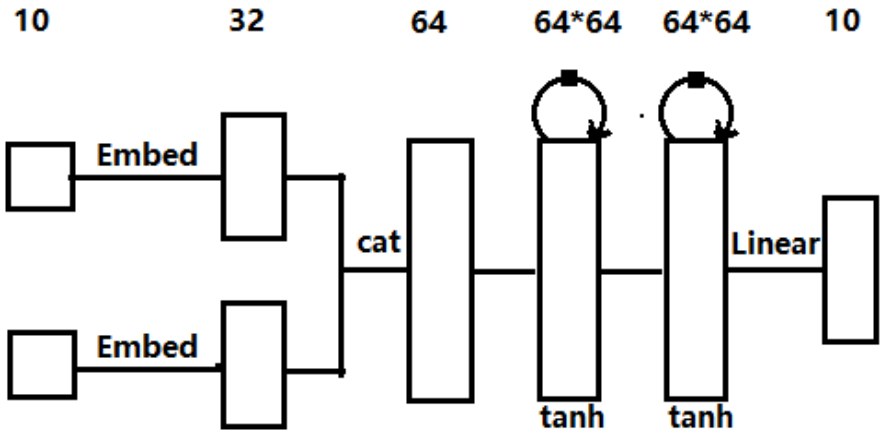
实验报告

实验目的

使用RNN来实现对两个十进制数的加法器；
改进网络参数，并改善网络结构

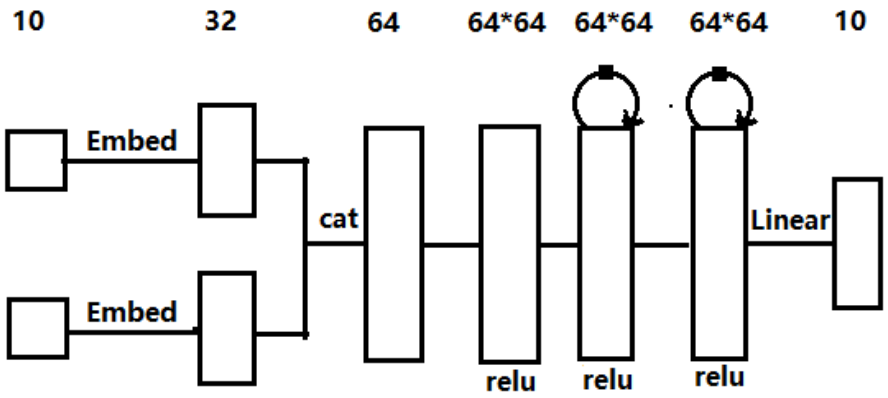
网络模型

普通网络模型



输入的两个10长度的one-hot张量先经过一个embed_layer得到两个32长度张量。
两个32长度张量经过一个cat操作合并得到一个64长度张量
该64长度张量经过两层64 × 64的RNN网络，得到一个64长度张量的输出
该64长度张量经过一个64 × 10的全连接层得到一个10长度张量作为输出

改进的网络模型



该模型在普通RNN的基础上，该模型在cat和RNN之间加上了一个64 × 64的全连接层，用relu作为激活函数

改进理由

考虑到在进行三个数的加法运算时，可以先对其中的两个数进行加法运算再让结果对另一个数进行加法运算。因此，这里先让两个加数的张量经过一个全　连接层作为一次加操作，再经过RNN网络对上一位的结果进行一次加操作。这样RNN网络所要学习的运算就从三个数的加法变成了两个数的加法，要学习的任务　更加简单。

