Assignment2

Author: Ruiqi Kuang

1.b Train LeNet5 on CIFAR10:

In this section, i will change the training methods and the learning rate respectively.

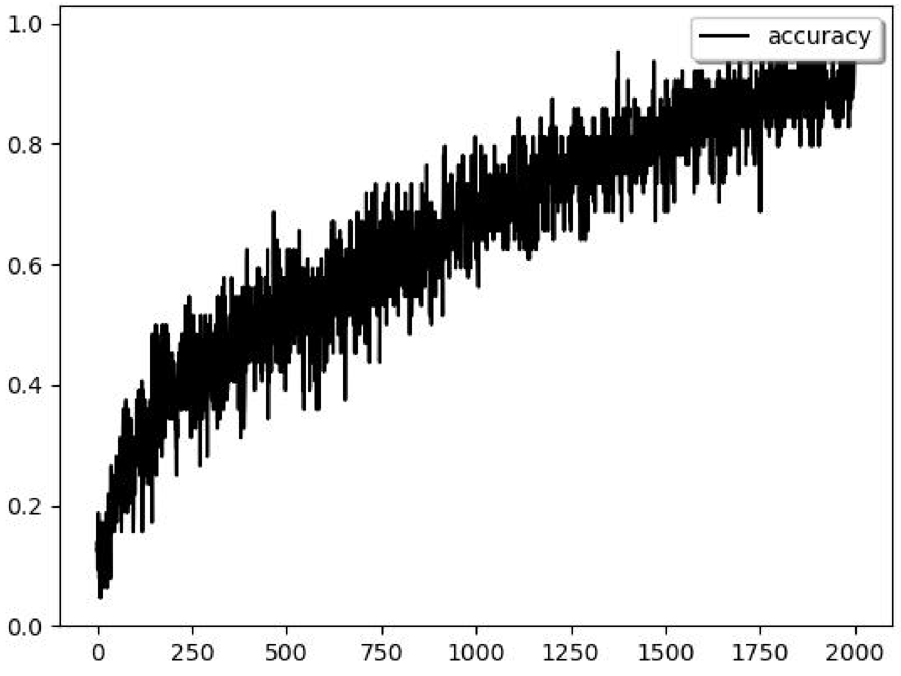
And as the experiments showed, the best hyper-parameters are:

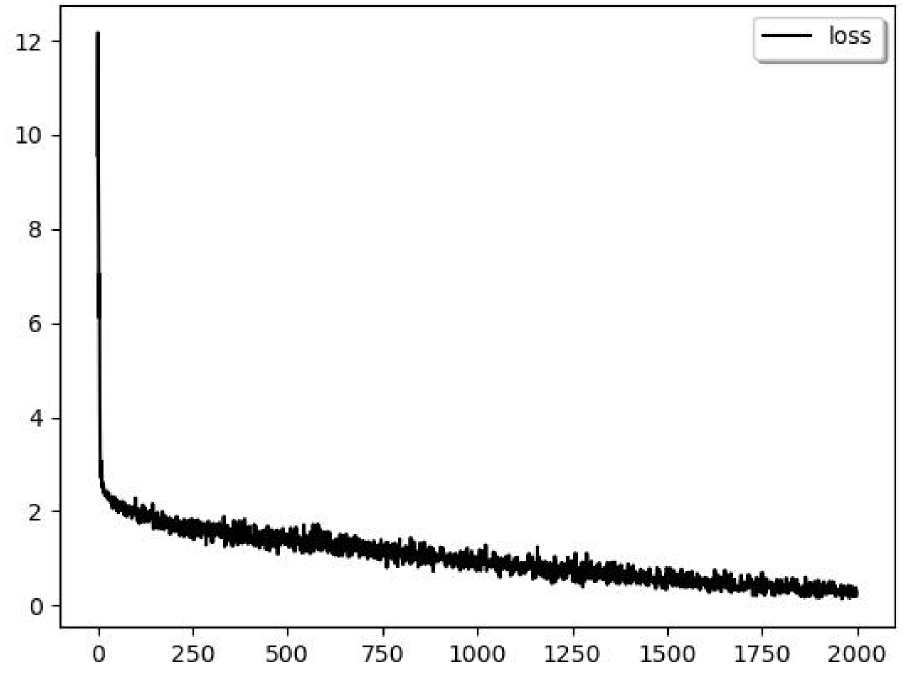
- Optimizer: *AdamOptimizer*
- *Learning rate: 1e-3*

Also. I found it interesting that in different learning rates(in this assignment, they are 1e-2, 1e-3, 1e-4), AdamOptimizer always work but GradientDescentOptimizer and AdagradOptimizer only work satisfactorily when learning rate is 1e-2.

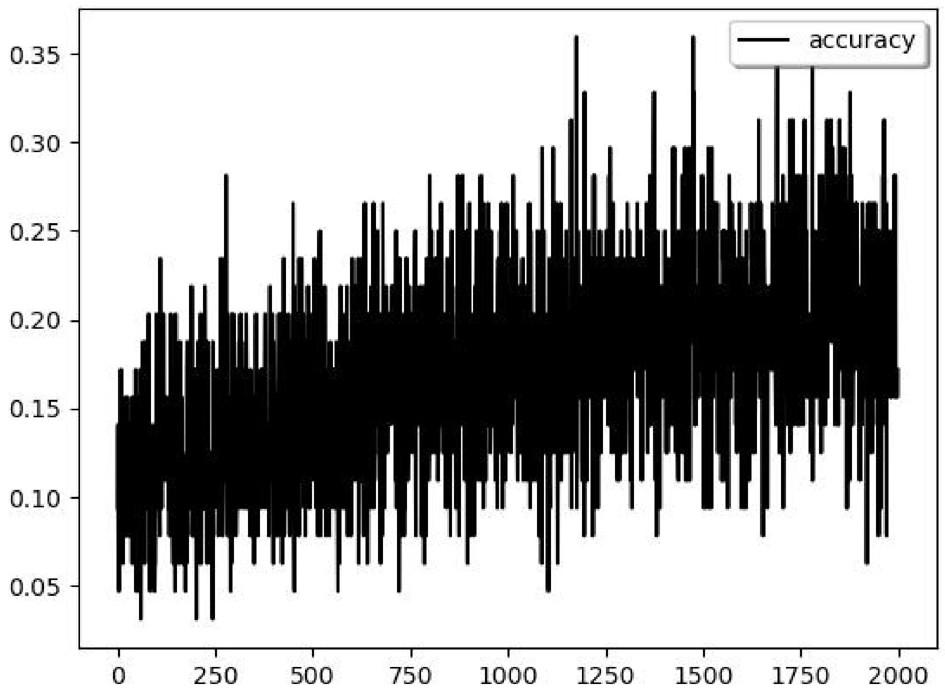
1) Fixed learning rate(1e-3)

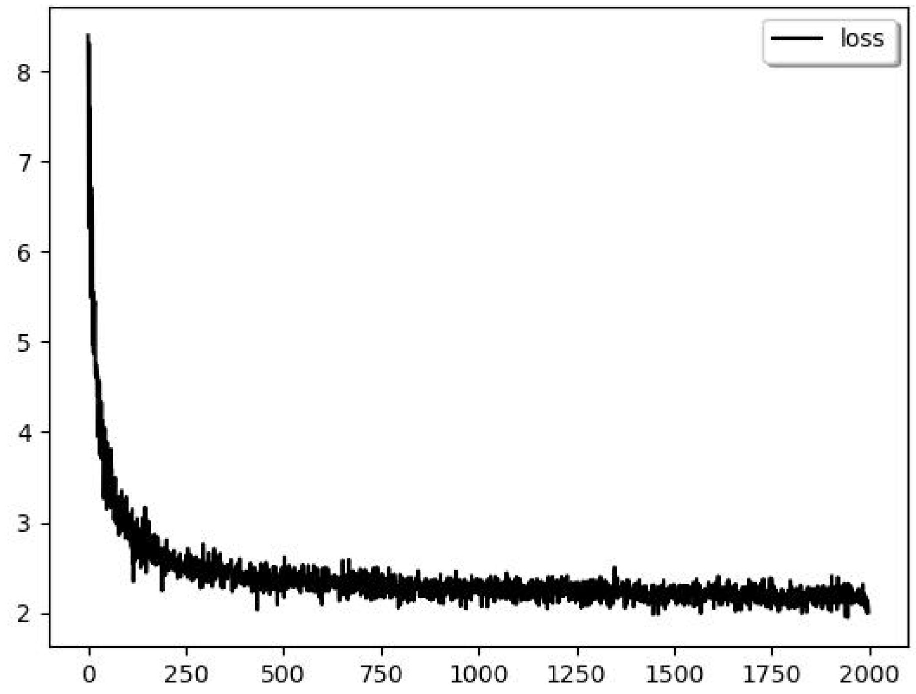
AdamOptimizer: test accuracy 0.536





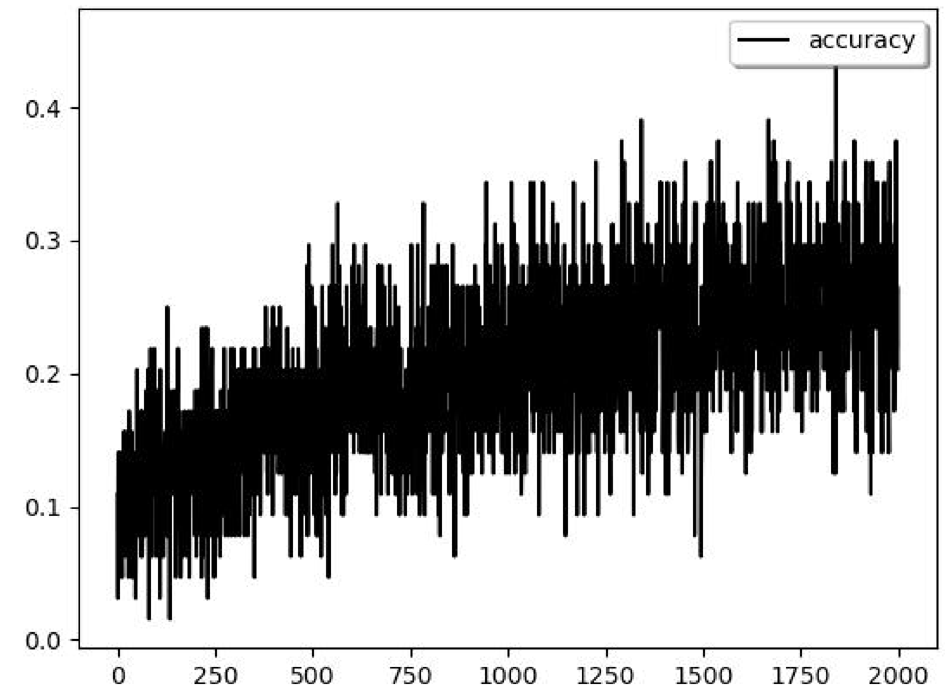
GradientDescentOptimizer: test accuracy 0.264

