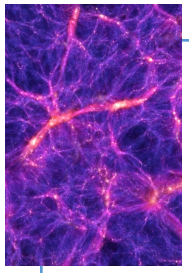
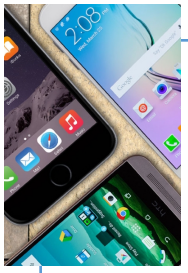
Neural Networks with Few Multiplications

Zhouhan Lin, Matthieu Courbariaux
Roland Memisevic, Yoshua Bengio
MILA, University of Montreal

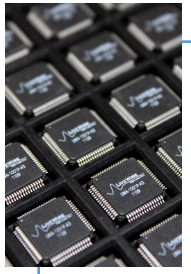
Why we don’t want massive multiplications?



Computationally expensive



Faster computation is likely to be crucial for further progress and for consumer applications on low-power devices.



A multiplier-free network could pave the way to fast, hardware friendly training of neural networks.

Various trials in the past decades…

- Quantize weight values (Kwan & Tang, 1993; Marchesi et al., 1993).
- Quantize states, learning rates, and gradients. (Simard & Graf, 1994)
- Completely Boolean network at test time (Kim & Paris, 2015
- Replace all floating-point multiplications by integer shifts. (Machado et al., 2015)
- Bit-stream networks (Burge et al., 1999) substituting weight connections with logical gates.
- ……

Binarization as regularization?

In many cases, neural networks only need very low precision

Low precision

Stochasticity comes with benefits

- Dropout, Blackout. [4][5]
- Noisy Gradients [3]
- Noisy activation functions [2]

stochasticity

Stochastic binarization?

Can we take advantage of the impreciseness of a binarization process so that we can have reduced computation load and extra regularization at the same time?

Our approach

Binarize weight values

- *BinaryConnect[*Courbariaux, et al., 2015*]* and *TernaryConnect*
- Binarize weights in the forward/backward propagations, but store a full-precision version of them in the backend.

Quantize backprop

- *Exponential quantization*
- Employ quantization of the representations while computing down-flowing error signals in the backward pass.

Our approach

Binarize weight values

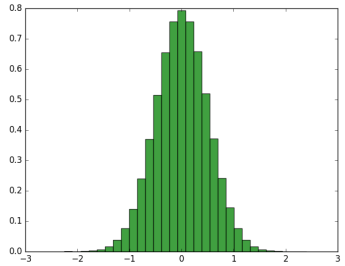
- *BinaryConnect[*Courbariaux, et al., 2015*]* and *TernaryConnect*
- Binarize weights in the forward/backward propagations, but store a full-precision version of them in the backend.

Quantize backprop

- *Exponential quantization*
- Employ quantization of the representations while computing down-flowing error signals in the backward pass.

