Assignment_2

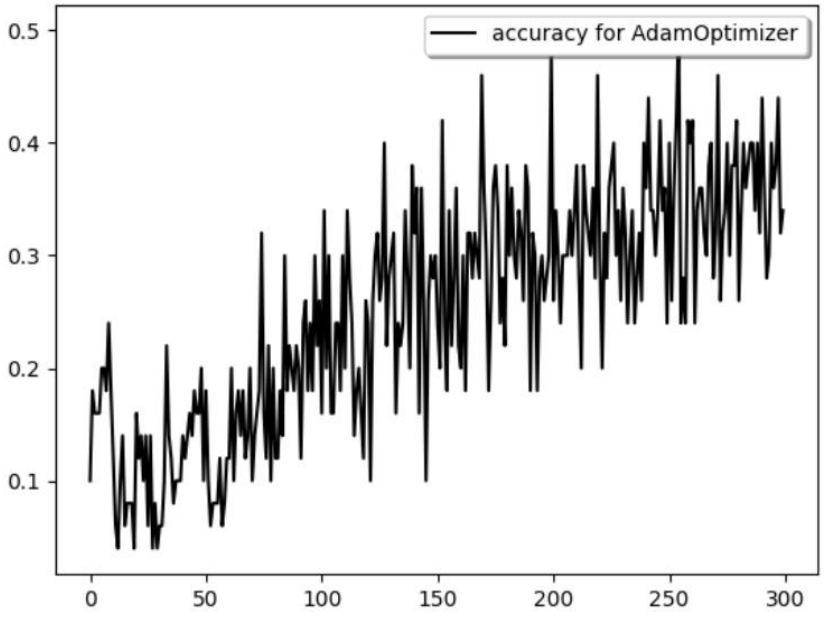
Zewen Xu

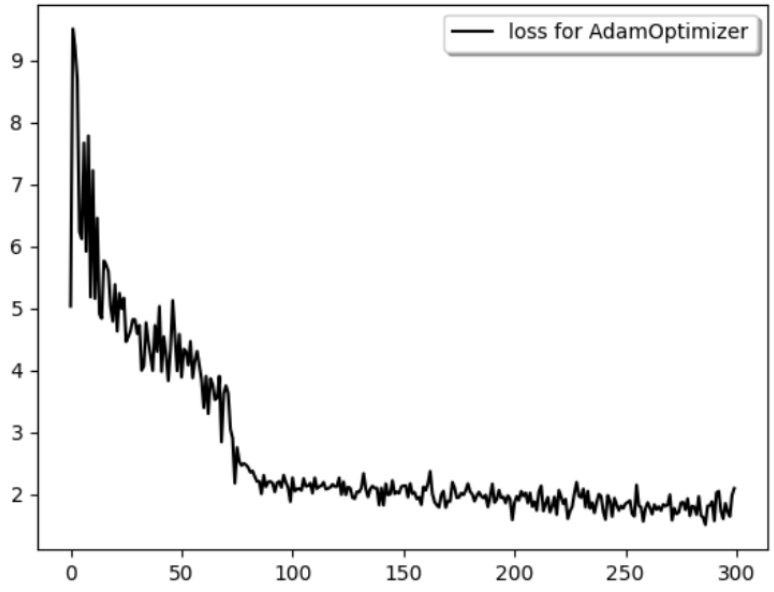
1 . a) b)

By testing(range = 300) the learning rate and optimizers, I've come up with 7 combinations. They're as follows:

1. AdamOptimizer with e^{-3} learning rate :

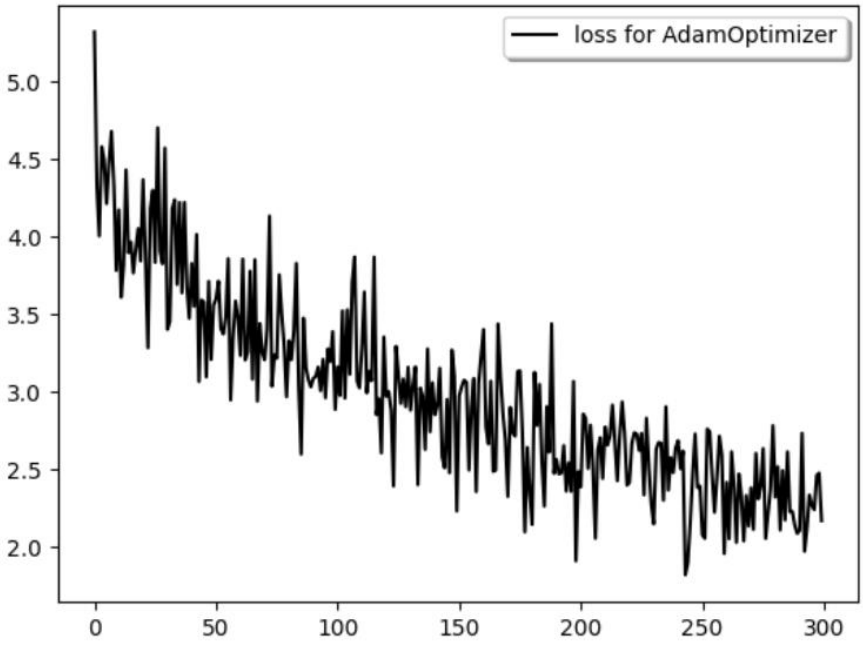
270th step, loss is 1.4507496356964111, train accuracy is 0.46000000834465027
280th step, loss is 1.6094765663146973, train accuracy is 0.47999998927116394
290th step, loss is 1.3954119682312012, train accuracy is 0.5
test accuracy 0.384
activation1: mean 0.0562139, variance 0.00547798
activation2: mean 0.00188613, variance 0.00033134

