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

Low-complexity DNNs - Learned Quantization & Model Compression

Naresh Shanbhag

Department of Electrical and Computer Engineering

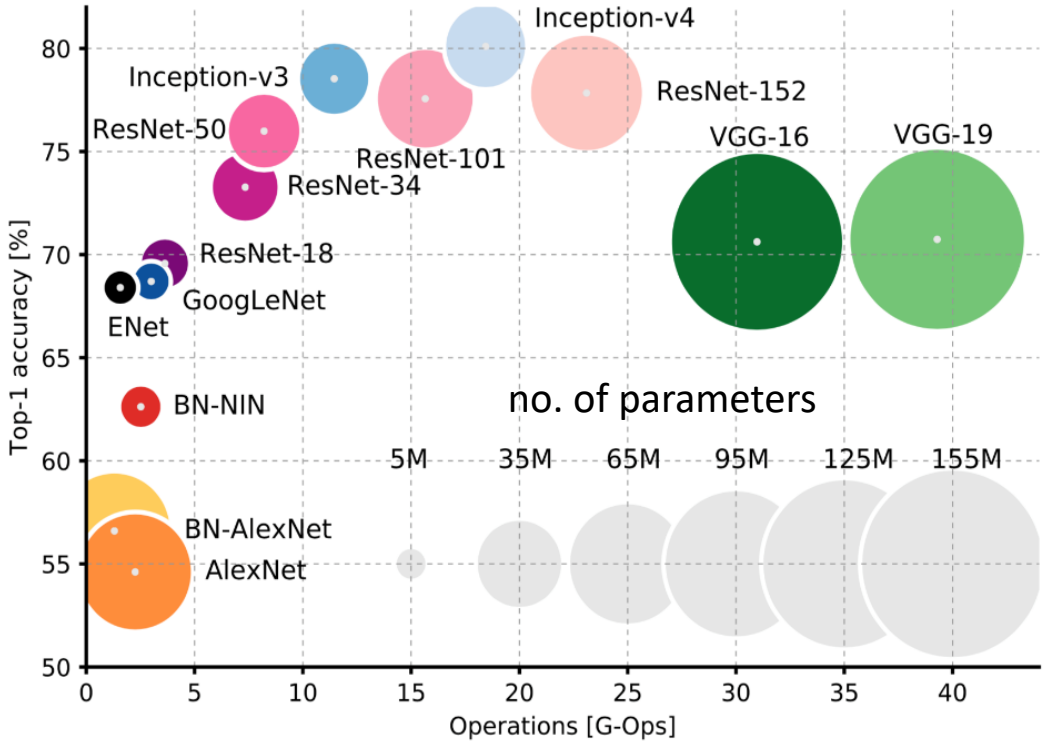
University of Illinois at Urbana-Champaign

http://shanbhag.ece.uiuc.edu

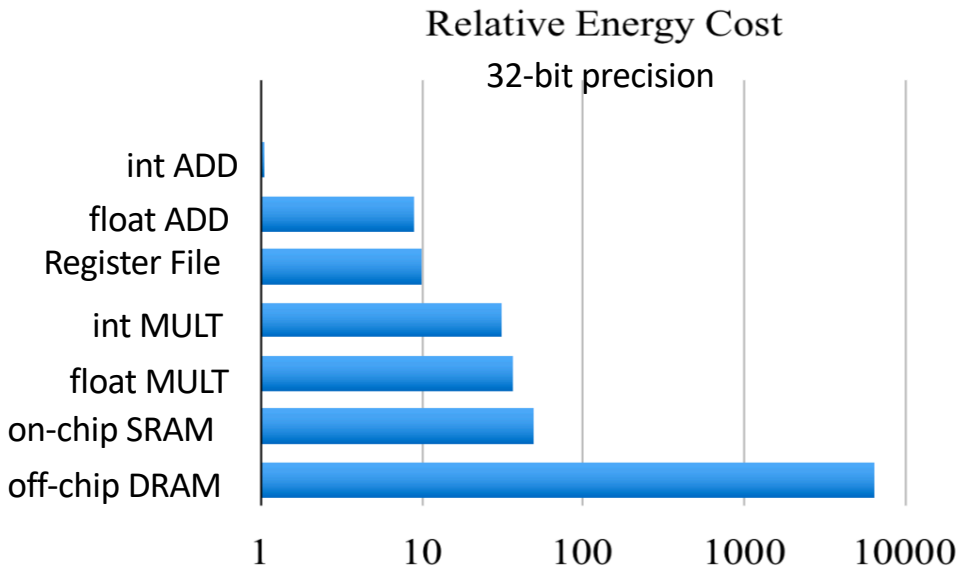
Low-Complexity DNNs

- Two approaches
- 1: design a low-complexity network from scratch
  - based on design intuitions (MobileNet, SqueezeNet)
- 2: reduce the complexity of a large network
  - binarization/ternarization
  - model compression
  - factorization
  - distillation
- need to address complexity vs. accuracy trade-off

Memory Access Energy in DNNs



[Canziani-arXiv-2017]



[Horowitz-ISSCC-2014]

- Require large no. of parameters → don't fit within on-chip SRAM
- off-chip DRAM accesses are 100x more energy expensive

Binarization and Ternarization

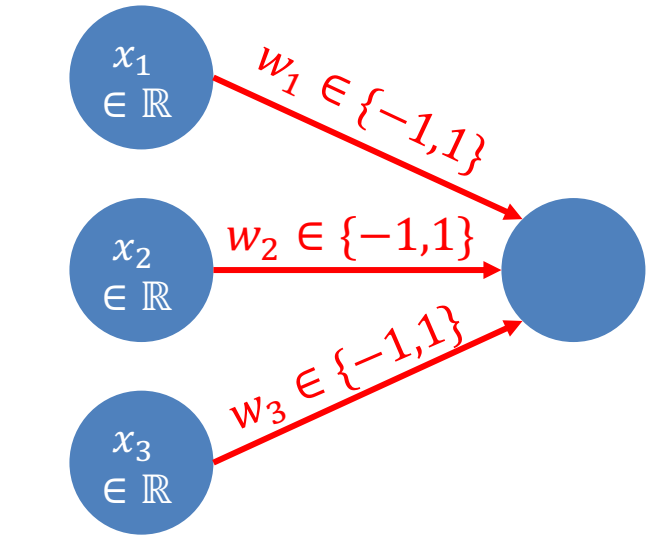
BinaryConnect: Training Deep Neural Networks with binary weights during propagations

NIPS'15

Matthieu Courbariaux
École Polytechnique de Montréal
matthieu.courbariaux@polymtl.ca

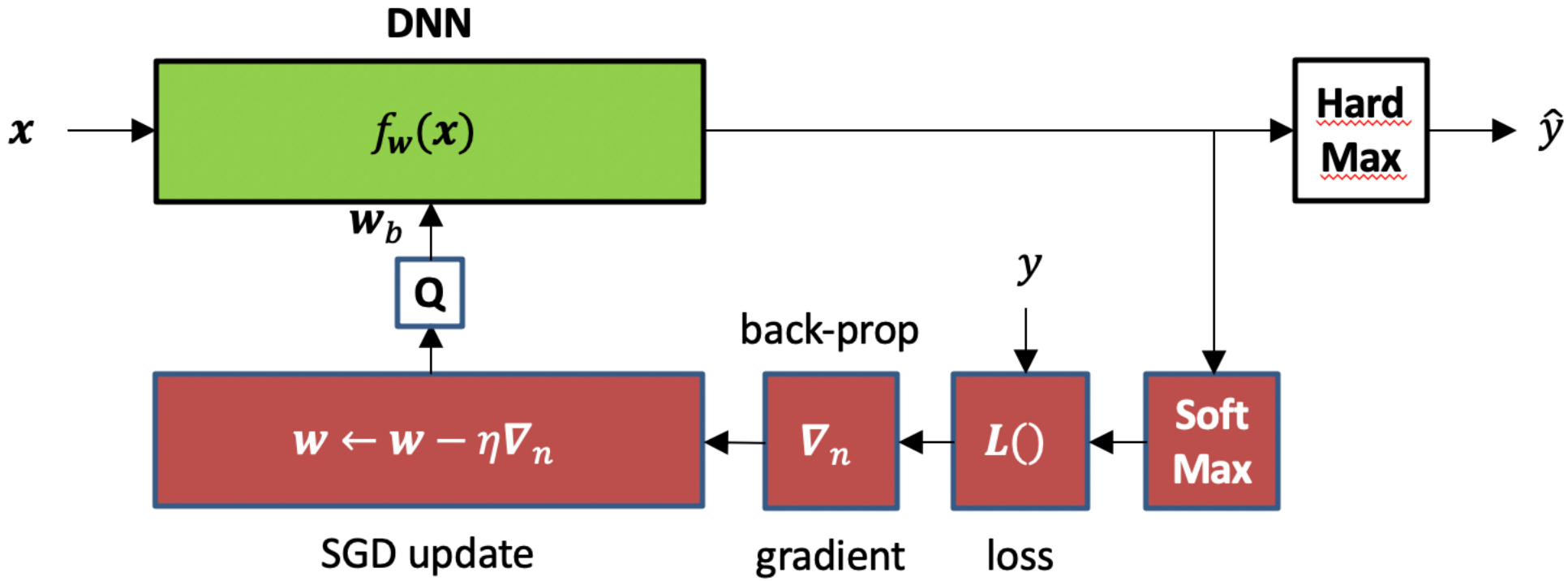
Yoshua Bengio
Université de Montréal, CIFAR Senior Fellow
yoshua.bengio@gmail.com

Jean-Pierre David
École Polytechnique de Montréal
jean-pierre.david@polymtl.ca



- famous for introducing *weight binarization*

Key Innovations



- weight binarization in forward path
- full precision backprop
- stochastic quantization