IndRNN

在比较relu和tanh作为激活函数的时候，发现RNN默认都是tanh作为激活函数，同时发现存在使用relu作为激活函数的IndRNN模型。

在传统RNN中，输出结果$out = \sigma(Wx + Uh + b)$

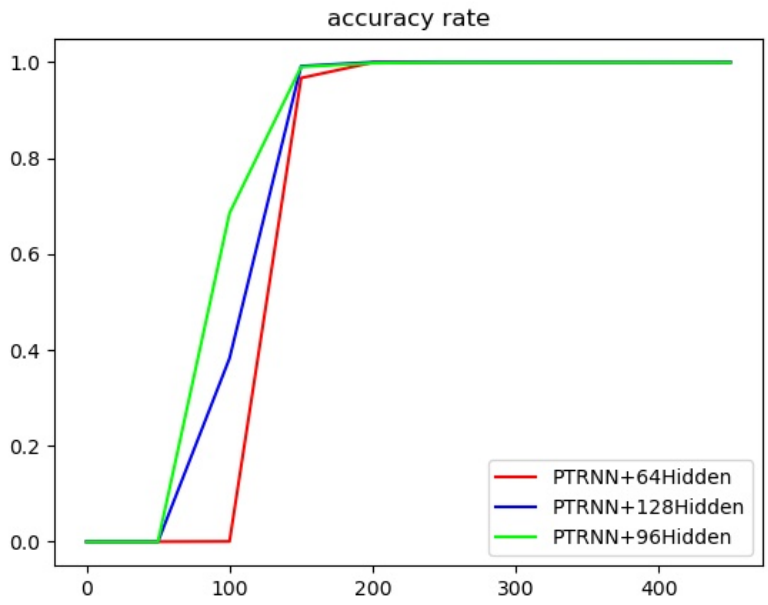
而在IndRNN中，输出结果$out = \sigma(Wx + U \bigodot h + b)$，其中$\bigodot$表示点积运算

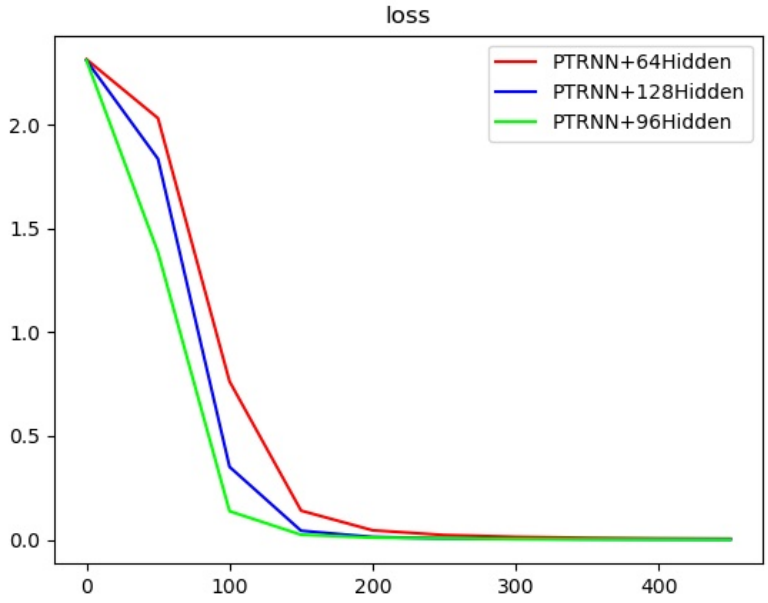
该网络处理比LTSM更长的序列信息，而且可以构建更深的网络。但在给任务中并没有使用IndRNN，原因会在测试结果中说明。

