Swin UNETR

Contents

Introduction

Architecture

Method

Experiments

2

Swin UNETR에 대한 발표를 시작하겠다. Introduction에서는 간략한 논문 소개 및 관련 데이터셋에 대해 말하고 Architecture에서는 모델에 대한 설명을 하겠다. Method에서는 모델과 학습에 관련된 기술을 말하고 Experiments에서는 성능과 평가지표에 대한 설명을 하겠다.

1 Introduction

- Swin UNETR: Swin Transformers for Semantic Segmentation of Brain Tumors in MRI Images
  - 2022년 Lecture Notes in Computer Science 학회에 게재된 논문
  - NVIDIA Team에서 개발, UNETR에 Swin Transformer를 적용한 모델을 제안함
  - Brain Tumer Segmentation에서 SOTA를 기록 (Multi-Organ에서도 SOTA)

Swin UNETR: Swin Transformers for Semantic Segmentation of Brain Tumors in MRI Images

Ali Hatamizadeh[1], Vishwesh Nath[1], Yucheng Tang[2], Dong Yang[1], Holger R. Roth[1], and Daguang Xu[1]

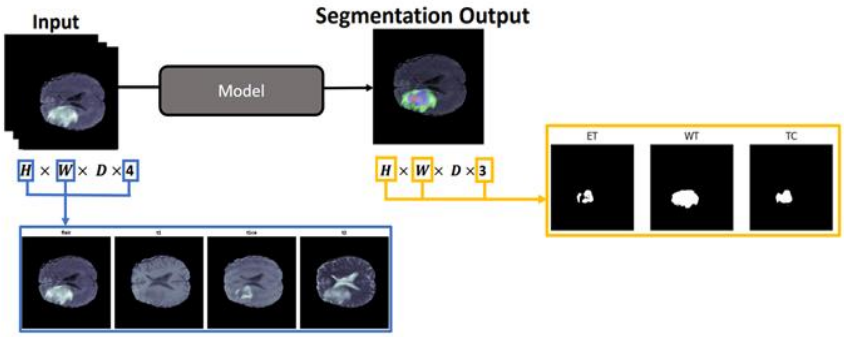
[1] NVIDIA
[2] Vanderbilt University
ahatamizadeh@nvidia.com

3

Swin UNETR은 2022년 LNCS 학회에 게재된 논문이고 NVIDIA 팀에서 개발을 하였다. 논문 제목만 봐도 알다싶이 UNETR에 Swin을 적용한 모델이고 현재 Brain Tumor Segmentation에서 SOTA를 기록하고 있다. 뿐 만 아니라 인체의 여러 기관을 보는 Multi-Organ Segmentation에서도 SOTA를 기록하고 있다고 한다.

1 Introduction

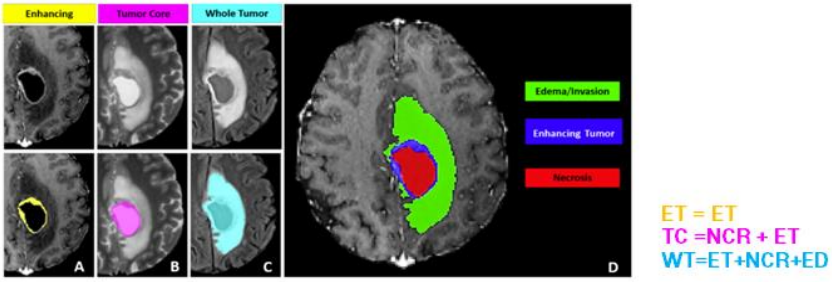
- Brain Tumor Segmentation
  - 다양한 Scanner로 뇌를 촬영한 3D Inputs를 받아, Brain Tumor를 Segmentation하는 Task
  - Output의 각 채널은 Brain Tumor의 범주에 해당하는 (ET, WT, TC)의 Predicted Segmentation Map
  - Dataset은 주로 BraTS를 이용함



Brain Tumor Segmentation은 다양한 Scanner로 뇌를 촬영한 3d input을 받아 Brain Tumor을 Segmentation하는 Task이다. Input 그림에서도 보다시피 뇌에 대한 MRI 사진이 flair, t1, t1ce등 서로 다른 스캐너와 프로토콜을 통해 출력되었다. 이런 이미지를 Multi-Modal Image라고 표현하기 때문에 이 분야는 멀티-모달 딥러닝이라고 표현할 수 있다. 그리고 output을 보면 각 채널별로 Brain Tumor의 범주에 대해 Segmentation Map을 예측한 것을 확인할 수 있다. 또한 input과 output에 대한 그림이 2D로 표현이 됐는데, 실제로는 3D Input과 Output을 처리한다. 따라서 input에는 flair,t1,t1ce,t2가 각각 D개만큼 존재하고, output도 ET,WT,TC에 대한 segmentation map이 D개만큼 존재한다.

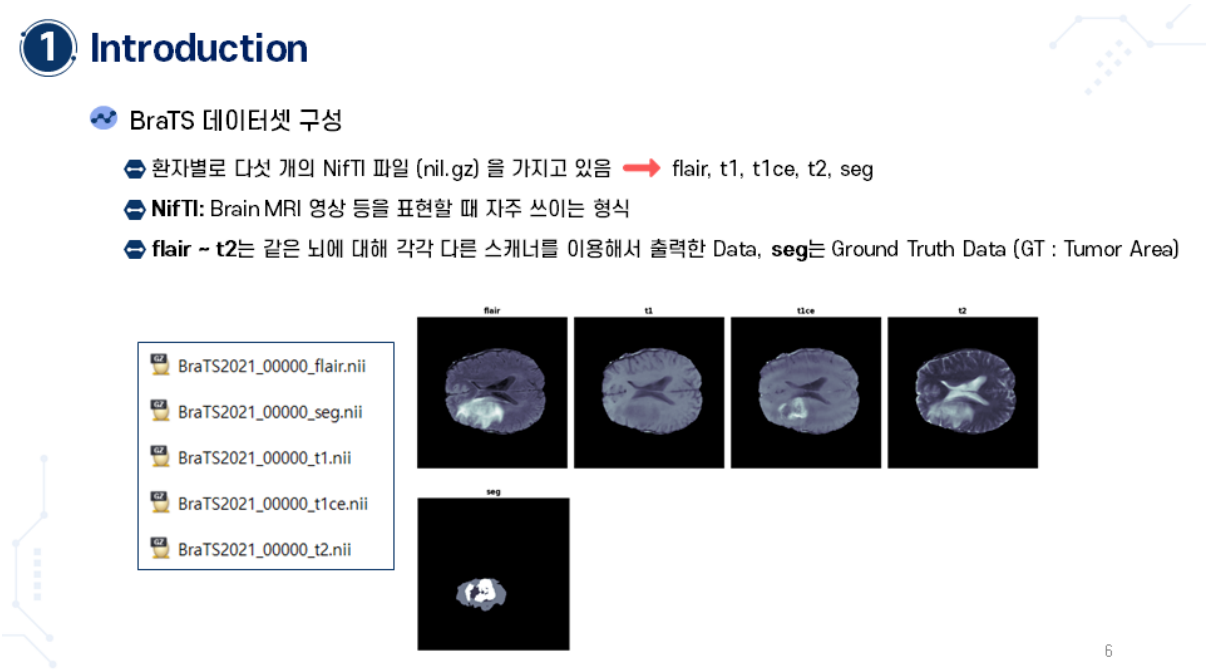
1 Introduction

- BraTS 데이터셋
  - Brain Tumor의 세 가지 범주에 대한 3D Segmentation Dataset
  - **Category for Data:** Edema(ED), Enhancing Tumor(ET), Necrosis(NCR)
  - **Category for Segmentation:** Enhancing(ET), Tumor Core(TC), Whole Tumor(WT)



