Open Software Project

**Assignment 11 Technical Report**

1976235 오진솔

1. Original U-net

|  |
| --- |
| #UNet.py  def \_\_init\_\_(self, in\_channels, out\_channels):  super(Unet, self).\_\_init\_\_()   *########## fill in the blanks (Hint : check out the channel size in practice lecture 15 ppt slides 5-6)* self.convDown1 = conv(in\_channels, 64)  self.convDown2 = conv(64, 128)  self.convDown3 = conv(128, 256)  self.convDown4 = conv(256, 512)  self.convDown5 = conv(512, 1024)  self.maxpool = nn.MaxPool2d(2, stride=2)  self.upsample = nn.Upsample(scale\_factor=2, mode=**'bilinear'**, align\_corners=True)  self.convUp4 = conv(1536, 512)  self.convUp3 = conv(768, 256)  self.convUp2 = conv(384, 128)  self.convUp1 = conv(192, 64)  self.convUp\_fin = nn.Conv2d(64, out\_channels, 1) |
| UNet 구조에 맞게 input\_channel, output\_channel 값 삽입 |
| forward 중 expansive path 부분  x = self.upsample(conv5) x = torch.cat([x, conv4], dim =1) *#######fill in here #######*  x = self.convUp4(x) x = self.upsample(x) x = torch.cat([x, conv3], dim=1) *#######fill in here #######*  x = self.convUp3(x) x = self.upsample(x) x = torch.cat([x, conv2], dim=1) *#######fill in here #######*  x = self.convUp2(x) x = self.upsample(x) x = torch.cat([x, conv1], dim=1) *#######fill in here #######*  x = self.convUp1(x) out = self.convUp\_fin(x) |
| torch.cat을 이용해서 두 결과를 합침 |

|  |
| --- |
| #modules.py  def train\_model(trainloader, model, criterion, optimizer,scheduler, device):  model.train()  for i, (inputs, labels) in enumerate(trainloader):  from datetime import datetime  inputs = inputs.to(device)  labels = labels.to(device=device, dtype=torch.int64)  criterion = criterion.cuda()  *##########################################  ############# fill in here (10 points) -> train  ####### Hint :  ####### 1. Get the output out of model, and Get the Loss* outputs = model(inputs)  loss = criterion(outputs, labels)   *####### 3. optimizer* optimizer.zero\_grad()   *####### 4. backpropagation* loss.backward()  optimizer.step()  scheduler.step()  *#########################################* |
| Getting model & loss, set optimizer, backpropagation |

|  |
| --- |
| def get\_loss\_train(model, trainloader, criterion, device):   model.eval()  total\_acc = 0  total\_loss = 0  for batch, (inputs, labels) in enumerate(trainloader):  with torch.no\_grad():  inputs = inputs.to(device)  labels = labels.to(device = device, dtype = torch.int64)  inputs = inputs.float()  *##########################################  ############# fill in here (5 points) -> (same as validation, just printing loss)  ####### Hint :  ####### Get the output out of model, and Get the Loss  ####### Think what's different from the above* outputs = model(inputs)  loss = criterion(outputs, labels)  *#########################################* outputs = np.transpose(outputs.cpu(), (0,2,3,1))  preds = torch.argmax(outputs, dim=3).float()  acc = accuracy\_check\_for\_batch(labels.cpu(), preds.cpu(), inputs.size()[0])  total\_acc += acc  total\_loss += loss.cpu().item()  return total\_acc/(batch+1), total\_loss/(batch+1) |
| train 의 loss를 구하는 과정 |

|  |
| --- |
| #get\_loss\_train 중 일부  inputs = inputs.to(device) labels = labels.to(device=device, dtype=torch.int64) *########################################## ############# fill in here (5 points) -> (validation) ####### Hint : ####### Get the output out of model, and Get the Loss ####### Think what's different from the above* outputs = model(inputs) loss = criterion(outputs, labels) *#########################################* |
| 위의 것과 다른 점은 잘 모르겠습니다. |
| #val\_model 중 일부  inputs = inputs.to(device) labels = labels.to(device=device, dtype=torch.int64) *########################################## ############# fill in here (5 points) -> (validation) ####### Hint : ####### Get the output out of model, and Get the Loss ####### Think what's different from the above* outputs = model(inputs) loss = criterion(outputs, labels) *#########################################* |
| 마찬가지로 다른 점을 잘 모르겠습니다. |

