寒假选修-深度学习任务

一、基础学习

1.1 基本需求

如果电脑有较好的GPU，可以选择在自己电脑上训练，但还是建议使用云服务器进行训练（以后训练模型肯定得学会在服务器上训练）。可以使用梯队培训时配置的AI Studio进行训练飞桨AI Studio。

使用云服务器需要在终端操作，需要掌握一定的linux基础知识。

1.2 深度学习理论基础（选做）

无论是选择哪种学习路径，都建议把**遇到的所有名词弄懂**，并用markdown用自己能懂的描述记录下来。

可以直接搜索引擎搜博客弄懂，也可以b站搜这个名词弄懂，但最推荐的还是通过“请用通俗易懂的语言描述一下xxx的基本概念和作用”这种提问方式问ChatGPT等大模型，如果还是弄不懂，可以再加上“请你举一个详细的例子解释，让高中生都能弄懂”等描述，它会成为最耐心的入门导师。

目前最推荐的是使用coze.com的gpt4，使用方法可以见梯队群中的“通过coze使用gpt4.pdf”。使用这个服务需要那个东西，如果没有，可以通过glados.rock注册获得一年学生免费pro账户， 配置方法可以搜索glados教育计划，如果搞不定可以私信我询问。

1.2.1 视频学习

如果没有基础，建议通过这个视频了解深度学习的基本流程和pytorch的基本使用方法。我第一次接触深度学习就是靠这个视频入门的。《PyTorch深度学习实践》

如果可投入的时间比较充裕，建议跟着这个视频学习并复现里面出现过的代码，如果能把里面出现的代码全部手敲一遍，对深度学习就基本了解了。

如果可分配的时间不算多，可以结合我在梯队群里发的“梯队培训_深度学习.pdf"中4.1.2往后部分，了解深度学习的基本过程，然后通过这个视频了解对应的代码实现。循环神经网络可以不看。

如果寒假的其他安排比较多，可以只看P1-P4,P6,P8,P10的效果和代码部分，不看数学推导。（但是这样做非常容易后面存在完全看不懂的部分，深度学习对没有基础的同学来说会比较难，建议还是从头一步一步看，我当初曾经想直接看卷积神经网络部分，结果名词一个都看不懂），循环神经网络可以不看。

1.2.2 PPT路径

如果不习惯跟着视频学，可以看我在梯队群中发的"计算机视觉课件.zip"中的1-6部分，其中1、3、4讲可以快速阅读，仅做了解；*2-图像处理基础* 、 *5-前馈神经网络* 和 *6-卷积神经网络* 需要细看，了解整体流程，并弄懂对应的代码部分。7和8可以不看。

二、实训任务

实训包含2.1、2.2两个任务，通过这两个任务，你将能：

- 弄懂深度学习从**网络的构建**、**各部件的实例化**到**最终训练验证**的代码流程
- 了解通过开源项目完成**YOLOv8训练**的全过程

2.1 任务：重排代码并构建网络

在此任务中，你需要：①根据**2.1.2 整体流程**的提示，将**2.1.3 代码片段**中乱序的数个代码块重新排序，组成正确的**模型训练脚本**；②完成**网络结构的构建**（参考2.1.4，需要根据样例自行搭建）。最后③**成功运行训练脚本**，得到最终的**模型文件**，**并上传到评测网站得到分数。**到时将有排行榜实时显示分数，最终通过排名加权准确率计算此任务得分。对于评测网站的说明参见2.1.5小节。

如果对深度学习比较了解，可以在搭建完成后修改网络的各个模块及参数，如尝试不同的优化器和损失函数；使用不同的训练轮数和学习率；尝试使用不同的层数和结构等。

最终需要提交代码和报告。代码需要进行1:1注释(可以使用copilot，但你自己必须弄懂这段注释)，每行有具体作用的代码都需要解释**代码的作用**，**使用的方法**，**此方法各参数的作用**，**为什么这里要使用这个参数。**并用markdown写一份总结，格式和长度无要求，如果报告内容与代码对不上或与排行榜成绩有严重不符，排行榜成绩无效。总结内容包含：