ET = ET
TC =NCR + ET
WT=ET+NCR+ED

Brain Tumor를 다루는 주된 데이터셋은 BraTS라고 하고, 이 데이터셋은 Brain Tumor의 세 가지 범주를 다루고 있다. 오른쪽 큰 사진이 실제 데이터셋에서 제공하는 범주인데, Edema, Enhancing Tumor, Necrosis를 다루고 있다. ED는 뇌종양이 넓게 분포되어 있는 형태고, ET는 고리형태이거나 속이 빈 형태이다. NCR같은 경우 ET 근처에 속이 채워진 채로 존재한다. 그런데 실제로 Segmentation을 할 때는 왼쪽 여섯 개의 그림과 같이, ET, TC, WT로 범주를 나눈다고 한다. ET는 그대로이고, WT는 NCT과 ET를 합친 영역이며, WT는 모든 범주를 포함하고 있다고 생각하면 된다.

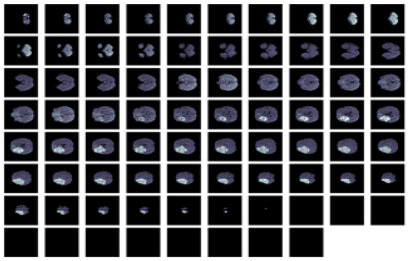


이 데이터셋은 약 1500명의 환자를 다루고 있는데, 환자별로 다섯개의 NifTI 파일을 가지고 있다. NifTI은 주로 뇌 영상을 저장하는데 사용하는 파일 형식이고, 중요한 점은 이 파일을 다섯 개나 가지고 있다는 점이다. 이 중 seg는 Segmentation Map에 대한 정답에 해당이 되고 조금 있다가 다시 알아보겠다. 그리고 flair에서 t2까지는 똑같은 뇌 사진을 서로 다른 Scanner와 프로토콜을 통해 출력을 한 것인데, 이렇게 한 이유는 뇌 종양이 걸렸다고 하더라도 특정 방식에서만 확인이 될 수 있기 때문이라고 한다. 그림에서도 보다시피, flair와 t2에서는 뇌 종양의 영역이 표시되지만 다른 방식은 잘 표시가 되지 않는다. 그리고 상황에 따라 t1과 t1ce에서만 잘 보이는 경우도 있다고 한다. 따라서 멀티모달 이미지를 입력으로 줌으로써 더 정확하게 Brain Tumor를 예측하려고 하는 것이다.

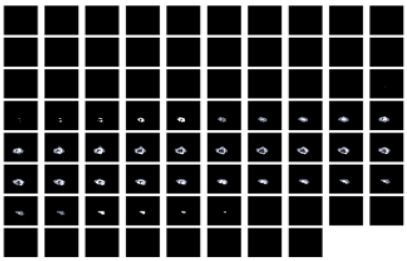
1 Introduction

BraTS 데이터 시각화

- Brain Tumor Data(3D)는 155장의 2D Data로 이루어져 있음
- 3차원 뇌를 155장의 단면으로 표현했다고 생각하면 됨
- 환자마다 H x W x D 의 데이터 (H,W: 해상도<240>, D: 뇌의 단면 개수<155>) 를 다섯 개 지님 (flair, t1, t1ce, t2, seg)



한 환자에 대한 Flair Data 일부 스캔 ($\frac{155}{2}$ Data)



한 환자에 대한 Seg Data 일부 스캔 ($\frac{155}{2}$ Data)

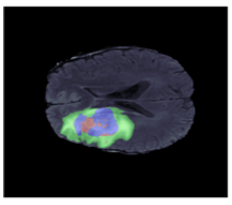
그리고 MRI 영상 자체가 3차원이다 보니까, 실제 데이터는 2D 사진 155장으로 저장이 되어있다. 그래서 3차원 뇌를 155장의 단면으로 표현했다고 생각하면 된다. 그리고 멀티모달 이미지이기 때문에 환자마다 HxWxD를 다섯 개 지닌다고 생각하면 된다. 아래는 시각화한 사진인데 155장의 사진 중에서 절반만큼만 출력한 것이다. 앞에서부터 78장을 차례대로 뽑은 것은 아니고 한 장씩 건너 뛰면서 출력을 한 것이다.

1 Introduction

Color Segmentation Map

- seg에는 어떤 픽셀이 어떤 범주인지에 대한 정보를 지니고 있음 (0~4)
- 이 정보를 이용해서 Color Segmantation Map (240x240x3)을 구성할 수 있음
- 1 = NCR -> Red, 2= ED -> Green, 4 = ET -> Blue

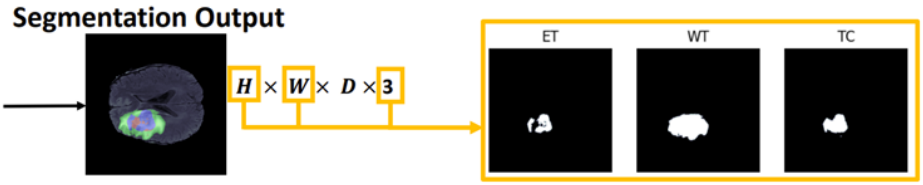
```
flair=flair_data[:,:,80]
seg=seg_data[:,:,80]

color_segmentation=np.zeros((240,240,3))
gray_segmentation=seg

color_segmentation[gray_segmentation == 1] = [255,0,0] # Red (Necrotic Tumor Core, NCR)
color_segmentation[gray_segmentation == 2] = [0,255,0] # Green (Edema, ED)
color_segmentation[gray_segmentation == 4] = [0,0,255] # Blue (Enhancing Tumor, ET)

plt.title('color segmentation map with flair')
plt.axis('off')
plt.imshow(flair,cmap='bone')
plt.imshow(color_segmentation,alpha=0.3,cmap='bone')
```



그리고 seg데이터는 어떤 픽셀이 어떤 범주인지에 대한 정보를 지니고 있다. 그래서 픽셀별로 0~4사이의 숫자를 가지고 있는데 0은 정상, 1은 NCR, 2는 ED, 4는 ET이며 3은 존재하지 않는다. 이 정보를 이용하면 Color Segmentation Map도 구성할 수 있다. 해상도는 240x240이고 뇌종양의 범주 세 개를 각각 RGB로 표현하면, 240x240x3인 정답 컬러 영상을 제작할 수 있다.

