Hardware Acceleration of Neural Networks via Pre-defined Sparsity

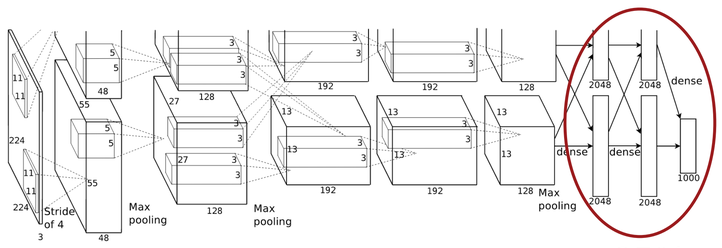
USC University of Southern California

Sourya Dey, Keith Chugg, Peter Beerel

USC Viterbi School of Engineering

(1) Introduction

Neural networks (NNs) are critical drivers of new machine learning technologies. Modern NNs are gigantic with millions of parameters, leading to large memory and computational complexity during training. This is particularly true for the fully connected (FC) classification layers (CLs) occurring near the outputs of a typical NN. E.g. >95% of connections in AlexNet are in the FC layers.

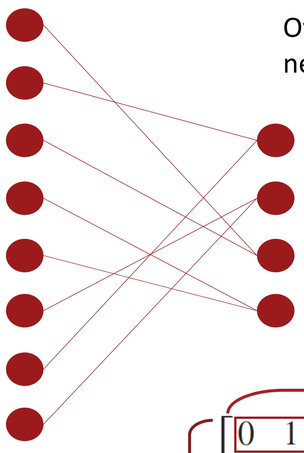


Existing methods to reduce parameters:

- Dropout
- Hashing
- Pruning
- Regularization

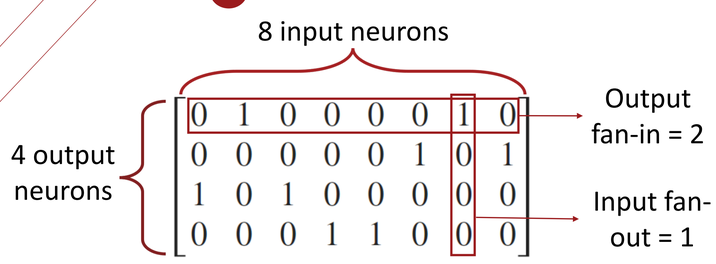
Additional training processing required
Need to store complete network at some time

(2) Our Approach: Pre-defined Sparsity



Overall density of this network = 8/32 = 25%

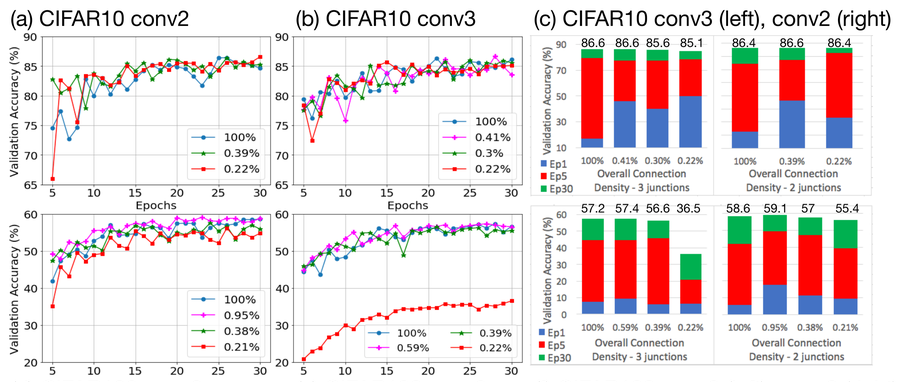
Plan sparse CL connection pattern from beginning by designing pseudo-random adjacency matrix