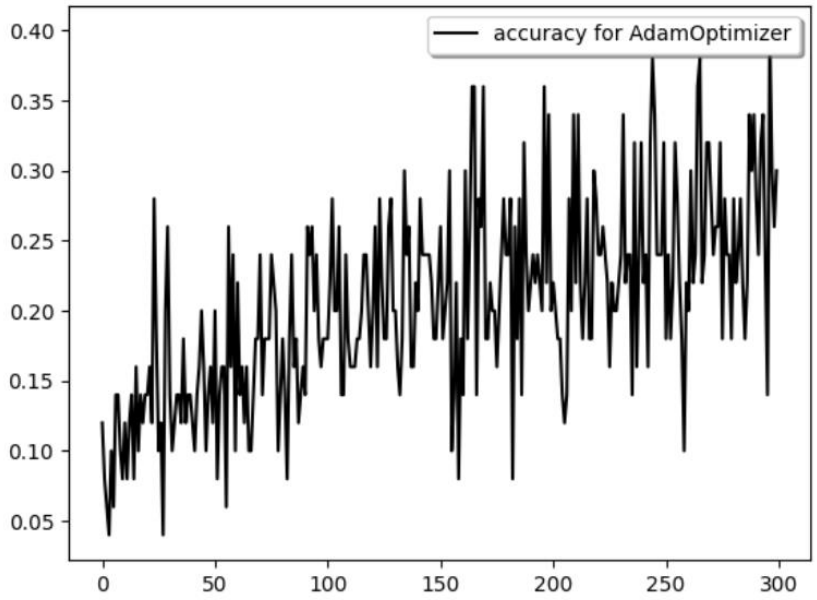




2. AdamOptimizer with e^{-4} learning rate :

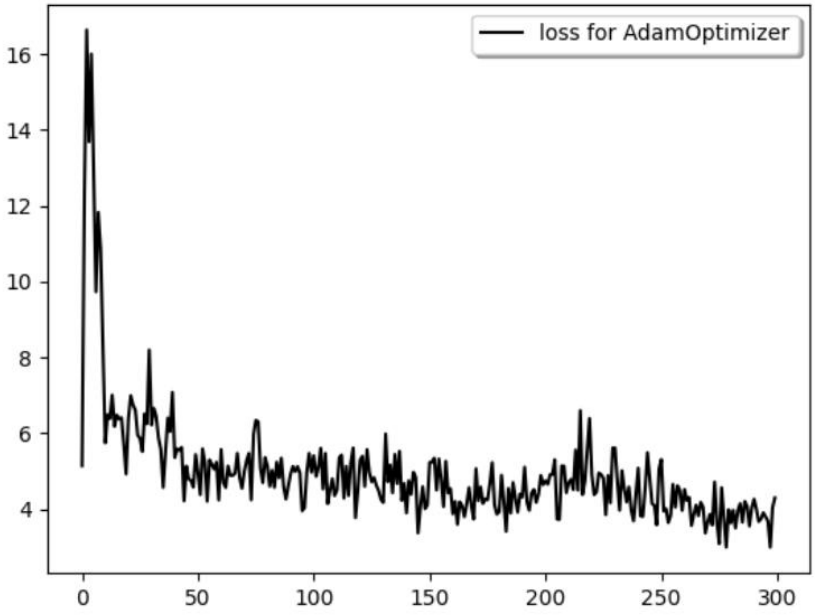
270th step, loss is 2.2398884296417236, train accuracy is 0.30000001192092896
280th step, loss is 2.9213852882385254, train accuracy is 0.1599999964237213
290th step, loss is 2.4343132972717285, train accuracy is 0.18000000715255737
test accuracy 0.303
activation1: mean 0.133522, variance 0.0152843
activation2: mean 0.0977329, variance 0.0239588

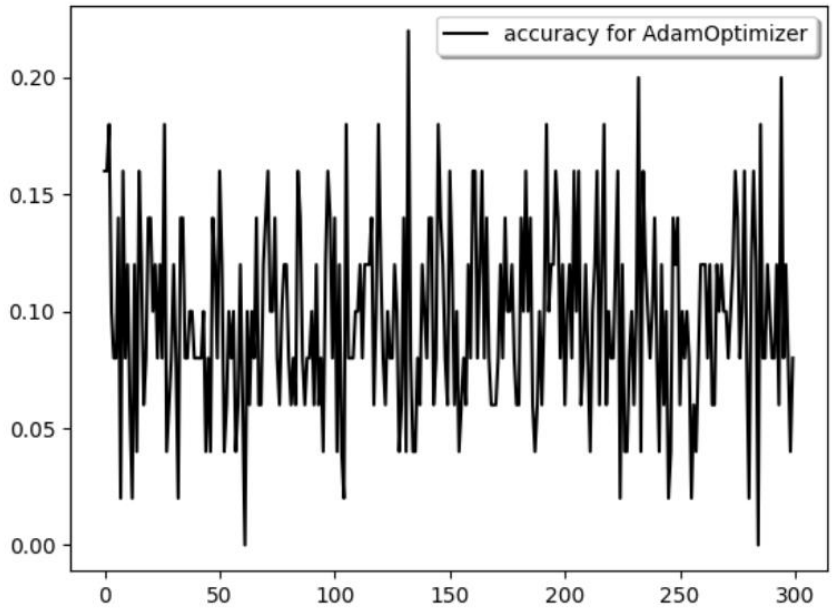




3. AdamOptimizer with e^{-2} learning rate :

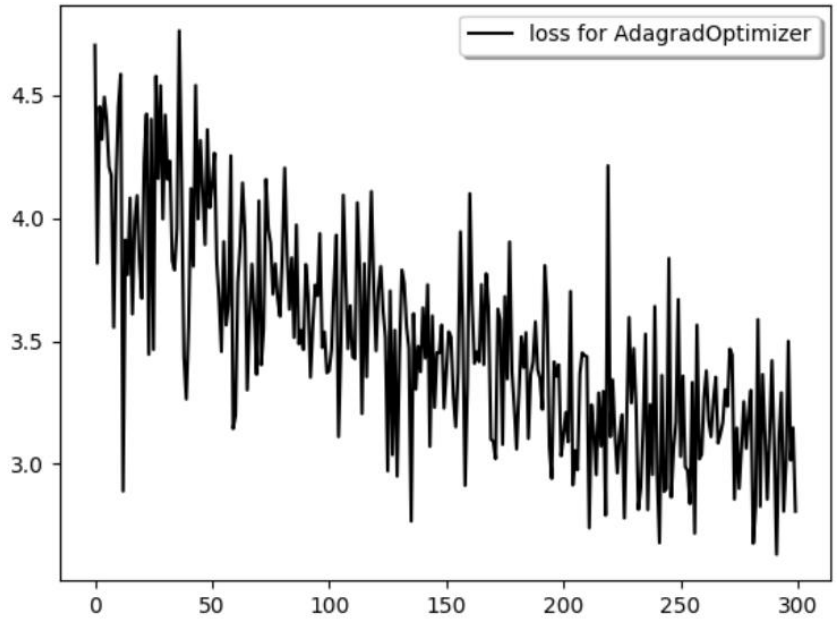
280th step, loss is 3.639279365539551, train accuracy is 0.019999999552965164
290th step, loss is 4.2674784660339355, train accuracy is 0.07999999821186066
test accuracy 0.1
activation1: mean 0.365196, variance 0.211727
activation2: mean 6.2687, variance 69.472

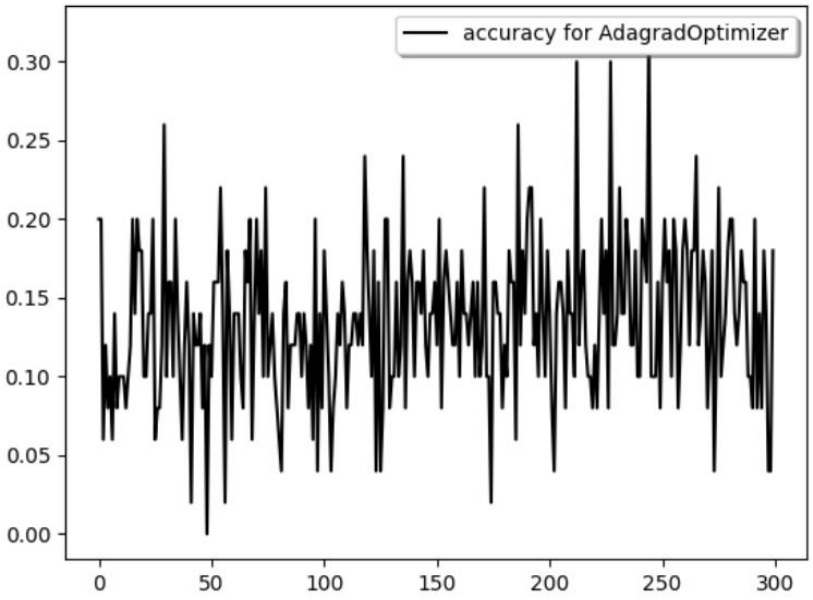




4. AdagradOptimizer with e^{-3} learning rate

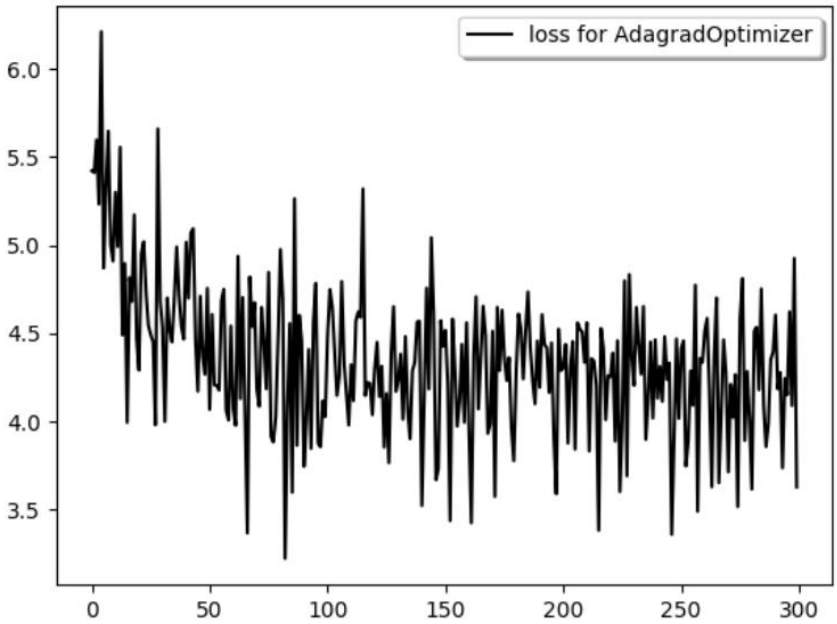
270th step, loss is 3.8379430770874023, train accuracy is 0.07999999821186066
280th step, loss is 2.7121284008026123, train accuracy is 0.23999999463558197
290th step, loss is 3.530604839324951, train accuracy is 0.14000000059604645
test accuracy 0.264
activation1: mean 0.113943, variance 0.0208963
activation2: mean 0.0795688, variance 0.0197604