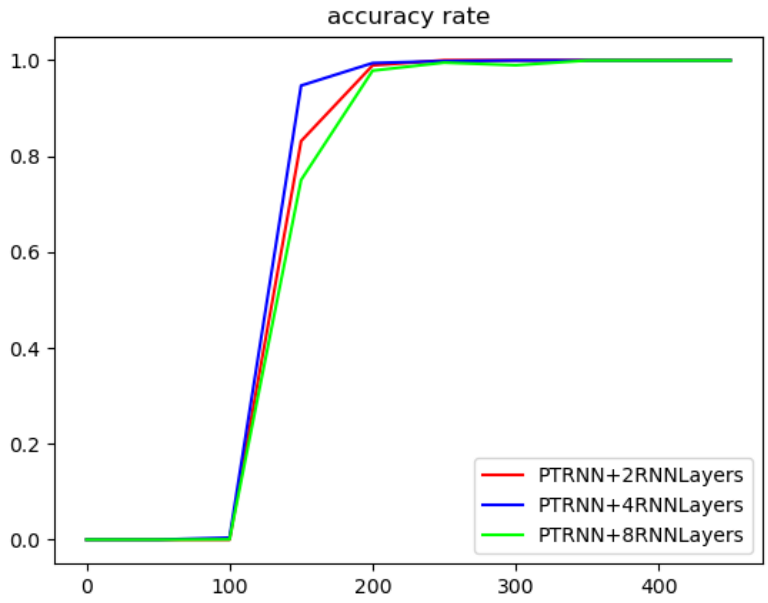
测试结果

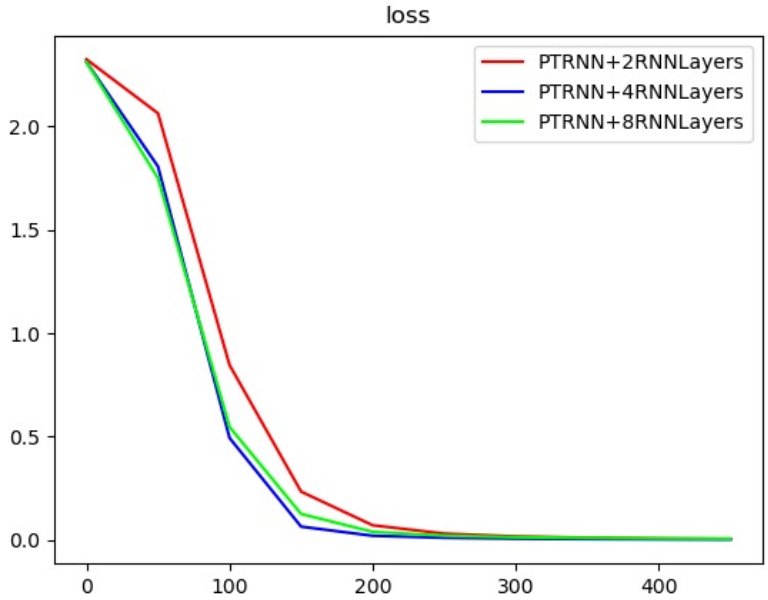
训练参数：进行2000次迭代，每次batch大小为200。对10000组数据进行测试。

由于加法位数较少时准确率很高，为了便于比较，默认条件下测试、训练的加法位数为50位。

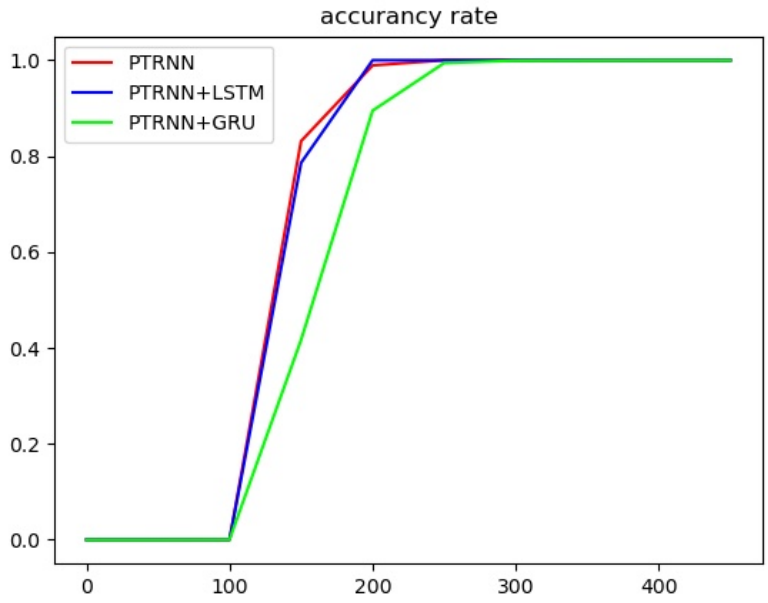

accuracy rate
1.0
0.8
0.6
0.4
0.2
0.0
0
100
200
300
400
PTRNN+64Hidden
PTRNN+128Hidden
PTRNN+96Hidden

