CSE 190 - Neural Network - Homework 3

Monica Gonsalves, Ning Ma, Xiaoyu(Justin) Liu

November 2015

1 AWS

As instructed, this is the instance we are using for this assignment
InstanceID: i-86147742
DNS: ec2-54-218-33-90.us-west-2.compute.amazonaws.com
IP: 54.218.33.90

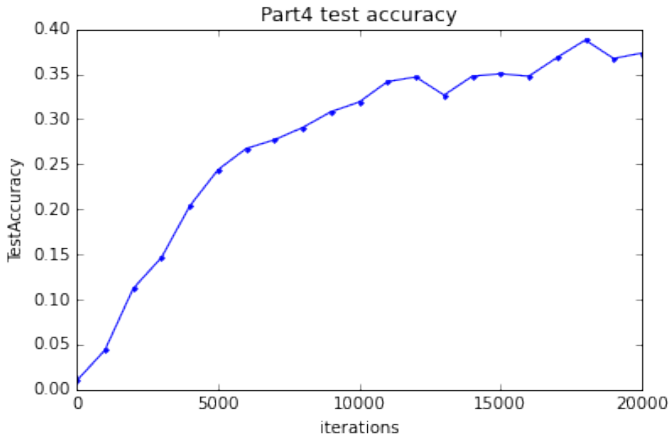
2 Load Data

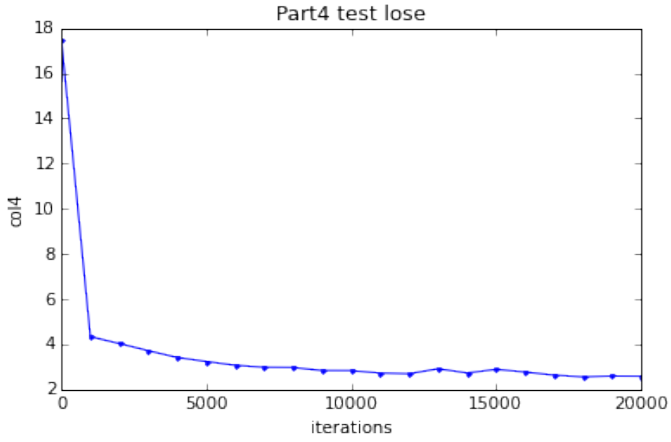
We split train and test into training.txt and testing.txt, where each txt file consists of lines of ("/path/to/png":String label:Int) pairs. In addition, we also transform into lmdb format.

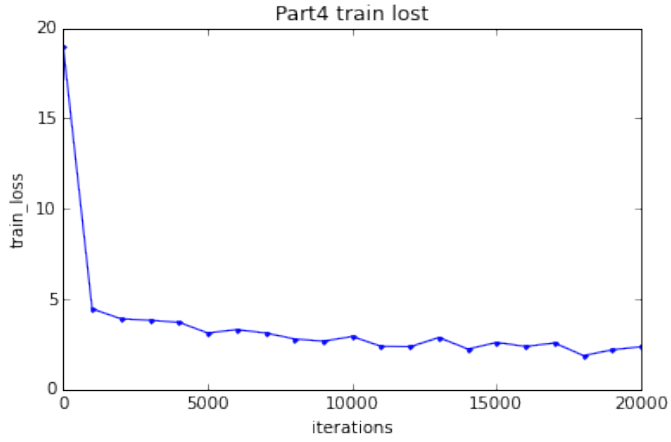
3 Build Network

Layer	Type	Input	Kernel	Filter	Nonlinearity	Pooling	Stride	Size	Output	Parameters
1	Conv	32*32*3	2*2	32	relu	MAX	2	3	15*15*32	416
2	Conv	15*15*32	5*5	64	relu	AVE	2	3	6*6*64	51264
3	Conv	6*6*64	2*2	128	relu	AVE	2	3	2*2*128	32896
4	FC	2*2*128	1*1						128	65664
5	FC	128	1		softmax				100	12900

4 Train Network



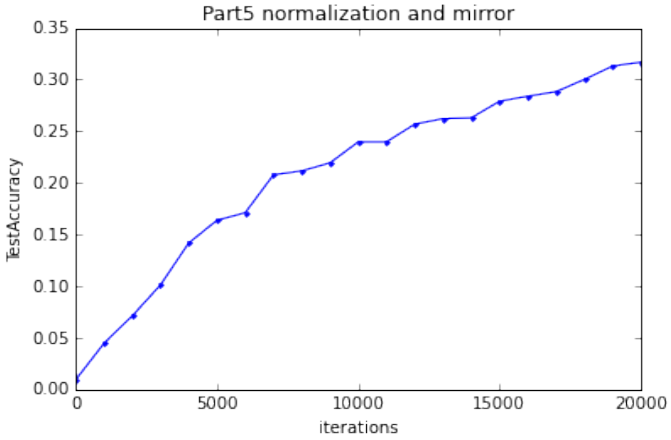




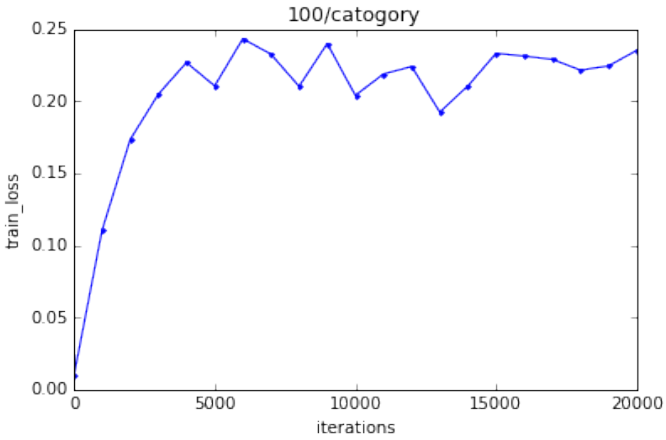
First one is test accuracy, second one is test loss, and the last one is train loss, all against iterations

