COMP 576 Assignment 2

ZiZhong Yan (zy46)

1) Visualizing a CNN with CIFAR10

b)
Optimizer: Gradient Descent
LR: 1e-3
Iter: 2000
Result:

```
test accuracy 0.212
```

Optimizer: AdamOptimizer
LR: 1e-3
Iter: 2000
Result:

```
test accuracy 0.491
```

Since Adam shows better accuracy than Gradient Descent, I will be trying to change the learning rate on this optimizer.

Optimizer: AdamOptimizer
LR: 1e-4
Iter: 2000
Result:

```
test accuracy 0.362
```

Optimizer: AdamOptimizer
LR: 1e-6
Iter:2000
Result:

```
test accuracy 0.118
```

So it seems like LR:1e-3 performed the best. I will increase the Iteration on AdamOptimizer with 1e-3 LR.

Optimizer: AdamOptimizer
LR: 1e-3
Iter:6000

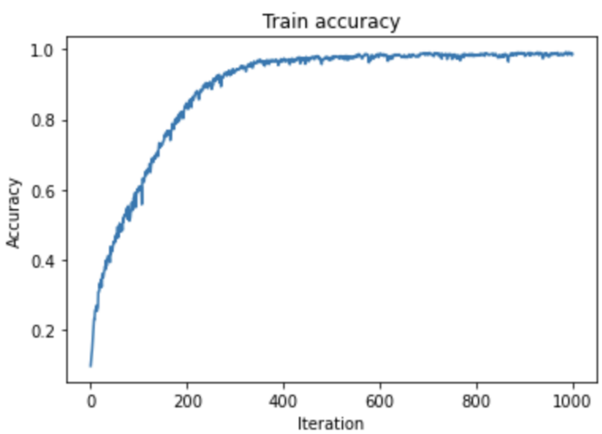
Result:

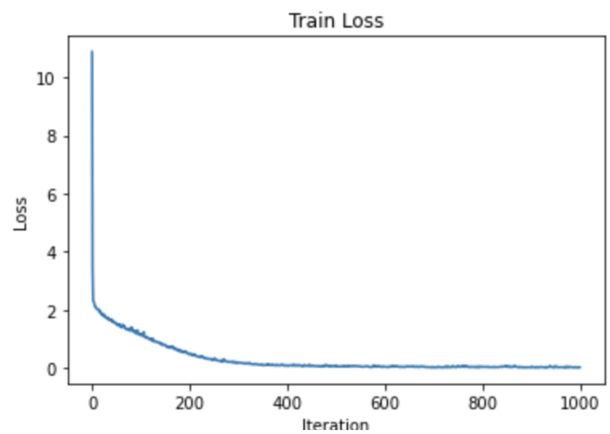
```
test accuracy 0.482
```

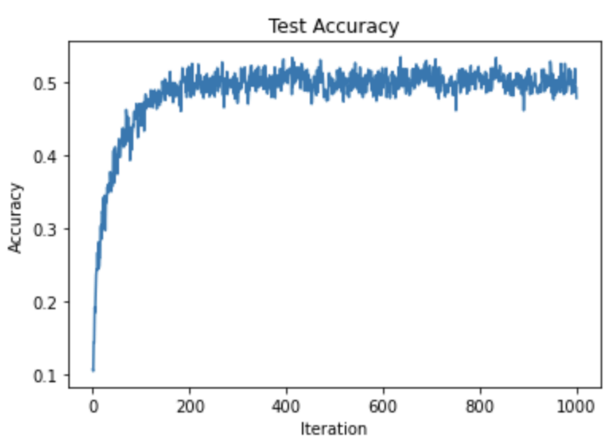
Iter:10000

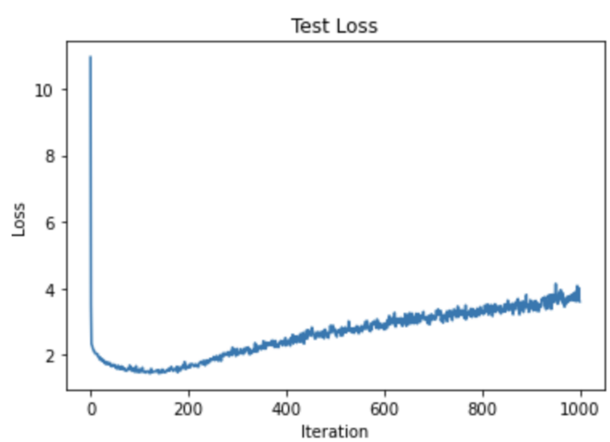
```
test accuracy 0.499
```

Plotting:









c)