Binarization Schemes

deterministic binarization

full-precision

$$w_b = \begin{cases} +1 & \text{if } w \geq 0, \\ -1 & \text{otherwise.} \end{cases}$$

stochastic binarization

full-precision

$$w_b = \begin{cases} +1 & \text{with probability } p = \sigma(w), \\ -1 & \text{with probability } 1 - p. \end{cases}$$

"hard (piece-wise linear) sigmoid"

$$\sigma(x) = \text{clip}(\frac{x+1}{2}, 0, 1)$$

- stochastic binarization postulated to reduce accumulation of bias

1. Forward propagation: forward propagation with binary weights

$w_b \leftarrow \text{binarize}(w_{t-1})$

For $k = 1$ to L, compute a_k knowing a_{k-1}, w_b and b_{t-1}

2. Backward propagation: back-prop and weight update in full precision

Initialize output layer's activations gradient $\frac{\partial C}{\partial a_L}$

For $k = L$ to 2, compute $\frac{\partial C}{\partial a_{k-1}}$ knowing $\frac{\partial C}{\partial a_k}$ and w_b

3. Parameter update:

Compute $\frac{\partial C}{\partial w_b}$ and $\frac{\partial C}{db_{t-1}}$ knowing $\frac{\partial C}{\partial a_k}$ and a_{k-1}

$w_t \leftarrow \text{clip}(w_{t-1} - \eta \frac{\partial C}{\partial w_b})$ $\frac{\partial C}{\partial w_t}$ cannot be computed

$b_t \leftarrow b_{t-1} - \eta \frac{\partial C}{\partial b_{t-1}}$

- weight binarization in forward path only
- gradients and weight accumulators are full precision

Results – Error rates

Method	MNIST	CIFAR-10	SVHN
No regularizer	$1.30 \pm 0.04\%$	10.64%	2.44%
BinaryConnect (det.)	$1.29 \pm 0.08\%$	9.90%	2.30%
BinaryConnect (stoch.)	$1.18 \pm 0.04\%$	**8.27%**	2.15%
50% Dropout	$1.01 \pm 0.04\%$		
Maxout Networks [29]	0.94%	11.68%	2.47%
Deep L2-SVM [30]	**0.87%**		
Network in Network [31]		10.41%	2.35%
DropConnect [21]			1.94%
Deeply-Supervised Nets [32]		9.78%	**1.92%**

- state-of-the-art results at the time of publication
- binarization acts as regularizer (improves generalization)
- tested on small datasets

NIPS'2016

Binarized Neural Networks

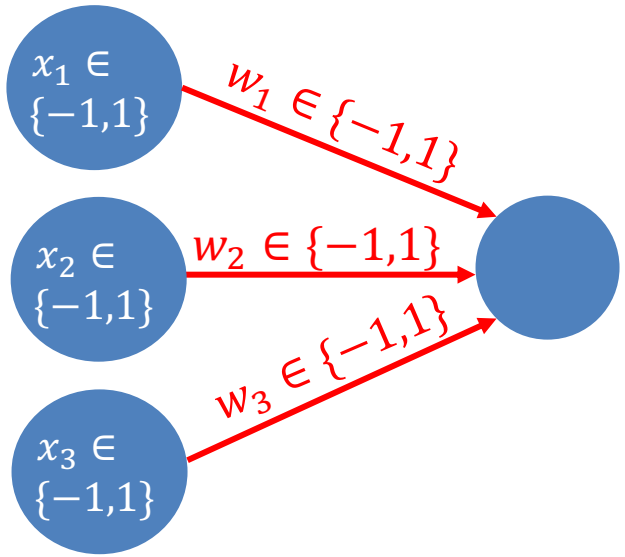
Itay Hubara[1]*
itayh@technion.ac.il

Matthieu Courbariaux[2]*
matthieu.courbariaux@gmail.com

Daniel Soudry[3]
daniel.soudry@gmail.com

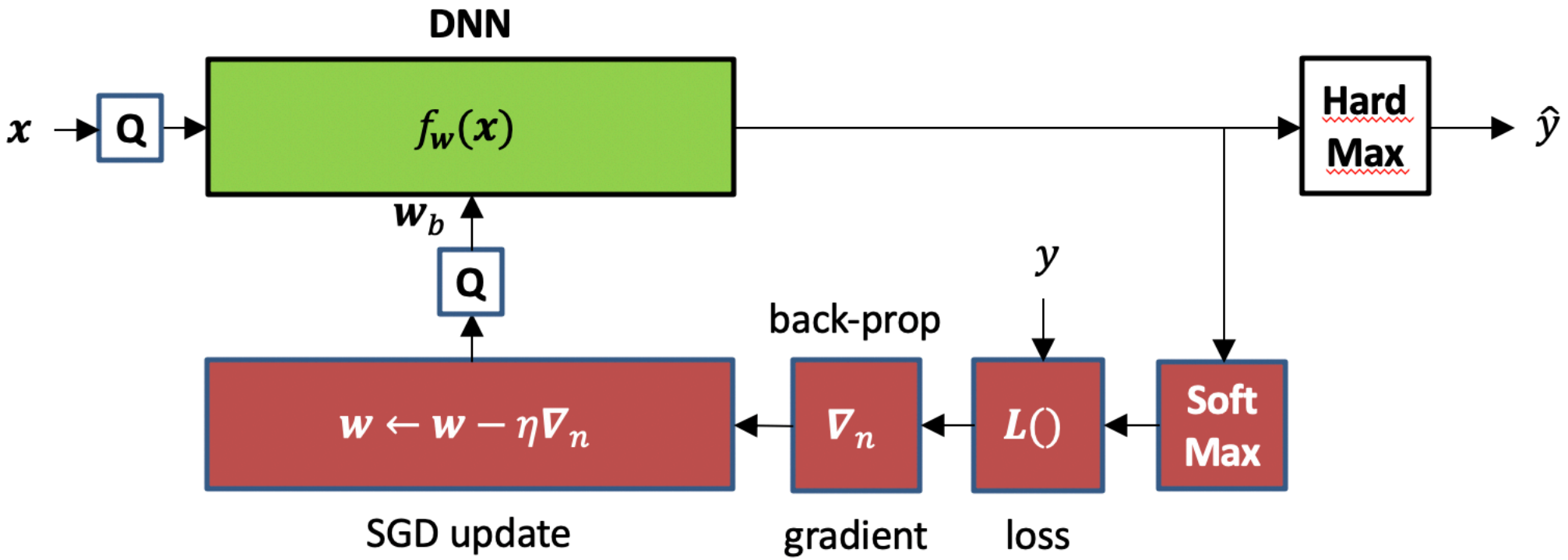
Ran El-Yaniv[1]
rani@cs.technion.ac.il

Yoshua Bengio[2,4]
yoshua.umontreal@gmail.com



- builds on top of BinaryConnect and introduces *activation binarization*

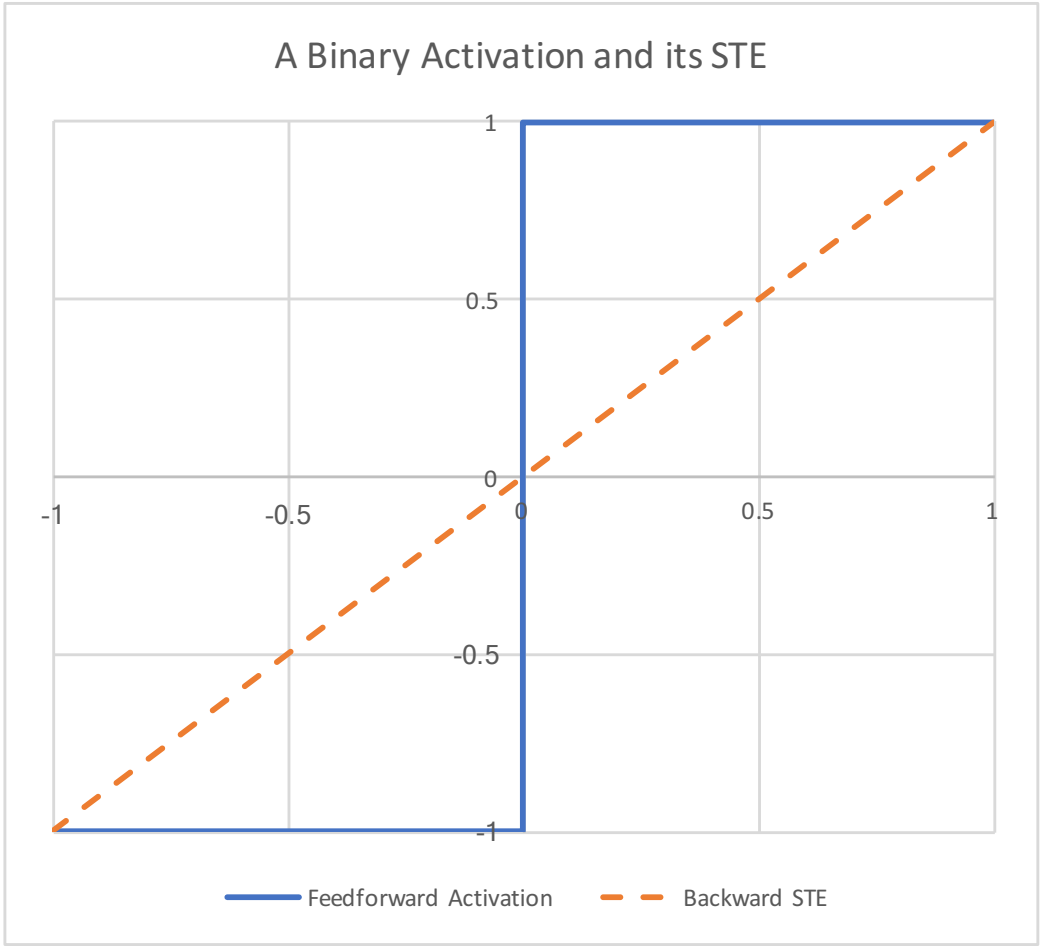
Key Innovations



- weight & activation binarization in forward path
- back-prop in full precision using *straight-through estimator* for differentiation

Straight-through Estimator

- treat the binary activation as a stochastic unit
- an estimate for its gradient is the identity function
- bypasses the problem of non-differentiability of binary activations
- surprisingly – it works!



