寒假深度学习任务总结

2.1 手写数字识别

任务流程

> 由于函数包含多个模块，为了方便备份和调试，我采用 `jupyter notebook` 来进行实验。
> 关于学习中的一些基本概念，我在学习视频时做过记录，就不放在这里进行叙述了。

模块包含库的导入和超参数的设置，训练和测试数据集加载，结合残差网络的卷积神经网络设计，训练模块和本地测试模块

库的导入和超参数的设置

导入pytorch相关模块，包括含有MINIST数据集的与图像和视频相关的模块torchversion，以及包含激活函数的torch.functional，包含深度学习基本框架的torch.nn
然后设置一些训练必须的超参数,包括学习率，epoch数，训练集和测试集的mini_batch大小。同时，模型在开始训练时，所有可学习参数都为随机数，定了随机种子之后，所有以随机方式生成的参数（比如网络权重的初始化、数据的随机打乱等）在每次运行时都会有固定的值，保证了实验的重复性。

训练和测试数据集的加载

从torch.utils工具包中导入DataLoader，

```
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)
```

使用了torch.utils.data.DataLoader类
`At the heart of PyTorch data loading utility is the torch.utils.data.DataLoader class. It represents a Python iterable over a dataset`
————官方文档
构造函数中比较基本的参数为数据集，batch_size， 每次训练后是否打乱顺序，这个类将数据集和取样器合并，可以通过此对象来获取训练样本。
torchvision.datasets中有MINIST数据集，首先需要对数据进行预处理，利用torchvision.transforms.Compose方法对数据依次进行操作，增强数据的随机性

```
epochs = 10 # 设置总共需要训练多少轮
batch_size_train = 32# 每次训练mini_batch的大小
batch_size_test = 10000 # 每次测试所用样本的大小
# learning_rate = 0.0011 # 学习率
# learning_rate = 0.0005 # 学习率
learning_rate = 0.01  # 学习率
log_interval = batch_size_train # 按照mini_batch的大小打印测试结果
random_seed = 1 # 将随机种子设置为 1。可以根据需要选择任何整数值作为随机种子。
torch.manual_seed(random_seed) # 在使用相同的随机种子、相同的环境和相同的代码时，
                               #能保证得到相同的随机数序列。
```

模型设计

深深体会到了奥卡姆剃刀原理，大道至简。
有试过将网络设计很多层，在GPT的直到下加了一些较为复杂的函数，最后模型太大甚至有时会卡爆评测系统，最终实测效果也很一般。简简单单两层卷积效果良好
主要包含卷积块的设计和神经网络的设计。
卷积块有卷积，激活，最大池化层，

模型有卷积，激活，最大池化层，和残差网络的叠加重复，最后接一个全连接层到最终输出。
希望有机会可以看看其他同学的想法。

不同参数比较

- **激活函数**：使用ReLU函数的效果明显由于Sigmoid函数
- **损失函数**：在多分类问题上，交叉熵损失函数更优
- **优化器**：Adam优化器简单粗暴，加入动量后可以提升模型性能。但最后结果采用SGD效果也不错
- **学习率**：不同优化器适合的学习率不同。Adam适合较小的学习率，如1e4级别，而SGD适合较大的学习率如0.1
- **epoch与过拟合**：在本地测试方面，基本上是训练轮数越多，本地测试效果越好。但我最多一次试着在模型上跑了80个epoch，得分只有80+。我本地测试效果最好时候可以达到99.63%，轮数没有很多，但离谱的是这个模型在线准确率只有可怜巴巴的40+。

深切体会

只有对本质有了更深的理解才能创造出更好的模型，才能有所创新。

第一次

激活函数 RELU
epoch 12 lr 0.01
本地准确率 99%

```
loss_f = nn.CrossEntropyLoss()
optimizer = optim.SGD(model.parameters(),lr=learning_rate)
```

姓名			得分		上传时间			状态	
贾竟一			95		02-28 19:04			Success	
0	1	2	3	4	5	6	7	8	9
9	8	10	10	10	10	10	10	10	8

