《人工智能软件开发与实践》

（ 2023 学年 秋季 学期）

作

业

报

告

学　　号：_

姓　　名：____

班　　级：___

任课教师：____

作业报告

实验名称：　使用 CLIP 模型进行零样本图像分类

成绩：

实验类别： 验证/综合型实验　　**实验要求：** 1 人 1 组　**时间：** 2023 年 9 月 22 日

一、　实验目的

使用开源的中文 CLIP 模型，分别在已有的标准数据集和自己收集的图片集上做零样本图像分类任务。

二、实验内容

1. 下载数据集

数据集为 ELEVATER Benchmark 的图像分类基准的中文版，共包括 20 个图像分类数据集，如 Caltech-101、CIFAR-10、CIFAR-100、MNIST 等。已有开源的整理好的数据集，可直接接入 Chinese CLIP 的代码进行零样本分类。可选择其中的 1 至 2 个数据集进行实验（建议选择 CIFAR-100 数据集）。

Chinese-CLIP/zeroshot_dataset.md at master · OFA-Sys/Chinese-CLIP · GitHub

数据集说明

我们将20个数据集分别置于20个文件夹中，统一打包上传，用户通过点击上述链接即可下载全部数据。`ELEVATER_all.zip`解压后，将得到每个数据集的zip压缩包。选择对应的压缩包再次解压后，每个文件夹的内容如下所示：

```
${dataset_name}
├── index.json  # 个别数据集包含这个文件，仅用于提交ELEVATER benchmark
├── label_cn.txt  # 中文标签名文件，每一行一个类别名
├── label.txt  # 英文标签名文件，每一行一个类别名
├── test/
│   ├── 000/
│   ├── 001/
│   └── 002/
└── train/
    ├── 000/
    ├── 001/
    └── 002/
```

`${dataset_name}`表示每个数据集的文件夹路径，如`cifar-100`，里面包括`train`和`test`两个文件夹，每个文件夹包含了以id编号命名的文件夹，分别代表每一个类别。另外还包含3个文件，分别为中文标签名文件`label_cn.txt`和英文标签名文件`label.txt`。其中：

- 类别数在10个及以下的情况下，如10，类别的id分别为[0-9]
- 类别数在10个以上的情况下，如100，类别的id分别为[000-099]，即向左补零到3位数。这是为了保证我们的id是以字典序进行排序
- 每个id对应的类别标签名为标签文件中的第${id}行（0-index），如`0`即对应标签文件中的第0行的类别名，`099`对应的是标签文件的第99行类别名。

训练和测试集文件夹内包含的子文件夹用字典序排序的原因是因为我们的代码使用了torchvision的dataset，默认文件夹内数据按照类别归类子文件夹，按照文件名以字典序排序。

标签文件包含中文版和原版两个文件，我们的代码仅需使用`label_cn.txt`，`label.txt`仅供参考。文件内容为每一行1个类别名，示例如下：

```
飞机
汽车
……
```

`index.json`仅用于提交ELEVATER benchmark使用，且并非每个数据集都包含此文件。该文件的原因是ELEVATER官方评测部分数据集的测试集样本顺序经过调整，如需保证提交结果正常需要调整样本顺序。如遇到数据集包含此文件，则可在测试运行命令中加上`index.json`即可。

类似地，如您自行准备ImageNet数据，请将上述中文和英文标签文件放入`${dataset_name}`，并在其中创建相应文件夹，如`train`和`test`，将图片按照类别归档并放入对应文件夹，并保证其按字典序排序，如`000-999`，实现的文件结构和上述示例保持一致，即可实现零样本分类的数据准备。

2. 参考示例 Jupyter Notebook 和开源项目介绍，将开源代码下载至本地、搭建好本地环境后对自己选择的图像数据集做零样本图像分类实验。（建议选择参数量较少的预训练模型 CN-CLIP$_{\text{ViT-B/16}}$）。

https://clip-cn-beijing.oss-cn-beijing.aliyuncs.com/others/Chinese-CLIP-on-MUGE-Retrieval.ipynb

GitHub - OFA-Sys/Chinese-CLIP: Chinese version of CLIP which achieves Chinese cross-modal retrieval and representation generation.

CLIP（Contrastive Language-Image Pretraining）模型的结构是一个深度学习模型，具有卓越的多模态能力，能够处理文本和图像之间的关联。CLIP 的结构可以分为以下几个关键组件：

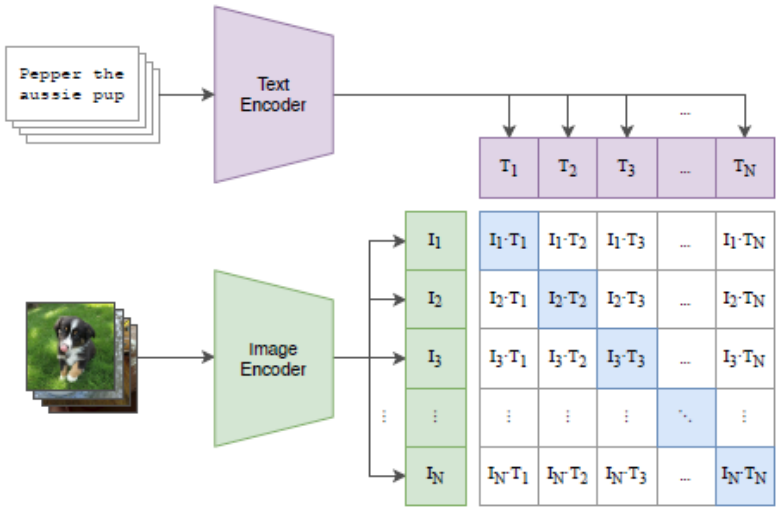
（1）. 图像编码器（Image Encoder）：采用视觉识别领域中常用的预训练模型，如 ResNet 或 Vision Transformer（ViT）。这个编码器将输入的图像转化为高维的特征向量，捕捉图像的语义信息。