Visualize first convolutional layer's weight:

First Layer Stat:

```
Statistics of The Activations on the Convolutional Layer
Max: 0.7514117
Min: 0.0
Mean: 0.0357248
Variance: 0.003353536
Standard Deviation: 0.057909723
```

Second Layer Stat:

```
Statistics of The Activations on the Convolutional Layer
Max: 0.8053479
Min: -0.8971613
Mean: -0.089508004
Variance: 0.02076852
Standard Deviation: 0.14411287
```

2) Paper Summarize

Paper: Visualizing and Understanding Convolutional Networks

By: Matthew D Zeiler, Rob Fergus

Visualization with a Deconvnet:
As mentioned by Matthew and rob “We 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”. A deconvet is a convnet that goes through the same process (filtering, pooling) but in reverse.

- Unsupervised learning
- Probe of already trained convnet
- To examine convnet, deconvnet attach to each layer, provide path back to image pixel

- Filtering
    - “ the deconvnet uses transposed versions of the same filters, but applied to the rectified maps, not the output of the layer beneath”
- Rectification
    - “The convnet uses relu non-linearities, rectifying the feature maps, ensuring the feature maps are always positive. To obtain valid feature reconstructions at each layer, pass the reconstructed signal through a relu non-linearity.”
- Upooling
    - “max pooling operation is non-invertible, can obtain an approximate inverse by recording the locations of the maxima within each pooling region in a set of switch variables”

Convnet Visualization:

- Feature Invariance
    - Image translate, rotate, and scaled by varying degree. Small changes influence the first layer greatly, but little influence on the top layer.
- Feature Evolution during training
    - “Visualizes the progression during training the strongest activation”
- Feature visualization
    - Each layer show hierarchical feature, different layer capture different aspect of images

Architecture Selection:
“While visualization of a trained model gives insight into its operation, it can also assist with selecting good architectures in the first place”

Correspondence Analysis:
Deep models differ from many existing recognition approaches. No explicit mechanism for establishing correspondence between object parts in diff images. Matthew and Rob explored

using Hamming distance, lower value indicates greater consistency in the change from masking operation. Their results show a model established form of correspondence.

Experiment:
Matthew and Rob experimented on ImageNet 2012. They explored the architecture of the model, and tried to replicate Krizhevsky's validation set. Achieved error rate within 0.1%. Present novel way to visualize model activity.

3) MNIST

a) Setup:

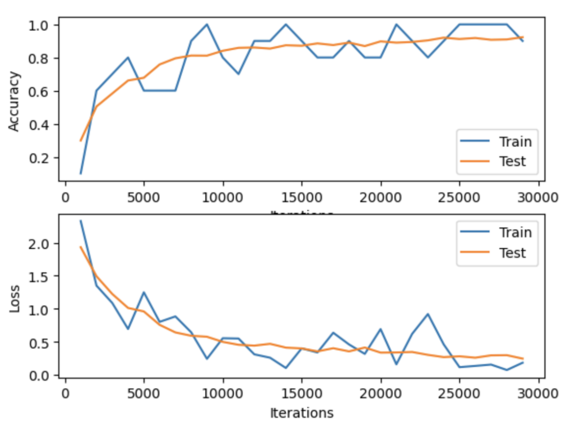
```
mnist = input_data.read_data_sets('MNIST_data', one_hot=True)

learningRate = 1e-3
trainingIters = 30000
batchSize = 10
displayStep = 100

nInput = 28 # we want the input to take the 28 pixels
nSteps = 28  # every 28
nHidden = 64 # number of neurons for the RNN
nClasses = 10 # this is MNIST so you know
```

