华东师范大学数据科学与工程学院上机实践报告

课程名称：当代人工智能	**年级：22 级**	**上机实践成绩：**
指导教师：李翔	**姓名：高雅凡**	
上机实践名称：实验五	**学号：10225501401**	**上机实践日期：**
上机实践编号：	**组号：**	

- **我的 github 库：**gyf-123239/Modern-Artificial-Intelligence

一、实验内容与设计思想

本项目旨在解决基于配对的文本和图像数据进行情感分类的任务，情感标签分为三类：positive、neutral 和 negative。为了更好利用文本和图像这两种信息，本项目设计了一个多模态融合模型**(BERT + ResNet50)**，将两种模态的信息进行有效融合，提升情感分类的准确性。

二、使用环境

- Python 版本：使用的 Python 版本为 3.9.21，并通过 conda 虚拟环境进行管理。
- CUDA 版本：使用的 CUDA 版本为 11.5.2，cuDNN 版本为 8.3.2，支持 GPU 加速。
- PyTorch 版本：使用的是 PyTorch GPU 版本，确保支持 CUDA。

三、实验过程

3.1 总体的实验过程

（1）数据加载与预处理：

- 加载训练数据，**文本数据**通过 BertTokenizer 进行编码，转化为模型输入的格式；**图像数据**通过 torchvision.transforms 库进行处理，并进行**缩放、随机翻转和旋转**等数据增强。
- 进行训练集和验证集的划分，使用 train_test_split 进行 80%-20% 的划分。
- 创建数据加载器：训练集和验证集分别通过 DataLoader 创建，支持批量加载。

（2）模型初始化与训练：

- 模型构建：我定义了一个多模态模型 MultimodalModel，该模型结合了文本模型

（BERT）和图像模型（ResNet50）。根据需要，模型可以**只使用文本数据**、**只使用图像数据**，或者**两者结合**。

- 文本模型：BERT 提供了一个基于 transformer 的语言表示模型。
- 图像模型：使用了 ResNet50 作为图像特征提取网络，去除最后一层分类头，保留前面的卷积层和全连接层提取特征。
- 融合策略：通过 torch.cat 将文本特征和图像特征拼接，送入一个全连接层进行分类。
- 模型训练：训练过程采用**交叉熵损失函数**和 **AdamW 优化器**，并使用**早停法**防止过拟合。训练结束后，通过绘制 **loss 曲线图**，可视化模型训练效果。

（3）模型预测：

- 在测试阶段，使用训练好的模型对测试集进行预测，输出每个样本的预测标签（tag）。预测结果通过 save_predictions 函数保存到文件（test_predictions.txt）中。

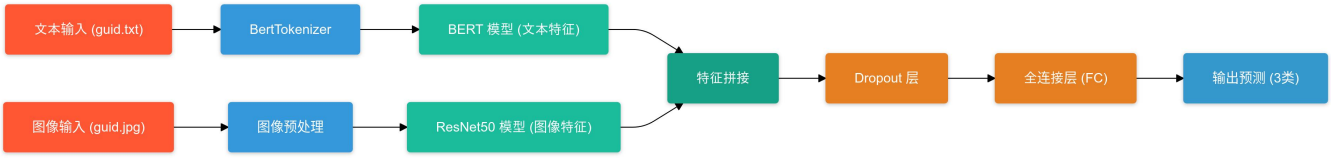
3.2 模型设计

模型结构表：

模块	输入	处理方式	输出
文本输入	文本数据 (guid.txt)	使用 BERT Tokenizer 对文本进行编码	输入到 BERT 模型
BERT 模型	Tokenized Text	提取文本的上下文信息，通过 Transformer 模型提取文本特征	文本特征向量 (Size: 768)
图像输入	图像数据 (guid.jpg)	图像经过预处理 (Resize, Flip, Rotation 等)	输入到 ResNet50 模型
ResNet50 模型	预处理后的图像	提取图像特征，移除分类头，保留中间特征层	图像特征向量 (Size: 2048)
特征融合	文本特征，图像特征	将文本特征和图像特征连接在一起	连接后的特征向量 (Size: 2816)
Dropout 层	融合后的特征向量	Dropout 正则化	去除部分神经元的激活

全连接层	**Dropout 后的特征向量**	**经过全连接层进行分类**	**分类输出** **(3 类情感标签)**

模型结构图：



总结：该多模态融合模型结合了文本和图像特征，通过多模态学习有效地对数据进行分类。它先提取每种模态的特征，再将它们融合起来，进行最终的预测。

为什么会设计这样的模型：

（1）模型效果好

通过之前在课程中曾经完成过的实验和学习过的模型，我们可以发现对于文本分类任务来说，BERT 能够通过预训练的语言模型理解文本的上下文和语义，模型效果较好；对于图像处理任务来说，ResNet50 是一个强大的卷积神经网络，我可以通过移除 ResNet50 的最后一层分类头，利用它来提取图像特征，取得较好的效果。

（2）多模态特征融合

我将来自文本和图像的特征通过 torch.cat()进行拼接，融合了两种模态的信息。这种特征级融合方法使得模型能够利用文本和图像信息的互补性，从而提高预测的准确性。

