**当代人工智能实验五-多模态情感分析**

**--10215501435 杨茜雅**

**实验任务：**

●给定配对的文本和图像，预测对应的情感标签。

●三分类任务：positive, neutral, negative。

示例：



'??? #stunned #sunglasses #gafas #gafasdesol \n' Positive

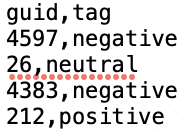
**实验数据集：**

●匿名数据集（实验五数据.zip）

●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: 数据的guid和对应的情感标签。



●test\_without\_label.txt：数据的guid和空的情感标签。



**实验要求：**

●设计一个多模态融合模型。

●自行从训练集中划分验证集，调整超参数。

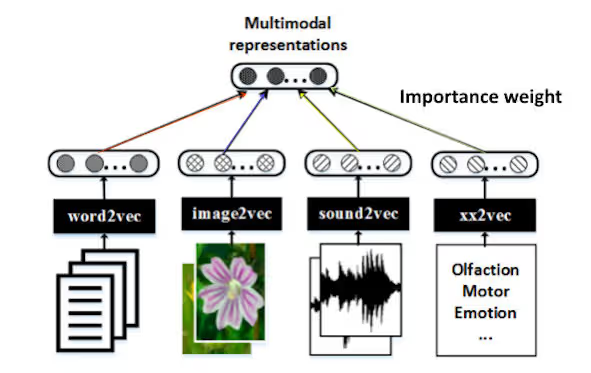
●预测测试集（test\_without\_label.txt）上的情感标签。

**代码仓库地址：**

<https://github.com/Lily127Yang/Contemporary-AI---Project-5>

**实验思路：**

本次实验要求实现的是一个多模态融合模型。多模态模型，顾名思义，是指能够处理多种类型数据（如图像、文本、声音等）的深度学习模型。与传统的深度学习模型相比，多模态大模型在训练过程中能够同时处理多种类型的数据，从而提高模型的泛化能力和鲁棒性。这使得多模态大模型在自然语言处理、计算机视觉、语音识别等领域具有广泛的应用前景。



我首先对数据集进行预处理，把所有的文字数据整理到一个文件中再把图像文字转化成可以训练的数据。接着对数据集进行探索，查看各类标签的占比并且划分训练集和验证集，得出neutral标签占比较少的结论，后续进行探索改进。接着开始搭建模型。

**模型结构**

模型以BERT和ResNet-152为基础，分别处理文本和图像输入。接着，通过自注意力机制实现多模态信息的融合。让我们逐步解析这个模型的关键部分。除了多模态处理外，该模型还支持单一模态的处理，即只有文本输入或只有图像输入。在这两种情况下，模型分别提取单一模态的特征，并通过分类器得到输出。

**特征提取**

模型首先通过BERT处理文本输入，获取文本的隐藏状态。这些隐藏状态包含了文本的语义信息，为后续的多模态融合做准备。对于图像输入，模型使用了预训练的ResNet-152模型提取图像的特征。通过对图像特征进行池化和线性变换，得到图像的隐藏状态。

**特征融合**

接下来，模型将文本和图像的隐藏状态进行拼接，构成共同的特征表示。通过设置attention\_mask，模型实现了对文本中padding部分的处理，并使用self-attention机制进行多模态融合。

**分类**

最后，模型分别提取多模态融合后的图像和文本特征，并通过线性变换进行分类，得到最终的输出。

不仅如此，我还将标签为 neutral 的数据条数做复制，再进⾏模型的训练和拟合。也进⾏了消融实验，分别对于仅⽂本和仅图像的准确率进⾏了探索，最后做出本次实验的总结。

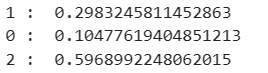
**1、****数据预处理**

**1.1****观察数据集**

train.txt共有4000条数据。为了方便模型预测之后计算准确率，将情绪标签按分类映射为0，1，2



并且查看每种情绪的占比如下。



data ⽂件夹中有若⼲对图⽚和txt⽂件，⽂件名相同的⽂件⼀⼀对应。两个txt⽂件分别是有情感标签的数据和需要被预测的无情感标签的数据，⽂件之间通过唯一的guid实现一一对应。



