实验五：多模态情感分析

10225501437 寇璟奕 当代人工智能

Github 仓库：https://github.com/yyy09k/AI.git

Ⅰ 实验背景与目的

随着社交媒体和多媒体数据的普及，情感分析不仅依赖于文本，还需要结合图像等多模态信息。传统的文本情感分析方法无法全面捕捉情感信息。

本实验通过多模态融合，结合 RoBERTa（文本）和 ResNet50（图像），构建一个情感分析模型。实验的目的是验证多模态数据融合在情感分析中的效果，并通过对比实验探讨仅文本、仅图像和多模态融合的性能差异。

Ⅱ 模型选择与实现

一、模型概述

1 架构

我采用了多模态深度学习模型，将文本与图像信息相结合，通过 **RoBERTa** 和 **ResNet50** 模型提取文本和图像特征，并将它们融合用于情感分类。简单架构图示如下：

```
[开始]
    ↓
[输入：文本 + 图像]
    ↓
[文本处理：RoBERTa 提取文本特征]
    ↓
[图像处理：ResNet50 提取图像特征]
    ↓
[特征融合：拼接文本和图像特征]
    ↓
[分类：全连接层进行情感分类]
    ↓
[输出：情感预测（正面、负面、中立）]
    ↓
[结束]
```

2 模型设计亮点

①多模态特征融合：将来自文本和图像的特征有效融合。文本特征通过 RoBERTa 提取，图像特征通过 ResNet50 提取。通过将两种模态的特征拼接后，模型能够在最终的分类层前进行联合训练，利用多模态数据互补的优势提高情感分类的性能。

实现方式：使用 torch.cat 函数将提取后的文本和图像特征向量拼接在一起；拼接后的特征向量送入全连接层进行最终分类。通过这种方式，模型能够同时捕捉文本和图像中的情感信号，提升情感分析的鲁棒性和准确性。

```
combined_features = torch.cat((image_features, text_features), dim=1)
```

②利用预训练模型加速训练：利用 RoBERTa 和 ResNet50 两种强大的预训练模型，通过迁移学习加速训练过程并提高模型的准确性，避免过拟合。RoBERTa 通过在大规模语料库上的预训练，可以捕捉到丰富的文本语义信息，而 ResNet50 则可以从图像中提取高级特征。

③学习率调度器：为在训练中优化学习率，我引入**学习率调度器（ReduceLROnPlateau）**。该调度器根据验证集上的损失变化动态调整学习率。当验证损失停止下降时，学习率自动降

低，可有效避免训练中学习率过高导致的震荡或过低导致的学习缓慢问题，提升了模型收敛性。

```
scheduler = ReduceLROnPlateau(optimizer, mode='min', patience=2, factor=0.1, verbose=True)
```

④TensorBoard 可视化训练过程：集成 TensorBoard，能够实时可视化训练过程中的损失、准确率等指标，这使得模型的训练过程更加可控且易于调优。且提供直观的图表，可以帮助监控每个 epoch 的训练表现，及时发现潜在的问题并进行调整。

实现方式：在训练过程中，我使用 SummaryWriter 将训练过程中的损失、准确率等关键指标记录到 TensorBoard 中。通过图表，可以清晰地看到训练的进展，并基于此做出学习率调整、过拟合检测等。

```
writer = SummaryWriter(log_dir='./runs/experiment_1')
```

⑤collate_fn 数据处理优化：针对多模态数据，我设计了自定义的 collate_fn 函数，动态处理变长文本输入和图像数据。此设计确保了每个批次的数据在输入模型前能够正确对齐，特别是在处理不同模态（文本和图像）时，能够保持数据的一致性和高效性。

实现方式：对文本输入，collate_fn 动态计算最大文本长度并对文本进行填充，使所有文本输入具有相同长度；对图像输入，torch.stack() 被用于批量堆叠图像数据，确保每次迭代时图像的维度一致。

```
images = torch.stack(images, 0)  def collate_fn(batch):
```

⑥Dropout 防止过拟合：在模型的全连接层中引入 nn.Dropout(0.5)，每次前向传播时，随机丢弃 50% 的神经元，以减少模型对训练数据中噪声的过度拟合，从而提升模型在未见数据上的泛化能力。

二、？为什么选择这两个模型

1 RoBERTa 模型的选择

①文本处理方面，是对 BERT 模型的优化：通过更大规模的训练数据、更长时间的训练及去除 BERT 中的 Next Sentence Prediction 任务，显著提高了性能。

