Project 2

Hui Wang, s210331, Jianting Liu, s210141, Jiahao Li, s210313

1 Introduction

In this project, we are working with the LIDC-IDRI segmentation dataset, which consists of 2D patches from CT scans where for each patch, lesions have been manually segmented by 4 out of 12 different radiologists. First, we design and use the first annotation to train a U-Net network for segmentation. Then we implement the following measures for validating segmentation performance: Dice overlap, Intersection over Union, Accuracy, Sensitivity and Specificity. In order to improve the segmentation, We try to use data augmentation, positive weights to make up for class imbalance, other loss functions, varying architecture and various optimization tools.

This dataset is a good representation of the typical ambiguities that appear in CT scans, For each scan, 4 radiologists (from a total of 12) provided annotation masks for lesions that they independently detected and considered to be abnormal, so there may be disagreement between annotators. In order to compare distribution of segmentation, we use generalized energy distance.

2 Architecture

In the U-Net, we use input image of size 128x128x3. Hence the size at various locations will differ from that in the original paper but the core components remain the same. Figure 1 is the detail of the architecture.

We also build two others neural network by consulting to basic model.

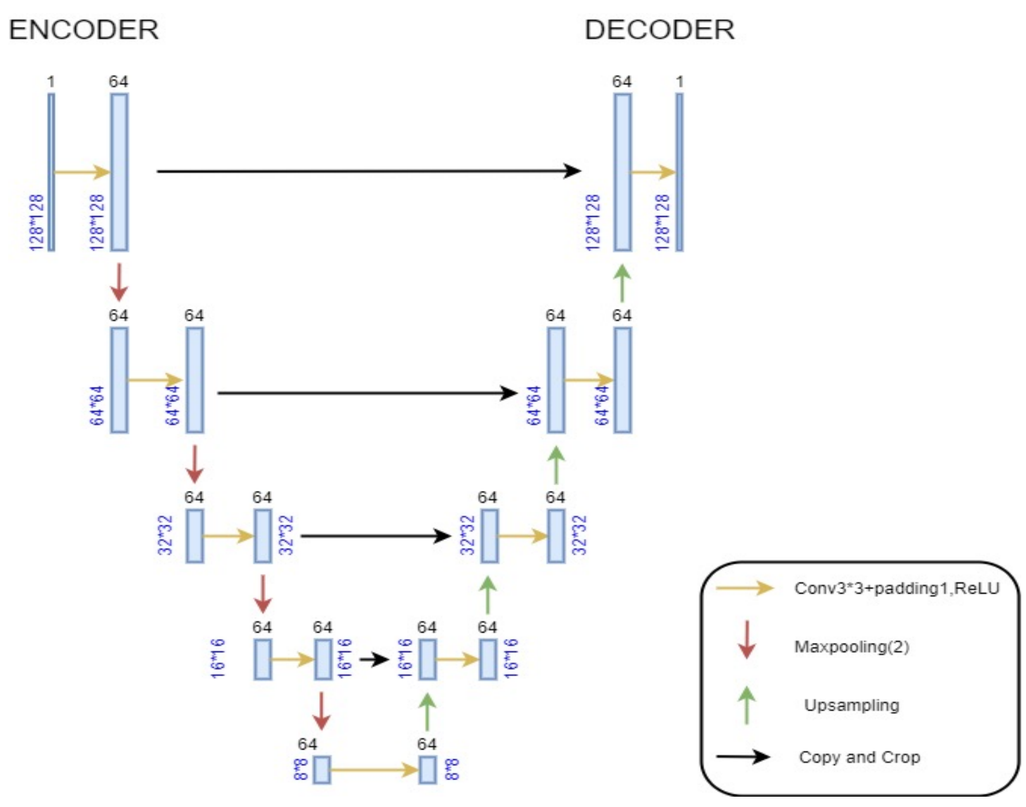

Figure 1 U-Net Architecture

3 Generalized Energy Distance

In order to compare distribution of segmentation, we use generalized energy distance. We extract annotation data drawn by 4 experts and then we get 4 different predictions given by four different models, 4 different ground truth(masks of lesion) which are relevant to the predictions. As d(x,y) = (1-IoU), we calculate IoU seperately between all predictions and all ground truth, among predictions and among ground truth to get the d(x,y), then according to the formula of GED, we can compute the distance.

We choose images in the first batch and we get the result that the smallest distance occurs in the second image with **0.0625**, the largest distance occurs in the fifth images with **0.8233**.

