RICE UNIVERSITY, DEPARTMENT OF COMPUTER SCIENCE

Assignment 2, COMP 576

Chen Zeng(cz39)

October 15, 2018

1.b

There are three sets of experiments to search for good hyper-parameters:

- learning rate: 1e-4, 1e-3, 1e-2
- training methods: AdamOptimizer, GradientDescentOptimizer

The results are shown below and the final best hyper-parameters are:

- learning rate: 1e-3
- training methods: AdamOptimizer

As for MomentumOptimizer, it's a little bit complex to set the parameters in it's learning_rate. Therefore, here we don't discuss too much on this kind of training method.

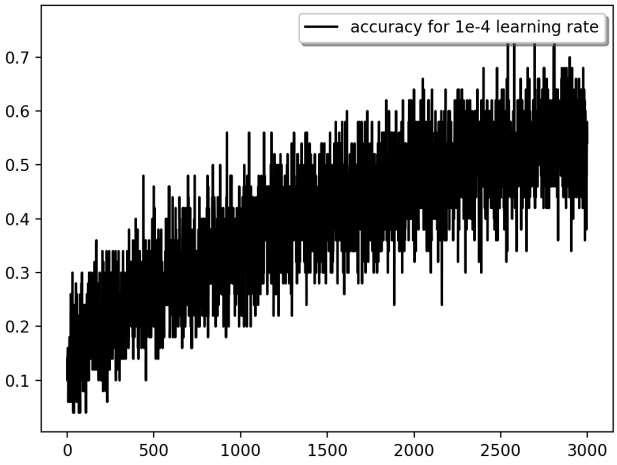

