ConvNetJS CIFAR-10 demo

Description

This demo trains a Convolutional Neural Network on the CIFAR-10 dataset in your browser, with nothing but Javascript. The state of the art on this dataset is about 90% accuracy and human performance is at about 94% (not perfect as the dataset can be a bit ambiguous). I used this python script to parse the original files (python version) into batches of images that can be easily loaded into page DOM with img tags.

This dataset is more difficult and it takes longer to train a network. Data augmentation includes random flipping and random image shifts by up to 2px horizontally and verically.

By default, in this demo we're using Adadelta which is one of per-parameter adaptive step size methods, so we don't have to worry about changing learning rates or momentum over time. However, I still included the text fields for changing these if you'd like to play around with SGD+Momentum trainer.

Report questions/bugs/suggestions to @karpathy.

Training Stats

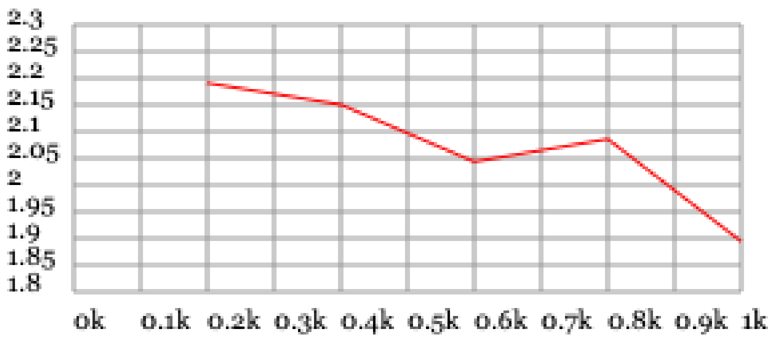
pause

Forward time per example: 8ms
Backprop time per example: 13ms
Classification loss: 1.92149
L2 Weight decay loss: 0.00102
Training accuracy: 0.34
Validation accuracy: 0.1
Examples seen: 1002
Learning rate: 0.01 change
Momentum: 0.9 change
Batch size: 4 change
Weight decay: 0.0001 change

save network snapshot as JSON
init network from JSON snapshot

load a pretrained network (achieves ~80% accuracy)

Loss:



clear graph

Instantiate a Network and Trainer

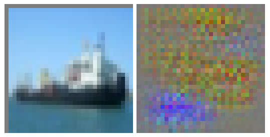
```
layer_defs = [];
layer_defs.push({type:'input', out_sx:32, out_sy:32, out_depth:3});
layer_defs.push({type:'conv', sx:5, filters:16, stride:1, pad:2, activation:'relu'});
layer_defs.push({type:'pool', sx:2, stride:2});
layer_defs.push({type:'conv', sx:5, filters:20, stride:1, pad:2, activation:'relu'});
layer_defs.push({type:'pool', sx:2, stride:2});
layer_defs.push({type:'conv', sx:5, filters:20, stride:1, pad:2, activation:'relu'});
layer_defs.push({type:'pool', sx:2, stride:2});
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:4, l2_decay:0.0001});
```

change network

Network Visualization

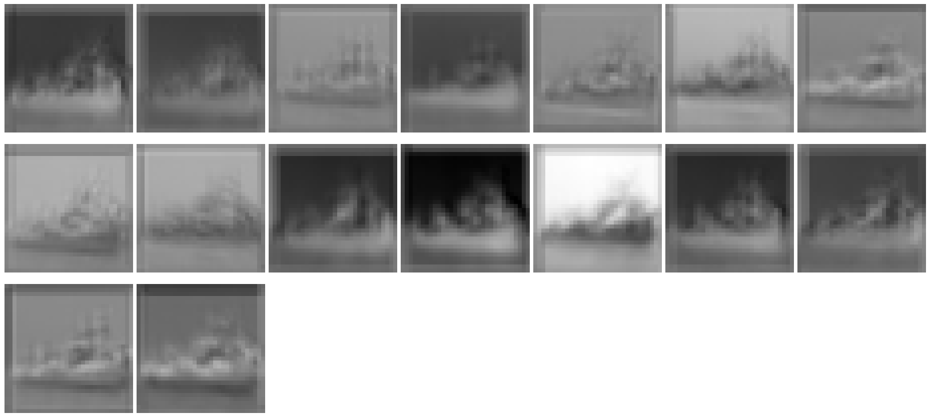
input (32x32x3)
max activation: 0.48823, min: -0.47648
max gradient: 0.00996, min: -0.00899

Activations:

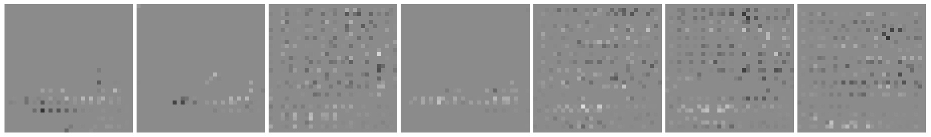


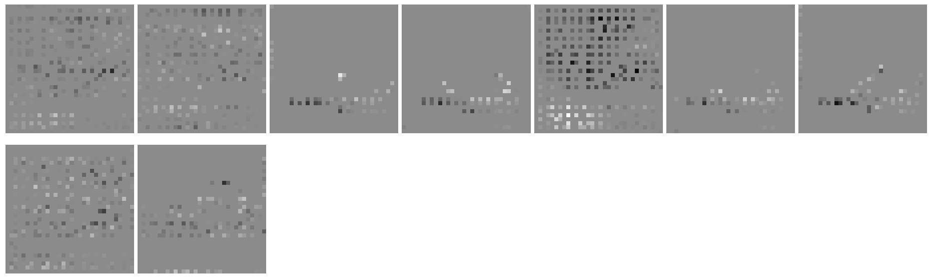
conv (32x32x16)
filter size 5x5x3, stride 1
max activation: 1.30636, min: -1.0832
max gradient: 0.00579, min: -0.00704
parameters: 16x5x5x3+16 = 1216

Activations:

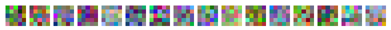


Activation Gradients:





Weights:



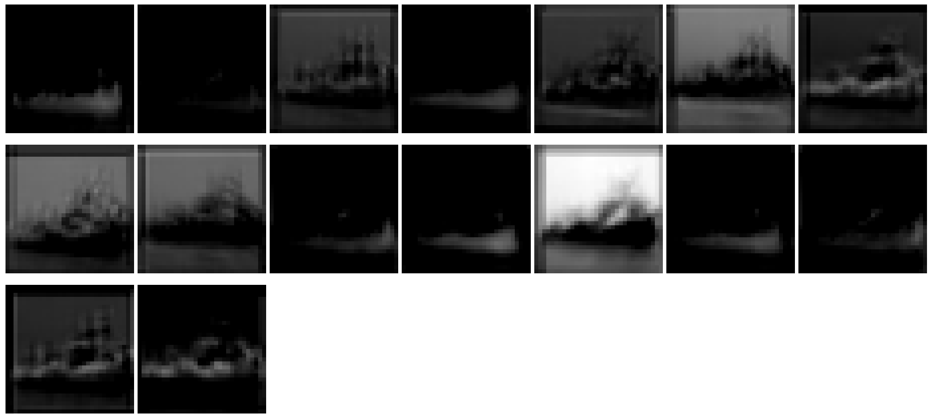
Weight Gradients:

relu (32x32x16)

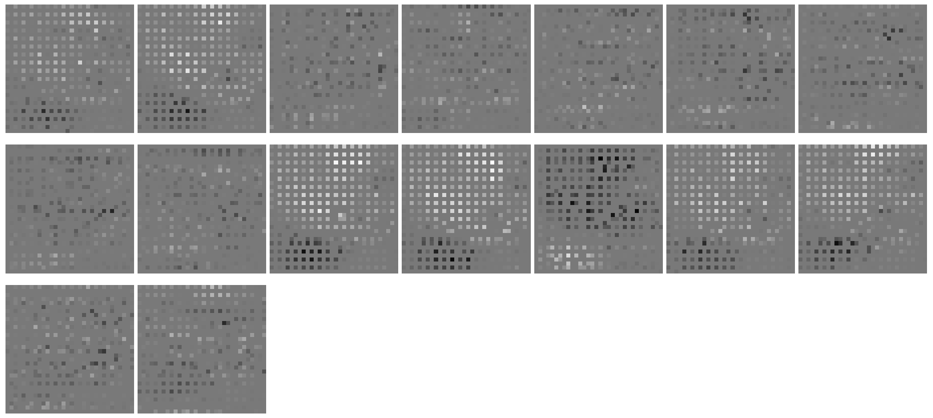
max activation: 1.30636, min: 0

max gradient: 0.00777, min: -0.00704

Activations:

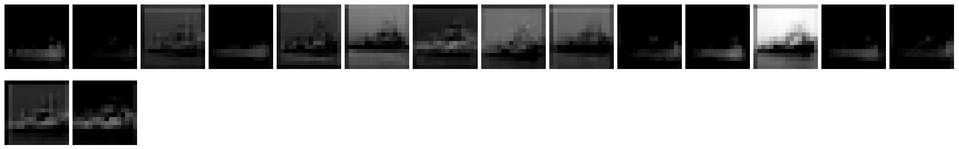


Activation Gradients:

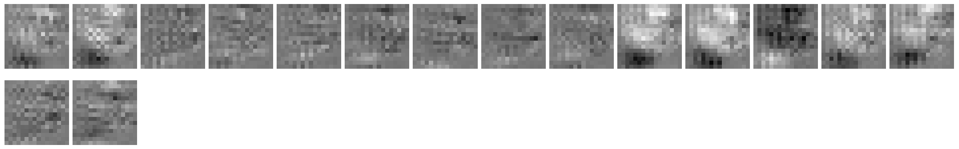


pool (16x16x16)
pooling size 2x2, stride 2
max activation: 1.30636, min: 0
max gradient: 0.00777, min: -0.00704

Activations:

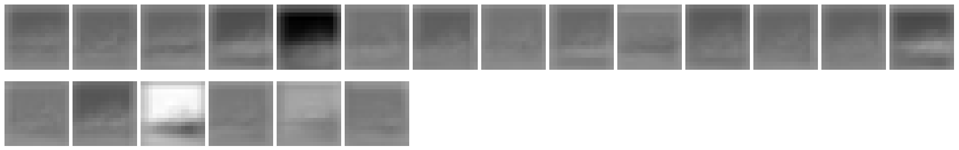


Activation Gradients:

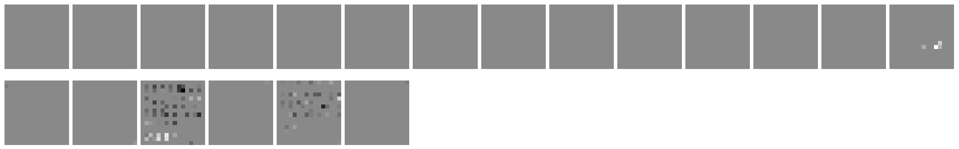


conv (16x16x20)
filter size 5x5x16, stride 1
max activation: 3.23825, min: -4.47611
max gradient: 0.01436, min: -0.01682
parameters: 20x5x5x16+20 = 8020

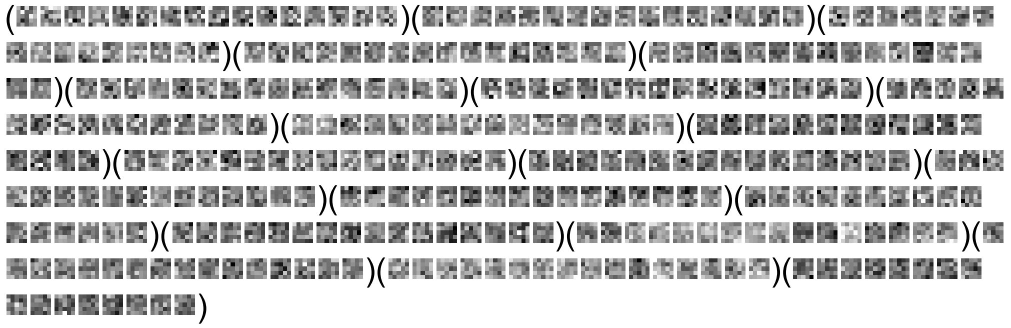
Activations:



Activation Gradients:



Weights:

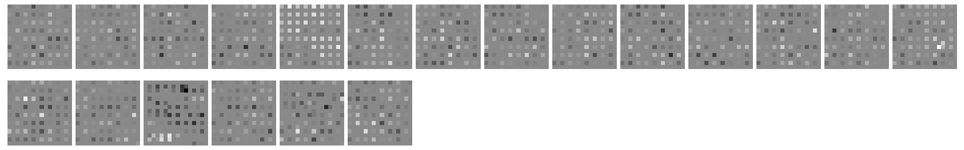


Weight Gradients:

relu (16x16x20)
max activation: 3.23825, min: 0
max gradient: 0.01436, min: -0.01682

Activations:

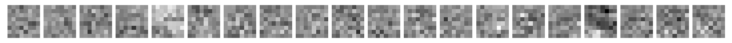
Activation Gradients:



pool (8x8x20)
pooling size 2x2, stride 2
max activation: 3.23825, min: 0
max gradient: 0.01436, min: -0.01682

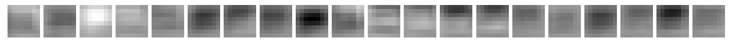
Activations:

Activation Gradients:



conv (8x8x20)
filter size 5x5x20, stride 1
max activation: 3.34058, min: -4.87992
max gradient: 0.0347, min: -0.03621
parameters: 20x5x5x20+20 = 10020

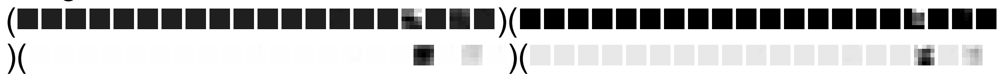
Activations:

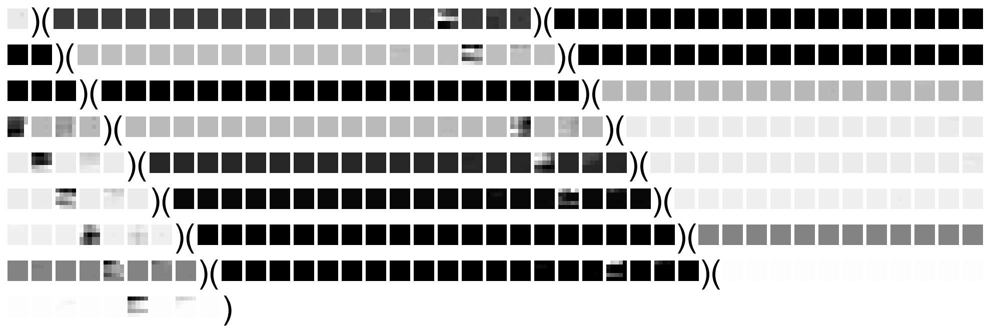


Activation Gradients:

Weights:

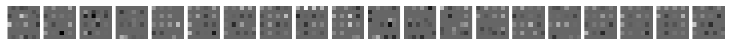
Weight Gradients:





relu (8x8x20)
max activation: 3.34058, min: 0
max gradient: 0.05703, min: -0.03851

Activations:

Activation Gradients:


pool (4x4x20)
pooling size 2x2, stride 2
max activation: 3.34058, min: 0
max gradient: 0.05703, min: -0.03851

Activations:

Activation Gradients:

fc (1x1x10)
max activation: 1.69487, min: -8.53016
max gradient: 0.10433, min: -0.19265
parameters: 10x320+10 = 3210

Activations:

Activation Gradients:

softmax (1x1x10)
max activation: 0.80735, min: 0.00002
max gradient: 0, min: 0

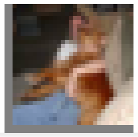
Activations:

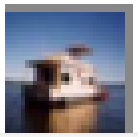
Example predictions on Test set

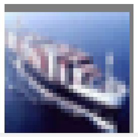
test accuracy based on last 200 test images: 0.275

truck
ship
horse

frog
cat
deer

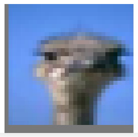

frog
cat
deer


ship
truck
airplane


ship
truck
car

ship
car
truck

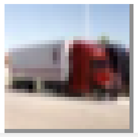
truck
car
ship

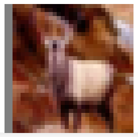

ship
airplane
truck

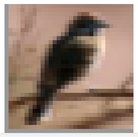
horse
deer
bird

frog
deer
dog

frog
deer
dog


truck
horse
ship

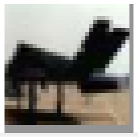

frog
cat
deer


frog
cat
horse

airplane
ship
horse

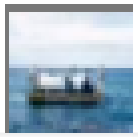
ship
airplane
truck

truck
car
airplane

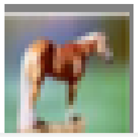

truck
horse
car

deer
frog
dog

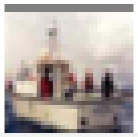
frog
deer
dog


ship
airplane
car

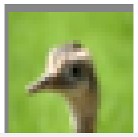
ship
airplane
car


frog
bird
deer

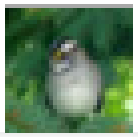
frog
deer
bird


horse
truck
car

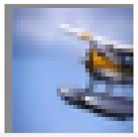
frog
horse
truck


frog
horse
truck

horse
truck
car


ship
airplane
truck

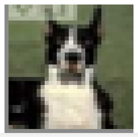
ship
airplane
car


ship
airplane
car

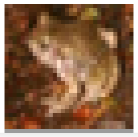
ship
airplane
truck

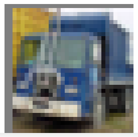
deer
frog
cat

airplane
car
ship

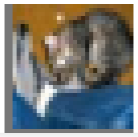

deer
frog
cat

deer
truck
frog


cat
deer
dog


airplane
car
ship

ship
airplane
car


car
airplane
cat