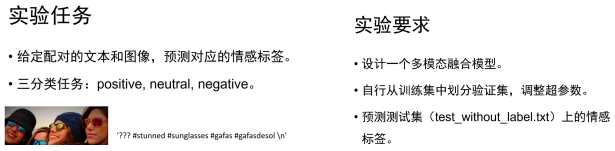
华东师范大学数据科学与工程学院上机实践报告

课程名称：当代人工智能	**年级：**2022 级	**上机实践成绩：**
指导教师：李翔	**姓名：**李芳	
上机实践名称：多模态情感分析	**学号：**10214602404	**上机实践日期：**
上机实践编号：05	**组号：**	2025.01.17

一、目的



二、内容与设计思想

1．Github 地址：user110laolemon/AIhw5 （我的仓库会在 1.22 日作业截止之后从私有仓库更改为共有仓库，由于上传总是超时，所以训练模型仅保留了表现较好的两个，删除了 python 缓存）

2．构造一个多模态融合模型：多模态输入要处理文本和图像，我需要先找到这两方面表现比较好的模型（Roberta 是改进的 bert 模型，更有利于情感分析等 NLP 任务、ConvNeXt 是之前做 CNN 时表现比较好且能够处理复杂视觉特征的模型）分别写出处理单种数据的代码，再结合在一起加上多头注意力机制等构建多模态融合模型，这样能够捕捉到更多特征、模态关系，还可以适应我给定的输入场景，灵活性和适应性更高。

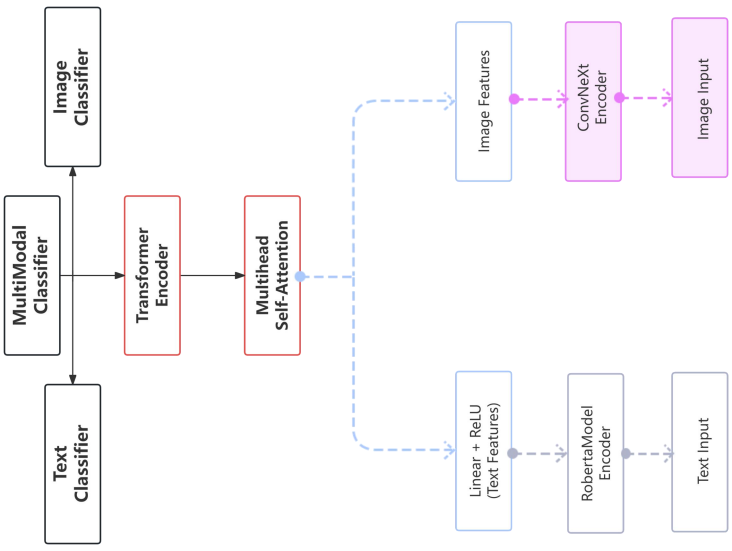
三、使用环境

Vscode + python

四、实验过程

1．文件代码解析

（1）Model.py：（为了节省空间，所以需要旋转一下图像）



①TextOnly 类，继承 nn.Module:

1）初始化：

调用父类的初始化方法、加载预训练的 RobertaModel、设置 RobertaModel 的参数为可训练、定义一个顺序容器，包含线性层和 ReLU 激活函数

2）前向传播：

使用 RobertaModel 对输入进行编码、获取编码器的最后一层隐藏状态、获取池化后的输出、将池化后的输出通过线性层和激活函数进行变换、返回隐藏状态和变换后的输出

②ImgOnly 类，继承 nn.Module：

1）初始化：

调用父类的初始化方法、加载预训练的 ConvNeXt 模型、设置 ConvNeXt 模型的参数为可训练

2）前向传播：

使用 ConvNeXt 模型对输入进行编码、返回编码后的输出

③MultiModalModel 类，继承 nn.Module：

1）初始化：

调用父类的初始化方法、初始化文本模型模块、初始化图像模型模块、定义多头自注意力机制、定义线性层用于生成键和值、定义 Transformer 编码器层、定义图像分类器、定义文本分类器、定义多模态分类器

