深度学习第三次实验作业报告

课程/2023 深度学习　学号/SA23229086　学生/郎文翀
大数据学院计算机技术班
2023 年 12 月 17 日星期六

1 实验要求

编写 RNN 的语言模型，并基于训练好的词向量，编写 RNN 模型用于文本分类并进一步设置不同超参数分析不同的超参数对模型预测推理的标签 Top1 准确率的影响。

2 实验设计

2.1 设计需求

本次实验选择 Pytorch 实现，并使用 GPU 版本进行训练，最后采用 tensorbaord 完成绘图与实验结果分析，具体程序设计需求如下：

数据集处理：数据集使用助教提供的 yelp3 数据集，由于并未提前划分，因此需要手动将数据集按照 6:3:1 划分为训练集，验证集和测试集。

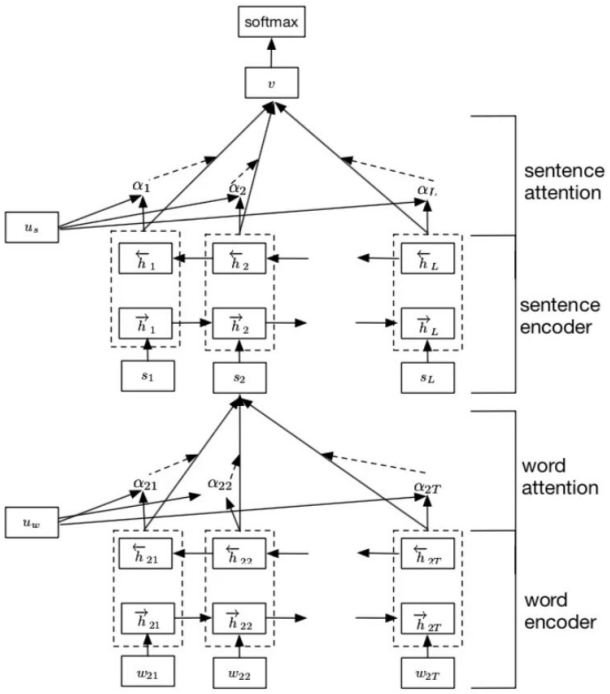
模型设计：本次实验需要参考 HAN 论文自行设计并实现模型。具体需要首先对文本进行词语划分并为不同的词语设置不同的标记进行编码后使用 embedding vector 进一步输入到 HAN 模型中结合 RNN 与 attention 机制实现模型对不同文本中语气词与最终评分的推理预测。

模型参数设置：尝试设置不同的超参数包括模型深度，编码向量宽度，是否使用 dropout 等分析不同方法对模型的影响。

分批次训练：每 Epoch 数据分为 256 一组进行训练。

2.2 HAN 论文分析

由于本次实验参考论文《"Hierarchical attention networks for document classification》实现，因此首先简单讲解 HAN 论文中的模型架构。再 HAN 中，模型设计分层清晰，主要分为四个阶段，具体如下图演示，包括 word encoder,word attention,sentence encoder,sentence attention 四个阶段。显然就是针对不同的文本对象分别进行自己的 RNN 编码与自注意力实现。

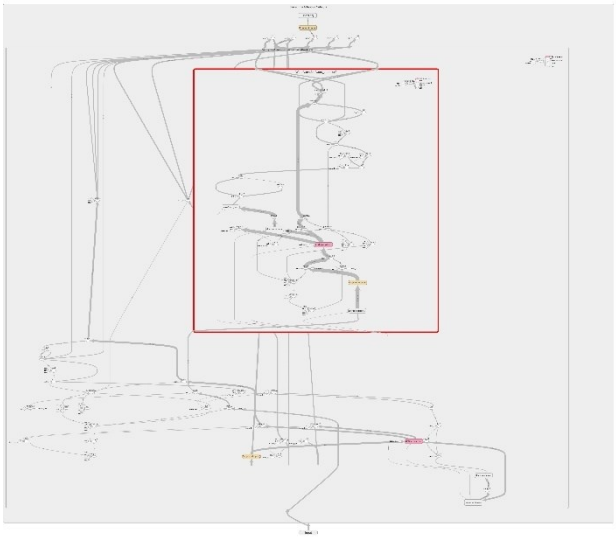


1.word encoder：对词汇进行编码，建立词向量。接着用双向 GRU 从单词的两个方向汇总信息来获取单词的注释，因此将上下文信息合并到句子向量中。
2.word attention：接着对句子向量使用 Attention 机制。
3.sentence encoder：与上面一样，根据句子向量，使用双向 GRU 构建文档向量。
4.sentence attention：对文档向量使用 Attention 机制。
5.softmax：常规的输出分类结果
其中潜在向量 h 主要是由双向 RNN 编码每一个 embedding 后的 word 张量<hleft,hright>拼接而成，注意力系数 α 由每一个潜在向量 h 与随机上下文初始化的 u 点乘计算得到，而每一个句子由不同的 word 的 α 加权和生成，

得到 s 潜在向量后再次执行 sentence 层的 encoder 与 attention，因此最后每一个评论文本 doc 的潜在向量 v 由 s 的 α 加权和得到，之后再走一层全连接生成 size 为 5 的预测 label 与 gt label 进行交叉熵计算得到 loss 反向传播优化模型中的参数，并使用 top1 计算准确率分析。

2.3 模型设计与实现

首先我们需要对文本进行词汇的划分与编码，生成每一个 word 对应的 vector 并 embedding 后得到 w 作为模型的输入。在本次实验中，首先执行 create_input_files.py 文件调用 utils.py 中封装好的工具函数首先对文本进行划分，之后对出现频率大于 5 的词汇 word 加入到 word_map 中并绑定为一 id，之后即可生成一个 word_map.json 文件查看所有需要编码的词汇，之后我们使用与训练好的 word2vec 模型对 word_map.json 中的每一个词汇进行编码生成每一个 word 对应的唯一张量 w 作为输入，这个模型可以保证输入词汇之间的编码后的张量 w 分布均匀且特征距离差异较大。编码后的数据文件会保存到 dataset 文件夹下的 pth 文件中，对应的编码模型保存在 word2vec_model 文件。之后我们在 main.py 中读入编码后的向量 w 文件作为输入训练我们参考 HAN 设计的文本-评分分类模型。我们设计并实现的模型图如下所示，对比 HAN 论文中的模型图证明实现架构相一致。

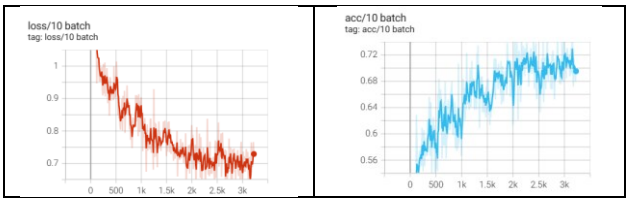


3 实验分析

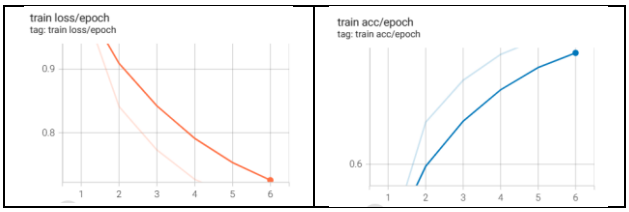
3.1 baseline 模型

我们选取 embedding 向量 size 为 200，rnn_layer_size 为 1，rnn_size 为 50，atten_size 为 100（注意 rnn_size 宽度一定为 atten_size 的 1/2 因为潜在向量最终是 left 与 right 拼接生成作为 attention 的输入），batch_size 为 256，学习率为 0.001，训练 6 个 epoch，每 epoch 后学习率衰减为原学习率的 0.5 作为 baseline 模型进行分析，以下是损失/准确率/学习率等展示图，具体日志请查看 logs 文件夹。

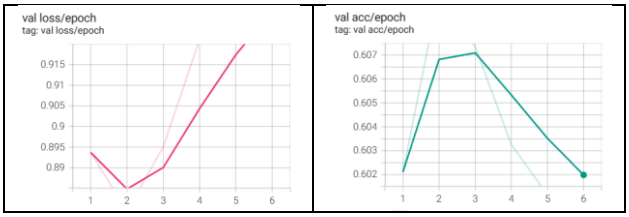
每 10 Batch 损失/准确率



Train 损失/准确率



Val 损失/准确率

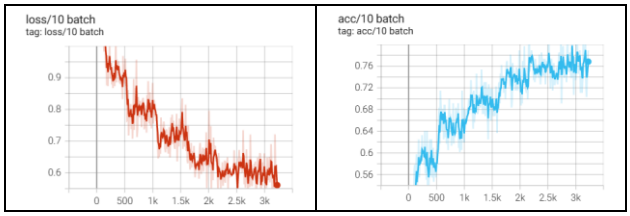


观察上图发现在 6 个 epoch 训练过程中模型在训练集上的损失不断下降并且准确率缓慢上升最终达到 75%，但是在第 2 epoch 后模型在 val 数据集上损失开始上升且准确率下降，说明模型已经在第 2 epoch 后收敛达到最佳性能，之后是由于过拟合导致其在验证集上的表现开始下降，因此我们应该选取第 2 epoch 的模型参数保存并在测试集上进行测试最终得到测试准确率为 60.2%。

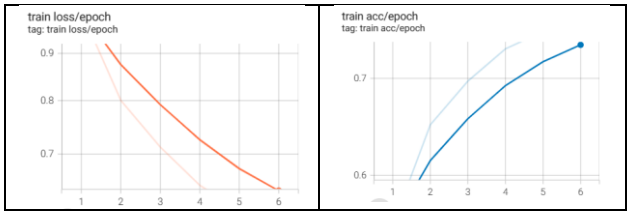
3.2 w/o dropout

之后我们在 baseline 基础上取消 dropout 操作观察模型的变化，损失与准确率图像如下所示：

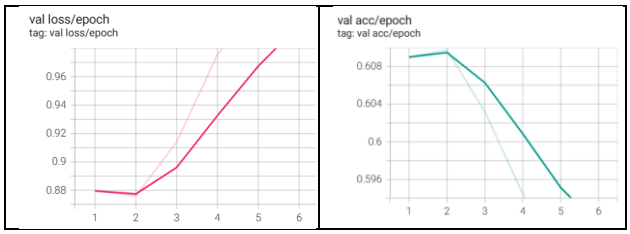
每 10 Batch 损失/准确率



Train 损失/准确率



Val 损失/准确率

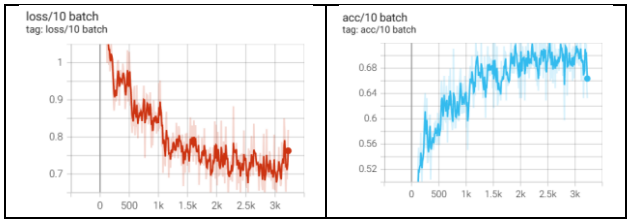


发现 dropout 操作对模型并无太大影响，分析原因是因为此次模型使用了 attention 机制而非深度 RNN 网络因此模型参数两并不大，出现梯度消失的情况概率较小，因此不加入 dropout 操作模型仍可迅速收敛并且最终在测试集的准确率表示也高达 60.0%。

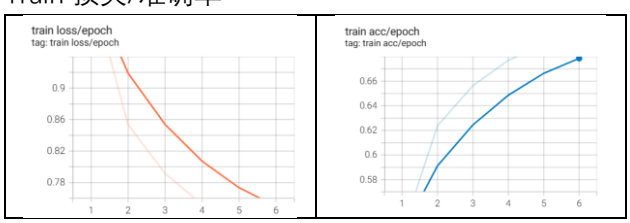
3.3 deep RNN

在 baseline 中我们对词汇 w 与语句 s 的 RNN encoder 均仅使用了一层双向 RNN，因此推测可能提升双向 RNN 深度可能有助于进一步加大潜在编码向量的差异距离从而提升模型的性能，因此这里我们尝试将 rnn_layer_size 翻倍。

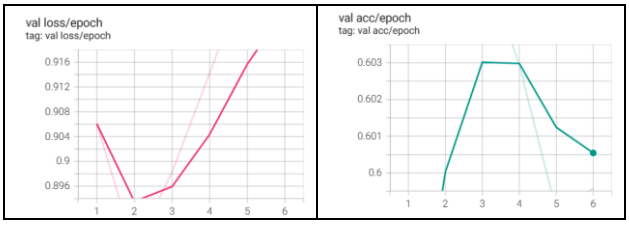
每 10 Batch 损失/准确率



Train 损失/准确率



Val 损失/准确率

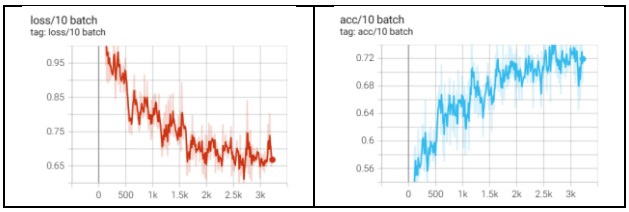


对比发现加深双向 RNN 编码层数并没有显著提升模型的性能，反而造成模型在测试集上的最终 Top1 准确率略微降低为 60.1%。略微浮动属于正常现象，因此表明在 baseline 模型中 encoder 层仅为一层已可以实现不同词汇与语句的有效编码。

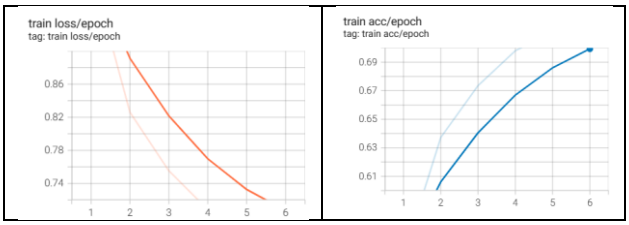
3.4 largesize latent tensor

为了尝试进一步提升模型的性能，我们还尝试加宽潜在编码张量的 size 宽度，尝试在 baseline 基础上将 atten_size,rnn_size 等翻倍，以下是损失/准确率图像：

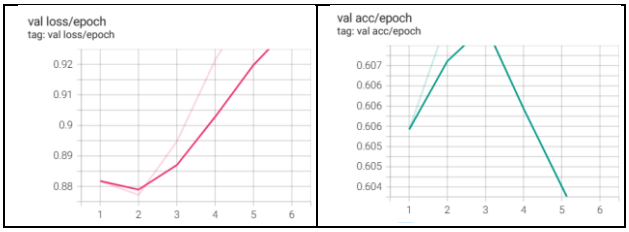
每 10 Batch 损失/准确率



Train 损失/准确率



Val 损失/准确率



发现在增大编码后潜在张量的宽度后并没有明显提升模型的分类性能。结合以上所有对比实验我们认为模型已经在 baseline 基础上达到性能饱和，如果想进一步提升模型的性能可能需要在数据方面进行优化例如增加更多训练数据，增加训练数据的分布多样性以保证模型学习到更多有效分类信息。

3.5 注意力机制可视化

我们额外提供一个 classify.py 可以可视化训练后的模型对输入的语句的词汇注意力系数，如下图所示形容词部分字体更大表示模型确实有效的将更多的注意力放在了带有主管性质的形容词上：

测试文本："*Artichoke Gratine: The corn chips were amazing, lightly salted and crisp. The dip was a bit too garlicky and runny for my liking. Ate a few bites of this but could not see myself eating the entire thing solo.*Spicy Buffalo ""Wings"": first things first... do not let looks dismay you... true it looks gross but they taste legit! The flavor of the buffalo sauce was perfect, although could have been spicier. And the cucumber ranch dipping sauce was perfectly creamy and lightly flavored as to not overpower the ""wings"". This dish is a must try!!*Vegan Chili Fries: the fries are thin cut and tasty. The chili sauce was good, at first, but I quickly got sick of the flavor. This could be because I was never a huge chili fan even back when I ate meat. Hmm, I think you are better off ordering the thyme fries.

*Crab Puffs: Perfectly crisp with a delicious creamy filling. Another must try!"

4. Requirements.txt

```
absl-py==2.0.0
autopep8==2.0.4
cachetools==5.3.1
certifi==2023.7.22
charset-normalizer==3.3.0
colorama==0.4.6
contourpy==1.1.1
cycler==0.12.1
fonttools==4.43.1
google-auth==2.23.3
google-auth-oauthlib==1.1.0
grpcio==1.59.0
idna==3.4
importlib-metadata==6.8.0
importlib-resources==6.1.0
kiwisolver==1.4.5
Markdown==3.5
MarkupSafe==2.1.3
matplotlib==3.8.0
netron==7.2.5
numpy==1.26.1
oauthlib==3.2.2
```

```
packaging==23.2
Pillow==10.1.0
protobuf==4.23.4
pyasn1==0.5.0
pyasn1-modules==0.3.0
pycodestyle==2.11.1
pyparsing==3.1.1
python-dateutil==2.8.2
requests==2.31.0
requests-oauthlib==1.3.1
rsa==4.9
six==1.16.0
tensorboard==2.15.0
tensorboard-data-server==0.7.1
tomli==2.0.1
tqdm==4.66.1
typing_extensions==4.8.0
urllib3==2.0.7
Werkzeug==3.0.0
zipp==3.17.0
```