Comparison between local binary CNN and vanilla CNN

Jianhao Peng, Hanchao Yu, You Wu

jianhao2@illinois.edu,
hanchao2@illinois.edu,
youw3@illinois.edu

Background Knowledge

- Local Binary Pattern(LBP) is widely used in the face recognition community as a feature of images.

- It is basically a weighted sum of a geometric sequence.
  - $LBP(x_c, y_c) = \sum_{n=0}^{L-1} s(i_n, i_c) \times 2^n$
- Where (x_c, y_c) is the center pixel.

*pictures from tensorflow.org

Background Knowledge

- $LBP(x_c, y_c) = \sum_{n=0}^{L-1} s(i_n, i_c) \times 2^n$
- Where (x_c, y_c) is the center pixel.
- Constraints:

*pictures from tensorflow.org

Background Knowledge

- $LBP(x_c, y_c) = \sum_{n=0}^{L-1} s(i_n, i_c) \times 2^n$
- Where (x_c, y_c) is the center pixel.
- Constraints:
  - Fixed base.

*pictures from tensorflow.org

Background Knowledge

- $LBP(x_c, y_c) = \sum_{n=0}^{L-1} s(i_n, i_c) \times 2^n$
- Where (x_c, y_c) is the center pixel.

- Constraints:
  - Fixed base. 2 in the above case.

*pictures from tensorflow.org

Background Knowledge

- $LBP(x_c, y_c) = \sum_{n=0}^{L-1} s(i_n, i_c) \times 2^n$
- Where (x_c, y_c) is the center pixel.
- Constraints:
  - Fixed base. 2 in the above case.
  - Fixed pivot.

*pictures from tensorflow.org

Background Knowledge

- $LBP(x_c, y_c) = \sum_{n=0}^{L-1} s(i_n, i_c) \times 2^n$
- Where (x_c, y_c) is the center pixel.
- Constraints:
  - Fixed base. 2 in the above case.
  - Fixed pivot. (x_c, y_c)

*pictures from tensorflow.org

Background Knowledge

- $LBP(x_c, y_c) = \sum_{n=0}^{L-1} s(i_n, i_c) \times 2^n$
- Where (x_c, y_c) is the center pixel.
- Constraints:
  - Fixed base. 2 in the above case.
  - Fixed pivot. (x_c, y_c)
  - Fixed ordering of neighbor pixels.

*pictures from tensorflow.org

Background Knowledge

- The LBP function can actually be implemented via convolution!
-

$$LBP(x_c, y_c) = \sum_{n=0}^{8-1} s(i_n, i_c) \times 2^n$$

*pictures from tensorflow.org

Background Knowledge

- The LBP function can actually be implemented via convolution!
-

$$LBP(x_c, y_c) = \sum_{n=0}^{8-1} s(i_n, i_c) \times 2^n$$

- $F(\boldsymbol{x}) = \sum_{i=1}^{8} \sigma(\boldsymbol{b_i} * \boldsymbol{x}) \times \boldsymbol{V}_i$

*pictures from tensorflow.org

Background Knowledge

- The LBP function can actually be implemented via convolution!
-

$$LBP(x_c, y_c) = \sum_{n=0}^{8-1} s(i_n, i_c) \times 2^n$$

- $F(\boldsymbol{x}) = \sum_{i=1}^{8} \sigma(\boldsymbol{b_i} * \boldsymbol{x}) \times \boldsymbol{V}_i$

- Where $\boldsymbol{V} = [2^7, 2^6, \dots, 2^0]$ is the weight vector.

*pictures from tensorflow.org

Background Knowledge

- Equivalent form of LBP function:
  - $F(\boldsymbol{x}) = \sum_{i=1}^{8} \sigma(\boldsymbol{b_i} * \boldsymbol{x}) \times \boldsymbol{V}_i$

- Benefits:

*pictures from tensorflow.org

Background Knowledge

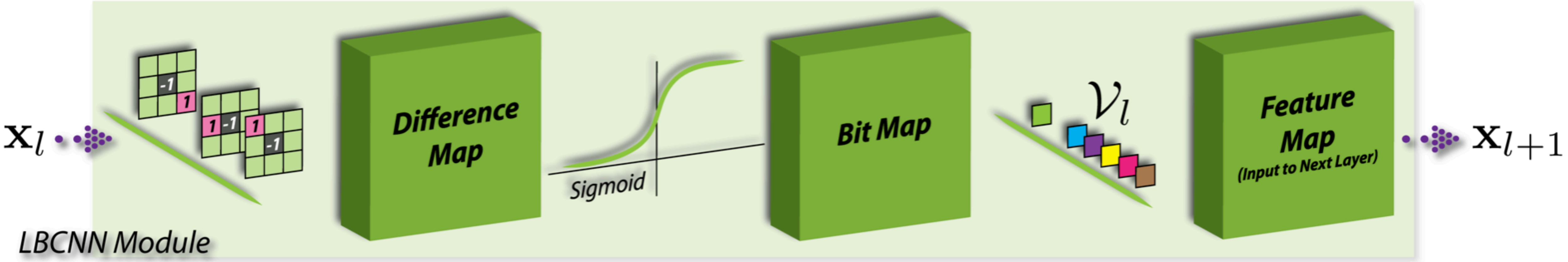
- Equivalent form of LBP function:
  - $F(\boldsymbol{x}) = \sum_{i=1}^{8} \sigma(\boldsymbol{b_i} * \boldsymbol{x}) \times \boldsymbol{V}_i$
- Benefits:
  - Flexible base! Learn $\boldsymbol{V}_i$ by a NN.
  - Learnable ordering of neighbor pixels
  - Learnable pivot.

*pictures from tensorflow.org

Background Knowledge

- Equivalent form of LBP function:
  - $F(\boldsymbol{x}) = \sum_{i=1}^{8} \sigma(\boldsymbol{b_i} * \boldsymbol{x}) \times \boldsymbol{V}_i$

- Benefits:
  - Flexible base! Learn $\boldsymbol{V}_i$ by a NN.
  - Learnable ordering of neighbor pixels
  - Learnable pivot.



