빅바이오 part2-1,
딥러닝실습 (tensorflow),

Convolutional Neural Networks

2016-4-5

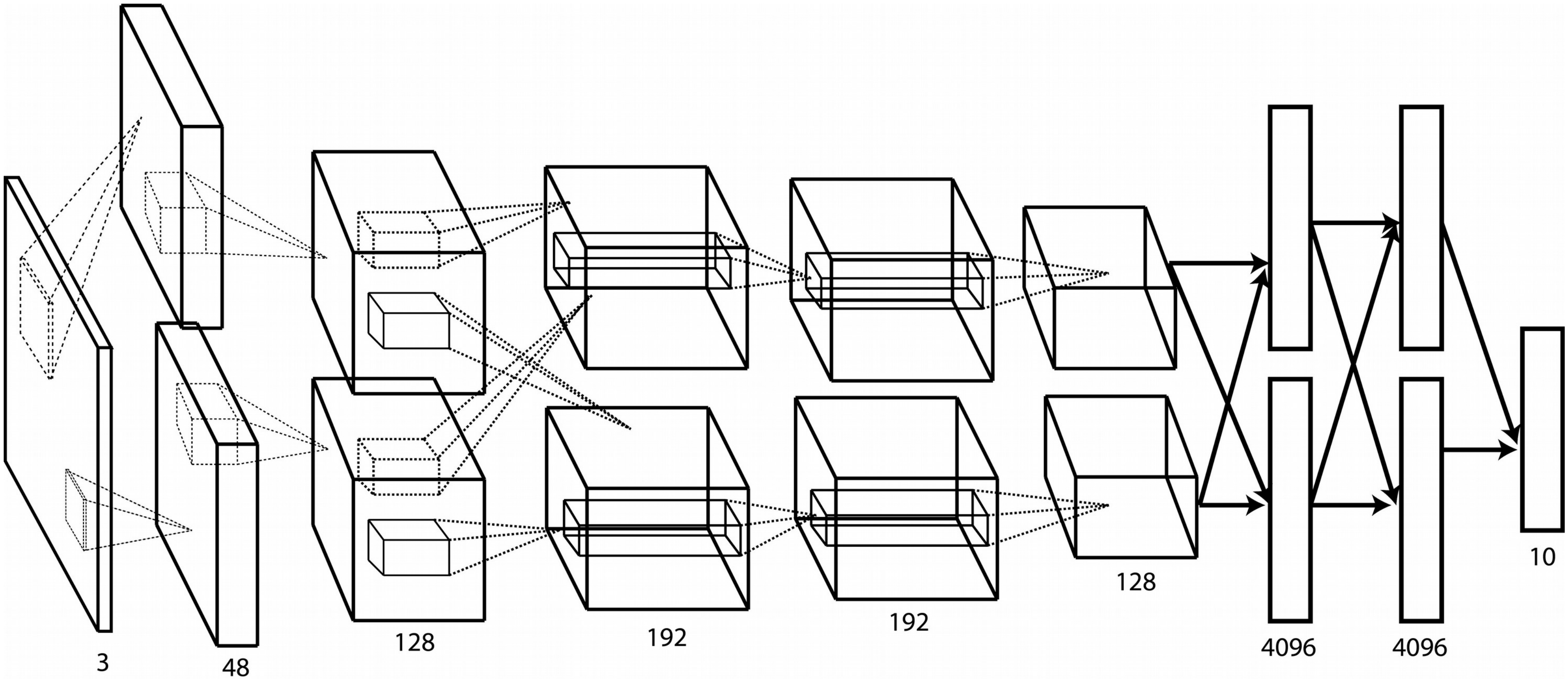
한성국

- Little bit more about Convolution Neural Networks
- Convolution Neural Networks Tutorial of Tensorflow

