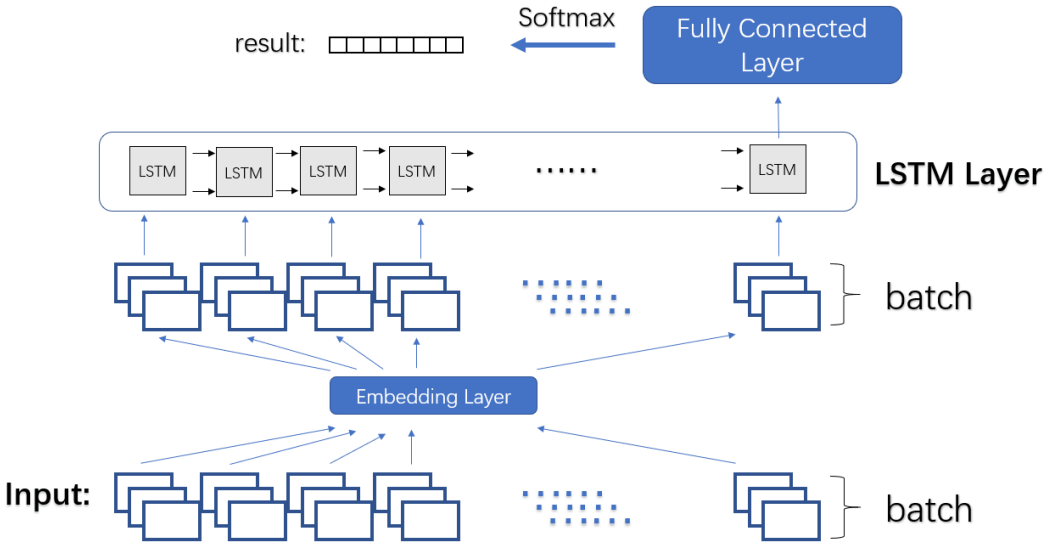
情感分析任务 实验报告

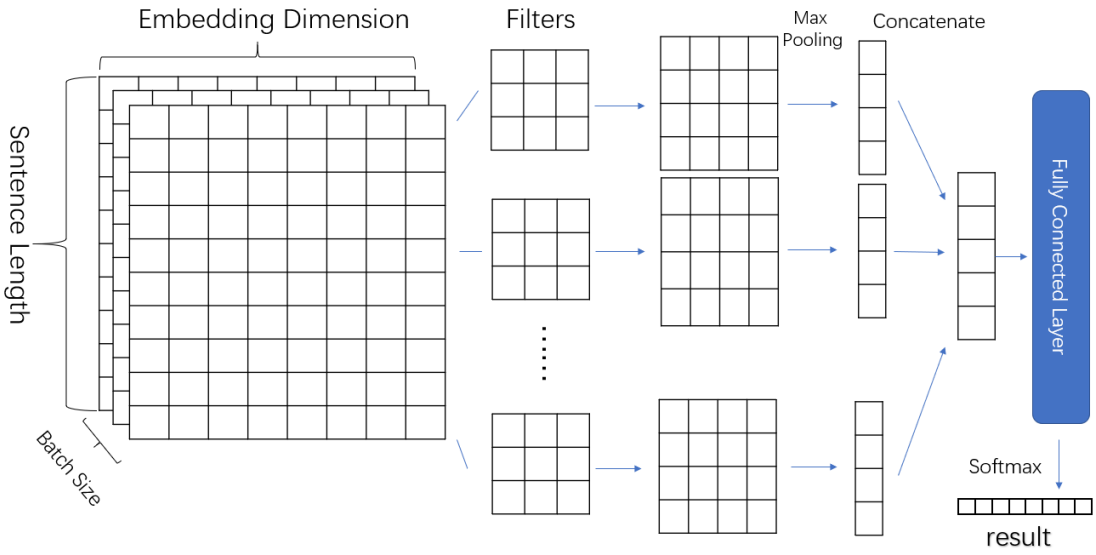
一、模型结构图及流程分析

预处理：将每篇文章取前 120 词，不足的补<unk>（词向量为零向量）
RNN：
结构如下



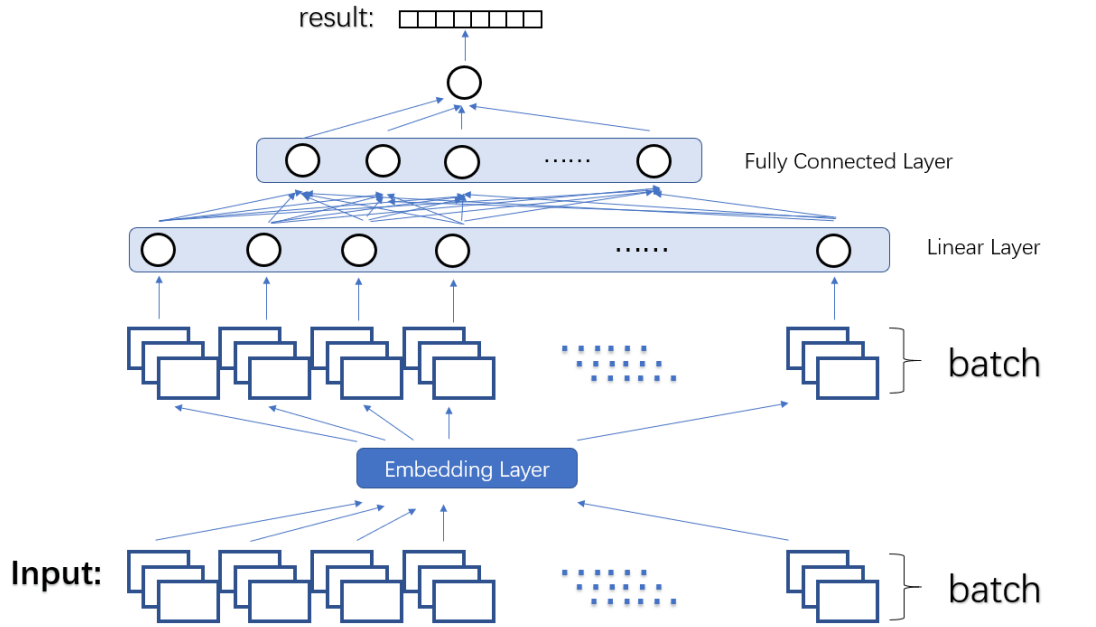
将每批文章的词通过词嵌入层变为以词向量形式表示，作为输入进入 LSTM 层；将 LSTM 层最后一个单元得到的h_n作为输入进入全连接层进行分类，输出的 8 维向量经过 softmax 后得到每个类的预测概率。

CNN:
结构如下：



将每个词对应的词向量作为行，构建文章对应的矩阵。将六个卷积核得到的矩阵最大池化得到六个向量，拼接成一个向量后进入全连接层进行分类，输出 8 维向量经过 softmax 后得到每个类的预测概率。

baseline（MLP）：
结构如下



同上处理输入文章后，通过一层线性层将每篇文章的词向量组成的矩阵映射为维度等于文章长度的向量；再通过一层全连接层，最后经过线性神经元输出 8 维向量，softmax 后即为每类的预测概率。

二、实验效果

参数设置：200epoch，batch_size=10, learning rate=0.001, dropout=0.2, optimizer=SGD, loss=CrossEntropyLoss。

准确率：
RNN：correct: 1141/2227 accuracy: 0.512
CNN：correct: 1182/2227 accuracy: 0.530

F-score
采用 macro f1-score；未被成功预测的类 f1-score 视为 0.
RNN：0.209（对情感“同情”、“温馨”未有成功预测）
CNN：0.280（对情感“温馨”未有成功预测）

相关系数：
RNN：所有文章的预测结果和实际标签的相关系数平均值为 0.54
CNN：所有文章的预测结果和实际标签的相关系数平均值为 0.53

参数设置：500epoch, batch_size=10, learning rate=0.001, dropout=0.2, optimizer=SGD, loss=MSELoss
RNN：
准确率：correct: 1222/2227 accuracy 0.549
f-score：0.160
平均相关系数：0.55

因时间原因，未对 CNN 进行 500epoch 训练。

三、比较参数效果
对 batch size, dropout rate 及所用优化器算法进行比较：
迭代次数 20 次
将 train 中序号为 3 的倍数的文章作为验证集。
LSTM：

	batch size=100	150	200
dropout=0.2 optimizer=SGD	(0.162, 0.062, 0.06)	(0.446, 0.084, 0.44)	(0.085, 0.038, 0.17)
dropout=0.2 optimizer=Adam	(0.146, 0.066, 0.13)	(0.465, 0.089, 0.41)	(0.449, 0.106, 0.32)
dropout=0.4 optimizer=SGD	(0.400, 0.090, 0.23)	(0.259, 0.090, 0.19)	(0.054, 0.041, 0.01)
dropout=0.4 optimizer=Adam	(0.321, 0.085, 0.27)	(0.473, 0.094, 0.40)	(0.187, 0.060. 0.20)

CNN:

	batch size=100	150	200
dropout=0.2 optimizer=SGD	(0.470, 0.080, 0.42)	(0.472, 0.096, 0.43)	(0.460, 0.112, 0.36)
dropout=0.2 optimizer=Adam	(0.075, 0.053, -0.14)	(0.053, 0.031, -0.05)	(0.077, 0.050, -0.14)

dropout=0.4 optimizer=SGD	(0.470, 0.080, 0.44)	(0.476, 0.088, 0.43)	(0.367. 0.096, 0.25)
dropout=0.4 optimizer=Adam	(0.084, 0.056, -0.18)	(0.146, 0.067, 0.04)	(0.179, 0.095, 0.20)

可见，两种模型均在 dropout=0.4, batch size=150 时取得最高正确率；在 dropout=0.2, batch size=200 时取得最高 f-score（因为某些类型标签显著多，这基本代表能够成功预测更多类的情感标签）。对于 LSTM 模型，优化器算法应选用 Adam；而对于 CNN，应选择 SGD。

四、比较 baseline 与 CNN、RNN 效果差异：

baseline
参数设置：epoch=100, batch_size=10, dropout=0.2, learning rate=0.001, optimizer=Adam, loss=CrossEntropyLoss。
效果：
f_score: 0.130（对情感“同情”、“无聊”、“难过”、“新奇”、“温馨”未有成功预测）
correct: 1044/2227 accuracy: 0.469
average correlation rate: 0.51
平均每个 epoch 用时：63 sec

与 RNN、CNN 比较：在 f_score、正确率及平均相关系数上效果都略差；每次迭代时间略长，但总训练时间较短；对许多次数较少的情感标签无法成功预测。

五、问题思考

1.训练应在模型收敛时停止。
我的做法：固定迭代次数（RNN、CNN 迭代 200 次，MLP 迭代 100 次）
固定迭代次数
优点：便于操作。
缺点：可能造成过拟合，模型可能尚未收敛，可能进行了不必要的训练。

通过验证集调整：
优点：当验证集和测试集分布接近，终止条件设置得当时，能使训练至模型收敛后立即停止。
缺点：当验证集和测试集中数据不近似为独立同分布时，通过验证集进行调整没有意义；不易判断训练终止的条件。

2.实验参数使用均匀分布初始化（服从分布$U(-\sqrt{k},\sqrt{k})$，其中 k 为 1/hidden_size）

任何情况下参数均值都应初始化为零；
高斯分布初始化适用于使用 ReLU 作为激活函数的网络；
正交初始化适用于 LSTM 层的参数。

3.防止过拟合的方法一般有：
dropout：
在每次前向输出时，以一定概率随机删除线性层中的神经元。具体做法为以一定概率将线性层输出置为零。

提前终止：迭代过程中，当验证集的准确率不再上升时，即提前终止训练。具体做法为若连续 n 次迭代准确率均未超过最高准确率则终止训练，n 可取 10, 20, 30 等。

正规化：即将原先的损失函数加上一项网络参数的范数。
设原先损失函数为 L_0，参数为 w

对于 L1 正规：$L = L_0 + \frac{\lambda}{n}\sum |w|$,

对于 L2 正规：$L = L_0 + \frac{\lambda}{2n}\sum w^2$,

其中 n 为参数总数，λ为超参数

扩展训练集：
通过重采样、加入随机噪声等方法人为增加数据集，使得训练出的模型更有泛化能力。

4.分析 CNN, RNN 和 MLP 的优缺点：
同样训练 200epoch 时，CNN 在准确率、f-score 方面效果比 RNN 好，而 RNN 网络在相关系数上比 CNN 略高；然而 RNN 在使用均方误差训练 500 个 epoch 后准确率和相关系数都高于训练了 200 个 epoch 的 CNN，但 f-score 反而下降；MLP 的效果整体差于 CNN、RNN，但所需训练时间较短，网络结构也更简单。

六、心得体会

1. 通过此次实验，我掌握了 RNN 和 CNN 的基本结构，了解了使用人工神经网络进行深度学习的基本方法，学习了 pytorch 框架的使用。
2. 我学习了使用 GPU 进行计算。即使是 940MX，计算时间也缩减为 CPU 计算的 1/5，使得超参数对比变为可能。
3. 通过对超参数的对比，我发现不同超参数对模型训练效果的影响非常大，遗憾由于训练时间限制不能测试更多不同参数效果（如学习率、激活函数的选取、损失函数的选取和网络架构等）。
4. 由于实验提供数据集很小，且各标签数量相差极大（标签 3 占据近 50%），必须在训练时通过重采样增加 3 以外标签的数量，才能学习到其他标签的特征。
5. 使用均方误差函数训练了 500 个 epoch 后的 RNN 在准确率和相关系数上有明显提高，遗憾没有时间验证增加迭代次数和使用其他损失函数是否能使 CNN 也有较大提高。