②对于情感分析任务，文本数据常含有复杂的上下文关系：而 RoBERTa 为 Transformer 架构，能更好地捕捉文本中的长程依赖，并提取出情感信息。相比于传统的 LSTM 或 CNN 模型，提供了更精准的文本特征表示。

★RoBERTa 的适配性：

①情感分析任务中，文本具有较强的上下文依赖性。RoBERTa 通过自注意力机制能够捕捉这些长程依赖关系。

②RoBERTa 是预训练模型，其在大规模数据集上的训练使它能从语言学的角度更好地理解文本。因此，我选择 RoBERTa 模型提取文本中的情感特征。

2 ResNet50 模型的选择

ResNet50 是一种深度卷积神经网络，采用残差连接技术，并避免了深层网络中的梯度消失问题，能够从图像中提取出情感相关的特征（面部表情、场景背景等），有助于模型更好理解情感。

★ResNet50 的适配性：ResNet50 能很好从图像中提取出情感相关的特征，与 RoBERTa 提取的文本特征相结合，形成完整的情感表达。与传统的图像分类模型相比，ResNet50 的残差结构更有效地解决了深层网络训练中的问题，因此我认为非常适配于本次实验任务。

Ⅲ 实验过程与问题解决

一、实验思路

1 数据加载与预处理

使用 MultiModalDataset 类加载图像和文本数据，并通过 collate_fn 函数进行批次处理。文本数据通过 RoBERTa 分词器进行编码，并进行 padding；图像数据通过 torch.stack()合并。确保两个模态的数据格式一致，适合输入模型。

2 模型构建

构建多模态情感分析模型 MultiModalModel，分别使用 RoBERTa 和 ResNet50 提取文本和图像特征，特征通过拼接的方式融合，最终通过全连接层进行分类（positive、negative、neutral）。

3 训练与优化

使用 Adam 优化器和交叉熵损失函数进行训练。训练过程中，用 tqdm 包为训练和验证过程添加进度条，帮助实时查看训练进度。进度条提供每个 batch 的处理时间以及当前训练状态，便于观察训练的速度与效率。同时结合学习率调度器（ReduceLROnPlateau）动态调整学习率，并使用 Dropout 技术防止过拟合。

4 验证与评估：

每个 epoch 结束后，评估模型在验证集上的表现，并记录训练损失及准确率。利用 TensorBoard 进行实时可视化，帮助监控训练过程。

5 测试与结果输出

训练完成后，加载最优模型，在测试集上进行情感预测，结果保存至 result.txt 文件中。

```
Testing: 100%|██████████| 256/256 [01:16<00:00,  3.33batch/s]
Created result.txt and added results
```

二、实验问题及解决措施

Q1 【文本编码问题 UnicodeDecodeError】

默认用 utf-8 编码加载文本文件时，出现 UnicodeDecodeError 错误，文本编码不一致，无法正确解析文本文件的编码。

Solution：

我尝试了多种编码方式读取文本文件，最终确定文本文件编码为 ISO-8859-1 并指定在代码中，确保能够正确读取文件内容。

```
with open(text_path, 'r', encoding='ISO-8859-1') as file:
    text = file.read()
```

Q2 【ValueError：传递 tokenizer 参数出错】

调用 RoBERTa 分词器时，出现 ValueError，提示 Too many dimensions (expected 1 or 2)。

Solution：

传递给分词器的文本参数应符合要求：为字符串并且使用了 return_tensors="pt" 返回 PyTorch 张量，以避免维度不匹配。

```
text_input = self.tokenizer(text, return_tensors="pt", padding=True, truncation=True, max_length=512)
```

Q3 【FileNotFoundError: [Errno 2] No such file or directory】

在图像文件的路径上。路径拼接时使用了混合的分隔符 \ 和 /，路径错误，导致了

Windows 系统无法正确找到文件。因为路径分隔符 Windows 会自动选择 \，而 Linux/macOS 则选择 /。

Solution：

检查路径分隔符，并使用 os.path.join()自动处理路径拼接问题，确保路径分隔符的一致性。该方法会根据操作系统自动选择正确的路径分隔符，确保路径在不同操作系统下兼容。

```
image_path = os.path.join(self.data_folder, f"{guid}.jpg")
text_path = os.path.join(self.data_folder, f"{guid}.txt")
```

Q4 【IndexError: Target 2726 is out of bounds】

模型的输出类别数量和实际标签的类别数量不匹配，导致标签的值超出了模型输出的类别范围。

Solution：

实验中使用的损失函数是 CrossEntropyLoss，它要求标签值必须在 [0, num_classes-1] 范围内。而 num_classes 设置为 3，因此标签值的最大值需小于 3。通过调整模型的输出层，确保其输出类别数为 3（即正面、负面、中立）。