2）前向传播：

如果只有文本输入：使用文本模型进行编码、使用文本分类器进行分类、返回文本分类结果

如果只有图像输入：使用图像模型进行编码、使用图像分类器进行分类、返回图像分类结果

如果同时有文本和图像输入：使用文本+图像模型进行编码、将文本和图像编码结果拼接、使用文本+图像+多模态分类器进行分类、返回多模态分类结果

3）多头自注意力机制：

使用线性层生成文本和图像的键和值、将文本和图像的键+值+查询堆叠、使用多头自注意力机制进行计算、返回注意力输出

4）Transformer 编码器：

将文本和图像的编码结果堆叠、使用编码器进行编码、返回编码结果

（2）Dataprocess.py：

①MyDataset 类创建自定义数据集，包含文本和图像的编码；load_json（）加载 JSON 文件转换为数据列表；load_data（）加载数据转换为 MyDataset；save_data（）保存预测结果文件；read_text_file（）读取文本文件内容；transform_data（）把数据转换为包含 guid、文本、标签和图像路径的字典。

②主函数读取数据，进行数据文件转换，最后保存为 JSON 文件

（3）Train_evaluate.py：

①train_and_test（）训练模型并在验证集上评估性能，保存最佳模型，并绘制损失曲线图；evaluate（）评估模型在验证集上的性能，计算并返回验证损失和准确率；get_test（）测试模型在不同场景下的准确率，并保存预测结果；calculate_accuracy（）计算预测结果的准确率；plot_loss_curve（）绘制训练和评估损失曲线图

②主函数：打印设备信息、加载训练集和验证集、创建数据加载器。根据 config 配置选择模式：do_train，则训练模型并评估性能；do_test，则加载测试集，测试模型在不同场景下的性能，并打印测试结果

（4）Config.py：超参设置、文件路径等信息。

使用流程：首先需要在代码文件中添加训练模型、训练曲线、预测结果三个文件夹；

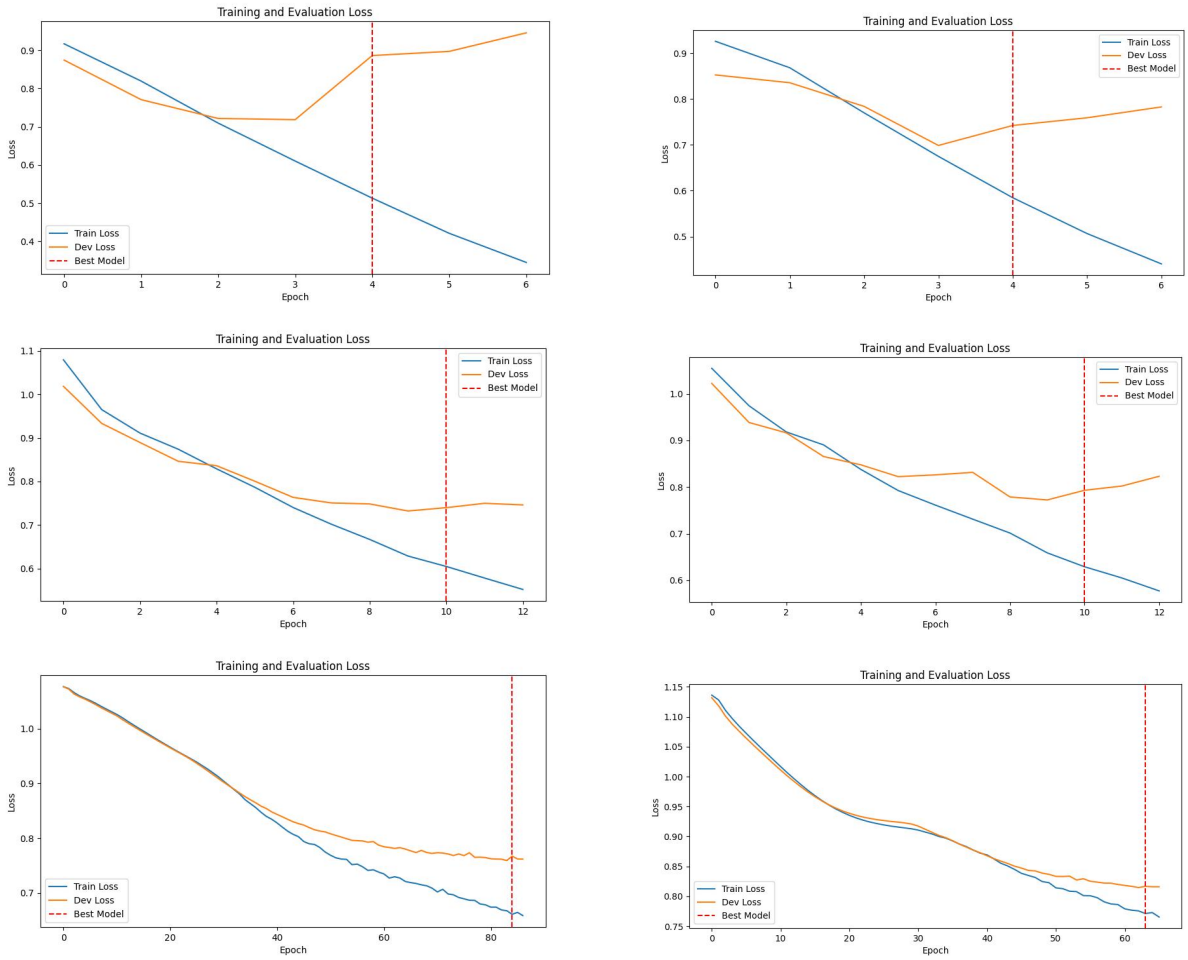
然后更改 config.py 文件中的 lr、dropout 参数设置大小，可以改变训练步长和正则化程度；