- 用自己的话描述整段代码的全流程，包含哪些板块
- 分板块描述板块作用，实现流程，使用方法，方法解释
- 网络结构，每一层的作用，为什么选用这些层实现
- 尝试的不同网络结构对应的分数，并尝试分析其中原因

2.1.1 原始任务描述

任务要求：

设计一个卷积神经网络，并在其中使用ResNet模块，在MNIST数据集上实现10分类手写体数字识别。

注意事项：

1. 深度学习框架任选。
2. 不能直接导入现有的ResNet网络。
3. 可以尝试不同的激活函数、训练超参数等，至少尝试两种，观察并比较网络性能。
4. 实验报告需包含神经网络架构、每一轮mini-batch训练后的模型在训练集和测试集上的损失、最终的训练集和测试集准确率，不同设计变化导致的网络性能差异，以及对应的实验分析。

2.1.2 整体流程

```
导入依赖模块
设定参数
加载数据集
设计Residual模块
    ......
定义完整的卷积神经网络(ResNet_CNN)
    ......
实例化网络模型
定义损失函数
定义优化器
定义训练函数
定义测试函数
执行训练函数和测试函数
```

2.1.3 代码片段

```
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
model = ResNet_CNN().to(device)
loss_f = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(),lr=learning_rate)
```

```
if __name__ == '__main__':
    epochs=10
    train(epochs)
    test()
```

```
# 训练模型
def train(epochs):
    for epoch in range(epochs):
        model.train()
        for batch_idx,(data,target) in enumerate(train_loader):
            data,target = data.to(device), target.to(device)
            optimizer.zero_grad()
            output = model(data)
            loss = loss_f(output,target)
            loss.backward()
            optimizer.step()
            if batch_idx % log_interval == 0:
                print('Train Epoch:{} [{}/{} ({:.0f}%)]\tLoss:{:.6f}'.format(
                    epoch,batch_idx*len(data),len(train_loader.dataset),
                    100.*batch_idx/len(train_loader),loss.item()
                ))
```

```
test_loader = torch.utils.data.DataLoader(
    torchvision.datasets.MNIST('./data/', train=False, download=True,
                               transform=torchvision.transforms.Compose([
                                   torchvision.transforms.ToTensor(),
                                   torchvision.transforms.Normalize(
                                       (0.1307,), (0.3081,))
                               ])),
    batch_size=batch_size_test, shuffle=True)
```

```
epochs = 10
batch_size_train = 64
batch_size_test = 1000
learning_rate = 0.01
log_interval = 10
random_seed = 1
torch.manual_seed(random_seed)
```

```
train_loader = torch.utils.data.DataLoader(
    torchvision.datasets.MNIST('./data/', train=True, download=True,
                               transform=torchvision.transforms.Compose([
                                   torchvision.transforms.ToTensor(),
                                   torchvision.transforms.Normalize(
                                       (0.1307,), (0.3081,))
                               ])),
    batch_size=batch_size_train, shuffle=True)
```

```
import torch
import torchvision
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader
```

```
# 保存模型 state_dict()是一个字典，保存了网络中所有的参数
    # 转换并保存为torch.jit的模型
    example_input = torch.rand(1, 1, 28, 28).to(device)
    traced_model = torch.jit.trace(model, example_input)
    torch.jit.save(traced_model, "traced_model.pt")
```