Results

Data set	MNIST	SVHN	CIFAR-10
Binarized activations+weights, during training and test			
BNN (Torch7) stochastic	1.40%	2.53%	10.15%
BNN (Theano) deterministic	0.96%	2.80%	11.40%
Committee Machines' Array (Baldassi et al., 2015)	1.35%	-	-
Binarized weights, during training and test			
BinaryConnect (Courbariaux et al., 2015)	1.29± 0.08%	2.30%	9.90%
Binarized activations+weights, during test			
EBP (Cheng et al., 2015)	2.2± 0.1%	-	-
Bitwise DNNs (Kim & Smaragdis, 2016)	1.33%	-	-
Ternary weights, binary activations, during test			
(Hwang & Sung, 2014)	1.45%	-	-
No binarization (standard results)			
No regularization	1.3± 0.2%	2.44%	10.94%
Gated pooling (Lee et al., 2015)	-	1.69%	7.62%

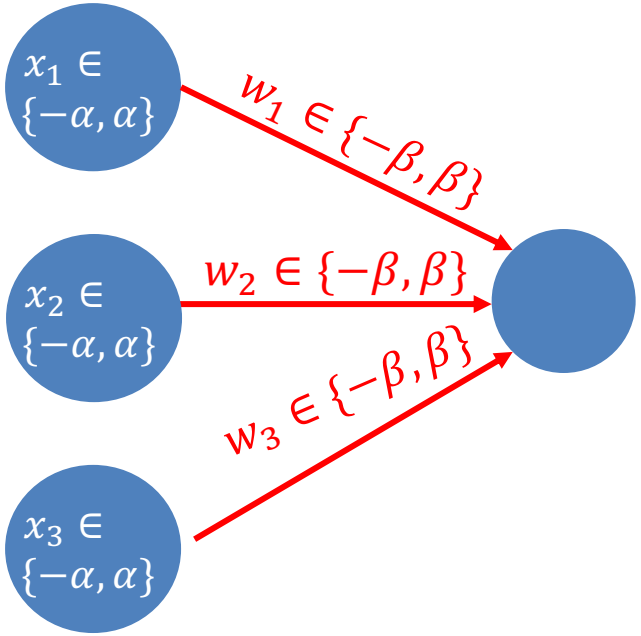
- much like BinaryConnect a year earlier:
  - state-of-the-art results at the time
  - tested on small datasets

XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks

[ECCV'16]

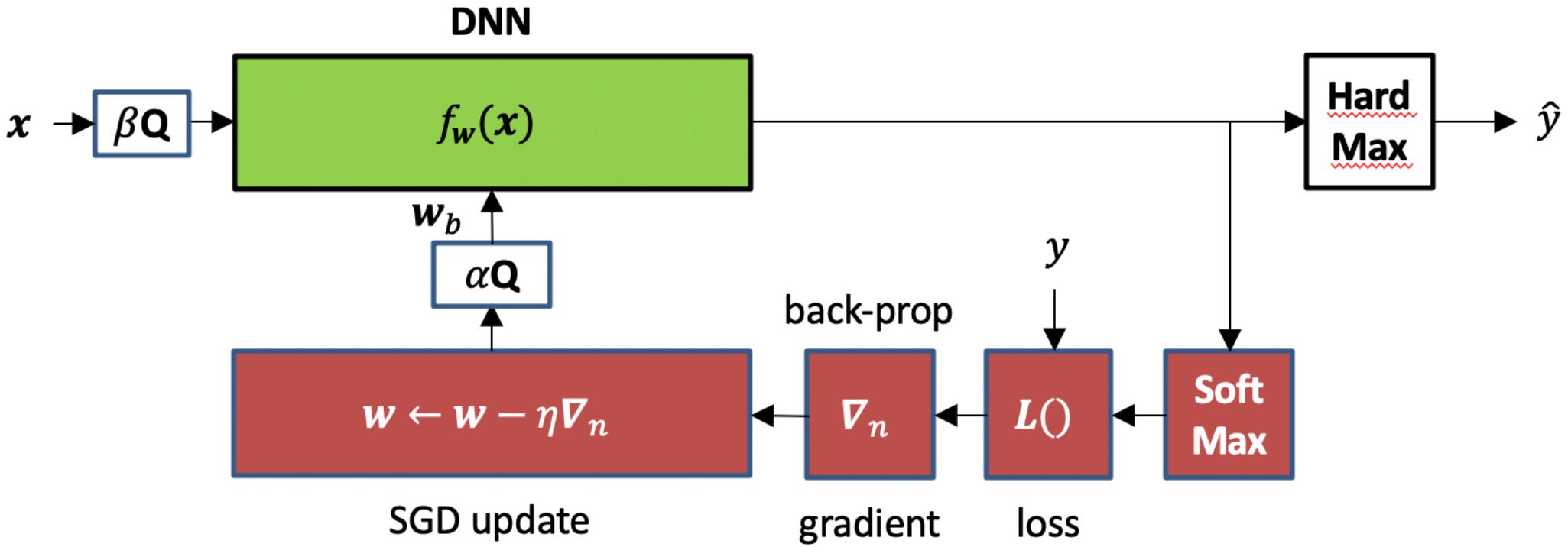
Mohammad Rastegari†, Vicente Ordonez†, Joseph Redmon*, Ali Farhadi†*

Allen Institute for AI†, University of Washington*
{mohammadr,vicenteor}@allenai.org
{pjreddie,ali}@cs.washington.edu



- famous for improving the accuracy of binarized networks

Key Innovation



- include scaling factors (α, β) in all binary tensors (weight and activations in forward path)
- learn the scaling factors

Binarization via Optimization

- weight binarization via L2 minimization between full precision and scaled binary (**BWN**)

$$J(\mathbf{B}, \alpha) = \|\mathbf{W} - \alpha\mathbf{B}\|^2$$
$$\alpha^*, \mathbf{B}^* = \underset{\alpha,\mathbf{B}}{\operatorname{argmin}} J(\mathbf{B}, \alpha)$$

$$\alpha^* = \frac{\left|\|\mathbf{W}\|\right|_1}{N}$$
$$\mathbf{B}^* = sign(\mathbf{W})$$

- weight/activation binarization via dot-product difference minimization between full precision and scaled binary

$$\alpha^*, \mathbf{B}^*, \beta^*, \mathbf{H}* = \underset{\alpha,\mathbf{B},\beta,\mathbf{H}}{\operatorname{argmin}} \|\mathbf{X} \odot \mathbf{W} - \beta\alpha\mathbf{H} \odot \mathbf{B}\|$$

Results

Classification Accuracy(%)									
Binary-Weight				Binary-Input-Binary-Weight				Full-Precision	
BWN		BC[11]		XNOR-Net		BNN[11]		AlexNet[1]	
Top-1	Top-5	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5
56.8	**79.4**	35.4	61.0	**44.2**	**69.2**	27.9	50.42	56.6	80.2

	ResNet-18		GoogLenet	
Network Variations	top-1	top-5	top-1	top-5
Binary-Weight-Network	60.8	83.0	65.5	86.1
XNOR-Network	51.2	73.2	N/A	N/A
Full-Precision-Network	69.3	89.2	71.3	90.0

- results reported on **ImageNet**
- near SOTA achieved only for binary-weighted AlexNet
- all other models have severe accuracy degradation compared to baseline

Ternary weight networks (TWN)

Fengfu Li and Bo Zhang
Institute of Applied Math., AMSS, CAS
Beijing, China
lifengfu12@mails.ucas.ac.cn;
b.zhang@amt.ac.cn

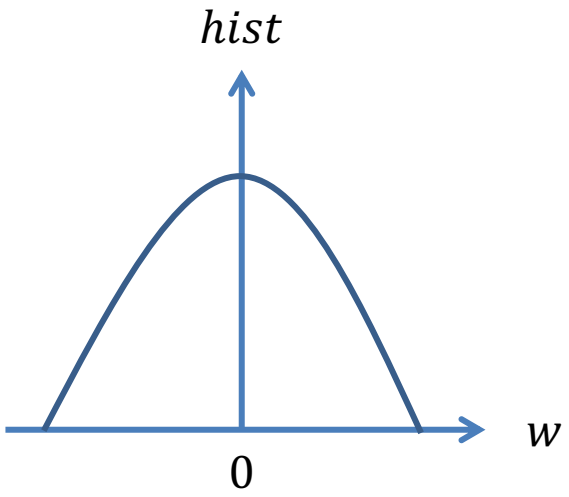
Bin Liu
Moshanghua Tech Co., Ltd.
Beijing, China
liubin@dress-plus.com

arXiv 2017

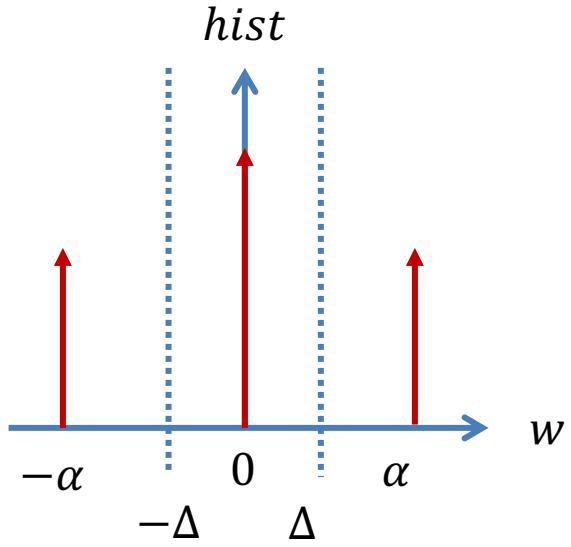
- First work to suggest weight ternarization ($w \in \{-\alpha, 0, \alpha\}$)
- higher expressiveness than binary (3x3 filter 2^9 vs. 3^9 templates)
- model compression: 2-b storage per weight -> 16x (32x) compression vs. 32-b (64-b) floating pt.
- multiply-accumulate complexity similar to BWNs (0 is not processed)

