人工智能导论——文本情感分类 实验报告

滕启成 计16 2021010837

实验目的

利用CNN、RNN等神经网络模型实现文本情感二分类，掌握深度学习框架的基本使用方法，了解不同神经网络模型的结构与特点。

实验环境

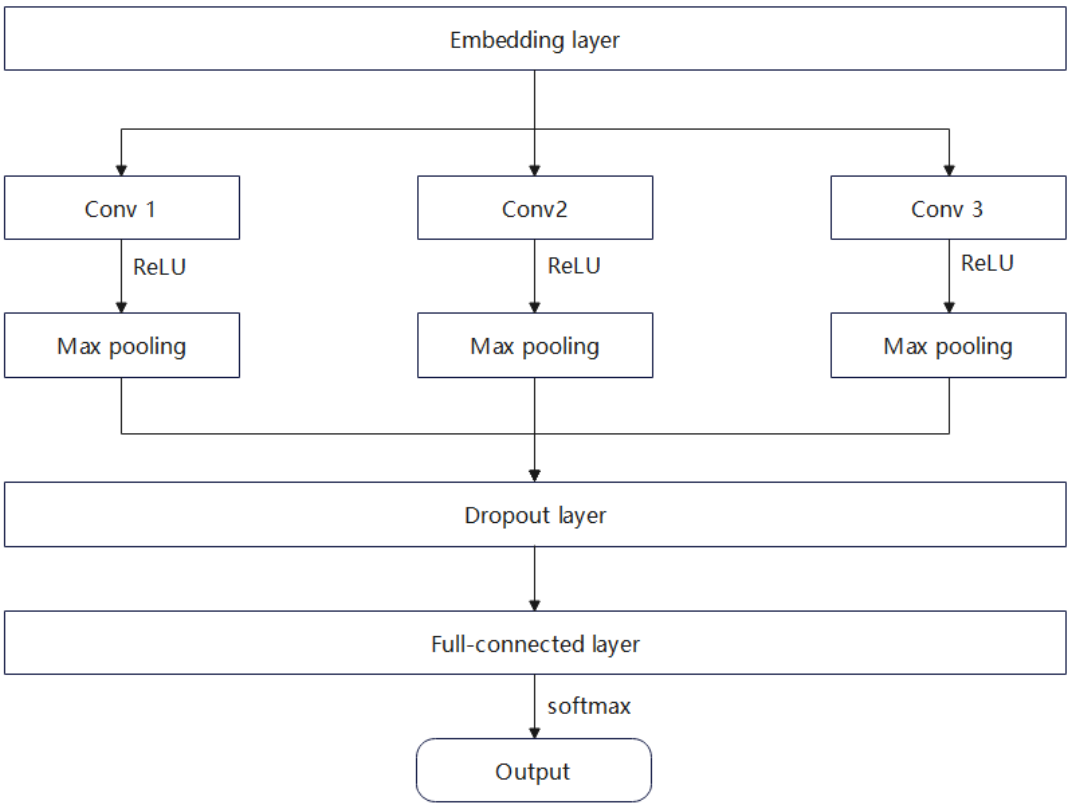
本次实验在Windows 11系统上进行，深度学习框架为pytorch 1.12.1。

由于本机的GPU出现了未解决的bug，模型训练暂且在CPU上进行。

模型结构

TextCNN

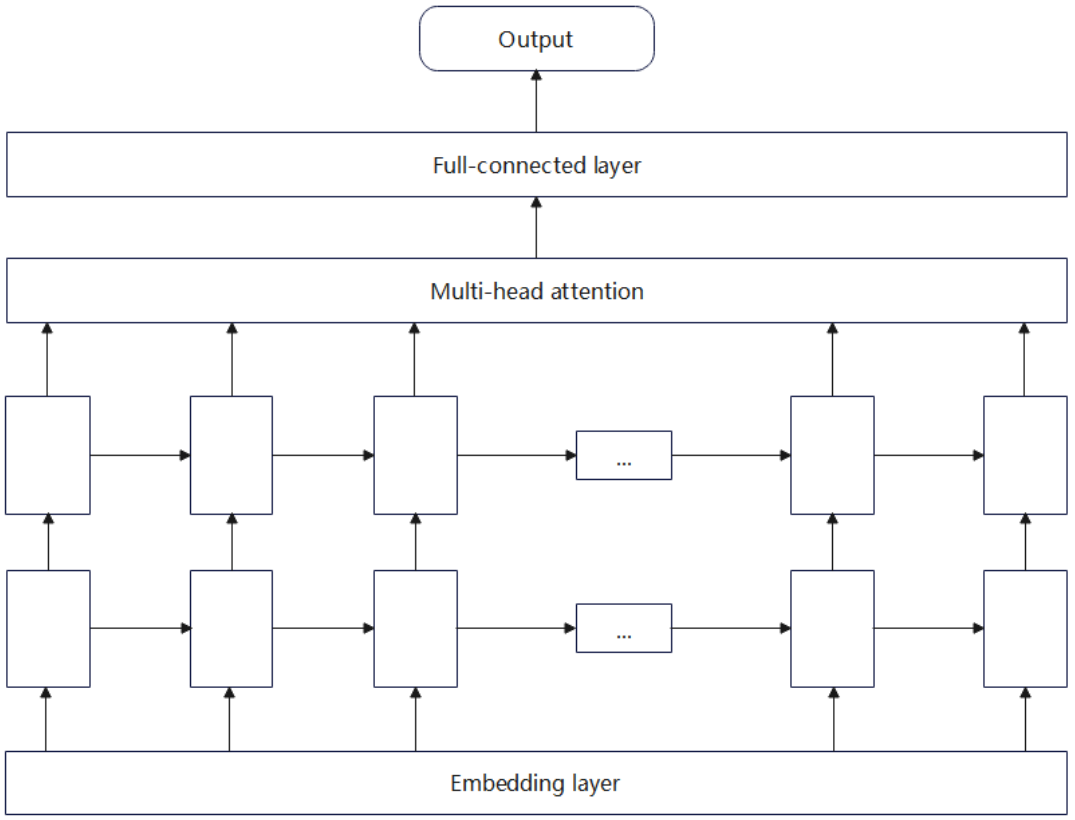
TextCNN即用于处理文本的卷积神经网络（Convolution Neuron Network），其主要结构如下图所示：



- 嵌入层：将输入的句子（词汇用token表示）转化为对应的50维词向量；
- 卷积层：分别用高度3,5,7的卷积核（各20个）对嵌入后的句子张量进行卷积，激活函数使用ReLU；
- 池化层：对三个卷积结果进行一维最大池化，得到的一维向量拼接起来；
- dropout层：以0.3的概率随机将一些神经元的值置为0，用于降低模型复杂度，减小过拟合；
- 全连接层：输入为60个神经元，输出为2个神经元，用于最终的分类，分类结果使用softmax处理后输出。

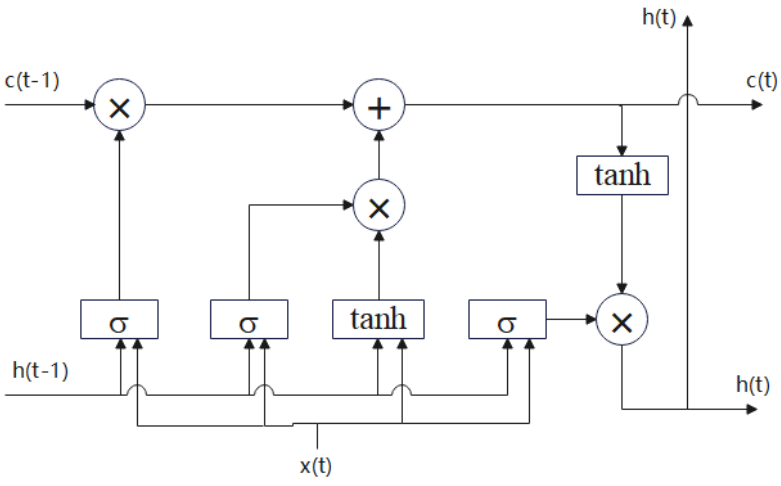
RNN

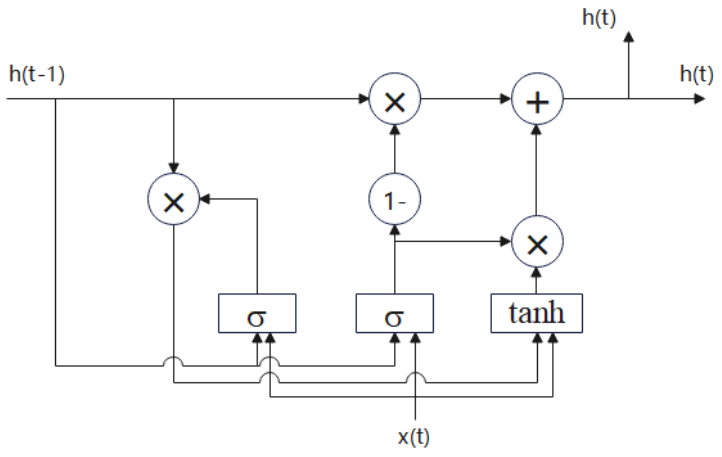
RNN即循环神经网络（Recurrent Neuron Network），多用于处理文本等序列问题。本次实验中实现了LSTM和GRU两种RNN模型，两种模型的外部结构基本相同。



- 嵌入层：同上；
- RNN层：共两层，第二层接受第一层的输出作为输入；
- 多头注意力机制：利用每个RNN单元的输出，以输出的softmax为权值对RNN输出进行加权和操作；
- 全连接层：用于分类，功能同TextCNN。

LSTM和GRU的区别主要在于单元的结构不同。





图中左侧为LSTM单元结构，右图为GRU单元结构。

MLP

MLP即全连接神经网络（多层感知机），采用纯线性层实现（结构图略），主要作为baseline与CNN和RNN进行对比。

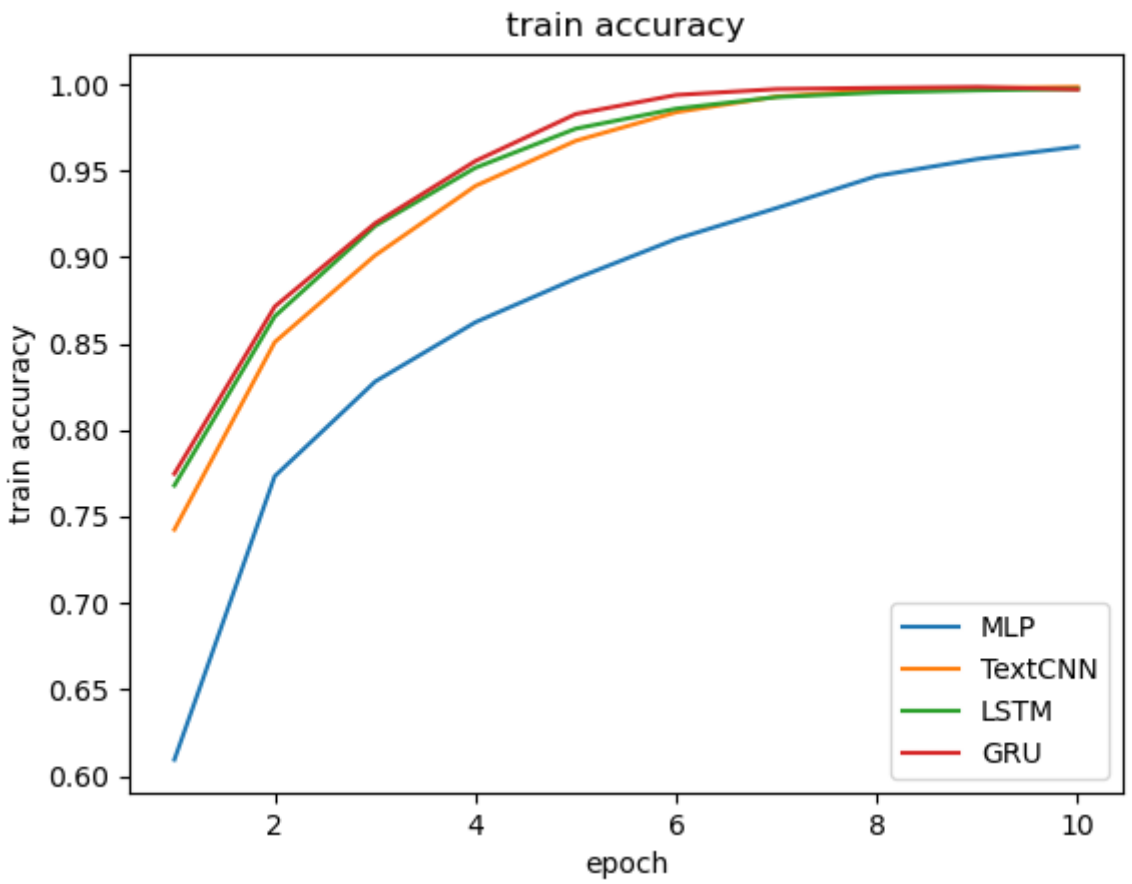
本实验中MLP只有一个隐含层，含有100个神经元，激活函数为sigmoid，dropout概率为0.3。

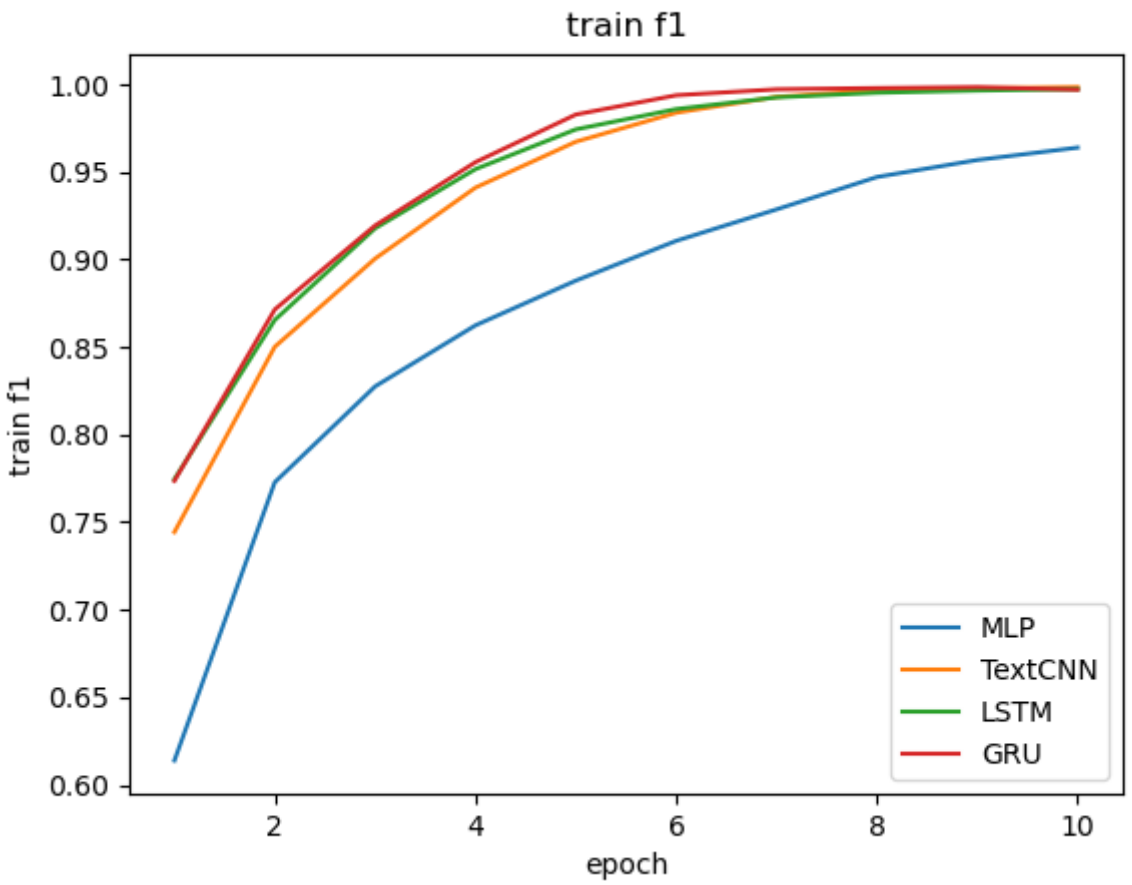
实验结果

模型对比实验中，初始学习率采用推荐值10^{-3}，Epoch设为10，batch_size为50，句子最大长度为60（长句截断，短句补齐）。

评价指标为准确率（accuracy）和F1-score（精准度与召回率的调和平均），实验中使用matplotlib.pyplot进行可视化。

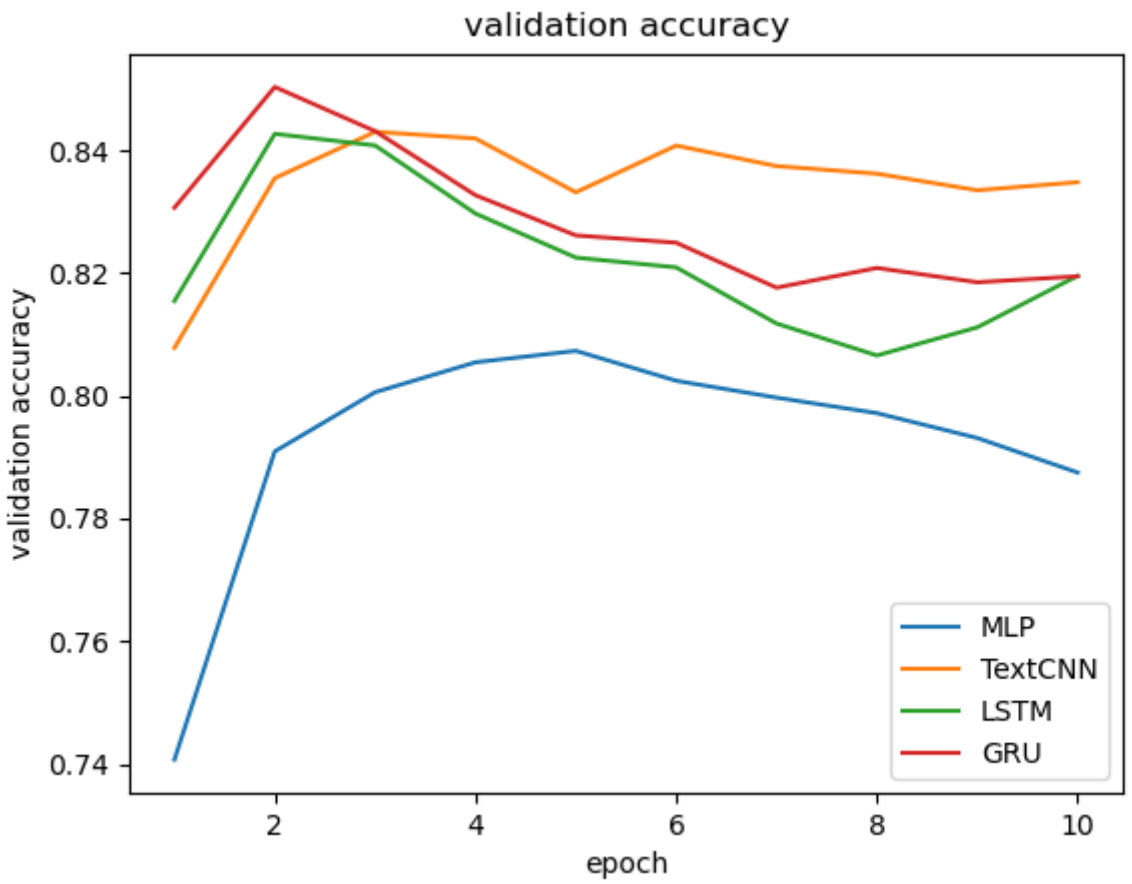
训练集结果

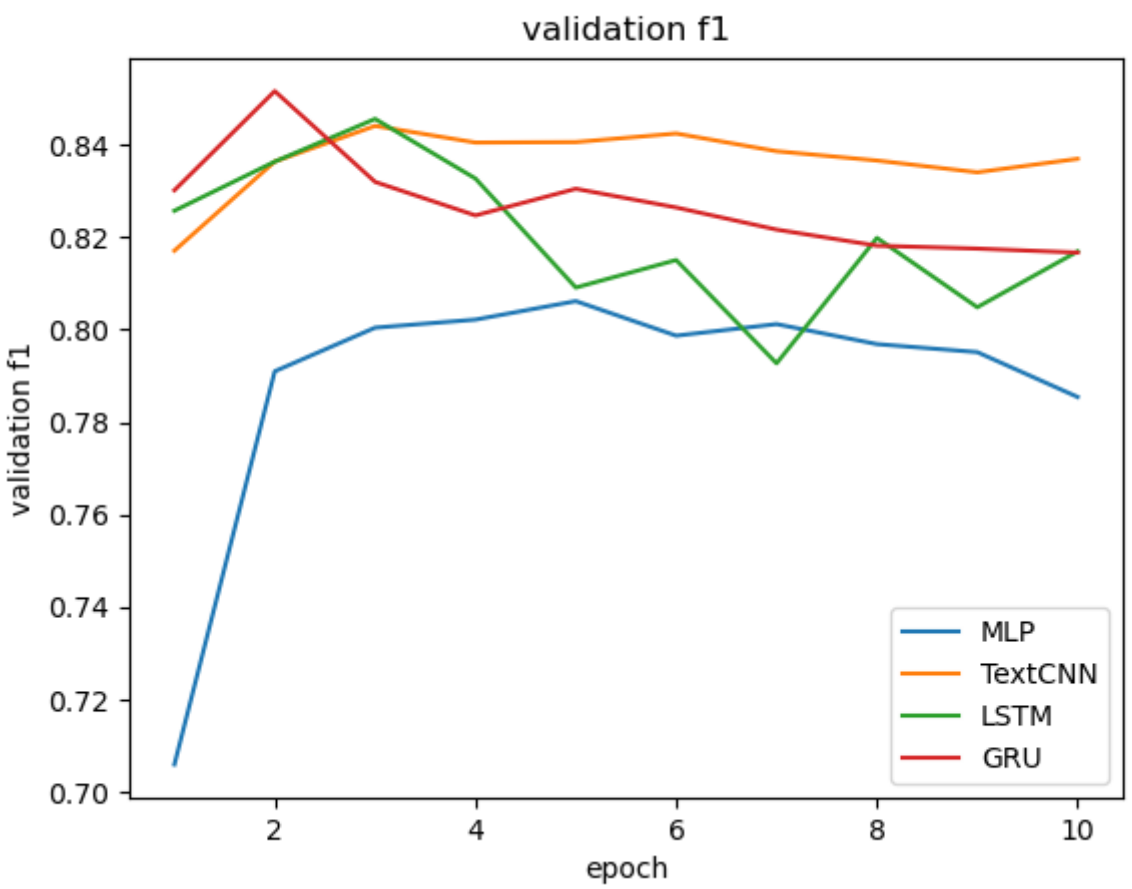




训练集的准确率和F1-score主要用来比较不同模型的收敛速度。从上面的折线图可以看出，LSTM和GRU两种RNN模型的收敛速度不相上下（GRU稍快一些），TextCNN略慢一些但差别不大，MLP收敛速度最慢，10个epoch后还没有完全收敛。

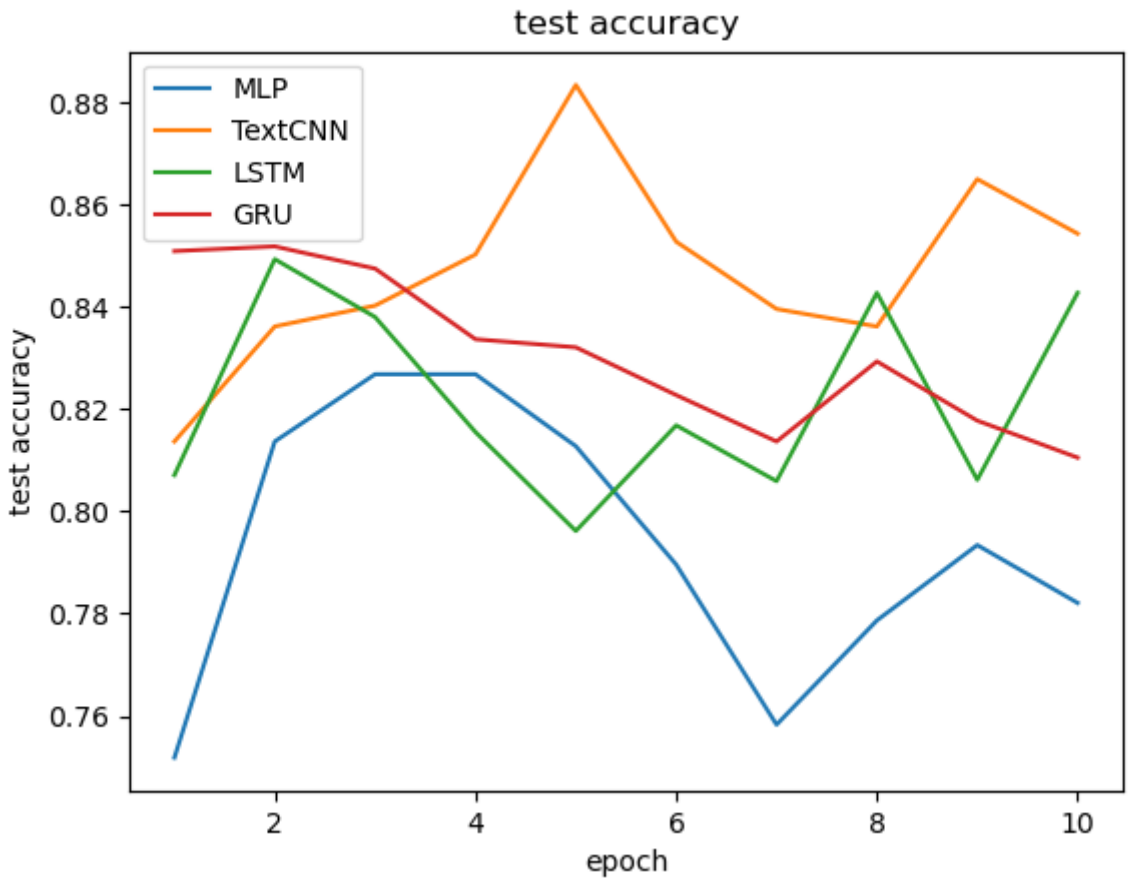
验证集结果

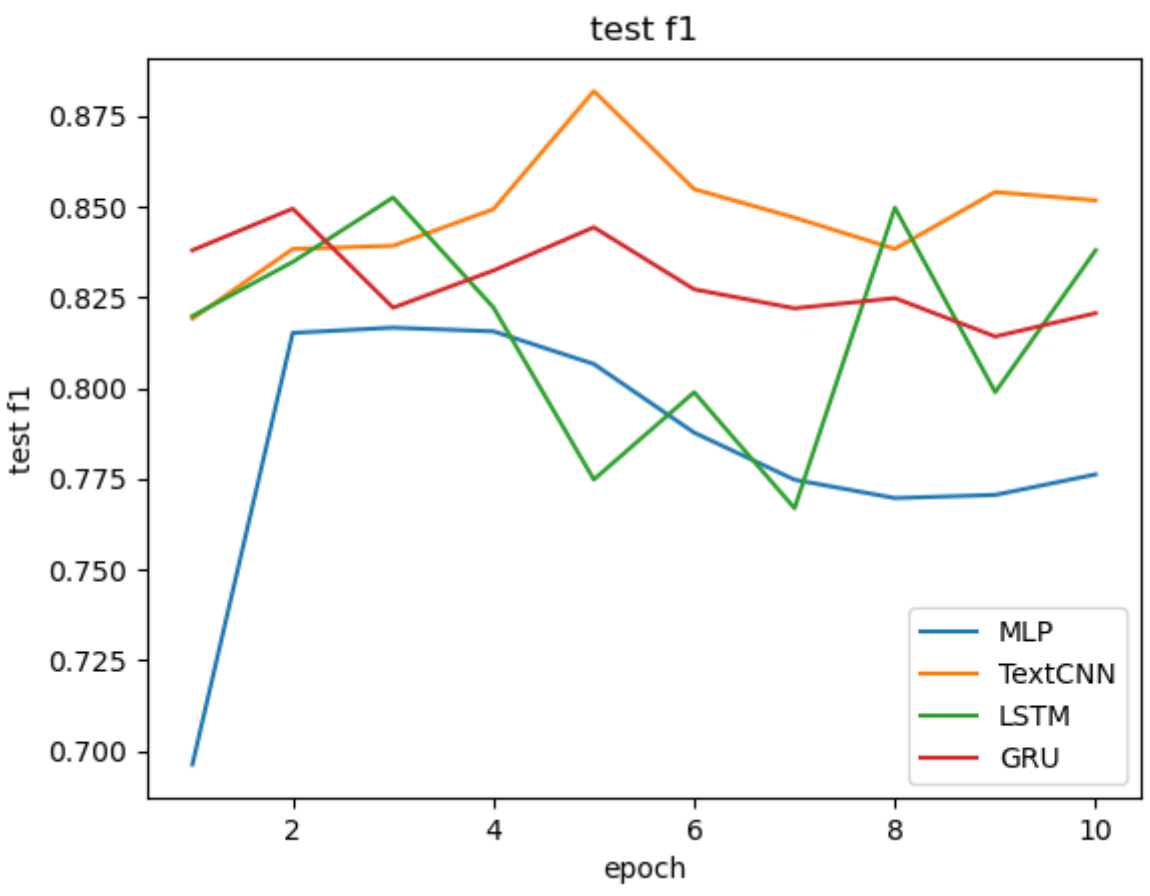




从验证集的结果可以看出不同模型出现过拟合的情况。RNN两种模型出现过拟合最快、最严重，MLP其次，TextCNN的过拟合现象最不明显。

测试集结果





综合验证集和测试集的表现，TextCNN的表现最好，准确率最大值可超过0.88，平均准确率和F1-score都在0.83~0.85附近；RNN其次，平均准确率和F1-score在0.82左右；MLP最差，两项指标很少突破0.8。两种RNN模型中，LSTM的表现波动性更大，GRU相对比较稳定。

实验结果总结如下：

- 收敛速度上，RNN最快，CNN其次，MLP最慢；
- 过拟合程度上，CNN最轻，MLP其次，RNN最严重；

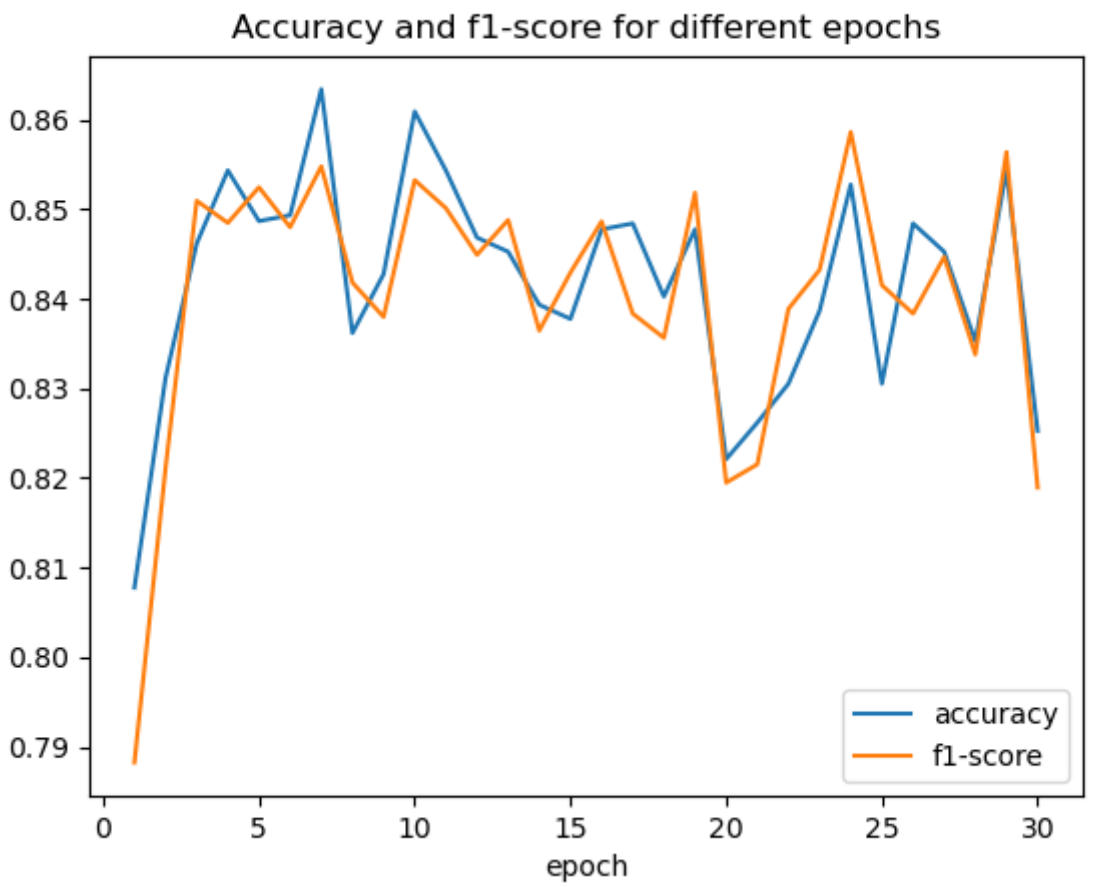
- 综合表现上，CNN最好，RNN其次，MLP最差。

参数调整

本实验主要涉及四个参数：迭代次数、初始学习率、batch_size和句子长度。下面探究除batch_size外的三个参数对于正确率的影响。

迭代次数

以TextCNN为例，迭代30次的测试集准确率和F1-score变化如下图：



可以看出，在第5个epoch左右时模型就已经基本收敛，在第10个epoch后出现了一定程度的过拟合，正确率总体略呈下降趋势。因此，在本实验中，迭代10次左右基本合适。

句子长度

本次实验可由命令行参数指定句子的最大长度，长度不足的用占位符补齐，长度过长的进行截断。下面是TextCNN和LSTM在最大长度40,60,80,100时的表现，准确率和F1-score均为10个epoch中测试集上的最大值。

模型/准确率/长度	**40**	**60**	**80**	**100**
TextCNN	0.830	0.863	0.864	0.871
LSTM	0.821	0.856	0.854	0.861

模型/F1-score/长度	**40**	**60**	**80**	**100**
TextCNN	0.834	0.868	0.859	0.865
LSTM	0.804	0.845	0.853	0.853

最大长度设为40时由于信息丢失过多，模型准确率较低；但最大长度超过60后，继续增大长度限制并没有给模型带来很大的提升。这与电影评论的文本特点有关，大部分影评的长度不会很长，而且情感表达也比较直白，将长度限制在60左右就能够得到很好的效果。

对于LSTM而言，本实验中固定最大长度的做法一定程度上限制了它的发挥，如果将长度限制去除，LSTM可能会有更好的表现。

初始学习率

学习率可以理解为每次梯度下降时行进的步长，通常由初始值开始指数衰减。下面是在不同初始学习率r_0的情况下TextCNN模型在前10个epoch里的最大准确率和F1-score：

评价指标/$-\lg r_0$	1	2	3	4	5
准确率	0.531	0.858	0.872	0.829	0.736
F1-score	0.677	0.861	0.872	0.818	0.739

可以看出默认初始学习率10^{-3}时表现最好。学习率太大导致模型精度太差，而学习率太小又会导致收敛过慢。

问题思考

训练停止时机

最初采用的方法是观察验证集的结果，当验证集准确率连续若干次（如10次）没有明显提升，且恰好要出现过拟合现象时停止训练。但是根据之前的训练与测试结果，CNN和RNN的收敛速度都很快，在第3个epoch左右就基本达到收敛，因此最终直接采用固定迭代次数的方式决定训练停止时机。

本实验任务较为简单，数据量不算特别大，因此训练停止的时机问题不算特别重要。但在实际生产及科研场景中，大模型的训练对于资源的消耗是非常巨大的，训练的停止问题就变得十分重要。固定迭代次数相比较而言更容易实现，但是迭代次数本身不好确定，可能出现欠拟合或者过拟合现象；而通过验证集测试结果调整迭代次数能够有效提升模型准确度，减少欠拟合与过拟合，但是操作难度较大，而且验证集结果本身并不稳定，无法完全避免欠拟合与过拟合现象。

参数初始化

对于采用梯度下降法的神经网络而言，好的参数初始化可以加快收敛速度，减少模型的训练时间。一般来说，神经网络参数的初始化主要有以下几种方式：

- 全零初始化。这种方式是绝对不能使用的，因为全部初始化为零会导致在反向传播的过程中梯度均匀回传，所有权重的更新保持同步，导致同一隐含层的所有神经元无法区分，神经网络发生退化（称为对称失效）。这样无论训练多少轮都不会得到令人满意的效果。
- 标准初始化（随机初始化）。有两种方式：高斯分布初始化和均匀分布初始化。高斯分布初始化即按照期望为0的高斯分布（正态分布）对权值进行初始化$W \sim N(0, \sigma^2)$，而均匀分布则是按照$[-r, r]$上的均匀分布进行初始化$W \sim U(-r, r)$，其中$r = \sqrt{3\sigma^2}$。在网络结构较为简单时，标准初始化就可以获得很好的效果，但网络结构较为复杂时会出现问题。比如在使用sigmoid或者双曲正切等激活函数时，方差过小导致神经元输出过小，经过多层之后逐渐趋于0，而方差过大又导致神经元输出落在激活函数较为平坦的区域，产生梯度消失问题。
- Xavier初始化。Xavier初始化相比于标准初始化的好处在于，标准初始化在向前传播的过程中输出值方差会不断增大，而Xavier初始化可以在理论上保持输出值方差的不变。假设对应层输入、输出神经元的个数分别为n_{in}, n_{out}，Xavier初始化的公式如下：

$$W \sim N\left(0, \frac{2}{n_{in} + n_{out}}\right)$$

$$W \sim U\left(-\sqrt{\frac{6}{n_{in} + n_{out}}}, \sqrt{\frac{6}{n_{in} + n_{out}}}\right)$$

  使用这种初始化方式，在传播层数较多时仍然可以保证参数基本服从正态分布，有利于模型的训练。

- Kaiming初始化。这也是pytorch对于本次实验中所有模型的默认初始化方法（稍有改动）。实际应用中发现，Xavier初始化在使用ReLU作为激活函数时会导致输出逐渐向0偏向，导致梯度消失。针对ReLU函数的这一问题，何恺明等提出只考虑输入神经元的Kaiming初始化：

$$W \sim N\left(0, \frac{2}{n_{in}}\right)$$

$$W \sim U\left(-\sqrt{\frac{6}{n_{in}}}, \sqrt{\frac{6}{n_{in}}}\right)$$

  这样可以有效解决ReLU函数的梯度消失问题。

- 正交初始化。由于正交矩阵具有范数保持性（$\|Qx\| = \|Q^T x\| = \|x\|$），将权值初始化为正交矩阵可以有效防止梯度消失和梯度爆炸的问题。具体而言，对于权值矩阵W，可先用高斯分布获得一个随机矩阵，然后执行矩阵的QR分解$W = QR$将其写为列正交矩阵Q和上三角矩阵R的乘积，用Q作为初始的权值矩阵。正交初始化在训练较深的神经网络时应用非常广泛。

过拟合问题

过拟合是深度学习中常见的问题。当模型过于复杂、参数过多时，往往导致模型的泛化能力降低，而训练集上迭代次数过多、训练集数据量不够或不均匀也可能导致模型学习到非共性的特征，这些都是过拟合产生的常见原因。

解决过拟合问题的方法主要有下面几种：

- 使用dropout。在某些层上加入dropout层，随机将一些神经元的输出值置为0，暂时舍弃这些神经元，可以有效降低模型的复杂度。
- 引入验证集。在训练过程中引入验证集（仅测试，不进行反向传播），每次迭代后验证训练集的准确率，可以实时监测过拟合现象的发生情况，在合适时机停止训练。
- 损失函数中加入正则化项。一般模型越复杂，正则化项就越大，在损失函数中引入正则化项可以避免模型复杂度上升。

模型对比

本次实验中三种基本模型对比如下：

- MLP实现最简单，训练时计算速度最快；但是MLP的上限比较低，隐含层神经元过少导致学习能力不够，过多又容易出现过拟合，同时固定长度的语义理解方式容易导致语义丢失，在长文本处理方面应用场景有限。
- CNN相比MLP的参数量显著减少，过拟合现象减轻，计算速度也比较快，而且CNN能够有效提取句子的局部特征，非常擅长较短文本的处理；但当文本长度增加时，提取局部特征的理解方式会忽略距离较远的文本之间的联系，同样导致语义丢失，因此CNN并不擅长非常长的文本处理。
- RNN采用与人理解句意类似的方式处理文本，可以提取较长文本的特征，在处理长文本时优势显著，同时可变长度输入也使得模型更加灵活，收敛速度也比较快；但是RNN模型相对复杂，容易出现过拟合，而且时序性导致其无法并行计算，训练时间长，因此在短文本处理上表现不如CNN。

心得体会

深度学习是当下科研和生产领域非常火热的技术，作为计算机系学生，到大二才第一次较为系统地了解深度学习并进行实践，感到有些惭愧（当然我并不清楚一个“合适”的节奏是怎样的，或者是否存在一个“合适”的节奏）。在这个各个专业都在炼丹的年代，深度学习似乎已经成为大学生的必备技能。

这部分的学习和作业之间有着比较大的gap，主要原因在于课上内容仅仅涉及理论部分，而实验所需的框架并未进行讲解。要完成这样一次作业，既要理解神经网络的基本原理和几种模型的结构，又要对所用的框架有所熟悉。只学理论的话，作业完成将会非常困难，比如手写BP算法，这在实际应用中既复杂又没必要。但同时，仅仅对照网上的教程学习框架使用也是不够的，只会用轮子而不知道轮子的运行原理，就可能出现很多难以捉摸的bug，例如不了解BP算法的过程就可能陷入全零初始化的陷阱。从这个角度而言，这次作业让我对于计算机领域“造轮子”“用轮子”之间的关系有了更加深入的理解。