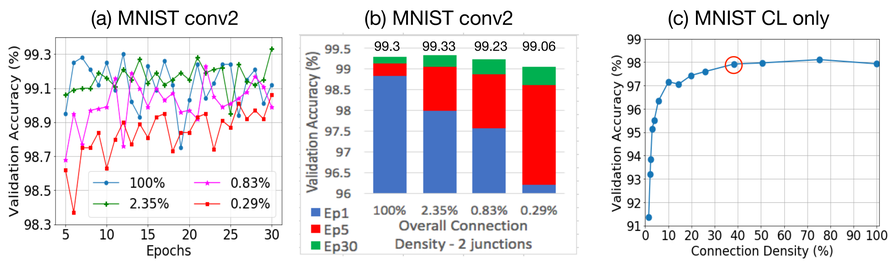


(3) Merits and Analysis of Pre-defined Sparsity

Classification layer (CL) connection density of a network can be reduced to low values using pre-defined sparse connection patterns, without any performance degradation. This leads to tremendous parameter (memory) savings, and in several cases, considerable operational (computational) savings as well. For example, the following networks achieve the same accuracy as their fully connected CL counterparts [3]:

- CNN with 2 CLs: 0.22% density (CIFAR10)
- CNN with 3 CLs: 0.59% density (CIFAR100)
- CNN with 2 CLs: 2.35% density (MNIST)
- Only 3 CLs (no conv layers): 38.3% density (MNIST)





Net	Junction fan-outs	CL Junction densities (%)	Overall CL density (%)
CIFAR10 conv2	1, 1	0.2, 6.3	**0.22**
	1, 8	0.2, 50	0.39
CIFAR100 conv2	1, 1	0.2, 0.8	0.21
	1, 8	0.2, 6.3	0.38
	1, 32	0.2, 25	**0.95**
CIFAR10 conv3	1, 1, 1	0.2, 0.4, 6.3	0.22
	1, 1, 8	0.2, 0.4, 50	0.3
	1, 2, 16	0.2, 0.8, 100	**0.41**
CIFAR100 conv3	1, 1, 1	0.2, 0.4, 0.8	0.22
	1, 1, 16	0.2, 0.4, 13	0.39
	1, 2, 32	0.2, 0.8, 25	**0.59**
MNIST conv2	1, 5	0.1, 50	0.29
	4, 10	0.5, 100	0.83
	16, 10	2, 100	**2.35**
MNIST CL only	42, 10	38, 100	**38.3**
	2, 10	1.8, 100	3.02

Less relative importance of CLs ➔ More sparsity possible

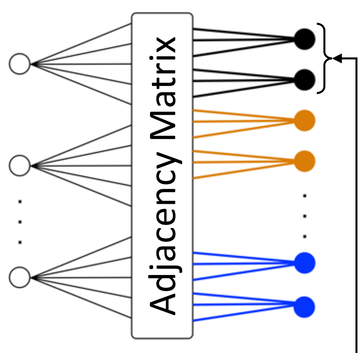
Effects of pre-defined sparsity on accuracy of networks training on CIFAR (left top), and MNIST (left middle). Key: 'conv2', 'conv3' = 'Convolutional network followed by 2 or 3 CLs'. 'CL only' = 'Only 3 CLs.
Tables showing some configurations we tried and their overall densities (right top), and savings achieved due to pre-defined sparsity (bottom).

Net	CLs / Total Layers	Conv Params (M)	Conv Ops (B)	FC CL Params (M)	Sparse CL Params (M)	Overall Param % Reduction	Overall Op % Reduction
MNIST CL only	2/2	0	0	0.07	0.03	61.7	**61.7**
MNIST conv2	2/6	0.05	0.1	2.47	0.06	**95.63**	18.29
CIFAR10 conv2	2/17	1.15	0.15	2.11	0.005	64.63	1.35
CIFAR100 conv2	2/17	1.15	0.15	2.16	0.03	64.76	1.38
CIFAR10 conv3	3/18	1.15	0.15	2.23	0.009	65.83	1.43
CIFAR100 conv3	3/18	1.15	0.15	2.26	0.013	65.99	1.45

(4) Memory Organization

Define degree of parallelism z for each junction to trade-off hardware complexity with processing time ➔ Flexible FPGA-suited architecture [1-2]

- Any parameter set *p* (weights, activations, deltas etc) is arranged in a bank of *z* memories.
- Adjacency matrix distributes weights across a junction in a pre-defined pseudo-random order.
- A bank of memories can be read in natural order (same row for all), or permuted order (any row).
- Must read only 1 entry from each memory in 1 clock cycle, otherwise **clash** => Processing stall.