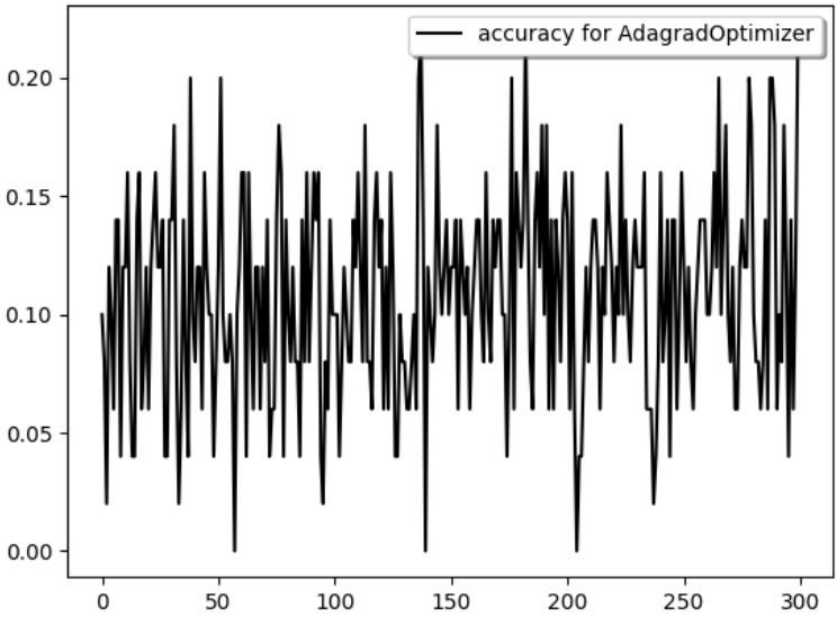




5. AdagradOptimizer with e^{-4} learning rate

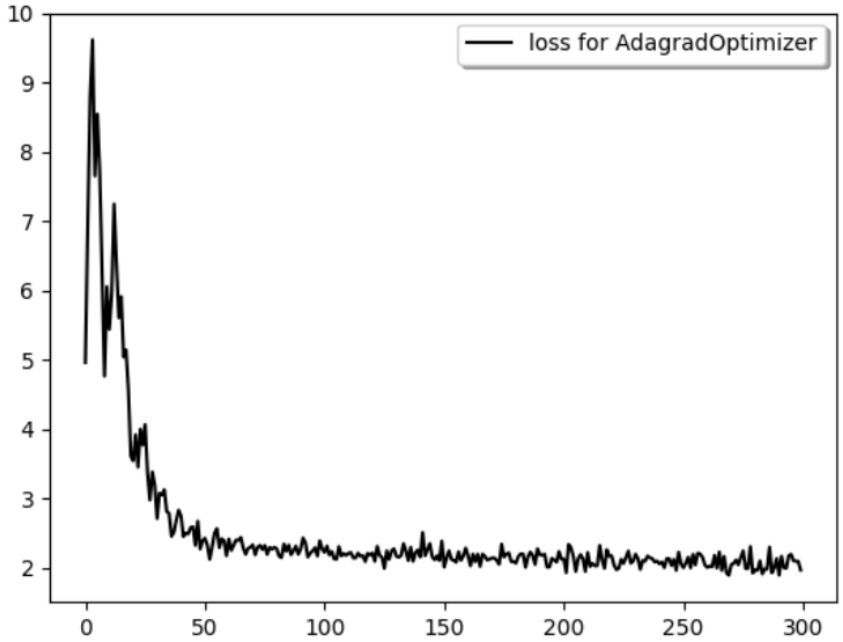
270th step, loss is 3.8292064666748047, train accuracy is 0.14000000059604645
280th step, loss is 3.767287254333496, train accuracy is 0.11999999731779099
290th step, loss is 4.227090835571289, train accuracy is 0.07999999821186066
test accuracy 0.136
activation1: mean 0.180201, variance 0.033469
activation2: mean 0.250849, variance 0.167544

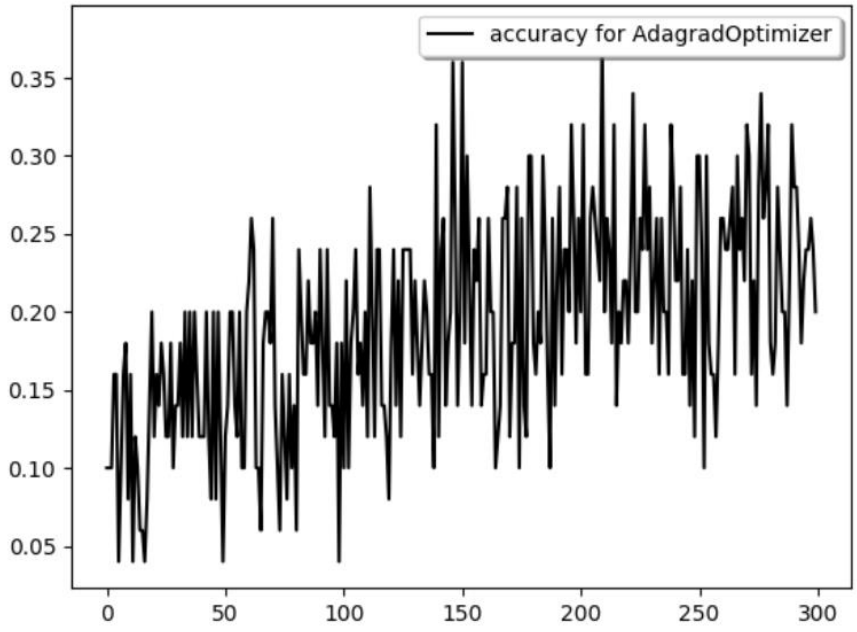




6. AdagradOptimizer with e^{-2} learning rate

270th step, loss is 1.9175626039505005, train accuracy is 0.3199999928474426
280th step, loss is 1.989504098892212, train accuracy is 0.30000001192092896
290th step, loss is 1.992216944694519, train accuracy is 0.30000001192092896
test accuracy 0.324
activation1: mean 0.0382021, variance 0.0036608
activation2: mean 0.00494298, variance 0.00055867