首次提交，分数还不错，后面一直没有优化多少。

```
# 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(1, 16, kernel_size=5)
        self.conv2 = nn.Conv2d(16, 32, kernel_size=5)

        self.relu = nn.ReLU()

        self.res1 = ResidualBlock(16, 16, stride=1)
        self.res2 = ResidualBlock(32, 32, stride=1)

        self.mp = nn.MaxPool2d(2)

        # 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(512, 10)

    def forward(self,x):

        x = self.conv1(x)
        x = self.relu(x)
        x = self.mp(x)
        x = self.res1(x)

        # n个通道，每个通道1*1，输出n*1*1
        #x = self.avg_pool(x)

        x = self.conv2(x)
        x = self.relu(x)
        x = self.mp(x)
        x = self.res2(x)


        # 将数据拉成一维
        x = x.view(x.size(0),-1)
        x = self.fc(x)
        return x
# 初代
# 构建包含ResidualBlock的CNN
class ResNet_CNN(nn.Module):
    def __init__(self,num_classes=10):
        super(ResNet_CNN,self).__init__()

        # Convolutional layer with 1 input channel, 8 output channels and a kernel size of 5
        self.conv1 = nn.Conv2d(1, 8, kernel_size=5)

        # Convolutional layer with 8 input channels, 16 output channels and kernel size of 5
        self.conv2 = nn.Conv2d(8, 16, kernel_size=5)

        # Convolutional layer with 16 input channels, 32 output channels and kernel size of 5
        self.conv3 = nn.Conv2d(16, 32, kernel_size=5)

        # Activation function: ReLU (Rectified Linear Unit)
        self.relu = nn.ReLU()

        # Residual blocks that learn to model the identity function, enabling the training of very deep models by preventing vanishing gradients problem.
        self.res1 = ResidualBlock(8, 8, stride=1)
        self.res2 = ResidualBlock(16, 16, stride=1)
        self.res3 = ResidualBlock(32, 32, stride=1)

        # MaxPooling layer with a size of 2 which reduces the spatial dimensions by taking the maximum value over a window of size 2x2
        self.mp = nn.MaxPool2d(2)

        # A fully connected or dense layer, which brings together all the information to make a classifier, it has 512 input features and 10 output features
        self.fc = nn.Linear(512, 10)

    def forward(self,x):
        # Passes the input image through the first convolutional layer then apply the ReLU activation function and max pooling
        x = self.relu(self.conv1(x))
        x = self.mp(x)
        x = self.res1(x)

        # Passes the result through the second convolutional layer then apply the ReLU activation function
```

```
        x = self.relu(self.conv2(x))
        x = self.res2(x)

        # Passes the result through the third convolutional layer then apply the ReLU activation function
        x = self.relu(self.conv3(x))
        x = self.res3(x)

        # Flatten the tensor into a 1D tensor to feed it into the fully connected layer
        x = x.view(x.size(0),-1)
        x = self.fc(x)

        return x
```

后续修改了多组模型，有些模型记录在了jupyter_notebook，但感觉这个简简单单的模型就挺好。

2.2 YOLO训练模型及视频推理

> 2024.3.8 贾竟一

主要有环境配置，数据集处理，模型训练，视频预测四个环节。在整个过程中，出现了一些奇葩的Bug，特意记录一下并作反思。

环境配置

- 已经在pycharm中指定了解释器，也可以在解释器下看待已经安装的包，但在运行代码时仍无法识别导入的包。但后续项目结束之后检查又恢复正常。
  解决方法：代码不多，使用命令行运行
- 此外本项目的依赖不需要特殊安装。在poetry解析并安装依赖文件后发现所有依赖都已经安装完毕。

数据集处理

- 一开始对yolo数据集的格式不熟悉，搞不懂conversion脚本怎么用，直到看到一个配置视频才发现学长已经全部预处理完毕，只需要简单设置一下文件格式即可。环境配置方面可能视频比博客能看到更多细节。
- 此外，我出于方便，并未采用脚本自动划分数据集，私认为经验上的比例，没只需要经验上正确。后来看到有同学利用脚本划分，不仅可以很好控制划分比例，也可以将数据集打乱，也是学到一招。
  最终模型对于一些静止小车识别不是很好，也有一些错误识别，自然有硬件条件限制的原因，但算法上主要原因应该没有随机打乱数据集。

模型训练

困难重重。

- 飞桨的云服务器安装包很慢，研究半天我也不知道文档中所说的“切换”如何实现，我理解是在本地anconda切换到云服务器上，但GPT说要找到root和ip地址。无果，遂在本地训练。
- 中文路径命问题。
  每次训练结束后，都报错显示无法找到weights文件夹，文件夹正常，路径正常，但格式不正常。假期装了固态硬盘，并将桌面移动到E盘，但移动后桌面路径一直为中文，改名后仍为中文，且无法移动。模型的runs文件一直会在E盘桌面创建，重新配置虚拟环境后仍旧如此。

```
cay=0.0)
Image sizes 640 train, 640 val
Using 8 dataloader workers
Logging results to E:\桌面\ultralytics\runs\detect\train5
Starting training for 1 epochs...

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
        1/1      2.16G      1.326      1.559     0.9259         12        640: 100%|██████████| 91/91 [00:25<00:00,  3.
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 9/9 [00:03<0
                   all        273       1336       0.99      0.686      0.911      0.604
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "E:\桌面\ultralytics\ultralytics\engine\model.py", line 644, in train
    self.trainer.train()
  File "E:\桌面\ultralytics\ultralytics\engine\trainer.py", line 208, in train
    self._do_train(world_size)
  File "E:\桌面\ultralytics\ultralytics\engine\trainer.py", line 432, in _do_train
    self.save_model()
  File "E:\桌面\ultralytics\ultralytics\engine\trainer.py", line 494, in save_model
    torch.save(ckpt, self.last)
  File "E:\桌面\ultralytics\ultralytics\utils\patches.py", line 82, in torch_save
    return _torch_save(*args, **kwargs)
  File "D:\anaconda\envs\YOLO-v8\lib\site-packages\torch\serialization.py", line 440, in save
    with _open_zipfile_writer(f) as opened_zipfile:
  File "D:\anaconda\envs\YOLO-v8\lib\site-packages\torch\serialization.py", line 315, in _open_zipfile_writer
    return container(name_or_buffer)
  File "D:\anaconda\envs\YOLO-v8\lib\site-packages\torch\serialization.py", line 288, in __init__
    super().__init__(torch._C.PyTorchFileWriter(str(name)))
RuntimeError: Parent directory E:\桌面\ultralytics\runs\detect\train5\weights does not exist.
>>>
```

```
13 个用法   Glenn Jocher *
def get_save_dir(args, name=None):
    """Return save_dir as created from train/val/predict arguments."""
    args.save_dir = r"D:\ultralytics\ultralytics\runs\save_dir" # 新加
    if getattr(args, "save_dir", None):
        save_dir = args.save_dir
    else:
        from ultralytics.utils.files import increment_path

        project = args.project or (ROOT.parent / "tests/tmp/runs" if TESTS_RUNNING else RUNS_DIR) / args.task
        name = name or args.name or f"{args.mode}"
        save_dir = increment_path(Path(project) / name, exist_ok=args.exist_ok if RANK in (-1, 0) else True)

    return Path(save_dir)
```

解决：无奈之下翻开模型源码，在 `cfg\__init__.py` 中将保存地址强行改到D盘，解决了中文路径的问题。

- 显存问题。3050Ti显卡只有4G显存，设置 `batch = 32` || `yolom.pt` 都会显示缓存分配不足。 `batch` & `yolon.pt` 强有力地制约着模型训练性能。解决：增加训练轮数，缩小batch，使用最基本的模型。
  同时，每次训练后都会出现显存不够分配问题，甚至短时间重启也无法解决问题。

```
                 all        273       1336      0.958      0.925      0.978       0.76

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      10/20      2.17G      0.786     0.4569     0.8168          3        640: 100%|██████████| 91/91 [00:14<00:00,  6.
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 9/9 [00:01<0
                   all        273       1336      0.966       0.93      0.979       0.75
Closing dataloader mosaic
Traceback (most recent call last):
  File "<string>", line 1, in <module>
  File "C:\Users\86153\AppData\Local\Programs\Python\Python311\Lib\multiprocessing\spawn.py", line 120, in spawn_main
    exitcode = _main(fd, parent_sentinel)
               ^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\86153\AppData\Local\Programs\Python\Python311\Lib\multiprocessing\spawn.py", line 130, in _main
    self = reduction.pickle.load(from_parent)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\86153\AppData\Local\Programs\Python\Python311\Lib\site-packages\torch\__init__.py", line 122, in <module>
    raise err
OSError: [WinError 1455] 页面文件太小，无法完成操作。 Error loading "C:\Users\86153\AppData\Local\Programs\Python\Python311\Lib\site-packages\torch\lib\cufft64_10.dll" or one of i
ts dependencies.
```

解决：加入 `torch.cuda.empty_cache()` 手动释放。

- 训练集和测试集的问题。
  一开始对这两个概念有所混淆，觉着都是评估模型性能的数据，不如把更多的数据用来训练。索性直接砍掉测试集，将验证集作为测试集。后面了注意到自己的best永远是last，感觉有些不对劲，查阅后发现验证集是调整超参数的，测试集是检验模型最终效果的。

视频预测

- 结果分析：存在一处固定位置的错误识别，遮挡小车识别效果不好。
- 视频预测的结果对于我来说似乎没有太大的优化空间，有空以后挂云服务器进行训练可以多尝试一下 。
- 我在本地尝试训练了三个模型，在第一次尝试的基础上分别增加了训练轮数和批次大小。三次训练结果都未能识别视频开始静止+遮挡的小车，但将bacth调小，训练轮数增加后，出现了错误识别的情况，但与此同时对于远处一些小车的识别准确率也提升了一些。

反思

- **关于习惯**。做了很多尝试，但缺乏记录，事后难以复盘。
- **关于Bug**。Bug并不会像魔术师手里的鸽子一样凭空出现，只不过没有弄懂他为什么会产生。遇到问题不应该摆烂和畏难，查阅资料也解决不了就要多多尝试。希望在本学期专业课学习之后，对计算机有更深入的理解。
- **关于Debug**。追本溯源，代码不是黑箱，总有方法可以绕过一下障碍，短期迫不得已也可以更改项目源码。
- **关于学习渠道**。GPT可以提供局部+个性化已经一些通用问题的解答。但缺乏宏观上+个性化的解决方案。有时Google is all we need。
  https://blog.csdn.net/xiyou__/article/details/121287909?spm=1001.2101.3001.6650.2&utm_medium=distribute.pc_relevant.none-task-blog-2~default~CTRLIST~Rate-2-121287909-blog-120538267.235^v43^pc_blog_bottom_relevance_base2&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2~default~CTRLIST~Rate-2-121287909-blog-120538267.235^v43^pc_blog_bottom_relevance_base2&utm_relevant_index=5