Deep Learning

Lecture 6: Model and Dataset Compression

National University of Singapore

CEG5304/EE6934

Instructor: Joey Tianyi Zhou

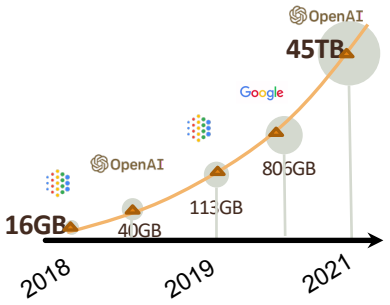
Network Compression and Speedup

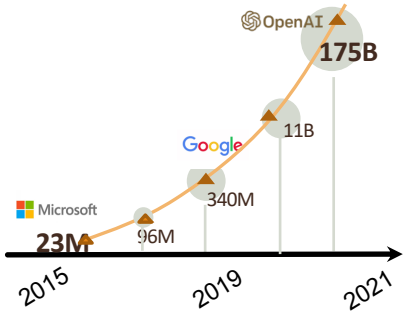
Present AI is Not Sustainable!

Current AI trajectory

Big data to the extreme

GPT-3 trained on 300 billions tokens





Big model to the extreme

175 billion model parameters

Issues

❖ **Memory& Storage**

Memory = 786 GB

Storage = 45 TB

❖ **Computing Power**

GPT-3 model = 355 years

GPT-3 model = $ 4.6M

❖ **Environment**

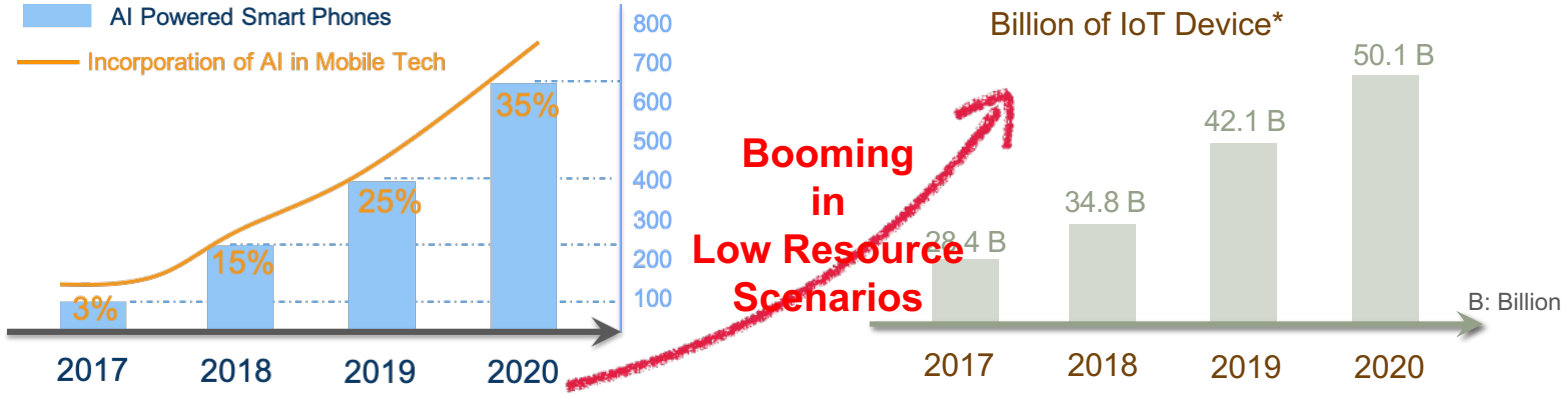
650	SG-NY Flight
126,000	US Car/Year
626,155	Single AI model

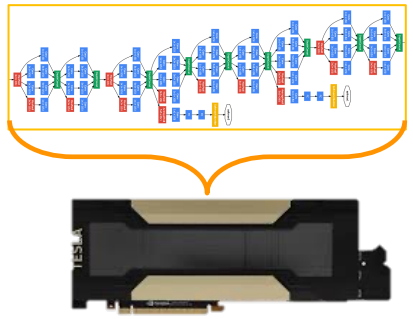
CO_2 Emission* → *lbs*

** From OpenAI and Icao*

Confidential

Low Resource Computing





Big Model

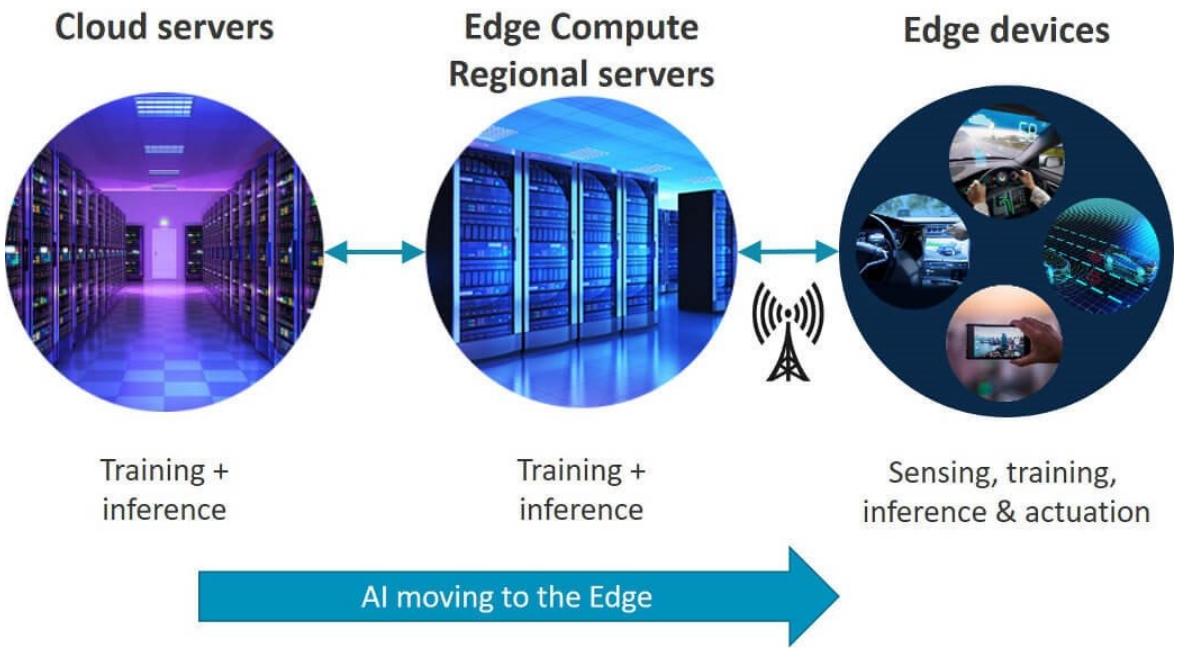
Memory:32GB
Computation: TFLOPS/s

Mobile AI

Memory:4GB
Computation: GFLOPS/s

Confidential

A New Era of AI on Edge Device



- Cite from “https://community.arm.com/”

Popularity

- cell phones
- wearable device
- IoT devices

Locally processed

- low latency
- data security
- low cost

Challenges

Edge device

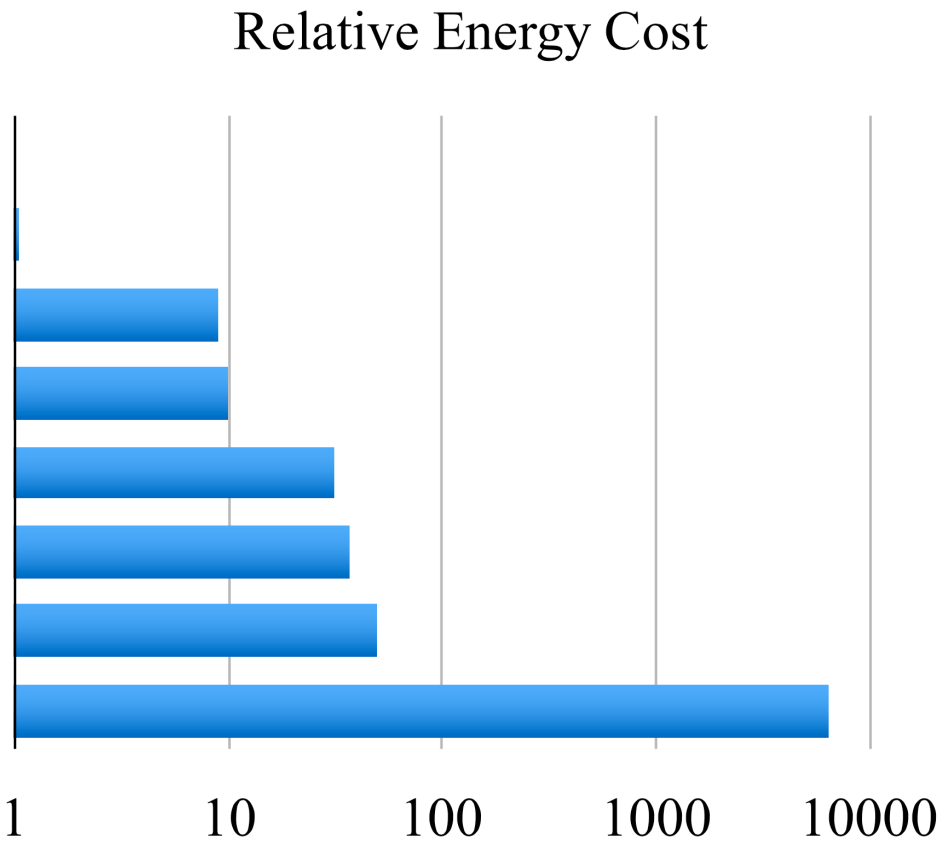
- Most of them supported by battery, demand high energy efficiency
- Sensitive to cost (area overhead)

Complicated CNN model

- VGG-16:
  - 528MB weights stored in off-chip memory
  - 1.5×10^{10} MAC (multiply-accumulate) operations
- Low energy efficiency
- Low area efficiency

Why smaller models?

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



Source: http://isca2016.eecs.umich.edu/wp-content/uploads/2016/07/4A-1.pdf

Outline

Weight Pruning

Quantization method

Pruning + Quantization + Encoding

Hardware-aware Compression

Outline

Weight Pruning

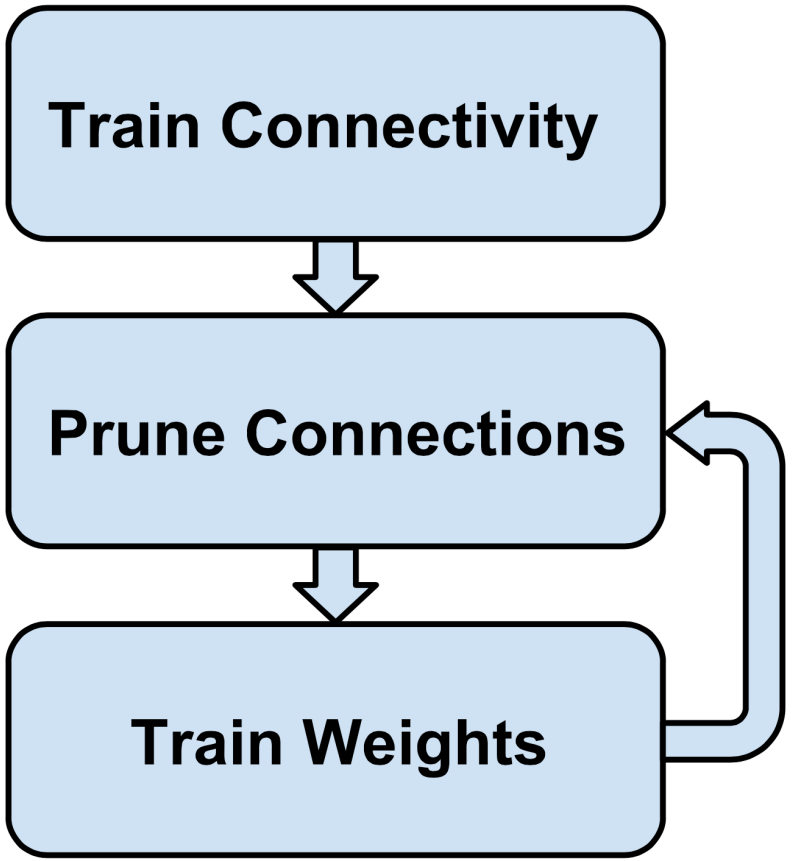
- Magnitude-based method
  - Iterative pruning + Retraining
  - Pruning with rehabilitation
- Hessian-based method

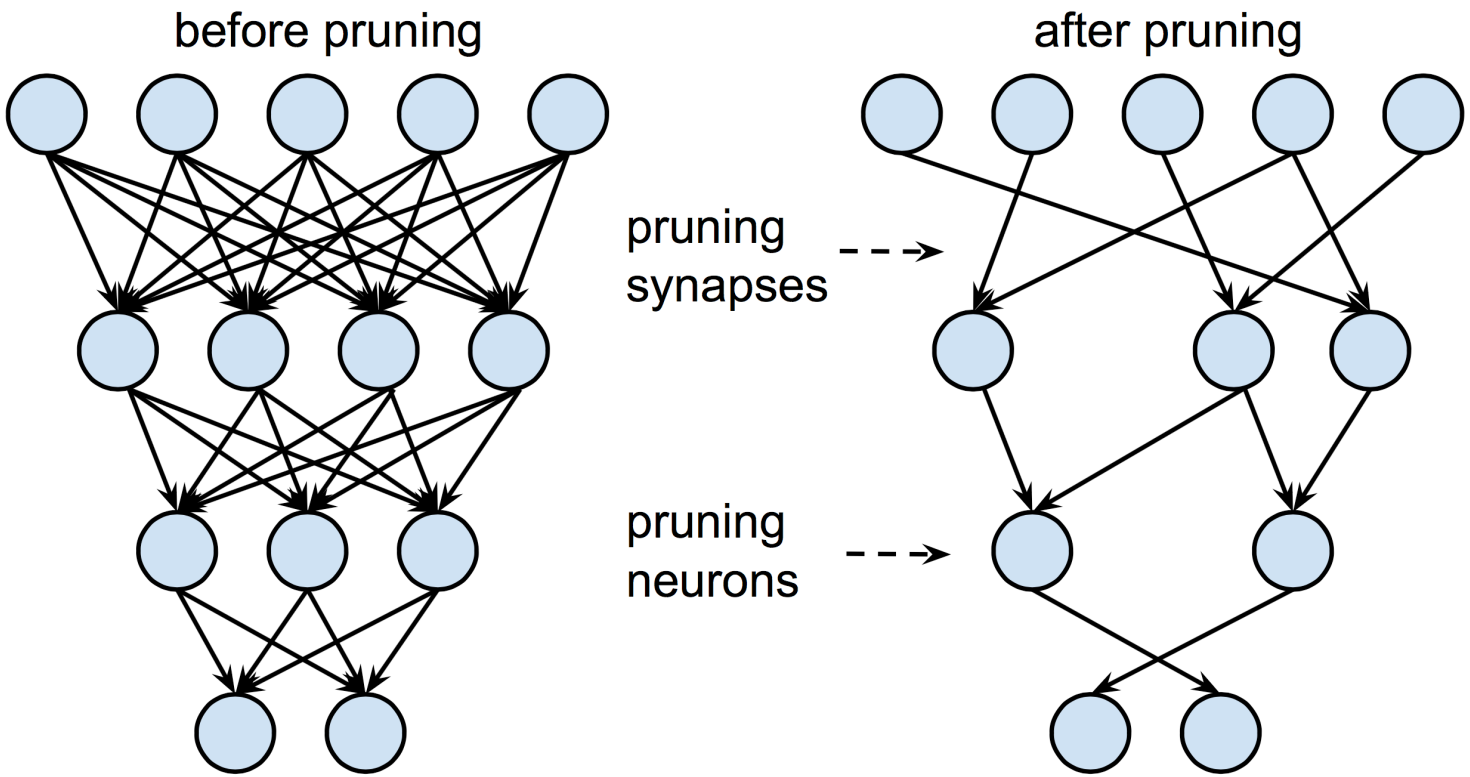
Quantization method

Pruning + Quantization + Encoding

Hardware-aware Compression

Magnitude-based method: Iterative Pruning + Retraining





Han, Song, et al. "Learning both weights and connections for efficient neural network." NIPS. 2015.

Magnitude-based method: Iterative Pruning + Retraining (Algorithm)

1. Choose a neural network architecture.
2. Train the network until a reasonable solution is obtained.
3. Prune the weights of which magnitudes are less than a threshold τ.
4. Train the network until a reasonable solution is obtained.
5. Iterate to step 3.

Han, Song, et al. "Learning both weights and connections for efficient neural network." NIPS. 2015.