Figure 0.1: learning_rate=1e-4

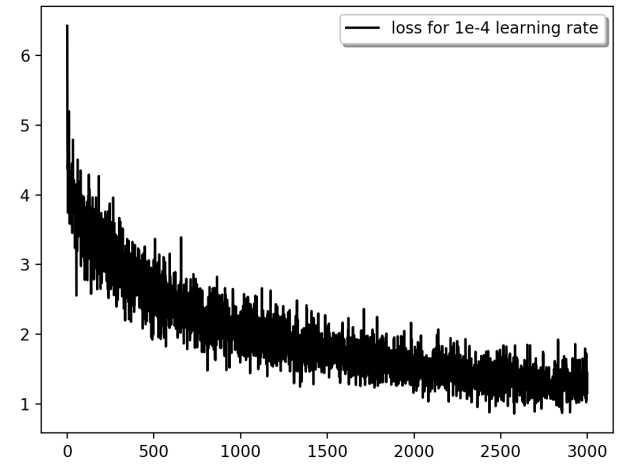

Figure 0.2: learning_rate=1e-4

```
test accuracy 0.427
```

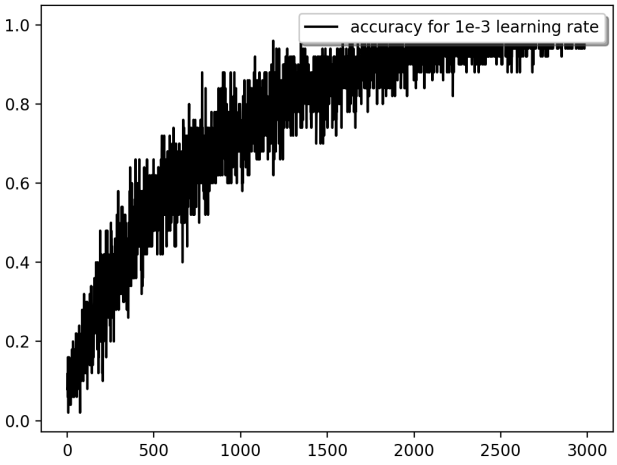

Figure 0.3: learning_rate=1e-3

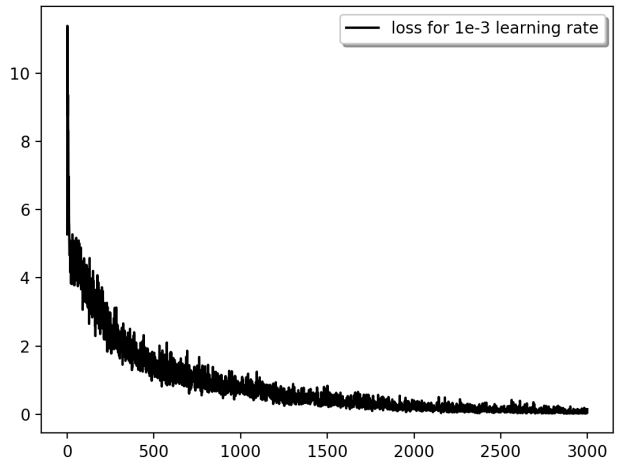

Figure 0.4: learning_rate=1e-3

```
test accuracy 0.456
```

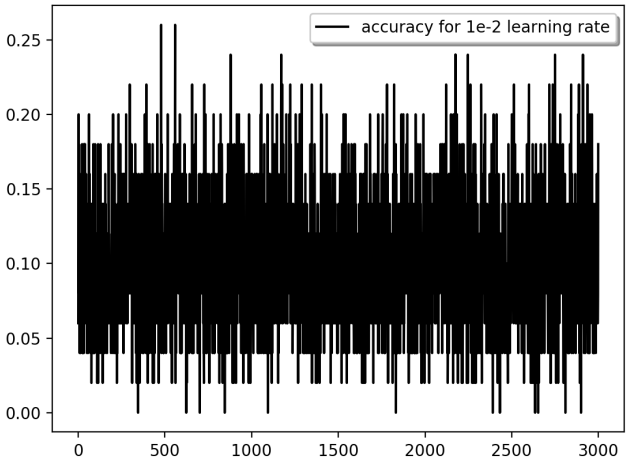

Figure 0.5: learning_rate=1e-2

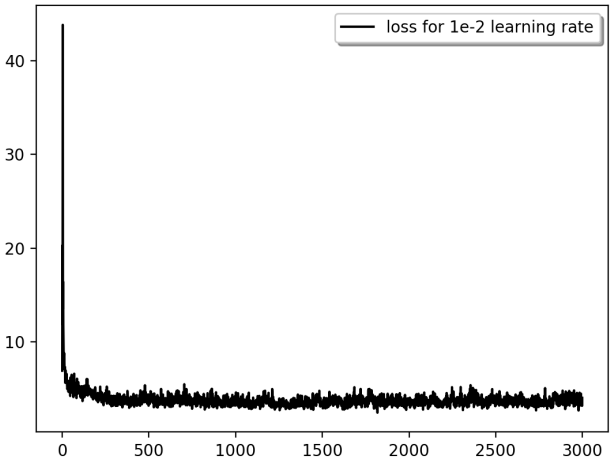

Figure 0.6: learning_rate=1e-2

```
test accuracy 0.1
```

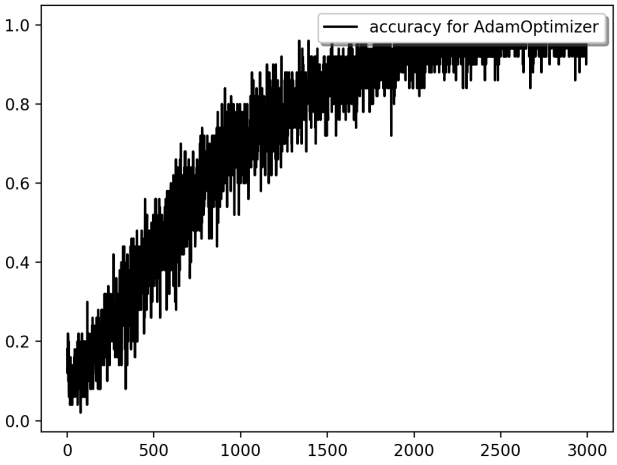

Figure 0.7: learning_method=AdamOptimizer

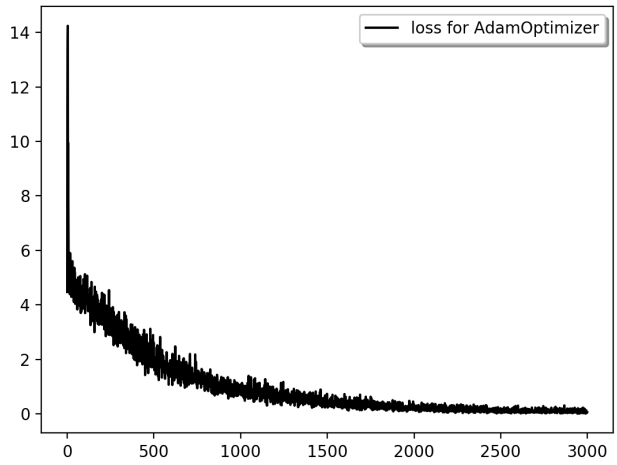