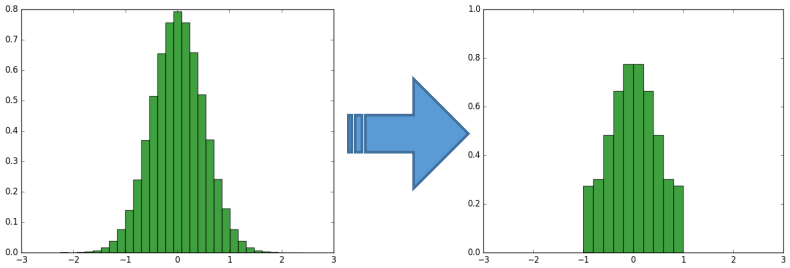
*

Binarize Weight Values



Original weight histogram

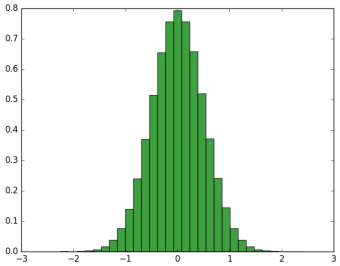
Binarize Weight Values



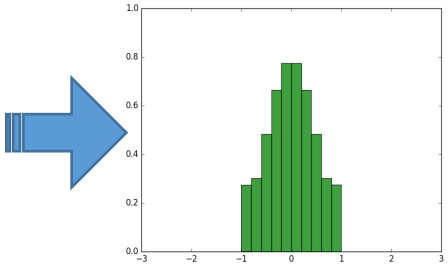
Original weight histogram

weight clipping

Binarize Weight Values

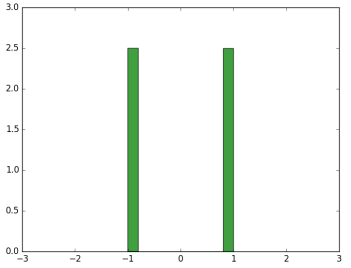


Original weight histogram

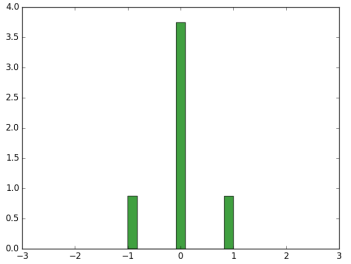


weight clipping

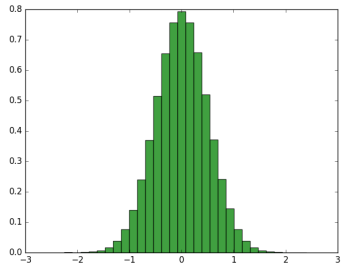
BinaryConnect



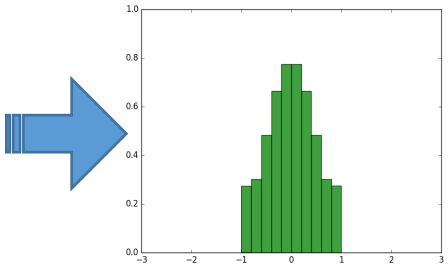
TernaryConnect



Binarize Weight Values

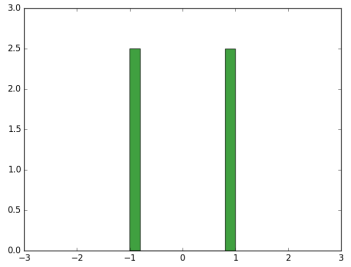


Original weight histogram



weight clipping

BinaryConnect



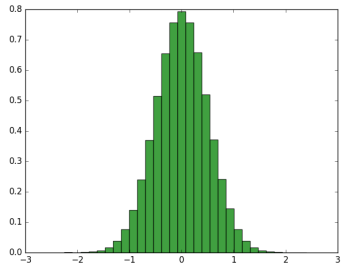
Stochastic

- $P(W_{ij} = 1) = \frac{w_{ij}+1}{2}$
- $P(W_{ij} = -1) = 1 - P(W_{ij} = 1)$

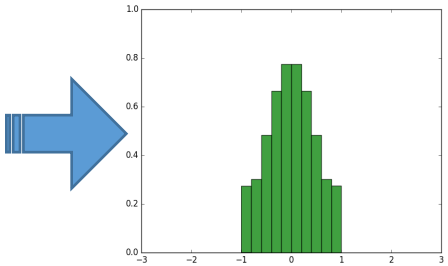
Deterministic

- $W_{ij} = \begin{cases} 1 & w_{ij} > 0 \\ -1 & otherwise \end{cases}$

Binarize Weight Values

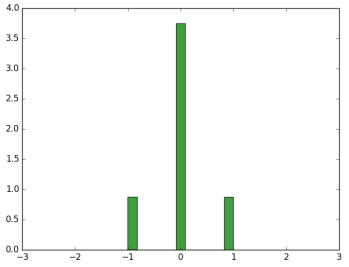


Original weight histogram



weight clipping

TernaryConnect



Stochastic

- If $w_{ij} > 0$:
  - $P(W_{ij} = 1) = w_{ij}$
  - $P(W_{ij} = 0) = 1 - w_{ij}$
- Else:
  - $P(W_{ij} = -1) = -w_{ij}$
  - $P(W_{ij} = 0) = 1 + w_{ij}$

Deterministic

- $W_{ij} = \begin{cases} 1 & w_{ij} > 0.5 \\ 0 & -0.5 < w_{ij} \leq 0.5 \\ -1 & w_{ij} \leq -0.5 \end{cases}$