Theoretical guarantee of LBC module

Theorem 3.5. *Let* $\mathbf{B} \in \mathbb{R}^{m \times N}$ *be a Bernoulli random matrix with the same subgaussian parameter* c *in (6), and* $\mathbf{x} \in \mathbb{R}^N$ *be a fixed vector and* $\|\mathbf{x}\|_2 > 0$*, with* $N = p \cdot h \cdot w$*. Let* $\boldsymbol{\xi} = \mathbf{B}\mathbf{x} \in \mathbb{R}^m$*. Then, for all* $t \in (0, 1)$*, there exists a matrix* $\mathbf{B}$ *and an index* $i \in [m]$ *such that*

$$\mathbb{P}\left(\xi_i \geq \underbrace{\sqrt{(1-t)}\|\mathbf{x}\|_2}_{>0}\right) \geq 1 - 2\exp(-\tilde{c}t^2 m) \tag{8}$$

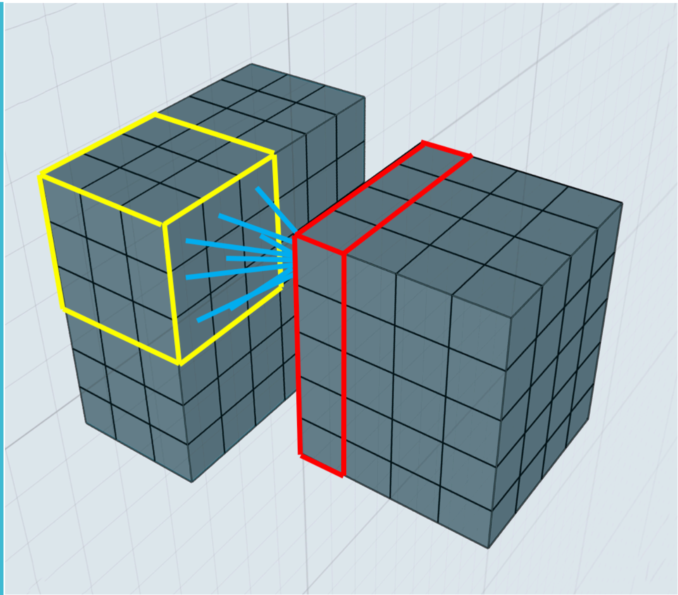
Theoretical guarantee of LBC module

Theorem 3.5. *Let* $\mathbf{B} \in \mathbb{R}^{m \times N}$ *be a Bernoulli random matrix with the same subgaussian parameter* c *in (6), and* $\mathbf{x} \in \mathbb{R}^N$ *be a fixed vector and* $\|\mathbf{x}\|_2 > 0$*, with* $N = p \cdot h \cdot w$*. Let* $\boldsymbol{\xi} = \mathbf{B}\mathbf{x} \in \mathbb{R}^m$*. Then, for all* $t \in (0, 1)$*, there exists a matrix* $\mathbf{B}$ *and an index* $i \in [m]$ *such that*

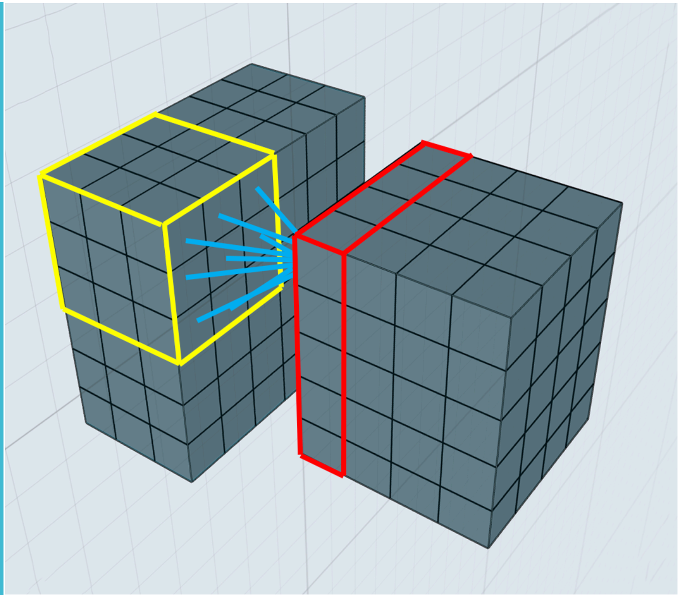
$$\mathbb{P}\left(\xi_i \geq \underbrace{\sqrt{(1-t)}\|\mathbf{x}\|_2}_{>0}\right) \geq 1 - 2\exp(-\tilde{c}t^2 m) \tag{8}$$

Conclusion:
LBC module is a good approximation of Convolution layer in NN

Reduction in Number of learnable weights



Reduction in Number of learnable weights



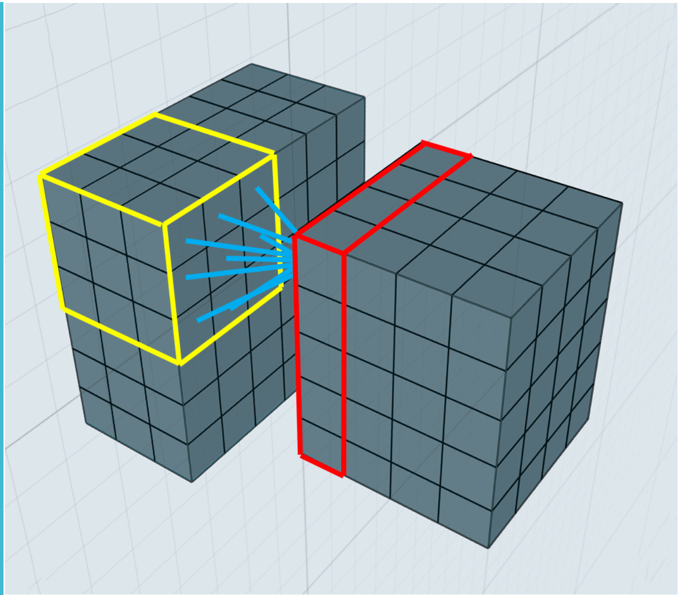
Input: p-channel tensor

Filter size: kH * kW

Number of different filters:
q [depth of next layer]