Key Innovation

before ternarization



after ternarization



Full-precision

ternary

$$w_l^t = \begin{cases} W_l : \tilde{w}_l > \Delta_l \\ 0 : |\tilde{w}_l| \leq \Delta_l \\ -W_l : \tilde{w}_l < -\Delta_l \end{cases}$$

Ternarization via Optimization

- Find the optimal ternary approximation $\alpha \boldsymbol{w}_t$ from the full precision weight vector $\boldsymbol{w} \in \mathbb{R}^n$:

$$\alpha^*, \boldsymbol{w}_t^* = \underset{\alpha, \boldsymbol{w}_t}{\operatorname{argmin}} \|\boldsymbol{w} - \alpha \boldsymbol{w}_t\|_2^2$$

$$\text{where } \alpha > 0, \boldsymbol{w}_t \in \{-1, 0, 1\}^n$$

- the use of a full precision scale α (inspired from XNORnet) improves the test accuracy compared to a strictly ternary weighted network, at the expense of an extra multiplication/DP
- Training of the network uses STE same as previous networks

Simulation Set-up

	MNIST	CIFAR-10	ImageNet
network architecture	LeNet-5	VGG-7	ResNet-18(B)
α → weight decay	1e-4	1e-4	1e-4
mini-batch size of BN	50	100	64 ($\times 4$)[2]
η → initial learning rate	0.01	0.1	0.1
learning rate decay[3] epochs	15, 25	80, 120	30, 40, 50
γ → momentum	0.9	0.9	0.9

- weight decay (L_2 regularization)
- multi-step learning rate schedule
- SGD with momentum

$$\boldsymbol{g}_n = \alpha \boldsymbol{w}_{n-1} + \nabla_{\boldsymbol{w}} J(\boldsymbol{w}; \boldsymbol{x}, y)$$

$$\boldsymbol{v}_n = \gamma \boldsymbol{v}_{n-1} + \eta \boldsymbol{g}_n$$

$$\boldsymbol{w}_{n+1} = \boldsymbol{w}_n - \boldsymbol{v}_n$$

Results

	MNIST	CIFAR-10	ImageNet (top-1)	ImageNet (top-5)
TWNs	**99.35**	**92.56**	**61.8 / 65.3**	**84.2 / 86.2**
BPWNs	99.05	90.18	57.5 / 61.6	81.2 / 83.9
FPWNs	99.41	92.88	65.4 / 67.6	86.76 / 88.0
BinaryConnect	98.82	91.73	-	-
Binarized Neural Networks	88.6	89.85	-	-
Binary Weight Networks	-	-	60.8	83.0
XNOR-Net	-	-	51.2	73.2

- LeNet5 (MNIST), VGG7 (CIFAR-10), and ResNet18(ImageNet) network architectures
- achieved state-of-the art results at the time of publication

TRAINED TERNARY QUANTIZATION (TTQ)

ICLR 2017

Chenzhuo Zhu*
Tsinghua University
zhucz13@mails.tsinghua.edu.cn

Song Han
Stanford University
songhan@stanford.edu

Huizi Mao
Stanford University
huizi@stanford.edu

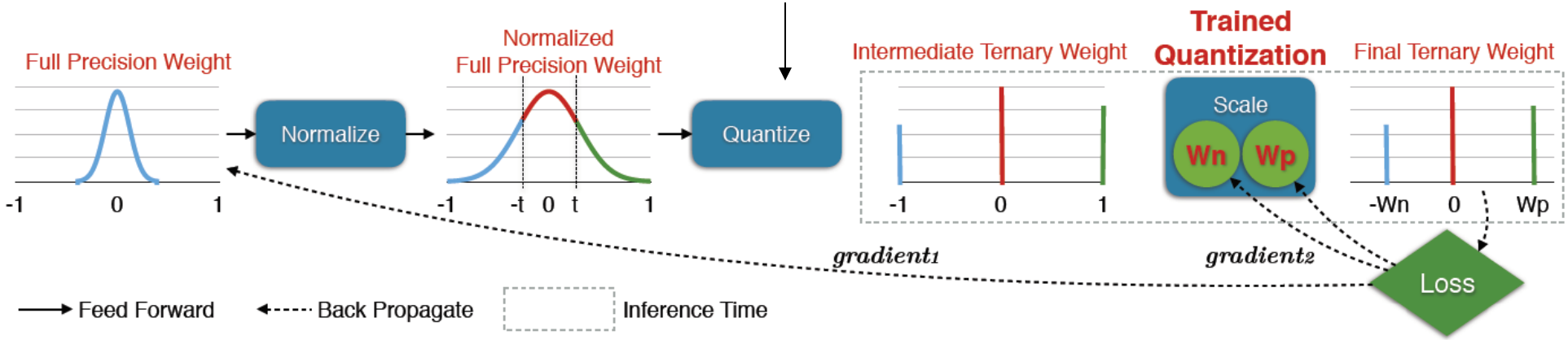
William J. Dally
Stanford University
NVIDIA
dally@stanford.edu

$$w_l^t = \begin{cases} W_l^p : \tilde{w}_l > \Delta_l \\ 0 : |\tilde{w}_l| \leq \Delta_l \\ -W_l^n : \tilde{w}_l < -\Delta_l \end{cases}$$

- similar to TWN but with asymmetric levels W^p & W^n
- learns the scale factors and thresholds via backprop

Key Innovation

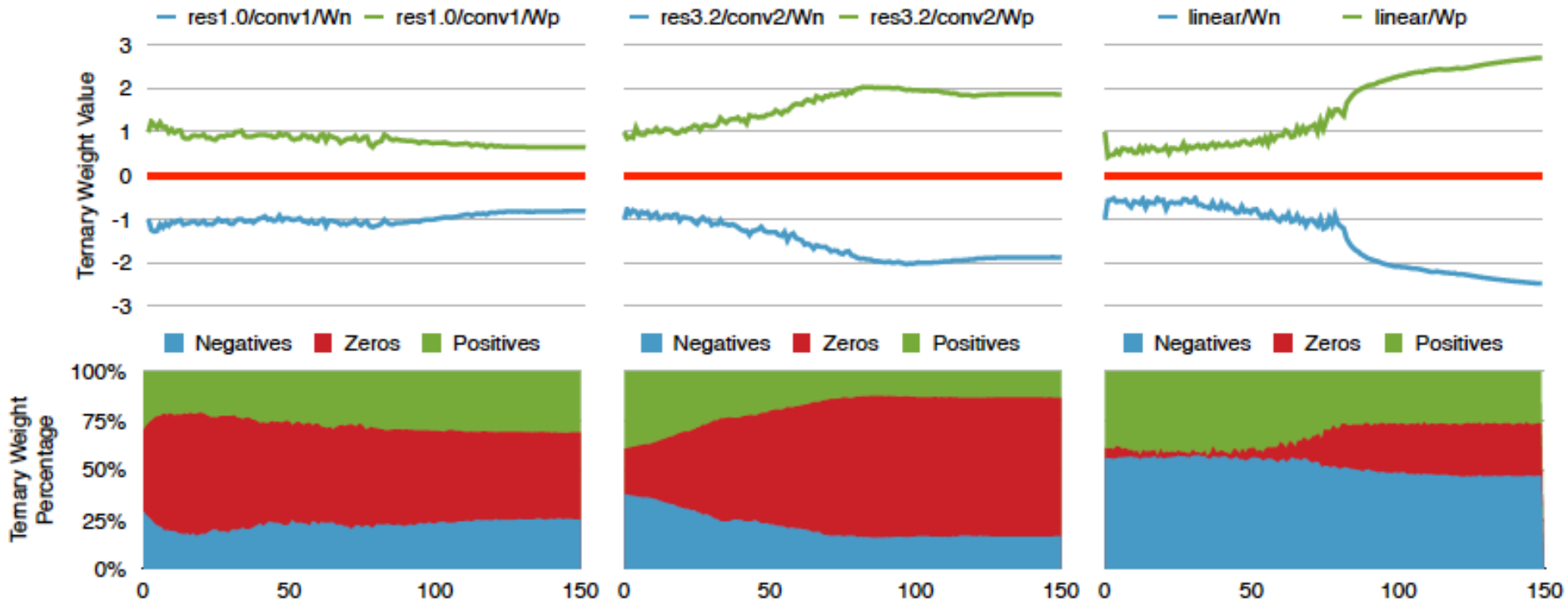
$$\Delta_l = t \times \max(|\tilde{w}|)$$



- S1: normalize full-precision weights in each layer
- S2: ternary quantize - intermediate ternary weights (t is a hyperparameter – set to 0.05 in simulations)
- S3: ternary assignment – assign $+1 \rightarrow W^p$ and $-1 \rightarrow W^n$
- S4: SGD update – scale factors (W^p & W^n) and full-precision weights
- Repeat S1
- use ternary weights only during inference

Simulations

(ResNet-20 on CIFAR-10)



- fraction of +ve, -ve and 0 weights varies over iterations, and across layers

Results

CIFAR10

Model	Full resolution	Ternary (Ours)	Improvement
ResNet-20	8.23	**8.87**	**-0.64**
ResNet-32	7.67	**7.63**	**0.04**
ResNet-44	7.18	**7.02**	**0.16**
ResNet-56	6.80	**6.44**	**0.36**