Goals of this Tutorials

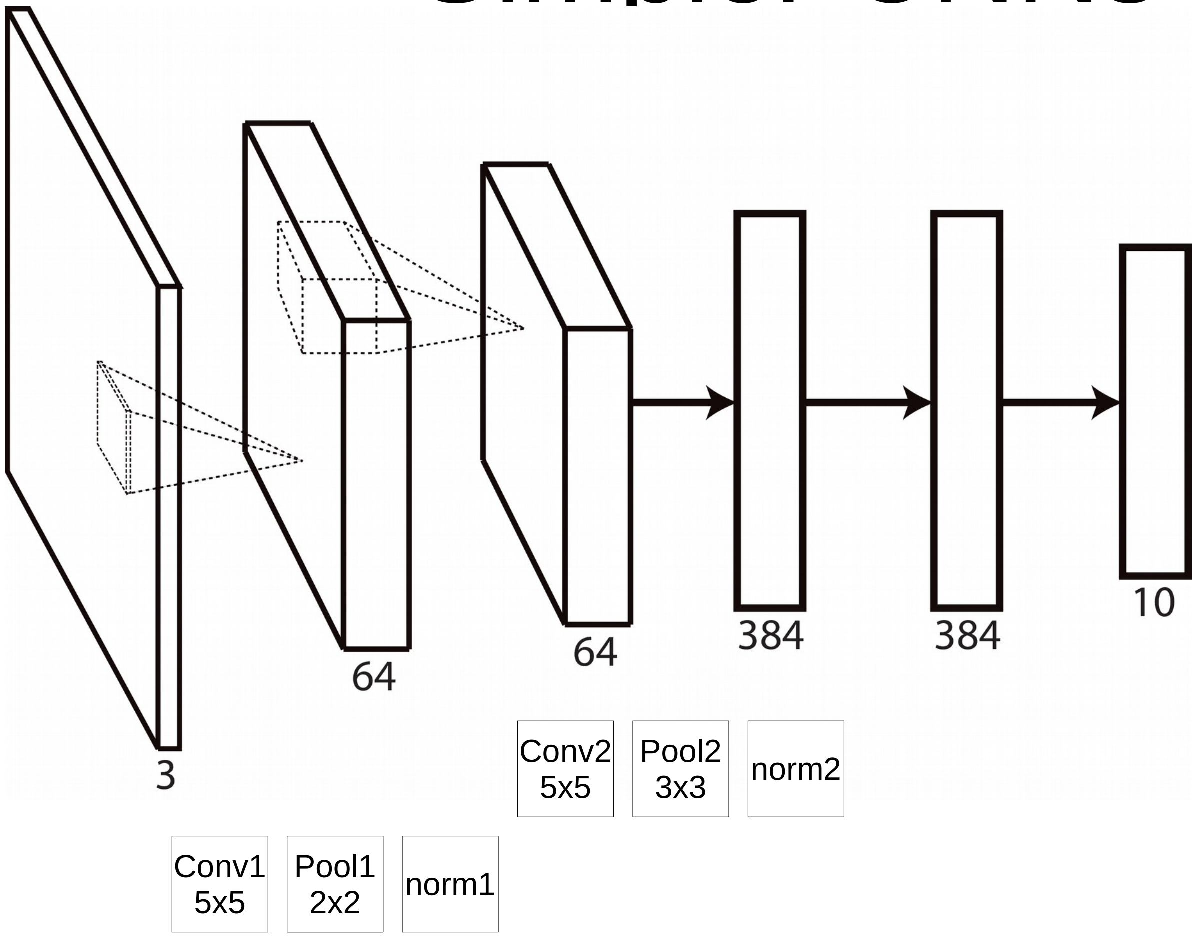
- Network architecture, training and evaluation
- Start from this template.
- Highlights
  - Simpler Convolutional Neural Networks (CNNs).
  - Convolution, relu, maxpooling, local response normalization.
  - Moving average → Enhancement in prediction performance.
  - Implementation of a learning rate schedule.
  - Prefetching queue for input data (large-scale training).
  - Multi-GPU Training (c.f. distributed tensorflow recently released).

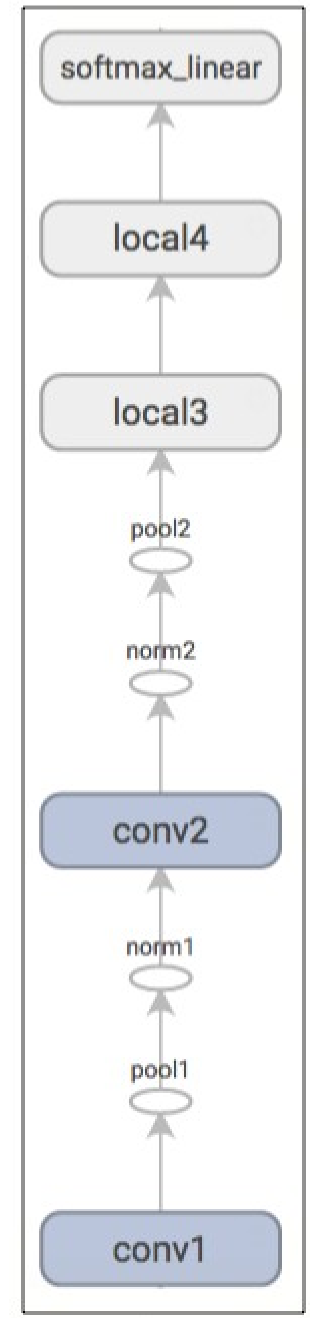
Convolutional Neural Networks



- 5 convolution layers + 2 pooling layers + 2 FC layers.
- Dropout.
- Rectified linear unit.
- Local response Normalization.
- ILSVRC 2012 result

