Assingment2 RNN实现加法运算

task1 RNN简单实现加法运算

实现过程

1. 通过forward函数进行前向传播

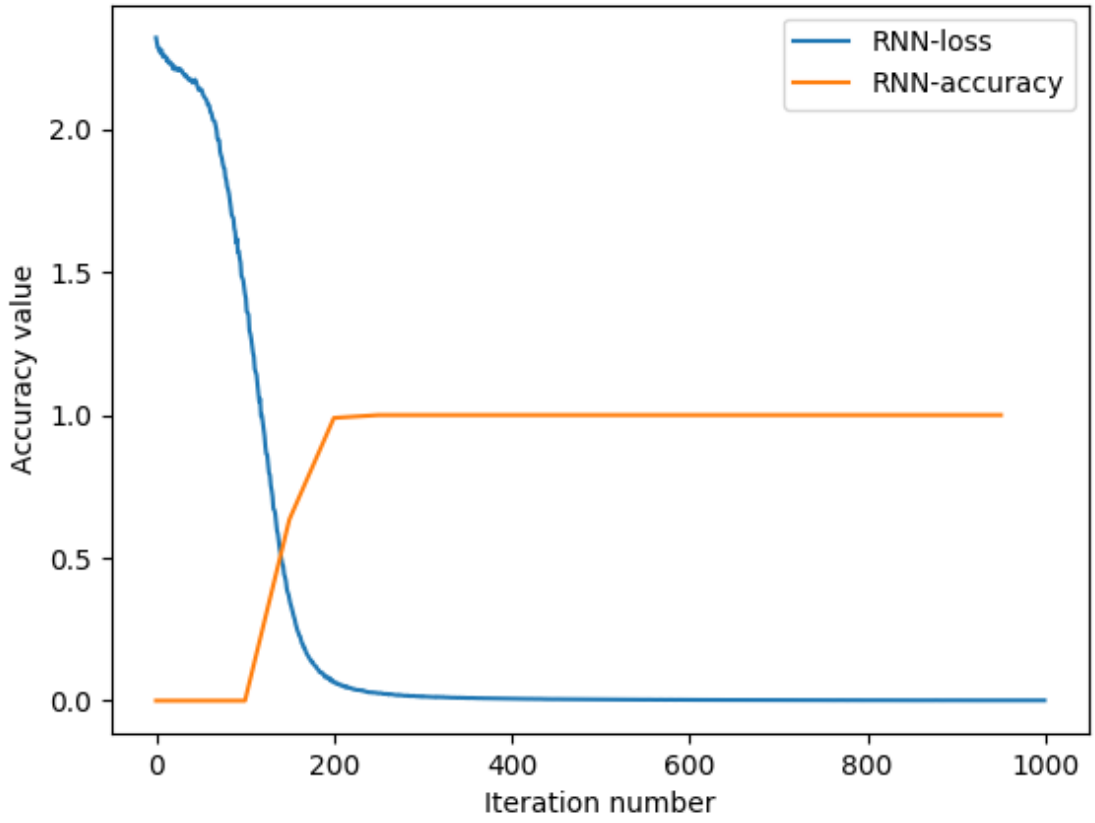
```
    #num1.shape=[200,11]
    num1=self.embed_layer(num1)
    num2=self.embed_layer(num2)
    #num1.shape=[200,11,32]
    #将每一位数字embedding到32维更高的嵌入空间中
    input = torch.cat((num1, num2), 2).transpose(0,1)
    #input.shape=[11,200,64]
    #将input 按照 seq_length=num_length,batch_size=200,input_size=64的顺序进行输入
    output, _ = self.rnn(input)
    #output.shape=[11,200,64]
    logits = self.dense(output)
    #将输出数字从高维dense回低维
    #logits.shape=[11, 200, 10]
    logits = logits.transpose(0,1).clone()
    #logits shape=[200,11,10]
    return logits
```

通过embed_layer将每一位数字嵌入到更高维的向量
每一个时间序列的输入为两个数的对应数字的数字向量的连接，数字的位数决定了输入序列的长度
将数据组织为 [seq_length=num_length,batch_size=200,input_size=64] 的顺序输入到RNN模型中
通过dense将output中的数字输出从高维映射为10维，每一维代表输出为该数字的概率

