HW4 – W251 – Submitted by abhisha@berkeley.edu

1. Name all the layers in the network, describe what they do.

There are 2 layers of convolution and pooling. Convolution layer helps in extracting the relevant features of the image based on a “filter” - things like edges and lines. Pooling helps in size reduction of the output which goes into the next layer’s input. The idea of max pooling is to extract the max value from the convolved matrix – this will indicate a strong signal if the features in the image overlapped well with the filters and will indicate a weak signal if the features in the image didnt overlap well with the filter. The last layer Is a softmax layer which helps in assigning probabilities to each of the 10 digit classes. The class with the highest probability is chosen.

Baseline model from https://cs.stanford.edu/people/karpathy/convnetjs/demo/mnist.html

```
layer_defs = [];

layer_defs.push({type:'input', out_sx:24, out_sy:24, out_depth:1});

layer_defs.push({type:'conv', sx:5, filters:8, stride:1, pad:2, activation:'relu'});

layer_defs.push({type:'pool', sx:2, stride:2});

layer_defs.push({type:'conv', sx:5, filters:16, stride:1, pad:2, activation:'relu'});

layer_defs.push({type:'pool', sx:3, stride:3});

layer_defs.push({type:'softmax', num_classes:10});


net = new convnetjs.Net();

net.makeLayers(layer_defs);


trainer = new convnetjs.SGDTrainer(net, {method:'adadelta', batch_size:20,
l2_decay:0.001});
```

Training Stats

pause

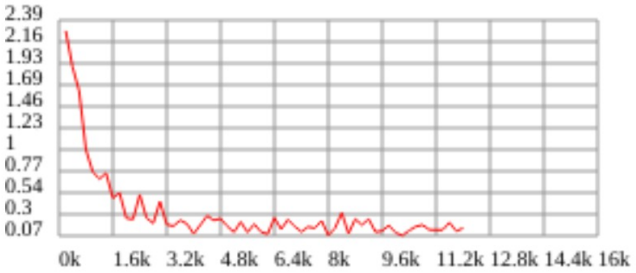
Forward time per example: 5ms
Backprop time per example: 6ms
Classification loss: 0.11704
L2 Weight decay loss: 0.00219
Training accuracy: 0.94
Validation accuracy: 0.9
Examples seen: 12121

Learning rate: 0.01 change
Momentum: 0.9 change
Batch size: 20 change
Weight decay: 0.001 change

save network snapshot as JSON
init network from JSON snapshot

original

Loss:



clear graph

2. Experiment with the number and size of filters in each layer. Does it improve the accuracy?

Training Stats

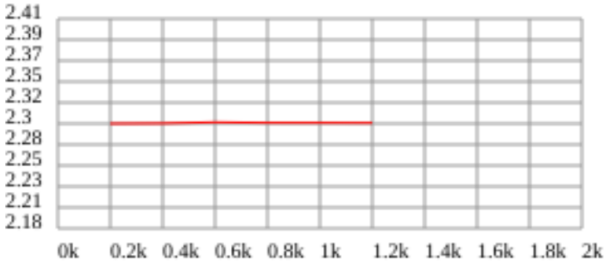
pause

Forward time per example: 7ms
Backprop time per example: 11ms
Classification loss: 2.29946
L2 Weight decay loss: 0.00142
Training accuracy: 0.06
Validation accuracy: 0.13
Examples seen: 1338

Learning rate: 0.01 change
Momentum: 0.9 change
Batch size: 20 change
Weight decay: 0.001 change

save network snapshot as JSON
init network from JSON snapshot

Loss:



clear graph

Instantiate a Network and Trainer

```
layer_defs = [];
layer_defs.push({type:'input', out_sx:24, out_sy:24, out_depth:1});
layer_defs.push({type:'conv', sx:10, filters:16, stride:1, pad:2, activation:'relu'});
layer_defs.push({type:'pool', sx:2, stride:2});
layer_defs.push({type:'conv', sx:15, filters:32, stride:1, pad:2, activation:'relu'});
layer_defs.push({type:'pool', sx:3, stride:3});
layer_defs.push({type:'softmax', num_classes:10});

net = new convnetjs.Net();
net.makeLayers(layer_defs);

trainer = new convnetjs.SGDTrainer(net, {method:'adadelta', batch_size:20, l2_decay:0.001});
```

In this case, we increased the size of filters and number of filters in both layers. We see the model has performed very badly. There is likely some error in the model because the error didnt even go down. One possibility is lack of training examples – we only saw 1338 examples.

