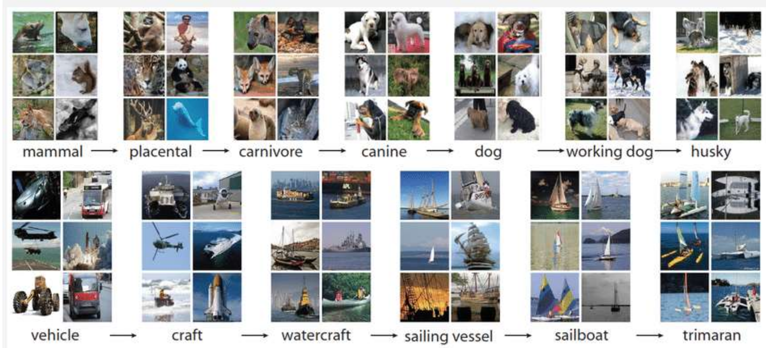
Lightweighting DL Models

June 04-11, 2021
Kyong-Ha Lee
bart7449@gmail.com

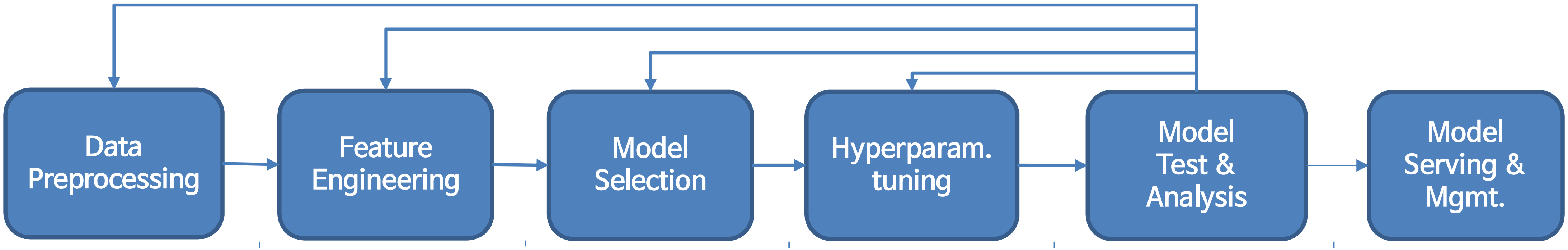
I. DL process

- 대량의 Labeled data의 필요
  - 주로 지도 학습 기반으로 모델 훈련을 위해 대량의 정답 집합이 요구됨
- 반복적인 단계별 시행착오 과정의 연속
  - 개발자가 요구하는 정확도에 도달할 때까지 여러 단계에서 사람이 개입하면서 trial-or-error 방식으로 반복적인 훈련, 테스트 과정을 수행
  - *반복적인* Human-In-The-Loop 프로세스



- 학습과 추론에 높은 수준의 전산 자원 요구
  - 많은 계층, 많은 파라미터 개수, 그에 따른 연산 수 및 메모리 증가
    - 예) BERT Large – 24 layers, 340M parameters, koBERT의 경우 V100 GPU x 32, horovod(w/ InfiniBand)로 학습에 약 한달 소요
- 학습 과정은 Batch 과정으로 수행
  - 데이터 변화에 따른 모델 재훈련 필요

Challenge 1 : Human-In-The-Loop Process



Data Preprocessing

- Data transforming
- Data cleansing
- Data labeling
- Problem
  - Inconsistency
  - Noise
  - Missing values
  - Imbalanced data
  - Outliers
  - Highly correlated data
  - Target leakage
  - Data size vs. memory size

Feature Engineering

- Estimating feature importance
- Feature selection
- Data vectorization
- Removing Highly correlated features
- Domain-specific features like POS in NLP

Model Selection

- Selecting a proper model from model collection
- Using pre-trained model
- Model design/ modification

Hyperparam. tuning

- Selecting right values for various tuning knobs :
  - learning rate
  - regularization (weight decay)
  - batch size
  - # of epoch
  - dropout rate
  - optimizer
  - no. of layers
  - no. of units
  - activation

  . . .

Model Test & Analysis

- Model accuracy
- Train loss
- Model fairness

Model Serving & Mgmt.