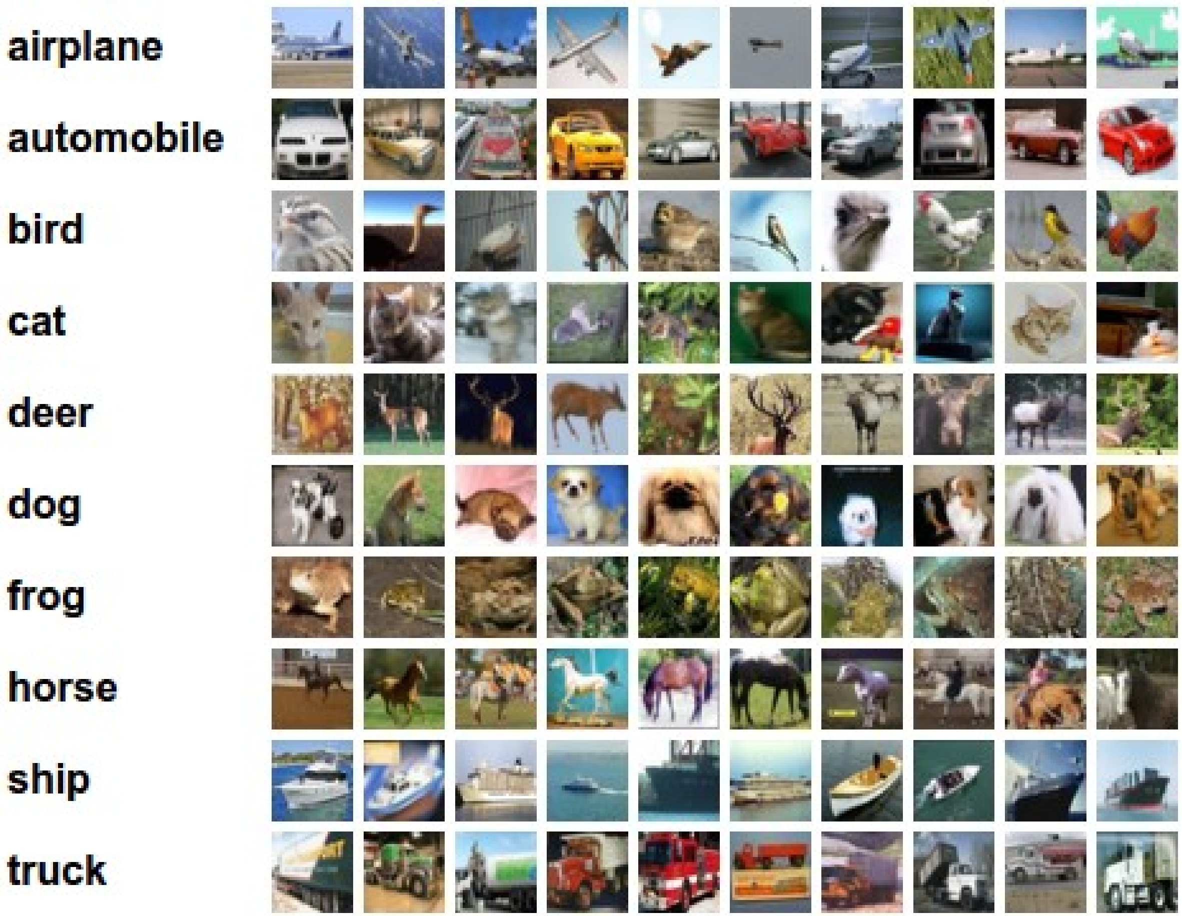
Simpler CNNs





- # of Parameters ~ 1M << # of parameters Alexnet, googlenet, VGGnet.
- Operations: 19.5M.
- 86% prediction accuracy.

CIFAR-10 data set



- 32x32pixels
- 10 classes
- Total dataset 60,000(Training :50000, Testing:10000)
- 6000/class