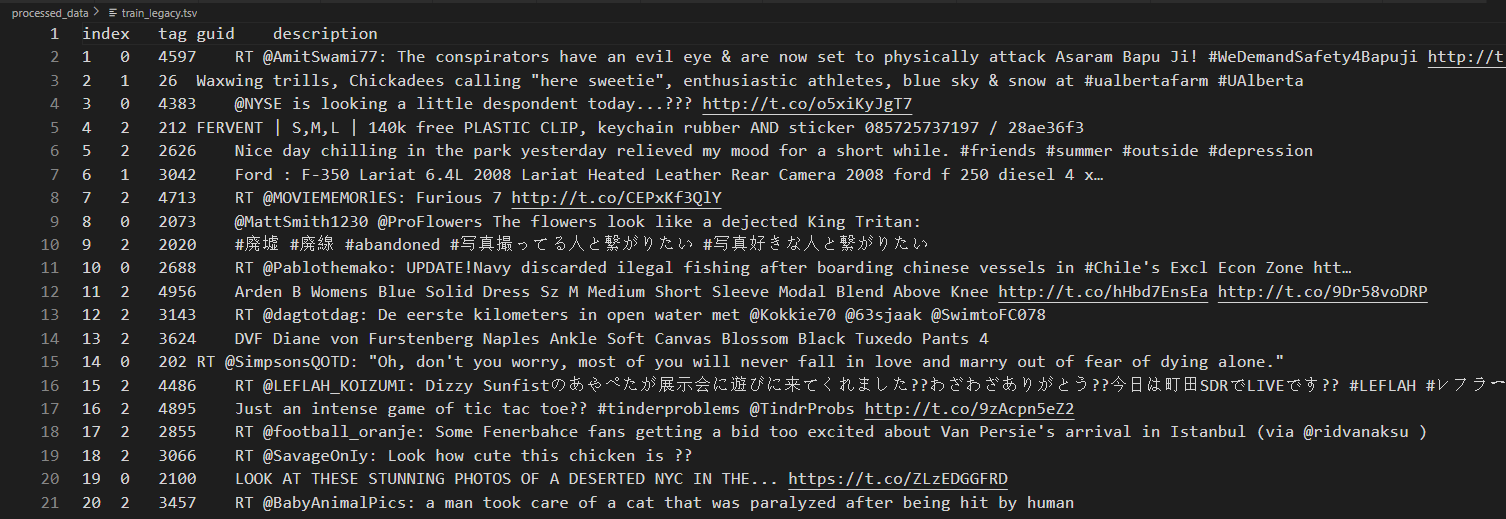
data.txt中的⽂件是⼀个⼀个句⼦，语句中存在一些对情感分类无用的字符串，例如：@xxx、#、http...等等，所以不能使⽤ csv 格式保存。为了把所有的⽂字数据都整理到⼀个⽂件中，我打算使⽤以 tab 为分隔符的 tsv ⽂件格式保存。

读取⽂件并⽣成 tsv 的⽅法在 generate\_data.py ⽂件夹下。下面说明以80%的训练集为例，20%的测试集和验证集做法一致。

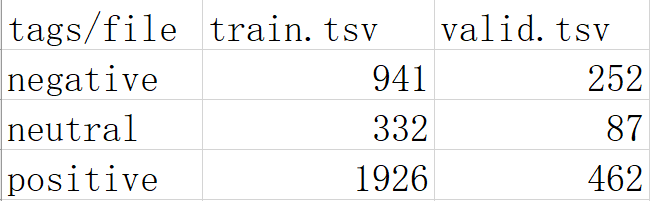
生成训练集（train.tsv）：写入表头后，遍历前3200行的训练数据（大约是80%的数据，用于8:2的训练/验证集划分）。对于每一行，它分离出标识符（guid）和标签（tag），读取对应的文本文件，并将文本内容和标签写入新的.tsv文件。



训练集和验证集以8：2的比例被划分，分别保存在 train.tsv, valid.tsv下，可以看到，每⼀⾏的第⼀列是 index，第⼆列是 tag转换成的数字，第三列是guid，第四列是⽂字描述。



查看 train.tsv, valid.tsv 中的 tag 种类，以及每个种类所占的个数

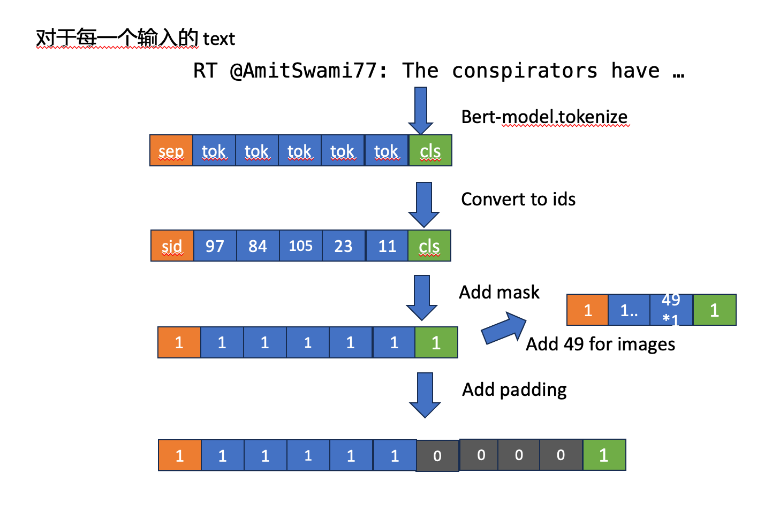


发现 neutral 标签所占的⽐例略少，如果真正在数据集上进⾏训练的话，可能会更容易造成过拟合。对于样本较为缺失的标签，⼀般的做法是**在数据集中重复这些较少的数据较多次**。尽管听上去没有什么理论⽀持，但是在实际操作时，这种⽅法是真实有效的。在许多 kaggle 竞赛上都得到了证实。