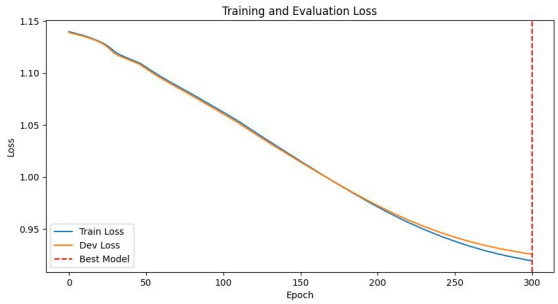
接着 do_train、do_test 是否为真代表运行 train_evaluate.py 文件主函数中的训练模式还是预测模式；

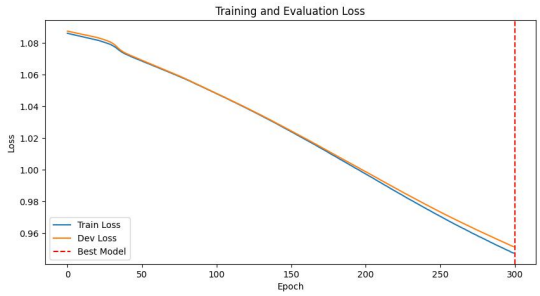
最后 mode 参数设置为 train 或者 test 表示加载的数据是训练集还是测试集数据集，当训练模式时，使用 train mode，加载转化训练集划分为训练和验证数据。

2．结果分析

（1）训练验证损失（dropout 从左到右 0.1、0.3；lr 从上到下 1e-5、1e-6、1e-7、1e-8）







lr 从大到小，步长从大到小，训练速度从快到慢，但是，超过一定区间，过小无法收敛，学习无效，过大速度太慢，耗时太长（最左一列，学习率过大，根本未收敛；最右一列，学习率太小，epoch 设置 300 的情况下都没触发早停，效率过低）。虚线 plt 绘制不精确，仅供参考