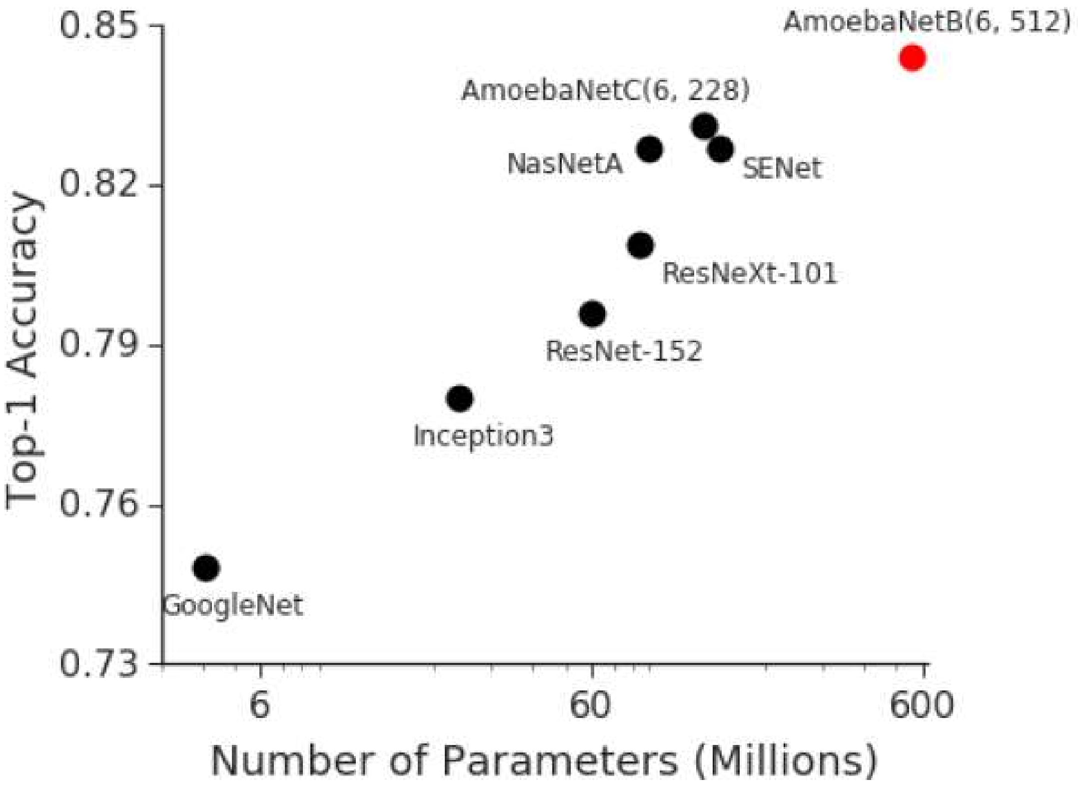
- Deployment
- Metadata mgmt.
- Model version control & mgmt.
- Lightening model for Mobile env.

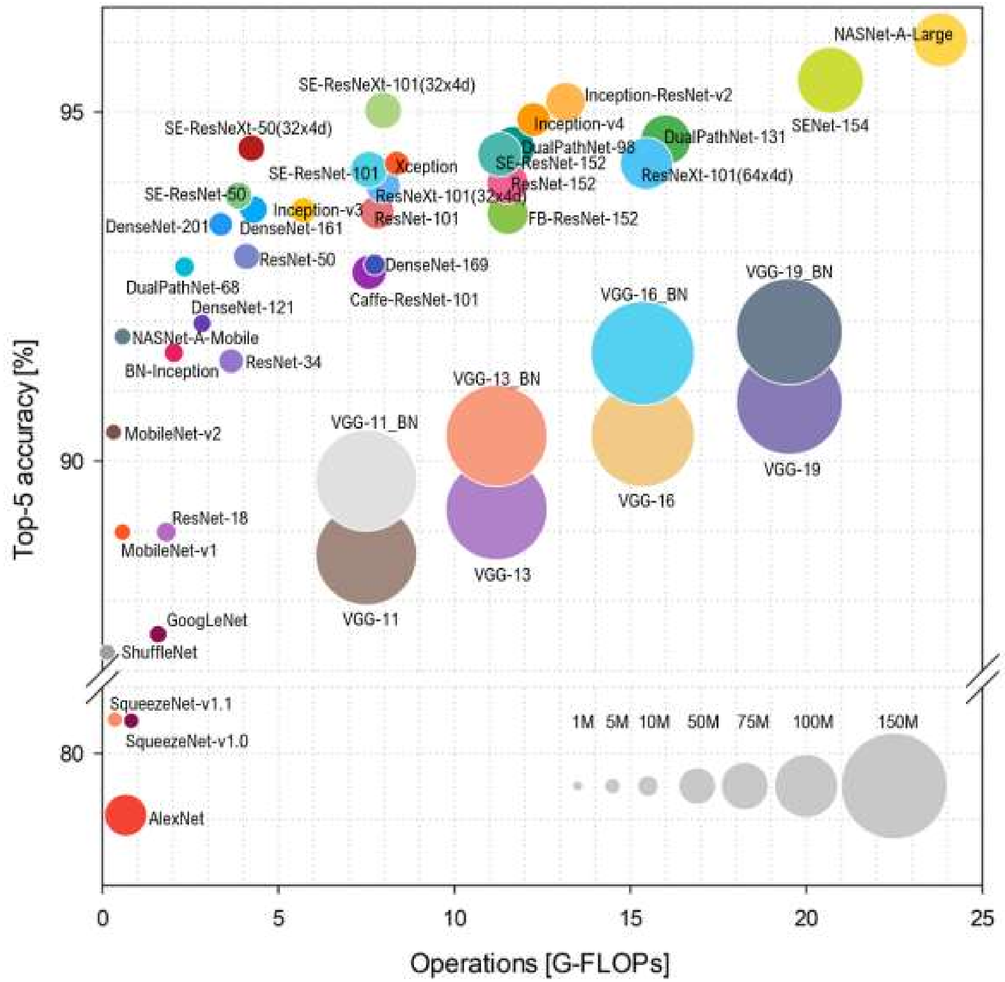
[여러 단계에서 사람의 개입이 요구되며 반복적인 작업]

