人工智能中的编程 Task2

Task 2 中使用的多GPU环境：Kaggle网站([https://www.kaggle.com/]下create选择notebook) notebook中Settings下accelerator 选择 GPU T4x2
Kaggle notebook免费版只支持单进程，故Task2中数据并行采用DP而非DDP

实验一：Task_2/Task_2_1_T4.py , Task_2/Task_2_1_T4x2 分别为在单块T4和T4x2上训练的代码，Task_2/Task_2_1_log_T4.txt,Task_2/Task_2_1_log_T4x2.txt 为两者的运行日志

模型结构为LeNet

```
class LeNet(torch.nn.Module):
        def __init__(self):
                super().__init__()
                self.conv1 = nn.Conv2d(3, 6, stride=1,padding=1, kernel_size=3)
                self.pool = nn.MaxPool2d(2, 2)
                self.conv2 = nn.Conv2d(6, 16, stride=1, padding=1, kernel_size=3)
                self.fc1 = nn.Linear(1024, 120)
                self.fc2 = nn.Linear(120, 84)
                self.fc3 = nn.Linear(84, 10)
                self.relu = nn.ReLU()

        def forward(self ,x):
            x = self.pool(self.relu(self.conv1(x)))
            x = self.pool(self.relu(self.conv2(x)))
            x = torch.flatten(x, 1)
            x = self.relu(self.fc1(x))
            x = self.relu(self.fc2(x))
            x = self.fc3(x)
            return x
```

根据日志记录，20个epoch，相同超参数下，单T4的train_acc 68.0% , test_acc 63.1%
两块T4 DP 的train_acc 69.5% , test_acc 64.0%
可以看到两者在acc上结果是相近的

单T4的每个epoch训练的平均时间是12s左右，而两块T4 DP 的每个epoch训练的平均时间是17.8s左右
可以看到两块T4 DP 的训练速度比单T4的训练速度反而慢了，这可能是因为 1.模型结构比较简单，数据集相对规模较小 2.batch_size设置较小
3 GPU数量较少

导致并未充分利用多GPU的计算力，反而进程中数据传输到GPU的开销较大，超过了GPU并行计算的加速程度

实验二：为了验证实验一的结论，实验二在更为复杂的模型上进行，并且batch_size从50上调至200

Task_2/Task_2_2_T4.py,Task_2/Task_2_2_T4x2 分别为在单块T4和T4x2上训练的代码，Task_2/Task_2_2_log_T4.txt,Task_2/Task_2_2_log_T4x2.txt 为两者的运行日志

```
class VGG(nn.Module):
    def __init__(self):
        super().__init__()
        # 1. define multiple convolution and downsampling layers
        # 2. define full-connected layer to classify
        self.cnn = nn.Sequential()
        def cnn_block(in_channels,out_channels):
            return nn.Sequential(
            nn.Conv2d(in_channels,out_channels,kernel_size=3,padding=1),
            nn.BatchNorm2d(out_channels),
            nn.ReLU(),
            nn.Conv2d(out_channels,out_channels,kernel_size=3,padding=1),
            nn.BatchNorm2d(out_channels),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2,stride=2)
            ) if out_channels<256 else nn.Sequential(
            nn.Conv2d(in_channels,out_channels,kernel_size=3,padding=1),
            nn.BatchNorm2d(out_channels),
            nn.ReLU(),
            nn.Conv2d(out_channels,out_channels,kernel_size=3,padding=1),
            nn.BatchNorm2d(out_channels),
            nn.ReLU(),
            nn.Conv2d(out_channels,out_channels,kernel_size=3,padding=1),
            nn.BatchNorm2d(out_channels),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2,stride=2)
            )
        pre_channels = 3
        for i in range(5):
            out_channels = min(2**(i+6),512)
            self.cnn.add_module(f'cnn_block_{i}', cnn_block(pre_channels, out_channels))
            pre_channels = out_channels
        self.fcnn=nn.Sequential(
            nn.Linear(512,256),
            nn.ReLU(),
            nn.Dropout(),
            nn.Linear(256,128),
            nn.ReLU(),
            nn.Dropout(),
            nn.Linear(128,10)
            )
    def forward(self, x: torch.Tensor):
    # x: input image, shape: [B * C * H* W]
```

```
        # extract features
        # classification
            out1 = self.cnn(x)
            out1 = out1.flatten(start_dim=1)
            out = self.fcnn(out1)
            return out
```

根据日志记录，20个epoch，相同超参数下，单T4的train_acc 98.6% , test_acc 77.1%
两块T4 DP 的train_acc 98.2% , test_acc 75.5%
可以看到两者在acc上结果仍然是相近的

单T4的每个epoch训练的平均时间是28.7s左右，而两块T4 DP 的每个epoch训练的平均时间是23.6s左右
此时两块T4 DP 的训练速度比单T4的训练速度更快，但是加速程度并未是训练时间缩短至1/2，这仍然是进程数据传输到GPU的开销耗时
从而当模型复杂度提高，batch_size设置较大时，多GPU的并行计算优势会更明显，加速效果更显著