香港科技大學
THE HONG KONG
UNIVERSITY OF SCIENCE
AND TECHNOLOGY

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

BHNN: a Memory-Efficient Accelerator for Compressing Deep Neural Network with Blocked Hashing Techniques

Jingyang Zhu[1], *Zhiliang Qian*[2*] , and Chi-Ying Tsui[1]

[1] The Hong Kong University of Science and Technology, Hong Kong
[2] Shanghai Jiao Tong University, Shanghai, China

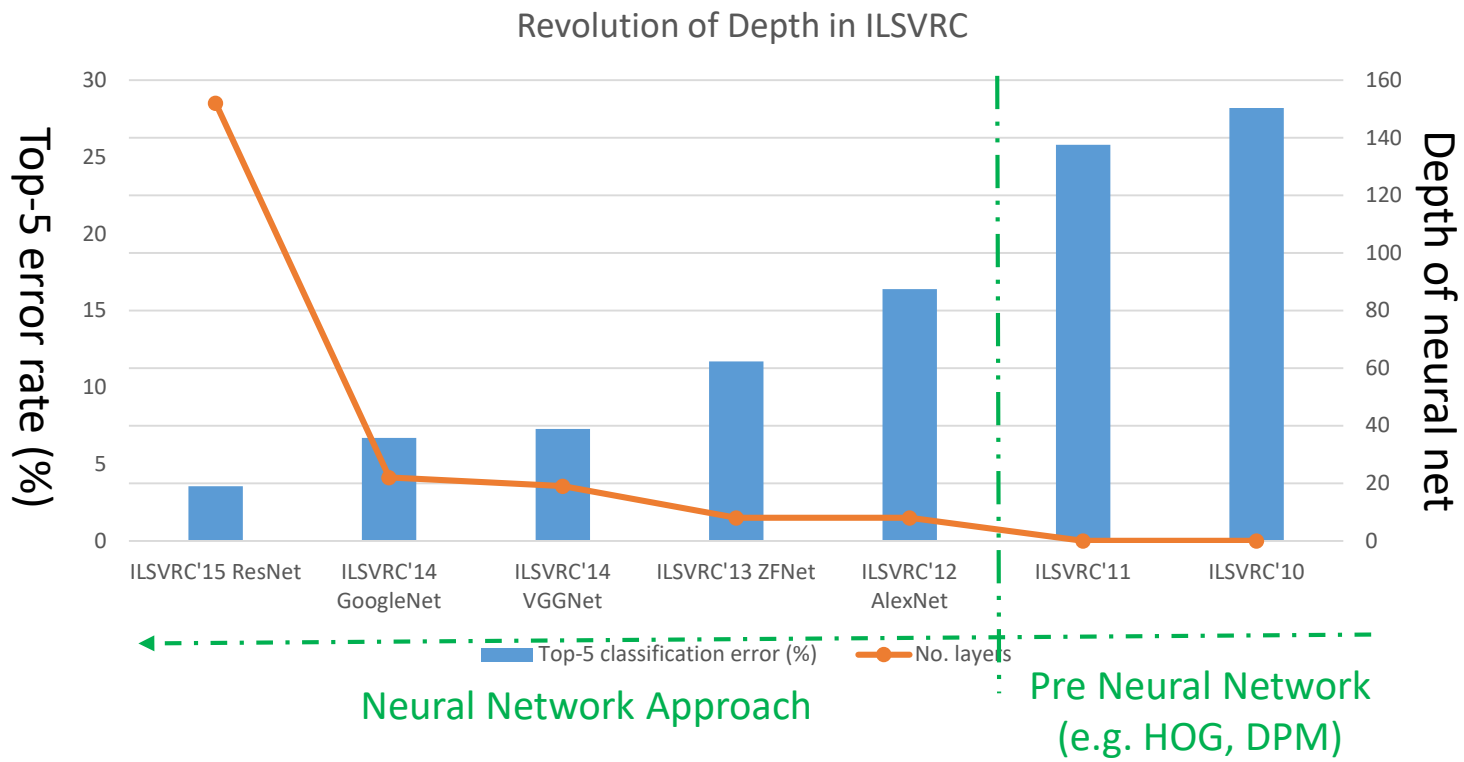
IEEE/ACM ASP-DAC 2017, 19th Jan., 2017, Chiba

Outline

- Introduction
- Related work
- Blocked hash neural network: algorithm
- Blocked hash neural network: hardware architecture
- Experiment results
- Conclusion

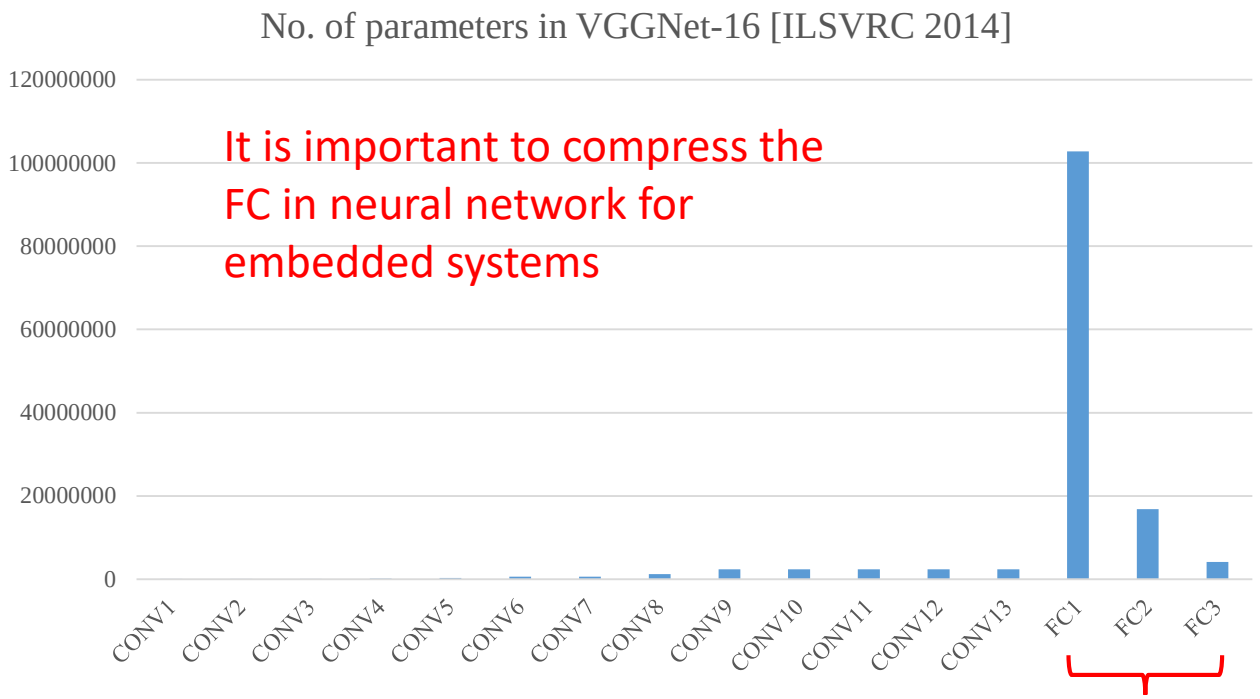
The renaissance of neural network

- Big data & powerful machine (Moore's Law)
- The increasing scale of neural networks in ImageNet visual recognition challenge



Three elementary layers in neural network

- Convolutional layer (CONV): mostly computation-intensive
- Pooling layer (POOL)
- Fully-connected layer (FC): mostly memory-intensive