（2）. 文本编码器（Text Encoder）：是一个多层的 Transformer 模型，用于将文本输入（例如描述图像的自然语言文本）转化为文本特征向量。这个编码器有助于理解文本的语义。

（3）. 多模态嵌入层（Multimodal Embedding Layer）：CLIP 将图像特征向量和文本特征向量进行连接，以形成多模态嵌入，从而在一个共享的嵌入空间中将文本和图像相关联。这个共享的嵌入空间允许模型衡量文本与图像之间的相似性。

（4）. 对比学习（Contrastive Learning）：CLIP 采用对比学习的方法，通过最大化相关图像和文本对的相似性以及最小化不相关对的相似性来训练模型。这有助于模型学习如何将相关的文本与图像嵌入更接近，而将不相关的文本与图像嵌入更远。

下图展示了 CLIP 模型的结构。实验中的中文 CLIP 模型与原 CLIP 模型结构相同，不同的是中文 CLIP 是在大规模的中文的图文训练集上预训练的，因此可以更好处理与中文有关的任务。



3. 下载使用了预训练模型 CN-CLIP$_{\text{ViT-B/16}}$ 在 CIFAR-100 数据集上微调后的模型参数再次进行实验，

并将其与直接使用预训练模型 CN-CLIP$_{ViT-B/16}$ 进行实验得到的分类准确率进行对比。

4. 自己搜集一些图片和对应标签，根据零样本数据分类的数据文档里的格式要求整理好数据的格式，然后再进行零样本分类实验。

5. （可选）在不微调模型的情况下如果想要提高零样本图片分类的准确率，可以通过制定合适的提示性模板来实现，具体操作可以通过阅读并修改源码 cn_clip/eval/cvinw_zeroshot_templates.py 来实现。

三、实验中每一步的操作（拷贝至此处，同时作为附件和报告再一份单独的记录）

a） 实验数据

① 加载已有的标准数据集，选择 1 至 2 个
② 记录自己收集的图片数据的类别和数量

b） 使用模型进行零样本图像分类

① 选择预训练模型进行实验
② 使用在 CIFAR-100 上微调后的模型再在 CIFAR-100 上进行实验，也可以尝试用它在别的数据集上实验，验证一下微调会对原来的预训练模型的泛化性造成什么影响

c） 修改提示性模板提高零样本图像分类准确性（如果有）

对哪些数据集的提示性模板做了什么样的修改

四、程序运行结果（将程序运行结果的截图拷贝至此处）

a） 实验数据

标准数据集选择 CIFAR-100，flower，flowers（flower 为老师提供的那个很小的花卉数据，flowers 是比较大的那个）此外自己的数据集选择了昆虫数据（insects），一共有三类，其类别和数量为：白星花金龟（176 张），蝗总科（517 张），四带虎天牛（157 张）。

b） 使用模型进行零样本图像分类

①模型运行调整

由于 Chinese-CLIP 预设的预测脚本无法直接在 Windows 系统中运行，我们对 evaluation.py 读取命令行参数部分进行了一些修改。首先对于 parse_args()中原始必需的变量，我们设置了一些默认值，如：

```
def parse_args():
    ds = "cifar-100"
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--datapath",
        type=str,
        # required=True,
        default="DATA/datasets/"+ds+"/test",
```

```
        help="Path to the test set for conducting zero shot evaluation.",
    )
    parser.add_argument(
        "--dataset",
        type=str,
        # default="imagenet",
        default=ds,
        help="Specified dataset.",
    )
```

