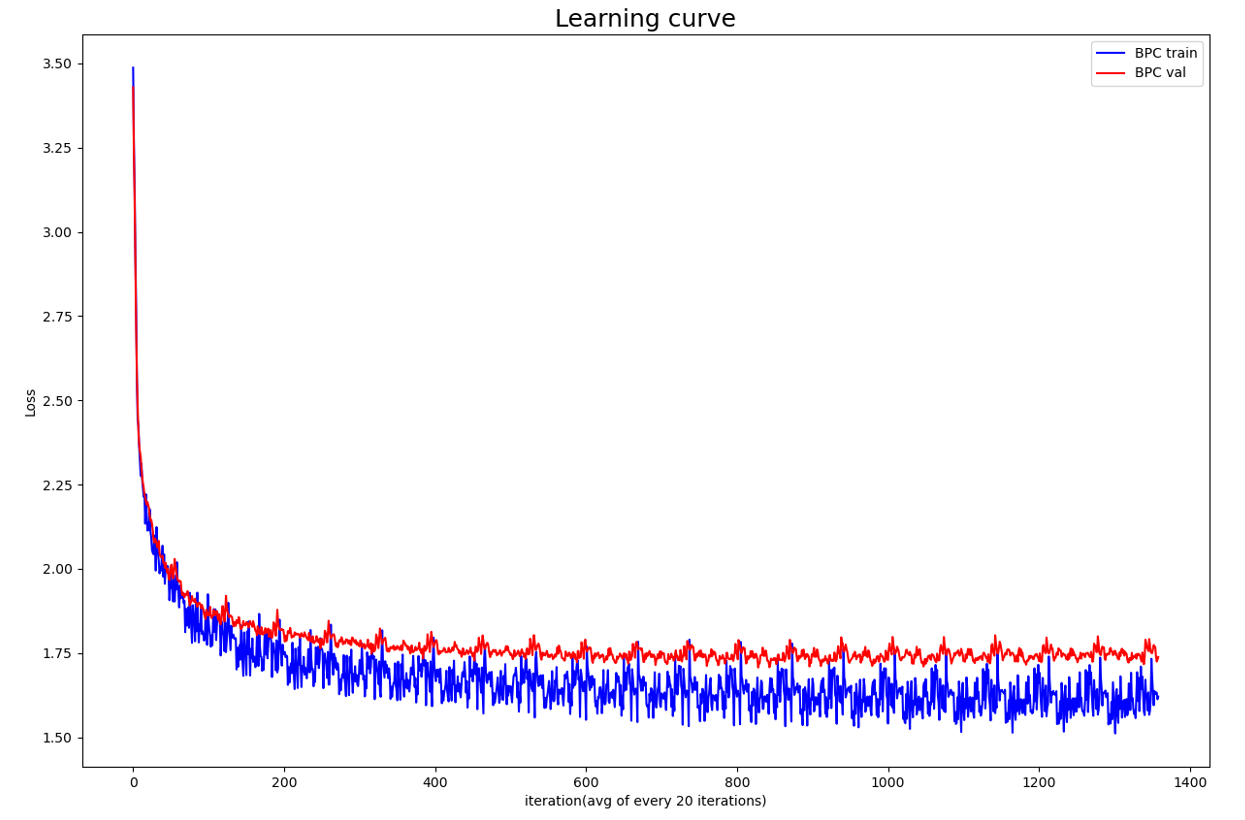
STANDARD RNN

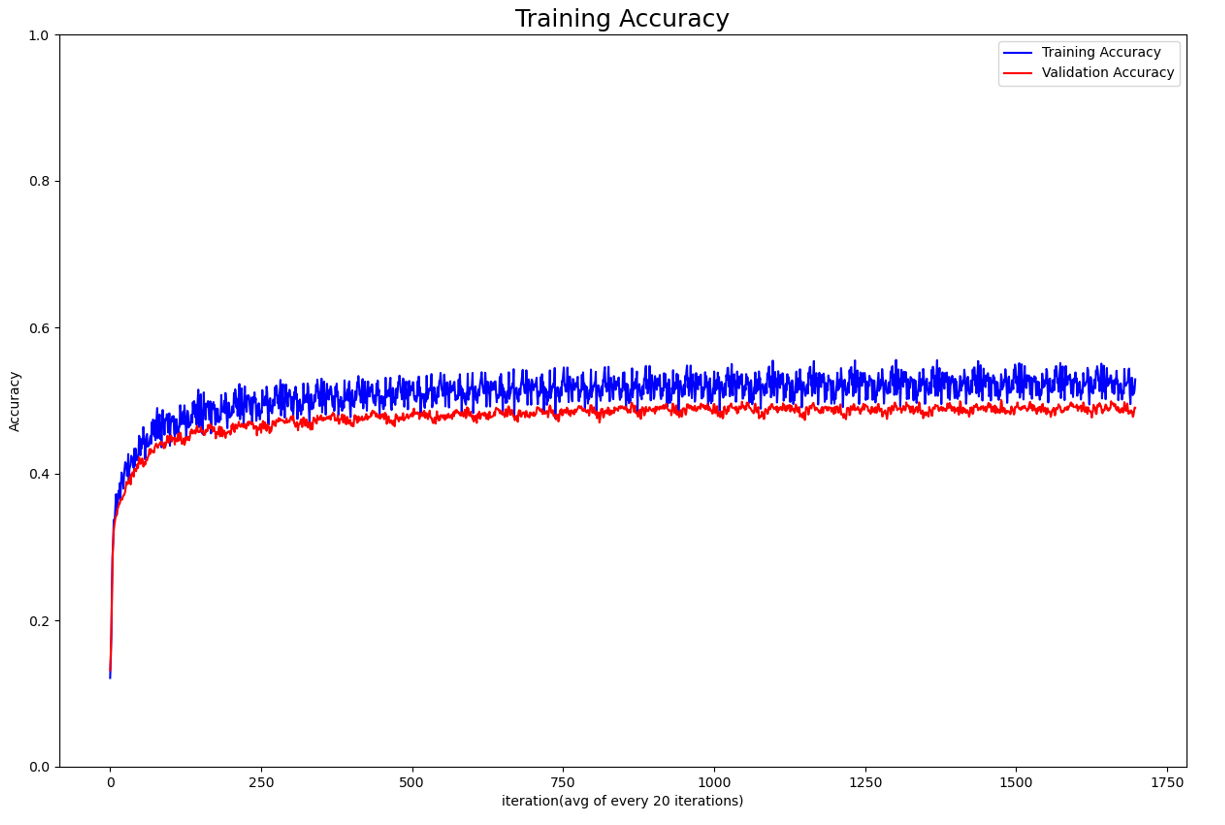
1. Plots of learning curve, error rate(accuracy)

Architecture:

