文本情感分类实验报告

（云盘链接：https://cloud.tsinghua.edu.cn/d/239adda583054cdea441/）

模型结构以及流程分析

TextCnn

模型结构如下

```
self.embedding = nn.Embedding(num_embeddings=vocabulary_size, embedding_dim=50)
self.cnn_list = [
    nn.Conv2d(
        in_channels=1,
        out_channels=num_filter,
        kernel_size=(size, word_vector_dim),
    )
    for size in filter_sizes
]
```

```
self.fully_connect = nn.Sequential(
    nn.Dropout(0.5),
    nn.Linear(
        in_features=num_filter * len(filter_sizes),
        out_features=2 * num_filter * len(filter_sizes),
    ),
    nn.ReLU(),
    nn.Linear(
        2 * num_filter * len(filter_sizes), num_filter * len(filter_sizes)
    ),
    nn.ReLU(),
    nn.Linear(num_filter * len(filter_sizes), 2),
)
```

主要由三部分组成，一个embdding层，一个可自定义数量的Conv2d列表和一个全连接结构,且为了防止过拟合添加了适当的Dropout

LSTM

模型结构如下

```
self.embedding = nn.Embedding(num_embeddings=vocabulary_size, embedding_dim=50)
self.lstm = nn.LSTM(
    word_vector_dim, hidden_layer_dim, layer_nums, batch_first=True, dropout=0.5
)
self.fully_connect = nn.Sequential(
    nn.Linear(hidden_layer_dim, 2 * hidden_layer_dim),
    nn.ReLU(),
    nn.Dropout(0.5),
    nn.Linear(2 * hidden_layer_dim, hidden_layer_dim),
    nn.ReLU(),
    nn.Dropout(0.5),
    nn.Linear(hidden_layer_dim, 2),
)
```

主要由三部分组成，一个embdding层，一个lstm层，和全连接结构，由于lstm很容易过拟合，所以做了较多的Dropout

MLP

模型结构如下

```
self.embedding = nn.Embedding(num_embeddings=vocabulary_size, embedding_dim=50)
self.max_length = max_length
```

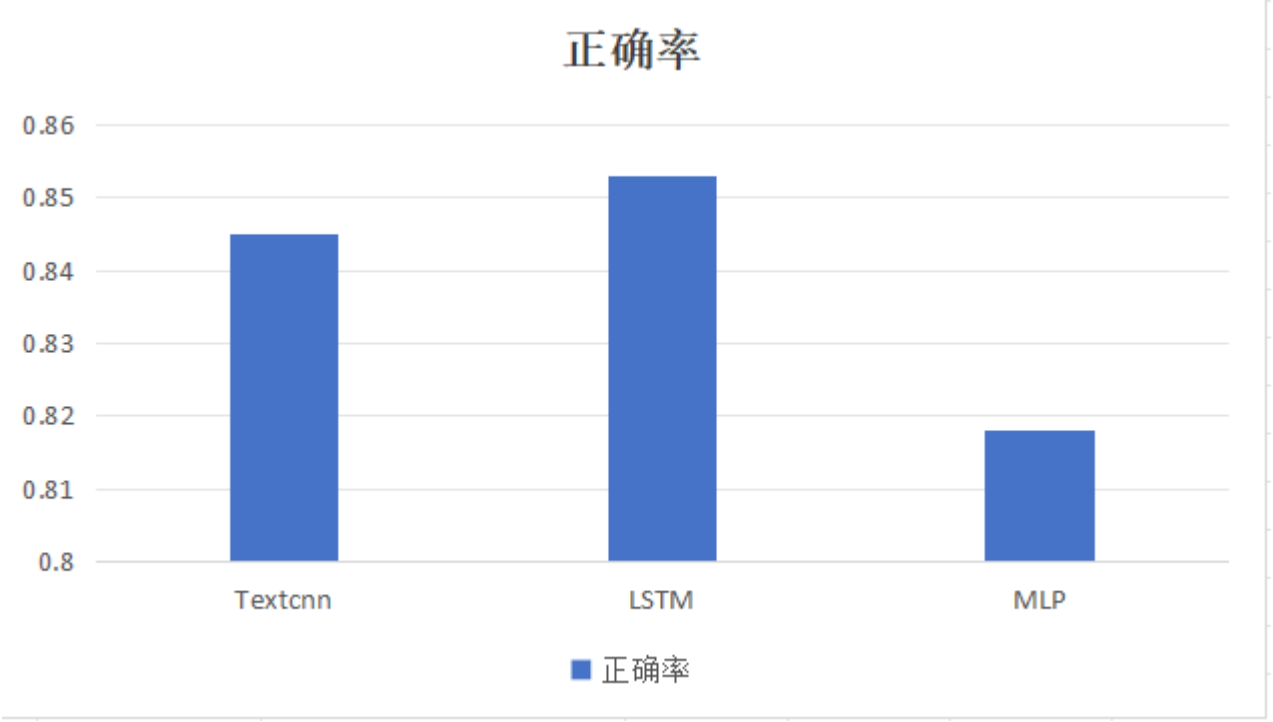
```
self.fully_connect = nn.Sequential(
    nn.Linear(max_length * 50, 32 * 50),
    nn.ReLU(),
    nn.Dropout(0.7),
    nn.Linear(32 * 50, 16 * 50),
    nn.ReLU(),
    nn.Linear(16 * 50, 8 * 50),
    nn.ReLU(),
    nn.Linear(8 * 50, 4 * 50),
    nn.ReLU(),
    nn.Dropout(0.7),
    nn.Linear(4 * 50, 2 * 50),
    nn.ReLU(),
    nn.Linear(2 * 50, 2),
)
```

