Assignment 6

Neural Networks and Deep Learning

CSCI 5922

Fall 2017

Student: **Bo Cao**

bo.cao-1@colorado.edu
boca7588@colorado.edu
Github: https://github.com/BryanBo-Cao/neuralnets-deeplearning

Part 1

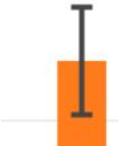
Set your code up to train a net given H and N. Each time you run the code, it should randomize the initial weights and generate a random training set of 10000 examples of length N. Also generate a random test set of 10000 examples of length N.

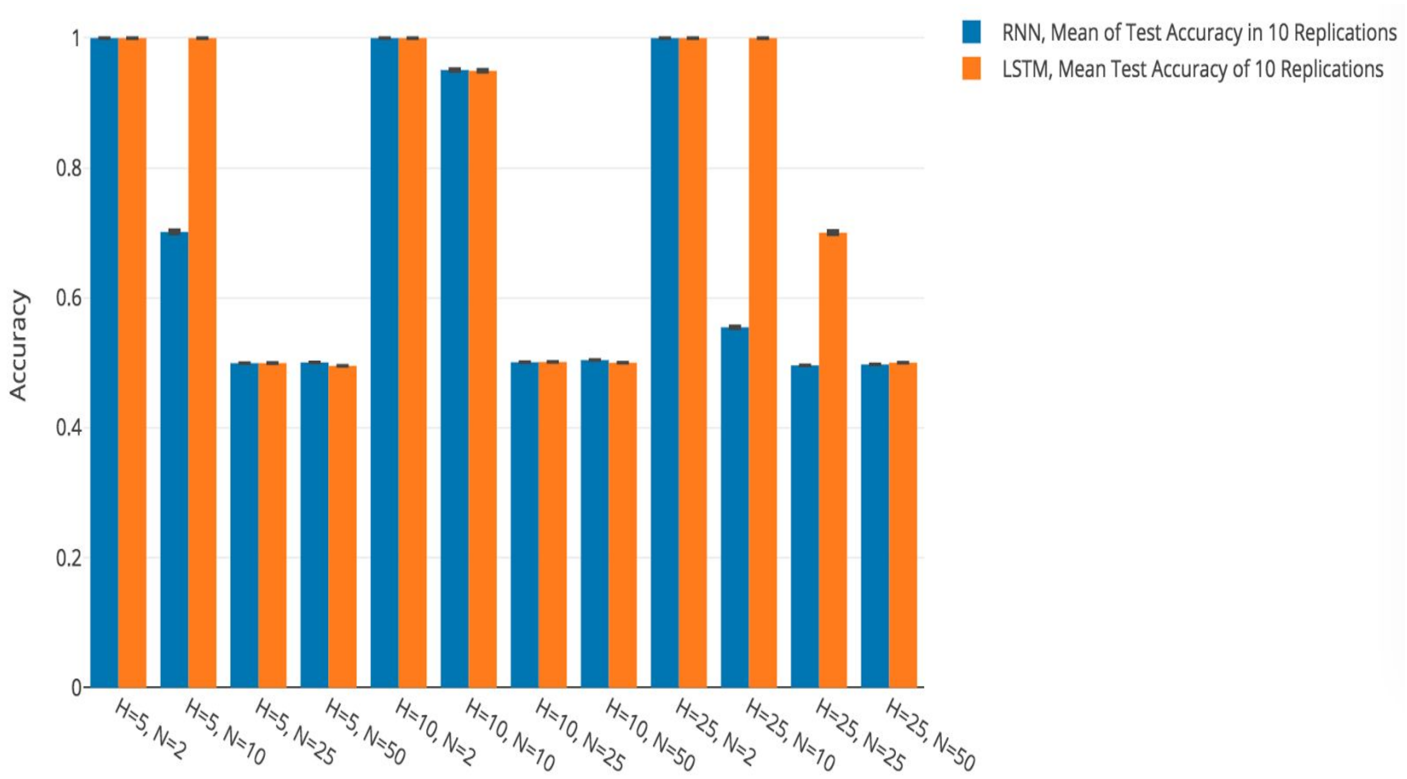
Train your net for H ∈ {5, 25} and for N ∈ {2, 10, 25, 50}. Use an RNN with tanh activation functions. For each combination of H and N, run 10 replications of your simulation.

