Report for PA3 Sentiment Classification

温佳鑫 计 84

2017010335

Department of Computer Science and Technology

wenjx17@mails.tsinghua.edu.cn

一、模型结构图及流程分析

1．MLP

[结构]:embedding+linear(,512)+dropout+relu+linear(512,32)+relu+linear(32,8)

[流程分析]:设置 fix_length 后输入即可，无特殊操作。

2．RNN

[结构]:embedding+dropout + biGRU + linear(2*hidden_size,8)

[流程分析]:直接输入即可，无特殊操作

3．RNN+Attention

[结构]:attention 实现借鉴自论文《Attention-Based Bidirectional Long Short-Term Memory Networks for Relation Classification》，主要公式如下。

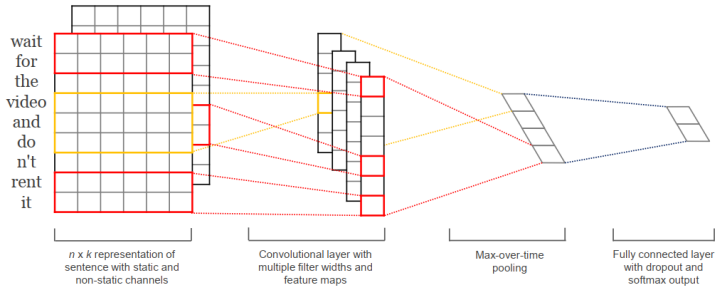
$$M = \tanh(H) \tag{9}$$
$$\alpha = softmax(w^T M) \tag{10}$$
$$r = H\alpha^T \tag{11}$$

[流程分析]:经过 RNNcell(biGRU or biLSTM)后，加入 attention 计算，再接一层 Linear 得到概率输出。

4．CNN(TextCNN)

[结构]:使用了 TextCNN 的模型结构, 卷积层超参数设置与论文中一致，即使用 filter windows of 3,4,5 with 100 feature maps each

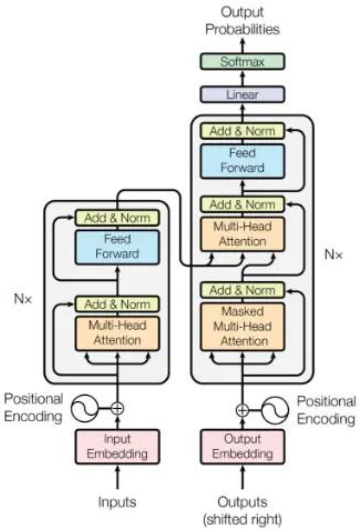
[流程分析]：首先用三种卷积核卷积，各得到 100 个 feature map, relu 激活一次，经过 maxpooling 层得到三个 100 维向量，concatenate 后,再过一层 dropout 和全连接，最终得到 8 维的概率输出。



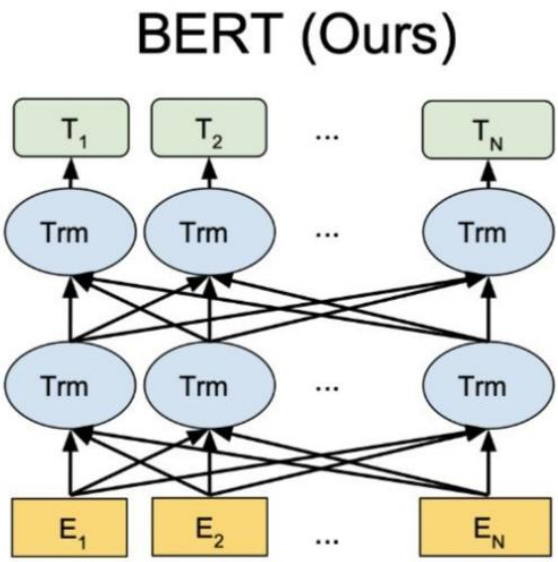
5．BERT

[结构]:bert+Linear

transformer 结构:



bert 结构:



[流程分析]:截取序列头部 128 位和尾部 382 位，头尾分别加入 cls 和 sep，直接输入即可,BERT 输出 768 维向量，经过线性层得到 8 维的概率输出

二、实验结果

model	词向量	Fix_length	epoch	Learning rate	Batch size	Weigth decay	accuracy	F1-score	correlation
MLP	mixed	1560	20	1e-4	128	1e-3	0.5722	0.5722	0.5792
CNN	sogou	512	20	1e-3	50	0.0005	0.6052	0.6052	0.6445
RNN(biGRU)	None	None	20	1e-3	16	1e-5	0.5192	0.5192	0.5258
BERT(head+tail)	None	512	20	1e-6	6	0.0005	**0.6232**	**0.6232**	**0.6799**
biGRU+Attention	sogou	None	100	1e-3	32	1e-3	0. 5976	0. 5976	0. 6259
LSTM+selfAttention	sogou	None	100	1e-4	32	1e-4	0. 5789	0. 5798	0. 6075

