今天阅读了两篇笔记，https://cs231n.github.io/convolutional-networks/ 和 https://cs231n.github.io/transfer-learning/ 。

第一篇介绍了卷积神经网络的相关知识，了解了三种主要的层：卷积层、池化层和全连接层。对每个层的功能有了一定的了解。通过笔记中卷积过程的动图演示，了解了一个7 * 7 * 3 的输入图片通过卷积变成 3 * 3 * 2的输出的过程。

阅读第二篇笔记是因为在PyTorch的两个图像方面的教程中反复提到了迁移学习，并在教程中给出了笔记地址。学习了有关迁移学习的内容，了解了其必要性，以及迁移学习的使用方式。

跟着PyTorch官网上Image下的TORCHVISION OBJECT DETECTION FINETUNING教程，编写了简历数据集的代码，但还没有全部完成。

由于之前有过PyTorch的使用经验，因此今天就开始了代码的编写。但是不知为何不能在Cuda上进行计算。

Table of Contents:

阅读时遇到的不理解的词

ConvNet： Convolutional Network

spatial size： 指图片的尺寸，应该指宽、高、深度。

卷积神经网络

“

- *INPUT [32x32x3] will hold the raw pixel values of the image, in this case an image of width 32, height 32, and with three color channels R,G,B.*
- *CONV layer will compute the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input volume. This may result in volume such as [32x32x12] if we decided to use 12 filters.*

为何使用了12个卷积核，深度变成12?

“

1. *First, the **depth** of the output volume is a hyperparameter: it corresponds to the number of filters we would like to use, each learning to look for something different in the input. For example, if the first Convolutional Layer takes as input the raw image, then different neurons along the depth dimension may activate in presence of various oriented edges, or blobs of color. We will refer to a set of neurons that are all looking at the same region of the input as a **depth column** (some people also prefer the term fibre).*

输出的深度是多少，Filter的数量就是多少。每一个Filter对图像（feature map）进行卷积操作后，得到一个32*32的图像（activation map），用不同filter得到的activation map堆叠起来，得到32*32*12的输出。

卷积过程,输入和输出大小关系

$(W-F+2 P) / S+1$，这个公式得到的是输出图像的二维大小。例如7*7的输入，用3*3的filter，步长1，pad0，卷积后，得到的输出大小是5 * 5。

W —— 输入的每个二维 feature map 大小是 W * W

F —— Filter(Kernel) 大小为 F * F

P —— 填充的大小，输入图像四周各往外延伸P

S —— 步长，Filter 移动的步长

参数共享

一个filter在移动过程中，filter的参数不变。例如3 * 3 * 1的filter，在某个卷积某层使用其对一个图进行卷积的过程中，9个参数的值不变。
https://www.zhihu.com/question/47158818/answer/670431317 中详细介绍了参数共享。

Transfer_Learning

在PyTorch官网的教程中，很多次提到了这个词。

https://cs231n.github.io/transfer-learning 中对Transfer Learning进行了说明，解释了其必要性。

问题：很少有人能够拥有庞大的数据集对完全随机的初始网络参数进行训练。同时，在我看来从0开始训练，也需要耗费很长的时间，并且效果不确定。

其主要含义是使用预训练好的卷积模型，这个卷积模型已经有很好的特征提取能力。

三种主要使用方式

ConvNet as fixed feature extractor： 删除原有全连接层，将其余的网络用于特征提取，自定义全连接层，选择需要的特征产生输出。我认为这是最简单的一种使用方式。

Fine-tuning the ConvNet： 使用新数据继续训练预训练的模型，通过反向传播修改网络权值。

Pretrained models： 没太看懂，似乎只是分享模型参数，让其他人根据这些这些发布的参数调整自己的模型参数。

使用建议

同时，文档中给出了微调模型的建议，主要依据是新数据集的大小，以及新数据集与原始数据集是否相似。

TORCHVISION_OBJECT_DETECTION_FINETUNING项目训练数据集构造

项目地址： https://pytorch.org/tutorials/intermediate/torchvision_tutorial.html#torchvision-object-detection-finetuning-tutorial

