**情感分析实验报告**

**张锦黔**

**201604014013**

**一、实验环境和数据准备**

**1.1 用Anaconda配置TensorFlow和Gensim环境**

各主要程序版本：

* python 3.7
* AnacondaNavigator
* TensorFlow 1.13.1
* Genism 3.4.0

**1.2 维基中文百科语料库**

来源：<https://dumps.wikimedia.org/zhwiki/latest/>

数据规模：3273626段中文语料，参考CSDN博客进行了分词、过滤非汉语字符的处理

**二、卷积神经网络（CNN）的设计和实现**

**2.1 数据预处理**

**(1) 词向量训练：**

使用维基中文百科语料库，用Gensim提供的word2vec方法进行词向量的训练。

在题目说明文档提供的词向量参考链接中，作者使用了长度为300的词向量。这为我设计自己的词向量提供了启发：由于个人电脑的性能有限，出于节省模型训练和后续神经网络训练时间的目的，对词向量的规模做了缩小。

缩小时考虑了以下方面：1.保证词向量的基本作用；2.减小计算规模；3.最终的神经网络模型尚未定型，在卷积神经网络中张量的规模会在多次池化时经历多次缩小，为方便后续调试时能重复利用此词向量模型，向量长度应当能被多次整除，故尽量选择2^n作为长度。（后面对文本长度的选取也遵循了这一思路）

经过尝试，设定词向量的长度为128，低频词频数为10，窗口宽度为5。

**(2) 文本分割：**

在提供的实验数据中，每个新闻文本为一行，分为时间戳、标签、文本内容三部分，相互以“\t”分隔，文本内部各词以空格分隔。使用readlines( )函数分行读取实验数据，split( )函数分别以\t和空格作为分隔符，完成对文本的分割，得到相应的list。

经过几个版本实验结果的对比，又在文本分割这一步增加了一个环节：词汇去重，即使每个词只在list中出现一次，在生成list时考察新加入list的词汇，若已经在list中存在则不再加入。经过试验，词汇去重可以有效降低一些高频中性词的影响，并缩短文本规模，节约训练模型的时间开支。

**(3) 情感分类：**

我选择了将情感分析转化为分类问题的方案。通过split(“\t”)得到的列表，其第二个元素为情感标签文本。使用re.findall()匹配其中的整数字符串“d+”，得到九个整数，其中第一项为后8项之和。去除第一项并进行数据格式转换后，取最大值的索引（0~7）即可代表文本的情感标签。

**(4) 文本转化为词向量**

通过split(“\t”)得到的列表，其第三个元素过split()函数即可得到以文本中每个单词作为元素的列表。

在最初的实验版本中，经统计，列表平均长度为426，最大长度为2490，截取取长度1024可保证取95%左右的文本。因此最初将列表长度统一为1024。

几个改进版本后，引入了词汇去重方案。1024作为列表长度的另一个缺点在于向很多数据的末尾补了大量的0值。我认为，大量0值对模型的干扰作用等同于高频停用词，因此必须去除。

经统计，列表做完词汇去重之后的平均长度为241，因此决定将长度定为256，这时处理后每个列表数据中的非0值占比平均值达到了77%。

统一长度后，加载训练好的word2vec词向量，将每个列表转化为256\*128的numpy数组，数据格式为float64。

**(5) 生成.tfrecords文件**

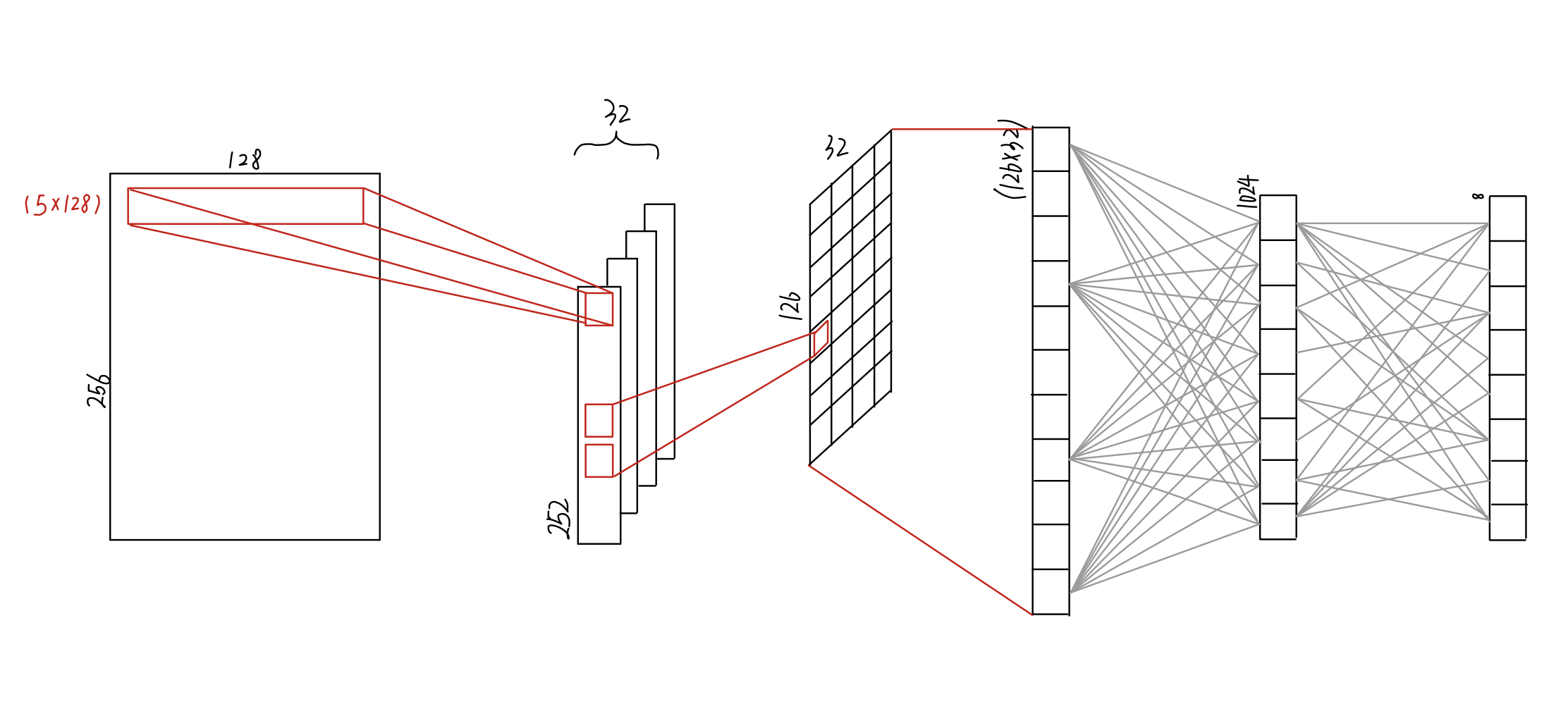
参考TensorFlow官网提供的例程，将步骤（4）中生成的numpy数组转化为bytes格式作为数据，将步骤（3）中得到的情感标签作为标签，生成tfrecords文件。经过对sinanews.train和sinanews.test两个文件的处理，生成了train.tfrecords和test.tfrecords两个二进制数据文件，可被TensorFlow神经网络用作的dataset文件。

**2.2 网络设计和代码实现**

使用TensorFlow提供的Estimator API，搭建CNN神经网络。作为高级API，Estimator可以简化搭建神经网络的工作，降低代码量。

参考TensorFlow官网提供的例程，我编写了自己的模型函数cnn\_framework和训练/测试时的输入函数train\_input/evaluate\_input，然后利用Estimator提供的分类器和训练函数等接口完成CNN神经网络的训练与测试。

cnn\_framework实现了以下描述的神经网络（全连接层的连接关系以简略方式在图中画出）：



**(1) 输入层：**

输入一个256\*128尺寸的张量，通道数为1。该张量读取自.tfrecords文件，经过cast函数转换为float64数据类型，并使用reshape函数完成尺寸转换。

**(2) 卷积层：**

卷积核尺寸：3\*128，数量为32，由输入的一个256\*128张量生成32个252\*1张量。

**(3) 池化层：**

池化器尺寸：2\*1，步长[2,1]，将卷积后的张量池化，生成32个126\*1张量。

**(4) 全连接层：**

将池化后的32个张量投射到1024个神经元，并应用丢弃正则化，丢弃正则化率为0.4。

**(5) 输出层：**

将1024个神经元的值投射到8个神经元中，得到神经网络的计算结果。

**(6) 激活函数：**

使用tanh激活函数。

**(7) 损失函数：**

使用交叉熵函数作为损失函数。

**(8) 优化器：**

使用随机梯度下降算法进行优化。

**(8) 评估指标：**

准确率：调用tf.metrics.accuracy( )函数获得

精确率：调用tf.metrics.precision( )函数获得

召回率：调用tf.metrics.recall( )函数获得

相关系数：调用tf.contrib.metrics.streaming\_pearson\_correlation( )函数获得

**2.3 结果分析**

**迭代1200次：**

准确率（accuracy）达到57.63%

精确率（precision）为94.39%

召回率（recall）为92.35%

由精确率和召回率计算F-Score为0.9336

相关系数为0.398

**三、递归神经网络（RNN）的设计和实现**

**3.1 数据预处理**

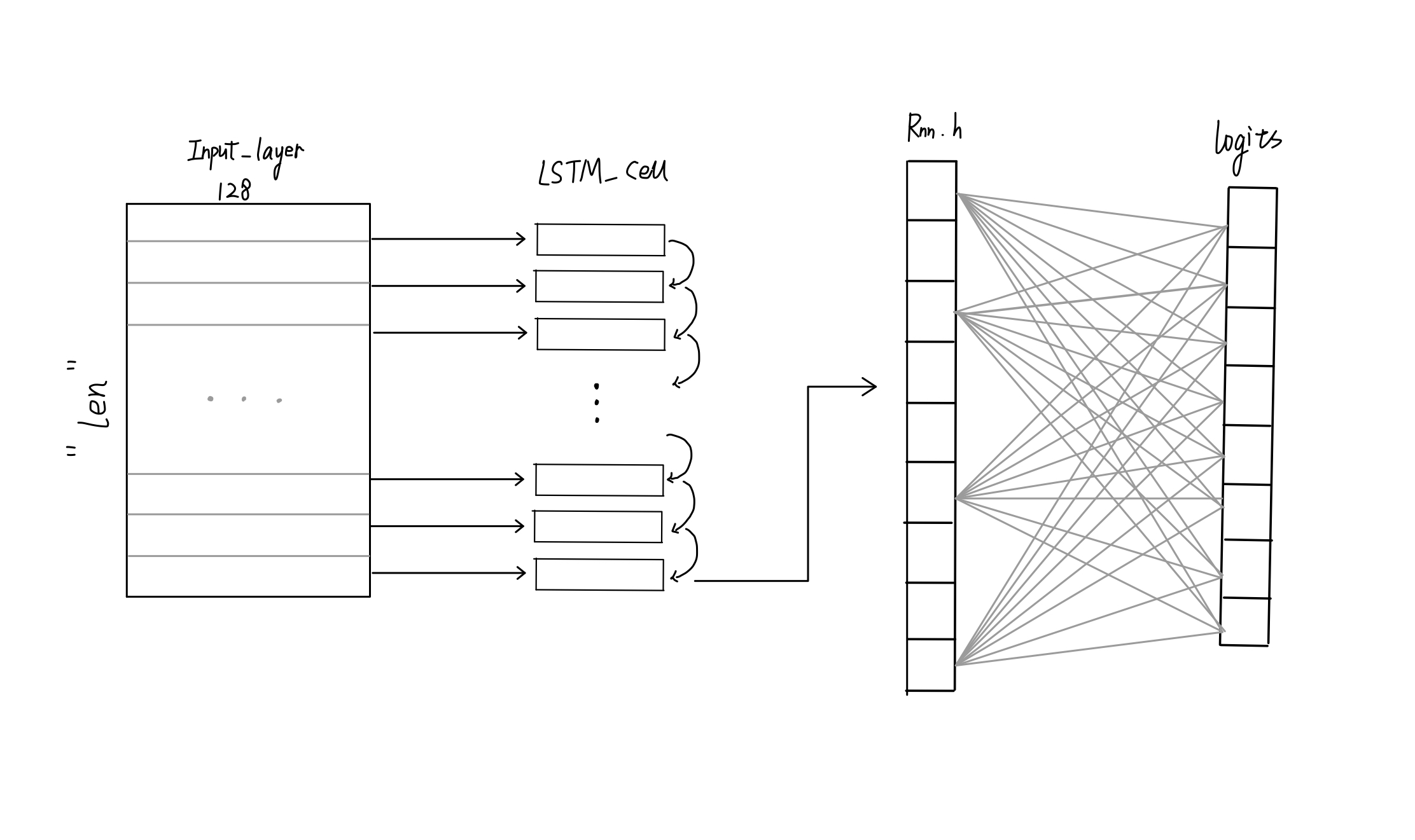
在CNN的数据预处理工作中，完成了将实验数据文本转化为可用于训练的.tfrecords文件的工作。在此基础上，结合RNN网络的特点，进行RNN网络的数据预处理。

**(1) 动态数据长度**

RNN比CNN更易受末尾补0的影响，因此应当更彻底地去除这一现象。在写入.tfrecords文件的数据中增加“len”特征，并传递给Estimator的模型函数，在每次训练时达到len所指定的长度就结束一次处理即可。

**3.2 网络设计和代码实现**

TensorFlow官网给出了RNN相关函数的介绍和实例。参考这些实例，我编写了自己的模型函数rnn\_framework和训练/测试时的输入函数train\_input/evaluate\_input，然后利用Estimator提供的分类器和训练函数等接口完成了RNN神经网络的训练与测试（最后两层之间的全连接以简略方式画出）：



**(1) 输入层**

从.tfrecords文件中读取的二进制数组，经过cast函数转换为float64类型后，用reshape函数完成尺寸转换即可。

**(2) LSTM神经元**

使用TensorFlow提供的BasicLSTMCell获得基本LSTM神经元。

**(3) 递归处理**

TensorFlow提供了dynamic\_rnn进行LSTM神经元的递归处理，实现RNN的核心处理功能。

**(4) 输出层**

将处理后的rnn张量投射到8个神经元上得到神经网络的计算结果

**(5) 损失函数**

使用交叉熵函数作为损失函数。

**(6) 优化器**

使用Adagrad优化算法进行优化

**3.3 结果分析**

准确率(accuracy)达到47.76% (1200次迭代)

精确率（precision）为85.05%

召回率（recall）为100%

由精确率和召回率计算F-Score为0.919

相关系数为0.375

四、思考与分析

**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。**

我认为应当在准确率达到最大值时停止实验。在实验中，我首先通过较少的迭代次数检查代码的可用性，之后以100次为间隔，不断增加迭代次数，训练出不同的模型。在CNN部分，模型的准确率在1200步时达到最高，之后开始下降；在RNN部分，模型准确率在1200步时达到最高，之后开始下降。

**2）实验参数的初始化是怎么做的？不同的方法适合哪些地方？（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）**

实验参数的初始化在Estimator中自动执行。

零均值初始化是使所有参数的均值为0，可以减少梯度弥散问题，使信号在网络中传递得更深。

高斯分布初始化适用于tanh/sigmoid等激活函数，可有效防止激活函数进入饱和区。

正交初始化在RNN中经常使用，可以解决梯度消失、梯度爆炸问题

**3）过拟合是深度学习常见的问题，有什么方法可以方式训练过程陷入过拟**

**合。**

过拟合现象最明显地体现在模型准确率经过一定程度的提升之后进入震荡或下降，是由于模型学习到了一些训练集具备的独有特征，而这些特征在测试集中不具备且对正确分类没有帮助。因此在训练过程中应当不断检测准确率等评价指标，及早停止训练，避免模型出现过拟合。

**4） 试分析CNN，RNN，全连接神经网络（MLP）三者的优缺点。**

在文本分类模型中，CNN的优点在于进行局部特征的提取，缺点是难以训练整句文本的特征；RNN的优点在于能保持训练的连贯性，便于识别整句文本的特征，缺点是更易受噪声干扰，以及“梯度消失”问题。MLP的优点是模型结构简单，缺点是随着神经元的增多，训练参数量越来越大，越来越难以取得较好的结果。

五、心得体会

在设计和实现CNN、RNN网络的过程中，我最大的体会有两点：

1.一个简单而充分调整过的模型，其性能要胜过一个复杂、难以驾驭的模型。在最初的试验版本中，我设计了多层隐藏层的CNN，但由于这个复杂的CNN模型难以进行调试，各项参数对它最终性能的影响难以分析，几经试验都得不到理想的成果，准确率在30%左右徘徊。经过模型的简化，再对学习率、激活函数、优化器等进行不断修改尝试，最终只有一层卷积层和一层池化层的CNN模型准确率接近50%，相比最初的版本有了较大的提升。

2.对实验数据的处理非常重要。神经网络的性能不单单依赖于网络自身，还依赖于输入其中的数据。在实验过程中，我对实验数据的处理方案经过了多次修改，进行过降噪后的试验数据使神经网络的性能得到大幅提升，达到了最终的56%~57%。