ImageNet

AlexNet

Error	Full precision	1-bit (DoReFa)	2-bit (TWN)	2-bit (Ours)
Top1	42.8%	46.1%	45.5%	**42.5%**
Top5	19.7%	23.7%	23.2%	**20.3%**

ResNet18

Error	Full precision	1-bit (BWN)	2-bit (TWN)	2-bit (Ours)
Top1	30.4%	39.2%	34.7%	**33.4%**
Top5	10.8%	17.0%	13.8%	**12.8%**

LQ-Nets: Learned Quantization for Highly Accurate and Compact Deep Neural Networks

ECCV 2018

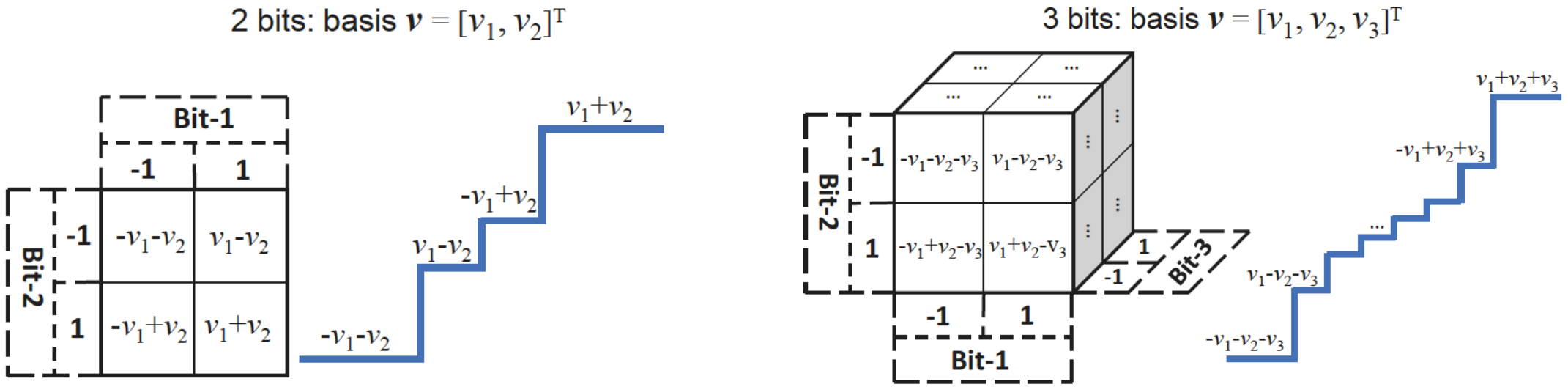
Dongqing Zhang*, Jiaolong Yang*, Dongqiangzi Ye*, and Gang Hua

Microsoft Research

zdqzeros@gmail.com jiaoyan@microsoft.com eowinye@gmail.com ganghua@microsoft.com

- extended binary weights to “multi-bit” representation
- DPs are still binary weighted (hardware friendly), but weight precision is multibit (higher accuracy)

Key Innovation



- jointly learns weights and quantizer parameters
- quantization as a dot-product - $x_q =< \boldsymbol{vb} >$

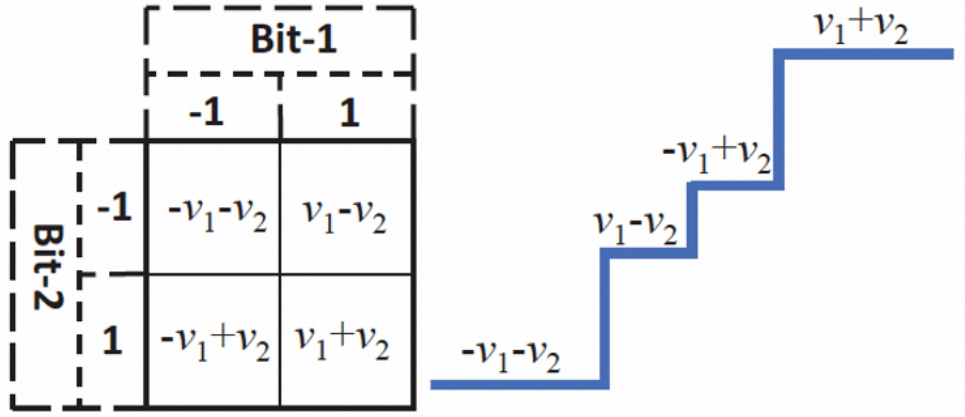
standard quantization

bit vector

$$q = \left\langle \begin{bmatrix} 1 \\ 2 \\ \dots \\ 2^{K-1} \end{bmatrix}, \begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_K \end{bmatrix} \right\rangle$$

fixed basis

2 bits: basis $\boldsymbol{v} = [v_1, v_2]^{\mathrm{T}}$



- quantization as a dot-product

learnable basis

$$Q_{\text{ours}}(x, \mathbf{v}) = \mathbf{v}^{\mathrm{T}}\mathbf{e}_l, \qquad \text{if } x \in (t_l, t_{l+1}],$$

- optimum thresholds – mean of adjacent quantization levels (recall LM algorithm)

LQNets via Optimization

- Similar to XNORNets and TWN, the quantization is optimization based (estimating the scales and the quantization assignment):

full precision weights $x \in \mathbb{R}^{K \times N}$

$$\mathbf{v}^*, B^* = \arg\min_{\mathbf{v}, B} \left\| B^{\mathrm{T}}\mathbf{v} - \mathbf{x} \right\|_2^2, \quad s.t.\ B \in \{-1, 1\}^{K \times N}$$

optimal basis vector optimal binary assignment matrix

- Also uses the STE to train the network in the presence of quantization
- Can be used for both weights and activations

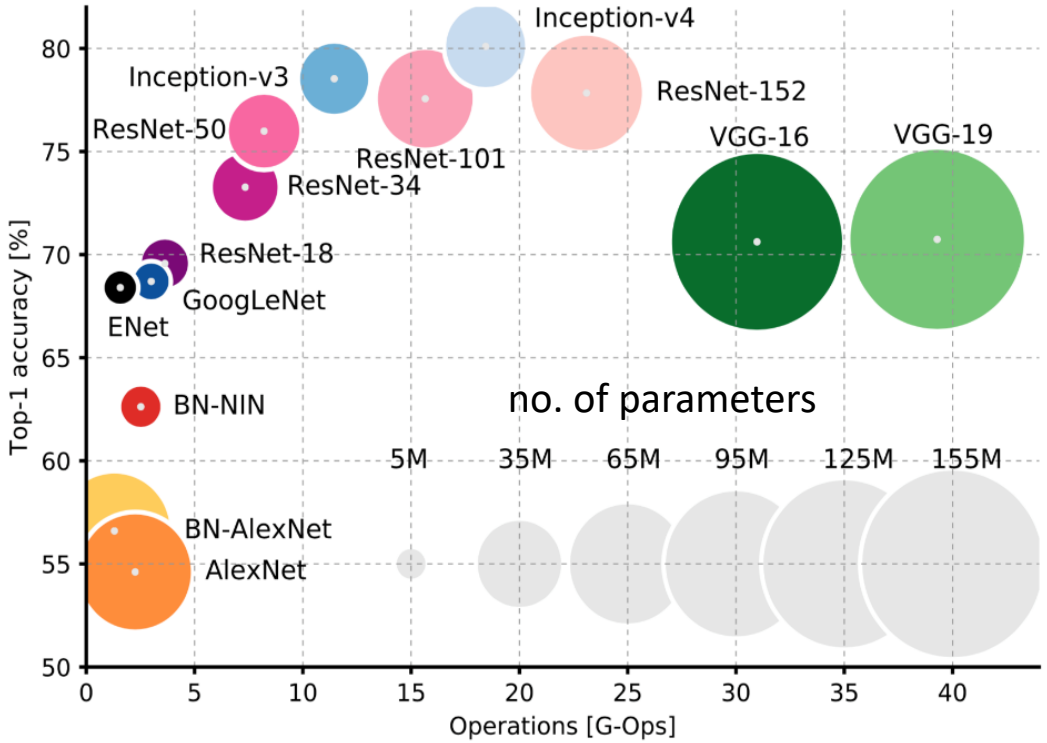
Results

Methods	Bit-width (W/A)	Accuracy(%) Top-1	Top-5
ResNet-18			
FP* [15]	32/32	69.6	89.2
FP-ours	32/32	70.3	89.5
BWN [39]	1/32	60.8	83.0
TWN [29]	2/32	61.8	84.2
TWN† [29]	2/32	65.3	86.2
TTQ† [54]	2/32	66.6	87.2
Ours	2/32	**68.0**	**88.0**
Ours	3/32	**69.3**	**88.8**
Ours	4/32	**70.0**	**89.1**
XNOR-Net [39]	1/1	51.2	73.2
DoReFa-Net‡ [53]	1/2	53.4	-
DoReFa-Net‡ [53]	1/4	59.2	-
HWGQ [3]	1/2	59.6	82.2
ABC-Net [33]	3/3	61.0	83.2
ABC-Net [33]	5/5	65.0	85.9
Ours	1/2	**62.6**	**84.3**
Ours	2/2	**64.9**	**85.9**
Ours	3/3	**68.2**	**87.9**
Ours	4/4	**69.3**	**88.8**