Code template

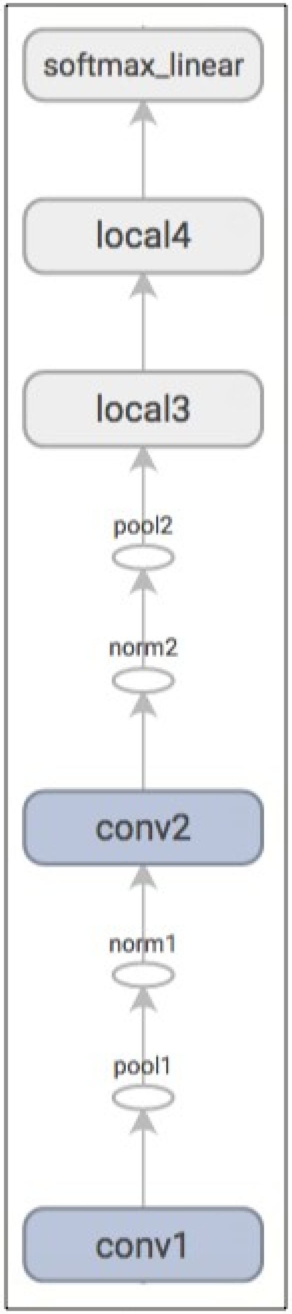
File	
cifar10_input.py	Reads the native CIFAR-10 binary file format.
cifar10.py	Builds the CIFAR-10 model.
cifa10_train.py	Trains a CIFAR-10 model on a CPU or GPU.
cifar10_multi_gpu_train.py	Trains a CIFAR-10 model on multiple GPUs.
cifar10_eval.py	Evaluates the predictive performance of a CIFAR-10 model.

Model inputs: Image preprocessing (cifar10_input.py)

- Cropping
- Approx. Whitened to zero mean and unity std.
- Randomly Flip
- Image brightness
- Image contrast.

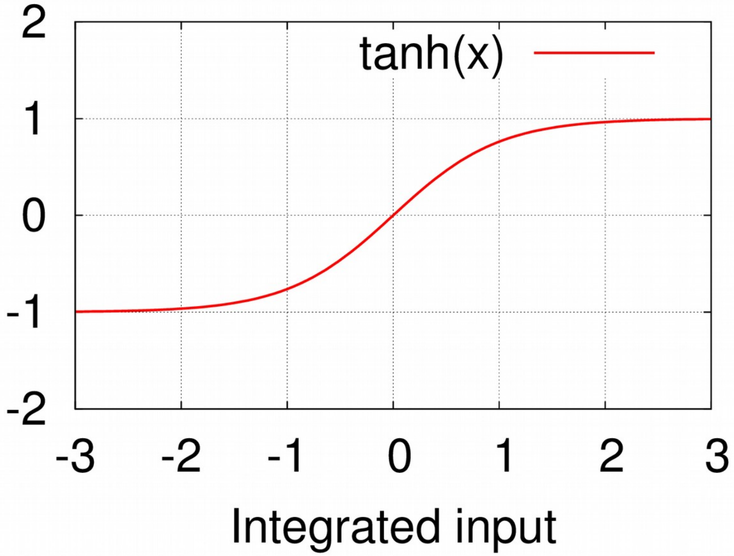
Model Prediction (cifar10.py)

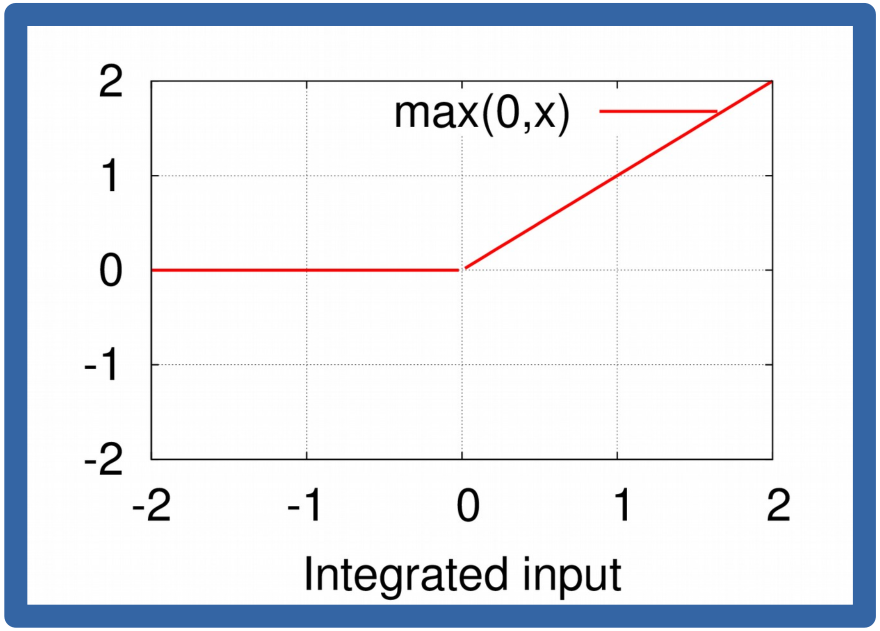
Layer Name	
conv1	Convolution, RELU.
pool1	Max pooling(cf. Average pooling).
norm1	Local response normalization.
conv2	Convolution and Retified linear activation.
norm2	Local response normalization.
pool2	Max pooling(cf. Average pooling).
loca3	Fully connected, RELU.
local4	Fully connected, RELU.
softmax_linear	Exercise1: check to normalized predictions.

