LRADNN: High-Throughput and Energy-Efficient Deep Neural Network Accelerator using Low Rank Approximation

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

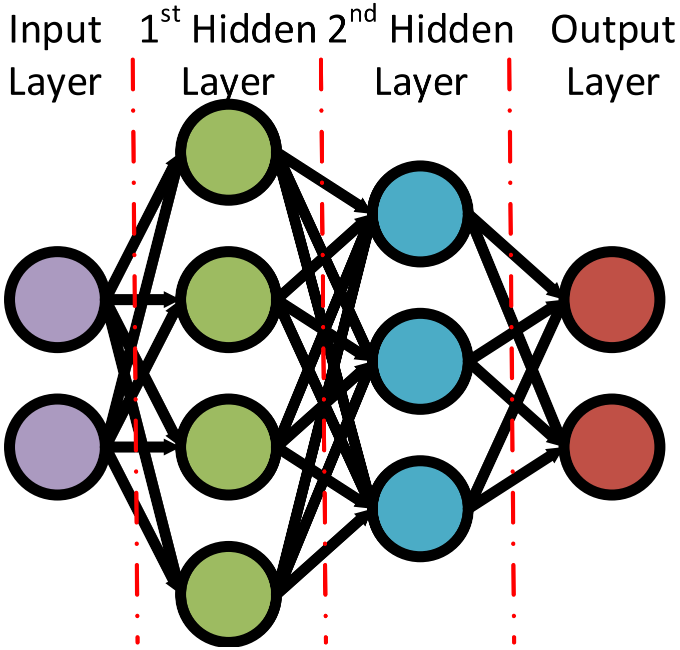
IEEE/ACM ASP-DAC 2016, 28th Jan., 2016, Macao

Outline

- Introduction
- Related work
- Motivation of LRADNN
- Low rank approximation (LRA) predictor
- LRADNN: hardware architecture
- Experiment results
- Conclusion

Deep neural network (DNN) and hardware acceleration

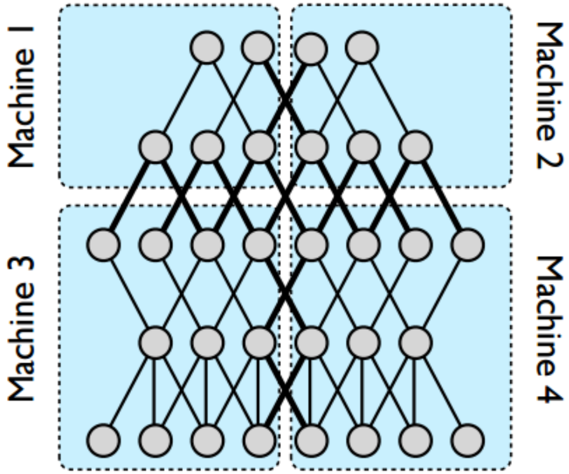
- Layer-wise organization with hierarchical feature extractions



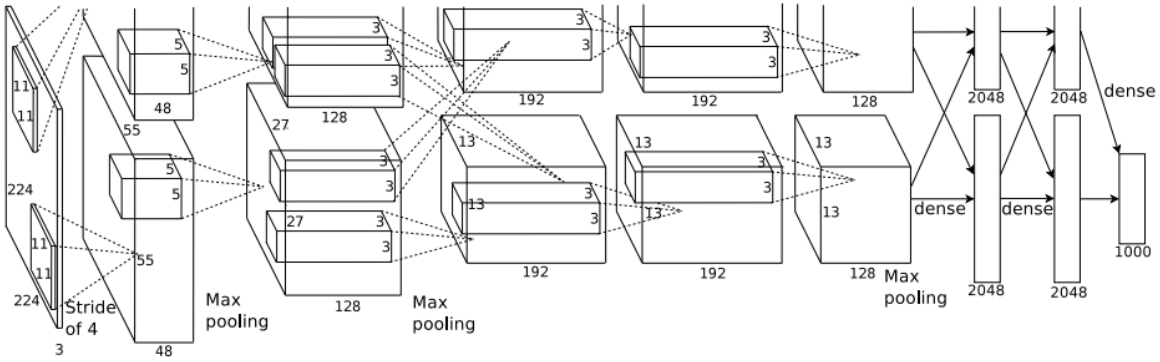
Deep neural network (DNN) and hardware acceleration

- Layer-wise organization with hierarchical feature extractions
- Hardware acceleration
  - CPU clusters: Google brain
  - GPU clusters: AlexNet
  - ASIC: IBM TrueNorth

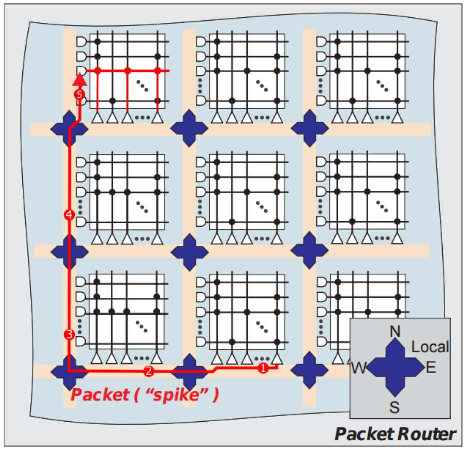
Google brain



AlexNet



IBM TrueNorth



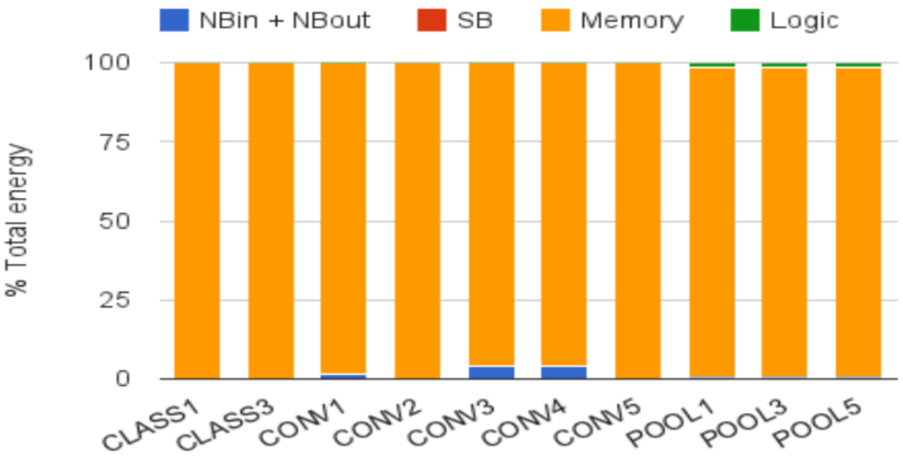
Outline

- Introduction
- Related work
- Motivation of LRADNN
- Low rank approximation (LRA) predictor
- LRADNN: hardware architecture
- Experiment results
- Conclusion