Challenge 2: Model Size Matters!

KISTI 한국과학기술정보연구원 Korea Institute of Science and Technology Information www.kisti.re.kr

- CNN-based image classifiers





Recent advances have shown that ever-larger DNN models lead to better task performance and past progress in visual recognition tasks has also shown **a strong correlation between the model size and accuracy**

* An analysis of Deep Neural Network Models for Practical Applications, A. Canziani et al., April 2017

** Benchmark analysis of representative deep neural network architecture, Blanco et al., Oct. 2018

Challenge 3: Time & Costs

KISTI 한국과학기술정보연구원 Korea Institute of Science and Technology Information www.kisti.re.kr

Long training time limits ML researcher's productivity

Correlation btw. #layers and time

Model	Error rate	Training time
ResNet18	10.76%	2.5 days
ResNet50	7.02%	5 days
ResNet101	6.21%	1 week
ResNet150	6.16%	1.5 weeks

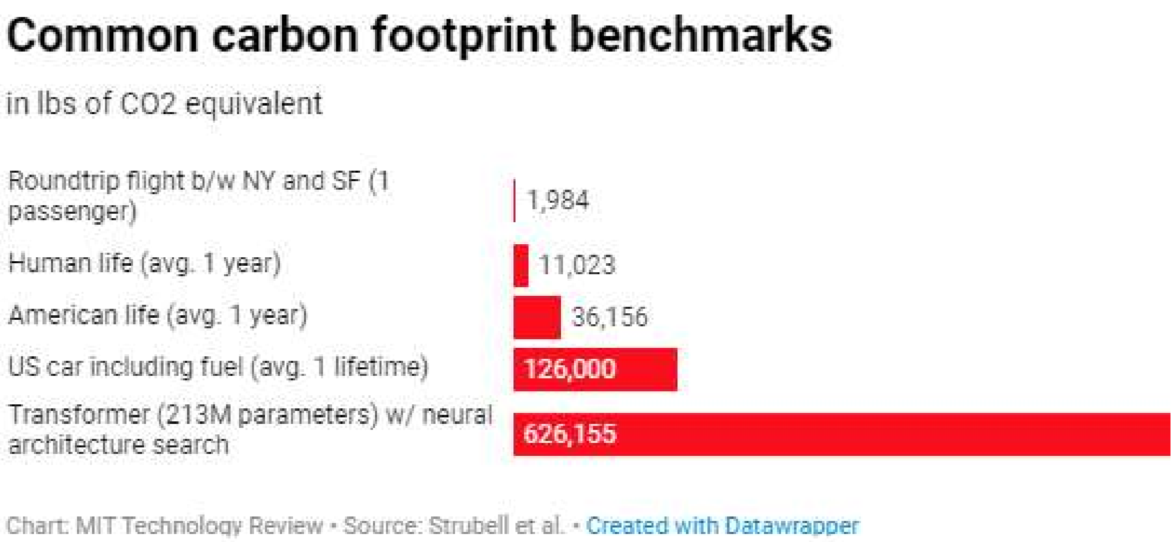
* M40 GPU, fb.resnet.torch

- KoBERT (SKT, Oct. 2019)
  - 24 layers, 340M parameters
  - 1 month with 32 V100 GPUs interconnected with Horovod (w/ infiniBand)