1 Introduction

- Segmentation Map for Deep Learning
  - 실제 딥 러닝으로 학습할 때는 (NCR, ED, ET) 범주 대신 (ET, TC, WT) 범주를 사용함
  - 딥 러닝 모델은 ET, WT, TC 에 대응되는 세 개의 채널을 가지는 Segmentation Output을 생성함
  - Brain Tumor Segmentation은 각 채널 별로 Pixel별 Tumor를 예측하는 Task



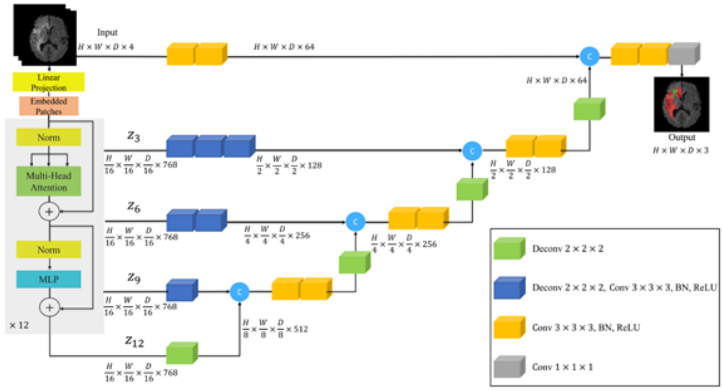
Final segmentation output consists of 3 output channels corresponding to ET,WT and TC sub-regions.

9

그런데 실제로 학습할 때는 ET, TC, WT를 사용한다고 앞에서 설명했다. 따라서 Segmantion Output 그림이 NCR, ED, ET에 대한 컬러 영상으로 표현이 될 지라도, 실제로는 각 채널별로 ET, WT, TC에 대한 Segmantion Map을 예측한다고 생각하면 된다.

2 Architecture

- UNETR
  - 3D UNET의 수축 경로를 Transformer 구조로 대체한 모델
  - Transformer의 특성 상 수축된 Shape이 일정하므로, Skip Connection 단계에서 서로 다른 Deconvolution을 진행함
  - → Attention 연산을 통해 Global 정보를 더 잘 획득하므로, 넓게 분포된 Tumor 영역도 잘 예측함

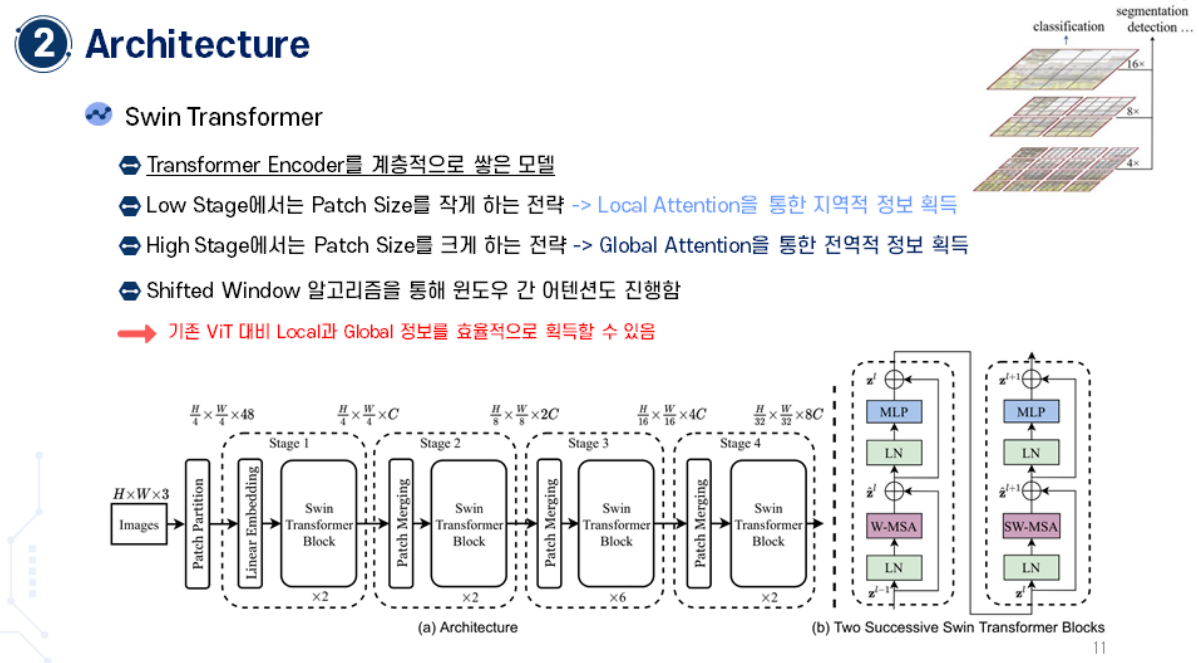


10

이제 본격적으로 모델 아키텍쳐를 소개하겠다. Swin UNETR을 설명하기 전에, UNETR과 Swin Transformer에 대해 간단하게 알아보겠다. 먼저 UNETR은 3D UNET의 수축 경로를 Transformer 구조로 대체한 모델이다. 그런데 수축 경로는 원래 Feature Map의 Resolution이 1/2씩 줄어야되지만, Transformer의 특성 상 Input과 Output Shape이 같으므로 수축된 Shape도 일정하다. 그런데 확장 경로에서는 기존 모델과 똑같이 Upsampling을 해나가기 때문에, Skip Connection 단계에서 Shape

을 맞추기 위해 서로 다른 Deconvolution을 사용한다. 가장 윗단에서는 Deconv를 세 번하고 가장 아랫단에서는 Deconv를 한 번만 하는 방식을 택하였다. 이 모델은 Transformer의 Attention 연산을 통해 Global 정보를 더 잘 획득하므로, 넓게 분포된 Tumor 영역도 잘 예측할 수 있다는 장점을 지닌다.

P. s. 다만 UNet의 취지가 Segmentation을 할 때 효율적으로 local 정보와 global 정보를 이용하자는 것이기 때문에 TransUNET처럼 Transformer를 거치기 전에 CNN으로 local정보를 파악하고 skip connection 단계에서도 CNN과 Transformer의 Feature를 모두 이용하는 것이 더 낫지 않았을까라고 개인적으로 생각한다.



다음은 Swin Transformer이다. 이 모델은 ViT 이후 제시된 모델로 단순하게 생각하면 Transformer Encoder를 계층적으로 쌓았다고 생각하면 된다. 따라서 FPN(Feature Pyramid Network)처럼 계층에 따라 다른 특징을 지니고 있는데, Low Stage에서는 Patch 사이즈를 작게하고, 이러한 작은 Patch들로 이루어진 윈도우 내부에서 Attention을 진행하기 때문에 지역적 정보를 획득할 수 있다. 반대로 High Stage에서는 Patch 사이즈를 크게 하는 전략을 해서 Global Attention(큰 영역에서의 Attention)을 통한 전역적 정보를 획득할 수 있게 하였다. 반면, 기존의 ViT는 항상 Global Attention만 하기 때문에 Local 정보를 획득하기에는 Swin이 훨씬 좋다.

