ECE 598NSG/498NSU
Deep Learning in Hardware
Fall 2020

Finite-precision DNNs (Inference)

Naresh Shanbhag
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign

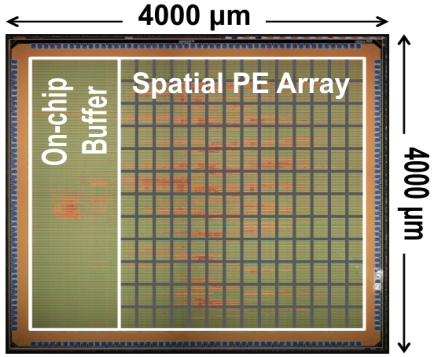
http://shanbhag.ece.uiuc.edu

Today

- fixed-point DNNs in literature
- evaluating impact of quantization on DNN accuracy, complexity and storage

Machine Learning in Reduced Precision

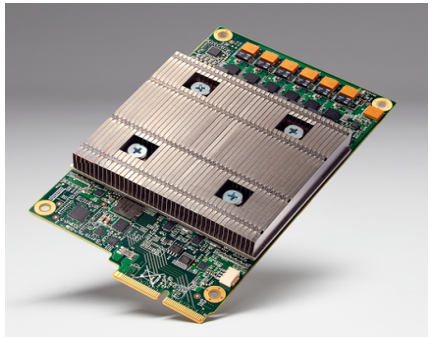
MIT's Eyeriss



[ISSCC'16]

16b fixed-point
(inference)

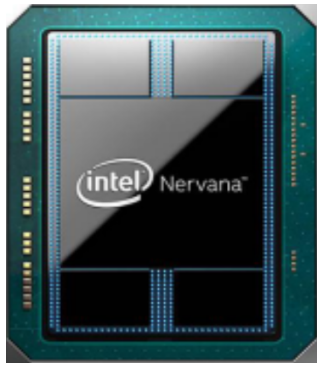
Google's TPU



[ISCA'17]

8b fixed-point
(inference)
16b floating-point
(training)

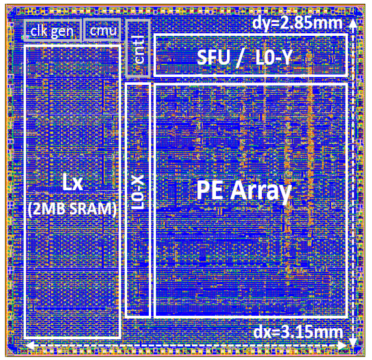
Intel's NNP



[NIPS'17]

16b flexpoint
(training)

IBM's AI Core



[VLSI'18]

16b floating-point
(training)

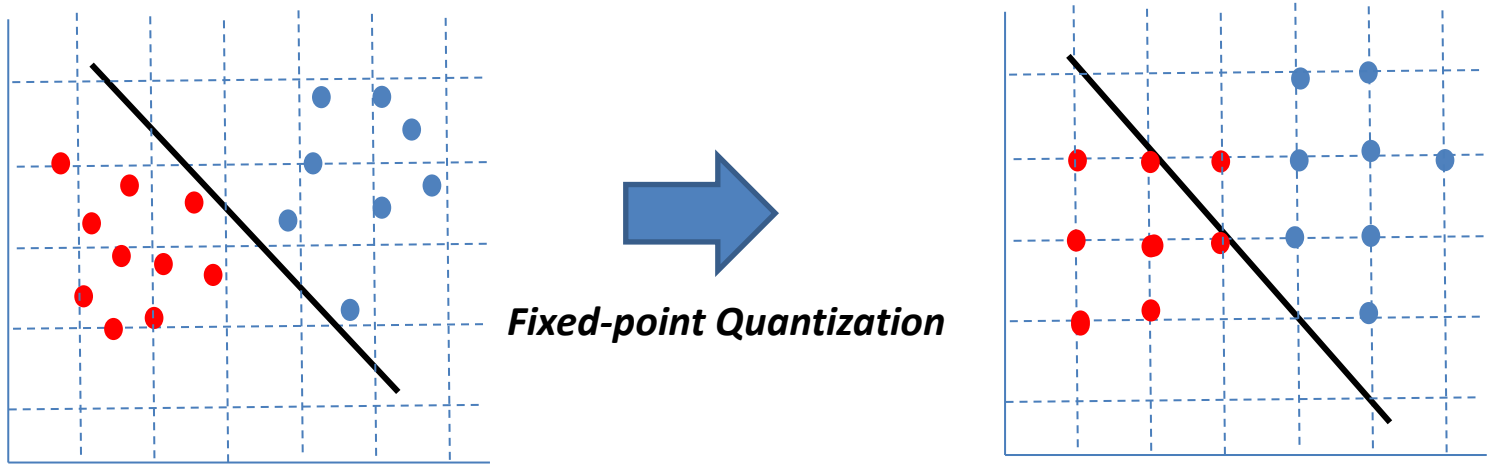
Are these the minimum precisions required? - **NO**

Can minimum precision requirements be determined analytically? - **YES**

Current Approaches

no theoretical guarantees on accuracy heavily simulation-based

- Stochastic Rounding during training [Gupta, ICML'15 – Hubara, NIPS'16]
  → Difficulty of training in a discrete space
- Trial-and-error approach [Sung, SiPS'14]
  → Exhaustive search is expensive
- SQNR based precision allocation [Lin, ICML'16]
  → Lack of precision/accuracy understanding

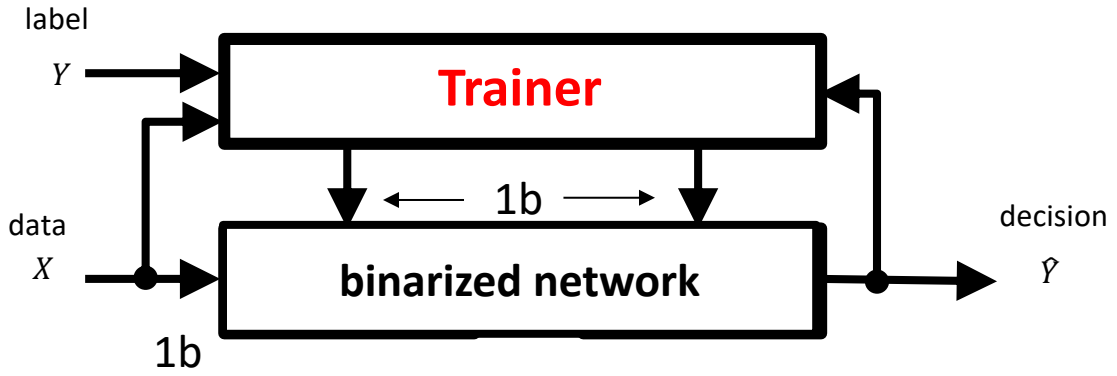


Binarized Neural Networks

Binarized Neural Networks: Training Neural Networks with Weights and Activations Constrained to +1 or −1

Matthieu Courbariaux*[1] MATTHIEU.COURBARIAUX@GMAIL.COM
Itay Hubara*[2] ITAYHUBARA@GMAIL.COM
Daniel Soudry[3] DANIEL.SOUDRY@GMAIL.COM
Ran El-Yaniv[2] RANI@CS.TECHNION.AC.IL
Yoshua Bengio[1,4] YOSHUA.UMONTREAL@GMAIL.COM

[arxiv, March 2016]

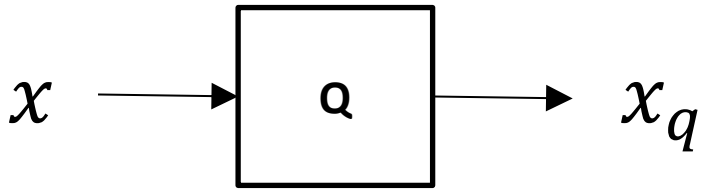


- weights and activations are both restricted to $\{\pm 1\}$
- train network to find best set of binary weights
- note: training is done with floating-point computations

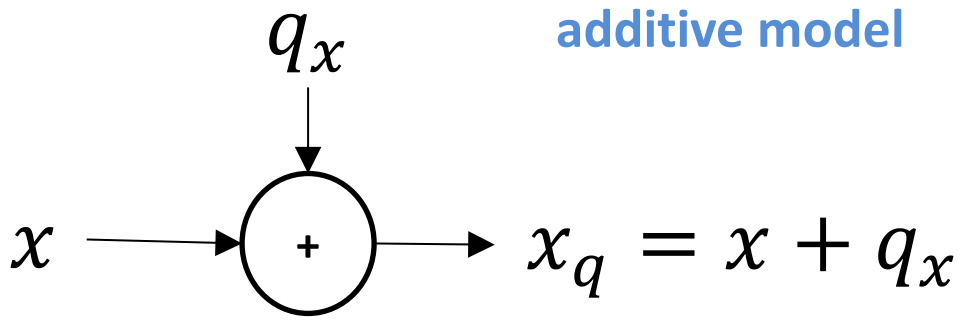
Quantization Noise Analysis

Fixed-point Model

quantizer symbol



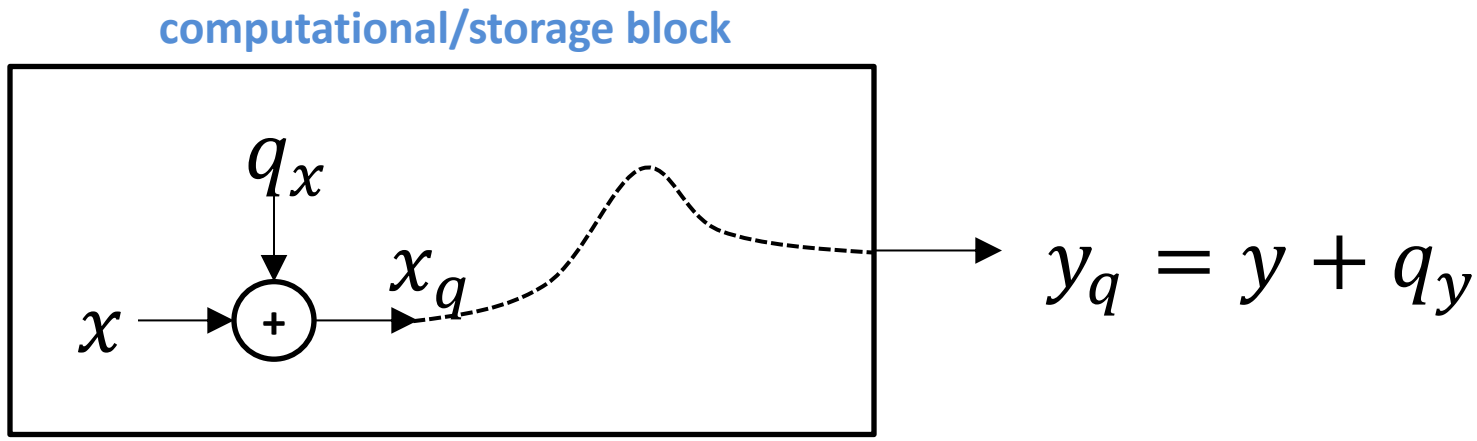
additive model



$$SQNR_x = 10 \log_{10} \left[\frac{\sigma_x^2}{\sigma_{q_x}^2} \right]$$

- x : floating-point or analog valued or infinite-precision scalar data
- q_x: quantization noise in x
- x_q: quantized value of x
- additive model: q_x is assumed to be independent of x
- $SQNR$: signal-to-quantization noise ratio → accuracy measure

Quantization Noise Analysis

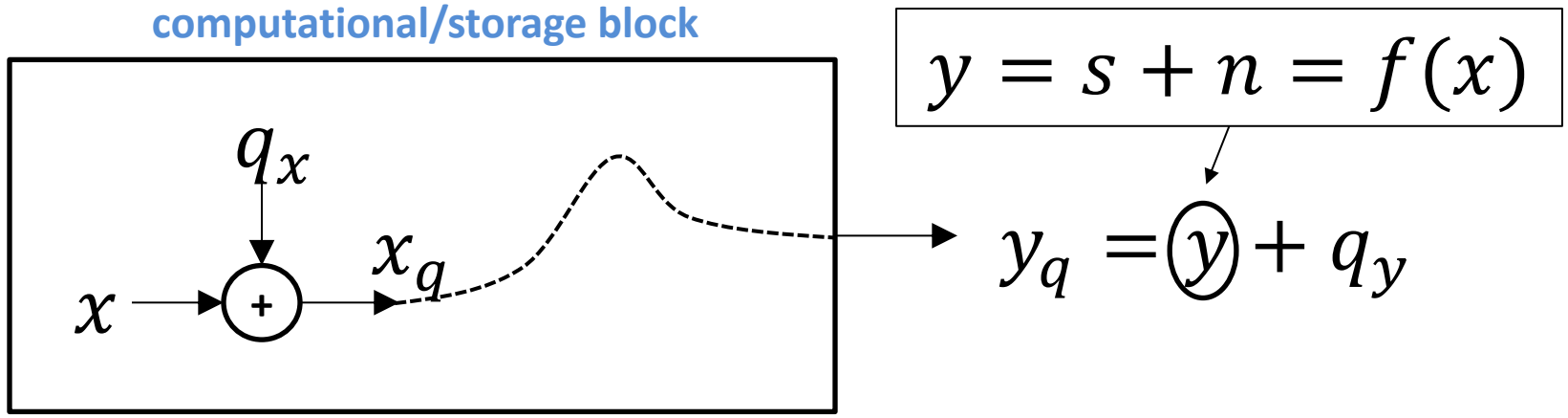


$$SQNR_y = 10 \log_{10} \left[\frac{\sigma_y^2}{\sigma_{q_y}^2} \right]$$

$q_y = f(q_x) = \frac{\partial y}{\partial x} q_x$ (first-order term in Taylor series expansion of $f(x)$)

- determine $\sigma_{q_y}^2$ as a function of $\sigma_{q_x}^2$: need to find $q_y = f(q_x)$
- contribution of q_x to output quantization noise q_y: $\sigma_{q(x \to y)}^2$
- sum all such contributions → total output quantization noise

True SQNR Requirements – Application Dependent



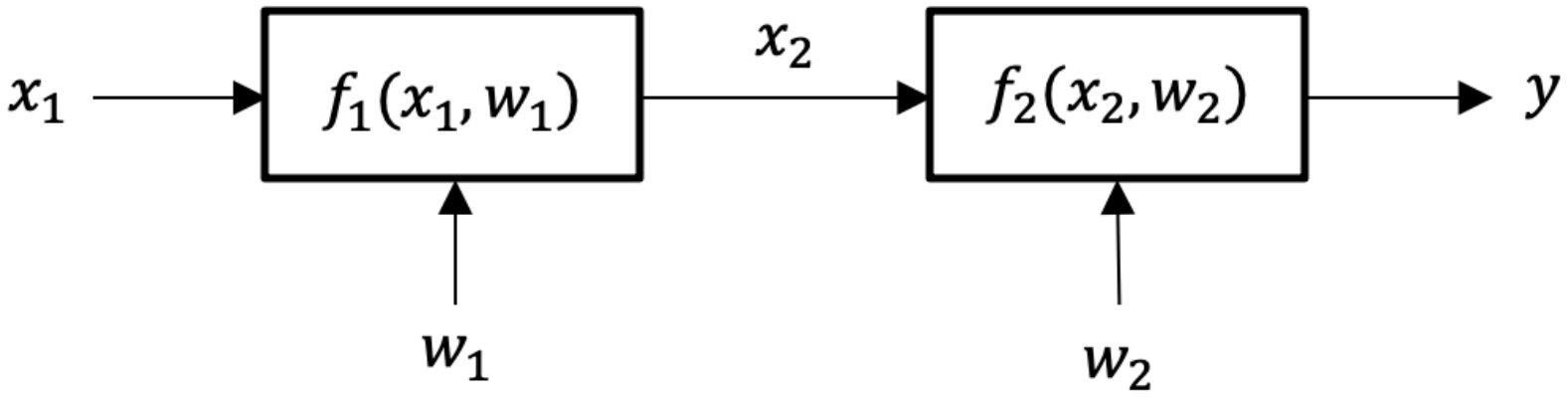
- Output SNR (SNR_y) requirements set by application:

$$SNR_y = 10\log_{10}\left[\frac{\sigma_s^2}{\sigma_n^2}\right]$$

- Need to set $SQNR_y \gg SNR_y$, e.g., $SQNR_y = \boldsymbol{SNR_y} + 6dB$
- How to compute SQNR for a dot product?

Noise Propagation via Chain Rule

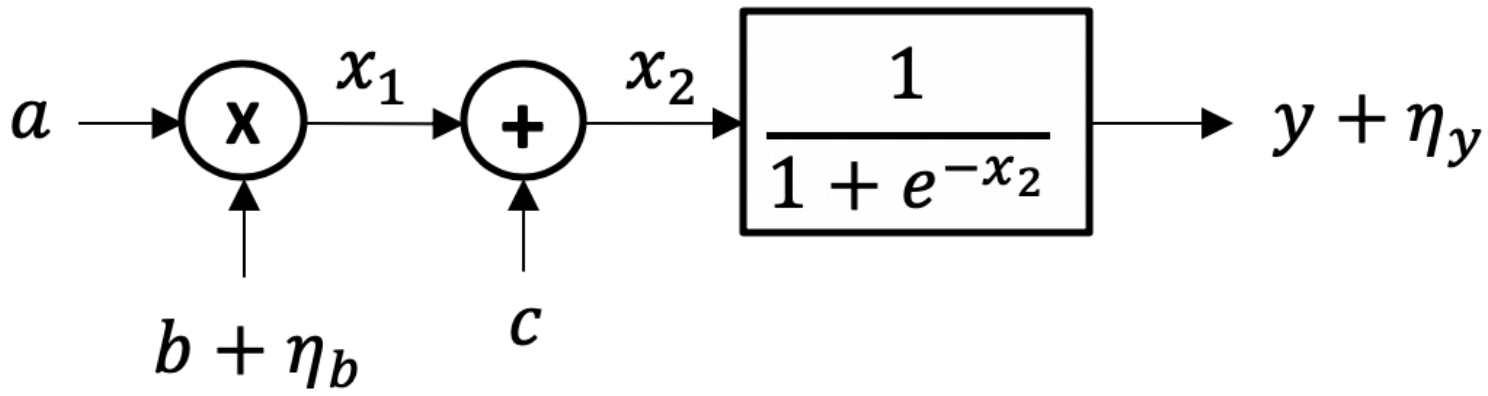
f_1 and f_2 are differentiable functions



$$\frac{\partial y}{\partial x_1} = \frac{\partial y}{\partial x_2} \times \frac{\partial x_2}{\partial x_1} = \frac{\partial f_2(x_2, w_2)}{\partial x_2} \times \frac{\partial f_1(x_1, w_1)}{\partial x_1}$$

$$\frac{\partial y}{\partial w_1} = \frac{\partial y}{\partial x_2} \times \frac{\partial x_2}{\partial w_1} = \frac{\partial f_2(x_2, w_2)}{\partial x_2} \times \frac{\partial f_1(x_1, w_1)}{\partial w_1}$$

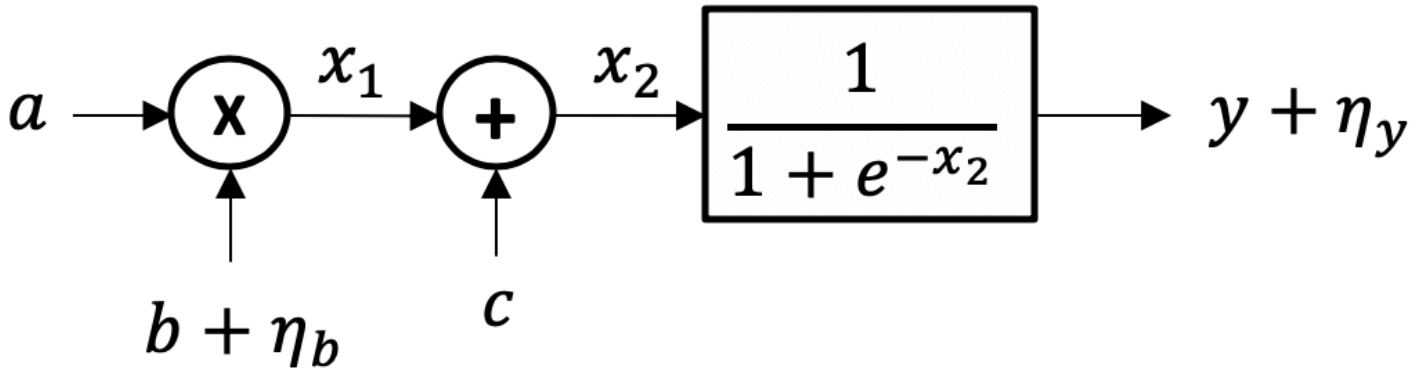
Example: Non-linear System with a Noisy Input



- noisy input: b

- noise propagation:

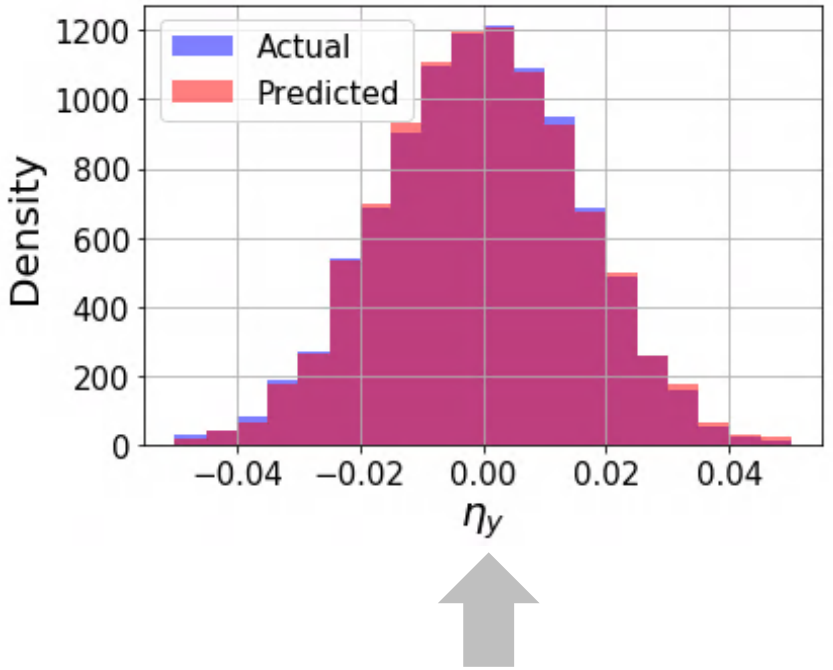
$$\eta_y \approx \frac{\partial y}{\partial b}\eta_b \; ; \qquad \frac{\partial y}{\partial b} = \frac{\partial y}{\partial x_2} \times \frac{\partial x_2}{\partial x_1} \times \frac{\partial x_1}{\partial b}$$



- $\frac{\partial y}{\partial b} = \frac{\partial y}{\partial x_2} \times \frac{\partial x_2}{\partial x_1} \times \frac{\partial x_1}{\partial b}$
- $\frac{\partial y}{\partial x_2} = y(1-y); \frac{\partial x_2}{\partial x_1} = 1; \frac{\partial x_1}{\partial b} = a$
- $\frac{\partial y}{\partial b} = ay(1-y) \rightarrow \eta_y = ay(1-y)\eta_b$

  $= N_G \eta_b =$ (noise gain) × (noise amplitude)

$\eta_b \sim N(0,0.01)$



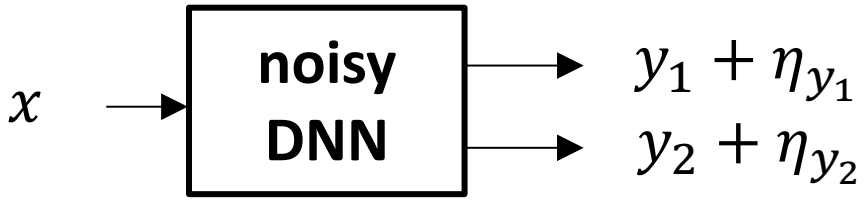
Close match between empirical simulations and analysis

How to use Noise Gains?