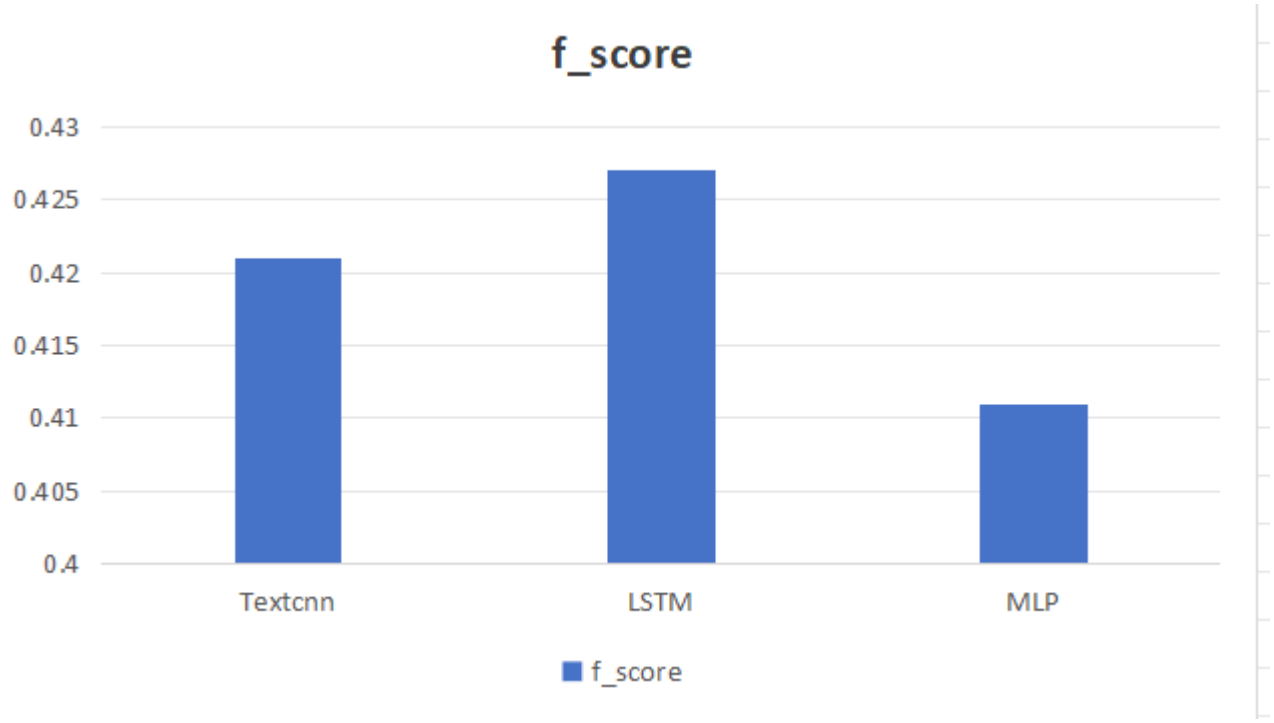
主要由两部分构成，embdding层和全连接结构，在全连接结构部分，采用深度高而非深度浅但是每一层参数都多的结构，经过测试，这种较深的结构能够取得更好的效果