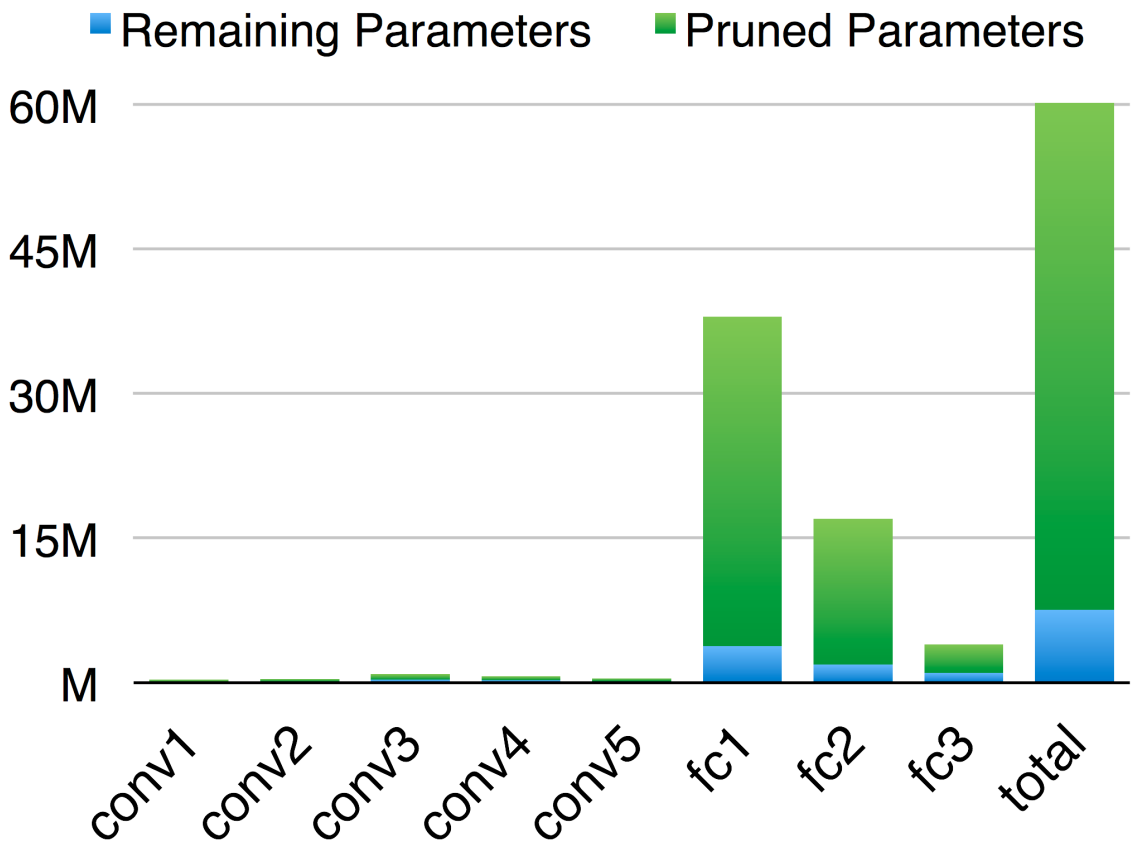
Magnitude-based method: Iterative Pruning + Retraining (Experiment: Overall)

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

Han, Song, et al. "Learning both weights and connections for efficient neural network." NIPS. 2015.

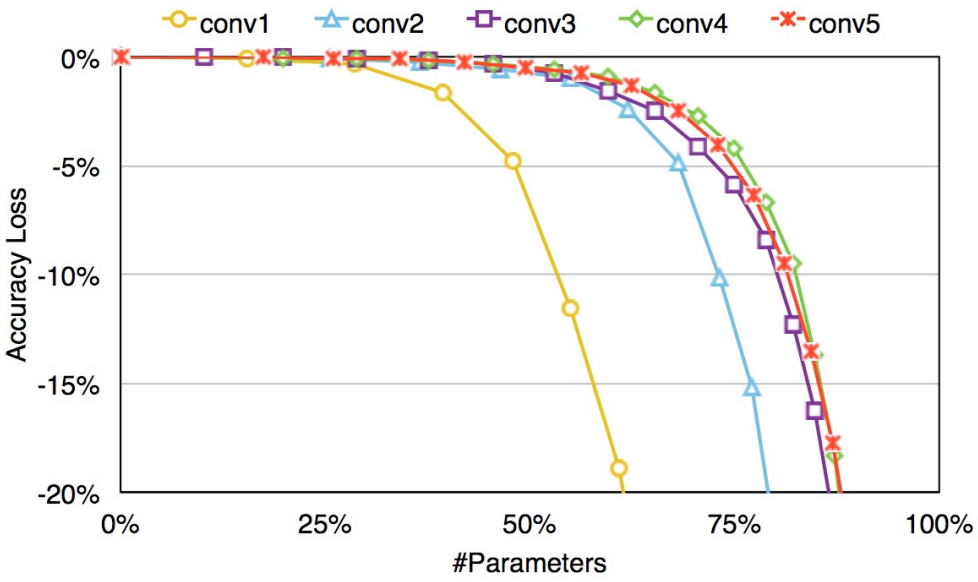
Magnitude-based method: Iterative Pruning + Retraining (Experiment: AlexNet)

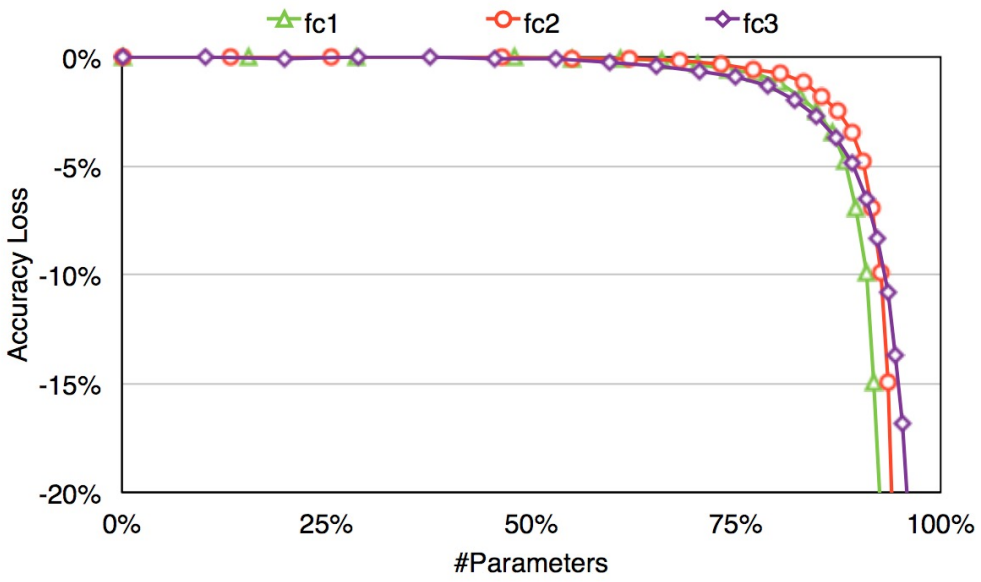
Layer	Weights	FLOP	Act%	Weights%	FLOP%
conv1	35K	211M	88%	84%	84%
conv2	307K	448M	52%	38%	33%
conv3	885K	299M	37%	35%	18%
conv4	663K	224M	40%	37%	14%
conv5	442K	150M	34%	37%	14%
fc1	38M	75M	36%	9%	3%
fc2	17M	34M	40%	9%	3%
fc3	4M	8M	100%	25%	10
Total	61M	1.5B	54%	11%	30%



Han, Song, et al. "Learning both weights and connections for efficient neural network." NIPS. 2015.

Magnitude-based method: Iterative Pruning + Retraining (Experiment: Tradeoff)





Han, Song, et al. "Learning both weights and connections for efficient neural network." NIPS. 2015.

Pruning with rehabilitation: Dynamic Network Surgery (Motivation)

Pruned connections have no chance to come back.

Incorrect pruning may cause severe accuracy loss.

Avoid the risk of irretrievable network damage .

Improve the learning efficiency.

Guo, Yiwen, et al. "Dynamic Network Surgery for Efficient DNNs." NIPS. 2016.

Pruning with rehabilitation: Dynamic Network Surgery (Formulation)

W_k denotes the weights, and T_k denotes the corresponding 0/1 masks.

$$\min_{W_k, T_k} L(W_k \odot T_k) \qquad s.t.\ \ {T_k}^{(i,j)} = h_k\left({W_k}^{(i,j)}\right), \forall (i,j) \in \mathfrak{T}$$

- $\odot$ is the element-wise product. $L(\cdot)$ is the loss function.

Dynamic network surgery updates only W_k. T_k is updated based on $h_k(\cdot)$.

$$h_k\left({W_k}^{(i,j)}\right) = \begin{cases} 0 & a_k \geq \left|{W_k}^{(i,j)}\right| \\ {T_k}^{(i,j)} & a_k \leq \left|{W_k}^{(i,j)}\right| \leq b_k \\ 1 & b_k \leq \left|{W_k}^{(i,j)}\right| \end{cases}$$

Not important

To be learnt

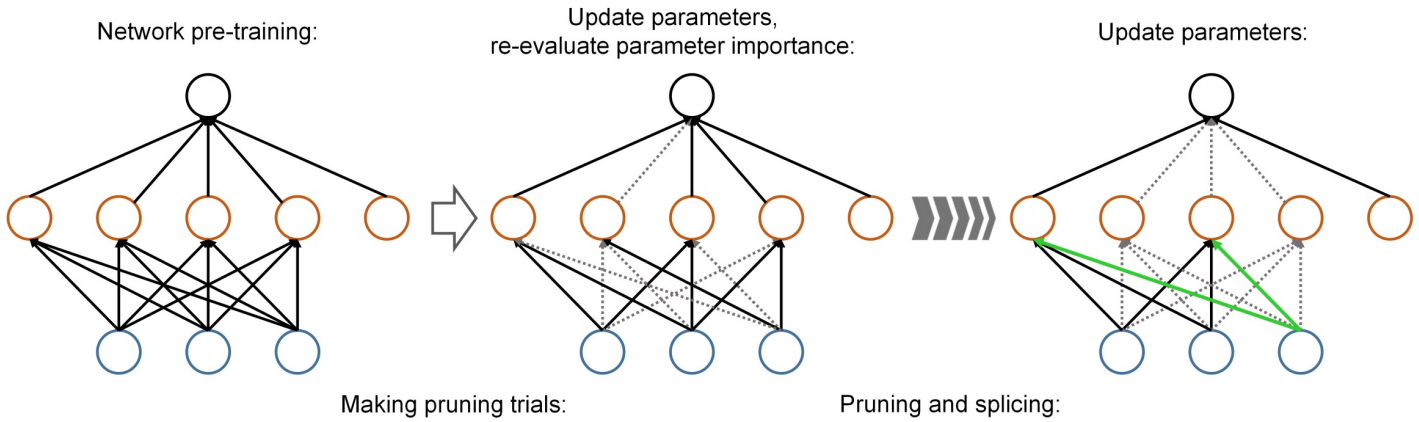
important

- a_k is the pruning threshold. $b_k = a_k + t$, where t is a pre-defined small margin.

Guo, Yiwen, et al. "Dynamic Network Surgery for Efficient DNNs." NIPS. 2016.

Pruning with rehabilitation: Dynamic Network Surgery (Algorithm)

1. Choose a neural network architecture.
2. Train the network until a reasonable solution is obtained.
3. Update T_k based on $h_k(\cdot)$.
4. Update W_k based on back-propagation.
5. Iterate to step 3.



Guo, Yiwen, et al. "Dynamic Network Surgery for Efficient DNNs." NIPS. 2016.

Outline

Weight Pruning

Quantization method

- Full Quantization
  - Fixed-point format
  - Code book
- Quantization with full-precision copy

Pruning + Quantization + Encoding

Hardware-aware Compression

Full Quantization : Fixed-point format

Limited Precision Arithmetic

- $[QI.QF]$, where QI and QF correspond to the integer and the fractional part of the number.
- The number of integer bits (IL) plus the number of fractional bits (FL) yields the total number of bits used to represent the number.
- WL = IL + FL.
- Can be represented as $\langle IL, FL \rangle$.
- $\langle IL, FL \rangle$ limits the precision to FL bits.
- $\langle IL, FL \rangle$ sets the range to $[-2^{IL-1}, 2^{IL-1} - 2^{-FL}]$.

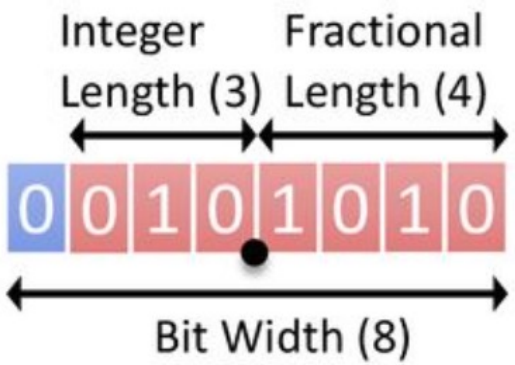
Gupta, Suyog, et al. "Deep Learning with Limited Numerical Precision." ICML. 2015.

Full Quantization

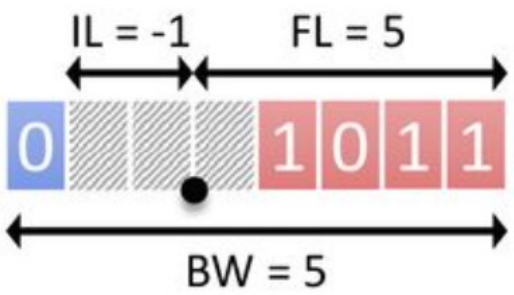
Fixed-point number format

The number of integer bits (IL) plus the number of fractional bits (FL) yields the total number of bits used to represent the number.

- Bit Width = Fractional (Word) Length + Integer (Word) Length + 1 (Sign)
- Approximate real value $\boldsymbol{x}$ by computing $\hat{\boldsymbol{x}} = \mathrm{Round}\left(\boldsymbol{x} \cdot 2^{\mathrm{FL}}\right) \cdot 2^{-\mathrm{FL}}$
- Representable range: $-2^{\mathrm{IL}} \leq \hat{\boldsymbol{x}} \leq 2^{\mathrm{IL}} - 2^{-\mathrm{FL}}$ with precision/step size $2^{-\mathrm{FL}}$



$= 42 \cdot 2^{-4} = 2.625$



$= 11 \cdot 2^{-5} = 0.34375$

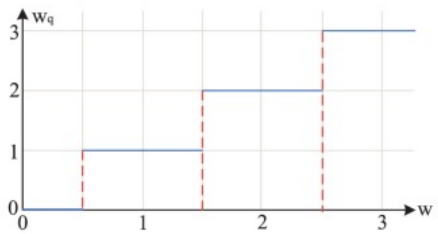
High precision value 2.62511234567, multiplied by 2^4 then we get 42.0017975307. Then around (), get 42. We can convert 42 into 0101010 this binary array.

Full Quantization : Fixed-point format (Rounding Modes)

Define $\lfloor x \rfloor$ as the largest integer multiple of $\epsilon = 2^{-FL}$.

Round-to-nearest:

- $Round(x, \langle IL, FL \rangle) = \begin{cases} \lfloor x \rfloor & \lfloor x \rfloor \leq x \leq \lfloor x \rfloor + \frac{\epsilon}{2} \\ \lfloor x \rfloor + \epsilon & \lfloor x \rfloor + \frac{\epsilon}{2} \leq x \leq \lfloor x \rfloor + \epsilon \end{cases}$



(a) The traditional rounding to the nearest mode, using the mean of adjacent quantization points as the critical value.

Stochastic rounding (unbiased):

- $Round(x, \langle IL, FL \rangle) = \begin{cases} \lfloor x \rfloor & w.p. \quad 1 - \frac{x - \lfloor x \rfloor}{\epsilon} \\ \lfloor x \rfloor + \epsilon & w.p. \quad \frac{x - \lfloor x \rfloor}{\epsilon} \end{cases}$

If x lies outside the range of $\langle IL, FL \rangle$, we saturate the result to either the lower or the upper limit of $\langle IL, FL \rangle$:

Gupta, Suyog, et al. "Deep Learning with Limited Numerical Precision." ICML. 2015.

Multiply and accumulate (MACC) operation

During training:

1. $\boldsymbol{a}$ and $\boldsymbol{b}$ are two vectors with fixed point format $\langle IL, FL \rangle$.

2. Compute $z = \sum_{i=1}^{d} a_i b_i$.
   - Results a fixed point number with format $\langle 2\times IL, 2\times FL \rangle$.