- XLNet (Yang, arXiv 19 Jun 2019)
  - 340 million parameters
  - Training : 2.5 days with 512 TPU v3 chips for 500k steps
  - 512 TPU x 2.5 days x $8 = $ 245,000
- Gpipe (Huang, NIPS Dec. 2019)
  - 556 million parameters
- NASNet (Barret, CVPR June 2018)
  - 800 GPU, 28 days training
- GPT-3 (OpenAI, 2020)
  - 175B parameters, required 3.14E23 FLOPS for training
  - At theoretical 28 TFLOPS for V100, 355 GPU-years and cost $4.6M for a single training run
  - 700GB memory to store it in FP32

Challenge 4: Energy Efficiency

- AlphaGo : 1,920 CPUs and 280 GPUs, $3,000 electric bill per game
- "Training a single DL model can emit as much carbon as 5 cars in their lifetimes" – MIT Tech. Review, 2019



The estimated costs of training a model

	Date of original paper	Energy consumption (kWh)	Carbon footprint (lbs of CO2e)	Cloud compute cost (USD)
Transformer (65M parameters)	Jun, 2017	27	26	$41-$140
Transformer (213M parameters)	Jun, 2017	201	192	$289-$981
ELMo	Feb, 2018	275	262	$433-$1,472
BERT (110M parameters)	Oct, 2018	1,507	1,438	$3,751-$12,571
Transformer (213M parameters) w/ neural architecture search	Jan, 2019	656,347	626,155	$942,973-$3,201,722
GPT-2	Feb, 2019	-	-	$12,902-$43,008

Note: Because of a lack of power draw data on GPT-2's training hardware, the researchers weren't able to calculate its carbon footprint.

Table: MIT Technology Review • Source: Strubell et al. • Created with Datawrapper

Summary

- **DL development is naturally an Iterative HITL process**
  - Learning and test processes are performed in a batch job
  - Usually developed in a trial-and-error fashion
  - **Massive labeled data are required** for learning accurate DL models

- **Recent models become much larger**
  - Even a single model is composed of 175 billion parameters (i.e., GPT-3, ~652GBs)
  - All the SIZE, TIME, and COST grow very fast beyond our resources
    - Learning models tend to become harder with a limited budget and time

- **Expensive computational resources are required**
  - Energy efficiency is also an issue

II. 모델 경량화: Overview

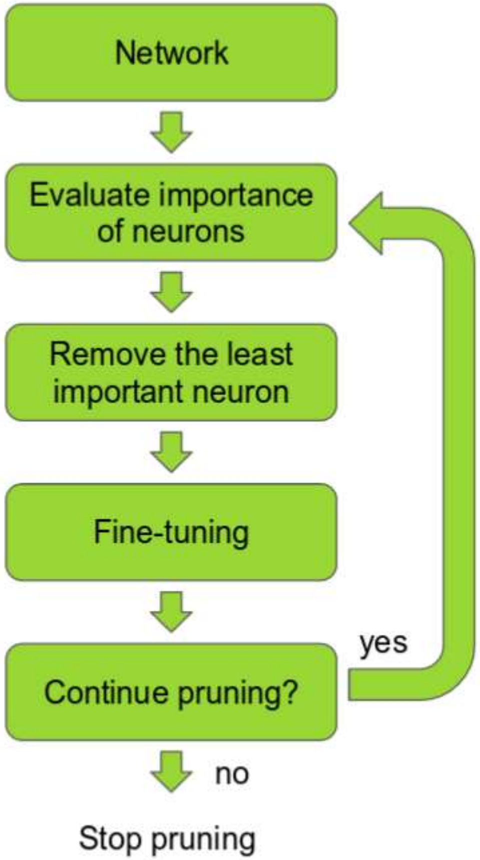
- 기술의 목적
  - 기존 full-strength 모델로부터 정확도 손실을 최소화하면서 보다 고속의 추론이 가능하고 메모리, 에너지 면에서 **효율적인 경량화된 모델**의 생성
  - 모바일 단말 등 저성능의 전산자원에서의 구동을 목적으로 함
  - 모델 압축(model compression)이라고도 함
- 모델 경량화 기법의 분류