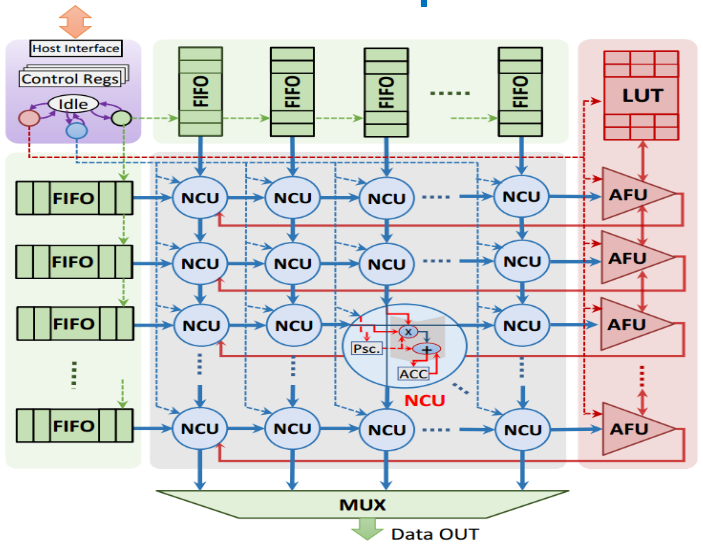
Related work

- DianNao: a general accelerator for DNN
  - Large energy consumption in memory access
- AxNN: energy-efficient accelerator by approximating resilient neurons
  - Only reduce power in datapath!

Energy consumption in DianNao



AxNN datapath

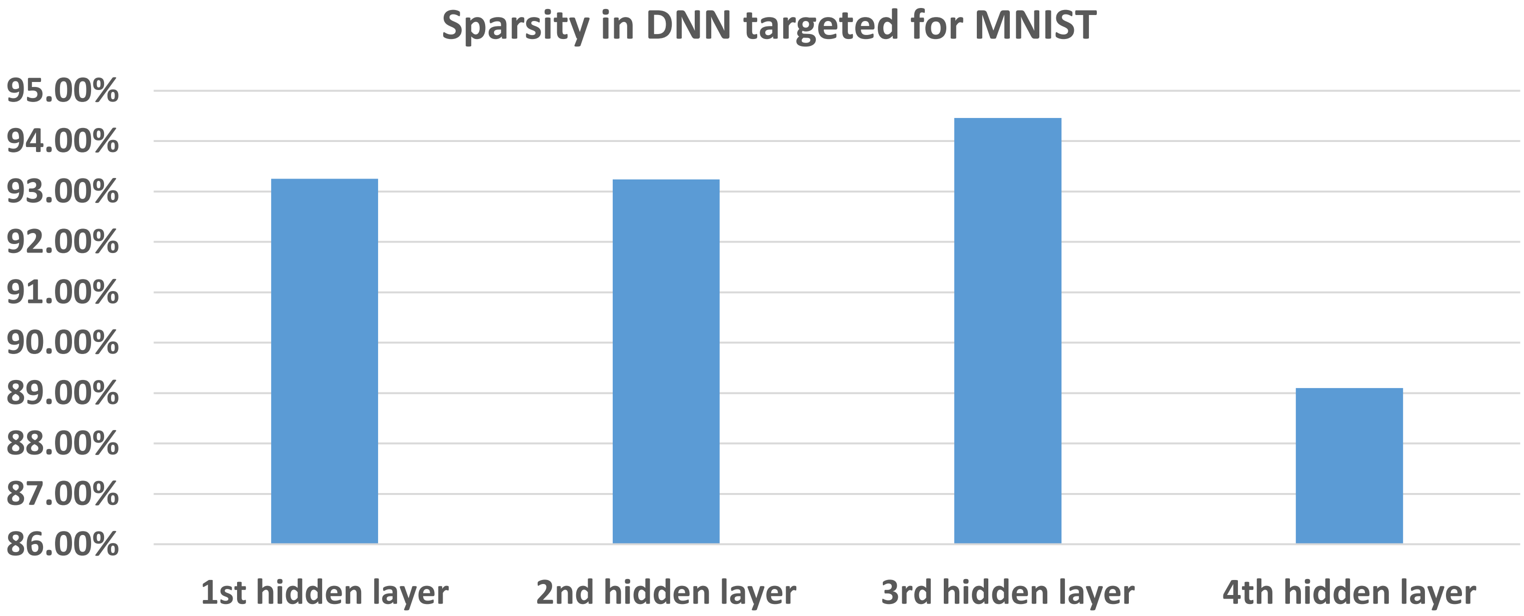


Outline

- Introduction
- Related work
- Motivation of LRADNN
- Low rank approximation (LRA) predictor
- LRADNN: hardware architecture
- Experiment results
- Conclusion

Motivation: sparsity in DNN

- Intrinsic sparsity in DNN to avoid overfitting / better feature extractions
    - Sparsity: proportion of inactive neurons (activation = 0)