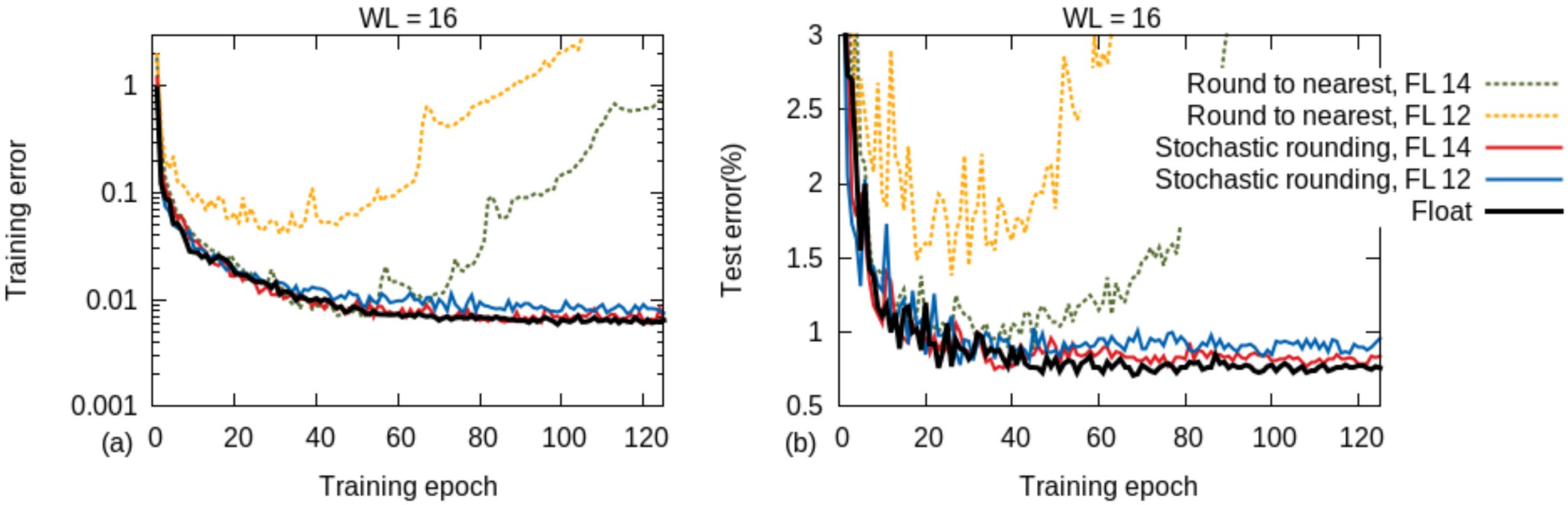
3. Covert and round z back to fixed point format $\langle IL, FL \rangle$. (Discard highest IL from left + FL from right hand side due to high possibility of being zero.)

During testing:

With fixed point format $\langle IL, FL \rangle$.

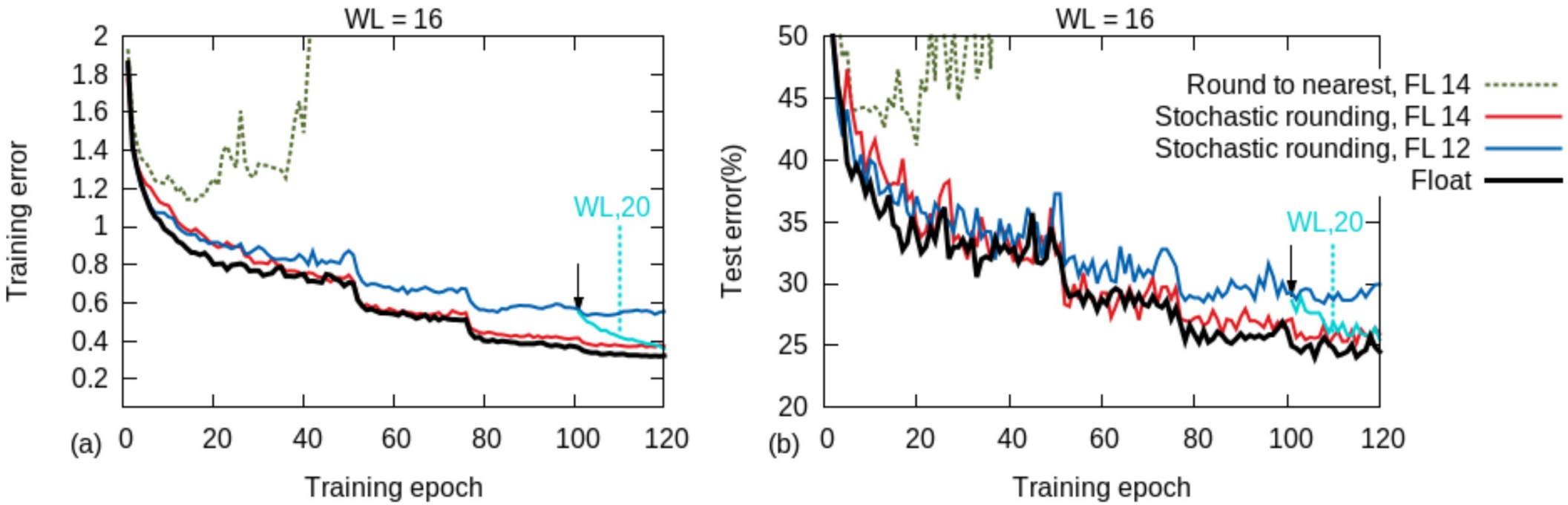
Gupta, Suyog, et al. "Deep Learning with Limited Numerical Precision." ICML. 2015.

Full Quantization: Fixed-point format (Experiment on MNIST with CNNs)



Gupta, Suyog, et al. "Deep Learning with Limited Numerical Precision." ICML. 2015.

Full Quantization: Fixed-point format (Experiment on CIFAR10 with fully connected DNNs)



Gupta, Suyog, et al. "Deep Learning with Limited Numerical Precision." ICML. 2015.

Outline

Weight Pruning

Quantization method

- Full quantization
- Quantization with full-precision copy
  - Binnaryconnect
  - BNN

Pruning + Quantization + Encoding

Hardware-aware Compression

Quantization with full-precision copy: Binaryconnect (Motivation)

Use only two possible value (e.g. +1 or -1) for weights.

Replace many multiply-accumulate operations by simple accumulations.

Fixed-point adders are much less expensive both in terms of area and energy than fixed-point multiply-accumulators.

Courbariaux, et al. "Binaryconnect: Training deep neural networks with binary weights during propagations." NIPS. 2015

Quantization with full-precision copy: Binaryconnect (Binarization)

Deterministic Binarization:

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

Stochastic Binarization:

- $w_b = \begin{cases} +1 & with\ probability\ p = \sigma(w_b) \\ -1 & with\ probability\ 1 - p \end{cases}$
- $\sigma(x) = clip\left(\frac{x+1}{2}, 0, 1\right) = max\left(0, min\left(1, \frac{x+1}{2}\right)\right)$

Stochastic binarization is more theoretically appealing than the deterministic one, but harder to implement as it requires the hardware to generate random bits when quantizing.

Courbariaux, et al. "Binaryconnect: Training deep neural networks with binary weights during propagations." NIPS. 2015

Quantization with full-precision copy: Binaryconnect

BinaryConnect only **binarize** the weights during the **forward** and **backward** propagations (steps 1 and 2) but **not** during the **parameter update** (step 3).

Courbariaux, et al. "Binaryconnect: Training deep neural networks with binary weights during propagations." NIPS. 2015

Quantization with full-precision copy: Binaryconnect

1. Binarize weights and perform forward pass.
2. Back propagate gradient based on binarized weights.
3. Update the full-precision weights.
4. Iterate to step 1.

Courbariaux, et al. "Binaryconnect: Training deep neural networks with binary weights during propagations." NIPS. 2015

Quantization with full-precision copy: Binaryconnect

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%**

Courbariaux, et al. "Binaryconnect: Training deep neural networks with binary weights during propagations." NIPS. 2015

Quantization with full-precision copy: Binarized Neural Networks (Motivation)

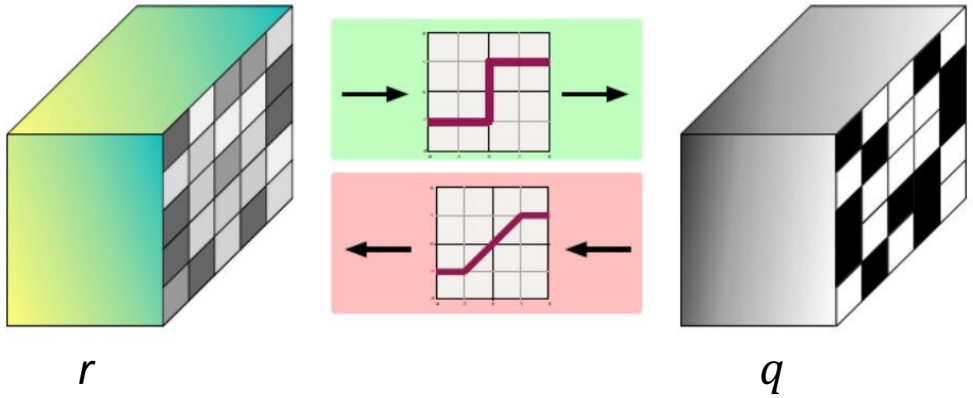
Neural networks with **both binary weights and activations** at run-time and when computing the parameters' gradient at train time.

Courbariaux, Matthieu, et al. "Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1." arXiv preprint arXiv:1602.02830 (2016).

Quantization with full-precision copy: Binarized Neural Networks

Propagating Gradients Through Discretization (“straight-through estimator ”)

- $q = Sign(r)$
- Estimator g_q of the gradient $\frac{\partial C}{\partial q}$
- Straight-through estimator of $\frac{\partial C}{\partial r}$
  - $g_r = g_q 1_{|r| \leq 1}$
  - Can be viewed as propagating the gradient through *hard tanh*
  - Where $\frac{\partial C}{\partial r} = \frac{\partial C}{\partial q} \frac{\partial q}{\partial r}$

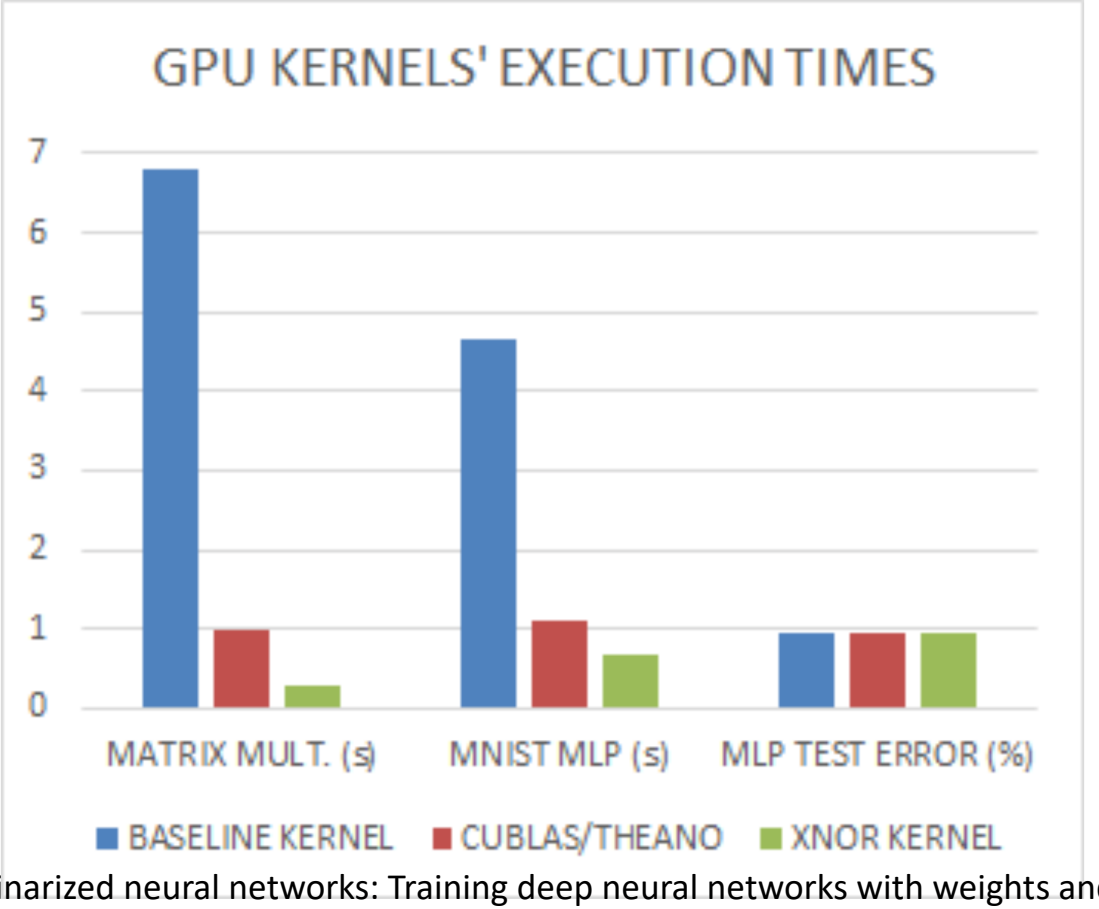


Replace multiplications with bit-shift

- Replace batch normalization with shift-based batch normalization
- Replace ADAM with shift-based AdaMax

- Courbariaux, Matthieu, et al. "Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1." arXiv preprint arXiv:1602.02830 (2016).
- Tian Huang, Tao Luo, Joey Tianyi Zhou, “RCT: Resource Constrained Training for Edge AI”, In TNNLS 2022.

Quantization with full-precision copy: Binarized Neural Networks



Courbariaux, Matthieu, et al. "Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1." arXiv preprint arXiv:1602.02830 (2016).