Number of Weights:
p * kH * kW * q

Reduction in Number of learnable weights

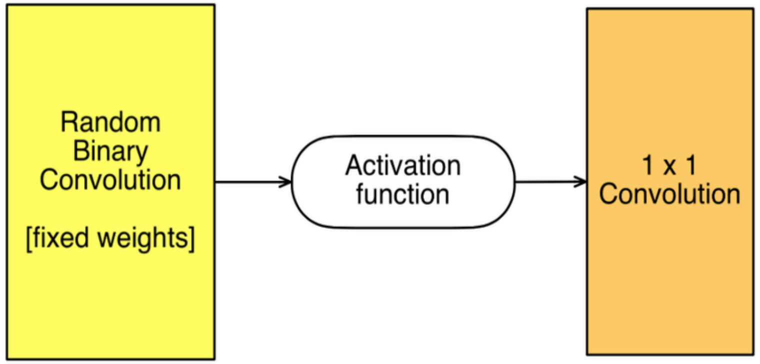


Input: p-channel tensor

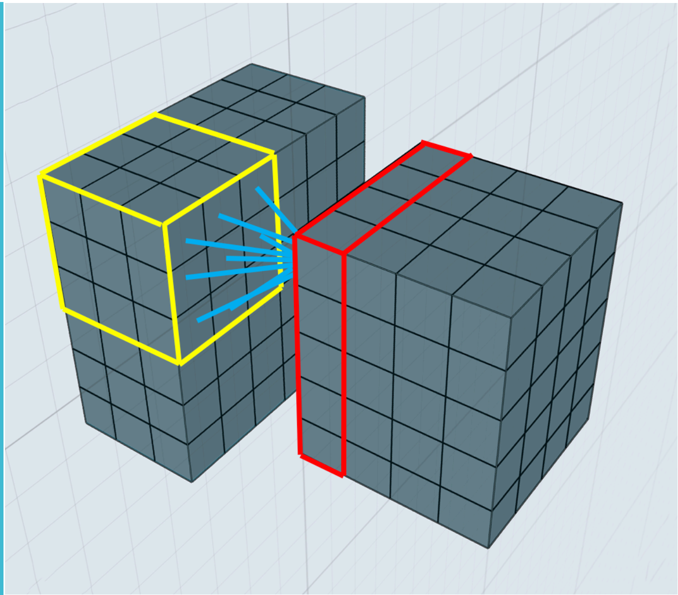
Filter size: kH * kW

Number of different filters:
q [depth of next layer]

Number of Weights:
p * kH * kW * q



Reduction in Number of learnable weights

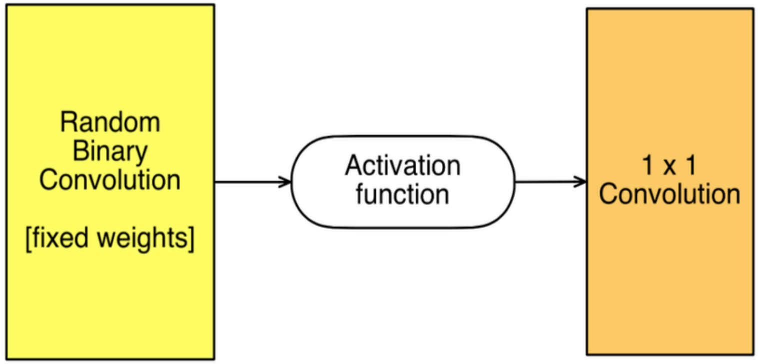


Input: p-channel tensor

Filter size: kH * kW

Number of different filters: q [depth of next layer]

Number of Weights: p * kH * kW * q

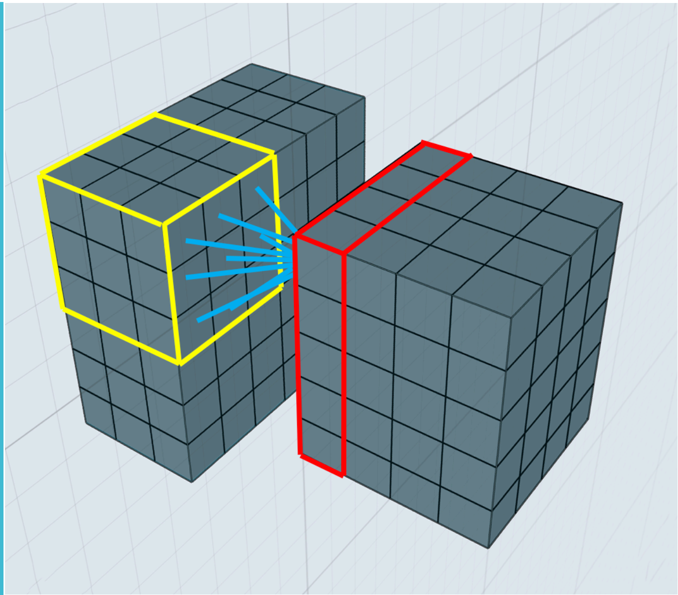


Input: p-channel tensor

Number of binary filters: m

Number of Weights: m * q

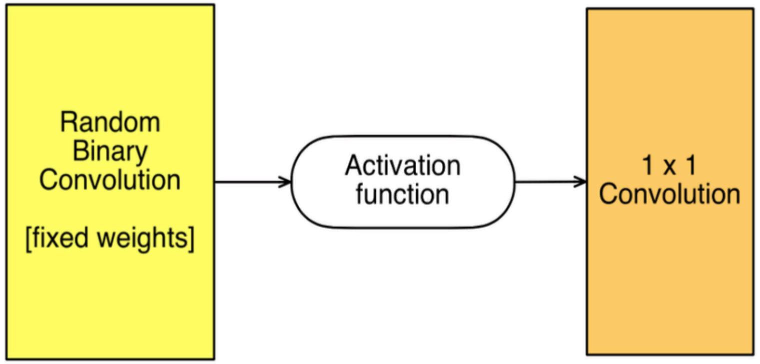
Reduction in Number of learnable weights



Input: p-channel tensor

Filter size: kH * kW

Number of different filters: q [depth of next layer]