实验结果

三个模型的最佳情况对比

最佳情况的条状图如下:





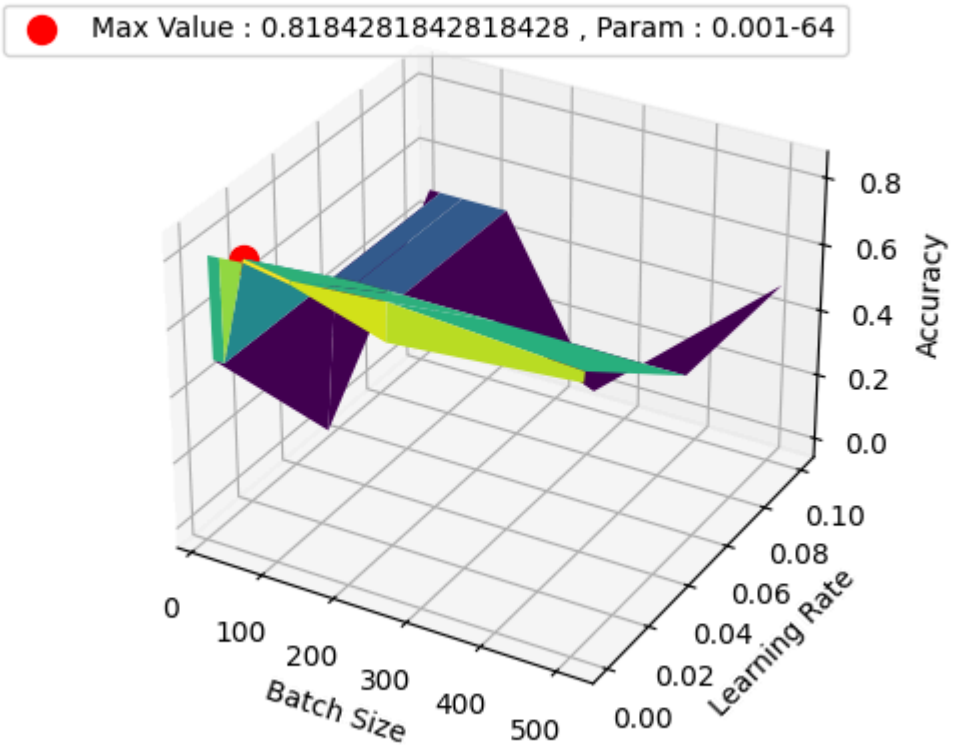
可以看到三种模型在最佳情况下，LSTM有着更好的效果，MLP则效果相对较差，这与LSTM本身参数量和记忆遗忘机制有着较大的关联，且LSTM更适合处理序列性文本的任务。