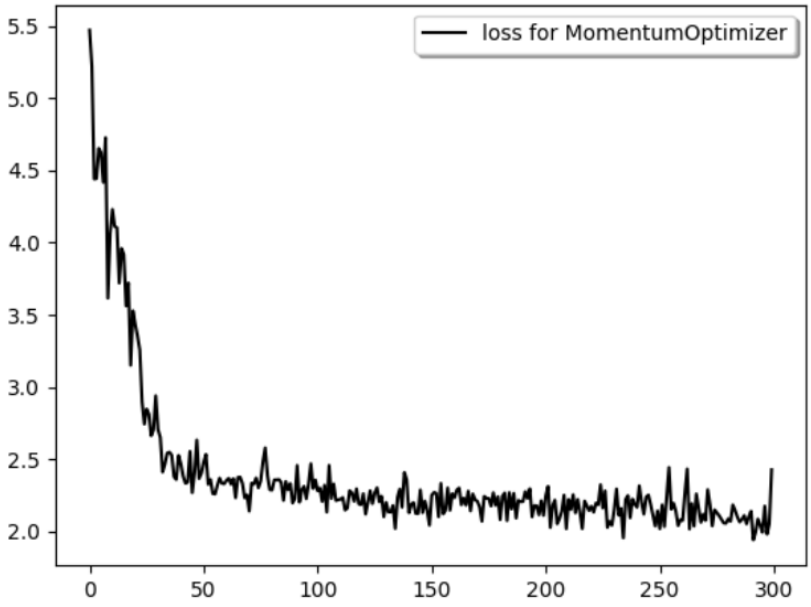


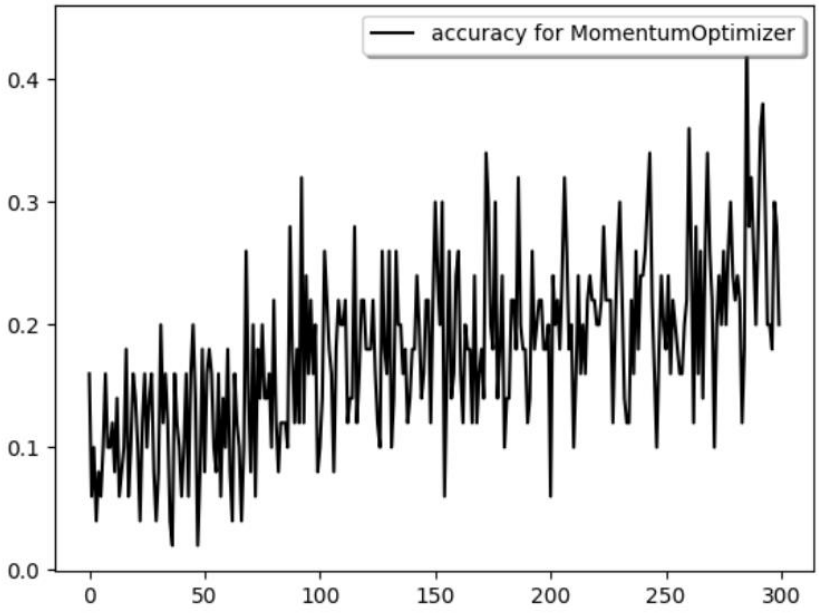


7. MomentumOptimizer

different combination between momentum rate and learning rate have been executed. However, out of limited report page, only the best outcome of them is to displayed. It's 0.5 momentum rate with e^{-2} learning rate.

270th step, loss is 2.065484046936035, train accuracy is 0.3199999928474426
280th step, loss is 2.1424686908721924, train accuracy is 0.23999999463558197
290th step, loss is 1.9169467687606812, train accuracy is 0.30000001192092896
test accuracy 0.295
activation1: mean 0.0381124, variance 0.0076643
activation2: mean 0.00357606, variance 0.000508919



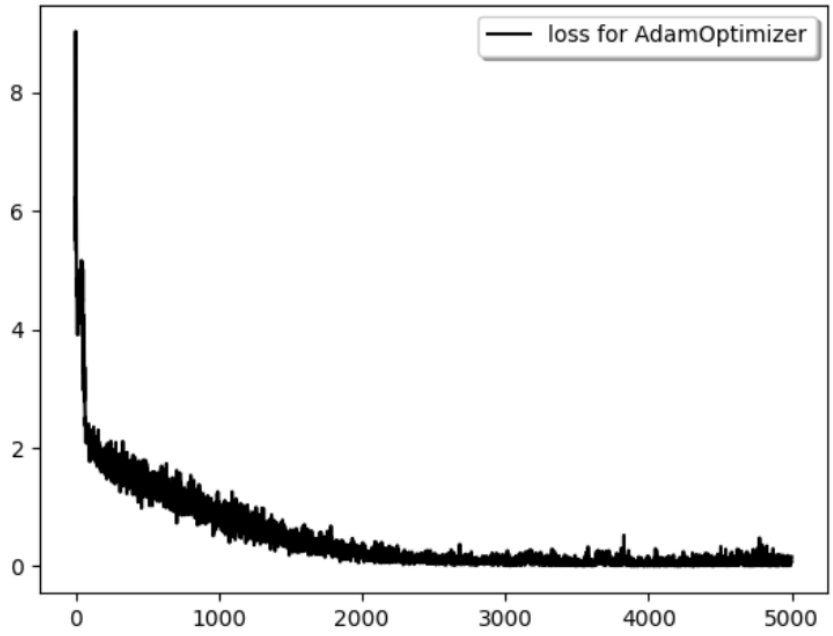


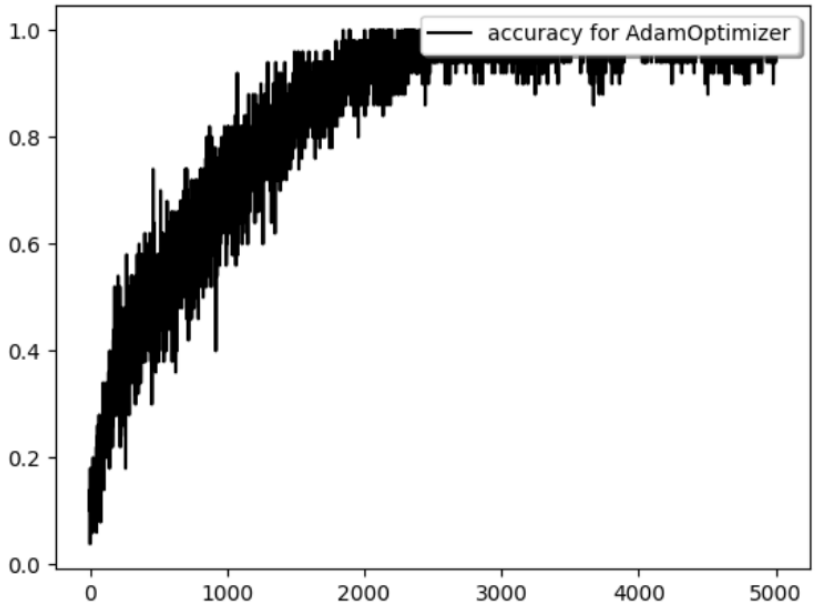
8. Summary

through the displayed comparison, the AdamOptimizer with e^{-3} learning rate performs the best. Then resize the range to 5000.

Train 4970th step, loss is 0.08374033868312836, train accuracy is 0.9599999785423279
4980th step, loss is 0.08861750364303589, train accuracy is 0.9800000190734863
4990th step, loss is 0.07235841453075409, train accuracy is 0.9800000190734863

test accuracy 0.417
activation1: mean 0.0335829, variance 0.00541334
activation2: mean 0.00481346, variance 0.00161275

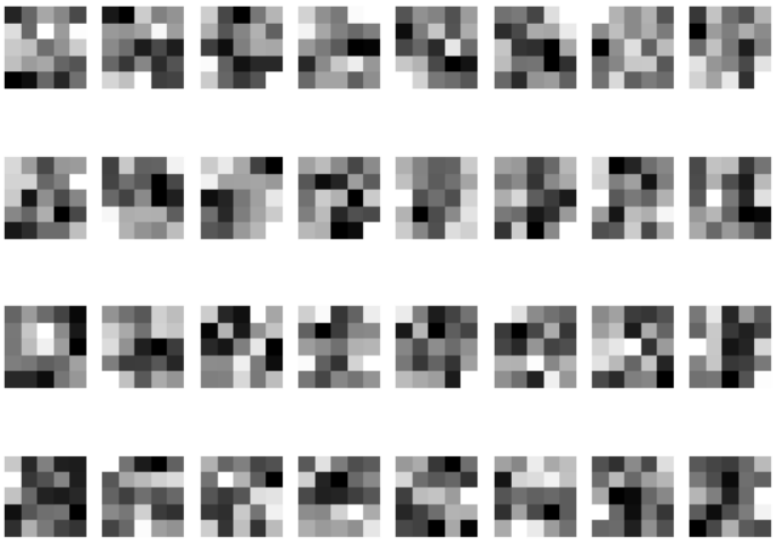




It's obvious that the cooperation between AdamOptimizer and e^{-3} learning rate could achieve a good outcome. The process almost become stable after 2000 steps learning.

