Option 1: UW-Madison GI Tract Image Segmentation

Jichen Li CWID:1046534

May 8, 2022

Abstract

I choose a Kaggle competition as my final project which is UW-Madison GI Tract Image Segmentation. The link is shown below.UW-Madison GI Tract Image Segmentation

1 Introduction

In 2019, an estimated 5 million people were diagnosed with a cancer of the gastro-intestinal tract worldwide. Of these patients, about half are eligible for radiation therapy, usually delivered over 10-15 minutes a day for 1-6 weeks. Radiation oncologists try to deliver high doses of radiation using X-ray beams pointed to tumors while avoiding the stomach and intestines. With newer technology such as integrated magnetic resonance imaging and linear accelerator systems, also known as MR-Linacs, oncologists are able to visualize the daily position of the tumor and intestines, which can vary day to day. In these scans, radiation oncologists must manually outline the position of the stomach and intestines in order to adjust the direction of the x-ray beams to increase the dose delivery to the tumor and avoid the stomach and intestines. This is a time-consuming and labor intensive process that can prolong treatments from 15 minutes a day to an hour a day, which can be difficult for patients to tolerate—unless deep learning could help automate the segmentation process. A method to segment the stomach and intestines would make treatments much faster and would allow more patients to get more effective treatment.

2 The work I have done

2.1 Choosing model

I spent some time to learn which model can be used to this task. This task belongs to Semantic Segmentation tasks. So I compared three models theoretically

1. FCN
2. Deeplabv3+
3. U-net

FCN[1] stacks a bunch of convolutional layers in a encoder-decoder fashion. The encoder down-samples the image using strided convolution giving a compressed feature representation of the image, and the decoder upsamples the image using methods like transpose convolution to give the segmented output. The FCN network upsamples only once. i.e. it has only one layer in the decoder.

DeepLabv3+[2] is a state-of-the-art semantic segmentation model having encoder-decoder architecture. The encoder consisting of pretrained CNN model is used to get encoded feature maps of the input image, and the decoder reconstructs output, from the essential information extracted by encoder, using upsampling.

The U-Net[3] build upon the concept of FCN. Its architecture, similar to the above encoder-decoder architecture, can be divided into three parts:

1. The contracting or downsampling path consists of 4 blocks where each block applies two 3x3 convolution (+batch norm) followed by 2x2 max-pooling. The number of features maps are doubled at each pooling layer (after each block) as 64 -¿ 128 -¿ 256 and so on.

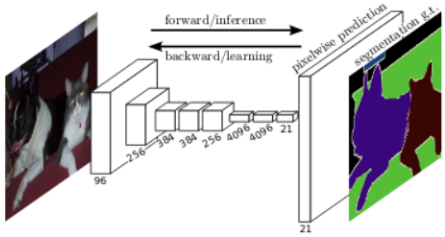

Figure 1: Fully convolutional network architecture

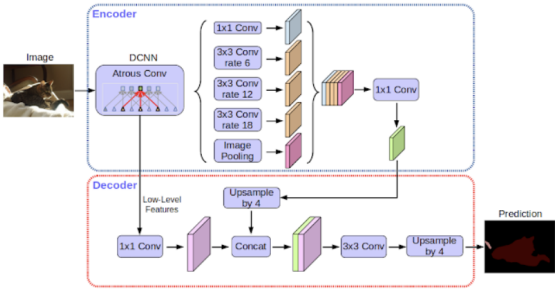

Figure 2: Deeplab architecture

2. The horizontal bottleneck consists of two 3x3 convolution followed by 2x2 up-convolution.

3. The expanding or upsampling path, complimentary to the contracting path, also consists of 4 blocks, where each block consists of two 3x3 conv followed by 2x2 upsampling (transpose convolution). The number of features maps here are halved after every block.

Imran Ahmed et al have done a research[4] which showed the difference between these three deep learning model. They evaluated three models in top view person images task.The output results are nearly same for all three models. All three models show good results for multiple person images.

And the evaluation results of all three models with different measuring parameters are showed in Figure 5.

Even though all three models can perform well on the semantic segmentation task, the inference time shows that DeepLabV3 and U-Net are slightly slower than the FCN model. That is, DeepLabV3 spends more computing resources. So I choosed U-net in this task.