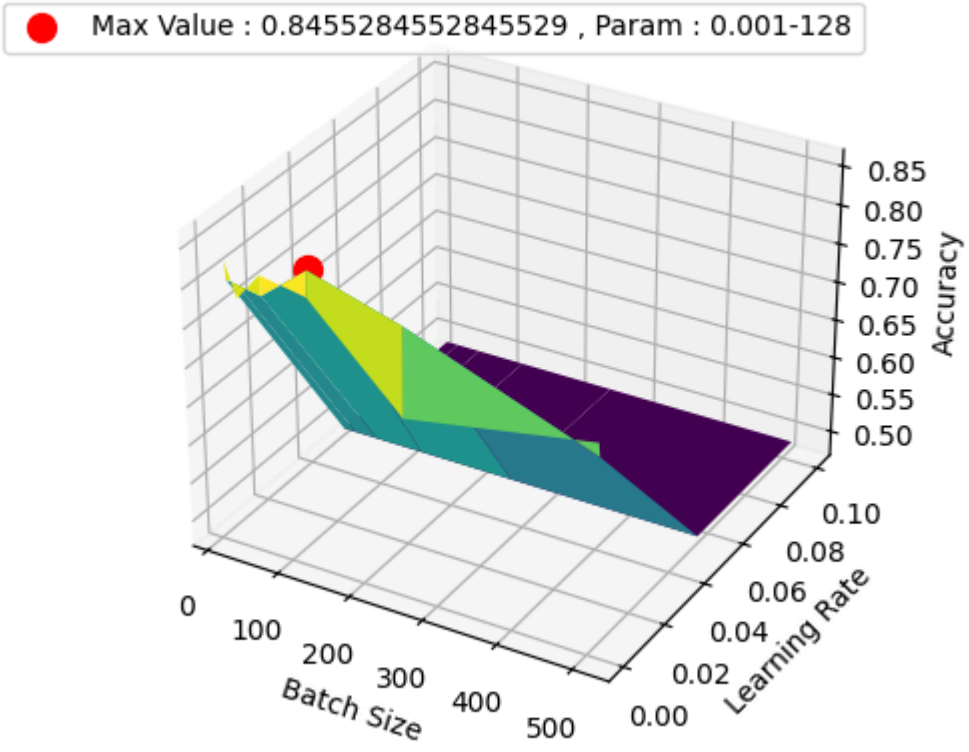
不同超参对实验结果的影响

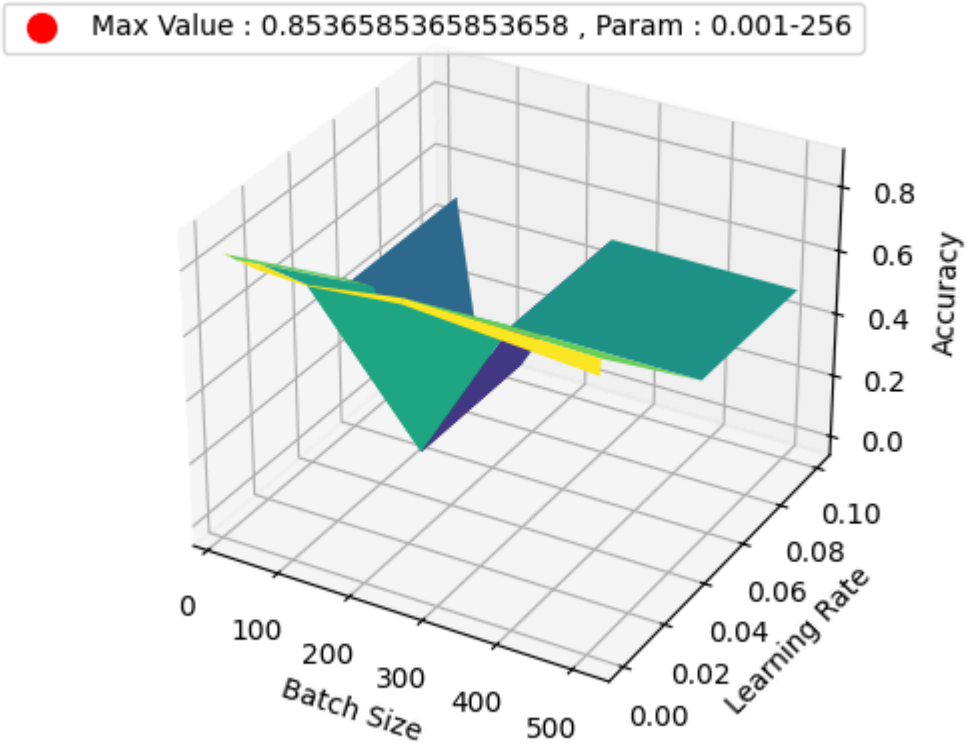
本次实验调参的范围如下

```
lr_list = [0.1, 0.05, 0.001, 0.0005]
batchsize_list = [512, 256, 128, 64, 32, 16]
```