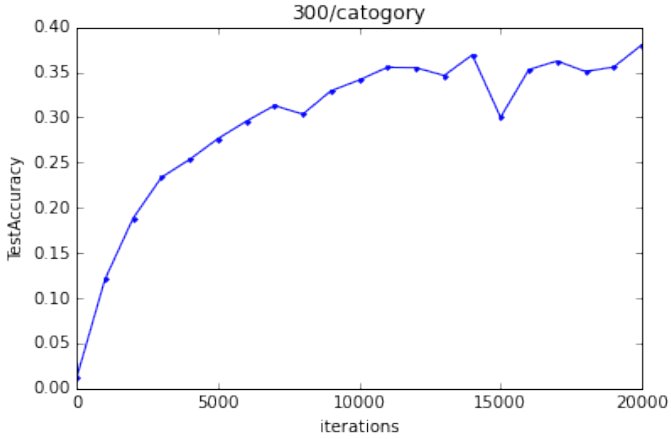
5 PrePocessing Input Data

(a) After making the normalizaton, subtracting mean and randomly applying mirror, we can see the accuracy converges slower than the original data. However, according to the trend, the accuracy of network using normalized data will be higher than the one using original data, if we increase the iteration number.



(b) We changed the number of training images per category to 100 and 300 respectively, and get the following results. The test accuracy decreases as we decrease the number of images in each category to 100. When we increase the number of images in each category to 300, we get a test accuracy which is better than the 100 per category data.





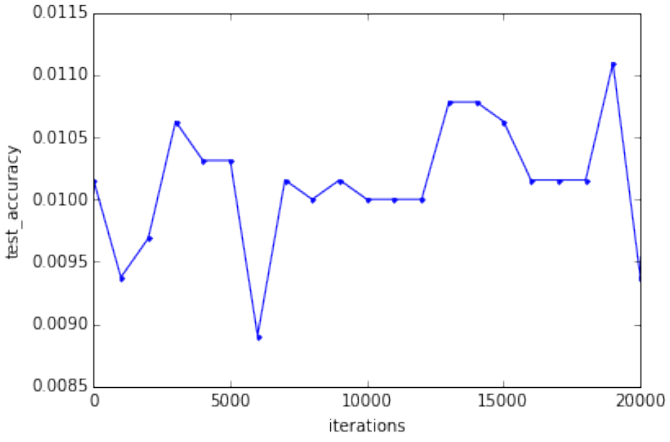
6 Network Structure

Number of parameters in part 3: 163140

Only one conv net, total number of parameters: 174436

Layer	Type	Input	Kernel	Filter	Nonlinearity	Pooling	Stride	Size	Output	Parameters
1	Conv	32*32*3	6*6	128	relu	MAX	2	10	3*3*128	13952
2	FC	3*3*128	1*1						128	147584
3	FC	128	1		softmax				100	12900

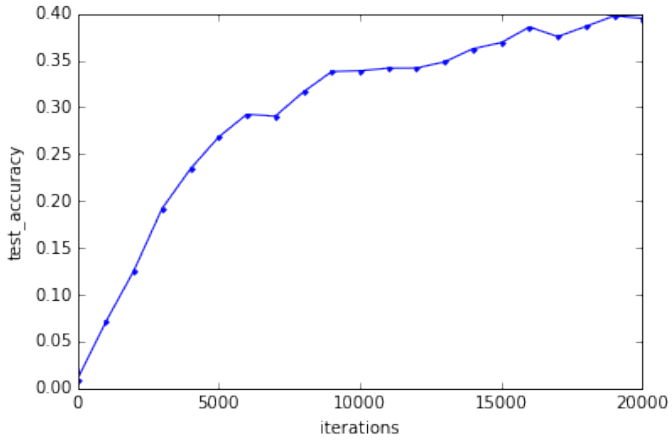
Here is the test accuracy vs iteration plot



One more conv net, total number of parameters: 167300

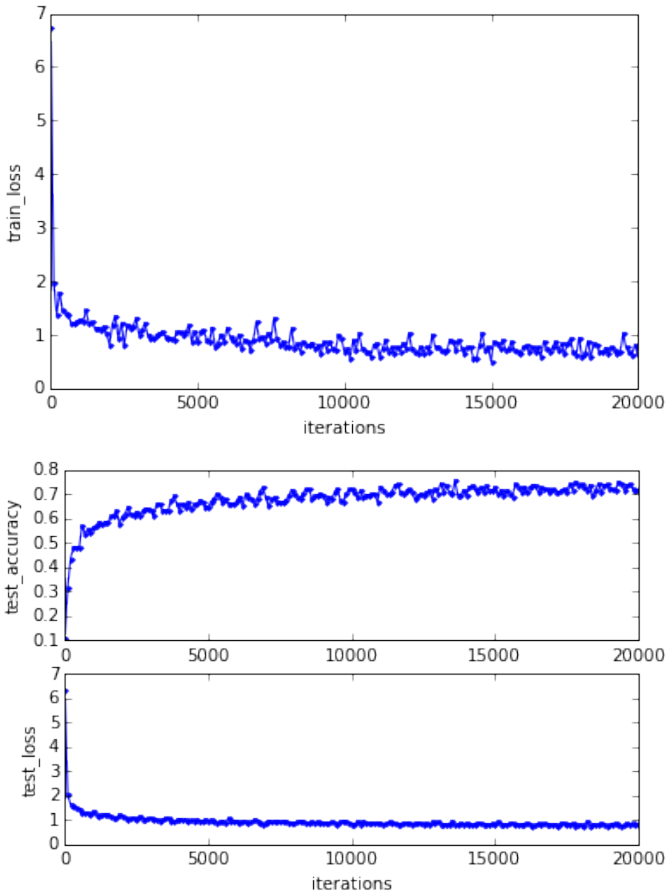
Layer	Type	Input	Kernel	Filter	Nonlinearity	Pooling	Stride	Size	Output	Parameters
1	Conv	32*32*3	2*2	32	relu	MAX	2	3	15*15*32	416
2	Conv	15*15*32	3*3	64	relu	AVE	1	3	13*13*64	18496
3	Conv	13*13*64	3*3	64	relu	AVE	2	3	5*5*64	36928
4	Conv	5*5*64	2*2	128	relu	AVE	2	2	2*2*128	32896
5	FC	2*2*128	1*1						128	65664
6	FC	128	1		softmax				100	12900

Here is the test accuracy vs iteration plot



Clearly, compare to the original model in part 3, set the number of conv layer to 1 gives terrible test results. On the other hand, adding one more conv layer to the original model gives slightly better accuracy. We are able to achieve to 40% toward the end of 20000th iterations. So we can conclude that depth is important to some extent.