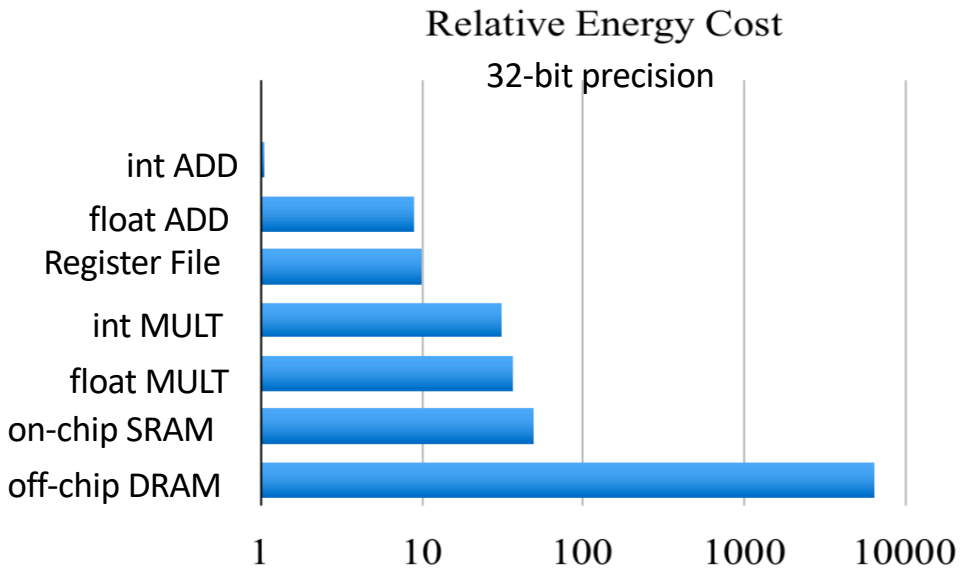
- State-of-the art results (at the time) for ResNet18 on ImageNet
- 2/32 denotes 2b weights and full precision activations
- Outperforms TWN and TTQ at iso-bit precision (2/32)

Model Compression

Memory Access Energy in DNNs



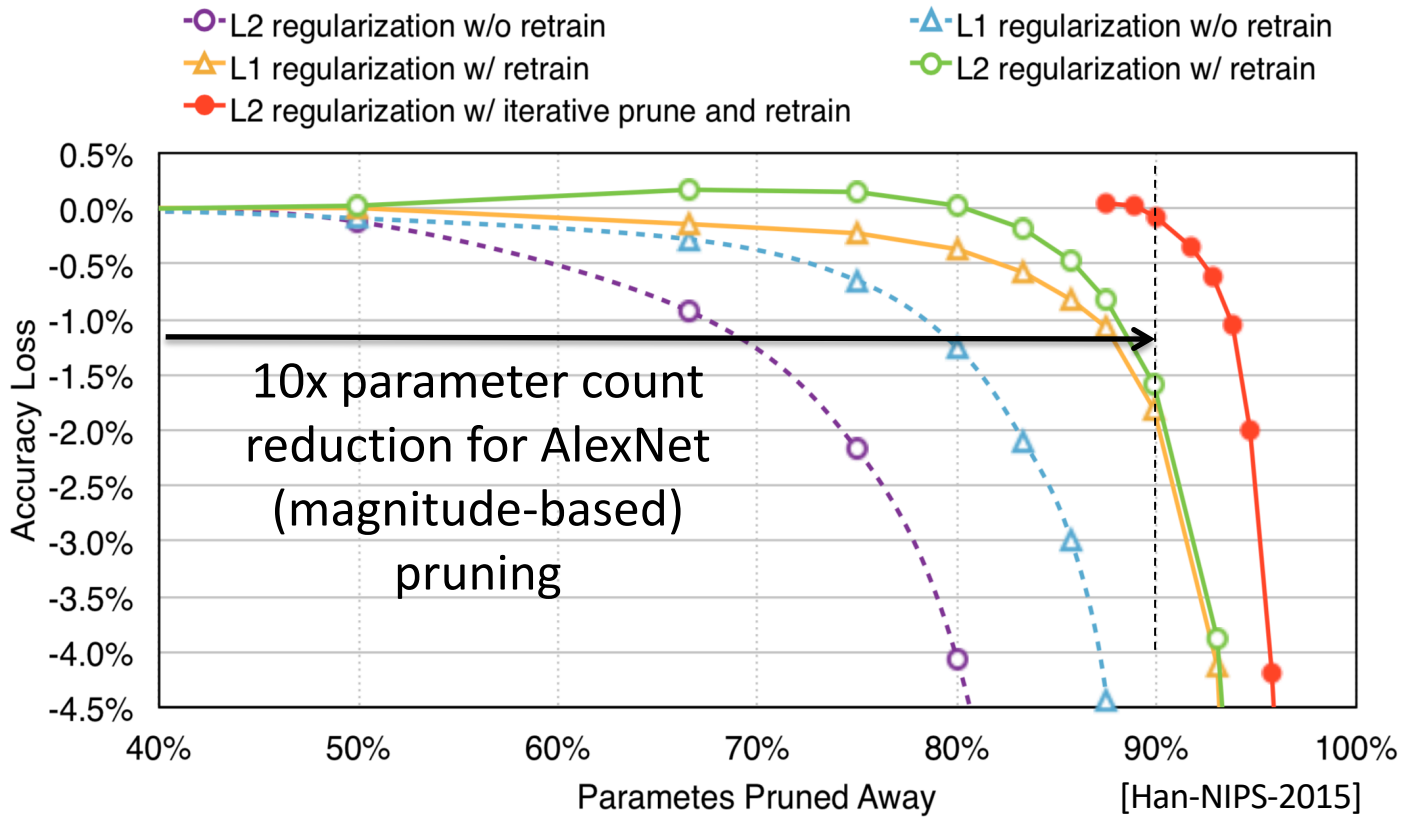
[Canziani-arXiv-2017]



[Horowitz-ISSCC-2014]

- Require large no. of parameters → don't fit within on-chip SRAM
- off-chip DRAM accesses are 100x more energy expensive

The Compression Opportunity



- DNNs are over-parametrized
- magnitude-based pruning: requires retraining of several hundred epochs
- retraining: expensive and slow, requires original training data

Learning both Weights and Connections for Efficient Neural Networks

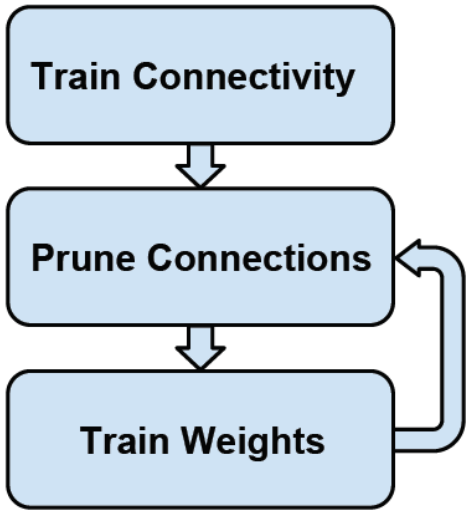
[NeurIPS'15]

Song Han
Stanford University
songhan@stanford.edu

Jeff Pool
NVIDIA
jpool@nvidia.com

John Tran
NVIDIA
johntran@nvidia.com

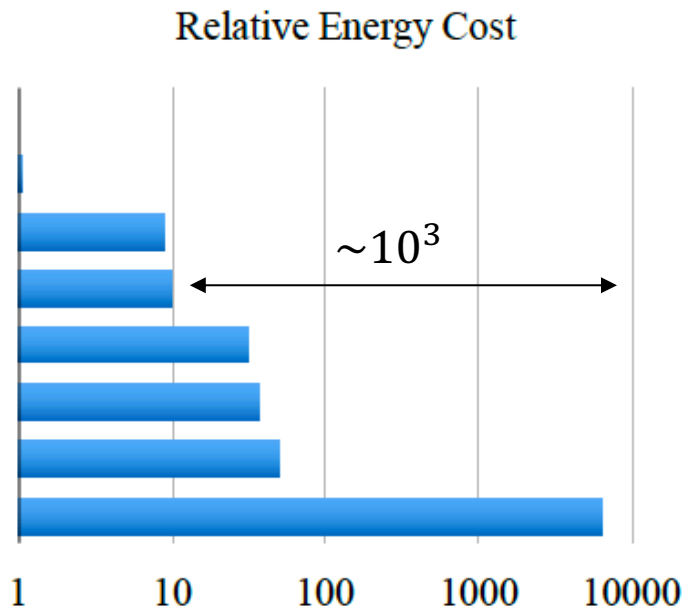
William J. Dally
Stanford University
NVIDIA
dally@stanford.edu



