FCN

全卷积神经网络将中间层转置卷积成输入尺寸的原因

- **保持空间对应关系**
- **逐像素分类**
- **多尺度信息融合**
- **端到端的学习方式**

输入图像先进入神经网络取特征，然后通过1×1卷积保持特征的情况下将通道数转化为类别数。

再通过转置卷积变化为输入图像尺寸。

双线性插值

将图片放大最常用的上采样方法。

首先将输出图像上的一点(x,y)映射到输入图像的对应映射点(x’,y‘)上，

然后寻找(x’,y‘)的四个最近的像素，通常是(x’,y‘)、(x’+1,y‘)、(x’,y‘+1)、(x’+1,y‘+1)

输出图像在坐标(x,y)上的像素依据输入图像上这4个像素及其与(x’,y’)的相对距离来计算。

如果我们用(x', y')表示输入图像在点(x', y')处的像素值，那么输出图像在点(x, y)处的像素值(x, y)可以表示为：

$$I(x,y) = (1-(x'-\lfloor x'\rfloor)) \times (1-(y'-\lfloor y'\rfloor)) \times I(\lfloor x'\rfloor, \lfloor y'\rfloor) + (x'-\lfloor x'\rfloor)\times \\ (1-(y'-\lfloor y'\rfloor)) \times I(\lfloor x'\rfloor+1, \lfloor y'\rfloor) + (1-(x'-\lfloor x'\rfloor)) \times (y'-\lfloor y'\rfloor) \times I(\lfloor x'\rfloor, \lfloor y'\rfloor+1) + \\ (x'-\lfloor x'\rfloor) \times (y'-\lfloor y'\rfloor) \times I(\lfloor x'\rfloor+1, \lfloor y'\rfloor+1)$$

其中$\lfloor x'\rfloor$和$\lfloor y'\rfloor$表示坐标向下取整。

Code

代码建议看看论文自己再撸一遍

```python
import torch.nn as nn
import torch.nn.functional as F
import torch
import torchvision
from d2l import torch as d2l
from torchvision.models import ResNet18_Weights

# torchvision中pretrained=True参数已经被废弃了，所以使用weights参数
pretrained_net = torchvision.models.resnet18(weights=ResNet18_Weights.IMAGENET1K_V1)
print(list(pretrained_net.children())[-3:])
# 最后一层是线性，倒数第二层是maxpooling
net = nn.Sequential(*list(pretrained_net.children())[:-2])

X = torch.rand(size=(1, 3, 320, 480))
print(net(X).shape)

num_classes = 21
net.add_module('final_conv', nn.Conv2d(512, num_classes, kernel_size=1))
net.add_module('transpose_conv',
               nn.ConvTranspose2d(
                 num_classes, num_classes,
                 kernel_size=64, stride=32, padding=16))
# Stride=32:上采样卷积操作中有一部分重叠, 提高精度, 简称上采样32倍

# 双线性插值
def bilinear_kernel(in_channels, out_channels, kernel_size):
    factor = (kernel_size + 1) // 2
    if kernel_size % 2 == 1:
        center = factor - 1
    else:
        center = factor - 0.5
    og = (torch.arange(kernel_size).reshape(-1, 1),
          torch.arange(kernel_size).reshape(1, -1))
    filt = (1 - torch.abs(og[0] - center) / factor) * \
           (1 - torch.abs(og[1] - center) / factor)
    weight = torch.zeros((in_channels, out_channels, kernel_size, kernel_size))
    weight[range(in_channels), range(out_channels), :, :] = filt
    return weight

# 定义一个转置卷积层, 使用双线性插值初始化卷积核, 放大图像尺寸
conv_trans = nn.ConvTranspose2d(3, 3, kernel_size=4, padding=1, stride=2, bias=False)
conv_trans.weight.data.copy_(bilinear_kernel(3, 3, 4))
img = torchvision.transforms.ToTensor()(d2l.Image.open('Images/微信图片
_20240823151107.jpg'))
X = img.unsqueeze(0)
Y = conv_trans(X)

# C, H, W -> H, W, C
out_img = Y[0].permute(1, 2, 0).detach()
d2l.set_figsize()
print('input image shape:', img.permute(1, 2, 0).shape)
d2l.plt.imshow(img.permute(1, 2, 0))
print('output image shape:', out_img.shape)
d2l.plt.imshow(out_img)

W = bilinear_kernel(num_classes, num_classes, 64)
net.transpose_conv.weight.data.copy_(W)
batch_size, crop_size = 32, (320, 480)
train_iter, test_iter = d2l.load_data_voc(batch_size, crop_size)

# 定义损失函数, 使用交叉熵损失, 并在特征图上按通道进行平均
def loss(inputs, targets):
    return F.cross_entropy(inputs, targets, reduction='none').mean(1).mean(1)

num_epochs, lr, wd, devices = 5, 0.001, 1e-3, d2l.try_all_gpus()

trainer = torch.optim.SGD(net.parameters(), lr=lr, weight_decay=wd)

d2l.train_ch13(net, train_iter, test_iter, loss, trainer, num_epochs, devices)

def predict(img):
    X = test_iter.dataset.normalize_image(img).unsqueeze(0)
    pred = net(X.to(devices[0])).argmax(dim=1)
    return pred.reshape(pred.shape[1], pred.shape[2])

def label2image(pred):
    colormap = torch.tensor(d2l.VOC_COLORMAP, device=devices[0])
    X = pred.long()
    return colormap[X, :]

# 正常是需要这么下载并解压的，但是我这里下载不了，所以我把数据集下载到了本地，并把路径改成了
本地路径
# 下载链接为http://host.robots.ox.ac.uk/pascal/VOC/voc2012/VOCtrainval_11-May-2012.tar
# d2l.DATA_HUB['voc2012'] = (d2l.DATA_URL + 'VOCtrainval_11-May-2012.tar',
#                            '4e443f8a2eca6b1dac8a6c57641b67dd40621a49')
#
# voc_dir = d2l.download_extract('voc2012','VOCdevkit/VOC2012')
voc_dir = 'D:\\data\\voc_dir\\VOCdevkit\\VOC2012'

test_images, test_labels = d2l.read_voc_images(voc_dir, False)

n, imgs = 4, []

for i in range(n):
    crop_rect = (0, 0, 320, 480)
    X = torchvision.transforms.functional.crop(test_images[i], *crop_rect)
    pred = label2image(predict(X))
    imgs += [X.permute(1, 2, 0), pred.cpu(),
torchvision.transforms.functional.crop(test_labels[i], *crop_rect).permute(1, 2, 0)]

d2l.show_images(imgs[::3] + imgs[1::3] + imgs[2::3], 3, n, scale=2)
d2l.plt.show()
```