人工智能实践

实验报告

（2022学年秋季学期）

教学班级	计科2班	专业（方向）	计算机科学与技术
学号	**20337263**	姓名	**俞泽斌**

一、 实验题目

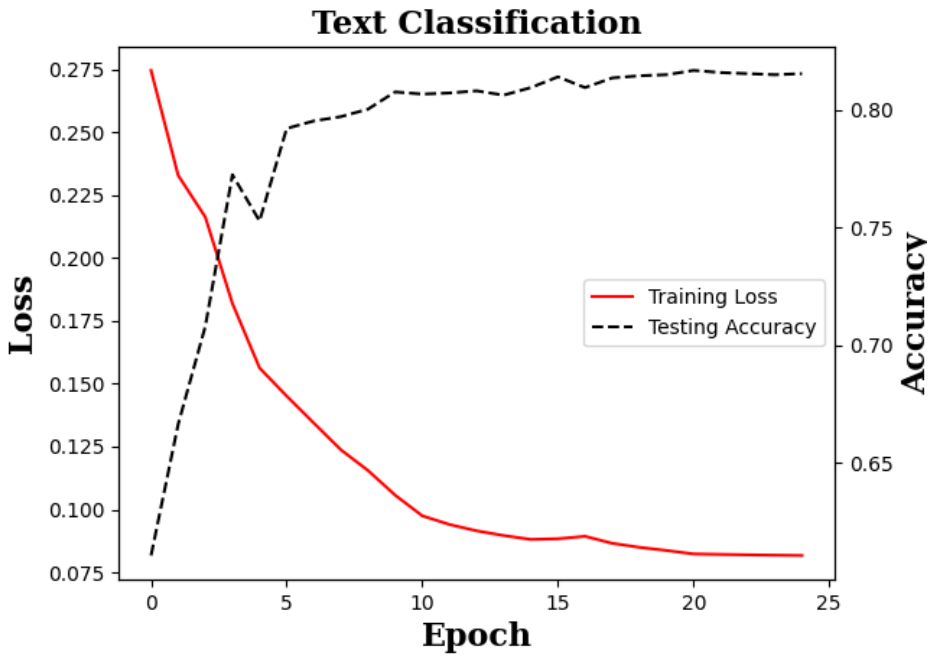
运行LSTM和TRANSFORMER新闻分类程序（源程序请在课程群下载），并对比实验结果。以下选作：尝试修改程序（如增加TRANSFORMER深度，修改网络中的残差定义、修改激活函数、修改超参数等），并对比修改后与修改前的结果。

二、 实验内容

1、程序运行

LSTM：

执行后得到结果为下图



具体的正确率

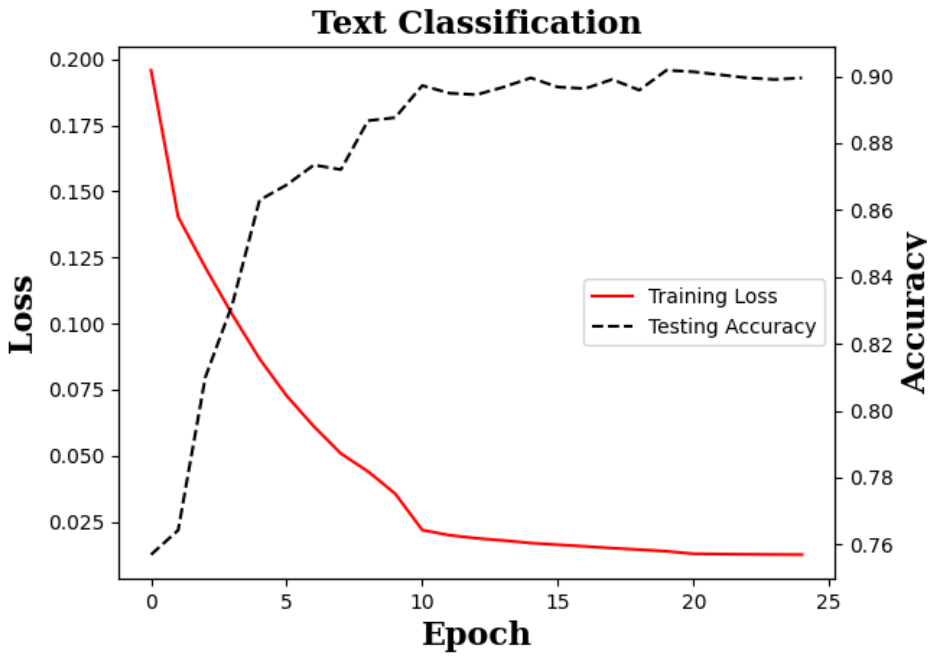
```
[Epoch:   0/ 25] Training Loss: 0.274, Testing Loss: 0.238, Training Acc: 0.538, Testing Acc: 0.611
[Epoch:   1/ 25] Training Loss: 0.233, Testing Loss: 0.214, Training Acc: 0.606, Testing Acc: 0.667
[Epoch:   2/ 25] Training Loss: 0.216, Testing Loss: 0.186, Training Acc: 0.645, Testing Acc: 0.708
[Epoch:   3/ 25] Training Loss: 0.182, Testing Loss: 0.156, Training Acc: 0.716, Testing Acc: 0.772
[Epoch:   4/ 25] Training Loss: 0.156, Testing Loss: 0.154, Training Acc: 0.757, Testing Acc: 0.753
[Epoch:   5/ 25] Training Loss: 0.145, Testing Loss: 0.137, Training Acc: 0.773, Testing Acc: 0.792
[Epoch:   6/ 25] Training Loss: 0.134, Testing Loss: 0.131, Training Acc: 0.783, Testing Acc: 0.795
[Epoch:   7/ 25] Training Loss: 0.124, Testing Loss: 0.128, Training Acc: 0.791, Testing Acc: 0.797
[Epoch:   8/ 25] Training Loss: 0.116, Testing Loss: 0.120, Training Acc: 0.802, Testing Acc: 0.800
[Epoch:   9/ 25] Training Loss: 0.106, Testing Loss: 0.115, Training Acc: 0.818, Testing Acc: 0.808
```

```
[Epoch:  18/ 25] Training Loss: 0.085, Testing Loss: 0.112, Training Acc: 0.856, Testing Acc: 0.815
[Epoch:  19/ 25] Training Loss: 0.084, Testing Loss: 0.111, Training Acc: 0.859, Testing Acc: 0.815
[Epoch:  20/ 25] Training Loss: 0.082, Testing Loss: 0.112, Training Acc: 0.862, Testing Acc: 0.817
[Epoch:  21/ 25] Training Loss: 0.082, Testing Loss: 0.113, Training Acc: 0.861, Testing Acc: 0.816
[Epoch:  22/ 25] Training Loss: 0.082, Testing Loss: 0.113, Training Acc: 0.862, Testing Acc: 0.815
[Epoch:  23/ 25] Training Loss: 0.082, Testing Loss: 0.113, Training Acc: 0.862, Testing Acc: 0.815
[Epoch:  24/ 25] Training Loss: 0.082, Testing Loss: 0.113, Training Acc: 0.862, Testing Acc: 0.815
Figure figure/LSTM_classifier_23-15-08-12.png is saved.
```