不同超参下三种模型训练后可以达到的最优正确率分布如下

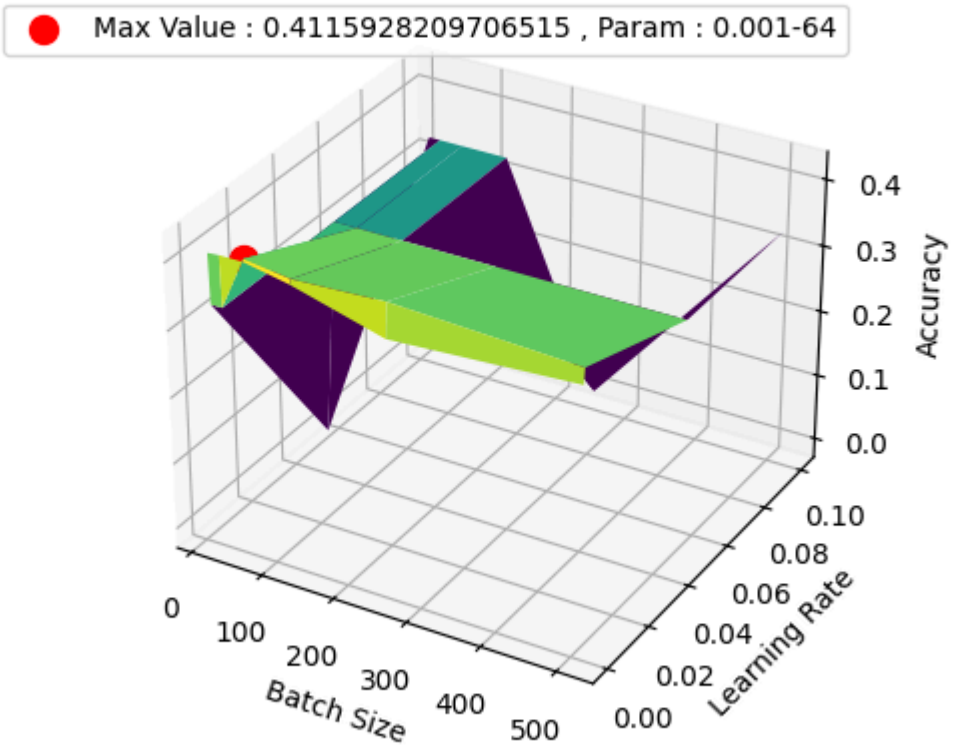


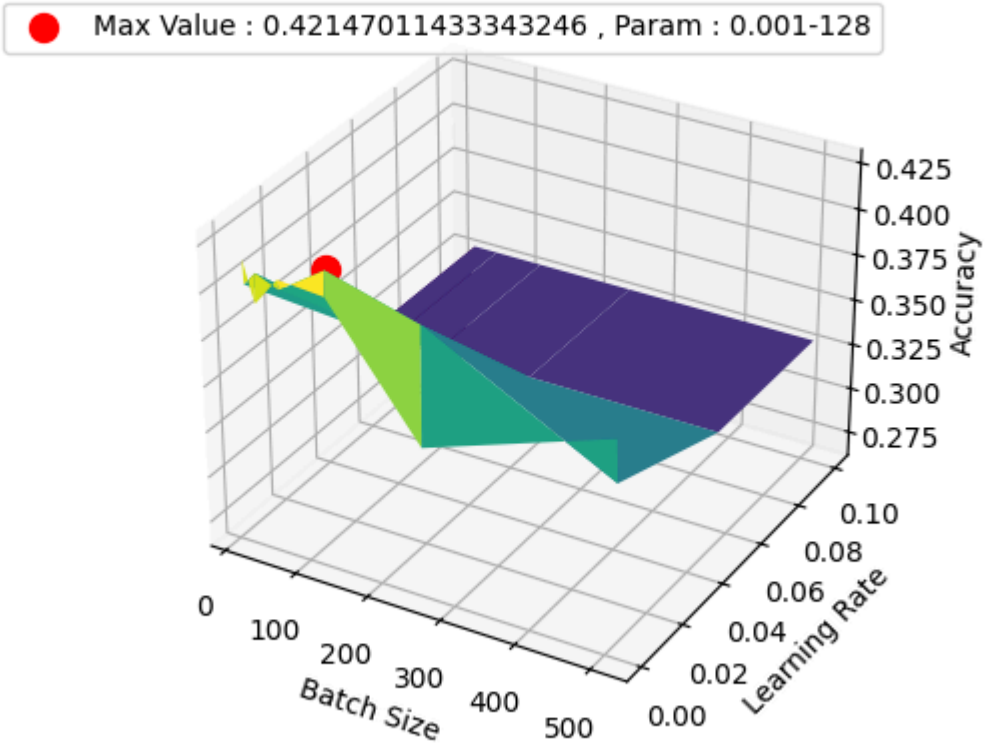


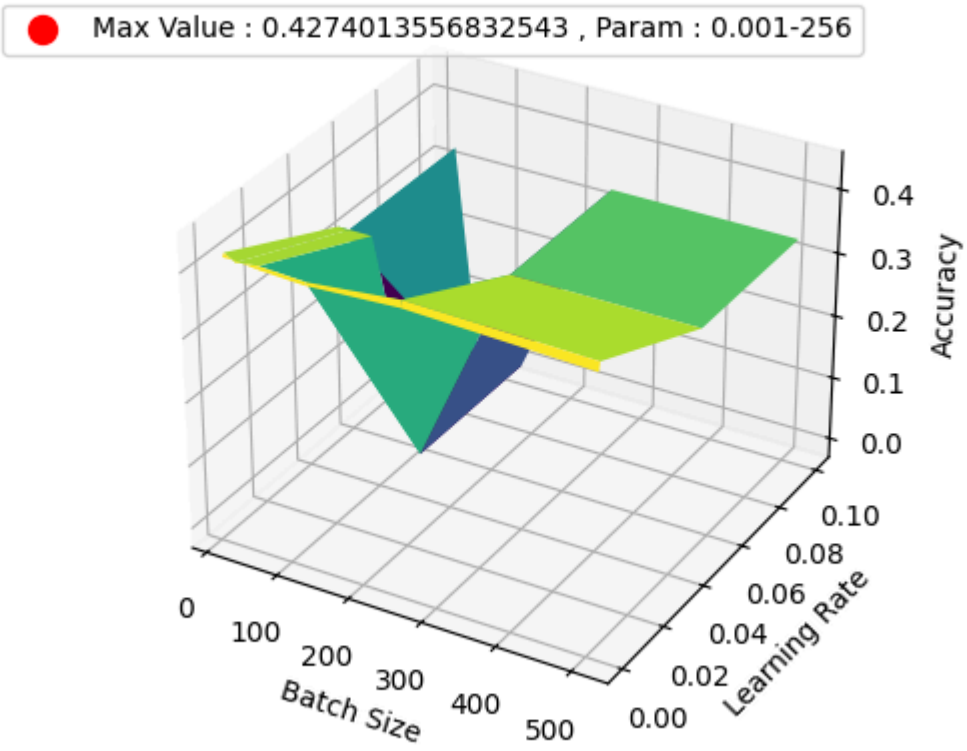


据图做如下分析

- 可以看到三种模型达到最优时batchsize这一参数并不相同，而学习率则相同，且正确率与参数之间有着较大的关系。
- 从图中可以看出对于较大的学习率似乎并没有取得很好的训练效果（这与我使用的早停机制有关，在后续分析）。
- 结论：使用适当的初始化超参有着重要的意义，在算力较为充足的情况下，可以考虑使用第三方的调参工具（如optuna）辅助调参。 不同超参下三种模型可以达到的最优f_score分布如下