三、参数比较

1. 词向量

从 https://github.com/Embedding/Chinese-Word-Vectors)选取了三个词向量文件。分别为

(1) sogou: Sogou News 搜狗新闻

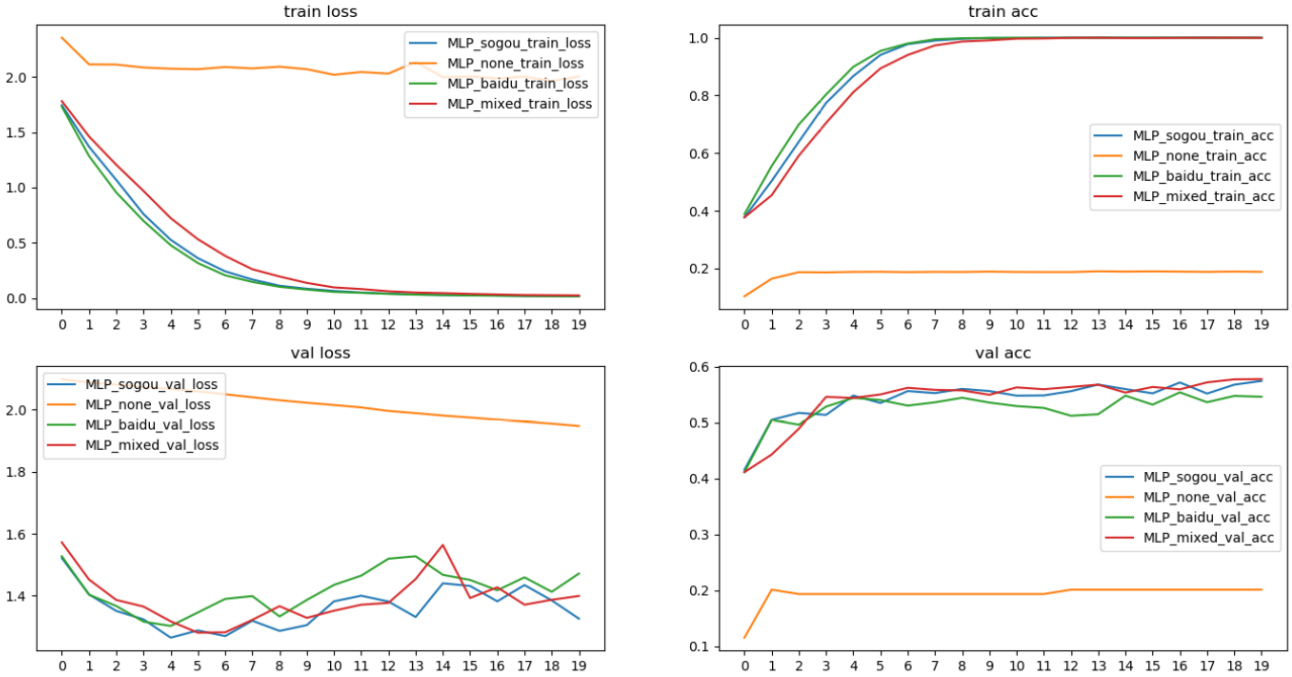
(2)baidu: Baidu Encyclopedia 百度百科

(3)mixed: Mixed-large 综合

选取 MLP,RNN 两个模型，分别测试以上三种词向量及不使用词向量的效果。

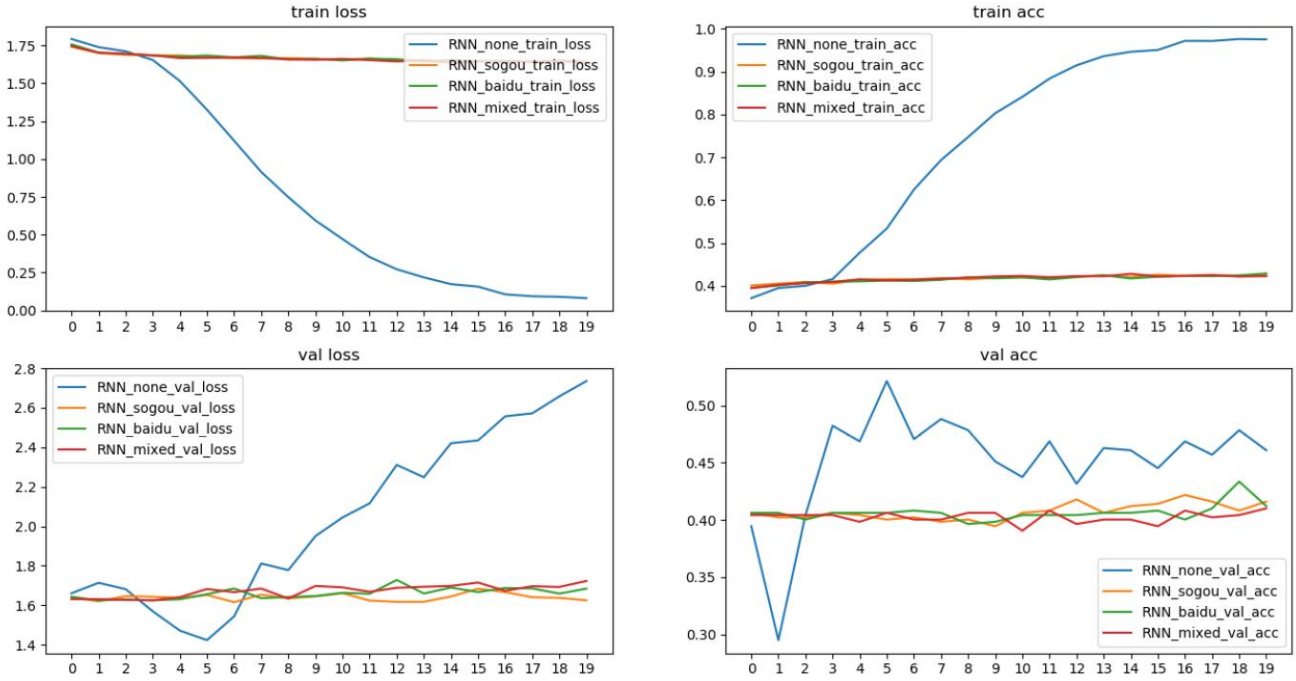
1.1 测试 MLP 使用不同词向量的效果

预训练词向量对模型的效果有大幅度提升，三种词向量之间差别不大。



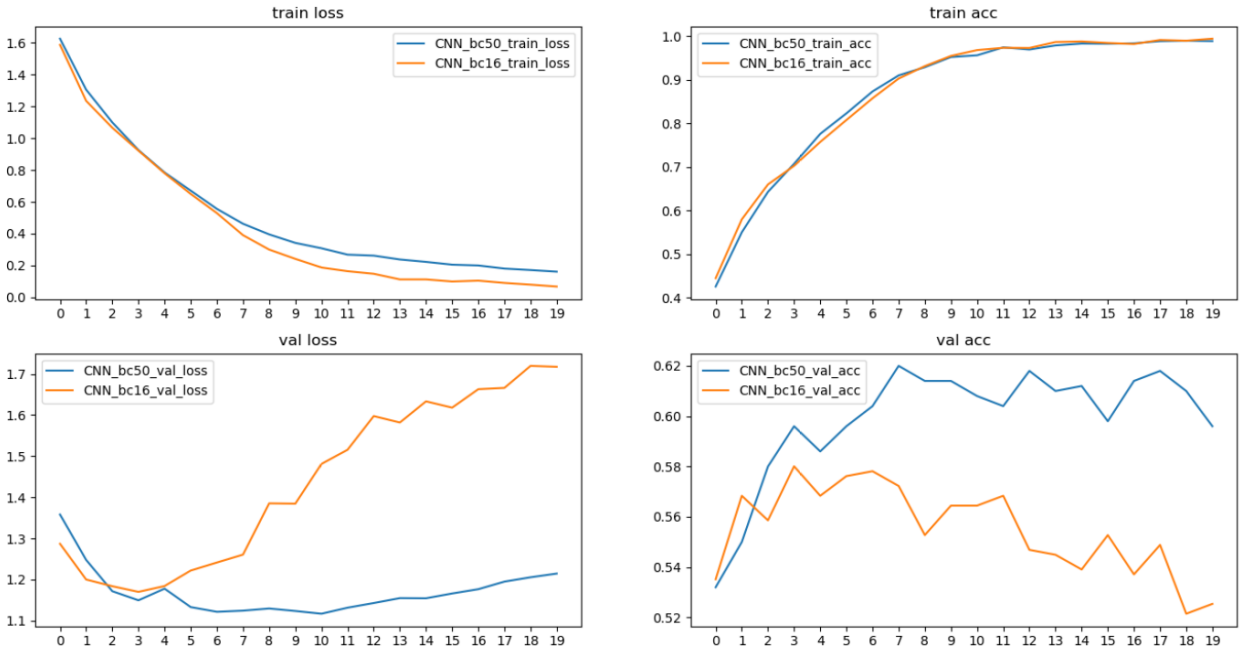
1.2 测试 RNN(biGRU)使用不同词向量的效果

三种词向量之间差别不大，但与 MLP 不同的是，不使用预训练词向量的训练效果反而更好。



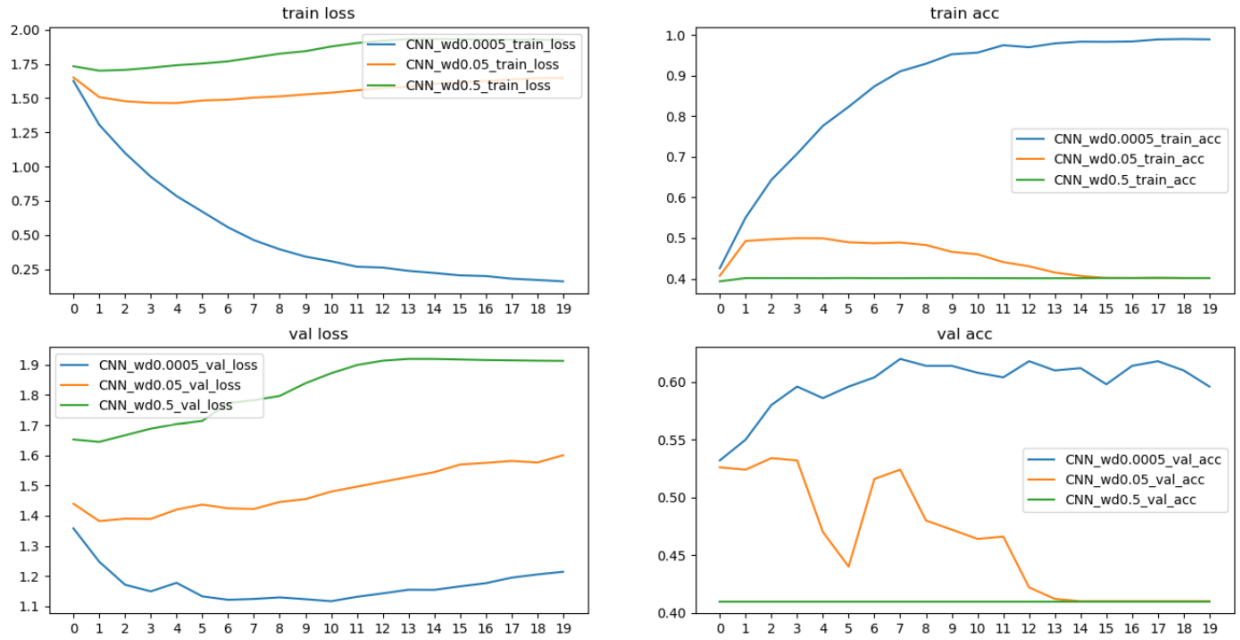
2. batch size

通常可以认为,在显存允许的情况下，batch_size 越大，越能够代表样本总体，求得的梯度方向可以更准确地朝向极值所在的方向，优化的效果更好。我以 CNN 模型为基础，记录 batch size 50 和 batch size 16 的效果，曲线图如下。从验证集的表现可以看到 batch size 50 的设定下模型有更好的训练效果。



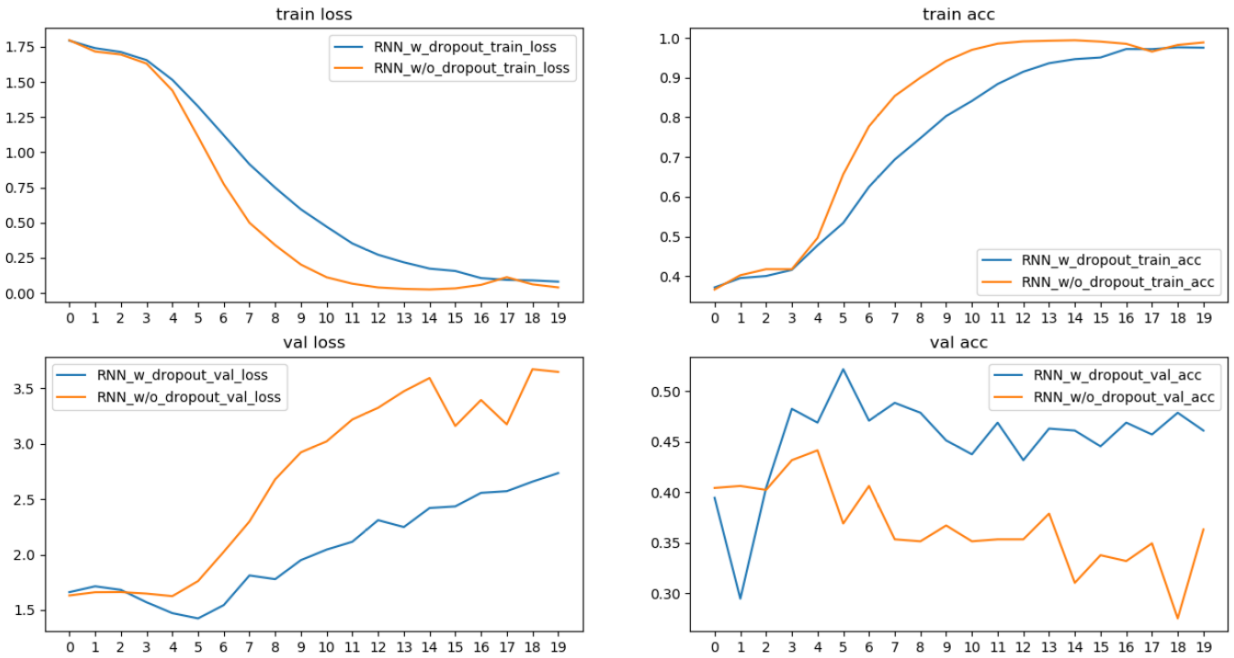
3. weight decay

weight decay，即 L2 正则化，是防止过拟合的有效手段。我以 CNN 模型为基础，分别尝试了 0.5,0.005,0.0005 的 weight decay 设定，曲线图如下。可以看到随着 weight decay 的增大，模型在训练集上的表现明显变差。



4. dropout

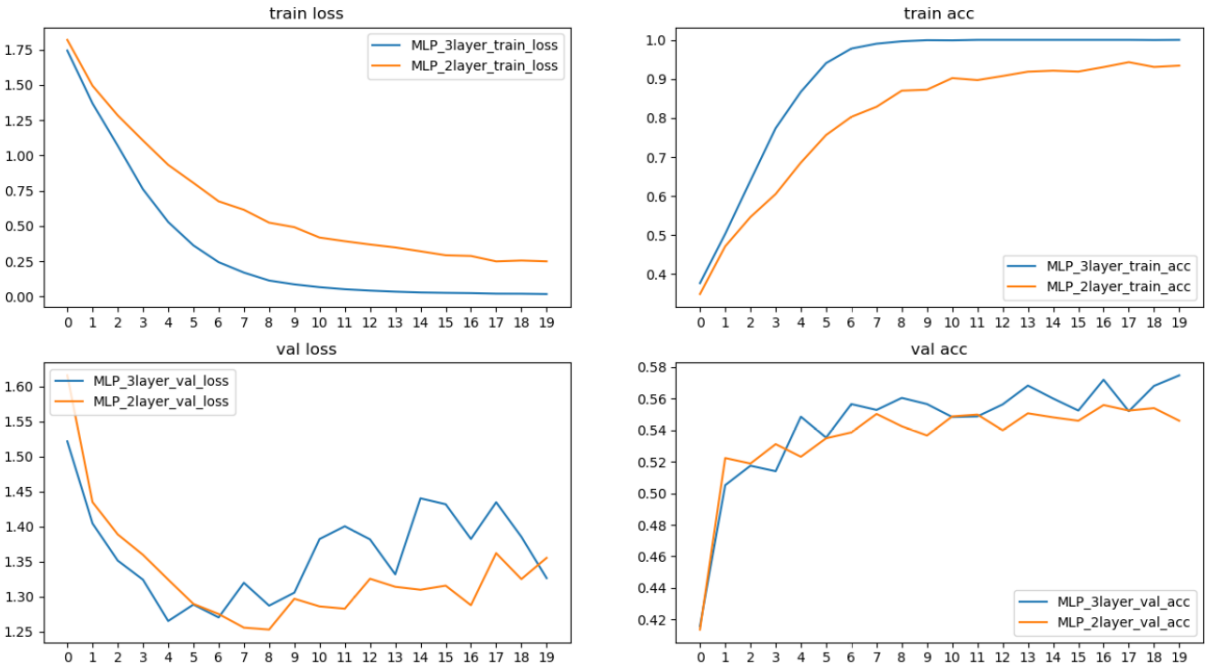
以 RNN 模型为基础，测试 dropout 的效果，曲线图如下。可以看到 dropout 有效的抑制了模型的过拟合情况。



5. 网络层数

网络越深越能够学习到深层特征，模型会有更好的表现。我选取 MLP 模型为基础，分

别测试两层与三层 mlp 的效果，曲线图如下，可以看到 3 层 MLP 有明显提升。



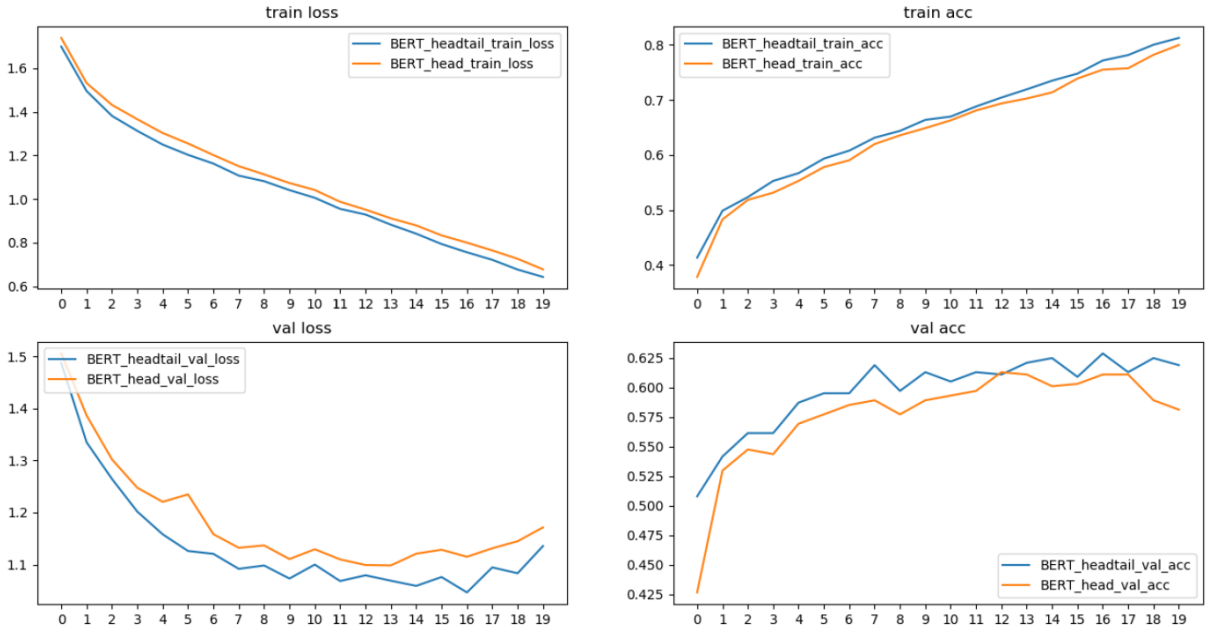
6. BERT head/ head+tail

BERT 对输入有 512 的长度限制，本次作业的数据集中大部分数据都超过了这一限制。

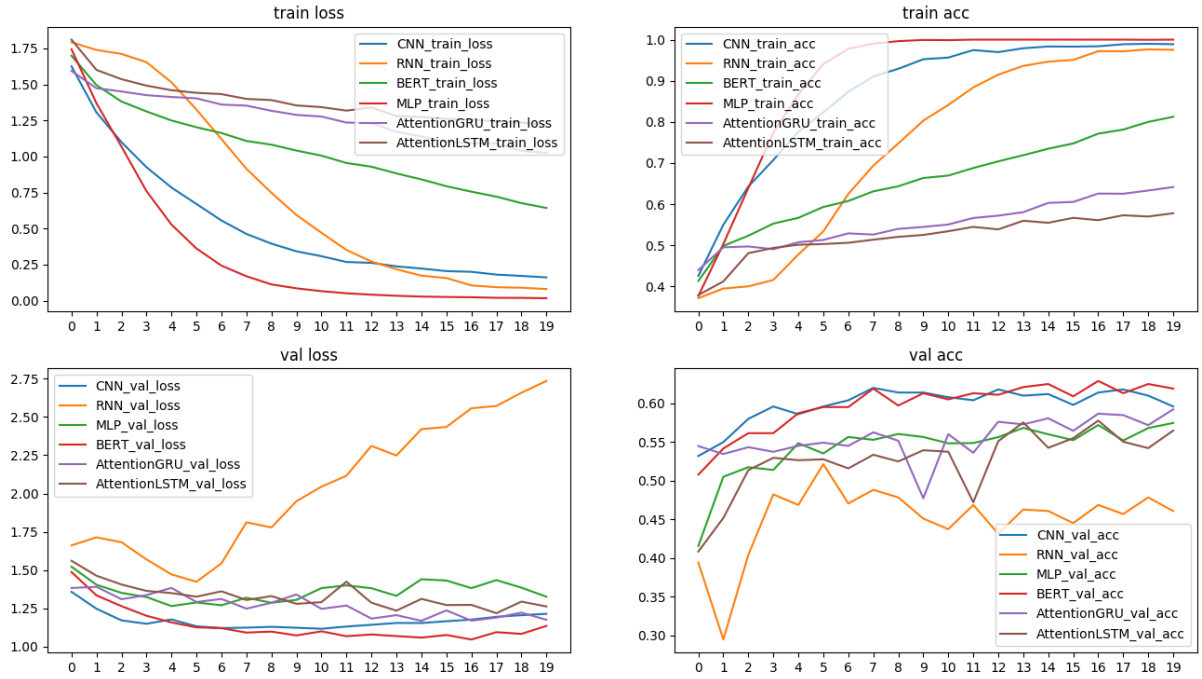
BERT 处理长文本常见的做法有从头截断 512，拼接头尾，等等。

我分别尝试了这两种做法，曲线图如下，可以看到拼接头尾的效果要优于从头截断。

同时我也尝试了第三种做法，将长文本以固定长度(如以 256 的长度截断)，同时每段之间保留 64 的 overlap，目的是一定程度上保留上下文信息，使得各段不完全独立，随后分别输入 BERT，拼接每一段的输出(768 维的向量),做 max_pooling，再接一层 MLP 分类，但由于显存限制，我最多只能将 batch size 设为 2，acc 在 47 左右停滞不动，无法测试模型的真实效果， 因此未加入报告中。



四、比较 baseline 模型与 CNN,RNN 模型的效果差异



1.准确率

BERT 准确率最高，第二是 TextCNN，第三第四是加入 attention 机制的 RNN（相较于基本 RNN 有了明显提升），然后是 MLP，最后是 RNN。

2.收敛情况

图中 BERT 和加入 attention 机制的 RNN 在 20 个 epoch 内在训练集上还未收敛，收敛速度较慢，其余模型在 20 各 epoch 内均已收敛。

(BERT 与加入 attention 机制的 RNN 我均做过 100 个 epoch 的实验，前者虽然最后在训练集上会收敛到近似 1 的值，但验证集上的效果不佳，最终选择的模型参数依然来自于 20 个 epoch 的训练结果；后者最后会在训练集上收敛到近似 0.65 的值，验证集上的准确率稍有提高，最终选择的模型参数来自于 100 个 epoch 的训练结果。）