뿐 만 아니라 윈도우 내부에서만 Attention을 하면, 윈도우가 겹치는 구간에서는 특징을 찾을 수 없기 때문에, Shifted Window라는 알고리즘을 통해 윈도우 간 어텐션도 진행하였다.

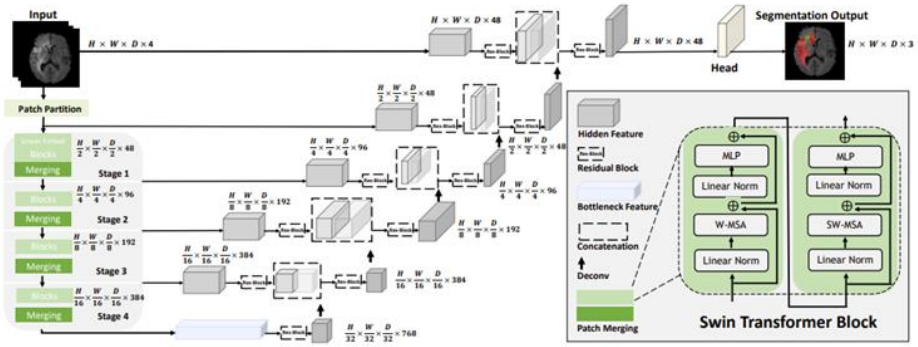
따라서 기존 ViT 대비 Local과 Global 정보를 효율적으로 획득할 수 있게 한 모델이라고 생각하면 된다.

2 Architecture

Swin UNETR

- UNETR에 Swin Transformer를 적용한 모델 (수축 경로에서 사용함)
- CNN을 이용한 수축 경로처럼 각 Stage별로 Output이 줄어들어 바로 Skip Connection이 가능함
  (그러나 논문에서는 ResBlock을 한 번 더 거침)

→ UNETR보다 Local 정보를 더 잘 활용하므로 더욱 정교한 Segmentation이 가능함



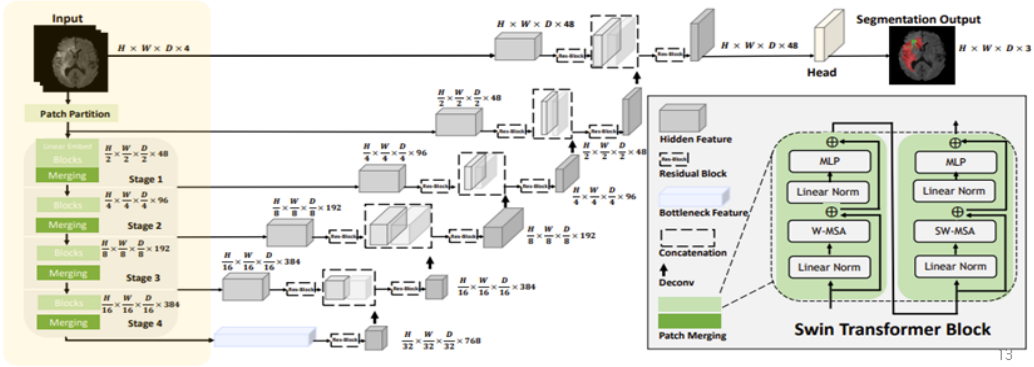
12

Swin UNETR은 UNETR의 수축 경로에서 Swin을 적용한 모델이다. Swin은 Patch Merging이라는 기술이 있기 때문에 각 Stage별로 Output Resolution이 줄어든다. 따라서 별도의 Deconv작업없이 Skip Connection이 가능하다. 그런데 이 논문에서는 ResBlock을 한 번 더 거친 구조를 사용한다. 그래서 이 모델은 Swin을 사용했기 때문에, UNETR보다 Local 정보를 더 잘 활용하므로 더욱 정교한 Segmentation이 가능하다는 장점이 있다. 물론 Global 정보도 유지한채로 Local정보를 활용할 수 있다는 의미이다.

2 Architecture

Contracting Path (수축 경로)

- 점진적으로 넓은 범위를 보며 Feature들을 추출하는 경로 (Local Feature -> Global Feature)
- Swin Transformer를 사용하며 Output Shape이 2배씩 작아짐

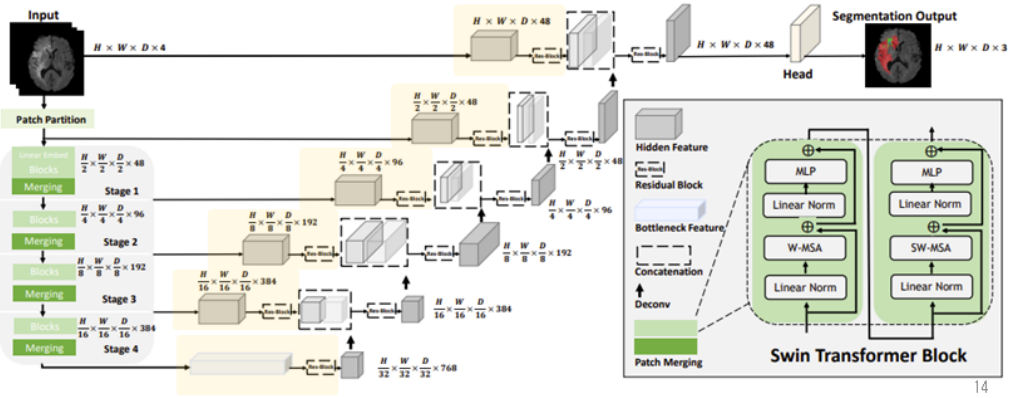


13

좀 더 자세하게 알아보자면, 수축 경로는 점진적으로 이미지의 넓은 범위를 보며 Feature를 추출하는 경로이다. stage1에 가까울수록 Local Feature를 추출하며, stage4에 가까울수록 Global Feature를 추출한다. Swin을 사용하였으며 기존 CNN처럼 Output Shape은 2배씩 작아진다.

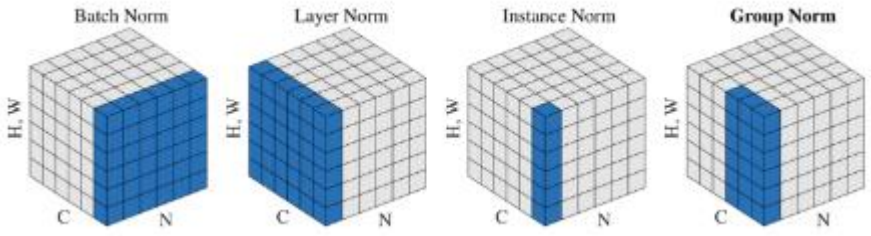
2 Architecture

- Bottle Neck & Connection
  - 수축 경로에서 확장 경로로 전환 & 연결되는 구간
  - [Instance Normalization -> Two 3x3x3 Conv Layers]로 구성된 Res-Block을 거침
    (평균과 표준편차는 batch와 channel과는 무관하게 각 데이터에 대해서만 normalization)



Bottle Neck과 Connection은 각각 수축 경로에서 확장 경로로 전환되는 구간, 연결되는 구간으로, 수축 경로에서 나온 feature를 확장 경로로 전달하는 역할을 한다. 그런데 논문에서는 단순히 전달하지 않고 Instance Normalizaton과 두 개의 3x3x3 Conv layer를 거치는 Res-Block을 추가시켰다. 여기서 Instance Normalization이란 평균과 표준편차를 구할 때 batch와 channel과는 무관하게 구한 뒤 표준화를 하는 것을 의미한다.