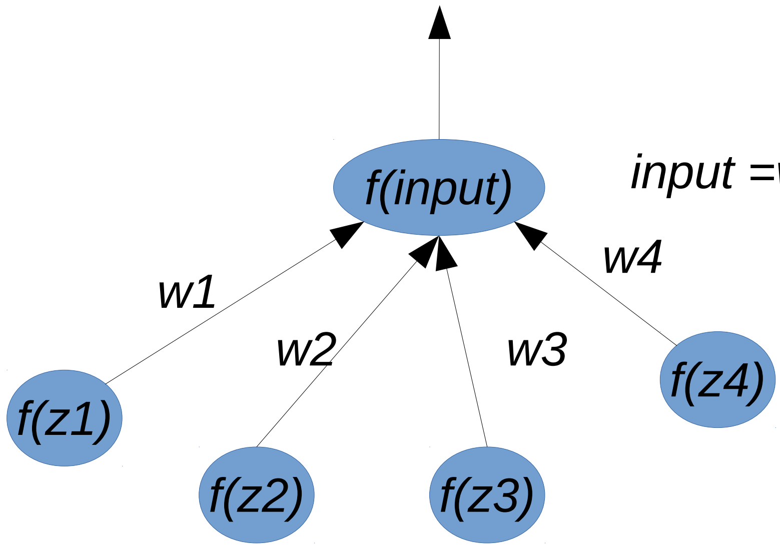


Exercise2: change the FC structure to locally connected at the local3 and local4. Dropout.

Neurons

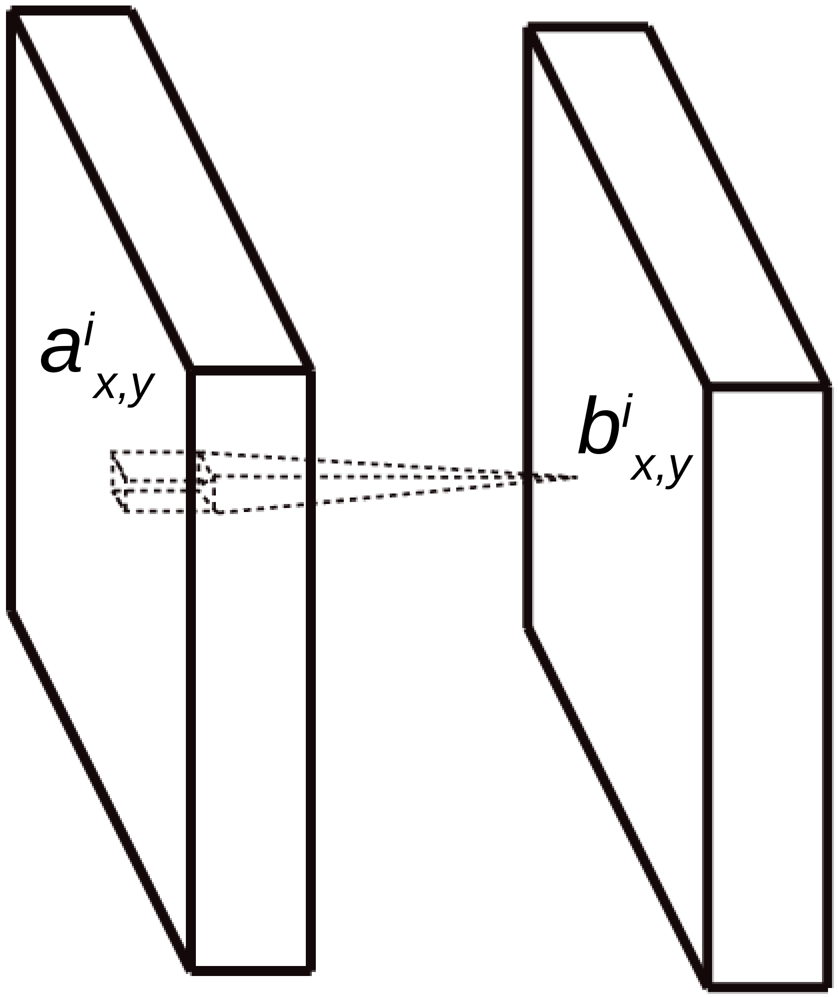






input =w1f(z1)+ w2f(z2)+w3f(z3)+w4f(z4)

Local response normalization



$$b^i_{x,y} = a^i_{x,y} \Big/ \left(k + \alpha \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} (a^j_{x,y})^2 \right)^{\beta}$$

- Neuron activations are normalized within the feature maps.
- Stable to quite large learning rate → Faster learning.

EX1 and EX2

- Learning curve.
- Prediction accuracy.