2. 损失函数
   预测数值的数字和真实数值数字之间的交叉熵
3. 训练
   batch_size设为200，每一个batch进行一次进行一次梯度回传，对参数进行更新，共进行1000个batch训练
4. 模型评估标准
   通过在测试集上的测试，预测数值和真实数值完全一样则为预测正确，来计算模型在测试集中的Accuracy进行模型效果评价

实现结果

下图展示了在1000个epoch中train-loss和test-accuracy的变化情况。
每一个batch都进行一次train loss的计算，每50个bacth计算一次test accuracy。

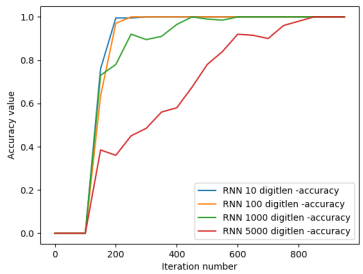


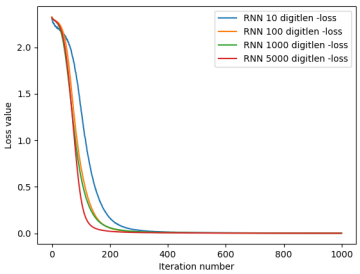
可以看出在100-200个batch中模型的能力提高到足以准确地计算加法。在0-400batch,模型的loss快速下降。由于train数据每一个batch都不同，所以不存在过拟合问题，在200个batch之后test accuracy恒为1。

task2 改变加法中数字长度 & RNN优化

1. 数字长度

通过改变数字长度来观察模型能力，取数字分别为10，100，1000，5000进行训练，结果如下。
可以看出随着数字的增加，模型在测试集上accuracy=1所需的训练次数也增加，但是模型loss的下降所需batch数量减小。





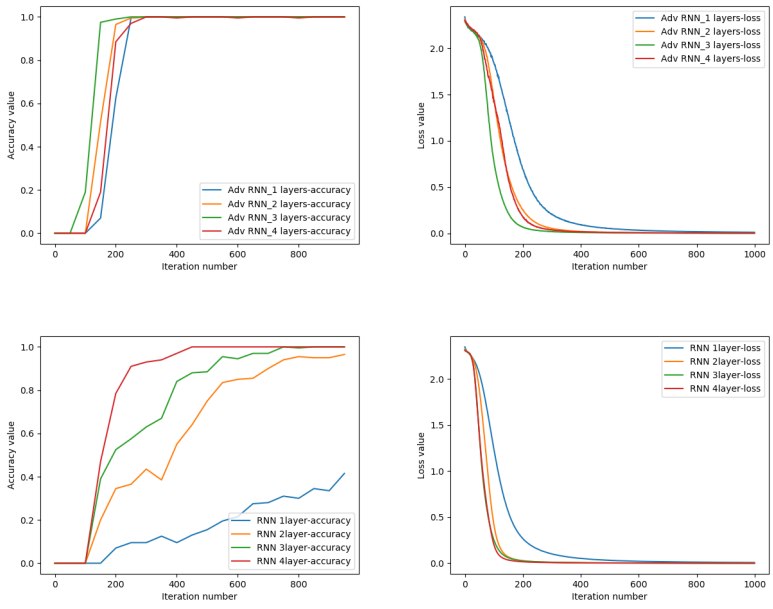
分析随着数字长度增加，loss下降需要的batch数量减少的原因是，数字长度增加，input序列增强，在每一轮的训练过程中有更多数据用于训练模型，所以loss下降速度变快。

分析随着数字长度增加，test Accuracy 上升需要的batch数量增加的原因是，数字长度增加，导致模型学习需要的能力增强，需要在更多位上保持完全正确性，导致需要训练的次数增加，需要的batch数量增加。

2.RNN优化

1. RNN layer

通过改变self.rnn层中RNN模型层数，进行模型效果优化，取layer=1，2，3，4进行训练，实验结果如下，在实验过程中，数字长度取值分别为10和5000，训练1000个batch。



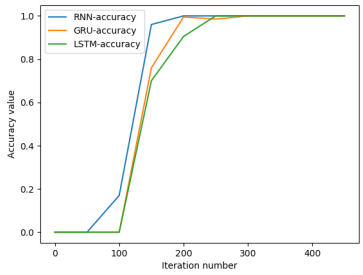
可以看出在两种数字长度情况下，最优的RNN模型layer数不同，在长度为100的数字训练过程中最优模型的RNN layer数为3，在长度为5000的数字训练过程中最优模型的RNN layer数为4。

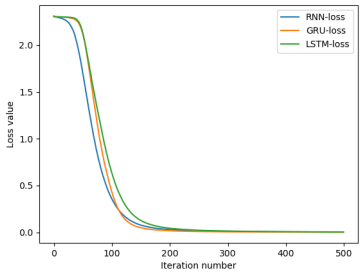
分析：

1. 由于长度为100数字相加的任务比较简单，在layer=3时RNN模型可以很好地完成任务，并且不会有过多的参数影响训练过程，所以在train loss下降和test accuracy增长方面layer=3都是表现较好的模型。
2. 长度为5000的数字相加任务较难，需要的参数量更多，才能更快地学好该任务，所以layer=4的表现更好。
3. 长度为5000的数字相加为什么loss不同layer train loss下降速度没有明显差异，而test Accuracy中layer4明显好于其他模型？分析可能是因为layer4由于拥有更多的参数，所以在学习的过程中更不容易过拟合，进而学习input和output的真实规律，所以在test上表现比其他模型要好。layer3和layer2同理，在test Accuracy上表现依次下降

2. RNN模型

通过改变self.rnn层中使用的具体RNN模型，分别测试nn.RNN和nn.LSTM，nn.GRU模型的效果。实验结果如下，在实验过程中数字长度取值为100，RNN层数均取值为3。



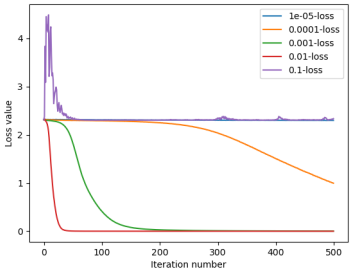


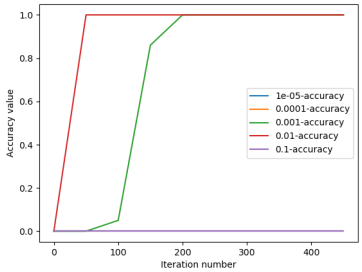
可以看出RNN模型的效果好于GRU，好于LSTM

分析：由于时间限制只在数字长度为100的数据上进行实验，分析实验结果，较为简单的RNN模型结果优于GRU优于最复杂的LSTM模型，可能是因为进行加法运算并不需要“长期”记忆，只需要记住进位信息就可以很好的完成任务，所以较为简单的模型已经可以很好的完成任务，而较为复杂的模型需要更多的数据来训练参数。

3. 学习率

通过改变学习率，观察RNN模型在不同学习率下，对该任务的学习能力和速度。实验结果如下，在实验过程中数字长度取值为100，RNN模型为nn.RNN,层数取为3。





可以看出在learning rate过大和过小时，模型的表现都不好。

分析：

1. 在learning rate过小时，lr=1e-5或1e-4导致模型收敛速度过慢，在500个epoch时仍未能收敛。
2. 在learning rate过大时，lr=1e-1会导致模型难以学习到合适的参数，loss无法降低，导致模型无法收敛

代码使用参数

```
 #默认模型
 python3 source.py --model pt --digit_len 10 --model RNN --lr 1e-3 --layer 3 --epoch 1000 --batch_size 200 --test_batch_size 200
```

digit_len 数字长度
model 使用模型 ('RNN','LSTM','GRU')
lr learning rate
layer 模型RNN部分层数
epoch 训练batch个数
batch_size batch size

test_batch_size 测试数据batch size

可以通过调用test1，test2，test3，test4函数完成上述实验