- to quantize DNNs with minimum precision subject to an accuracy loss budget
- to update corresponding weight parameter (SGD/back-propagation)

DNN Decision Error due to Noise

If $y_1 > y_2$ declare Class 1
else declare Class 2



If $y_1 + \eta_{y_1} > y_2 + \eta_{y_2}$ declare Class 1
else declare Class 2

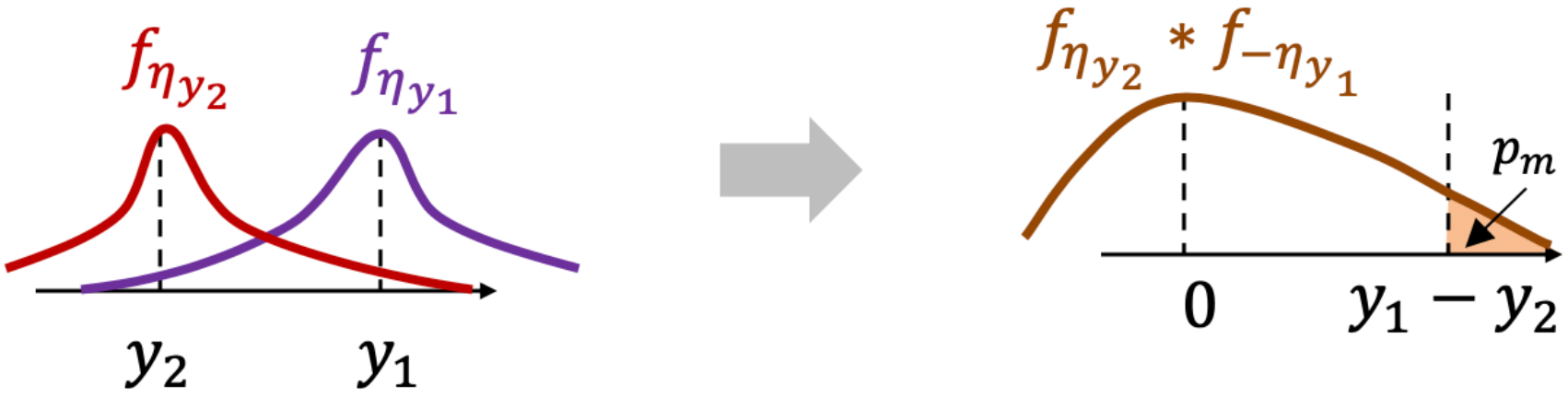
If $y_1 > y_2$ but $y_1 + \eta_{y_1} < y_2 + \eta_{y_2}$ → decision error (**mismatch**)

- sensitivity of DNNs to noise → probability of output reordering → DNN makes decision error (how to calculate?)

what is the mismatch probability?

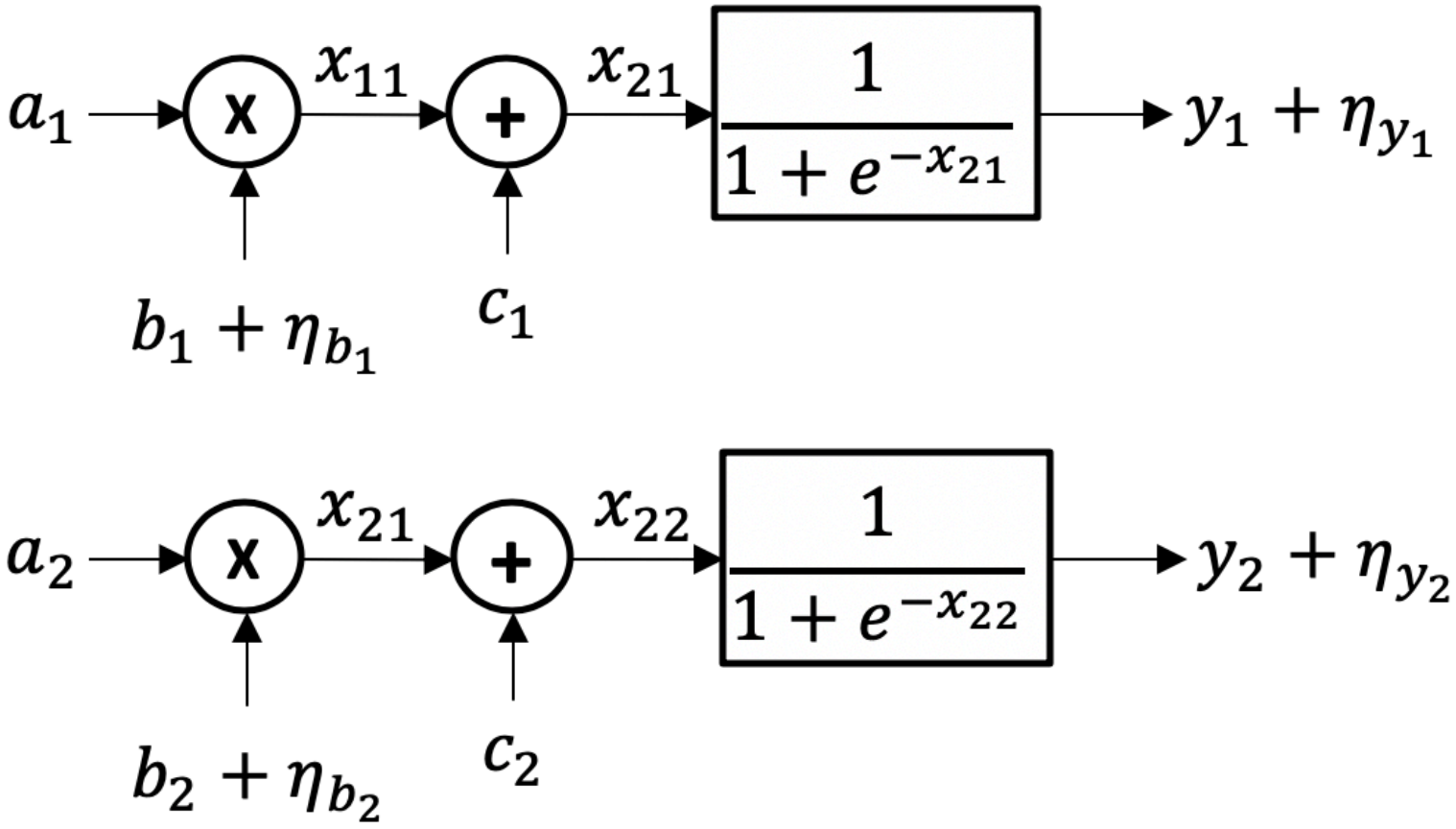
$A = \{y_1 > y_2\}$: an event; $A_m = \{y_1 + \eta_{y_1} < y_2 + \eta_{y_2} | A\}$: mismatch event

$p_m = \Pr\{y_1 + \eta_{y_1} < y_2 + \eta_{y_2} | A\} = \Pr\{\eta_{y_2} - \eta_{y_1} > y_1 - y_2 | A\}$: mismatch probability



- if η_{y_1} and η_{y_2} are independent → $f_{\eta_{y_2} - \eta_{y_1}} = f_{\eta_{y_2}} * f_{-\eta_{y_1}}$
- Calculate p_m

Example



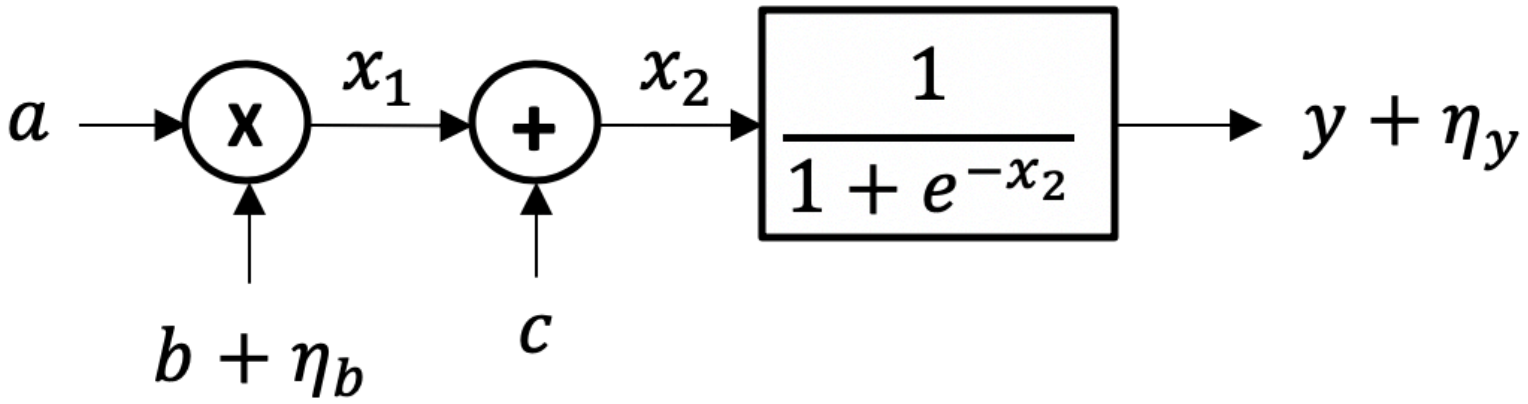
- what is the probability $\Pr\{\eta_{y_2} - \eta_{y_1} > y_1 - y_2 | A\}$?