Another try gives us the following

Training Stats

pause

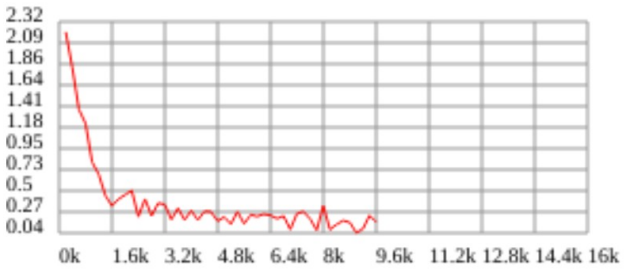
Forward time per example: 11ms
Backprop time per example: 16ms
Classification loss: 0.05581
L2 Weight decay loss: 0.00314
Training accuracy: 1
Validation accuracy: 0.94
Examples seen: 9701

Learning rate: 0.01 change
Momentum: 0.9 change
Batch size: 20 change
Weight decay: 0.001 change

save network snapshot as JSON
init network from JSON snapshot

increased filters
and filter size=4

Loss:



clear graph

Instantiate a Network and Trainer

```
layer_defs = [];
layer_defs.push({type:'input', out_sx:24, out_sy:24, out_depth:1});
layer_defs.push({type:'conv', sx:4, filters:16, stride:1, pad:2, activation:'relu'});
layer_defs.push({type:'pool', sx:2, stride:2});
layer_defs.push({type:'conv', sx:4, filters:32, stride:1, pad:2, activation:'relu'});
layer_defs.push({type:'pool', sx:3, stride:3});
layer_defs.push({type:'softmax', num_classes:10});

net = new convnetjs.Net();
net.makeLayers(layer_defs);

trainer = new convnetjs.SGDTrainer(net, {method:'adadelta', batch_size:20, l2_decay:0.001});
```

change network

This is more reasonable, for the above, we increased filters and we changed the filter size to 4 in both layers.

3. Remove the pooling layers. Does it impact the accuracy?

Training Stats

pause

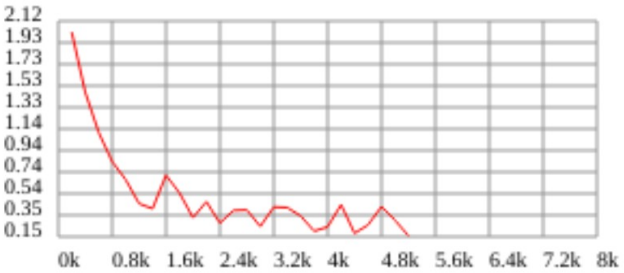
Forward time per example: 11ms
Backprop time per example: 20ms
Classification loss: 0.18601
L2 Weight decay loss: 0.0025
Training accuracy: 0.94
Validation accuracy: 0.83
Examples seen: 5331

Learning rate: 0.01 change
Momentum: 0.9 change
Batch size: 20 change
Weight decay: 0.001 change

save network snapshot as JSON
init network from JSON snapshot

```
removed pooling
```

Loss:



clear graph

Instantiate a Network and Trainer

```
layer_defs = [];
layer_defs.push({type:'input', out_sx:24, out_sy:24, out_depth:1});
layer_defs.push({type:'conv', sx:5, filters:8, stride:1, pad:2, activation:'relu'});
layer_defs.push({type:'conv', sx:5, filters:16, stride:1, pad:2, activation:'relu'});
layer_defs.push({type:'softmax', num_classes:10});

net = new convnetjs.Net();
net.makeLayers(layer_defs);

trainer = new convnetjs.SGDTrainer(net, {method:'adadelta', batch_size:20, l2_decay:0.001});
```

change network

Accuracy decreases when we remove pooling, which is interesting

4. Add one more conv layer. Does it help with accuracy?

Training Stats

pause

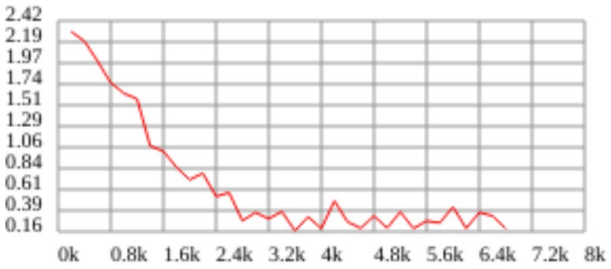
Forward time per example: 3ms
Backprop time per example: 8ms
Classification loss: 0.17683
L2 Weight decay loss: 0.00273
Training accuracy: 0.95
Validation accuracy: 0.83
Examples seen: 6891