Make a graph of mean % correct on the test set for the different values of H and N. I'll be more impressed if you plot not only the mean but also the standard error of the mean (= standard deviation of the 10 replications divided by sqrt(10)).

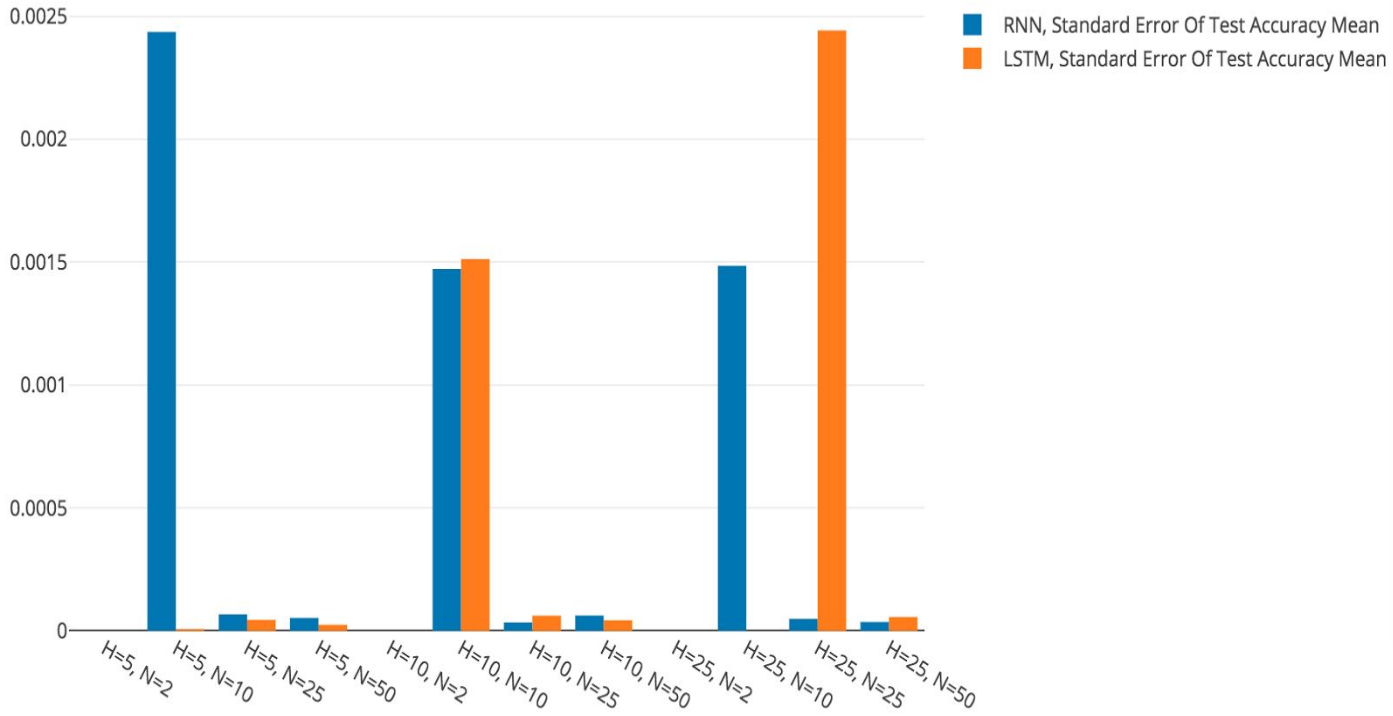
Part 2

Repeat the experiment of Part 1, but use LSTM neurons instead of standard tanh neurons in the recurrent layer.