1 . c)

Visualization of the first layer's weights:



activation1: mean 0.0335829, variance 0.00541334
activation2: mean 0.00481346, variance 0.00161275

2

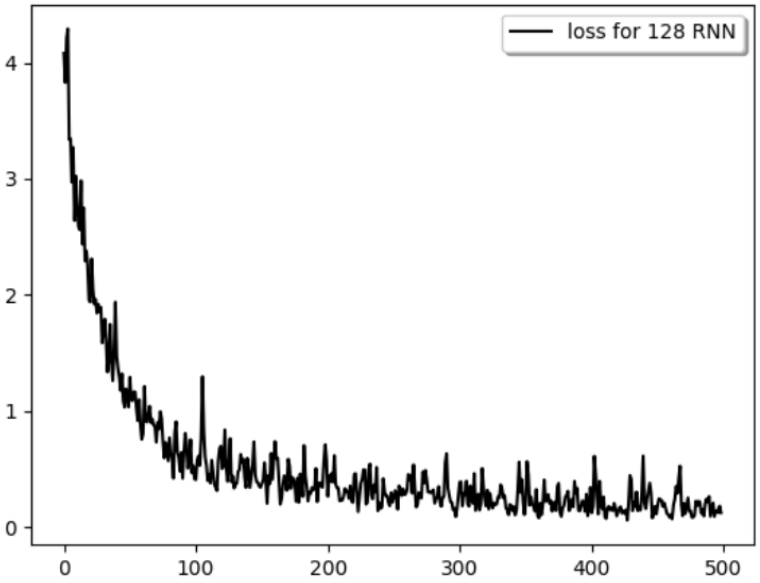
Summary:

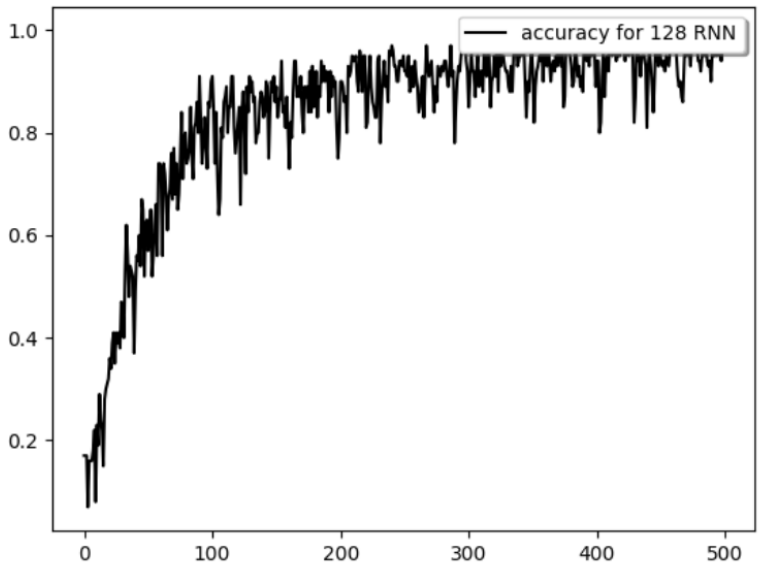
In this paper, the authors present a novel visualization scheme to help understand existing methodologies, which involves deconvnet visualization, unpooling.The authors visualize feature activation on the ImageNet validation set. This paper is useful for work on feature visualization, feature evolution during training, and feature invariance. It is also inspiring for our future model selection and parameter tuning. Also, they mentioned a experiment they've carried out on ImageNet 2012, where they worked on the feature extraction, and deep into structures, and analyze the discriminatory ability in Imagenet-retrained model.

3 a)b)

1.1 Learning rate – 128, RMSPropOptimizer

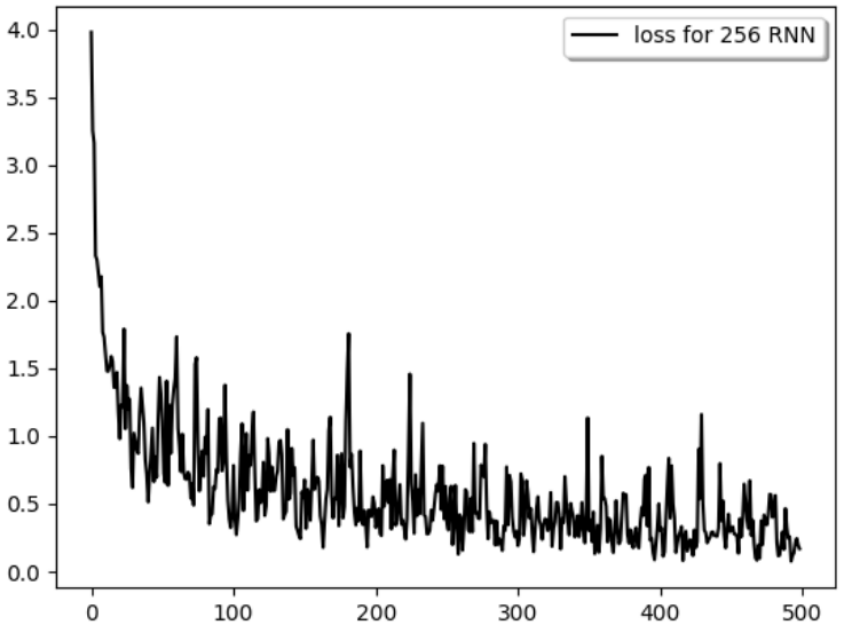
Iter 47000, Minibatch Loss= 0.098883, Training Accuracy= 0.99000
Iter 48000, Minibatch Loss= 0.224050, Training Accuracy= 0.94000
Iter 49000, Minibatch Loss= 0.264112, Training Accuracy= 0.90000
Optimization finished
Testing Accuracy: 0.8898

