Arkansas AI-Campus Method for the 2019 Kidney Tumor Segmentation Challenge

Jason L Causey[1], Jonathan Stubblefield[1], Tomonori Yoshino[1], Alejandro Torrico[2], Jake A Qualls[1], and Xiuzhen Huang[1]

[1] Arkansas State University
[2] University of Arkansas

Abstract. Our Arkansas AI-Campus team participants the 2019 Kidney Tumor Segmentation Challenge (KiTS19) during the past 4 months. Here the paper provides a summary of our methods and validation results for this grand challenge in biomedical imaging analysis.

1 Introduction

The 2019 Kidney Tumor Segmentation Challenge provide a good platform for encouraging computational approach development for automatic kidney tumor segmentation with patients CT scans. In this manuscript we provide our method to address the challenging question. Our method is based on neural network models and trained by the dataset provided by the KiTS19 Challenge[1].

2 Materials and Methods

In this section we first describe the dataset we used and then present our model for the biomedical imaging problem based on neural network.

2.1 Dataset

The data were provided by the KiTS19 Challenge organization[1]. There are 300 patients of CT data. 210 patients data were made available to the competition teams, and the remaining 90 patients data are used for testing predictions; no segmentation information was provided for these.

We used the 210 patient CT scans corresponding ground truth provided by the KiTS19 Challenge organizers[1] for our training and validation. Our validation group was set aside before training began by selecting 20% (N=42) of available patients at random. The same validation group was isolated and used for validation on all models. The remaining 168 patients were used as our training group for all models.

2.2 Methods

We investigated several model architectures as possible solutions to this challenge. Primarily, we looked at two high-level configurations: Mask-RCNN[2] and U-Net[3]. We decided on an ensemble of U-Net models as our final configuration after testing many variations. We discuss our experience with Mask-RCNN further in the Discussion section, as well as our rationale for ultimately choosing U-Net.

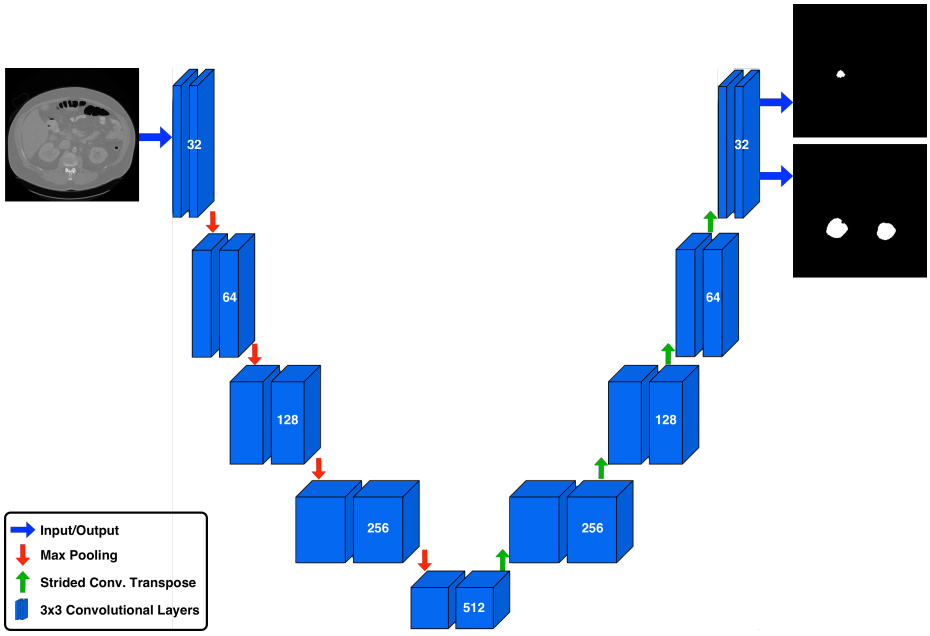

Fig. 1. U-Net architecture.

Our final ensemble consists of two U-Net models working in tandem, followed by a post-processing "cleanup" phase to minimize prediction artifacts. All of our U-Nets share the same structural architecture, as shown in fig. 1. Both models in our ensemble were trained on axial slices, differing in the number of epochs trained and the interpretation of the output masks from each. One model was tasked with predicting the kidney and tumor masks separately in its two output channels. We will refer to this as the "K/T" model. The other model was trained to predict the combined kidney+tumor mask on the first output channel, and the tumor portion on the second output channel. We will refer to this as the "KT/T" model. The output from the two models was combined such that both models voted equally for the inclusion of any individual mask voxel, and voxels receiving a vote from either model were included in the result sent to the post-processing stage.