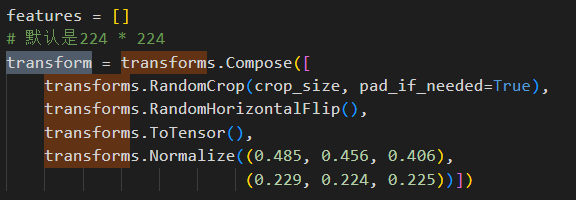
**后续的计划是将标签为 neutral 的数据条数做复制，再进⾏模型的训练和拟合，观察对⽐效果。**

**1.2****把⽂字图像转化为可以训练的数据**

我们读取刚刚得到的 tsv ⽂件，并且对其中的 text 进⾏ tokenize, convert\_to\_ids, add\_padding 等⼀系列操作，最后再加上⼀条数据，这条数据会留49个空格给图像（图像经过处理后⻓度都为 49）

****

对于图像数据，我们统⼀将其先转换为 224\*224 的⼤⼩，⽅便模型进⾏训练。

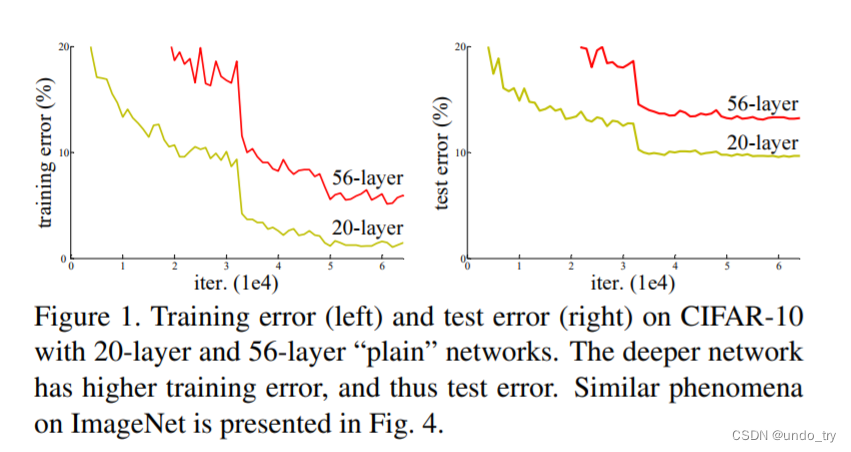


对图像使⽤ RandomCrop ⽅法统⼀尺⼨到 crop\_size 以后，进⾏随机⽔平翻转，转换成 torch.tensor 数据格式，并且进⾏归⼀化。

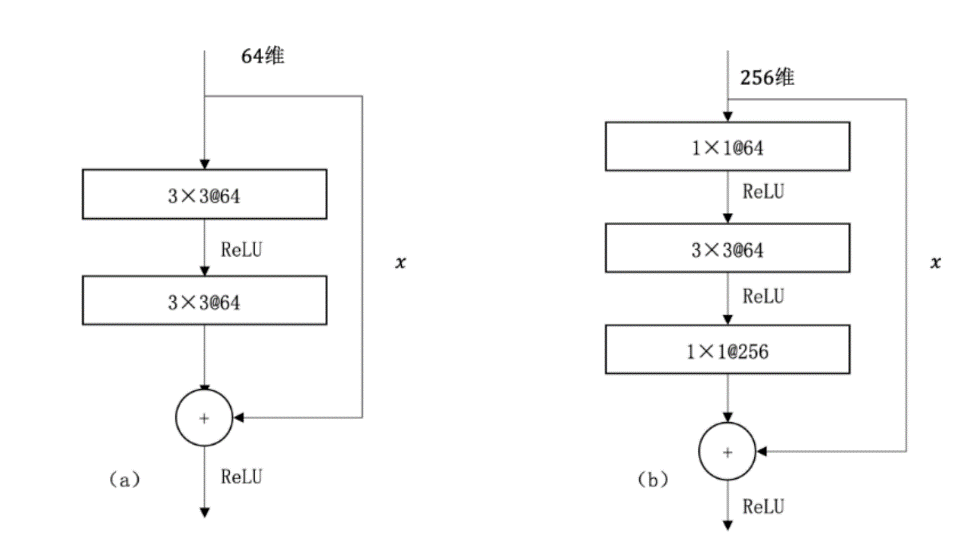
**2、****模型构建 BERT+RESNET-152**

我们在⽂字部分选取的预训练模型是 bert-base-uncased, 图⽚部分选取的预训练模型是 resnet-152, **因为在实验三中我所得到的结论是 resnet 是⼀个准确率很⾼并且参数相对较⼩、不容易出现过拟合的模型。**