Outline

Matrix Factorization

Weight Pruning

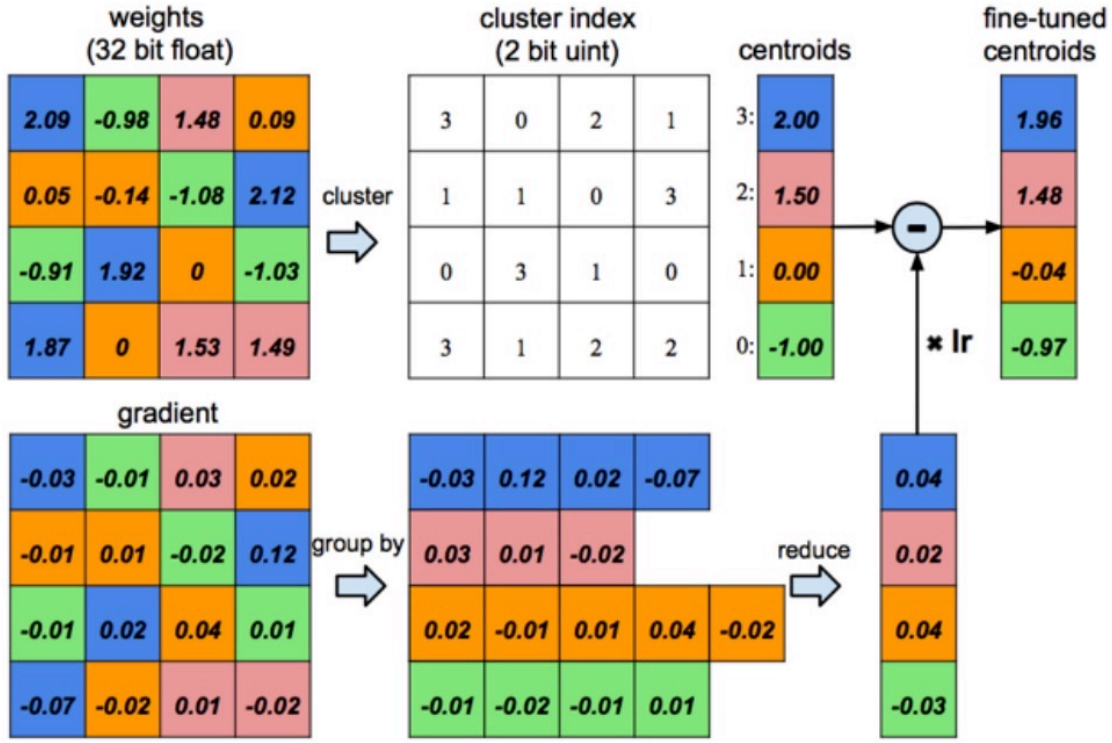
Quantization method

Pruning + Quantization + Encoding

- Deep Compression

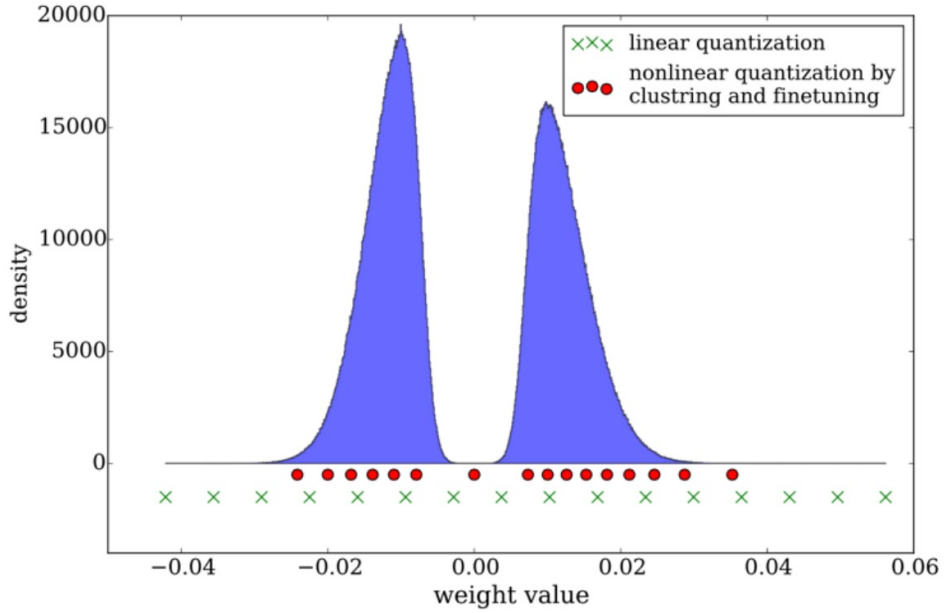
Hardware-aware Compression

Pruning + Quantization + Encoding: Deep Compression



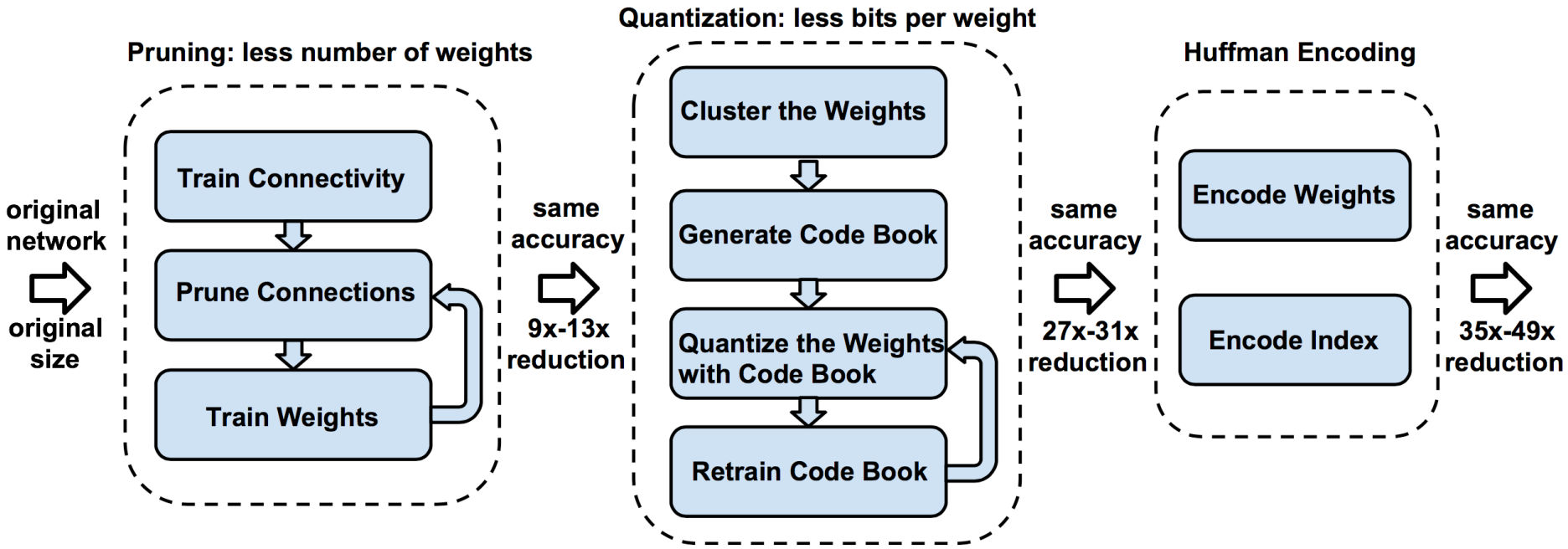
Weight sharing by scalar quantization (top) and centroids fine-tuning (bottom)

Finetune Centroids



Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding

Pruning + Quantization + Encoding: Deep Compression



Huffman code is a type of optimal prefix code that is commonly used for loss-less data compression. It produces a variable-length code table for encoding source symbol. As in other entropy encoding methods, more common symbols are represented with fewer bits than less common symbols, thus save the total space

Han, etal., Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding, ICLR 2016

Pruning + Quantization + Encoding: Deep Compression

1. Choose a neural network architecture.
2. Train the network until a reasonable solution is obtained.
3. Prune the network with magnitude-based method until a reasonable solution is obtained.
4. Quantize the network with k-means based method until a reasonable solution is obtained.
5. Further compress the network with Huffman coding.

Courbariaux, Matthieu, et al. "Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1." arXiv preprint arXiv:1602.02830 (2016).

Pruning + Quantization + Encoding: Deep Compression

Table 1: The compression pipeline can save 35× to 49× parameter storage with no loss of accuracy.

Network	Top-1 Error	Top-5 Error	Parameters	Compress Rate
LeNet-300-100 Ref	1.64%	-	1070 KB	
LeNet-300-100 Compressed	1.58%	-	**27 KB**	**40×**
LeNet-5 Ref	0.80%	-	1720 KB	
LeNet-5 Compressed	0.74%	-	**44 KB**	**39×**
AlexNet Ref	42.78%	19.73%	240 MB	
AlexNet Compressed	42.78%	19.70%	**6.9 MB**	**35×**
VGG-16 Ref	31.50%	11.32%	552 MB	
VGG-16 Compressed	31.17%	10.91%	**11.3 MB**	**49×**

Table 2: Compression statistics for LeNet-300-100. P: pruning, Q:quantization, H:Huffman coding.

Layer	#Weights	Weights% (P)	Weight bits (P+Q)	Weight bits (P+Q+H)	Index bits (P+Q)	Index bits (P+Q+H)	Compress rate (P+Q)	Compress rate (P+Q+H)
ip1	235K	8%	6	4.4	5	3.7	3.1%	2.32%
ip2	30K	9%	6	4.4	5	4.3	3.8%	3.04%
ip3	1K	26%	6	4.3	5	3.2	15.7%	12.70%
Total	266K	8%(12×)	6	5.1	5	3.7	3.1% (**32×**)	2.49% (**40×**)

Table 3: Compression statistics for LeNet-5. P: pruning, Q:quantization, H:Huffman coding.

Layer	#Weights	Weights% (P)	Weight bits (P+Q)	Weight bits (P+Q+H)	Index bits (P+Q)	Index bits (P+Q+H)	Compress rate (P+Q)	Compress rate (P+Q+H)
conv1	0.5K	66%	8	7.2	5	1.5	78.5%	67.45%
conv2	25K	12%	8	7.2	5	3.9	6.0%	5.28%
ip1	400K	8%	5	4.5	5	4.5	2.7%	2.45%
ip2	5K	19%	5	5.2	5	3.7	6.9%	6.13%
Total	431K	8%(12×)	5.3	4.1	5	4.4	3.05% (**33×**)	2.55% (**39×**)

Short Break 15 mins

Outline

Matrix Factorization

Weight Pruning

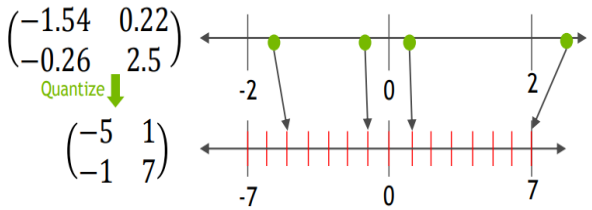
Quantization method

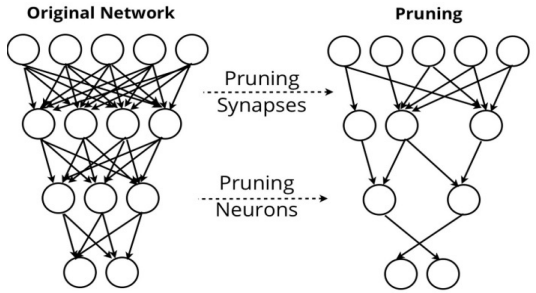
Pruning + Quantization + Encoding

Hardware-aware Compression

- Energy-aware Compression with Data-flows

Model Compression





Quantization

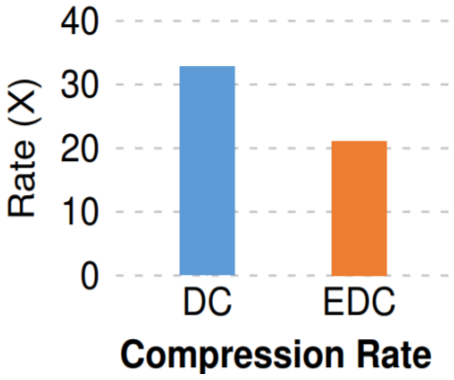
- Lower the precision
- Floats (FP32) scaled to integers (INT8)

Pruning

- Drop small value weights
- Fine tune and recover

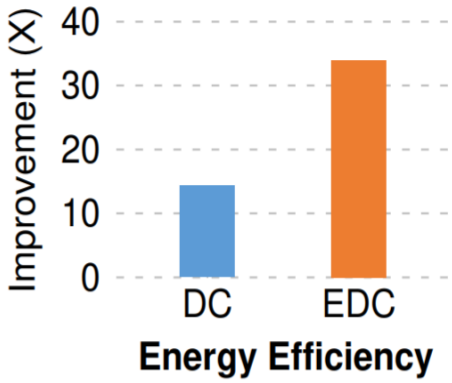
Model Size e.g. 8 bit * 1000 weights = 8k models size

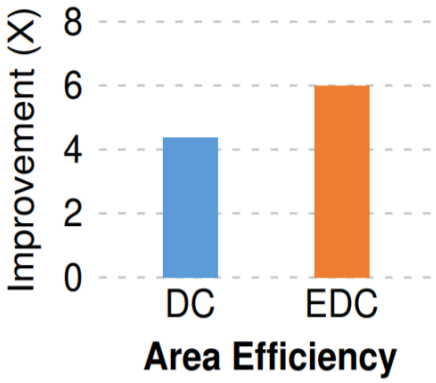
Model Size Vs. Energy/Area



DC
Deep Compression
(Han et al., 2015)

EDC
EDCompress





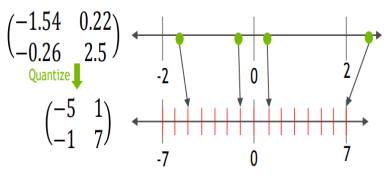
e.g. 8 bit * 1000 weights = 8k models size

Traditional model compression
- Target on the storage of weights
- Fit on edge devices
- Deep Compression- 33X on LeNet05 (Quantization + Pruning)

- Energy-aware model compression with dataflow (EDCompress)
  - Target on energy/area
  - Related to dataflow
  - Diversity of dataflow types

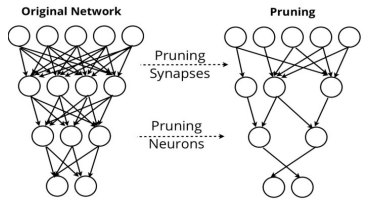
Energy-aware Model Compression

Traditional model compression focuses on **model size.**



1. Quantization

- Lower the precision ($FP32 >> INT8$)

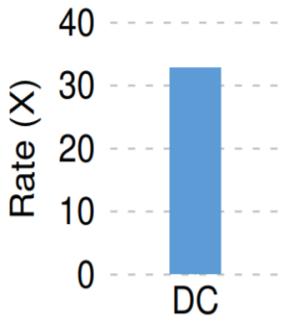


2. Pruning

- Drop small weights

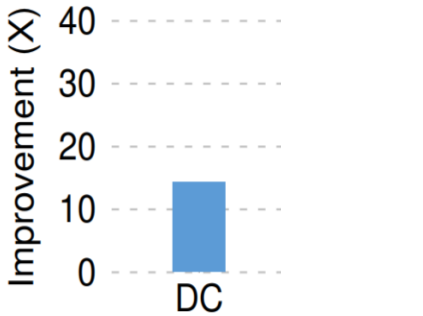
Smaller Size ≠ Lower Energy

Why?

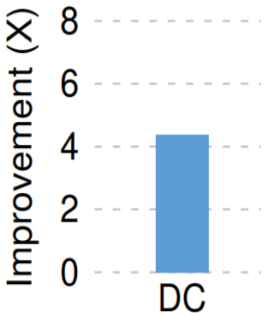


Compression Rate

DC (Deep Compression) (*Han et al., ICLR-16, Best Paper*)



Energy Efficiency



Area Efficiency

compressed 33× model size, but only saved 10× energy. Battery?

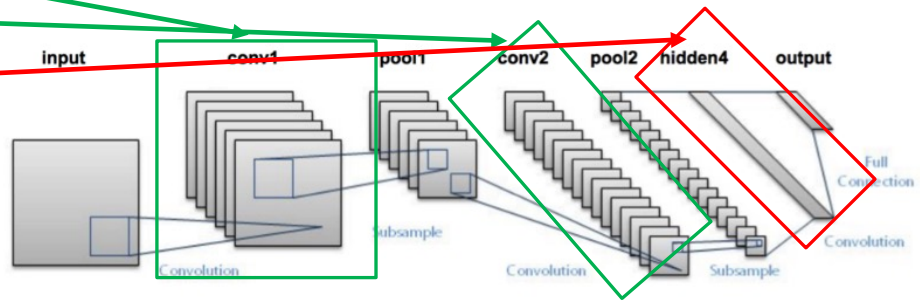
Confidential

Energy breakdown

A close look on model size and energy breakdown.

	Model Size	Energy
C1	0.1 %	28 %
C2	0.5 %	56 %
L1	93 %	13 %
L2	6.4 %	3 %

- Target on model size (Traditional methods)
  - Fully Connection
- Target on energy
  Conv Layers



A closer look

We experimentally observe that a large portion of the energy is consumed on the **data movement** (e.g., around 72% in VGG-16) in computing the **convolution** layers.

Data movement ⟵ **Dataflow design**

What is Dataflow

Algorithm 1 Computation of a typical convolutional layer

```
for c_o in range (C_O) do
  for c_i in range (C_I) do
    for x in range (X) do
      for y in range (Y) do
        for f_x from -(F_X-1)/2 to (F_X-1)/2 do
          for f_y from -(F_Y-1)/2 to (F_Y-1)/2 do
            O[c_o][x][y]+=
            I[c_i][x+f_x][y+f_y]×W[c_o][c_i][f_x][f_y]
```

Table 1. Notation table of Algorithm 1

C_O	# of Output channel	C_I	# of Input channel
X	Width of feature map	Y	Height of feature map
F_X	Width of filter	F_Y	Height of filter

Dimensions in activation/weights

MAC (multiply–accumulate)

- $C_O \cdot C_I \cdot X \cdot Y \cdot F_X \cdot F_Y$ iterations
- Speedup by multiple processing elements (PEs)

Given a matrix of PEs

- Mapping
- Unloop any two loops
- $C_6^2 = 15$ combinations

Each combination-> dataflow

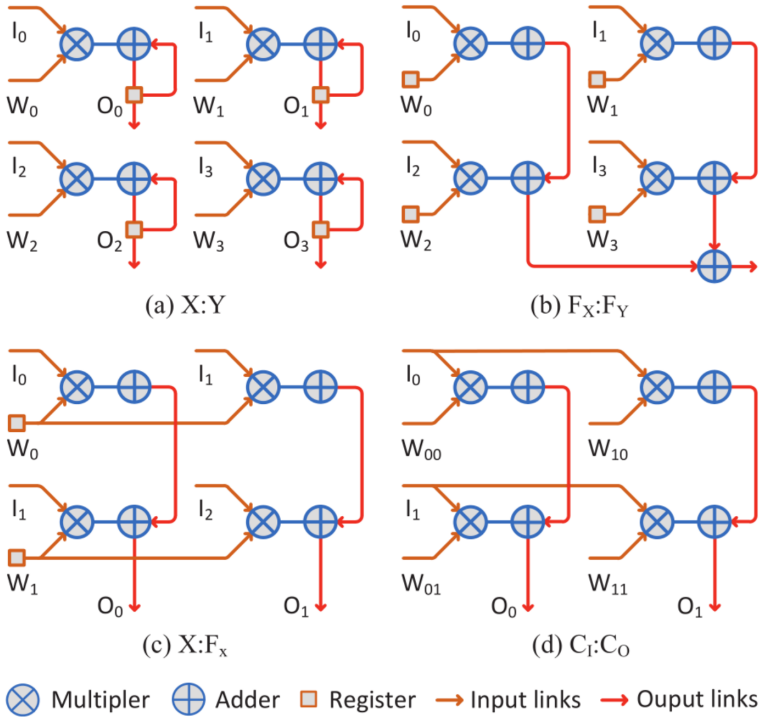
Significance of Dataflow

Table 2. List of four popular dataflow types. Reference work are [1-2] (Du et al., 2015) (Song et al., 2018), [3] (Qiu et al., 2016), [4-6] (Chen et al., 2016) (Gao et al., 2017) (Li et al., 2016a), [7-12] (Chen et al., 2014) (Jouppi et al., 2017) (Zhang et al., 2015) (Alwani et al., 2016) (Shen et al., 2016) (Suda et al., 2016)