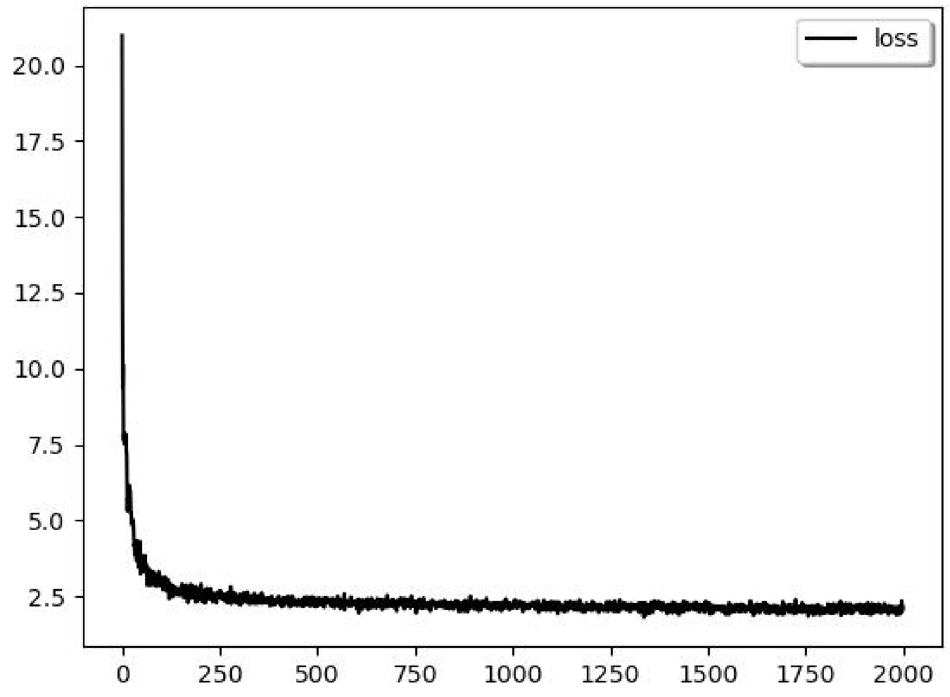




So weird, Gradient Descent didn’t work in this situation.

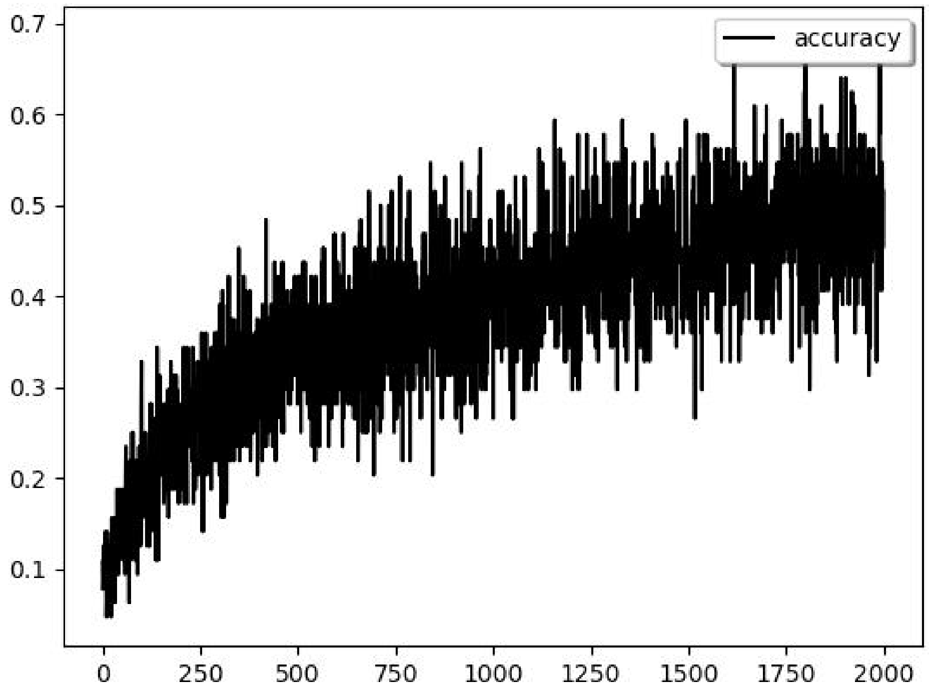
AdagradOptimizer: test accuracy 0.312

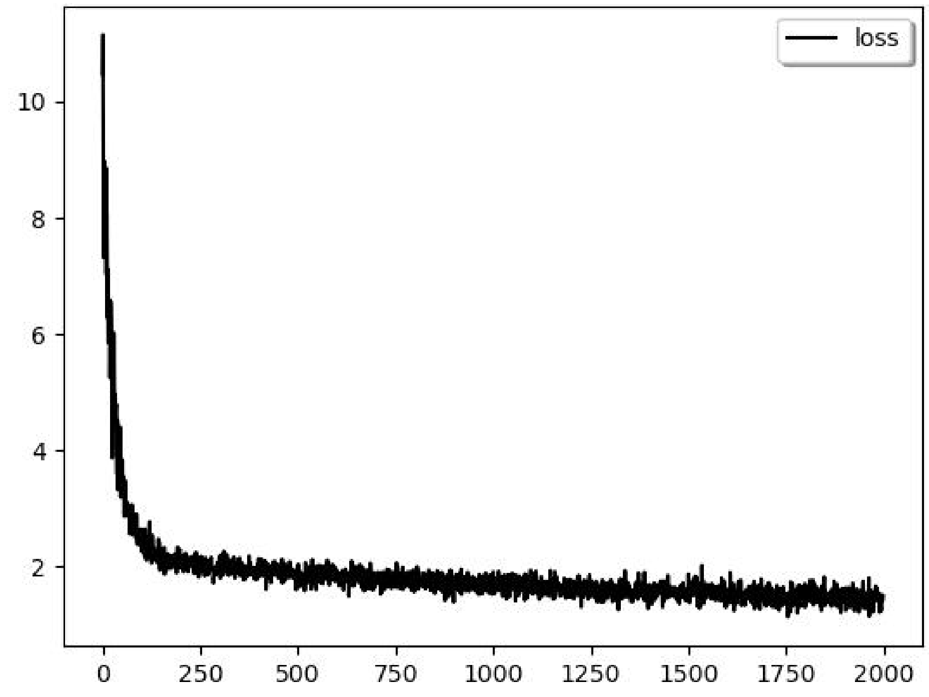




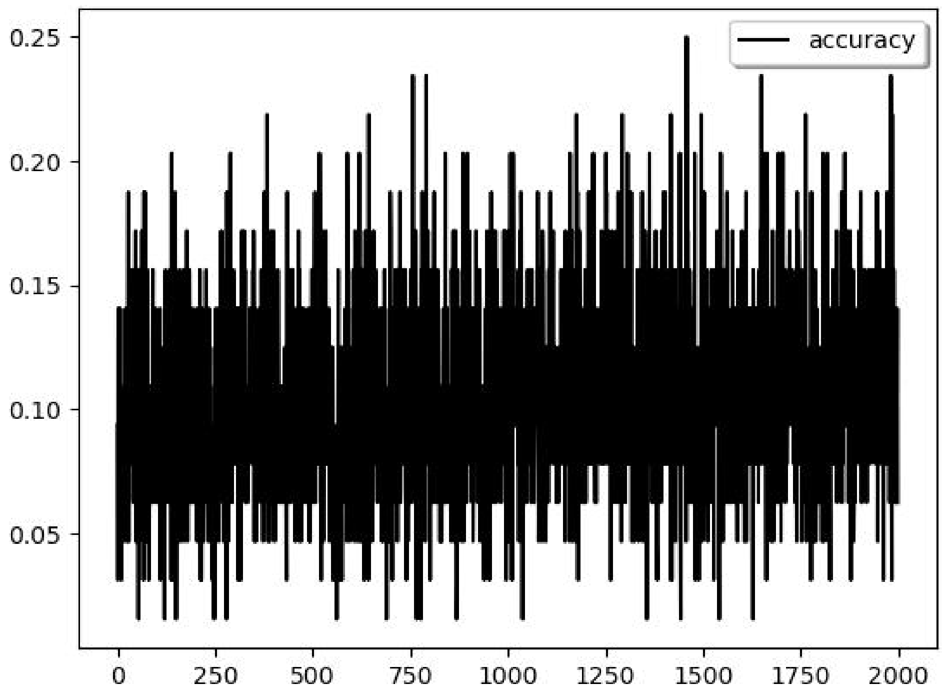
2) Fixed learning rate(1e-4)

