UNet_Arch_Saurabh

June 25, 2020

```
In [17]: import torch
         import torch.nn as nn


         # convulution network created with ReLU activation funtion


         def double_conv(in_c,out_c):
           conv = nn.Sequential(
               nn.Conv2d(in_c,out_c,kernel_size=3),
               nn.ReLU(inplace=True),
               nn.Conv2d(out_c,out_c,kernel_size=3),
               nn.ReLU(inplace=True)

           )
           return conv


         def crop_image(tensor,target_tensor):
           target_size= target_tensor.size()[2]
           tensor_size= tensor.size()[2]
           delta = tensor_size-target_size
           delta= delta//2
           return tensor[:,:,delta:tensor_size-delta,delta:tensor_size-delta]    # take all bat


         #model creation

         class UNet(nn.Module):
           def __init__(self):
             super(UNet,self).__init__()

             self.maxpool_2x2 = nn.MaxPool2d(kernel_size=2, stride =2) # intializing max pooli
             self.down_conv_1 = double_conv(1,64)         # initializing 5 conv layers
             self.down_conv_2= double_conv(64,128)
             self.down_conv_3 = double_conv(128,256)
             self.down_conv_4 = double_conv(256,512)
             self.down_conv_5 = double_conv(512,1024)
```

```
        self.up_trans_1 = nn.ConvTranspose2d(in_channels=1024,out_channels=512,
                                             kernel_size=2,stride=2)
        self.up_conv_1= double_conv(1024,512)
        self.up_trans_2 = nn.ConvTranspose2d(in_channels=512,out_channels=256,
                                             kernel_size=2,stride=2)
        self.up_conv_2= double_conv(512,256)
        self.up_trans_3 = nn.ConvTranspose2d(in_channels=256,out_channels=128,
                                             kernel_size=2,stride=2)
        self.up_conv_3= double_conv(256,128)

        self.up_trans_4 = nn.ConvTranspose2d(in_channels=128,out_channels=64,
                                             kernel_size=2,stride=2)
        self.up_conv_4= double_conv(128,64) # In the end you will get a 64 channel image

         #at the you have output layer

        self.out =nn.Conv2d(in_channels=64,out_channels=2,
                            kernel_size=1)

    # forward pass for feed forward neural network

        def forward(self,image):

        #batch_size,c,h,w
        #encoder
        # Each of the Conv layer is followed by maxpooling.

        X1= self.down_conv_1(image) #
       # print(X1.size())             #Here after 1st convulation our image size is 568 X
        X2= self.maxpool_2x2(X1)
        X3= self.down_conv_2(X2) #
        X4= self.maxpool_2x2(X3)
        X5= self.down_conv_3(X4) #
        X6= self.maxpool_2x2(X5)
        X7= self.down_conv_4(X6) #
        X8= self.maxpool_2x2(X7)
        X9= self.down_conv_5(X8)
       # print(X9.size())             #here after 5th convulation our image size is 28X28

        #decoder

        x= self.up_trans_1(X9)
        y=crop_image(X7,x)
        x= self.up_conv_1(torch.cat([x,y],1))

        #print(X7.size())    # we have to concatenate with X9 but the size is different t
        #print(y.size())
        x= self.up_trans_2(x)
        y=crop_image(X5,x)
        x= self.up_conv_2(torch.cat([x,y],1))

        x= self.up_trans_3(x)
        y=crop_image(X3,x)
        x= self.up_conv_3(torch.cat([x,y],1))

        x= self.up_trans_4(x)
        y=crop_image(X1,x)
        x= self.up_conv_4(torch.cat([x,y],1))

        x=self.out(x)
        print(x.size())
        return x




    if __name__ == "__main__":
      image = torch.rand((1,1,572,572))
      model = UNet()
      print(model(image))
```

```
torch.Size([1, 2, 388, 388])
tensor([[[[0.0777, 0.0769, 0.0780,  ..., 0.0780, 0.0794, 0.0808],
          [0.0783, 0.0749, 0.0808,  ..., 0.0776, 0.0772, 0.0776],
          [0.0782, 0.0802, 0.0762,  ..., 0.0817, 0.0762, 0.0767],
          ...,
          [0.0766, 0.0805, 0.0795,  ..., 0.0742, 0.0772, 0.0766],
          [0.0806, 0.0805, 0.0797,  ..., 0.0808, 0.0783, 0.0773],
          [0.0758, 0.0786, 0.0791,  ..., 0.0812, 0.0819, 0.0820]],

         [[0.1142, 0.1133, 0.1058,  ..., 0.1105, 0.1095, 0.1080],
          [0.1096, 0.1081, 0.1079,  ..., 0.1099, 0.1112, 0.1090],
          [0.1104, 0.1137, 0.1091,  ..., 0.1104, 0.1103, 0.1117],
          ...,
          [0.1090, 0.1080, 0.1086,  ..., 0.1082, 0.1089, 0.1057],
          [0.1103, 0.1098, 0.1065,  ..., 0.1100, 0.1113, 0.1105],
          [0.1133, 0.1138, 0.1088,  ..., 0.1117, 0.1134, 0.1104]]]],
       grad_fn=<MkldnnConvolutionBackward>)
```

In [ ]: