人工智能导论

基于深度学习的情感分析

实验报告

聂鼎宜

计96

2019011346

---

0 前置说明

本报告所有实验均在下述环境中完成：

> MacOS Big Sur
>
> Python 3.6
>
> Pytorch 1.7

本报告所有实验基于 v2 数据集完成。

对数据集进行的预处理包括：

1. 忽略空白语句；
2. 忽略似乎是数据集生成阶段混入其中的一些无意义数据 (如用 [] 符号包裹的那些句子)；
3. 忽略所有阿拉伯数字；
4. 忽略所有标点符号，这可能意味着一些英语中实际不能被拆分的词 (didn't, alice's, seven-eleven) 被拆分 (didn +

t, alice + s, seven + eleven);
5. 仅使用小写字母;
6. padding 字符统一定义为 <pad>;
7. 未知字符统一定义为 <unk>;

训练前均会对数据集进行上述预处理，然后建立词汇表 (vocabulary，即词对下标的一一映射，一般 <pad> 的下标规定为 0)。

1 三种模型的结构与流程分析

1.0 一些共性

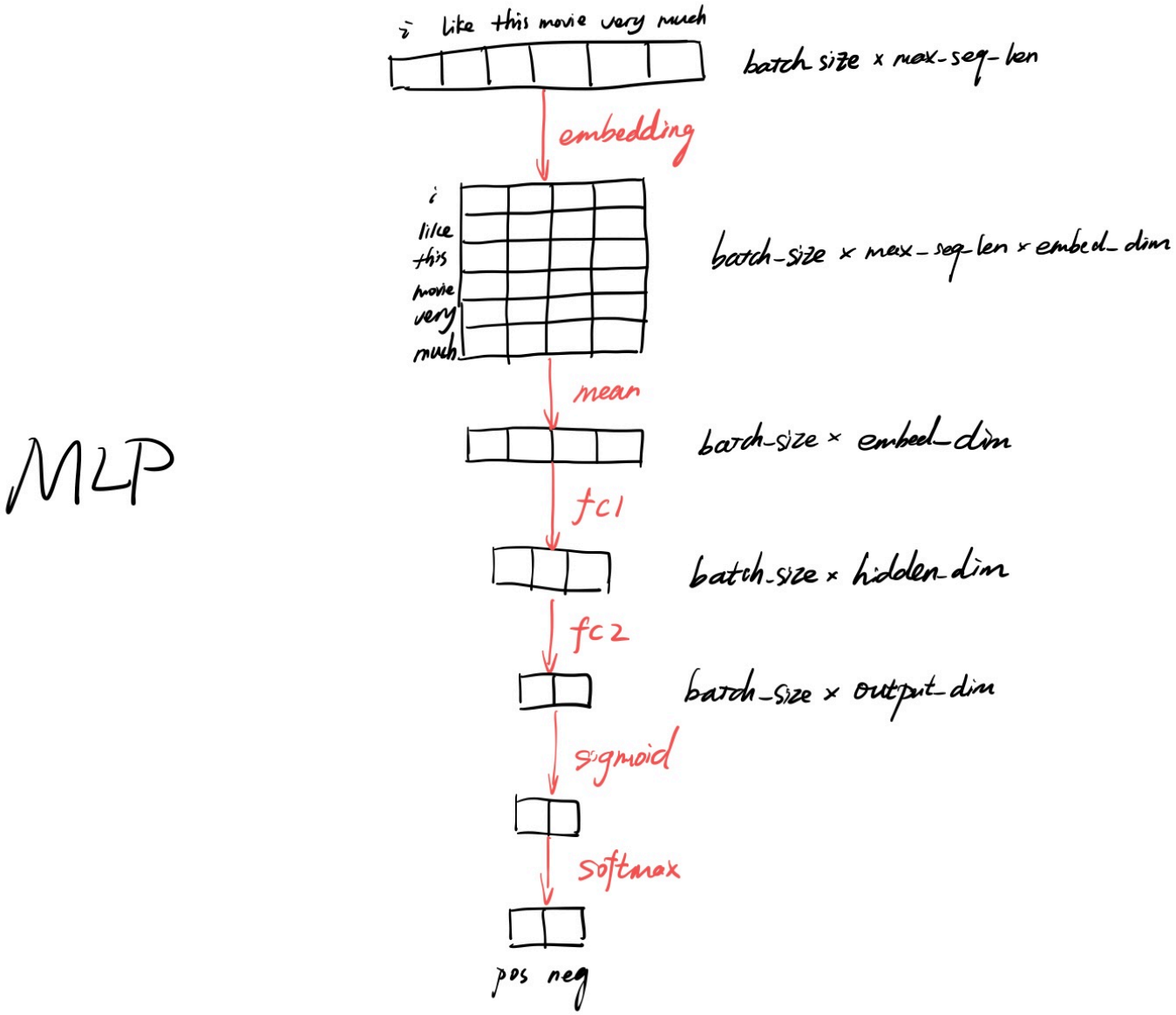
由于目标任务是一个多分类任务，查阅文献得知多分类任务神经网络的全连接层后往往接激活层 (通常用 sigmoid) + softmax 层的组合，这样输出的结果可以直接作为概率使用，因此本实验的所有模型均这样设计。考虑到模型的可应用性，在处理分类目标种数的时候进行了 +1 处理，即除了训练数据集里出现过的 label 种类，模型还可能输出一个 **other 种类**，代表它无法确定的情感分析结果。

另外，出于实现的简便易行性考虑，每个模型的第一层均为**随机初始化**的 embedding 层，来完成词的 index 到词向量 (word vector) 的映射。下述内容中除非特别说明，embedding 层的输出维数即词向量的维数统一为 400。对于一个 minibatch 中的数据，它们在通过 embedding 层后应该是一个 batch_size * max_seq_length * embed_dim 的 tensor。

在训练时，使用的优化器是 Adam (自适应实在是太省心了)，损失函数是 BCE，同时有采用梯度裁剪 (bp 前将过于大的梯度值限制在一个合适水平)。

1.1 MLP

MLP 作为一个对照用 baseline，实现比较朴素，它仅含一层隐含层，而且没有引入 dropout。结构和流程可以表示如下:

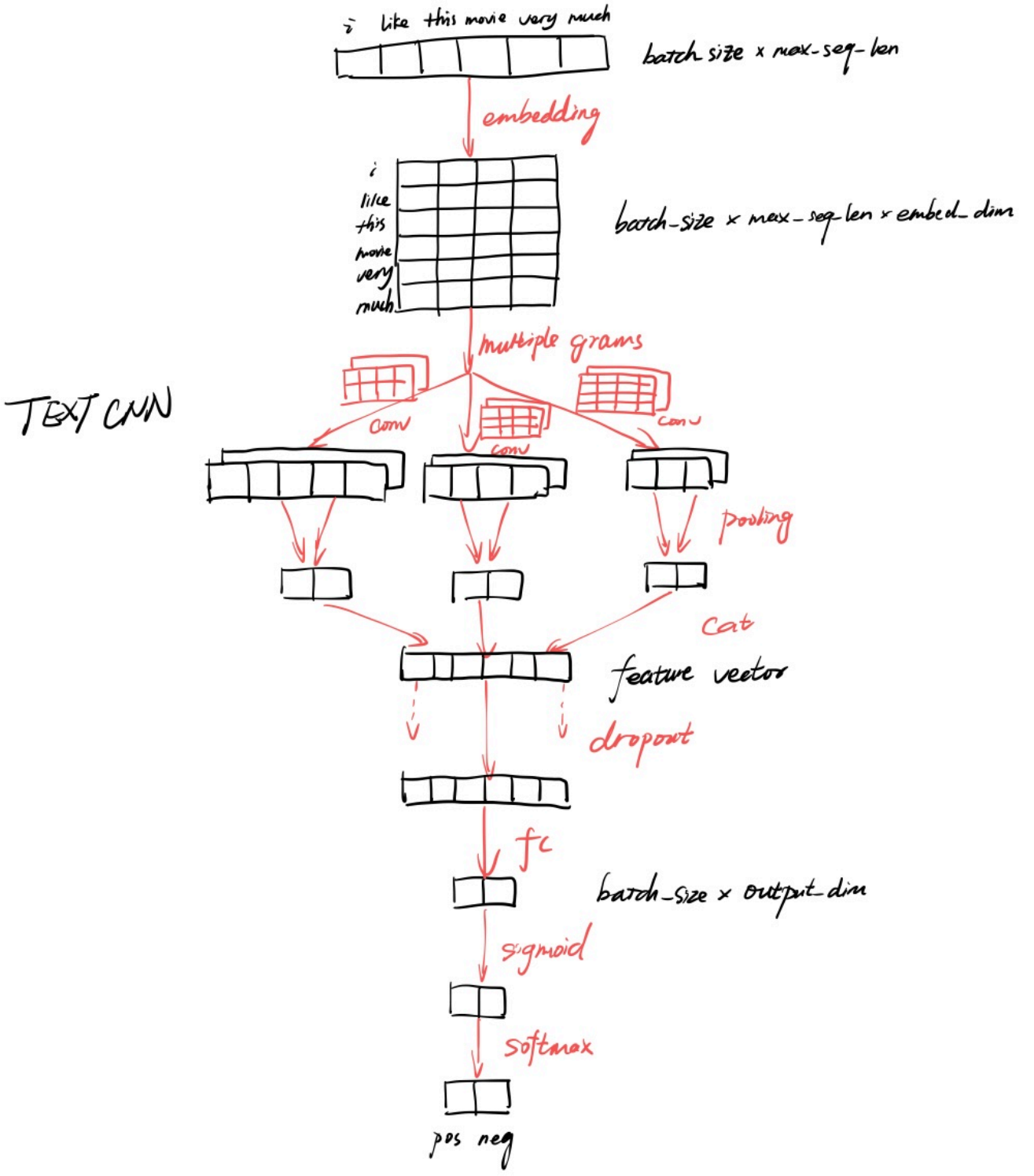


几点解释：

- 列向量出于美观考虑也画成行向量的形状，实际以模型的真实结构为准，下同。
- embedding 层的输出已在 1.0 中进行解释，上图给出的是 batch_size = 1 即一次性只输入一个句子的示意图。注意，当 batch_size > 1 时，为了让 minibatch 内的句子可以 stack，长度不足的句子将被 pad 至与 minimatch 内最长句子等长。这意味着一些句子矩阵的末几行甚至十几行会是全 0，下同。
- mean 这一步其实是一种简单粗暴的降维：将一个句子的所有词向量取平均“压扁”为一个词向量，输入后续全连接层。可以理解为直接用句子各词向量在词向量空间中的重心作为句子的特征向量输入全连接层。
- 上图其实是一个二分类的示意，实际任务并不是二分类而是多分类，但在代码实现上输出维数其实是可调的。

1.2 Text CNN

Text CNN 的基本思想是把一个个句子转为词向量 stack 成的二维矩阵后，将它视为与图像地位等同的一种结构，在卷积与卷积核的参与下完成特征提取。实验设计的 CNN 结构和流程如下：

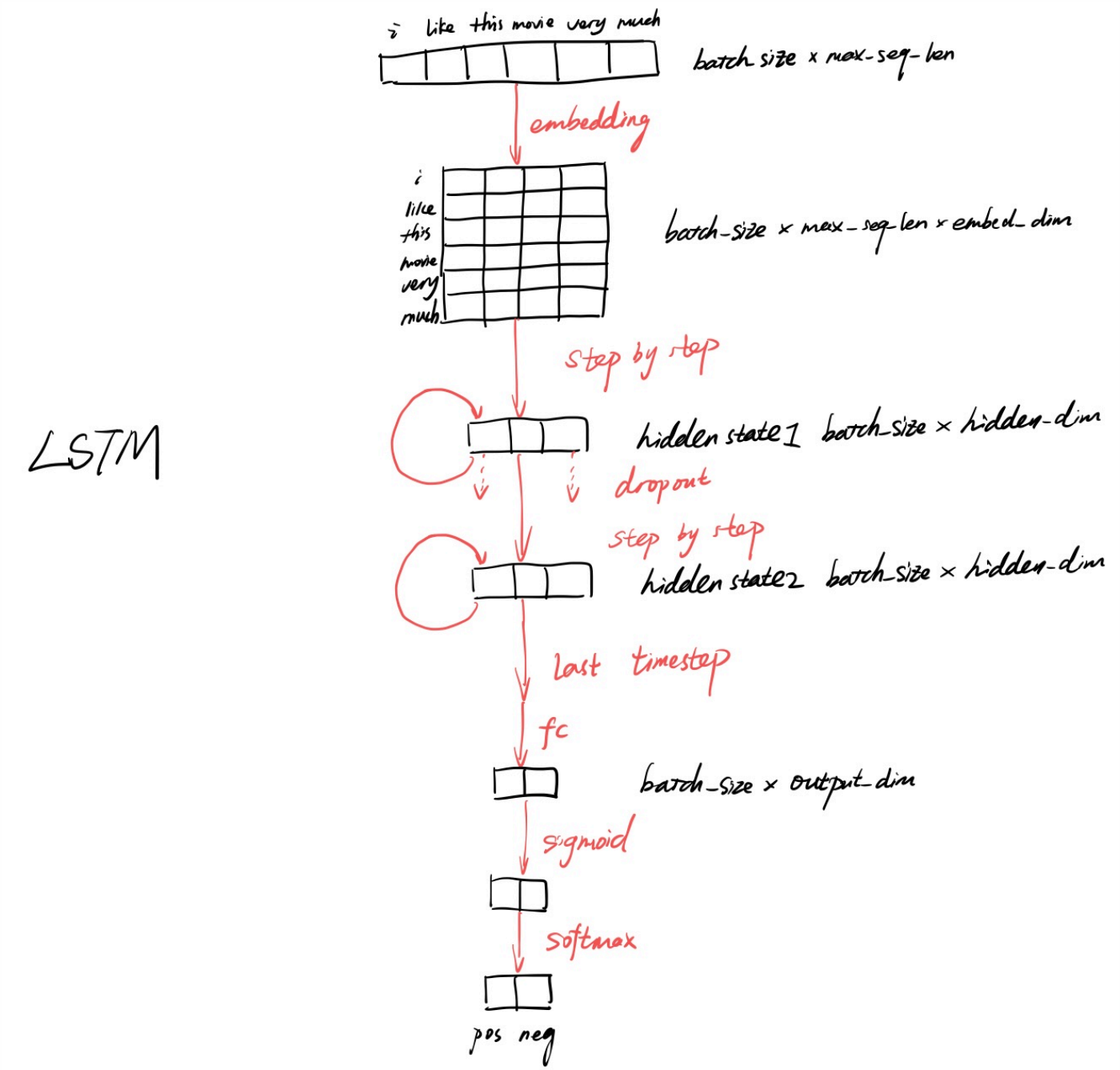


几点追加解释：

- 图中给出的是 2-gram、3-gram、4-gram 对应的卷积核数量均为 2。
- 相关文献对卷积 stride 似乎没有什么深入的研究，这里全部置 1，最多地保留特征。
- pooling 层采用朴素的最大池化。

1.3 LSTM (RNN)

LSTM 作为一种循环神经网络，采用多个可循环的神经元串行的结构设计，适合于读入时序性 (timestep) 的输入。它的结构与流程示意如下：



几点追加解释：

- lstm 层的输入是一步步 (step by step) 进行的，甚至可以是双向同时步进的，每次读入一个 / 首位各一个词向量。lstm 层可以有多个 (图中为 2 个)，它们之间的传递也是一步步进行的。层间可以应用 dropout。
- 注意，对于一个 minibatch 同时输入的情景，对 batch 内的各句子没有一个统一的 last timestep，各句子的最后一个词向量输入 lstm 层后就可以停止循环了，否则后续的 padding 全 0 的向量会影响 lstm 层的输出（这点从 lstm 的公式中可以轻易推出）。
- 图中展示的是 lstm 的隐状态维数为 3 的情形。

2 三种模型的效果评估

2.0 一些说明

在实验所用的数据集上，模型被初始化为一个 7 分类（实际是 8）分类任务的分类器。对于 random guess，准确率应该在 14%。

由于每次训练开始时 embedding 层的参数是随机初始化的，模型的效果可能有所波动。

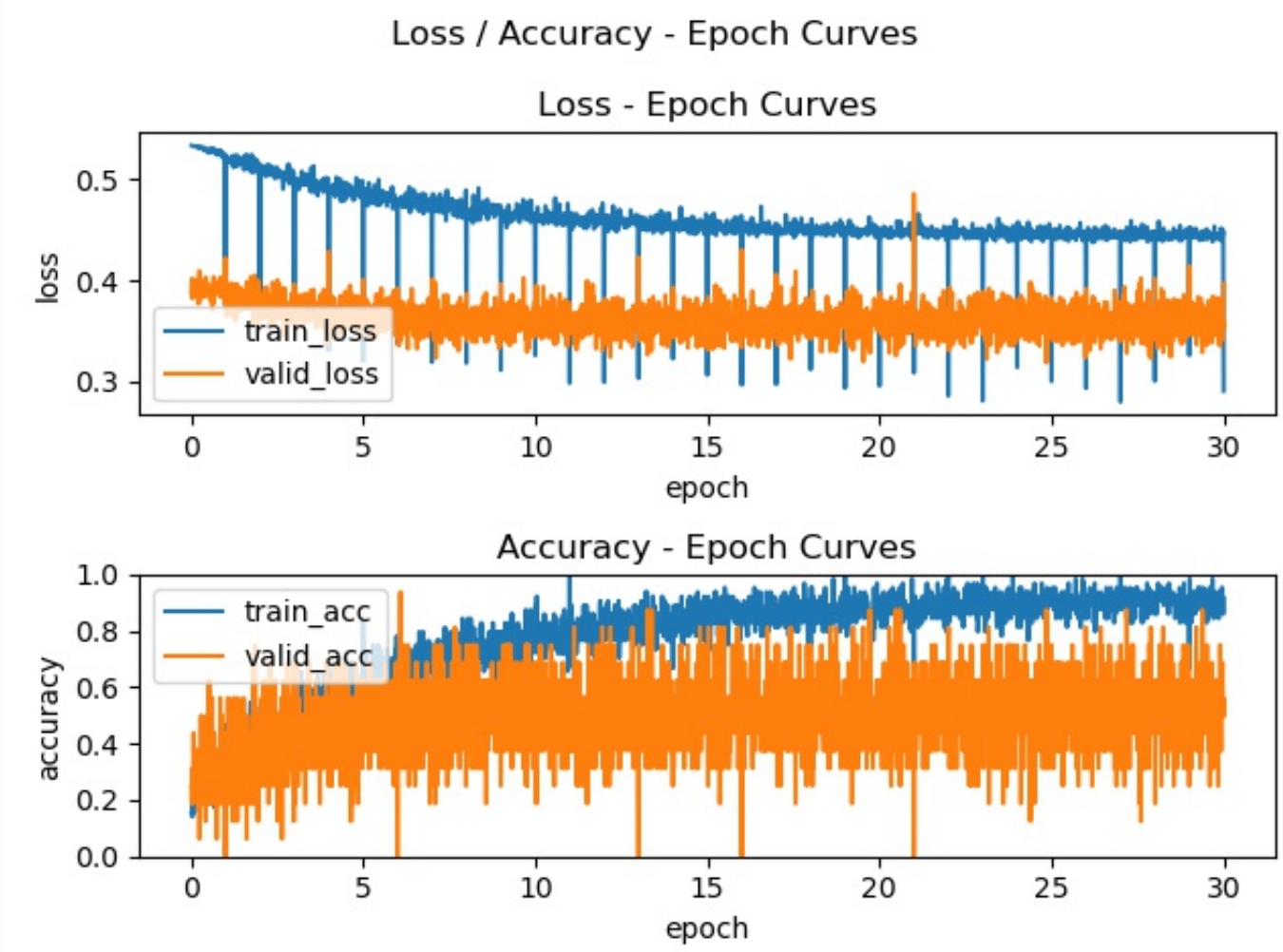
下文的 F1 score 均使用 sklearn 库函数计算给出。

2.1 MLP

设置隐层维数为 256，初始化 MLP，经过大约 30 个 epoch 的训练，各项指标可达：

Total accuracy	F1 score (macro average)	F1 score (micro average)
约 53%	约 0.48	约 0.53

其中一次实验的损失和准确率曲线图：



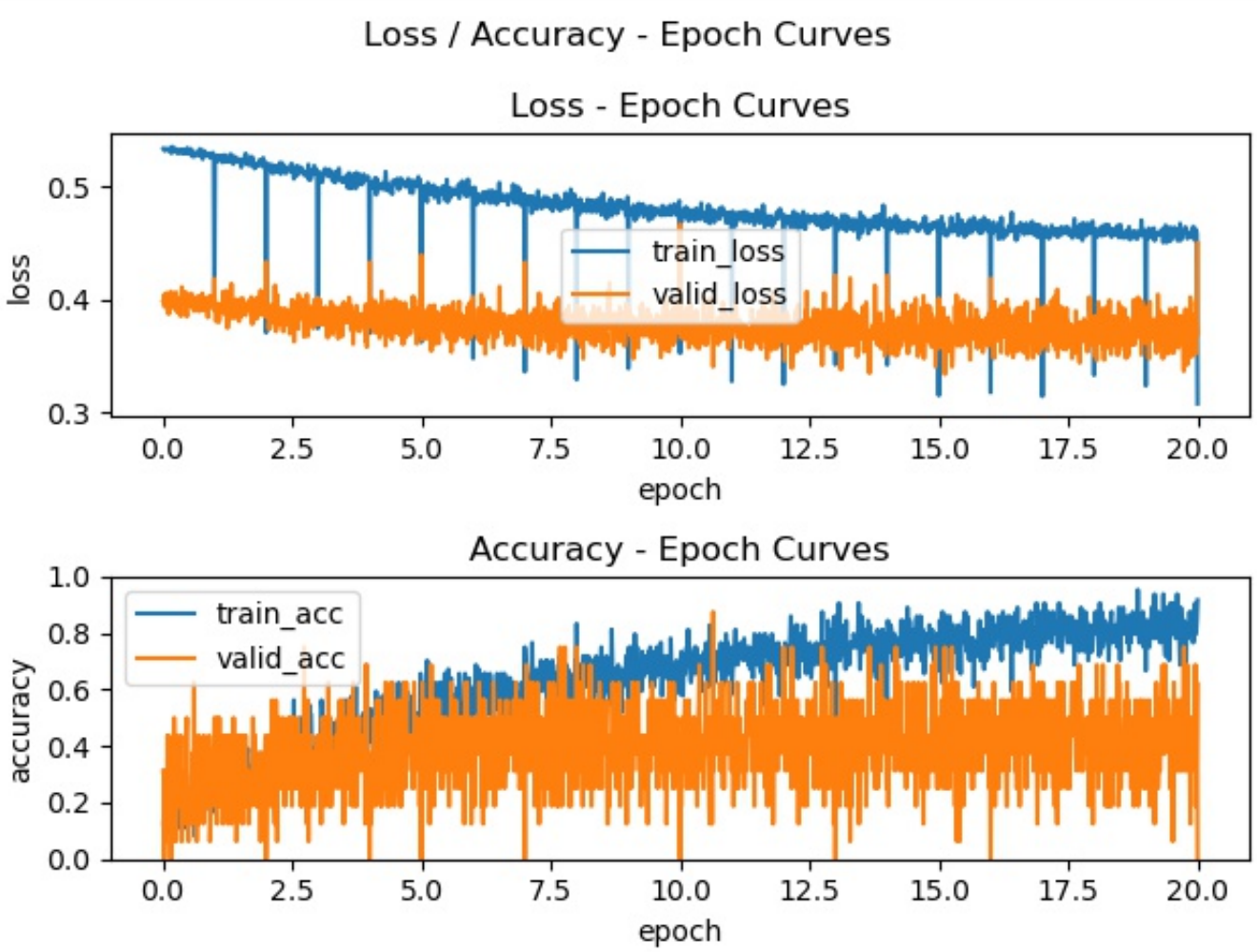
效果意外的很好？

> 曲线中有一些有规律的毛刺，这是因为这些曲线是每一个 iteration 采样一个点而不是每个 epoch，这会导致每个 epoch 的最后一个 iteration 因为 size 略小而偏移较多，可以忽视。

2.2 Text CNN

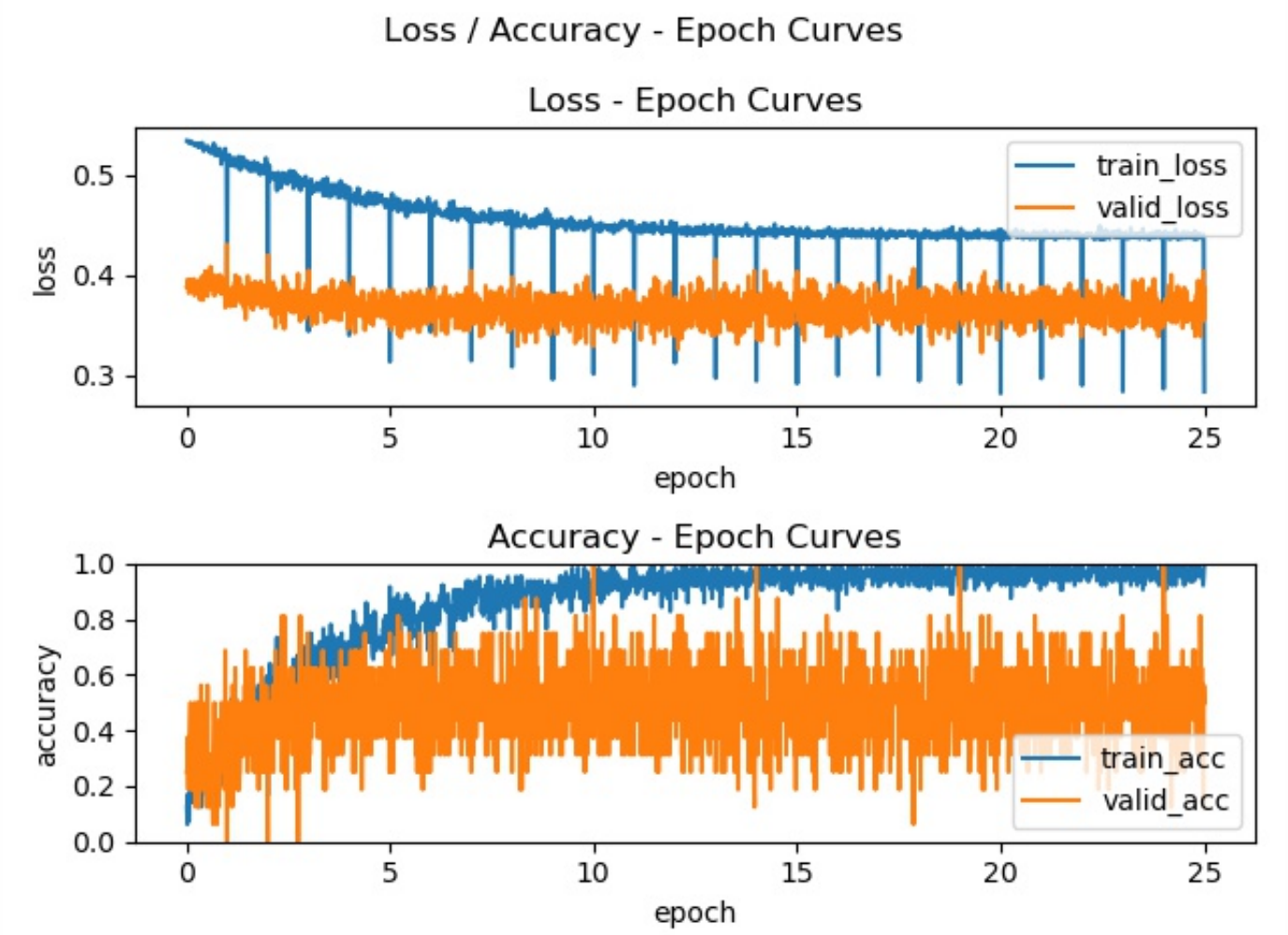
通过之前的拼音输入法作业了解到，对于 n-gram 的MHH 模型，n = 4 以上的表现基本不再上升，于是训练 Text CNN 一开始采用 kernel size 为 2、3、4 的卷积核各 2 个的配置，并暂时不开启 dropout。训练 20 个 epoch 基本收敛，结果如下：

Total accuracy	F1 score (macro average)	F1 score (micro average)
约 43%	约 0.38	约 0.43



进一步增加 kernel 数量至每种各 10 个，训练 25 个 epoch 左右，结果如下：

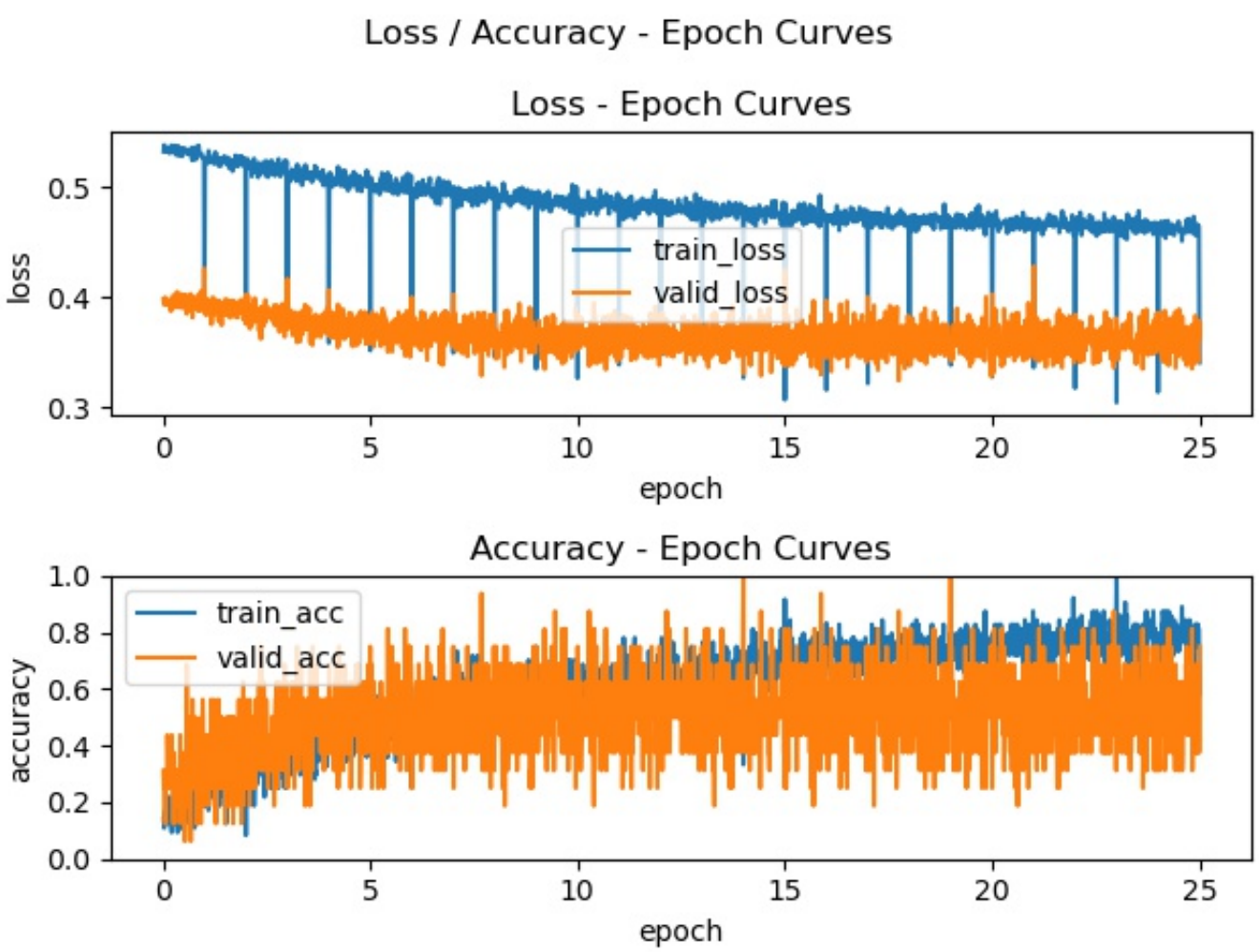
Total accuracy	F1 score (macro average)	F1 score (micro average)
约 48%	约 0.45	约 0.48



可以看到，training accuracy 远高于 validation accuracy，且后者收敛后前者还在增长，这说明存在严重的过拟合。

如果开启 dropout 呢？设置 dropout 层概率为 0.5，训练 25 个 epoch，结果如下：

Total accuracy	F1 score (macro average)	F1 score (micro average)
约 53%	约 0.48	约 0.53

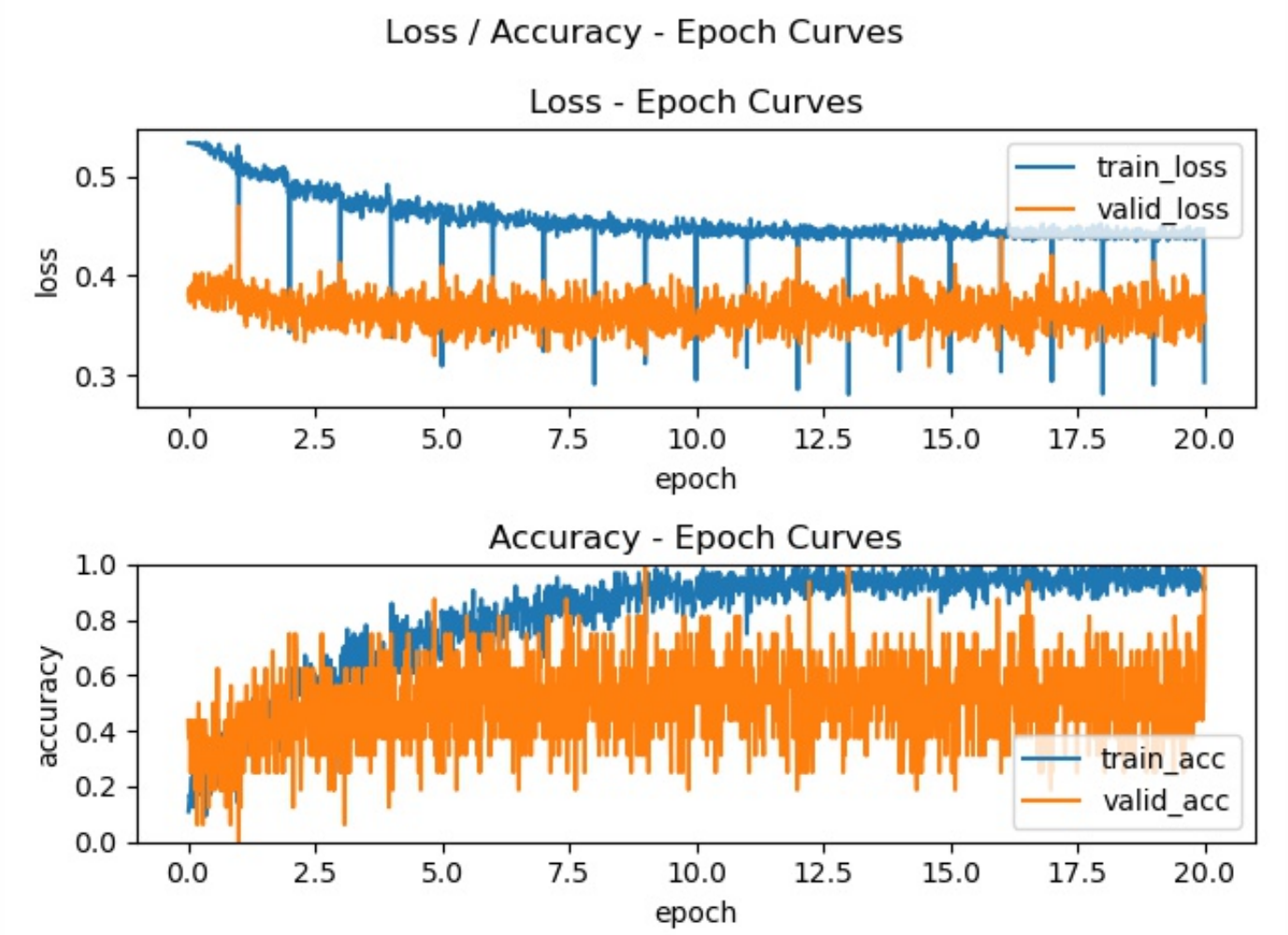


过拟合现象得到缓解，性能与 baseline 持平。

2.3 LSTM (RNN)

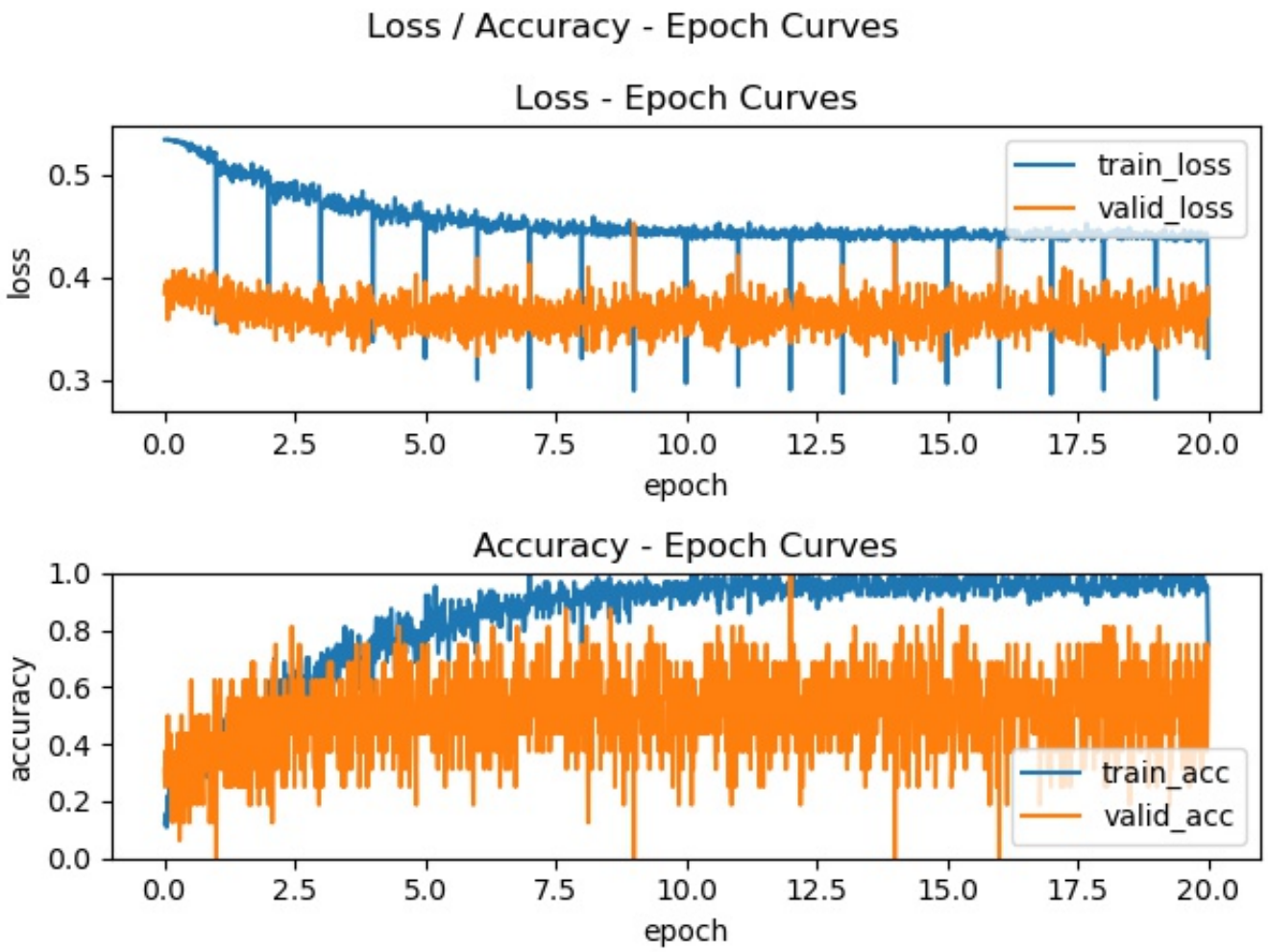
LSTM 的训练是最花时间的。首先进行了一组 2 lstm 层、不采用双向、无 dropout 的实验。隐状态维数 256，结果如下：

Total accuracy	F1 score (macro average)	F1 score (micro average)
约 52%	约 0.46	约 0.52



同样看出有过拟合倾向。设置 dropout = 0.3，同时开启双向模式，再次训练：

Total accuracy	F1 score (macro average)	F1 score (micro average)
约 54%	约 0.48	约 0.54



3 超参数调节

代码实现时和做实验时都考虑了各种超参数，包括学习率、训练周期、神经元层数、各层维数、dropout 概率等等。之前也做过几个深度学习和神经网络的项目。一个直观的感受是，**参数不是越多越好**。

模型的 capacity 应该与目标问题的 complexity 匹配，才能获得最好的效果。其实稍微查看一下实验所用的数据集可以发现，内容的质量并不是很高，包括许多意义不明的句子和不匹配的标签，甚至有奇奇怪怪的拼写错误。在这种情况下，模型的表达力过强反而会导致难以预料的过拟合或其他负面效果。因此，选取适当的超参数成为一个考虑的重心。

结合第 2 节的结果以及本人做了但没有完整记录的其他实验，可以得出以下几点总结规律:

- embedding 层维数不需要太大，较小的 embedding 空间也有很好的表达力，且能大大减少参数数量，加速训练，减少过拟合。
- dropout 是降低过拟合的有效手段。查阅文献可以了解到，dropout 的原理在于，随机丢弃一些层间权值效果约等于同时训练多个不同模型，综合考虑它们的输出结果进行决策，类似于一个多专家系统。一些具有过拟合倾向的神经元的意见很可能被其他更理智的神经元的意见吞没，从而优化测试表现。
- 在 CNN 中，卷积核的数量是一个关键要素。卷积核的出现大大减少了参数数量，参数共享从机制上要求了卷积核数量不能过少，过少的卷积核往往直接意味着过少的特征被捕获，模型的效果自然不会太好。
- 虽然 LSTM 相较于普通 RNN 已经一定程度上解决了“记忆”和“遗忘”的问题，但双向 LSTM 仍较单向的有更好的表现，这是很容易理解的。拿情感分类举例的话，一些句子的情感很可能集中在句首或是句尾，比如英语里经常出现的 "unfortunately..." 后面加一大段陈述句的句子，对于这样的句子，反向的过程比正向捕捉的情感可能更

加强烈。综合考虑两者的意见总是好的。

还有许多值得探索的子问题无法尽述于此，相信我会在之后的学习中继续总结反思。

4 三种模型的横向对比

上一节的分析中提到，受限于数据集质量，在其上解决多分类问题的上限恐怕不会太高。结果确实如此，CNN 和 LSTM 都不比 MLP 高出多少。

LSTM 的表现最稳定，毕竟 RNN 就是一类专长于处理时序输入的模型，而自然语言处理恰好契合它的应用场景。

CNN 在参数不当的情况下表现最差，可能是因为语言的特征比图像的特征更加抽象复杂，少数的卷积核无法完全表达，并且这种模型受到 padding 的影响较大（相当于大部分输入的后半部分都是纯白的无意义图像）。

5 问题思考

1. 如何控制实验训练的停止时间? 简要陈述你的实现方式，并试分析固定迭代次数与 early stopping 等方法的优缺点。

   我基本采用的是 early stopping。当 loss 不再降低（尤其是我用的优化器还是广受好评的 adam 时），继续训练往往只会加剧过拟合。及时观察训练曲线图、及时停止训练是效率最高的训练策略。对于一些后续可能会部署应用的模型，early stopping 能保证他们的泛用性最高、稳定性最好。但固定迭代次数也有它的优点，可以清楚地比较不同模型、不同参数给训练带来的影响。

2. 过拟合和欠拟合是深度学习常见的问题，有什么方法可以解决上述问题。

   过拟合：dropout、early stopping、减少模型参数数量等。

   欠拟合：学习率过高（导致无法收敛在极值点）、模型参数过少、训练周期过短都可能导致欠拟合问题。解决方法就是反其道而行之。

3. 试分析梯度消失和梯度爆炸产生的原因，以及对应的解决方式。

   原因：激活函数使用 sigmoid 时，其特性决定导数很可能远小于 1，连乘后就会导致反向传播过程中梯度消失。参数初始化不合适（如过小/过大）也会引起梯度消失/梯度爆炸。

   解决：激活函数使用大多数情况下更优秀的 ReLU、引入正则项（如 l2 范数）、梯度裁剪等。本实验中使用 Adam 优化器也一定程度上避免了这些坑。

4. 试分析 CNN，RNN，全连接神经网络(MLP)三者的优缺点与各自适用的场景。

   诚如第 4 节所言，

   CNN：适合图像处理，因为图像的本质就是二维矩阵，具体应用如手写识别、人脸识别等。缺点在于设计门槛高、结构复杂。

   RNN：适合处理时序输入，比如翻译、语音识别、计算机作曲等。缺点在于无法并行多线程。

   MLP：适合解决一些非线性的朴素问题，比如函数逼近与预测分析、模式识别、数据压缩等。缺点在于无法识别更高维度上的特征。

6 心得体会

之前我做过一些神经网络的项目，也包括基于 RNN 的情感分析。但用 MLP 和 CNN 进行自然语言处理还是第一次。通过直观的横向比较加深了对三种模型各自优劣的理解，也进一步学习到了一些调参技巧、训练技巧和优化方法。对深度学习框架的进一步熟悉，让我看到了今后利用深度学习技术解决自己感兴趣的问题和需求的更大可能。