Motivation: sparsity in DNN

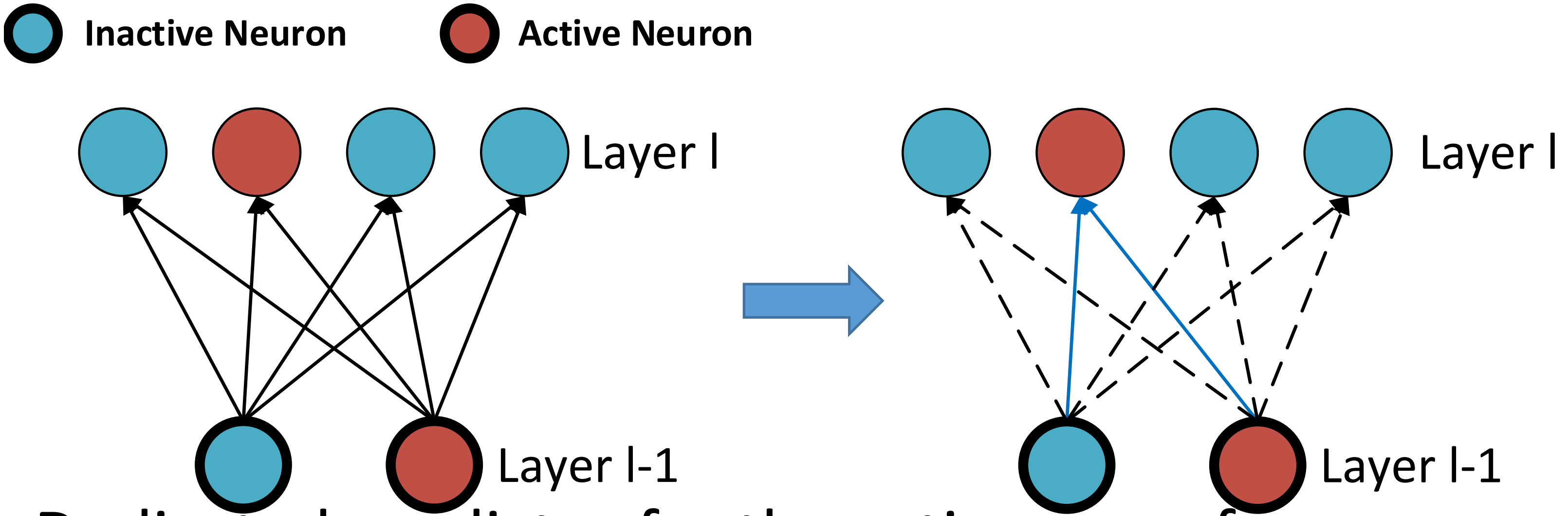
- Intrinsic sparsity in DNN to avoid overfitting / better feature extractions
  - Sparsity: proportion of inactive neurons (activation = 0)
- Conventional computation

$$a_i^{(l+1)} = \sigma\left(\sum_{j=1}^{s_l} W_{ij}^{(l)} a_j^{(l)}\right), \overbrace{i \in [1, s_{l+1}]}^{\text{Regardless of the activeness of neurons}}$$

  - Arithmetic ops / neuron: s_l mults, $s_l - 1$ adds, 1 nonlinear
  - Memory accesses / neuron: $2s_l$ (weights & acts)

Motivation: bypass unnecessary operations

- Dynamically bypass the inactive neurons



- Dedicated predictor for the activeness of neurons
  - Simple: less arithmetic operations involved
  - Accurate: small performance degradation

Outline

- Introduction
- Related work
- Motivation of LRADNN
- Low rank approximation (LRA) predictor
- LRADNN: hardware architecture
- Experiment results
- Conclusion

Low rank approximation (LRA) predictor [1]

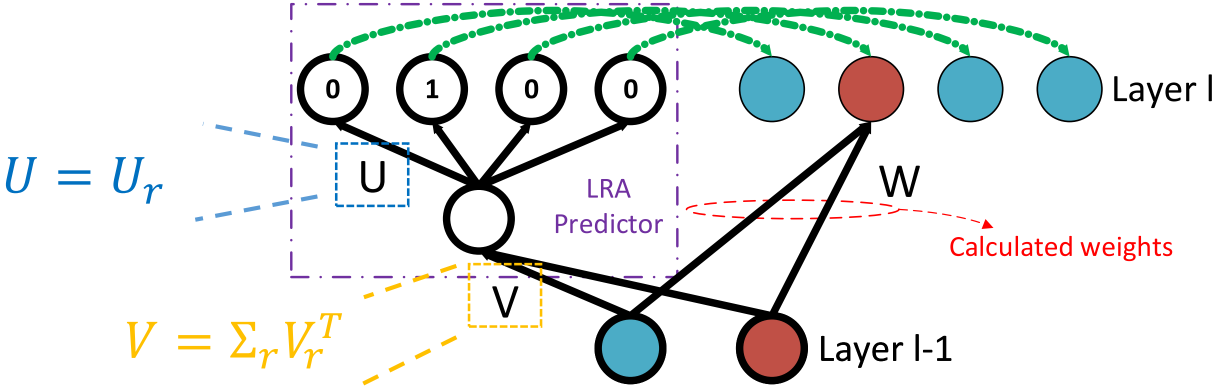
- Approximation for synaptic weights W

$$\min_{\widetilde{W}} |\widetilde{W} - W|_F$$

$$s.t.\ \operatorname{rank}\big(\widetilde{W}\big) = r$$

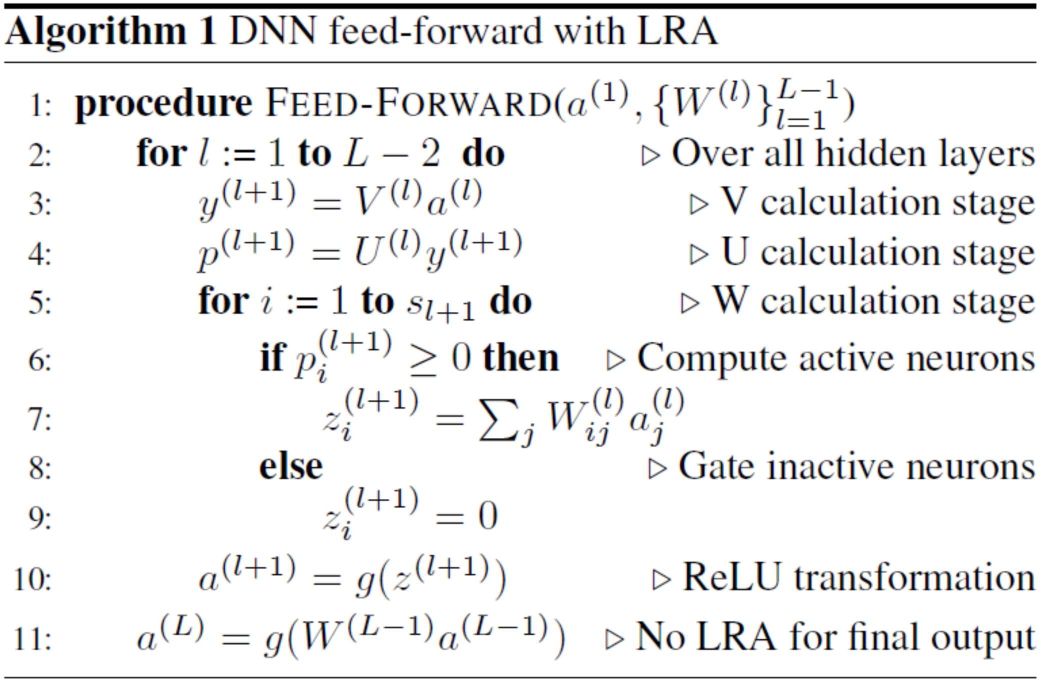
where r is a hyper-parameter controlling the prediction accuracy vs. computation complexity