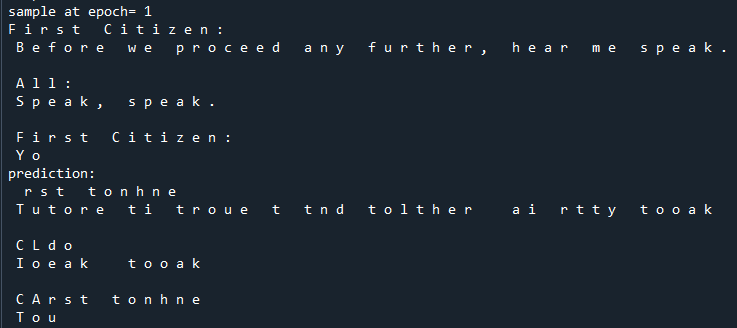
{hidden states=128,hidden layers=1,

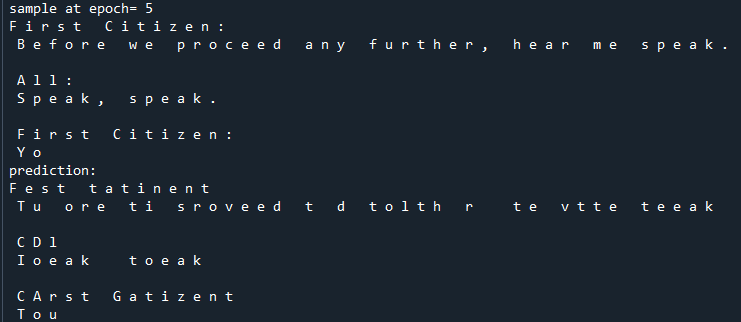
seq len=100-1, Batch size=32,lr=0.003}

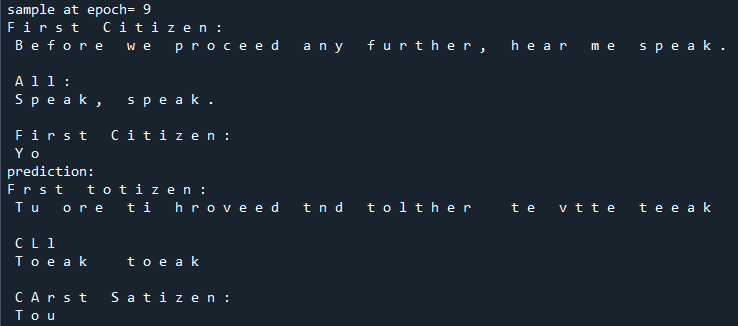