残差神经⽹络的主要贡献是发现了 “退化现象(Degradation)”



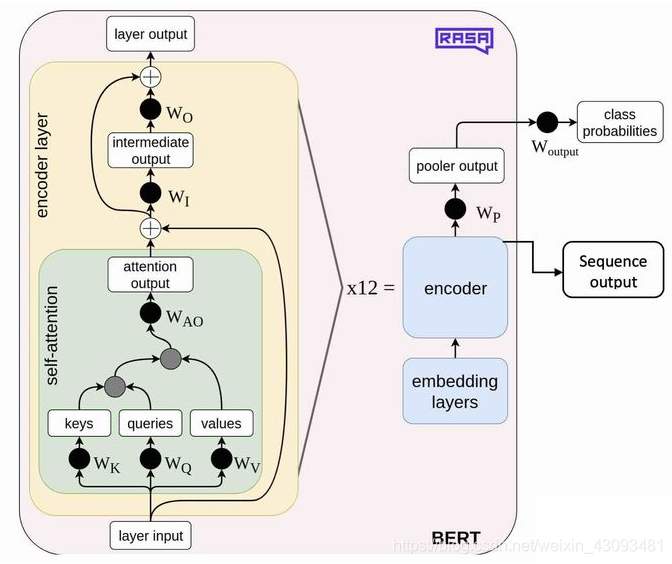
它针对退化现象发明了 “快捷连接 (Shortcutconnection)”，极⼤的消除了深度过⼤的神经⽹络训练困难问题。神经⽹络的 “深度” ⾸次 突破了 100 层、最⼤的神经⽹络甚⾄超过了 1000 层。⾮线性转换极⼤的提⾼了数据分类能⼒，但是， 随着⽹络的深度不断的加⼤，我们在⾮线性转换⽅⾯已经⾛的太远，竟然⽆法实现线性转换。显然，在 神经⽹络中增加线性转换分⽀成为很好的选择，于是，ResNet 团队在 ResNet 模块中增加了快捷连接分⽀，在线性转换和⾮线性转换之间寻求⼀个平衡。这不仅提高了网络的性能，也为深度学习的未来发展提供了新的思路和方向。ResNet的成功应用于多种计算机视觉任务，如图像分类、物体检测和图像分割，证明了其强大的功能和通用性。



**2.1** **Bert模型搭建**

分为以下步骤：

把输⼊先做 embedding，并且将 token embeddings, segmentation embeddings, position embeddings 进⾏加和。将三种不同的 embedding 进⾏加和与单独的 embedding 输⼊相⽐，使⽤了不同的⻆度和⽅式进⾏了编码，可以让模型学到更多的特征信息。

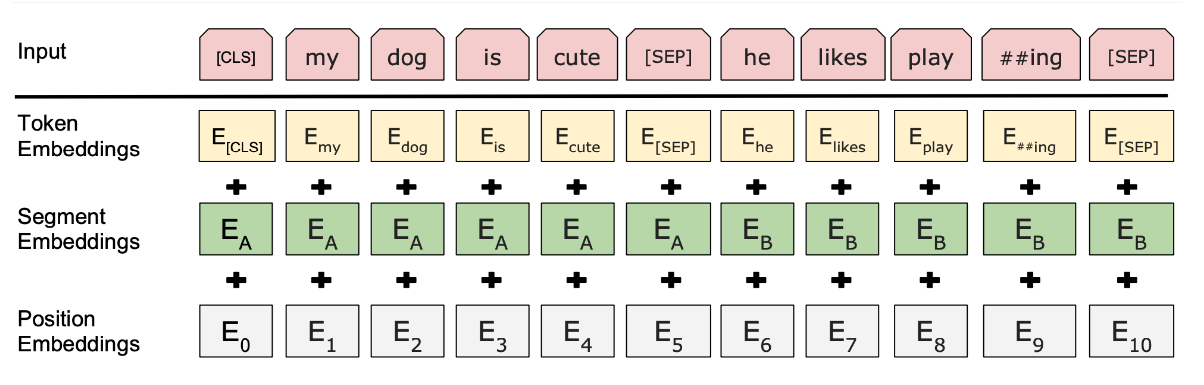


●Token Embeddings：⽤于将输⼊的单词序列转换为向量表示，作⽤是将每个输⼊单词都表示为⼀个向量，

包含了该单词的语义信息。

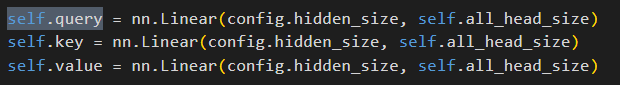
●Segment Embeddings：⽤于区分两个句⼦中相同位置的单词，作⽤是将不同句⼦中相同位置的单词进⾏区分，避免不同句⼦中的单词产⽣⼲扰。