- A nice closed form solution based on SVD



[1] Davis A, Arel I. Low-Rank Approximations for Conditional Feedforward Computation in Deep Neural Networks[J]. arXiv preprint arXiv:1312.4461, 2013.

Feedforward pass with LRA

- 

Algorithm 1 DNN feed-forward with LRA

```
procedure FEED-FORWARD($a^{(1)}, \{W^{(l)}\}_{l=1}^{L-1}$)
    for $l := 1$ to $L-2$ do                      ▷ Over all hidden layers
        $y^{(l+1)} = V^{(l)} a^{(l)}$                  ▷ V calculation stage
        $p^{(l+1)} = U^{(l)} y^{(l+1)}$                ▷ U calculation stage
        for $i := 1$ to $s_{l+1}$ do                 ▷ W calculation stage
            if $p_i^{(l+1)} \geq 0$ then              ▷ Compute active neurons
                $z_i^{(l+1)} = \sum_j W_{ij}^{(l)} a_j^{(l)}$
            else                                  ▷ Gate inactive neurons
                $z_i^{(l+1)} = 0$
        $a^{(l+1)} = g(z^{(l+1)})$                     ▷ ReLU transformation
    $a^{(L)} = g(W^{(L-1)} a^{(L-1)})$                 ▷ No LRA for final output
```

- Take LRA into offline training (backpropagation) to improve the performance

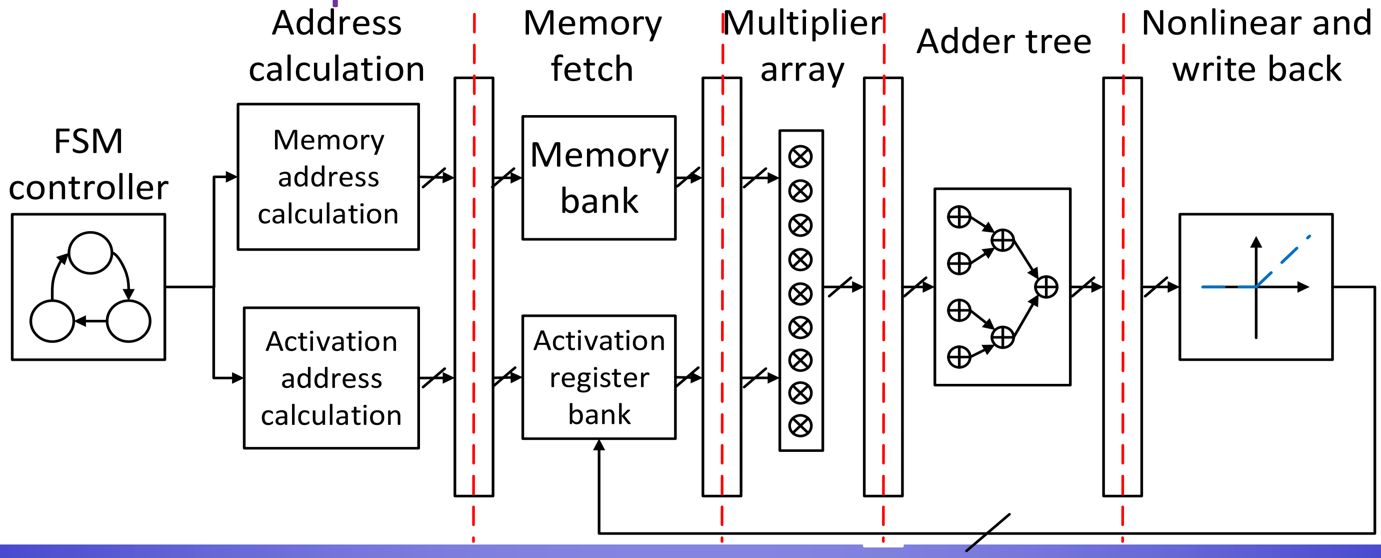
Outline

- Introduction
- Related work
- Motivation of LRADNN
- Low rank approximation (LRA) predictor
- **LRADNN: hardware architecture**
- Experiment results
- Conclusion

LRADNN: hardware architecture for LRA predictor

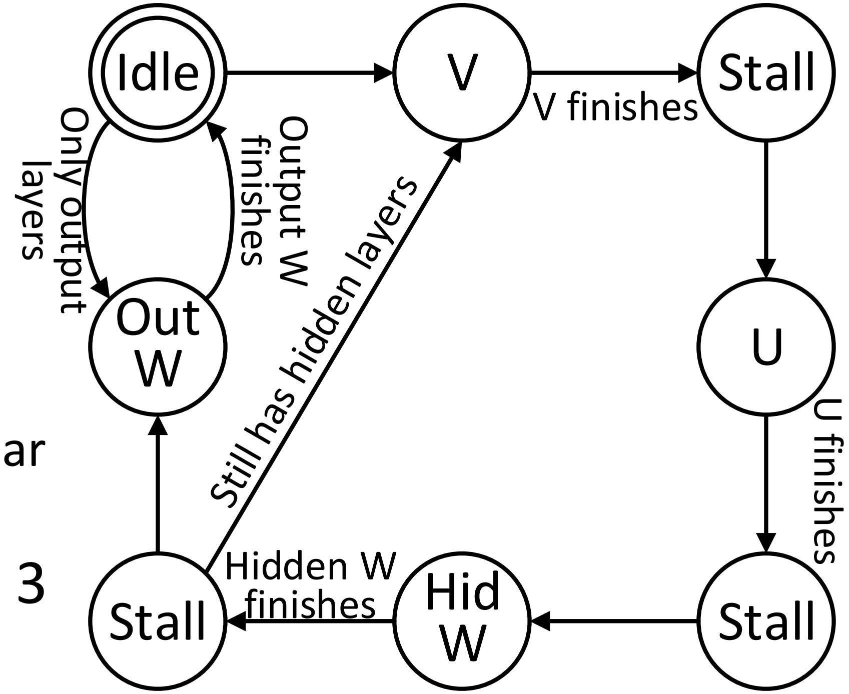
- Top view of the architecture: 5-stage pipeline
  - Address calculation
  - Memory fetch
  - Multiplication
  - Addition
  - Nonlinear operation and write back