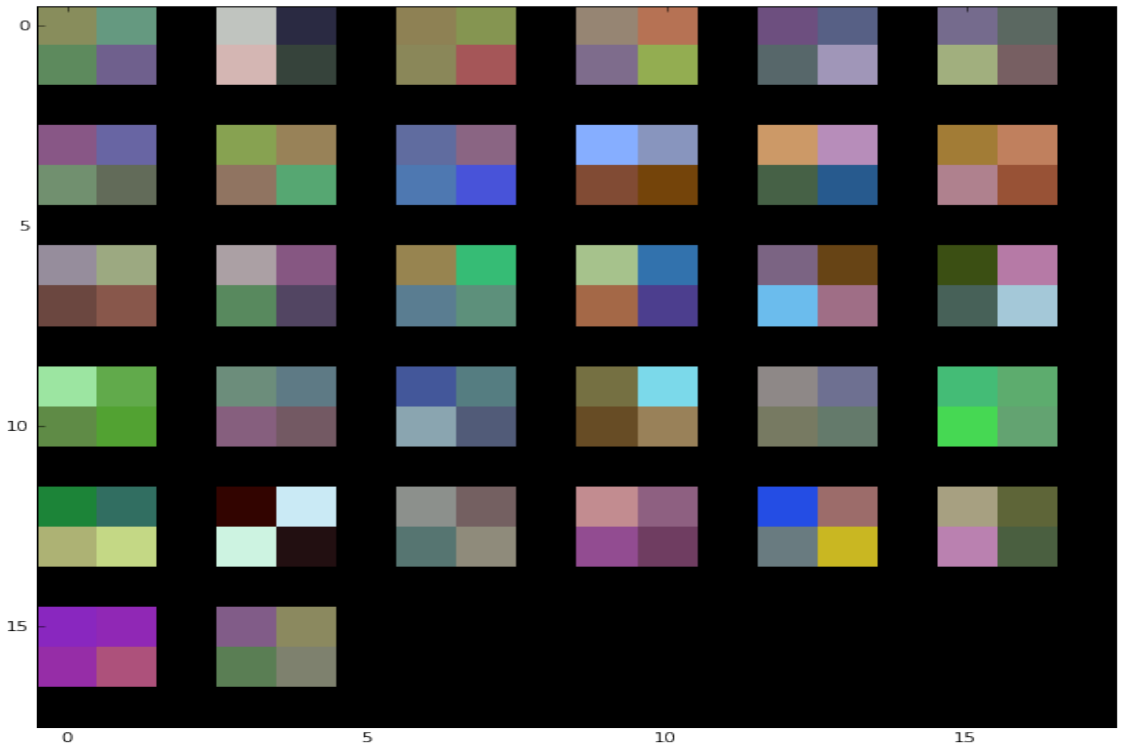
7 Network Fine-Tuning

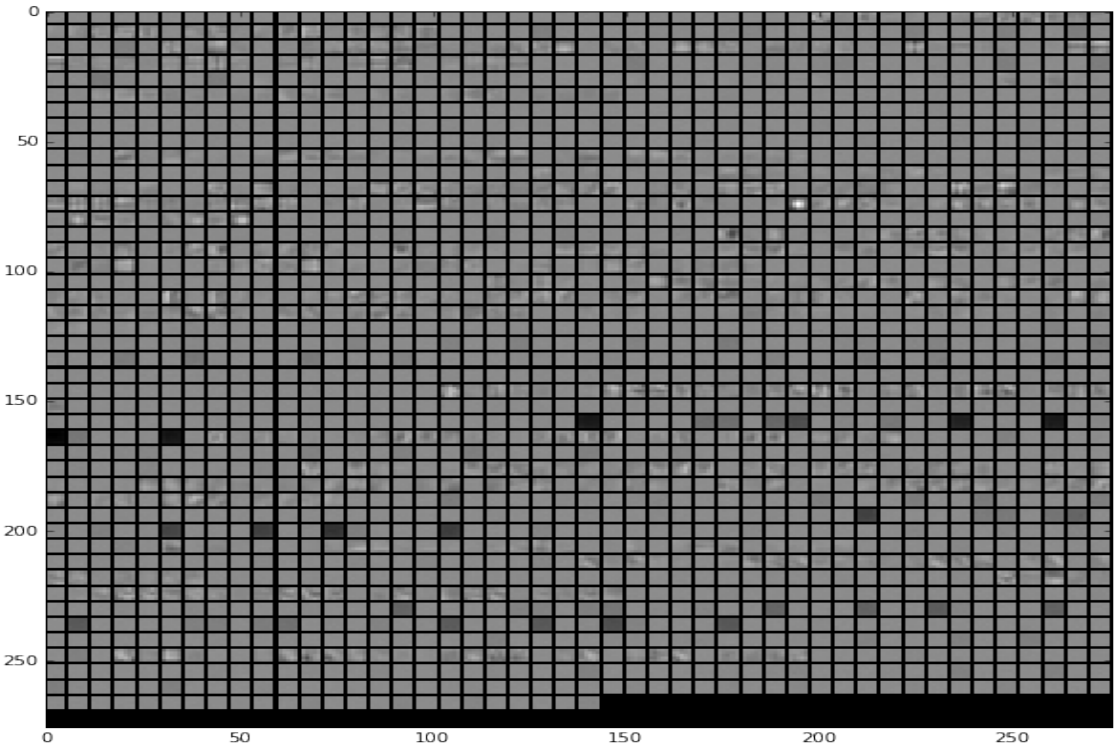


Clearly, test accuracy on cifar10 converges to 70% which is really impressive! In comparison, our model using the cifar100 data had a test accuracy of 40%İn addition, the default cifar10 model gives us 75% test accuracy, and we got 70% So cifar100 model generalizes pretty well on cifar10.

8 Feature visualization

We followed the tutorial posted on the assignment
Here are the filters from first two layers


0
5
10
15
0
5
10
15



They look different.