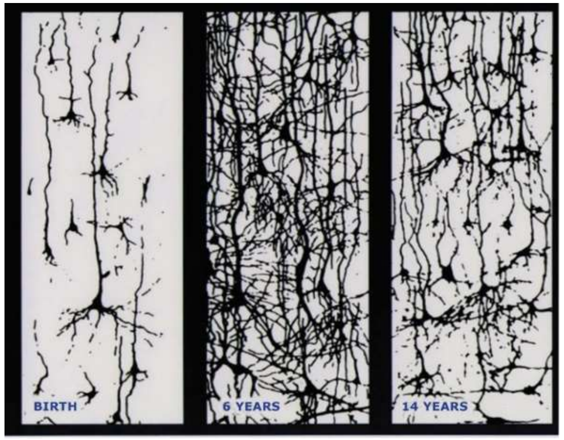
Approaches	Technique	Appraoch
Pruning	• Remove weights or neurons under threshold	• From pre-trained model
Weight sharing	• Cluster weights and encode centroid of the weights	• From pre-trained model
Quantization	• Substitute weight, activations or gradients with lower bit-widths	• From pre-trained model • From scratch
Distillation	• Teacher-student model	• From pre-trained model
Low-rank approximation	• Tensor decomposition on convolutional tensor	• From pre-trained model
Sparse regularization	• Learn a structure-regularized version of CNN to achieve speed-up with little accuracy loss	• From pre-trained model
Compact Network Design	• Revise networks to be more computationally efficient	• From scratch

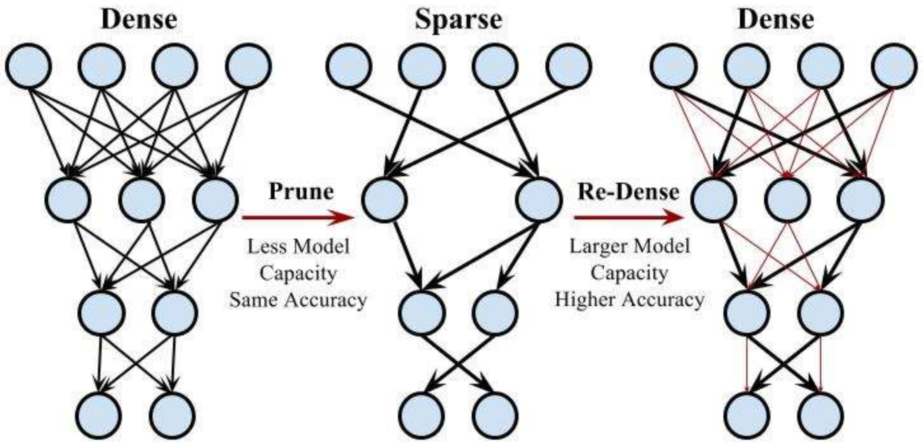
[모바일 단말에의 이식 및 추론을 위한 초기 목적]

II. 모델 경량화: Pruning

- Pruning unimportant neurons in DNN
  - Motivated by how real brain learns
  - **Remove** weights which $|weight| < threshold$
  - **Retrain** after pruning weights
  - Learn effective connections by iterative pruning







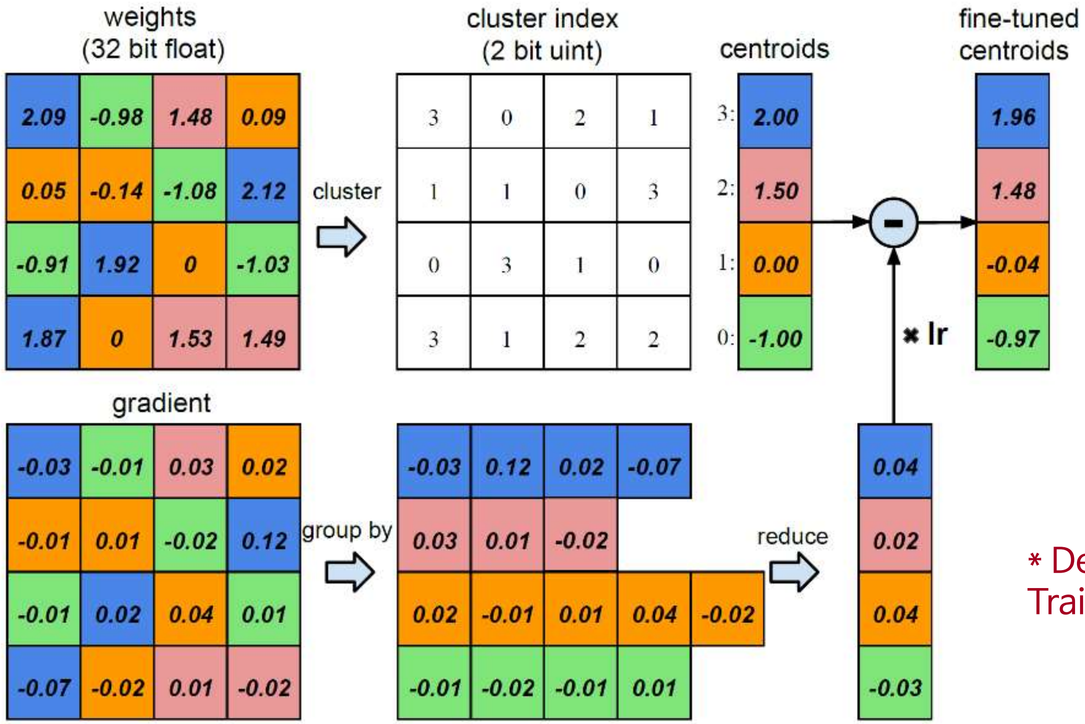
[기존 모델의 가공 및 재학습을 통한 모델 경량화]