- early work on pruning network connections to save storage costs

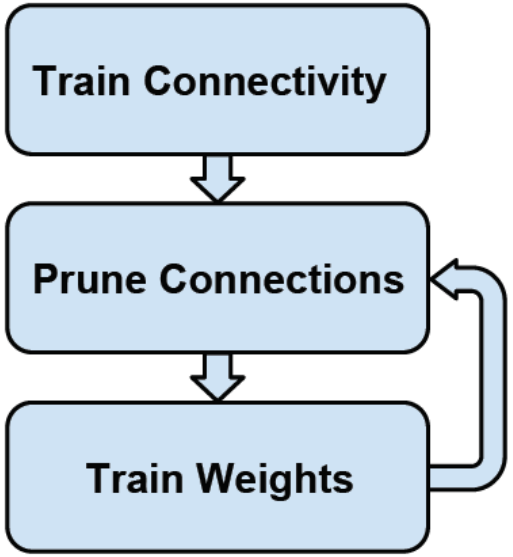
45nm CMOS process

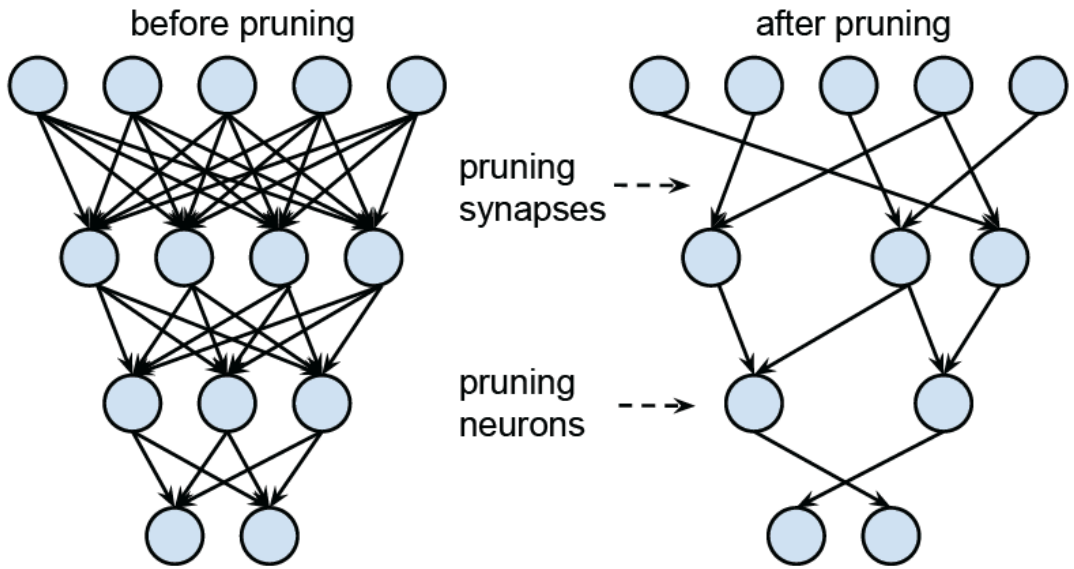
Operation	Energy [pJ]	Relative Cost
32 bit int ADD	0.1	1
32 bit float ADD	0.9	9
32 bit Register File	1	10
32 bit int MULT	3.1	31
32 bit float MULT	3.7	37
32 bit SRAM Cache	5	50
32 bit DRAM Memory	**640**	**6400**



- memory access is 3-orders-of-magnitude more than arithmetic
- 1B parameter network → (20Hz) × (10^9) × (640pJ) = 12.8W for DRAM accesses only

Approach





- Step 1 – vanilla training with L2 regularization
- Step 2 – prune low-weight connections
- Step 3 – re-train sparse network with scaled drop-out ratio

# of connections $\longrightarrow C_i = N_i N_{i-1}$ and $D_r = D_o \sqrt{\frac{C_{ir}}{C_{io}}}$ ← original network, ← pruned network

# of neurons

Results

Network	Top-1 Error	Top-5 Error	Parameters	Compression Rate
LeNet-300-100 Ref	1.64%	-	267K	
LeNet-300-100 Pruned	1.59%	-	**22K**	**12×**
LeNet-5 Ref	0.80%	-	431K	
LeNet-5 Pruned	0.77%	-	**36K**	**12×**
AlexNet Ref	42.78%	19.73%	61M	
AlexNet Pruned	42.77%	19.67%	**6.7M**	**9×**
VGG-16 Ref	31.50%	11.32%	138M	
VGG-16 Pruned	31.34%	10.88%	**10.3M**	**13×**

- CONV layer parameters are fixed; prune and re-train FC layers
- 9×-to-13× reduction (> 90%) in parameter complexity

Layer	Weights	FLOP	Act%	Weights%	FLOP%
	original	original	sparsity	pruned	pruned
fc1	235K	470K	38%	8%	8%
fc2	30K	60K	65%	9%	4%
fc3	1K	2K	100%	26%	17%
Total	266K	532K	46%	**8%**	**8%**
				12×	**12×**

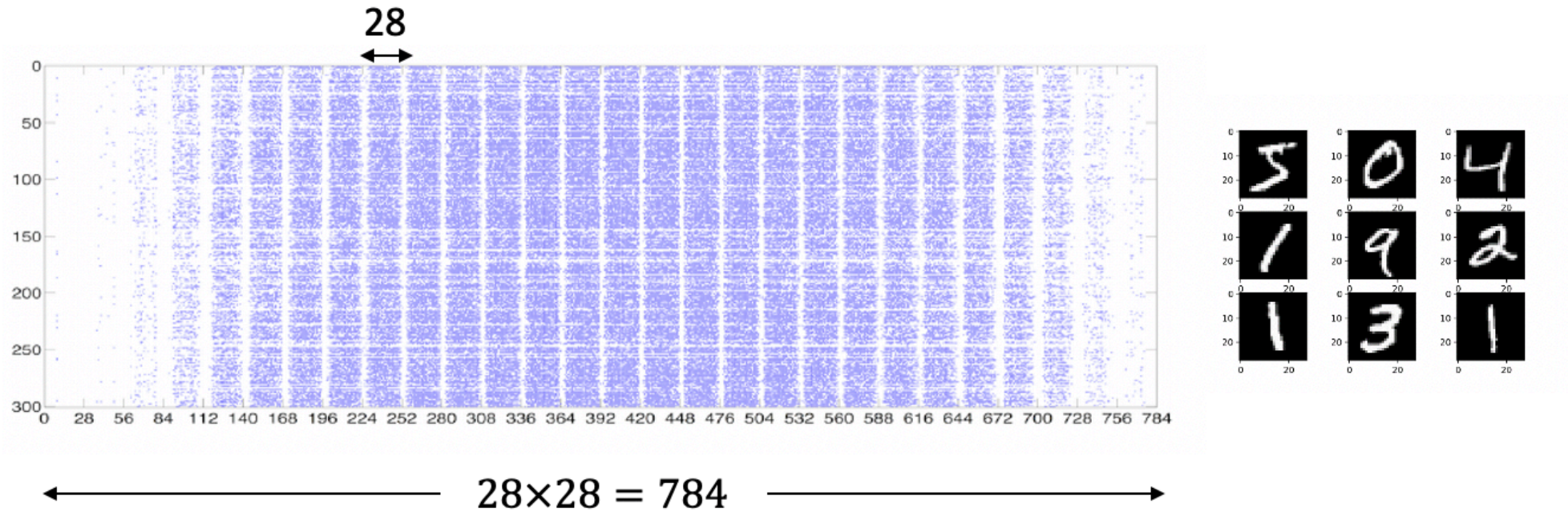
LeNet-300-100

Layer	Weights	FLOP	Act%	Weights%	FLOP%
conv1	0.5K	576K	82%	66%	66%
conv2	25K	3200K	72%	12%	10%
fc1	400K	800K	55%	8%	6%
fc2	5K	10K	100%	19%	10%
Total	431K	4586K	77%	**8%**	**16%**
				12×	**6×**

LeNet-5

- re-training with 1/10 of the original learning rate

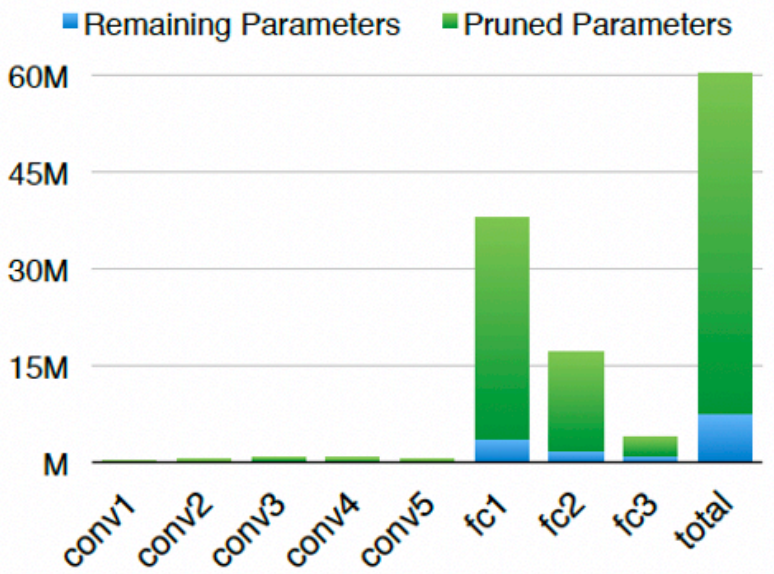
Sparsity Pattern



- fc1 layer of LeNet-300-100 → 300×784 matrix-vector multiply
- banded structure with width 28 corresponding to 28×28 input
- left and right sparsity due to unimportant regions at the top and bottom of the image