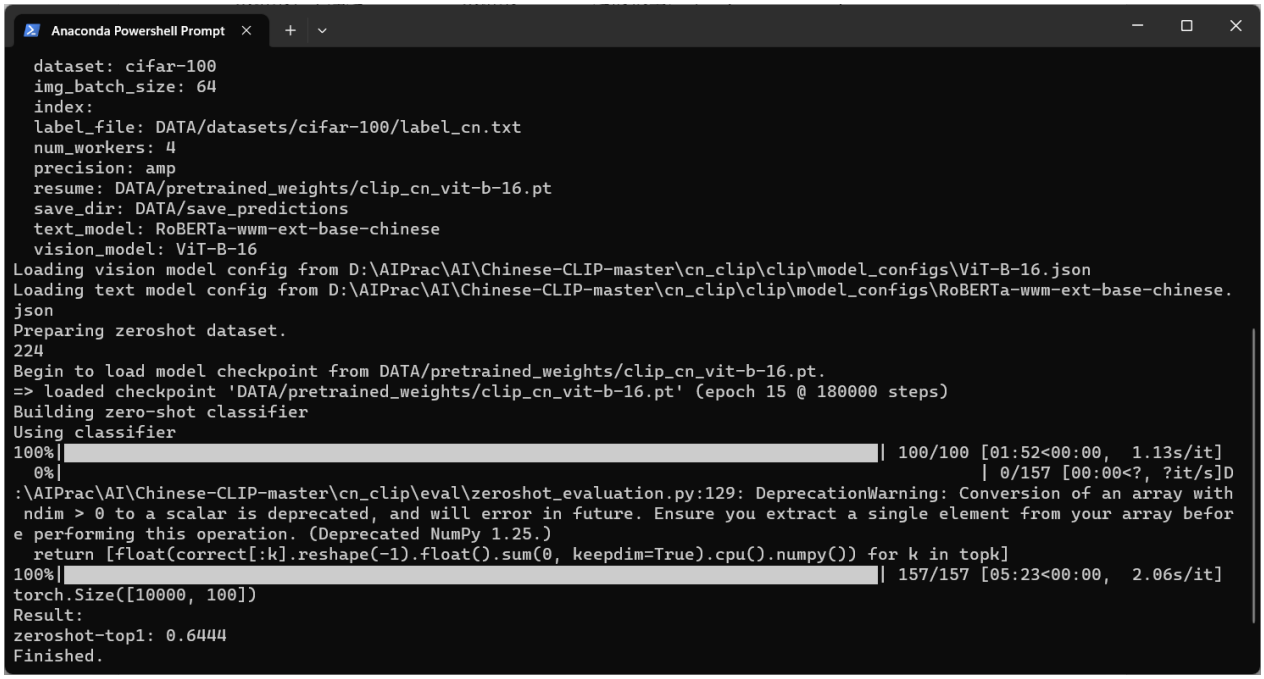
而在主要程序运行前对读入参数做进一步处理：

```
if __name__ == "__main__":
    args = parse_args()
    ds = args.dataset
    args.label_file = "DATA/datasets/"+ds+"/label_cn.txt"
    args.datapath = "DATA/datasets/"+ds+"/test"
    args.resume = "DATA/pretrained_weights/"+args.resume
```

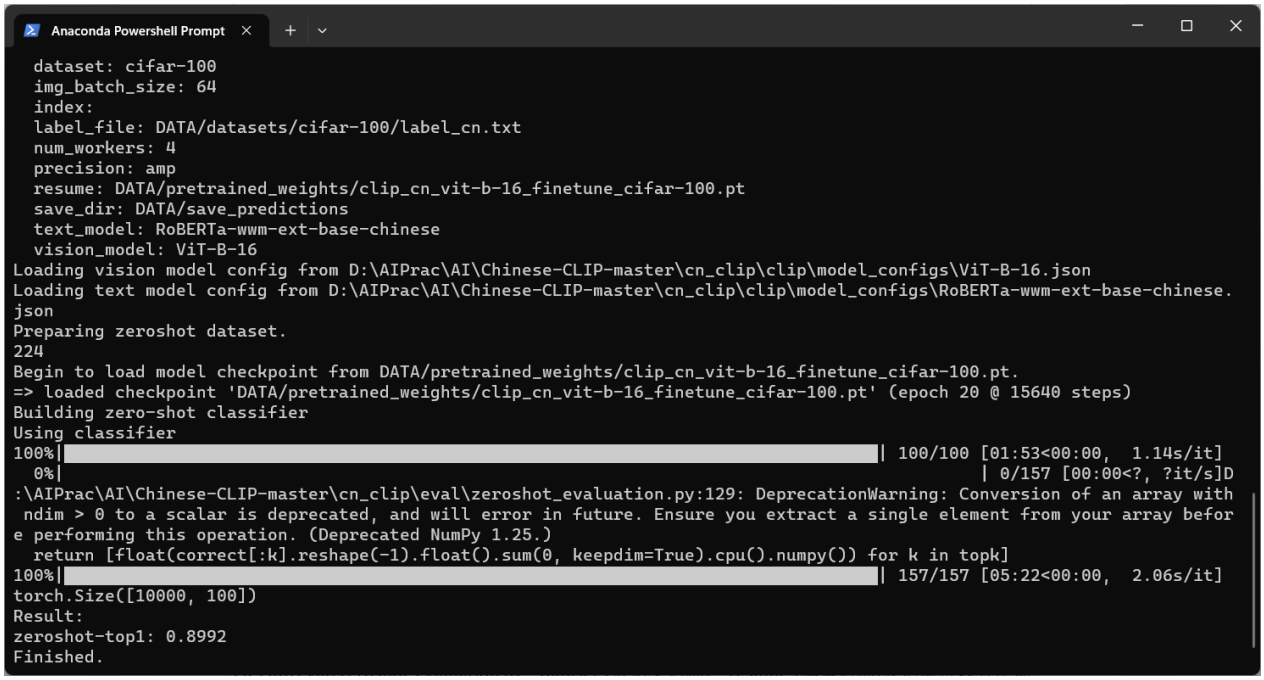
这样只要模型放在 DATA\pretrained_weights 目录，数据集按标准格式整理好并放在 DATA\datasets\{数据集名称}，通过包含--dataset 和—resume 参数的命令行指令 python cn_clip/eval/zeroshot_evaluation.py --dataset {数据集名称} --resume {模型名称}我们就可以方便地选取所用的模型，例如 python cn_clip/eval/zeroshot_evaluation.py --dataset flower --resume clip_cn_vit-b-16_finetune_cifar-100.pt。若不提供—dataset 及—resume 参数，默认选取 CIFAR-100 数据集和未经过 CIFAR-100 数据集 finetune 过的模型，即 clip_cn_vit-b-16.pt。

②进行实验

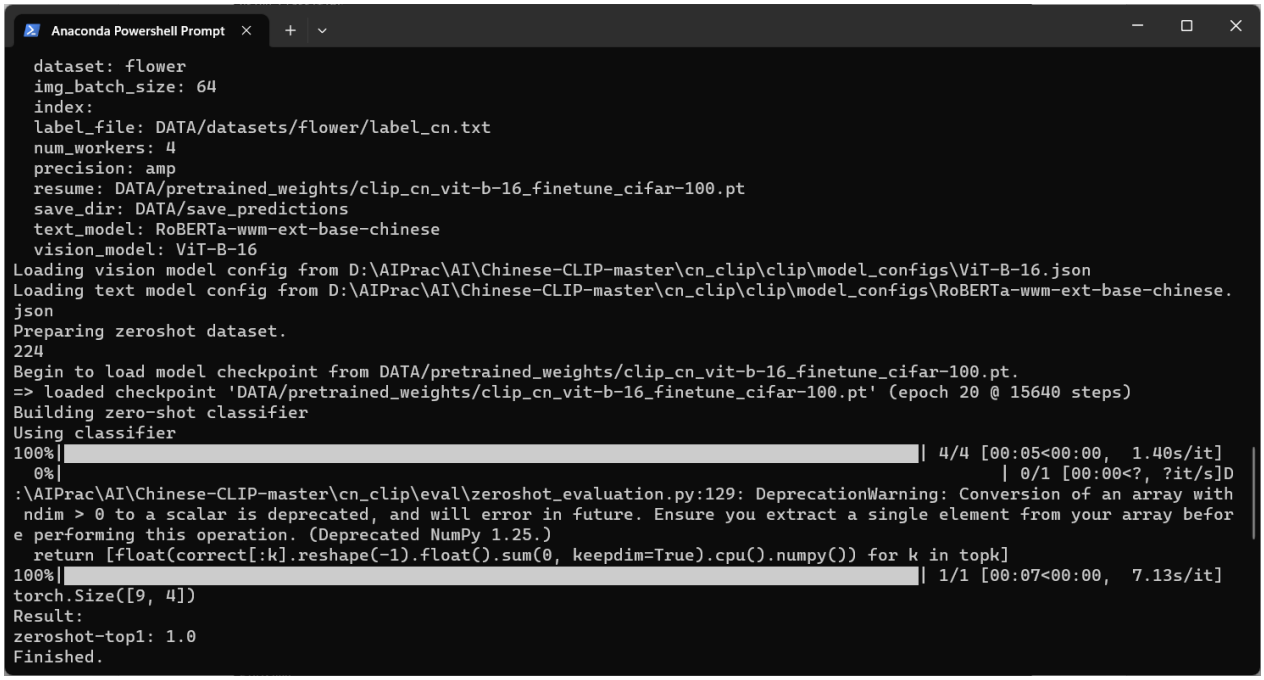
首先，使用预训练模型 CN-CLIP$_{\text{ViT-B/16}}$ 在 CIFAR-100 数据集上进行测试，所用指令为 python cn_clip/eval/zeroshot_evaluation.py --dataset cifar-100，结果如下，与 Chinese-CLIP 官方数据一致。

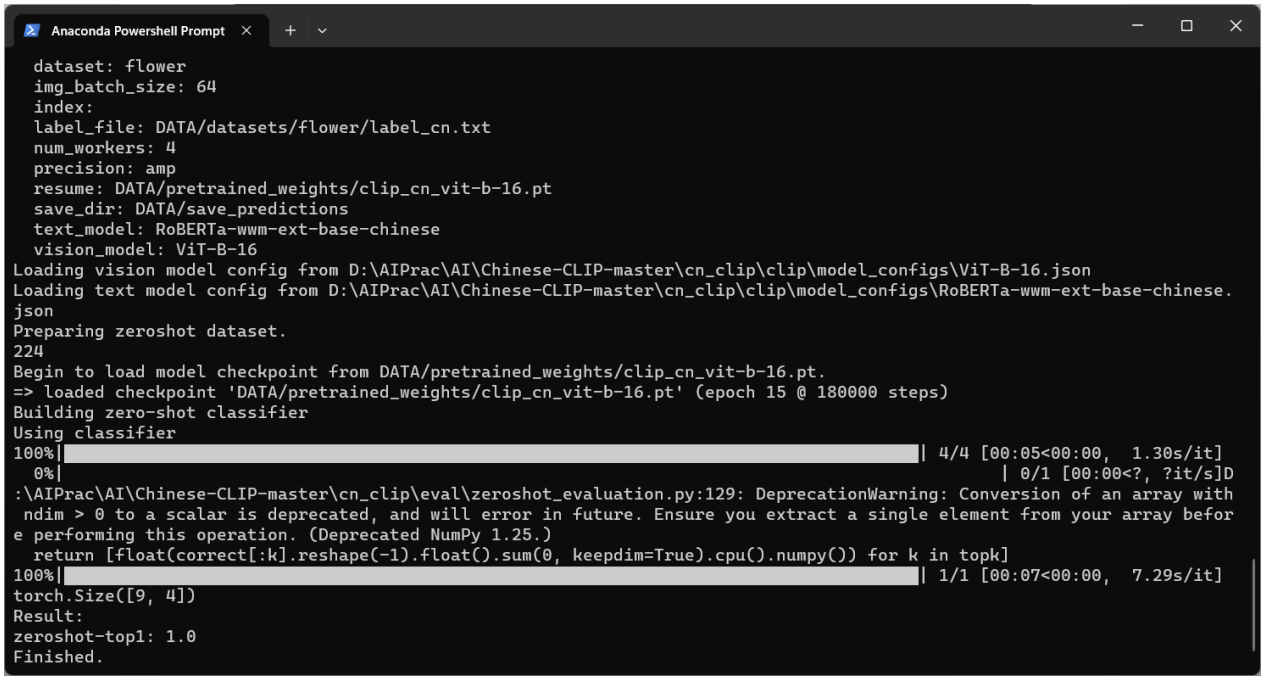


再用 finetune 后的模型权重测试，指令为 python cn_clip/eval/zeroshot_evaluation.py --dataset cifar-100 –-resume clip_cn_vit-b-16_finetune_cifar-100.pt，结果如下，不出意外地正确率提升了许多。



