Colinear Convolution Layer

Pang Liang

October 10, 2014

1 Cifar10 Database

The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. (See http://www.cs.toronto.edu/~kriz/cifar.html)

Fig 1 are the classes in the dataset, as well as 10 random images from each.

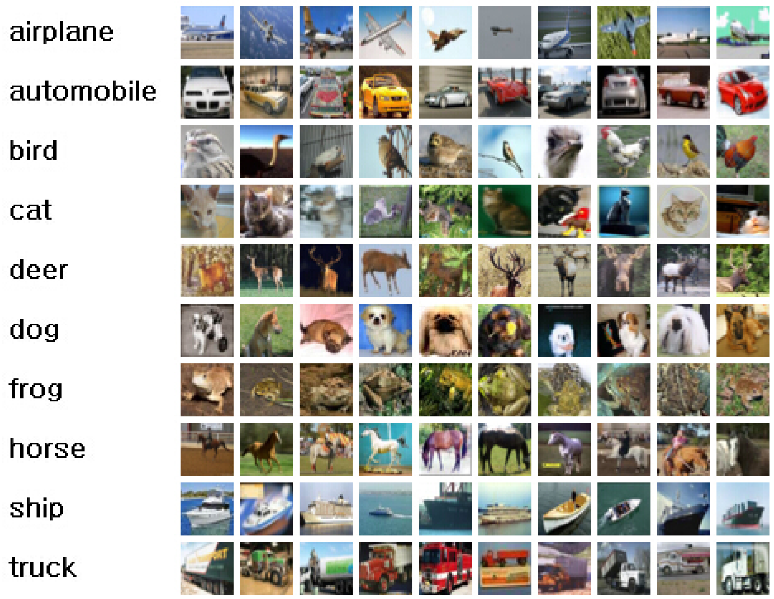

Figure 1: Example of cifar10 database.

2 Caffe Framework of Deep Learning

Caffe is a deep learning framework developed with cleanliness, readability, and speed in mind. (See http://caffe.berkeleyvision.org)

It implement many kind of layers list below:

- Vision Layers

  - Convolution $\surd$
  - Pooling $\surd$
  - Local Response Normalization (LRN) $\surd$

- Loss Layers
  - Softmax $\surd$
  - Sum-of-Squares / Euclidean
  - Hinge / Margin
  - Sigmoid Cross-Entropy

- Activation / Neuron Layers
  - ReLU / Rectified-Linear and Leaky-ReLU $\surd$
  - Sigmoid
  - TanH / Hyperbolic Tangent

- Common Layers
  - Inner Product $\surd$
  - Splitting
  - Flattening
  - Concatenation
  - Slicing

3 Fast Version of NN Structure for Cifar10

The Cifar10 model Fig 2 is a CNN that composes layers of convolution, pooling, rectified linear unit (ReLU) nonlinearities, and local contrast normalization with a linear classifier on top of it all.

The inputs of the model are not the raw images or raw images normalized into $[0,1]$, but the images preprocess by the image mean operation. In other word, the pixel at same position average over all images, so the input pixel value range is $[-127,127]$. In the end of the network we use Softmax loss as the classification loss.

4 Colinear Convolution Layer in Formula

4.1 Feed Forward

$$\begin{aligned} a_{in} &= e_i F_n e_i^T + S_n e_i^T + b_n \\ &= \textstyle\sum_x \sum_y e_{ix} f_{xyn} e_{iy} + \sum_x s_{xn} e_{ix} + b_n \end{aligned} \tag{1}$$

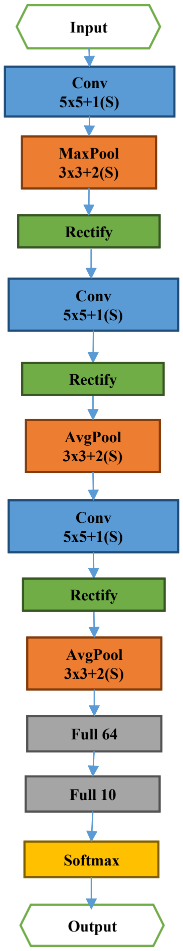

Figure 2: Cifar10 quick model.