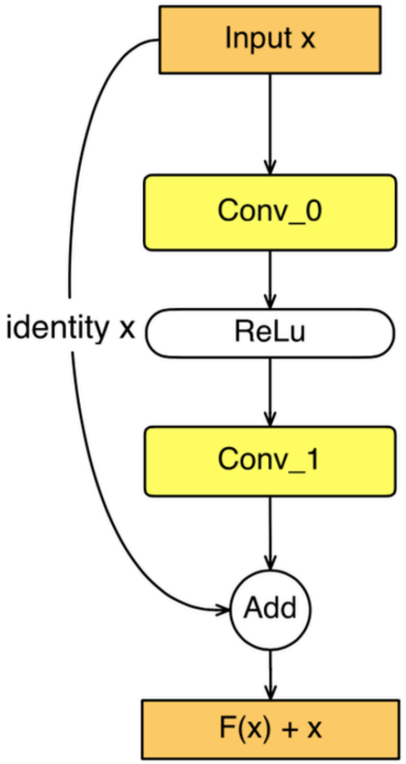
Input: p-channel tensor

Number of binary filters: m

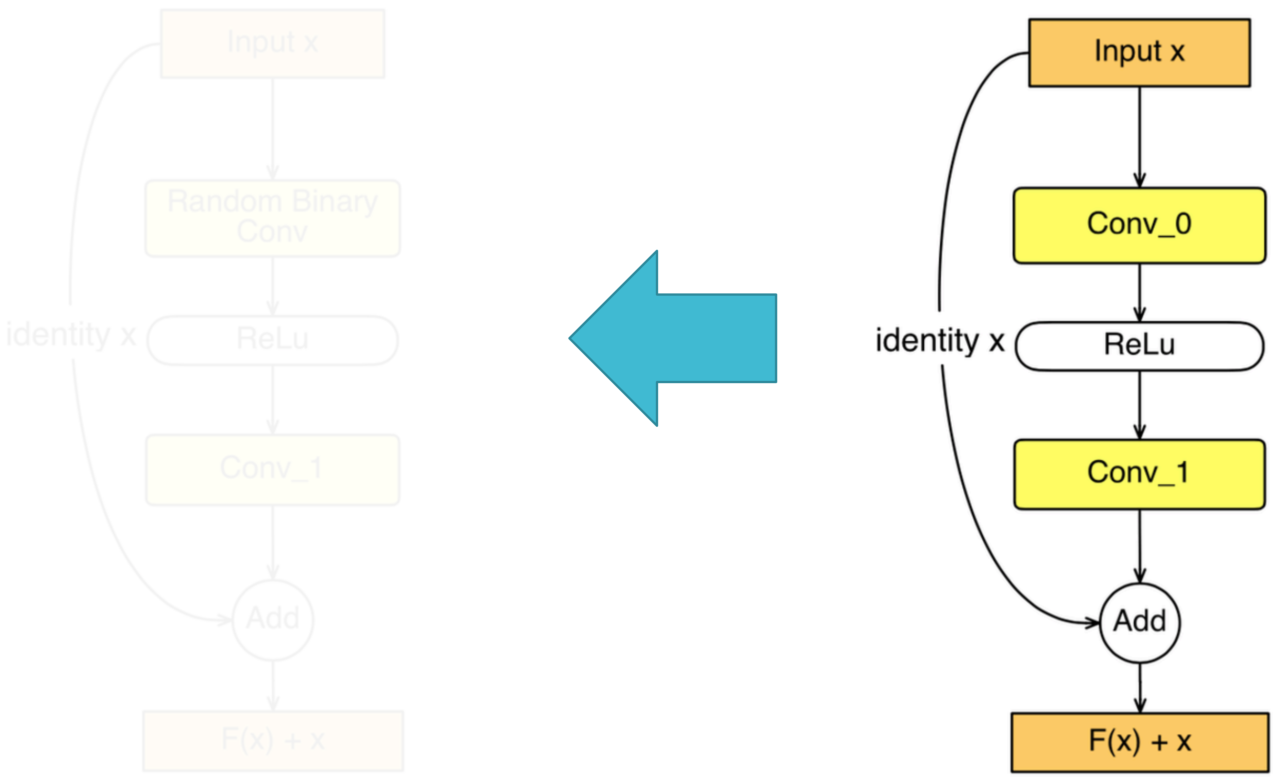
Number of Weights: p * kH * kW * q

Number of Weights: m * q

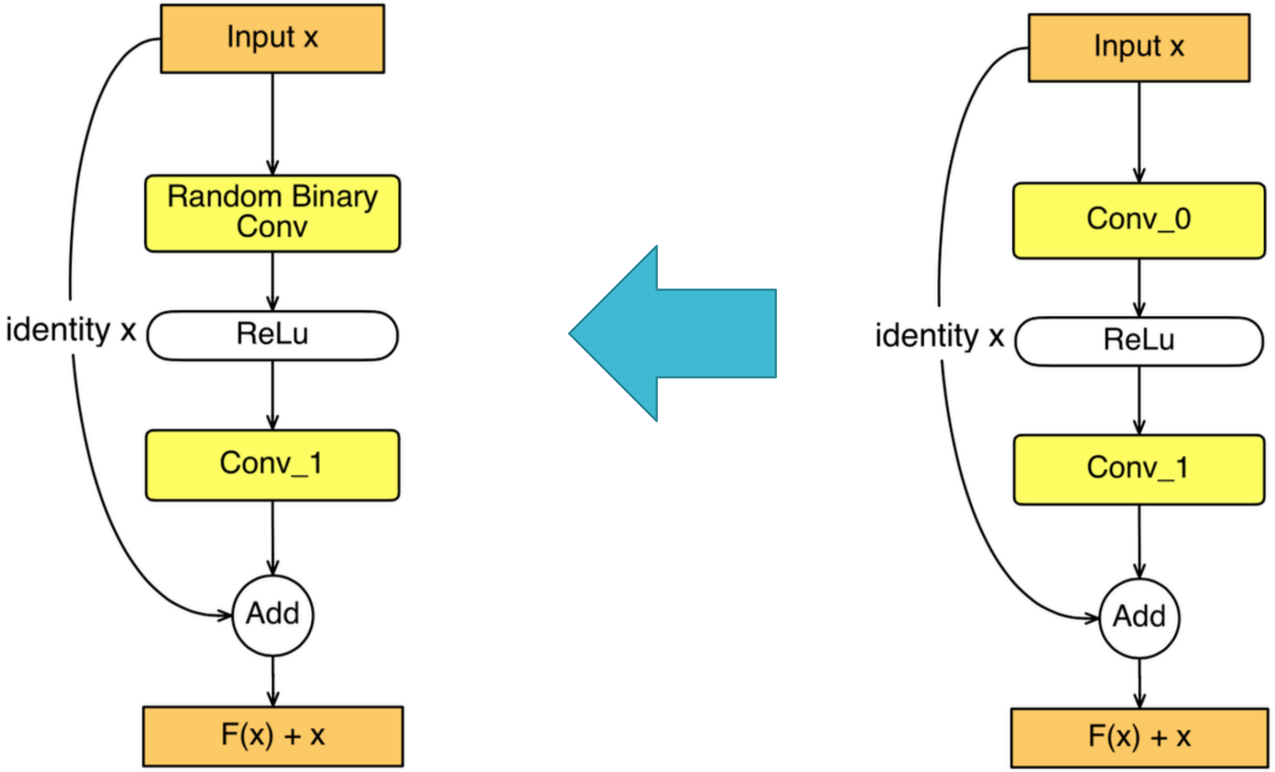
LBC Residual Block



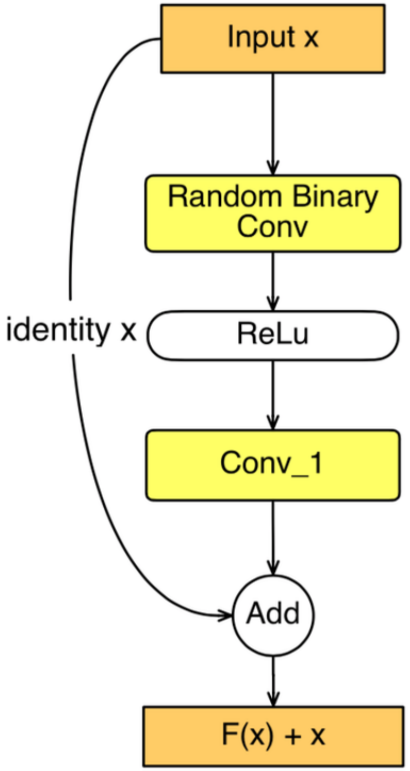
LBC Residual Block



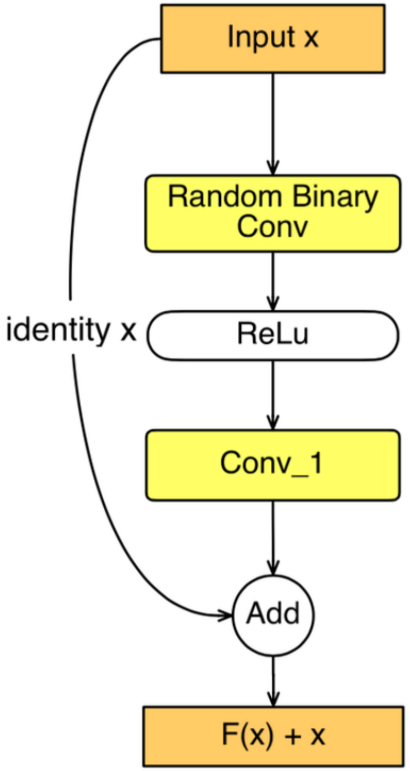
LBC Residual Block



