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**：大三　　**上机实践成绩**：
指导教师：李翔　　**姓名**：刘蔚璁　　**学号**：10225501443

实验五：多模态情感分析

[摘要]：本实验旨在通过多模态融合模型结合文本与图像数据，完成情感分析三分类任务。我们基于匿名数据集，设计并实现了一种融合双向注意力机制的多模态深度学习模型，对训练数据进行预处理、特征提取与融合，并在验证集上进行了性能评估和消融实验。结果表明，多模态融合模型相较于单模态方法在准确率和 F1 分数上均有显著提升，验证了模型在情感分析任务中的有效性。

[GitHub]：https://github.com/Gnociew/Multimodal_sentiment_analysis

一、目标与要求

- 设计一个多模态融合模型，在给定的配对文本和图像上预测对应的情感标签
- 自行从训练集中划分验证集，调整超参数
- 在验证集上进行消融实验

二、实验设计与模型构建

- **数据集描述**

  本次使用的是匿名数据集，包含 4000 条训练数据和 511 条测试数据。

  - 数据集组织如下：
    - data 文件夹：包括所有的训练文本和图片，每个文件按照唯一的 guid 命名。
    - train.txt：训练数据的 guid 和对应的情感标签。
    - test_without_label.txt：测试数据的 guid 和空的情感标签。
  - 经统计发现训练集存在标签分布不均的情况：

positive	negative	neutral
2388	1193	419

表 1 训练集标签分布情况

  - 其中文本数据多为短文本，长度不一，可能包含社交网络标签、URL 等信息。

- **数据集划分**

  由于数据集中仅提供了训练数据和测试数据，因此需将训练数据进一步划分为训练集和验证集。因为数据集较小，为了能充分地利用数据，本实验采用五折交叉验证，最终选用验证效果最优的模型用于测试。注意要设置随机种子保证结果可复现。

  - 使用 Scikit-learn 提供的用于实现交叉验证的函数 KFold 进行划分

```
kfold = KFold(n_splits=k_folds, shuffle=True, random_state=seed)
kfold.split(train_dataset)
```

- **数据预处理**

  为了构建适用于多模态任务的数据集，实验中设计了一个自定义数据集类，实现对输入数据的预处理和格式化。具体操作如下：

  - **图片处理**

    将输入图片统一调整为 224x224 像素大小以适配模型的输入要求，并将图片从 PIL 格

式转换为 PyTorch 的张量格式，最后对其进行标准化，将像素值分布归一化到标准正态分布，有助于加速训练和提高模型性能。

- **文本处理**

  对原始文本进行清洗，去除噪声和多余信息，如 URL、@用户标签、转发标记等，接着将文本转为小写，并去除常见的停用词，以此减少噪声，保留重要的语义信息。最后使用 BertTokenizer 将文本编码为模型可接受的格式。

模型构建

在本次任务中，我构建了两个多模态融合模型。第一个模型基于 PyTorch 的基础层进行搭建，其中图像特征提取采用 ResNet 架构，文本特征提取则使用 Transformer 的 Encoder 模块。第二个模型使用 Huggingface 提供的预训练模型，分别采用 Vision Transformer (ViT) 提取图像特征和 BERT 提取文本特征。此外，这两种模型均实现了三种特征融合方法，包括直接拼接、交叉注意力以及双向交叉注意力。

接下来我将介绍自己搭建的模型的架构。

- **整体架构**

  该模型使用 `torch.nn` 模块中提供的神经网络模块和工具进行搭建，自定义的神经网络模型需要继承 `nn.Module`，并提供 `forward` 方法来定义数据如何在模型中前向传播。因为模型有明显的模块结构，所以先分别搭建各个模块，最后进行整合。

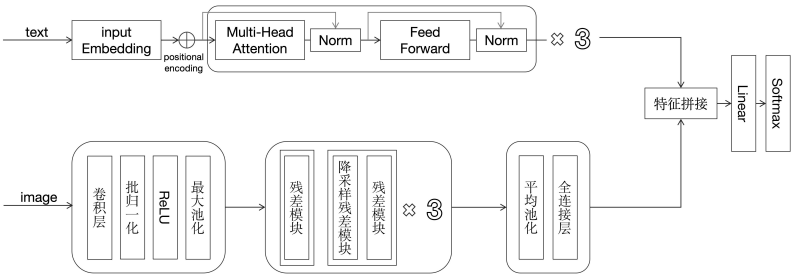

图 1 多模态融合模型搭建示意图

- **图片特征提取模块**

  该模块采用 ResNet 进行图片特征提取，可以分成残差块和整体网络的搭建，这个网络的详细解释和构建在 Project3 的实验报告中已体现，区别在于最后的全连接层不映射到具体类别，而是映射到特征维度便于拼接，以下是残差块的搭建示意图：