4.2 Back Propagation

$$\sum_n \frac{\partial E}{\partial a_{in}} = \epsilon_{in} \tag{2}$$

- Error Propagation

$$\frac{\partial E}{\partial e_{ix}} = \sum_n \frac{\partial E}{\partial a_{in}} \cdot \frac{\partial a_{in}}{\partial e_{ix}} = \sum_n \epsilon_{in}((\sum_y (f_{xyn} + f_{yxn})e_{iy}) + s_{xn}) \tag{3}$$

- Weight Update

$$\frac{\partial E}{\partial f_{xyn}} = \sum_i \frac{\partial E}{\partial a_{in}} \cdot \frac{\partial a_{in}}{\partial f_{xyn}} = \sum_i \epsilon_{in} e_{ix} e_{iy} \tag{4}$$

$$\frac{\partial E}{\partial s_{xn}} = \sum_i \frac{\partial E}{\partial a_{in}} \cdot \frac{\partial a_{in}}{\partial s_{xn}} = \sum_i \epsilon_{in} e_{ix} \tag{5}$$

$$\frac{\partial E}{\partial b_n} = \sum_i \frac{\partial E}{\partial a_{in}} \cdot \frac{\partial a_{in}}{\partial b_n} = \sum_i \epsilon_{in} \tag{6}$$

See details in Fig 3.