$$a_i^{(l+1)} = \sigma\left(\sum_{j=1}^{s_l} W_{ij}^{(l)} \times a_j^{(l)}\right)$$



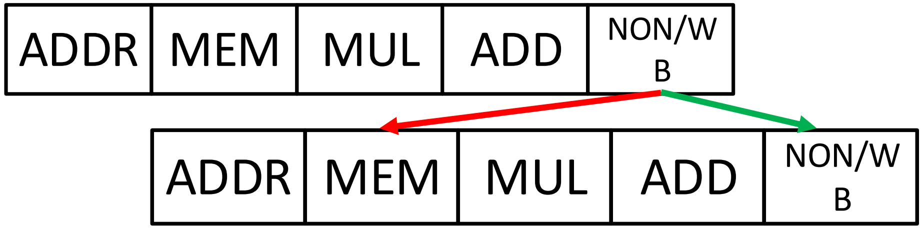
Status controller

- 3 calculation statuses
  - V computation
  - U computation
  - W computation
    - Hidden layer
    - Output layer (no nonlinear operation)
  - Data dependency (delay 3 CCs)



$p^{(l)}=U^{(l)}V^{(l)}z^{(l)}$

$a^{(l+1)}=\mathrm{LRA}(p^{(l)}, a^{(l)})$



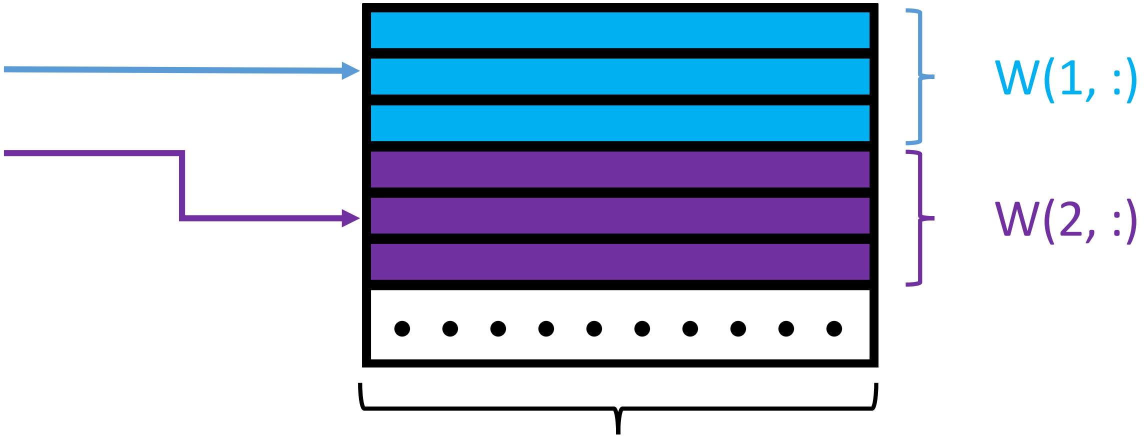
ADDR	MEM	MUL	ADD	NON/WB

ADDR	MEM	MUL	ADD	NON/WB

Address calculation: memory organization

- Memory word width: parallelism of the accelerator

$$\begin{bmatrix} W_{11} & \cdots & W_{1n} \\ W_{21} & \cdots & W_{2n} \\ \vdots & \ddots & \vdots \\ W_{m1} & \cdots & W_{mn} \end{bmatrix}$$



One word: parallelism of the accelerator

- For a specified weight W_{ij}

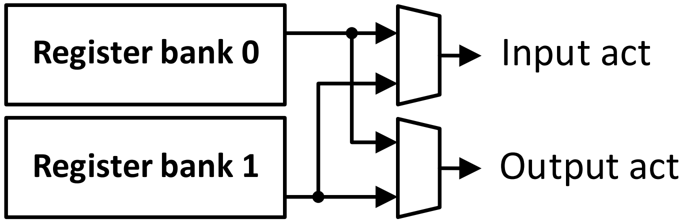
$$\mathrm{addr}\left(W_{ij}\right) = i\lceil \mathrm{n/p} \rceil + \lfloor j/p \rfloor$$

# word lines for an output neuron

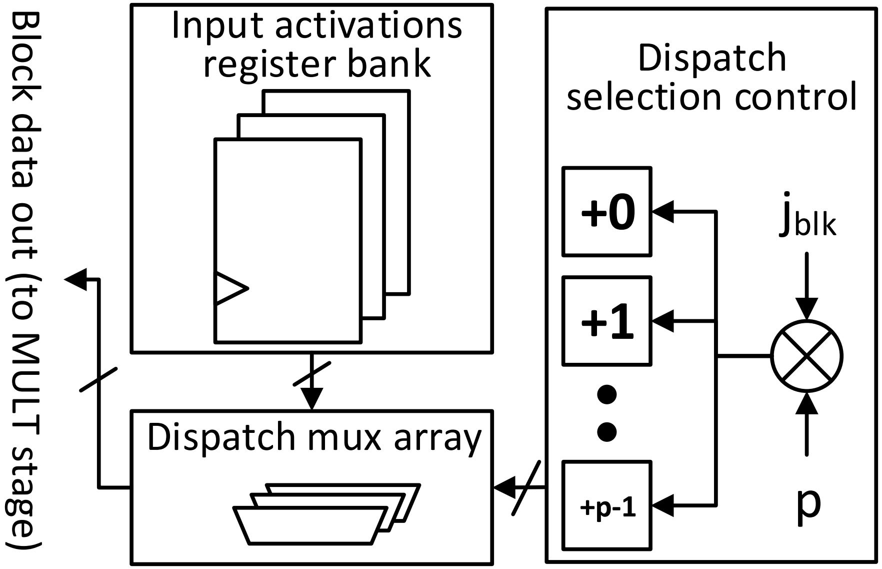
Column block $\mathrm{j_{blk}}$

Memory access: activation register banks

- Two physical register banks are interleaved to two logical registers: input and output activations