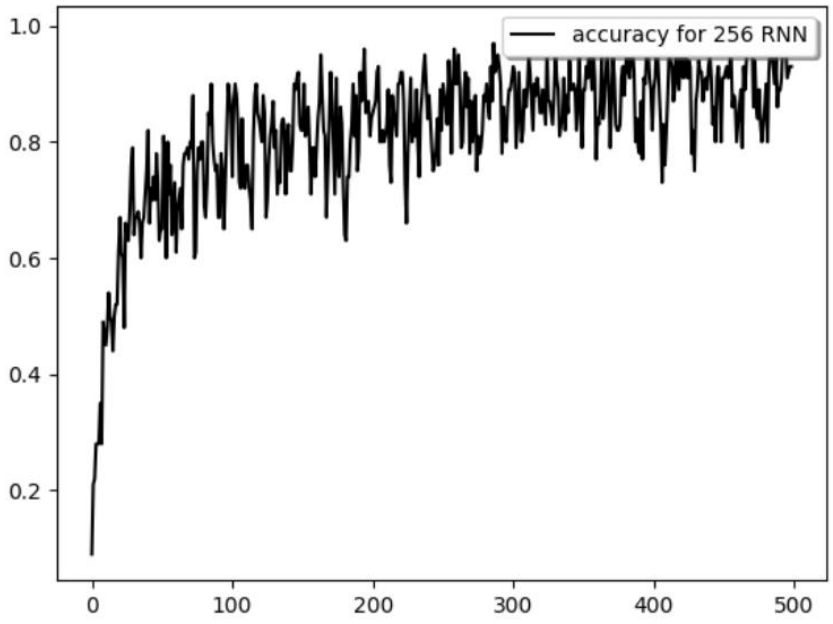




1.2 Learning rate – 256, RMSPropOptimizer

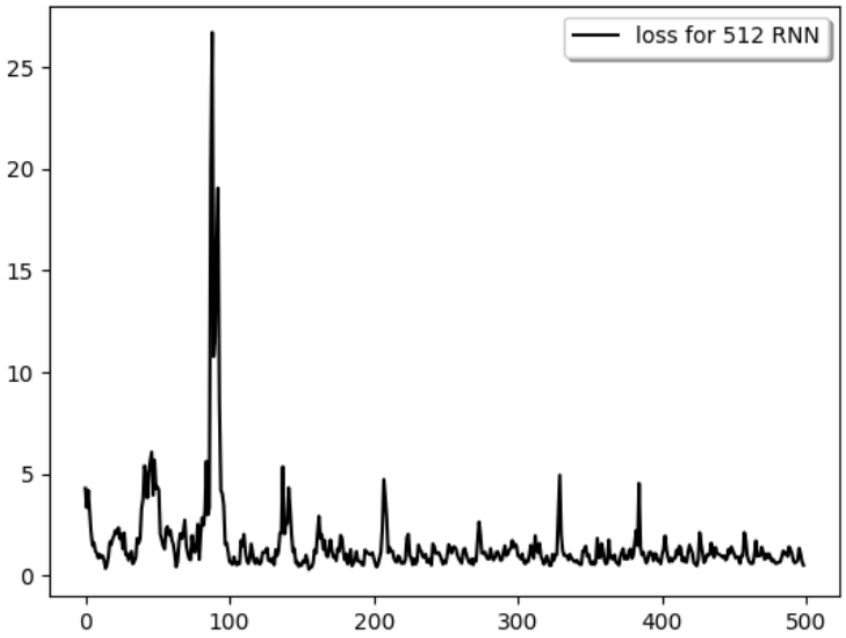
Iter 47000, Minibatch Loss= 0.195663, Training Accuracy= 0.96000
Iter 48000, Minibatch Loss= 0.401581, Training Accuracy= 0.86000
Iter 49000, Minibatch Loss= 0.325927, Training Accuracy= 0.89000
Optimization finished
Testing Accuracy: 0.854

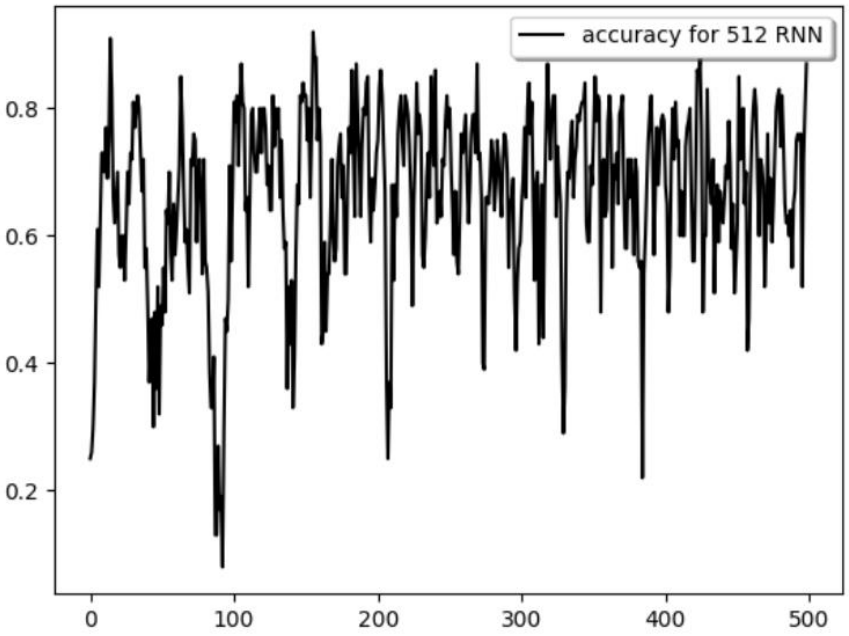




1.3 Learning rate – 512, RMSPropOptimizer

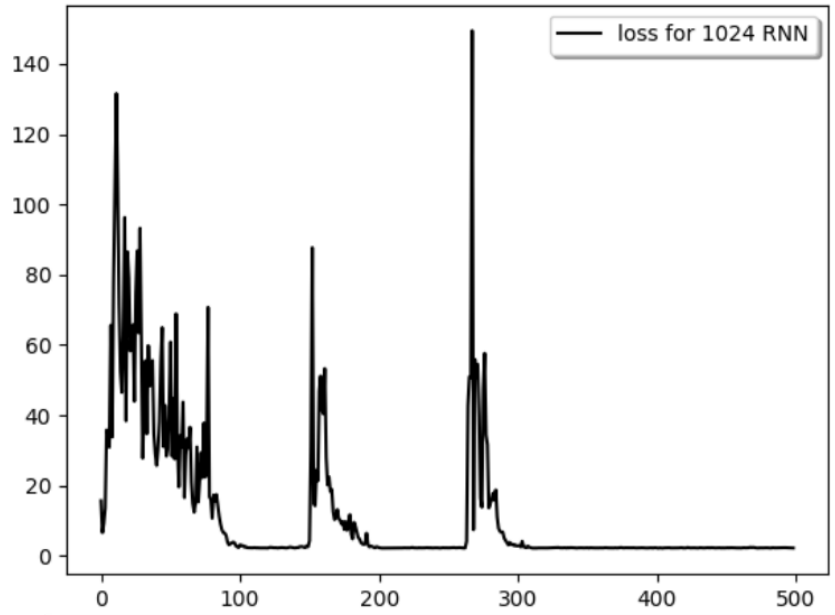
Iter 47000, Minibatch Loss= 1.401921, Training Accuracy= 0.52000
Iter 48000, Minibatch Loss= 0.592011, Training Accuracy= 0.83000
Iter 49000, Minibatch Loss= 1.271250, Training Accuracy= 0.65000
Optimization finished
Testing Accuracy: 0.6321

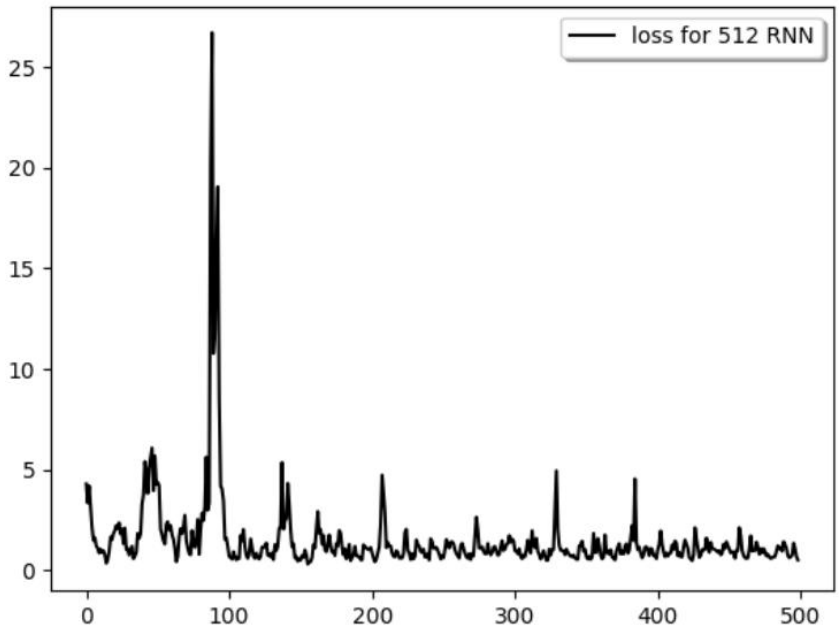




1.4 Learning rate 1024, RMSPropOptimizer

Iter 47000, Minibatch Loss= 2.334334, Training Accuracy= 0.14000
Iter 48000, Minibatch Loss= 2.342498, Training Accuracy= 0.14000
Iter 49000, Minibatch Loss= 2.349804, Training Accuracy= 0.14000
Optimization finished
Testing Accuracy: 0.1028