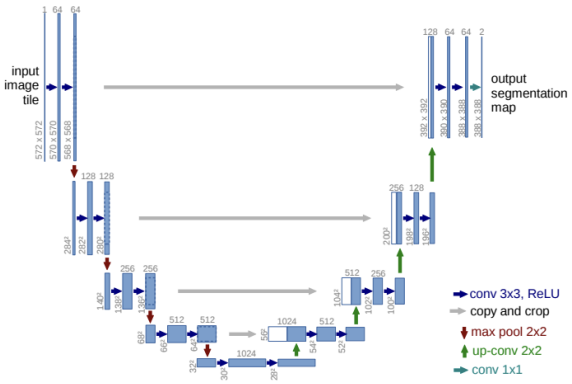

Figure 3: U-net architecture

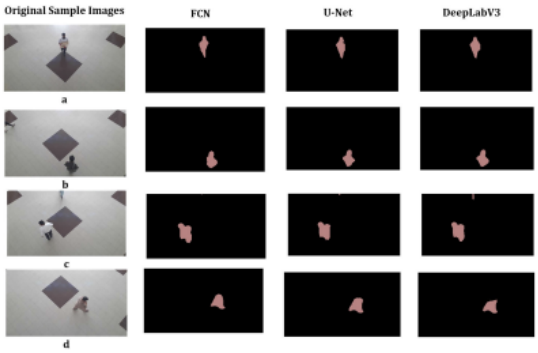

Figure 4: Outputs of three models in top view person images task

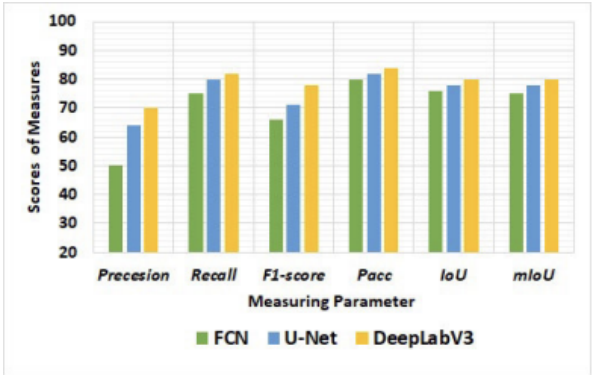

Figure 5: Evaluation results of three models

2.2 Data preprocessing

In order to reduce the file size, mask training data used run-length encoding,which is RLE, on the pixel values. It using pairs of values that contain a start position and a run length. E.g. '1 3' implies starting at pixel 1 and running a total of 3 pixels (1,2,3). So I need to transform the RLE into grayscale format. The outputs is shown in figure 6.

The training data contains many invalid data. For some reason, there are lots of pictures with no masks that cannot be used in the training. So I cleaned up the invalid data.

There I made I mistake in the beginning. I forgot to normalize the input data first, which cause the model to not converge. And what's more is you need to fulfill the background pixel and add the mask of background into training data as well. So if you want to segment 3 classes, you need to prepare 4 masks as the training data which include the mask of background.