只截取了其中的一部分数据，当然我们也能通过曲线很清晰地看出来loss的递减以及正确率的提高，说明整个LSTM在不断迭代向最优值接近

Transformer：

执行后得到结果为下图



具体的正确率

```
[Epoch:   0/ 25] Training Loss: 0.196, Testing Loss: 0.141, Training Acc: 0.667, Testing Acc: 0.757
[Epoch:   1/ 25] Training Loss: 0.140, Testing Loss: 0.136, Training Acc: 0.753, Testing Acc: 0.764
[Epoch:   2/ 25] Training Loss: 0.121, Testing Loss: 0.110, Training Acc: 0.790, Testing Acc: 0.810
[Epoch:   3/ 25] Training Loss: 0.103, Testing Loss: 0.103, Training Acc: 0.823, Testing Acc: 0.832
[Epoch:   4/ 25] Training Loss: 0.087, Testing Loss: 0.089, Training Acc: 0.856, Testing Acc: 0.863
[Epoch:   5/ 25] Training Loss: 0.073, Testing Loss: 0.085, Training Acc: 0.882, Testing Acc: 0.868
[Epoch:   6/ 25] Training Loss: 0.061, Testing Loss: 0.082, Training Acc: 0.898, Testing Acc: 0.873
[Epoch:   7/ 25] Training Loss: 0.051, Testing Loss: 0.080, Training Acc: 0.914, Testing Acc: 0.872
[Epoch:   8/ 25] Training Loss: 0.044, Testing Loss: 0.076, Training Acc: 0.927, Testing Acc: 0.887
```

```
[Epoch:  14/ 25] Training Loss: 0.017, Testing Loss: 0.073, Training Acc: 0.978, Testing Acc: 0.899
[Epoch:  15/ 25] Training Loss: 0.016, Testing Loss: 0.076, Training Acc: 0.980, Testing Acc: 0.897
[Epoch:  16/ 25] Training Loss: 0.016, Testing Loss: 0.076, Training Acc: 0.980, Testing Acc: 0.896
[Epoch:  17/ 25] Training Loss: 0.015, Testing Loss: 0.074, Training Acc: 0.983, Testing Acc: 0.899
[Epoch:  18/ 25] Training Loss: 0.014, Testing Loss: 0.077, Training Acc: 0.984, Testing Acc: 0.896
[Epoch:  19/ 25] Training Loss: 0.014, Testing Loss: 0.075, Training Acc: 0.985, Testing Acc: 0.902
[Epoch:  20/ 25] Training Loss: 0.013, Testing Loss: 0.076, Training Acc: 0.987, Testing Acc: 0.901
[Epoch:  21/ 25] Training Loss: 0.013, Testing Loss: 0.076, Training Acc: 0.987, Testing Acc: 0.900
[Epoch:  22/ 25] Training Loss: 0.013, Testing Loss: 0.076, Training Acc: 0.986, Testing Acc: 0.899
[Epoch:  23/ 25] Training Loss: 0.013, Testing Loss: 0.076, Training Acc: 0.986, Testing Acc: 0.899
[Epoch:  24/ 25] Training Loss: 0.013, Testing Loss: 0.077, Training Acc: 0.986, Testing Acc: 0.899
```

只截取了其中的一部分数据，当然我们也能通过曲线很清晰地看出来loss的递减以及正确率的提高，说明整个transformer在不断迭代向最优值接近

与之前的LSTM进行对比，可以明显地发现transformer所得到的的初值更好，并且收敛速度相对较快，在第14次时已经基本处于收敛状态，只有一些微小的调整，而LSTM得到的loss就比较高，收敛速度相对较慢，最后的正确率也比transformer小一点，同时因为训练有一定的误差存在，所以在最后的几次训练中存在acc偶然出现降低的情况也在合理范围内，但都是不断优化 的过程