Our approach

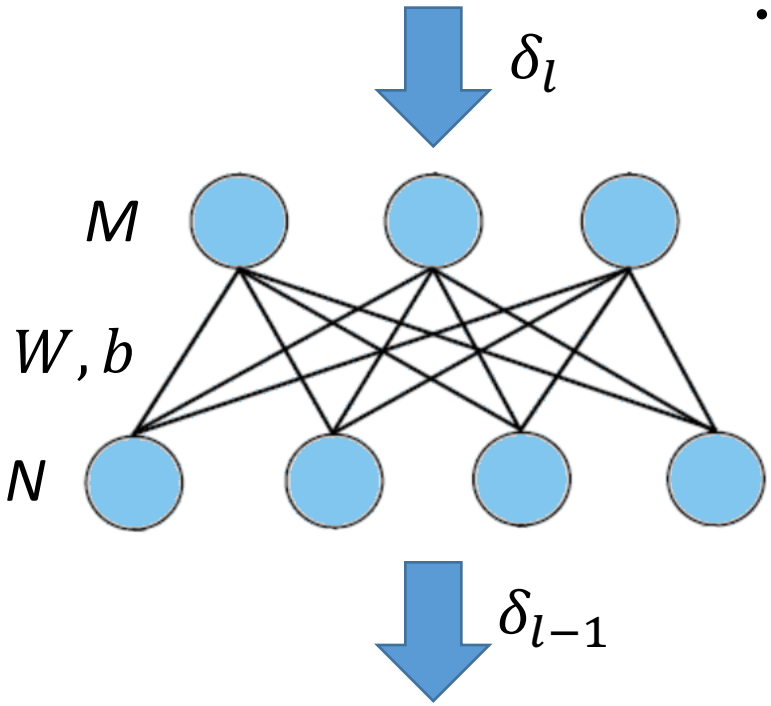
Binarize weight values

- *BinaryConnect[*Courbariaux, et al., 2015*]* and *TernaryConnect*
- Binarize weights in the forward/backward propagations, but store a full-precision version of them in the backend.

Quantize backprop

- *Exponential quantization*
- Employ quantization of the representations while computing down-flowing error signals in the backward pass.

Quantized Backprop



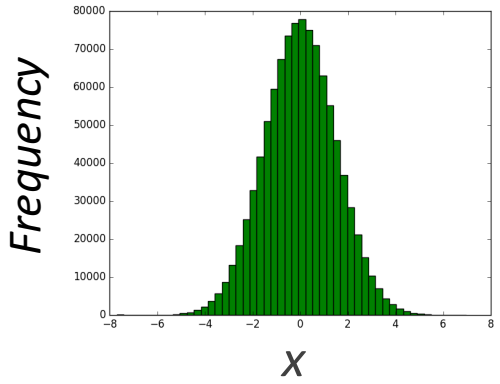
- Consider the update you need to take in the backward pass of a given layer, with *N* input units and *M* outputs:

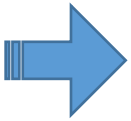
$$\Delta W = [\eta \delta_l \circ h'(Wx + b)] \cdot x^T \quad (1)$$

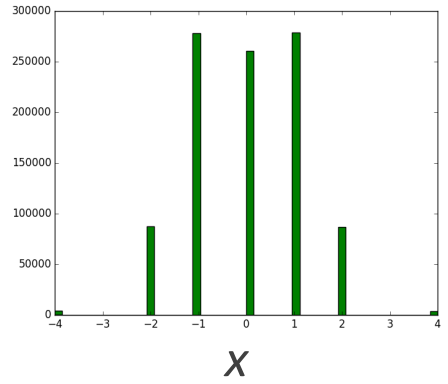
$$\Delta b = \eta \delta_l \circ h'(Wx + b) \quad (2)$$

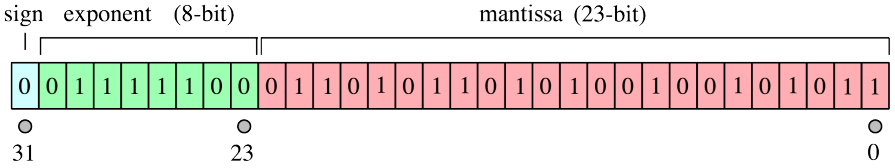
$$\delta_{l-1} = [W^T \cdot \delta_l] \circ h'(Wx + b) \quad (3)$$