2.1.4 网络结构示例

注意不要修改命名，否则其他部分可能会对不上。如果写"删去了"，说明此处建议还原，不代表只需要填充这个地方，还有其他地方需要自行设计。各个方法的参数需要自行调整以匹配自己的设计。建议先弄懂维度、通道等概念，各个方法的参数代表什么，每个方法的作用后填充。

```
# 残差模块
class ResidualBlock(torch.nn.Module):

    def __init__(self, in_channels, out_channels, stride=1):
        super(ResidualBlock, self).__init__()

        self.conv1 = nn.Conv2d(in_channels, out_channels, kernel_size=3,
stride=stride, padding=1, bias=False)

        self.relu = nn.ReLU(可选参数)


    def forward(self,x):
        # 保存输入数据，采用恒等映射
        identity = x

        #
        out =self.conv1(x)

        out =self.relu(out)


        # 还原结果
        删去了一句
```

```
        out = self.relu(out)
        # 返回结果
        return out
```

```
# 构建包含ResidualBlock的CNN
class ResNet_CNN(nn.Module):
    def __init__(self,num_classes=10):
        super(ResNet_CNN,self).__init__()


        self.conv1 =nn.Conv2d(自行设计参数)

        self.relu = nn.ReLU()

        self.res2 = ResidualBlock(自行设计参数)

        # 用一个自适应均值池化层将每个通道维度变成1*1，此句可选
        self.avg_pool = nn.AdaptiveAvgPool2d((1,1))

        self.fc = nn.Linear(需要自己根据网络调整参数)
    def forward(self,x):
        x = self.conv1(x)

        x = self.relu(x)

        x = self.res2(x)

        # n个通道，每个通道1*1，输出n*1*1
        x = self.avg_pool(x)
        # 将数据拉成一维
        x = x.view(x.size(0),-1)
        x = self.fc(x)
        return x
```

2.1.5 提交方式

通过（http://8.141.7.104:5246/upload）进行提交。登录账号和密码是姓名和QQ账号。需填写姓名并上传.pt格式文件，网站将自动评分并将得分更新到排行榜和提交列表中。注意填写正确姓名和尽量减少提交次数，服务器算力有限。

2.2 任务：训练模型并完成推理

在此任务中，你需要在本地或云服务器（推荐）配置Ultralytics的YOLO相关环境，将数据集转换为yolo格式并训练出一个雷达视角识别整车的模型，并对所给雷达赛场内录视频进行推理，需提交报告和推理后视频。

2.2.1 环境配置

在本地或云服务器配置环境，云服务器推荐使用免费的飞桨AI Studio或视个人情况租用服务器（如autodl），配置环境可以使用CPU启动建议先在免费云服务器上熟悉基本操作（折腾坏了能迅速重开）。

建议先安装anaconda，后创建虚拟环境并切换，参照Quickstart - Ultralytics YOLOv8 Docs进行环境配置。

需要注意的是，建议第一步先通过nvidia-smi等方式查看本机/云服务器的cuda版本，见右上角cuda version。Ultralytics包需要pytorch环境，安装的pytorch版本一定要和cuda 版本匹配，具体情况自行搜索教程。

```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 537.58                 Driver Version: 537.58       CUDA Version: 12.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name                     TCC/WDDM | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util  Compute M. |
|                                         |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA GeForce RTX 3050 ...   WDDM | 00000000:01:00.0 Off |                  N/A |
| N/A   44C    P0              12W /  55W |      0MiB /  4096MiB |      0%      Default |
|                                         |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

2.2.2 数据集准备

我们会将后续在CVAT上由全体梯队新标注的雷达赛场整车标注数据汇总并添加部分旧数据作为此次任务的数据集，分发给完成了后续标注任务的同学。未完成标注任务的同学可以私信完成的同学获取此数据集，或通过RoboMaster官方论坛（RoboMaster论坛）等渠道获取数据集。

（或许会导出为coco格式让他们自己转为yolo格式）上网自行了解yolo数据集的文件组织方式和标注格式（建议顺便了解下coco，上交格式等），将数据集整理转换为可训练的格式，可使用工具包（ezthor/Data-Augmentation ）中的conversion部分脚本进行转换。

2.2.3 修改配置文件

自行根据Ultralytics的官方文档中quickstart部分修改两个yaml文件，注意调整文件夹的路径正确。

2.2.4 训练模型

根据quickstart部分开始训练，建议了解screen的基本操作后在云服务器中挂screen进行训练。

2.2.5 推理

根据官网中modes下的predict部分学习如何对视频进行推理，推理视频到时候会发到群里。将推理并可视化后的视频保存。

2.2.6 提交内容

需要将报告和视频放入文件夹中压缩后提交，命名为xxxx（未定）

报告中需要有自己的配置过程截图，并将遇到的问题和解决方式（参考的资料附上）写上，最后需要总结整个流程和自己的经验。