9 Conclusion

In this assignment we are introduced to convolution neural network and the caffe framework. We practiced using caffe in a cloud computing enviroment(AWS) and got experience with implementing and deploying CNN architecture. Caffe is a fast and (maybe) scalable deep learning framework that allows people quickly set up and run different experiments. It may not be very intuitive to use at first, but after reading tutorials we were able to understand the structure of the framework better. With our new understanding, it became clear Caffe is a handy framework.

Appendices

PART 2, 3 - 4
Filename: prepare.py

```
def prepareData(imgs,labels,filename="training.txt",directory_name="trainData"):
    f = open(filename, "w+")
    for i, (img,label) in enumerate(zip(imgs,labels)):
        x_img = img.reshape((32,32,3),order="F").transpose((1,0,2))

        x_img_filename = "{0}/image_{1}.png".format(directory_name,i)

        # save file as x_img_file_name to location to
        # subdirectory called directory_name
        imsave(x_img_filename, x_img)

        # path to image, relative to current working
        # directory
        file_path = "{0}/{1}".format(os.getcwd(),x_img_filename)

        # write path to image and the image
        # label to training.txt, on the
        # same line.
        f.write("{0} {1}\n".format(file_path,label))
    f.close()

    # move the file we created into the hw3 folder
    os.rename(filename, os.getcwd() + "/caffe/hw3/" + filename);
```

Filename: train_cifar100.sh

```
#!/usr/bin/env sh
TOOLS=/mnt/caffe/build/tools
#/mnt/caffe/build/tools/caffe train \
$TOOLS/caffe.bin train \
    --solver=/mnt/caffe/hw3/cifar100_solver.prototxt
```

Filename: cifar100_solver.prototxt

```
net: "/mnt/caffe/hw3/train_val.prototxt"
test_iter: 100
test_interval: 1000
base_lr: 0.001
momentum: 0.9
weight_decay: 0.004
lr_policy: "fixed"
display: 1000
max_iter: 20000
snapshot: 10000
snapshot_prefix: "/mnt/caffe/hw3/snapshots/cifar100_quick"
solver_mode: GPU
```

Filename: train_val.prototxt

```
name: "hw3"
layer {
  name: "cifar100"
  type: "ImageData"
  top: "data"
  top: "label"
  include {
    phase: TRAIN }
  image_data_param {
    source: "/mnt/caffe/hw3/training.txt"
    batch_size: 64
  }
}
layer {
  name: "cifar100"
  type: "ImageData"
  top: "data"
  top: "label"
  include {
    phase: TEST }
  image_data_param {
    source: "/mnt/caffe/hw3/test.txt"
    batch_size: 64
  }
}

layer {
  name: "conv1"
  type: "Convolution"
  bottom: "data"
  top: "conv1"
  param {
    lr_mult: 1
  }
  param {
    lr_mult: 2
  }
  convolution_param {
    num_output: 32
    kernel_size: 2
    stride: 1
    weight_filler {
      type: "xavier"
    }
    bias_filler {
      type: "constant"
```

```
    }
  }
}
layer {
  name: "pool1"
  type: "Pooling"
  bottom: "conv1"
  top: "pool1"
  pooling_param {
    pool: MAX
    kernel_size: 3
    stride: 2
  }
}
layer {
  name: "relu1"
  type: "ReLU"
  bottom: "pool1"
  top: "pool1"
}
layer {
  name: "conv2"
  type: "Convolution"
  bottom: "pool1"
  top: "conv2"
  param {
    lr_mult: 1
  }
  param {
    lr_mult: 2
  }
  convolution_param {
    num_output: 64
    kernel_size: 5
    pad: 1
    stride: 1
    weight_filler {
      type: "gaussian"
      std: 0.01
    }
    bias_filler {
      type: "constant"
    }
  }
}
layer {
```

```
  name: "relu2"
  type: "ReLU"
  bottom: "conv2"
  top: "conv2"
}
layer {
  name: "pool2"
  type: "Pooling"
  bottom: "conv2"
  top: "pool2"
  pooling_param {
    pool: AVE
    kernel_size: 3
    stride: 2
  }
}
layer {
  name: "conv3"
  type: "Convolution"
  bottom: "pool2"
  top: "conv3"
  param {
    lr_mult: 1
  }
  param {
    lr_mult: 2
  }
  convolution_param {
    num_output: 128
    kernel_size: 2
    stride: 1
    weight_filler {
      type: "gaussian"
      std: 0.01
    }
    bias_filler {
      type: "constant"
    }
  }
}
layer {
  name: "relu3"
  type: "ReLU"
  bottom: "conv3"
  top: "conv3"
}
```

```
layer {
  name: "pool3"
  type: "Pooling"
  bottom: "conv3"
  top: "pool3"
  pooling_param {
    pool: AVE
    kernel_size: 3
    stride: 2
  }
}
layer {
  name: "ip1"
  type: "InnerProduct"
  bottom: "pool3"
  top: "ip1"
  param {
    lr_mult: 1
  }
  param {
    lr_mult: 2
  }
  inner_product_param {
    num_output: 128
    weight_filler {
      type: "gaussian"
      std: 0.1
    }
    bias_filler {
      type: "constant"
    }
  }
}
layer {
  name: "ip2"
  type: "InnerProduct"
  bottom: "ip1"
  top: "ip2"
  param {
    lr_mult: 1
  }
  param {
    lr_mult: 2
  }
  inner_product_param {
    num_output: 100
```

```
        weight_filler {
          type: "gaussian"
          std: 0.1
        }
        bias_filler {
          type: "constant"
        }
      }
    }
    layer {
      name: "accuracy1"
      type: "Accuracy"
      bottom: "ip2"
      bottom: "label"
      top: "accuracy1"
      include {
        phase: TEST
      }
    }
    layer{
       name: "accuracy2"
       type: "Accuracy"
       bottom: "ip2"
       bottom: "label"
       top: "accuracy2"
       include {
           phase: TRAIN
        }
    }
    layer {
      name: "loss"
      type: "SoftmaxWithLoss"
      bottom: "ip2"
      bottom: "label"
      top: "loss"
    }
```