$$\eta_{y_2} - \eta_{y_1} = a_2 y_2(1 - y_2)\eta_{b_2} - a_1 y_1(1 - y_1)\eta_{b_1}$$

assume $\eta_{b_1} \perp \eta_{b_2}$ (independent) with identical variance σ^2

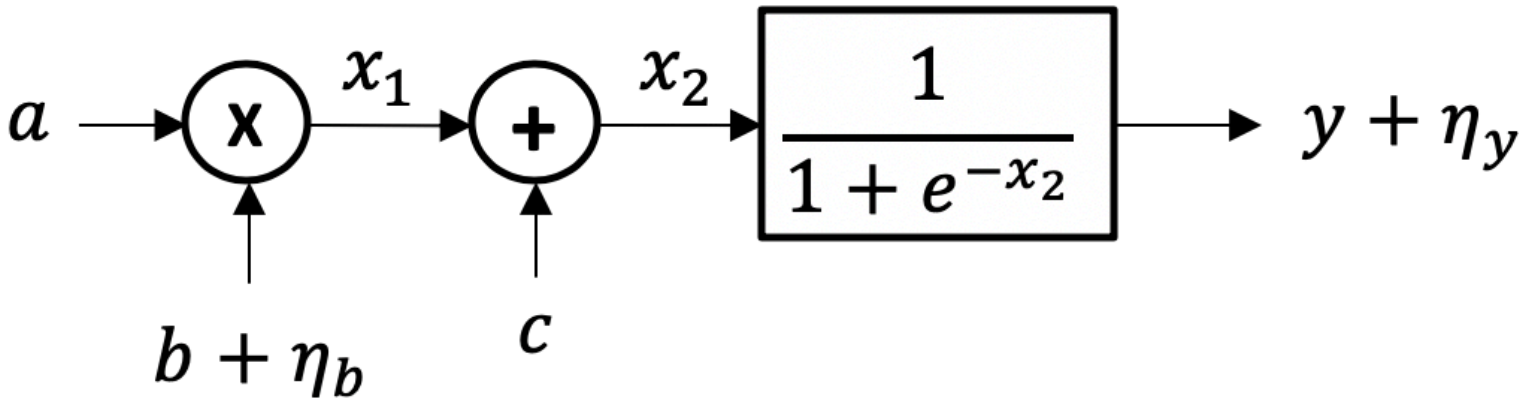
$$Var(\eta_{y_2} - \eta_{y_1}) = [a_1^2 y_1^2(1 - y_1)^2 + a_2^2 y_2^2(1 - y_2)^2]\sigma^2$$

$$\Pr\{\eta_{y_2} - \eta_{y_1} > y_1 - y_2 | A\} < \frac{1}{2}\frac{Var(\eta_{y_2} - \eta_{y_1})}{(y_1 - y_2)^2}$$



$$\Pr\{\eta_{y_2} - \eta_{y_1} > y_1 - y_2 | A\} < \underbrace{\frac{[a_1^2 y_1^2 (1-y_1)^2 + a_2^2 y_2^2 (1-y_2)^2]}{2(y_1 - y_2)^2}}_{\text{noise gain } (N_G)} \sigma^2$$

- noise gain yields upper bounds on accuracy loss



- $a = [a_1, a_2] = [0.2, -0.6]$
- $b = \begin{bmatrix} 0.5, & 0 \\ 0, & -0.5 \end{bmatrix}, c = [0, 0]$
- $y = [y_1, y_2] = [0.5249, 0.5744]$
- $G = \frac{[a_1^2 y_1^2 (1-y_1)^2 + a_2^2 y_2^2 (1-y_2)^2]}{2(y_1 - y_2)^2} = 4.9049$
- $\Pr\{\eta_{y_2} - \eta_{y_1} > y_1 - y_2 | A\} < 0.01 \rightarrow B_b \geq 4$

$$\Pr\{\eta_{y_2} - \eta_{y_1} > y_1 - y_2 | A\} < 4.9049\, \sigma^2$$

$$\sigma^2 = \frac{\Delta^2}{12}, \qquad \Delta = 2^{1-B_b}$$

Impact of Quantization on Inference Accuracy

Analytical Guarantees on Numerical Precision of Deep Neural Networks

Charbel Sakr Yongjune Kim Naresh Shanbhag

Sakr, C., Kim, Y., & Shanbhag, N., "Analytical Guarantees on Numerical Precision of Deep Neural Networks", In *International Conference on Machine Learning (ICML 2017)* (pp. 3007-3016).

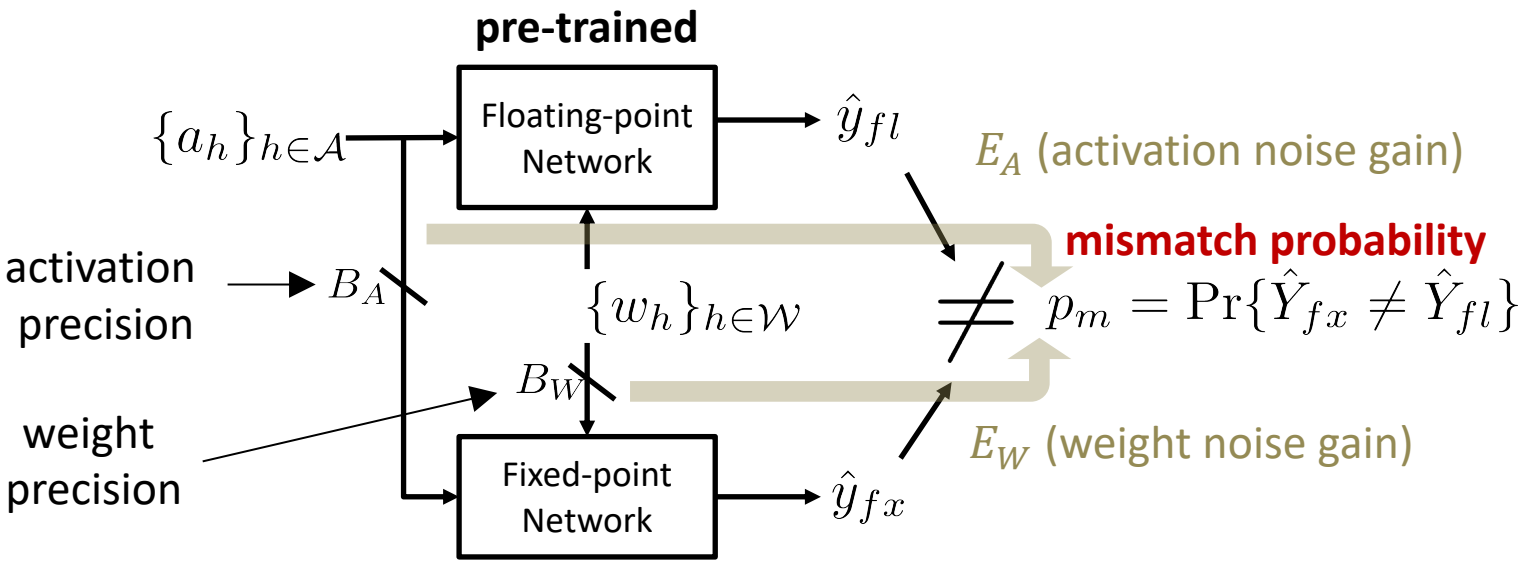
code available at https://github.com/charbel-sakr

AN ANALYTICAL METHOD TO DETERMINE MINIMUM PER-LAYER PRECISION OF DEEP NEURAL NETWORKS

Charbel Sakr and Naresh Shanbhag

(ICASSP'2018)

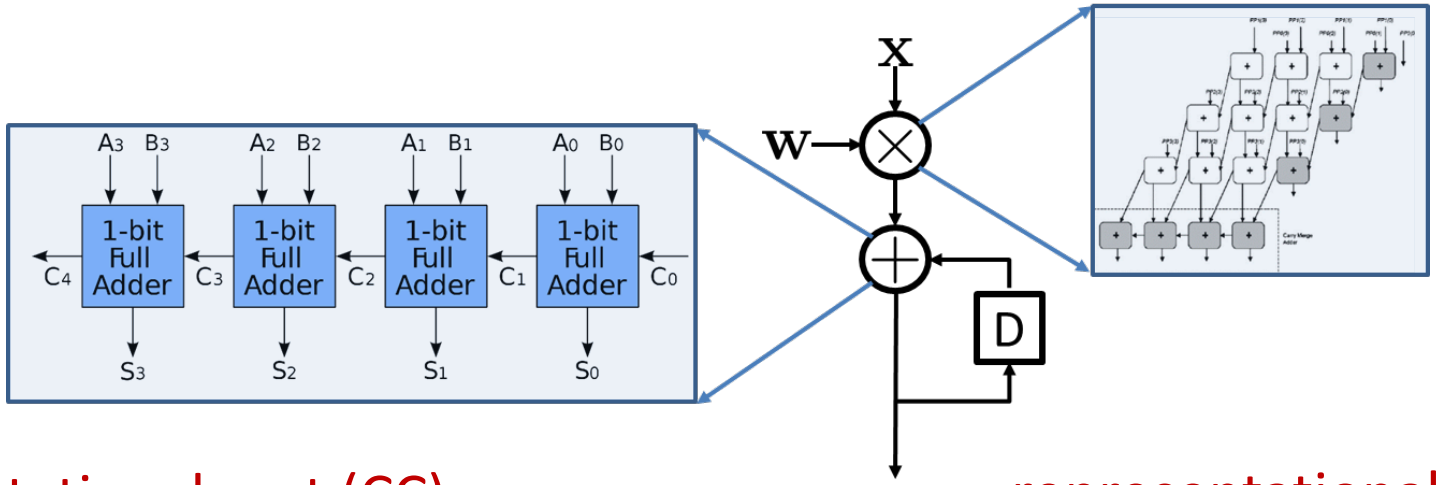
Precision Analysis Framework



- no retraining; per-layer precision; activation vs. weight trade-off
- noise gains computed via one standard backprop iteration
- minimizing p_m can be done via noise equalization (NE)

Hardware Complexity Metrics

- easy to compute from an algorithmic description



computational cost (CC)

(# of 1-b FAs)

# of dot products

dot product dimension

$$\sum_{l=1}^{L} N_l \left(D_l B_l^{(a)} B_l^{(w)} + (D_l - 1)\left(B_l^{(a)} + B_l^{(w)} + \log_2 D_l - 1 \right) \right)$$

activation precision

weight precision

representational cost (RC)

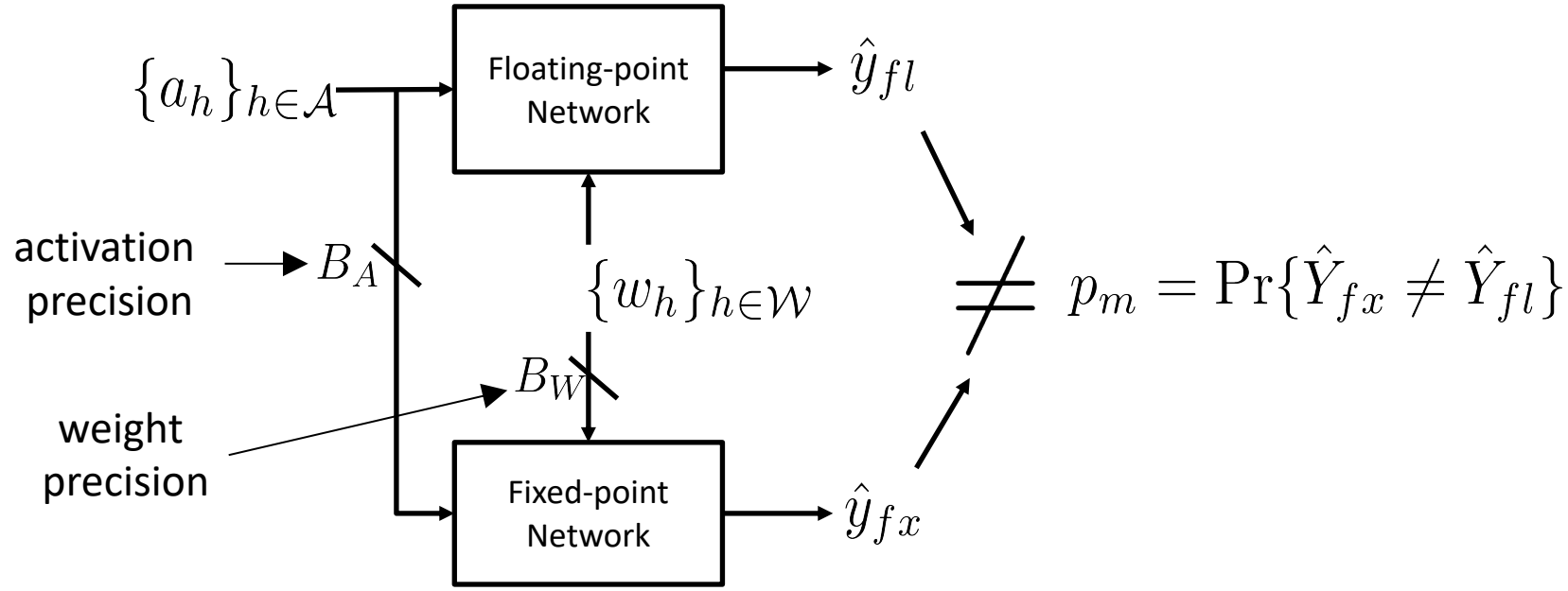
(# of bits of storage)

