**DANN算法解读**

## Paper

特征提取器用于将源域和目标域的特征全部提取出来，然后根据提取的特征分别对域和标签种类进行预测，然后根据预测结果反向传播将提取域特征的参数减弱，将特征标签的提取器的参数加强

学习一个模型能够学习一个好的可迁移特征，满足两个条件：

1. Domain-invariance - 面对这些特征，你无法区分它们是来自目标域还是源域。
2. Discriminativeness - 利用这些特征，你可以很好的完成分类任务。

## Code

1. **torch.backends.cudnn.deterministic = True**

这个用于提升计算的时间，防止每次都在随机数里面搜索结果

1. **torchvision.transforms**

是pytorch中的图像预处理包。一般用Compose把多个步骤整合到一起

主要是有对图像进行处理，然后将图像转化为tensor后对数据进行处理的函数操作。

1. **torch.utils.data.DataLoader**

主要用于将输入数据转化为pytorch训练中可以用的数据（该接口主要用来将自定义的数据读取接口的输出或者PyTorch已有的数据读取接口的输入按照batch size封装成Tensor）

**4）iter（）函数用法**

iter()函数用来返回指定对象的迭代器，有两种用法：iter(iterable)和iter(callable, sentinel)，前者要求参数必须为序列或者有自己的迭代器，后者会持续调用参数callable直至其返回sentinel。

譬如给列表加iter后调用next可以实现按顺序取值 。

**5）输入模型训练的数据形式**

[-0.4242, -0.4242, -0.4242, ..., -0.4242, -0.4242, -0.4242]]]]), tensor([7, 6, 0, 6,

这也是预处理为pytorch训练用的数据调用出来的格式

**6）err.backward()函数**

**7）sys.stdout.write（）用法**

不仅可以作为输出还可以进行返回字符串长度

**8）打印出的dataloader\_source dataloader\_target 形式**

<torch.utils.data.dataloader.DataLoader object at 0x000002334956DAF0>

<torch.utils.data.dataloader.DataLoader object at 0x000002334C888E80>

**9）assert dataset\_name in ['MNIST', 'mnist\_m']**

断言可以在条件不满足程序运行的情况下直接返回错误，而不必等待程序运行后出现崩溃的情况，例如我们的代码只能在 Linux 系统下运行，可以先判断当前系统是否符合条件。

**10）add\_module(name, module)**

方法: add\_module(name, module)

Adds a child module to the current module. 将一个子模块添加到当前模块中.

The module can be accessed as an attribute using the given name.

通过给定的名字,我们就可以以访问属性的方式来访问该模块.

Parameters 参数

name (string) – name of the child module. The child module can be accessed from this module using the given name

module (Module) – child module to be added to the module.

# 11). nn.BatchNorm2d

BatchNorm（批规范化）主要是为了加速神经网络的收敛过程以及提高训练过程中的稳定性。通常用于解决多层神经网络中间层的协方差偏移(Internal Covariate Shift)问题，类似于网络输入进行零均值化和方差归一化的操作，不过是在中间层的输入中操作而已。使一批（Batch）feature map满足均值为0，方差为1的分布规律。这样不仅数据分布一致，而且避免发生梯度消失。

# 12）nn.MaxPool2d

**输出尺寸=(输入尺寸-filter尺寸+2\*padding)/stride+1**

**class torch.nn.MaxPool2d(kernel\_size, stride=None, padding=0, dilation=1, return\_indices=False, ceil\_mode=False)**

kernel\_size(int or tuple)：取最大值（max pooling）的窗口大小

**stride(int or tuple, optional)：**窗口移动的步长。默认值为kernel\_size（上一个参数我们确定了滑动窗口的大小，现在我们来确定这个窗口如何进行滑动。如果不指定这个参数，那么默认步长跟最大池化窗口大小一致。如果指定了参数，那么将按照我们指定的参数进行滑动。

**padding (int or tuple, optional)：**输入的每一条边补充0的层数

**dilation (int or tuple, optional)：**控制窗口中元素步长的参数

**return\_indices：**如果为True，会返回输出最大值的位置索引，对于上采样（torch.nn.MaxUnpool2d）操作会有帮助。

**ceil\_mode：** 如果等于True，计算输出信号大小的时候，会使用向上取整，代替默认的向下取整的操作。

**13）nn.ReLU(True)：**

inplace = True 时,会修改输入对象的值,所以打印出对象存储地址相同

nplace = False 时,不会修改输入对象的值,而是返回一个新创建的对象

inplace = True ,会改变输入数据的值,节省反复申请与释放内存的空间与时间,只是将原来的地址传递,效率更好

**14）nn.Linear**

PyTorch的nn.Linear（）是用于设置网络中的**全连接层的**，**需要注意在二维图像处理的任务中，全连接层的输入与输出一般都设置为二维张量，形状通常为[batch\_size, size]，不同于卷积层要求输入输出是四维张量**。

# 15）nn.Conv2d

out\_size = （in\_size - K + 2P）/ S +1

**K为kernel\_size的设置值，P为padding的设置值，S为stride的设置值**

**class torch.nn.Conv2d(in\_channels, out\_channels, kernel\_size, stride=1, padding=0, dilation=1, groups=1, bias=True,padding\_mode='zeros')**

**in\_channels(int)**：输入图像的channel（通道数），例如，RGB图像通道数为3

**out\_channels(int)：** 输出图像（特征层）的channel

**kernel\_size(int or tuple)**：kernel（卷积核）的大小，kennel\_size=5，意味着卷积大小（5,5）/5×5，kennel\_size=（2,3），意味着卷积大小（2，3）/2×3 ，即非正方形卷积

**stride(int or tuple，optional)：** 卷积的步长，默认为1，stride=2,意味着步长上下左右扫描皆为2， stride=（2,3），左右扫描步长为2、上下为3

**padding(int or tuple，optional)：**四周pad的大小，默认为0，在卷积之前补0，四周都补0，

**dilation(int or tuple，optional)**： kernel元素间的距离，默认为1（dilation翻译为扩张，有时候也称为“空洞”1）

**groups(int ，optional)：**将原始输入channel划分成的组数，默认为1

**bias(bool，optional)**：如果是True，则输出的bias可学，默认为True。卷积后是否加偏移量

**padding\_mode：**默认为“zeros”，填充0

**16）view函数**

view()的作用相当于numpy中的reshape，重新定义矩阵的形状。

view中一个参数定为-1，代表动态调整这个维度上的元素个数，以保证元素的总数不变

**17) torch.nn.Dropout**

可以看到torch.nn.Dropout对所有元素中每个元素按照概率0.5更改为零，   
而torch.nn.Dropout2d是对每个通道按照概率0.5置为0，

18) model.eval()

在模型中，我们通常会加上Dropout层和batch normalization层，在模型预测阶段，我们需要将这些层设置到预测模式，model.eval()就是帮我们一键搞定的，如果在预测的时候忘记使用model.eval()，会导致不一致的预测结果。

**19)pytorch实现梯度反转层(Gradient Reversal Layer)**