Max Value : 0.4115928209706515 , Param : 0.001-64
Accuracy
0.4
0.3
0.2
0.1
0.0
Batch Size
0
100
200
300
400
500
Learning Rate
0.00
0.02
0.04
0.06
0.08
0.10


Max Value : 0.42147011433343246 , Param : 0.001-128
Accuracy
0.425
0.400
0.375
0.350
0.325
0.300
0.275
Batch Size
0
100
200
300
400
500
Learning Rate
0.00
0.02
0.04
0.06
0.08
0.10



可见三种模型f_score达到最优的参数与正确率达到最优的参数完全一致，一定程度上说明二者作为模型性能的评价标准都是合理的。

投票机制

考虑到三种模型的架构之间有着较大的区别，故模型对于输入应该有着较为不同的概率分布，可能答对的问题会有这不同，故尝试让三个模型投票来做出答案。具体机制为对于同一个输入，让三个模型分别作答（投票），然后取投票数多的作为正确答案，实验结果如下。

```
正确率: 0.856
f_score: 0.429
```

可见虽然有提升，但是提升的很有限，随后我查看了投票后仍然错误的情况，发现在大多数情况下都为三个模型同时答错，仅有少量情况存在某一个模型独自答对的情况。

问题思考

训练的停止时机

- 一个最为直接的方法是固定epoch数，但这样会存在两个问题，当给定的epoch数过多时，会对训练数据过拟合，导致模型泛化能力下降；当epoch数过少时，则会出现欠拟合的情况。
- **使用validation数据集:** 一个较为合适的方法是基于validation来设置早停的机制，我的实现方法为，每训练完一个epoch就让模型在validation数据集上测试一遍，并将结果的正确率与最佳正确率作对比，如果连续 `early_stopping`（超参）次没有增加，则停止运行，并保存在validation集上运行结果最好的模型。这种早停机制的使用可以使得模型在欠拟合时继续训练，而过拟合后及时停止。
- 对上文中0.1学习率的解答，由于0.1这个学习率过大，即使使用带有学习率衰减的优化器Adam依然无法保证在前几轮中取得较好的训练效果，故此时由于`early_stopping`的存在，就早早停止了训练，故使用这种早停机制也可以帮助排除一些不合适的初始化超参。

实验参数的初始化

- 对于nn.embdding层，可以使用提供的词向量模型进行初始化，如此做可以很大程度上减少模型训练需要的epoch数

过拟合问题的解决方式

- 首先最为直接的方法就是降低模型的复杂程度，当模型过于复杂，参数量过多时，就会容易对训练数据过拟合。
- 如果不想降低模型的参数量，则可以在适当的地方增加Dropout层，如此做可以在训练时不会每个batch都更新所有神经元的参数，可以起到防止过拟合的作用
- 使用早停机制，早停机制的使用可以使得训练过程自动地在过拟合以后停止，且保存泛化能力最好的模型，从而在模型架构不变的情况下找到没有发生过拟合的最佳参数。

不同神经网络之间的优缺点

MLP

- 优点： 模型结构简单，参数量相对较少，训练速度较快
- 缺点： 泛化能力较差，对于任务的解决程度不如其余较为复杂的模型

TextCnn

- 优点： 经过测试，即使不使用已有预训练好的词向量模型也可以很快（指使用较少的epoch数）得到很好的训练效果（即由于其分辨图片的能力很强，对于词向量质量的依赖性不是很大），且最终正确率较高。且实验过程中过拟合现象并不严重。
- 缺点： 参数量较大，训练速度较慢。

LSTM

- 优点： 由于其记忆遗忘机制的优势，可以达到最高的正确率，适合处理序列化的任务。
- 缺点： 很容易对训练数据过拟合，且经过测试，如果不使用预训练好的词向量初始化embdding层，需要较多的epoch数模型能力才会涌现。

心得体会

- 搞懂了一直不是很明白的LSTM架构
- 对于训练时各项参数调节，训练机制的设计有了一定程度的了解