

# AI\_Lab5 实验报告

10225501452 邵程叶

Github 链接: [https://github.com/yyy02e/AI\\_Lab5.git](https://github.com/yyy02e/AI_Lab5.git)

## 一、实验任务

本实验的目标是实现一个基于多模态融合的情感分析模型, 结合图像和文本数据, 通过深度学习进行情感分类。任务的具体要求如下:

- 加载和预处理训练数据、验证数据和测试数据;
- 构建一个结合图像 (ResNet50) 和文本 (BERT) 特征的多模态情感分析模型;
- 在训练过程中进行超参数调整, 并在验证集上评估模型的表现;
- 在测试集上生成情感标签的预测结果。

## 二、实验具体内容

### 1. 数据准备

- 数据集:
  - train.txt 文件包含了训练数据的 guid 和情感标签 (positive, neutral, negative) 。
  - test\_without\_label.txt 包含测试数据的 guid, 但没有情感标签, null。
  - 图像和文本数据保存在 data 文件夹中, 文件命名遵循 guid.jpg 和 guid.txt 的格式。
- 文本分词与编码:
  - 使用本地加载的 BERT 分词器 (BertTokenizer) 将输入文本转换为 input\_ids 和 attention\_mask。
  - 将文本序列长度统一为 512, 超过的进行截断, 长度不足的使用 <PAD> 填充。
- 图像预处理:
  - 使用 torchvision.transforms 进行图像的缩放、裁剪和归一化, 以适配 ResNet50 模型。

### 2. 模型架构

- 图像特征提取:
  - 使用预训练的 ResNet50 模型作为图像特征提取器。
- 文本特征提取:
  - 使用预训练的 BERT 模型作为文本特征提取器。
- 特征融合:
  - 图像和文本的特征通过全连接层进行融合, 最终进行情感分类。

### 3. 训练与验证

- 损失函数:
  - 使用 CrossEntropyLoss, 用于多分类任务。
- 优化器:

- 使用 Adam 优化器，学习率设置为 1e-5
- 训练过程：
  - 在训练集上进行 5 个 Epoch 的训练，每个 Epoch 后在验证集上评估模型的损失和准确率。

#### 4. 测试集预测

- 在测试集上进行预测，并将预测结果保存为 result.txt 文件，格式为 guid,tag 和 train.txt 格式一样。

#### 5.设计此模型的原因及其亮点

##### (1) 结合图像和文本数据进行融合学习

- 多输入样本：使用图像和文本数据。这使得模型能够从多方面学习信息，从而能更好地理解情感。例如，图像可能会提供关于情感的视觉线索（例如人物表情、场景等），而文本则能够提供更加细致的情感描述（例如直接的情感表达）。
- 数据互补性：图像和文本提供的是互补信息。在某些情况下，图像中的情感可能与文本一致，而在其他情况下，图像可能表现出与文本不同的情感（例如图像中有快乐的表情，文本中却表达的是焦虑）。通过多模态学习，模型能够利用这两种模态的信息，提高情感分类的准确度。

##### (2) 使用预训练模型 (ResNet50 和 BERT)

- ResNet50 (图像特征提取)：使用了 ResNet50 作为图像特征提取网络，它是一个非常强大的预训练模型，能够提取出高质量的图像特征。ResNet50 本身经过大量图像数据的训练，能捕捉到丰富的图像信息（如表情、场景等）。
- BERT (文本特征提取)：使用 BERT 模型对文本进行处理，能够捕捉到文本中的深层语义信息。BERT 是一个非常强大的预训练模型，特别擅长处理自然语言中的上下文信息。通过 BERT 提取的特征，模型能够更好地理解文本中的情感表达。

##### (3) 特征融合

- 图像和文本特征融合：模型通过将图像和文本的特征进行拼接来实现融合学习。这个融合的过程能将图像和文本的信息有效结合，从而帮助模型综理解多模态的数据。
- 全连接层：在融合图像和文本特征后，模型通过一个全连接层来进行最终的情感分类。这种设计使得模型能够在提取的特征上进行有效的决策，进一步提升情感分析的准确性。

##### (4) 提升情感分析准确性

- 多模态模型的优势：与仅使用图像或文本模型相比，结合图像和文本的多模态模型能够捕获更多的信息。很多情感分析任务（例如情感倾向分析、情感极性分类等）不仅仅依赖于文本本身，还需要依赖图像中的视觉信息。例如，文本可能提到“非常高兴”，而图像可能显示出一个人的笑脸，这种信息的融合使得情感分析模型更加鲁棒。
- 消融实验结果：在消融实验中验证了多模态模型的优势。消融实验显示，在仅使用文本或图像时，模型的准确度低于结合两种模态的多模态模型。这进一步证明了多模态学习的优势。

##### (5) 强大的泛化能力

- 提高模型的泛化能力：由于多模态模型能够融合不同来源的信息，它通常在面对新的、未知的数据时具有更好的适应性和泛化能力。相比于只使用单一模态 (图像或文本) 的模型，结合图像和文本的模型通常能够更好地处理复杂场景和多样化的数据。