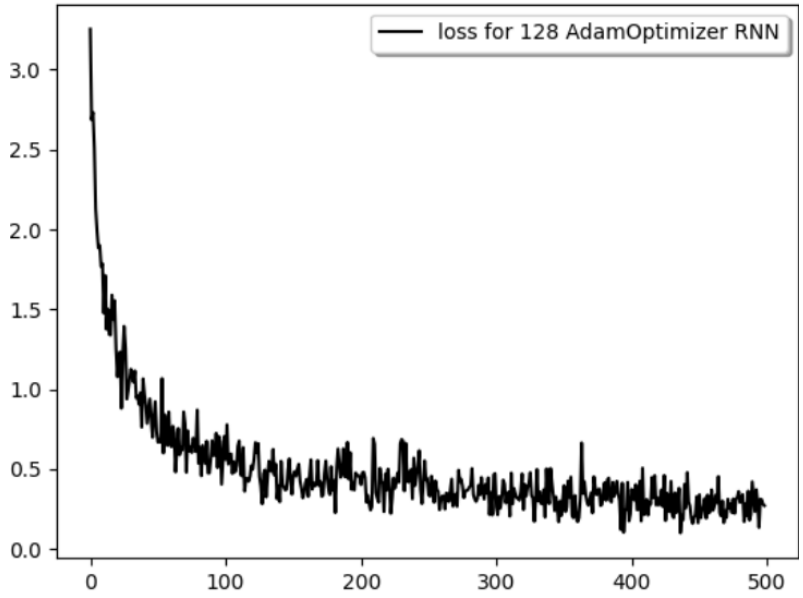


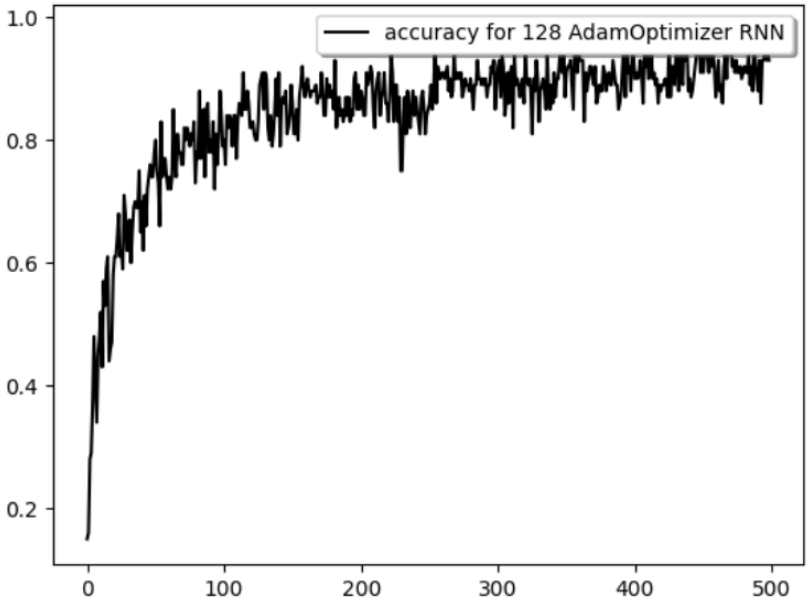
1.5 Summary

The accuracy rate and learning spped stays the best when learning rate is 128.

2.1 Learning rate – 128, AdamOptimizer

Iter 47000, Minibatch Loss= 0.240110, Training Accuracy= 0.95000
Iter 48000, Minibatch Loss= 0.360987, Training Accuracy= 0.91000
Iter 49000, Minibatch Loss= 0.420948, Training Accuracy= 0.90000
Optimization finished
Testing Accuracy: 0.9135





2.2 Summary

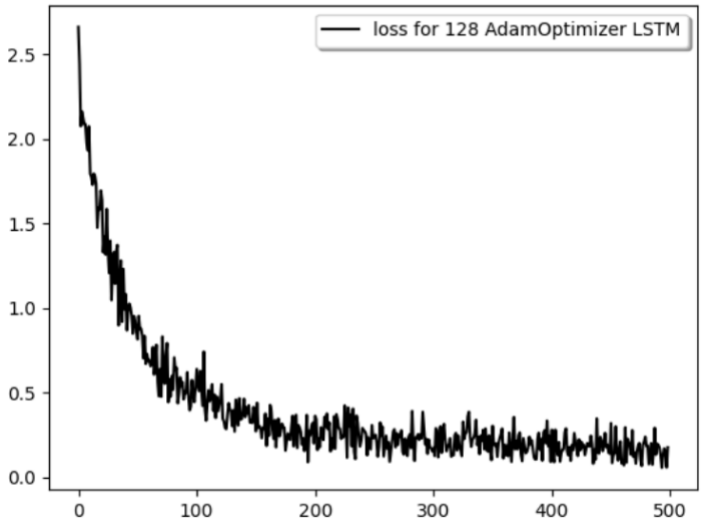
AdamOptimizer outweights RMSPropOptimizer in all rounds. With experiments in part 1, it's decided to adopt AdamOptimizer as the default optimizer.

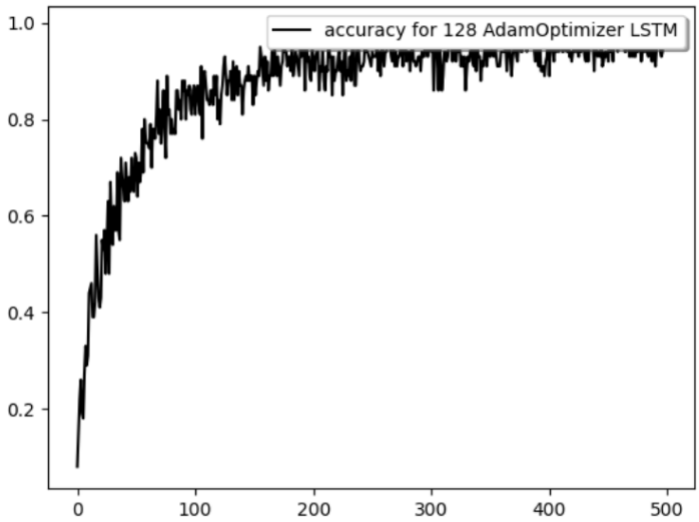
3. According to the results, the parameters are set as follows:

- Number of nodes in the hidden layer: 128(best), 256, 512, 1024
- Learning rate: e^{-3}
- Number of iterations: 50000
- Cost (hint use softmax cross entropy with logits): softmax cross entropy
- Optimizer: RMSPropOptimizer, AdamOptimizer(best)