2、程序修改

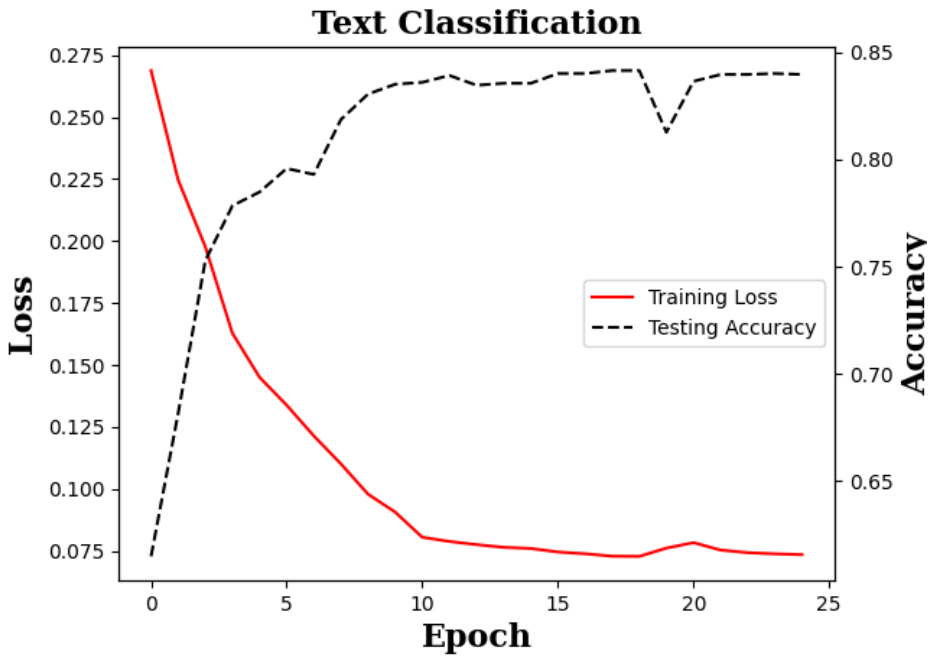
LSTM：

（1）维度修改：

将初始维度改为128，隐藏层维度改为64

```
    embedding_dim = 128  #token转换的维度，把token映射成96维度的向量。embedding layer
    hidden_dim = 64  #隐藏状态的维度
    sentence_len = 32 #多于32就截断，少于32就补0
```

执行结果




```
[Epoch:   0/ 25] Training Loss: 0.269, Testing Loss: 0.230, Training Acc: 0.556, Testing Acc: 0.615
[Epoch:   1/ 25] Training Loss: 0.225, Testing Loss: 0.203, Training Acc: 0.628, Testing Acc: 0.682
[Epoch:   2/ 25] Training Loss: 0.198, Testing Loss: 0.171, Training Acc: 0.674, Testing Acc: 0.753
[Epoch:   3/ 25] Training Loss: 0.163, Testing Loss: 0.151, Training Acc: 0.745, Testing Acc: 0.778
[Epoch:   4/ 25] Training Loss: 0.145, Testing Loss: 0.142, Training Acc: 0.772, Testing Acc: 0.785
[Epoch:   5/ 25] Training Loss: 0.134, Testing Loss: 0.129, Training Acc: 0.783, Testing Acc: 0.796
[Epoch:   6/ 25] Training Loss: 0.121, Testing Loss: 0.129, Training Acc: 0.794, Testing Acc: 0.793
[Epoch:   7/ 25] Training Loss: 0.110, Testing Loss: 0.111, Training Acc: 0.816, Testing Acc: 0.819
[Epoch:   8/ 25] Training Loss: 0.098, Testing Loss: 0.102, Training Acc: 0.841, Testing Acc: 0.831
```

```
[Epoch:  14/ 25] Training Loss: 0.076, Testing Loss: 0.096, Training Acc: 0.878, Testing Acc: 0.836
[Epoch:  15/ 25] Training Loss: 0.075, Testing Loss: 0.095, Training Acc: 0.879, Testing Acc: 0.840
[Epoch:  16/ 25] Training Loss: 0.074, Testing Loss: 0.095, Training Acc: 0.880, Testing Acc: 0.840
[Epoch:  17/ 25] Training Loss: 0.073, Testing Loss: 0.094, Training Acc: 0.881, Testing Acc: 0.841
[Epoch:  18/ 25] Training Loss: 0.073, Testing Loss: 0.096, Training Acc: 0.881, Testing Acc: 0.841
[Epoch:  19/ 25] Training Loss: 0.076, Testing Loss: 0.104, Training Acc: 0.874, Testing Acc: 0.813
[Epoch:  20/ 25] Training Loss: 0.078, Testing Loss: 0.100, Training Acc: 0.863, Testing Acc: 0.836
[Epoch:  21/ 25] Training Loss: 0.075, Testing Loss: 0.098, Training Acc: 0.881, Testing Acc: 0.840
[Epoch:  22/ 25] Training Loss: 0.074, Testing Loss: 0.098, Training Acc: 0.882, Testing Acc: 0.840
[Epoch:  23/ 25] Training Loss: 0.074, Testing Loss: 0.098, Training Acc: 0.882, Testing Acc: 0.840
[Epoch:  24/ 25] Training Loss: 0.073, Testing Loss: 0.098, Training Acc: 0.882, Testing Acc: 0.840
Figure figure/LSTM_classifier_23-15-45-01.png is saved.
```

