Edge-Aware Network for Kidneys and Kidney Tumor Semantic Segmentation

Andriy Myronenko and Ali Hatamizadeh

NVIDIA, Santa Clara, CA
{amyronenko,ahatamizadeh}@nvidia.com

Abstract. Automated segmentation of kidneys and kidney tumors is an important step in quantifying the tumor’s morphometrical details to monitor the progression of the disease and accurately compare decisions regarding the kidney tumor treatment. Manual delineation techniques are often tedious, error-prone and require expert knowledge for creating unambiguous representation of kidneys and kidney tumors segmentation. In this work, we propose an end-to-end boundary aware fully Convolutional Neural Networks (CNNs) for reliable kidney and kidney tumor semantic segmentation from arterial phase abdominal 3D CT scans. We propose a segmentation network entailing an encoder-decoder that specifically accounts for organ/tumor edge information by devising a dedicated network edge branch and edge-aware loss terms. We have evaluated our model on 2019 MICCAI KiTS Kidney Tumor Segmentation challenge dataset. Based on our own data split, we achieved 0.970 dice for kidney & tumor, and 0.834 dice for tumor segmentation.

1 Introduction

Kidney cancer accounted for nearly 175,000 deaths worldwide in 2018 [1], and it is projected that 14,770 deaths will occur due to the disease in 2019 in the US [8]. Current kidney tumor treatment planning include Radical Nephrectomy (RN) and Partial Nephrectomy (PN). In RN, both the tumor and the affected kidney are removed whereas in PN the tumor is removed but kidneys are saved [9]. Although RNs were historically prevalent as a standard treatment procedure for kidney tumors, new capabilities for earlier detection of the tumors as well as advancements in surgery has made PNs a viable treatment approach [4].

Automated segmentation of kidneys and kidney tumors can help physicians to obtain accurate morphometrical details of the tumor in an efficient and reliable manner as the manual delineation process is often tedious and error-prone. The decision for kidney tumor treatment plan can be made by leveraging such important tumor’s morphometrical information. Recently, deep learning approaches for semantic image segmentation have demonstrated prominent results in medical image analysis for various applications [7,5,6,2]. The powerful non-linear feature extraction capabilities of CNNs along with the effectiveness of the encoder-decoder architectures have made it possible to employ CNNs for challenging segmentation tasks.
Fig. 1. Example of an axial slice of 3D CT images of two patients in KiTS dataset. Red color indicates kidneys, green color indicates tumor region.

Kidney Tumor Segmentation Challenge (KiTS 2019) provides data of multiphase 3D CTs, voxel-wise ground truth labels, and comprehensive clinical outcomes for 300 patients who underwent nephrectomy for kidney tumors between 2010 to 2018 at University of Minnesota [4]. 210 patients were randomly selected for the training set and the remaining 90 patients were left as a testing set. The annotation was performed in the transverse plane with regular subsampling of series in the longitudinal direction with roughly 50 annotated slices depicting the Kidney for each patient. The labels for excluded slices were computed by using a contour interpolation algorithm [4]. Figure 1 illustrates 2D axial view of the example images from two patients in the training set of KiTS 2019.

2 Methods

2.1 CNN Architecture

Our network consists of the main segmentation branch and the additional boundary stream that processes the feature maps at the boundary level (see Figure 2). The main branch follows [6] with asymmetric encoder-decoder structure. The input to the encoder is a 176x176x176 crop which is initially fed into a 3x3x3 convolution with 16 filters. Feature maps are then extracted at each resolution by feeding them into a residual block [3] followed by a strided 3x3x3 convolution (for downsizing and doubling of feature dimension). The bottom of the encoder entails four consecutive residual blocks that are connected to the decoder. The extracted feature maps in the decoder are upsampled using bilinear interpolation and added with feature maps from the encoder. The output of the decoder, along with the output of the boundary stream are fed into the fusion module
(see Figure 3). Finally, the output of the fusion module is fed into a 1x1x1 convolution with 2 channels where channel-wise sigmoid activation \( \sigma(X) = \frac{1}{1+\exp(-X)} \) determines the probability of each voxel belonging to kidneys&tumor or only tumor classes.

### 2.2 Boundary Stream

The purpose of the boundary stream is to highlight the edge information of the feature maps extracted in the main encoder by leveraging an additional attention-driven decoder. The attention-gated layers in every resolution of the boundary stream process the feature maps that are learned in the main encoder as well as the output of the previous attention-gated layer. For the first attention-gated layer, we first concatenate the output of the encoder with its previous resolution and feed it into a residual block. In the attention gated layer, each input is first fed into a 3x3x3 convolutional layer with matching number of feature maps and then fused together, followed by ReLU. The output of the ReLU is fed into a 1x1x1 convolution layer followed by sigmoid function \( \sigma \) to obtain the attention map. Consecutively, an element-wise multiplication between the boundary stream feature maps and the computed attention map results in the output of the attention-gated layer.