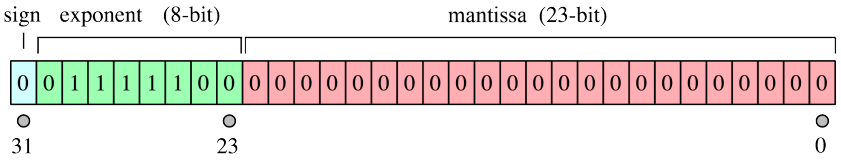
Exponential Quantization



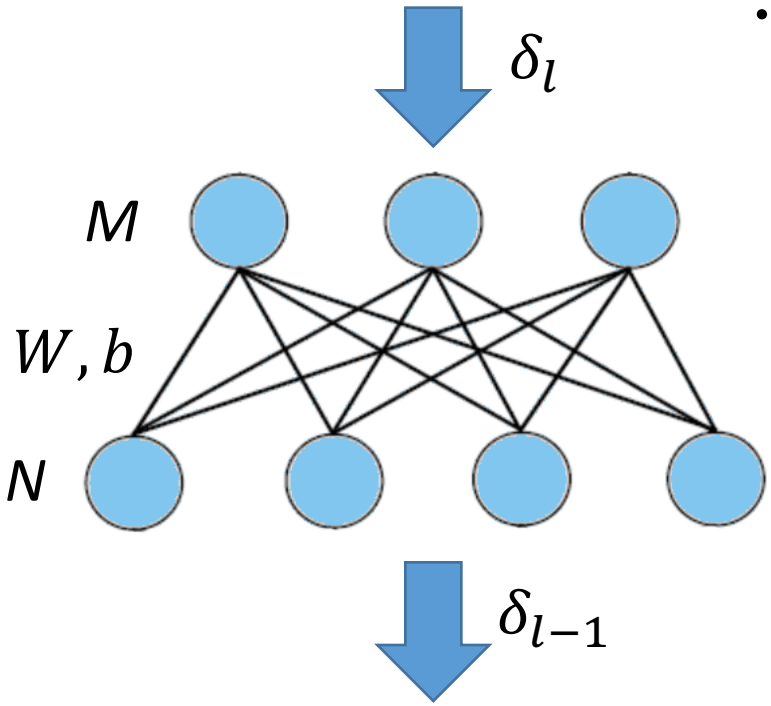








Quantized Backprop



- Consider the update you need to take in the backward pass of a given layer, with *N* input units and *M* outputs:

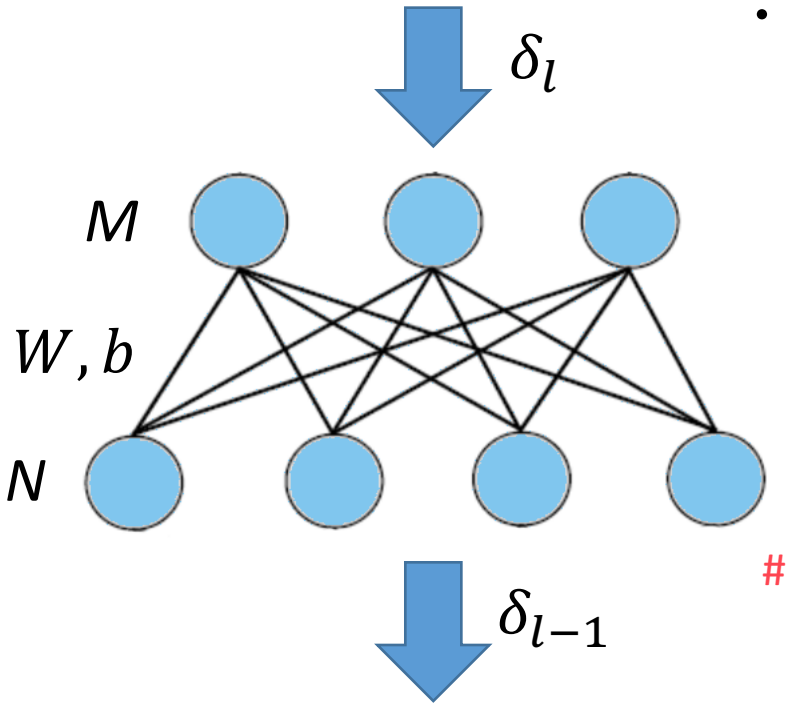
$$\Delta W = [\eta \delta_l \circ h'(Wx + b)] \cdot x^T \quad (1)$$

$$\Delta b = \eta \delta_l \circ h'(Wx + b) \quad (2)$$

$$\delta_{l-1} = [W^T \cdot \delta_l] \circ h'(Wx + b) \quad (3)$$

  - It is hard to bound the values of h', thus make it hard to decide how many bits it will need.
  - We choose to quantize x.