Ⅳ 实验结果分析及思考

一、多模态融合模型实验结果分析

实验训练过程结果如图：

```
Training Epoch 1/5: 100%|██████████| 1600/1600 [38:19<00:00,  1.44s/batch]
Validating Epoch 1/5: 100%|██████████| 400/400 [01:40<00:00,  3.98batch/s]
Epoch 1/5, Train Loss: 0.8765447524003684, Train Accuracy: 60.78%
Validation Loss: 0.8439121453231201, Validation Accuracy: 63.12%
Model saved to model.pth
Training Epoch 2/5: 100%|██████████| 1600/1600 [38:55<00:00,  1.46s/batch]
Validating Epoch 2/5: 100%|██████████| 400/400 [01:43<00:00,  3.87batch/s]
Epoch 2/5, Train Loss: 0.752145981953945, Train Accuracy: 67.41%
Validation Loss: 0.6883869412727655, Validation Accuracy: 72.25%
Model saved to model.pth
Training Epoch 3/5: 100%|██████████| 1600/1600 [38:55<00:00,  1.46s/batch]
Validating Epoch 3/5: 100%|██████████| 400/400 [01:29<00:00,  4.45batch/s]
Epoch 3/5, Train Loss: 0.6086835673474706, Train Accuracy: 75.19%
Validation Loss: 0.7451830855803564, Validation Accuracy: 66.88%
Model saved to model.pth
Training Epoch 4/5: 100%|██████████| 1600/1600 [44:06<00:00,  1.65s/batch]
Validating Epoch 4/5: 100%|██████████| 400/400 [01:30<00:00,  4.43batch/s]
Epoch 4/5, Train Loss: 0.47436886908719317, Train Accuracy: 81.88%
Validation Loss: 0.8105163441202603, Validation Accuracy: 67.12%
Model saved to model.pth
Training Epoch 5/5: 100%|██████████| 1600/1600 [47:14<00:00,  1.77s/batch]
Validating Epoch 5/5: 100%|██████████| 400/400 [01:30<00:00,  4.44batch/s]
Epoch 5/5, Train Loss: 0.3518319985931157, Train Accuracy: 86.72%
Validation Loss: 0.891021683823783, Validation Accuracy: 70.25%
Model saved to model.pth
```

1 准确率变化分析

Epoch1+2

最初的训练中我并未添加正则化 Dropout 及 Weight Decay，第一个 Epoch 中 val accuracy 达到了 68%，但在第二个 Epoch 中就出现了过拟合，val accuracy 降至 62%。因此我在全连接层中添加了 Dropout 并在训练过程中设置了 L2 正则化（weight_decay），同时还在图像数据预处理时采用了数据增强以增强模型泛化能力。

```
# 图像预处理，数据增强
transform = transforms.Compose([
    transforms.RandomHorizontalFlip(),  # 随机水平翻转
    transforms.RandomRotation(30),  # 随机旋转
    transforms.ColorJitter(brightness=0.2, contrast=0.2, saturation=0.2, hue=0.2),  # 随机颜色变化
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
])
```

现在的 Epoch1 中 val accuracy 相比之前下降至 63.12%。分析原因：正则化的加入，尤其 Weight Decay（L2 正则化）在损失函数中增加权重惩罚项，限制了参数的增长，对模型的学习进行了约束，且 Dropout 在训练过程中会随机丢弃神经元（0.5），因此虽然减少了模型过拟合的风险，但导致模型在训练初期无法迅速完全拟合训练数据中的特征

可以看到 Epoch2 中的 val accuracy 迅速增长至 72.25%，远远超过了最初的训练准确率，说明：模型已逐渐适应训练数据的结构，同时正则化的作用开始显现，限制了过拟合，帮助模型在验证集上取得更好的泛化能力。