# of weights

$$\sum_{l=1}^{L} \left(R_l^{(a)} B_l^{(a)} + R_l^{(w)} B_l^{(w)} \right)$$

# of activations

Weight vs. Activation Precision Trade-off



$$p_m \leq \Delta_A^2 E_A + \Delta_W^2 E_W$$

$$\Delta_A = 2^{-(B_A-1)}$$
$$\Delta_W = 2^{-(B_W-1)}$$

- activation and weight precision can be traded-off for fixed accuracy loss

Noise Equalization

$$p_m \leq \Delta_A^2 E_A + \Delta_W^2 E_W$$

$$B_A - B_W = round\left(\log_2 \sqrt{\frac{E_A}{E_W}}\right)$$

- ensures both activations and weights contribute equally to accuracy loss

Per-Layer Noise Equalization

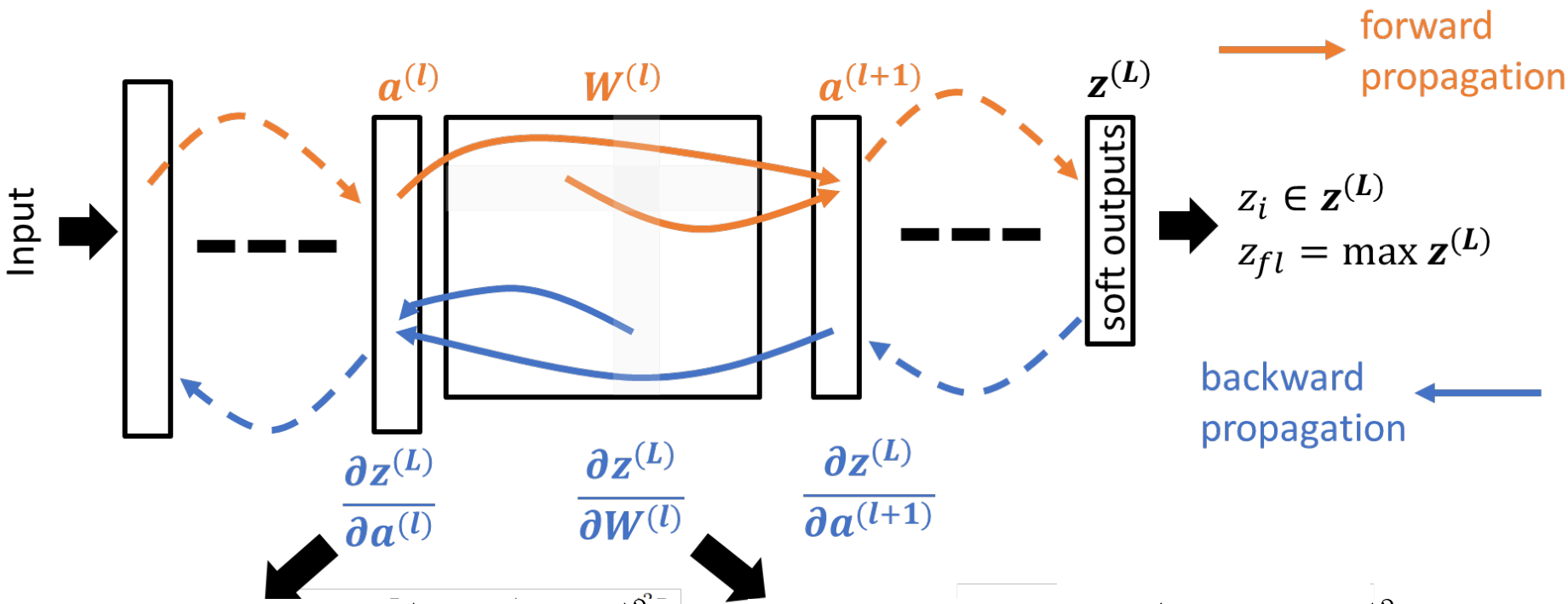
- in the case of per-layer precisions:

$$p_m \leq \sum_{l=1}^{L} \left(\Delta_{A,l}^2 E_{A,l} + \Delta_{W,l}^2 E_{W,l} \right)$$

- noise equalization:

$$B_{A,l} = round\left(\log_2 \left(\sqrt{\frac{E_{A,l}}{E_{\min}}} \right) \right) + B_{\min} \qquad B_{W,l} = round\left(\log_2 \left(\sqrt{\frac{E_{W,l}}{E_{\min}}} \right) \right) + B_{\min}$$

Noise Gain Calculation via Backprop



$$E_A = \mathbb{E}\left[\sum_{\substack{i=1 \\ i \neq \hat{Y}_{fl}}}^{M} \frac{\sum_{h \in \mathcal{A}} \left| \frac{\partial (Z_i - Z_{\hat{Y}_{fl}})}{\partial A_h} \right|^2}{24 |Z_i - Z_{\hat{Y}_{fl}}|^2}\right] \qquad E_W = \mathbb{E}\left[\sum_{\substack{i=1 \\ i \neq \hat{Y}_{fl}}}^{M} \frac{\sum_{h \in \mathcal{W}} \left| \frac{\partial (Z_i - Z_{\hat{Y}_{fl}})}{\partial w_h} \right|^2}{24 |Z_i - Z_{\hat{Y}_{fl}}|^2}\right]$$

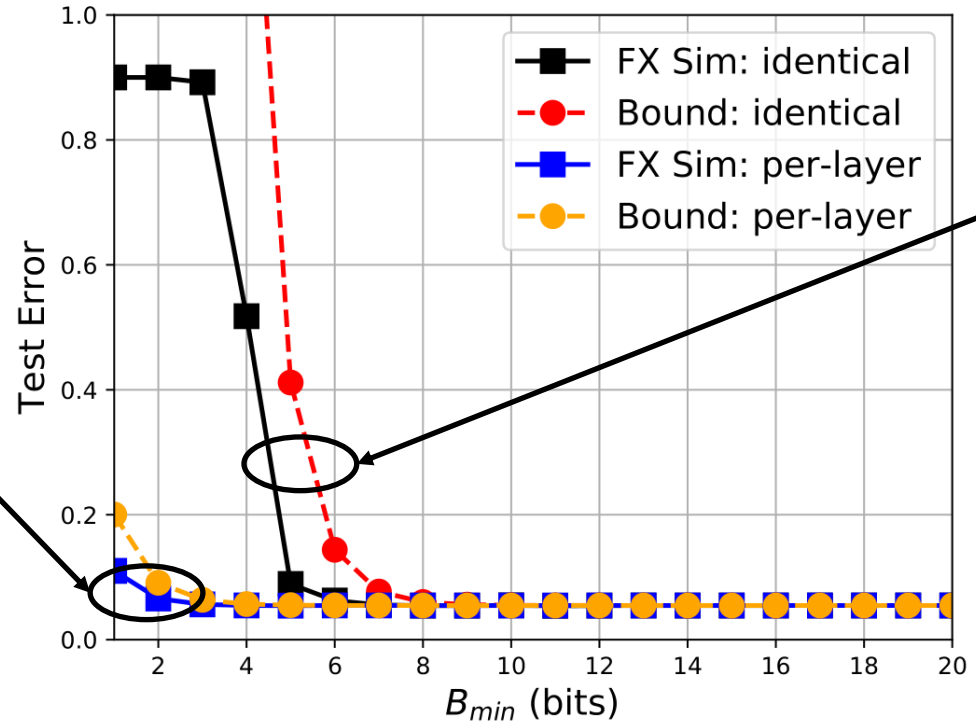
Lesson 1 – Precision Trade-offs Captured Analytically