Quantized Backprop



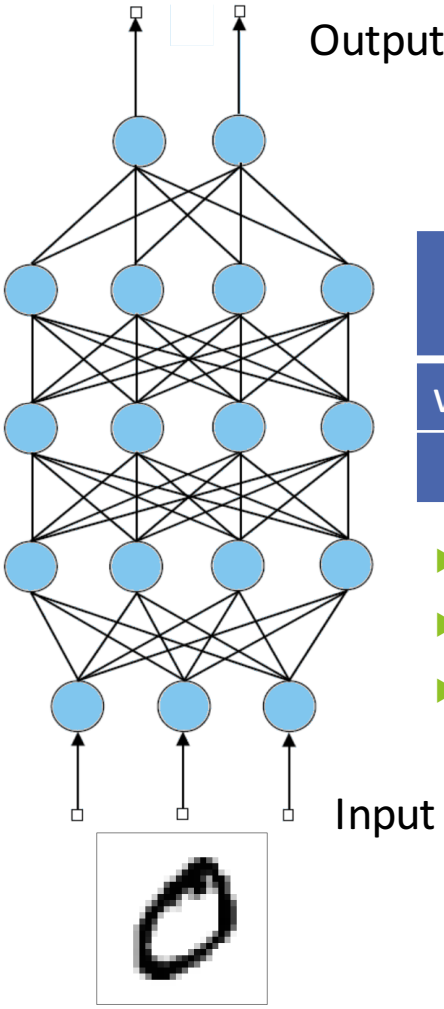
- Consider the update you need to take in the backward pass of a given layer, with *N* input units and *M* outputs:

$$\Delta W = [\eta \delta_l \circ h'(Wx + b)] \cdot x^T \quad (1)$$
$$\Delta b = \eta \delta_l \circ h'(Wx + b) \quad (2)$$
$$\delta_{l-1} = [W^T \cdot \delta_l] \circ h'(Wx + b) \quad (3)$$

multiplications: 2M M

- **3M** multiplications in total
- A standard backprop would have to compute all the multiplications, requiring **2MN + 3M** multiplications.