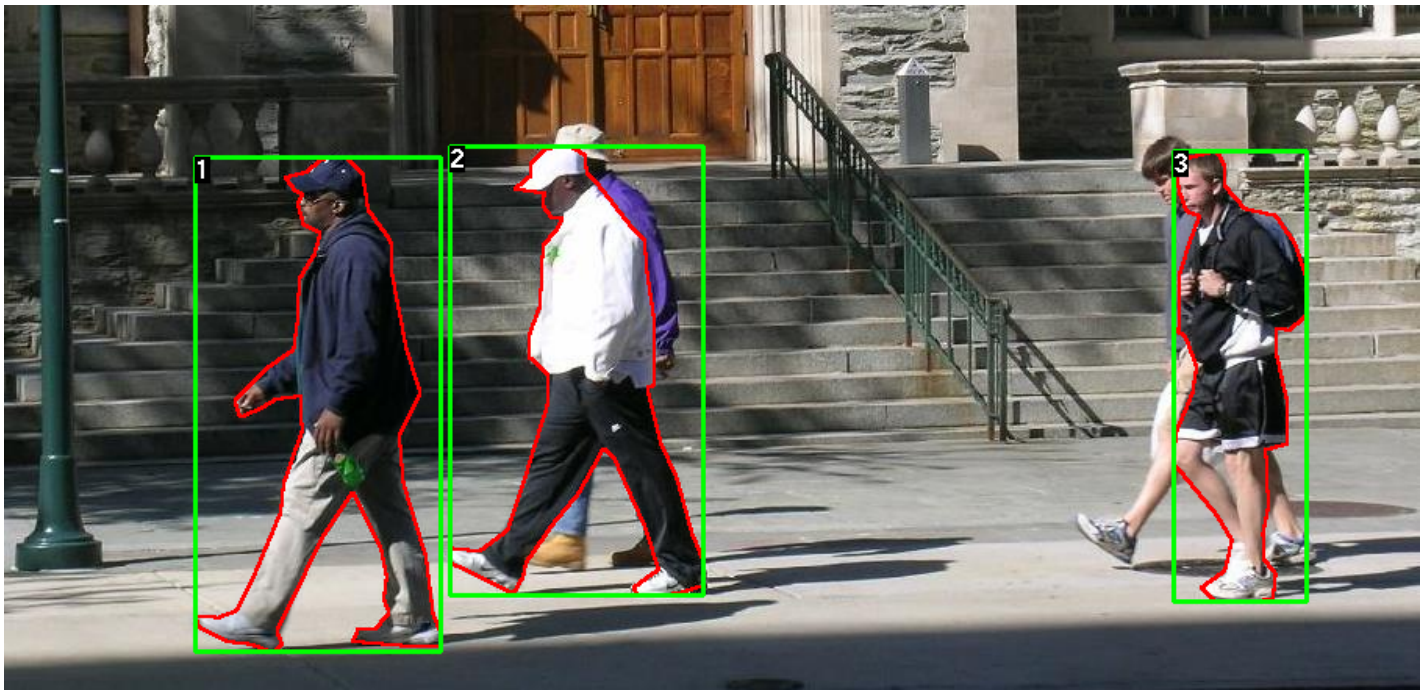
get_transform函数

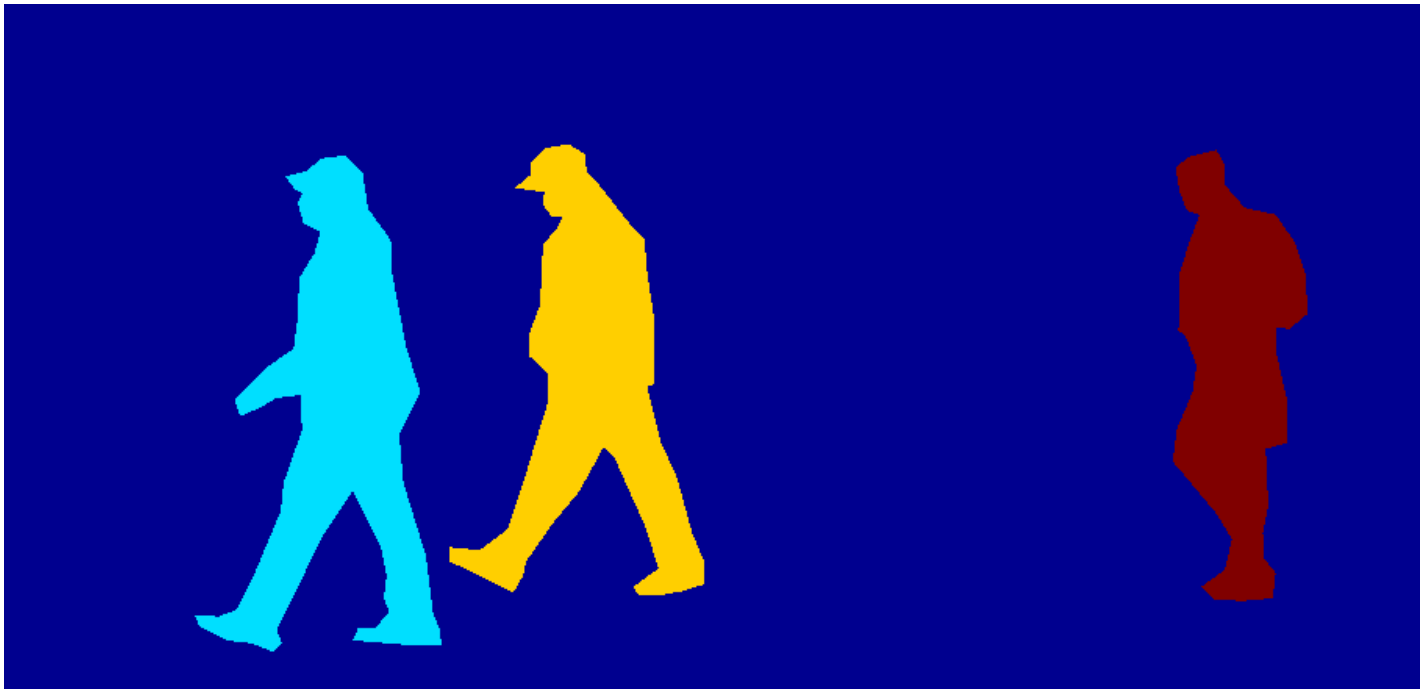
```
def get_transform(train):
    transforms = []
    transforms.append(T.ToTensor())
    if train:
        transforms.append(T.RandomHorizontalFlip(0.5))
    return T.Compose(transforms)
```

RandomHorizontalFlip(p) : 依概率对图片进行翻转。

掩码图片处理

示例项目提供的数据集中每张彩色图片有其对应的掩码图片





1.将掩码图片以如下代码形式打开后：

```
mask = Image.open(mask_path)# convert the PIL Image into a numpy array
mask = np.array(mask)
'''
[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
 '''
```

