Hubmap-Hacking the Kidney, 3rd Place Solution

Shujun He, Sejun Song

This write-up aims to provide a summary of the 3rd place solution in the Hubmap-Hacking the Kidney competition hosted on Kaggle by InnovationDigi. Our team name is 'what's goin on'. Our final public leaderboard score is 0.9187, and our final private leaderboard score is 0.9502.

1 Background

Our team consists of 2 members: Shujun He and Sejun Song. Shujun is a PhD student in the department of chemical engineering at Texas A&M University in College Station, TX, and Sejun Song is a data engineer, in Seoul in the Republic of Korea. Both of us have partipated in various Kaggle competitions before and placed in the medal zone numerous times. Our previous experience with Kaggle competitions definitely helped us succeed in this competition, since we're already familiar with the notebook submission system and competition format. It is difficult to estimate how much time we spent on this competition due to the down time caused by labeling issues, but it's fair to say on average we spent somewhere between 10-20 hours per week.

2 Model Summary

This section will provide a summary of our modeling methods. Our approach is based on Convolutional Neural Networks, and our training/inference pipelines are coded with the pytorch library, while using some utilities from the fastai library. Rasterio is used to read the tiff images one chunk at a time during inference to avoid memory issues in the submission notebook.

2.1 Preprocessing

The code we use for preprocessing was originally released by iafoss at `https://www.kaggle.com/iafoss/256x256-images`, which basically cuts the large tiff images into small square tiles so they can be used in batches during training (Figure 1). Tiles are also filtered based on a saturation threshold of 40. The edges of tiff images are padded with white space to ensure an integer number of tiles. RLE encoded masks are converted to PNGs. Our final submissions use tiles of size 1024x1024 while downsampling the original image by two times. The preprocessed dataset is released as a Kaggle dataset at `https://www.kaggle.com/shujun717/hubmap-3rd-place-processed-dataset`.

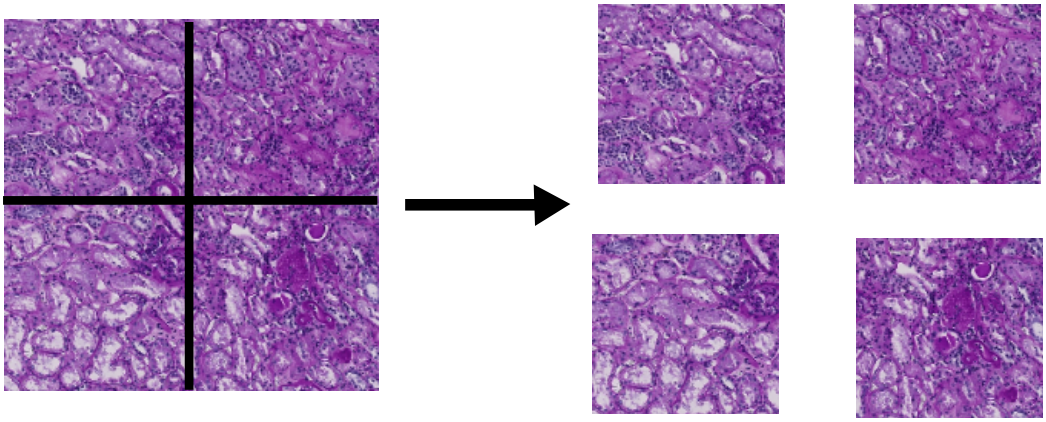

Figure 1: Large tiff images are cut into small square tiles

2.2 Models

Our final submission is an ensemble of 2 sets of 5-fold models using the U-Net[1] architecture with backbones from resnext family (resnext50_32x4d and resnext101_32x4d)[2]. Additionally, Feature Pyramid Network[3] is added to provide skip connections between upscaling blocks of the decoder, atrous spatial pyramid pooling (ASPP)[4] to enlarge receptive fields, and pixel shuffle[5] instead of transposed convolution to avoid artifacts.