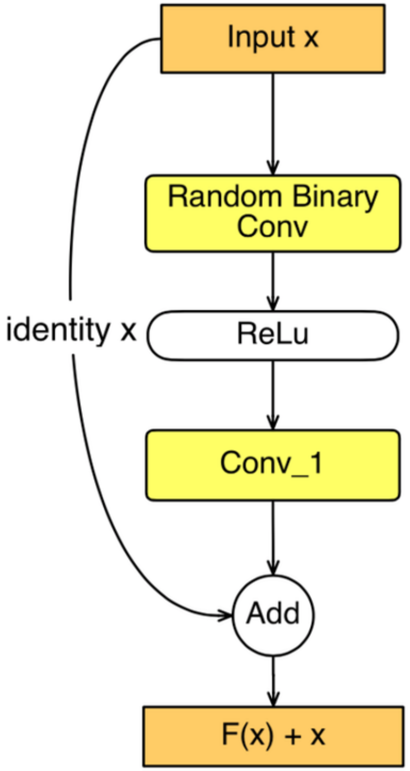
LBC Residual Block

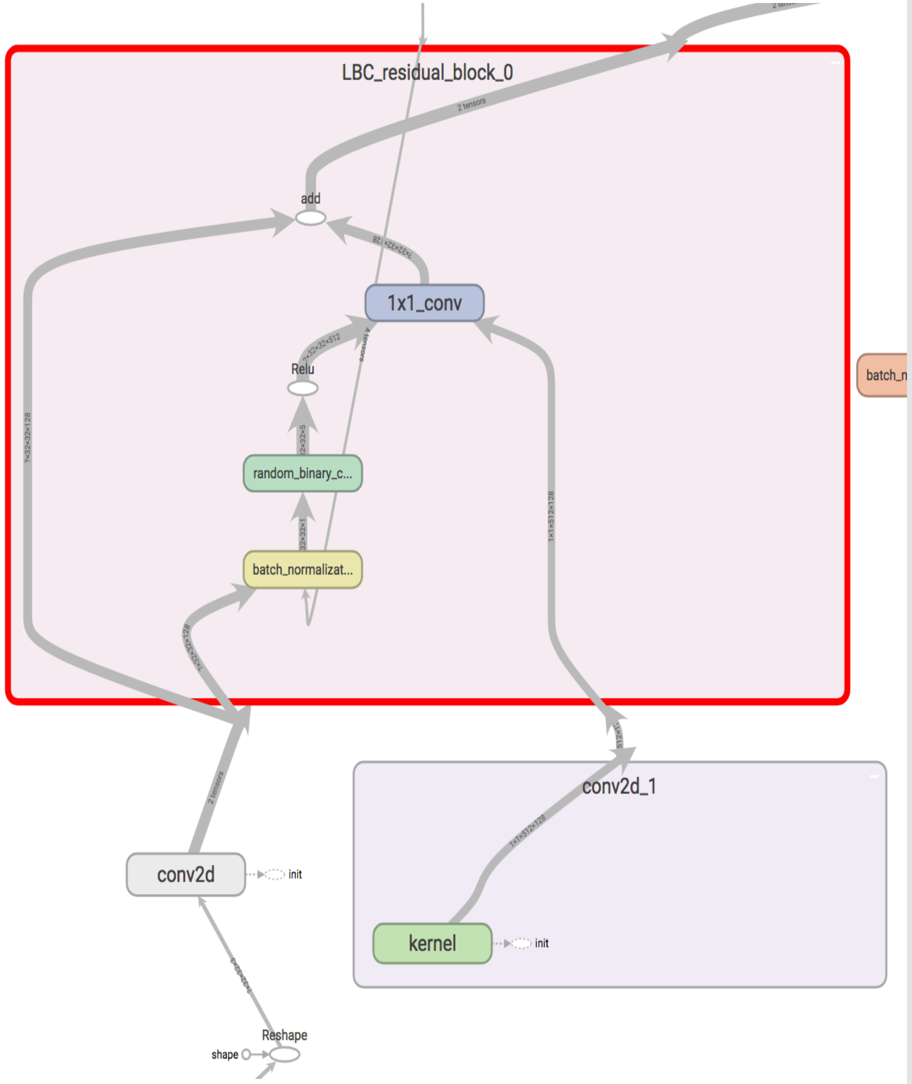


LBC Residual Block



LBC Residual Block

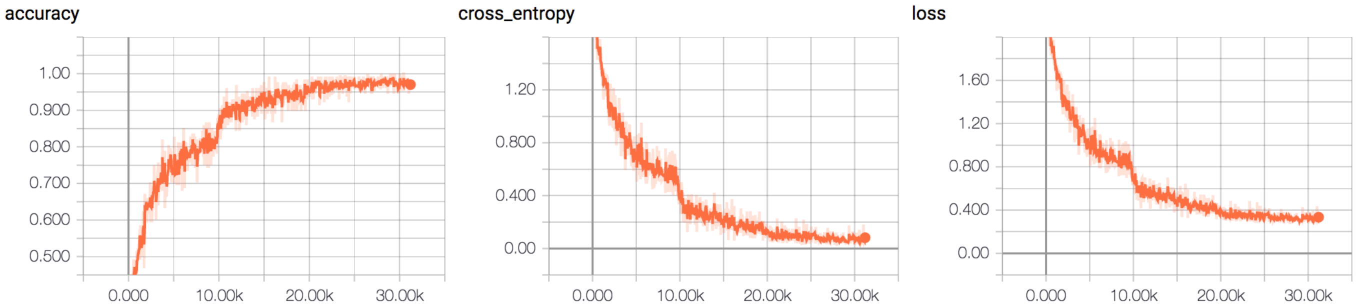




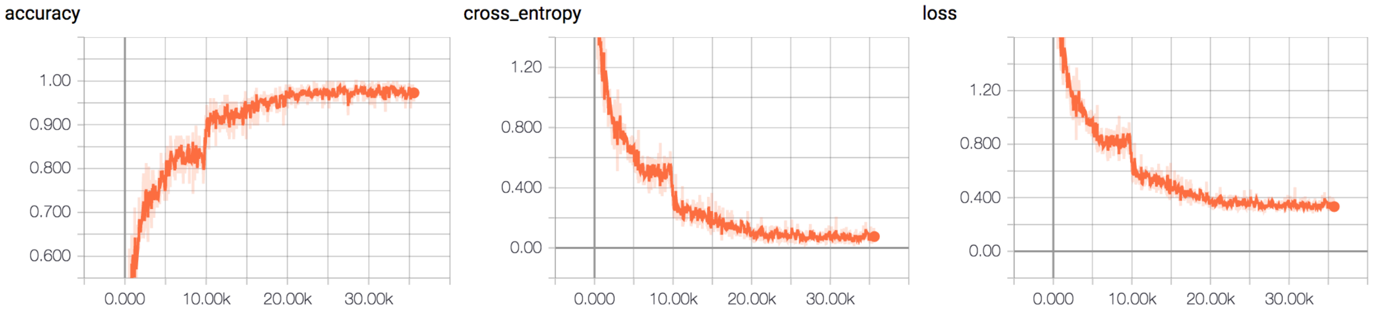
Results of LBC_ResNet on CIFAR-10

different number of binary filers

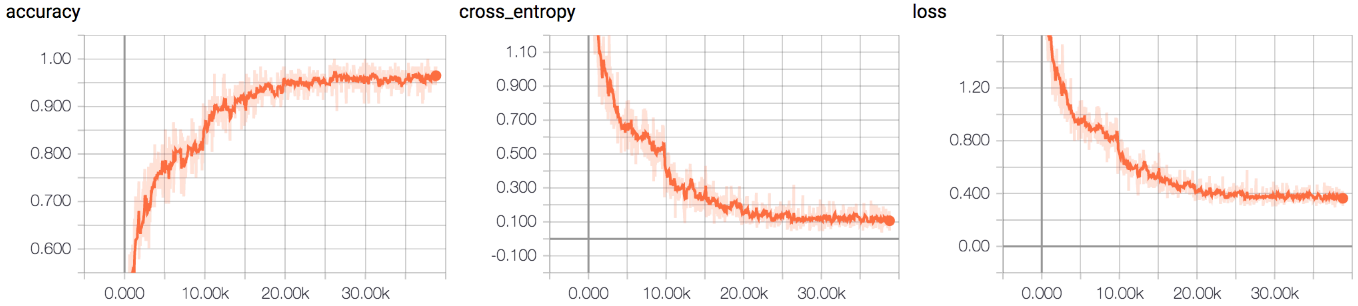
number_of_b = 512



number_of_b = 256



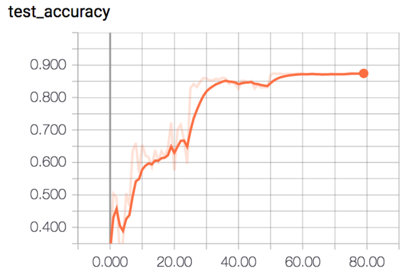
number_of_b = 128

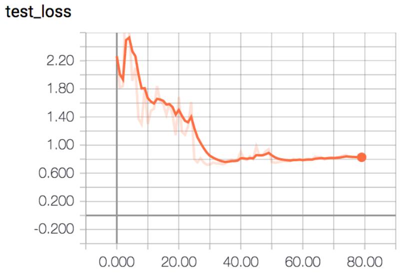


Results of LBC_ResNet on CIFAR-10

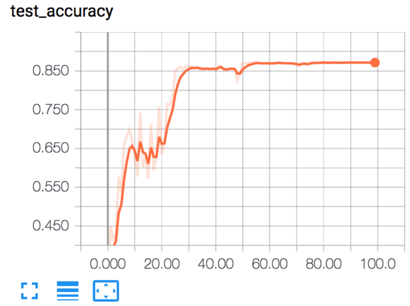
different number of binary filers

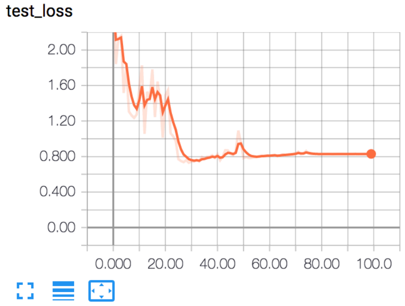
number_of_b = 256



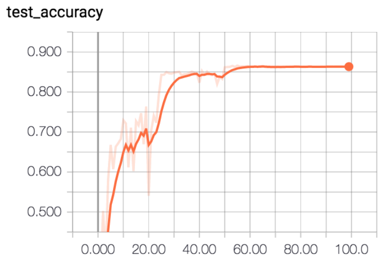


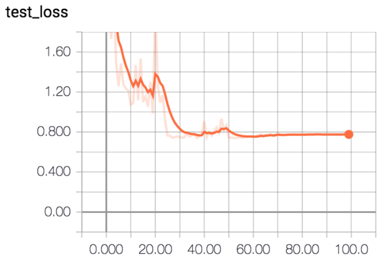
number_of_b = 256





number_of_b = 128





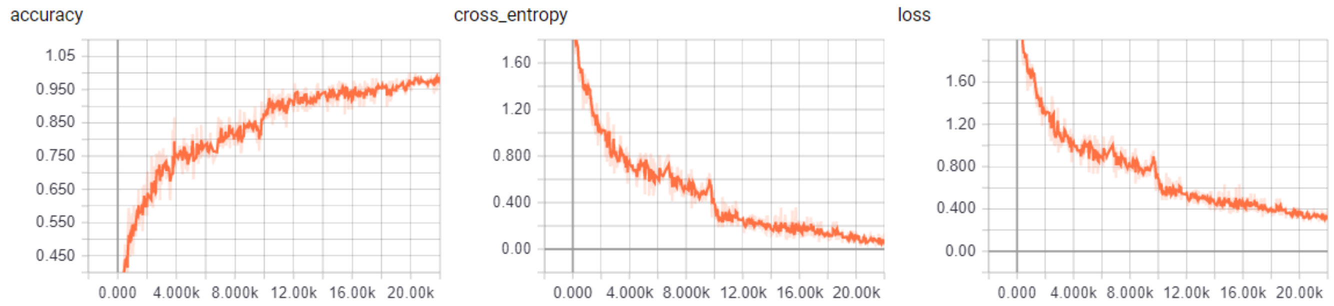
Observation:

1. Increasing the number of binary filters slightly increases the test accuracy. (~1%)
2. The training procedure was not significantly affected.
3. Both training and test accuracy/loss saturated, but didn't show overfitting.
4. In their paper, their claim to have a higher (+4% of ours) accuracy.