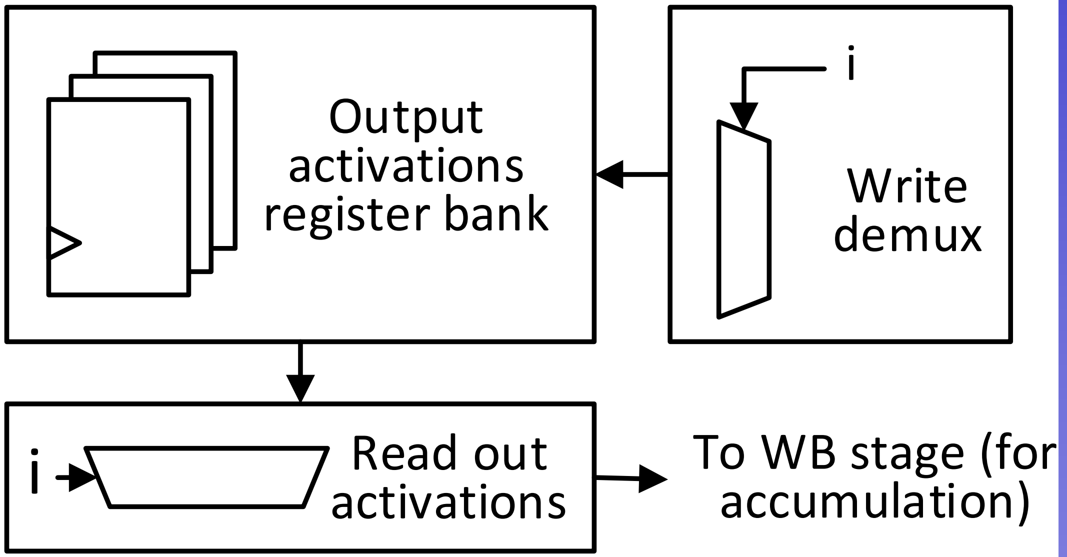


- Input activations


Block data out (to MULT stage)
Input activations register bank
Dispatch mux array
Dispatch selection control
+0
+1
+p-1
j_{blk}
p

Memory access: activation register banks (cont.)

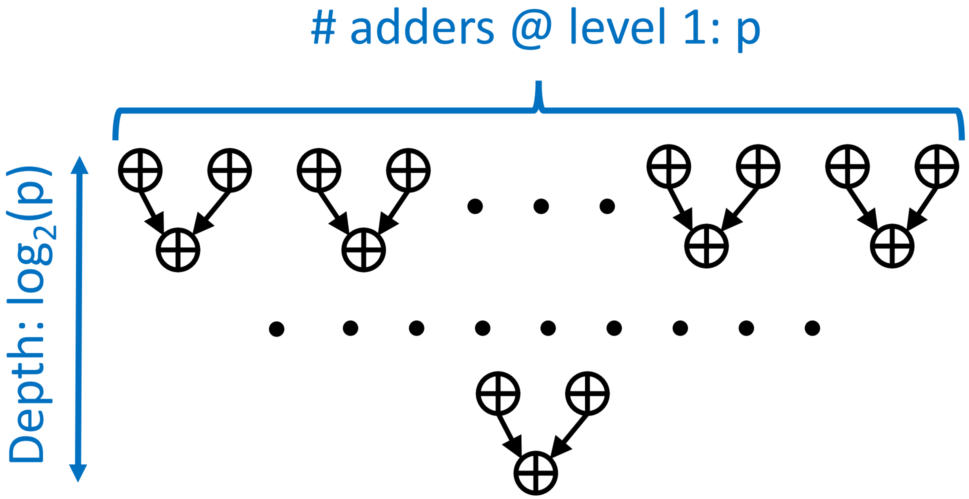
- Output activations
- Extra register for LRA scheme ($\leq 5\%$)
  - Tmp1: $\mathrm{tmp1} = V^{(l)} z^{(l)}$
  - Tmp2: $\mathrm{tmp2} = U^{(l)} \mathrm{tmp1}$
  - Predictor: $p^{(l)} = \mathrm{tmp2} > 0$

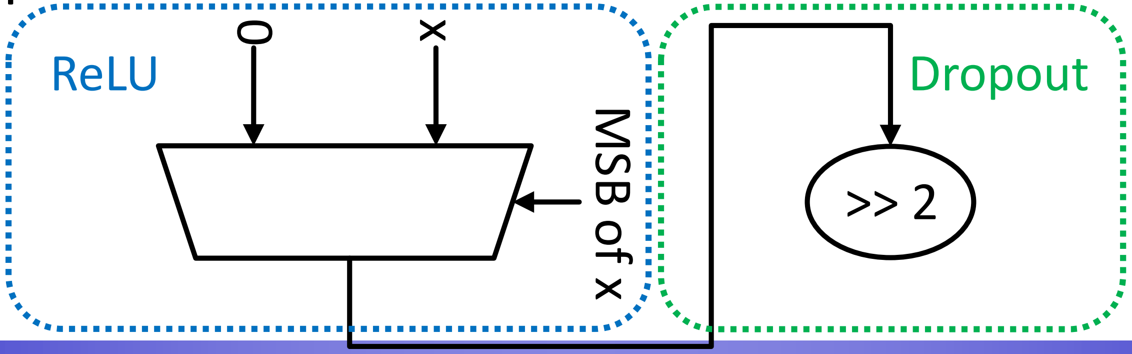

Output activations register bank
i
Write demux
i
Read out activations
To WB stage (for accumulation)