$$B_{A,l} = \log_2 \sqrt{\frac{E_{A,l}}{E_{\min}}} + B_{\min}$$

&

$$B_{W,l} = \log_2 \sqrt{\frac{E_{W,l}}{E_{\min}}} + B_{\min}$$

CIFAR-10 using ResNet 18

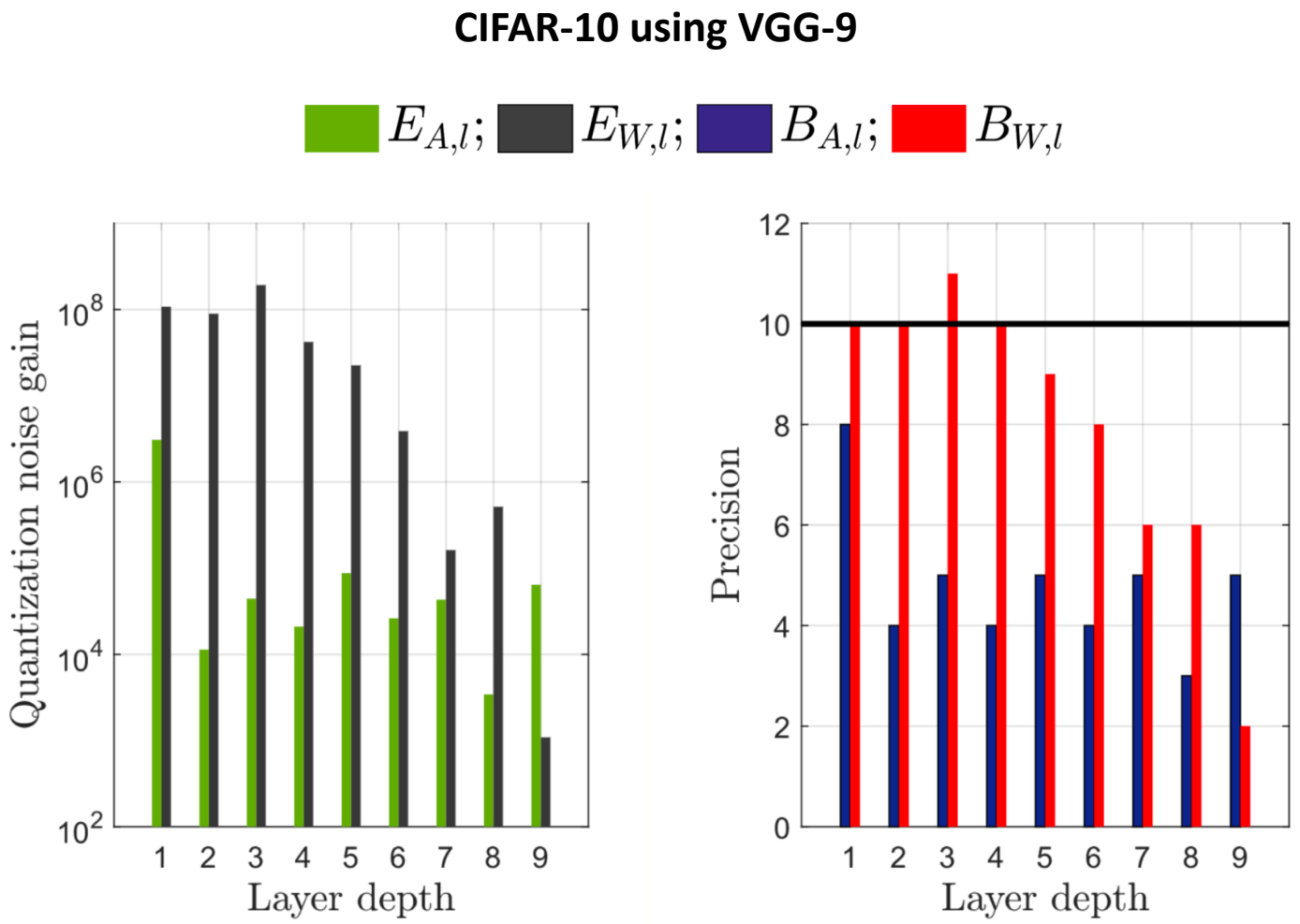


$$B_{A,l} = B_{W,l} = B_{\min} \text{ for } l = 1 \ldots L$$

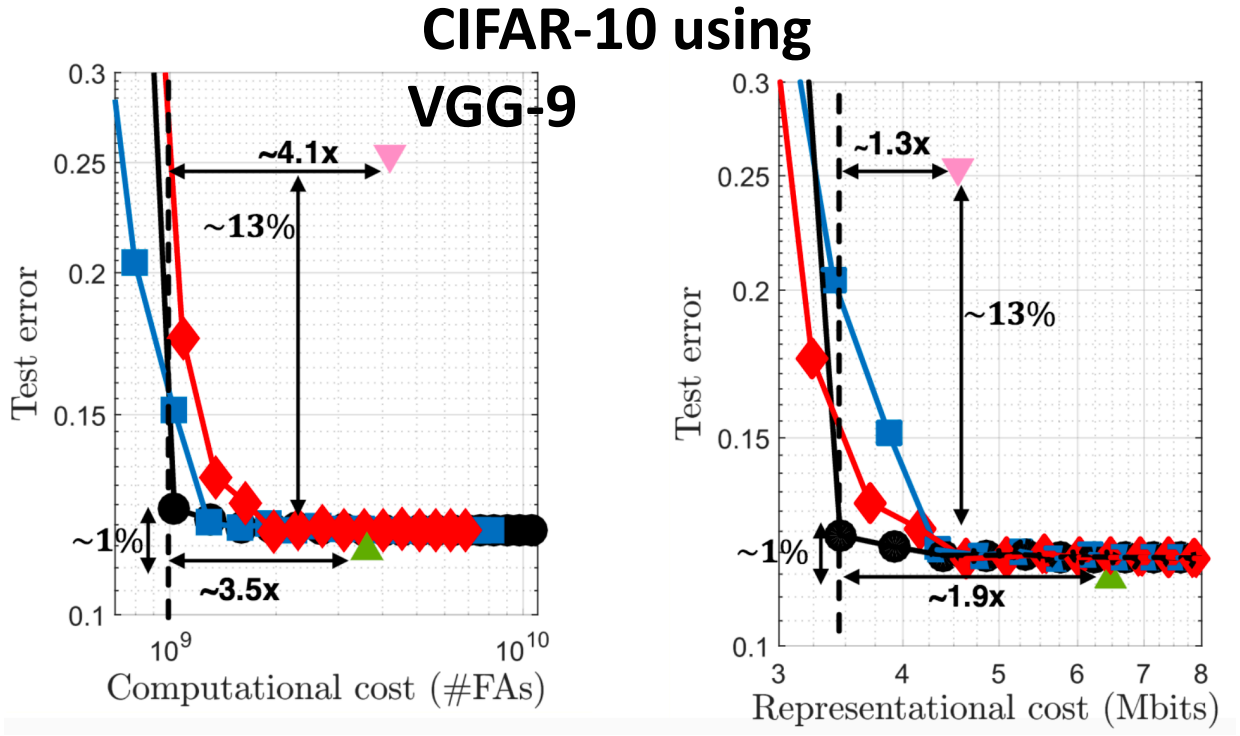
- bound predicts minimum precision within 1~2 bits
- bound reveals trade-offs between network precisions
- trade-offs captured by relative values of noise gains

Lesson 2 – Precision Decreases with Depth

- weights typically require more precision than activations
- precision decreases because early perturbations are most destructive

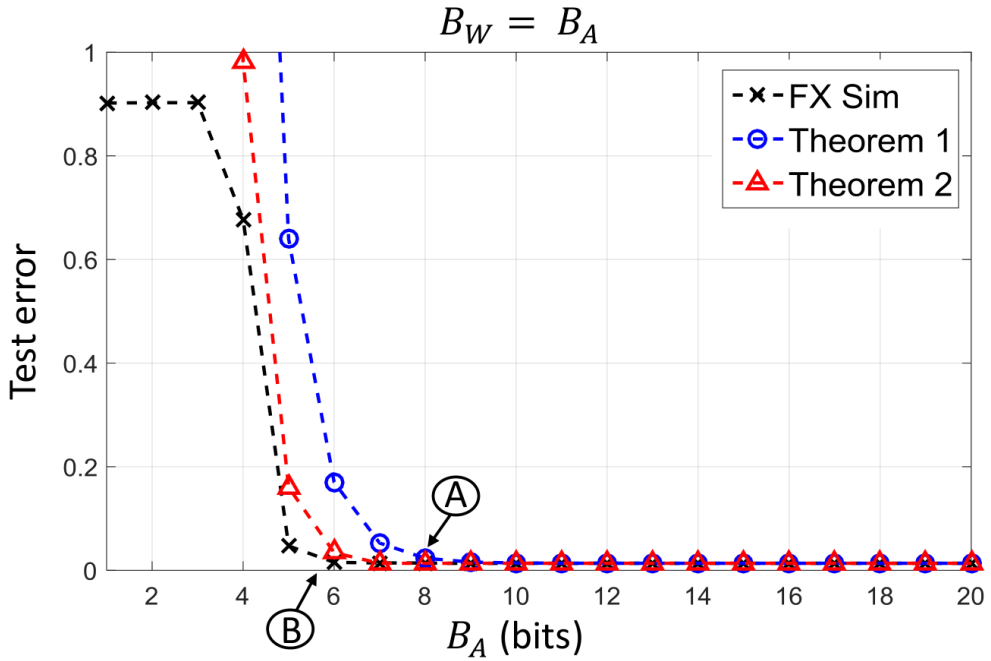


Lesson 3 – BinaryNets are More Complex than Minimum Precision Networks



- up to **3.5x lower** complexity & **2X** lower storage over BinaryNet at **iso-accuracy**
- empirically observed by [Moons, Verhelst]

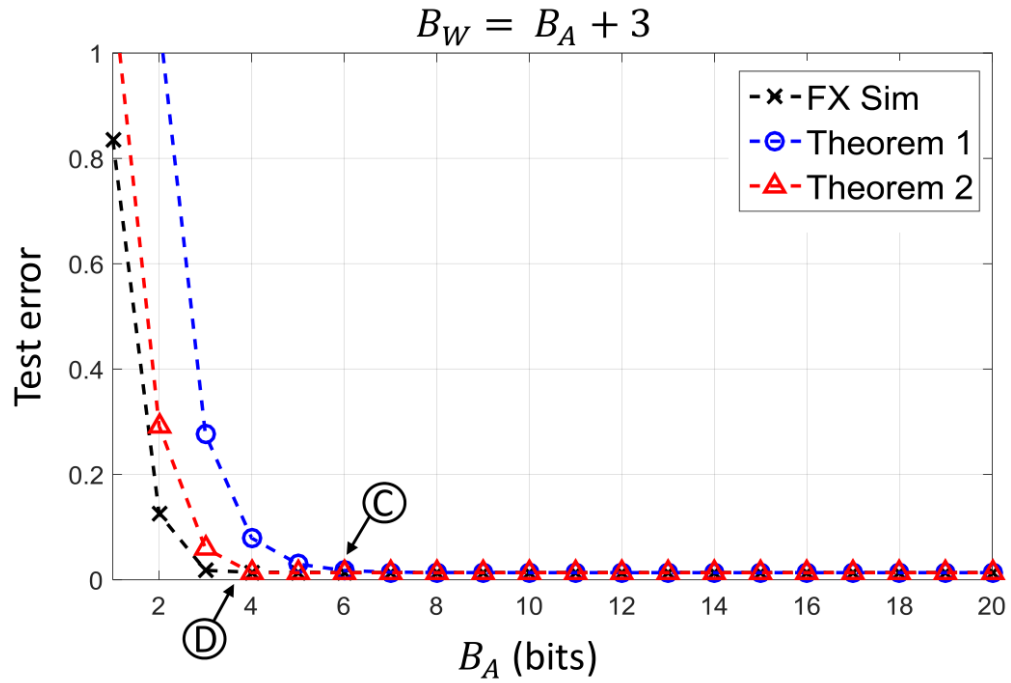
Simulations – MNIST



A: $B_W = B_A; p_m \leq 1\%$ (Theorem 1)
B: $B_W = B_A; p_m \leq 1\%$ (Theorem 2)
C: $B_W = B_A + 3; p_m \leq 1\%$ (Theorem 1)
D: $B_W = B_A + 3; p_m \leq 1\%$ (Theorem 2)

MLP : 784 – 512 – 512 – 512 – 10

- upper bounds are valid
- activation precision reduced by 3 bits with no accuracy degradation

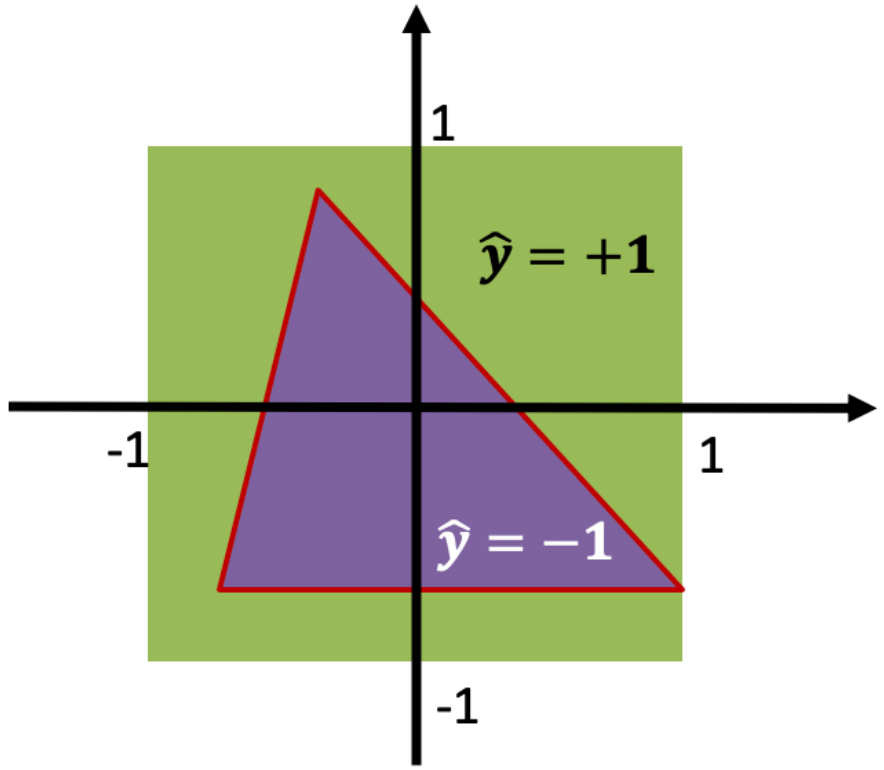


Results – MNIST

Precision Assignment	Test error (%)	Computational Cost (10^6 FAs)	Representational Cost (10^6 bits)
Floating-point	1.36	N/A	N/A
$(8, 8)$	1.41	82.9	7.5
$(6, 6)$	1.54	53.1	**5.63**
$(6, 9)$	**1.35**	72.7	8.43
$(4, 7)$	1.43	**44.7**	6.54
SQ $(16, 16)$ (Gupta et al., 2015)	1.4	533	28
BN $(1, 1)$ (Hubara et al., 2016b)	1.4	117	10