(Transformer에서는 LayerNorm을 채택하므로 Batch와 무관하게 평균과 표준편차를 구한다.)

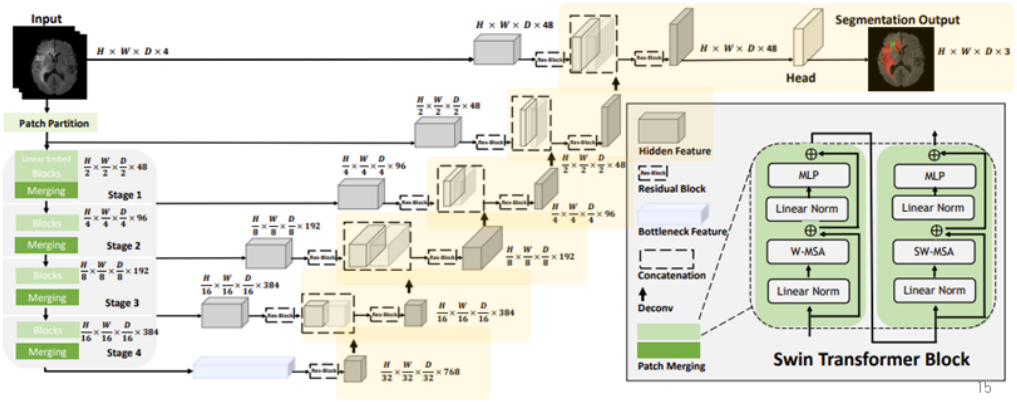


https://wdprogrammer.tistory.com/67

2 Architecture

Expanding Path (확장 경로)

- Global Feature를 앞 단의 Local Feature들과 차례차례 결합하는 경로
- Deconvolution과 Res-Block을 사용하며 Output Shape이 2배씩 커짐



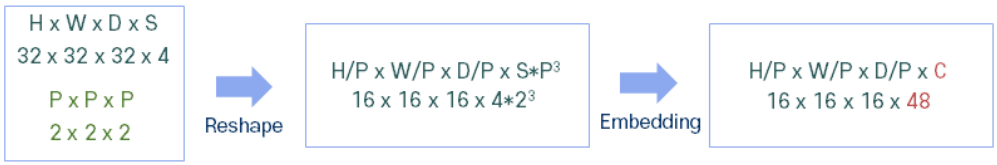
15

확장 경로는 Bottleneck Feature로부터 얻은 Global정보에 앞 단(윗 단)의 Local Feature들을 차례차례 결합하는 경로이다. 이 때 결합을 위해 Shape을 맞춰주기 위해서 Deconvolution을 진행한다. 또한 Res-Block을 사용하며 특징을 다시 한 번 추출한다. 확장 경로의 초반에는 Global Feature정보만 다시 추출하게 되지만, 후반으로 갈수록 Global에 점점 더 정교한 Local Feature들이 결합되면서 Global과 Local 정보가 적절하게 담긴 Feature Map을 형성할 수 있다.

3 Method

Patch Partition & Linear Embedding

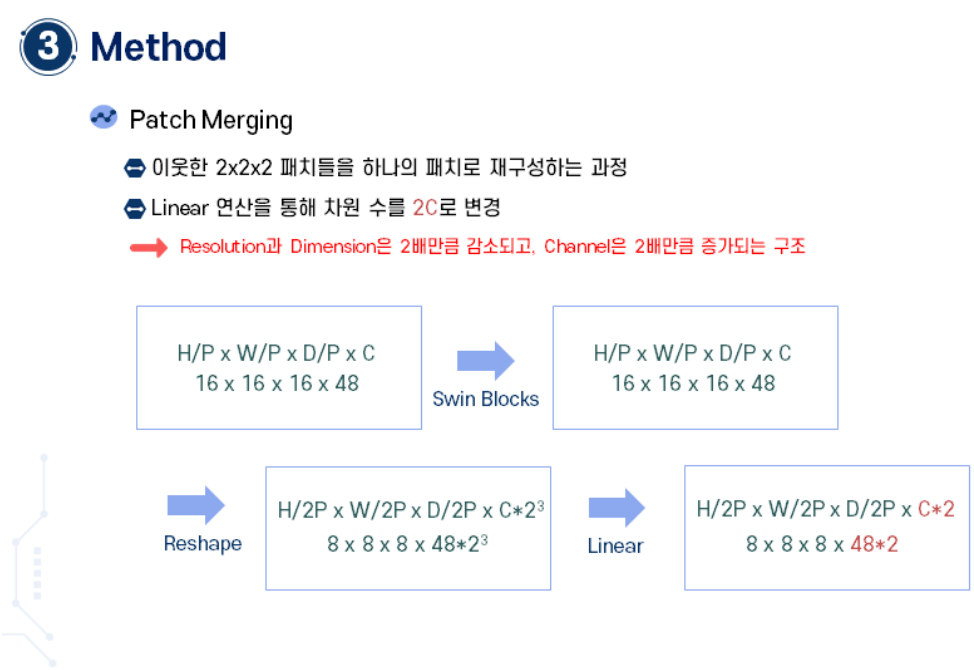
- Patch Partition: Input을 크기가 P인 패치들로 분할 후 변환하는 과정 (Pixel 기준 -> Patch 기준)
- Linear Embedding: 채널축을 C로 재구성하는 과정 (벡터의 차원이 C가 되는 과정)
- 논문에서 초기 패치 크기(P)는 2, C는 48로 주어짐