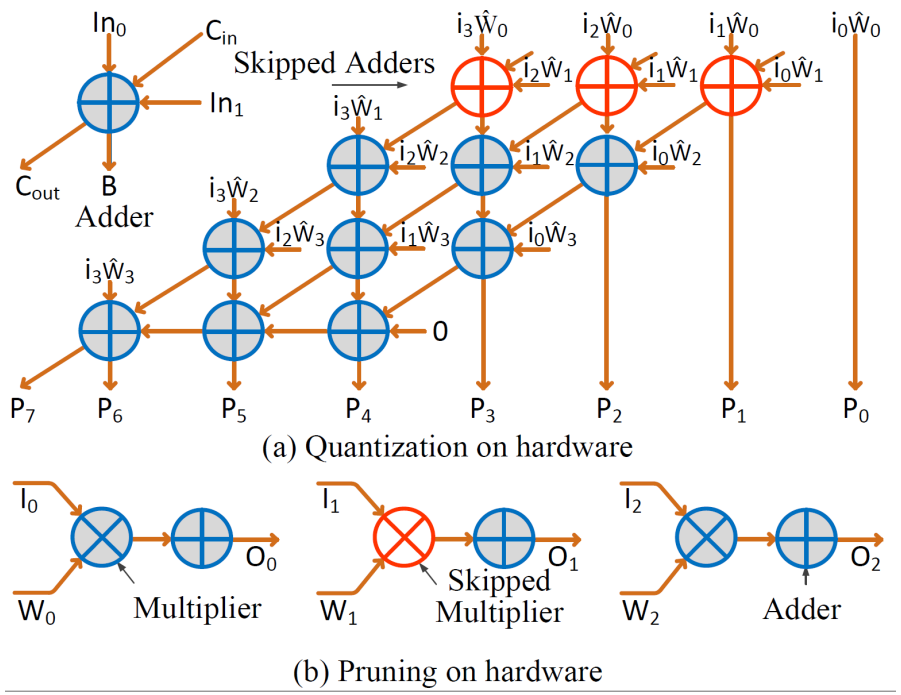
Dataflow	Applied by	Dataflow	Applied by
$X : Y$	[1-2]	$F_X : F_Y$	[3]
$X : F_X$	[4-6]	$C_I : C_O$	[7-12]

- MAC: 3 inputs + 1 output
- Data reuse
  - Register: temporarily store input/output
  - Reduce data movement

Different dataflow → Different compression



Effects of Compression Methods on Dataflow



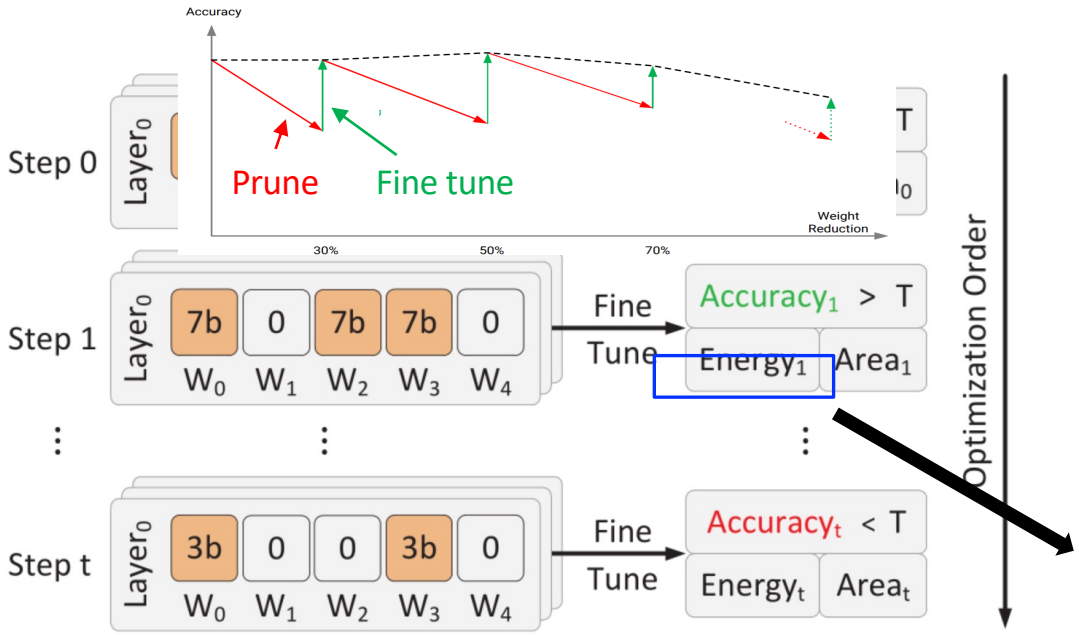
Dataflow with different data movement

Quantization
- Weights : n bit -> n-Δn bit
- Skip Δn rows of adders
- Reduce $\Delta n/n$ data movement

Pruning
- Skip multiplier if weight = 0
- No need to transmit that weight

Demand a way to combine and apply

Multi-Step Optimization

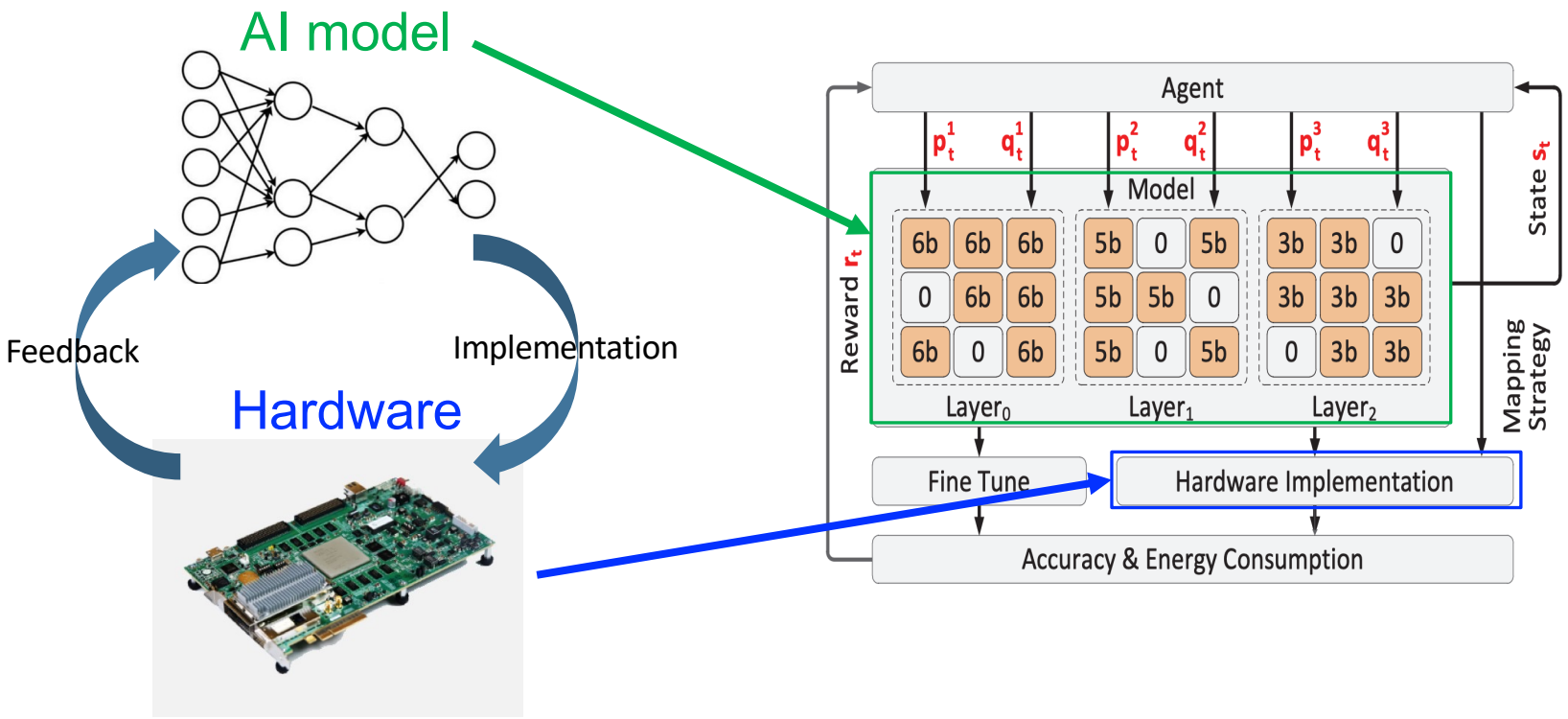


EDCompress

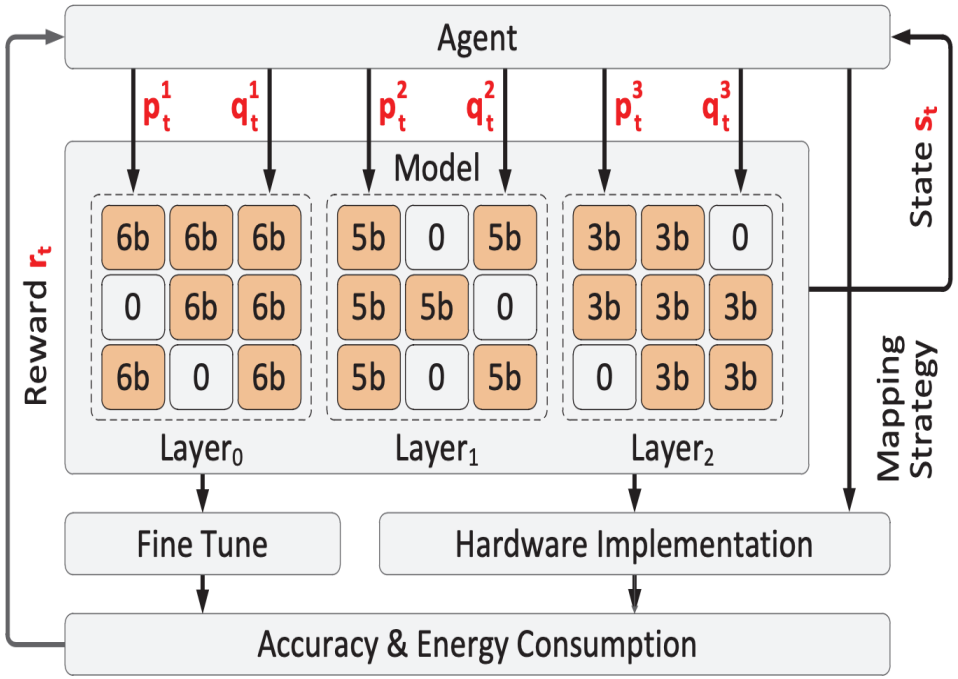
- Two degrees of freedom: quantization & pruning
- Prune: 100%->0%
- Quantization: 8 bit -> 1 bit
- Fine-tune :re-train a few epochs
- Like playing the game (Reinforcement Learning)

How to get the energy consumption as supervision?

Reinforcement Learning



Reinforcement Learning



Multi-step problem/Game

- discount factor in each step

$$Q_t^l = Q_0^l + \sum_{i=0}^{t-1} q_i^l \gamma^i$$

$$P_t^l = P_0^l + \sum_{i=0}^{t-1} p_i^l \gamma^i$$

Actor-Critic

- Action
  - Δpruning amount +/-
  - Δquantization bit +/-

$$a_t = (\bigcup_{l=0}^{L-1} \{q_t^l\}) \cup (\bigcup_{l=0}^{L-1} \{p_t^l\})$$

- State
  - History of actions
  - History of rewards

$$s_t = (\bigcup_{t-\tau}^{m=t} \bigcup_{l=0}^{L-1} \{Q_m^l\}) \cup (\bigcup_{t-\tau}^{m=t} \bigcup_{l=0}^{L-1} \{P_m^l\}) \cup (\bigcup_{t-\tau}^{m=t} \{r_m\}) \cup \{t\}$$

- Reward
  - Accuracy Improvement Energy Improvement

$$r_t = (\alpha_t/\alpha_{t-1})^\lambda \cdot \beta_{t-1}/\beta_t$$

Experiment Setup

Algorithm setup:

Soft Actor Critic (SAC)

Models
- Complex: VGG-16
- Simple: LeNet-5
- Computation efficiency: MobileNet

Data Set
- MNIST
- CIFAR10
- ImageNet

Hardware setup:

Xilinx Virtex UltraScale FPGA

Xilinx XPE toolkit

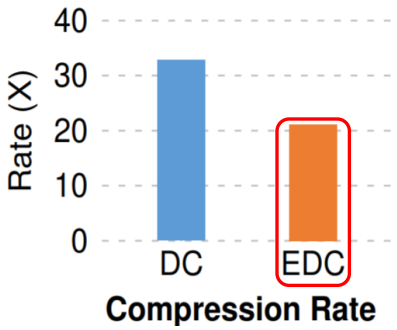
Logic
- Registers
- M × N Multiplier (Nbit weight, Mbit feature)
- M/2 × (N + 1) LUTs*

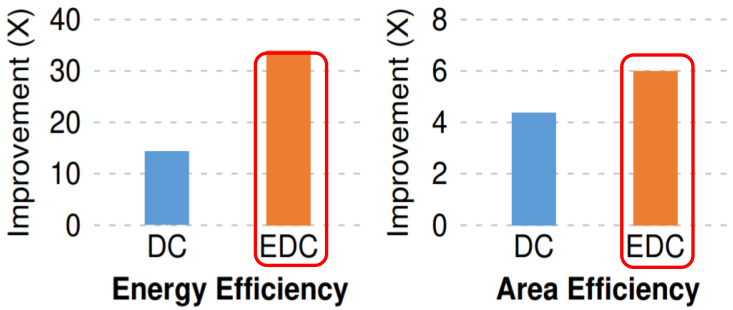
Memory
- All weights + the maximum activation

*Ref: E. G. Walters, "Array Multipliers for High Throughput in Xilinx FPGAs with 6-input LUTs," Computers, vol. 5, no. 4, p. 20, 2016.

Experimental Results

Smaller Size ≠ Lower Energy





We compress less.

We save much more energy & area.

DC-Deep Compression (*Han et al., ICLR-16, Best Paper*)
EDC- Energy-aware Model Compression with Dataflow (Ours)

Comparison with the State-of-the-Arts

EDCompress Vs. HAQ [29] on ImageNet using MobileNet

Dataflow	Norm. Energy [29]	Norm. Energy Ours	Norm. Area [29]	Norm. Area Ours
$X : Y$	5.44	**1.41**	26.1	**5.27**
$F_X : F_Y$	6.31	**1.81**	2.53	**1.00**
$X : F_X$	6.32	**1.81**	2.53	**1.00**
$C_I : C_O$	4.48	**1.00**	505	**92.0**
Top-1 Accuracy	64.8	68.3	64.8	68.3
Top-5 Accuracy	85.9	88.3	85.9	88.3

[29] K. Wang et al., "HAQ: Hardware-Aware Automated Quantization with Mixed Precision," in CVPR, 2019, pp. 8612–8620.