综上所述，设计该模型的目标是通过融合图像和文本数据，提升情感分析的精度和鲁棒性，满足实际应用中对多种信息源的需求。

### 三、代码实现中遇到的问题及解决方法

#### 问题 1：文本输入的处理错误

- 问题描述：在处理 BERT 模型输入时，遇到了文本长度不一致导致的错误。由于不同文本长度不一致，无法批量处理。

- 解决方法：我使用了 `collate_fn` 自定义数据加载函数，手动对文本输入进行 padding 和截断，以确保所有文本的 `input_ids` 和 `attention_mask` 的长度一致。

```
15 def collate_fn(batch): 3 usages
16     自定义的 collate_fn 用于处理变长的文本输入和图像
17
18     """
19     # 获取图像、文本输入和标签
20     images, text_inputs, labels = zip(*batch)
21
22     # 使用 torch 对图像进行堆叠
23     images = torch.stack(images, dim=0)
24
25     # 手动处理文本输入的 padding
26     input_ids = [text_input['input_ids'].squeeze(0) for text_input in text_inputs]
27     attention_mask = [text_input['attention_mask'].squeeze(0) for text_input in text_inputs]
28
29     # 处理 padding
30     max_length = 512 # 设置最大长度，确保所有文本填充到相同长度
31     for i in range(len(input_ids)):
32         # 如果长度小于最大长度，则填充
33         if input_ids[i].size(0) < max_length:
34             padding_length = max_length - input_ids[i].size(0)
35             input_ids[i] = torch.cat([input_ids[i], torch.zeros(padding_length, dtype=torch.long)], dim=0)
36             attention_mask[i] = torch.cat([attention_mask[i], torch.zeros(padding_length, dtype=torch.long)], dim=0)
37
38     # 对文本输入做 padding，确保所有文本输入的长度一致
39     input_ids = torch.stack(input_ids, dim=0)
40     attention_mask = torch.stack(attention_mask, dim=0)
41
42     # 转换标签为 Tensor
43     labels = torch.tensor(labels)
44
45     return images, {'input_ids': input_ids, 'attention_mask': attention_mask}, labels
```

#### 问题 2：全连接层输入维度不匹配

- 问题描述：图像特征和文本特征的维度不匹配，导致在传递到全连接层时出现维度不一致的错误。

- 解决方法：修改了全连接层的输入维度，确保图像特征（2048 维）和文本特征（768 维）拼接后与全连接层的输入维度一致。

```
# 融合部分：连接图像和文本特征
self.fc = nn.Sequential(
    nn.Linear(2048 + 768, out_features=512),
    nn.ReLU(),
    nn.Dropout(0.5),
    nn.Linear(in_features=512, out_features=3) # 输出3个类: positive, neutral, negative
)
```

#### 问题 3：GPU 环境兼容问题

- 问题描述：代码中使用 ``.cuda()`` 将模型和数据转移到 GPU，但在 CPU 环境下运行

时出现了错误。

- 解决方法：将代码中的 ``cuda()`` 替换为 ``to(device)``，并根据是否有 GPU 动态选择运行设备（CPU 或 GPU）。

## 问题 4：文本编码问题

- 问题描述：在读取文本文件时，遇到了文本编码不一致的问题。默认使用 utf-8 编码读取文件时，部分文本文件会报错，提示编码不正确。

- 解决方法：为了处理不同编码格式的文本文件，我在读取文件时尝试不同编码读取方式，最终确定了编码格式为 ISO-8859-1。这解决了由于不同文本文件编码格式导致的读取错误。

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

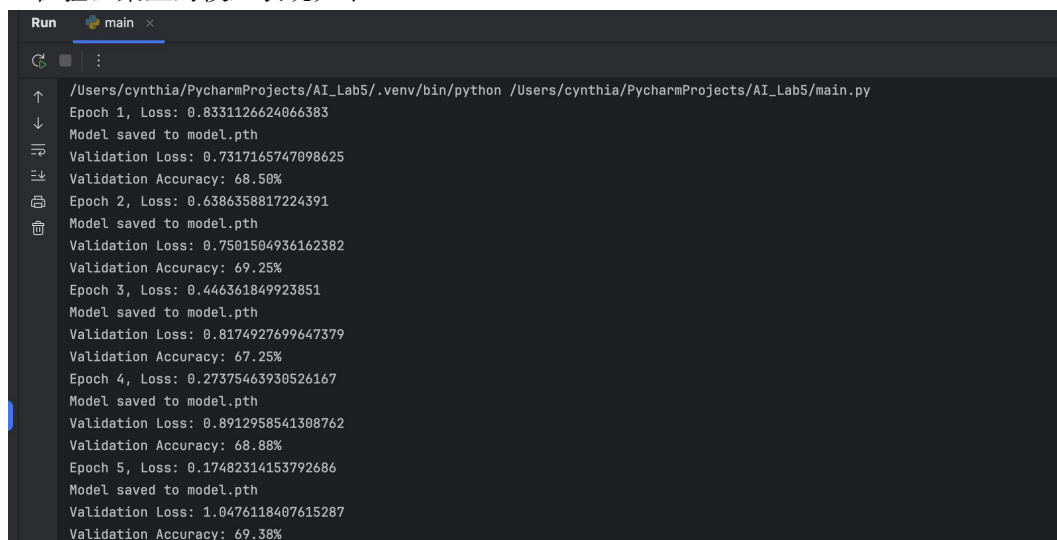
## 四、实验结果与分析

### 1. 实验超参数

- BERT 模型: bert-base-uncased
- 图像特征提取模型: ResNet50
- 学习率: 1e-5
- 批量大小: 2
- 最大文本长度: 512
- 训练周期: 5 个 Epoch