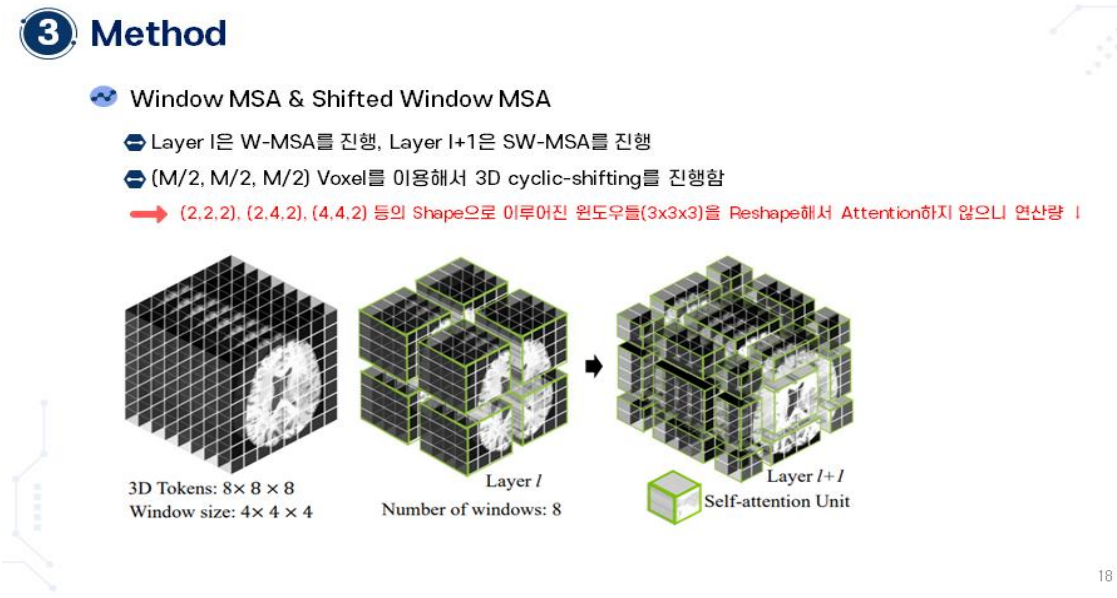
16

지금부터는 Swin에 사용된 기술을 설명하겠다. 먼저 Patch Partition과 Embedding은 Swin Block에 들어오기 전에 거치는 과정이다. Path Partition은 Input을 크기가 P인 패치들로 분할 후 변환하는 과정으로, Input의 각 원소가 픽셀이었다면, 이것을 Patch 기준으로 변환시키는 것이다. 예를 들어

32x32x32인 입력은 2x2x2 패치가 width, height, dimension 방향으로 각각 16개있다고 생각할 수 있다. 따라서 Patch기준으로 Reshape하면 16x16x16이 되는 것이고, Resolution과 Dimension이 줄어든 대신 채널인 S가 늘어나므로 S는 4에서 $4*2^3$이 된다. 또한 여느 Transformer와 다를 바 없이 Embedding을 진행해서 채널을 C로 재구성한다고 한다. 논문에서 C는 48로 주어진다.



다음은 Swin Blocks를 설명하기 전에, Patch Merging부터 설명하겠다. Patch Merging은 이웃한 2x2x2 패치들을 하나의 패치로 재구성하는 과정이다. 따라서 16x16x16인 Shape은 8x8x8이 된다. Patch Partion과 마찬가지로, Resolution과 Dimension이 줄어든 대신 채널인 C가 늘어나므로 C는 48에서 $48*2^3$이 된다. 또한 Linear 연산을 통해 채널은 2C로 다시 변경된다.



다음은 Swin Block에서 사용되는 W-MSA와 SW-MSA에 대한 설명이다. 처음 Input이 32x32x32라

고 가정했기 때문에, 이 그림은 Swin의 stage2인 8x8x8(patch partion과 patch merging을 한 번씩 거친 상태)이라고 생각하면 편하다.

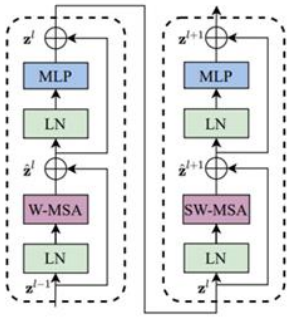
Layer l 그림을 보면, 이전에 8x8x8개의 패치들이 크기가 4인 윈도우들로 재구성이 되서, 윈도우가 총 2x2x2개 만들어진 상태이고, 각 윈도우 내에서 어텐션을 진행하겠다는 의미이다.

Layer l+1에서는 width, height, dimension기준으로 각각 윈도우 개수를 한 개씩 늘려서 크기가 제각각인 윈도우가 총 3x3x3개 만들어지는데, 이렇게 윈도우를 늘리는 이유는 layer l에서 윈도우끼리 겹치는 부분은 어텐션을 계산할 수 없었기 때문이다. 그러나 진짜 이 27개의 윈도우에 대해 어텐션 연산을 모두 진행하면, 크기가 제각각인 윈도우들을 다시 (4,4,4)로 Reshape해야 되기 때문에 연산량이 엄청나게 늘어난다. 따라서 크기가 (M/2,M/2,M/2)인 Voxel(윈도우)을 우측 하단으로 Shift시키는 방식으로 결국 Layer l과 같은 Shape을 구성한다. 그럼 결국 크기가 (4,4,4)인 윈도우 8개로 어텐션을 진행할 수 있는 것이다.

3 Method

Swin Transformer Blocks

- 윈도우 내 어텐션(W-MSA)과 윈도우 간 어텐션(SW-MSA)을 순차적으로 수행하는 구조
- 원본 Transformer와 달리 LayerNorm을 먼저 수행함



$$\hat{z}^l = \text{W-MSA}(\text{LN}(z^{l-1})) + z^{l-1}$$
$$z^l = \text{MLP}(\text{LN}(\hat{z}^l)) + \hat{z}^l$$
$$\hat{z}^{l+1} = \text{SW-MSA}(\text{LN}(z^l)) + z^l$$
$$z^{l+1} = \text{MLP}(\text{LN}(\hat{z}^{l+1})) + \hat{z}^{l+1}.$$

19

앞서 설명한 W-MSA와 SW-MSA를 다루는 모듈이 Swin Transformer Blocks이다. 이 Blocks은 두 개의 Block으로 구성되며 한 개는 W-MSA를 하고 한 개는 SW-MSA를 한다. 또한 수식을 보면 원본 Transformer와 달리 LayerNorm을 먼저 수행한다는 점에서 차이가 있다.

3 Method

- Swin UNETR configurations
  - 실제 윈도우 사이즈(M)는 7을 이용함
  - Stage별 Number of Blocks는 유지되지만 Number of Heads는 점점 늘어남
  - ViT-Base (Layer가 12개)의 Params가 86M인 것을 고려하면 효율적인 연산을 수행함

Embed Dimension	Feature Size	Number of Blocks	Window Size	Number of Heads	Parameters	FLOPs
768	48	[2,2,2,2]	[7,7,7]	[3,6,12,24]	61.98M	394.84G

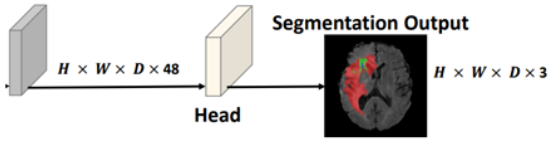
Table 1. Swin UNETR configurations.

ppt랑 논문에서는 이해를 위해 윈도우 사이즈를 4로 지정하였는데, 실제 코드에서는 7을 이용했다고 한다. 또한 Stage별 Number of Blocks는 2로 유지가 되고, Number of Heads는 점점 늘어나는 구조를 채택했다. 파라미터는 61M으로, ViT-Base에 비하면 효율적인 모델이라고 말할 수 있다.

3 Method

- Residual Block
  - Res-Block은 전환 구간, 연결 구간, 확장 경로에 사용됨
  - Instance Normalization Layer와 두 개의 3x3x3 Conv Layer로 구성됨

- Final Segmentation Output
  - Head는 1x1x1 Convolution으로 이루어짐
  - 이후 Sigmoid Activation Function을 이용함 (0 ~ 1사이의 값으로 예측함)



다음은 전환구간, 연결구간, 확장 경로에서 사용된 Residual Block을 다시 알아보겠다. 이 Block은 Instance Normalization Layer와 두 개의 3x3x3 Conv Layer로 구성된다.

