1. Appendix

1.1. Hyper-paramater Space

To make our entry matrix memory management simple, we fixed batch size as 100 and hence it was not included in search space.

1.1.1. LeNet

Table 1. Hyper-parameter search space for LeNet network

Hyper-parameter	Scale	Min	Max
Learning rate	log	5e-5	5
Conv1 L2 penalty	log	5e-5	5
Conv2 L2 penalty	log	5e-5	5
Momentum	log	0.001	0.99

We have used caffe's implementation of LeNet network [5] and the details of our search space can be found in table 1. Once a hyper-parameter set is selected, we train a new Lenet (from scratch) using that set for 10000 iterations to report final validation error.

1.1.2. AlexNet

Table 2. Hyper-parameter search space used for CQ

Hyper-parameter	Scale	Min	Max
Learning rate	log	5e-5	5
Conv1 L2 penalty	log	5e-5	5
Conv2 L2 penalty	log	5e-5	5
Conv3 L2 penalty	log	5e-5	5
FC1 L2 penalty	log	5e-5	5
FC2 L2 penalty	log	5e-5	5

Table 3. Hyper-parameter search space for AlexLRN

Hyper-parameter	Scale	Min	Max
Learning rate	log	5e-5	5
Conv1 L2 penalty	log	5e-5	5
Conv2 L2 penalty	log	5e-5	5
Conv3 L2 penalty	log	5e-5	5
FC1 L2 penalty	log	5e-5	5
LRN:			
Scale	log	5e-6	5
Power	linear	0.01	3

[5]The model specification is available at https://github.com/BVLC/caffe/blob/master/examples/mnist/lenet_train_test.prototxt

Table 4. Hyper-parameter search space used for AlexBN network

Hyper-parameter	Scale	Min	Max
Learning rate	log	5e-5	5
Momentum	log	5e-5	5
Conv1 L2 penalty	log	5e-5	5
Conv2 L2 penalty	log	5e-5	5
Conv3 L2 penalty	log	5e-5	5
FC1 L2 penalty	log	5e-5	5

The details of our search space for CQ network [6], AlexBN [7] and AlexLRN [8] can be found in table 2, 4 and 3 respectively. Once a hyper-parameter set is selected, we train a new CQ, AlexLRN or AlexBN (from scratch) using that selected configuration for 4000, 60000 and 60000 iterations respectively to report final validation error. We have replaced sigmoid with ReLu for AlexBN because we observed that ReLu helps in converging quickly than sigmoid activation function and also found that applying batch normalization after ReLu and pooling layers helps to get better accuracy.

1.1.3. MLP

The search space for MLP can be found in Table 5. It has only one hidden, input and output layer with an optional dropout layer which was also used in meprop. Once a hyper-parameter set is selected, we train a new MLP (from scratch) using that selected configuration for 20 epochs.

Table 5. Hyper-parameter search space used in hyperband for MLP

Hyper-parameter	Scale	Min	Max
Hidden neurons	linear	100	8192
Dropout	linear	0	0.999
top-K	linear	0	100

1.2. Proof

Total execution time without our technique is given by

$$N(t_f + t_b)$$

Time spent by CNN when our technique is applied consists of 3 parts:

[6]The model specification is available at https://github.com/BVLC/caffe/blob/master/examples/cifar10/cifar10_quick_train_test.prototxt

[7]The model specification is available at https://github.com/BVLC/caffe/blob/master/examples/cifar10/cifar10_full_sigmoid_train_test_bn.prototxt

[8]The model specification is available at https://github.com/BVLC/caffe/blob/master/examples/cifar10/cifar10_full_train_test.prototxt

1. time spent in pre-raining interval is

$$p_t(t_f + t_b)$$

2. time spent during calculating E is

$$\frac{N - p_t}{Z + 1}(t_f + t_b + t_e)$$

3. where as the new execution time of CNN after applying STAN is

$$(N - p_t)\frac{Z}{Z + 1}(t_b + (t_f - t_i))$$

So the speedup is given by

$$\frac{N(t_f + t_b)}{p_t(t_f + t_b) + \frac{N-p_t}{Z+1}(t_f + t_b + t_e) + (N - p_t)\frac{Z}{Z+1}(t_b + (t_f - t_i))}$$

$$\frac{N(t_f + t_b)}{N(t_f + t_b) + \frac{N-p_t}{Z+1}t_e - (N - p_t)\frac{Z}{Z+1}t_i}$$

$$\frac{N(t_f + t_b)}{N(t_f + t_b) - \frac{N-p_t}{Z+1}(Zt_i - t_e)}$$