Carnegie Mellon

Energy Aware Computing Research Group

Quantized Deep Neural Networks for Energy Efficient Hardware-based Inference

Authors: Ruizhou Ding, Zeye Liu, R. D. (Shawn) Blanton, Diana Marculescu

Presenter: R. D. (Shawn) Blanton

Department of Electrical and Computer Engineering

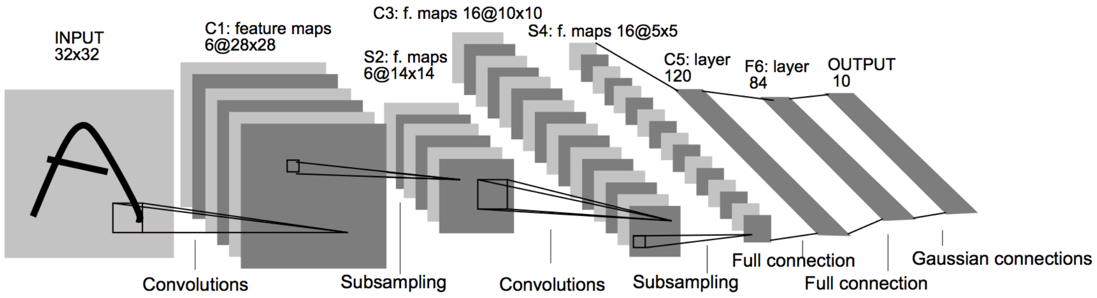
Carnegie Mellon University

rblanton@andrew.cmu.edu

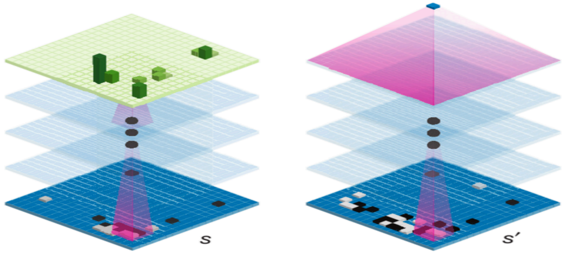
Why energy-efficient Deep Neural Networks

- **Deep neural network (DNN) has been an useful machine learning model for many classification applications**

DNN example: LeNet [LeCun *et al*., 1998]



- **Many classification systems require low power, area and storage**
  - Google's AlphaGo: 13-layer architecture; ~4 million parameters



[Silver *et al*., 2016]

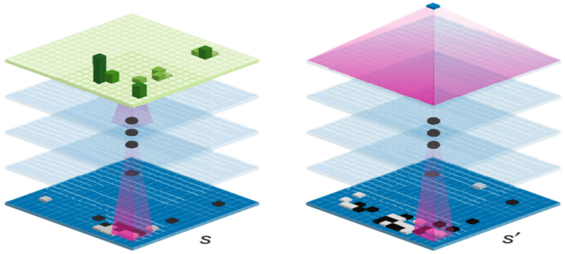
Why energy-efficient Deep Neural Networks

- **Deep neural network (DNN) has been an useful machine learning model for many classification applications**

  DNN example: LeNet [LeCun *et al*., 1998]

Main goal: to reduce energy, area and storage for DNNs' hardware implementation

- **Many classification systems require low power, area and storage**
  - Google's AlphaGo: 13-layer architecture; ~4 million parameters



[Silver *et al*., 2016]

Outline

- **DNN quantization**
- **LightNNs**
  - Approximation
  - Training approach
- **Experiment**
  - Setup
  - Results for accuracy, storage, energy, and area
- **Guideline for model selection**
- **Conclusion**

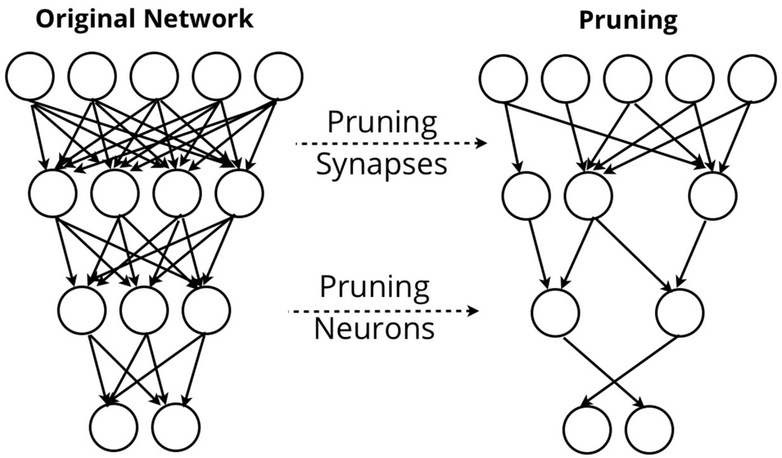
Energy vs. Accuracy

- **DNNs' energy consumption mainly comes from two sources:**
  - Data movement & logic computation

Energy vs. Accuracy

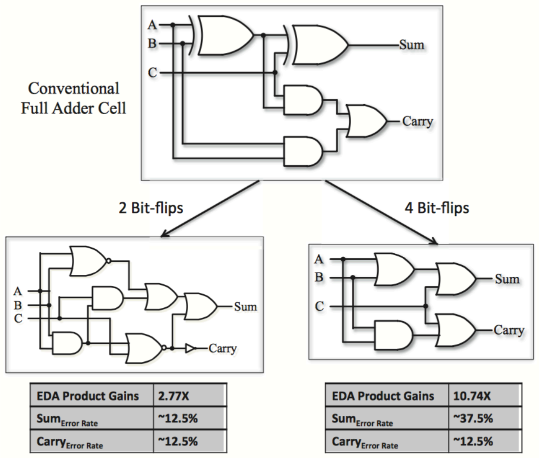
- **DNNs' energy consumption mainly comes from two sources:**
  - Data movement & logic computation

Reducing memory accesses



Deep compression
[H. Song, *et al.*, 2015]

Reducing computation energy



Inexact circuits
[Z. Du, *et al.*, 2014]

DNN quantization

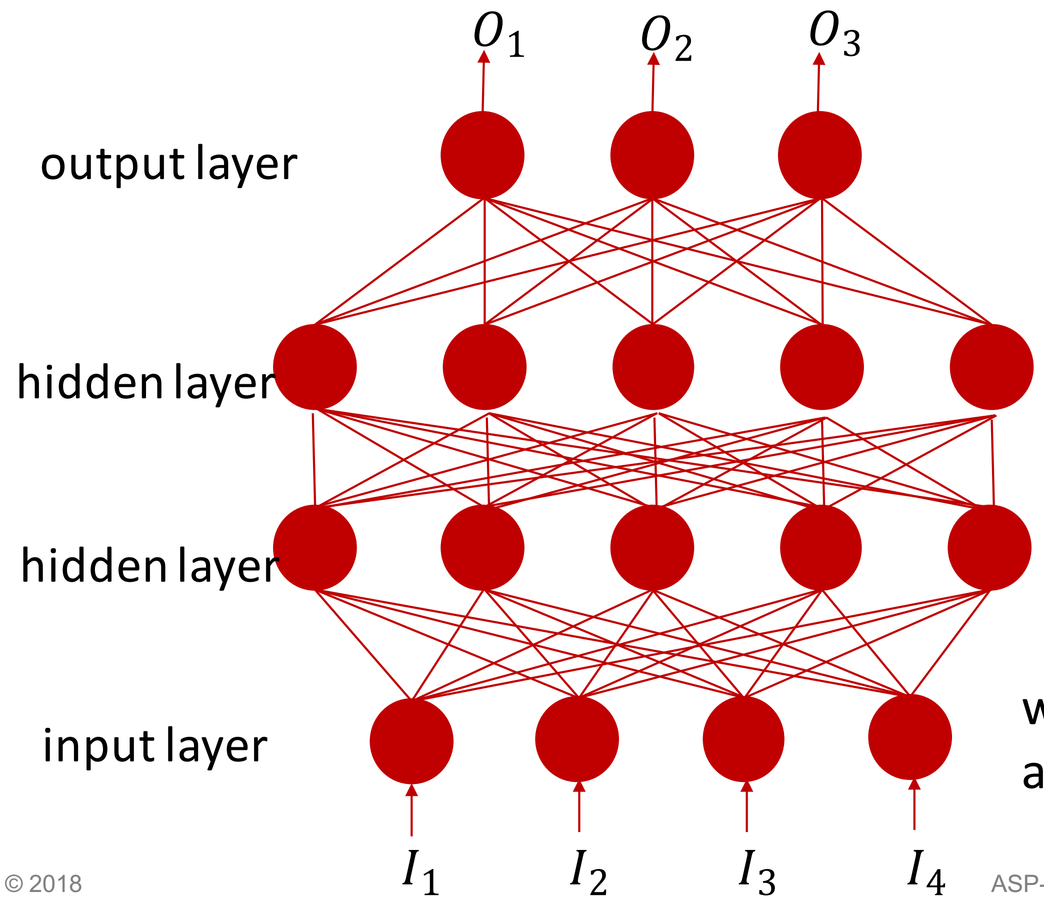
- Reducing data movement
- Reducing logic energy

A unified solution: **DNN quantization**

DNN quantization

- Reducing data movement
- Reducing logic energy

A unified solution: **DNN quantization**



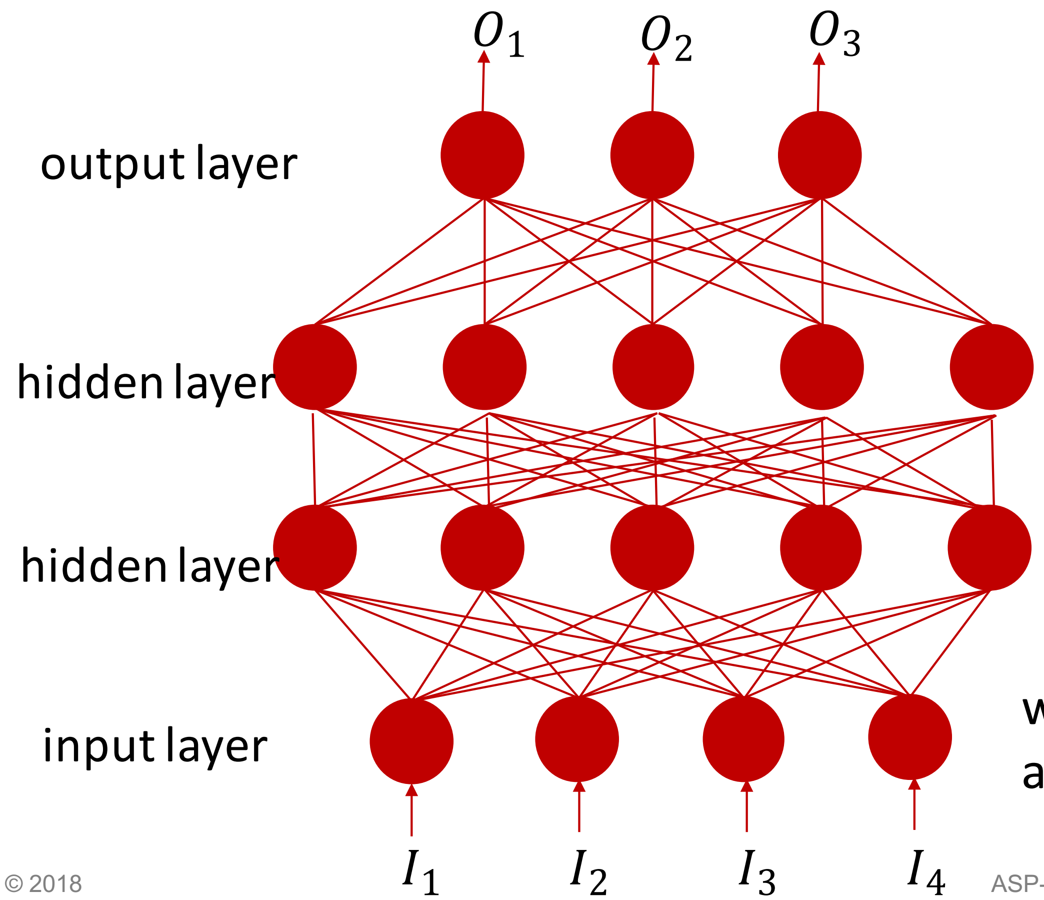
weights: connections between two consecutive layers
activations: output values of each hidden layer

DNN quantization

- Reducing data movement
- Reducing logic energy

A unified solution: **DNN quantization**

Constraining weights and/or activations to discrete values



weights: connections between two consecutive layers
activations: output values of each hidden layer

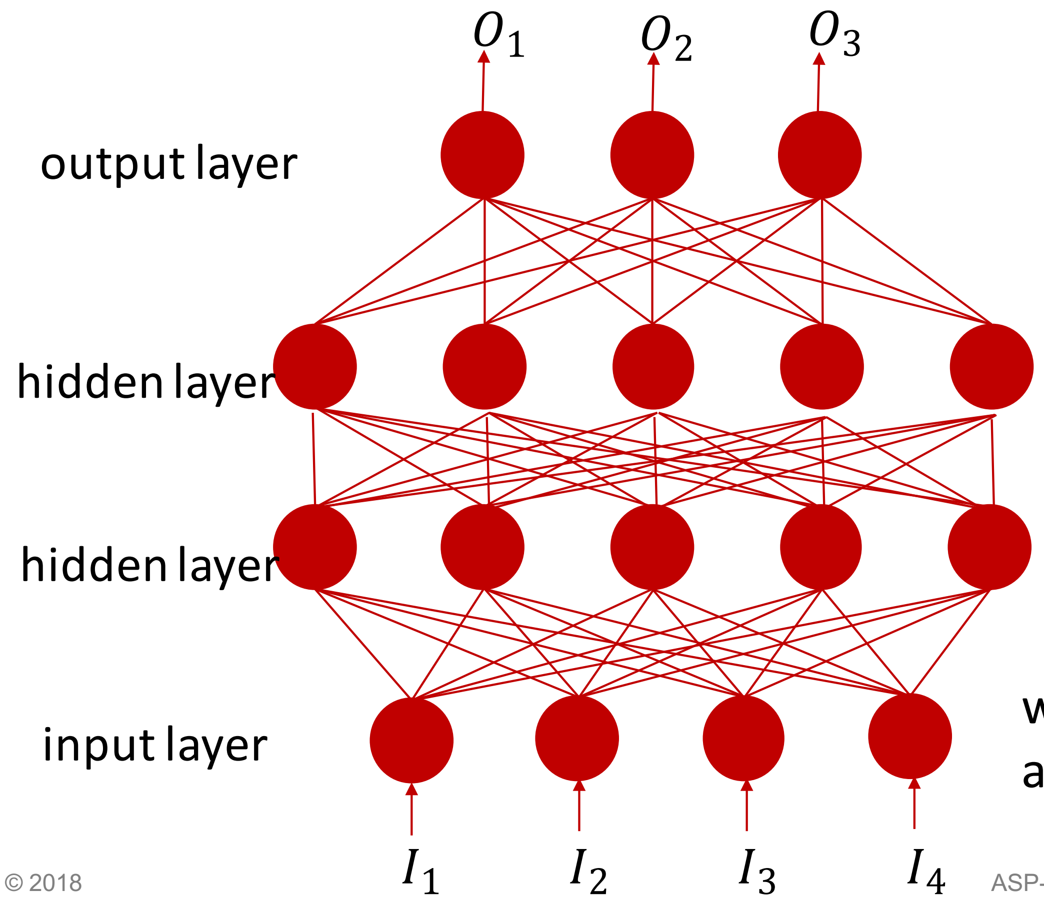
DNN quantization

- Reducing data movement
- Reducing logic energy

A unified solution: **DNN quantization**

Constraining weights and/or activations to discrete values

Examples: binarized DNNs, fixed-point DNNs, ternary-weight DNNs, etc.

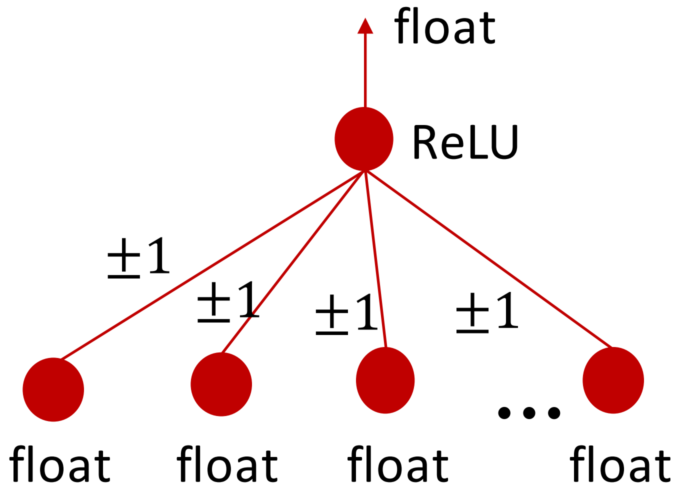


weights: connections between two consecutive layers
activations: output values of each hidden layer

Binarized Neural Networks (BNNs)

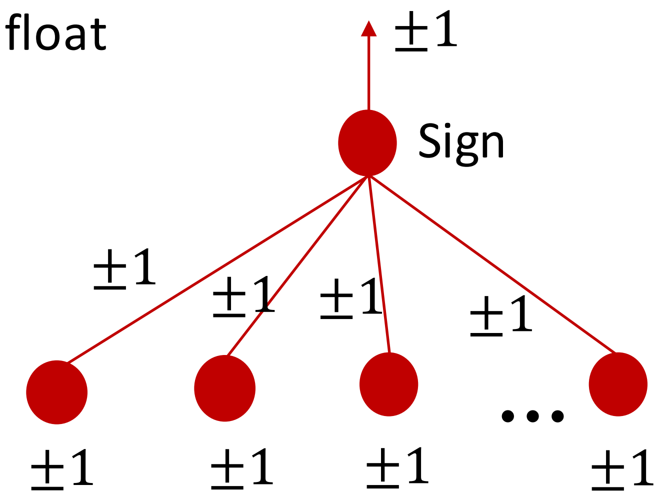
- BNNs reduce both memory and computation, but may lose much accuracy
- Two types of BNNs:
  - Type I: BinaryConnect
    - Weights are ± 1

      [Courbariaux, *et al.*, 2015]



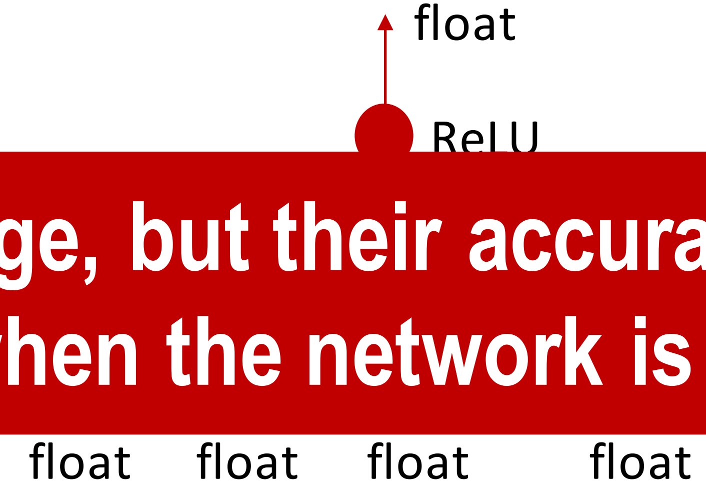
  - Type II: BinaryNet
    - Weights and activations are ± 1

      [Hubara, *et al.*, 2016]



Binarized Neural Networks (BNNs)