4.1 LSTM

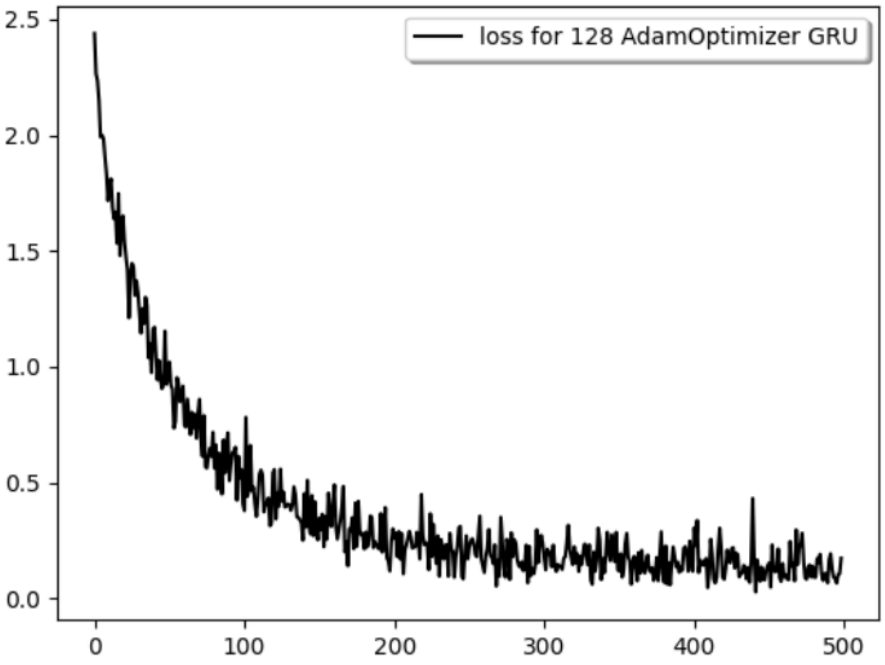
Iter 47000, Minibatch Loss= 0.119915, Training Accuracy= 0.96000
Iter 48000, Minibatch Loss= 0.237956, Training Accuracy= 0.94000
Iter 49000, Minibatch Loss= 0.195998, Training Accuracy= 0.93000
Optimization finished
Testing Accuracy: 0.9505

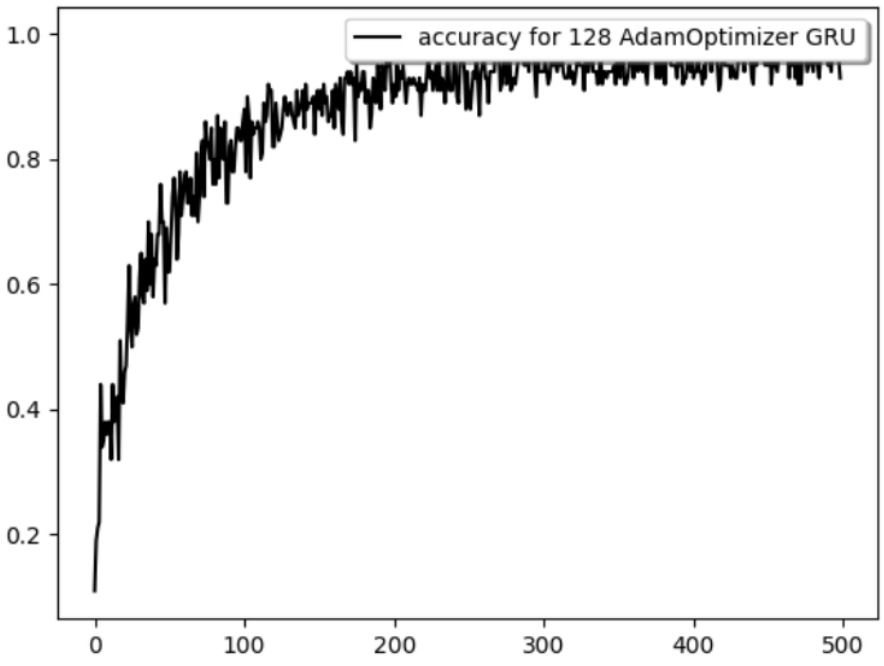




4.2 GRU

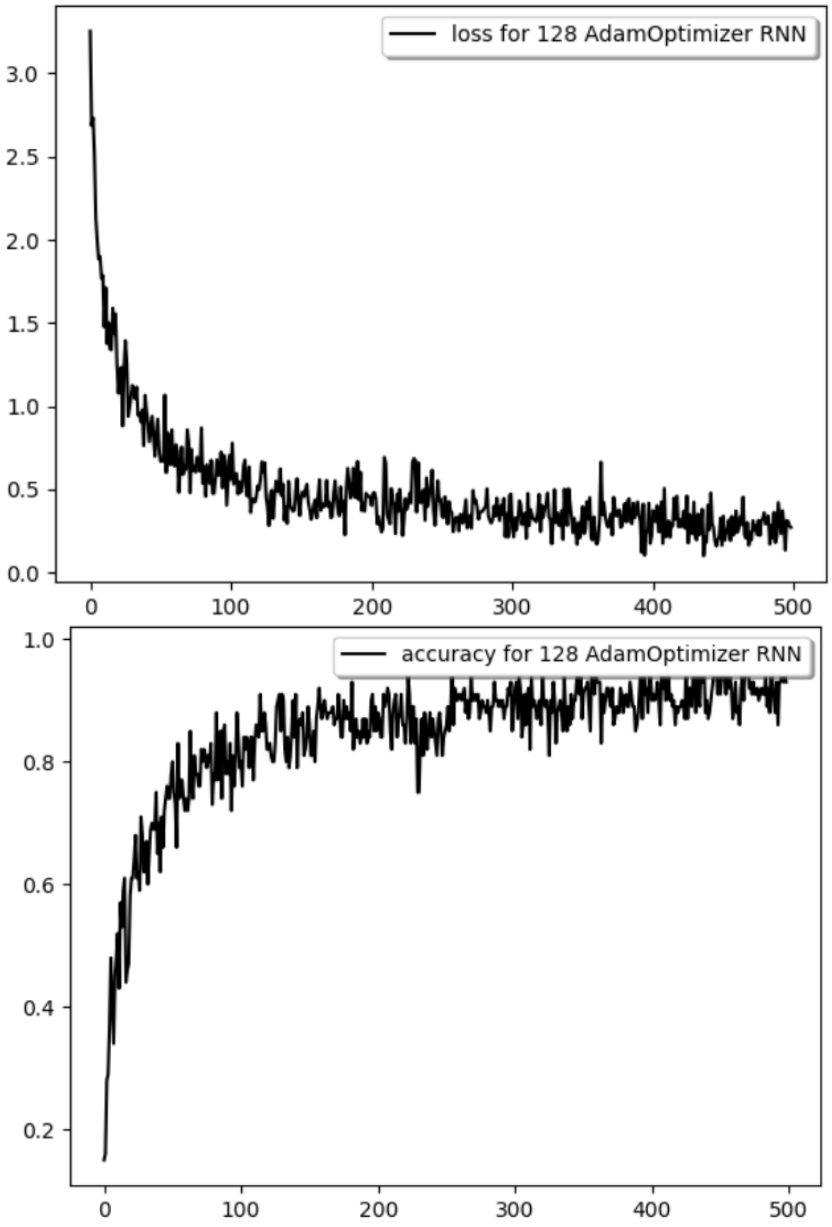
Iter 47000, Minibatch Loss= 0.146785, Training Accuracy= 0.97000
Iter 48000, Minibatch Loss= 0.138639, Training Accuracy= 0.96000
Iter 49000, Minibatch Loss= 0.067714, Training Accuracy= 0.99000
Optimization finished
Testing Accuracy: 0.9572





4.3 RNN

Iter 47000, Minibatch Loss= 0.098883, Training Accuracy= 0.99000
Iter 48000, Minibatch Loss= 0.224050, Training Accuracy= 0.94000
Iter 49000, Minibatch Loss= 0.264112, Training Accuracy= 0.90000
Optimization finished
Testing Accuracy: 0.8898



4.4 Summary

The LSTM and GRU shows much better result in testing accuracy. On the other hand, three of them show almost the samelearnign speed as the iteration going on, and LSTM may be a little overfitting. In all, LSTM and GRU are beteer.

3c)

Comparision:

After comparing the above outcomes and analyzing with the convnet in assignment 1, RNNs are better at handling dependencies between lines, while CNNs are slightly better at handling complex hierarchies (layers).

RNN seems to be more efficient in terms of learning speed and final testing accuracy, although RNN tends to be overfitting in some accasions. CNN could be used to imitate some “human” processing while RNN tends to be adopted to process monotype information like images and audios.