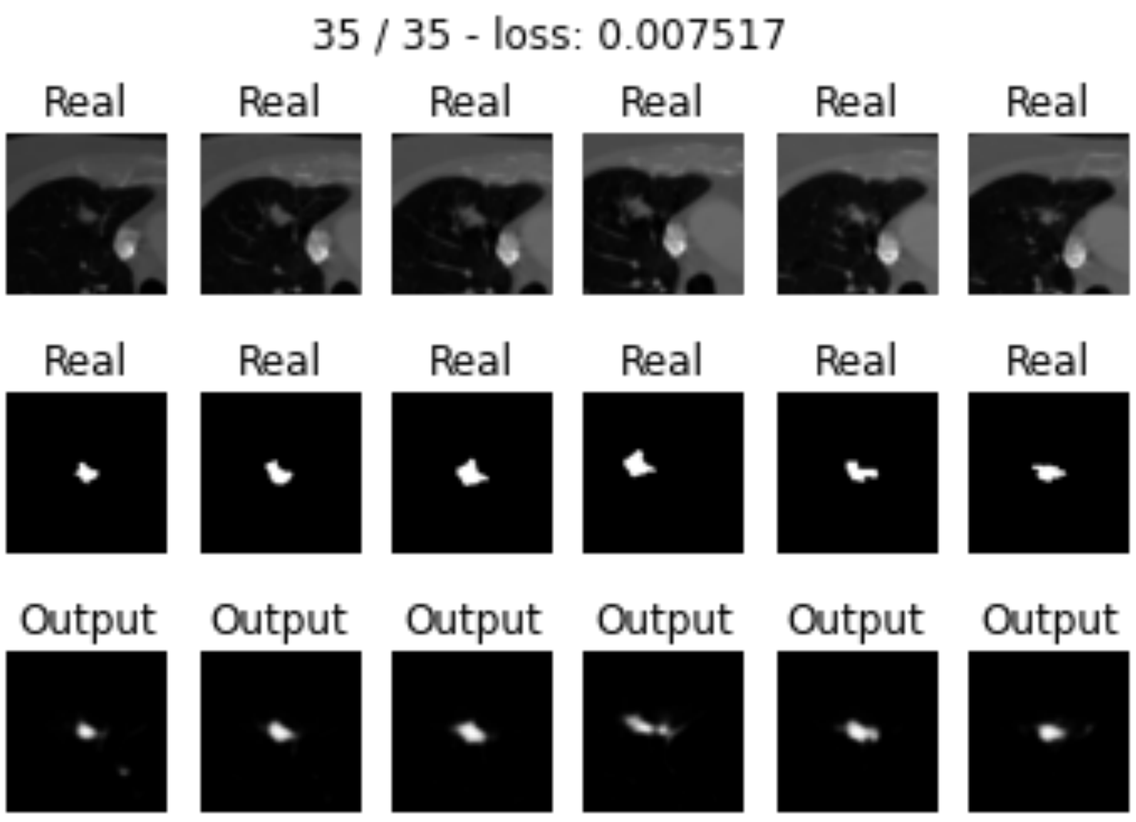

Figure 2 Six Examples of U-net2 Model

4 Performance Evaluation

We use following measures for validating segmentation performance: Dice overlap, Intersection over Union, Accuracy, Sensitivity and Specificity.

Because our dataset has a class imbalance, accuracy or specificity can become unreliable metrics for measuring our performance. In the dataset we had 99.41% pixel in the training data belonging to background, we could build a model that was 99% accurate by just saying everything belonged to background. Under this situation, IoU, Dice and Sensitivity can work better.

4.1 Performance with different model

We try 3 different models, which are greatly inspired by Exercise2. The U-net1 network's architecture is shown is figure1. U-net2 network is similar as U-net, we replace max-pooling by convolutions with stride=2, and replace upsampling by transpose-convolutions also with stride=2. The third network is EncDec network, which is a simple encoder-decoder architecture for segmentation. The performance of different networks are shown below. From the table we can find that U-net2 performance best and EncDec net performance worst. In figure 2, we plot some examples of U-net2 modle

TYPE	IoU	Accuracy	Sensitivity	Specificity	Dice
Training EncDec	0.009737	0.994194	0.009738	1	0.018283
Training Unet1	0.553601	0.997027	0.633380	0.999159	0.710756
Training Unet2	0.587021	0.997242	0.680310	0.999068	0.737654
Test EncDec	0.004272	0.993603	0.004272	1	0.007914
Test Unet1	0.337190	0.995893	0.44552	0.999205	0.519424
Test Unet2	0.398440	0.995831	0.507808	0.998832	0.541382
Validation EncDec	0.002153	0.993886	0.002153	1	0.004133
Validation Unet1	0.375602	0.9955634	0.480039	0.998957	0.511369
Validation Unet2	0.385147	0.995508	0.515788	0.998522	0.519038

Table 1 Performance with Different Model

4.2 Data Augmentation

In order to improve the segmentation, we try to use data augmentation consist of horizontal flip for the dataset image. In our case data augmentation doesn't greatly improve performance.

TYPE	IoU	Accuracy	Sensitivity	Specificity	Dice
Training unet1	0.553601	0.997027	0.633380	0.999159	0.710756
Training with DA	0.574035	0.997225	0.631919	0.999369	0.724347
Test unet1	0.337190	0.995893	0.44552	0.999205	0.519424
Test withDA	0.384914	0.995931	0.457262	0.999256	0.522299
Validation unet1	0.375602	0.9955634	0.480039	0.998957	0.511369
Validation with DA	0.364638	0.995694	0.476031	0.999164	0.503237

Table 2 Performance with/without Data Augmentation

4.3 Optimizer

We tried both SGD and Adam, it founds out Adam optimizer works much better than SGD.

TYPE	IoU	Accuracy	Sensitivity	Specificity	Dice
Training with SGD	0.0	0.994122	0.0	1.0	0.0
Training with Adam	0.574035	0.997225	0.631919	0.999369	0.724347
Test with SGD	0.0	0.993549	0.0	1.0	0.0
Test with Adam	0.384914	0.995931	0.457262	0.999256	0.522299
Validation with SGD	0.0	0.993860	0.0	1.0	0.0
Validation with Adam	0.364638	0.995694	0.476031	0.999164	0.503237

Table 3 Performance with SGD/ Adam

4.4 Loss Function

We tried both binary cross-entropy and focal loss function. It turns out binary cross-entropy loss function performers better than focal loss function

TYPE	IoU	Accuracy	Sensitivity	Specificity	Dice
Training with BCE	0.574035	0.997225	0.631919	0.999369	0.724347
Training with focal	0.493608	0.996352	0.613084	0.998596	0.658538
Test with BCE	0.384914	0.995931	0.457262	0.999256	0.522299
Test with focal	0.384066	0.995598	0.511904	0.998646	0.533412
Validation with BCE	0.364638	0.995694	0.476031	0.999164	0.503237
Validation with focal	0.355375	0.995348	0.481648	0.998464	0.490879

Table 4 Performance with BCE/ focal