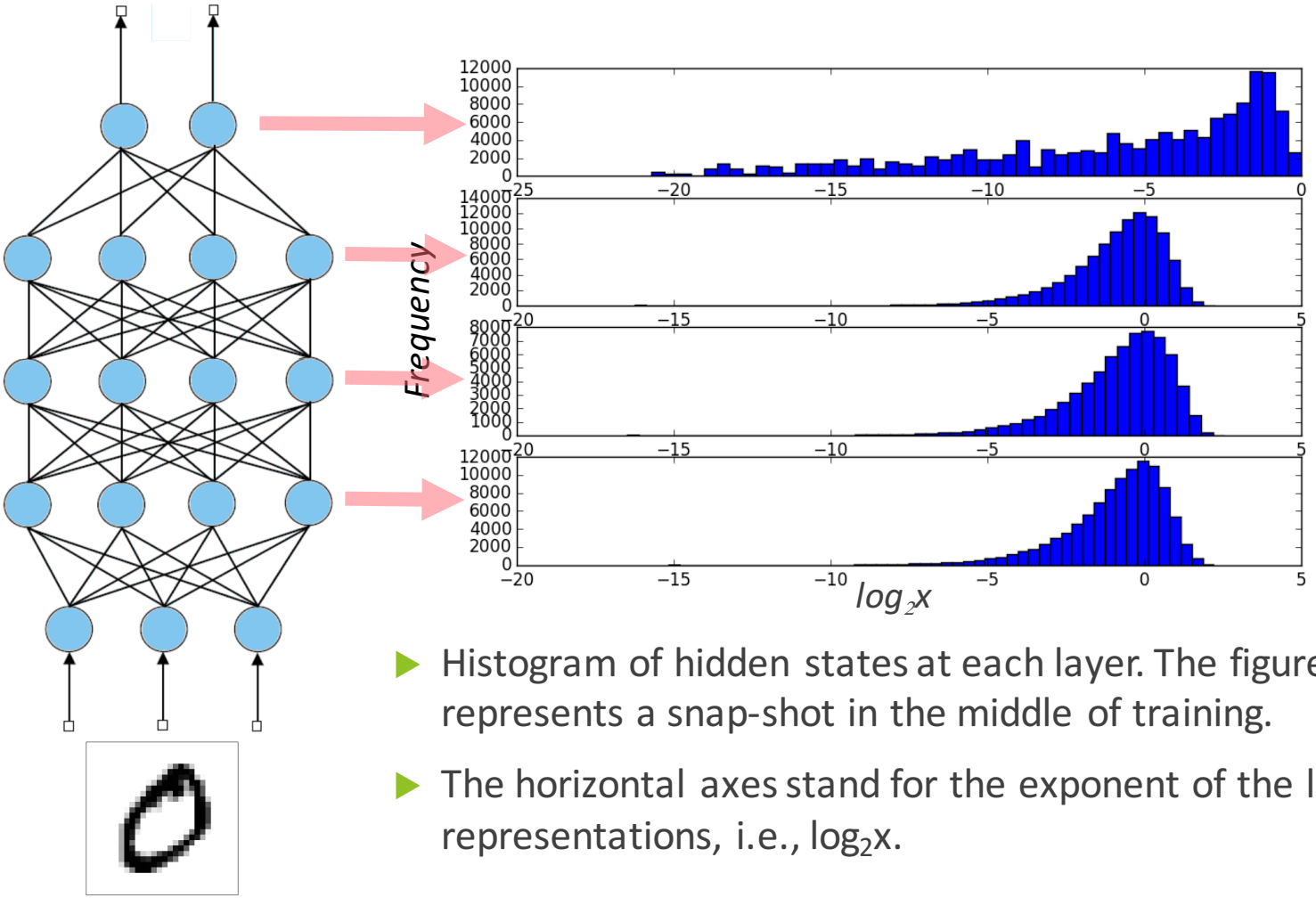
How many multiplications saved?



	Full precision	Ternary connect + Quantized backprop	ratio
without BN	1.7480×10^9	1.8492×10^6	0.001058
with BN	1.7535×10^9	7.4245×10^6	0.004234

- MLP with ReLU, 4 layers (784-1024-1024-1024-10)
- Assume that standard SGD are used as the optimization algorithm.
- BN stands for Batch Normalization

