to the pixel at(x, y, z).

Since the tumor will connect with the kidney and each person will only have at most two kidneys. Therefore, first, we merge the kidneys and tumor of the segmentation result and ignore the background. Then each case takes only two largest components as the new result. However, the case may have only one kidney. For this reason, we will remove if this component are small than 0.1 times of the largest component,

1.4 Data Augmentation

We use data augmentation to improve the variability and diversity of training data, which contains Horizontal Flip, Random Brightness Contras, Random Gamma, Grid Distortion and Shift Scale Rotate.

2 Implementation Detail

The model is implemented on PyTorch. We employed cross-entropy function for ResUNet and generalized dice loss for DenseUNet as the loss function. The initial learning rate was 0.0001 and multiplied by 0.1 when loss is not decrement in 5 epoch. Each model was trained for 100 epochs.

Both models were trained from scratch using the Adam optimization algorithm. Data augmentation is perform during model training of each sample. The input for ResNet is the original slice image (512×512), and the input for DenseUNet is slice image after ROI crop and padding to 512×512. Batch size is 32. We are using 4 NVIDIA Tesla V100 GPU with 32GB memory of each GPU for training.

Our source code is available at https://github.com/nitsaick/kits19-challenge after the results announcement.