Logical variable	Tmp1	Tmp2	Predictor
Physical location	Tmp1 reg	Output act	Predictor reg
Size (depth x width)	rank x FP width	# acts x FP width	# acts x 1

Computational stages

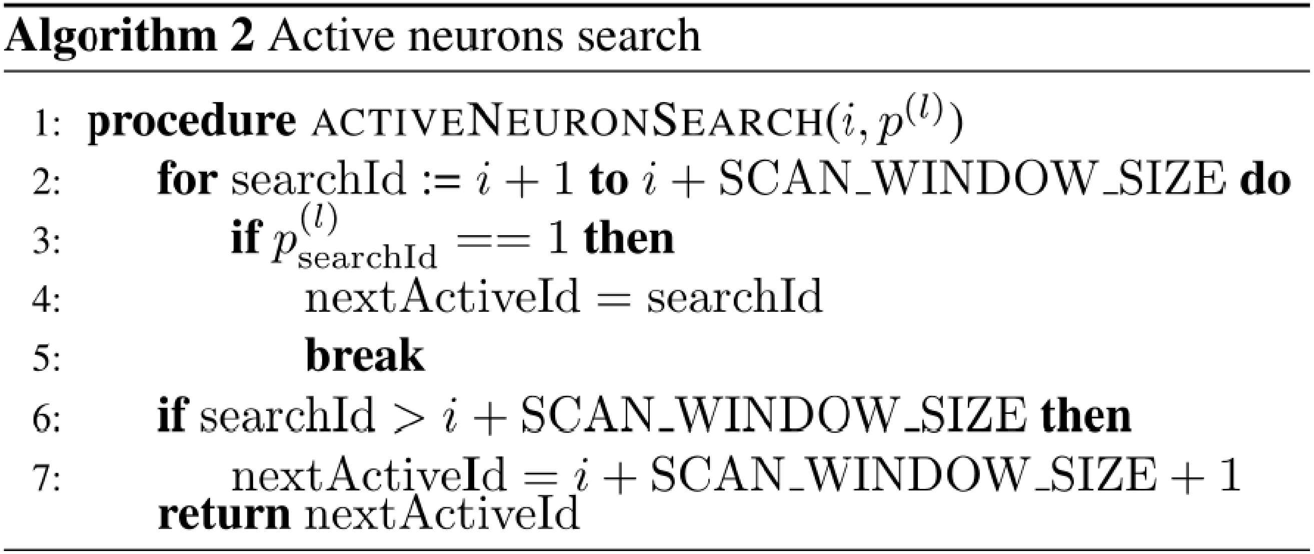
- Parallel multiplication
  - p multipliers, determining the parallelism of the accelerator
- Merging operation
  - Adder tree
- Nonlinear operation
  - ReLU
  - Dropout





Active neurons search: behavior level

- Priority encoder based search



Algorithm 2 Active neurons search

```
procedure ActiveNeuronSearch($i, p^{(l)}$)
    for searchId := $i + 1$ to $i$ + SCAN_WINDOW_SIZE do
        if $p^{(l)}_{\mathrm{searchId}} == 1$ then
            nextActiveId = searchId
            break
    if searchId > $i$ + SCAN_WINDOW_SIZE then
        nextActiveId = $i$ + SCAN_WINDOW_SIZE + 1
    return nextActiveId
```

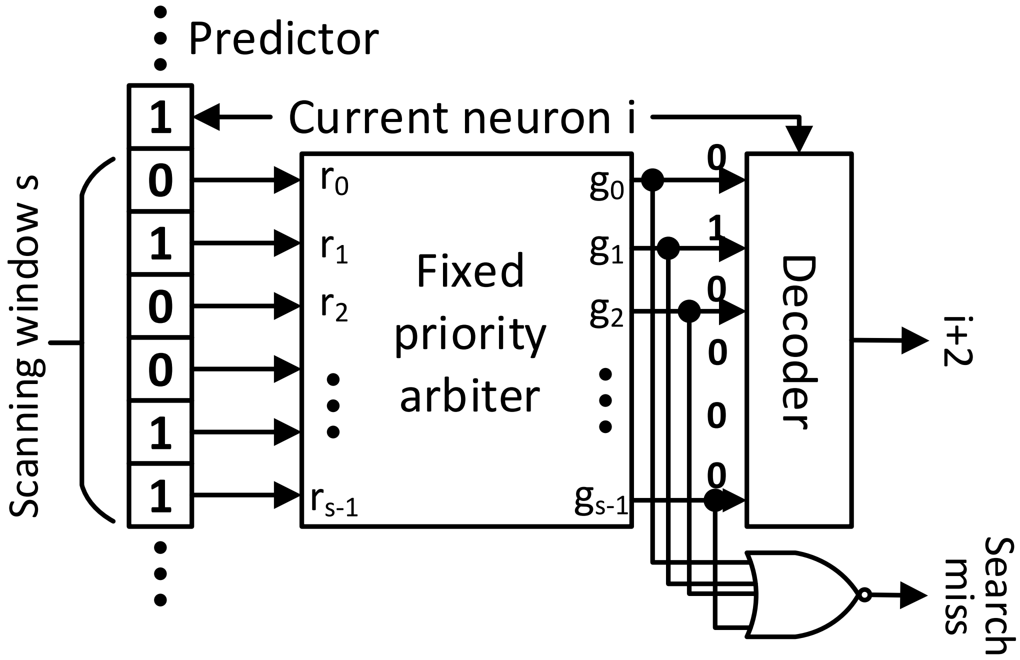
Priority encoder search

Search miss

- Search miss penalty: 1 CC

Active neurons search: hardware level

- Priority encoder based search
  - Higher priority assigned to LSB
- Decoder
  - One hot to binary decoder



Outline

- Introduction
- Related work
- Motivation of LRADNN
- Low rank approximation (LRA) predictor
- LRADNN: hardware architecture
- Experiment results
- Conclusion

Simulation setup

- Behavior level simulation on MATLAB
  - Offline training
  - Fixed point simulation
- RTL implementation using Verilog
  - Technology node: TSMC 65nm LP standard cell
  - Memory model: HP CACTI 6.5
- Comparison of behavior, pre-synthesis and post-synthesis simulations
- Power evaluation based on the post-synthesis simulation (extracted switching activities)
- 3 real benchmarks are tested on LRADNN: MNIST, Caltech101, and SVHN

