Multiple Sclerosis Lesion Segmentation using 3D Unet

Yuxi Luo

May 2019

1 Introduction

Multiple Sclerosis (MS) is a chronic disease of central nervous system, for which a patient's status requires radiologist experts to manually delineate MS lesions from the corresponding Magnetic Resonance Imaging (MRI). There are several problems for the traditional clinician approach: images from different centers often have different modalities and qualities with different MRI standards; radiologist experts often have their own discretion for some potential lesion areas, which adds to inter-expert variability... Therefore, performing delineation on large datasets is almost impossible, and automatic segmentation methods are in crucial need.

There are several challenges aiming to provide robust automatic segmentation methods, which should be resistant to different MRI image qualities, and should reflect a generalization of inter-expert variability for the final segmentation result. Often the data set provided by these challenges is very small, therefore the model should prepare to extract as much features as possible from the small training data set.

This project intends to improve results from the MICCAI 2016 challenge, using 3D Unet model to provide a more accurate outputs for MS segmentation challenge.

2 Data

This project uses the training dataset provided by the MICCAI 2016 challenge. There are 3 centers that has different scanners to reflect different MRI image qualities, which provide 4 modalities: FLAIR, DP, T1 and T2. Each center provides 5 images for training, which are delineated by 7 experts to reflect inter-expert variability of manual segmentation. In total, there are 15 images for training, where each image has 4 modalities and 7 manual segmentations. The challenge provides pre-processed data, which for each image computes a consensus segmentation from 7 manual segmentations, and standardizes each image with denoising, rigid registration, brain extraction, and bias field correction.

In particular, this project uses the pre-processed data provided by the challenge. According to the results from the challenge, FLAIR provides the most precise information of the images among all the provided modalities. Therefore, this project uses FLAIR modality for the 15 training images, and the consensus segmentations as ground-truth. A summary of the dataset is presented in Table 1.

Table 1: Details of each sequence and scanner for MS dataset

Scanner	**Matrix**	**Training cases**
GE Discovery 3T	512 * 512 * 224	5
Siemens Aera 1.5T	256 * 224 * 128	5
Philips Ingenia 3T	336 * 336 * 261	5

3 Model

3.1 Structure

This project builds on 3D Unet structure in Keras. Originally Unet structure only deals with 2D medical images, and the loss function is a combination of binary cross entropy and pixel-wise soft-max. The model used in this project modifies the structure, so that it could take 3D image input, and use Dice score loss as loss function to improve the quality of segmentation.

The model has 4 layers of contracting and expansive paths with 32 base filters. Each layers in contracting path has two (3, 3, 3) convolution followed by ReLu activation and (2, 2, 2) max pooling, and each layers in expansive paths has a (3, 3, 3) de-convolution for concatenated matrix with current layer and its corresponding contracted layer. At the final layer, (1, 1, 1) convolution is used to map the feature vector to 2 classes of output (lesion / non-lesion). The model should take input with arbitrary shape, since images provided by different centers varies by shape. Below is a summary of model structure with input size (64, 64, 64):

Table 2: Summary of model structure

Layer	**Name**	**Dimension of output**
0	Input	(1, 64, 64, 64)
Contract 1	Conv3D + ReLU	(32, 64, 64, 64)
Contract 1	Conv3D + ReLU	(64, 64, 64, 64)
Contract 1	Max pooling	(64, 32, 32, 32)
Contract 2	Conv3D + ReLU	(64, 32, 32, 32)
Contract 2	Conv3D + ReLU	(128, 32, 32, 32)
Contract 2	Max pooling	(128, 16, 16, 16)
Contract 3	Conv3D + ReLU	(128, 16, 16, 16)
Contract 3	Conv3D + ReLU	(256, 16, 16, 16)
Contract 3	Max pooling	(256, 8, 8, 8)
4	Conv3D + ReLU	(256, 8, 8, 8)
4	Conv3D + ReLU	(512, 8, 8, 8)
Expand 3	Deconvolution3D	(512, 16, 16, 16)
Expand 3	Concatenation	(768, 16, 16, 16)
Expand 3	Conv3D + ReLU	(256, 16, 16, 16)
Expand 2	Deconvolution3D	(256, 32, 32, 32)
Expand 2	Concatenation	(384, 32, 32, 32)
Expand 2	Conv3D + ReLU	(128, 32, 32, 32)
Expand 1	Deconvolution3D	(128, 64, 64, 64)
Expand 1	Concatenation	(192, 64, 64, 64)
Expand 1	Conv3D + ReLU	(64, 64, 64, 64)
0	Conv3D	(1, 64, 64, 64)

3.2 Patch training

The model is trained on NVIDIA Tesla P40 GPU with 64GB memory.

Initially the entire images were fed into the model for training. However, the dimension for the entire image is too large for such a model. After testing around the parameters, the GPU used for this project can only afford 1 layer.

Sliding-window approach is used instead. Each image is sliced to a number of patches for training. The patch size is set to (64, 64, 64) so that each patch is large enough to include a potential region of lesion, and the patch gap is set to 16 to allow some overlapping regions between two patches. Each image and its

corresponding ground-truth segmentation are zero-padded with respect to the patch size, and then sliced to generate patches for training. During the training stage, each patch is generated randomly to feed into the model.

4 Training log

Below are training logs for training 5 epochs with slight modifications on the use of dataset and loss function. Each epoch takes approximately 4 hrs.

4.1 Training on Dice Loss with all patches

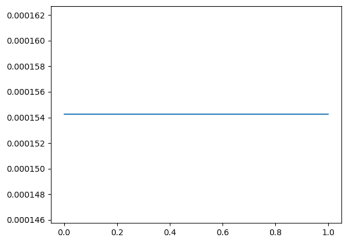
Initially the entire dataset of patches is used for training. However, the overall result after training is not promising, because most patches the model trained on do not contain any lesion region. Below are training log to train on entire dataset with dice loss:

Table 3: Training log on Dice Loss with all patches

Epoch	Training Dice Score	Training Loss	Validation Dice Score	Validation Loss
1	0.6220	0.3780	0.6039	0.3961
2	0.6385	0.3615	0.6039	0.3961
3	0.6385	0.3615	0.6039	0.3961
4	0.6385	0.3615	0.6039	0.3961
5	0.6385	0.3615	0.6039	0.3961

Even though the model is trained for several epochs, the training and validation dice score do not improve at all. The output segmentation is almost all black. Below is graph of threshold value vs. dice score, which demonstrates that the dice score, after thresholding, is still approaching to 0. Therefore, the model need to be modified so that it could learn useful information as much as possible.

Figure 1: Threshold vs Dice score



4.2 Training on Dice Loss with selected patches

Inspired by "A Multiscale Patch Based Convolutional Network for Brain Tumor Segmentation", which only uses patches that contains at least 0.01 % lesions, the dataset is then processed to generate patches only when the patch contains at least 0.01 % lesions. The output segmentation from this model improves a lot, and provides insights about lesion as we expected.

We can notice that after the third epoch, the model starts to overfit and the resulting dice score stays around 0.60 for the validation dataset. Therefore, the following post-processing is performed on outputs generated by weights from Epoch 3.

Table 4: Training log on Dice Loss with selected patches

Epoch	Training Dice Score	Training Loss	Validation Dice Score	Validation Loss
1	0.6566	0.3434	0.6013	0.3987
2	0.8843	0.1157	0.5048	0.4952
3	0.9240	0.0760	0.6145	0.3855
4	0.9425	0.0575	0.5981	0.4019
5	0.9544	0.0456	0.6021	0.3979

The final dice score for this model should be determined after merging the pacthes back to entire image, and calculate the overall dice with respect to ground truth segmentation. After enumerating through all possible threshold values, the optimal threshold value is 0.18, and final dice score becomes 0.8837.

Figure 2: Threshold vs Dice score

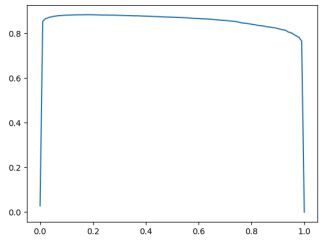
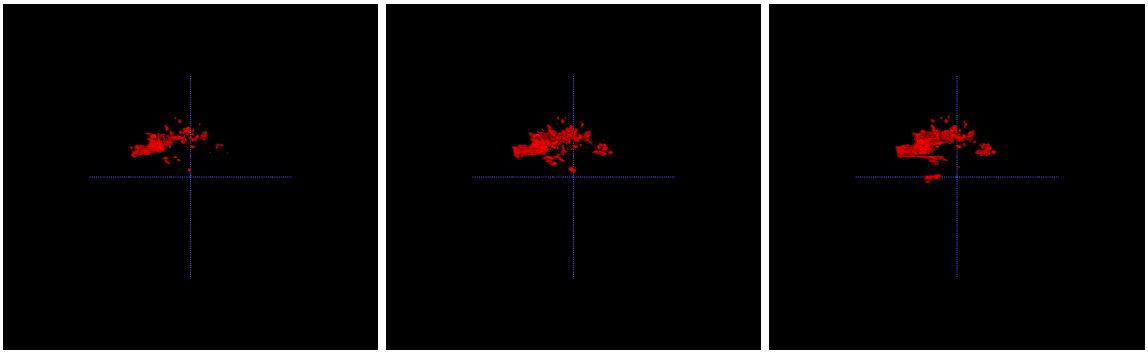


Figure 3: Validation image vs Ground truth on (336, 336, 261) image



(a) Merged output (b) Thresholded output (c) Ground truth

4.3 Training on Binary Cross Entropy with selected patches

The model is also modified to train on Binary Cross Entropy as the loss function instead of Dice Score, to test whether Binary Cross Entropy could serve as a better loss function for training. Since it is proved that training on all patches does not provide useful output, this model is trained on Binary Cross Entropy with selected patches.

Table 5: Training log on Binary Cross Entropy (select epoch 6

Epoch	Training Dice score	Training Loss	Validation Dice score	Validation Loss
1	0.5122	0.0245	0.5665	0.0353
2	0.8538	0.0041	0.5879	0.0518
3	0.9160	0.0025	0.5985	0.0580
4	0.9450	0.0017	0.6130	0.0550
5	0.9634	0.0012	0.6266	0.0626

The resulting validation dice score seems similar to the one trained with dice loss – around 0.62, but in reality, after the validation patches are merged and the optimal threshold is selected, the final dice score is 0.9824 with threshold value 0.59, which is way better than previous model.

Figure 4: Threshold vs Dice score

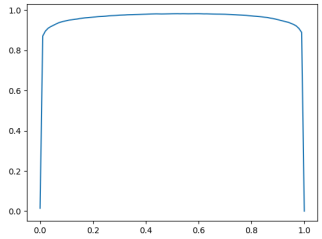
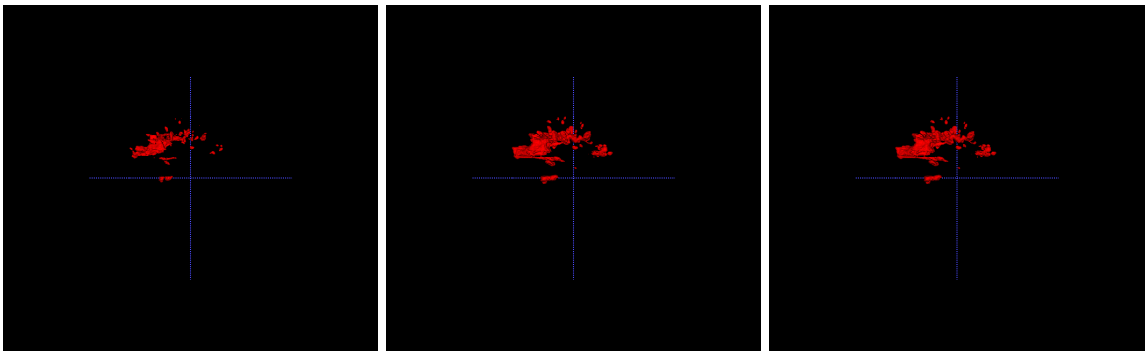


Figure 5: Validation image vs Ground truth on (336, 336, 261) image



(a) Merged output (b) Thresholded output (c) Ground truth

5 Conclusion

From the training history, the model should be trained on patches that contain at least 0.01 % lesion voxels. Although Dice Score is a standard way to measure the correctness of segmentation, it is surprising that the model using Dice Loss performs less good than that of using Binary Cross Entropy. Overall, the model structure generates decent outputs even with this kind of small dataset.

In conclusion, the final dice score of Dice Loss model is 0.8837 with threshold value of 0.18, and the final dice score of Binary Cross Entropy model is 0.9824 with threshold value of 0.59. Even though both dice scores are relatively high, the final dice score of Dice Loss model uses a very small threshold value comparing to that of Binary Cross Entropy model, which potentially indicates that Binary Cross Entropy should be a better loss function for this case.

It is worthy noted that the best dice score in the original challenge is 0.591. Although the testing dataset is not provided by the organizer, the resulting weight of this model should outperform the winners of the original challenge.

6 Future works

Training this model suffers from small number of dataset. In the future, this project could incorporate dataset from 2015 Longitudinal MS Lesion Segmentation Challenge, which provides additional training and testing images. However, the raw data should also be processed with similar techniques used in 2016 MICCAI challenge.

There are some insights from the result of the 2016 MICCAI challenge stating that, even though FLAIR modality provides the best quality, using multiple modalities could potentially improve the result of the model. Therefore, the model could use more channels for each image to include all the modalities for training.

Since the dataset is relatively small, K-fold cross validation could be performed to select the optimal hyper-parameters for this model to further improve the results of this model.

7 Sources

- Source code on Github:
- https://github.com/Skycocoo/MS-Lesion-Segmentation
- Training results:
- https://drive.google.com/drive/folders/1xjZnuejdtKepHYj6X9sVsGD2sir0QrLk?usp=sharing