모델의 Output같은 경우, Head Module을 거쳐 채널을 3으로 재구성하고, 활성화 함수로 Sigmoid를 채택하면서 각 채널에서 픽셀별로 Tumor Score를 0~1사이의 값으로 예측하게 된다.

3 Method

Loss Function

- Soft Dice Loss를 사용함
- Dice Loss = 1- Dice Score
- Dice Score: 정밀도와 재현율을 한 번에 표현한 점수 (2∗|G ∩ Y| / |G|+|Y|)
- G = Ground Truth, Y = Predicted, I = Voxel 수, J = Class 수

$$\mathcal{L}(G,Y) = 1 - \frac{2}{J}\sum_{j=1}^{J}\frac{\sum_{i=1}^{I} G_{i,j}Y_{i,j}}{\sum_{i=1}^{I} G_{i,j}^2 + \sum_{i=1}^{I} Y_{i,j}^2}.$$

모델에 사용되는 Loss Function은 Soft Dice Loss를 사용한다. Dice Loss는 1-Dice Score라고 생각할 수 있고, Dice Score는 정밀도와 재현율을 한 번에 표현한 점수로 1에 가까울수록 GT와 Predicted가 유사하다고 생각하면 된다. 파란색으로 나와있는 수식이 Dice Score 공식인데, Swin UNETR같은 경우, 여러 클래스도 고려해야 되기 때문에, 클래스마다 계산한 Dice Score를 모두 더한 뒤 클래스 개수인 J로 나누는 방식을 채택하고 있다. 그리고 3D input과 output이기 때문에 픽셀마다 처리하지 않고, Dimension 축으로 픽셀들을 묶어서 만든 복셀(Voxel) 단위로 계산을 하는 것 같다.

4 Experiments

Results

- BraTS 2021 Training Dataset으로 five-fold cross-validation를 진행함 → Fold 2가 가장 성능이 좋음
- 평균 Dice Score가 모든 범주 (ET, WT, TC) 에서 가장 높음

	Swin UNETR				nnU-Net				SegResNet				TransBTS			
Dice Score	ET	WT	TC	Avg.	ET	WT	TC	Avg.	ET	WT	TC	Avg.	ET	WT	TC	Avg.
Fold 1	**0.876**	**0.929**	**0.914**	**0.906**	0.866	0.921	0.902	0.896	0.867	0.924	0.907	0.899	0.856	0.910	0.897	0.883
Fold 2	**0.908**	**0.938**	**0.919**	**0.921**	0.899	0.933	**0.919**	0.917	0.900	0.933	0.915	0.916	0.885	0.919	0.903	0.902
Fold 3	**0.891**	**0.931**	**0.919**	**0.913**	0.886	0.929	0.914	0.910	0.884	0.927	0.917	0.909	0.866	0.903	0.898	0.889
Fold 4	**0.890**	**0.937**	**0.920**	**0.915**	0.886	0.927	0.914	0.909	0.888	0.921	0.916	0.908	0.868	0.910	0.901	0.893
Fold 5	**0.891**	**0.934**	**0.917**	**0.914**	0.880	0.929	**0.917**	0.909	0.878	0.930	0.912	0.906	0.867	0.915	0.893	0.892
Avg.	**0.891**	**0.933**	**0.917**	**0.913**	0.883	0.927	0.913	0.908	0.883	0.927	0.913	0.907	0.868	0.911	0.898	0.891

- BraTS 2021 Validation Dataset으로 Dice Score(↑)와 Hausdorff Distance(↓)를 계산함 (벤치마킹 서버에서 진행)

	Dice			Hausdorff (mm)		
Validation dataset	ET	WT	TC	ET	WT	TC
Swin UNETR	0.858	0.926	0.885	6.016	5.831	3.770

실험 결과이다. 보통 Segmentation을 할 때는 k fold cross validation을 진행하는 듯하다. 따라서 k를 5로 두고 실험해본 결과, fold2에서 가장 성능이 높게 나왔다.
또한 벤치마킹 서버에서 Dice Score와 Hausdorff 거리도 계산을 했는데, 이것도 SOTA인 수치라고 한다.

4 Experiments

Dice Score

- 정밀도와 재현율을 한 번에 표현하는 F-Score와 같은 지표, 1에 가까울 수록 Predicted와 GT가 유사함

$$DSC = \frac{2|X \cap Y|}{|X| + |Y|} = \frac{2 \times TP}{(TP + FP) + (TP + FN)}$$

White Segmentation Map:
GT (X)

Green Segmentation Map:
Predicted (Y)

23

마지막으로 Segmentation에 사용되는 평가지표에 대해 설명을 하겠다. 먼저 Dice Score는 정밀도와 재현율을 한 번에 나타내는 F-Score와 완전히 같은 지표이다. 1에 가까울수록 Predicted와 GT가 유사하다고 생각하면 된다. 그런데 이것을 시각화한 자료로 따져보면, 흰 색 손이 GT고 초록색 영역이 Predicted라고 할 때, 1에 가까울수록 초록색 영역은 손의 계형을 따른다고 이해할 수 있다. 그리고 공식을 보면, IoU와 유사하기 때문에, 그냥 IoU처럼 겹치는 비중이 많을수록 분자인 교집합이 커져 더 정확하게 예측했다고 생각하면 된다.

4 Experiments

Hausdorff Distance

- 두 집합 간의 거리를 결정하는 방식, 0에 가까울수록 Predicted 집합(X)과 GT 집합(Y)간의 거리가 작음
- 다른 집합에서 가장 가까운 점까지의 거리들 중 최댓값을 계산함 (가장 가까운 점은 상대적일 수 있음)

$$d_H(X,Y) = max(d_{XY}, d_{YX}) = max\left\{ \max_{x \in X} \min_{y \in Y} d(x,y), \max_{y \in Y} \min_{x \in X} d(x,y) \right\}$$