Me:
The graph is shown below, results using **RNN** are shown in **blue** while results using **LSTM** are shown in **orange**. Each setting's parameters (Hidden Unit Number -- H and Length of the Sequence -- N) are shown in x-axis. In each setting, the **mean of the accuracy** on test set of 10 replications is displayed in the shape of **bar** while the **standard error** of the mean is shown in the shape like  .

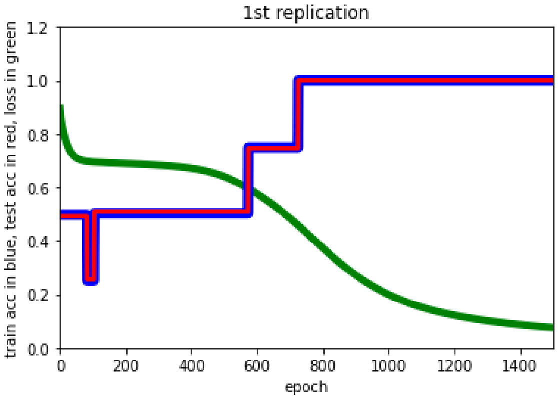


In addition, the standard error of Test Accuracy Mean is shown in the bar graph below.

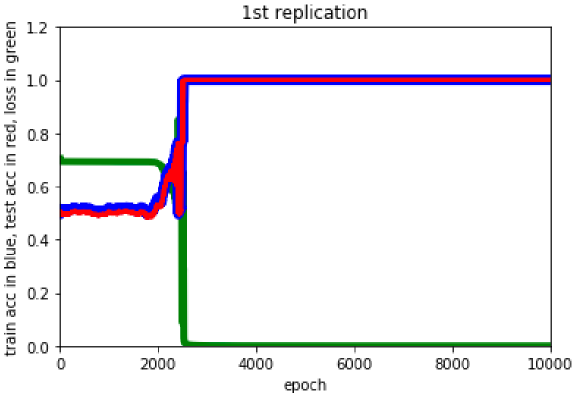


Setting	Neuron Type	Mean Test Accuracy of 10 Replications	Standard Error Of Test Accuracy Mean
H=5, N=2	RNN	1	0
	LSTM	1	0
H=5, N=10	RNN	0.70161	0.002436540276
	LSTM	0.99993	0.000002099990816
H=5, N=25	RNN	0.49971	0.00006719746627
	LSTM	0.49975	0.00004442352336
H=5, N=50	RNN	0.50083	0.00005271058064
	LSTM	0.49542	0.00002461626194
H=10, N=2	RNN	1	0
	LSTM	1	0
H=10, N=10	RNN	0.95085	0.001472168118
	LSTM	0.94958	0.001512600034
H=10, N=25	RNN	0.50109	0.00003448025091
	LSTM	0.50147	0.00006167343818
H=10, N=50	RNN	0.50432	0.00006225724239
	LSTM	0.50029	0.00004272805527
H=25, N=2	RNN	1	0
	LSTM	1	0
H=25, N=10	RNN	0.55488	0.001485280544
	LSTM	1	0
H=25, N=25	RNN	0.49628	0.00004903425463
	LSTM	0.70047	0.002442682385
H=25, N=50	RNN	0.4977	0.00003595554736
	LSTM	0.50041	0.00005641706288

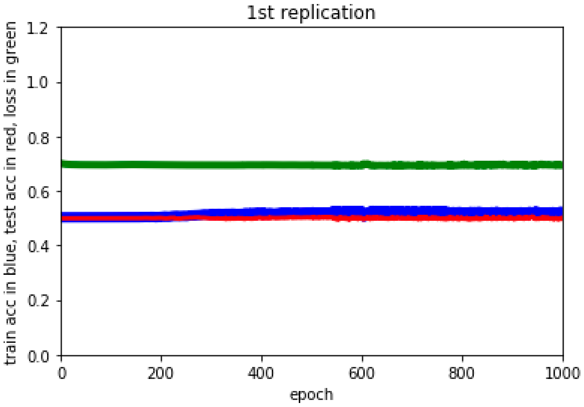
In the 1st replication of each setting, the **training** and **test accuracy** are displayed in **blue** and **red** respectively, as well as the **loss** are plotted in **green** as below:



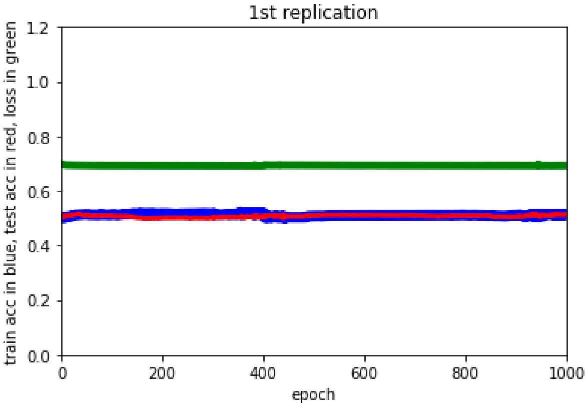
RNN H=5, N=2



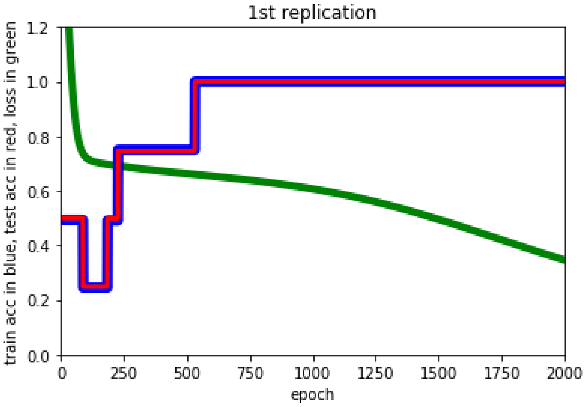
RNN H=5, N=10



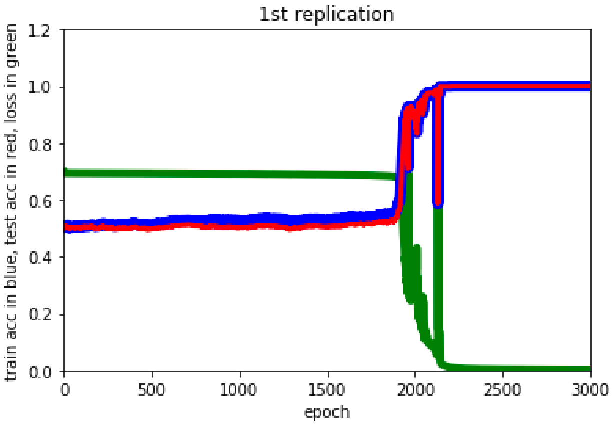
RNN H=5, N=25



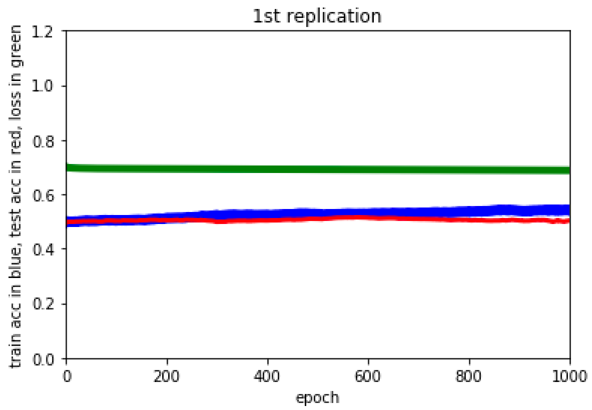
RNN H=5, N=50



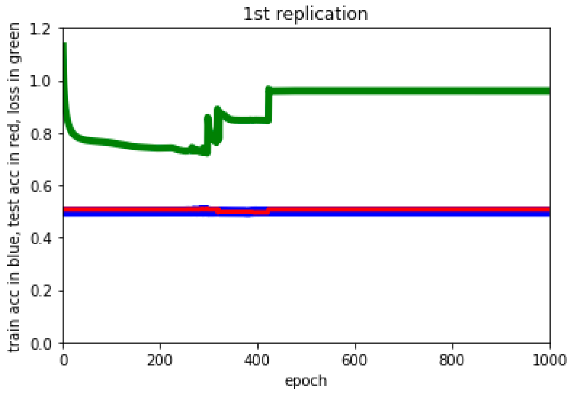
RNN H=10, N=2



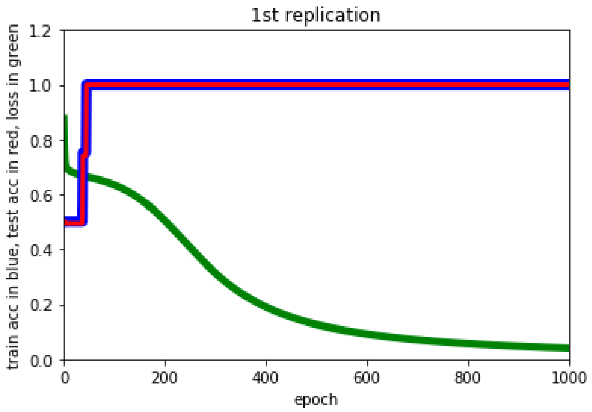
RNN H=10, N=10



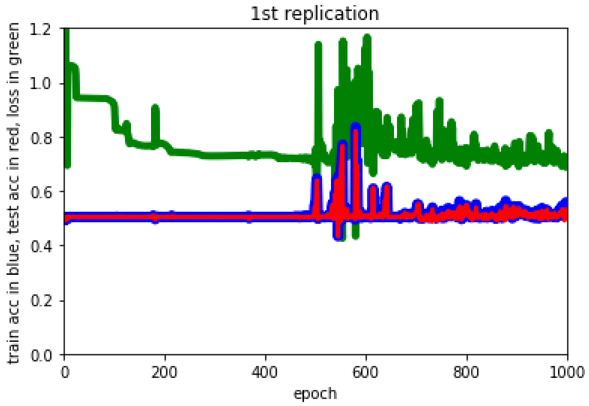
RNN H=10, N=25



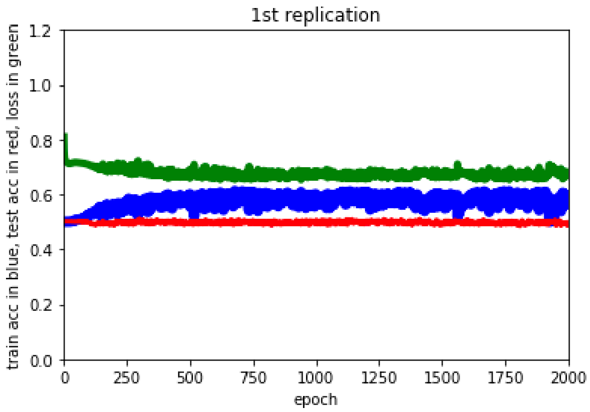
RNN H=10, N=50



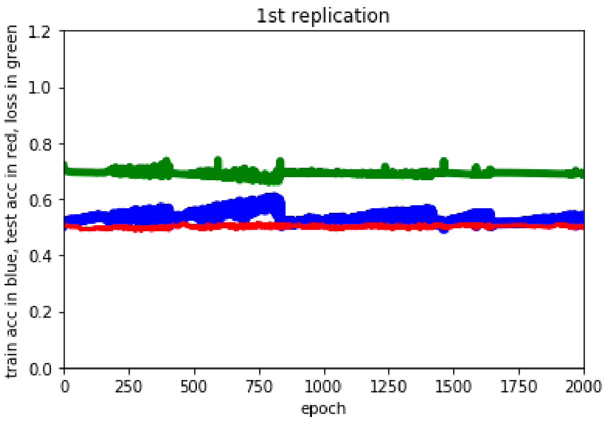