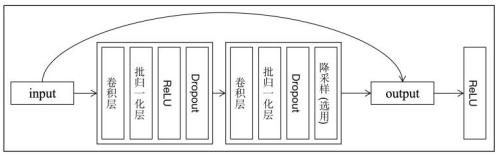

图 2 残差块搭建示意图

- **文本特征提取模块**

  该模块使用 Transformer 的 Encoder 模块进行特征提取，同样输出给定维度的特征，Transformer 的详细搭建过程同样在 Project4 的实验报告中体现了，故此处只给出 Encoder 部分的搭建示意图：

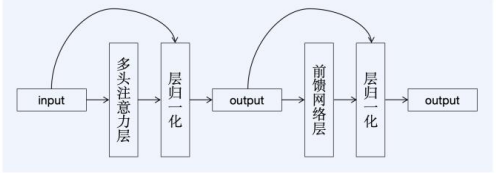

图 3 Encoder 层搭建示意图

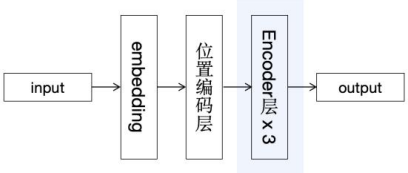

图 4 Encoder 搭建示意图

- **特征融合模块**

  上述操作在下游任务中提取出了不同模态数据的特征，为了充分利用不同模态间的信息互补性，需要在模型中引入了特征融合模块，用以将图像特征与文本特征有效地融合，输出富含多模态交互信息的表示向量，以提升分类性能。我实现了三种特征融合方法，具体如下：

  - **直接拼接**

    先将图像特征与文本特征在特征维度上使用 `torch.cat` 进行拼接，后接一层全连接层进行分类。该方法实现简单、计算开销较低，但无法显式建模图文间的细粒度交互，可能丢失部分跨模态信息。

  - **交叉注意力**

    因为相关有很多种不同的形式、不同的定义，所以考虑通过并行计算多组注意力，来捕获不同方面的相关性。使用图片数据作为 Query，文本数据作为 Key 和 Value，对每个头独立计算点积注意力，最后将多头输出通过拼接和线性投影恢复为原始维度。点积注意力的计算公式如下：

    $$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{Q \cdot K^T}{\sqrt{d_k}}\right) \cdot V$$

  - **双向交叉注意力**

    类似 ViLBERT、UNITER 等多模态预训练中的做法，先让图像 Query 文本，再让文本 Query 图像，多层堆叠之后得到深度融合的表示，这样可以对图像和文本细节有更深层理解，但计算成本更高，训练更耗时。

☺ **注意!** 图片特征提取中使用的是批归一化，而文本特征提取使用的是层归一化。

- **训练过程**

本实验旨在通过优化模型参数，提高分类任务的准确率。在训练过程中，采用监督学习的方式，使用**焦点损失函数 Sigmoid Focal Loss** 作为目标函数，最小化模型在训练集上的预测误差并利用 **Adam 优化器**进行参数更新。

整个训练分为多个迭代周期（epoch），每个 epoch 中，使用 DataLoader 分批次加载数据，模型依次处理训练数据集的所有批次，计算损失值并进行参数更新。同时，在每个 epoch 的末尾，模型会在验证集上评估性能，并通过早停机制监测模型的训练进展以防止过拟合。

- **训练细节**

  - **Sigmoid Focal Loss**

    在最开始我们提到，训练集的标签分布是**非常不平衡**的。在分类任务中。当某些类别的样本数量显著多于其他类别时，模型可能会偏向于预测出现频率较高的类别，而 Sigmoid Focal Loss 是基于二分类交叉熵损失的改进版本，用于解决目标检测中类别不平衡问题。当样本易于分类时（预测概率接近 1），损失会自动减小，避免易分样本对梯度的过多贡献，对难分样本的损失加权，促使模型重点学习这些样本，从而提升性能。

  - **早停机制**

    在训练初期，模型的性能通常随着迭代次数的增加而提高，然而，随着训练的继续，模型在训练集上的表现可能继续改善，但在验证集上可能出现性能下降的情况，即过拟合。所以我们应该对训练过程进行监控，因此引入早停机制。

    - ✓ **如何选择最佳停止点?**

      由于训练过程存在一定的随机性，因此不宜仅依据某一个 epoch 的验证集精度作

为判断标准。我们通常会参考多个 epoch 的平均验证性能，或者当验证集性能在连续若干个 epoch 内未得到改善时，提前终止训练。在本实验中，**我们设置当验证集准确率在 3 个 epoch 内未改善时提前终止训练。**

