图像分类训练脚本README

一、项目概述

本项目基于PyTorch框架搭建，专注于图像分类任务。它构建了一套完整的流程，涵盖数据加载、预处理、模型训练以及结果可视化。通过该项目，用户能够便捷地开展图像分类实验，并深入分析模型性能，为实际应用提供有力支持 。

二、文件结构

```
项目根目录
├── train.py                          # 主训练脚本
├── helper
│   └── models.py                     # 模型获取模块
├── tools
│   └── get_image_label_file.py       # 生成标签文件工具
│   └── load_image.py                 # 图像处理演示脚本
├── dataset
│   ├── images                        # 图像数据存储目录
│   └── label_file
│       ├── label_file_train.json     # 训练集标签文件
│       └── label_file_val.json       # 验证集标签文件
```

三、环境配置

核心依赖安装

```
pip3 install torch torchvision matplotlib pillow scikit-learn
```

额外工具使用说明

- **数据集预处理工具**：运行以下命令，可按指定比例划分数据集并生成标签文件。默认划分比例为7:3，需要可以去代码里面更改。

```
python3 tools/get_image_label_file.py
```

- **图像处理演示**：执行以下命令，可进行图像卷积与池化的演示操作。`--input_img`参数用于指定输入图像的路径。

```
python3 tools/load_image.py --input_img "./tools/test.jpg"
```

四、参数说明

运行脚本时，可通过命令行参数对训练过程进行灵活配置：

1. **数据参数**
   - `--base_path`：图像数据的根目录，默认值为`"./dataset/images"`。需设置为有效的文件路径字符串，以此明确图像数据的存储位置。
   - `--label_file_train`：训练集标签文件的路径，默认值为`"./dataset/label_file/label_file_train.json"`。应设置为有效的JSON文件路径字符串，该文件包含训练图像的标签信息。
   - `--label_file_val`：验证集标签文件的路径，默认值为`"./dataset/label_file/label_file_val.json"`。需设置为有效的JSON文件路径字符串，该文件包含验证图像的标签信息。
2. **模型参数**
   - `--model_name`：选择的模型架构名称，如`"custom"`、`"resnet34"`等。
   - `--load_pretrained`：选择的是否加载预训练模型的权重，是建议学习的关键。
   - `--num_classes`：分类任务的类别数，默认值为`4`。需设置为正整数，根据实际分类任务的类别数量进行调整。
3. **训练参数**
   - `--batch_size`：训练时的批次大小，默认值为`32`。应设置为正整数，可根据GPU显存大小进行调整。较大的批次大小能提高训练效率，但可能引发显存不足问题。
   - `--learning_rate`：初始学习率，默认值为`0.001`。需设置为正浮点数，作为训练过程中的关键超参数，其大小会影响模型的收敛速度和性能表现。
   - `--epochs`：训练的总轮次，默认值为`50`。应设置为正整数，可根据数据集大小和模型复杂度来调整训练轮次。
4. **实验管理参数**
   - `--save_fig_name`：训练结果图表的保存名称，默认值为`"exp1"`。可设置为任意字符串，用于区分不同的实验结果图表。
   - `--gpu`：是否使用GPU加速训练，默认值为`False`。取值为布尔值，若设置为`True`，且系统具备CUDA环境，则会使用GPU进行计算。

五、逻辑思路

1. **数据准备**
   - `ImageFolderDataset`类负责从指定路径加载图像数据及其对应的标签信息。
   - `get_dataset_urbanpipe`函数依据训练集和验证集的不同需求，分别实施数据增强和预处理操作，并创建数据加载器。
2. **模型初始化**
   - 借助`get_model`函数从`helper.models`模块获取指定的模型架构，并加载预训练权重。
   - 将模型转移至指定的计算设备（CPU或GPU）上。
3. **训练过程**
   - `train`函数执行单个epoch的训练过程，包括前向传播、损失计算、反向传播以及参数更新。
   - 记录训练损失和准确率，并在每个epoch结束时输出展示。
4. **验证过程**
   - `test`函数在验证集上评估模型性能，计算验证损失和准确率。
   - 记录验证损失和准确率，并在每次验证结束时输出展示。
5. **结果可视化**

- 训练结束后，运用`matplotlib`绘制训练损失和准确率曲线，以及验证损失和准确率曲线。
- 将绘制好的图表保存为指定名称的图像文件。

六、模型训练使用方法

1. 在命令行中运行脚本，并根据实际需求调整参数。例如：

```
python3 train.py --base_path="./dataset/images" --
label_file_train="./dataset/label_file/label_file_train.json" --
label_file_val="./dataset/label_file/label_file_val.json" --
model_name="resnet18" --load_pretrained  --num_classes=4 --batch_size=32 -
-learning_rate=0.001 --epochs=50 --save_fig_name="exp1" --gpu
```

经常调整的参数就下面几个，不写就按照默认的

```
python3 train.py  --model_name="resnet18" --load_pretrained  --
batch_size=32 --learning_rate=0.001 --epochs=50 --save_fig_name="exp1"
```

2. 训练结束后，项目根目录下会生成训练结果图表文件，文件名由`--save_fig_name`参数指定。