# Demo 使用说明文档

## 一、编程语言与运行环境：

编程语言：python3
深度学习框架：PyTorch
测试环境：在 linux & Windows10 均已经通过测试。
硬件要求：普通个人电脑即可，如有条件，可使用 GPU 加速训练

## 二、前期工作：

1. 环境配置：按照文档**“环境配置指南”**安装所需要的包（*这部分很重要，请仔细阅读安装指南）。
2. 数据集：将下载的数据集进行解压后，将 pickle_file 文件夹下的三个数据文件 **train_data, val_data, test_data 移动到代码的 data 文件夹下。**其中，images 文件夹对应的是可视化的图片数据，供大家分析使用，在成
3. 模型：我们提供了已经训练好的模型 **AlexNet_BN-45.41.pth.tar（位于 Result/AlexNet_BN-45.41 文件夹下）**，方便测试程序。

## 三、代码如何使用

提供的代码主要用于：训练模型、测试模型性能、生成测试集的 score 文件、计算模型的参数量与计算量。在每个模块我们介绍了关键参数以及提供了实例进行运行。

### 1. 训练模型（main.py）

说明：在训练模型时主要用到的文件、参数设置说明

**1.1 参数介绍**

```
--data    # 数据集文件所在路径，默认为代码中的 data 文件夹
--result  # 训练结果存储路径，默认为代码中的 Results 文件夹
--arch    # 网络架构，如 AlexNet_BN
--epochs  # 训练的 epoch 数，1 个 epoch 等于使用训练集中的全部样本训练一次
--batch-size  # 一次训练的样本数目
--lr      # 初始学习率的大小
--optimizer # 优化算法，默认 SGD，可以在相应部分添加自己的优化器
-- print-freq  # 输出训练情况的频率
--save-freq   # 保存模型的频率
--resume  # 继续从某个模型接着训练，须填写该模型路径
--cuda    # 判断是否有 GPU 可用
```

**1.2 运行代码**

以 AlexNet_BN 为例，运行以下代码即可训练模型。

```
python main.py --arch AlexNet_BN --epochs 160 --batch-size 128 --lr 0.1
```

输出如下：

```
learning rate:0.1
Epoch: [0][0/391]   Time 2.734 (2.734)  Data 2.153 (2.153)  Loss 4.6521 (4.6521)    Prec@1 0.000 (0.000)    Prec@5 3.125 (3.125)
Epoch: [0][10/391]  Time 0.617 (0.874)  Data 0.001 (0.196)  Loss 4.6923 (4.6703)    Prec@1 1.562 (1.136)    Prec@5 4.688 (5.469)
```

其中：Prec@1 预测排名第一的标签是正确标签的概率；Prec@5 代表的是预测排名前

五的标签中有正确标签的概率；Time，是指训练一个批次所需要的时间；Data，是加载数据所耗费的时间，loss 是损失函数的大小。

### 1.3 输出的文件说明

① checkpoint.pth.tar →保存了最新的训练模型
② model_best.pth.tar →保存了最好的训练模型
③ checkpoint_*.pth.tar →保存的一些中间模型
④ net-train.pdf →训练曲线
⑤ stats.mat →保存了训练结果（包含 loss、准确率）
注：在测试模型和生成测试集的 score 时，应选择“model_best.pth.tar”

## 2. 测试模型性能（evaluate_prediction.py）

说明： 使用训练好的模型在测试集上进行测试，得到文件 test_prediction.csv，在竞赛网页指定位置上传该文件即可得到排名。

### 2.1 参数介绍

--data # 数据集文件所在路径，默认为代码中的 data 文件夹
--result # 训练结果存储路径，默认为代码中的 Results 文件夹
--arch # 网络架构，如 AlexNet_BN， resnet20
--batch-size # 一次提取特征的的样本数目
--model-dir # 要测试的模型路径

### 2.2 运行代码

以 AlexNet_BN 为例，运行以下命令可生成文件 test_prediction.csv

```
python evaluate_prediction.py --arch AlexNet_BN --batch-size 128 --model-dir ./Results/AlexNet_BN-45.41/AlexNet_BN-45.41.pth.tar --result ./Results/AlexNet_BN-45.41
```

## 3. 生成测试集的 score（ evaluate_score.py）

说明： 使用训练好的模型在测试集上进行测试，得到预测分数文件 test_score.csv，这个文件最终**与源代码一起上交**。

### 3.1 参数介绍

--data # 数据集文件所在路径，默认为代码中的 data 文件夹
--result # 训练结果存储路径，默认为代码中的 Results 文件夹
--arch # 网络架构，如 AlexNet_BN， resnet20
--batch-size # 一次提取特征的的样本数目
--model-dir # 要测试的模型路径

### 3.2 运行代码

以 AlexNet_BN 为例， 运行以下命令可生成文件 test_score.csv

```
python evaluate_score.py --arch AlexNet_BN --batch-size 128 --model-dir ./Results/AlexNet_BN-45.41/AlexNet_BN-45.41.pth.tar --result ./Results/AlexNet_BN-45.41
```

## 4. 评估模型的参数量和 FLOPs（model_params_flops.py）

### 4.1 参数介绍

--arch　# 要评估的网络架构，如 AlexNet_BN

**4.2 运行代码**

以 AlexNet_BN 为例，运行以下命令即可得到如图所示的运行结果：红框标注的分别为模型的**参数量（2.3M）和 FLOPs（25.33MFLOPs）**。

python model_params_flops.py --arch AlexNet_BN


## 四、用户自定义

我们提供了一些空白文件、或可更改的函数以便用户改进网络架构、训练策略、优化算法、数据增广等实现更好的性能

1. **构建自己的网络架构**

   在空白文件“**./models/custom_model.py**”内，各位同学可以构建自己的网络架构

2. **数据增广**

   用户可以通过更改 image_preprocess.py 下的 **train_val_transforms()**函数去实现不同的数据增广方法

3. **训练策略**

   关于学习率，用户可通过修改函数 main.py()下的 adjust_learning_rate(optimizer,epoch)函数去实现 cosine 学习率、warm up 等方法。

4. **优化算法**

```
94      # define optimizer
95      if args.optimizer == 'SGD':
96          optimizer = optim.SGD(model.parameters(), args.lr, momentum=0.9, weight_decay=1e-4)
97      elif args.optimizer == 'custom':
98          """
99              You can achieve your own optimizer here
100         """
101         pass
102     else:
103         raise KeyError('optimization method {} is not achieved')
104
```

用户可在上图中红框部分实现自己的优化算法，可直接调用 PyTorch 中现有的函数。

注意：除以上提及的部分，所有代码可依据自身实际情况进行更改。

## 五、联系方式

对代码有疑问的同学可咨询：徐春荞　　QQ：2109899546