Cambridge Medium (2020), **37**, 1–5

CAMBRIDGE
UNIVERSITY PRESS

ARTICLE TYPE

Final Project - Biomedical Computer Vision

Sara Paratico*

Biomedical Engineering (Biomechanics and biomaterials track), Politecnico di Milano
*Corresponding author. Email: sara.paratico@mail.polimi.it

Abstract

The Kidney Tumor Segmentation challenge (KiTS19) aimed to develop a model capable of accurately segmenting both kidneys and kidney tumors in CT scans. In this project, I worked with deep learning techniques to tackle this task. Specifically, I employed the U-Net architecture, which is renowned for its effectiveness in medical image segmentation. Following the training phase, I performed predictions to assess the efficiency of the model in segmenting kidney and tumor regions.

Keywords: Segmentation, Medical imaging, Deep Learning, Computer Vision

1. Introduction

The kidney tumor segmentation challenge took place in 2019 and involved the development of a model capable of accurately segmenting kidneys and kidney tumors in CT scans. The dataset provided for the challenge consisted of 300 high-quality abdominal CT scans, with annotations available for 210 of these scans. The remaining 90 scans were used for evaluating the models submitted by the participants.

Considering the success of U-Net and its variants in segmentation tasks within the medical domain, I chosed to implement a multi-class U-Net model to effectively segment both kidneys and tumors. By leveraging the capabilities of this architecture, the goal was to achieve accurate and reliable segmentation results.

2. Methods

2.1 Preliminary visualization of data

The dataset provided by the challenge consists of 300 folders, each representing a case. Out of these, 210 folders contain two *"nii.gz"* files each. The first file, named *"imaging.nii.gz"*, contains the total volume obtained from the clinical examination, divided into two-dimensional slices. The second file, named *"segmentation.nii.gz,"* contains the corresponding segmentations of the images. The remaining 90 cases only contain the volumes without associated segmentations; they were meant for a final evaluation phase of model efficiency. Additionally, two *".py"* files are provided, with one named *"kits.json"* that contains the clinical data for the cases.

In the context of the project, the challenge provides specific functions to handle the dataset, such as the loading cases function, which allows loading the volume and segmentation of a specific case.

To become familiar with the dataset and understand its nature, the volume and segmentation of a single case were initially loaded to obtain the data dimensions. Furthermore, a slice of the volume and its corresponding segmentation were plotted to gain a visual understanding of the data. They are shown in *Figure 1*.

It was not considered necessary to perform a size check on the entire dataset from the beginning because preprocessing was planned to resize the data.

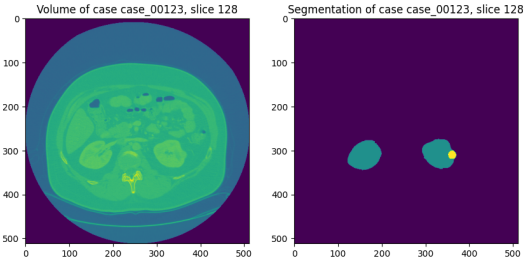

Figure 1. Plot of the 128th volume and segmentation of case 123

2.2 Preprocessing

I have developed two functions to handle the dataset: one for the volumes and the other for segmentations. These functions are responsible for loading the data, converting it into arrays, and applying preprocessing steps. Considering the potential heterogeneity of the data, I found it necessary to normalize the data and resize the images to a fixed height and width of [256, 256] pixels, resulting in square images.

In an attempt to improve the data quality, I also experimented with resampling the images to a common voxel spacing of 1x1x1mm. However, this approach significantly increased the computational load of the functions without yielding substantial improvements in the model's performance on the final results. Consequently, I decided to prioritize computational efficiency by opting for a streamlined code that was less computationally demanding.

2.3 Splitting in training and test datasets

These functions also create lists to store the individual slices in a columnar format, establishing the necessary structure for training the model. Both functions require two numerical indices as inputs, representing the boundaries of the selected cases range. By invoking each function, it's possible to retrieve an entire dataset and its corresponding labels in a single operation. I applied both functions twice: once to generate the training dataset and again to create the test dataset, which will be used during the prediction phase of the project.

2.3.1 Network

I implemented the U-Net architecture using Keras for the segmentation of kidneys and kidney tumors. The U-Net architecture is characterized by a "U" shaped structure, with a contraction path (also known as down-sampling) and an expansion path (also known as up-sampling). The contracting path reduces the image size and captures the relevant features, while the expanding path reconstructs the segmented image at a higher resolution using the extracted features. Its structure and specific layers are shown in *Figure 2*.

I worked on Google Colab, therefore it wasn't possible for me to upload the entire dataset. A subset of 150 cases was involved, 100 for training and 50 for test. In addition, during the training process, a portion of the training dataset (splitting validation rate equal to 0.1) was held out and used as a validation dataset to validate the training results.

The model was trained with batch size of 25 and training it for 25 epochs using GPU. To optimize the model, I chose the Adam optimizer with a learning rate of 1e-5.

Since the task involves multiclass segmentation, I employed categorical crossentropy as the loss function and the primary evaluation metric chosen for assessing the model's performance was accuracy.

To prevent overfitting and ensure the model's generalization, I incorporated early stopping. This technique monitored the validation loss, and if no improvement was observed for four consecutive epochs (with a patience value of 4), the training process was interrupted. This safeguard prevented unnecessary computation and allowed for a more efficient training process.