Figure 0.8: learning_method=AdamOptimizer

```
test accuracy 0.503
```

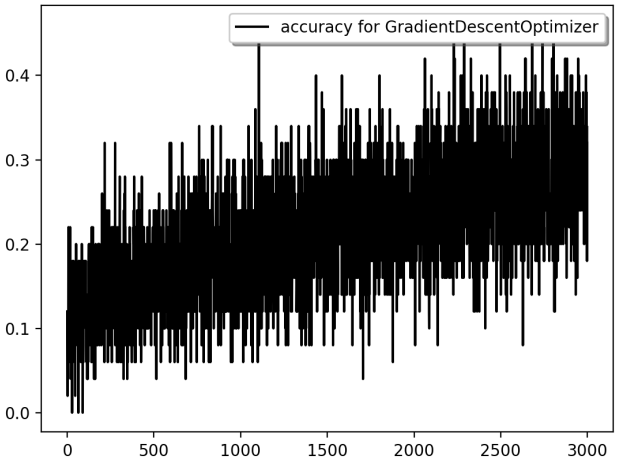

Figure 0.9: learning_method=GradientDescentOptimizer

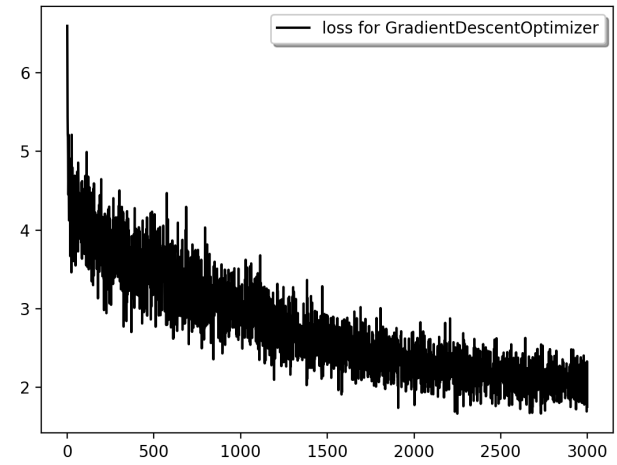

Figure 0.10: learning_method=GradientDescentOptimizer

```
test accuracy 0.33
```

1.c

Visualize the first convolutional layer's weights:

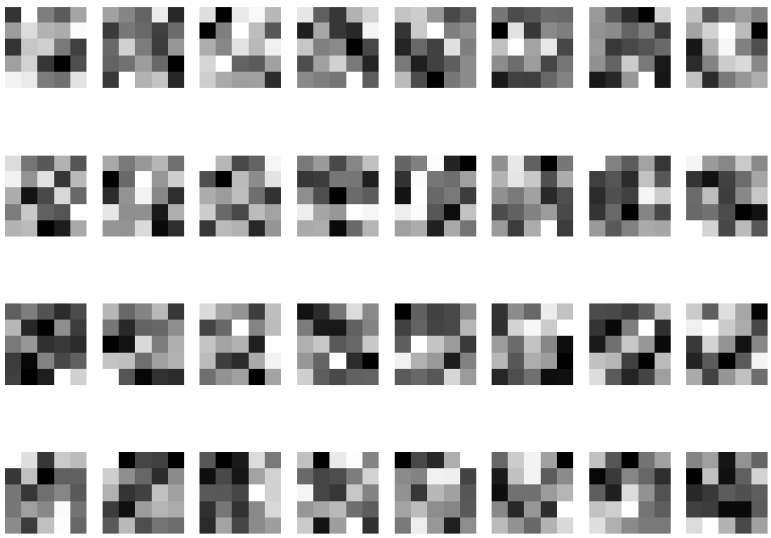

Figure 0.11: first layer weights

Statistics of the activations in the convolutional layers on test images:

```
activation1: mean -0.0139321, variance 0.00760955
activation2: mean -0.166392, variance 0.0450583
```

2 Paper Summarization

This paper is to solve the problems why Large Convolutional Network models perform so well, or how they might be improved. The authors introduce a novel visualization technique that gives insight into the function of intermediate feature layers and the operation of the classifier. The visualization technique they propose uses a multi-layered Deconvolutional Network (deconvnet) to project the feature activations back to the input pixel space.
The authors use standard fully supervised convnet models throughout the paper and these models map a color 2D input image via series layers to a probability vector. The top few layers of the network are conventional fully-connected networks and the final layer is a softmax classifier.
To visualize the features of each layers, the authors present a novel way to map these activities back to the input pixel space, showing what input pattern originally caused a given activation in the feature maps. To examine a convnet, a deconvnet is attached to each of its layers. To reconstruct the activity in each layer, unpool, rectify, and filter are repeated used until input pixel space is reached.

The model was trained on the ImageNet 2012 training set with cropped and resized. Later, they also showed how the ImageNet trained model can generalize well to other datasets.
The authors use the deconvnet to visualize the feature activations on the ImageNet validation set. The paper can help the work in Feature Visualization, Feature Evolution during Training, and Feature Invariance. Also, it can help in Architecture Selection, Occlusion Sensitivity, and Correspondence Analysis.

3.b

As for these experiments, I run it on colab provided by Google. The shared link is:

`https://colab.research.google.com/drive/1wdlC-92FsBbYaiO1O\_WgP729xwHK1VF-`

As for this section, I firstly run the mnist on the following three models:

- BasicLSTMCell
- GRUCell
- BasicRNNCell

The following images are the accuracy and loss curve under three models. After comparing, we find the performance on BasicLSTMCell and GRUCell are similar and they are better than the performance on BasicRNNCell.
After these three sets of experiments, I run the mnist under BasicRNNCell with number of neurons for the RNN as:

- 64 units
- 128 units
- 256 units

After comparing the experiments results, we can easily find when the number of neurons for the RNN is 128, the performance is the best.

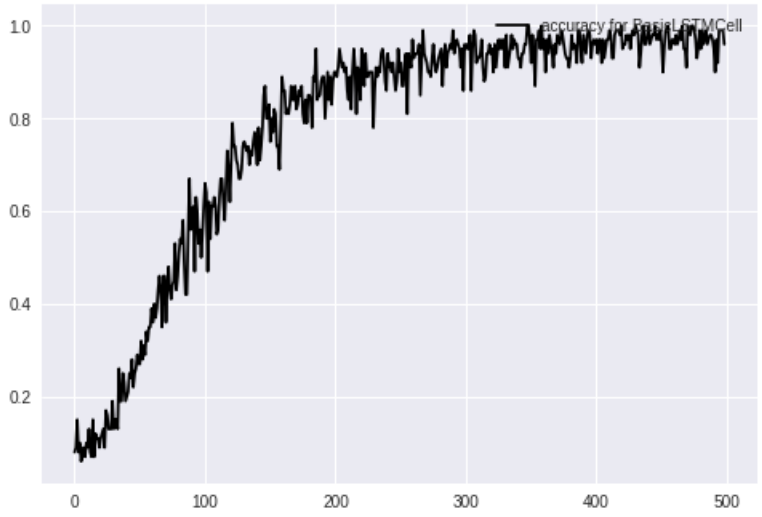

Figure 0.12: BasicLSTMCell accuracy

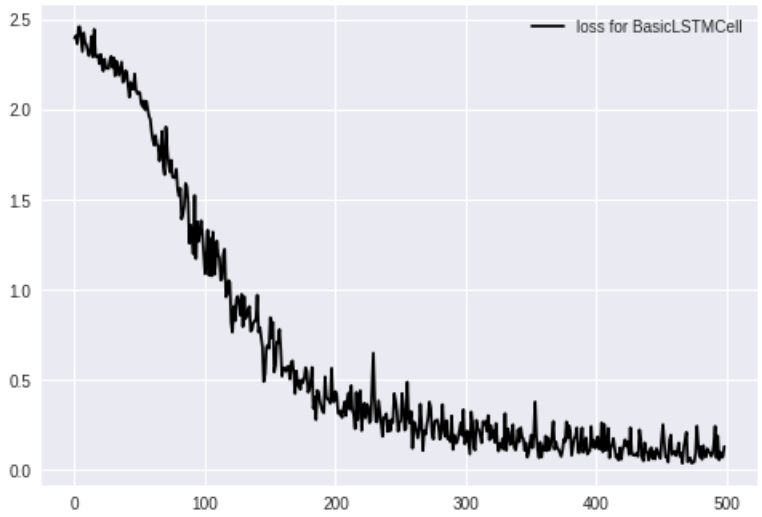

Figure 0.13: BasicLSTMCell loss

```
Testing Accuracy: 0.9423
```

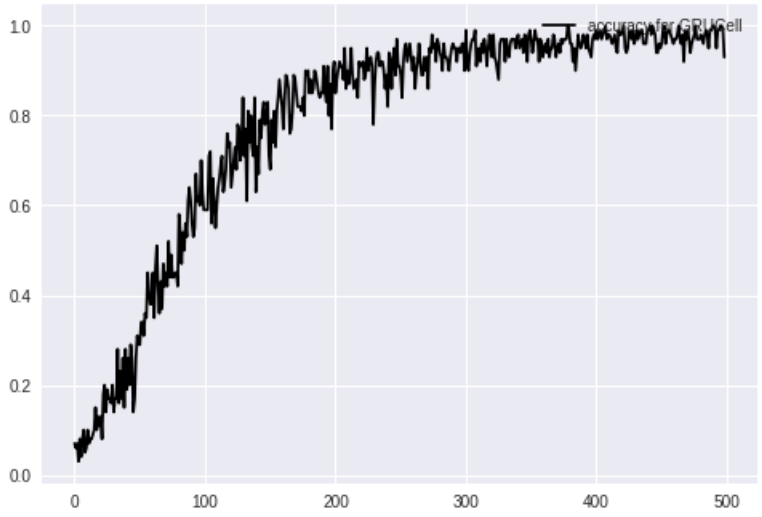

Figure 0.14: GRUCell accuracy

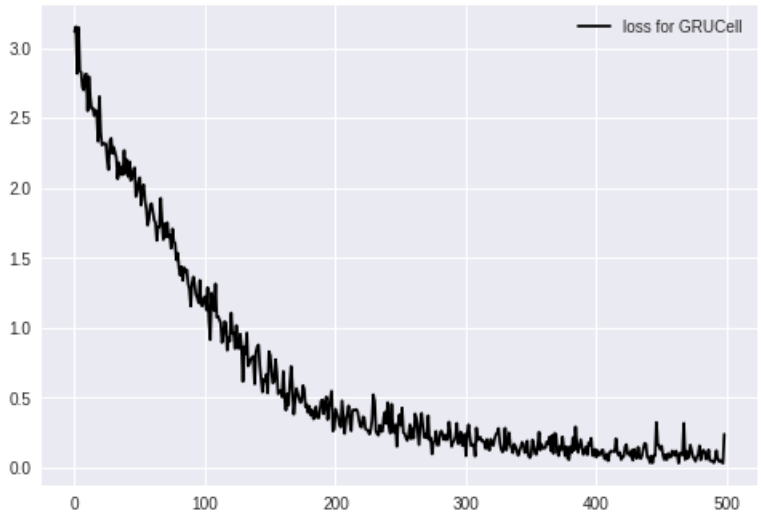

Figure 0.15: GRUCell loss

```
Testing Accuracy: 0.9341
```

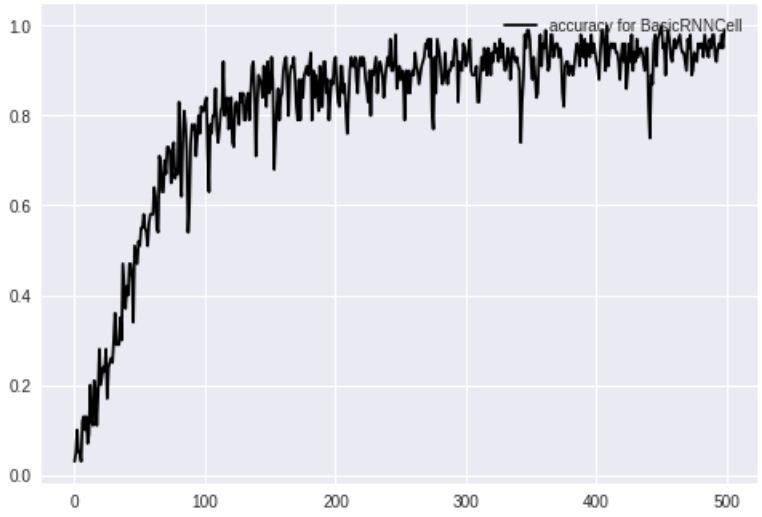

Figure 0.16: BasicRNNCell accuracy

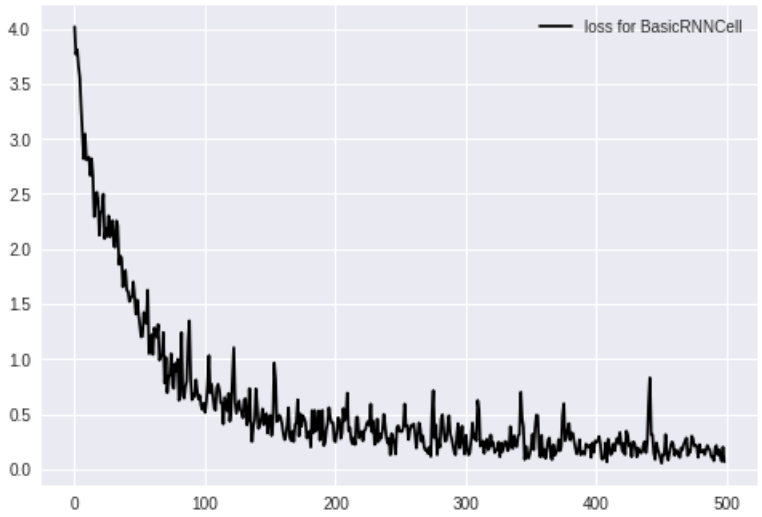

Figure 0.17: BasicRNNCell loss

```
Testing Accuracy: 0.9117
```

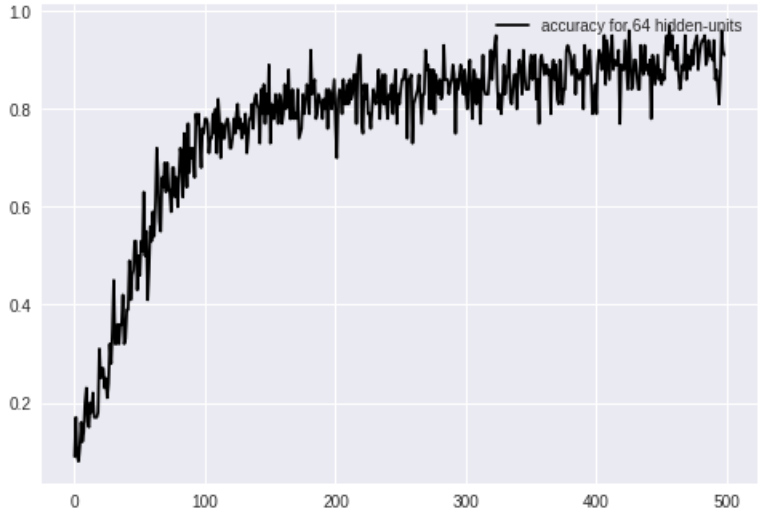

Figure 0.18: unit_64_acc

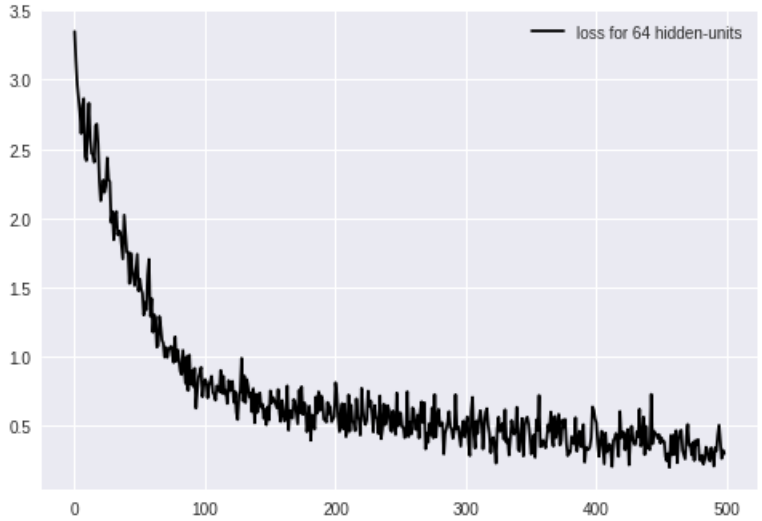

Figure 0.19: unit_64_loss

```
Testing Accuracy: 0.8717
```

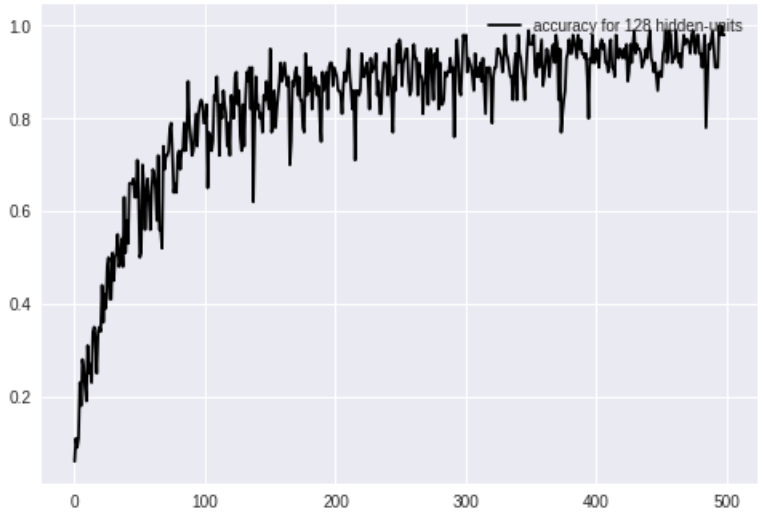

Figure 0.20: unit_128_acc

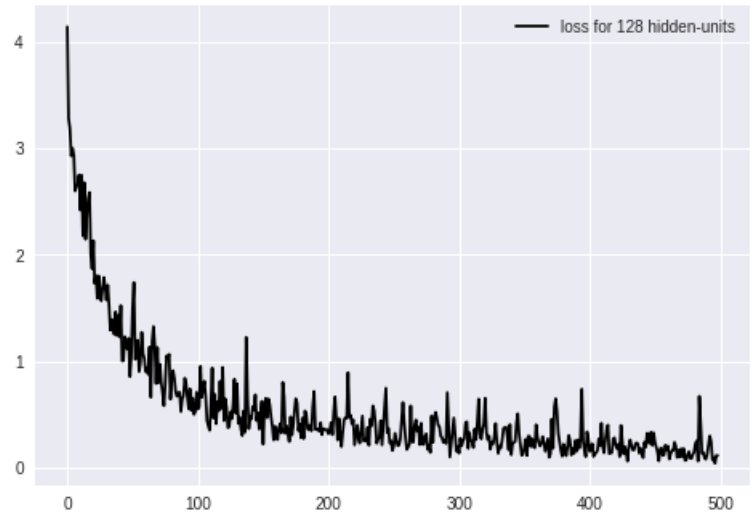

Figure 0.21: unit_128_loss

```
Testing Accuracy: 0.893
```

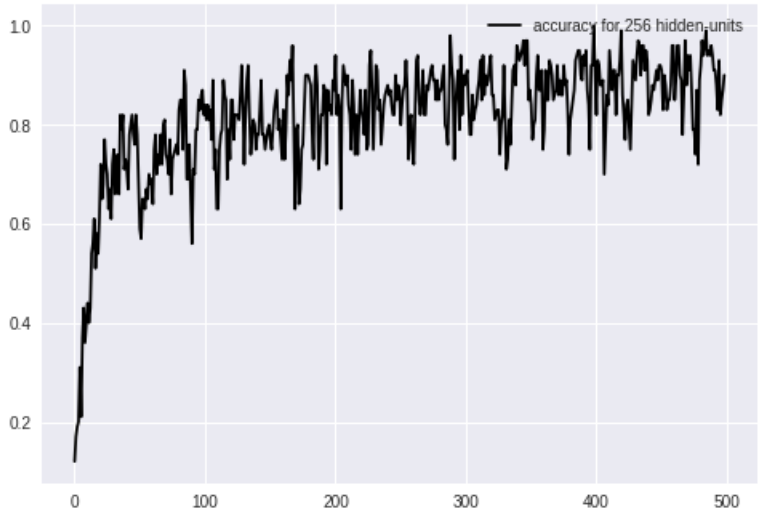

Figure 0.22: unit_256_acc

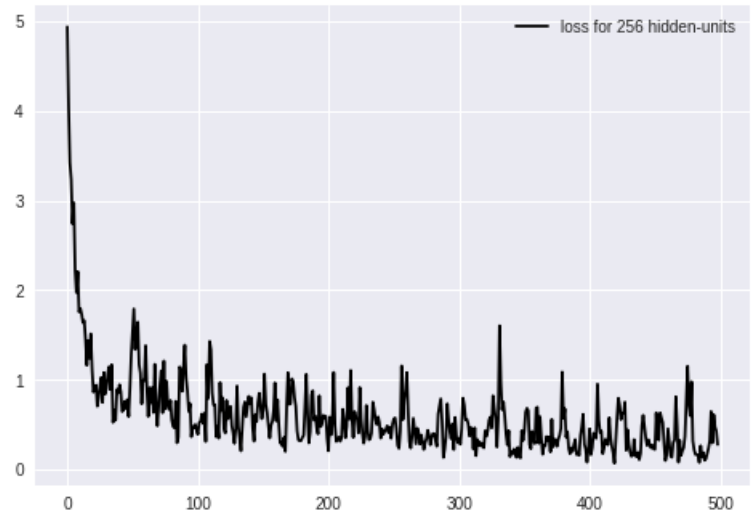

Figure 0.23: unit_256_loss

```
Testing Accuracy: 0.8337
```

3.c

As for similarities, both CNN and RNN are used on deep learning training task and the number of their layers can be very large. Also, both of them can work on image recognition or some other tasks. Last of all, some details in them are similar, like training methods, gradient

methods, etc.
As for difference, the following answer is referred from:

`https://datascience.stackexchange.com/questions/11619/rnn-vs-cnn-at-a-high-level`

CNN:

- CNN take a fixed size input and generate fixed-size outputs.
- CNN is a type of feed-forward artificial neural network - are variations of multilayer perceptrons which are designed to use minimal amounts of preprocessing.
- CNNs use connectivity pattern between its neurons is inspired by the organization of the animal visual cortex, whose individual neurons are arranged in such a way that they respond to overlapping regions tiling the visual field.
- CNNs are ideal for images and videos processing.

RNN:

- RNN can handle arbitrary input/output lengths.
- RNN, unlike feedforward neural networks, can use their internal memory to process arbitrary sequences of inputs.
- Recurrent neural networks use time-series information (i.e. what I spoke last will impact what I will speak next.)
- RNNs are ideal for text and speech analysis.