Finally, we post-processed the proposed mask by (1) filling gaps of width ≤ 2 in the tumor mask along each of the three axes, (2) computing and filling the convex hull of each connected region in the tumor mask, (3) removing any segmentations that occupy only a single "slice" along each of the three axes, (4) retaining only the largest five connected regions in the tumor mask, (5) computing the two largest connected regions that intersect with a kidney segmentation in the union of the tumor and kidney masks and using those two largest connected regions to filter all proposals, removing any proposed segmentation voxels outside these two regions.

This post-processing stage removed spurious predictions as well as filling in any missing interior regions in the tumor prediction.

2.2.1 Pre-processing We found that loading the NiFTi-format files for each patient was a bottleneck in our training process, so we pre-processed the images and saved the pre-processed versions in a format that could be read directly by the Numpy[4] package. For our axial models, we saved each axial "slice" in an individual Numpy file. This allowed us to load slices individually instead of loading an entire CT scan volume, further optimizing our loading times. For training with coronal and saggital views, we saved the entire CT volume for each patient in a single Numpy file. We optimized training on these views such that all possible slices for a single patient were used preferentially before moving to a different patient, so that we could reduce the impact of the longer load times.

Our pre-processing also included a window normalization of the CT image data which thresholded the raw Hounsfield units to the range $[-500, 500]$ and mapped the values to the numeric range $[0, 1]$ according to the formula:

$$v_{out} = \min(\max(v_{in}, -500), 500) \cdot \frac{1}{1000} + 0.5$$

This step must also be performed prior to inference with all our models, so it is part of the input stage for the inference algorithm.

For inference, our algorithm reads the NiFTi file directly; it is not necessary to cache the image in Numpy format at this stage. The window normalization step is required as a pre-processing step during inference.

2.2.2 Training Our training data consisted of 168 cases that included a ground-truth segmentation. For each of our models, we proceeded as follows using the Keras[Keras] deep learning framework in Python with the Tensorflow[Tensorflow] back-end.

1. Starting weights were "seeded" at random and trained for 8 epochs each. We continued this process until we found an initial model that seemed to be converging at a reasonable rate. Many starts did not converge in any

meaningful way within the first 8 epochs, and were discarded. In general, a good starting weights could be found in about 5 attempts.

2. All training for the axial models proceeded by dividing all available axial slices into two sets: "Positive" slices contained at least one segmented voxel of either tumor or kidney, and "Negative" slices contained no segmented voxels. We balanced our training set by randomly choosing enough slices from the positive and negative sets to create a 2:1 ratio of positive slices.
3. Image slices were augmented in the following ways (each augmentation had a 50% chance of being applied to any slice):
   - Randomly flipped vertically (this augmentation was disabled after ~135 epochs)
   - Randomly flipped horizontally
   - Randomly shifted up to 15% in both the vertical and horizontal directions
   - Randomly zoomed in/out up to 15% and re-cropped or padded with zeros to maintain image size (only used on epochs > 150 K/T and > 200 KT/T)
4. Models were trained using ~2000 slices per epoch. Training loss was a weighted cross-entropy loss where tumor segmentation errors were weighted 10x versus kidney segmentation errors. We also monitored a per-slice DICE metric to determine how training was proceeding.
5. After training the models until the training metrics indicated a performance plateau, we ranked the weights by training and validation dice metric, and chose several top ranked checkpoints for further testing. For both axial models, we eventually trained in excess of 250 epochs, but the later checkpoints were not always best.

Selected "best" weights were then used in an ensemble as described prevously; we chose one checkpoint from the K/T and KT/T models for our final ensemble.

We provide detailed instructions for training both our K/T model and the KT/T model in the "README.md" file contained in our source repository on Github (https://github.com/jcausey-astate/ai*campus*kits19). The best K/T weights occurred at epoch 150 and the best KT/T weights occurred at epoch 205.

2.2.3 Implementation details We utilized both local and cloud-based (Amazon Web Services) GPU instances to train our models. Our two local instances included a single NVIDIA Tesla P-40 GPU and a single NVIDIA Tesla V-100 GPU, respectively. We also utilized up to three concurrant AWS cloud instances using the Deep Learning AMI, with one NVIDIA Tesla P-100 on each instance.