Learning rate: 0.01 change
Momentum: 0.9 change
Batch size: 20 change
Weight decay: 0.001 change

save network snapshot as JSON
init network from JSON snapshot

added layer of conv

Loss:



clear graph

Instantiate a Network and Trainer

```
layer_defs = [];
layer_defs.push({type:'input', out_sx:24, out_sy:24, out_depth:1});
layer_defs.push({type:'conv', sx:5, filters:8, stride:1, pad:2, activation:'relu'});
layer_defs.push({type:'pool', sx:2, stride:2});
layer_defs.push({type:'conv', sx:5, filters:16, stride:1, pad:2, activation:'relu'});
layer_defs.push({type:'pool', sx:3, stride:3});
layer_defs.push({type:'conv', sx:5, filters:16, stride:1, pad:2, activation:'relu'});
layer_defs.push({type:'pool', sx:3, stride:3});
layer_defs.push({type:'softmax', num_classes:10});

net = new convnetjs.Net();
net.makeLayers(layer_defs);
```

change network

Network Visualization

Adding a new layer probably leads to some form of overfitting where we see high training accuracy but low test accuracy

5. Increase the batch size. What impact does it have?

Training Stats

pause

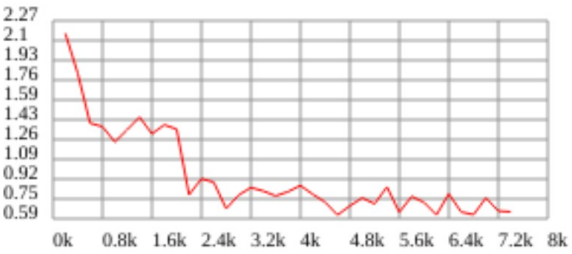
Forward time per example: 4ms
Backprop time per example: 9ms
Classification loss: 0.56118
L2 Weight decay loss: 0
Training accuracy: 0.89
Validation accuracy: 0.73
Examples seen: 7542

Learning rate: 0.01 change
Momentum: 0.9 change
Batch size: 2000 change
Weight decay: 0.001 change

save network snapshot as JSON
init network from JSON snapshot

batch size inc

Loss:



clear graph

Instantiate a Network and Trainer

```
layer_defs = [];
layer_defs.push({type:'input', out_sx:24, out_sy:24, out_depth:1});
layer_defs.push({type:'conv', sx:5, filters:8, stride:1, pad:2, activation:'relu'});
layer_defs.push({type:'pool', sx:2, stride:2});
layer_defs.push({type:'conv', sx:5, filters:16, stride:1, pad:2, activation:'relu'});
layer_defs.push({type:'pool', sx:3, stride:3});
layer_defs.push({type:'softmax', num_classes:10});

net = new convnetjs.Net();
net.makeLayers(layer_defs);

trainer = new convnetjs.SGDTrainer(net, {method:'adadelta', batch_size:20, l2_decay:0.001});
```

change network

Increasing batch size from 20 to 2000 leads to a very low validation accuracy relative to the other models.

6. What is the best accuracy you can achieve? Are you over 99%? 99.5%?

Training Stats

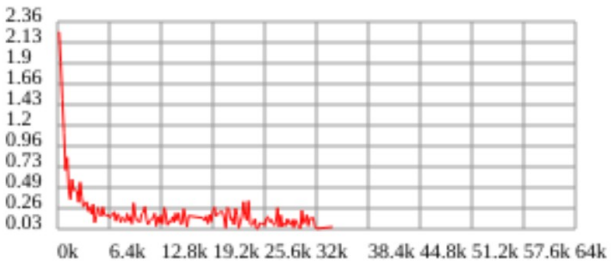
resume

Forward time per example: 4ms
Backprop time per example: 7ms
Classification loss: 0.04238
L2 Weight decay loss: -0.00001
Training accuracy: 0.99
Validation accuracy: 0.96
Examples seen: 35650

Learning rate: 0.1 change
Momentum: 0.9 change
Batch size: 2000 change
Weight decay: 0.01 change

save network snapshot as JSON
init network from JSON snapshot

Loss:



clear graph

No unfortunately my network only had a validation accuracy of 96% with the default network