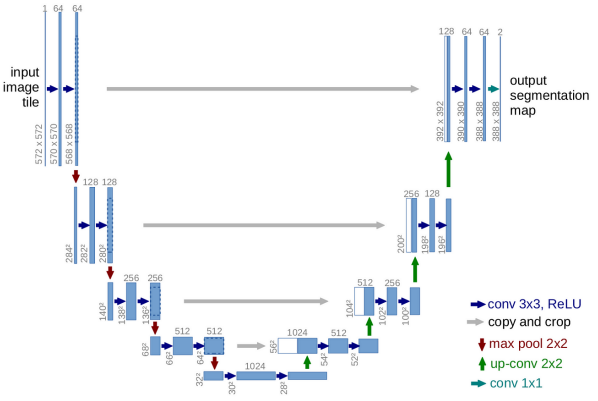

Figure 2. UNet — Line by Line Explanation

3. Results

The training process resulted in a training accuracy of 0.86 and a validation accuracy of 0.92. It is uncommon for the validation accuracy to be higher than the training accuracy, especially considering that I modified the range of cases examined compared to previous runs. If I had kept the same range, the training might have been affected as the model would have recognized the images and learned less. However, the losses are still relatively high, with training loss at 7.9 and validation loss at 8.5. Unlike accuracy, I expected the validation loss to be higher than training one. It is possible to see accuracy and loss behaviours over epochs in *Figure 3* and *Figure 4*. In *Figure 5* is also possible to see prediction results related to the unkown test dataset.

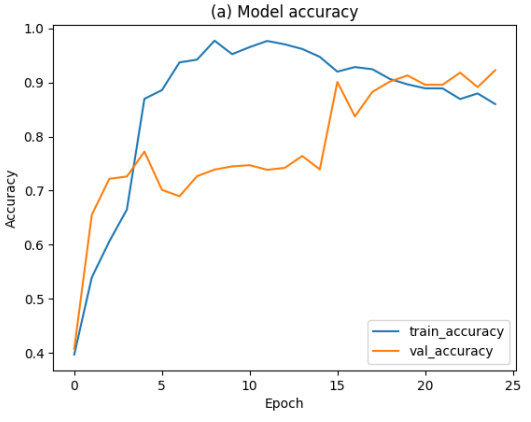

Figure 3. (a) Train and validation accuracy over epochs

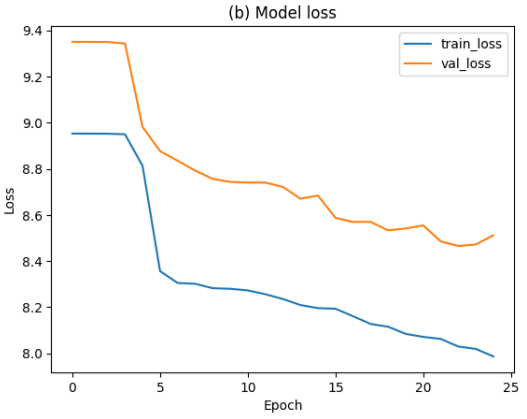

Figure 4. (b) Train and validation loss over epochs

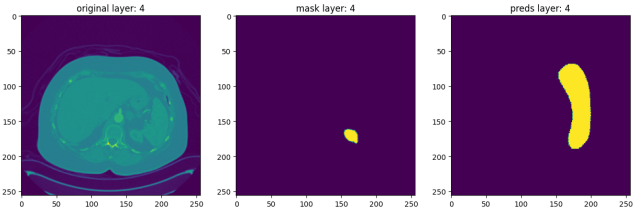

Figure 5. Comparison between a random test slice, its real segmentation and segmentation prediction

4. Conclusion

The chosen network architecture is known to be effective for this type of task, therefore I was expecting at least decent results like these.

Nevertheless, there are several potential improvements that could be implemented. Due to time constraints and limited experience, I have not implemented certain enhancements, such as data augmentation to increase image variety and make the learning process more challenging. Additionally, applying more preprocessing steps like filtering operations could be beneficial. Lastly, working with the entire dataset instead of just a portion of it would also be advantageous.

5. Discussion

Considering my limited experience in computer vision, particularly in segmentation tasks, I am already satisfied with the results I have achieved. Many steps in the process were new to me, and I have learned a great deal from this project. However, I recognize that these results may not be sufficient and I am certain that there are numerous ways to further improve them.

Initially, my plan was to explore more complex U-Net architectures, such as a 3D U-Net. The intention was to compare the results obtained from different architectures. However, due to constraints in terms of my skills, knowledge, and time availability, I was unable to pursue these ideas.

Despite these limitations, I am satisfied with the progress made and remain motivated to continue exploring and expanding my expertise in computer vision and segmentation tasks in the future.

6. Sitography

[1] https://kits19.grand-challenge.org/
[2] https://github.com/neheller/kits19

7. Bibliography

[1] Ronneberger et al. (2015). "U-Net: Convolutional Networks for Biomedical Image Segmentation. In International Conference on Medical Image Computing and Computer-Assisted Intervention" (pp. 234-241). Springer.

[2] Hesamian et al. (2019). "Deep learning techniques for medical image segmentation: Achievements and challenges". Journal of digital imaging, 32(4), 582-596