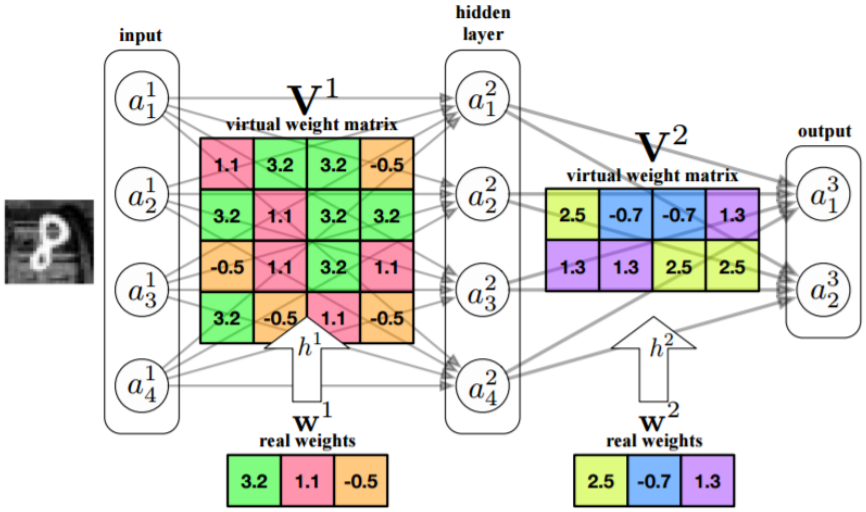


Outline

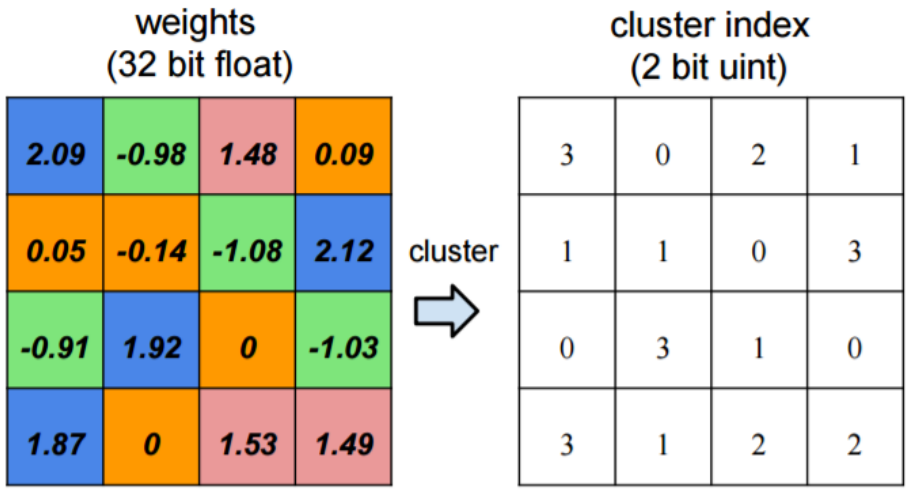
- Introduction
- Related work
- Blocked hash neural network: algorithm
- Blocked hash neural network: hardware architecture
- Experiment results
- Conclusion

Related work

- Hash Neural Network[1]: map real weights into virtual weights through hash function
- Deep compression[2]: map real weights into virtual weights through cluster index (obtained from k-means)



Hash neural network

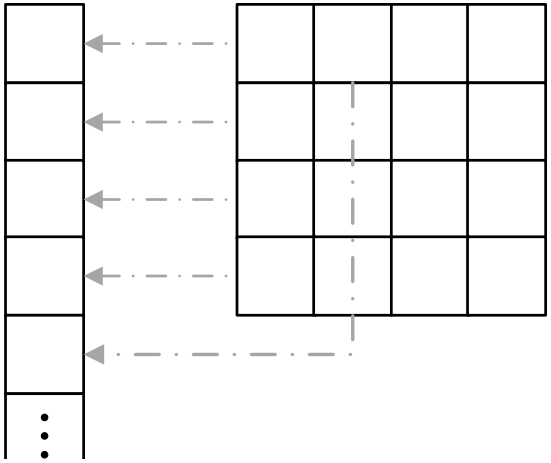


Deep compression

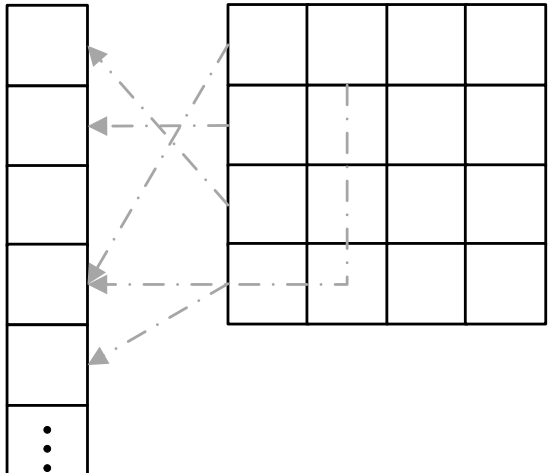
[1] Chen W, Wilson J T, Tyree S, et al. Compressing neural networks with the hashing trick[J].
[2] Han S, Mao H, Dally W J. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding[J].