PART 5

```
#trainNlz_val.prototxt
name: "hw3"
layer {
  name: "cifar100Nlz"
  type: "ImageData"
  top: "data"
  top: "label"
```

```
  include {
    phase: TRAIN
  }
  transform_param {
    mean_file: "trainNlzMean.binaryproto"
  }
  image_data_param {
    source: "/mnt/caffe/hw3/trainingNlz.txt"
    batch_size: 64
  }
}
layer {
  name: "cifar100Nlz"
  type: "ImageData"
  top: "data"
  top: "label"
  include {
    phase: TEST
  }
  transform_param {
    mean_file: "testNlzMean.binaryproto"
  }
  image_data_param {
    source: "/mnt/caffe/hw3/testNlz.txt"
    batch_size: 64
  }
}

layer {
  name: "conv1"
  type: "Convolution"
  bottom: "data"
  top: "conv1"
  param {
    lr_mult: 1
  }
  param {
    lr_mult: 2
  }
  convolution_param {
    num_output: 32
    kernel_size: 2
    stride: 1
    weight_filler {
      type: "xavier"
    }
```

```
    bias_filler {
      type: "constant"
    }
  }
}
layer {
  name: "pool1"
  type: "Pooling"
  bottom: "conv1"
  top: "pool1"
  pooling_param {
    pool: MAX
    kernel_size: 3
    stride: 2
  }
}
layer {
  name: "relu1"
  type: "ReLU"
  bottom: "pool1"
  top: "pool1"
}
layer {
  name: "conv2"
  type: "Convolution"
  bottom: "pool1"
  top: "conv2"
  param {
    lr_mult: 1
  }
  param {
    lr_mult: 2
  }
  convolution_param {
    num_output: 64
    kernel_size: 5
    pad: 1
    stride: 1
    weight_filler {
      type: "gaussian"
      std: 0.01
    }
    bias_filler {
      type: "constant"
    }
  }
```

```
}
layer {
  name: "relu2"
  type: "ReLU"
  bottom: "conv2"
  top: "conv2"
}
layer {
  name: "pool2"
  type: "Pooling"
  bottom: "conv2"
  top: "pool2"
  pooling_param {
    pool: AVE
    kernel_size: 3
    stride: 2
  }
}
layer {
  name: "conv3"
  type: "Convolution"
  bottom: "pool2"
  top: "conv3"
  param {
    lr_mult: 1
  }
  param {
    lr_mult: 2
  }
  convolution_param {
    num_output: 128
    kernel_size: 2
    stride: 1
    weight_filler {
      type: "gaussian"
      std: 0.01
    }
    bias_filler {
      type: "constant"
    }
  }
}
layer {
  name: "relu3"
  type: "ReLU"
  bottom: "conv3"
```

```
  top: "conv3"
}
layer {
  name: "pool3"
  type: "Pooling"
  bottom: "conv3"
  top: "pool3"
  pooling_param {
    pool: AVE
    kernel_size: 3
    stride: 2
  }
}
layer {
  name: "ip1"
  type: "InnerProduct"
  bottom: "pool3"
  top: "ip1"
  param {
    lr_mult: 1
  }
  param {
    lr_mult: 2
  }
  inner_product_param {
    num_output: 128
    weight_filler {
      type: "gaussian"
      std: 0.1
    }
    bias_filler {
      type: "constant"
    }
  }
}
layer {
  name: "ip2"
  type: "InnerProduct"
  bottom: "ip1"
  top: "ip2"
  param {
    lr_mult: 1
  }
  param {
    lr_mult: 2
  }
```

```
  inner_product_param {
    num_output: 100
    weight_filler {
      type: "gaussian"
      std: 0.1
    }
    bias_filler {
      type: "constant"
    }
  }
}
layer {
  name: "accuracy1"
  type: "Accuracy"
  bottom: "ip2"
  bottom: "label"
  top: "accuracy1"
  include {
    phase: TEST
  }
}
layer{
   name: "accuracy2"
   type: "Accuracy"
   bottom: "ip2"
   bottom: "label"
   top: "accuracy2"
   include {
       phase: TRAIN
    }
}
layer {
  name: "loss"
  type: "SoftmaxWithLoss"
  bottom: "ip2"
  bottom: "label"
  top: "loss"
}


#train100_val.prototxt
name: "hw3"
layer {
  name: "cifar100"
  type: "ImageData"
  top: "data"
```

```
  top: "label"
  include {
    phase: TRAIN }
  image_data_param {
    source: "/mnt/caffe/hw3/training100.txt"
    batch_size: 64
  }
}
layer {
  name: "cifar100"
  type: "ImageData"
  top: "data"
  top: "label"
  include {
    phase: TEST }
  image_data_param {
    source: "/mnt/caffe/hw3/test.txt"
    batch_size: 64
  }
}

layer {
  name: "conv1"
  type: "Convolution"
  bottom: "data"
  top: "conv1"
  param {
    lr_mult: 1
  }
  param {
    lr_mult: 2
  }
  convolution_param {
    num_output: 32
    kernel_size: 2
    stride: 1
    weight_filler {
      type: "xavier"
    }
    bias_filler {
      type: "constant"
    }
  }
}
layer {
  name: "pool1"
```

```
  type: "Pooling"
  bottom: "conv1"
  top: "pool1"
  pooling_param {
    pool: MAX
    kernel_size: 3
    stride: 2
  }
}
layer {
  name: "relu1"
  type: "ReLU"
  bottom: "pool1"
  top: "pool1"
}
layer {
  name: "conv2"
  type: "Convolution"
  bottom: "pool1"
  top: "conv2"
  param {
    lr_mult: 1
  }
  param {
    lr_mult: 2
  }
  convolution_param {
    num_output: 64
    kernel_size: 5
    pad: 1
    stride: 1
    weight_filler {
      type: "gaussian"
      std: 0.01
    }
    bias_filler {
      type: "constant"
    }
  }
}
layer {
  name: "relu2"
  type: "ReLU"
  bottom: "conv2"
  top: "conv2"
}
```

```
layer {
  name: "pool2"
  type: "Pooling"
  bottom: "conv2"
