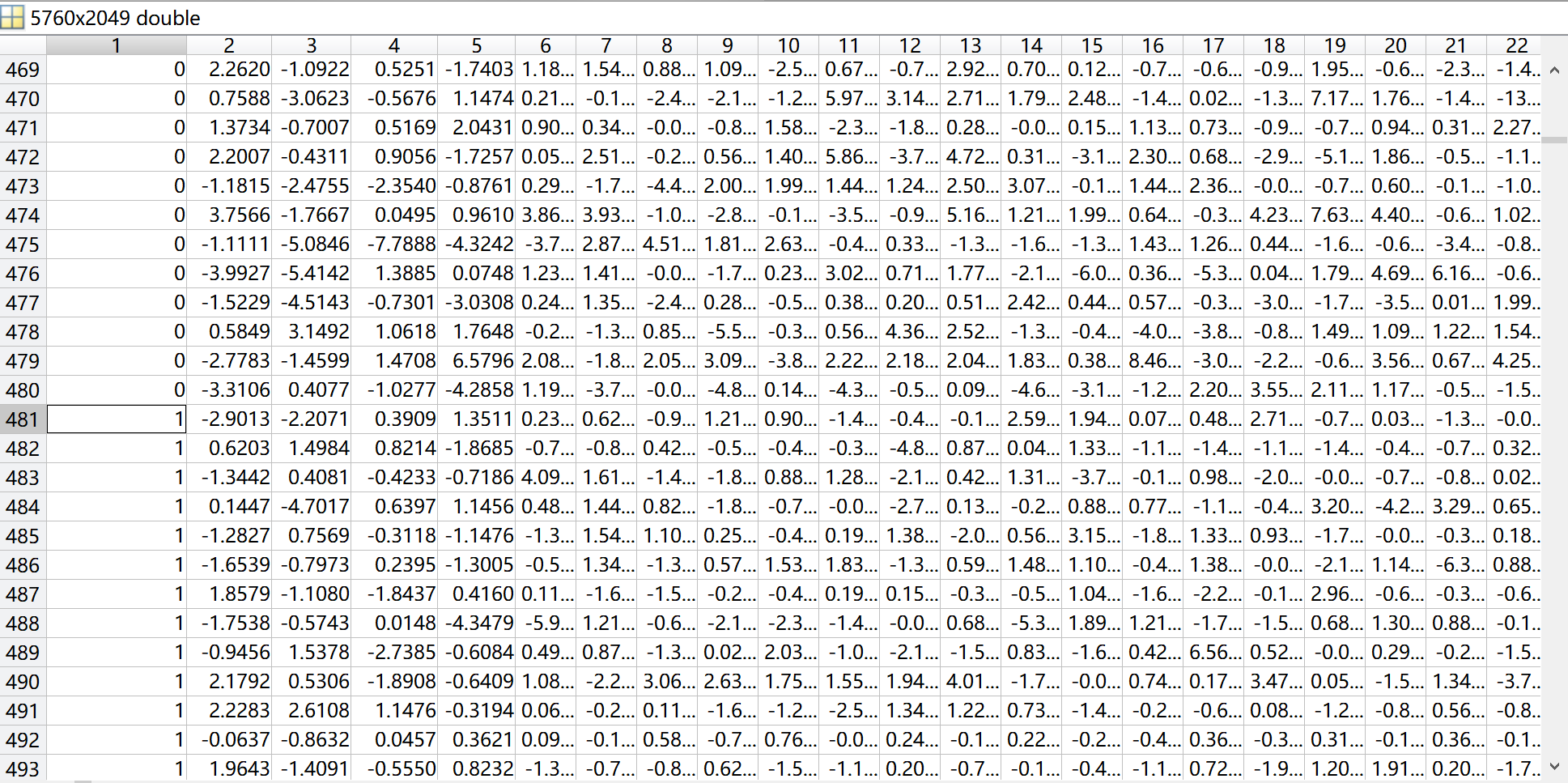
**环境配置：cuda11.1 + cuDNN 8.0.5 + Python3.7**

**数据集介绍：12类特征数据**

**数据来源于定轴齿轮箱采集振动信号，通过一定的数据预处理方法，得到训练集(Train\_datasets)和测试集（Test\_datasets），这里为了跑通迁移学习模型，暂将训练集作为源域数据集（source），测试集作为目标域数据集（target）。**

1. **源域数据集是.mat文件格式，每一行表示一个样本，标签在第一列，其他列都是数据（特征向量）。每类特征数据样本480个，所以源域数据集的大小为【480\*12 = 5760,2049】**
2. **目标域数据集中每类特征数据样本120个，数据集大小为【120\*12 = 1440,2049】**

****

**图1 源域数据集数据**

**数据集的导入，**这里是通过Tensordataset和DataLoader生成数据迭代器。

Conv1d(in\_channels=Ci, out\_channels=Co, kernel\_size=K, stride=s))

对于输入网络的数据:

input=torch.randn(N,Ci,Li)

网络参数的结构：

Weight [Co,Ci,K]

Bias [Co]