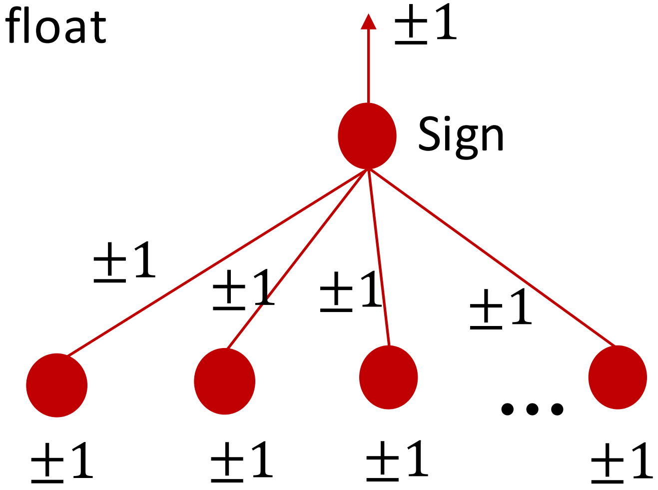
- **BNNs reduce both memory and computation, but may lose much accuracy**
- **Two types of BNNs:**



BNNs reduce storage, but their accuracy degrades especially when the network is small

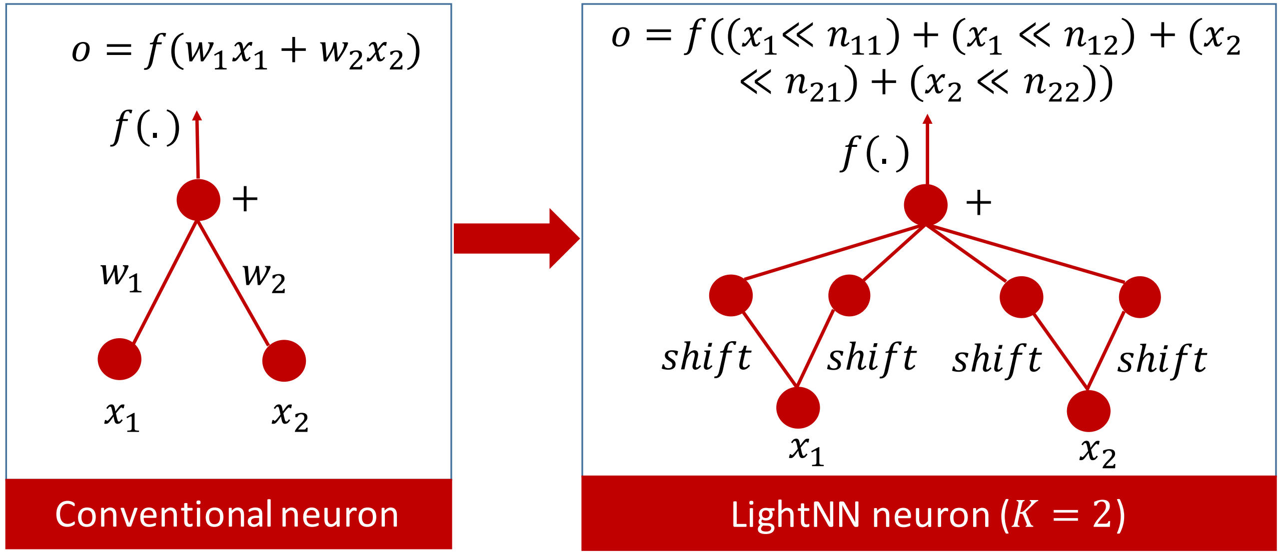
- ◆ Type II: BinaryNet
  - Weights and activations are ±1

  [Hubara, *et al.*, 2016]



LightNNs: constraining number of 1s in weights

- **Replacing multipliers with limited shift and add operators**
  - $w \cdot x = sign(w)(2^{n_1} + 2^{n_2} + \cdots + 2^{n_K}) \cdot x = sign(w)(x \ll n_1 + \cdots + x \ll n_K)$
  - We constrain K to be one or two
  - When $K = 1$, the equivalent multiplier is just a shift
  - When $K = 2$, the equivalent multiplier is two shifts and one add (shown below)



LightNNs: using stochastic rounding

- **Stochastic rounding strategy is used to constrain the floating point weights to have at most K 1s**
  - Stochastic rounding:

    [S. Gupta, *et al.*, 2015]

$$w = \begin{cases} w_h, & with\ prob\ p \\ w_l, & with\ prob\ 1-p \end{cases}$$

w: weight value
w_h: nearest higher legal value
w_l: nearest lower legal value
$p = \frac{w-w_l}{w_h-w_l}$

  - If $K = 1$, then $4 = 100_{(2)}$, ✓ $3 = 11_{(2)}$ ✗
  - 3 can be rounded to 2 or 4 both with 50% probability

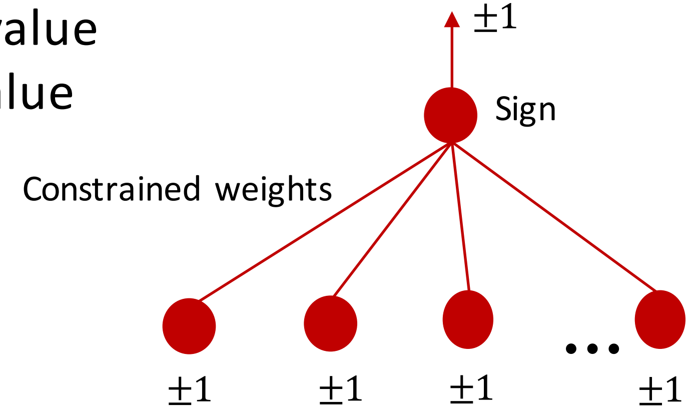
LightNNs: using stochastic rounding

- **Stochastic rounding strategy is used to constrain the floating point weights to have at most K 1s**
  - Stochastic rounding:

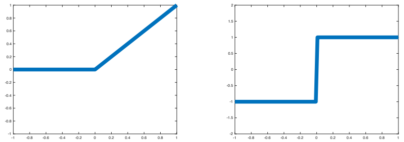
    [S. Gupta, *et al.*, 2015]

$$w = \begin{cases} w_h, & with\ prob\ p \\ w_l, & with\ prob\ 1-p \end{cases}$$

w: weight value
w_h: nearest higher legal value
w_l: nearest lower legal value
$p = \frac{w - w_l}{w_h - w_l}$

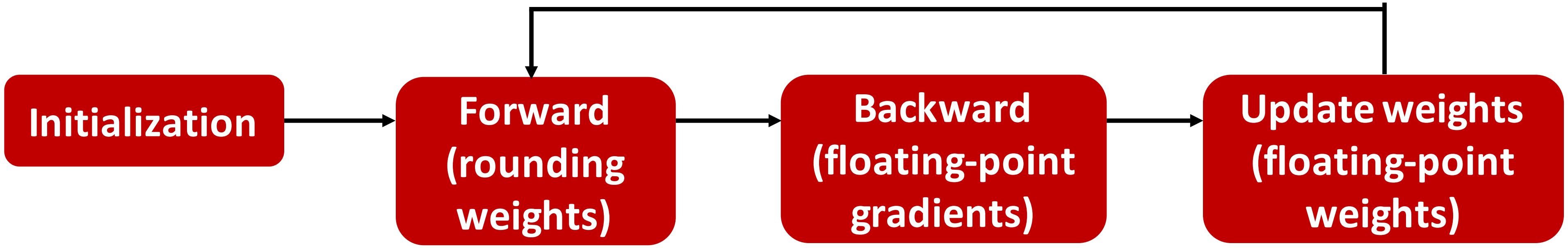


  - If $K = 1$, then $4 = 100_{(2)}$, ✓ $3 = 11_{(2)}$ ✗
  - 3 can be rounded to 2 or 4 both with 50% probability
- **LightNNs can also have either ReLU or Sign activation functions**