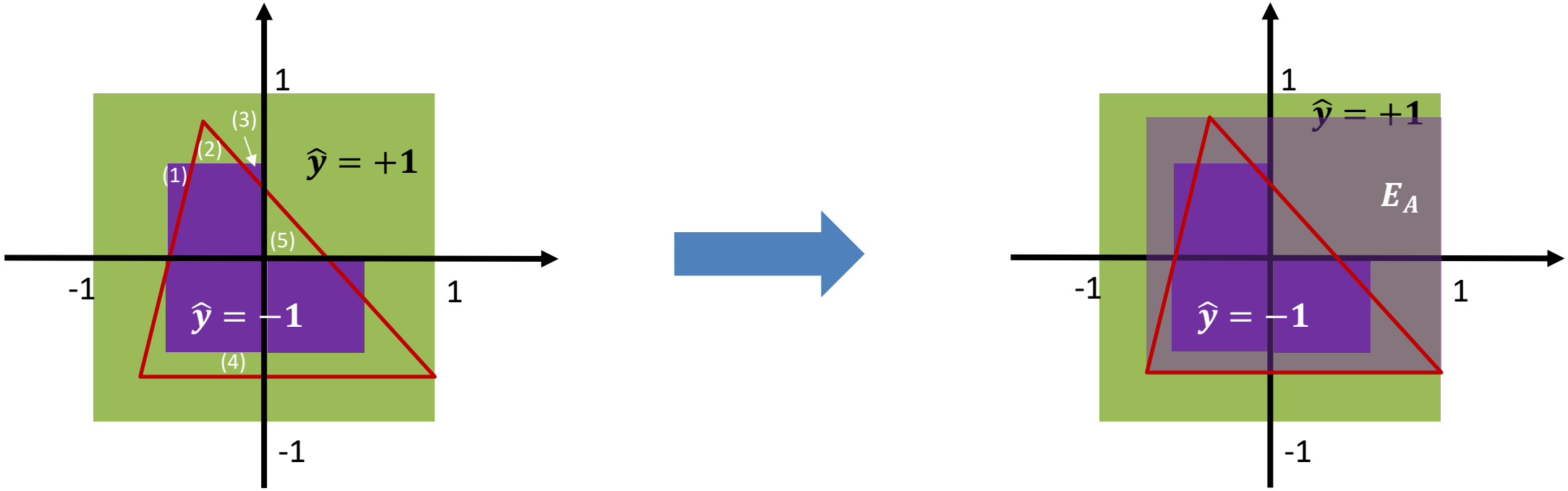
- no loss in accuracy
- ~2× savings in computational and rep. costs over BinaryNets!
- key finding: complexity scales *quadratically* with height (#neurons/layer)
  - BN (2048/512) is 4 times higher → 16 times more complex
  - Using up to 9 bits → still about 2 times less complex than BN

Visualization



- Assume input data uniformly distributed in $[-1,1]^2$
- 2 class problem

Input Quantization

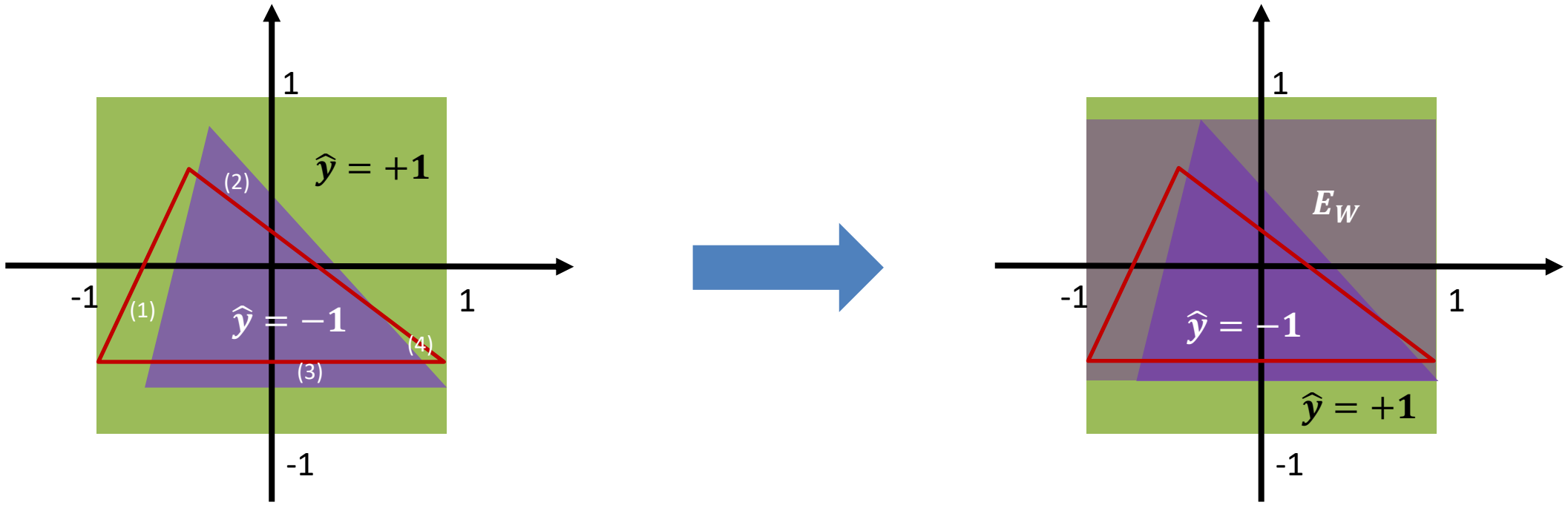


- p_m = combined area of (1), (2), (3), (4), (5)

- Hard to compute exactly → easy to compute an upper bound:

$$p_m \leq \Delta_A^2 E_A$$

Weight Quantization



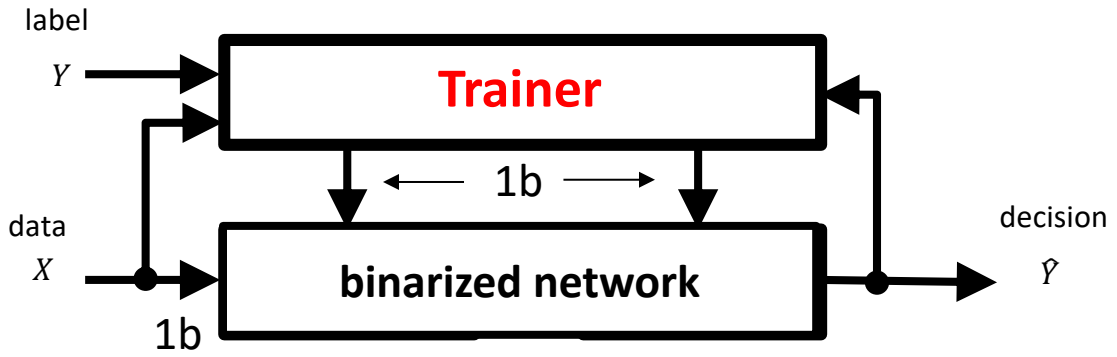
- p_m = combined area of (1), (2), (3), (4)
- hard to compute exactly → easy to compute an upper bound: $p_m \leq \Delta_W^2 E_W$

Binarized Neural Networks

Binarized Neural Networks: Training Neural Networks with Weights and Activations Constrained to +1 or −1

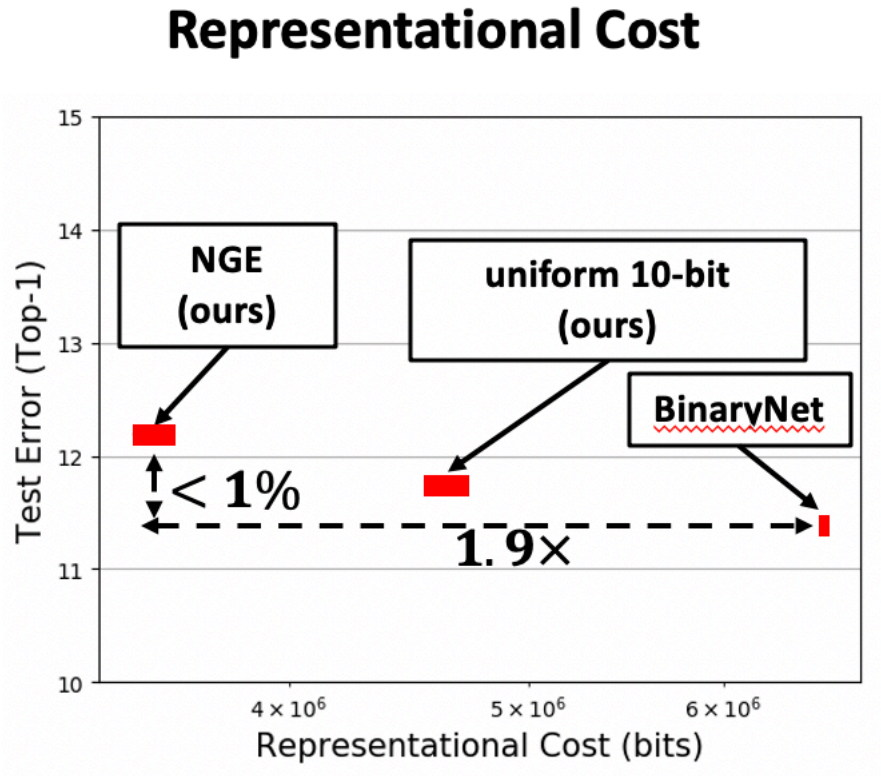
Matthieu Courbariaux*[1] MATTHIEU.COURBARIAUX@GMAIL.COM
Itay Hubara*[2] ITAYHUBARA@GMAIL.COM
Daniel Soudry[3] DANIEL.SOUDRY@GMAIL.COM
Ran El-Yaniv[2] RANI@CS.TECHNION.AC.IL
Yoshua Bengio[1,4] YOSHUA.UMONTREAL@GMAIL.COM

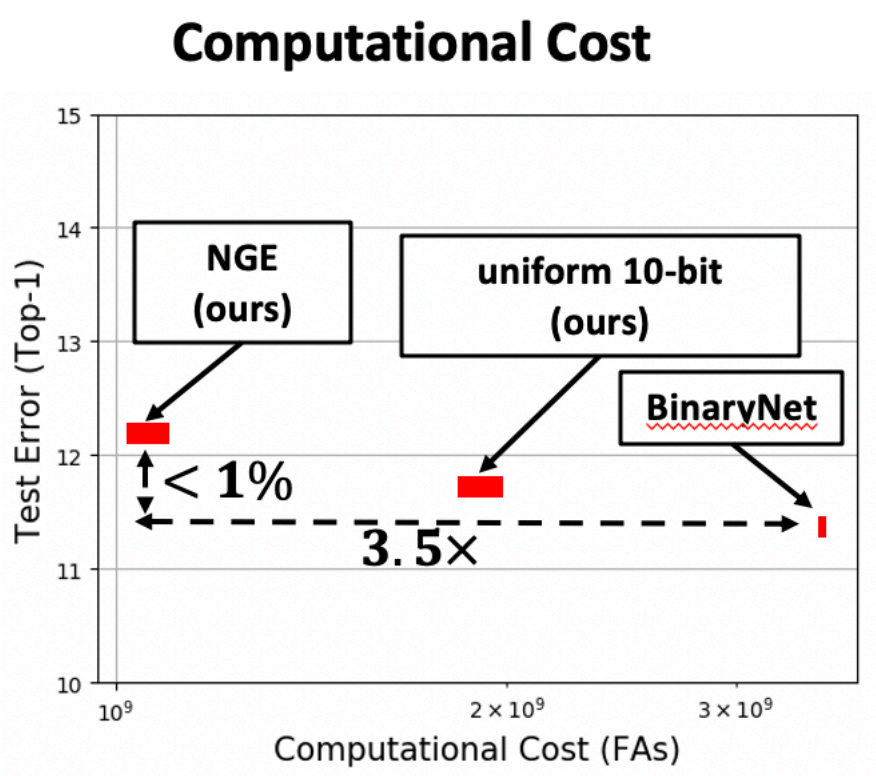
[arxiv, March 2016]