网络输出与参数和输入之间的关系：

Output [N,Co,Lo] , 其中Lo = [(Li-(K-1)+1)/s]+1 向下取整

1.数据来源于一维时间序列信号，数据和标签保存在同一文件下。

根据pytorch中一维卷积函数conv1d()的对数据的要求，需要对一维数据进行增维, 使数据通道变为1。

**输入数据的大小： data[batch\_size,channels,length]**

**X=torch.from\_numpy(datas[:,1:]).type(torch.FloatTensor)**

**data=X.unsqueeze(1)**

2.基于pytorch中的多分类交叉熵计算，需要标签为OneHot目标向量对应的数字。

criterion = nn.CrossEntropyLoss()

loss\_c = criterion(y\_src, target)

**y\_src：[batch\_size,classes] 源域数据预测输出**

**target：[batch\_size] 源域标签**

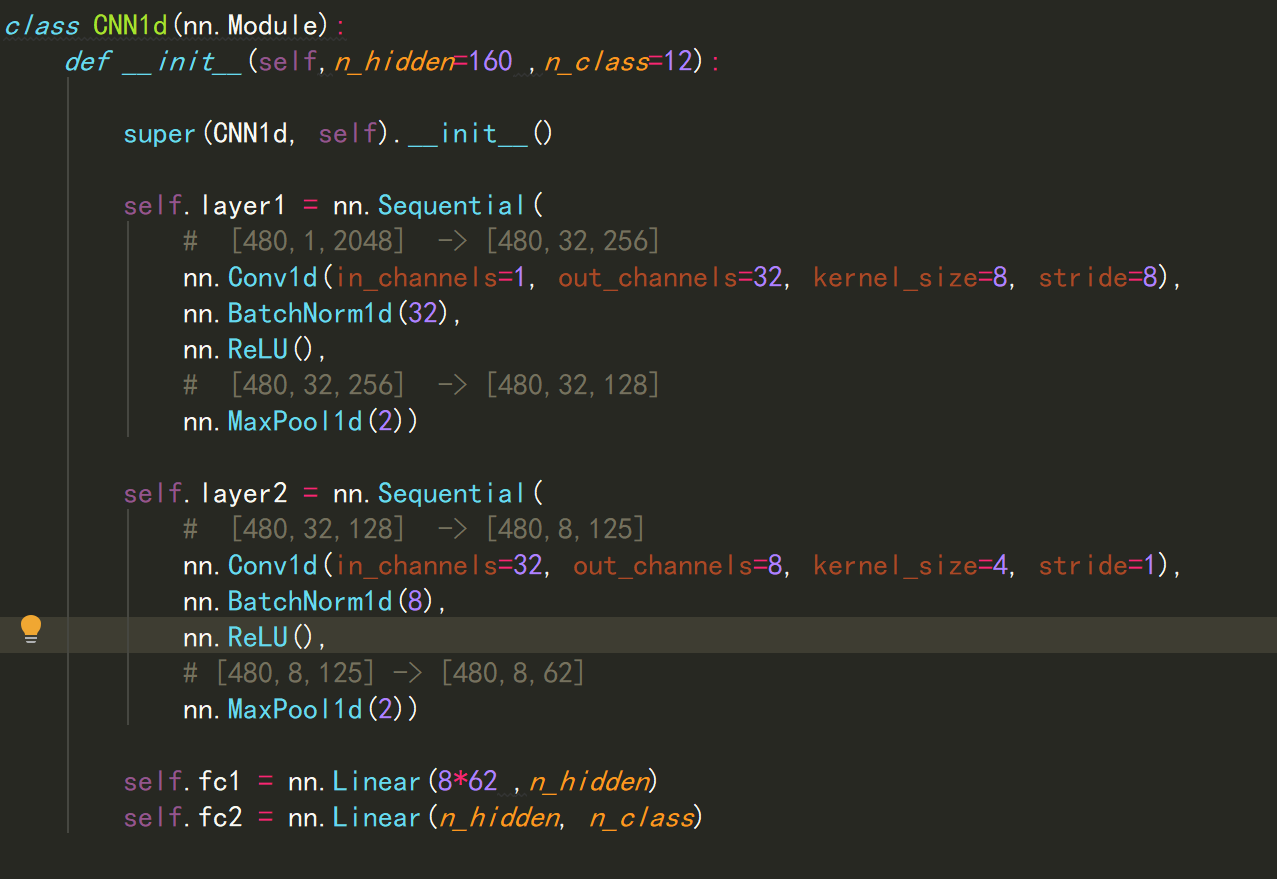
**因此需要对数据集的标签进行降维：**

**标签大小: label [batch\_size]**

**Y=torch.from\_numpy(datas[:, 0:1]).type(torch.LongTensor)**

**label = y.squeeze()**

**这里，我的问题是pytorch中定义网络结构需要输入数据与第一层卷积层的输入通道相一致，在keras定义中，却不需要，这是为什么？**

****

**图2 pytorch搭建的网络结构**



**图3 keras 搭建的网络结构**

**输入数据大小都是data[480 ,1, 2048] ，如果数据不是这样，对于pytorch网络输入会出现输入数据大小不匹配问题！**

**问题二：针对于pytorch中线性层（conv1d、BN、Relu、Maxpool1d）之后的第一层全连接层Linear的输入维度(这里是8\*62)是需要自己计算主动输入的吗？可以不可以通过网络结构自动填入这个参数？**

**问题三：我这里的网络是一次迭代就得到一次源域准确率和一次目标域准确率，这样的方式对不对？**

**问题四：这里的MMD最大均值偏差计算过程有点不太明白，还请老师指点一下，K核矩阵的计算过程，sigma是核间距吗？最后的mmd值与核矩阵的关系是怎样的？**