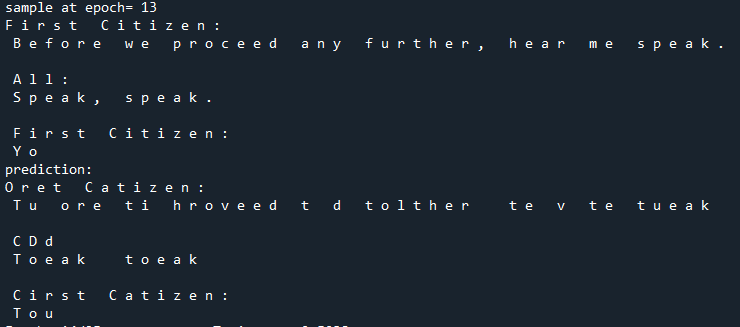


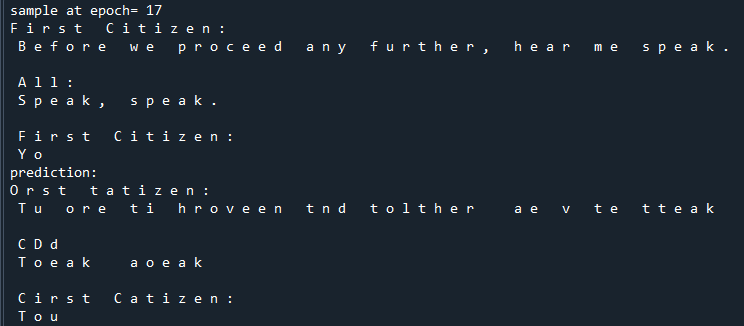
2.Breakpoints



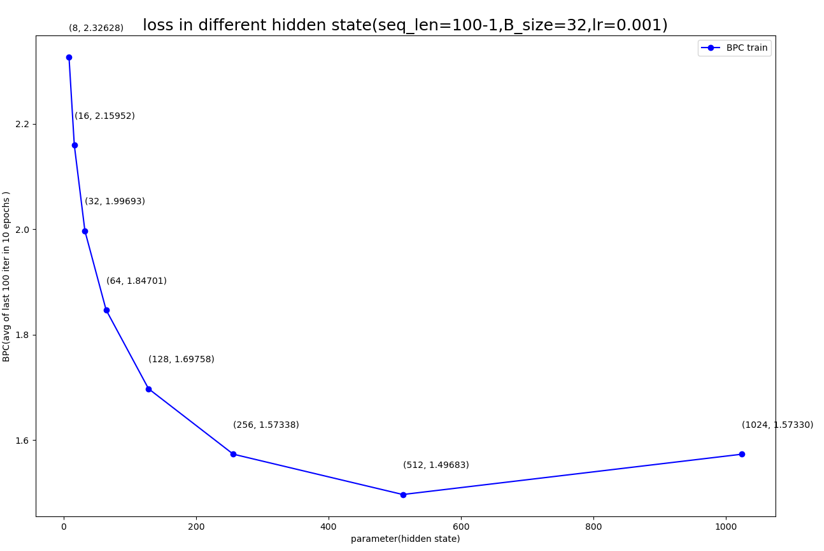


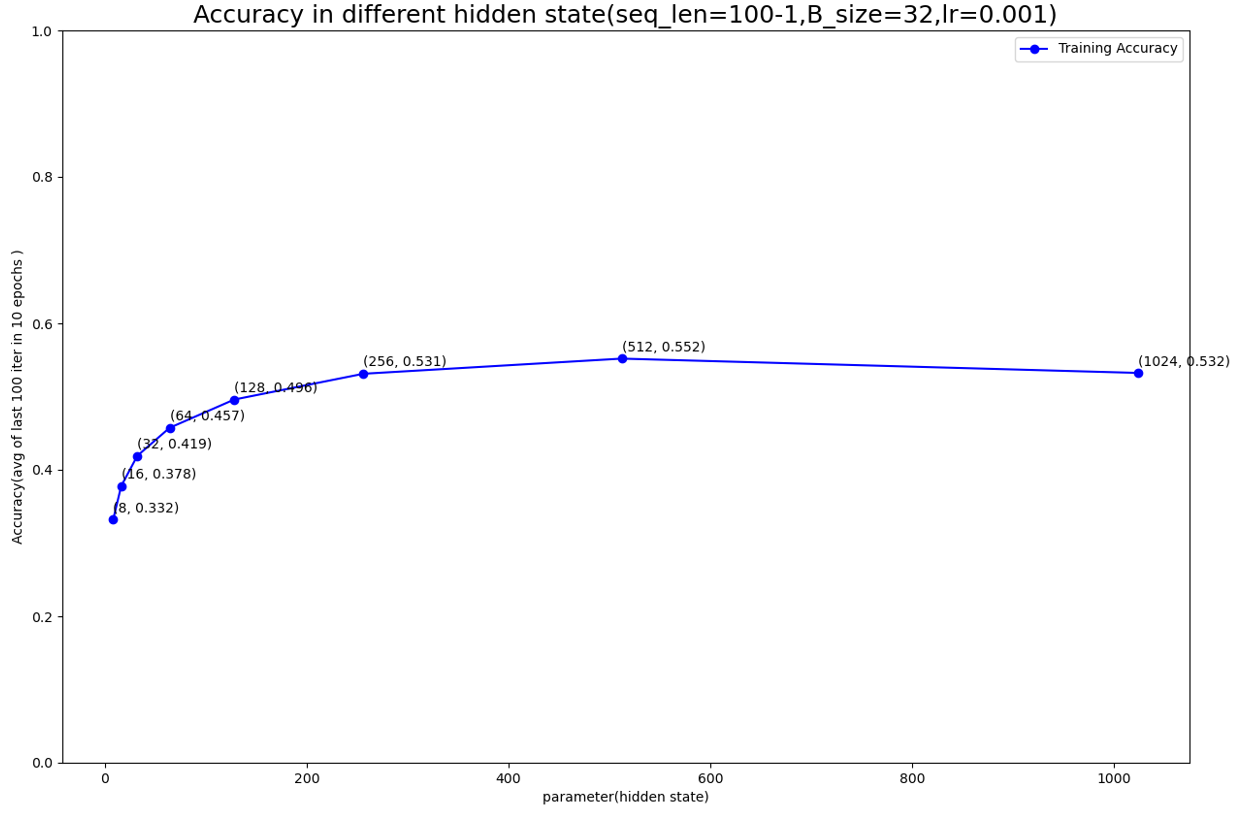


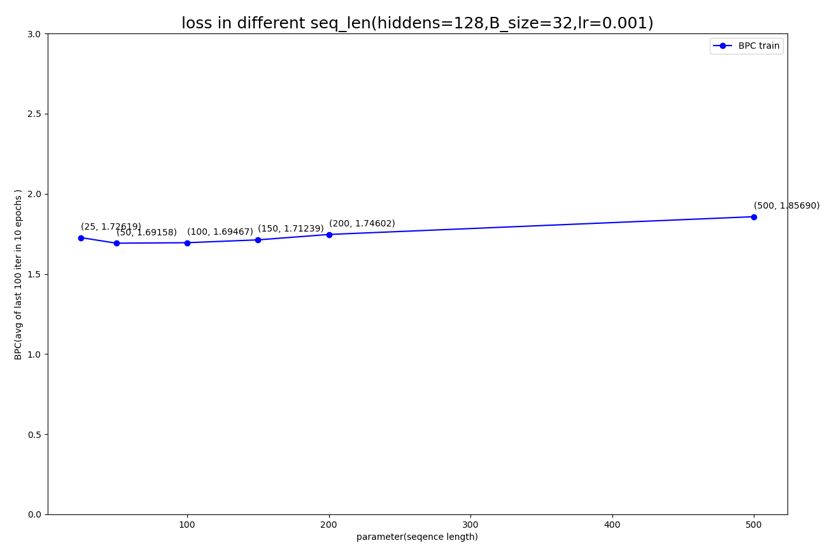


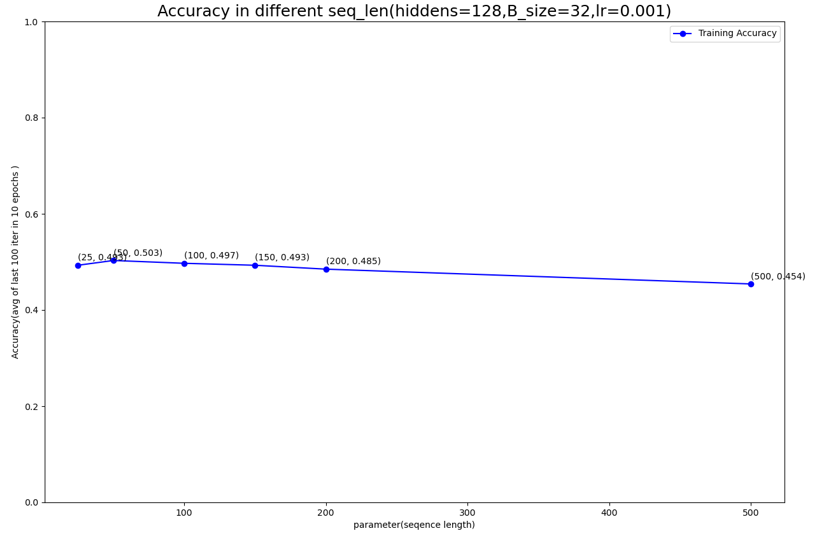


3.Changing parameter









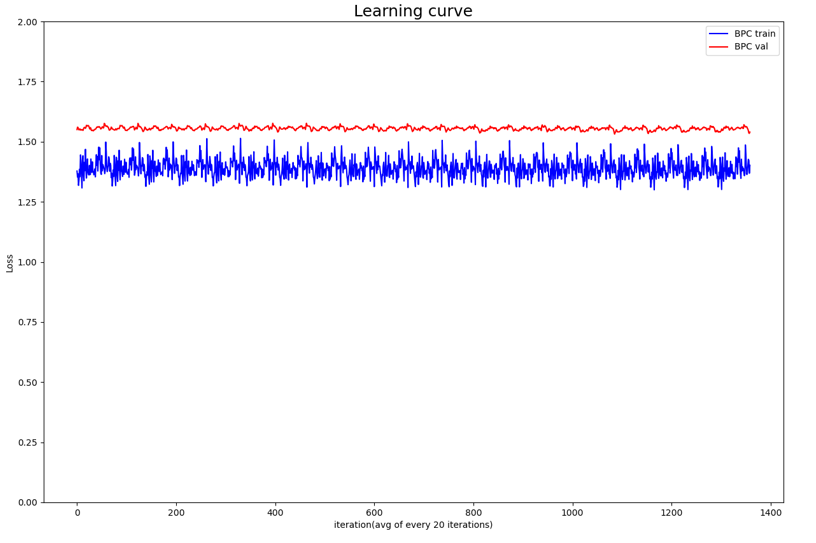
LSTM

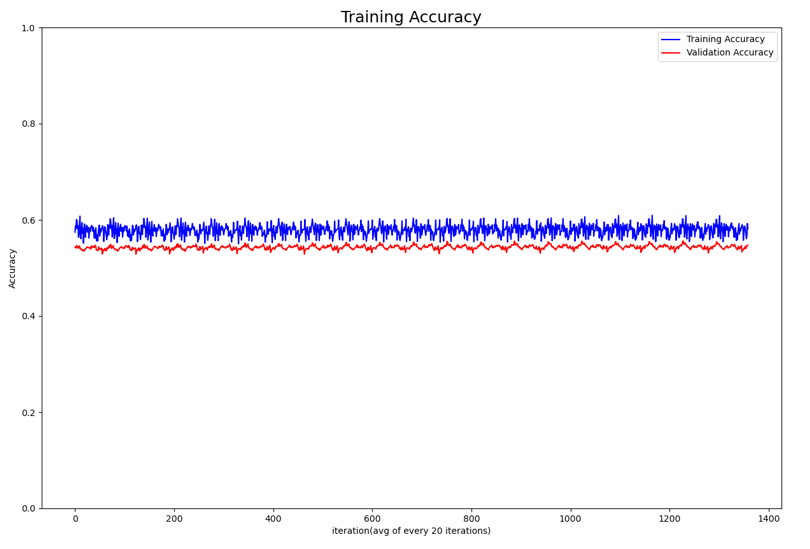
1.Plots of learning curve, error rate(accuracy)

Architecture:

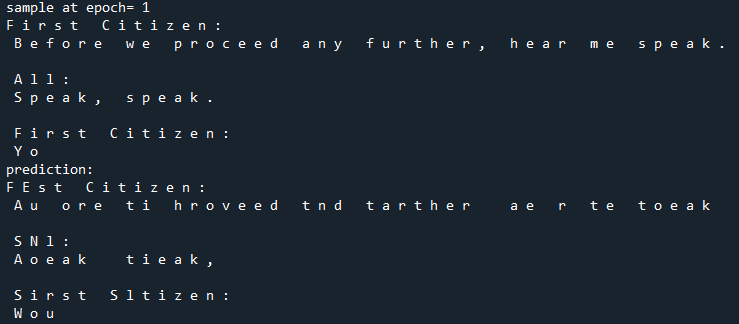
{hidden states=128, hidden layers=1,

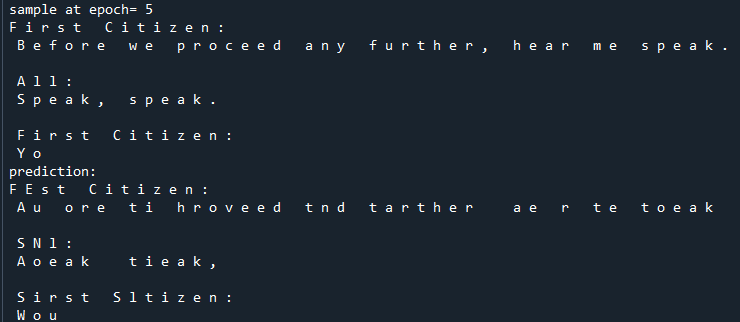
seq len=100-1, Batch size=32, lr=0.001}

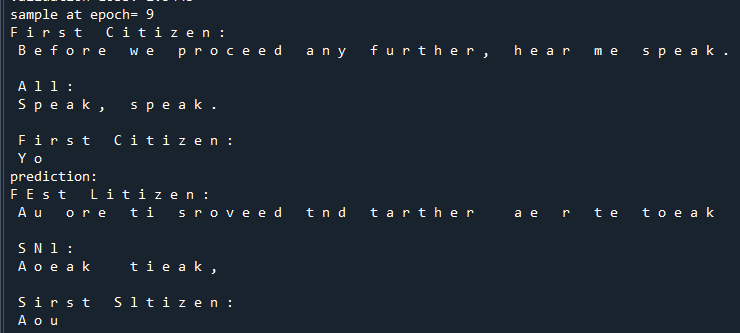


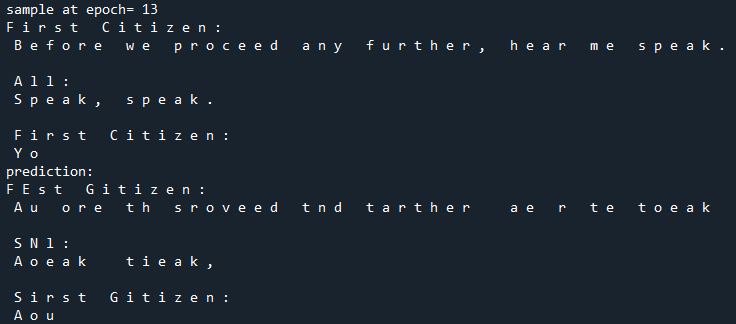


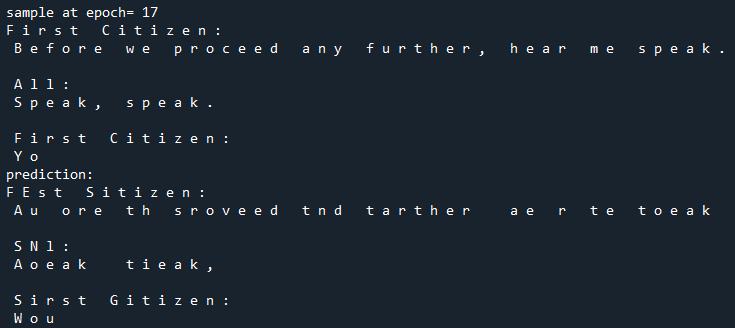
2. Breakpoints



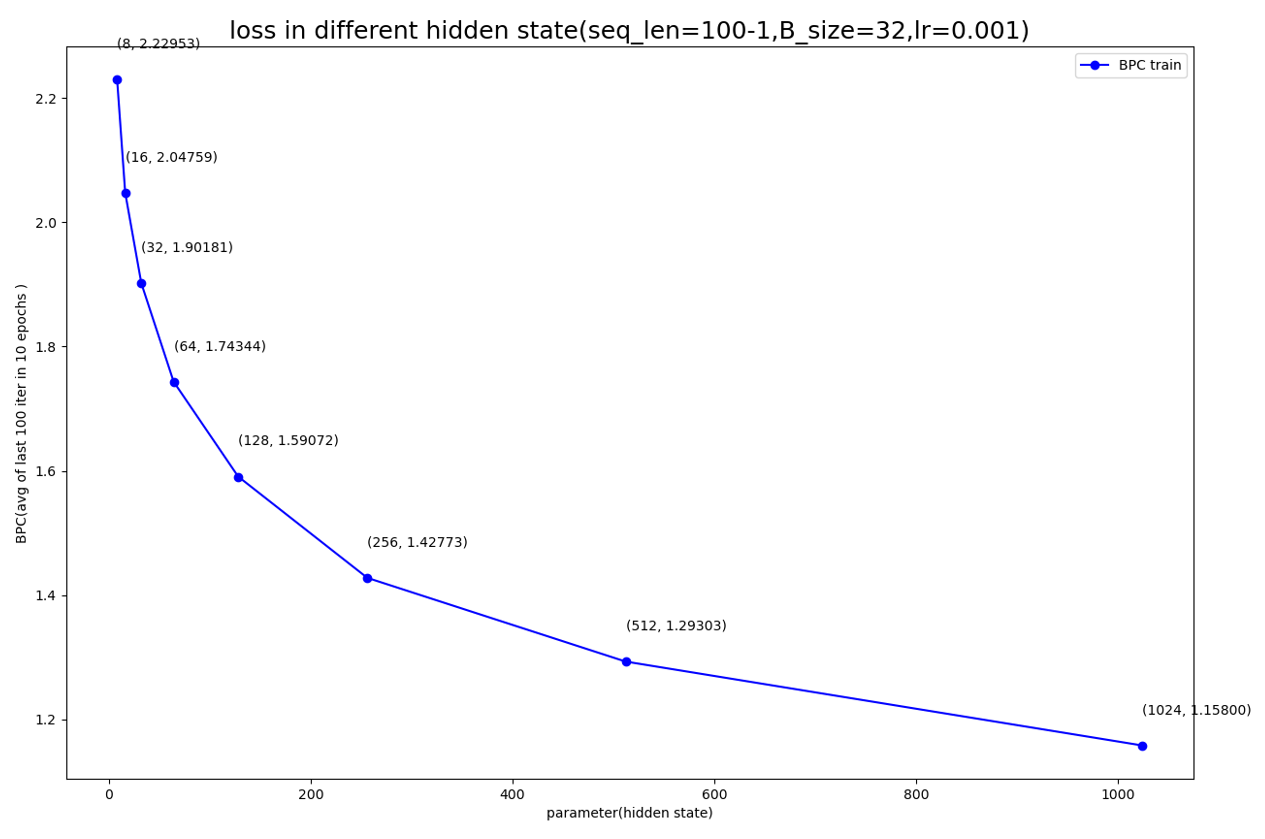


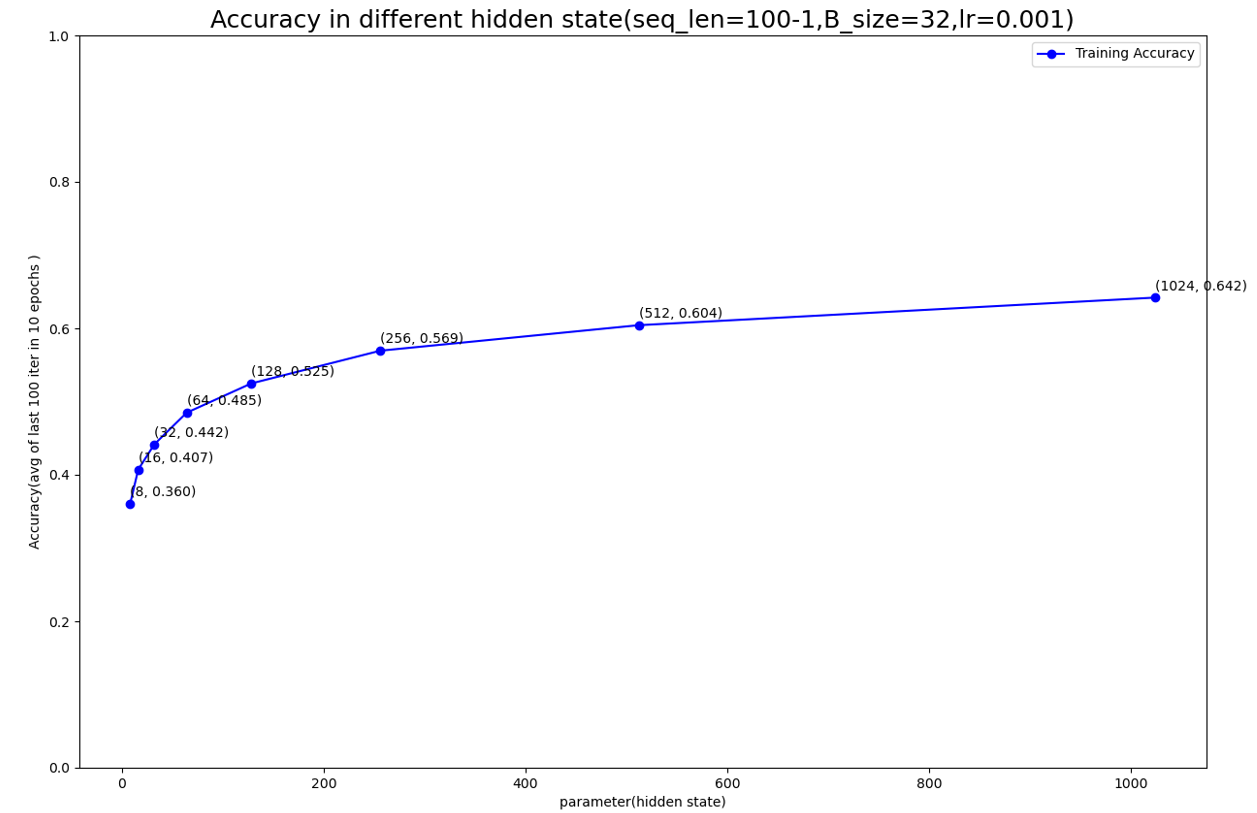


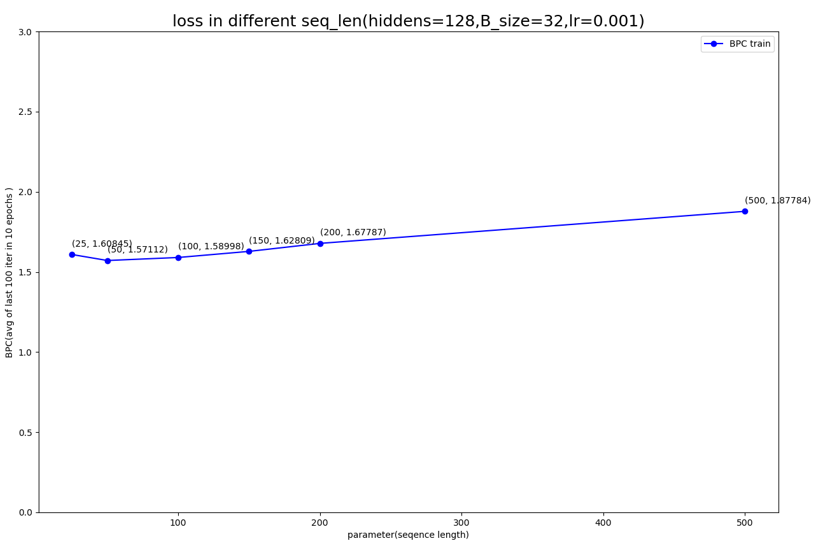


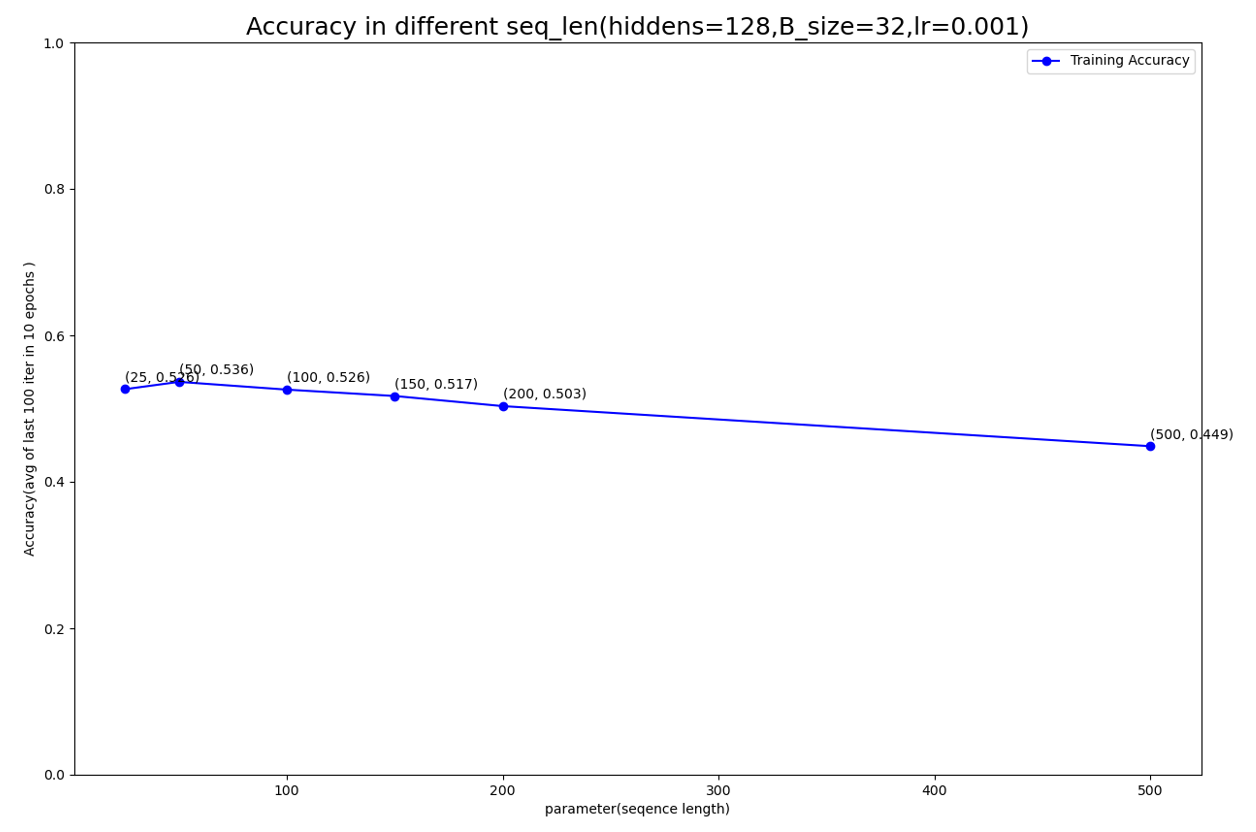


3.Changing parameter









Difference:

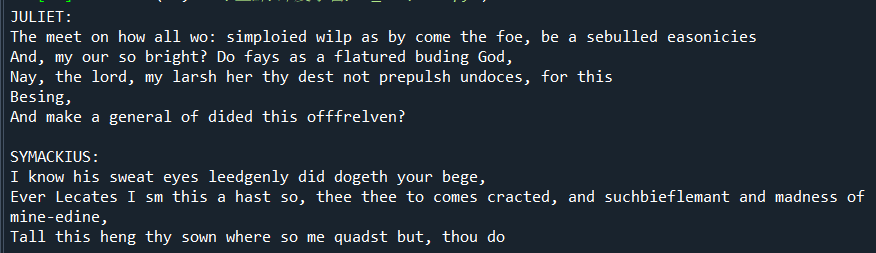
After observing the plot above.Maybe because the more parameter introduced by LSTM which provide it more capacity than RNN, LSTM perform better than RNN and LSTM converge faster then RNN (RNN loss are high in first 10 iterations but LSTM almost converge).

From this point, maybe we can conclude that RNN is hard to learn with large hidden size (Ex: perform bad when hidden size>512 and n\_layers>1). But LSTM get better result when increasing the hidden size

Priming the model:

Model(RNN): hidden size=128, seq\_len=500, lr=0.003,batch\_size=32,

Training epoch=20



Model(LSTM): hidden size=128, seq\_len=500, lr=0.001,batch\_size=32,

Training epoch=10