3 Testing Results

The table below shows our performance on our local validation group of 42 cases. Shown is the performance for each of the individual models in the ensemble, as

well as the ensemble itself. The performance of the individual models does not include the post-processing steps.

Model	K+T DICE, Std. Dev.	T DICE, Std. Dev.
K/T axial	0.927, 0.096	0.512, 0.293
KT/T axial	0.932, 0.072	0.517, 0.294
ensemble + post-proc.	0.949, 0.053	0.601, 0.292

4 Summary and Discussion

4.1 Medical Relevance

Renal cell carcinomas are aggressive cancers. The most important prognostic factor for patients is the stage of the cancer at the time of its discovery [5]. Most cancers are staged using the TNM method [5]. This segmentation algorithm should be able to provide a T (tumor) stage for the cancer. The segmenting algorithm will naturally be able to provide information about the tumor's size and location. Most kidney cancers are treated with surgery[6]. Radical nephrectomy used to be the standard of care for these cancers, but with recent improvements in technology and surgical techniques, partial nephrectomies are now available [6]. These have superior recovery and quality of life for the patient post-surgery [6]. This algorithm will assist surgeons in deciding whether or not a patient is a candidate for partial nephrectomy by helping decide if a sufficient margin of healthy tissue can be left behind to minimize risk of cancer recurrence [6].

4.2 Other Models

Mask-RCNN
We attempted to adapt Mask-RCNN[2] to the segmentation problem. This model was selected for its state-of-the-art ability to perform segmentation tasks. However, we encountered challenges in adapting this model to the problem of segmenting the kidney and the tumor. Much like the U-Net model, Mask-RCNN is a 2D model and was trained with individual slices of CT scans. However, Mask-RCNN assumes that every training image will contain at least one object of interest. Training errors occur if this is not the case. To work around this problem, we added a "dummy" mask consisting of a single pixel placed in a random position on each slice that did not contain kidney or tumor. The randomness was intended to prevent the model from perfectly learning the dummy mask's position.

After making these modifications, we were able to train Mask-RCNN models for the axial, coronal, and sagittal views. However, its performance was quickly outclassed by the U-Net. We suspect this is because Mask-RCNN has a much higher model capacity (and complexity) and trained more slowly.

Multi-View U-Net
While planning the U-Net ensemble, we planned to train models for all three views: Axial, saggital, and coronal. However, the coronal and saggital models did not reach the same level of performance as the axial model, and ensembles containing these models underperformed ensembles with axial models only. We suspect that the reason was that were using the original images and not the interplated dataset that allowed a common spacing in the Z-axis direction. This led to a much higher variance in the model's experience of anatomical structures for the non-axial views, reflected in the inability to reach adequate performance.

References

[1] N. Heller, N. Sathianathen, A. Kalapara, E. Walczak, K. Moore, H. Kaluzniak, J. Rosenberg, P. Blake, Z. Rengel, M. Oestreich, J. Dean, M. Tradewell, A. Shah, R. Tejpaul, Z. Edgerton, M. Peterson, S. Raza, S. Regmi, N. Papanikolopoulos, and C. Weight, "The KiTS19 Challenge Data: 300 Kidney Tumor Cases with Clinical Context, CT Semantic Segmentations, and Surgical Outcomes," *arXiv:1904.00445 [cs, q-bio, stat]*, Mar. 2019.

[2] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," *arXiv:1703.06870 [cs]*, Mar. 2017.

[3] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2015, pp. 234–241.

[4] S. van der Walt, S. C. Colbert, and G. Varoquaux, "The NumPy Array: A Structure for Efficient Numerical Computation," *Computing in Science Engineering*, vol. 13, no. 2, pp. 22–30, Mar. 2011.

[5] H. T. Cohen and F. J. McGovern, "Renal-Cell Carcinoma," *New England Journal of Medicine*, vol. 353, no. 23, pp. 2477–2490, Dec. 2005.

[6] M. M. Picken, L. Wang, and G. N. Gupta, "Positive Surgical Margins in Renal Cell CarcinomaTranslating Tumor Biology Into Clinical Outcomes," *American Journal of Clinical Pathology*, vol. 143, no. 5, pp. 620–622, May 2015.