Training LightNNs

- **Backpropagation algorithm is modified: in the forward pass, the weights are first rounded and then used for computation**



random $\boldsymbol{w_0}$

$$\boldsymbol{w_c} = round\,(\boldsymbol{w_{t-1}})$$

$\boldsymbol{w_{t-1}}$ is also saved for weights update

$$\frac{\partial l}{\partial \boldsymbol{w_c}}$$

l: loss function

$$\boldsymbol{w_t} = \boldsymbol{w_{t-1}} - \eta \frac{\partial l}{\partial \boldsymbol{w_c}}$$

η: learning rate

Experiment setup

- **Models**

Model	Weights	Activations	Intermediate results
Conventional DNN	floating	ReLU	floating
LightNN-2	$\pm(2^{-m_1}+2^{-m_2})$, $m_1, m_2 = 0,1,\ldots,7$	ReLU	floating
LightNN-1	$\pm 2^{-m}, m = 0,1,\ldots,7$	ReLU	floating
BinaryConnect	+1 or -1	ReLU	floating
LightNN-2-bin	$\pm(2^{-m_1}+2^{-m_2})$, $m_1, m_2 = 0,1,\ldots,7$	Sign	+1 or -1
LightNN-1-bin	$\pm 2^{-m}, m = 0,1,\ldots,7$	Sign	+1 or -1
BinaryNet	+1 or -1	Sign	+1 or -1

[Courbariaux, *et al.*, 2015]

[Hubara, *et al.*, 2016]

Experiment setup

- **Datasets and DNN configurations**

Dataset	**Configuration**	Detail
MNIST	**1-hidden**	One hidden layer with 100 neurons
	2-conv	Two convolution layers and two fully-connected layers
	3-hidden	Three hidden layers each with 4096 neurons
CIFAR-10	**3-conv**	Three convolution layers and one fully-connected layer
	6-conv	Six convolution layers and three fully-connected layers

Red: small configurations
Blue: large configurations

Test error results

- **In most cases, from good to bad: Conventional > LightNNs > BNNs**

		MNIST		
		1-hidden	2-conv	3-hidden
Number of parameters		79,510	431,080	36,818,954
Test error	Conventional	1.72%	0.86%	0.75%
	BinaryConnect	4.10%	4.63%	1.29%
	BinaryNet	6.79%	3.16%	0.96%

Test error results

- **In most cases, from good to bad: Conventional > LightNNs > BNNs**

		MNIST		
		1-hidden	2-conv	3-hidden
Number of parameters		79,510	431,080	36,818,954
Test error	Conventional	1.72%	0.86%	0.75%
	LightNN-2	**1.86%**	**1.29%**	**0.83%**
	LightNN-1	**2.09%**	**2.31%**	**0.89%**
	BinaryConnect	4.10%	4.63%	1.29%
	LightNN-2-bin	**2.94%**	**1.67%**	**0.89%**
	LightNN-1-bin	**3.10%**	**1.86%**	**0.94%**
	BinaryNet	6.79%	3.16%	0.96%

Test error results

- **In most cases, from good to bad: Conventional > LightNNs > BNNs**

		MNIST			CIFAR-10	
		1-hidden	2-conv	3-hidden	3-conv	6-conv
Number of parameters		79,510	431,080	36,818,954	82,208	39,191,690
Test error	Conventional	1.72%	0.86%	0.75%	21.16%	10.94%
	LightNN-2	**1.86%**	**1.29%**	**0.83%**		
	LightNN-1	**2.09%**	**2.31%**	**0.89%**		
	BinaryConnect	4.10%	4.63%	1.29%	43.22%	9.90%
	LightNN-2-bin	**2.94%**	**1.67%**	**0.89%**		
	LightNN-1-bin	**3.10%**	**1.86%**	**0.94%**		
	BinaryNet	6.79%	3.16%	0.96%	73.82%	11.40%

Test error results

- **In most cases, from good to bad: Conventional > LightNNs > BNNs**

		MNIST			CIFAR-10	
		1-hidden	2-conv	3-hidden	3-conv	6-conv
Number of parameters		79,510	431,080	36,818,954	82,208	39,191,690
Test error	Conventional	1.72%	0.86%	0.75%	21.16%	10.94%
	LightNN-2	**1.86%**	**1.29%**	**0.83%**	**24.62%**	**8.84%**
	LightNN-1	**2.09%**	**2.31%**	**0.89%**	**26.11%**	**8.79%**
	BinaryConnect	4.10%	4.63%	1.29%	43.22%	9.90%
	LightNN-2-bin	**2.94%**	**1.67%**	**0.89%**	**32.58%**	**10.12%**
	LightNN-1-bin	**3.10%**	**1.86%**	**0.94%**	**36.56%**	**9.05%**
	BinaryNet	6.79%	3.16%	0.96%	73.82%	11.40%

