模式识别与机器学习实验四——Transformer模型的应用

2021113117-王宇轩

1. 实验环境

- 操作系统: Windows
- 编程语言：Python
- IDE：PyCharm
- 设备：GPU

2. 文件列表

文件名	内容
coop.py	CoOp程序
zeroshot.py	零样本分类程序
实验报告.pdf	实验报告

3. 实验内容

3.1 CLIP代码介绍

官方Github仓库给出了使用CLIP模型进行图片零样本分类的示例代码：

```
import os
import clip
import torch
from torchvision.datasets import CIFAR100

# Load the model
device = "cuda" if torch.cuda.is_available() else "cpu"
model, preprocess = clip.load('ViT-B/32', device)

# Download the dataset
cifar100 = CIFAR100(root=os.path.expanduser("~/.cache"), download=True, train=False)

# Prepare the inputs
image, class_id = cifar100[3637]
image_input = preprocess(image).unsqueeze(0).to(device)
```

```
text_inputs = torch.cat([clip.tokenize(f"a photo of a {c}") for c in
cifar100.classes]).to(device)

# Calculate features
with torch.no_grad():
    image_features = model.encode_image(image_input)
    text_features = model.encode_text(text_inputs)

# Pick the top 5 most similar labels for the image
image_features /= image_features.norm(dim=-1, keepdim=True)
text_features /= text_features.norm(dim=-1, keepdim=True)
similarity = (100.0 * image_features @ text_features.T).softmax(dim=-1)
values, indices = similarity[0].topk(5)

# Print the result
print("\nTop predictions:\n")
for value, index in zip(values, indices):
    print(f"{cifar100.classes[index]:>16s}: {100 * value.item():.2f}%")
```

主要逻辑：

1. 加载数据集。
2. 定义模型，调用了clip.py中的load，此调用返回两个对象（即model和preprocess），分别为：clip模型本身，该模型对应的图像预处理函数，将图像处理为模型的图片编码器接受的输入。
3. 用preprocess处理图片获取图片输入；连接tokenize后的人工prompt（a photo of a）与数据集的类别名，作为文本编码器的输入。
4. 将两个输入输入进对应的编码器，获取输出。
5. 处理输出，得到相似度，输出分类结果。

对该repo中代码的组织的简要介绍：

主要代码都在`clip/`目录下，最主要的调用是clip.py，其中完成了模型的构建（包括下载等）、tokenize等主要函数；model.py中实现了构建模型需要的许多功能，最重要的CLIP类也在此处实现，并通过`build_model()`构建为模型，clip.py中许多功能是调用model完成的。simple_tokenizer.py实现了一个简单的tokenizer，本次的实验也对其进行了调用。

3.2 数据集处理

本次实验中对Caltech-101数据集的处理与实验2类似，只是把ImageFolder调用时传入的transform变换设为了CLIP模型传回的preprocess。运行时，请将数据集目录与coop.py和zeroshot.py调整一致（类别文件夹所在目录，此外，我在做实验的过程中和实验二一样，把BACKGROUND文件夹移除了）。

3.3 CoOp实现

代码见coop.py，将coop.py放于`./CLIP`目录下即可运行。主要的模型定义为CoOp类，其中，定义了两个模型类PromptLearner和TextEncoder，分别是可学习的上下文Embedding层组成的Prompt和能够处理修改后的文本的文本编码器。模型的定义、训练和测试评估的实现思路与前两次实验基本一致。

3.4 Zero-shot实现

代码见zeroshot.py，与coop.py一样将coop.py放于`./CLIP`目录下即可运行。与CoOp实现不同的是，只实现了一个Clip模型类，其中文本编码器和图片编码器就使用clip模型的`text_encoder()`和`image_encoder()`。

4. 实验结果与分析

三次实验的`seed`分别选取的是42、10、21。

对于0-shot分类，三次实验的平均分数（以正确率为衡量）为0.8085。

对于1-shot分类，三次实验在训练50个epochs后，损失函数值均降至0.0062，平均分数为0.8374。

对于2-shot分类，三次实验在训练100个epochs后，损失函数值均降至0.0034，平均分数为0.8458。

对于4-shot分类，三次实验在训练100个epochs后，损失函数值均降至0.0033，平均分数为0.8471。

绘图如下。