Potential issues of hash neural network and deep compression

- No spatial locality in hashed neural network



Organize the weights in adjacent locations in memory



Scattering hash map destroys the spatial locality in memory access

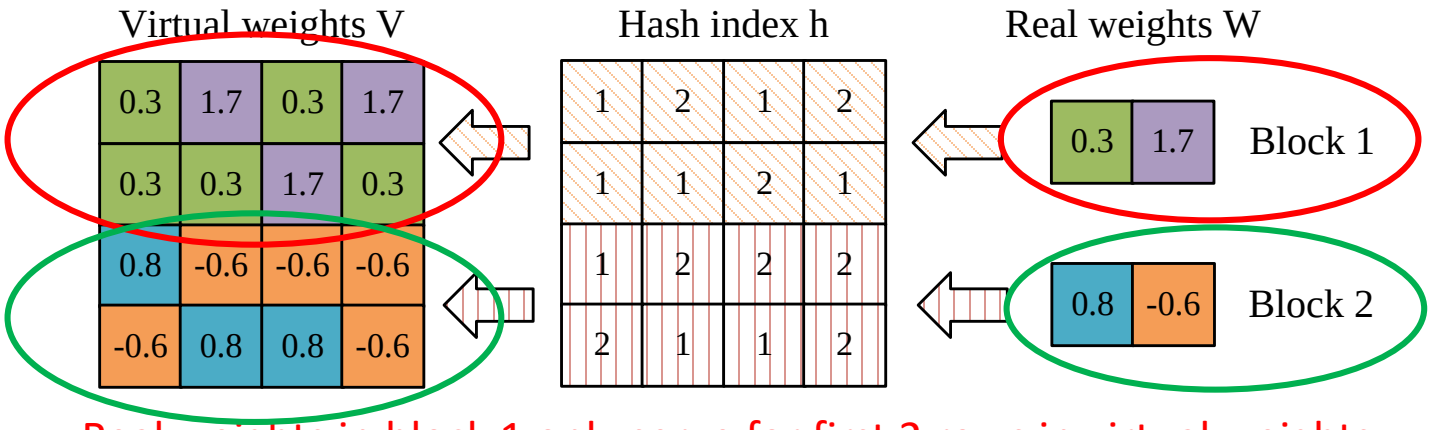
- Deep compression: additional storage of cluster index

Outline

- Introduction
- Related work
- Blocked hash neural network: algorithm
- Blocked hash neural network: hardware architecture
- Experiment results
- Conclusion

Our approach: blocked hash neural network

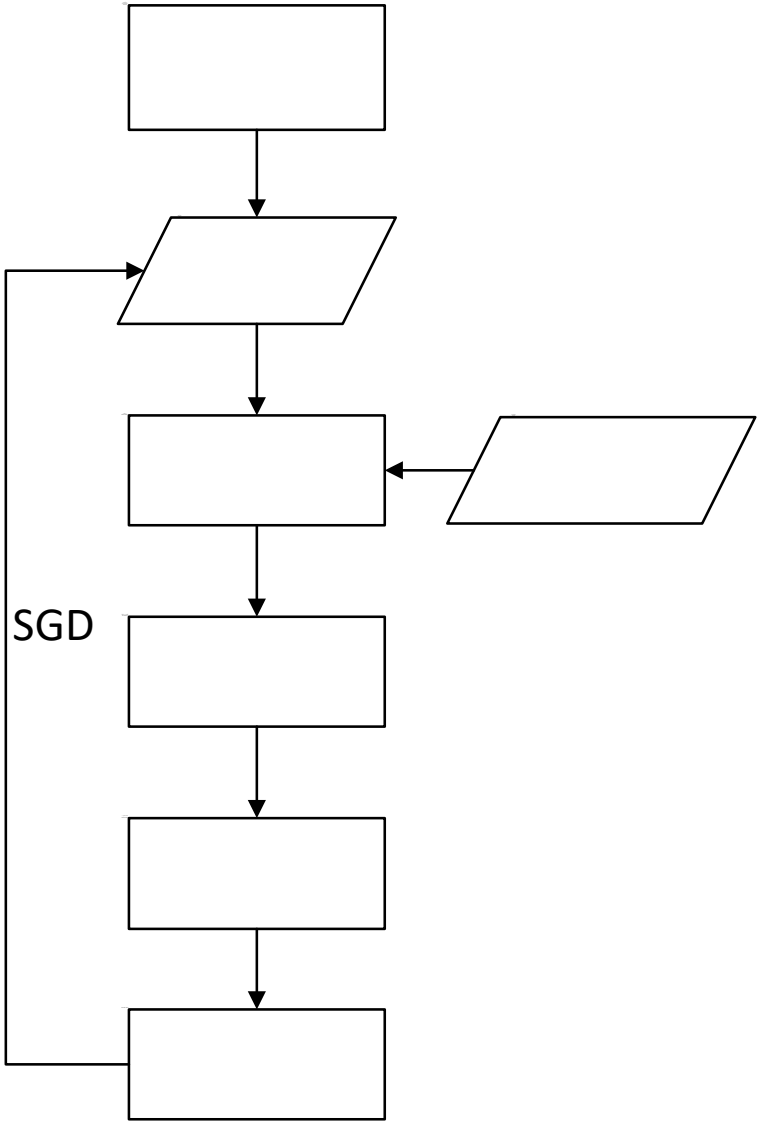
- Basic idea: preserve the spatial locality in hash neural network through the blocked constraint



  - Real weights in block 1 only serve for first 2 rows in virtual weights
  - Real weights in block 2 only serve for last 2 rows in virtual weights
- No cluster index storage is required since the mapping is through a predetermined hash functions
- Real weight block size b determines the tradeoff of locality and performance degradation