✧ **训练环境**

硬件环境：RTX 4090(24GB)　　软件环境：Python - 3.12.3

✧ **tqdm**

为了更直观地监控训练过程和模型收敛情况，本实验在训练和验证过程中引入了 tqdm 进度条库，提高了实验过程的可读性。

➢ **伪代码**

```
输入:数据集，模型名称，超参数
输出:最佳训练模型，测试集准确率

for fold in range(k folds):
    使用 Dataloader 将数据分批加载到模型中
    根据模型名称初始化对应模型
    定义损失函数和优化器

    for epoch in range(最大送代轮数):
        将模型设置为训练模式
        for image, text, labels in train loader:
            清空梯度
            前向传播，计算损失
            反向传播，更新模型参数
            累加训练损失
        计算并记录该轮平均训练损失

        将模型设置为验证模式
        for image, text, labels in val loader:
            前向传播，计算损失
            累加验证损失
            计算预测类别，统计准确率
        计算并记录平均验证损失和准确率

        if 当前验证准确率 > 最佳验证准确率:
            将未改善计数重置为零
            保存当前模型
        else:
            未改善计数 ++
            if 未改善计数>设定的阈值:
                结束训练
```

● **参数调整与结果评估**

为了优化模型性能，我通过设置对比实验对超参数进行了调整，通过对比**最佳验证准确率**寻找最佳参数组合，并进行消融实验以验证每个模态在整体模型中的贡献。

参数调整

➢ **注意力头数和前馈网络层数**

为了确定这三个参数的合适大小，我在固定学习率 0.0001、批次大小 32、编码器层数都为 3、丢弃率为 0.3、嵌入向量维度 512 的情况下进行测试：

注意力头数	前馈网络层数			
	256	512	1024	2048
4	57.00%	56.25%	60.12%	58.50%
8	50.88%	56.38%	60.50%	54.50%

表 2 不同参数组合下的验证准确率

✓ 增加头数可以让模型在更多的子空间中关注不同的语义关系，但头数过多可能导致模型过拟合或计算资源浪费。增加前馈网络层数能提升模型拟合更复杂特征的能力，但层数增加会显著提高计算成本，且梯度可能会变得不稳定。

➢ **学习率和特征维度**

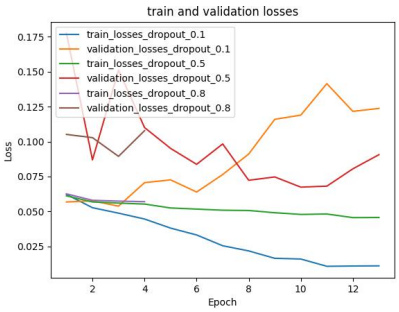

图 5 不同学习率下的损失折线图

图 6 不同特征维度下的损失折线图

❖ **模型对比**（注：self 代表自己搭建的模型，pretrain 代表预训练的模型）

模型	特征融合方法		
	直接拼接	交叉注意力	双向交叉注意力
self	58.50%	62.00%	63.88%
pretrain	67.75%	70.25%	73.03%

表 3 不同特征融合方法下的验证准确率

❖ **消融实验**

模型	指标	模态		
		all	image	text
self	acc	64.12%	58.50%	63.62%
	加权 f1	0.5782	0.4318	0.5812
pretrain	acc	68.75%	65.00%	67.62%
	加权 f1	0.6416	0.5943	0.6277

表 4 消融实验下的验证准确率和加权 F1 值

- **argparse 模块**

  为了提高实验的灵活性，本实验使用了 argparse 模块对模型和超参数进行管理。通过命令行参数传递实验配置，能够方便地选择模型、拼接方式、随机种子等。

三、讨论与总结

- **遇到的问题**

  ✧ 在最后调整训练过程的代码结构时出现了训练损失不下降的情况，重复实验后发现每次都与第一轮相同，这说明不是梯度爆炸或梯度消失，而是没有进行更新，后认真排查代码发现移动模型初始化位置的时候忘记移动优化器定义的位置，导致优化器定义在模型初始化前，无法绑定模型进行优化。

  ✧ 在使用自己搭建的两个模型进行特征拼接的时候遇到了非常多次维度不匹配的问题，解决方法就是在代码各处打印出向量维度，一点点进行调整。

- **总结**

  本次实验旨在设计和实现多模态融合模型，通过文本与图像特征的联合建模完成情感分析任务。实验过程中，我分别构建了基于自定义搭建与预训练模型的两种多模态融合框架，并结合三种特征融合方法进行对比分析。实验结果表明，双向交叉注意力效果最佳，且基于预训练模型的框架在所有特征融合方法下均优于自定义搭建的模型。此外，通过消融实验发现多模态信息融合显著提升了模型的分类能力。在实验过程中，针对数据不平衡问题引入 Sigmoid Focal Loss，有效缓解了类别偏差对模型性能的影响。本次实验不仅提升了我对多模态情感分析的理解，对各种问题的处理也让我对实际模型训练中的关键问题有了更系统的认识。