观察输出结果：训练集准确率逐步上升，且验证集准确率也逐步提升。表明模型逐渐找到了更加合理的参数配置，并且在训练数据和验证数据上都取得了较为一致的表现。

Epoch3-5

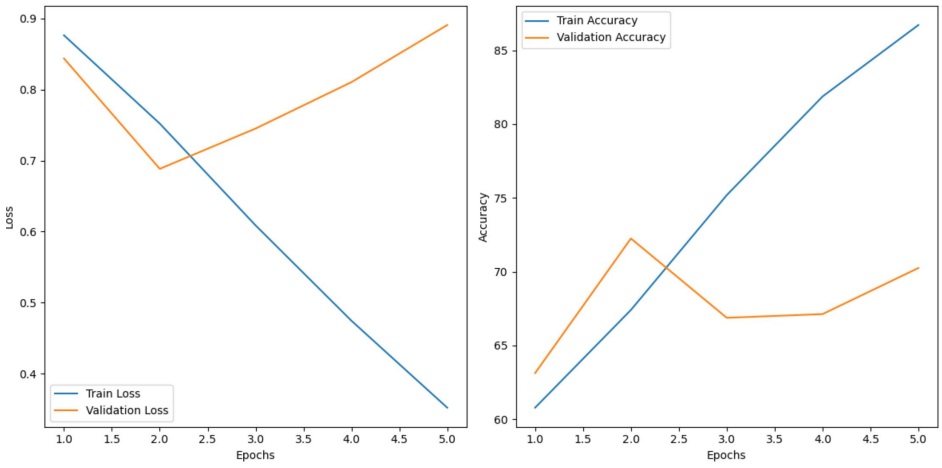
训练中后期，尤其当训练集准确率达到 75.19%（Epoch3）时，验证集准确率开始下降并最后回升至 70.25%。

分析原因：出现过拟合。模型在训练集上的表现越来越好，但在验证集上准确率下降。说明模型对训练集过度学习（噪声或特定模式），导致其在验证集上表现变差。（考虑引入 Batch Normalization 进一步增强泛化能力）

在 Epoch3 准确率波动后，Epoch4-5 再次回升，说明随着训练进行，模型的泛化能力得到改善。

2 训练过程可视化

如图：训练损失逐渐减少并趋于平稳，而验证损失的变化相对波动较大。准确率上文已讨论。



二、消融实验结果对比分析

1 仅文本模型（缺乏图像特征的辅助，整体性能逊色于多模态模型）

```
Training Epoch 1/5: 100%|██████████| 400/400 [12:55<00:00,  1.94s/batch]
Validating Epoch 1/5: 100%|██████████| 100/100 [00:53<00:00,  1.86batch/s]

Epoch 1/5, Train Loss: 0.8186424426361918, Train Accuracy: 64.12%
Validation Loss: 0.7219091865420342, Validation Accuracy: 70.25%
```

2 仅图像模型（性能最差，图像特征对于情感分析任务的作用有限）

```
Training Epoch 1/5: 100%|██████████| 1600/1600 [12:40<00:00,  2.10batch/s]
Validating Epoch 1/5: 100%|██████████| 400/400 [00:44<00:00,  9.03batch/s]

Epoch 1/5, Train Loss: 0.8926253518089652, Train Accuracy: 58.97%
Validation Loss: 0.8353577332943678, Validation Accuracy: 61.25%
```

3 对比

模型类型	优点	缺点	原因分析
多模态模型	- 图像和文本特征互补，提供丰富的上下文信息。 - 验证准确率较高，表现优异。 - 训练准确率较高，减少了过拟合风险。	- 训练与验证集之间仍存在差距，可能存在过拟合问题。 - 计算和内存消耗较大。	多模态融合模型有效地结合了图像和文本信息，在情感分析任务中表现出色，但训练过程中可能会出现过拟合现象，且计算成本较高。
仅文本模型	- 文本特征本身具有强大的情感分类能力。 - 相比图像模型，验证准确率表现较好。	- 训练准确率相对较低，且模型单一，信息量较少。	文本模型能够较好地捕捉句子的情感特征，但缺乏图像的辅助，导致整体表现逊色于多模态模型，特别是在训练集表现上。
仅图像模型	- 相对简单，计算资源消耗较低。 - 可作为图像分类的基本模型进行基线对比。	- 在情感分类任务中，图像信息不如文本信息有效，表现较差。	图像模型无法提供足够的情感信息，因此分类效果不佳，训练准确率和验证准确率远低于多模态和文本模型。

Ⅴ 结论

本次实验我成功实现了一个基于多模态数据的情感分析模型，结合了 RoBERTa 和 ResNet50 模型，分别从文本和图像中提取特征，并通过特征融合进行情感分类，最终达到了较好的性能。在实验过程中，我通过合理的解决方案保证了模型的稳定性和准确性。最终模型在验证集上展现出了较好的分类效果，尤其在多模态数据融合方面，展示了相较于单一模态的优势。

通过本次实验，我深入理解了多模态学习的挑战和优势，特别是在处理多种数据类型时的特征融合和模型优化技术。未来可以考虑引入更多的数据增强技术和更复杂的融合策略，以进一步提升模型性能。