dropout 从小到大，训练速度从慢到快，同时能防止过拟合现象，但是也是有一定限度，过小设置无意义，过大无法正常学习有效特征，效果反而还会下降。

（2）多模态融合模型在训练验证集上的结果以及消融实验结果（只输入文本、只输入图像数据）（从左到右 lr=0.6、0.7、0.8，从上到下 dropout=0.1、0.3）

```
  Text_only Test Accuracy: 0.2925

Scenario: Image Input Only
  Image_only Test Accuracy: 0.1734

Scenario: Text and Image Input
  Text_and_image Test Accuracy: 0.2925
预测完成.
```

```
  Text_only Test Accuracy: 0.3603

Scenario: Image Input Only
  Image_only Test Accuracy: 0.3334

Scenario: Text and Image Input
  Text_and_image Test Accuracy: 0.3603
预测完成.
```

```
  Text_only Test Accuracy: 0.3603

Scenario: Image Input Only
  Image_only Test Accuracy: 0.1122

Scenario: Text and Image Input
  Text_and_image Test Accuracy: 0.3603
预测完成.
```

```
  Text_only Test Accuracy: 0.3059

Scenario: Image Input Only
  Image_only Test Accuracy: 0.0666

Scenario: Text and Image Input
  Text_and_image Test Accuracy: 0.3059
预测完成.
```

```
  Text_only Test Accuracy: 0.3534

Scenario: Image Input Only
  Image_only Test Accuracy: 0.1778

Scenario: Text and Image Input
  Text_and_image Test Accuracy: 0.3534
预测完成.
```

```
  Text_only Test Accuracy: 0.3603

Scenario: Image Input Only
  Image_only Test Accuracy: 0.0663

Scenario: Text and Image Input
  Text_and_image Test Accuracy: 0.3603
预测完成.
```

从不同的数据类型来看，恰当的学习率可以提高预测准确性，从输出可以看出，模型在文字+图像和仅文字情况下表现差别不大，至少在保留四位有效数字后精确度一致，但是在仅图像数据时表现的较差，即使在表现优秀的超参下还是不如另两种数据类型。

我猜测一部分原因可能是由于数据集中可能存在类别不平衡，导致模型在某些类别上的表现较差，也可能因为我的多模态模型的架构比较简单，或者说没有充分利用图像信息，让 convnext 模型发挥出来。这是可以继续研究的小问题。

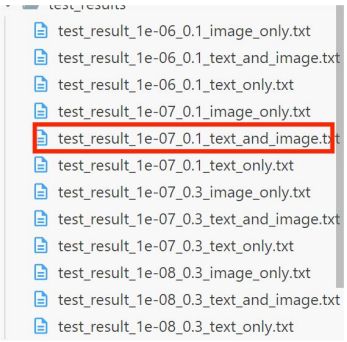
（3）测试文件结果预测（选训练时表现最好且效率高的超参配置 1e-7&0.1）改名为 test_result.txt。

```
  Train Loss: 0.7793, Train Accuracy: 0.6009
  Dev Loss: 0.8184, Dev Accuracy: 0.5863 (Epoch 61)
Epoch 62/300:
  Train Loss: 0.7770, Train Accuracy: 0.6012
  Dev Loss: 0.8168, Dev Accuracy: 0.5825 (Epoch 62)
Epoch 63/300:
  Train Loss: 0.7750, Train Accuracy: 0.6012
  Dev Loss: 0.8149, Dev Accuracy: 0.5825 (Epoch 63)
Epoch 64/300:
  Dev Loss: 0.8168, Dev Accuracy: 0.5850 (Epoch 64)
Epoch 65/300:
  Train Loss: 0.7731, Train Accuracy: 0.6088
  Dev Loss: 0.8161, Dev Accuracy: 0.5837 (Epoch 65)
Epoch 66/300:
  Train Loss: 0.7656, Train Accuracy: 0.6131
  Dev Loss: 0.8161, Dev Accuracy: 0.5863 (Epoch 66)
Early stopping triggered after 66 epochs.
Best model saved with dev loss: 0.8149
```

