Tex代在D/常思洋/tex代码中 名字是unet求解pde

一个文件为data.py :该文件是用来将训练数据转化为.npy 格式文件，便于网络使用数据。当然还有其他功能如数据增强等，程序中已经进行了解释。

另外一个文件：unet.py :该文件为unet网络结构定义与测试代码。

Unet\_parts.py 中的类

DoubleConv

Down

Up

OutConv 网络结构

Unet\_module.py 中的类

UNet

layer.py 中的类



unetCov2 unetUp

unet\_add.py 中的类

UNet\_Nested 嵌套的U-Net 里面会涉及到通道数 filters=[64,128,256,,512,1024]

如果使用以下输入权初始化策略，深度网络将会更早地收敛：

1.为给定层上的权值矩阵创建一个张量，并用从标准正态分布中随机选择的数字填充它。

2.将每个随机选择的数字乘以√2/√n，其中n是从上一层的输出(也称为“扇入”)进入给定层的连接数。

3.偏置张量初始化为零。

我们可以按照这些方向实现我们自己版本的kaiming初始化，并验证如果在我们假设的100层网络的所有层上使用ReLU，那么它确实可以防止激活输出爆炸或消失。  
def kaiming(m,h):

return torch.randn(m,h)\*math.sqrt(2./m)

x=torch.randn(512)

for I in range(100):

a=kaiming(512,512)

x=relu(a @ x)

x.mean(),x.std()

tool文件夹  
mytransforms.py

类：class MinMaxNormalize(object) 最小最大归一化  
class RandomFlip(object) 随机翻转（水平和垂直），对给定PIL图像，概率为0.5

class RandCrop\_overlap(object) 随机剪切，且是有覆盖的剪切

class RandCrop(object) 随机剪切

class Normalize(object): 最大值归一化  
class PadXYZ\_8(object) 对图片和标签在x，y，z 方向添加黑边。以便整除delta

为的是方便最后对训练图片测试

class PadZ\_8(object):对图片和标签在z 方向添加黑边。以便整除delta。为的是方便最后对训练图片测试，输入的image，label都是三个维度

class PadXYZ\_16(object):对图片和标签在x，y，z 方向添加黑边。以便整除delta。为的是方便最后对训练图片测试，输入的image，label都是三个维度  
class ResizeXoY(object): 对xoy平面插值，z轴不变  
class RandCutZ(object): z 轴剪切然后对 xoy 平面插值

class RandCutX(object): x 轴剪切然后对 zoy 平面插值

class Compose(object): 联合起来

# class ToTensor(object):

# """

# 将标签和图片转为pytorch的张量

# dataset 里面会转，这里转了不好对标签放缩

# """

# def \_\_call\_\_(self, image, label):

# image = torch.from\_numpy(img).float()

# label = torch.from\_numpy(np.array(label, dtype=np.int32)).long()

# return [image, label]

# Trainer.py

def print\_term\_txt(message, out\_txt):

# 训练函数

def train(args, model, device, criterion, optimizer, epoch, train\_loader):

return loss.item(), x, y, name, output.detach(), pred, Dices

# 测试函数

def test(args, model, device, criterion, epoch, test\_loader):

return batch\_loss, x, y, name, output.detach(), pred, Dices

# 保存预测结果. 2D 和 3D 输入有区别

# 这个是针对2D设计的

def save\_2D\_predict(args, save\_dir, model, device, data\_loader, color\_dict):

# 保存测试集或者训练集.预测结果.原图加边界，预测，标签，误差

# 自动创建文件夹

# batch szie 可以设置为 1

# 针对网络预测输出超过一个通道来做的，一个通道的不能做

# Layers.py

定义了两个类unetConv2，unetUp

# unetConv2(nn.Module)

def \_\_init\_\_(self, in\_size, out\_size, is\_batchnorm, n=2, ks=3, stride=1, padding=1):

判断是否批处理归一化，若是则conv结构为 nn.Conv2d nn.BatchNorm2d nn.ReLu

若不进行，则conv为 nn.Conv2d nn.ReLu   
使用kaiming方法初始化权重矩阵

def forward(self, inputs):

return x

# unetUp(nn.Module)

def \_\_init\_\_(self, in\_size, out\_size, is\_deconv, n\_concat=2)

判断是否转置卷积，若是则使用nn.ConvTranspose2d  
若不是，则为nn.UpsamplingBilinear2d nn.Conv2d 初始化权重矩阵

def forward(self, high\_feature, \*low\_feature):

return self.conv(outputs0)

# utils.py 初始化模型 weight bias

# unet\_model.py

一个UNet类，有定义通道数与类别数 这里是模型和损失修改的地方

# unet\_main.py

给出一些参数的值  
轮回数、最小像素点、类别数、学习率、三个路径（模型、输出、根目录），txt文件（训练、测试、val、）、轮回数的最大值、批处理尺寸、最后的模型、训练用的设备、最后的输出的txt文件、损失的名字，测试的dice的名字、信息。