PyTorch

- 上手简单
- 调试方便
- 文档友好

用PyTorch进行人脸分类

- 任务：正确分类10M人脸图片，包含100K人
- 步骤
  - 1.准备数据集
  - 2.搭建网络
  - 3.编写训练、验证代码

数据集

- torchvision.datasets
  - MNIST
  - Fashion-MNIST
  - EMNIST
  - COCO
  - LSUN
  - ImageFolder
  - DatasetFolder
  - Imagenet-12
  - CIFAR
  - STL10
  - SVHN
  - PhotoTour

```
root/dog/xxx.png
root/dog/xxy.png
root/dog/xxz.png

root/cat/123.png
root/cat/nsdf3.png
root/cat/asd932_.png
```

- MS-Celeb1M
  - ID1
  - ID2
    - 1.jpg
    - 2.jpg
    - 3.ipg

数据集-ImageFolder

```python
import torch
import torchvision.datasets as datasets
import torchvision.transforms as transforms

traindir = '/ssd/Dataset/MS-Celeb-1M'
normalize = transforms.Normalize(mean=[0.485, 0.456, 0.406])
train_dataset = datasets.ImageFolder(
    traindir,
    transforms.Compose([
        transforms.Resize(224),
        transforms.CenterCrop(224),
        transforms.RandomHorizontalFlip(),
        transforms.ToTensor(),
        normalize,
    ]))
```

搭建网络

- torch.nn
  - Parameters
  - Containers
  - Convolution layers
  - Pooling layers
  - Padding layers
  - Non-linear activations (weighted sum, nonlinearity)
  - Non-linear activations (other)
  - Normalization layers
  - Recurrent layers
  - Linear layers
  - Dropout layers
  - Sparse layers
  - Distance functions
  - Loss functions
  - Vision layers
  - DataParallel layers (multi-GPU, distributed)

- torch.nn.Module
  - 所有网络的基类
  - 编写新的网络
    - 继承 torch.nn.Module
    - 编写 初始化函数
    - 编写 前向传递

```
import torch.nn as nn
import torch.nn.functional as F

class Model(nn.Module):
    def __init__(self):
        super(Model, self).__init__()
        self.conv1 = nn.Conv2d(1, 20, 5)
        self.conv2 = nn.Conv2d(20, 20, 5)

    def forward(self, x):
       x = F.relu(self.conv1(x))
       return F.relu(self.conv2(x))
```

搭建网络-ResNet

- torchvision.models
  - VGG
  - ResNet
  - Alexnet

```
17  # 载入现有网络
18  net = torchvision.models.resnet34(pretrained=False)
19  # 重写最后一层FC
20  net.fc = torch.nn.Linear(512, 100000, bias=Flase)
21
22  # Loss公式 以及 更新器
23  criterion = torch.nn.CrossEntropyLoss()
24  optimizer = torch.optim.SGD(net.parameters(), args.lr,
25                              momentum=args.momentum,
26                              weight_decay=args.weight_decay)
```

编写训练代码

```
26  # 数据读取器
27  train_loader = torch.utils.data.DataLoader(
28          train_dataset, batch_size=args.batch_size, shuffle=(train_sampler is None),
29          num_workers=args.workers, pin_memory=True, sampler=train_sampler)
30
31  # 开始训练
32  for epoch in range(args.start_epoch, args.epochs):
33      # 训练模式
34      net.train()
35      for i, (input, target) in enumerate(train_loader):
36          # 前向传递，计算Loss
37          output = net(input)
38          loss = crite net
39          # 反向传导，更新参数
40          optimizer.zero_grad()
41          loss.backward()
42          optimizer.step()
```

资源

- 官方文档：https://pytorch.org/docs/stable/index.html
- 官方论坛：https://discuss.pytorch.org/
  - Bug
  - Method
- GitHub：https://github.com/pytorch/
  - Code
  - Example