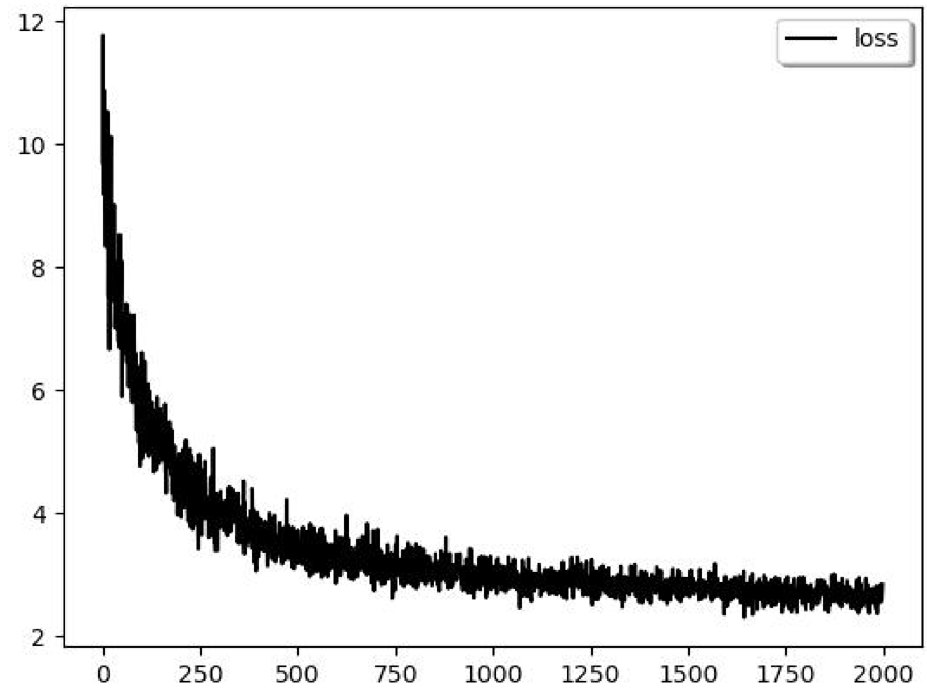
AdamOptimizer: test accuracy 0.474



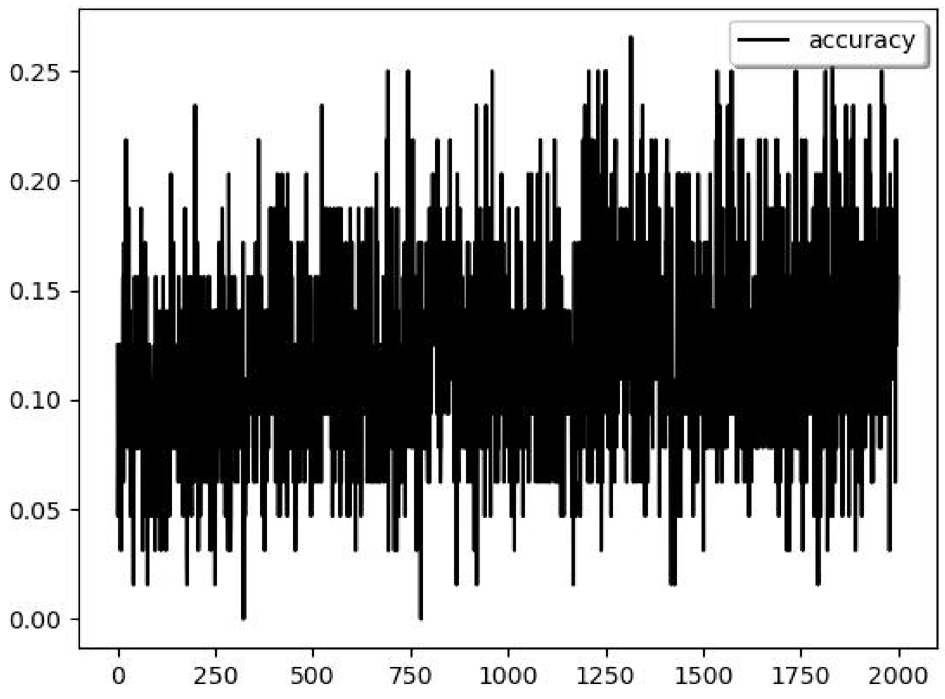


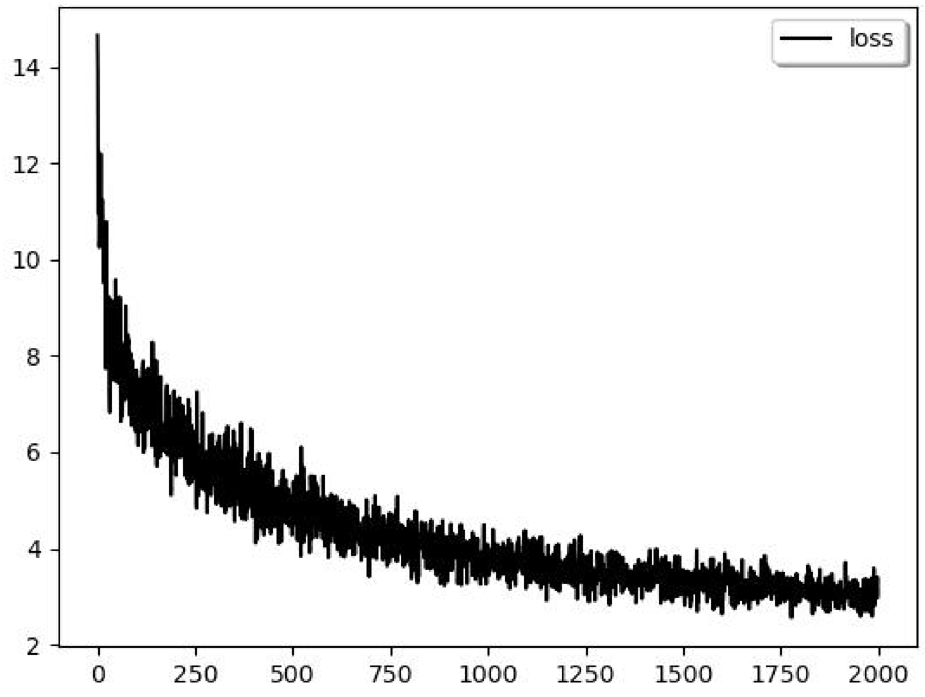
GradientDescentOptimizer: test accuracy 0.161





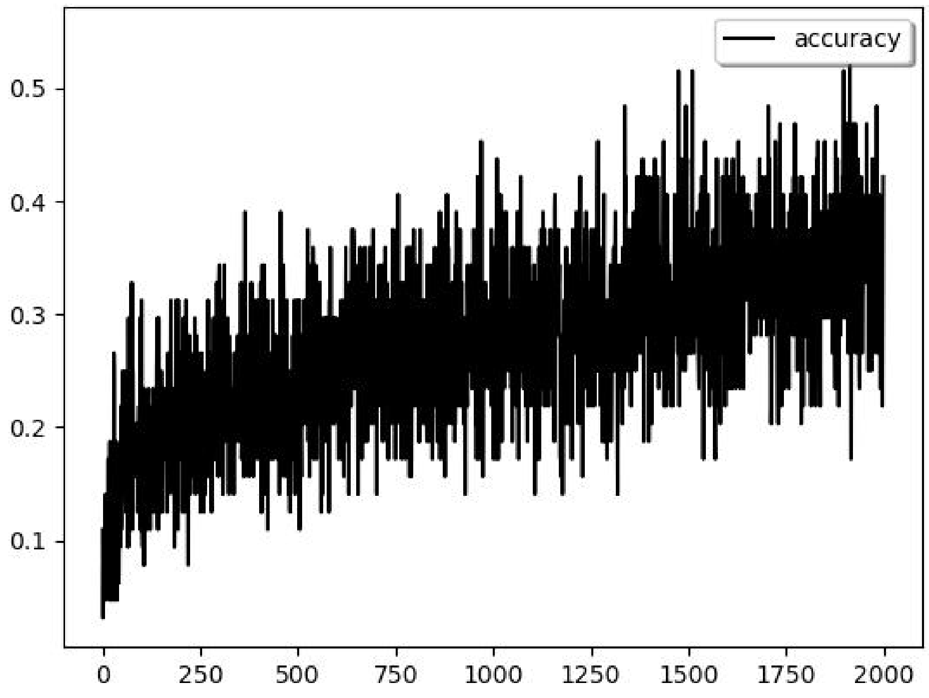
AdagradOptimizer: test accuracy 0.199

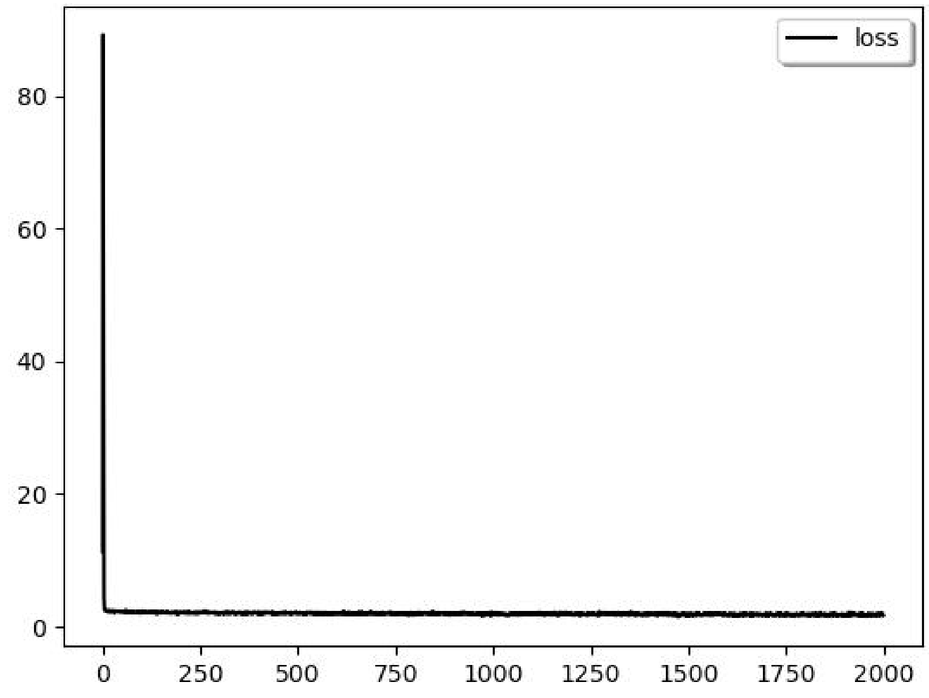




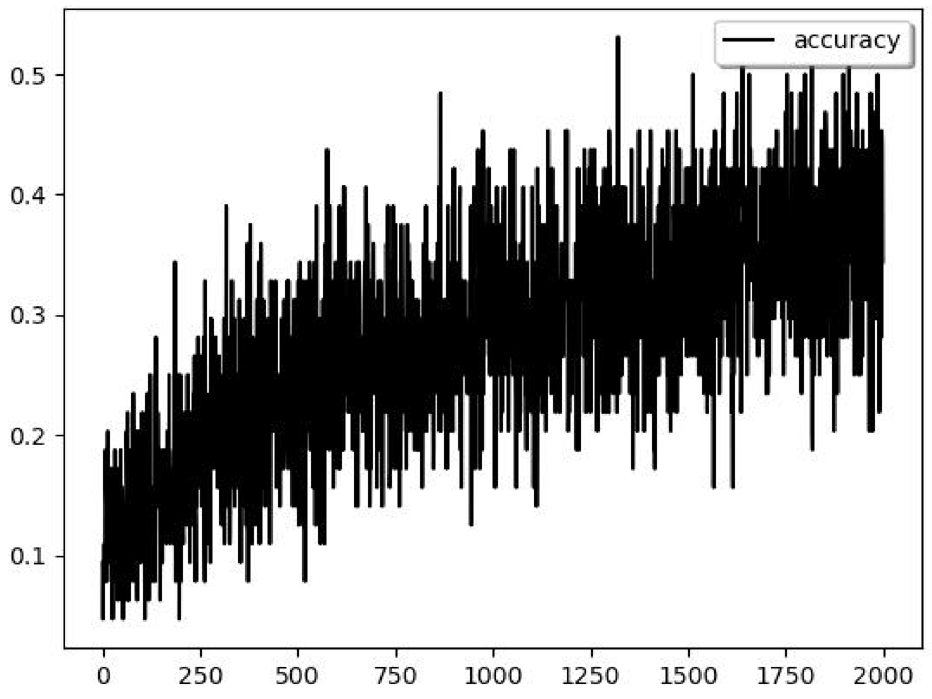
3) Fixed learning rate(1e-2)