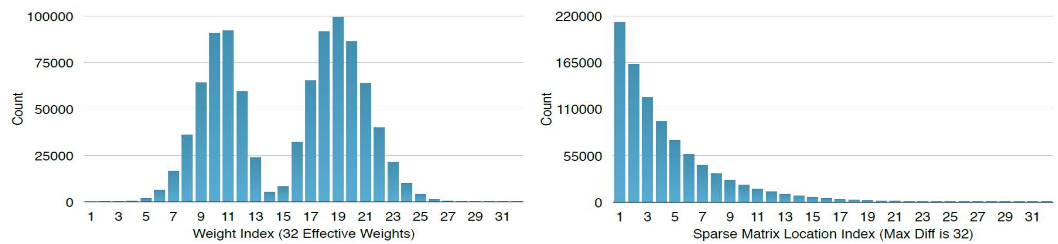
Network	Top-1 Error	Top-5 Error	Parameters	Compression Rate
LeNet-300-100 Ref	1.64%	-	267K	
LeNet-300-100 Pruned	1.59%	-	**22K**	**12**×
LeNet-5 Ref	0.80%	-	431K	
LeNet-5 Pruned	0.77%	-	**36K**	**12**×
AlexNet Ref	42.78%	19.73%	61M	
AlexNet Pruned	42.77%	19.67%	**6.7M**	**9**×
VGG-16 Ref	31.50%	11.32%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

- Song Han, Compressing and regularizing deep neural networks, 2016
- Pavlo Molchanov et. al., “Pruning Convolutional Neural Networks for Resource Efficient Inference,” ICLR 2017

II. 모델 경량화 : Weight Sharing (1)

- Quantization and Weight Sharing
  - Cluster weights and use **centroid** of clustered weights in indexed array
  - Update centroid weights with **summation** of corresponding gradients
- Huffman Coding
  - Quantized weights are biased
  - Achieve additional compression via encoding weights





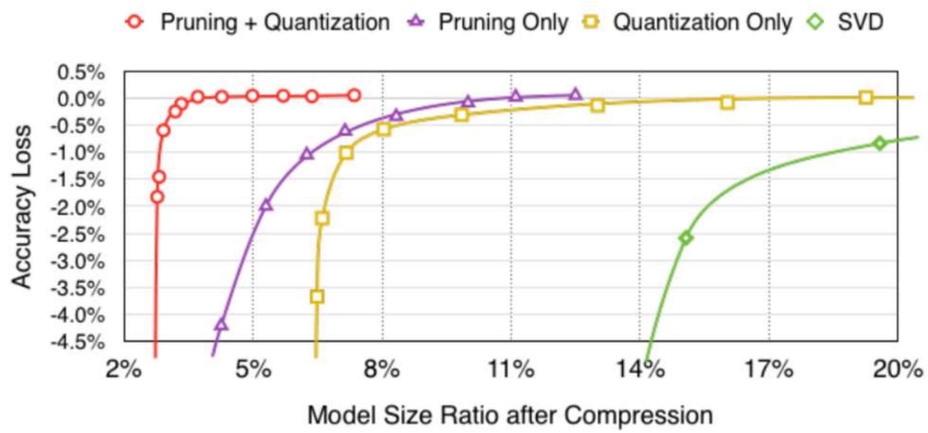
Biased weights

* Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding [Han et al., ICLR 2016]

II. 모델 경량화: Weight Sharing (2)

- Experiments
  - About 40x without critical accuracy loss
  - MNIST data with LeNet-300-100, LeNet-100
  - ImageNet data with AlexNet, VGG-16
- Does pruning and quantization make synergy?
  - Result becomes much better when using two methods together
  - Model can be compressed up to 3% of original size