ImageNet + MobileNet: **3.8X/3.9X** improvement on energy/area efficiency

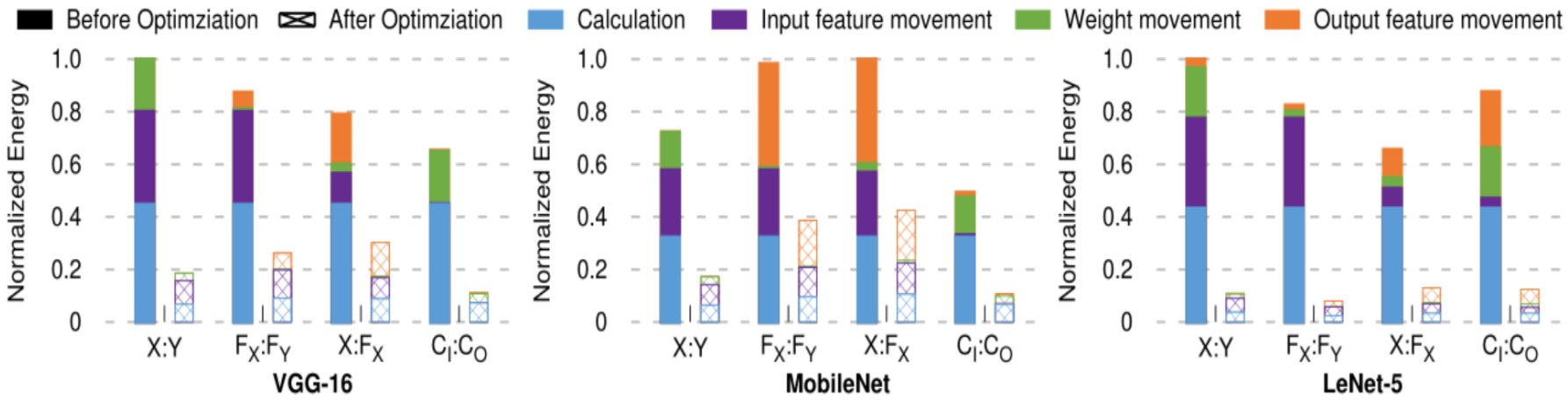
EDCompress Vs. previous work on CIFAR-10 using VGG-16

Dataflow	Norm. Energy [16]	Norm. Energy [17]	Norm. Energy Ours	Norm. Area [16]	Norm. Area [17]	Norm. Area Ours
$X : Y$	24.41	15.10	**1.69**	7.78	5.56	**1.00**
$F_X : F_Y$	22.61	14.42	**2.31**	6.42	4.20	**1.27**
$X : F_X$	22.17	15.10	**2.73**	6.42	4.20	**1.42**
$C_I : C_O$	19.68	12.21	**1.00**	434	431	**47.58**
Accuracy	93.1	93.4	91.3	93.1	93.4	91.3

[16] Li, Hao, et al. "Pruning Filters for Efficient ConvNets." ICLR 2017.
[17] Singh, Pravendra, et al. "Play and Prune: Adaptive Filter Pruning for Deep Model Compression." IJCAI 2019

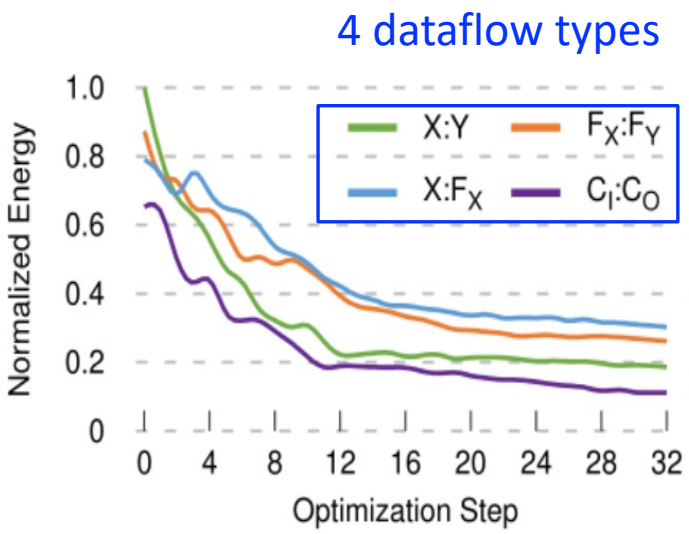
CIFAR-10 + VGG-16:
11X/6X improvement on energy/area efficiency

Before and After EDC



Up to **37X**, energy efficiency improvement

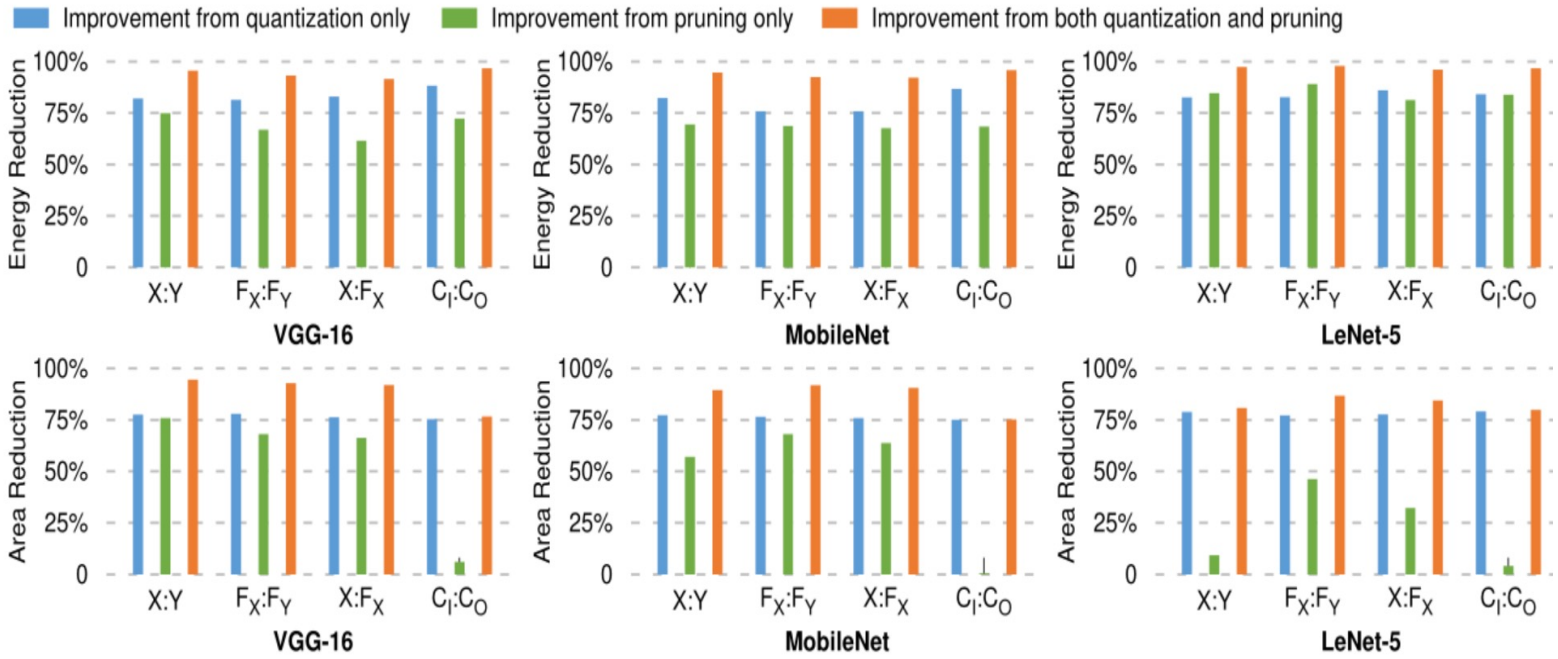
Dataflow vs Strategy



For different dataflow in hardware:

- Existing methods always have the **same** optimization strategy
- Ours: Optimization changes with **different** dataflow in hardware.

Insights on Quantization & Pruning



- Quant+ prune is the best!
- Quantization is **more efficient** than pruning.

Zhehui Wang, Tao Luo, Rick Goh, Joey Tianyi Zhou, **"EDCompress: Energy-Aware Model Compression with Dataflow"**, ***TNNLS***.

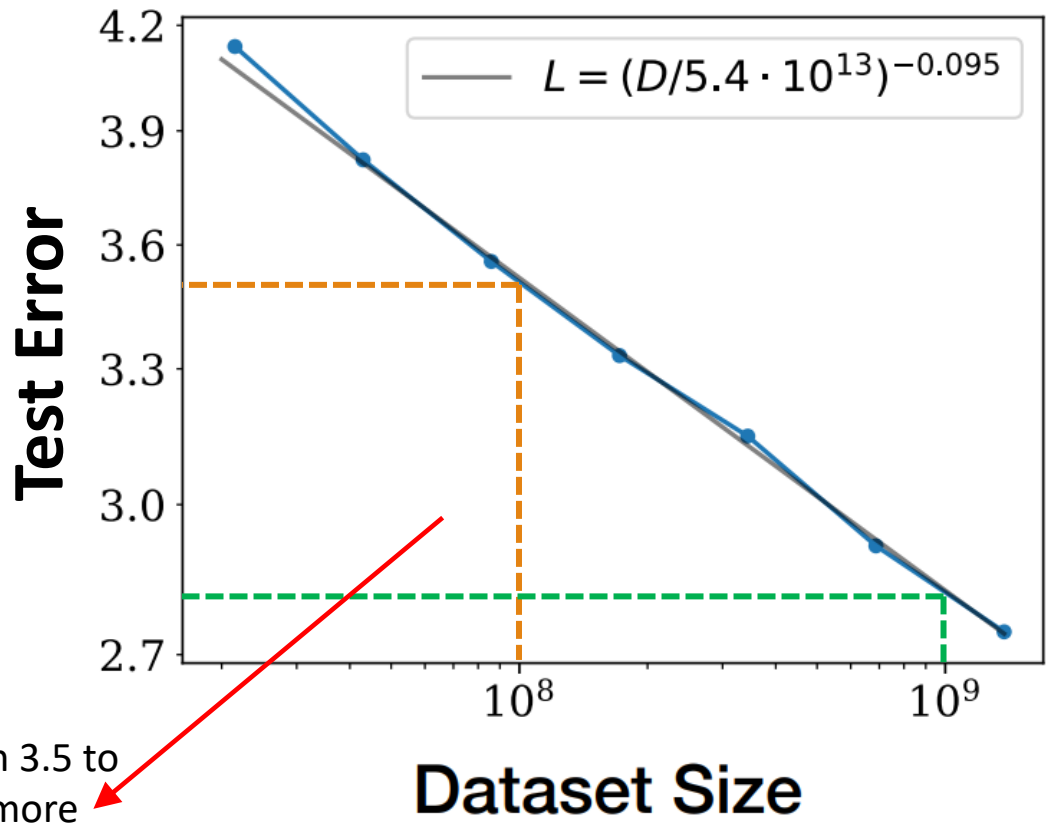
Dataset Compression

--- Pruning & Distillation

Motivation
Why smaller datasets?

- **Power Law Scaling between performance and dataset scale**

  Marginal improvement in performance requires an order of magnitude more data.



Drop of test error from 3.5 to 2.8 requires 10 times more training data.

Motivation
Why smaller datasets?

- **Limited computation and storage resources**

  For example, ImageNet-1k comprises 1,281,167 images across 1000 classes, consuming 130 GB of memory.

Networks	Processor	Time	Accuracy
Resnet-50	Tesla P100 x8	29 hours	75.3%
DeiT-S	V100 x4	3 days	81.2%

Outline

Dataset Pruning

Select a subset images in a full dataset without performance drop.

Dataset Distillation

Learn a few synthetic images (alter image pixels) to replace full dataset.

Outline

Dataset Pruning

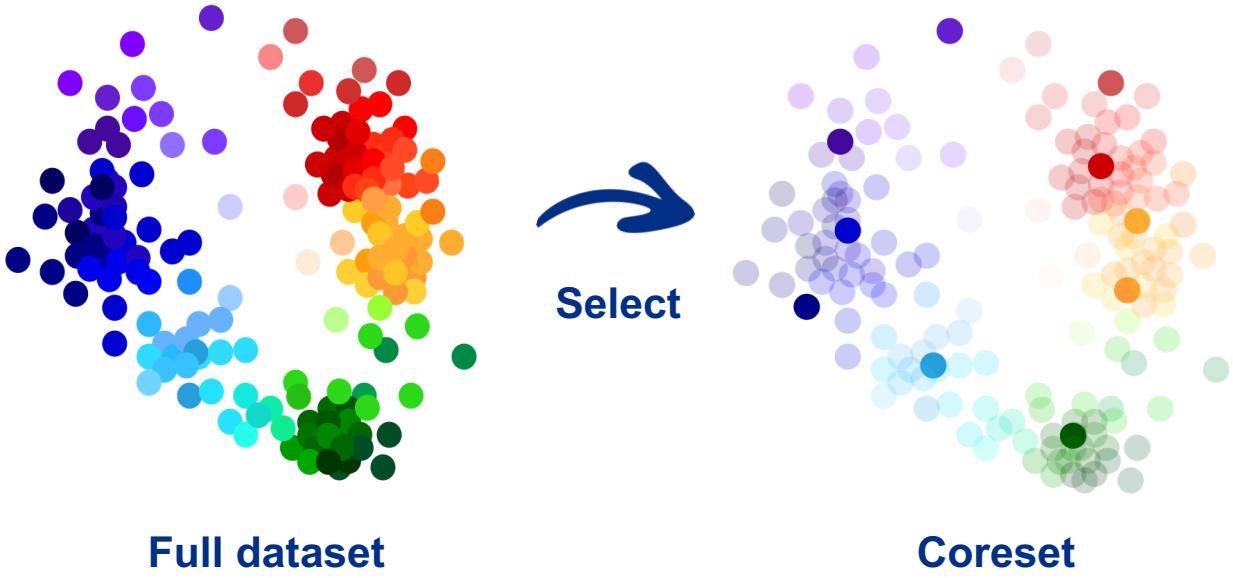
Select a subset images in a full dataset without performance drop.

- A Milestone Paper: Forgetting
- An Improvement Paper: Selection via Proxy
- An Improvement Paper: EL2N

Dataset Distillation

Learn a few synthetic images (alter image pixels) to replace full dataset.

Dataset Pruning



Also known as **Subset Selection**.

The Objective:

- **Select a subset images** in a full dataset without performance drop.

Dataset Pruning

A Milestone Paper: Forgetting (ICLR, 2019)

- Some samples are consistently forgotten across subsequent training;
- Some examples are never forgotten.

Prune unforgettable ones

Forgetting statistics:

Sample i undergoes a forgetting event when it is correctly classified in the current update, but misclassified in the next update.

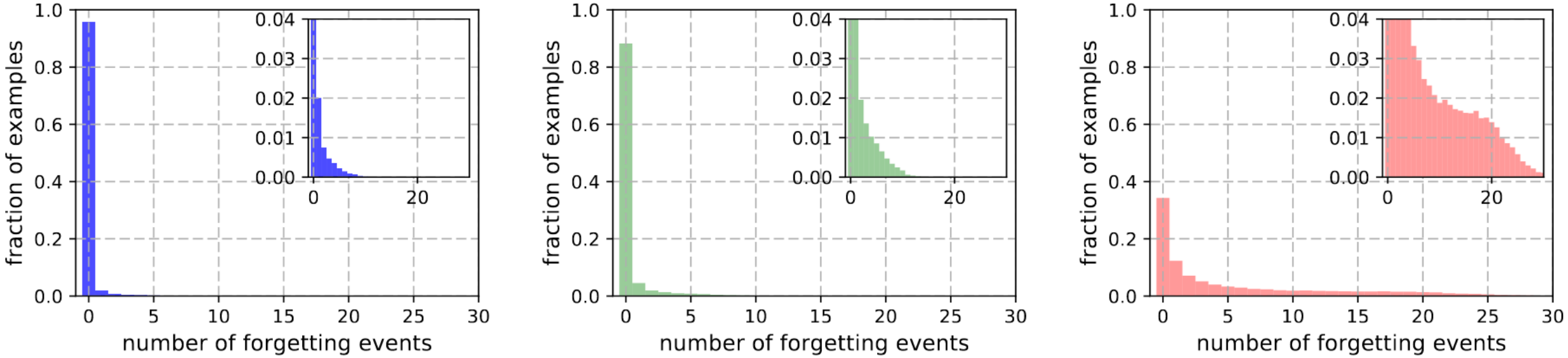
Algorithm 1 Computing forgetting statistics.

initialize $\text{prev_acc}_i = 0, i \in \mathcal{D}$
initialize forgetting $T[i] = 0, i \in \mathcal{D}$
while not training done **do**
  $B \sim \mathcal{D}$ # sample a minibatch
  for example $i \in B$ **do**
    compute acc_i
    if $\text{prev_acc}_i > \text{acc}_i$ **then**
      $T[i] = T[i] + 1$
    $\text{prev_acc}_i = \text{acc}_i$
  gradient update classifier on B
return T

T[i]: the number of forgetting events for sample i.