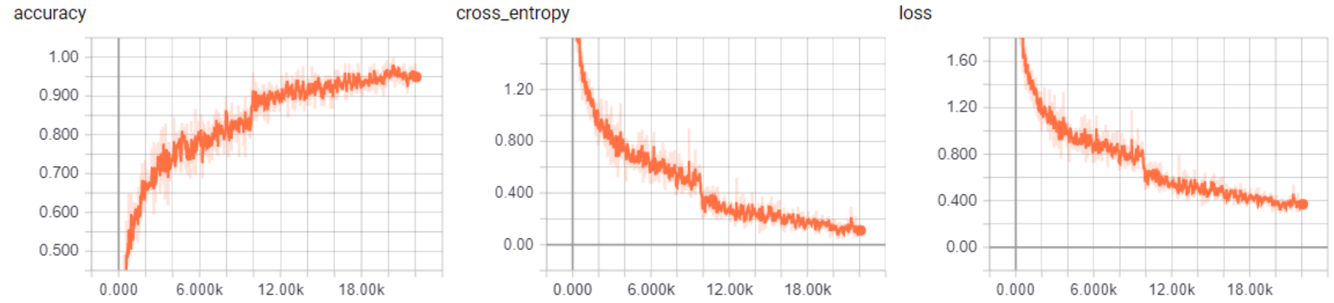
Results of LBC_ResNet on CIFAR-10

different number of depth

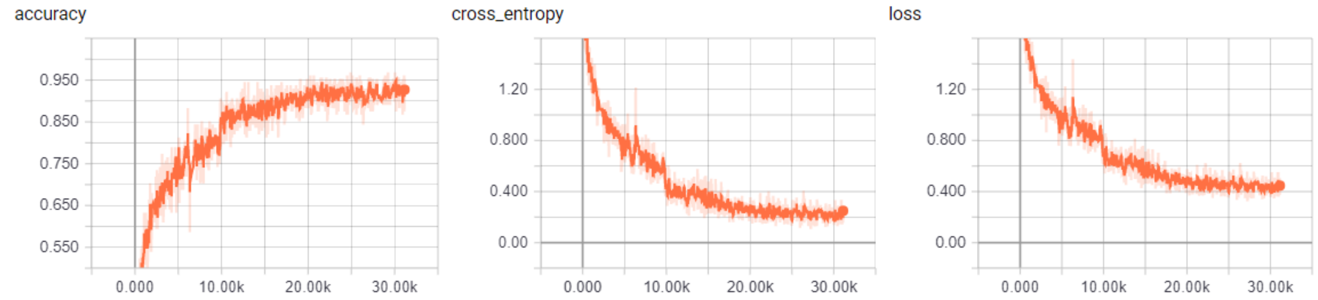
Depth = 15



Depth = 10



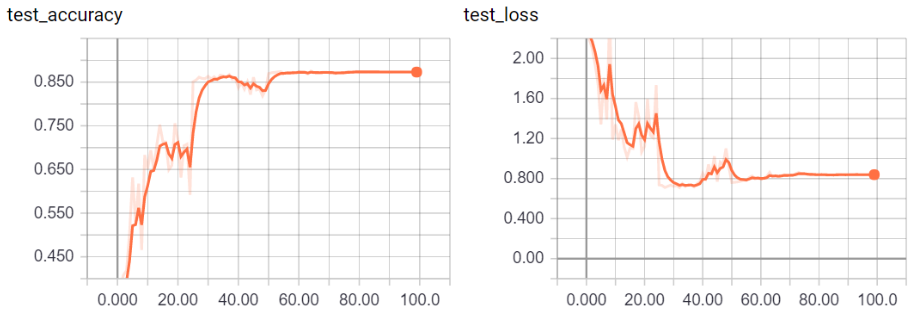
Depth = 5



Results of LBC_ResNet on CIFAR-10

different number of depth

Depth = 15



Depth = 10

Depth = 5

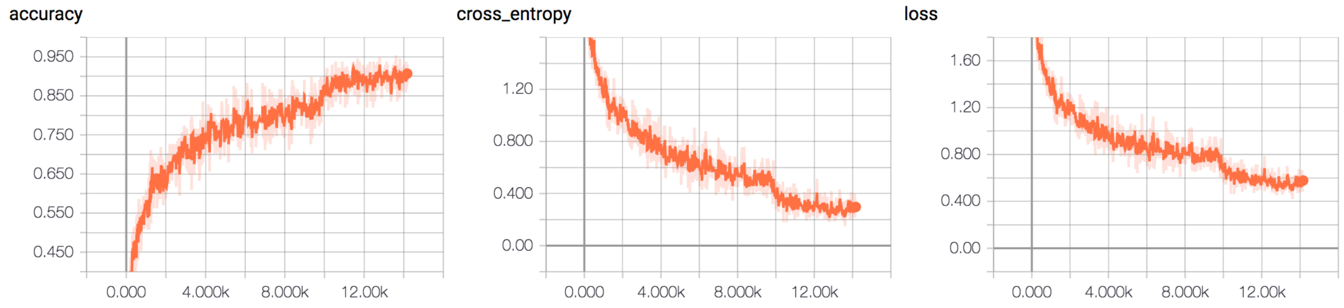
Observation:

1. The deeper the better.
2. Overfitting does not occur

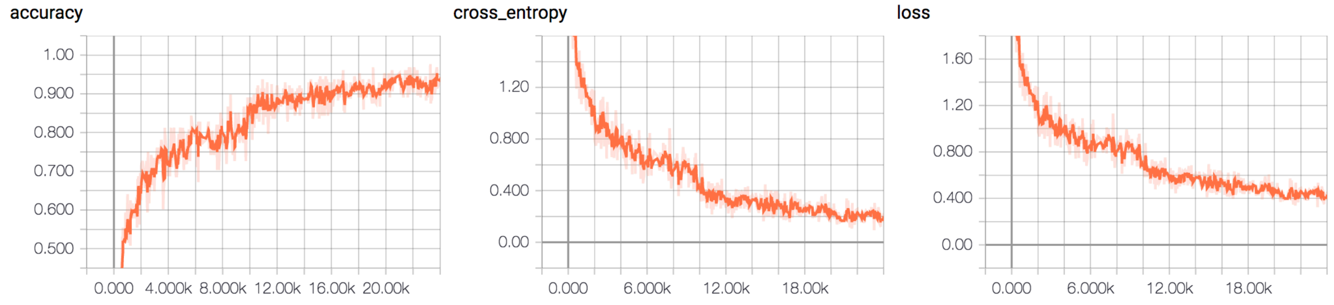
Results of LBC_ResNet on CIFAR-10

different sparsity

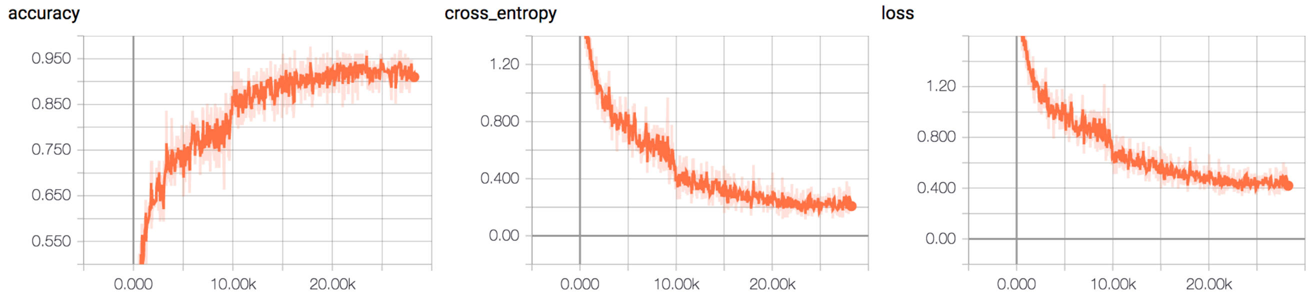
Sparsity = 0.5



Sparsity = 0.3



Sparsity = 0.1



Results of LBC_ResNet on CIFAR-10

different sparsity

Sparsity = 0.5

Sparsity = 0.3

Sparsity = 0.1

Observation:

1. Accuracy leap happens at almost the same training step.

2. No significant accuracy/loss difference with various sparsity.

Future Works

- 1. Fine tuning the learning rate, and hopefully the model can keep searching the optimum.

- 2. Try using the shared weight configuration in the paper, to see how this help with training/model size.

- 3. It still have time, test on other dataset to verify our observation and speculation

The End

Thank you!