### 2.3 Fusion Module

The purpose of the fusion module is to embed the highlighted boundary information in feature maps that are learned in the main stream. Inspired by [10],
we first concatenate the feature maps learned from main stream as well as the boundary stream before feeding it into a 3x3x3 convolutional layer followed by batch normalization and ReLU. The output is then routed to an identity branch as well as an additional branch in which we employ a global average pooling layer followed by two consecutive 1x1x1 convolutional layers with ReLU and sigmoid as the activation functions. Consequently, we perform an element-wise multiplication between the output of each of the branches before and feed it into a final 1x1x1 convolutional layer followed by a channel-wise sigmoid activation function. The fusion module is illustrated in Figure 3.

![Fig. 3. The fusion module to combine the main and the boundary streams.](image-url)

2.4 Loss Functions

Our proposed CNN jointly learns the boundaries and semantics by supervising the output of the main stream as well as the boundary stream. We use a dice loss function on the predicted outputs of the main stream as well as the boundary stream. The dice loss is as follows [5]:

$$L_{Dice} = 1 - \frac{2 \cdot \sum y_{true} \cdot y_{pred}}{\sum y_{true}^2 + \sum y_{pred}^2 + \epsilon}$$  \hspace{1cm} (1)

Where $y_{pred}, y_{true}$ denote the pixel-wise semantic predictions of the main stream and their corresponding labels, $\epsilon$ is a small constant to avoid division by zero and summation is carried over the total number of pixels.

Additionally, we add a weighted Binary Cross Entropy (BCE) loss to the boundary stream loss in order to deal with the imbalanced number of boundary and non-boundary pixels:
\[
L_{BCE} = -\beta \sum_{j \in y_+} \log P(y_{pred,j} = 1|x; \theta) \\
- (1 - \beta) \sum_{j \in y_-} \log P(y_{pred,j} = 0|x; \theta)
\] (2)

Where \(x, \theta, y_-\) and \(y_+\) denote the input image, CNN parameters, edge and non-edge pixel sets respectively. \(\beta\) is the ratio of non-edge pixels over the entire number of pixels and \(P(y_{pred,j})\) denotes the probability of the predicted class at pixel \(j\).

The total loss function that is minimized during training is computed by taking the average of losses for tumor-only and foreground class predictions.

3 Data Processing

We normalized the CT data to [-1, 1] range by dividing the intensity values by 1000 and clipping the values that fall outside this range. For training, images were re-sampled to 1x1x1mm isotropic resolution and re-sampled back to their original resolution after the inference. The re-sampled output size of the images was on average 512x512 in axial plane and 400 – 800 along the inf-sup direction.

4 Implementation Details

We have implemented our method in Pytorch\(^1\). Since the resampled CT image were often large, we used a 176x176x176 crop during training. The cropping region was centered on the kidney tumor label (with probability 0.8), on any foreground (with probability 0.1) and on background (with probability 0.1). We found it important to sample more frequently from the tumor region. The model was trained on 8 NVIDIA Tesla V100 16GB GPUs (DGX-1 server). We used a batch size of 8 and the Adam optimization algorithm with the initial learning rate of \(\alpha_0 = 5e^{-5}\) that was further decreased according to \(\alpha = \alpha_0 \times (1 - e/N_e)^{0.9}\) \(^6\), where \(e\) and \(N_e\) denote the current epoch counter total number of epochs (300 in our case).

For inference we used a sliding window technique with a crop size of 224x224 x224 and an overlapping step size of 128 to avoid the boundary artifacts. We also used test time augmentation (TTA) by mirror flipping the input image across its axes and averaging the output of the resulting segmentation probability maps. Finally, we used an ensemble of 5 models to further improve the results.

5 Preliminary Results

Table 1 shows evaluation results of our model. Figure 4 illustrates a typical segmentation result of our method and a ground truth (from the training set).

\(^1\) https://pytorch.org/
We split the training part of the data into our own subsets for training and validation. We show a single model performance, as well as ensemble of 5 models. We also follow the best practices, with test time augmentation (TTA), where the input the model was flipped (all combinations) and un-flipped and averaged after the inference. Finally the KiTS 2019 submission allowed to view 2 approximate scores (based on a small subset of true validation data), which allowed us to list the approximate scores of the single model and the ensemble. Generally the dice scores were similar and consistent between our own split and the approximate scores (see Table 1).
Table 1. Preliminary dice results based on our own data split as well as 2 approximate scores provided by KiTS 2019 submission portal.

<table>
<thead>
<tr>
<th>Evaluation</th>
<th>Dice</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kidneys&amp;Tumor</td>
<td>Tumor only</td>
</tr>
<tr>
<td>-----------------------------------------</td>
<td>------------</td>
</tr>
<tr>
<td>Our split (single model)</td>
<td>0.957</td>
</tr>
<tr>
<td>Our split (TTA + ensemble)</td>
<td>0.970</td>
</tr>
<tr>
<td>Approximate score (single model)</td>
<td>0.955</td>
</tr>
<tr>
<td>Approximate score (TTA + ensemble)</td>
<td>0.974</td>
</tr>
</tbody>
</table>

6 Conclusion

In this work, we describe an edge-aware segmentation network for automated segmentation of kidneys and kidney tumors. Our approach explicitly accounts for object edge information by using a dedicated boundary stream that processes the feature maps at the boundary level. We have trained and evaluated our model on KiTS 2019 Kidney Tumor Segmentation challenge (as part of MICCAI) and achieved 0.970 dice for kidney & tumor, and 0.834 dice for tumor segmentation based on our own data split.

References