Model Training (cifar10_train.py)

- Multinomial logistic regression (softmax)
- Loss function: cross-entropy
- Weight decay: L2 regularization
- Learning rate schedule: exponentially decays
- Loss Optimizer:
  - tf.train.GradientDescentOptimizer
  - tf.train.AdagradOptimizer
  - tf.train.AdamOptimizer

Launching and Training the Model

$ python cifar10_train.py --help

```
optional arguments:
  -h, --help            show this help message and exit
  --batch_size BATCH_SIZE
                        Number of images to process in a batch.
  --data_dir DATA_DIR   Path to the CIFAR-10 data directory.
  --train_dir TRAIN_DIR
                        Directory where to write event logs and
checkpoint.
  --max_steps MAX_STEPS
                        Number of batches to run.
  --log_device_placement [LOG_DEVICE_PLACEMENT]
                        Whether to log device placement.
  --nolog_device_placement
```

Launching and Training the Model

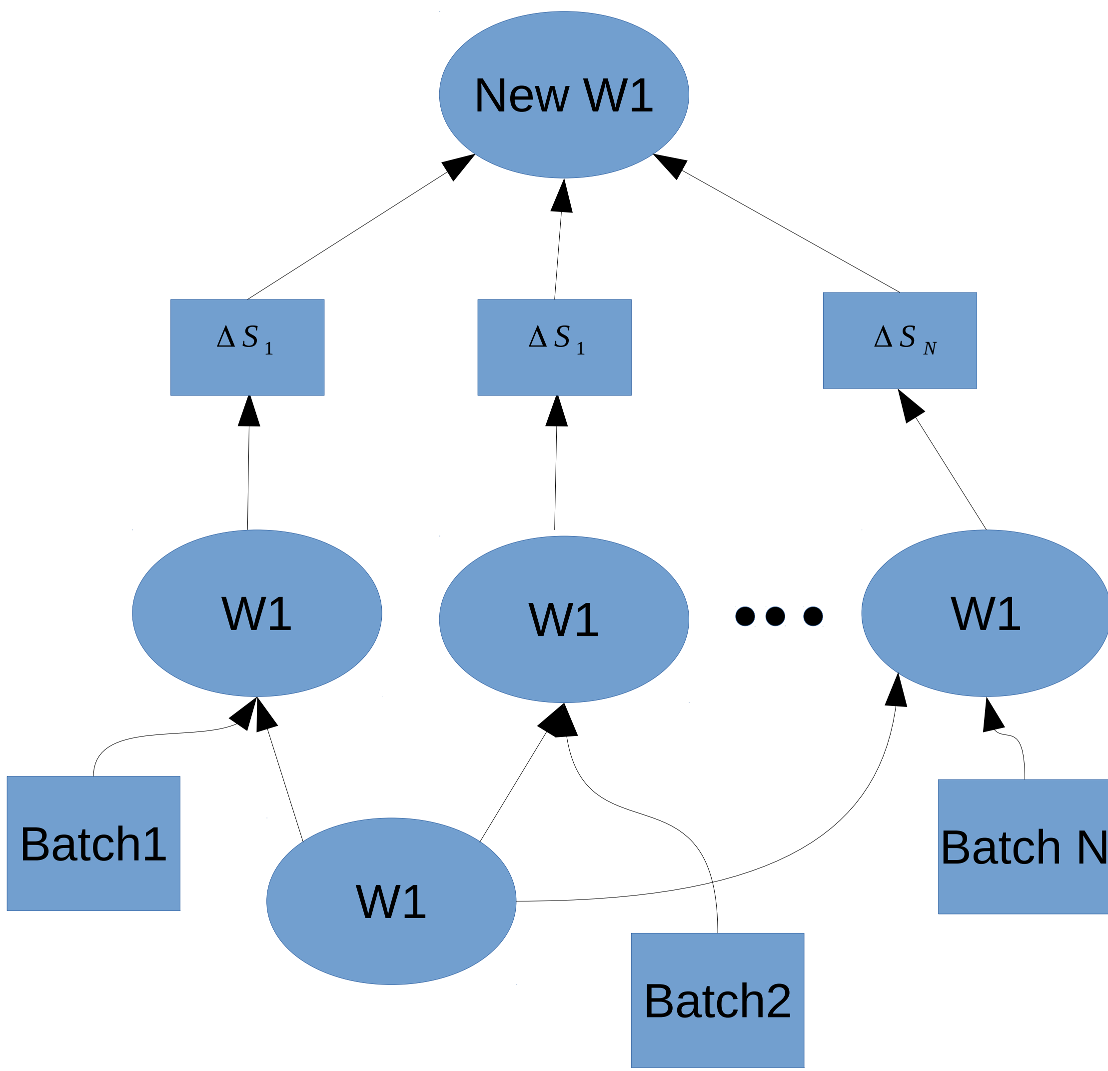