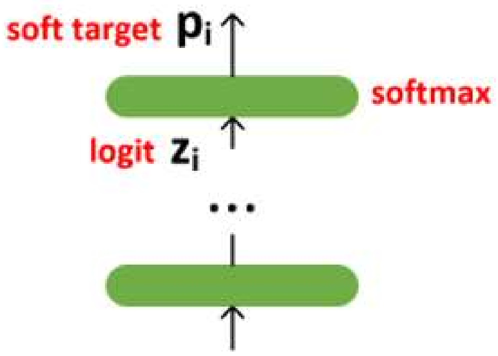
Network	Top-1 Error	Top-5 Error	Parameters	Compress Rate
LeNet-300-100 Ref	1.64%	-	1070 KB	
LeNet-300-100 Compressed	1.58%	-	**27 KB**	40×
LeNet-5 Ref	0.80%	-	1720 KB	
LeNet-5 Compressed	0.74%	-	**44 KB**	39×
AlexNet Ref	42.78%	19.73%	240 MB	
AlexNet Compressed	42.78%	19.70%	**6.9 MB**	35×
VGG-16 Ref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

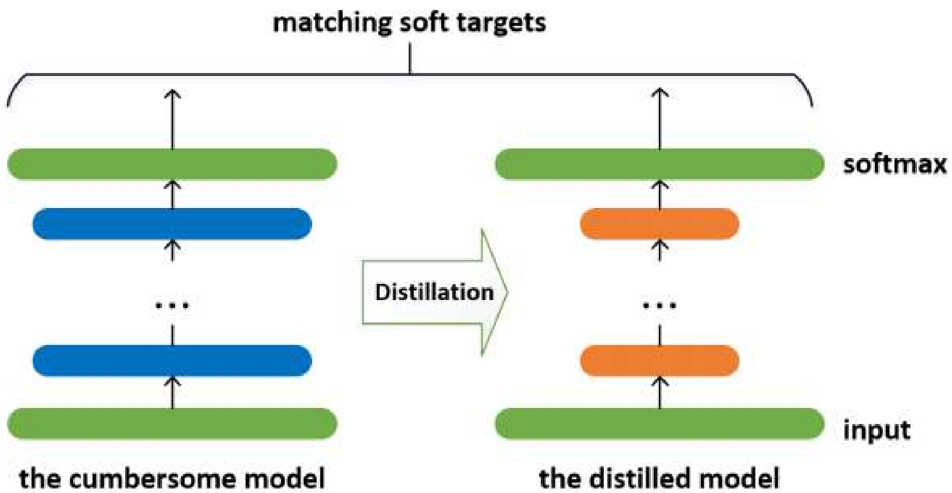


[기존 모델의 가공(클러스터링) 통한 모델 경량화]

II. 모델 경량화: Distillation (1)

- Transfer the generalization ability of the cumbersome model to a small model
  - Use the class probabilities produced by the cumbersome model as "soft targets" for training the small model
  - Use the same training set or separate "transfer set" for the transfer stage
- The Algorithm
  1. Feed teacher with data (T_1)
  2. Obtain soft targets from teacher (T_1)
  3. Train student on soft targets (T_1) with cross-entropy loss
  4. Use student with $T \langle T_1$





II. 모델 경량화: Distillation (2)

- **Model Compression via Distillation and Quantization** [Polino et al., ICLR'18]
  - Given trained deep neural network (DNN), called 'teachers', make a compressed 'student' model, with similar accuracy using quantization and distillation
  - Teacher = Original Deep Model
  - Student = Quantized Model

II. 모델 경량화: Distillation (3)

- Quantization Process
  - Map each data point to the nearest quantization point.
  - Given quantization function $\hat{Q}()$,