●Position Embeddings：⽤于表示单词在输⼊序列中的位置，作⽤是将输⼊序列中单词的位置信息编码成向量，以便模型能够理解单词在序列中的顺序关系。

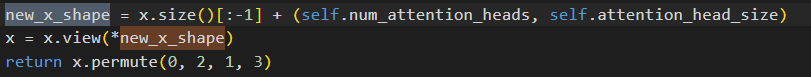


之后我们添加层归⼀化和dropout 防⽌过拟合与防⽌梯度爆炸或者消失，并且将输⼊的数据进⾏应⽤多头注意⼒机制。

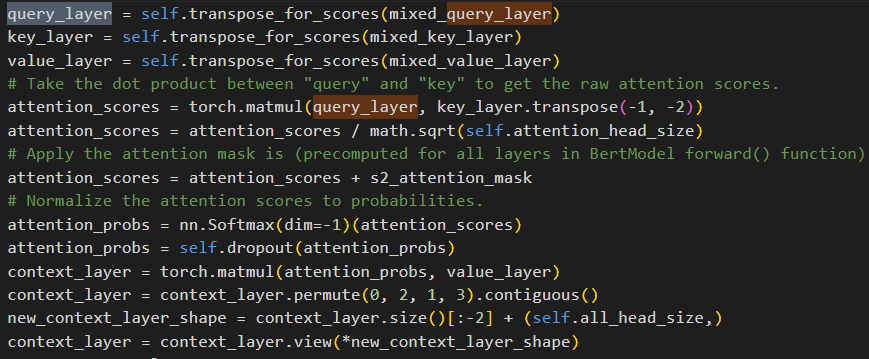
分别定义 query、key、value



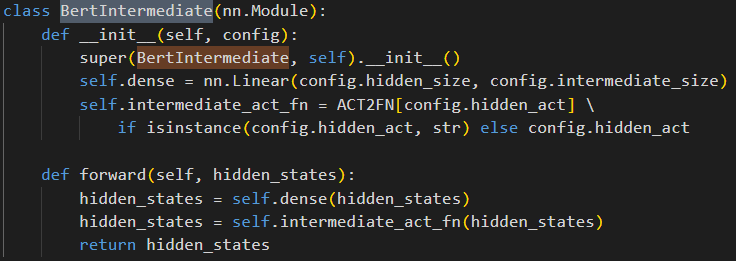
将 它们 进⾏multihead拆分，拆成12个头, 拆分后保存在key\_layer, query\_layer, value\_layer 中，维度为（1,12,128,64）



attention\_scores 矩阵是由 Q 和 K 矩阵相乘得到的，⽤于计算注意⼒分数，注意⼒权重经过 Softmax 函数和Dropout 操作后，我们可以得到每个单词对其他单词的注意⼒权重，并计算上下⽂向量。最终返回的是上下⽂向量，将每个注意⼒头的向量拼接在⼀起，形成⼀个更⼤的向量维度。