- weights and activations are both restricted to $\{\pm 1\}$
- train network to find best set of binary weights
- note: training is done with floating-point computations

Accuracy vs Complexity – Small Models (VGG-9 on CIFAR-10)

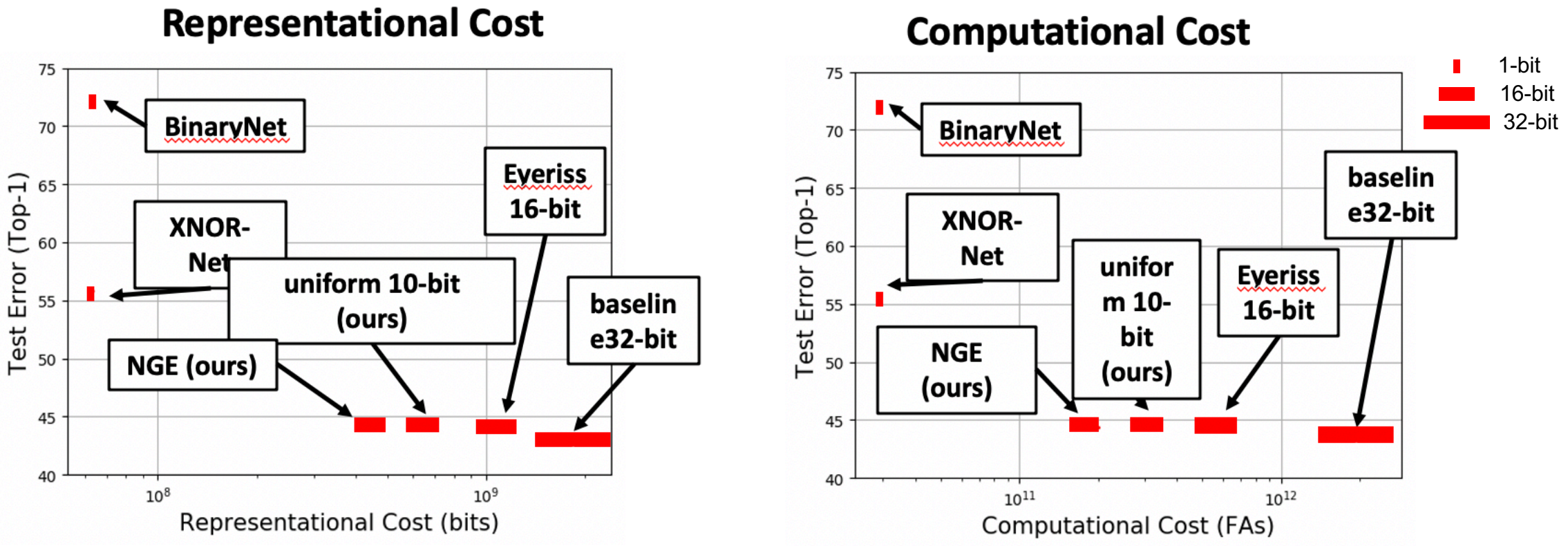




1-bit
10-bit

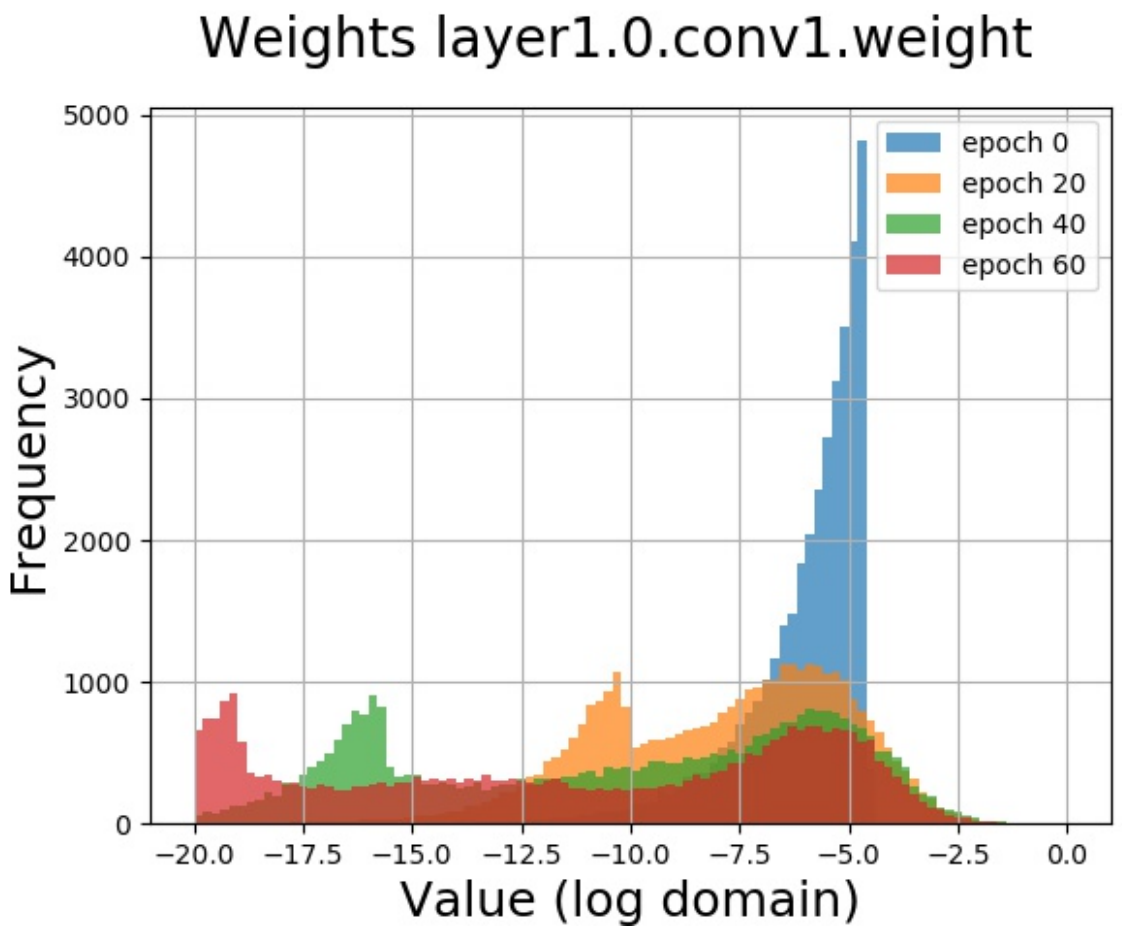
- BinaryNets achieve good accuracy on small datasets & networks because they use very large models!

Accuracy vs Complexity – Large Models (AlexNet on ImageNet)



- BinaryNets not great for large datasets and complex models
- iso-accuracy and small complexity can be achieved by starting with small network and using principled precision reduction

DNN Weight Distribution



- changes over time as during training → makes training more challenging than inference

Summary

- DNN minimum precision requirements can be determined analytically from FL-pt simulations
- weight and activation precisions trade-off
- precisions tend to decrease with depth
- different architectures provide different trade-offs
- BinaryNets tend to be much more complex than minimum precision networks

Course Web Page

https://courses.grainger.illinois.edu/ece598nsg/fa2020/
https://courses.grainger.illinois.edu/ece498nsu/fa2020/

http://shanbhag.ece.uiuc.edu

Classification

$$\hat{y} = \arg \max_{i=1,\ldots,M} z_i$$

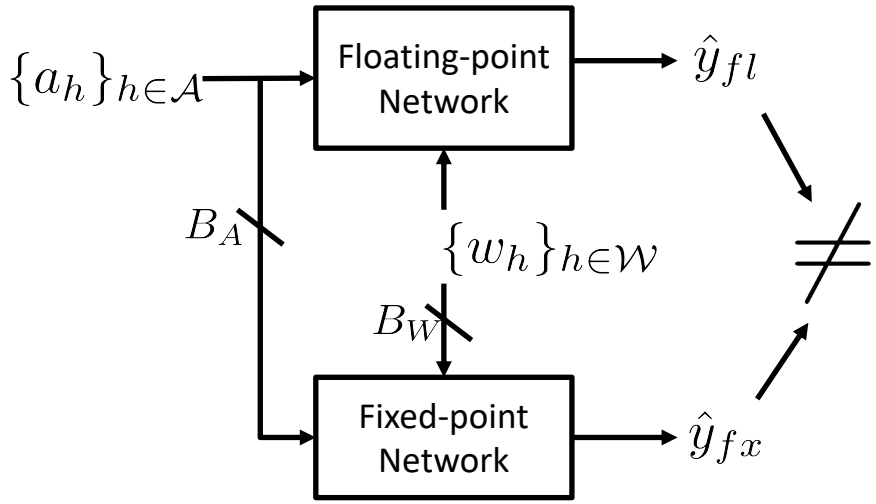
$$z_i = f\left(\{a_h\}_{h\in\mathcal{A}}, \{w_h\}_{h\in\mathcal{W}}\right)$$

Output Quantization

$$z_i + q_{z_i} = f\left(\{a_h + q_{a_h}\}_{h\in\mathcal{A}}, \{w_h + q_{w_h}\}_{h\in\mathcal{W}}\right)$$

$$q_{z_i} = \sum_{h\in\mathcal{A}} q_{a_h} \frac{\partial z_i}{\partial a_h} + \sum_{h\in\mathcal{W}} q_{w_h} \frac{\partial z_i}{\partial w_h}$$

Mismatch Probability



p_m: "Mismatch Probability"

$$p_m = \Pr\{\hat{Y}_{fx} \neq \hat{Y}_{fl}\}$$

Proof Sketch

- For one input, when do we have a mismatch?
  → If FL network predicts label "j"
  → But FX network predicts label "i" where $i \neq j$
    → This happens with some probability computed as follows:

$$\Pr\left(z_i + q_{z_i} > z_j + q_{z_j}\right)$$ (Output re-ordering due to quantization)

$$= \Pr\left(q_{z_i} - q_{z_j} > z_j - z_i\right) = \tfrac{1}{2}\Pr\left(|q_{z_i} - q_{z_j}| > |z_j - z_i|\right)$$ (Symmetry of quantization noise)

    → But we already know

$$q_{z_i} - q_{z_j} = \sum_{h\in\mathcal{A}} q_{a_h}\frac{\partial(z_i - z_j)}{\partial a_h} + \sum_{h\in\mathcal{W}} q_{w_h}\frac{\partial(z_i - z_j)}{\partial w_h}$$

    → Whose variance is

$$\frac{\Delta_A^2}{12}\sum_{h\in\mathcal{A}}\left|\frac{\partial(z_i - z_j)}{\partial a_h}\right|^2 + \frac{\Delta_W^2}{12}\sum_{h\in\mathcal{W}}\left|\frac{\partial(z_i - z_j)}{\partial w_h}\right|^2$$

    → Applying Chebyshev inequality + LTP yields the result