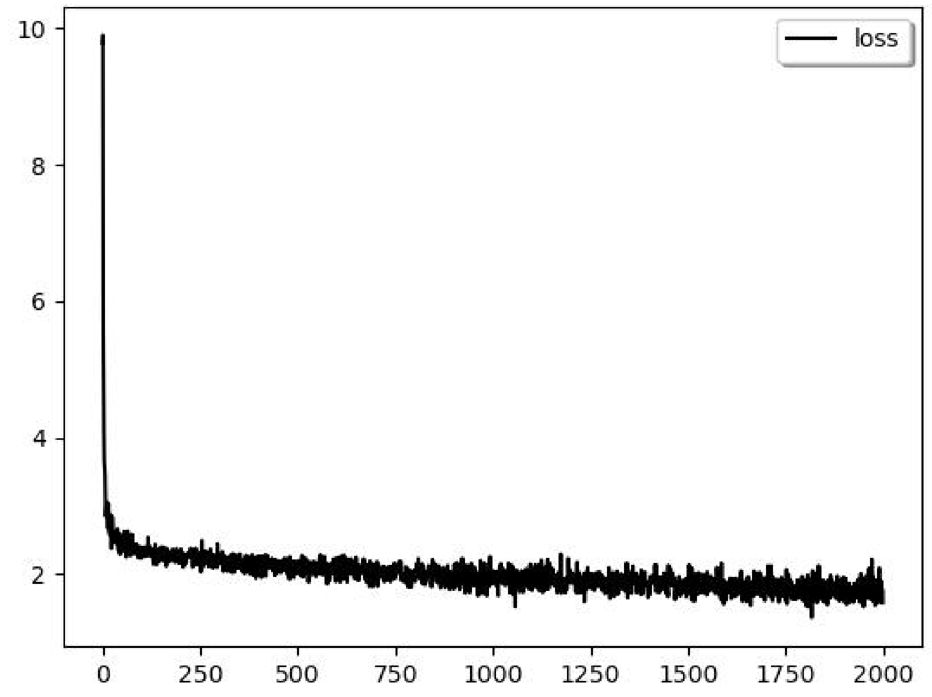
AdamOptimizer: test accuracy 0.305





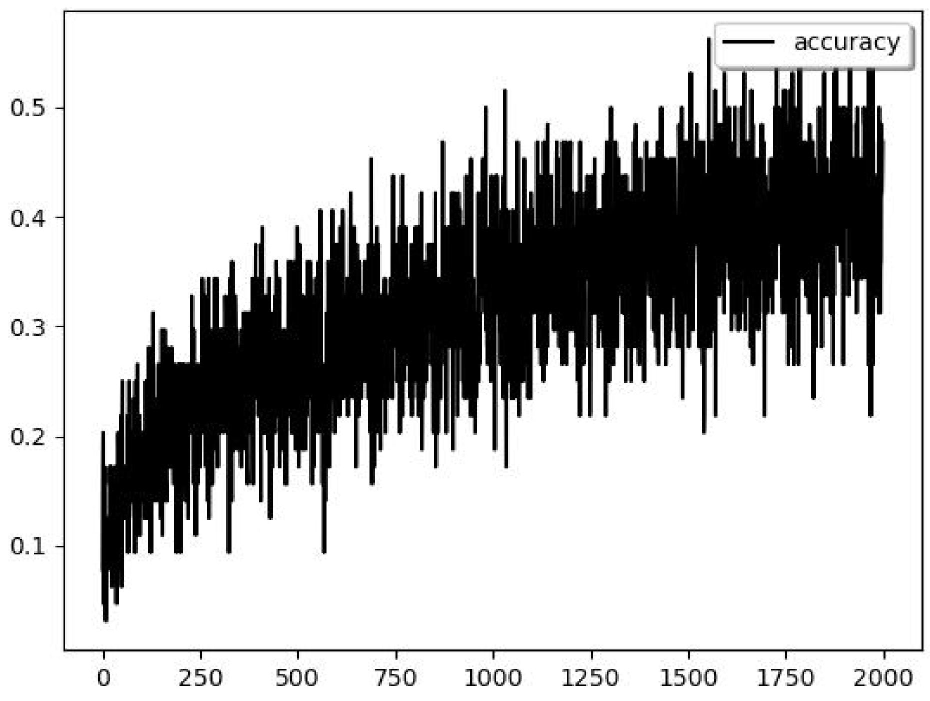
GradientDescentOptimizer: test accuracy 0.398

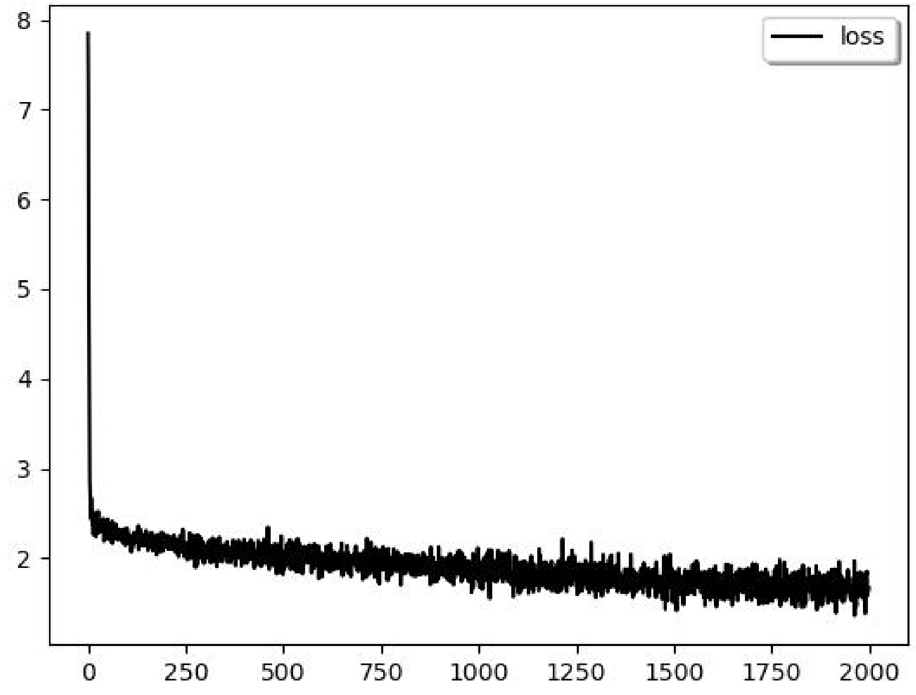




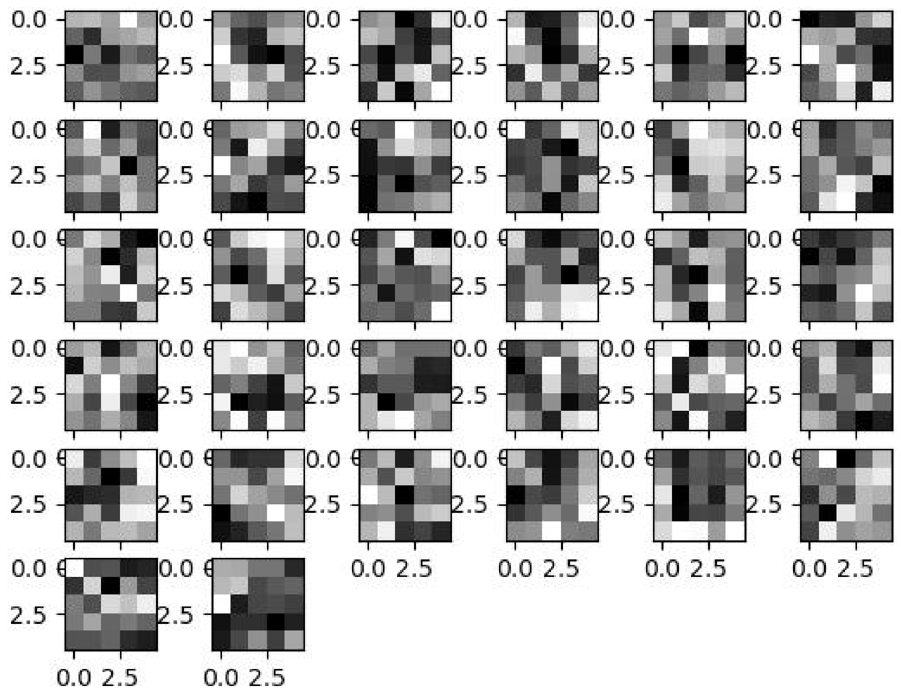
It works!

AdagradOptimizer: test accuracy 0.411





1.c Visualize the Trained Network:



2.

The visualization technique the authors propose uses a multi-layered Deconvolutional Network (deconvnet) to complete the process of inverse mapping from output features to input signals.

In the model, a deconvolutional network is attached to each layer of the convolutional network. First, the input image is passed through the convolutional network, and each layer is given the corresponding output. To test the features of a particular convolutional layer, all feature values are set to zero and then fed into the deconvnet connected to it, in the following order.

Unpooling: max pooling operation is usually irreversible. An approximate reversal method is used in the paper, a switch set is used to save the position of each maximum value in the original graph, and the unpooling process labels the maximum value back to the location of the record, and the remaining positions are complemented by 0.