RNN H=25, N=2



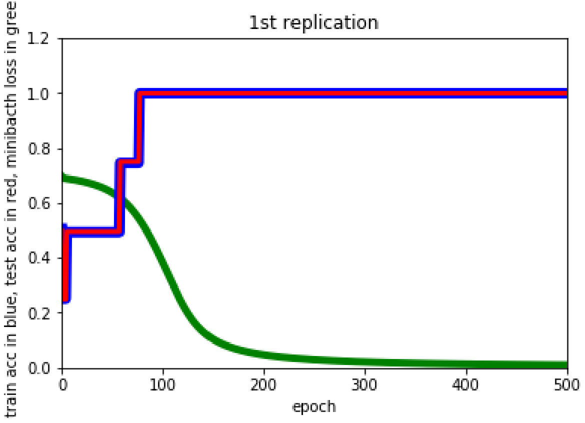
RNN H=25, N=10



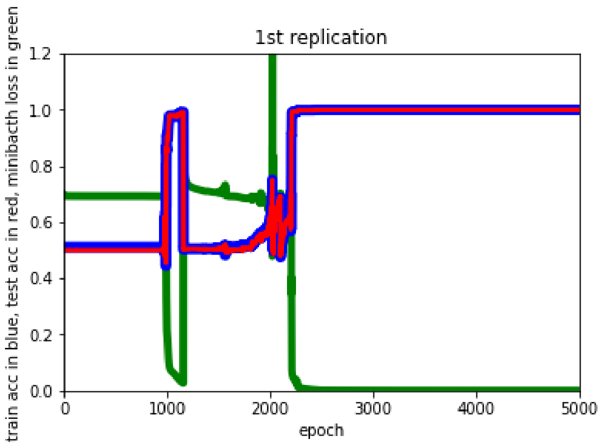
RNN H=25, N=25



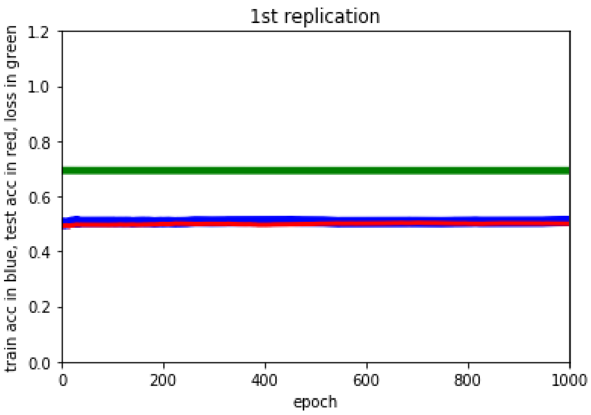
RNN H=25, N=50



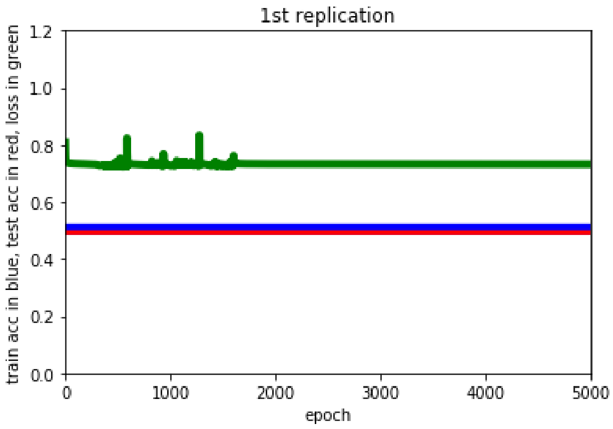
LSTM H=5, N=2



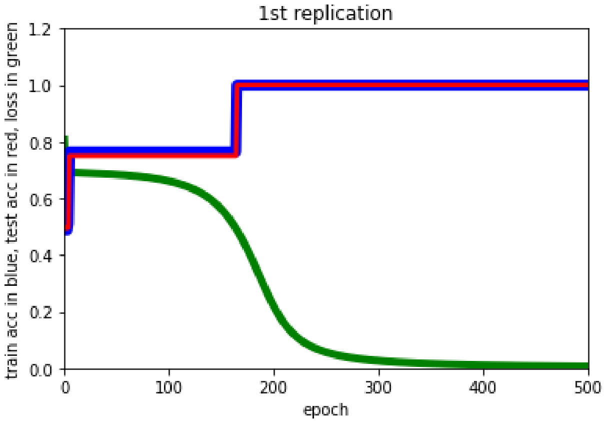
LSTM H=5, N=10



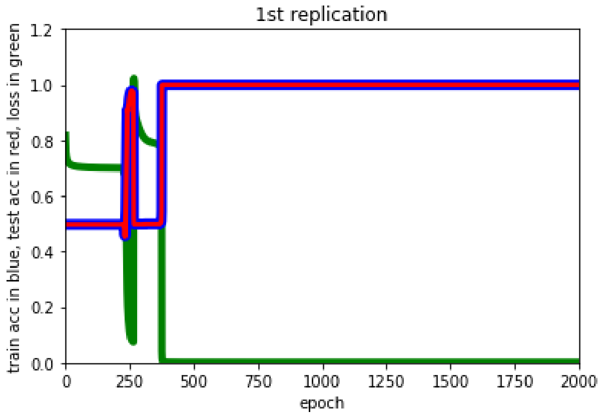
LSTM H=5, N=25



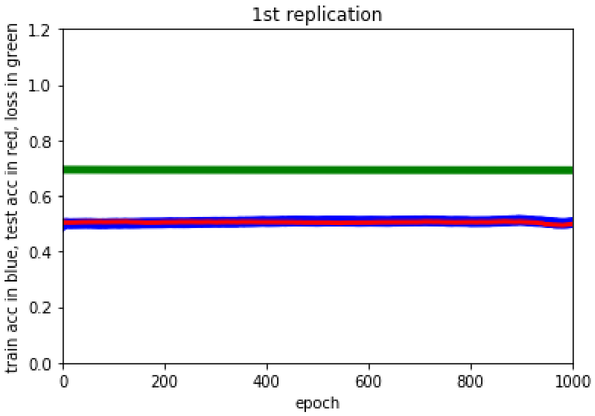
LSTM H=5, N=50



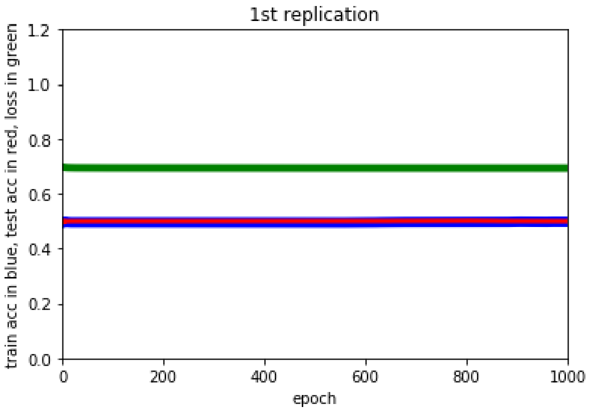
LSTM H=10, N=2



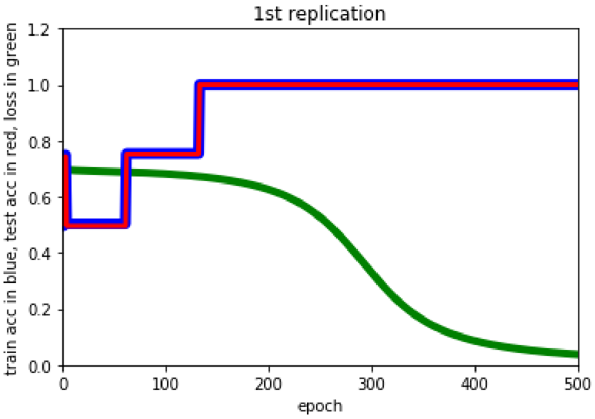
LSTM H=10, N=10



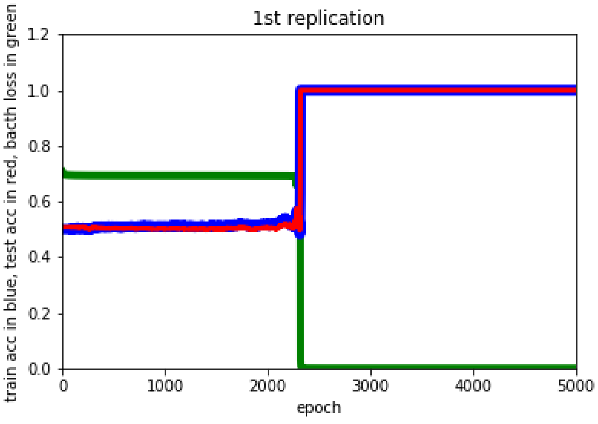
LSTM H=10, N=25



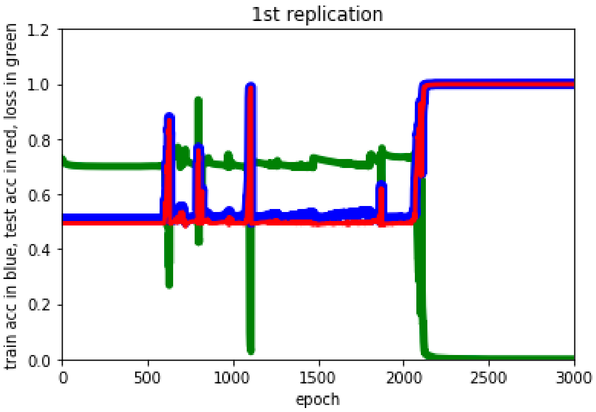
LSTM H=10, N=50



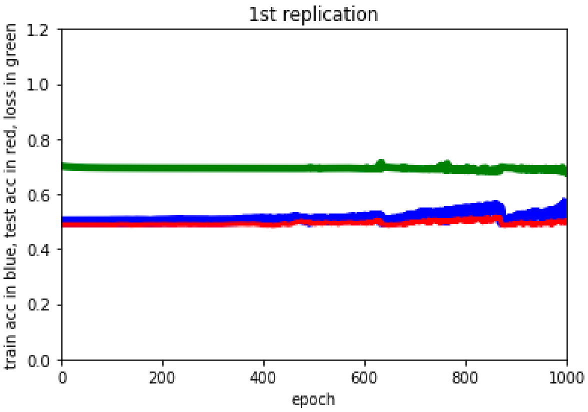
LSTM H=25, N=2



LSTM H=25, N=10



LSTM H=25, N=25



LSTM H=25, N=50

Discussion:

Notice that the train accuracy and test accuracy are kind of tracking each other , this is mainly due to the fact that for N = 2, 10, the length of sequence will be 4, 1024, while we randomly generated 10000 training and test set, so around 90% of the data will be exactly the same despite that the order is different. In addition, for N = 2, all settings network converged to 100% accuracy when being trained for enough epochs; but for N = 10, 25, 50, I think whether the network can learn and converge to a solution or not heavily depends on the initial weights -- sometimes it learned but sometimes the accuracy just jittered around 50%.

In general, for H = 25 and N >= 10, LSTM outperformed RNN, I think this is because LSTM can hold the information in a longer history than RNN, at this point LSTM could learn better on longer sequence.

Code:

All the codes are run in jupyter notebook in the directory of “assignment6-BoCao/src”. Results are already included in the “*.ipynb” files, file are named with setting parameters. For instance, “assign6_RNN_H25_N50.ipynb” is the code for 25 hidden units and 50 length of sequence.