Optimizer = Adam

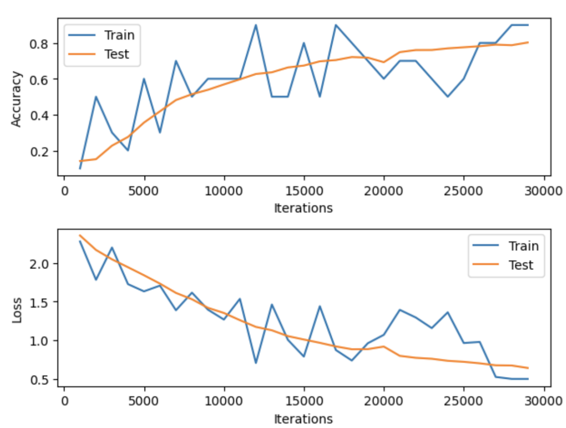
b) Using LSTM and GRU / Plot accuracy and loss of:

LSTM:

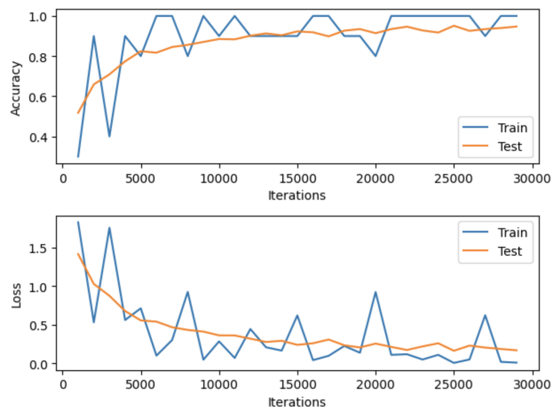
nHidden = 32



nHidden = 10

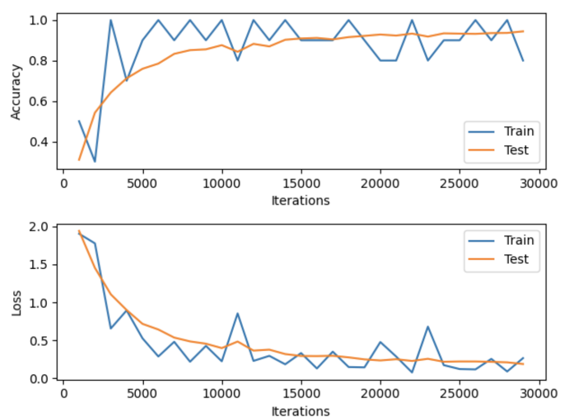


nHidden = 64

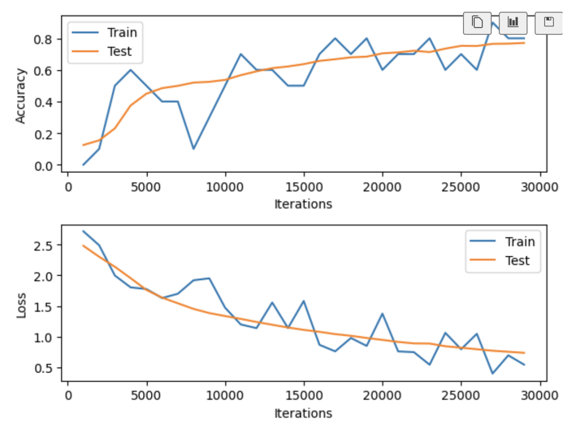


GRU:

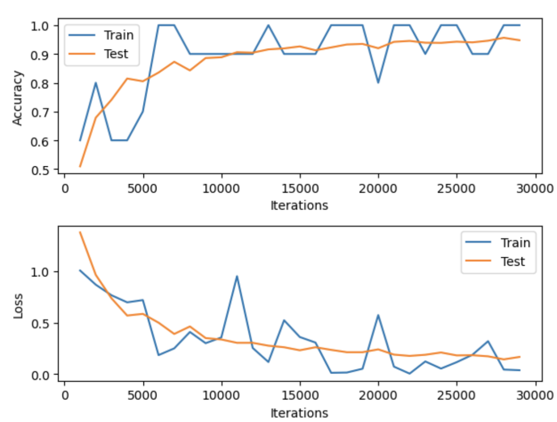
nHidden = 32



nHidden = 10



nHidden = 64



b) I noticed that GRU and LSTM perform similarly in both accuracy and loss. But LSTM has a higher accuracy at the beginning. According to the plot it looks like at nHidden layer = 32, the loss of LSTM is slightly higher.

c) The first thing I notice is, CNN has the same size input and same size output. CNN are feed forward using filter and pooling
CNN processes in a more hierarchical way, with steps. SoCNN are better in image processing
RNN can have different input and output. RNN is recurring, it recurse back to the network