3. overfit 情况

MLP 与 RNN 存在较强的 overfit 情况，TextCNN,BERT,尤其是加入 attention 机制的 RNN,overfit 情况有明显改善。

五、问题思考

1. 实验训练什么时候停止是最合适的？简要陈述你的实现方式，并试分析固定迭代次数与通过验证集调整等方法的优缺点。

(1) early stop

停止标准的设定方法有很多，最常见的一种是记录 $E_{opt}(t)$为迭代次数为 t 时取得的最好验证集误差，设第 t 次迭代的验证集误差 $E_{va}(t)$，定义泛化损失 $G(t) = 100*(E_{va}(t) / E_{opt}(t) – 1)$，即当前次迭代相较于此前最低误差的增长率，超过我们设定的阈值时就停止。

(2) 固定迭代次数

我选择的是后一种。

第一种方法相较于第二种方法，优点是可以一定程度上地减少训练时间，缺点也是由于迭代次数较少从而较为缺乏对模型真实性能的判断能力。事实上固定 epoch 的训练方式同样可以根据模型训练过程中在验证集的表现，对 max_epoch 做调整。

2. 实验参数的初始化时怎么做的？不同的方法适合哪些方法？（现有的初始化方法为零均值初始化，高斯分布初始化，正交初始化等）

Attention 中的权重矩阵初始化使用的是标准正态分布，其余参数初始化均用的是 pytorch 的默认初始化。线性层和卷积层使用的均是 kaiming initialization, RNN 使用的是均匀分布，代码可见于 linear.py, conv.py, rnn.py 的 reset_parameter 函数 。

kaiming initialization 是针对 relu 激活提出的，相比之下 xavier initialization 不太适合 relu 激活，更适合 tanh 激活。正交初始化主要用于解决深度网络梯度消失和梯度爆炸的问题，因此适用于 RNN。