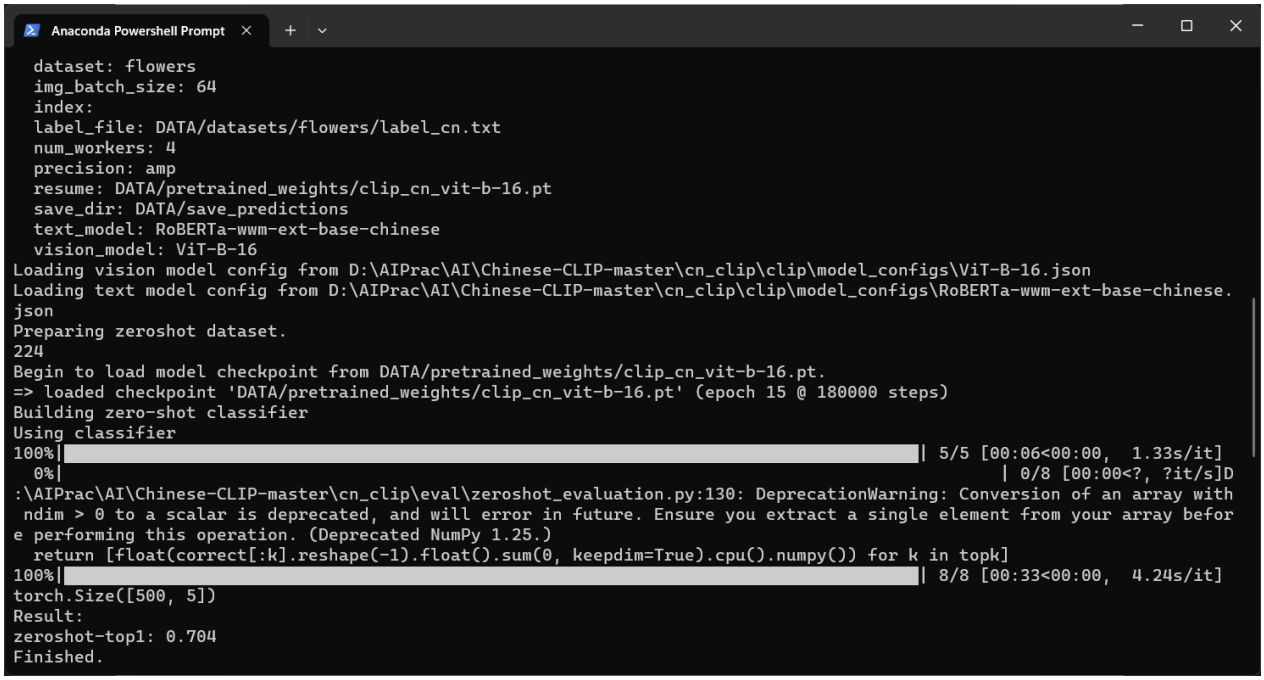
随后，采用 flower 数据集，在两个模型上的运行结果如下。



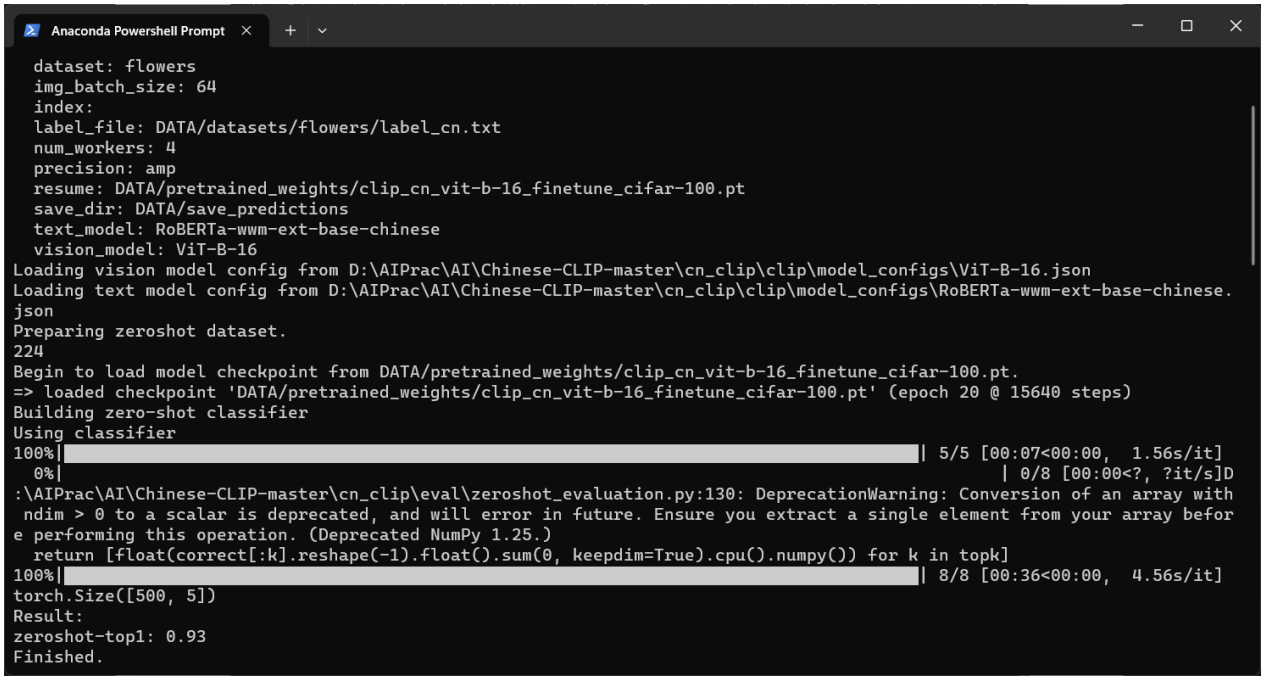


可见采用两种模型，flower 数据集的结果都是 1.0，主要原因为 flower 数据集较小。

对于较大的 flowers 数据集，未经微调模型的结果如下：

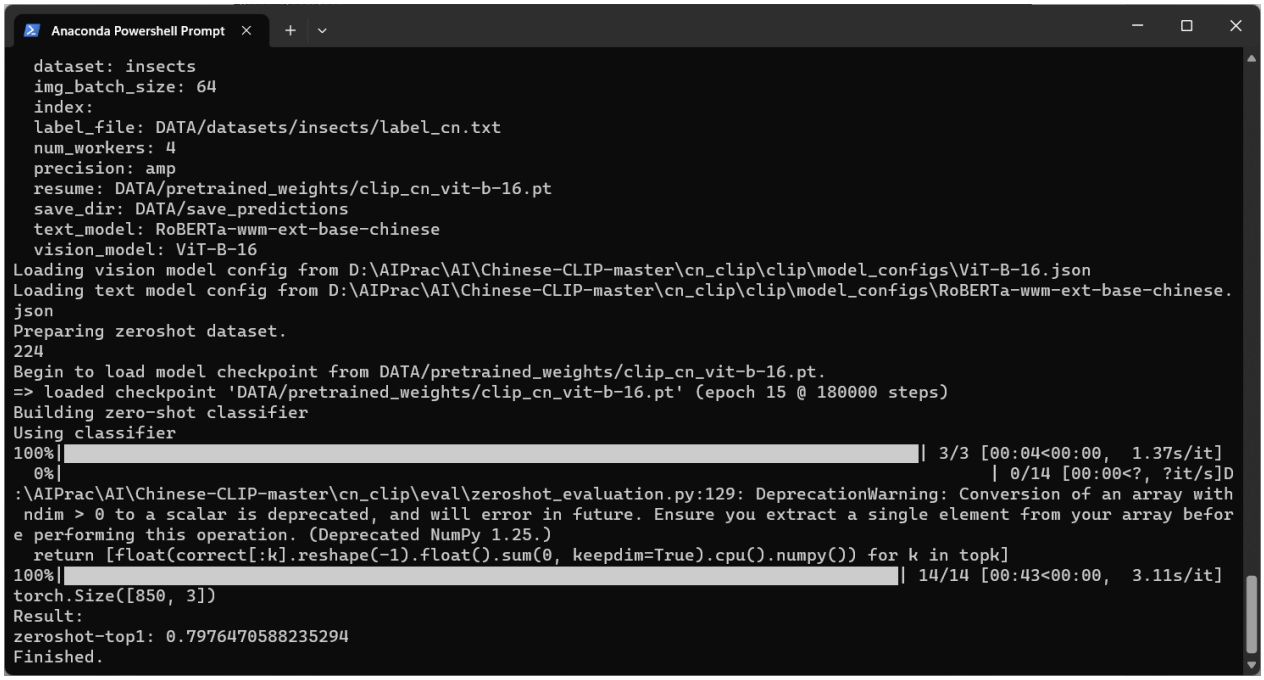


微调后的模型的结果：

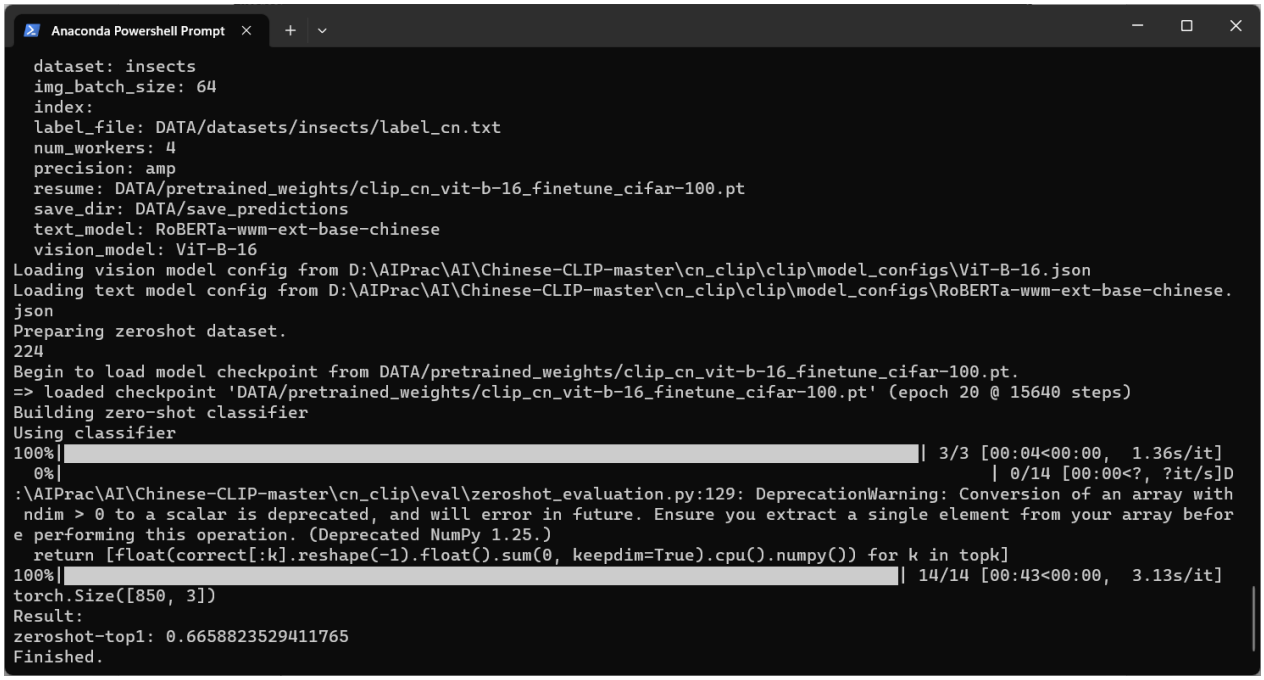


可见，微调过后模型的预测结果有了大幅的提升。显而易见的原因是 flowers 数据集中的五个类别都出现在了 CIFAR-100 数据集中，因此 CIFAR-100 数据集 finetune 过的模型对于 flowers 数据集有很好的预测结果。