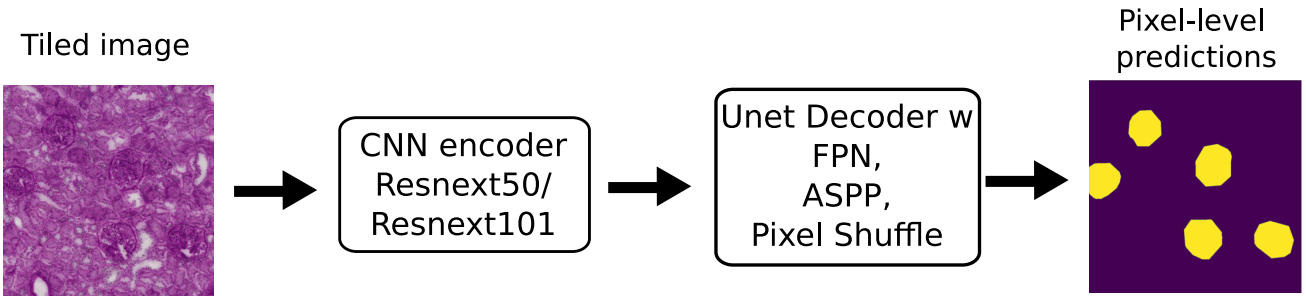

Figure 2: Visualization of architecture used

2.3 Training

We start training from weights pretrained in weakly-supervised fashion on 940 million public images followed by fine-tuning on ImageNet1K dataset [6]. We use Nvidia Apex (`https://github.com/NVIDIA/apex`) to enable mixed precision training, which usually allows for up to 50% reduction in training GPU memory consumption and 50% increase in training speed. One cycle learning rate scheduler[7] is used with pct_start set to 0.2, div_factor 1e2, and max_lr 1e-4. Models are trained for 50 epochs each. Heavy augmentations are used during training which include cutout[8] and numerous augmentations from the albumentations library at `https://github.com/albumentations-team/albumentations`:

- HorizontalFlip
- VerticalFlip
- RandomRotate90
- ShiftScaleRotate
- ElasticTransform
- GaussianBlur
- GaussNoise
- OpticalDistortion
- GridDistortion
- IAAPiecewiseAffine
- HueSaturationValue
- CLAHE
- RandomBrightnessContrast

Loss function used is binary cross entropy loss, and gradient norms are clipped at 1. For each fold, we select and save weights from the epoch with the best soft dice score. Normalization of images is done by subtracting from the mean and dividing by the standard deviation. Batch size is set to 16. For resnext50, training each model takes around two and half hours and resnext101, each model takes around three hour and a half on a system with 2 x RTX 3090, and AMD threadripper with 24 cores. In total, training the final ensemble took around 26 hours.

2.4 Inference

During inference, we cut the large tiff images into small tiles and load them with rasterio, and then we make predictions using expansion tiles, where instead of simply mapping the predictions from small tiles onto the large prediction mask, we expand the tile in all four directions before feeding it into the model and then only use the center region's predictions as inference output (Figure 3). In our final submission, we use tiles of size of 1024 and expansion of 256, which means the models make predictions for tiles of size 1280 and then we only map the center 1024x1024 region back to the big prediction mask.

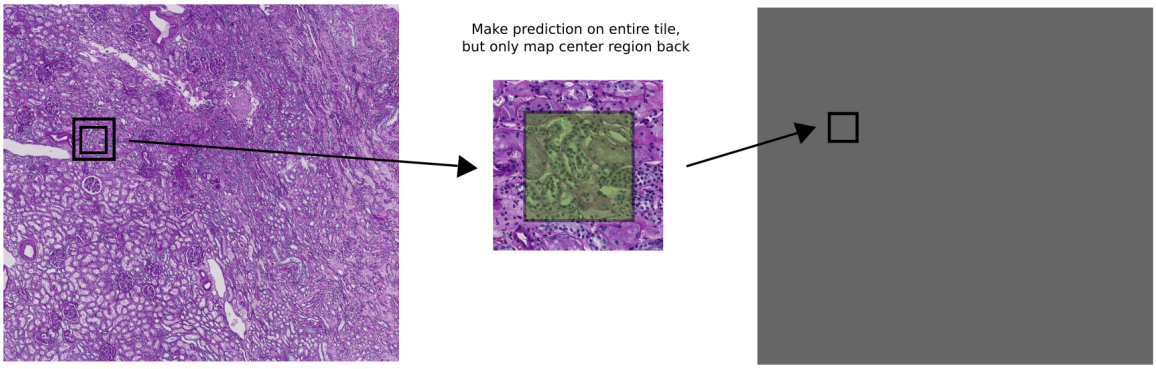

Figure 3: Visualization of how expansion tile is used during inference.