Training of blocked hash neural network

- Backpropagation into the real weights, derived from chain rules
  $\frac{\partial \mathcal{L}}{\partial W_k^{(l)}} = \sum_{i,j} \frac{\partial \mathcal{L}}{\partial V_{ij}^{(l)}} \frac{\partial V_{ij}^{(l)}}{\partial W_k^{(l)}} =$
  $\sum_{i,j} \delta_{ij}^{(l)} 1\{h(i,j,l) = k\}$
  where δ is the error term which can be obtained from backpropagation, and $1\{\cdot\}$ is the indicator function
- Training flow of blocked hash neural network



Hardware acceleration of blocked hash neural network: basic arithmetic operation

- The basic operation of feedforward pass is a matrix vector operation

```
for i = 0; i < n_out; i = i + 1 do
    act_out[i] = 0;
    for j = 0; j < n_in; j = j + 1 do
        act_out[i] += V_ij act_in[j];
    end
    act_out[i] = g(act_out[i]);
end
```

$V_{ij} = W_h^{blk}$

Algorithm 1: Pseudo code for the original FC layer.

- memory since we block the hash neural network

Hardware acceleration of blocked hash neural network: loop tiling

- Two-level memory hierarchy
  - On-chip SRAM: store all the real weights W (we can store weights on-chip since we compress the weights a lot)
  - On-chip register file: store only the current active real weight block

```
for i = 0; i < n_out; i = i + 1 do
    act_out[i] = 0;
    for j = 0; j < n_in; j = j + T_i do
        for t_j = j; t_j < min(j + T_j, n_in); t_j = t_j + 1 do
            act_out[i] += V_{it_j} act_in[t_j];
        end
    end
    act_out[i] = g(act_out[i]);
end
```

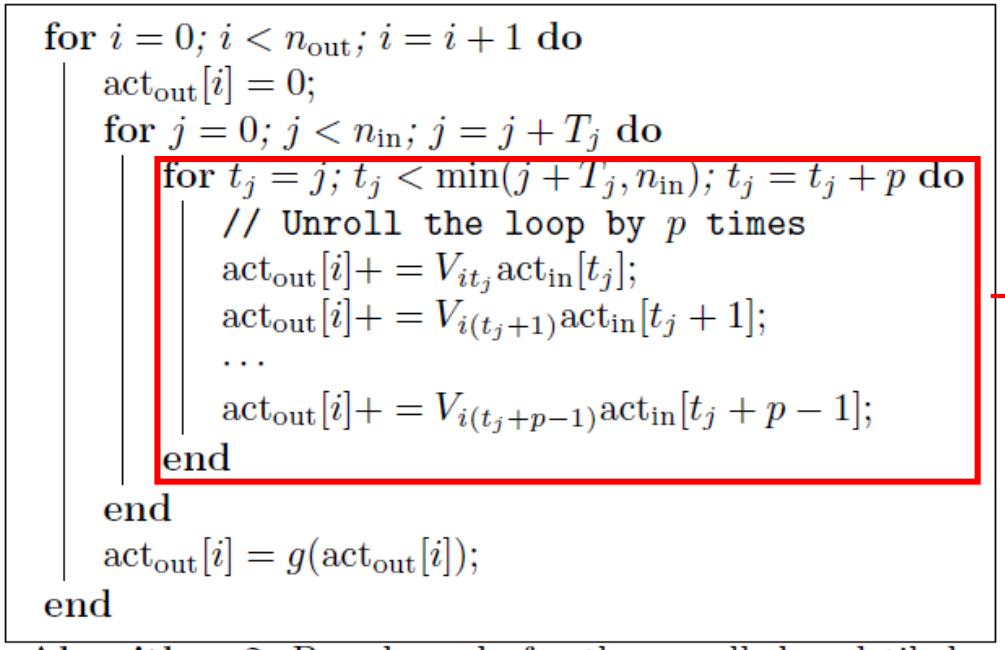
Algorithm 2: Pseudo code for the tiled FC layer.

Within the tiled loop, V_{it_j} comes from the same block

- Loop tiling size $T_j = b/cr$, where b is the real block size and cr is the desired compression rate

Hardware acceleration of blocked hash neural network: loop unrolling

- Unroll the loop to parallelize the computation



Algorithm 3: Pseudo code for the unrolled and tiled FC layer.

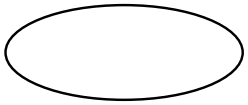
To be computed in p PEs

Outline

- Introduction
- Related work
- Blocked hash neural network: algorithm
- Blocked hash neural network: hardware architecture
- Experiment results
- Conclusion

Hardware architecture of blocked hash neural network

- 5 pipeline stages
  - Refresh & Hash: fetch the real weights from on-chip SRAM to local register file & hash computation for index
  - XBAR & RE In_{act}: crossbar access the corresponding real weights & read the input activations
  - Mult: compute the multiplication in parallel
  - Merge: compute the accumulation
  - ReLU & WB: nonlinear computation & write back to output activation



Simplified hash function for hardware implementation

- A general hash function is complex: large area & multiple cycles
- A simplified hash function is tolerable due to our simulation results

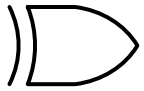
$$V_{ij}^{(l)} = W_{h(i,j,l)}^{(l)} \xi(i,j,l)$$

  - $h(i,j,l)$: $N^3 \rightarrow \{1, 2, \dots, b\}$ determines the mapping procedure. Uniformly distributes the index within range 1 to b:

$$h(i,j,l) = (i \oplus j \oplus l)\%b$$

  - $\xi(i,j,l)$: $N^3 \rightarrow \{-1, +1\}$ removes the bias of hash inner-product. Randomly generates the ± 1:

$$\xi(i,j,l) = LFSR\%2$$



Hardware block of hash generator

Outline

- Introduction
- Related work
- Blocked hash neural network: algorithm
- Blocked hash neural network: hardware architecture
- Experiment results
- Conclusion

Simulation setup

- Algorithm level verification on MATLAB
  - Verify the performance degradation under different compression rates
  - Offline train using stochastic gradient descent (SGD)
  - Numerical gradient check to guarantee the backpropagation is correct
  - Fixed point simulation of hardware architecture
- Hardware implementation by Verilog
  - Synthesize, place and routing using Xilinx Vivado
  - Prototype in Xilinx FPGA VC709
- Evaluate performance over 3 different datasets with variant neural network architecture only consisting of FC layers
  - Caltech 101 silhouettes
  - Handwritten MNIST
  - Handwritten MNIST challenging variant ROT

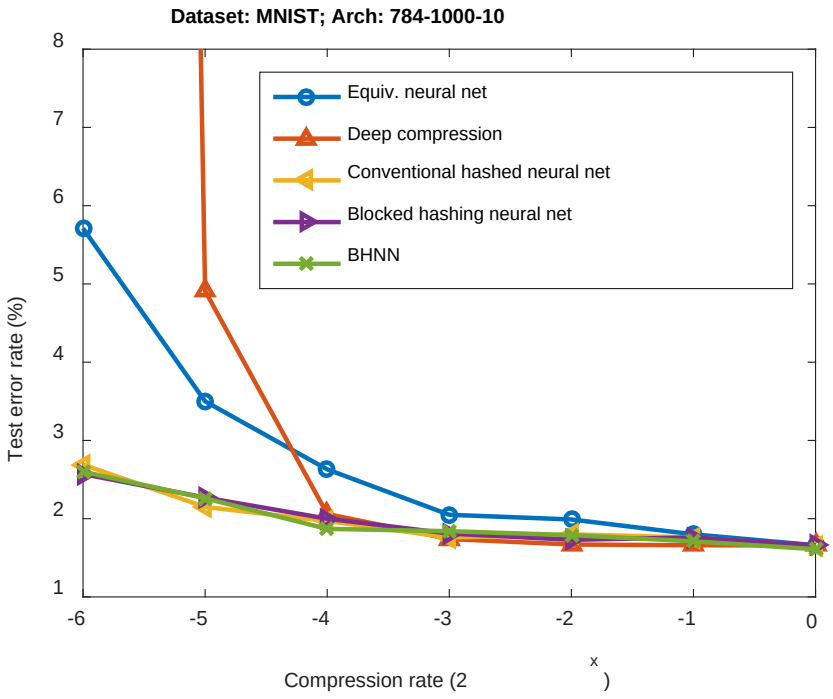
Algorithm-level compression performance

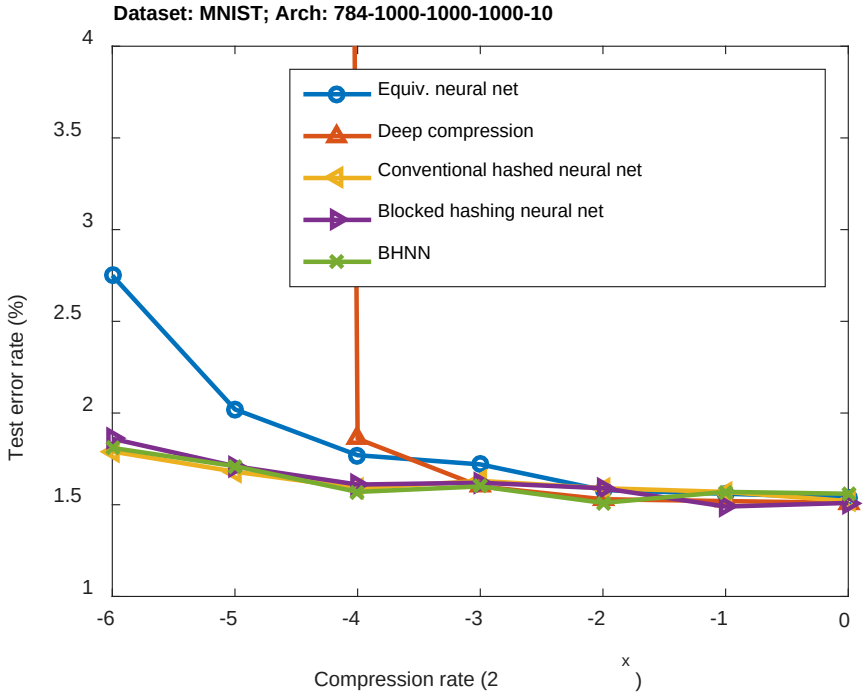
- Five neural network compression schemes
  - Equivalent neural net: plain neural network with equiv. size
  - Deep compression
  - Conventional hash network: with ideal hash function[*]
  - Blocked hash network: hash network with blocked constraints
  - BHNN: hardware-mimic blocked hash neural network

[*] Ideal hash function: open source xxhash @ https://github.com/Cyan4973/xxHash

Algorithm-level compression performance (cont.)

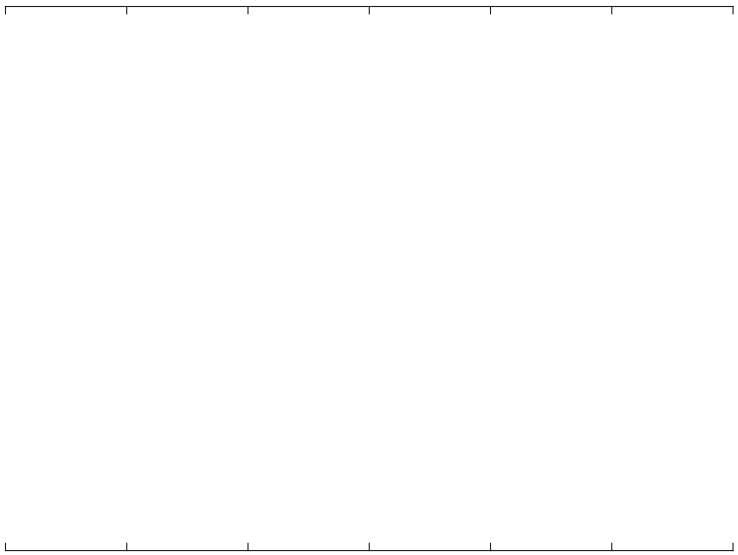
- Dataset: MNIST
- Over 2 different neural network architectures

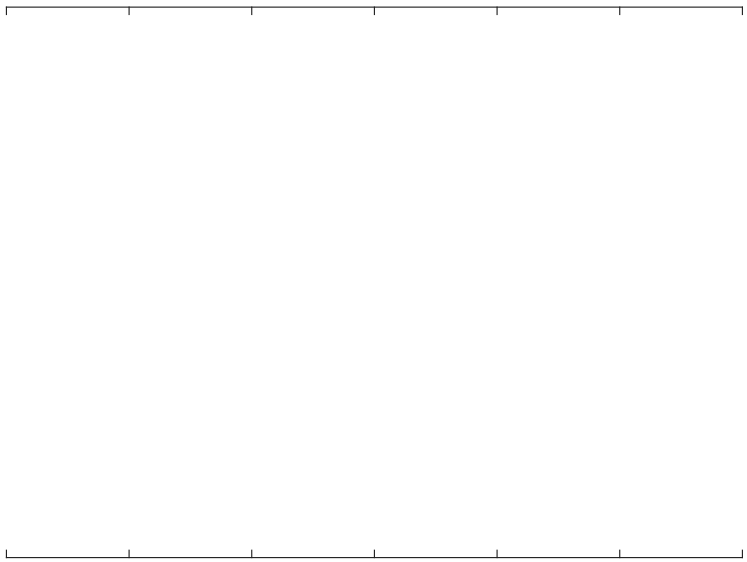




Algorithm-level compression performance (cont.)