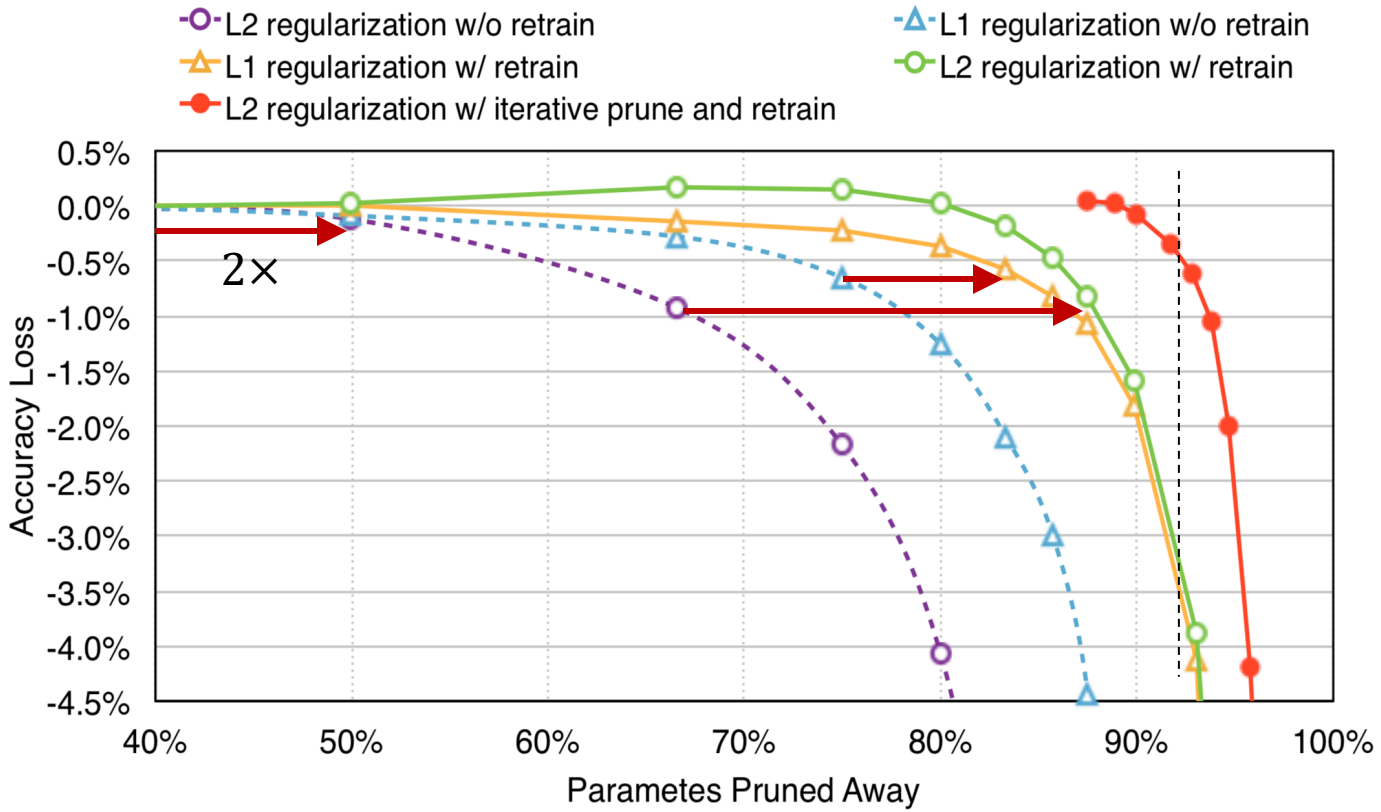
AlexNet on ImageNet

Layer	Weights	FLOP	Act%	Weights%	FLOP%
conv1	35K	211M	88%	84%	84%
conv2	307K	448M	52%	38%	33%
conv3	885K	299M	37%	35%	18%
conv4	663K	224M	40%	37%	14%
conv5	442K	150M	34%	37%	14%
fc1	38M	75M	36%	9%	3%
fc2	17M	34M	40%	9%	3%
fc3	4M	8M	100%	25%	10%
Total	61M	1.5B	54%	**11%**	**30%**
				9×	**3×**



- re-training with 1/100 of the original learning rate
- similar results for VGG-16

Accuracy vs. Pruning

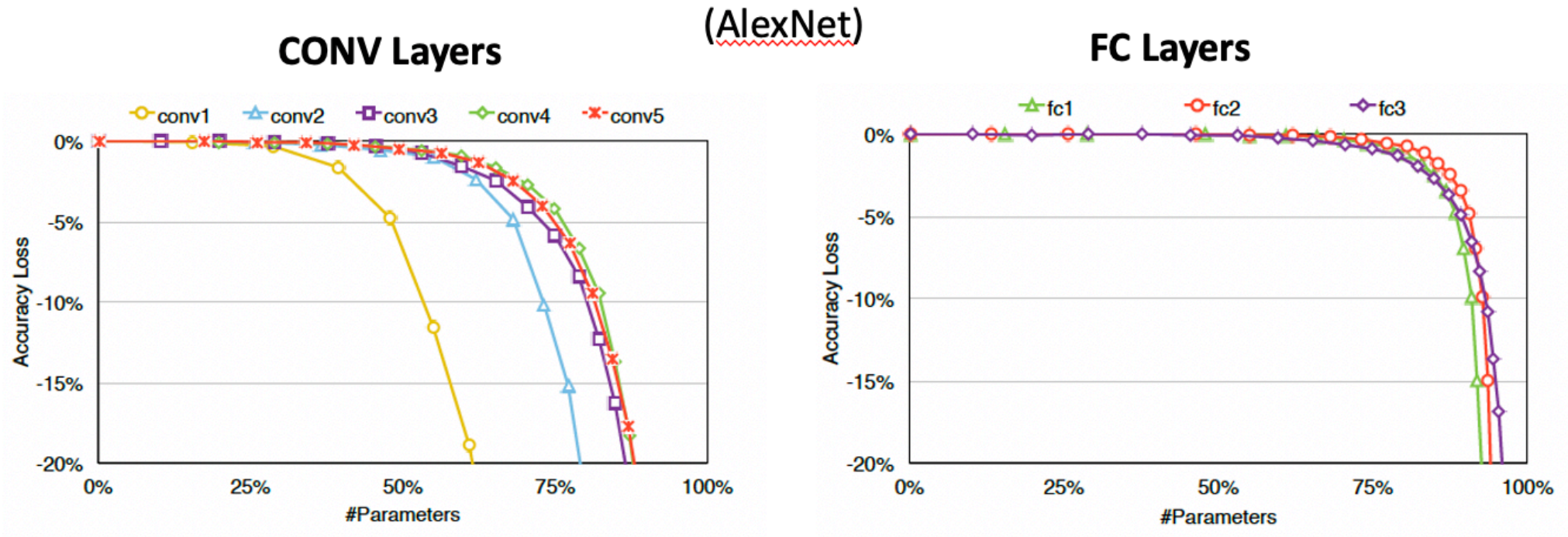


(AlexNet)

- 2X compression without retraining ("free lunch")
- for greater compression → retraining is important
- L1 regularizer better than L2 if pruning w.o. re-training
- L2 regularizer is better than L1 if pruning with re-training

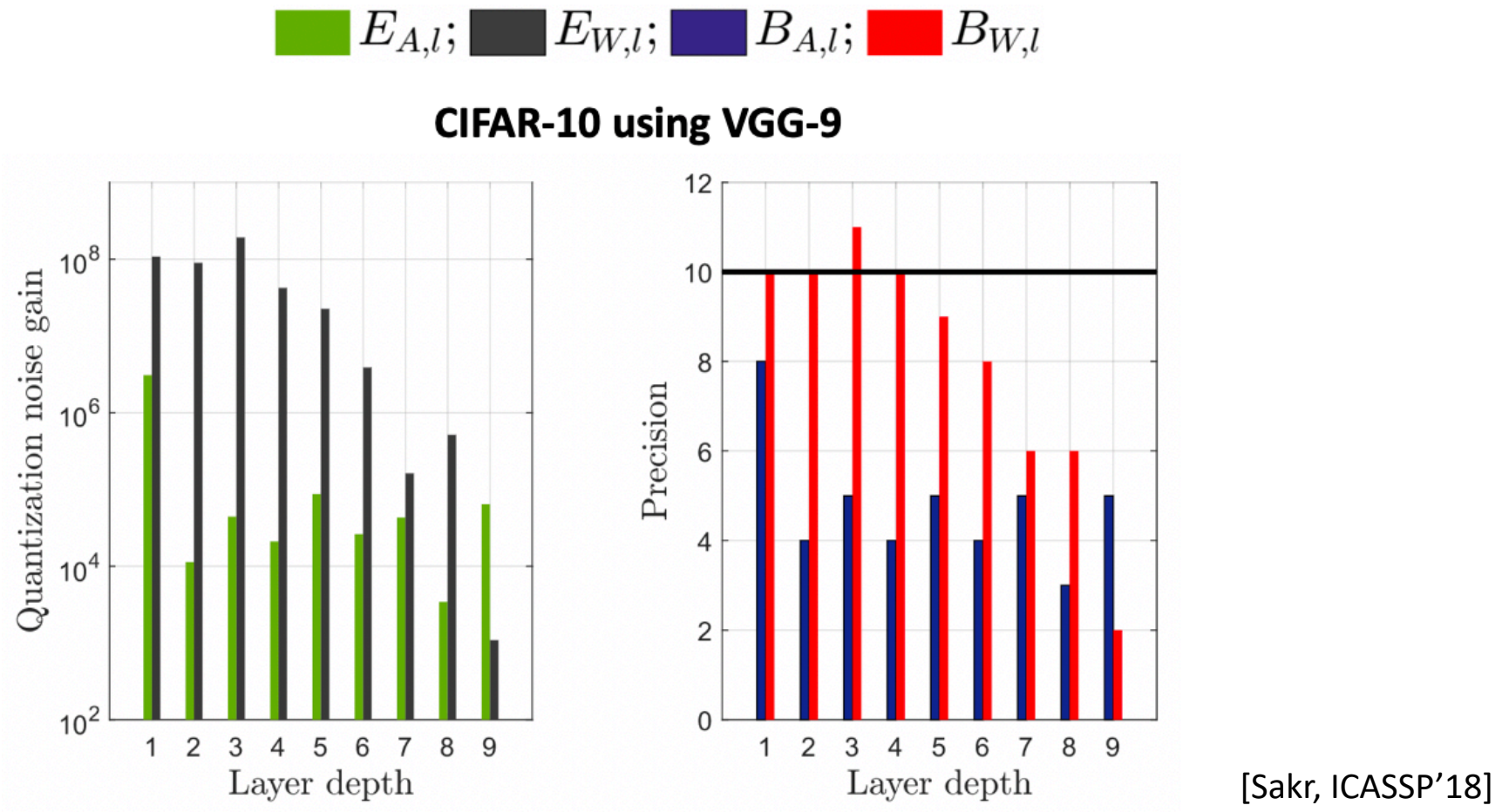
Pruning Sensitivity

(AlexNet)



- CONV layers more sensitive than FC layers (why?)
- CONV1 layer most sensitive (why?)….paper claims it has less redundancy due to having only 3 channels

Lecture 5 – Precision Decreases with Depth



[Sakr, ICASSP'18]

- precision decreases with depth → noise gains are higher in early layers

Summary

- DNNs are overprovisioned → many opportunities to compress network parameters
- binarization/ternarization compresses by reducing precision
- model compression prunes the network connections
- both need to trade-off accuracy with complexity
- both need re-training to minimize accuracy loss

Course Web Page

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

http://shanbhag.ece.uiuc.edu