While cutting tiff images into small tiles enables is both efficient and effective, it has one downside, which is edge effects. Early in the competition, we found that using larger tiles with the same downsampling factor resulted in both better cross validation results (Figure 4) and public leaderboard scores. At that point, we already had a vague feeling that the difference in performance between different tiles might be due to edge effects. Intuitively, if we consider the amount of edges to be quantified by the perimeter and the amount of information content (or predictions needed to make) by the area, then the amount of edges has linear scaling to the size of the tiles, while the amount of information content has quadratic scaling. In other words, the larger the tile size is, the smaller the edge effects become relatively. Therefore, using larger tiles is one way to alleviate edge effects, but a better way is to use expansion tiles, which takes the edges out of the equation. Our reasoning is validated by the private lb results, as using expansion tiles leads to very significant improvements both in public and private leaderboard (Table 1).

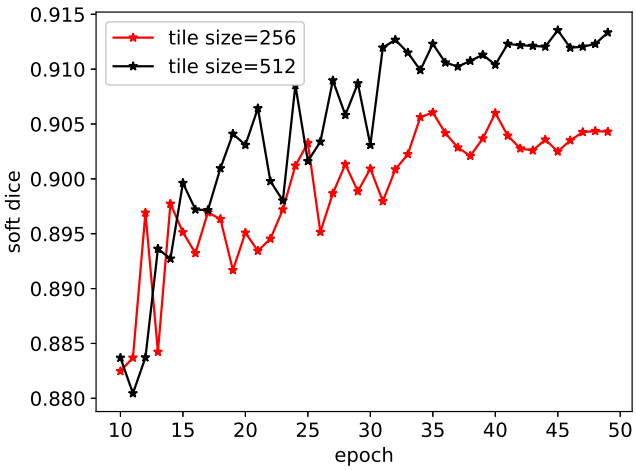

Figure 4: Comparison of validation performance of tile size 256 and 512 with 4 times downsampling

	Expansion tiles	Normal tiles
Cross Validation	0.946	0.944
Public LB	0.9187	0.9144
Private LB	0.9503	0.9467

Table 1: LB/CV performance comparison of making predictions with expansion tiles vs normal tiles using our final ensemble.

2.5 Interesting Insights

The biggest challenge the competition presents is the outlier image (d488c759a) in the public test set. Based on discussions in the forum, it seems that this image has scletoric glomeruli labeled, while the rest of the training/public test set only have non-scletoric glomeruli labeled. In order to fit well to the public test set, one would have to use some type of hand labeling, and it is clear from post competition discussions that some of the top teams on the public leaderboard were all doing hand labeling. Since we are not experts in biology, we never attempted to do hand labeling; instead, we used some of the hand labels that were publicly available. Using the hand labels to train led to much higher public lb score, boosting our lb score from 0.918 to 0.930 at the time. Beyond that, however, it was quite difficult to make improvements. With larger images, higher resolution, and the usage of expansion tiles, we were able to boost our public lb score to 0.936, which was still quite a bit lower than the top public lb scores that were in the 0.95 range.

We had an idea that the top lb scores were obtained using hand labels, so many competitors may have already overfitted to the public test set. Nevertheless, it was still possible for another outlier image to be in the private test set, so our final strategy was to make two very different submissions, one ensemble trained with hanb labels and one ensemble trained without. While training with hand labels, we used our previous best public test set predictions as pseudo labels, together with training images, to train for an additional 25 epochs, starting from a previous run with 4x downsampled images. On the other hand, we also trained models with only training data, which is our final best submission. Our final strategy was designed specifically for two senarios:

1. There are other outlier images in the private test set
2. There are no other outlier images in the private test set

The results with and without using hand are summarised in Table 2. In the end, our strategy paid off, resulting in a huge shake-up into the prize zone. Effectively using two submissions for two hypothetical senarios was what helped us win 3rd place in this competition.

	Wo hand labels	W hand labels
Public LB	0.9187	0.9362
Private LB	0.9503	0.9430

Table 2: LB performance comparison of with and without using hand labels

3 Acknowledgements

We would like to specially thank Maxim for releasing his preprocessing/training/inference notebooks, which were of tremendous help. Further, we would like to thank the organizers for hosting this competition.

References

[1] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015.

[2] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks, 2017.

[3] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. *CoRR*, abs/1612.03144, 2016.

[4] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *CoRR*, abs/1606.00915, 2016.

[5] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P. Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. *CoRR*, abs/1609.05158, 2016.

[6] Dhruv Kumar Mahajan, Ross B. Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. Exploring the limits of weakly supervised pretraining. In *ECCV*, 2018.

[7] Leslie N. Smith and Nicholay Topin. Super-convergence: Very fast training of residual networks using large learning rates. *CoRR*, abs/1708.07120, 2017.

[8] Terrance Devries and Graham W. Taylor. Improved regularization of convolutional neural networks with cutout. *CoRR*, abs/1708.04552, 2017.