1e-7,0.3

```
  Train Loss: 0.7260, Train Accuracy: 0.6006
  Dev Loss: 0.7954, Dev Accuracy: 0.5850 (Epoch 75)
Epoch 76/300:
  Train Loss: 0.7244, Train Accuracy: 0.6019
  Dev Loss: 0.7939, Dev Accuracy: 0.5850 (Epoch 76)
Epoch 77/300:
  Train Loss: 0.7203, Train Accuracy: 0.6059
  Dev Loss: 0.7960, Dev Accuracy: 0.5850 (Epoch 77)
Epoch 78/300:
  Train Loss: 0.7148, Train Accuracy: 0.6078
  Dev Loss: 0.7966, Dev Accuracy: 0.5850 (Epoch 78)
Epoch 79/300:
  Train Loss: 0.7164, Train Accuracy: 0.6066
  Dev Loss: 0.8002, Dev Accuracy: 0.5850 (Epoch 79)
Early stopping triggered after 79 epochs.
Best model saved with dev loss: 0.7939
训练完成.
```

1e-7,0.1



五、总结

1．代码实现时遇到了哪些 bug？如何解决的？

（1）最初训练完成时就进行测试文件预测，加载保存好的模型参数进行训练时一直出错，原因是因为 model 被_IncompatibleKeys 覆盖了，导致预测时模型加载出现问题，因此我后面把测试部分单拎出来放进了主函数，根据 config 的模式选择是否需要加载保存的模型进行预测，这样也更加灵活了

```
  Train Loss: 0.7656, Train Accuracy: 0.6131
  Dev Loss: 0.8161, Dev Accuracy: 0.5863 (Epoch 66)
Early stopping triggered after 66 epochs.
Best model saved with dev loss: 0.8149

Testing the model for different scenarios:

Scenario: Text Input Only
Traceback (most recent call last):
  File "c:\Users\admin\PycharmProjects\pythonProject\hw5\train_evaluate.py", line 188, in <module>
    train_and_test(train_dataloader, dev_dataloader, test_dataloader)
  File "c:\Users\admin\PycharmProjects\pythonProject\hw5\train_evaluate.py", line 79, in train_and_test
    get_test(model, test_dataloader, scenario="text_only")
  File "c:\Users\admin\PycharmProjects\pythonProject\hw5\train_evaluate.py", line 120, in get_test
    model.eval()
AttributeError: '_IncompatibleKeys' object has no attribute 'eval'
```

（2）开始时测试函数代码的处理逻辑不大对，导致模型在消融实验和非消融实验的三种数据类型下预测结果完全相同，不合理。后面修改了 get_test（）函数的逻辑，在接收数据时，根据场景选择来对特定数据集置空，从而达到不同场景不同预测方式的消融结果。

```
  Text_only Test Accuracy: 0.3603

Scenario: Image Input Only
  Image_only Test Accuracy: 0.3603

Scenario: Text and Image Input
  Text_and_image Test Accuracy: 0.3603
预测完成.
```

```
  Text_only Test Accuracy: 0.3534

Scenario: Image Input Only
  Image_only Test Accuracy: 0.3534

Scenario: Text and Image Input
  Text_and_image Test Accuracy: 0.3534
预测完成.
```

（3）GitHub 连接问题。更新远程仓库时发生连接超时错误、RPC 错误等。后面通过重新设置远程连接，增加缓冲区大小和超时时间，使用 VPN 加速的方法，多试了几次成功上传：

```
$ git push origin master
fatal: unable to access 'https://github.com/user110laolemon/AIhw5.git/': Failed to connect
to github.com port 443 after 21075 ms: Could not connect to server
```


```
$ git push origin master
fatal: unable to access 'https://github.com/user110laolemon/AIhw5.git/': OpenSSL SSL_r
ead: SSL_ERROR_SYSCALL, errno 0
```

2．实验心得

本次实验是这学期最后一次实验作业，相当于是之前学习的一个总结，把多模态模型进行融合，可以用到之前图像和文本相关模型训练和学习的知识。我选择的两种模型在单个领域中表现优秀，但是可能是因为融合的时候还是比较简单、层数比较少导致没有发挥出他们各自的性能，由于本次实验还是比较复杂的模型训练，我吸取了上次实验的教训，租用了云 GPU 服务器提高训练效率。本次实验收获很大，了解了融合不同模型训练时的细节，虽然训练结果没有我想的那么完美，但是从中学习到了很多经验，感谢这学期老师和助教的工作，谢谢！