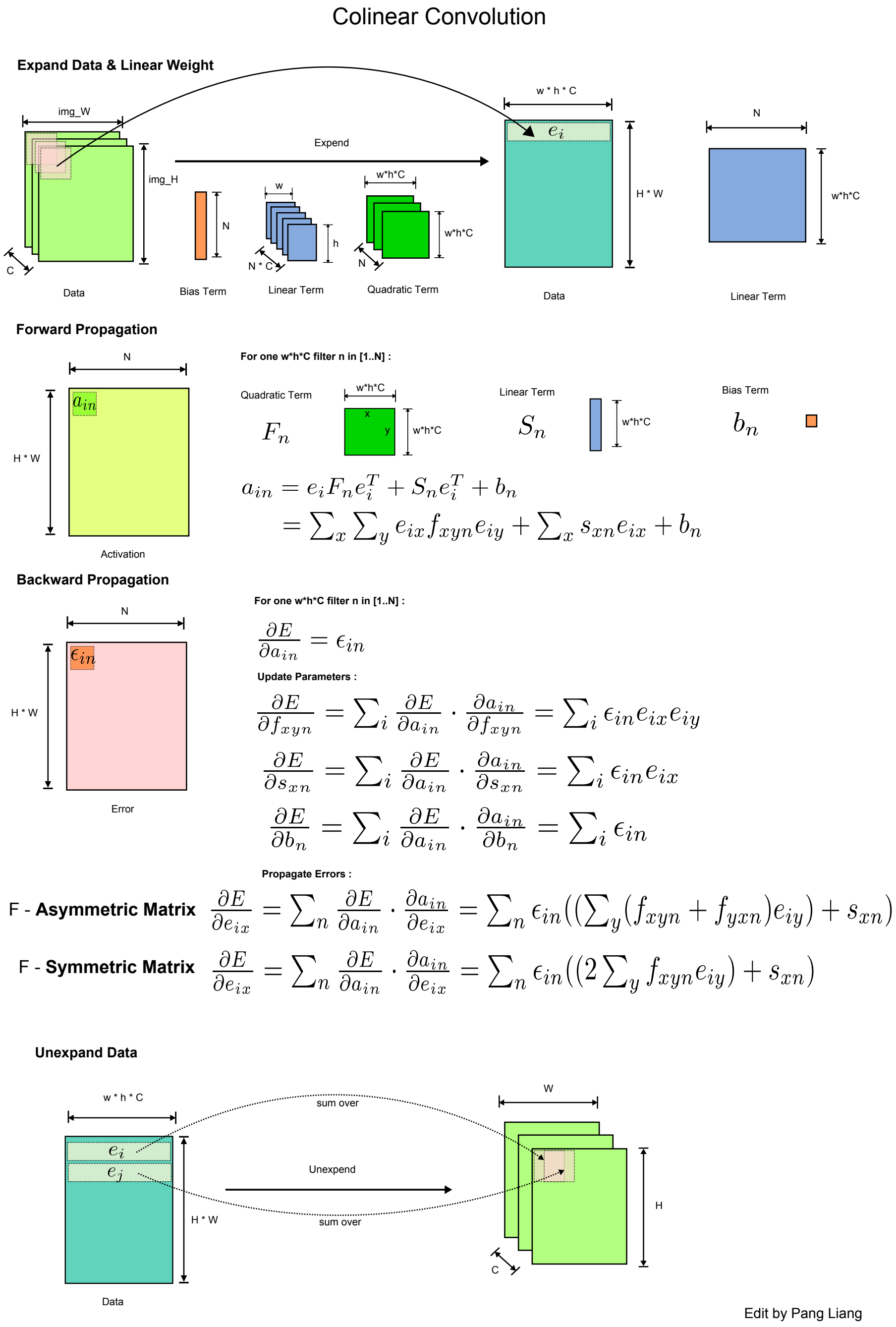
Colinear Convolution
Expand Data & Linear Weight
img_W
img_H
C
Data
Expend
N
Bias Term
w
h
N * C
Linear Term
w*h*C
w*h*C
N
Quadratic Term
w * h * C
e_i
H * W
Data
N
w*h*C
Linear Term
Forward Propagation
N
a_{in}
H * W
Activation
For one w*h*C filter n in [1..N] :
Quadratic Term
F_n
w*h*C
x
y
w*h*C
Linear Term
S_n
w*h*C
Bias Term
b_n
$a_{in} = e_i F_n e_i^T + S_n e_i^T + b_n$
$= \sum_x \sum_y e_{ix} f_{xyn} e_{iy} + \sum_x s_{xn} e_{ix} + b_n$
Backward Propagation
N
ϵ_{in}
H * W
Error
For one w*h*C filter n in [1..N] :
$\frac{\partial E}{\partial a_{in}} = \epsilon_{in}$
Update Parameters :
$\frac{\partial E}{\partial f_{xyn}} = \sum_i \frac{\partial E}{\partial a_{in}} \cdot \frac{\partial a_{in}}{\partial f_{xyn}} = \sum_i \epsilon_{in} e_{ix} e_{iy}$
$\frac{\partial E}{\partial s_{xn}} = \sum_i \frac{\partial E}{\partial a_{in}} \cdot \frac{\partial a_{in}}{\partial s_{xn}} = \sum_i \epsilon_{in} e_{ix}$
$\frac{\partial E}{\partial b_n} = \sum_i \frac{\partial E}{\partial a_{in}} \cdot \frac{\partial a_{in}}{\partial b_n} = \sum_i \epsilon_{in}$
Propagate Errors :
F - Asymmetric Matrix $\frac{\partial E}{\partial e_{ix}} = \sum_n \frac{\partial E}{\partial a_{in}} \cdot \frac{\partial a_{in}}{\partial e_{ix}} = \sum_n \epsilon_{in}((\sum_y (f_{xyn} + f_{yxn}) e_{iy}) + s_{xn})$
F - Symmetric Matrix $\frac{\partial E}{\partial e_{ix}} = \sum_n \frac{\partial E}{\partial a_{in}} \cdot \frac{\partial a_{in}}{\partial e_{ix}} = \sum_n \epsilon_{in}((2 \sum_y f_{xyn} e_{iy}) + s_{xn})$
Unexpand Data
w * h * C
e_i
e_j
H * W
Data
sum over
Unexpend
sum over
W
H
C
Edit by Pang Liang

Figure 3: Colinear Convolution Layer.

5 Experiments

We replace the first Convolution layer to our Colinear Convolution layer. Weights in quadratic term are initialed in range of $[0, 0.000001]$, for the sake of the input range $[-127.127]$ and form of $e*f*e$. We want the activations of the layer not too large, so the weights in quadratic term are very small. For the same reason, the weight decay is set to 1000.

In Fig 4 and Fig 5 ,the number 16 or 32 denotes the first convolution layer output channel number. The lines entitled with Base mean that the result produce by Convolution layer, and others without produce by Colinear Convolution layer. The experiments show that the original configuration of the network is better than what we proposed.But there're some points we should optimize, which will show in next section.

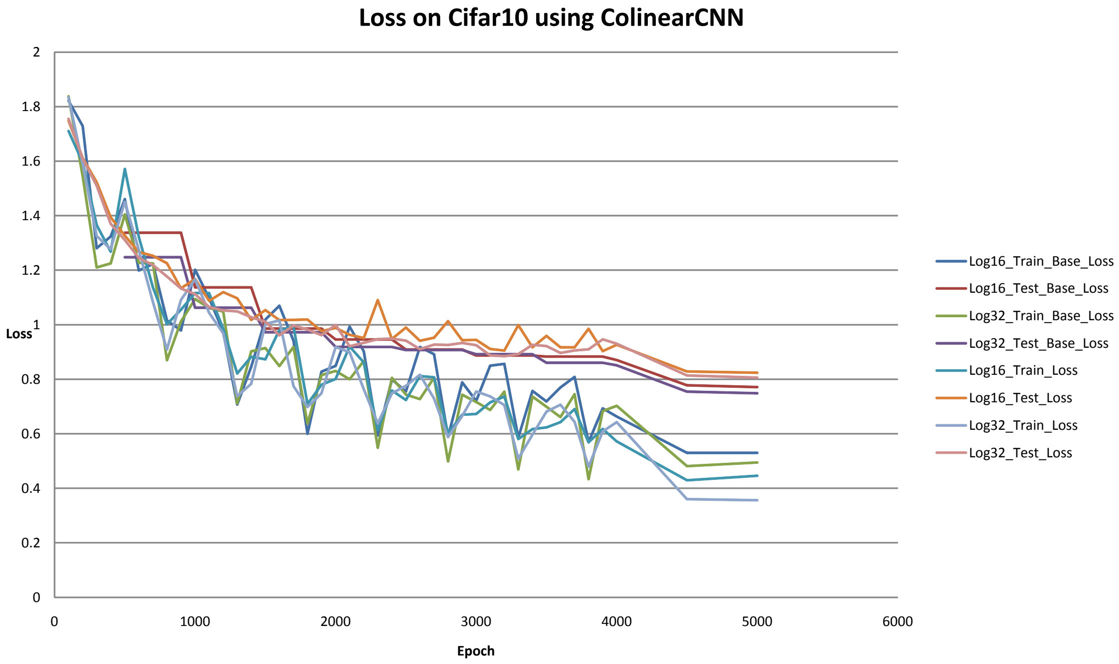

Figure 4: Train loss and Test loss of four net structures.

6 L1 Regularize on CCNN

Using L1 regularize on the parameters always produces a more sparse result, which meanings the bigger the regularizer α the more 0 in parameters. Different from the L2 regularize $\lambda||W||_2$, L1 regularize has the form of $\alpha||W||_1$. L2 regularize can derived everywhere, see Eq 7.

$$\frac{\partial\lambda||W||_2}{\partial W} = 2\lambda W \tag{7}$$

While L1 doesn't has this good property. Its derivative is Eq 8.

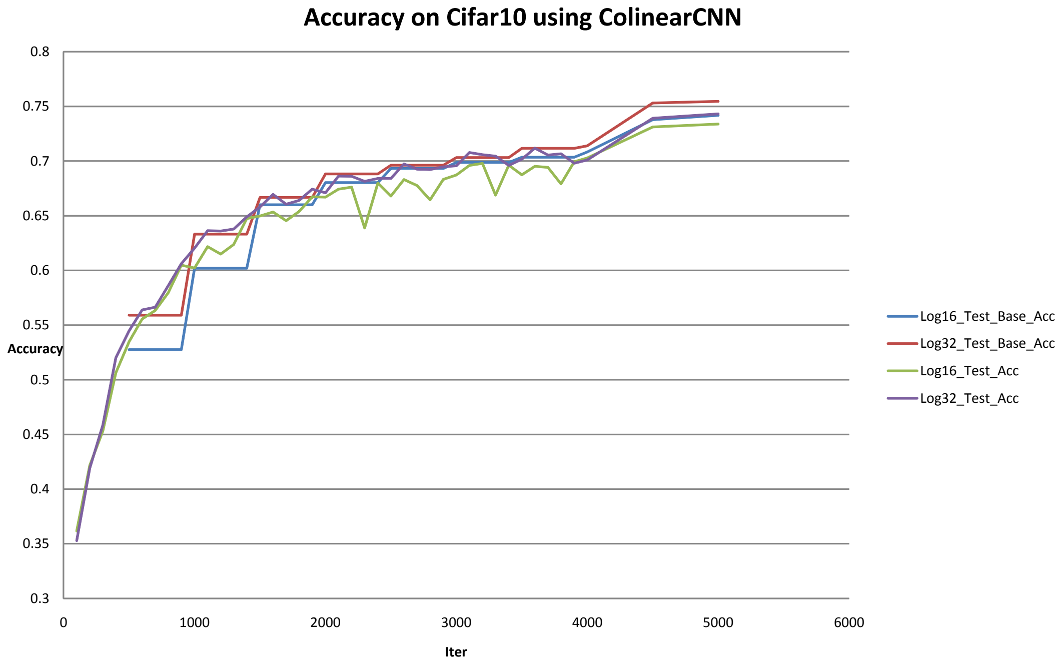

Figure 5: Test Accuracy of four net structures.

$$\frac{\partial \alpha ||W||_1}{\partial W} = \begin{cases} -\alpha & W < 0 \\ undefine & W = 0 \\ \alpha & W \end{cases} \tag{8}$$

So in order to implement L1 regularize, we need to know W's sign. For instance one parameter w_t at t's iterator, then we want to calculate next update of w_t denoted as w_{t+1}.

$$w_{t+1} = w_t + \Delta w - sign(w)\alpha \tag{9}$$

$$\hat{w_{t+1}} = w_t + \Delta w \tag{10}$$

then we need w_{t+1} and $w_t + \Delta w$ have the same sign. So the last term $-sign(w)\alpha$ will not change the sign. So we use Eq 11 for implement.

$$w_{t+1} = \begin{cases} max(\hat{w_{t+1}} - \alpha, 0) & \hat{w_{t+1}} >= 0 \\ min(\hat{w_{t+1}} + \alpha, 0) & \hat{w_{t+1}} < 0 \end{cases} \tag{11}$$

7 Symmetric Strategy on CCNN

Changing Asymmetric Matrix F to Symmetric Matrix is natural to us, because the relation between e_{ix} and e_{iy} should be same. That means $f_{xyn} = f_{yxn}$. So we decrease parameter number from N^2 to $N * (N + 1)/2$.

Then we need modify some formula in BP process.

- Error Propagation

$$\frac{\partial E}{\partial e_{ix}} = \sum_n \frac{\partial E}{\partial a_{in}} \cdot \frac{\partial a_{in}}{\partial e_{ix}} = \sum_n \epsilon_{in}((\sum_y 2 \cdot \left\{ \begin{array}{ll} f_{xyn} & x <= y \\ f_{yxn} & x > y \end{array} \right\} \cdot e_{iy}) + s_{xn}) \tag{12}$$

- Weight Update

$$\frac{\partial E}{\partial f_{xyn}} = \sum_i \frac{\partial E}{\partial a_{in}} \cdot \frac{\partial a_{in}}{\partial f_{xyn}} = \left\{ \begin{array}{ll} 2\sum_i \epsilon_{in} e_{ix} e_{iy} & x < y \\ \sum_i \epsilon_{in} e_{ix} e_{iy} & x = y \end{array} \right. \tag{13}$$

8 Mask the Parameters

Maybe this idea can be compared with dropout connections.

9 Restriction and Regularization of CCNN

1. Change the Asymmetric to Symmetric, in order to decrease the parameter numbers.
2. Change the L2 regularize to L1 regularize, for the 2 order relation is weaker than the 1 order relation and we need not that much parameters to model it.
3. Using mask matrix to erase the relations cross the channels.