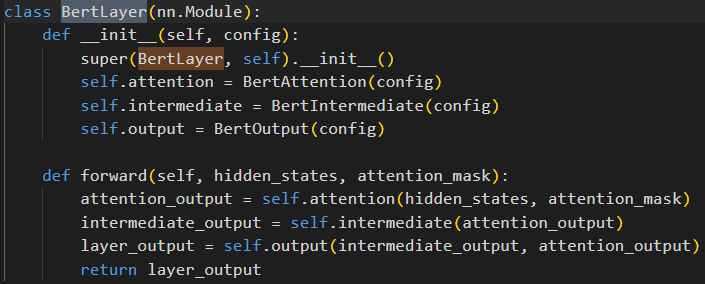


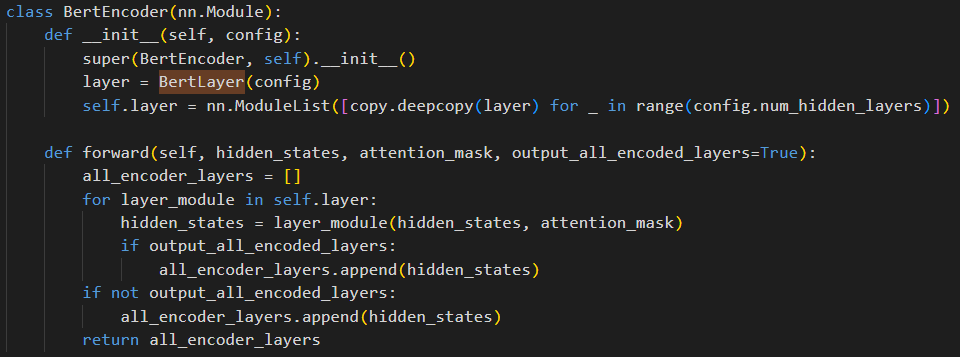
intermediate 层，也就是 Feedforward 层，⽤于对输⼊的隐藏状态向量进⾏⾮线性变换，提取更⾼层次的特征表示



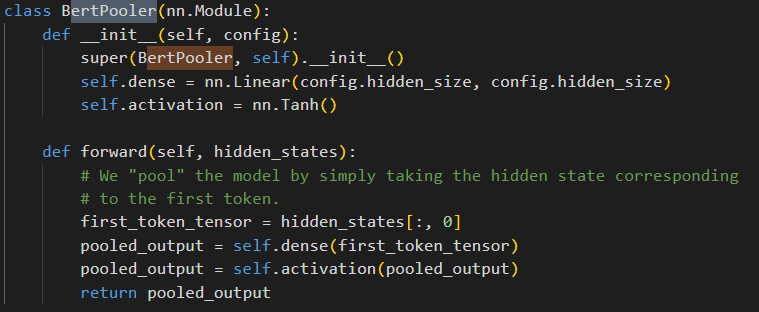
根据以上的代码，可以构建⼀个 bert encoder. bert encoder 层⼀层包括 bert input，bert layer 和 bert

output，其中 bert layer 层⼜包括 self-attention 层和 intermediate 和 output 层。

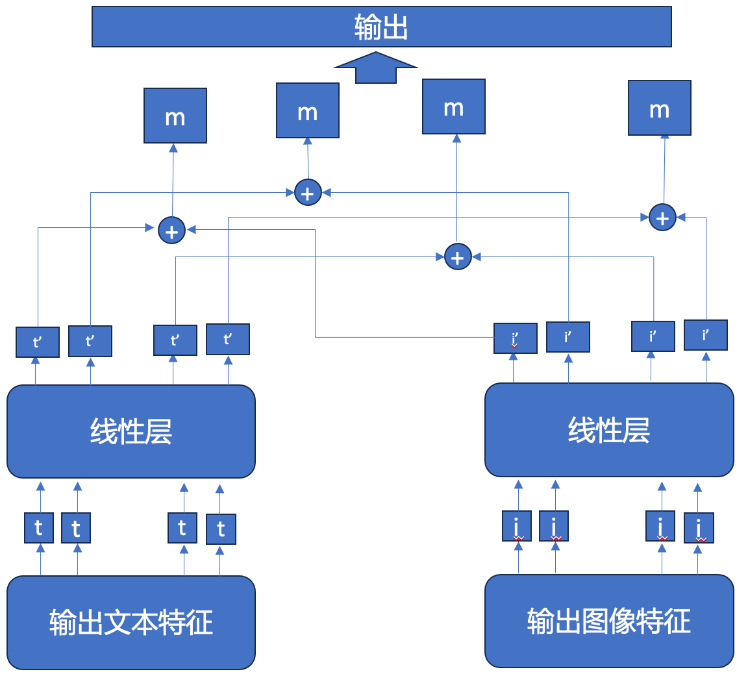




定义 pooler 层，作⽤是将模型的最后⼀层输出向量转换为⼀个固定⻓度的向量，⽤于表示整个输⼊序列的语义特征，可以⽤于下游任务的处理，如分类、序列标注等。



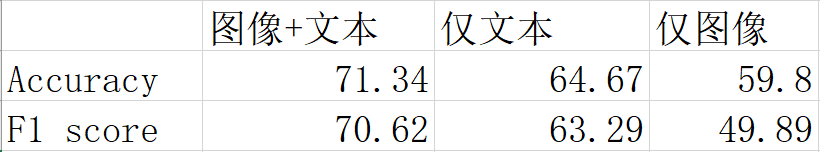
然后我们还需要接受多模态输⼊。我们可以将 ⽂本+图像的输⼊也做⼀个 cross attention。Transformer 解码从完整的输⼊序列开始，但解码序列为空。 交叉注意⼒将信息从输⼊序列引⼊到解码器的各层，以便它可以预测下⼀个输出序列标记。 然后解码器将令牌添加到输出序列中，并重复此⾃回归过程，直到⽣成 EOS 令牌。