3.3 模型亮点

- 强大的预训练模型：文本特征使用了 BERT 预训练模型，图像特征则使用了 ResNet50，这些都是在各自领域表现非常优秀的模型。
- Dropout 正则化：在多模态特征拼接后使用 Dropout 层，有助于提高模型的泛化能力，防止过拟合。
- 可调节：我的模型可以根据需求选择只使用文本或图像，或者两者同时使用。
- 早停法：训练模型时我使用了早停法，防止模型过拟合，节省训练时间。

3.4 模型调参

模型设计好之后，并不是一开始效果就很好，需要我进行超参数的调整，找到适合的超参数，才让模型效果变得更好，提升模型性能并控制过拟合。调整超参数：batch_size、learning_rate、patience、dropout_rate 等。

（1）学习率（1e-6）

经过对学习率的调整，我选择了 1e-6 作为该模型比较适合的学习率。

1e-5: `Early stopping... Best epoch: 2, Best Val Accuracy: 0.7125`

- 当学习率较大（1e-5）时，我们可以发现，模型很早就**过拟合**早停了，这可能是因为，更新幅度较大，直接跨越了损失函数的最小点，导致无法收敛，而开始过拟合。
- 当学习率较小（1e-7）时，我们可以发现，模型**收敛速度很慢**，因为更新幅度过小，导致在训练轮次内无法充分优化。
- 当学习率为 1e-6 时，这是我找到的一个比较好的学习率，此时模型效果较好。

```
Early stopping... Best epoch: 8, Best Val Accuracy: 0.7462
```

（2）dropout_rate（0.1）

0.1：`Early stopping... Best epoch: 8, Best Val Accuracy: 0.7462`

0.75：`Early stopping... Best epoch: 10, Best Val Accuracy: 0.7350`

0.5：`Early stopping... Best epoch: 9, Best Val Accuracy: 0.7362`

不同 dropout 的训练效果相差不大，但是 dropout 为 0.1 时准确率最高，因此我选择了 0.1。

- **最终多模态融合模型在验证集上的结果：准确率：0.7462**

3.5 消融实验结果

只输入文本：

```
Epoch [7/20], Train Loss: 0.6219, Train Acc: 0.7416, Val Loss: 0.6785, Val Acc: 0.7238
```

只输入图片：

```
Epoch [16/20], Train Loss: 0.7452, Train Acc: 0.6659, Val Loss: 0.7876, Val Acc: 0.6288
```

从消融实验结果可以看出，仅输入文本的效果较好，只略低于多模态融合模型；而只输入图片的效果略差，准确率较低。

这个结果表明，文本信息对情感分类任务有**较强的影响力**，且 BERT 模型能够有效提取情感相关特征；而图像信息的**贡献较小**，可能是因为图像与情感的相关性较低或提取的视觉特征不足以显著提升分类性能。因此，文本在该任务中的作用**更为关键**，图像仅作为补充信息。

3.6 实验中遇到的问题

（1）程序在 GPU 上运行时内存不足

- 调整 batch_size：减小 batch_size 到 8，减少了每次迭代的数据量，降低 GPU 内存需求。
- 设置 max_length：缩短了文本张量的长度，减少了文本处理的内存需求。

（2）遇到 utf-8 无法解码的字节

```
UnicodeDecodeError: 'utf-8' codec can't decode byte 0xa1 in position 139: invalid start byte
```

解决： `with open(text_path, 'r', encoding='utf-8', errors='ignore') as f:`

使用 errors='ignore'，遇到无法解码的字节序列，忽略它们，而不是抛出异常。

（3）加载数据错误

在划分训练集和验证集后，加载数据时会遇到报错，这需要我们重置索引，避免因索引错乱导致没有办法正常加载数据：

```
train_dataset.text_data = train_data.reset_index(drop=True)
```

（4）过拟合严重

```
Epoch [1/20], Train Loss: 0.7908, Train Acc: 0.6438, Val Loss: 0.6663, Val Acc: 0.7250
Epoch [2/20], Train Loss: 0.5853, Train Acc: 0.7612, Val Loss: 0.6350, Val Acc: 0.7175
Epoch [3/20], Train Loss: 0.4432, Train Acc: 0.8322, Val Loss: 0.6766, Val Acc: 0.7100
Epoch [4/20], Train Loss: 0.3100, Train Acc: 0.8919, Val Loss: 0.7866, Val Acc: 0.6950
Epoch [5/20], Train Loss: 0.1977, Train Acc: 0.9356, Val Loss: 0.8088, Val Acc: 0.7037
```

如上图，我刚开始设计的多模态融合模型过拟合问题非常严重，为此我采取了一些措施:

- 添加 dropout 层：

在 MultimodalModel 中，我添加了 self.dropout，并在模型的全连接层输出前应用了 Dropout。它可以通过随机丢弃部分神经元来防止模型过度依赖某些特定特征，从而减小过拟合的风险。

- 数据增强：

在 image_transform 中我使用了**图像增强技术：随机水平翻转、随机垂直翻转、随机旋转**。这些数据增强技术通过增加数据的多样性，帮助模型更好地泛化，降低了过拟合的可能性。

- 权重衰减：

在优化器中我使用了 weight_decay=0.01，这相当于 L2 正则化，它通过惩罚较大的模型权重，进一步防止了模型的过拟合。