## **PyTorch 学习笔记**

## **一、PyTorch 基础认知**

PyTorch 是基于 Python 的机器学习框架，具备两大关键特性：一是支持在 GPU 上进行 Tensor 计算，可加速高维矩阵运算；二是能自动算微分，助力模型训练。

开展模型训练时，需准备好模型架构、损失函数与最佳算法，流程依次为训练、验证、测试的迭代。

## **二、数据准备：Dataset 与 DataLoader**

### **（一）核心概念**

Dataset用于收集所有数据，DataLoader则多次读取Dataset，每次从Dataset取出一些实例及对应值，再将其整理成batches，使用时通过dataloader = DataLoader(dataset, batch\_size, shuffle=True)构建，需先从torch.utils.data导入Dataset和DataLoader，自定义数据集和数据加载器也依赖这些基础导入。

### **（二）自定义 Dataset 类**

创建MyDataset类继承Dataset，需实现三个方法：

\_\_init\_\_(self, file)：完成数据读取与预处理，将处理后数据存于self.data。

\_\_getitem\_\_(self, index)：按索引index返回self.data中对应数据项，实现数据按需获取。

\_\_len\_\_(self)：返回self.data长度，告知整个数据集大小 。

DataLoader 借助 Dataset 的\_\_getitem\_\_方法，按批次（batch\_size ）获取数据组成 mini - batch，如从索引 0 到 4 依次取数据构建批次 。

## **三、Tensor 操作**

Tensor 是 PyTorch 中高维矩阵，为数据存储形式，可通过shape查看大小，不同维度对应dim0（一维 ）、dim0 dim1（二维 ）、dim0 dim1 dim2（三维 ）等，且 PyTorch 里的dim 类似 NumPy 中的axis。

### **（一）创建 Tensor**

可将已有的 list 或 numpy.ndarray 转换，x = torch.tensor([[1,7],[1,1]])、x = torch.from\_numpy(np.array([[1,7],[1,1]]))。

也能生成全 0 或全 1 的 Tensor，x = torch.zeros([2,1])、x = torch.ones([1,2,5])。

### **（二）常见操作**

算术运算：支持z = x + y、z = x - y，还有幂运算y = x.pow(2)、求和y = x.sum()、求平均y = x.mean()。

维度变换：transpose用于转置，二维 Tensor 中x = x.transpose(0,1)可交换 0 维和 1 维；squeeze删掉长度为 1 的维度，如[1,2,3]经x = x.squeeze(0)变为[2,3]；unsqueeze增加长度为 1 的维度，[2,3]用x = x.unsqueeze(1)变为[2,1,3]。

拼接操作：cat可连接多个 Tensor，如x = torch.zeros([2,1,3])、y = torch.zeros([2,3,3])、z = torch.zeros([2,2,3])，通过w = torch.cat([x,y,z], dim = 1)按维度 1 拼接，得到形状为[2,6,3]的 Tensor 。

设备与精度：数据类型别混用（如 float、long ），可通过 x = x.to('cpu') 在 CPU 计算，x = x.to('cuda') 在 GPU 计算，用 torch.cuda.is\_available()检查 GPU ，还能通过 specify 'cuda:0'指定某块 GPU。