3D segmentation of tumors on kidneys

Tomáš Detko

ČVUT - FIT

detkotom@fit.cvut.cz

December 2, 2020

1 Introduction

This work aims to design, train and compare the performance of State-of-the-art models within the segmentation of CT scans of patients. It provides a theoretical basis and acquaints the reader. In this work great emphasis is placed on the overall process of development of machine learning models.

2 Dataset

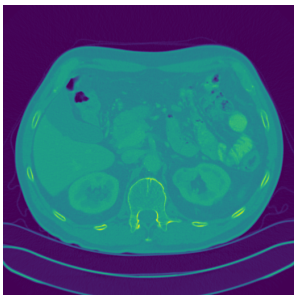
All data images used for training were obtained from the KiTS19 Challenge [1]. It is one of many machine learning competitions involving medical segmentation enthusiasts from many countries around the world.

The dataset provided by the organizers contains information on individual patients suffering from kidney cancer. Such a set consists of several CT images on which transverse cuts of the patient's abdominal cavity are captured. The thickness of these cuts is in the range of 1 mm - 5 mm, while within one patient this thickness is constant. In addition, the file contains segmentation data mask. These are pictures on which the kidneys and tumors are marked. All other organs in the images are painted black and represent background. Data were provided by the University of Minnesota Medical Center.

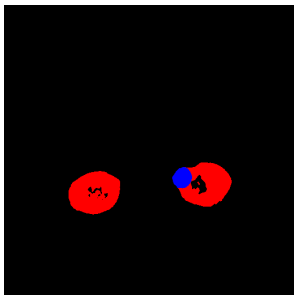
All files that contain images and ground truth labels are anonymized and saved in NIFTI format. The data file for one patient contains data in the format $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, where $x_k \in \mathbb{R}^{512\times512\times3}$ and $y_k \in \mathbb{R}^{512\times512\times3}$, $n = 45965$, x represents patient images and y represents ground truth segmentation masks of the background, kidneys and tumor. The whole dataset consists of 210 patients.

3 Data preprocessing

When training and testing new models, iteration speed through possible architectures and parameters is very important. That is why all data samples have been reduced from 512x512x3 to 128x128x3. Subsequently, all images were saved in .npy format.

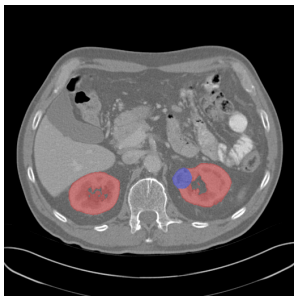


(a) Original picture from CT.



(b) Segmentation mask.

(c) CT scan in gray hue.



(d) After application of segmentationmask.

The OpenCV library was used to reduce dimension of the images. Several interpolation methods were tried and compared their results and adjusted them to correspond to the values in the original dataset. The reason we focused on different interpolation methods is the possible loss of information during transformations. Looking at the picture 2i, we can see that the tumor itself is much smaller compared to other objects. Such interpolation can negatively affect the quality of the CT scan dataset itself. If a tumor has specific features that distinguish it from other objects, then interpolation can destroy this information.

All images that do not contain a kidney or tumor are removed from the dataset during training. Models that were trained on data without this adjustment very often converged to black, because everything else (kidneys and tumors) was considered an anomaly and the weights of the models did not adapt to such an object.

The dataset undergoes further modification if we

decide to train a 3D model. Then, after removing the irrelevant images, 32-image blocks are created for each patient. Each such block has an overlap of 12 frames with the previous data block. The overlay ensures that the network can work with information that was processed in the previous cycle of the forward propagation algorithm. Thus, the overlay represents something like a context. After this processing, the blocks from all patients are concatenated to form a new dataset $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, where $x_k \in \mathbb{R}^{32\times128\times128\times3}$ and $y_k \in \mathbb{R}^{32\times128\times128\times3}$, without any other preprocessing n = 691.

4 Data augmentation

In tasks where we work with a large dataset, the DataGenerator class, which inherits properties from keras.utils.Sequence, is useful. Such generator ensures parallel processing. The data is read from the disk, augmented and the CPU sends it directly to the GPU. This principle speeds up training when more graphics cards and more CPU cores are available. DataGenerator class provides this parallel data feeding to the model. DataGenerator also takes care of the augmentation itself. Data images are horizontally flipped, they are rotated in the range (-5,5)degrees, zoom is used in the range (5,-5)percent of the image size, the image is shifted in the range (-10px,10px) in the horizontal and in the vertical direction. Such post processing of input data will ensure greater robustness of training process and the models.

5 Models

In thesis we experimented and compared results of architectures Unet2D [3], Unet3D, PSPNnet [5], DenseUnet [4], Deeplabv3+ [2].

When creating models, we divided them into two categories according to the number of learning parameters. If we compare the performance of 2 models that have same architecture and differ only in the number of learning parameters, then a larger model should have better results. That was exactly the reason why we decided to create 2 versions for PSPNet and DenseUnet networks. The input of all neural networks is modified to be able to receive images with a size of 128x128x3.