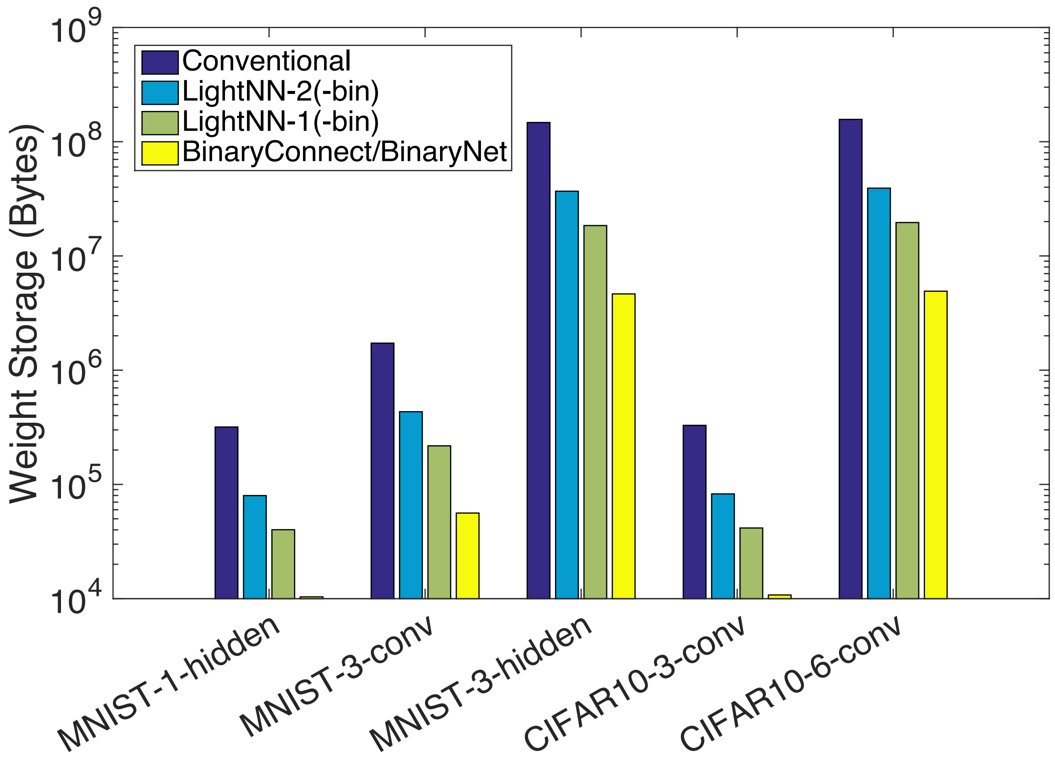
Test error results

- In most cases, from good to bad: Conventional > LightNNs > BNNs

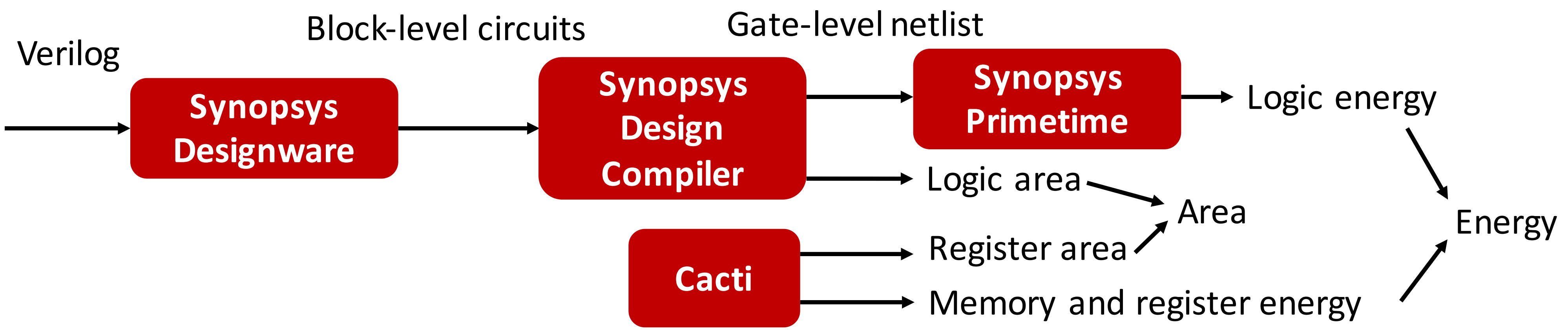
		MNIST			CIFAR-10	
		1-hidden	2-conv	3-hidden	3-conv	6-conv
Number of parameters		79,510	431,080	36,818,954	82,208	39,191,690
Test error	Conventional	1.72%	0.86%	0.75%	21.16%	10.94%
	LightNN-2	**1.86%**	**1.29%**	**0.83%**	**24.62%**	**8.84%**
	LightNN-1	**2.09%**	**2.31%**	**0.89%**	**26.11%**	**8.79%**
	BinaryConnect	4.10%	4.63%	1.29%	43.22%	9.90%
	LightNN-2-bin	**2.94%**	**1.67%**	**0.89%**	**32.58%**	**10.12%**
	LightNN-1-bin	**3.10%**	**1.86%**	**0.94%**	**36.56%**	**9.05%**
	BinaryNet	6.79%	3.16%	0.96%	73.82%	11.40%

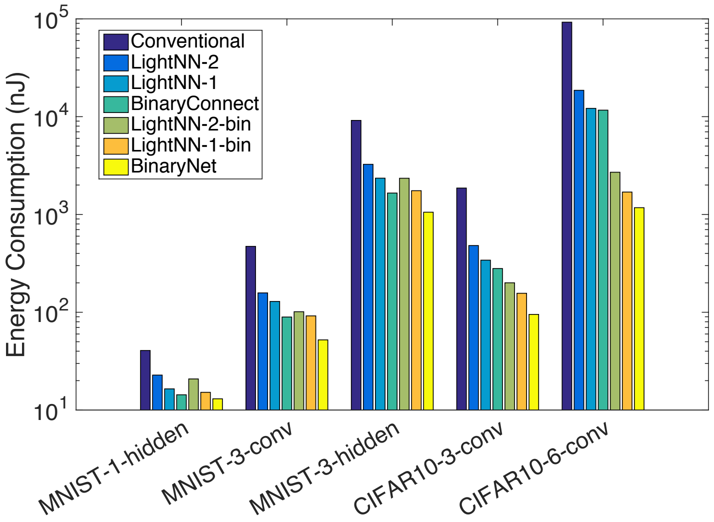
Total storage results

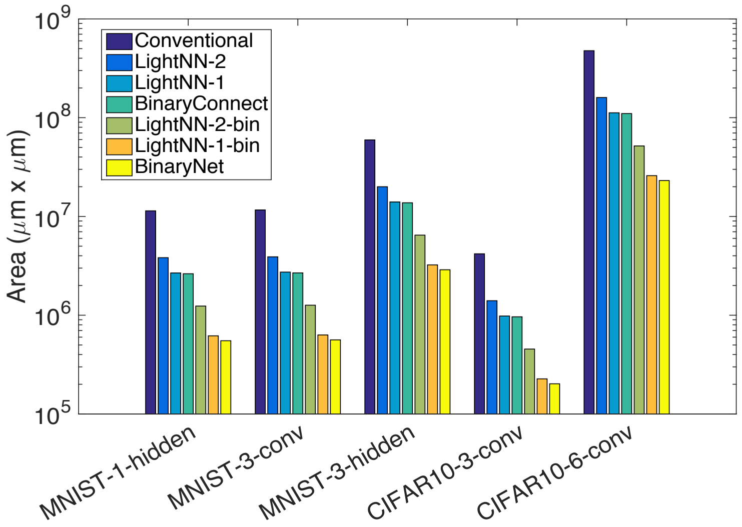
- BNN weight: 1 bit
- LightNN-1 weight: 4 bits
- LightNN-2 weight: 8 bits
- Conventional weight: 32 bits