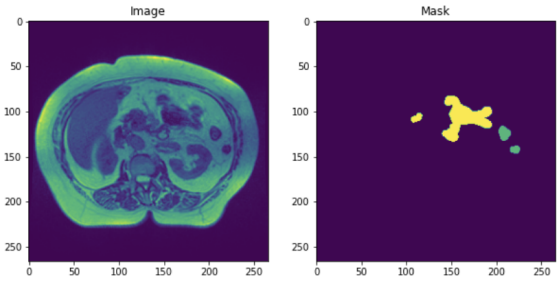

Figure 6: Transform RLE into grayscale format

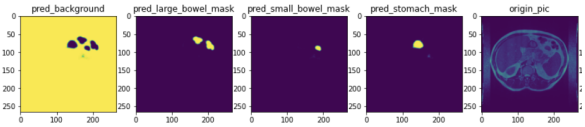

Figure 7: Add mask of background into training data

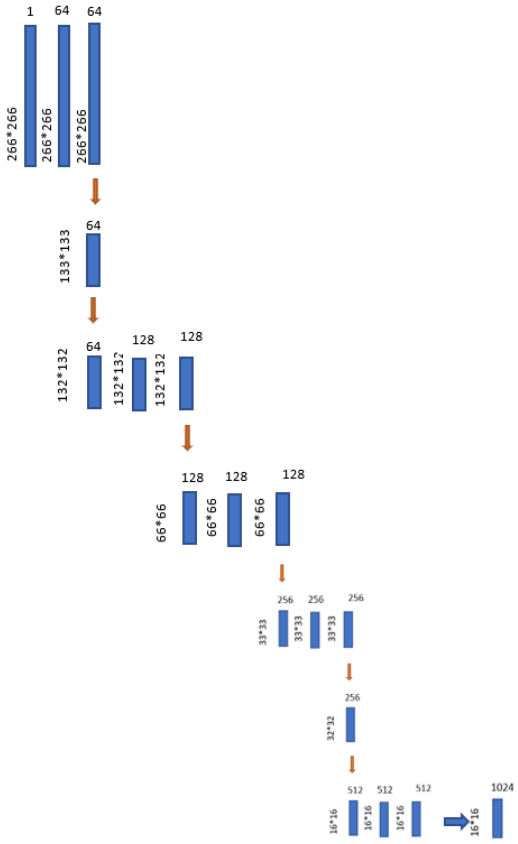

Figure 8: Encoder architecture

2.3 Architecture of my U-net model

The input data has three different sizes, 266*266, 276*276, 310*360. At first I used 266*266 for training. To input 266*266 pictures into U-net model, I tune the model network. I added MaxPolling and ZeroPadding layers to the model. So the encoder progress encodes the picture into 266*266, 133*133, 132*132, 66*66, 33*33, 32*32, 16*16 in sequence. And I also add Dropout layers to make the model more robust. The number of the total parameters is 26,806,068. The encoder architecture is shown as figure 8. The structure of the decoder is a mirror image of the structure of the encoder. every same size of encoder and decoder using concatenate layer to connect.

2.4 Training

At first I used CPU to train the model, which is too slow. It usually take me nearly ten hours to train. So I used kaggle notebook that can provide free GPU to training. By using GPU, training speed can be increased ten times. That helps a lot.

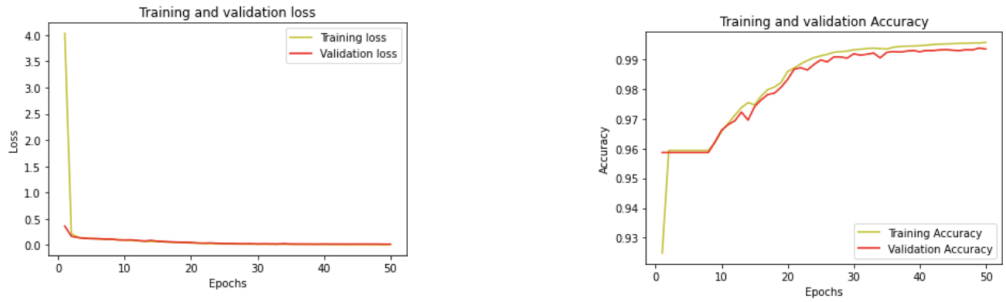

Figure 9: The loss and accuracy line

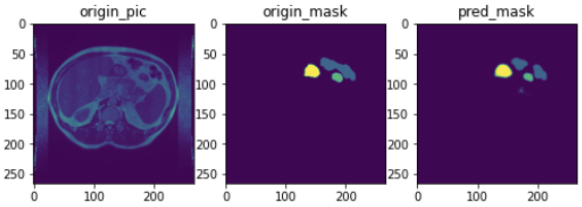

Figure 10: Random predict

2.5 Result

I choose categorical crossentropy as my loss function. After 50 epochs of training, the accuracy reached 0.9956 and the validation accuracy reached 0.9934. The loss and accuracy line graphs are shown as figure 9.

And I predict a random picture. The result showed that U-net can predict the result well.

The kaggle notebook I wrote is shown as below.My u-net sample code

2.6 What I've learned

I learned a lot in this final program. This is the first deep learning model I complete by my own. I acknowledge the arcitechture of FCN, DeepLabV3+ and U-net. And I will not forget to normalize the input data next time. And GPU is faster than CPU in training. I also learned how to tune the model in order to fit my data. There is a lot tricks here.

2.7 what's next

I plan to finish the kaggle competition after the final program. There are many things I can improve. First I need to adjust the size of input data. So I can using different size of training picture in the model. I need to transform my result into RLE format to verify my model in kaggle. And I need to tune my model layer trying to reach a better score.

References

[1] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[2] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 801–818.

[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*. Springer, 2015, pp. 234–241.

[4] I. Ahmed, M. Ahmad, F. A. Khan, and M. Asif, "Comparison of deep-learning-based segmentation models: Using top view person images," *IEEE Access*, vol. 8, pp. 136 361–136 373, 2020.