6 Used metrics

Dice coefficient (DC)

$$\frac{y\hat{y}}{y\hat{y} + \frac{1}{2}(1-y)\hat{y} + \frac{1}{2}y(1-\hat{y})} \tag{1}$$

Jaccard coefficient (JC)

$$\frac{y\hat{y}}{y\hat{y} + (1-y)\hat{y} + y(1-\hat{y})} \tag{2}$$

7 Results

Weighted dice coefficient loss (DCL_w) was used to train the final models. Such a loss function converged reasonably fast and with the same parameters gave the best results compared to other loss functions from 6. Weighted jaccard coefficient (JC_w) was chosen as a metric determining the final quality of the model. The best model was considered to be the one with the lowest error on the validation set when using JC_w.

Network	JC_w
PSPNet_s	0.8566
Unet2D	**0.8633**
DenseUnet_s	0.8522
Deeplabv3+	0.8281

Table 1: Performance of small models on test data.

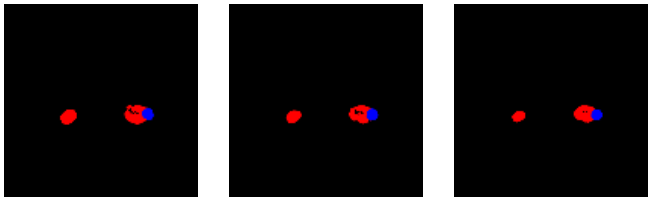
Network	JC_w
PSPNet_b	0.8654
Unet3D	**0.9113**
DenseUnet_b	0.8639

Table 2: Performance of big models on test data.

The fact that the Unet3D and Unet2D networks were optimized to the greatest extent may have contributed to the result. Unet2D network had the highest JC_w within the smaller models 1. Within models with a larger number of parameters, Unet3D won 2. This was the expected result for Unet3D. The architecture was designed to capture spatial information. Such a model can estimate the required tumor shape when processing image sequences, even if some data in the sequence are ambiguous and cannot be accurately determined using information from a single image. Thanks to 3D filters, the network was able to capture context that other networks could not.

8 Conclusion

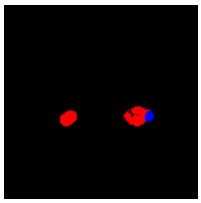
In this work, we focused on creating, training and comparing the performance of models designed for segmentation of organs and cancer. We created and trained models that are used when working with medical diagnostic scans.



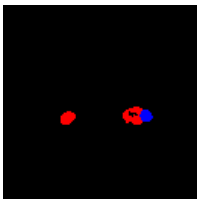
(a) Ground truth k_{th}.

(b) Ground truth k_{th+1}.

(c) Ground truth k_{th+2}.

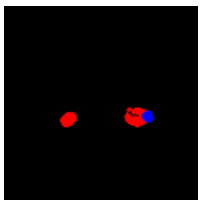


(d) Unet2D output k_{th}.

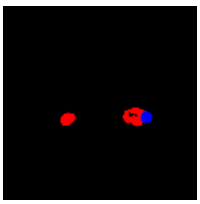


(e) Unet2D output k_{th+1}.

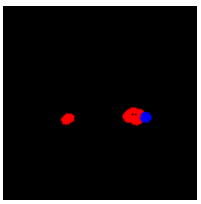
(f) Unet2D output k_{th+2}.



(g) Unet3D output k_{th}.



(h) Unet3D output k_{th+1}.



(i) Unet3D output k_{th+2}.

From all of the models, Unet3D came out as the most accurate model. The results of segmentation on the test data were above expectations. This model was able to segment tumors and kidneys with great accuracy. However, during segmentation, images that contained very small tumors were segmented as kidneys. This could be improved by training the model on the original dataset that has a higher resolution than the dataset on which these models were trained. Another possibility for improvement is the use of GAN for dataset augmentation.

Another goal was to ensure the reproducibility of the solution. This requirement was met thanks to a modification of the training script. All parameters of the dataset and the trained network are stored in a file and the imported libraries in which randomness is used are initialized by a predefined seed. The next step was to publish the source code as an open source https://gitlab.fit.cvut.cz/detkotom/mvi-sp.

9 Outline of future work

In the future, I would like to focus on training networks on the original image size (samples x 512 x 512 x 3). This will include modifying preprocessing scripts and modifying neural network architectures. To achieve even better results, it is worth considering combining a 3D network with elements of other State-of-the-art architectures.

Another important factor is the dataset. For augmentations, it would be interesting to use the CycleGAN or other types of GANs and compare the results on such trained models. Such an approach could provide improved model performance and a base line for segmentation in other medical applications when the dataset is small.

References

[1] Kits19. Challenge data. online, 2019. [cit. 2020–05–27] `https://kits19.grand-challenge.org/data/`.

[2] Chen Liang-Chieh, Zhu Yukun, Papandreou George, Schroff Florian, and Adam Hartwig. Encoder-decoder with atrous separable convolution for semantic image segmentation. online, 2018. [cit. 2020–05–27] `https://arxiv.org/abs/1802.02611`.

[3] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. online, 2015. [cit. 2020–05–27] `https://arxiv.org/abs/1505.04597`.

[4] Jegou Simon, Drozdzal Michal, Vazquez David, Romero Adriana, and Bengio Yoshua. The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. online, 2016. [cit. 2020–05–27] `https://arxiv.org/abs/1611.09326`.

[5] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. online, 2015. [cit. 2020–05–27] `https://arxiv.org/abs/1612.01105`.