Dataset Pruning

A Milestone Paper: Forgetting (ICLR, 2019)



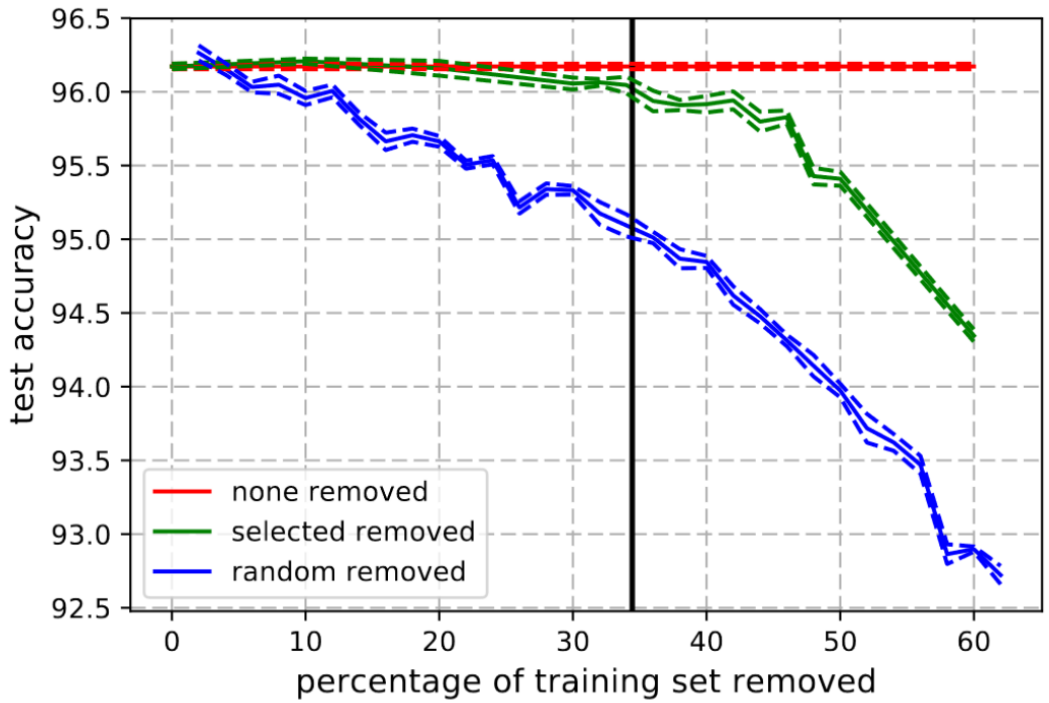
Histograms of forgetting events on (from left to right) MNIST, permutedMNIST and CIFAR-10

- **Most samples are unforgettable.**
- **More complex datasets contain significantly fewer unforgettable samples.**

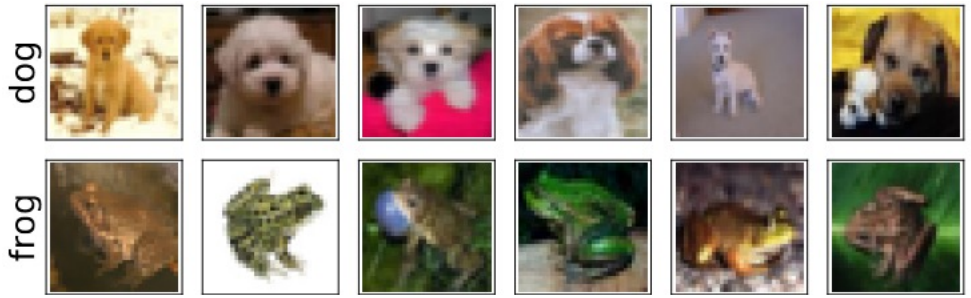
Dataset Pruning

A Milestone Paper: Forgetting (ICLR, 2019)

Performance on CIFAR-10 of ResNet18



The most unforgettable samples (CIFAR-10)



The most forgettable samples (CIFAR-10)

Forgettable samples seem to exhibit peculiar or uncommon features.

Example Forgetting during Deep Neural Network Learning, ICLR 2019.

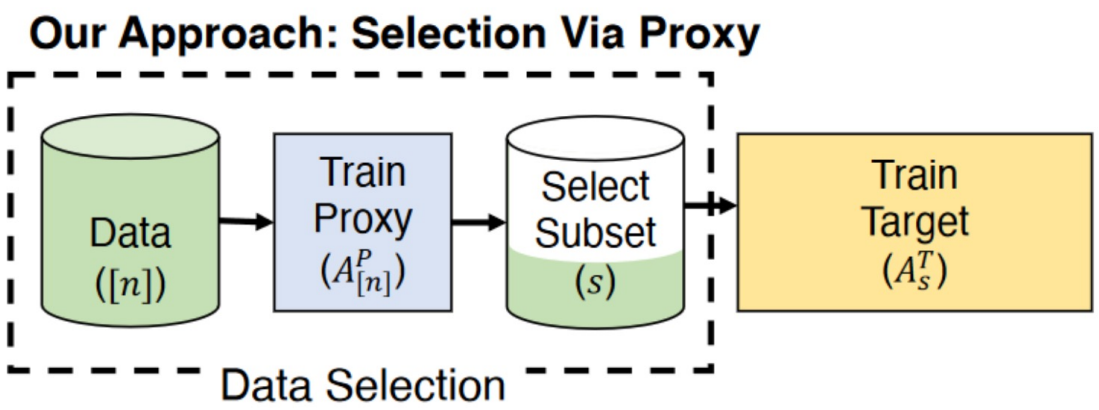
Dataset Pruning

Problem/Challenge of "Forgetting":

The collection of "Forgetting" statistics for large models is time-consuming.

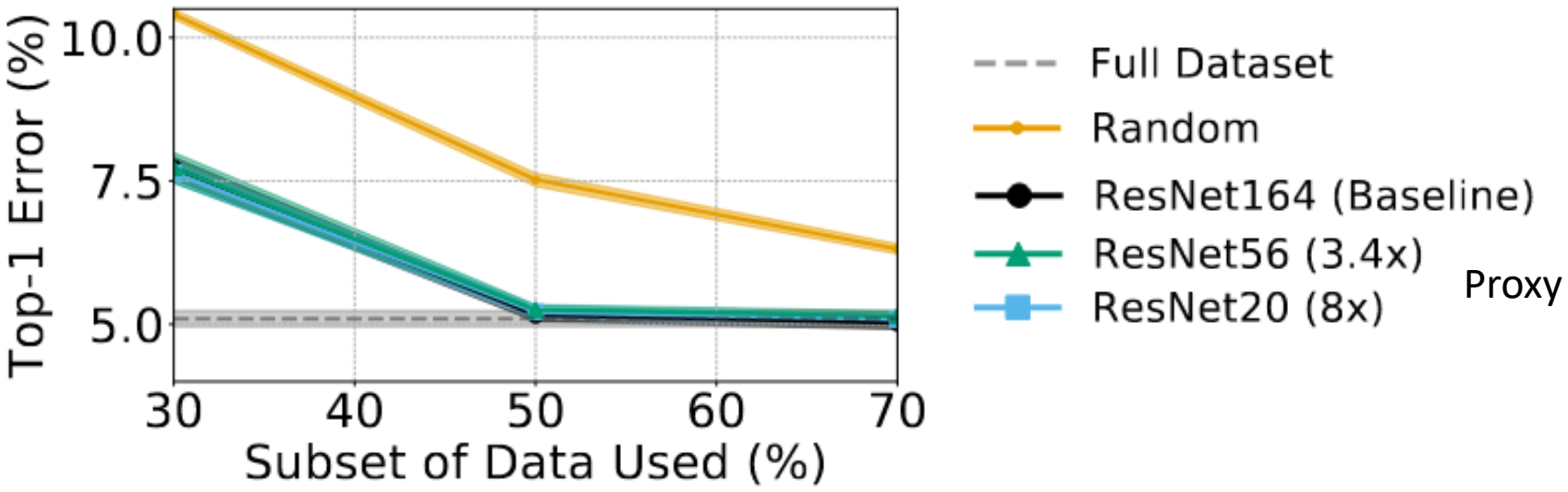
A Solution: Selection via Proxy (ICLR, 2020)

- Use a **smaller** proxy network $A^P_{[n]}$ with **fewer** epochs to speed up the training process.



Dataset Pruning

"Selection via Proxy" Performance



CIFAR10 forgetting events

Performance is on-par a large network.

Dataset Pruning

Problem/Challenge of "Forgetting":

Collecting "Forgetting" statistics necessitates a complete training.

A Solution: EL2N (NeurIPS, 2021)

EL2N (New Metric) : The **early (less than 20 epochs) error vector score**, averaged over several weight initializations.

$$\mathbb{E}\|p(\mathbf{w}_t, x) - y\|_2.$$

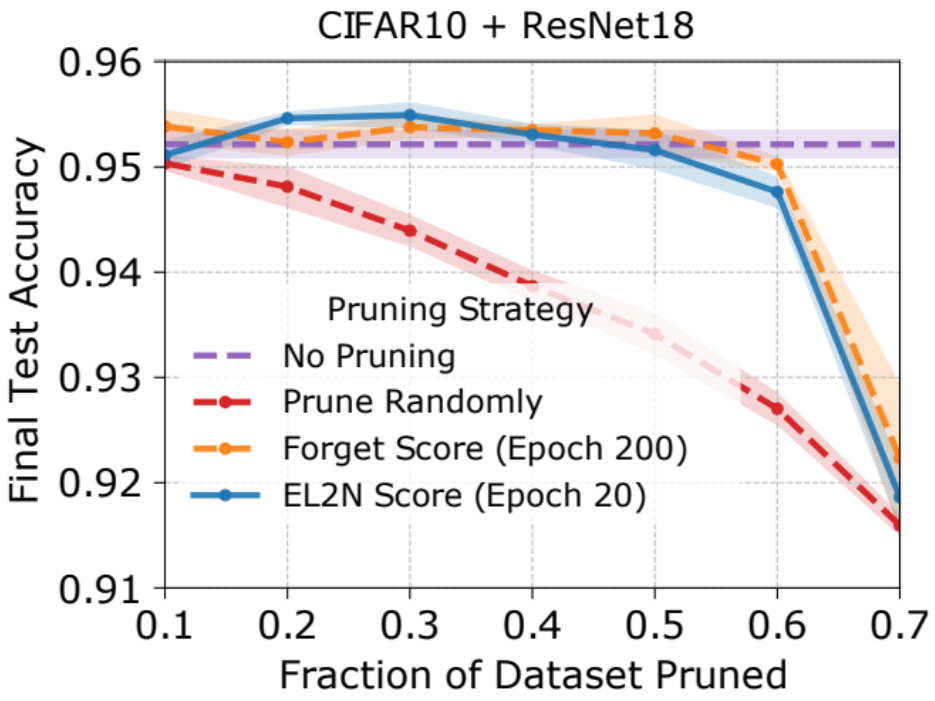
logits vector

one-hot label

Prune samples with smaller error vector scores.

Dataset Pruning: performance

EL2N Performance



Comparison of **forgetting scores at the end of training** and **EL2N scores early in training** (at epoch 20).

Outline

Dataset Pruning

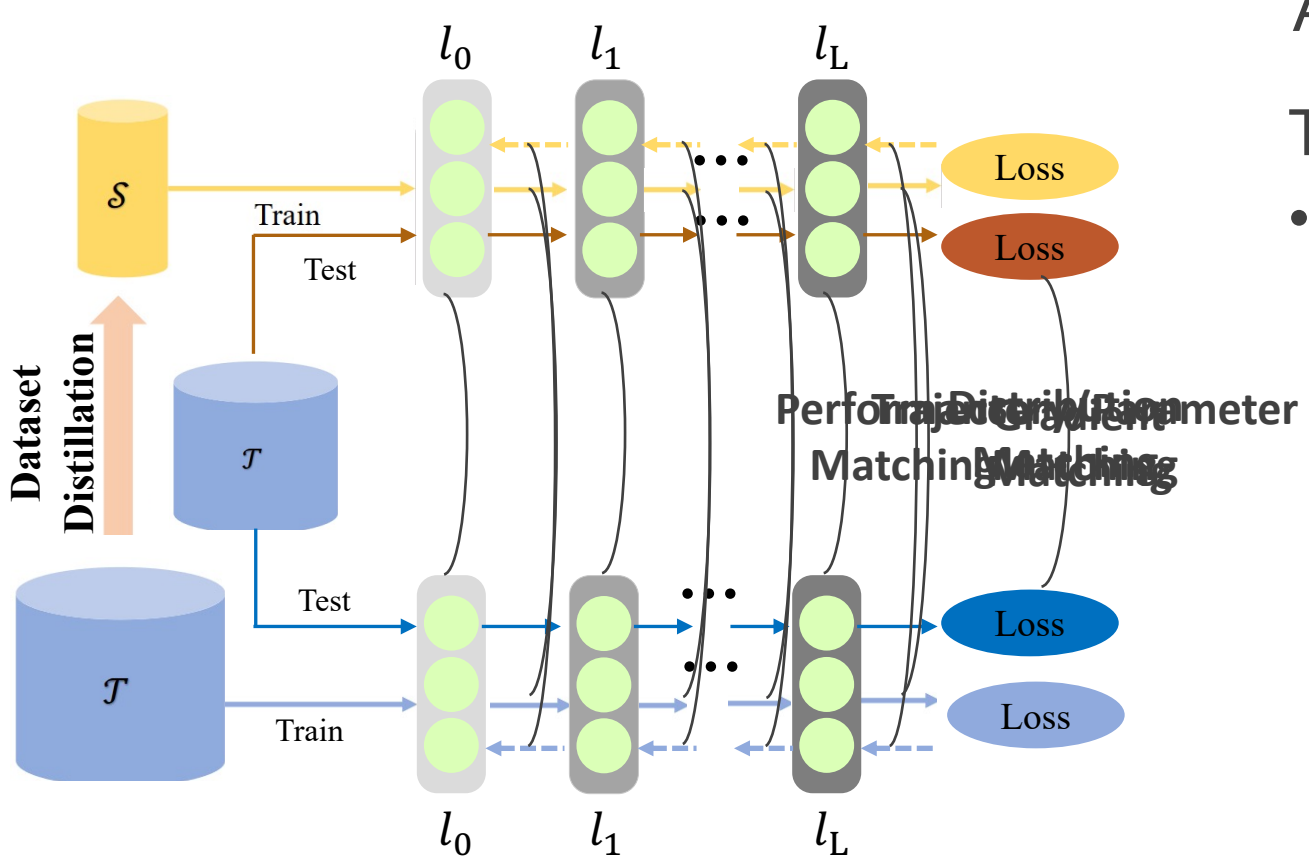
Select a subset images in a full dataset without performance drop.

Dataset Distillation

Learn a few synthetic images (alter image pixels) to replace full dataset.

- Performance Matching
- Gradient Matching
- Distribution Matching
- Trajectory Matching

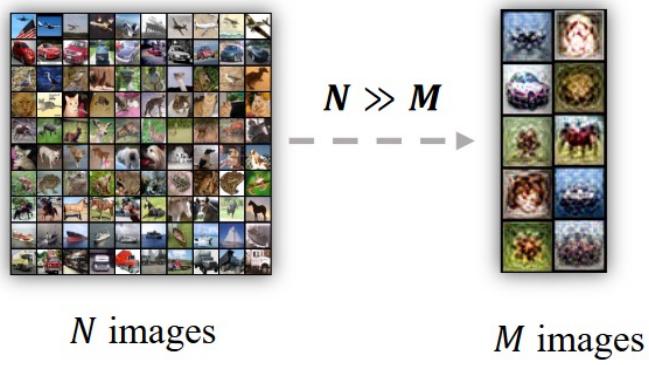
Dataset Distillation



Also known as **Dataset Condensation**.

The Objective:

- **Learn a few synthetic images (alter image pixels)** to replace full dataset.



1 image per class (IPC)

Dataset Distillation

Compared with Dataset Pruning:

Dataset Pruning Objective:

- **Learn a few synthetic images (alter image pixels)** to replace full dataset.

1 image per class (IPC)

Dataset Distillation Objective:

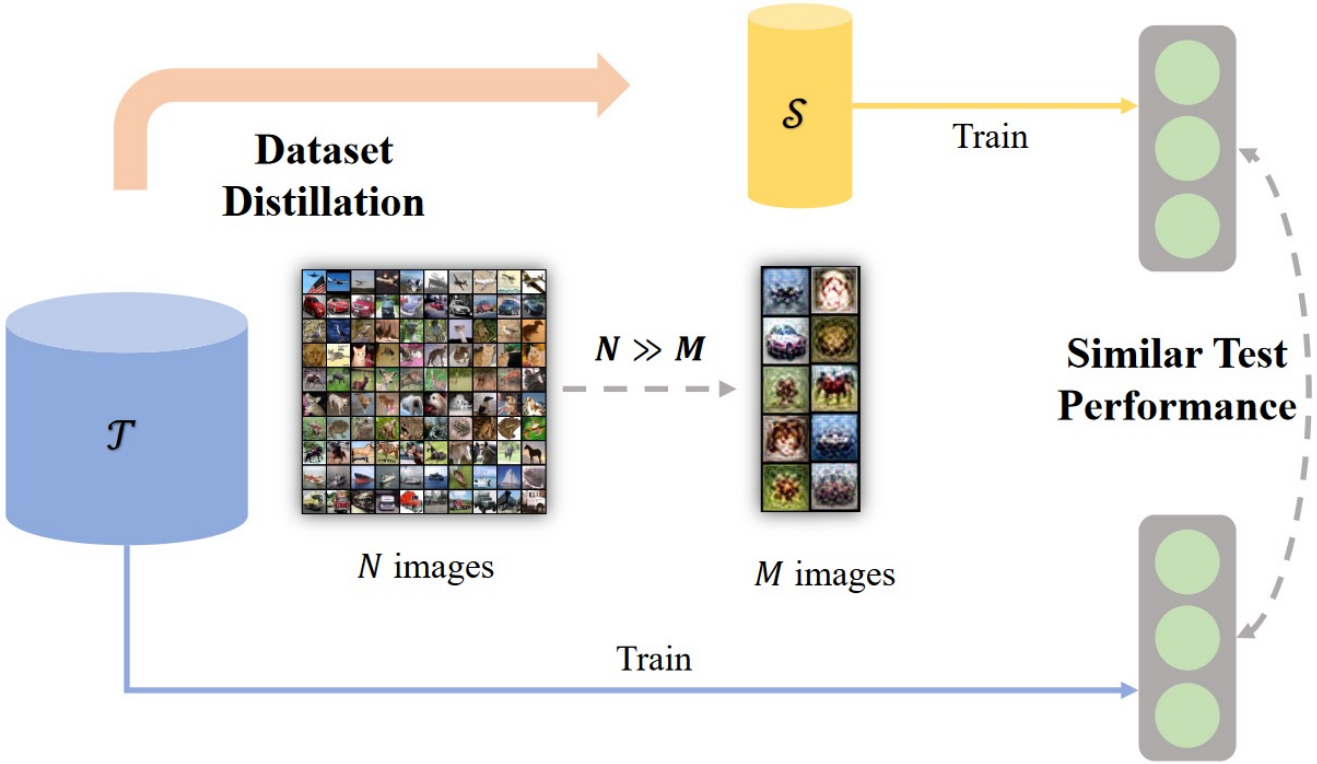
- **Select a subset images** in a full dataset without performance drop.