$ python cifar10_train.py

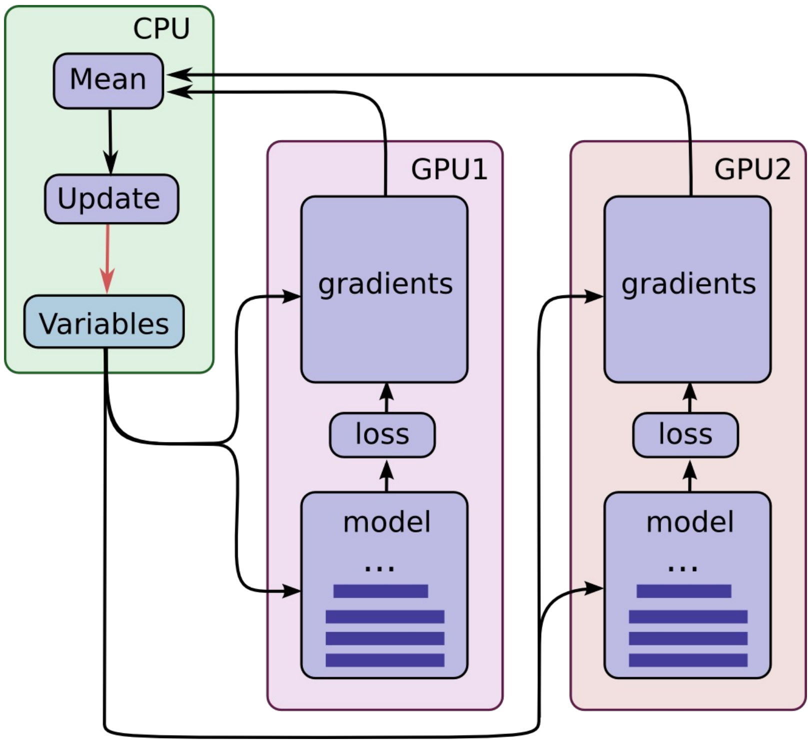
```
2016-04-02 00:03:50.972550: step 182840, loss = 0.27 (804.1 examples/sec; 0.159 sec/batch)
2016-04-02 00:03:52.596317: step 182850, loss = 0.22 (851.1 examples/sec; 0.150 sec/batch)
2016-04-02 00:03:54.144716: step 182860, loss = 0.30 (879.3 examples/sec; 0.146 sec/batch)
2016-04-02 00:03:55.798343: step 182870, loss = 0.29 (754.4 examples/sec; 0.170 sec/batch)
2016-04-02 00:03:57.381721: step 182880, loss = 0.26 (772.9 examples/sec; 0.166 sec/batch)
2016-04-02 00:03:58.935672: step 182890, loss = 0.35 (893.9 examples/sec; 0.143 sec/batch)
2016-04-02 00:04:00.487395: step 182900, loss = 0.29 (830.9 examples/sec; 0.154 sec/batch)
2016-04-02 00:04:02.218012: step 182910, loss= 0.29 (922.3 examples/sec; 0.139 sec/batch)
2016-04-02 00:04:03.771728: step 182920, loss = 0.34 (799.0 examples/sec; 0.160 sec/batch)
2016-04-02 00:04:05.343366: step 182930, loss = 0.27 (797.8 examples/sec; 0.160 sec/batch)
```

(GTX980Ti)