  top: "pool2"
  pooling_param {
    pool: AVE
    kernel_size: 3
    stride: 2
  }
}
layer {
  name: "conv3"
  type: "Convolution"
  bottom: "pool2"
  top: "conv3"
  param {
    lr_mult: 1
  }
  param {
    lr_mult: 2
  }
  convolution_param {
    num_output: 128
    kernel_size: 2
    stride: 1
    weight_filler {
      type: "gaussian"
      std: 0.01
    }
    bias_filler {
      type: "constant"
    }
  }
}
layer {
  name: "relu3"
  type: "ReLU"
  bottom: "conv3"
  top: "conv3"
}
layer {
  name: "pool3"
  type: "Pooling"
  bottom: "conv3"
  top: "pool3"
```

```
  pooling_param {
    pool: AVE
    kernel_size: 3
    stride: 2
  }
}
layer {
  name: "ip1"
  type: "InnerProduct"
  bottom: "pool3"
  top: "ip1"
  param {
    lr_mult: 1
  }
  param {
    lr_mult: 2
  }
  inner_product_param {
    num_output: 128
    weight_filler {
      type: "gaussian"
      std: 0.1
    }
    bias_filler {
      type: "constant"
    }
  }
}
layer {
  name: "ip2"
  type: "InnerProduct"
  bottom: "ip1"
  top: "ip2"
  param {
    lr_mult: 1
  }
  param {
    lr_mult: 2
  }
  inner_product_param {
    num_output: 100
    weight_filler {
      type: "gaussian"
      std: 0.1
    }
    bias_filler {
```

```
      type: "constant"
    }
  }
}
layer {
  name: "accuracy1"
  type: "Accuracy"
  bottom: "ip2"
  bottom: "label"
  top: "accuracy1"
  include {
    phase: TEST
  }
}
layer{
   name: "accuracy2"
   type: "Accuracy"
   bottom: "ip2"
   bottom: "label"
   top: "accuracy2"
   include {
       phase: TRAIN
    }
}
layer {
  name: "loss"
  type: "SoftmaxWithLoss"
  bottom: "ip2"
  bottom: "label"
  top: "loss"
}


#train300_val.prototxt

name: "hw3"
layer {
  name: "cifar100"
  type: "ImageData"
  top: "data"
  top: "label"
  include {
    phase: TRAIN }
  image_data_param {
    source: "/mnt/caffe/hw3/training300.txt"
    batch_size: 64
```

```
  }
}
layer {
  name: "cifar100"
  type: "ImageData"
  top: "data"
  top: "label"
  include {
    phase: TEST }
  image_data_param {
    source: "/mnt/caffe/hw3/test.txt"
    batch_size: 64
  }
}

layer {
  name: "conv1"
  type: "Convolution"
  bottom: "data"
  top: "conv1"
  param {
    lr_mult: 1
  }
  param {
    lr_mult: 2
  }
  convolution_param {
    num_output: 32
    kernel_size: 2
    stride: 1
    weight_filler {
      type: "xavier"
    }
    bias_filler {
      type: "constant"
    }
  }
}
layer {
  name: "pool1"
  type: "Pooling"
  bottom: "conv1"
  top: "pool1"
  pooling_param {
    pool: MAX
    kernel_size: 3
```

```
    stride: 2
  }
}
layer {
  name: "relu1"
  type: "ReLU"
  bottom: "pool1"
  top: "pool1"
}
layer {
  name: "conv2"
  type: "Convolution"
  bottom: "pool1"
  top: "conv2"
  param {
    lr_mult: 1
  }
  param {
    lr_mult: 2
  }
  convolution_param {
    num_output: 64
    kernel_size: 5
    pad: 1
    stride: 1
    weight_filler {
      type: "gaussian"
      std: 0.01
    }
    bias_filler {
      type: "constant"
    }
  }
}
layer {
  name: "relu2"
  type: "ReLU"
  bottom: "conv2"
  top: "conv2"
}
layer {
  name: "pool2"
  type: "Pooling"
  bottom: "conv2"
  top: "pool2"
  pooling_param {
```

```
    pool: AVE
    kernel_size: 3
    stride: 2
  }
}
layer {
  name: "conv3"
  type: "Convolution"
  bottom: "pool2"
  top: "conv3"
  param {
    lr_mult: 1
  }
  param {
    lr_mult: 2
  }
  convolution_param {
    num_output: 128
    kernel_size: 2
    stride: 1
    weight_filler {
      type: "gaussian"
      std: 0.01
    }
    bias_filler {
      type: "constant"
    }
  }
}
layer {
  name: "relu3"
  type: "ReLU"
  bottom: "conv3"
  top: "conv3"
}
layer {
  name: "pool3"
  type: "Pooling"
  bottom: "conv3"
  top: "pool3"
  pooling_param {
    pool: AVE
    kernel_size: 3
    stride: 2
  }
}
```

```
layer {
  name: "ip1"
  type: "InnerProduct"
  bottom: "pool3"
  top: "ip1"
  param {
    lr_mult: 1
  }
  param {
    lr_mult: 2
  }
  inner_product_param {
    num_output: 128
    weight_filler {
      type: "gaussian"
      std: 0.1
    }
    bias_filler {
      type: "constant"
    }
  }
}
layer {
  name: "ip2"
  type: "InnerProduct"
  bottom: "ip1"
  top: "ip2"
  param {
    lr_mult: 1
  }
  param {
    lr_mult: 2
  }
  inner_product_param {
    num_output: 100
    weight_filler {
      type: "gaussian"
      std: 0.1
    }
    bias_filler {
      type: "constant"
    }
  }
}
layer {
  name: "accuracy1"
```

```
  type: "Accuracy"
  bottom: "ip2"
  bottom: "label"
  top: "accuracy1"
  include {
    phase: TEST
  }
}
layer{
   name: "accuracy2"
   type: "Accuracy"
   bottom: "ip2"
   bottom: "label"
   top: "accuracy2"
   include {
       phase: TRAIN
    }
}
layer {
  name: "loss"
  type: "SoftmaxWithLoss"
  bottom: "ip2"
  bottom: "label"
  top: "loss"
}
```

PART 6

```
# One conv layer network:
# /mnt/caffe/hw3/justin/train_val.prototxt.one

name: "hw3"
layer {
  name: "cifar100"
  type: "ImageData"
  top: "data"
  top: "label"
  include {
    phase: TRAIN }
  image_data_param {
    source: "/mnt/caffe/hw3/training.txt"
    batch_size: 64
  }
}
```

```
layer {
  name: "cifar100"
  type: "ImageData"
  top: "data"
  top: "label"
  include {
    phase: TEST }
  image_data_param {
    source: "/mnt/caffe/hw3/test.txt"
    batch_size: 64
  }
}

layer {
  name: "conv1"
  type: "Convolution"
  bottom: "data"
  top: "conv1"
  param {
    lr_mult: 1
  }
  param {
    lr_mult: 2
  }
  convolution_param {
    num_output: 128
    kernel_size: 6
    stride: 2
    weight_filler {
      type: "xavier"
    }
    bias_filler {
      type: "constant"
    }
  }
}
layer {
  name: "pool1"
  type: "Pooling"
  bottom: "conv1"
  top: "pool1"
  pooling_param {
    pool: MAX
    kernel_size: 10
    stride: 2
  }
```

```
}
layer {
  name: "relu1"
  type: "ReLU"
  bottom: "pool1"
  top: "pool1"
}
layer {
  name: "ip1"
  type: "InnerProduct"
  bottom: "pool1"
  top: "ip1"
  param {
    lr_mult: 1
  }
  param {
    lr_mult: 2
  }
  inner_product_param {
    num_output: 100
    weight_filler {
      type: "gaussian"
      std: 0.1
    }
    bias_filler {
      type: "constant"
    }
  }
}

layer {
  name: "accuracy1"
  type: "Accuracy"
  bottom: "ip1"
  bottom: "label"
  top: "accuracy1"
  include {
    phase: TEST
  }
}
layer{
   name: "accuracy2"
   type: "Accuracy"
   bottom: "ip1"
   bottom: "label"
   top: "accuracy2"
```

```
    include {
        phase: TRAIN
    }
}
layer {
  name: "loss"
  type: "SoftmaxWithLoss"
  bottom: "ip1"
  bottom: "label"
  top: "loss"
}

# One more conv layer network:
# /mnt/caffe/hw3/justin/train_val.prototxt.more
name: "hw3"
layer {
  name: "cifar100"
  type: "ImageData"
  top: "data"
  top: "label"
  include {
    phase: TRAIN }
  image_data_param {
    source: "/mnt/caffe/hw3/training.txt"
    batch_size: 64
  }
}
layer {
  name: "cifar100"
  type: "ImageData"
  top: "data"
  top: "label"
  include {
    phase: TEST }
  image_data_param {
    source: "/mnt/caffe/hw3/test.txt"
    batch_size: 64
  }
}

layer {
  name: "conv1"
  type: "Convolution"
  bottom: "data"
  top: "conv1"
  param {
```

```
    lr_mult: 1
  }
  param {
    lr_mult: 2
  }
  convolution_param {
    num_output: 32
    kernel_size: 2
    stride: 1
    weight_filler {
      type: "xavier"
    }
    bias_filler {
      type: "constant"
    }
  }
}
layer {
  name: "pool1"
  type: "Pooling"
  bottom: "conv1"
  top: "pool1"
  pooling_param {
    pool: MAX
    kernel_size: 3
    stride: 2
  }
}
layer {
  name: "relu1"
  type: "ReLU"
  bottom: "pool1"
  top: "pool1"
}
layer {
  name: "conv2"
  type: "Convolution"
  bottom: "pool1"
  top: "conv2"
  param {
    lr_mult: 1
  }
  param {
    lr_mult: 2
  }
  convolution_param {
```

```
    num_output: 64
    kernel_size: 3
    pad: 1
    stride: 1
    weight_filler {
      type: "gaussian"
      std: 0.01
    }
    bias_filler {
      type: "constant"
    }
  }
}
layer {
  name: "relu2"
  type: "ReLU"
  bottom: "conv2"
  top: "conv2"
}
layer {
  name: "pool2"
  type: "Pooling"
  bottom: "conv2"
  top: "pool2"
  pooling_param {
    pool: AVE
    kernel_size: 3
    stride: 1
  }
}
layer {
  name: "conv3"
  type: "Convolution"
  bottom: "pool2"
  top: "conv3"
  param {
    lr_mult: 1
  }
  param {
    lr_mult: 2
  }
  convolution_param {
    num_output: 64
    kernel_size: 3
    stride: 1
    weight_filler {
```

```
      type: "gaussian"
      std: 0.01
    }
    bias_filler {
      type: "constant"
    }
  }
}
layer {
  name: "relu3"
  type: "ReLU"
  bottom: "conv3"
  top: "conv3"
}
layer {
  name: "pool3"
  type: "Pooling"
  bottom: "conv3"
  top: "pool3"
  pooling_param {
    pool: AVE
    kernel_size: 3
    stride: 2
  }
}
layer {
  name: "conv4"
  type: "Convolution"
  bottom: "pool3"
  top: "conv4"
  param {
    lr_mult: 1
  }
  param {
    lr_mult: 2
  }
  convolution_param {
    num_output: 128
    kernel_size: 2
    stride: 1
    weight_filler {
      type: "gaussian"
      std: 0.01
    }
    bias_filler {
      type: "constant"
```

```
    }
  }
}
layer {
  name: "relu4"
  type: "ReLU"
  bottom: "conv4"
  top: "conv4"
}
layer {
  name: "pool4"
  type: "Pooling"
  bottom: "conv4"
  top: "pool4"
  pooling_param {
    pool: AVE
    kernel_size: 2
    stride: 2
  }
}
layer {
  name: "ip1"
  type: "InnerProduct"
  bottom: "pool4"
  top: "ip1"
  param {
    lr_mult: 1
  }
  param {
    lr_mult: 2
  }
  inner_product_param {
    num_output: 128
    weight_filler {
      type: "gaussian"
      std: 0.1
    }
    bias_filler {
      type: "constant"
    }
  }
}
layer {
  name: "ip2"
  type: "InnerProduct"
  bottom: "ip1"
```

```
  top: "ip2"
  param {
    lr_mult: 1
  }
  param {
    lr_mult: 2
  }
  inner_product_param {
    num_output: 100
    weight_filler {
      type: "gaussian"
      std: 0.1
    }
    bias_filler {
      type: "constant"
    }
  }
}
layer {
  name: "accuracy1"
  type: "Accuracy"
  bottom: "ip2"
  bottom: "label"
  top: "accuracy1"
  include {
    phase: TEST
  }
}
layer{
   name: "accuracy2"
   type: "Accuracy"
   bottom: "ip2"
   bottom: "label"
   top: "accuracy2"
   include {
       phase: TRAIN
    }
}
layer {
  name: "loss"
  type: "SoftmaxWithLoss"
  bottom: "ip2"
  bottom: "label"
  top: "loss"
}
```