Whether the image pixels are learnable

Dataset distillationg, 2018.

Dataset Distillation

A Milestone Paper: Performance Matching (2018)

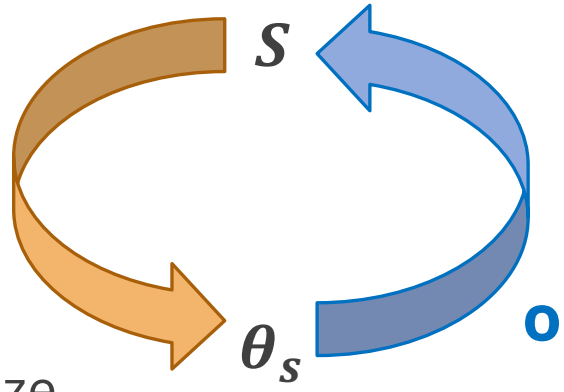


Minimizes the performance gap for models trained on the original full dataset T and the synthetic data S.

Dataset Distillation

Performance Matching (2018)

Formulate Dataset Distillation as a Bi-level Optimization-based Problem.



inner loop:

trains a network ϕ_{θ_s} to minimize the training loss on the distilled data S.

$$\boldsymbol{\theta}_{\mathcal{S}} = \arg\min_{\boldsymbol{\theta}} \mathcal{L}_{\mathcal{S}}(\boldsymbol{\theta})$$

$$\mathcal{L}_{\mathcal{S}}(\boldsymbol{\theta}) = \mathbb{E}_{(\tilde{\boldsymbol{x}},\tilde{\boldsymbol{y}})\in\mathcal{S}}[\ell(\phi_{\boldsymbol{\theta}_{\mathcal{S}}}(\tilde{\boldsymbol{x}}), \tilde{\boldsymbol{y}})]$$

outer loop:

to update S by minimizing the performance gap between models trained on the original full dataset $\boldsymbol{T}$ and the synthetic data S.

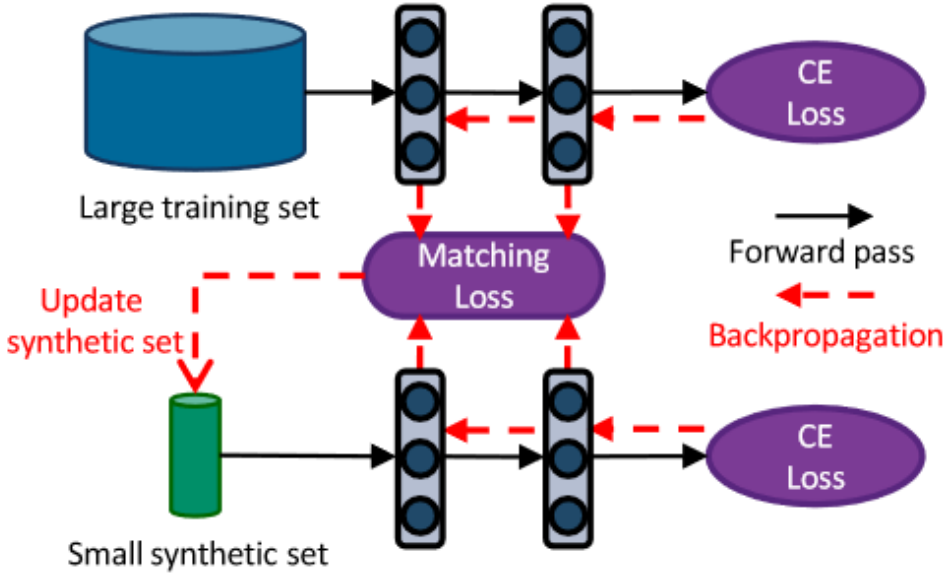
$$\arg\min_{\mathcal{S},|\mathcal{S}|}\left(\sup\{|\ell(\phi_{\boldsymbol{\theta}_{\mathcal{T}}}(\boldsymbol{x}), \boldsymbol{y}) - \ell(\phi_{\boldsymbol{\theta}_{\mathcal{S}}}(\boldsymbol{x}), \boldsymbol{y})|\}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{T}}\right)$$

Dataset Distillation

Problem/Challenge of "Performance Matching":

The granularity of performance-based matching is too coarse.

A Solution: Gradient Matching (ICLR 2021)



A network is updated using S and T for 1 step, respectively, and their resultant gradients with respective to $\boldsymbol{\theta}$ are encouraged to be consistent.

Gradient Matching utilizes a finer-grained matching.

Dataset condensation with gradient matching, ICLR 2021.

Dataset Pruning

"Gradient Matching" Performance

Dataset	Img/Cls	DD	Ours	Whole Dataset
MNIST	1	-	85.0±1.6	99.5±0.0
	10	79.5±8.1	**93.9±0.6**	
CIFAR10	1	-	24.2±0.9	83.1±0.2
	10	36.8±1.2	**39.1±1.2**	

Comparison to "Performance Matching" (DD in the table) in terms of testing accuracy (%).

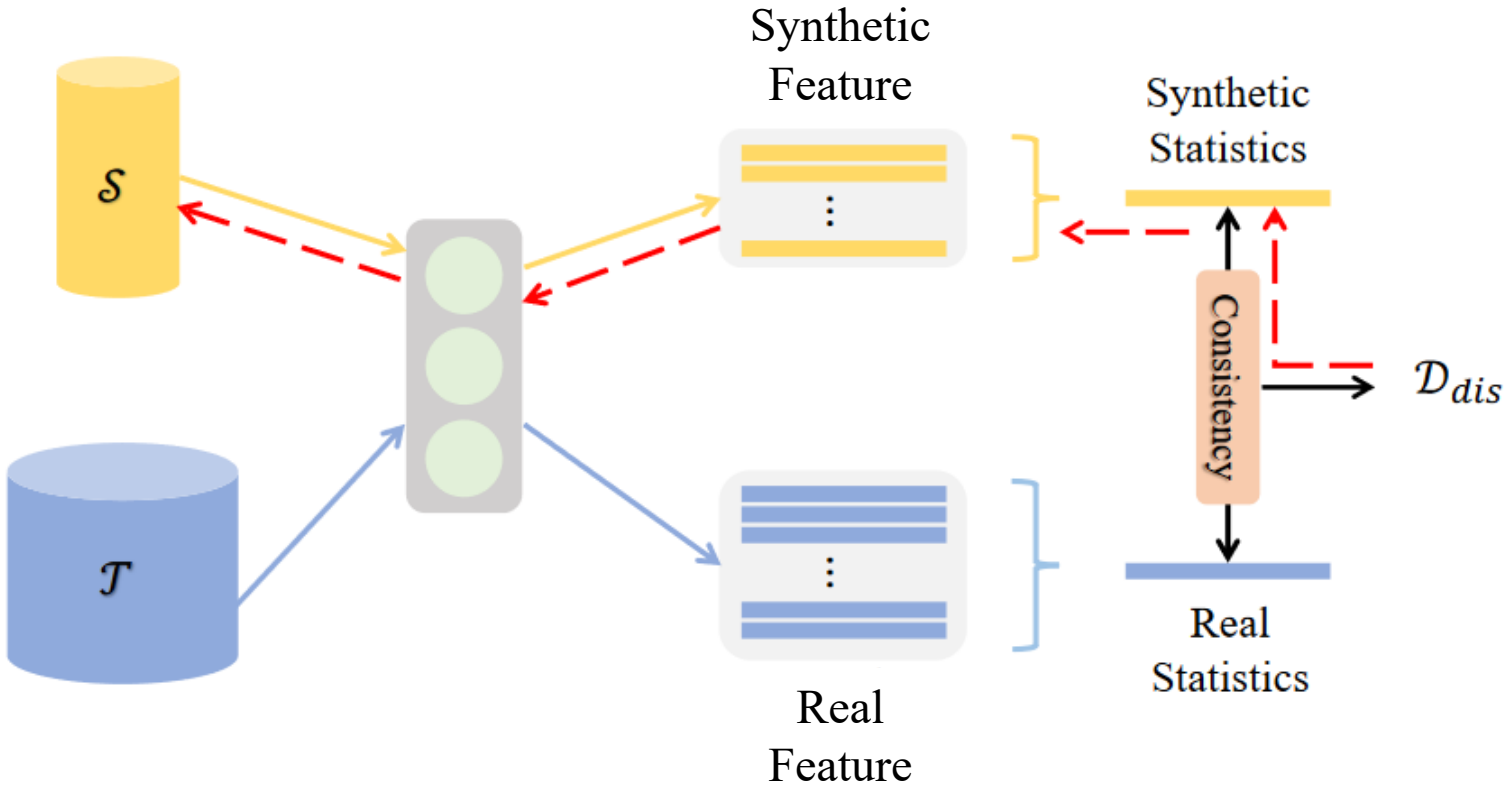
"Gradient Matching" trains 2 times faster than "Performance Matching" and requires 50% less memory on CIFAR10 experiments.

Dataset Distillation

Problem/Challenge of "Gradient Matching":

Matching gradients may result in poor generalization performance on unseen architectures.

A Solution: Distribution Matching (CVPR 2022)



Dataset Pruning

"Distribution Matching" Performance

	IPC	Ratio %	DD†	DC	CAFE
CIFAR10	1	0.02	-	28.3±0.5	30.3±1.1
	10	0.2	36.8±1.2	44.9±0.5	46.3±0.6
	50	1	-	53.9±0.5	55.5±0.6
CIFAR100	1	0.2	-	12.8±0.3	12.9±0.3
	10	2	-	25.2±0.3	27.8±0.3
	50	10	-	-	37.9±0.3

Comparison between "Distribution Matching" (CAFE in the table) with "Performance Matching" (DD in the table) and Gradient Matching" (DC in the table) in terms of testing accuracy (%).

Dataset Pruning

"Distribution Matching" Performance

	C\T	ConvNet	AlexNet	VGG11	ResNet18	MLP
DC	ConvNet	53.9±0.5	28.77±0.7	38.76±1.1	20.85±1.0	28.71±0.7
CAFE	ConvNet	**55.50±0.4**	**34.02±0.6**	**40.55±0.8**	**25.27±0.9**	**36.67±0.6**

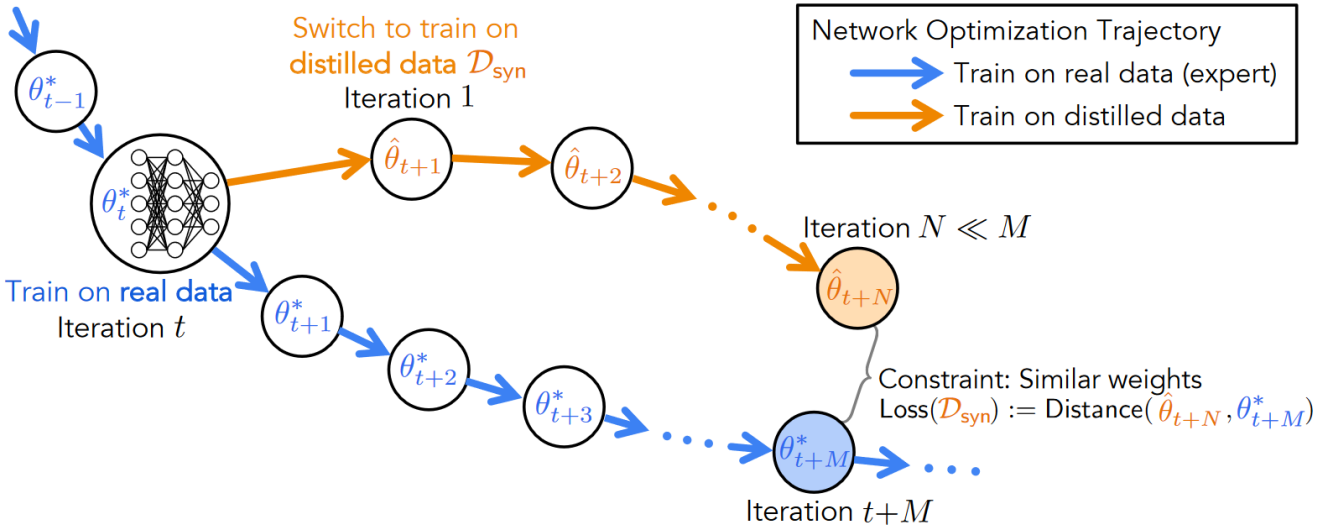
The testing performance (%) on **unseen** architectures. The 50 IPC synthetic set is learned on one architecture (C), and then tested on another architecture (T). "Gradient Matching" is represented by "DC" in the table.

Dataset Distillation

Problem/Challenge of "Gradient Matching":

Short-range matching (gradient matching) may accumulate errors over many training steps.

A Solution: Trajectory Matching (CVPR 2022)



$$\mathcal{L} = \frac{\|\hat{\theta}_{t+N} - \theta^*_{t+M}\|_2^2}{\|\theta^*_t - \theta^*_{t+M}\|_2^2}$$

Network parameters trained on distilled data

Network parameters trained on real data

Dataset Distillation

“Trajectory Matching” Performance

	Img/Cls	Ratio %	DD† [44]	DC [47]	CAFE [43]	Ours	Full Dataset
CIFAR-10	1	0.02	-	28.3 ± 0.5	30.3 ± 1.1	$\mathbf{46.3 \pm 0.8}^*$	84.8 ± 0.1
	10	0.2	36.8 ± 1.2	44.9 ± 0.5	46.3 ± 0.6	$\mathbf{65.3 \pm 0.7}^*$	
	50	1	-	53.9 ± 0.5	55.5 ± 0.6	$\mathbf{71.6 \pm 0.2}$	
CIFAR-100	1	0.2	-	12.8 ± 0.3	12.9 ± 0.3	$\mathbf{24.3 \pm 0.3}^*$	56.2 ± 0.3
	10	2	-	25.2 ± 0.3	27.8 ± 0.3	$\mathbf{40.1 \pm 0.4}$	
	50	10	-	-	37.9 ± 0.3	$\mathbf{47.7 \pm 0.2}^*$	
Tiny ImageNet	1	0.2	-	-	-	$\mathbf{8.8 \pm 0.3}$	37.6 ± 0.4
	10	2	-	-	-	$\mathbf{23.2 \pm 0.2}$	
	50	10	-	-	-	$\mathbf{28.0 \pm 0.3}$	

Comparison to “Performance Matching” (DD in the table), “Gradient Matching” (DC in the table) and “Distribution Matching” (CAFE in the table) in terms of testing accuracy (%).

Reading List

- Denton, Emily L., et al. "Exploiting linear structure within convolutional networks for efficient evaluation." Advances in Neural Information Processing Systems. 2014.
- Jin, Jonghoon, Aysegul Dundar, and Eugenio Culurciello. "Flattened convolutional neural networks for feedforward acceleration." arXiv preprint arXiv:1412.5474 (2014).
- Gong, Yunchao, et al. "Compressing deep convolutional networks using vector quantization." arXiv preprint arXiv:1412.6115 (2014).
- Han, Song, et al. "Learning both weights and connections for efficient neural network." Advances in Neural Information Processing Systems. 2015.
- Guo, Yiwen, Anbang Yao, and Yurong Chen. "Dynamic Network Surgery for Efficient DNNs." Advances In Neural Information Processing Systems. 2016.
- Gupta, Suyog, et al. "Deep Learning with Limited Numerical Precision." ICML. 2015.
- Courbariaux, Matthieu, Yoshua Bengio, and Jean-Pierre David. "Binaryconnect: Training deep neural networks with binary weights during propagations." Advances in Neural Information Processing Systems. 2015.
- Courbariaux, Matthieu, et al. "Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1." arXiv preprint arXiv:1602.02830 (2016).
- Han, Song, Huizi Mao, and William J. Dally. "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding." arXiv preprint arXiv:1510.00149 (2015).
- Iandola, Forrest N., et al. "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size." arXiv preprint arXiv:1602.07360 (2016).
- Zhehui Wang, Tao Luo, Joey Tianyi Zhou, Rick Goh, “EDCompress: Energy-Aware Model Compression with Dataflow”, IEEE TNNLS, 2022.
- Tian Huang, Tao Luo, Ming Yan, Joey Tianyi Zhou, Rick Goh, “RCT: Resource Constrained Training for Edge AI”, In TNNLS, 2022

That is the end of Part I for CEG5304/EE 6934.
Thank you!
Wish all of you success in Deep Learning!