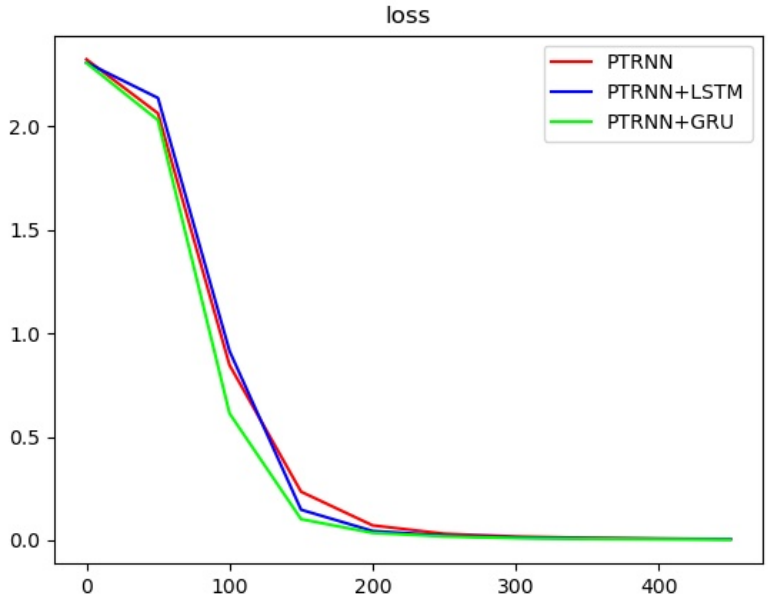

loss
2.0
1.5
1.0
0.5
0.0
0
100
200
300
400
PTRNN+64Hidden
PTRNN+128Hidden
PTRNN+96Hidden





分别扩大RNN隐藏层大小、增加RNN网络的层数，对模型进行测试。使用两种方法之后分别让模型用到更多的之前时刻时的结果、能进行更多的非线性变化，可以让模型的准确率变高，对损失函数降低方面也有较好的效果。另外，隐藏层大小为128的模型准确率、损失函数值都不如大小为96的模型，同样的情况也出现在了增加RNN的层数中。说明了需要学习的参数数量过多时会造成训练上的困难，需要更大的训练集才能训练出一个较为理想的参数。另外，训练较大规模的参数时，在测试中发现准确率、损失函数都出现了较大范围的波动，说明了参数过多时更容易受到训练数据的影响，也更容易出现过拟合的现象。





另外，由于在两数相加时，在随机数据中某位i上相加的结果很难影响到之后某一位相加的结果。因此，在随机数据中，长程依赖关系并不是十分明显。把普通模型中的RNN替换为LSTM和GRU之后，就可以发现使用有更长久记忆的模型在随机数据下并不能很好地提高模型地准确率。可能对于极端情况(9999... + 1)，长程依赖才会体现出其重要性。

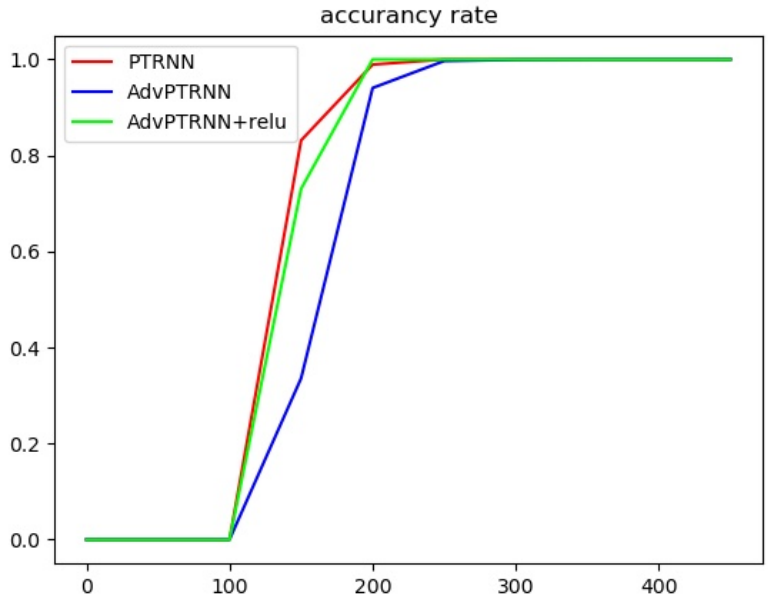
在对LSTM、GRU、RNN模型进行多位数字方面的准确率测试时，也发现了即使在更需要长期记忆的任务中，RNN仍然是一个十分理想的方法。