PART 7

Filename: caffe/hw3/monica/cifar10_solver.prototxt

```
net: "/mnt/caffe/hw3/monica/train_val.prototxt"
test_iter: 10
test_interval: 100
base_lr: 0.0001
momentum: 0.9
weight_decay: 0.004
lr_policy: "fixed"
display: 1000
max_iter: 25000
snapshot: 1000
snapshot_prefix:"/mnt/caffe/hw3/monica/snapshots/cifar10_"
solver_mode: GPU
```

Filename: caffe/hw3/monica/train_val.txt

```
name: "hw3"
layer {
  name: "cifar100"
  type: "Data"
  top: "data"
  top: "label"
  include {
    phase: TRAIN
  }
  transform_param {
    mean_file: "examples/cifar10/mean.binaryproto"
  }
  data_param {
    source: "examples/cifar10/cifar10_train_lmdb"
    batch_size: 64
    backend: LMDB
  }
}
layer {
  name: "cifar100"
  type: "Data"
  top: "data"
  top: "label"
  include {
    phase: TEST
  }
  transform_param {
    mean_file: "examples/cifar10/mean.binaryproto"
  }
```

```
  data_param {
    source: "examples/cifar10/cifar10_test_lmdb"
    batch_size: 64
    backend: LMDB
  }
}

layer {
  name: "conv1"
  type: "Convolution"
  bottom: "data"
  top: "conv1"
  param {
    lr_mult: 1
  }
  param {
    lr_mult: 2
  }
  convolution_param {
    num_output: 32
    kernel_size: 2
    stride: 1
    weight_filler {
      type: "xavier"
    }
    bias_filler {
      type: "constant"
    }
  }
}
layer {
  name: "pool1"
  type: "Pooling"
  bottom: "conv1"
  top: "pool1"
  pooling_param {
    pool: MAX
    kernel_size: 3
    stride: 2
  }
}
layer {
  name: "relu1"
  type: "ReLU"
  bottom: "pool1"
  top: "pool1"
```

```
}
layer {
  name: "conv2"
  type: "Convolution"
  bottom: "pool1"
  top: "conv2"
  param {
    lr_mult: 1
  }
  param {
    lr_mult: 2
  }
  convolution_param {
    num_output: 64
    kernel_size: 5
    pad: 1
    stride: 1
    weight_filler {
      type: "gaussian"
      std: 0.01
    }
    bias_filler {
      type: "constant"
    }
  }
}
layer {
  name: "relu2"
  type: "ReLU"
  bottom: "conv2"
  top: "conv2"
}
layer {
  name: "pool2"
  type: "Pooling"
  bottom: "conv2"
  top: "pool2"
  pooling_param {
    pool: AVE
    kernel_size: 3
    stride: 2
  }
}
layer {
  name: "conv3"
  type: "Convolution"
```

```
  bottom: "pool2"
  top: "conv3"
  param {
    lr_mult: 1
  }
  param {
    lr_mult: 2
  }
  convolution_param {
    num_output: 128
    kernel_size: 2
    stride: 1
    weight_filler {
      type: "gaussian"
      std: 0.01
    }
    bias_filler {
      type: "constant"
    }
  }
}
layer {
  name: "relu3"
  type: "ReLU"
  bottom: "conv3"
  top: "conv3"
}
layer {
  name: "pool3"
  type: "Pooling"
  bottom: "conv3"
  top: "pool3"
  pooling_param {
    pool: AVE
    kernel_size: 3
    stride: 2
  }
}
layer {
  name: "ip1"
  type: "InnerProduct"
  bottom: "pool3"
  top: "ip1"
  param {
    lr_mult: 1
  }
```

```
  param {
    lr_mult: 2
  }
  inner_product_param {
    num_output: 128
    weight_filler {
      type: "gaussian"
      std: 0.1
    }
    bias_filler {
      type: "constant"
    }
  }
}
layer {
  name: "ip2_cifar10"
  type: "InnerProduct"
  bottom: "ip1"
  top: "ip2_cifar10"
  param {
    lr_mult: 10
    decay_mult: 1
  }
  param {
    lr_mult: 20
    decay_mult: 0
  }
  inner_product_param {
    num_output: 10
    weight_filler {
      type: "gaussian"
      std: 0.1
    }
    bias_filler {
      type: "constant"
    }
  }
}
layer {
  name: "accuracy1"
  type: "Accuracy"
  bottom: "ip2_cifar10"
  bottom: "label"
  top: "accuracy1"
  include {
    phase: TEST
```

```
  }
}
layer{
   name: "accuracy2"
   type: "Accuracy"
   bottom: "ip2_cifar10"
   bottom: "label"
   top: "accuracy2"
   include {
       phase: TRAIN
    }
}
layer {
  name: "loss"
  type: "SoftmaxWithLoss"
  bottom: "ip2_cifar10"
  bottom: "label"
  top: "loss"
}
```

PART 8

```
def vis_square(data, padsize=1, padval=0):
    data -= data.min()
    data /= data.max()

    # force the number of filters to be square
    n = int(np.ceil(np.sqrt(data.shape[0])))
    padding = ((0, n ** 2 - data.shape[0]), (0, padsize), (0, padsize)) + ((0, 0
    data = np.pad(data, padding, mode='constant', constant_values=(padval, padva

    # tile the filters into an image
    data = data.reshape((n, n) + data.shape[1:]).transpose((0, 2, 1, 3) + tuple(
    data = data.reshape((n * data.shape[1], n * data.shape[3]) + data.shape[4:])

    plt.imshow(data)

import caffe
caffe_root = '../'
mynet = caffe.Net(caffe_root+'hw3/justin/train_val.prototxt',caffe_root+'hw3/jus

vis_square(mynet.params['conv1'][0].data.transpose(0, 2, 3, 1))
a = mynet.params['conv2'][0].data
vis_square(a.reshape(32*64, 5, 5))
```