最后采用我们收集的 insects 数据集进行实验。未微调过的 CN-CLIP$_{\text{ViT-B/16d}}$ 模型的结果如下：



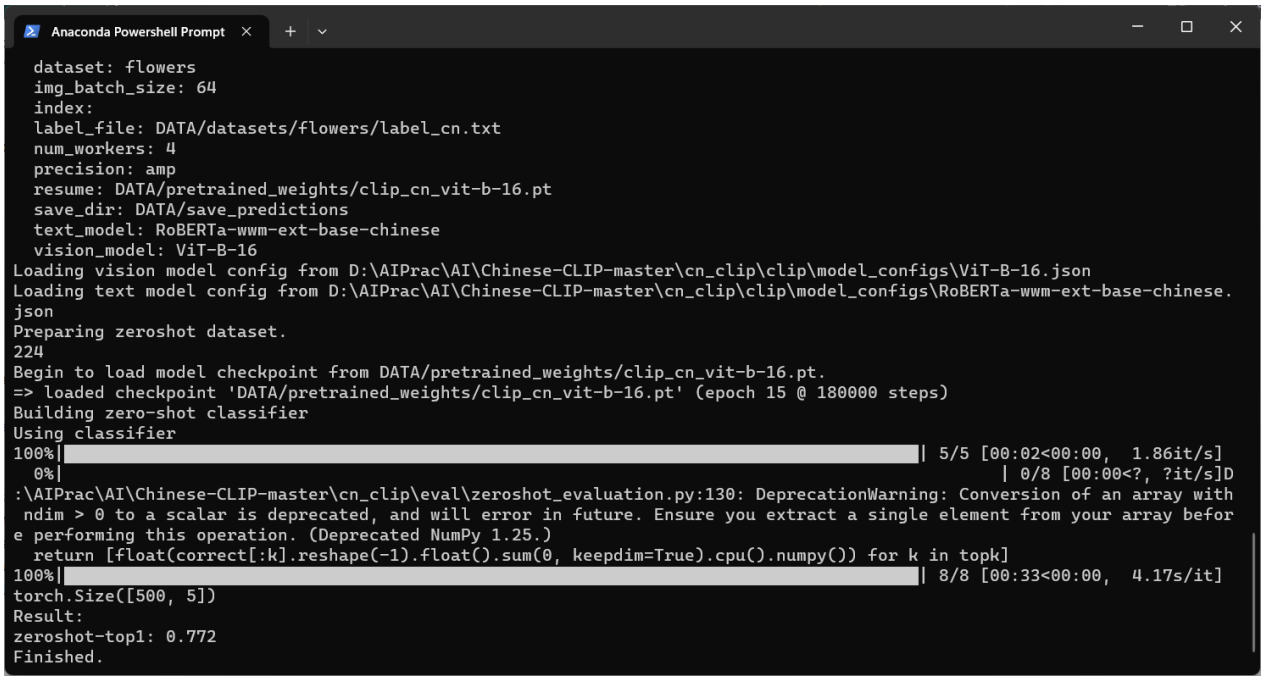
微调过的模型上的结果：



从结果可以看到，insects 数据集在微调后模型上预测的结果反不如未微调过的预训练模型，说明采用特定数据集对于模型权重进行微调会降低原本预训练模型的泛化性，尤其是微调过程中使用的目标任务和数据集与预训练模型的预训练任务和数据集差异较大的情况。这样，对于未见过的数据集，经微调后模型的表现可能会变差。

c） 修改提示性模板提高零样本图像分类准确性

首先，我们直接使用 cvinw_zeroshot_templates.py 中预设的提示性模板 `flower_templates`，在未微调的模型上进行 flowers 数据集的预测，结果如下：