Summary of the implementation

Micro-architecture parameters	Value
# multipliers (parallelism)	32
Depth of the activation register bank	1024
Layer index width (max layer no.)	3
Fixed point for the internal data path	Q7.12
Fixed point for W memory	Q2.11
Fixed point for U, V memory	Q5.8
W memory size	3.5MB
U, V memory size	448KB
Scanning window size	16
# V calculation registers (max rank)	128

Area and timing results

- Memory dominant for area and timing

	LRADNN (direct)	LRADNN
Total area (mm2)	51.94	64.18
Critical path (ns)	8.94	9.03

- Area overhead caused by U, V memory

$$448KB \times 2/3.5MB = 25\%$$

- Timing overhead caused by extra U, V, and W memory data MUX selection (~ 1% increase)

Training results on real applications

- Prediction loss: test error rate for LRA feedforward – test error rate for plain feedforward

	Caltech 101 silhouettes	MNIST	SVHN
Architecture	784-1023-101	784-1000-600-400-10	1023-1000-700-400-150-10
# connections	0.90M	1.63M	2.06M
Rank	50	50-35-25	100-70-50-25
Prediction loss (fixed point)	-0.09%	0.07%	-0.93%

Theoretical lower bounds for accelerators

- Number of cycles

$$\frac{\#\ \text{synptic connections in DNN}}{\text{parallelism}} = \frac{\sum_{l=1}^{L-1} s_{l+1}(1 + s_l)}{\text{parallelism}}$$

- Power consumption (only consider memory access power)

$$\frac{\mathrm{E_{FF}}}{t_{FF}} = \frac{\mathrm{E_{MEM}} \times \mathrm{n_{cyc}}}{T_{\mathrm{cyc}} \times \mathrm{n_{cyc}}}$$

E_{FF}: energy consumption during feedforward
t_{FF}: elapsed time for feedforward
n_{cyc}: number of ideal cycles
E_{MEM}: memory read energy per access

Timing and power results on real applications

- Power consumption: average post-synthesis simulations on the first 10 testing samples

Number of cycles

	Ideal	LRADNN (direct)	LRADNN
Caltech 101 silhouettes	28327	29840	23141
MNIST	50938	52971	30105
SVHN	64586	66245	49371

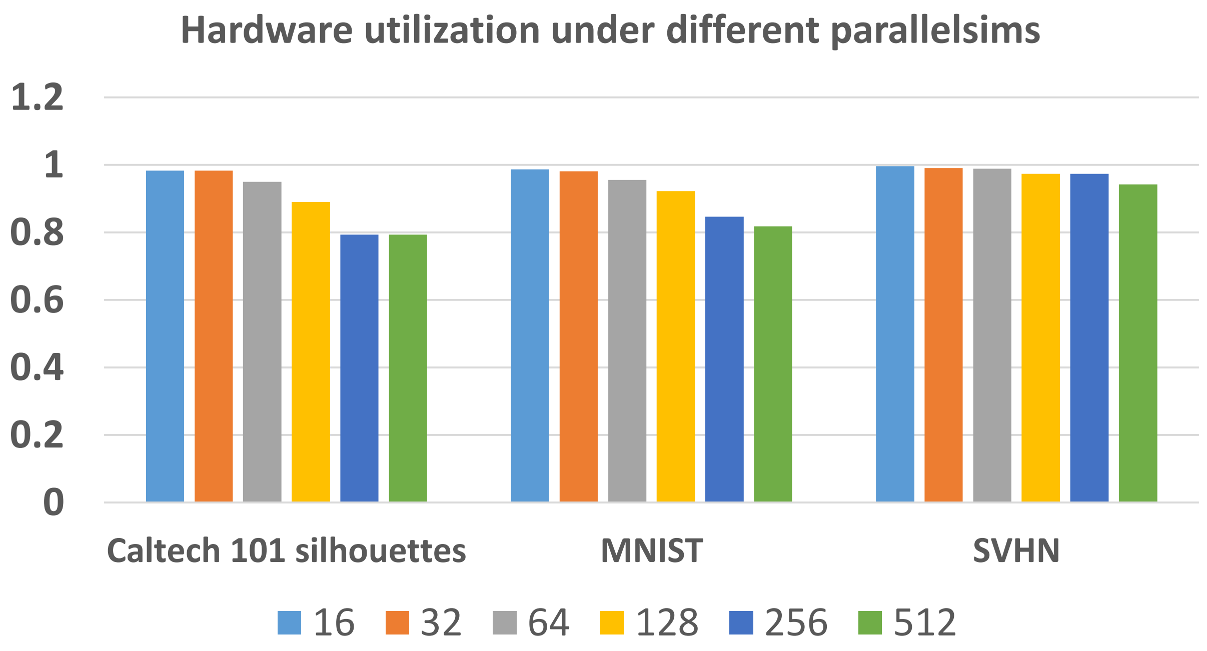
Power consumption (mW) / Energy consumption (mJ)

	Ideal	LRADNN (direct)	LRADNN
Caltech 101 silhouettes	517.76 / 0.15	551.61 / 0.16	487.88 / 0.11
MNIST	517.76 / 0.26	557.98 / 0.30	459.73 / 0.14
SVHN	517.76 / 0.33	561.42 / 0.37	438.37 / 0.22

Scalability for high parallelism of LRADNN

- Not fully utilize the hardware (multipliers) due to the memory alignment

$$\mathrm{U} = \frac{m \times n}{m \times p \times \lceil n/p \rceil}$$



Outline

- Introduction
- Related work
- Motivation of LRADNN
- Low rank approximation (LRA) predictor
- LRADNN: hardware architecture
- Experiment results
- Conclusion

Conclusion

- A general hardware accelerator LRADNN for DNN is proposed
- A time and power-saving accelerator based on LRA
  - 31% ~ 53% energy reduction
  - 22% ~ 43% throughput increase
- A better scheme compared to the existing work

- A better scheme compared to the existing work

	AxNN	LRADNN
Prediction loss	0.5%	< 0.1%
Energy improvement (w/o memory)	1.14x – 1.92x	1.18x – 1.61x
Energy improvement (w/ memory)	N.A.	1.45x – 2.13x

- Thank you

- Q & A