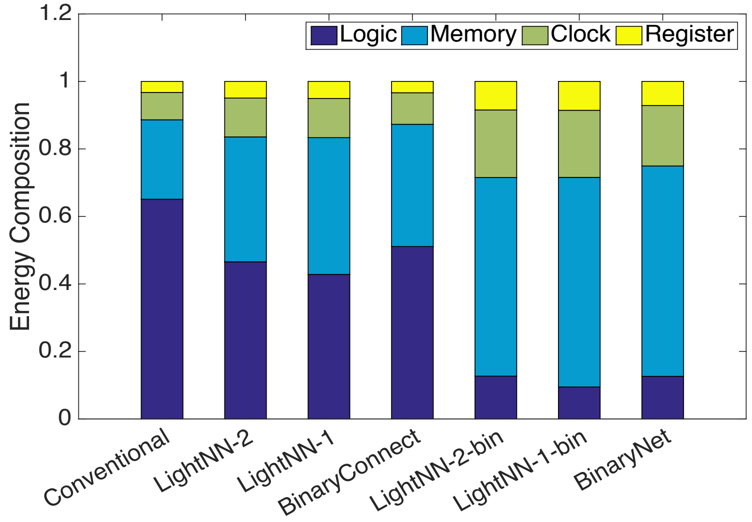


Energy and area synthesis results



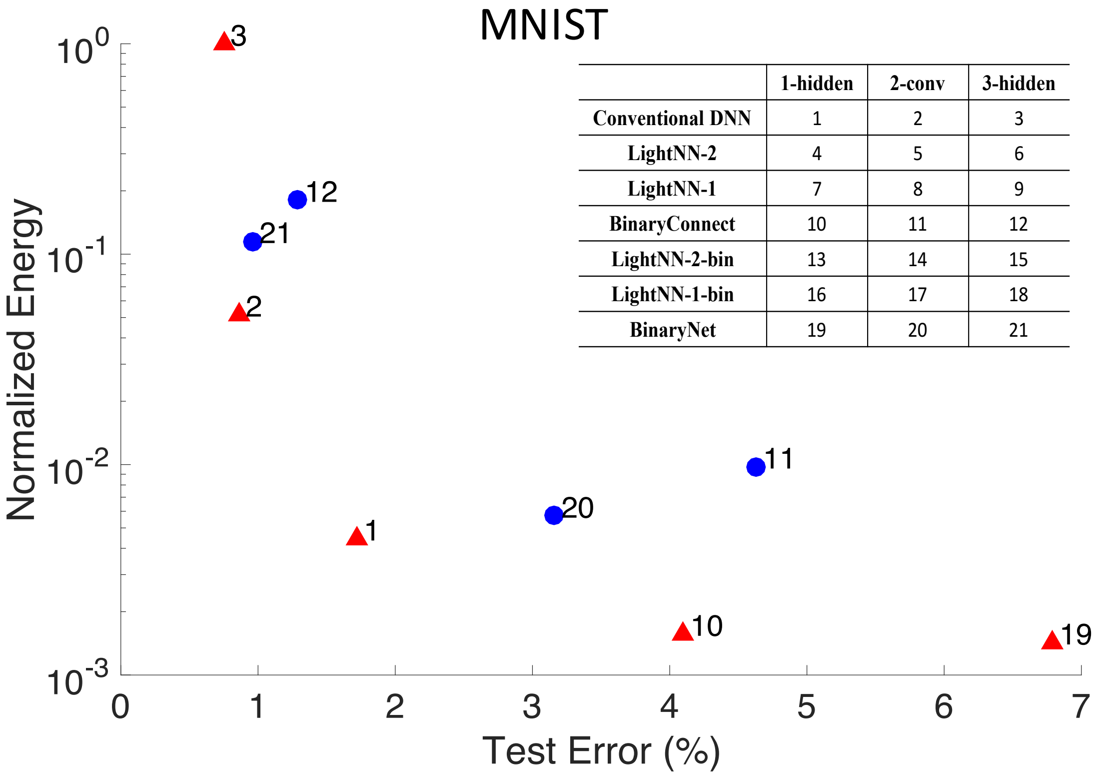




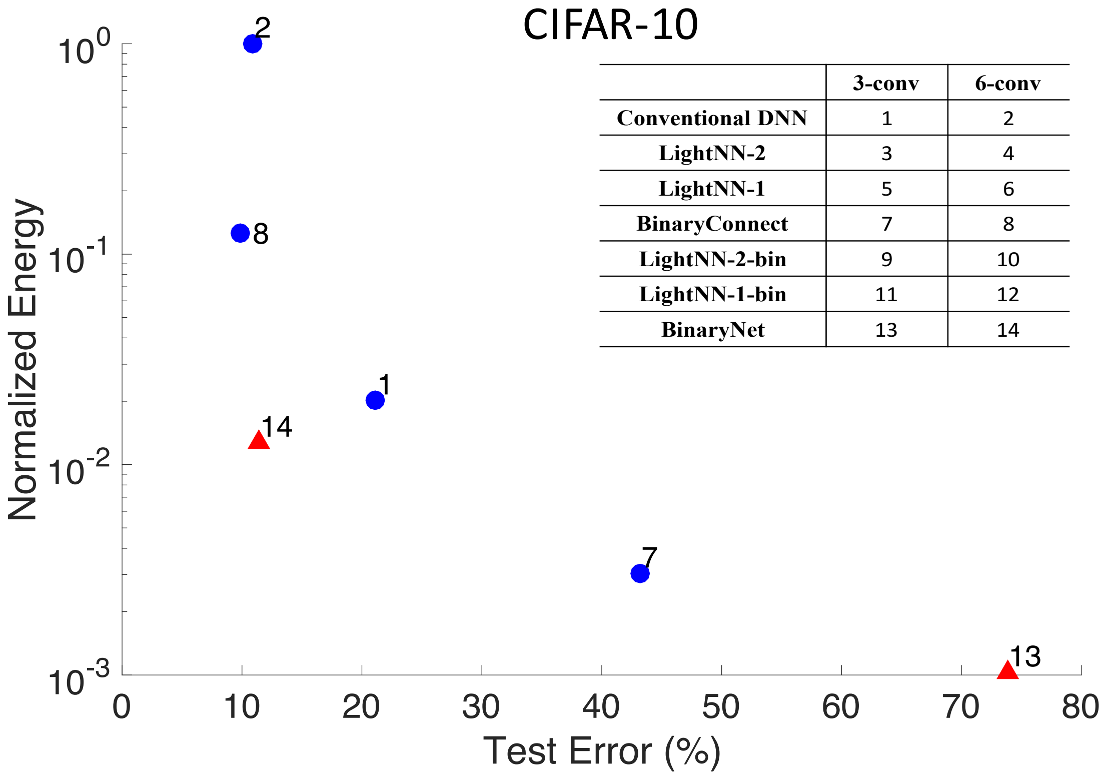


Guideline for model selection

- **Trade-off between accuracy and energy**



	1-hidden	2-conv	3-hidden
Conventional DNN	1	2	3
LightNN-2	4	5	6
LightNN-1	7	8	9
BinaryConnect	10	11	12
LightNN-2-bin	13	14	15
LightNN-1-bin	16	17	18
BinaryNet	19	20	21