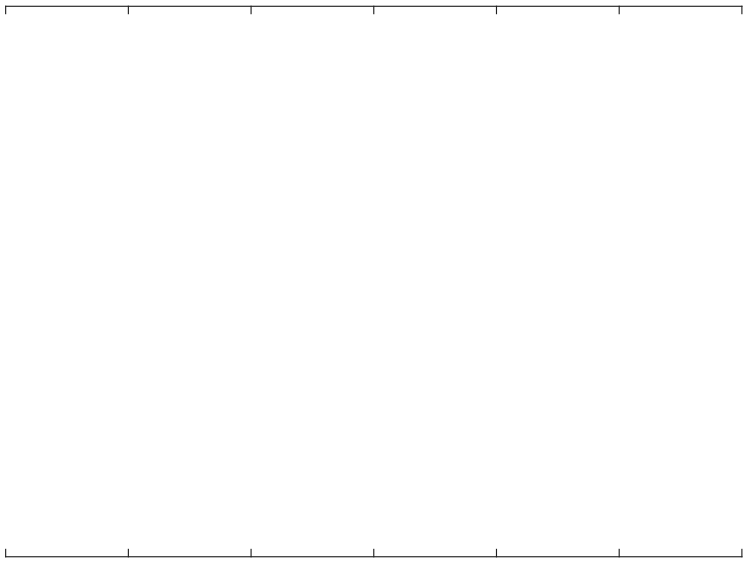
- Dataset: Caltech101
- Over 2 different neural network architectures

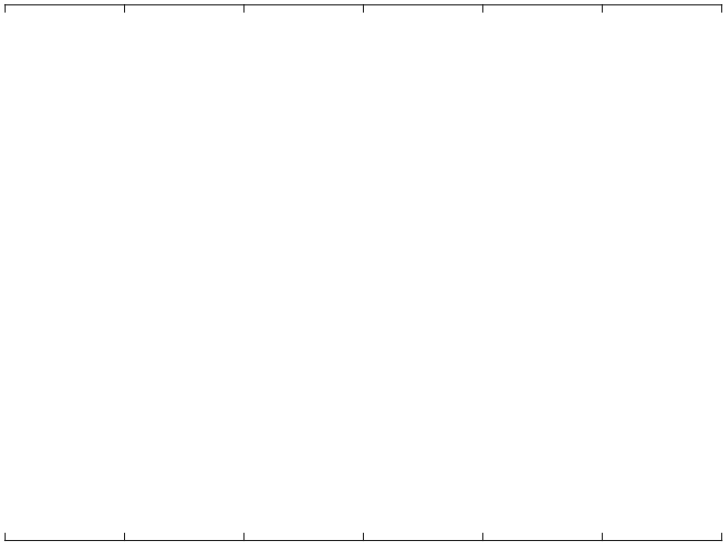




Algorithm-level compression performance (cont.)

- Dataset: ROT
- Over 2 different neural network architectures





Hardware implementation results

- Micro-architectural parameters on Xilinx VC709

Micro-architectural parameters	Value
Parallelism p	32
Real block size b	64
Quantization scheme	Q0.15 (16-bit)
Compression ratio	1/16
Clock freq	100MHz

- Resource utilization

Resource	DSP	BRAM	LUT	FF
Used	32	939	17671	13271
Utilization (%)	0.89	63.88	4.08	1.53

Hardware acceleration results

- Baseline: conventional hash (w/o spatial locality) on CPU & GPU

Arch / Platform	Intel i7 6700HQ	NVIDIA GTX 960M	Xilinx VC709
784-1000-10	5.54ms / image	1.17ms / image	0.25ms / image
784-1000-1000-1000-10	20.19ms / image	2.98ms / image	0.89ms / image
784-1000-101	6.31ms / image	1.25ms / image	0.28ms / image
784-1000-1000-1000-101	20.96ms / image	3.05ms / image	0.92ms / image

- FPGA has a speedup of over 20x and 3~5x over CPU and GPU under various neural network architectures

Outline

- Introduction
- Related work
- Blocked hash neural network: algorithm
- Blocked hash neural network: hardware architecture
- Experiment results
- Conclusion

Conclusion

- A novel compression algorithm for neural network
  - Spatial locality for the conventional hash neural network
  - Maintain the same performance with the conventional hash neural network and outperform the “deep compression” scheme under heavy compression region
- A hardware accelerator catering for the blocked hash neural network
  - Two-level memory hierarchy taking advantage of the spatial locality
  - Hardware simplification of the general hash function
  - Achieves ~20x and ~4x speedup compared with CPU and GPU