得到一个二维数组，除人外的背景对应0，目标1对应的背景是1，目标2对应的背景是2，目标3对应的背景是3：

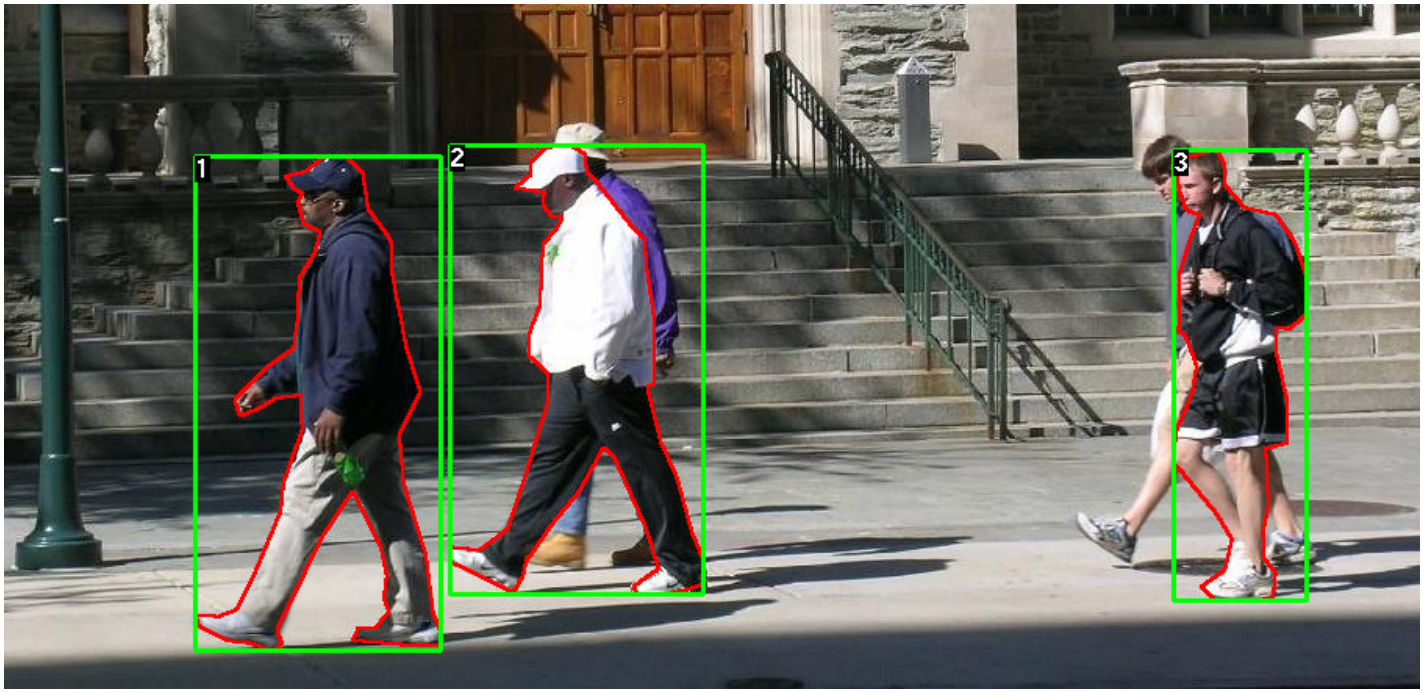
2.因此又使用了如下代码，得到了三个目标对应的id：

```
obj_ids = np.unique(mask)
# [0 1 2 3]
obj_ids = obj_ids[1:]
# [1 2 3]
```

3.去除原掩码图片中保存的目标id信息，即背景是0，人物是1。

```
masks = mask == obj_ids[:, None, None]
'''
[[[False False False ... False False False]
  [False False False ... False False False]
  [False False False ... False False False]
  ...
  [False False False ... False False False]
  [False False False ... False False False]
  [False False False ... False False False]]
 [[False False False ... False False False]
  [False False False ... False False False]
  [False False False ... False False False]
  ...
  [False False False ... False False False]
  [False False False ... False False False]
  [False False False ... False False False]]]
'''
```

4.将目标圈出来，xmin, xmax, ymin, ymax是方框的四个坐标：




```
for i in range(num_objs):
    pos = np.where(masks[i])
    xmin = np.min(pos[1])
    xmax = np.max(pos[1])
    ymin = np.min(pos[0])
    ymax = np.max(pos[0])
    boxes.append([xmin, ymin, xmax, ymax])
```

待解决问题

transforms函数的作用与使用

在__getitem__函数最后，调用了这个函数，应该是对图片进行翻转，提高训练集的多样性。但是输入参数是错误的。

不能使用gpu进行计算

torch.cuda.is_available()返回false。