Rectification: Same as the convolutional network, the reconstructed signal is fed to the relu nonlinear function, which guarantees the non-negativity of the output.

Filtering: Uses the same convolutional kernel as the convolutional network but transposed to perform the convolution operation, which is equivalent to flipping the parameter matrix horizontally and vertically.

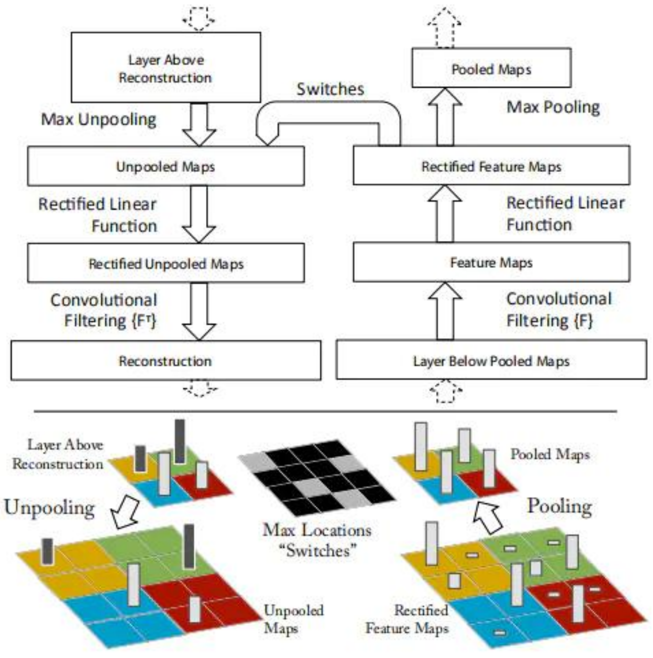


Figure 1. Top: A deconvnet layer (left) attached to a convnet layer (right). The deconvnet will reconstruct an approximate version of the convnet features from the layer beneath. Bottom: An illustration of the unpooling operation in the deconvnet, using *switches* which record the location of the local max in each pooling region (colored zones) during pooling in the convnet.

3.b

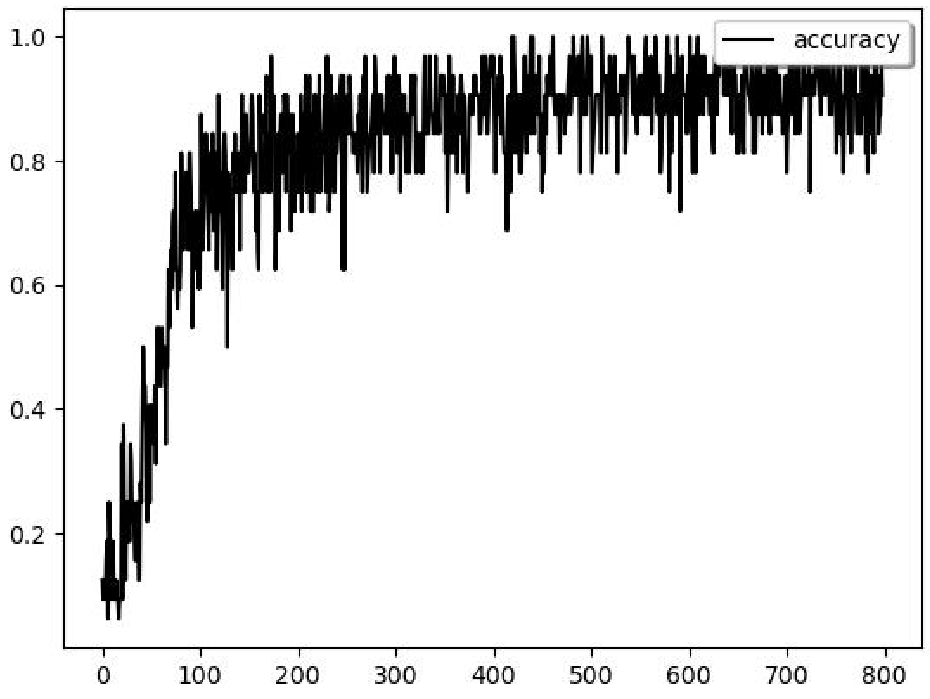
In this section, I will change the cell used in the model to train on the MNIST and the number of hidden neurons for the RNN.

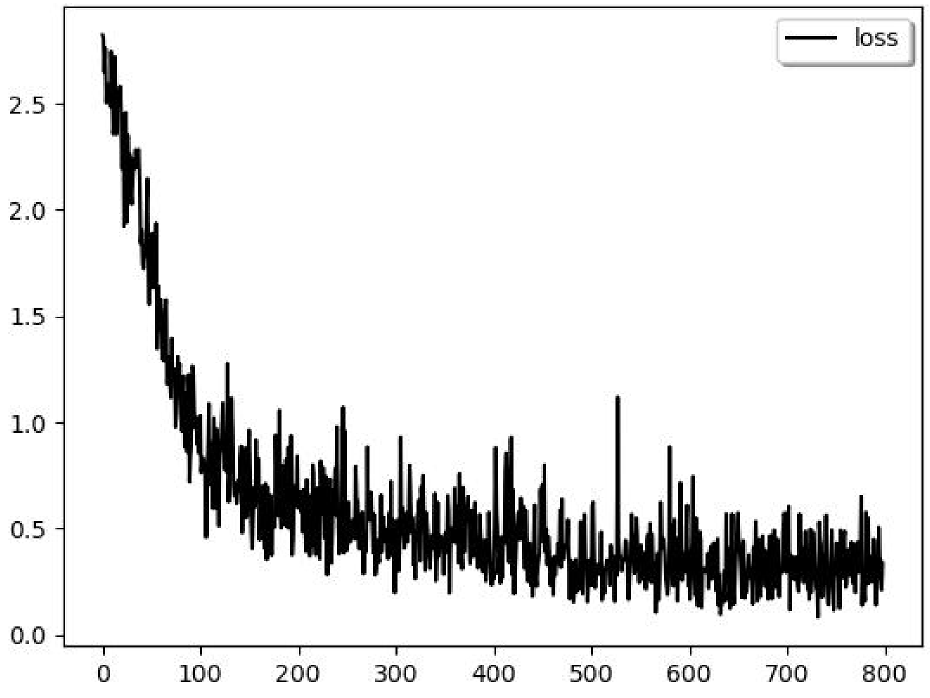
And as the experiments showed, the best hyper-parameters are:

- LSTM Cell: *GRUCell*
- *nHidden: 128*

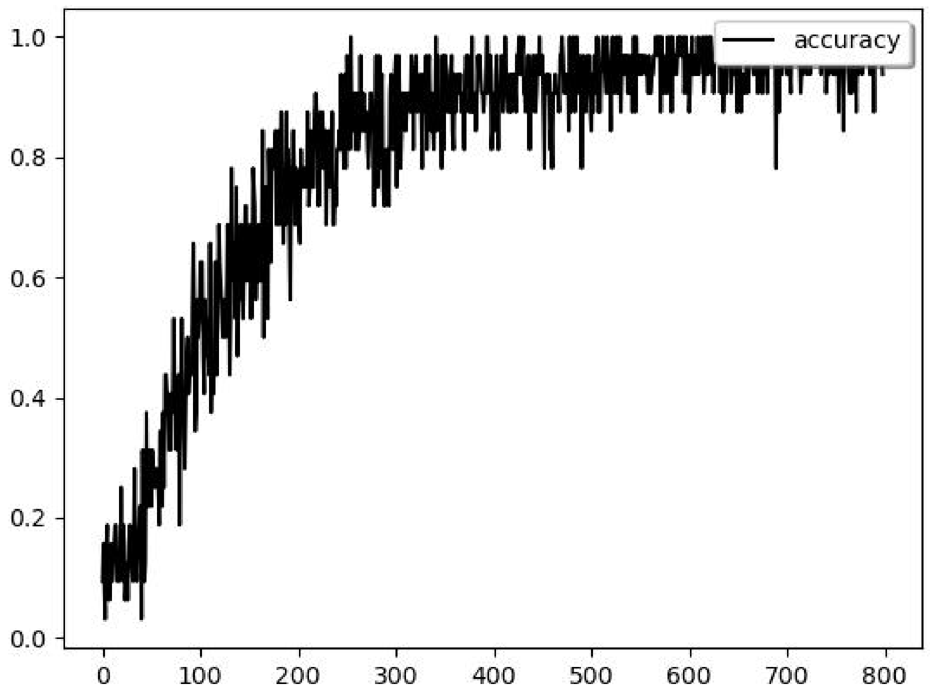
1) nHidden = 64

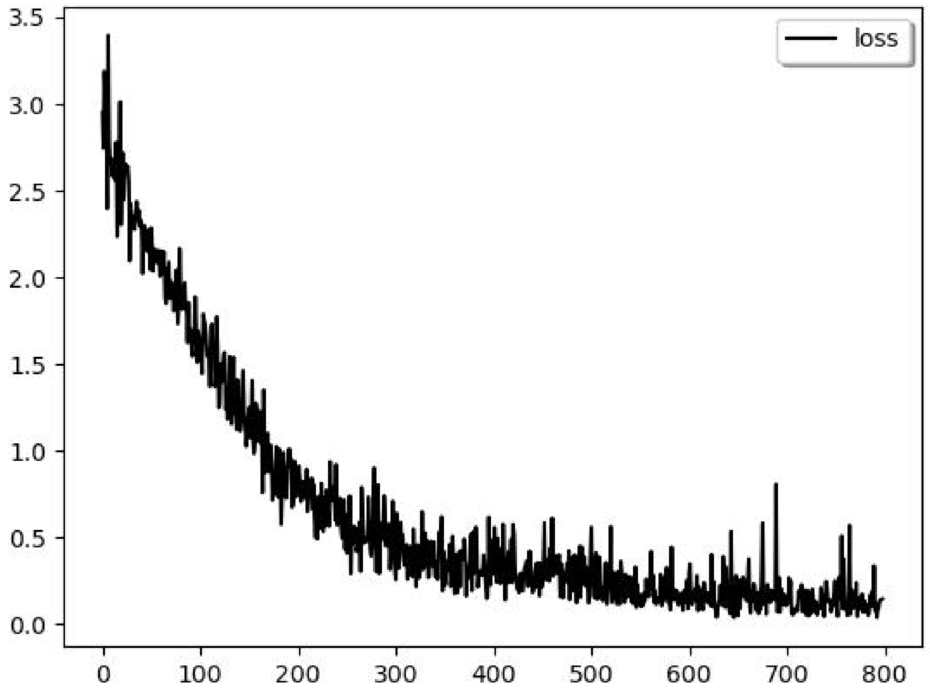
BasicRNNCell: Testing Accuracy: 0.8562



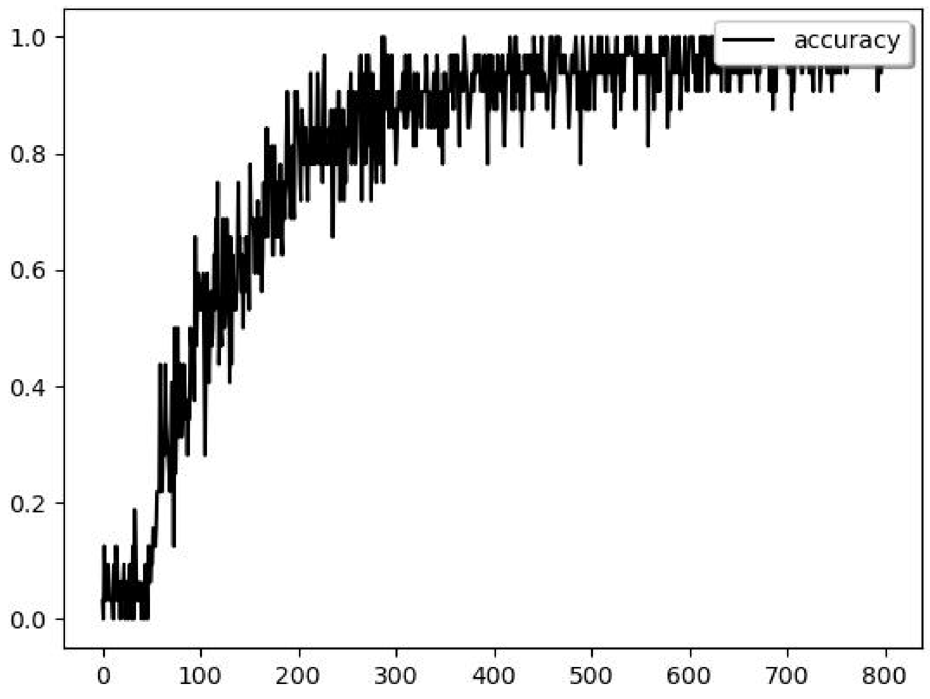


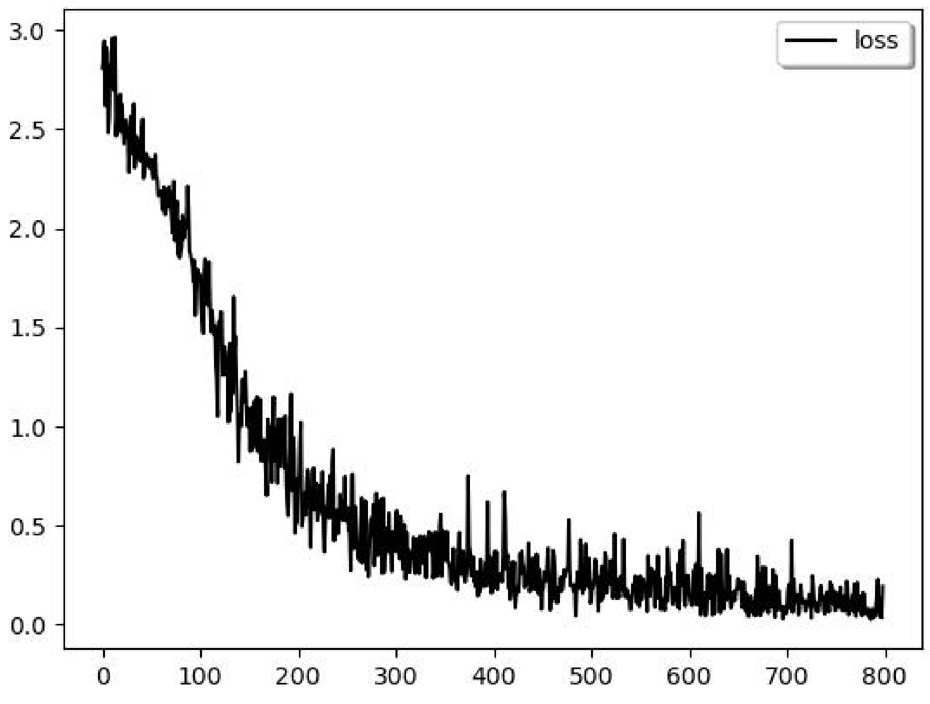
GRUCell: Testing Accuracy: 0.9346





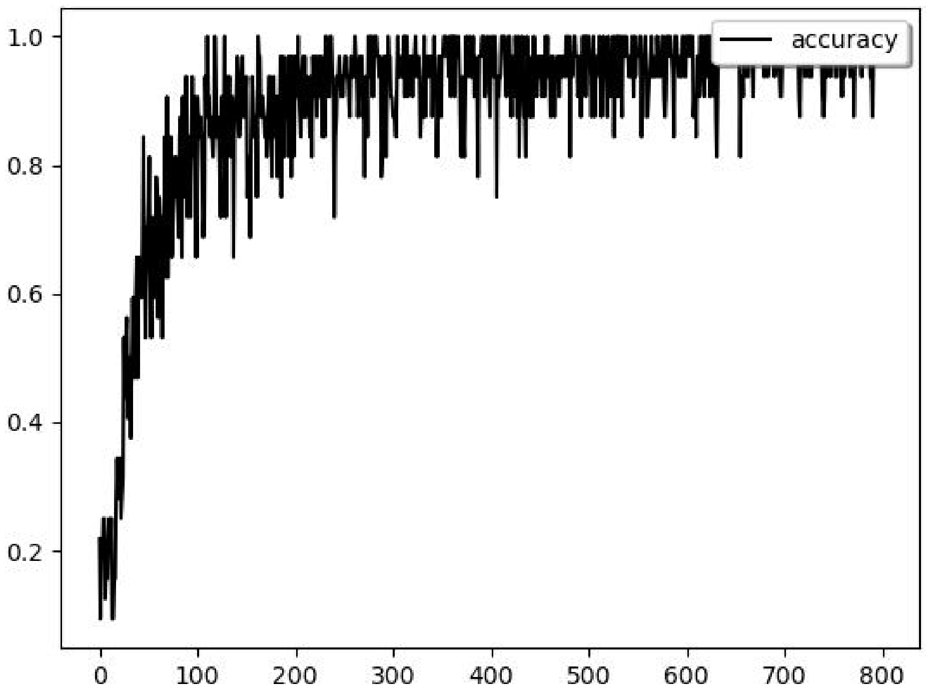
BasicLSTMCell: Testing Accuracy: 0.919

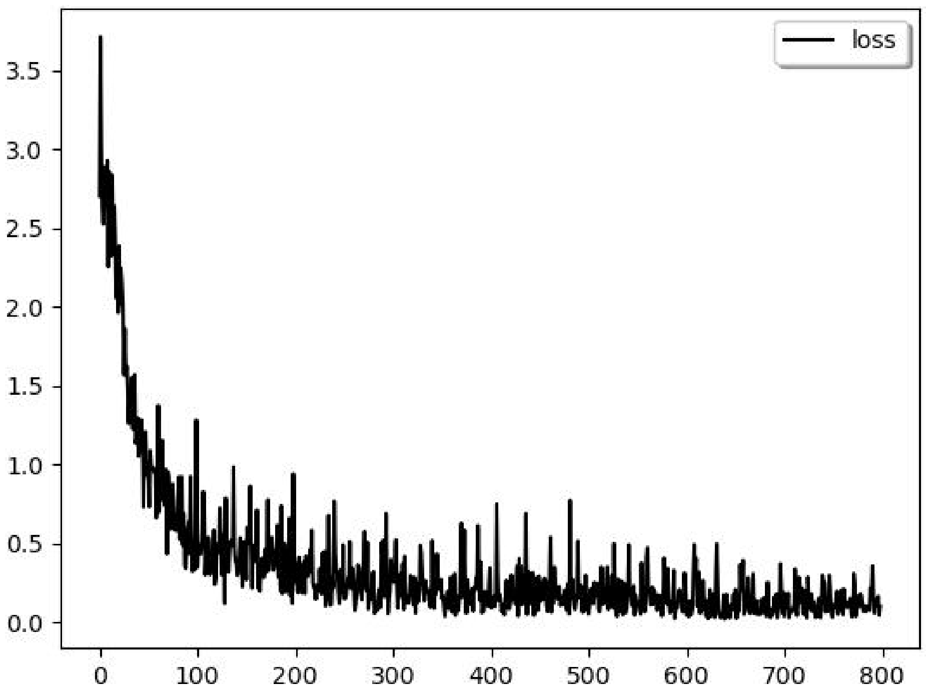




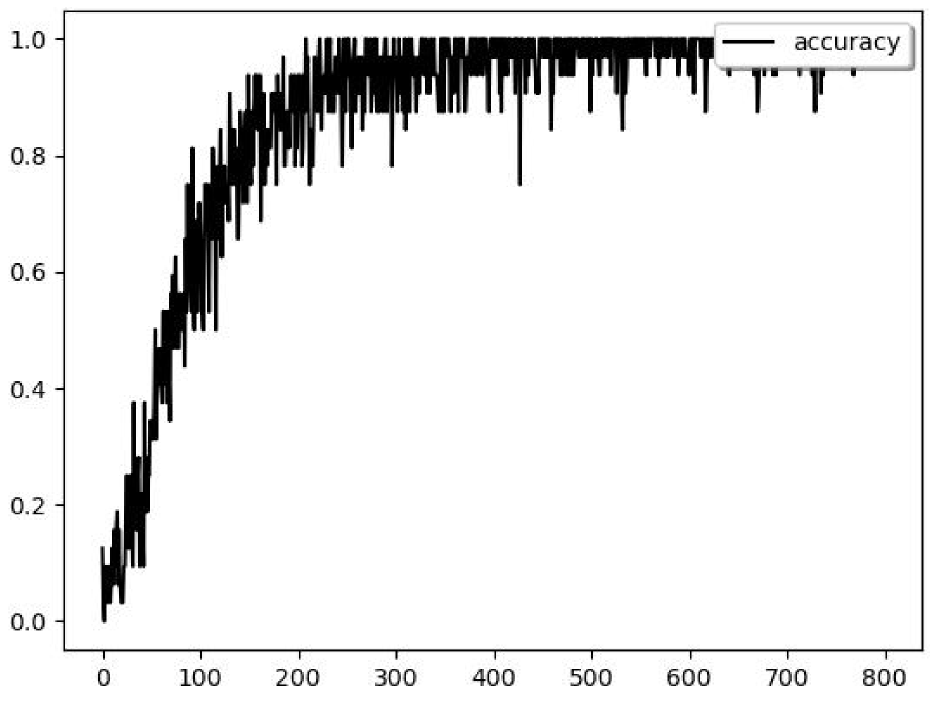
2) nHidden = 128

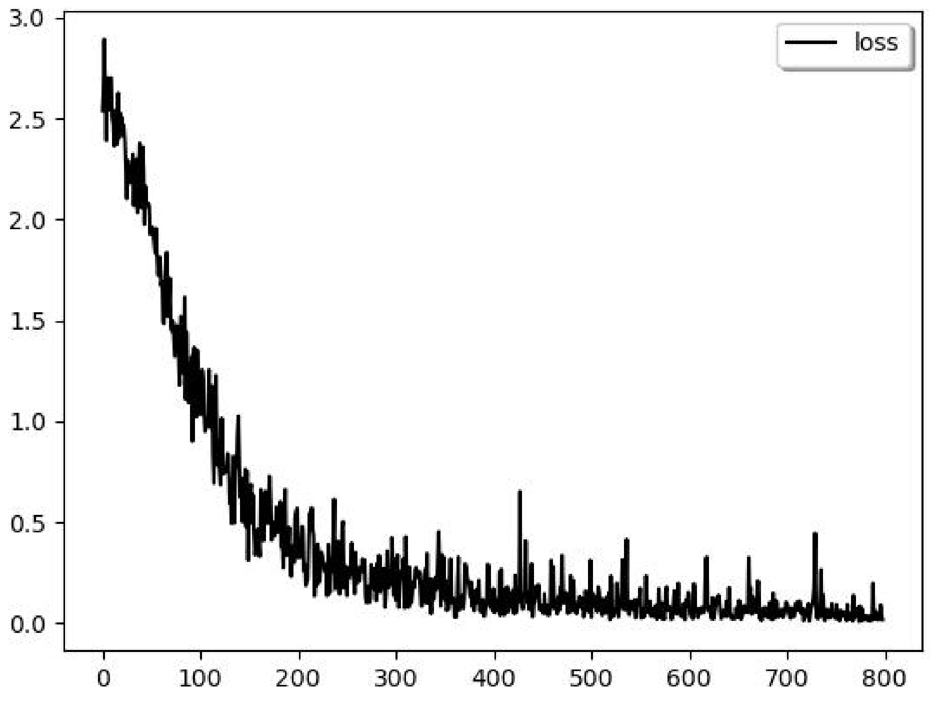
BasicRNNCell: Testing Accuracy: 0.8826