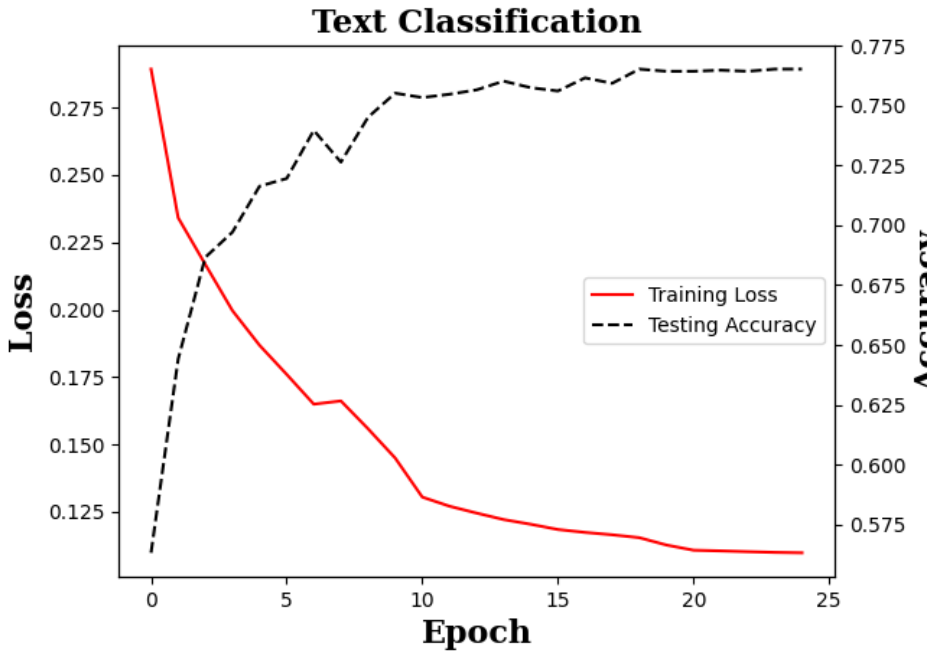
与上面最初的LSTM进行比较，可以看到此时的收敛速度变快，同时最后的结果也更好，具体体现在loss最终值更小以及正确率更高

但是由于维度地增加，会导致其中的计算复杂度增加，所以跑出这个结果所需要的时间更多

（2）激活函数修改

将原来输入门以及遗忘门的sigmoid激活函数改为relu

```
        for t in range(seq_sz):     # 将序列依次放入LSTM单元中
            x_t = x[:, t, :]
            # @ 为矩阵运算
            i_t = torch.relu(x_t @ self.w_i + h_t @ self.U_i + self.b_i)
            #i_t = torch.sigmoid(x_t @ self.w_i + h_t @ self.U_i + self.b_i)
# 输入门，调整输入的x_t及h_t比例  i_t = σ(W_i x_t+U_i h_(t−1)+b_i)
            f_t = torch.relu(x_t @ self.w_f + h_t @ self.U_f + self.b_f)
            #f_t = torch.sigmoid(x_t @ self.w_f + h_t @ self.U_f + self.b_f)
# 遗忘门，以往上一次的 f_t = σ(W_f x_t+U_f h_(t−1)+b_f )
            g_t = torch.tanh(x_t @ self.w_c + h_t @ self.U_c + self.b_c)
 # 补给记忆,中间结果 c ̃t = tanh(W_c x_t+U_c h_(t−1)+b_c)
            o_t = torch.sigmoid(x_t @ self.w_o + h_t @ self.U_o + self.b_o)
# 输出门，用于生成输出
            c_t = f_t * c_t + i_t * g_t     # 计算下一个cell state， 记忆状态c_t+1
c_t = f_t⊙c_(t−1)+i_t⊙c ̃t \\ c ̃t = tanh(W_c x_t+U_c h_(t−1)+b_c)
            h_t = o_t * torch.tanh(c_t)     # 计算下一个隐藏状态， h_t+1
h_t=o_t⊙tanh(c_t)
```




```
[Epoch:   0/ 25] Training Loss: 0.289, Testing Loss: 0.243, Training Acc: 0.512, Testing Acc: 0.563
[Epoch:   1/ 25] Training Loss: 0.234, Testing Loss: 0.219, Training Acc: 0.596, Testing Acc: 0.644
[Epoch:   2/ 25] Training Loss: 0.217, Testing Loss: 0.202, Training Acc: 0.634, Testing Acc: 0.687
[Epoch:   3/ 25] Training Loss: 0.200, Testing Loss: 0.190, Training Acc: 0.667, Testing Acc: 0.697
[Epoch:   4/ 25] Training Loss: 0.187, Testing Loss: 0.179, Training Acc: 0.696, Testing Acc: 0.716
[Epoch:   5/ 25] Training Loss: 0.176, Testing Loss: 0.180, Training Acc: 0.711, Testing Acc: 0.720
[Epoch:   6/ 25] Training Loss: 0.165, Testing Loss: 0.165, Training Acc: 0.731, Testing Acc: 0.740
[Epoch:   7/ 25] Training Loss: 0.166, Testing Loss: 0.173, Training Acc: 0.727, Testing Acc: 0.726
[Epoch:   8/ 25] Training Loss: 0.156, Testing Loss: 0.163, Training Acc: 0.744, Testing Acc: 0.745
[Epoch:   9/ 25] Training Loss: 0.145, Testing Loss: 0.157, Training Acc: 0.757, Testing Acc: 0.755
```

```
[Epoch:  13/ 25] Training Loss: 0.122, Testing Loss: 0.152, Training Acc: 0.801, Testing Acc: 0.760
[Epoch:  14/ 25] Training Loss: 0.120, Testing Loss: 0.152, Training Acc: 0.801, Testing Acc: 0.757
[Epoch:  15/ 25] Training Loss: 0.118, Testing Loss: 0.153, Training Acc: 0.805, Testing Acc: 0.756
[Epoch:  16/ 25] Training Loss: 0.117, Testing Loss: 0.151, Training Acc: 0.809, Testing Acc: 0.762
[Epoch:  17/ 25] Training Loss: 0.116, Testing Loss: 0.152, Training Acc: 0.810, Testing Acc: 0.759
[Epoch:  18/ 25] Training Loss: 0.115, Testing Loss: 0.151, Training Acc: 0.813, Testing Acc: 0.765
[Epoch:  19/ 25] Training Loss: 0.113, Testing Loss: 0.151, Training Acc: 0.818, Testing Acc: 0.764
[Epoch:  20/ 25] Training Loss: 0.111, Testing Loss: 0.151, Training Acc: 0.820, Testing Acc: 0.764
[Epoch:  21/ 25] Training Loss: 0.111, Testing Loss: 0.151, Training Acc: 0.821, Testing Acc: 0.765
[Epoch:  22/ 25] Training Loss: 0.110, Testing Loss: 0.151, Training Acc: 0.821, Testing Acc: 0.764
[Epoch:  23/ 25] Training Loss: 0.110, Testing Loss: 0.151, Training Acc: 0.822, Testing Acc: 0.765
[Epoch:  24/ 25] Training Loss: 0.110, Testing Loss: 0.151, Training Acc: 0.822, Testing Acc: 0.765
```

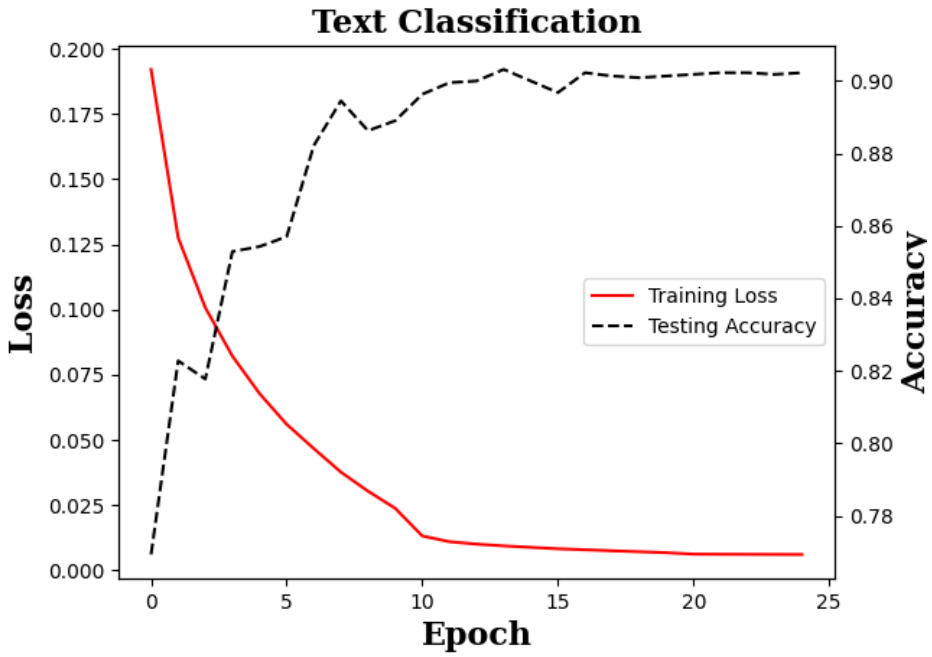
这个修改过后通过曲线我们也能看出来，曲线的收敛速度变慢了，通过具体数据，我们可以看到最后的loss在0.11左右，比原来未经修改的数据差了很多，正确率方面也比较差，收敛速度慢，具体可能是relu函数在第一象限内取值可能越来越大，导致最后的训练数据存在较大的偏差，也就是训练会被影响很大

Transformer:

（1）维度修改：

跟LSTM操作一样，这里采取的也是将初始维度改为128，隐藏层维度改为64

```
    embedding_dim = 128  #token转换的维度，把token映射成96维度的向量。embedding layer
    hidden_dim = 64  #隐藏状态的维度
    sentence_len = 32 #多于32就截断，少于32就补0
```




```
[Epoch:   0/ 25] Training Loss: 0.192, Testing Loss: 0.135, Training Acc: 0.682, Testing Acc: 0.769
[Epoch:   1/ 25] Training Loss: 0.128, Testing Loss: 0.108, Training Acc: 0.785, Testing Acc: 0.823
[Epoch:   2/ 25] Training Loss: 0.101, Testing Loss: 0.108, Training Acc: 0.829, Testing Acc: 0.818
[Epoch:   3/ 25] Training Loss: 0.082, Testing Loss: 0.092, Training Acc: 0.857, Testing Acc: 0.853
[Epoch:   4/ 25] Training Loss: 0.068, Testing Loss: 0.093, Training Acc: 0.882, Testing Acc: 0.854
[Epoch:   5/ 25] Training Loss: 0.056, Testing Loss: 0.094, Training Acc: 0.902, Testing Acc: 0.857
[Epoch:   6/ 25] Training Loss: 0.047, Testing Loss: 0.078, Training Acc: 0.927, Testing Acc: 0.882
[Epoch:   7/ 25] Training Loss: 0.038, Testing Loss: 0.078, Training Acc: 0.937, Testing Acc: 0.894
[Epoch:   8/ 25] Training Loss: 0.030, Testing Loss: 0.080, Training Acc: 0.951, Testing Acc: 0.886
[Epoch:   9/ 25] Training Loss: 0.024, Testing Loss: 0.082, Training Acc: 0.960, Testing Acc: 0.889
```

```
[Epoch:  16/ 25] Training Loss: 0.008, Testing Loss: 0.077, Training Acc: 0.991, Testing Acc: 0.902
[Epoch:  17/ 25] Training Loss: 0.008, Testing Loss: 0.078, Training Acc: 0.992, Testing Acc: 0.901
[Epoch:  18/ 25] Training Loss: 0.007, Testing Loss: 0.079, Training Acc: 0.992, Testing Acc: 0.901
[Epoch:  19/ 25] Training Loss: 0.007, Testing Loss: 0.079, Training Acc: 0.993, Testing Acc: 0.901
[Epoch:  20/ 25] Training Loss: 0.006, Testing Loss: 0.080, Training Acc: 0.994, Testing Acc: 0.902
[Epoch:  21/ 25] Training Loss: 0.006, Testing Loss: 0.080, Training Acc: 0.994, Testing Acc: 0.902
[Epoch:  22/ 25] Training Loss: 0.006, Testing Loss: 0.080, Training Acc: 0.994, Testing Acc: 0.902
[Epoch:  23/ 25] Training Loss: 0.006, Testing Loss: 0.080, Training Acc: 0.994, Testing Acc: 0.902
[Epoch:  24/ 25] Training Loss: 0.006, Testing Loss: 0.080, Training Acc: 0.994, Testing Acc: 0.902
```

与上面最初的Transformer进行比较，可以看到此时的收敛速度随迭代次数的增加而变快，最后的结果也更好，具体体现在loss最终值更小以及正确率更高但是由于维度地增加，会导致其中的计算复杂度增加，所以跑出这个结果所需要的时间更多。

其实正确率的提高只有一点点，loss的降低倒是挺明显，同时从曲线上来看的收敛速率变快