实操教程｜PyTorch实现断点继续训练

CV开发者都爱看的 极市平台 2023-01-29 22:00:21 发表于广东 手机阅读

作者｜HUST小菜鸡@知乎（已授权）

来源｜https://zhuanlan.zhihu.com/p/133250753

编辑｜极市平台

极市导读

本文整理了pytorch实现断电继续训练时需要注意的要点，附有代码详解。

最近在尝试用CIFAR10训练分类问题的时候，由于数据集体量比较大，训练的过程中时间比较长，有时候想给停下来，但是停下来了之后就得重新训练，之前师兄让我们学习断点继续训练及继续训练的时候注意epoch的改变等，今天上午给大致整理了一下，不全面仅供参考

```
Epoch:  9 | train loss: 0.3517 | test accuracy: 0.7184 | train time: 14215
Epoch:  9 | train loss: 0.2471 | test accuracy: 0.7252 | train time: 14309
Epoch:  9 | train loss: 0.4335 | test accuracy: 0.7201 | train time: 14403
Epoch:  9 | train loss: 0.2186 | test accuracy: 0.7242 | train time: 14497
Epoch:  9 | train loss: 0.2127 | test accuracy: 0.7196 | train time: 14591
Epoch:  9 | train loss: 0.1624 | test accuracy: 0.7142 | train time: 14685
Epoch:  9 | train loss: 0.1795 | test accuracy: 0.7170 | train time: 14780
绝望！！！！！训练到了一定次数发现训练次数少了，或者中途断了又得重新开始训练
```

一、模型的保存与加载

PyTorch中的保存（序列化，从内存到硬盘）与反序列化（加载，从硬盘到内存）

torch.save主要参数：obj：对象 、f：输出路径

torch.load 主要参数 ： f：文件路径 、map_location：指定存放位置、 cpu or gpu

模型的保存的两种方法：

1、保存整个Module

```
torch.save(net, path)
```

2、保存模型参数

```
state_dict = net.state_dict()
torch.save(state_dict , path)
```

二、模型的训练过程中保存

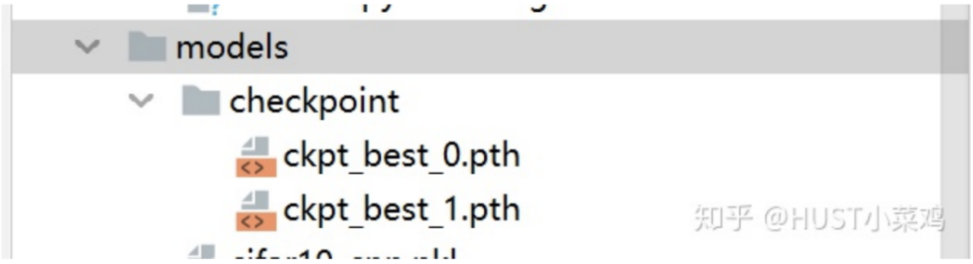
```
checkpoint = {
        "net": model.state_dict(),
        'optimizer':optimizer.state_dict(),
        "epoch": epoch
    }
```

将网络训练过程中的网络的权重，优化器的权重保存，以及epoch 保存，便于继续训练恢复

在训练过程中，可以根据自己的需要，每多少代，或者多少epoch保存一次网络参数，便于恢复，提高程序的鲁棒性。

```
checkpoint = {
        "net": model.state_dict(),
        'optimizer':optimizer.state_dict(),
        "epoch": epoch
    }
    if not os.path.isdir("./models/checkpoint"):
        os.mkdir("./models/checkpoint")
    torch.save(checkpoint, './models/checkpoint/ckpt_best_%s.pth' %(str(ep
```

通过上述的过程可以在训练过程自动在指定位置创建文件夹，并保存断点文件



三、模型的断点继续训练

```
if RESUME:
    path_checkpoint = "./models/checkpoint/ckpt_best_1.pth"  # 断点路径
    checkpoint = torch.load(path_checkpoint)  # 加载断点

    model.load_state_dict(checkpoint['net'])  # 加载模型可学习参数

    optimizer.load_state_dict(checkpoint['optimizer'])  # 加载优化器参数
    start_epoch = checkpoint['epoch']  # 设置开始的epoch
```

指出这里的是否继续训练，及训练的checkpoint的文件位置等可以通过argparse从命令行直接读取，也可以通过log文件直接加载，也可以自己在代码中进行修改。关于argparse参照我的这一篇文章：

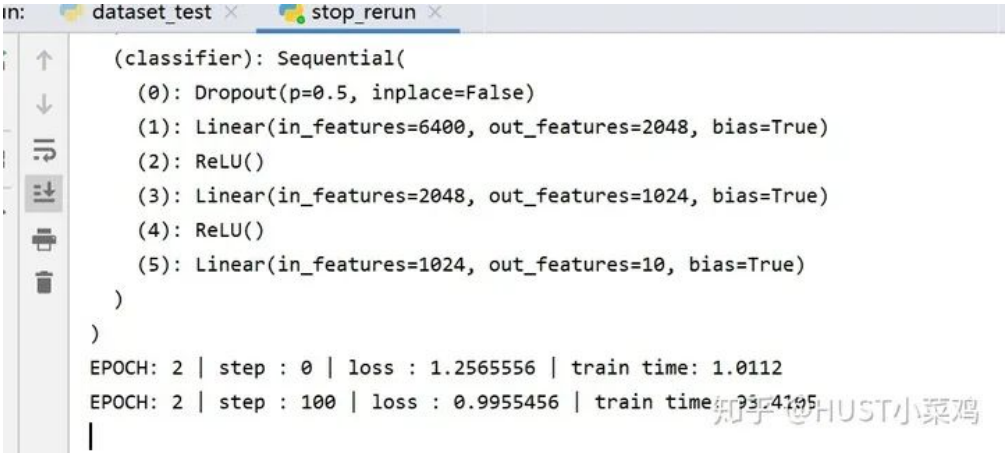
HUST小菜鸡：argparse 命令行选项、参数和子命令解析器

https://zhuanlan.zhihu.com/p/133285373

四、重点在于epoch的恢复

```
start_epoch = -1


if RESUME:
    path_checkpoint = "./models/checkpoint/ckpt_best_1.pth"  # 断点路径
    checkpoint = torch.load(path_checkpoint)  # 加载断点

    model.load_state_dict(checkpoint['net'])  # 加载模型可学习参数

    optimizer.load_state_dict(checkpoint['optimizer'])  # 加载优化器参数
    start_epoch = checkpoint['epoch']  # 设置开始的epoch



for epoch in  range(start_epoch + 1 ,EPOCH):
    # print('EPOCH:',epoch)
    for step, (b_img,b_label) in enumerate(train_loader):
        train_output = model(b_img)
        loss = loss_func(train_output,b_label)
        # losses.append(loss)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
```

通过定义start_epoch变量来保证继续训练的时候epoch不会变化



断点继续训练

一、初始化随机数种子

```python
import torch
import random
import numpy as np

def set_random_seed(seed = 10,deterministic=False,benchmark=False):
    random.seed(seed)
    np.random(seed)
    torch.manual_seed(seed)
    torch.cuda.manual_seed_all(seed)
    if deterministic:
        torch.backends.cudnn.deterministic = True
    if benchmark:
        torch.backends.cudnn.benchmark = True
```

关于torch.backends.cudnn.deterministic和torch.backends.cudnn.benchmark详见

Pytorch学习0.01:cudnn.benchmark= True的设置

https://www.cnblogs.com/captain-dl/p/11938864.html

pytorch---之cudnn.benchmark和cudnn.deterministic_人工智能_zxyhhjs2017的博客

https://blog.csdn.net/zxyhhjs2017/article/details/91348108

Benchmark 模式会提升计算速度，但是由于计算中有随机性，每次网络前馈结果略有差异。

```
torch.backends.cudnn.benchmark = True
```

如果想要避免这种结果波动，设置

```
torch.backends.cudnn.deterministic = True
```

benchmark用在输入尺寸一致，可以加速训练，deterministic用来固定内部随机性

二、多步长SGD继续训练

在简单的任务中，我们使用固定步长（也就是学习率LR）进行训练，但是如果学习率lr设置的过小的话，则会导致很难收敛，如果学习率很大的时候，就会导致在最小值附近，总会错过最小值，loss产生震荡，无法收敛。所以这要求我们要对于不同的训练阶段使用不同的学习率，一方面可以加快训练的过程，另一方面可以加快网络收敛。

采用多步长 torch.optim.lr_scheduler的多种步长设置方式来实现步长的控制，lr_scheduler的各种使用推荐参考如下教程：

【转载】 Pytorch中的学习率调整lr_scheduler,ReduceLROnPlateau

https://www.cnblogs.com/devilmaycry812839668/p/10630302.html

所以我们在保存网络中的训练的参数的过程中，还需要保存lr_scheduler的state_dict，然后断点继续训练的时候恢复

```python
#这里我设置了不同的epoch对应不同的学习率衰减，在10->20->30，学习率依次衰减为原来的0.
lr_schedule = torch.optim.lr_scheduler.MultiStepLR(optimizer,milestones=[
```

```
optimizer = torch.optim.SGD(model.parameters(),lr=0.1)

for epoch in range(start_epoch+1,80):
    optimizer.zero_grad()
    optimizer.step()
    lr_schedule.step()

    if epoch %10 ==0:
        print('epoch:',epoch)
        print('learning rate:',optimizer.state_dict()['param_groups'][0][
```

lr的变化过程如下：

```
epoch: 10
learning rate: 0.1
epoch: 20
learning rate: 0.010000000000000002
epoch: 30
learning rate: 0.0010000000000000002
epoch: 40
learning rate: 0.00010000000000000003
epoch: 50
learning rate: 1.0000000000000004e-05
epoch: 60
learning rate: 1.0000000000000004e-06
epoch: 70
learning rate: 1.0000000000000004e-06
```

我们在保存的时候，也需要对lr_scheduler的state_dict进行保存，断点继续训练的时候也需要恢复lr_scheduler

```
#加载恢复
if RESUME:
    path_checkpoint = "./model_parameter/test/ckpt_best_50.pth"  # 断点路径
    checkpoint = torch.load(path_checkpoint)  # 加载断点

    model.load_state_dict(checkpoint['net'])  # 加载模型可学习参数

    optimizer.load_state_dict(checkpoint['optimizer'])  # 加载优化器参数
    start_epoch = checkpoint['epoch']  # 设置开始的epoch
    lr_schedule.load_state_dict(checkpoint['lr_schedule'])#加载lr_schedule



#保存
for epoch in range(start_epoch+1,80):

    optimizer.zero_grad()

    optimizer.step()
```

```
    lr_schedule.step()


    if epoch %10 ==0:
        print('epoch:',epoch)
        print('learning rate:',optimizer.state_dict()['param_groups'][0][
        checkpoint = {
            "net": model.state_dict(),
            'optimizer': optimizer.state_dict(),
            "epoch": epoch,
            'lr_schedule': lr_schedule.state_dict()
        }
        if not os.path.isdir("./model_parameter/test"):
            os.mkdir("./model_parameter/test")
        torch.save(checkpoint, './model_parameter/test/ckpt_best_%s.pth'
```

三、保存最好的结果

每一个epoch中的每个step会有不同的结果，可以保存每一代最好的结果，用于后续的训练

第一次实验代码

```
RESUME = True

EPOCH = 40
LR = 0.0005


model = cifar10_cnn.CIFAR10_CNN()

print(model)
optimizer = torch.optim.Adam(model.parameters(),lr=LR)
loss_func = nn.CrossEntropyLoss()

start_epoch = -1


if RESUME:
    path_checkpoint = "./models/checkpoint/ckpt_best_1.pth"  # 断点路径
    checkpoint = torch.load(path_checkpoint)  # 加载断点

    model.load_state_dict(checkpoint['net'])  # 加载模型可学习参数

    optimizer.load_state_dict(checkpoint['optimizer'])  # 加载优化器参数
    start_epoch = checkpoint['epoch']  # 设置开始的epoch



for epoch in  range(start_epoch + 1 ,EPOCH):
    # print('EPOCH:',epoch)
    for step, (b_img,b_label) in enumerate(train_loader):
        train_output = model(b_img)
```

```
        loss = loss_func(train_output,b_label)
        # losses.append(loss)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        if step % 100 == 0:
            now = time.time()
            print('EPOCH:',epoch,'| step :',step,'| loss :',loss.data.num

    checkpoint = {
        "net": model.state_dict(),
        'optimizer':optimizer.state_dict(),
        "epoch": epoch
    }
    if not os.path.isdir("./models/checkpoint"):
        os.mkdir("./models/checkpoint")
    torch.save(checkpoint, './models/checkpoint/ckpt_best_%s.pth' %(str(e
```

更新实验代码

```
optimizer = torch.optim.SGD(model.parameters(),lr=0.1)
lr_schedule = torch.optim.lr_scheduler.MultiStepLR(optimizer,milestone
start_epoch = 9
# print(schedule)


if RESUME:
    path_checkpoint = "./model_parameter/test/ckpt_best_50.pth"  # 断点
    checkpoint = torch.load(path_checkpoint)  # 加载断点

    model.load_state_dict(checkpoint['net'])  # 加载模型可学习参数

    optimizer.load_state_dict(checkpoint['optimizer'])  # 加载优化器参数
    start_epoch = checkpoint['epoch']  # 设置开始的epoch
    lr_schedule.load_state_dict(checkpoint['lr_schedule'])

for epoch in range(start_epoch+1,80):

    optimizer.zero_grad()

    optimizer.step()
    lr_schedule.step()


    if epoch %10 ==0:
        print('epoch:',epoch)
        print('learning rate:',optimizer.state_dict()['param_groups'][
        checkpoint = {
            "net": model.state_dict(),
```

```
            'optimizer': optimizer.state_dict(),
            "epoch": epoch,
            'lr_schedule': lr_schedule.state_dict()
        }
        if not os.path.isdir("./model_parameter/test"):
            os.mkdir("./model_parameter/test")
        torch.save(checkpoint, './model_parameter/test/ckpt_best_%s.pt
```

公众号后台回复“CNN综述”获取67页综述深度卷积神经网络架构

极市干货

技术干货：损失函数技术总结及Pytorch使用示例｜深度学习有哪些trick？｜目标检测正负样本区分策略和平衡策略总结

实操教程：GPU多卡并行训练总结（以pytorch为例）｜CUDA WarpReduce 学习笔记｜卷积神经网络压缩方法总结

点击阅读原文进入CV社区

收获更多技术干货

阅读原文

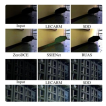
喜欢此内容的人还喜欢

YOLOv5帮助母猪产仔2 南京农业大学研发母猪产仔检测模型并部署到

极市平台

ICCV23｜将隐式神经表征用于低光增强，北大张健团队提出NeRCo

极市平台



9个数据科学中常见距离度量总结以及优缺点概述

极市平台