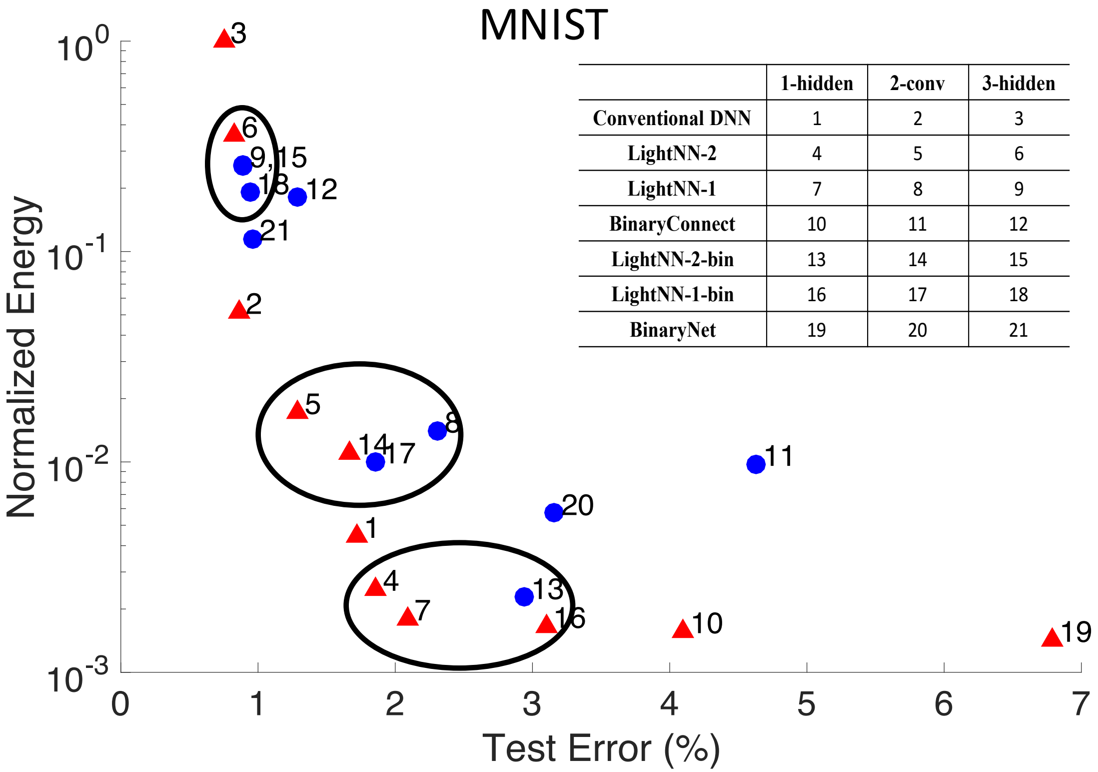


	3-conv	6-conv
Conventional DNN	1	2
LightNN-2	3	4
LightNN-1	5	6
BinaryConnect	7	8
LightNN-2-bin	9	10
LightNN-1-bin	11	12
BinaryNet	13	14

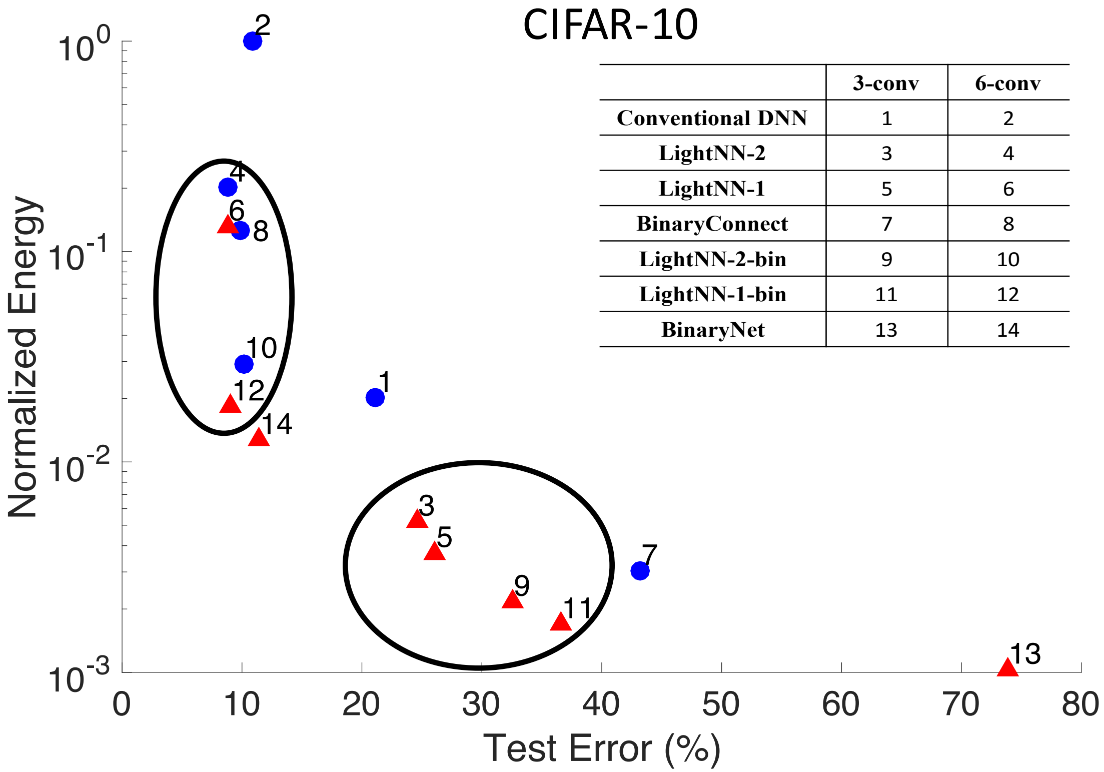
Red triangles are Pareto-optimal

Guideline for model selection

- **Trade-off between accuracy and energy**



	1-hidden	2-conv	3-hidden
Conventional DNN	1	2	3
LightNN-2	4	5	6
LightNN-1	7	8	9
BinaryConnect	10	11	12
LightNN-2-bin	13	14	15
LightNN-1-bin	16	17	18
BinaryNet	19	20	21



	3-conv	6-conv
Conventional DNN	1	2
LightNN-2	3	4
LightNN-1	5	6
BinaryConnect	7	8
LightNN-2-bin	9	10
LightNN-1-bin	11	12
BinaryNet	13	14

Red triangles are Pareto-optimal

Conclusion

- DNN quantization can reduce energy consumption for hardware-based DNN inference
- LightNNs replace the multipliers with more energy-efficient operators
- LightNNs provide more options for hardware designers to select DNN models based on their accuracy and resource constraints

Thank you!