|  |
| --- |
| #segmentation mask  *########################################### ############# fill in here (10 points) ####### Hint : ####### convert segmentation mask into r,g,b (both for image and predicted result) ####### image should become temp\_rgb, result should become temp\_label ####### You should use cls\_invert[]* if temp\_l[j][k] == 0 :  temp\_rgb[j][k] = (img[j][k][0].item()\*100, img[j][k][1].item()\*100, img[j][k][2].item()\*100)  else :  temp\_rgb[j][k] = cls\_invert[temp\_l[j][k]]  for i in range (3):  if temp\_rgb[j][k][i] < 0 :  temp\_rgb[j][k][i] = 0  elif temp\_rgb[j][k][i] > 255 :  temp\_rgb[j][k][i] = 255  temp\_label[j][k] = cls\_invert[temp\_l[j][k]] *#########################################* |
| temp\_rgb 에는 해당 픽셀에 라벨이 있을 경우 라벨을, 없을 경우 원본 이미지를 출력함  (원본 이미지 위에 라벨을 그린 모습)  temp\_label 에는 라벨만 출력함 |

|  |
| --- |
| #main.py  *##### fill in here ##### ##### Hint : Initialize the model (Options : UNet, resnet\_encoder\_unet)* model = Unet(in\_channels = 3,out\_channels = 22) PATH = **'../trained\_model/UNet\_trained\_model.pth'** *#model = UNetWithResnet50Encoder() #PATH = '../trained\_model/resnet\_encoder\_unet.pth' #########################################################################* |
| Unet과 resnet\_encoder\_unet 중 하나를 선택해서 사용.  UNet의 경우에는 (3,22)로 초기화한다.  checkpoint 경로를 지정한다. |
| *# Loss Function ##### fill in here -> hint : set the loss function ##### done* criterion = nn.CrossEntropyLoss()  *# Optimizer ##### fill in here -> hint : set the Optimizer ##### done* optimizer = torch.optim.Adam(model.parameters(), lr=learning\_rate) scheduler = StepLR(optimizer, step\_size=4, gamma=0.1) |
| loss function은 Cross Entropy Loss를 사용하고,  Optimizer 는 Adam을 사용한다. |
| *##### fill in here ##### ##### Hint : load the model parameter, which is given* checkpoint = torch.load(PATH, map\_location = device) model.load\_state\_dict(checkpoint) |
| checkpoint를 불러온다. |
| if epoch % 4 == 0:  savepath2 = savepath1 + str(epoch) + **".pth"** *##### fill in here ##### done  ##### Hint : save the model parameter* torch.save(model.state\_dict(), savepath2) |
| model parameter를 저장한다. |

UNet result

|  |  |
| --- | --- |
| CUDA out of memory error 때문에 부득이하게 batch size = 1로 두고 실행하였습니다. | |
| result | label |
|  |  |
|  |  |
|  | |

2. resnet\_encoder\_unet

|  |
| --- |
| if self.downsample:  self.layer = nn.Sequential(  *##########################################  ############## fill in here  # Hint : use these functions (conv1x1, conv3x3)  #########################################* conv1x1(in\_channels, middle\_channels, 2, 0),  conv3x3(middle\_channels, middle\_channels, 1, 1),  conv1x1(middle\_channels, out\_channels, 1, 0)  )  self.downsize = conv1x1(in\_channels, out\_channels, 2, 0)  else:  self.layer = nn.Sequential(  *##########################################  ############# fill in here  #########################################* conv1x1(in\_channels, middle\_channels, 1, 0),  conv3x3(middle\_channels, middle\_channels, 1, 1),  conv1x1(middle\_channels, out\_channels, 1, 0)  )  self.make\_equal\_channel = conv1x1(in\_channels, out\_channels, 1, 0) self.activation = nn.ReLU(inplace=True) |
| convolution block을 생성하는 과정.  Assignment 10 과 같음. |
| def forward(self, x, with\_output\_feature\_map=False): *#256* out1 = self.layer1(x)  out1, indices = self.pool(out1)  out2 = self.layer2(out1)  out3 = self.layer3(out2)  x = self.bridge(out3) *# bridge* x = self.UpConv1(x)  x = torch.cat([x, out2], dim =1)*#######fill in here ####### hint : concatenation (Practice Lecture slides 6p)* x = self.UnetConv1(x)  x = self.upconv2\_1(x, output\_size=torch.Size([x.size(0),256,64,64]))  x = self.upconv2\_2(x)  x = torch.cat([x, out1], dim=1) *######## hint : concatenation (Practice Lecture slides 6p)* x = self.UnetConv2\_2(x, output\_size=torch.Size([x.size(0), 128, 128, 128]))  x = self.UnetConv2\_2(x, output\_size=torch.Size([x.size(0), 128, 256, 256]))  x = self.UnetConv2\_3(x)  x = self.UnetConv3(x)  return x |
| forward 과정. Torch.cat을 이용하여 두 결과를 합침 |

Result

|  |  |
| --- | --- |
|  | |
| result | label |
|  |  |
|  |  |
|  | |