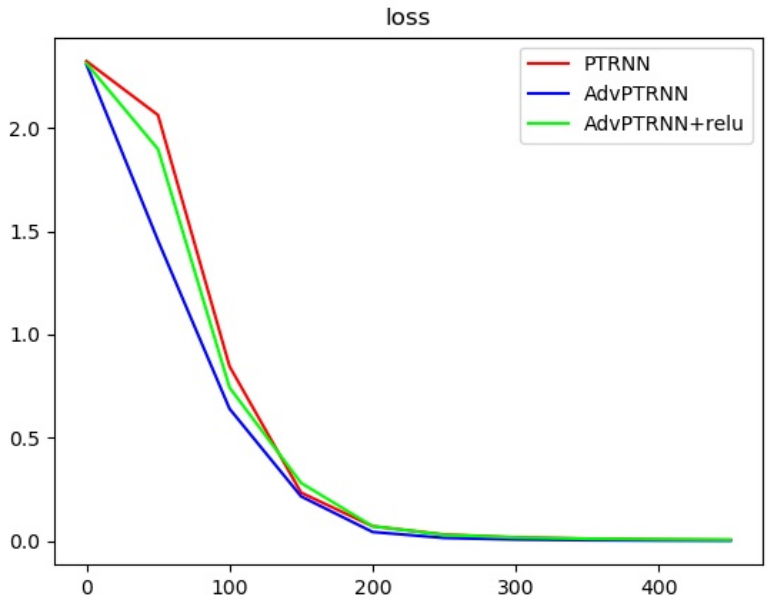
训练位数:6位,500次迭代,每次batch_size位200.

测试位数	RNN	LSTM	GRU
10	1.0	1.0	1
50	1.0	1.0	0.995
100	1.0	1.0	0.995
200	1.0	0.999	0.994

测试位数	RNN	LSTM	GRU
500	1.0	1.0	0.981
1000	1.0	0.999	0.968

可见使用RNN作为网络模型在这类简单的任务中有很好的效果.另外,GRU在该项测试中出现了较大的下降，推测是因为在该类任务中更新门、重置门都需要取到较大的值，在训练中GRU很难学到这么大的参数。





在测试中，发现改进模型在损失函数上能够以较小的参数增加(一个64×64的全连接层)，与测试数据有较高的拟合度。但是，在准确率上，改进模型并不能发挥很好的作用，而且当使用relu作为激励函数时，尽管会让损失值进一步降低，但也会导致准确率在较小的测试集中表现得较差。

由于ReLU在求导结果为1或0，在深层网络中即使有多层ReLU也不会减少导数结果。但在RNN中由于共享参数，导致求导时会出现权重矩阵W连乘的情况，如果W有一个大于1的特征值，会出现梯度爆炸的现象。因此，IndRNN把每层参数相互独立，在求导 $\frac{dJ_n}{dh_n,t} = \frac{dJ_n}{dh_n,t'} u_n^{(t'-t)} \prod_{k=t}^{t'-1} \sigma'_{n,k+1}$，可以通过调节$u$来防止出现梯度爆炸。

但是在测试中，使用IndRNN的模型测得的准确率十分低下，batch_size为200，加数位数为10位进行训练，发现分别迭代1000、2000、3000、4000次时的准确率为0.129、0.5174、0.8533、0.9366。可能是自己编写的模型中有错误，最终并未使用IndRNN作为改进的结果.

运行说明

默认运行命令：

```
python source.py
```

详细参数设置见source内部说明

参考文献

https://arxiv.org/pdf/1803.04831.pdf