Mem 0	Mem 1				Mem z-1
p_0	p_1	p_2			p_{z-1}
p_z					
p_{2z}					
p_{3z}					
p_{4z}					p_{5z-1}

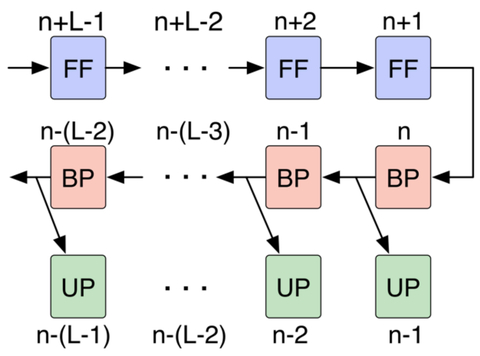
z activations read in permuted order each cycle

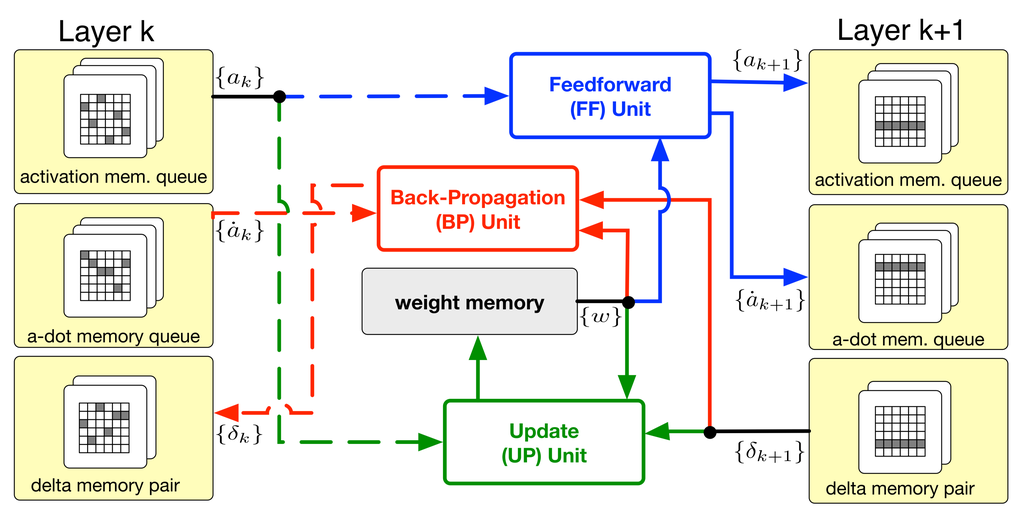
Prevent clashes through proper adjacency matrix design

(5) Hardware Acceleration - Parallelism and Pipelining

- 3 network processes - Feedforward (FF), Backpropagate (BP), Update (UP)
- Parallel execution in a junction by reusing weight memory (figure below)
- Pipelined execution of different inputs across junctions (figure right)

Speedup = 3J, where J = No. of CL junctions





(6) Our Existing Work

[1] S. Dey, Y. Shao, K. M. Chugg, and P. A. Beerel, "Accelerating training of deep neural networks via sparse edge processing," in *Proc. ICANN*. Springer, 2017, pp. 273–280.

[2] S. Dey, P. A. Beerel, and K. M. Chugg, "Interleaver design for deep neural networks," in *Proc. Asilomar Conference on Signals, Systems and Computers*. 2017.

[3] S. Dey, K.-W. Huang, P. A. Beerel, and K. M. Chugg, "Characterizing sparse connectivity patterns in neural networks," in *arXiv:1711.02131*. 2017. Submitted for publication in ICLR 2018.

Contact: souryade@usc.edu, chugg@usc.edu, pabeerel@usc.edu