**3、****消融实验结果**

分别只输入文本或图像数据，观察多模态融合模型在验证集会获得怎样的表现

输⼊ 图像+⽂本，仅⽂本，仅图像 的准确率和 f1 值分别如下：



可以发现，我们的模型在多模态问题上的表现较好，判断正确率在 0.7 左右。并且远⾼于两个单模态的模型输出结果。推测⽂本数据表现好于图像数据的可能性是**，⼀个图像可以表示更多范围的情绪，⽽⽂本似乎略少⼀些**。

**3.1****对数据集的标签进⾏探索，训练**

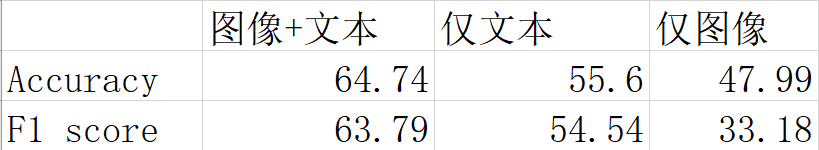
**前⽂提到， neutral 标签所占的⽐例略少，如果真正在数据集上进⾏训练的话，可能会更容易造成过拟合。对于样本较为缺失的标签，⼀般的做法是在数据集中重复这些较少的数据较多次。**

定义了 fix\_overfit ⽅法，⽤于给训练集和验证集进⾏缺失标签的数据填补。把训练数据和验证数据⾥⾯ label 为 1的数据先添加在末尾，然后将⾥⾯的⾏随机排序⽣成新⽂件。

修改以后，训练数据集中，label 为 0,1,2 的数据分别为 941 995 1926 条。验证数据集中，label 为 0,1,2 的数据分别为 252 260 461 条。**相⽐于以前 2:1:3 的数据分布⽐，现在 1:1:2 的数据分布⽐似乎更加健壮。**

同时，发现数据集中有多条⽇语、俄语数据，我将其都翻译成了英⽂，来提升模型的准确率。

进⾏训练与验证，得到的结果如下：



发现得到的准确率⽐原数据集更低，这也⽐较合理，因为原本的数据集，与其说是⼀个三分类问题，不如更加靠近⼀个⼆分类问题。假设模型有先验知识，知道各种标签所对应的数据⽐例是 2:1:3, 那么准确率的期望值就是**0.5\*0.5 + 0.33\*0.33 + 0.16\*0.16 = 0.375。** ⽽如果知道各种数据的⽐例是 1:1:2, 准确率的期望值是 **0.25\*0.25 +0.25\*0.25 + 0.5\*0.5 = 0.365**, 会略低⼀些。

但是双模态和仅⽂本的表现稍微还符合这样的⽐例，但是仅图像就太低了。F1 score 到了 33.18，不如随机猜测。因此判断这个模型的效果不好。

我们来查看预测⽣成的数据，在 predict ⽂件夹下。旧数据集和新数据集所对应的预测结果分别是

text\_without\_label\_legacy.txt 和 text\_without\_label.txt 。发现前者中，标签为 positive 的较多，并且三种标签分布得较为均匀。⽽后者中，标签为 neutral 的过多，和原始数据的分布不太吻合，舍弃。

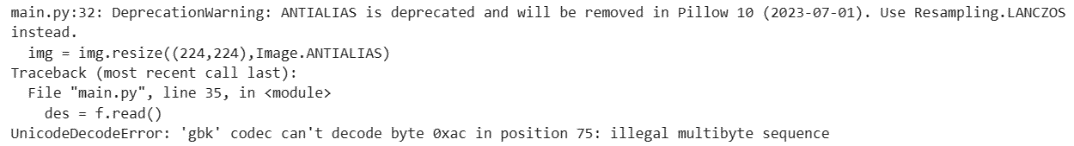
所以⽬前来讲，**在这个例⼦中，使⽤原数据集表现更好**。

**4、****总结**

在本次实验中，我用多模态融合模型对具体数据进⾏了建模，使⽤ bert模型 和 resne-152模型对于⽂本和图像分别进⾏了变换并且进⾏向量拼接，最后喂进模型训练，取得了较好的表现。同时，我对于数据集进⾏了探索，发现如果重复标签为 neutral 的数据过多，⽆论是训练结果还是预测结果都不太理想。我们也进⾏了消融实验，分别对于仅⽂本和仅图像的准确率进⾏了探索，对于多模态模型的表现有了更深刻的了解。