结果从原来的 0.704 提升到了 0.772，较为可观。

接着，我们为 insects 数据集编写了一个简单的提示性模板，如下图所示：

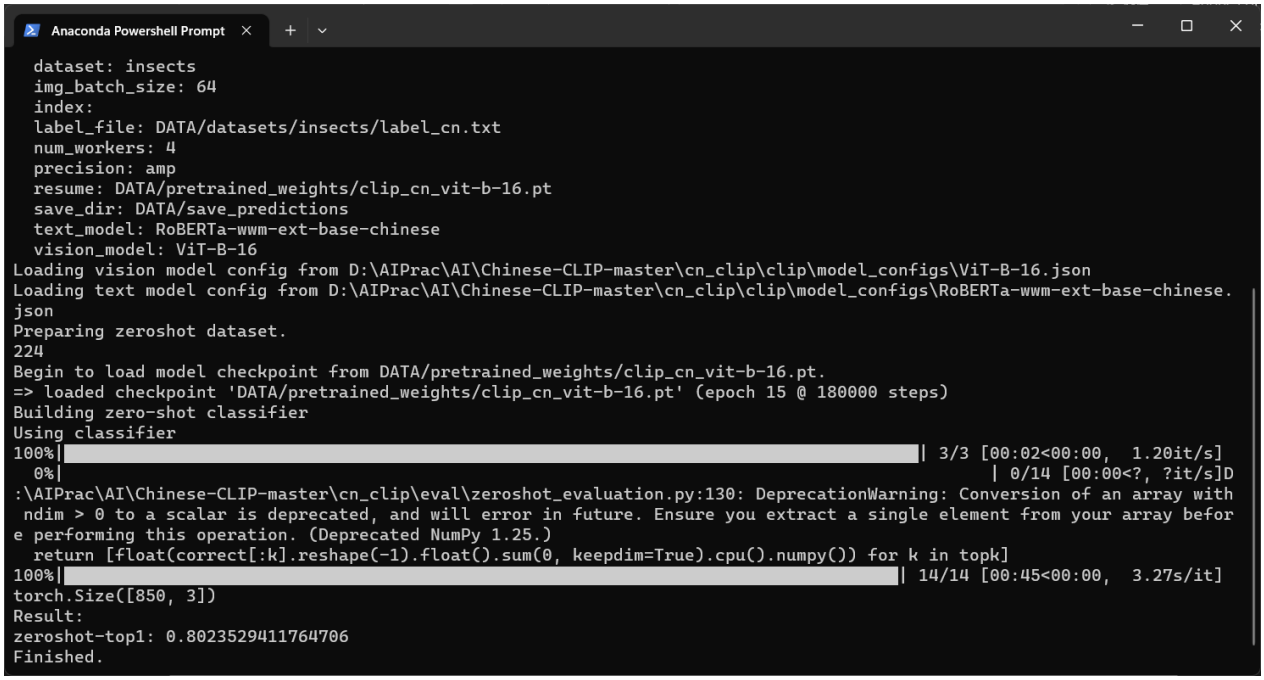
```
insects_templates = [
    lambda c: f"一张{c}的好照片",
    lambda c: f"一种叫{c}的昆虫的照片",
    lambda c: f"一种叫{c}的昆虫的特写照片",
    lambda c: f"质量好的{c}的照片",
    lambda c: f"一种叫{c}的昆虫的高质量照片",
    lambda c: f"一种叫{c}的昆虫的近距离照片",
    lambda c: f"一种叫{c}的昆虫的近距离照片",
    lambda c: f"{c}在植物上",
    lambda c: f"一种叫{c}的昆虫的高清照片",
]
```

并在 zeroshot_evaluation.py 中相应地做了添加，使程序识别、启用此模板：

```
from cn_clip.eval.cvinw_zeroshot_templates import (
    openai_templates,
    flower_templates,
    food_templates,
    aircraft_templates,
    eurosat_templates,
    country211_templates,
    insects_templates
)
```

```
template_dict = {
    "fgvc-aircraft-2013b-variants102": aircraft_templates,
    "food-101": food_templates,
    "oxford-flower-102": flower_templates,
    "eurosat_clip": eurosat_templates,
    "resisc45_clip": eurosat_templates,
    "country211": country211_templates,
    "openai": openai_templates,
    "insects": insects_templates
}
if args.dataset in template_dict.keys():
    templates = template_dict[args.dataset]
else:
    templates = template_dict['openai']
```

加入了此简单的提示性模板后，insects 数据集在未经微调的预训练模型上的预测结果有了一些提高，从 0.79 提高到了 0.80，如下图。



六、心得体会和遇到的困难

了解了 CLIP 模型的基本原理和使用方法。前期时对于 CLIP 模型的使用遇到了一些困难。