根据pytorch网站的信息，单个LSTM细胞的结构包括：

一个输入门

一个遗忘门

一个信息向量

新的细胞向量

一个输出门

以及一个新的隐藏状态

LSTM网络的初始化参数包含：input\_size代表输入特征的维度；hidden\_size代表隐藏状态的维度，num\_layers代表LSTM细胞的数量，或者说LSTM网络的层数；bias(bool)代表是否启用偏置，默认为使用；batch\_first(bool)代表输入输出张量是否将batch前置，默认为false；dropout(浮点)代表添加一层几率为dropout的dropout层，默认为0；bidirectional(bool)代表这个LSTM网络是否是双向的，默认为false；proj\_size代表将LSTM映射到指定的大小，默认为0.

输入包含input,(h\_0,c\_0)，序列输入 input表示时间序列数据，有三种可能的形状：

1.未批处理输入：(L, )

2.批处理输入：(L, N, )，其中L为序列长度，N为批大小，为输入特征维度。

可选的输入h\_0代表初始的隐藏状态，c\_0代表细胞的初始状态。

输出一个元组 (output,(h\_n, c\_n))：

序列输出output包含每个时间步的隐藏状态，其结构为

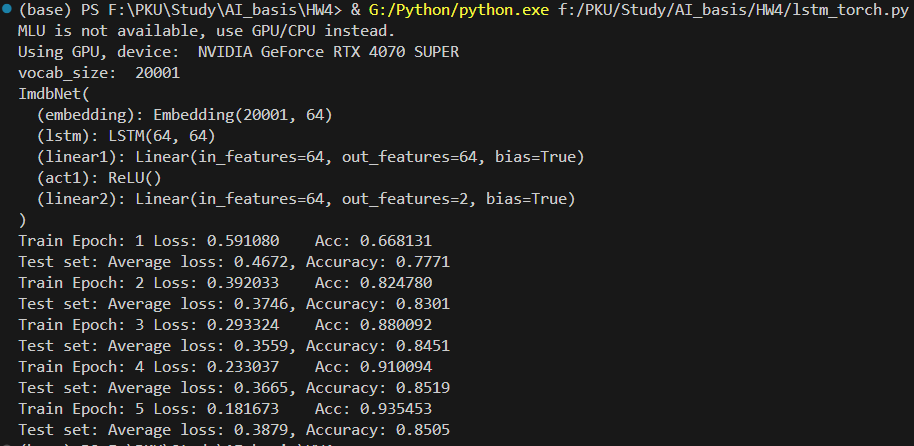
未批处理：(L, Hout)

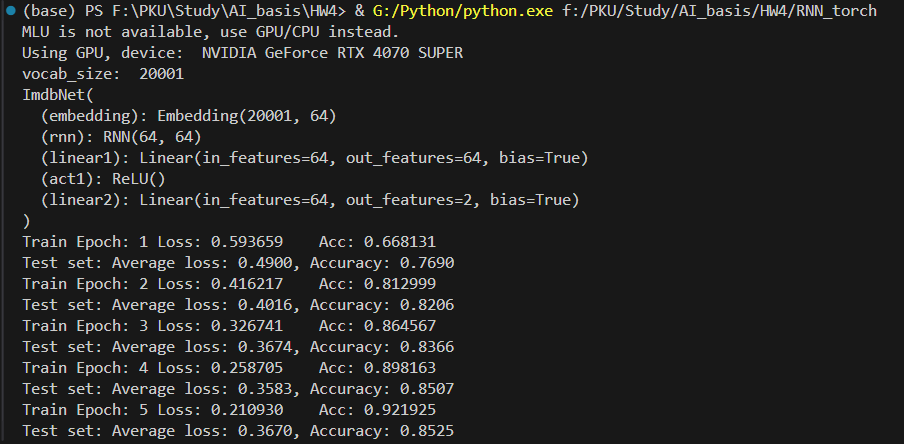
批处理 (L, N, Hout) 其中L为序列长度，N为批大小，为输入特征维度。

h\_n表示最终的隐藏状态，而c\_n代表最终的细胞状态。

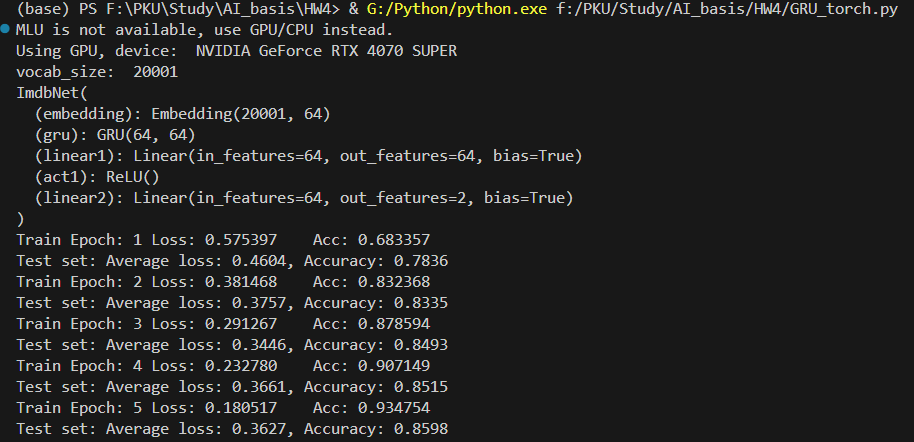
（在以上的分析中略去了bidirectional和batch\_first的影响，均设为默认值）

LSTM,RNN,GRU对比试验

LSTM的结果如图：

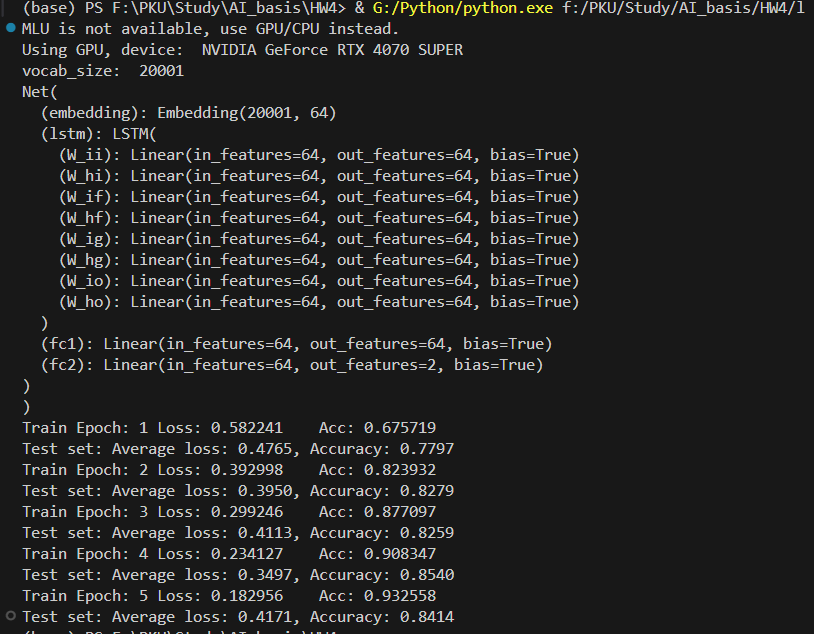
RNN的结果如图：

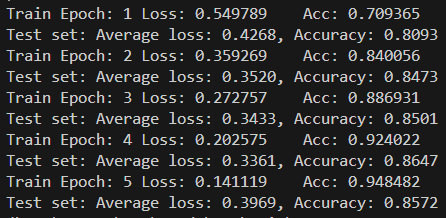
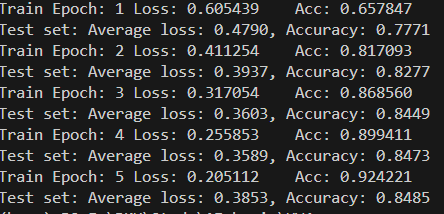
GRU的结果如图：

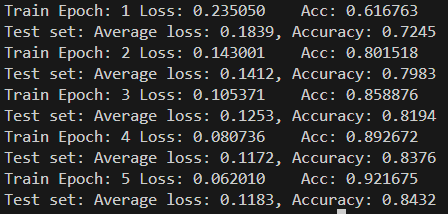


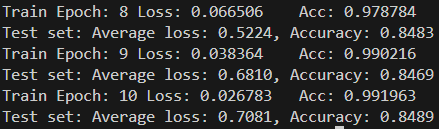
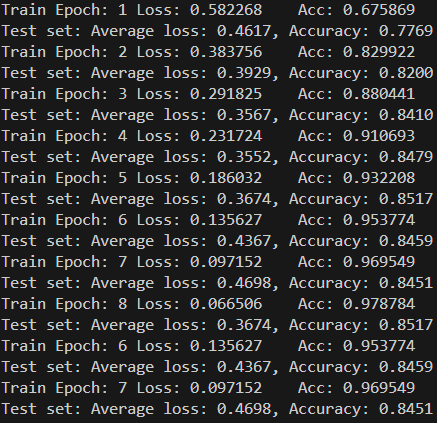
三种模型在给定数据集的情况下并没有本质上性能的差别，GRU略胜一筹。

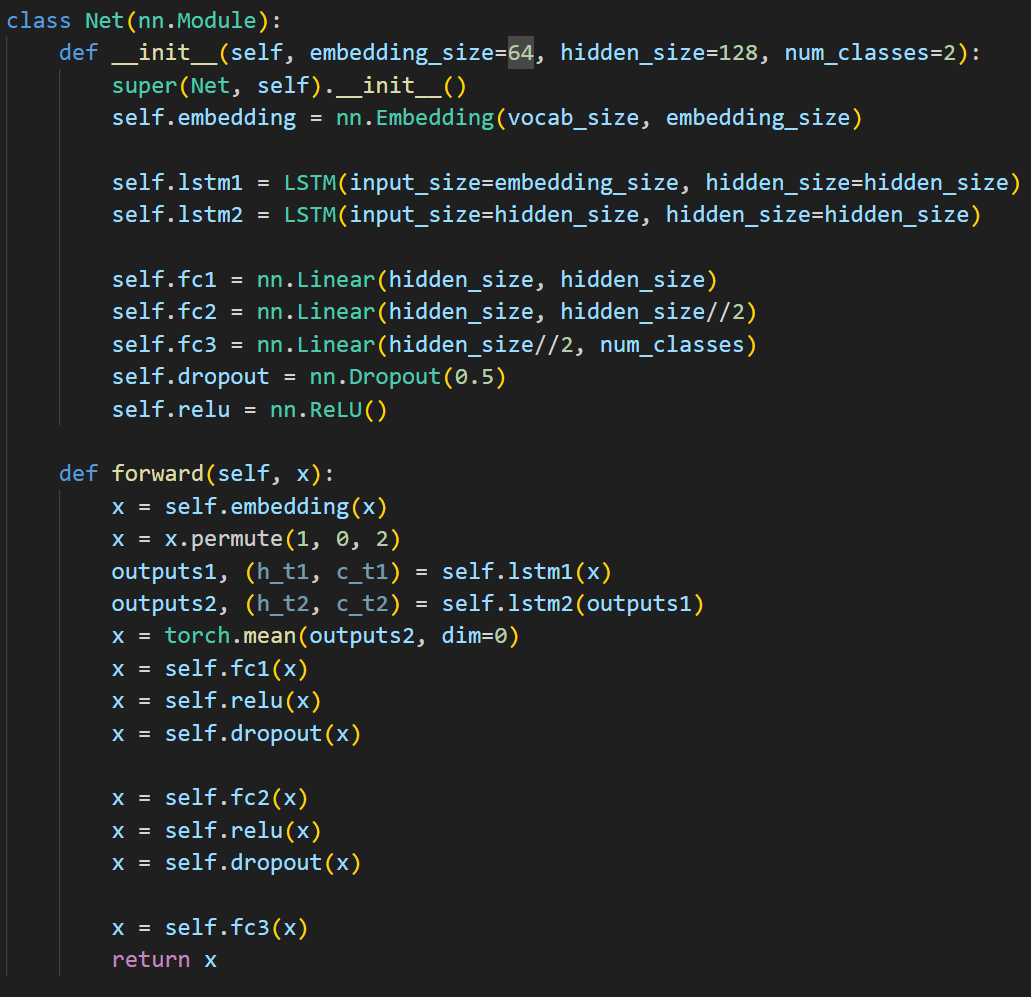
LSTM手写运行结果：（未改变任何网络结构、损失函数、超参数）

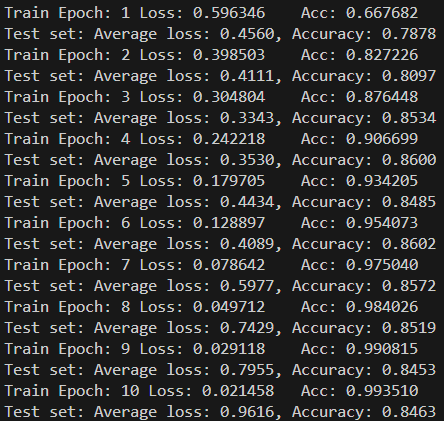


接着将隐藏层的维度从64改为256，观察到训练进展较快，但最终效果与64差不太多；再将其改为16，观察到整体效果与64相差无几。

若将损失函数改为MSE，观察到一开始的进展较慢，但由于CrossEntropy在后期准确率不增反降，导致最终效果也和交叉熵损失函数差不多。可能是因为二分类问题比较简单，交叉熵和MSE都能够相对较好的描述损失程度；且训练时间比较短，不同损失函数之间的差异还没有体现出来。

在原先的各个超参数保持不变的情况下，将训练的epoch数增加到10，观察到虽然epoch数增加到了10，但是性能在epoch=5之后，虽然对训练集的拟合程度不断上升，但是测试集的性能保持不变，甚至有下降趋势，可能是该网络结构过于简单，不能支撑多个epoch的训练，也可能是数据量不足且训练集和测试集之间存在一定差异。

因此将网络结构略作调整，改为双层lstm，加入更多的全连接层和激活函数，并且添加了dropout层，如图继续10个epoch的训练，此时观察到

测试集依旧保持在85%左右的准确率，说明后者可能是准确的。