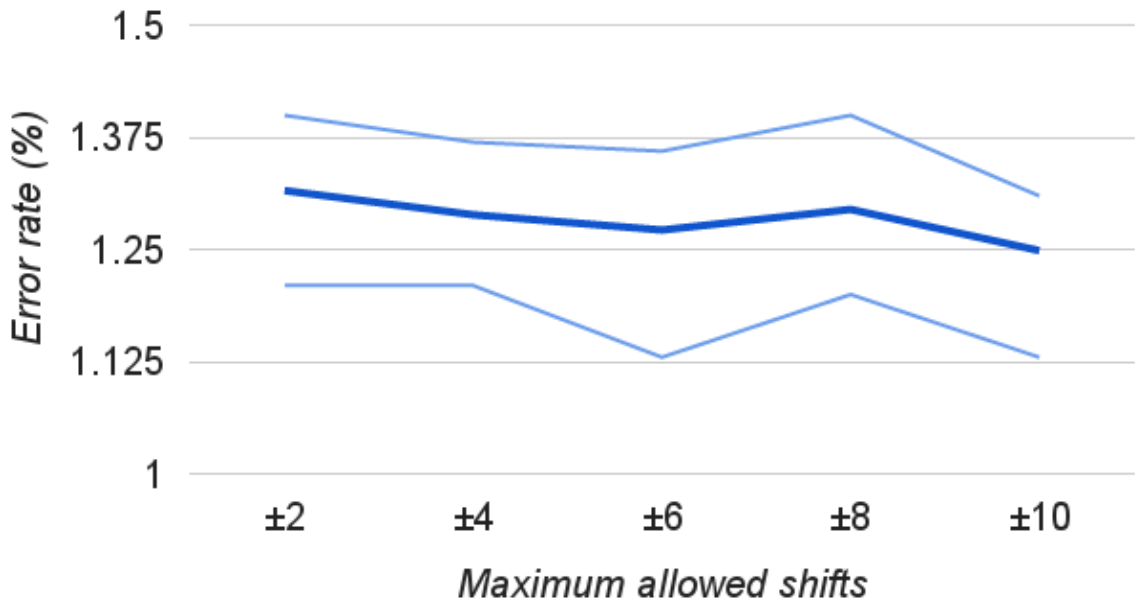
Range of Hidden Representations



- Histogram of hidden states at each layer. The figure represents a snap-shot in the middle of training.
- The horizontal axes stand for the exponent of the layers' representations, i.e., $\log_2 x$.

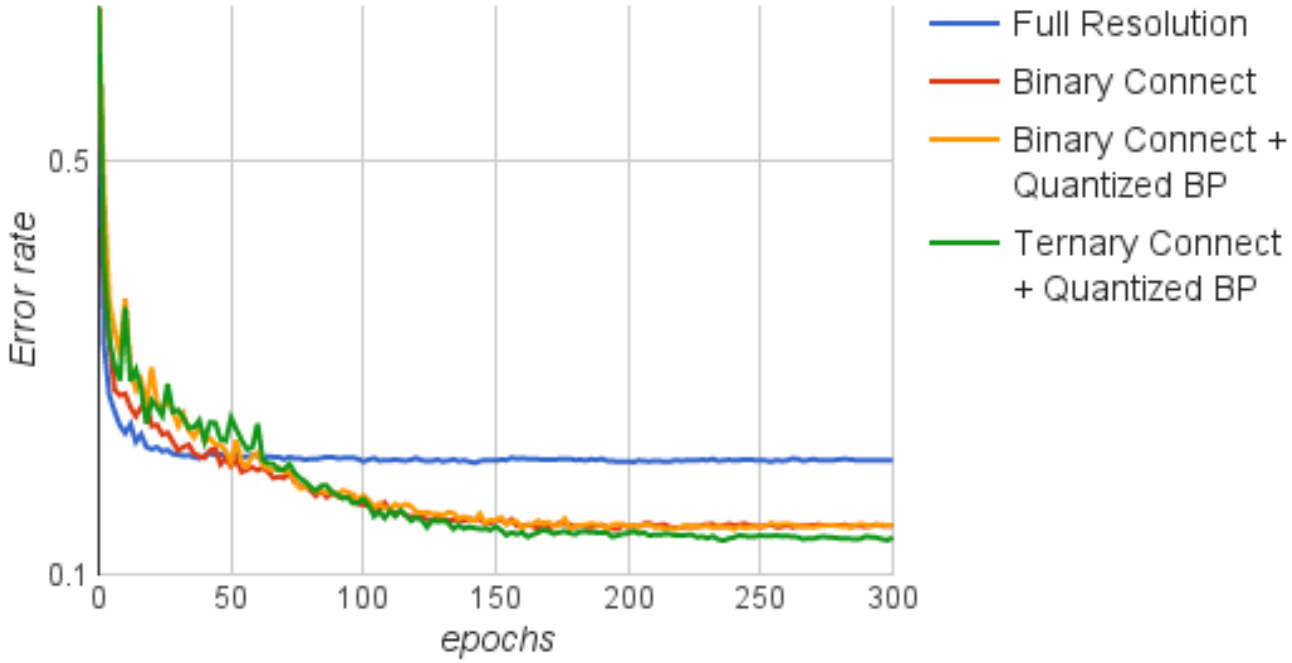
The Effect of Limiting the Range of Exponent

- Constraining the maximum allowed amount of bit shifts in quantized backprop.



General Performance

	Full precision	Binary connect	Binary connect + Quantized backprop	Ternary connect + Quantized backprop
MNIST	1.33%	1.23%	1.29%	1.15%
CIFAR10	15.64%	12.04%	12.08%	12.01%
SVHN	2.85%	2.47%	2.48%	2.42%



Related Works & Recent Advances

- Binarize both weights and activations [Courbariaux, et al., 2016]
- Exponential quantization over the forward pass.
- Larger, more serious datasets.
- Actual dedicated hardware realization.

Any questions?

References, Code & More: