https://github.com/zty08111220/Multimodal-Fusion-Model

实验五：多模态情感分析

模型设计

初始设计

文本处理

- 文本特征提取：
  - 使用 `BertModel` 从 `bert-base-uncased` 文件夹中加载预训练的 BERT 模型（`bert-base-uncased`）。
  - BERT 模型的输出是一个隐藏状态张量，形状为 `(batch_size, sequence_length, hidden_size)`，其中 `hidden_size` 为 768。
  - 通过取 `[CLS]` 标记对应的隐藏状态（即 `last_hidden_state[:, 0, :]`）作为整个文本的表示。
  - 将 `[CLS]` 标记的 768 维向量通过一个全连接层（`nn.Linear(768, 256)`）降维到 256 维。
- 文本编码：
  - 使用 `BertTokenizer` 对输入文本进行编码，生成输入张量（`input_ids`, `attention_mask` 等）。
  - 文本编码的最大长度设置为 128，超出部分会被截断，不足部分会用填充标记（`[PAD]`）补齐。

图像处理

- 图像特征提取：
  - 使用 `resnet50` 从 `torchvision` 中加载预训练的 ResNet-50 模型。
  - ResNet-50 的最后一层全连接层被替换为一个新的全连接层（`nn.Linear(self.img_model.fc.in_features, 256)`），将输出维度从 2048 维降维到 256 维。
  - 图像经过 ResNet-50 后，输出一个 256 维的特征向量。
- 图像预处理：
  - 调整大小为 224x224 像素。
  - 转换为张量（`transforms.ToTensor()`）。
  - 归一化（`transforms.Normalize`）。

多模态融合

- 特征拼接：
  - 将文本特征（256 维）和图像特征（256 维）在特征维度上进行拼接（`torch.cat`），得到一个 512 维的融合特征。
  - 将拼接后的特征通过一个全连接层（`nn.Linear(512, 128)`）进一步降维到 128 维。

分类器

- 分类：
  - 将融合后的 128 维特征通过一个全连接层（`nn.Linear(128, num_classes)`）映射到类别数（`num_classes=3`）的维度。
  - 输出为每个类别的 logits（未归一化的分数），可以用于计算交叉熵损失。

模型修改

一开始学习率设置为1e-4，batch_size设置为32，验证损失随着epoch上升。

为了解决验证损失随着epoch上升这个问题。增加dropout层和优化器的权重减益。减小了学习率。对图像数据进行增强（随机水平翻转、旋转）。使用学习率调度器`ReduceLROnPlateau`，在验证损失不再下降时降低学习率。然后我想会不会是模型复杂度太高了，就把文本特征提取部分的维度，图像特征提取部分的维度，多模态融合部分的维度都减半。

复杂度高的模型，会在epoch=3时验证损失和验证准确率同时上升。这是因为在过拟合的早期阶段，模型可能开始记住训练数据的噪声，导致验证损失上升。然而，如果验证集中的样本与训练集相似，模型可能仍然能够正确分类这些样本，导致验证准确率上升。

超参数选择

尝试batch_size(16,32,64,128)模型复杂度(高,低)。一共8种情况。数据记录在`batch_size和模型复杂程度.txt`中。

一开始我想选择batch_size=128，复杂度高的模型。因为这种情况的过拟合程度最轻，直到epoch=6验证损失才上升，验证损失较低，验证准确率较高。数据集较大且计算资源充足。后面我测学习率的时候第一遍用的是简单模型，忘记换回复杂模型了。后面我又对比了一下相同学习率下两种模型的表现，发现简单模型验证损失和验证准确率表现较好。最后选择batch_size=128，复杂度低的模型。这种情况的过拟合程度是第二轻的。

尝试learning_rate(1e-4,1e-5,1e-6)。1e-4验证损失很高。1e-6验证损失下降太慢了，10个epoch还没过拟合。选择learning_rate=1e-5。再在1e-5周围选择学习率。数据记录在`学习率.txt`中。

尝试learning_rate(8e-6,9e-6,1e-5,1.1e-5,1.2e-5)。选择learning_rate=8e-6。原因是训练损失和验证损失下降较为平稳，验证准确率逐步提升，且没有明显的过拟合现象。

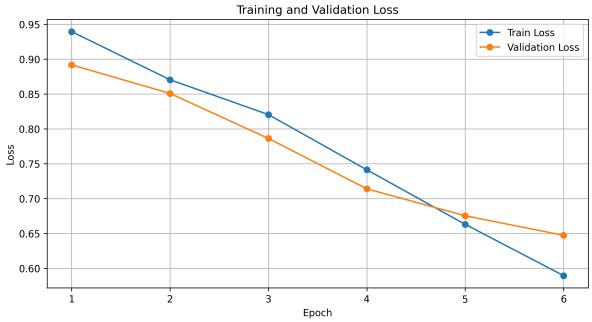
epoch=7或者epoch=6时过拟合，选择epoch=6。后面每次尝试的时候就看运气，只记录epoch=6时不过拟合的模型。这可能是因为我对图像数据进行增强。

最终选择：batch_size=128，lr=8e-6，模型复杂度低，epoch=6。

结果分析

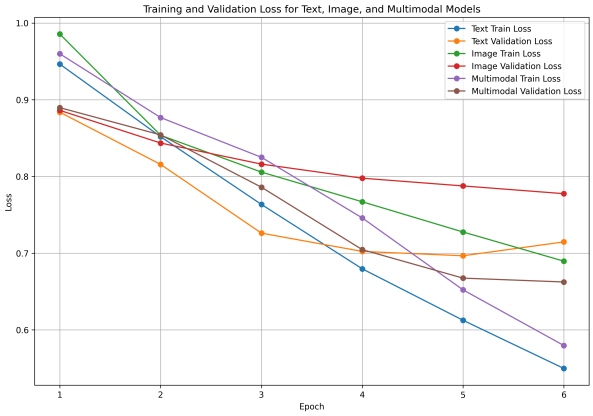
多模态融合模型

```
Validation Loss: 0.6474, Validation Accuracy: 0.7375
```



随着epoch的增加，模型对于训练集和验证集的拟合都在改善，且没有过拟合，验证损失与训练损失之间的差距并不是很大，这表明模型具有合理的泛化能力。

文本模型，图像模型和多模态融合模型



训练损失：文本<多模态<图像

验证损失：多模态<文本<图像

从图中可以看出，无论是仅输入图像数据还是仅输入文本数据，模型都能够从中学习到一定的特征。

文本模型过拟合了。原因可能是正则化不足，我没有加dropout或者权重衰减，只用了全连接。

图像模型表现较差可能是因为图像特征可能无法充分捕捉数据中的有用信息。

对比单一模态模型（如仅使用文本或图像数据的模型），多模态融合模型验证损失更低。这意味着多模态模型能够有效地结合来自不同类型的输入数据（如文本和图像）的特征，形成更加全面且准确的特征表示，这间接增加了正则化效果。

bug

1编码格式使用utf-8报错。用`check.py`打印出所有不重复的编码方式IBM866TIS-620,Big5,Windows-1254,Windows-1253,EUC-JP,ascii,Windows-1252,MacRoman,IBM866,KOI8-R,ISO-8859-1,ISO-8859-9,Windows-1251,GB2312。定义需要尝试的编码方式列表，并遍历尝试。

2网上租gpu跑，得上传bert模型，云上下载不了报错了。本地下载好装上去，读取就行了。

3读取测试数据中guid列以浮点数形式读了，导致读取了1.0.txt文件。读取训练数据的时候还是好好的，代码都一样的。读取的时候强制把读到的数改成整数就行了。

4尝试batch_size=256发现租的gpu内存不够了，跑不了。这是是因为进程已经退出，但一部分显存未释放。 终止Jupyter Notebook主进程重启内核，释放了被占用的显存。再次尝试发现还是不够，这就是因为租的不太行。`ps -ef`找出主进程进程id。`kill -9 id`杀死主进程。后面确定了batch_size=128。我每次跑都得重启内核，不然显存不够。

5文本模型，图像模型和多融合模型放在一起训练显存不够。分三次训练，把loss存在txt文件里，最后读取三个txt文件绘图。