Multiple GPU cards Implemetation

- Model replica





$python cifar10_multigpu_train.py –num_gpus n

1 gtx980ti+1gtx960

```
sghan@aturing: ~/tensorflo
sghan@aturing: ~/tensorflow-master/tensorflow/models/image/cifar10 116x
untime/gpu/gpu_bfc_allocator.cc:83] GPU 0 memory begin
o 0xcde320e67
: step 0, loss = 4.68 (13.2 examples/sec; 9.681 sec/bat

untime/gpu/pool_allocator.cc:244] PoolAllocator: After
ount=612093 evicted_count=1000 eviction_rate=0.00163374
n rate=0.00182484
: step 10, loss = 4.66 (909.0 examples/sec; 0.141 sec/b

: step 20, loss = 4.64 (1023.9 examples/sec; 0.125 sec/batch)
: step 30, loss = 4.63 (928.2 examples/sec; 0.138 sec/batch)
: step 40, loss = 4.61 (845.4 examples/sec; 0.151 sec/batch)
: step 50, loss = 4.59 (949.6 examples/sec; 0.135 sec/batch)
: step 60, loss = 4.57 (953.3 examples/sec; 0.134 sec/batch)
: step 70, loss = 4.55 (828.5 examples/sec; 0.154 sec/batch)
: step 80, loss = 4.53 (818.1 examples/sec; 0.156 sec/batch)
: step 90, loss = 4.51 (878.3 examples/sec; 0.146 sec/batch)
: step 100, loss = 4.50 (801.3 examples/sec; 0.160 sec/batch)
: step 110, loss = 4.48 (816.1 examples/sec; 0.157 sec/batch)
: step 120, loss = 4.46 (848.6 examples/sec; 0.151 sec/batch)
: step 130, loss = 4.44 (839.9 examples/sec; 0.152 sec/batch)
: step 140, loss = 4.43 (877.6 examples/sec; 0.146 sec/batch)
: step 150, loss = 4.41 (940.3 examples/sec; 0.136 sec/batch)
: step 160, loss = 4.39 (753.7 examples/sec; 0.170 sec/batch)
: step 170, loss = 4.38 (821.0 examples/sec; 0.156 sec/batch)
: step 180, loss = 4.36 (914.7 examples/sec; 0.140 sec/batch)
: step 190, loss = 4.34 (877.5 examples/sec; 0.146 sec/batch)
: step 200, loss = 4.33 (896.4 examples/sec; 0.143 sec/batch)
: step 210, loss = 4.31 (876.2 examples/sec; 0.146 sec/batch)
: step 220, loss = 4.29 (920.7 examples/sec; 0.139 sec/batch)
: step 230, loss = 4.28 (883.5 examples/sec; 0.145 sec/batch)
: step 240, loss = 4.26 (946.6 examples/sec; 0.135 sec/batch)
: step 250, loss = 4.25 (862.0 examples/sec; 0.148 sec/batch)
: step 260, loss = 4.23 (948.4 examples/sec; 0.135 sec/batch)
: step 270, loss = 4.22 (881.7 examples/sec; 0.145 sec/batch)
```

1 gtx980ti

```
sghan@aturing: ~/ten
@aturing: ~/tensorflow-master/tensorflow/models/image/cifar10 102x52
: step 0, loss = 4.68 (11.0 examples/sec; 11.595 sec/batch)
: step 10, loss = 4.66 (925.5 examples/sec; 0.138 sec/batch
: step 20, loss = 4.64 (872.3 examples/sec; 0.147 sec/batch
: step 30, loss = 4.61 (789.1 examples/sec; 0.162 sec/batch
: step 40, loss = 4.60 (751.7 examples/sec; 0.170 sec/batch
: step 50, loss = 4.59 (792.2 examples/sec; 0.162 sec/batch
: step 60, loss = 4.57 (957.6 examples/sec; 0.134 sec/batch
: step 70, loss = 4.55 (1014.2 examples/sec; 0.126 sec/batc
: step 80, loss = 4.53 (1022.2 examples/sec; 0.125 sec/batc
: step 90, loss = 4.51 (1027.0 examples/sec; 0.125 sec/batc
: step 100, loss = 4.48 (1013.8 examples/sec; 0.126 sec/bat
: step 110, loss = 4.48 (777.9 examples/sec; 0.165 sec/batc
: step 120, loss = 4.46 (857.1 examples/sec; 0.149 sec/batc
: step 130, loss = 4.44 (824.8 examples/sec; 0.155 sec/batc
: step 140, loss = 4.42 (798.4 examples/sec; 0.160 sec/batc
: step 150, loss = 4.41 (988.2 examples/sec; 0.130 sec/batc
: step 160, loss = 4.39 (744.8 examples/sec; 0.172 sec/batc
: step 170, loss = 4.37 (751.4 examples/sec; 0.170 sec/batc
: step 180, loss = 4.36 (763.3 examples/sec; 0.168 sec/batc
: step 190, loss = 4.34 (892.6 examples/sec; 0.143 sec/batc
: step 200, loss = 4.32 (712.3 examples/sec; 0.180 sec/batc
: step 210, loss = 4.31 (710.7 examples/sec; 0.180 sec/batc
: step 220, loss = 4.30 (824.7 examples/sec; 0.155 sec/batc
: step 230, loss = 4.27 (691.6 examples/sec; 0.185 sec/batc
: step 240, loss = 4.27 (723.0 examples/sec; 0.177 sec/batc
: step 250, loss = 4.25 (728.8 examples/sec; 0.176 sec/batc
: step 260, loss = 4.23 (743.4 examples/sec; 0.172 sec/batc
: step 270, loss = 4.22 (772.6 examples/sec; 0.166 sec/batc
: step 280, loss = 4.20 (677.5 examples/sec; 0.189 sec/batc
: step 290, loss = 4.18 (694.3 examples/sec; 0.184 sec/batc
```

References

- Tensorflow.org
- Vincent Dumoulin and Francesco Visin,“A guide to convolution arithmetic for deep learning”, arXiv:1603.07285 [stat.ML]
- Mattew D. Zeiler and Rob Fergus, “Visualizing and Understanding Convolutions Networks”, arXiv:1311.2901 [cs.CV]