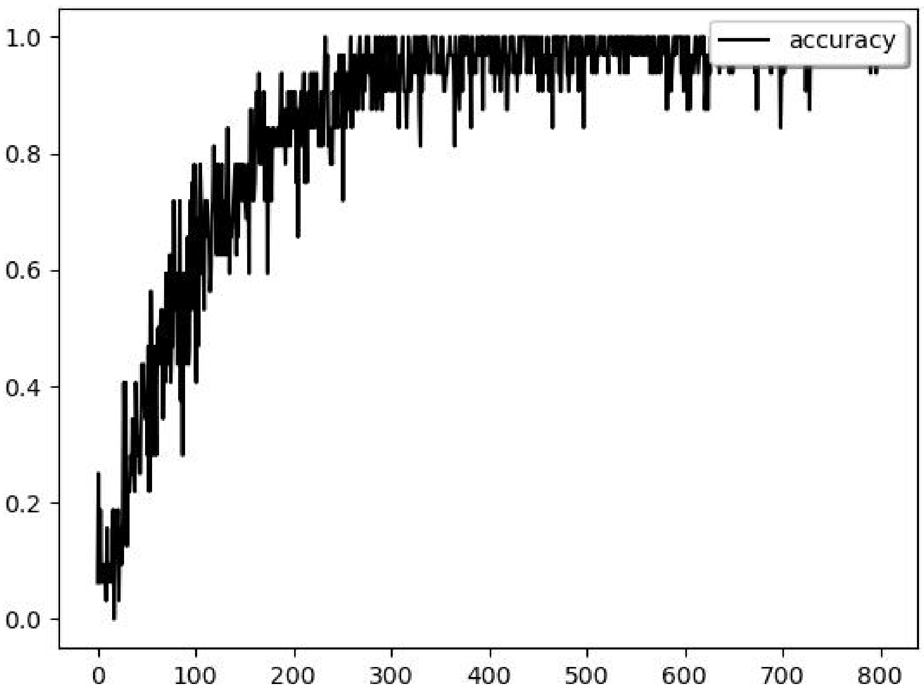


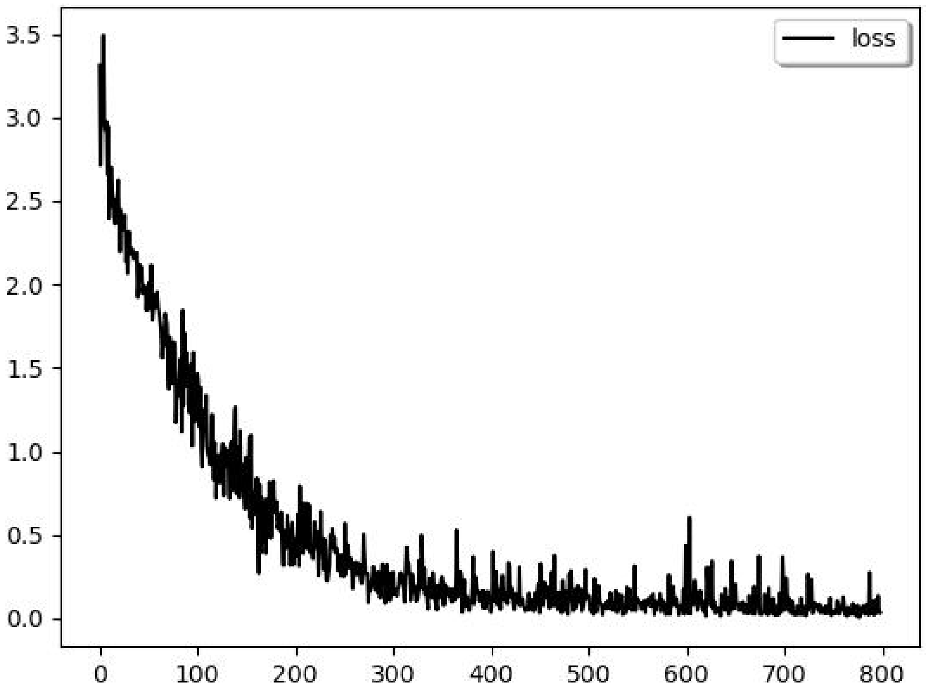
GRUCell: Testing Accuracy: 0.9411





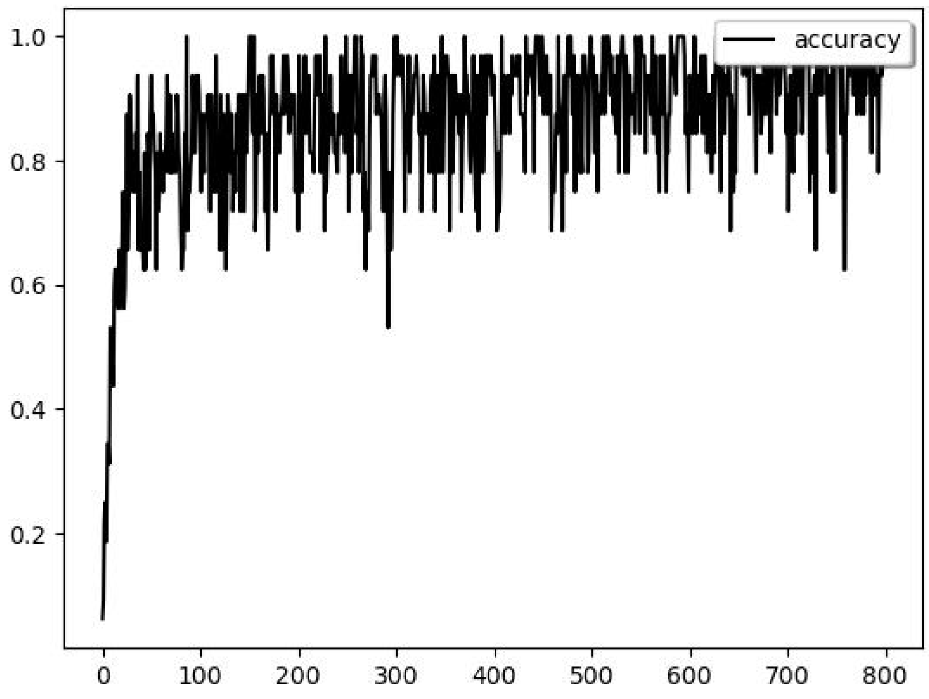
BasicLSTMCell: Testing Accuracy: 0.9395

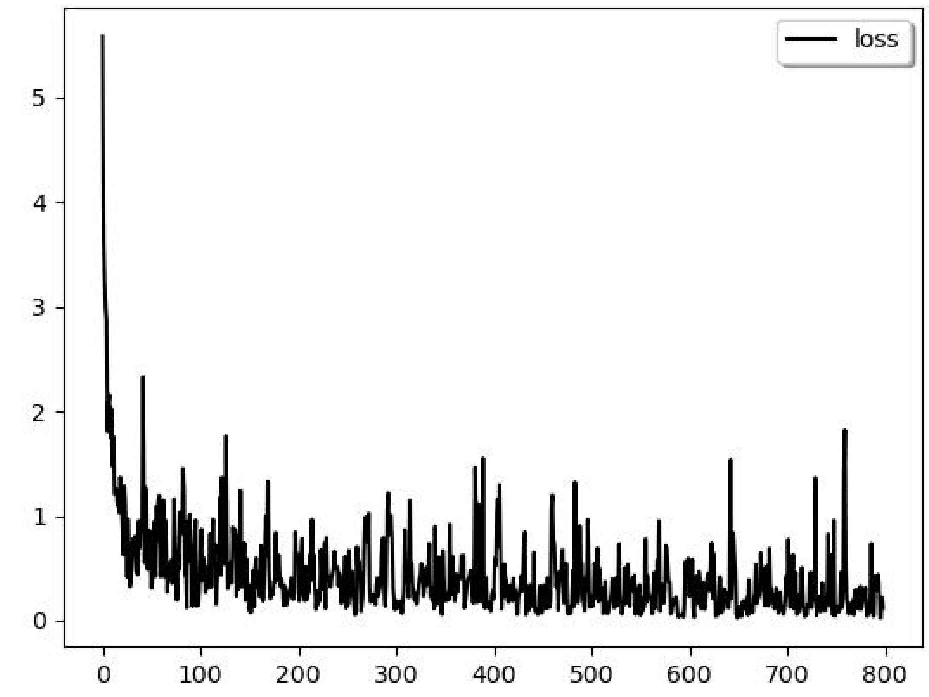




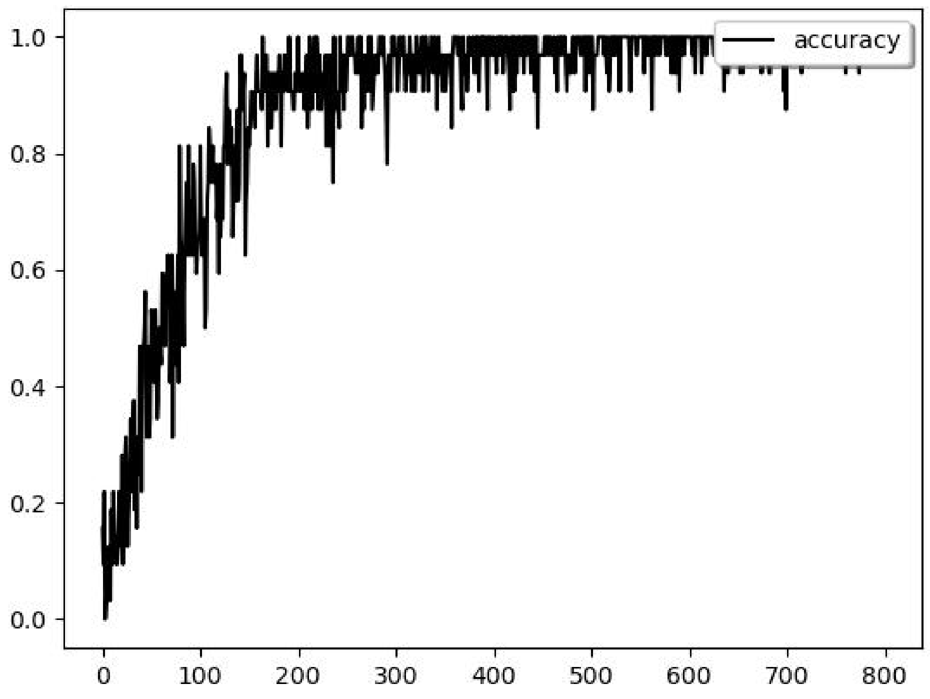
3) nHidden = 256

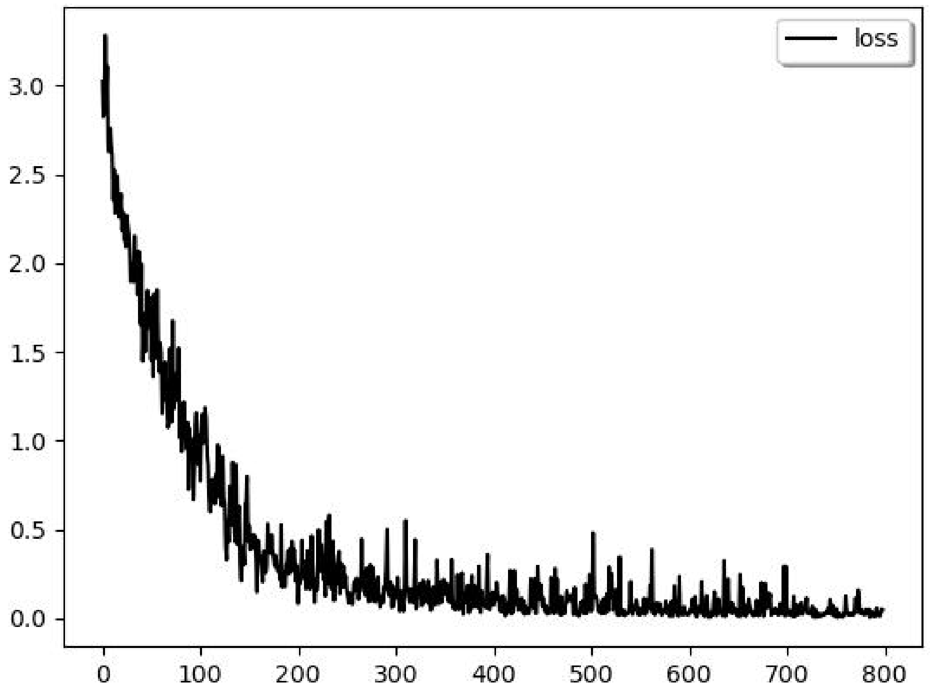
BasicRNNCell: Testing Accuracy: 0.8483



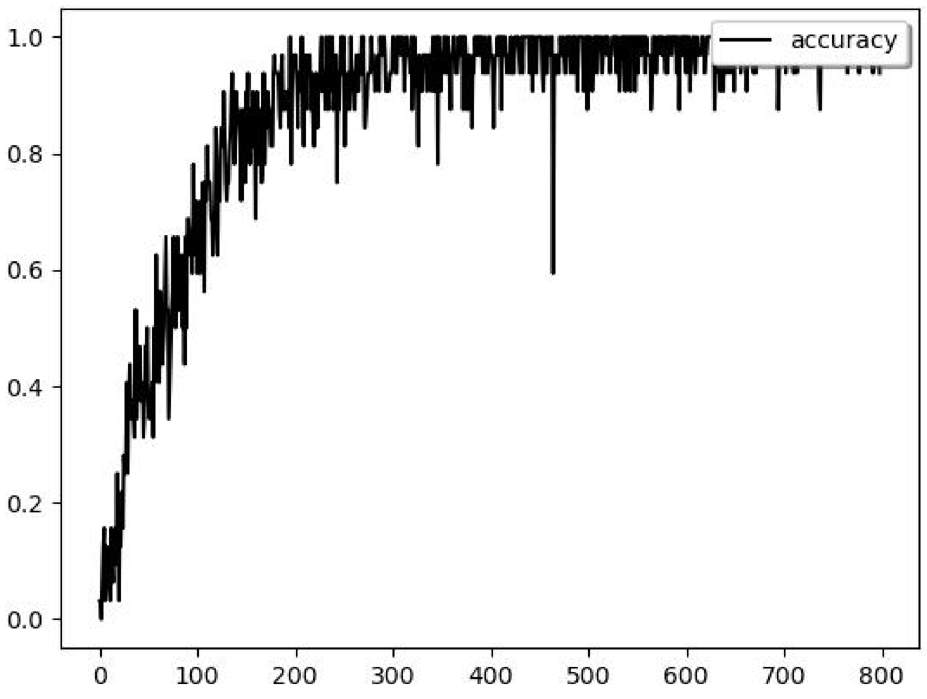


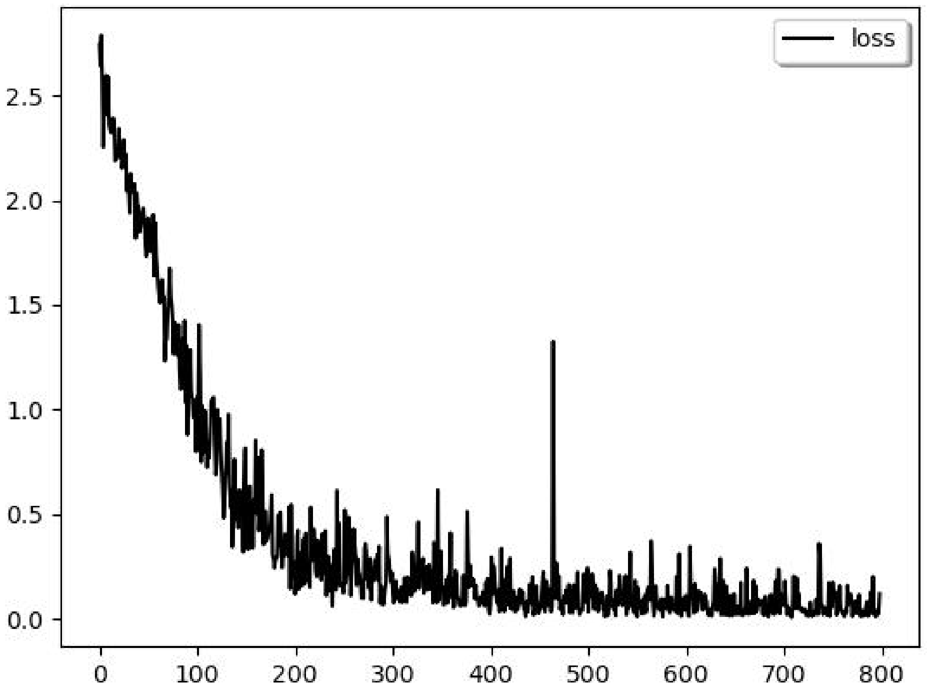
GRUCell: Testing Accuracy: 0.9322





BasicLSTMCell: Testing Accuracy: 0.9117





3.c

Similarities.

1. Both are extensions of traditional neural networks.
2. Forward computation to generate results and backward computation for model updating.
3. Each layer of neural network can have multiple neurons coexisting horizontally and multiple layers of neural networks connected vertically.

Differences.

1. CNN has spatial extension; RNN temporal extension, neurons with multiple temporal outputs computed.
2. RNN can be used to describe the output of a continuous state in time with memory function, while CNN is for static output.
3. CNN ca reach much deeper depth, RNN has limited depth.