**5、****遇到的问题及解决措施**

5.1 读入图片描述时，出现无法解码的问题



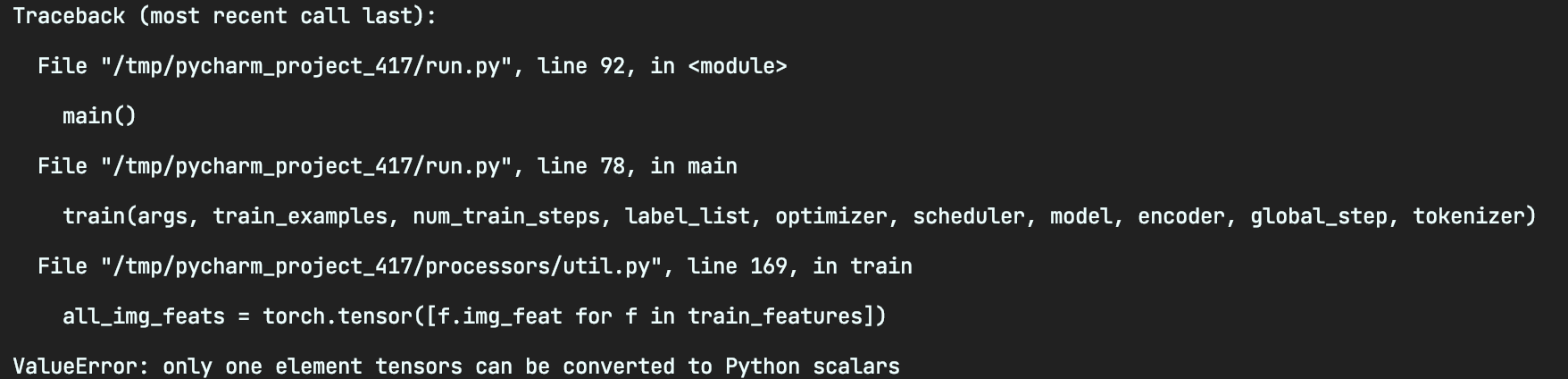
解决方案：根据代码给出的提示信息，可以看出错误的意思是：Unicode的解码（Decode）出现错误了，以gbk编码的方式去解码（该字符串变成Unicode），但是此处通过gbk的方式，却无法解码（can't decode）.''illegal multibyte sequence"的意思是非法的多字节序列，也就是说无法解码了。出现这样的错误，可能是要处理的字符串本身不是gbk编码，却是以gbk编码去解码。比如，字符串本身是utf-8的，但用gbk去解码，必然出错。故采用更宽范围的编码进行解码， with open('./data/' +str(guid) + '.txt', encoding='gb18030') as f: ，问题即可解决。

5.2 运行模型时，出现以下问题



解决方案：该问题出现的原因是构建的模型类没有实例化就使用了，进行实例化之后问题解决

5.3 使⽤torch.tensor转换时，忘记之前已经通过torch.transformer转换成了⼀系列torch.tensor数据类型



解决方案：list 列表中每⼀个元素都是⼀个 tensor 类型的数据，所以应该使⽤ torch.stack ⽅法将这些 list ⾥⾯的元素拼接起来变成⼀个 torch.tensor 类型的数据。

**六、这样设计模型的****原因和亮点**

**原因：**

本项目需要分别提取文本的特征以及图片的特征，将二者特征融合之后再进行情绪的分类。对于文本进行情感分析，就是需要得到整个文本的上下文的综合信息，想到Bert模型在处理文本信息时，内部采用自注意力机制，最后对每个字输出一个向量，并且在句子开头添加了一个特殊符号'CLS'，该符号对应的向量中包含了整个句子的综合信息。所以，进行情绪分类的文本特征考虑采用'CLS'的向量信息。对于图片的特征提取，考虑使用在ImageNet上预训练过的Resnet，之所以选择Resnet是因为我在实验三中得出过resnet 是⼀个准确率很⾼并且参数相对较⼩、不容易出现过拟合的模型这一结论。越深的网络提取的特征越抽象，越具有语义信息，Resnet中有残差结构，该结构使得训练更深的网络时缓解梯度消失或者梯度爆炸的问题。

**亮点：**

进行实验之后，我对数据集进行了探索优化并且重新带入模型进行实验。我认为数据预处理也是“建模”的一部分，在以后的学习或者工作中我们有可能会拿到标签分布不均匀的数据，那我们是否可以对数据进行一些合理的加工而不是一定要使用raw data呢？带着这样的想法，我定义了 fix\_overfit ⽅法，⽤于给训练集和验证集进⾏缺失标签的数据填补，使数据分布变得更为健壮，但是最后的准确率比原本的模型更低，对此我也做出了自己认为的可能的解释。虽然结果对模型没有实际帮助，但这只证明这样的加工方法不适用于本次实验，不证明不适用于其他的情况，不管怎样，这给我以后处理数据提供了一个新的思路。