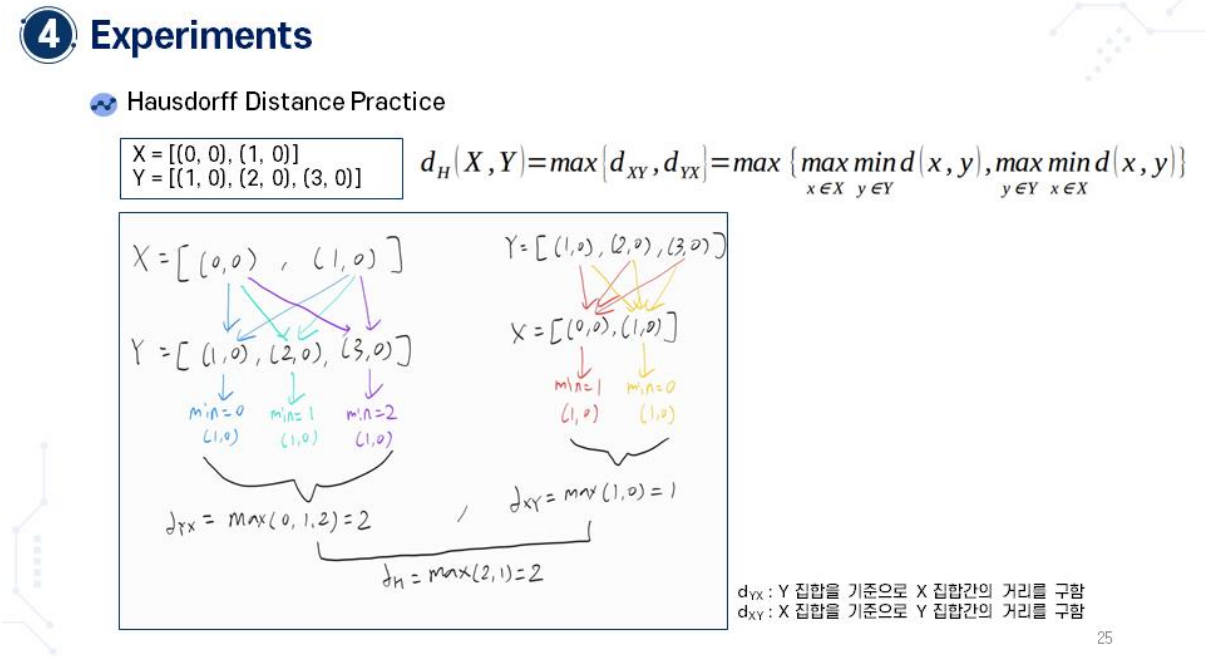
```
Max [
   Max [ Min [ Distance( x, y ) for y in Y ] for x in X ],
   Max [ Min [ Distance( x, y ) for x in X ] for y in Y ]
]
```

① y는 고정한 채로, x를 변경하면서 Distance(x,y)의 Min 값을 계산
② 위 과정을 모든 y에 대해 진행함
③ 모든 Min 값에 대해 Max 값을 계산함 -> d_{YX}
④ d_{XY}와 d_{YX}중 큰 값을 d_H로 선정함

24

다음은 Hausdorff 거리이다. Predicted 영역과 GT 영역을 마치 집합인 것처럼 생각하고, 두 집합 간의 거리가 작을수록 정확하게 예측했다고 따지는 것이다. 두 집합간의 거리를 결정할 때는 다

른 집합에서 가장 가까운 점까지의 거리들 중 최댓값을 이용하는데, 공식은 아래와 같다. 이 공식을 수도코드로 따져보면 1~4의 설명과 같다. 그런데 이 부분은 한 번 연습을 해보면서 설명을 하겠다.



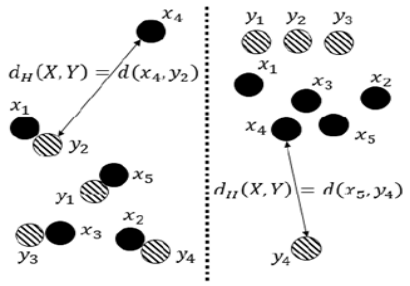
X집합과 Y집합이 위와 같을 때 Hausdorff 거리를 구하는 방법이다.
먼저 d_{YX}부터 계산을 해보겠다. 처음에는 Y를 (1,0)으로 고정한채로 X를 변경하면서 Distance의 최솟값을 계산하는 것이다. 계산 결과, (1,0)은 X가 (1,0)일 때 Distance가 가장 낮게 나온다. 이 과정을 모든 Y에 대해 진행해보면, 각각 최솟값은 0,1,2가 나온다. 이렇게 나온 최솟값들 중에서 가장 큰 값이 d_{YX}가 되는 것이다. 그리고 d_{XY}는 X를 고정한채로 똑같이 계산하면 된다.
그렇게 이렇게 나온 d_{YX}와 d_{XY}는 각각 2,1이 되는데, 여기서 최댓값이 Hausdorff 거리가 된다.

필자가 이 거리를 한 문장으로 표현하면 '다른 집합에서 가장 가까운 점까지의 거리들 중 최댓값'이라고 말했다. d_{YX}를 기준으로 이 문장을 다시 곱씹어보면, 다른 집합(X)에서 가장 가까운 점은 (1,0)이 되고, 그 거리들은 0,1,2가 된다. 여기서 가장 큰 값은 2가 된다. 그런데 이 값은 Y 집합을 기준으로 X 집합간의 거리를 계산한 것이기 때문에, 반대로, X집합을 기준으로 Y 집합간의 거리를 계산한 d_{XY}도 계산함으로써 두 값들 중 더 큰 값을 채택하는 것이다.

그리고 이 집합의 원소는 1차원 공간에 표현이 되기 때문에, 다른 집합에서 가장 가까운 점이 한 개로 좁혀지지만, 만약 2차원 이상이면 가장 가까운 점은 여러 개가 될 수도 있다. 그 예시가 다음 슬라이드와 같다.

4 Experiments

- Hausdorff Distance Problem
  - Outlier에 취약하다
  - 왼쪽과 오른쪽의 X,Y 집합 간의 d_H는 Outlier(x_4,y_4)를 제외하면 왼쪽이 훨씬 작음, 포함해서 계산하면 왼쪽이 더 큼
    - Distance를 계산할 때 0.95를 곱해서 Outlier에 덜 민감하도록 개선함 (95% HD)



X집합과 Y집합간의 거리를 계산한다고 할 때, x1과 가장 가까운 점은 y2고, x5와 가장 가까운 점은 y1이다. 이런 식으로 다른 집합에서 가장 가까운 점이 여러 개 있을 수 있다. 그래서 집합의 각 원소마다 다른 집합의 가장 가까운 점(원소)까지의 거리를 계산하고 이렇게 나온 거리들 중에서 최댓값을 채택한다고 생각하면 된다. 그런데 이 슬라이드의 제목에서도 보다시피 문제점이 있다. 바로 Outlier에 취약하다는 점인데, 원래 outlier인 x4와 y4를 제외하면, 왼쪽과 오른쪽 그림에서 hausdorff 거리는 왼쪽이 훨씬 작아야 정상이다. 그러나 outlier가 나오면서 왼쪽의 housdorff 거리는 $d(x_4,y_2)$가 되기 때문에, 왼쪽이 더 커지는 문제가 발생한 것이다.

그래서 Segmentation에서 hausdorff 거리를 평가 지표로 사용할 때는 Distance에 0.95를 곱하는 방식인 95% HD를 사용한다. 이렇게 하면 전체적으로 Distance가 작아지기 때문에 Outlier에 덜 민감해질 수 있다.

5 Reference

- Swin UNETR: Swin Transformers for Semantic Segmentation of Brain Tumors in MRI Images
  - https://arxiv.org/pdf/2201.01266.pdf
- Dice Score
  - http://machinelearningkorea.com/2019/07/13/%ED%8F%89%EA%B0%80%EC%A7%80%ED%91%9C-dice/
- Hausdorff Distance
  - https://structseg2019.grand-challenge.org/Evaluation/
  - https://npclinic3.tistory.com/7
  - https://dhpark1212.tistory.com/entry/Hausdorff-Distance

레퍼런스는 다음과 같다. 이상으로 발표를 마치겠다.