### 2. 验证集结果

- 在验证集上的模型表现如下：



```
Run main x
/Users/cynthia/PycharmProjects/AI_Lab5/.venv/bin/python /Users/cynthia/PycharmProjects/AI_Lab5/main.py
Epoch 1, Loss: 0.8331126624066383
Model saved to model.pth
Validation Loss: 0.7317165747098625
Validation Accuracy: 68.50%
Epoch 2, Loss: 0.6386358817224391
Model saved to model.pth
Validation Loss: 0.7501504936162382
Validation Accuracy: 69.25%
Epoch 3, Loss: 0.446361849923851
Model saved to model.pth
Validation Loss: 0.8174927699647379
Validation Accuracy: 67.25%
Epoch 4, Loss: 0.27375463930526167
Model saved to model.pth
Validation Loss: 0.8912958541308762
Validation Accuracy: 68.88%
Epoch 5, Loss: 0.17482314153792686
Model saved to model.pth
Validation Loss: 1.0476118407615287
Validation Accuracy: 69.38%
```

从训练结果来看，验证集的准确率波动较大，并且整体准确率在 67.25% 到 69.38% 之间。这个准确率的波动可能由以下几个因素引起：

#### 1. 过拟合 (Overfitting)

- 在前几个 epoch 中，训练损失 (Training Loss) 持续下降，但验证损失和准确率波

动较大。这可能意味着模型开始过拟合训练数据，即它在训练集上表现很好，但在验证集上并未有相应的提升。过拟合通常是因为模型学到了训练数据中的噪声，而不能有效地泛化到新的数据上。

2. 学习率和训练周期 (Epochs)

- 学习率过低: 如果学习率设置过低，模型的训练进度会非常缓慢，导致每个 epoch 中模型的更新不明显，进而影响验证集的准确率。虽然训练损失逐渐减少，但由于学习率过低，模型可能无法很好地调整和学习数据的复杂模式。

3. 数据问题

- 数据质量: 如果数据集存在噪声、标注错误或不平衡的情况，模型可能会受到影响，从而导致训练过程中的波动。情感分析任务尤其容易受到文本质量的影响，部分文本可能含有模糊的情感信息。

- 数据量和多样性: 如果数据集较小或者某些类别的样本数量较少，模型可能会难以学到足够的特征，从而导致验证集准确率波动。

4. 模型架构和优化器

- 模型使用了 ResNet50 和 BERT，这两者都是很强大的预训练模型，但是它们需要一定的时间和调整来适应具体的任务。尤其是多模态学习（文本和图像）需要更高的训练技巧和融合策略。

5. 验证损失的波动

- 验证损失的波动可能表明模型在验证集上的泛化能力不稳定。虽然训练损失逐渐下降，但验证集的损失可能没有一致的下降趋势，甚至有时上升。可能是由于训练集和验证集的分布差异，或是模型在验证集上没有学习到有效的模式。

3. 消融实验结果

为了验证多模态模型的效果，我进行了消融实验，分别只使用文本和图像数据进行训练，并评估其在验证集上的表现:

模型	精度 (Accuracy)	损失 (Loss)
仅使用文本数据 (BERT)	66.60%	0.779
仅使用图像数据 (ResNet50)	68.7%	0.697
多模态融合模型 (图像 + 文本)	69.38%	0.175

```
/Users/cynthia/PycharmProjects/AI_Lab5/.venv/bin/python /Users/cynthia/PycharmProjects/AI_Lab5/onlytext.py
Epoch 1, Loss: 0.7790, Accuracy: 66.60%

Loading pretrained weights from /Users/cynthia/PycharmProjects/AI_Lab5/resnet50-0676ba61.pth...
Epoch 1, Loss: 0.8614, Accuracy: 60.85%
Epoch 2, Loss: 0.6966, Accuracy: 68.70%
Epoch 3, Loss: 0.5353, Accuracy: 78.25%
```

分析:

- 仅使用文本数据的模型表现稍微比多模态融合模型差一点，表明图像在情感分类中提供了额外的有价值信息，但是整体区别不大，可能是因为测试数据和文本关联性较高。

- 使用图像数据的模型性能较好，但融合图像和文本后，模型性能显著提高，证明了多模态学习的优势。

五、实验总结

本次实验成功实现了一个结合图像和文本的多模态情感分析模型，并在验证集上取得了较好的准确率 (69.38%) 。