3. 过拟合是深度学习常见的问题，有什么方法可以防止训练过程陷入过拟合。

3.1 Data-level: data augmentation

效果是多样的，可以减弱原始训练集中的 bias(比如训练集中带有 no,not 等词的句子均被标注为 negative),可以丰富训练数据的多样性，更符合真实场景。尽管相较于图像，自然语言的数据增强由于其离散性要更为困难，但目前也有许多可以尝试的方式，比如 word-level 的同义词替换，删除，插入；sentence-level 的改写，插入等等。

3.2 Model-level

(1)简化模型结构，比如减少网络层数，减少神经元个数等等。

(2)batch-normalization

(3)Dropout:训练过程中以概率 p 抑制一些神经元的输出，使得模型不会过分依赖某些局部特征，增强模型的泛化能力。

(4)正则化：比如增大 weigth decay,具体效果此前已通过曲线图展示。

3.3 训练过程:

观测验证集 loss,acc 的变化，及时停止

4. 试分析 CNN，RNN，全连接神经网络(MLP)三者的优缺点。

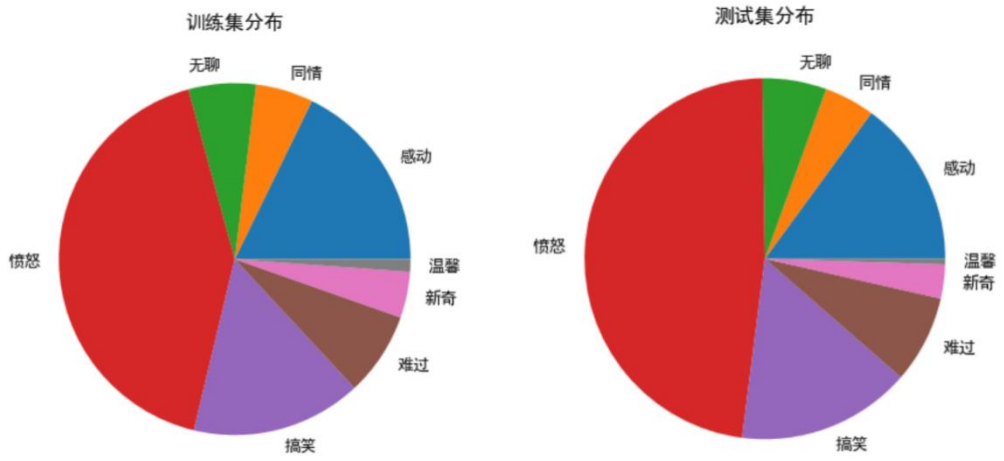
	CNN	RNN	MLP
优点	训练快，参数规模小，有很强的局部特征提取能力，当分类由句中的几处 key-phrase 决定时(比如 not,happy)，CNN 更适合。	输入长度可变，擅长处理 sequential data。当分类由整句决定时，RNN 更适合。	训练快，可以学习到全局的特征信息。
缺点	输入长度固定	训练较慢，容易出现梯度消失和梯度爆炸的问题。	输入长度固定，参数规模大

六、心得体会

之前上人工神经网络课程时也做过情感分类任务，但主要考察的是 RNNcell 的实现，对 accuracy 要求很低。本次实验为了提升 acc 做了大量的尝试，收获了很多。

首先我对调参的重要性，调参带来增益的上限，模型结构设计的重要意义，以及模型设计中的 trick 有了更为深刻的认识。过程中我也多次阅读 pytorch 源代码，增强了对深度学习框架的理解。在 BERT 的实验过程中，我认识到大型预训练模型本身的局限性(刚刚公布的 GPT-3 应该 forward 都跑不起来了)。我还学习了不平衡数据集的一些处理方法。

（未在前面报告中提到的一点是，我还统计了训练集与测试集类别的分布，观察到的现象可以概括为以下两点:（1）存在明显的 unbalance 现象（2）训练集与测试集的类别分布十分相似。于是我尝试在交叉熵的计算过程中根据各类别在数据集中的频率加入权重(即 CrossEntropyLoss 中的 weight 参数)，但效果并不佳。）



最开始学深度学习的时候感觉很玄学，但随着学习的深入，我发现越来越多十分有趣且有意义的想法，对深度学习的研究也越来越感兴趣。这学期也开始在跟师兄做项目，希望以后可以做出自己的成果。感谢老师助教的指导！