$$Q(v) = \alpha \hat{Q}\left(\frac{v-\beta}{\alpha}\right) + \beta.$$

where $\alpha = max_i v_i - min_i v_i$, and $\beta = min_i v_i$

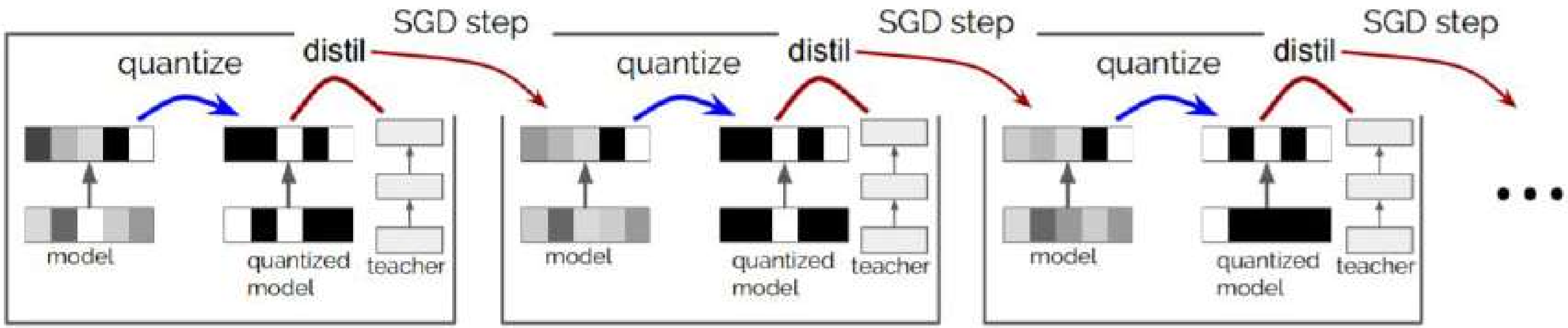
- The range of $\hat{Q}()$ function is [0, 1]
  - Example : if we quantize v in [10, 50] into 5 quantization points, each v is mapped to the nearest point between [10, 20, 30, 40, 50]
  - Uniform vs non-uniform quantization
    - Quantization points can be learned from data

II. 모델 경량화: Distillation (4)

- Quantized Distillation

Algorithm 1 Quantized Distillation

1: **procedure** QUANTIZED DISTILLATION
2: *Let w be the network weights*
3: *loop*
4: $w^q \leftarrow \text{quant\_function}(w, s)$
5: *Run forward pass and compute distillation loss $l(w^q)$*
6: *Run backward pass and compute* $\frac{\partial l(w^q)}{\partial w^q}$
7: *Update original weights using SGD* ***in full precision*** $w = w - \nu \cdot \frac{\partial l(w^q)}{\partial w^q}$
8: *Finally quantize the weights before returning:* $w^q \leftarrow \text{quant\_function}(w, s)$
9: **return** w^q



II. 모델 경량화: Distillation (5)

CIFAR10 accuracy

		2 bits	4 bits	8 bits
Student model 1 1M param - 4 MB 84.5% - 88.8% X5.25	PM Quant.(No bucket)	9.30 %	67.99 %	88.91 %
	PM Quant. (with bucket)	10.53 %	87.18 %	88.80 %
	Quantized Distill.	82.4 %	88.00 %	88.82 %
	Differentiable Quant.	80.43%	88.31 %	—
Student model 2 0.3M param - 1.27 MB 80.3% - 84.3% X16.5	PM Quant. (No bucket)	10.15 %	68.05 %	84.38 %
	PM Quant. (with bucket)	11.89 %	81.96 %	84.38 %
	Quantized Distill.	74.22 %	83.92 %	84.22 %
	Differentiable Quant.	72.79 %	83.49 %	—
Student model 3 0.1M param - 0.45 MB 71.6% - 78.2% X46.66	PM Quant. (No bucket)	10.15 %	61.30 %	78.04 %
	PM Quant. (with bucket)	10.38 %	72.44 %	78.10 %
	Quantized Distill.	67.02 %	77.75 %	77.92 %
	Differentiable Quant.	57.84 %	77.36 %	—

Teacher model : 5.3M param., 21MB, accuracy 89.71%

OpenNMT dataset Bleu score and perplexity (ppl)

		2 bits	4 bits
Student model 1 81.6M param - 326 MB 14.97 - 16.13 BLEU	PM Quant.(No bucket)	0.00 - $2 \cdot 10^{17}$ ppl	0.24 - $2 \cdot 10^{6}$ ppl
	PM Quant. (with bucket)	4.12 - 125.1 ppl	16.29 - 26.2 ppl
	Quantized Distill.	0.00 - 6645 ppl	15.73 - 25.43 ppl
	Differentiable Quant.	0.7 - 249 ppl	15.01 - 28.8 ppl
Student model 2 64.8M param - 249 MB 14.22 - 15.48 BLEU	PM Quant. (No bucket)	0.00 - $5 \cdot 10^{8}$ ppl	6.65 - 71.78 ppl
	PM Quant. (with bucket)	1.72 - 286.98 ppl	15.19 - 28.95 ppl
	Quantized Distill.	0.00 - 4035 ppl	15.26 - 29.1 ppl
	Differentiable Quant.	0.28 - 306 ppl	13.86 - 31.33 ppl
Student model 3 57.2M param - 228 MB 12.45 - 13.8 BLEU	PM Quant. (No bucket)	0.00 - $3 \cdot 10^{8}$ ppl	5.47 - 106.5 ppl
	PM Quant. (with bucket)	0.24 - 1984 ppl	12.64 - 36.56 ppl
	Quantized Distill.	0.14 - 731 ppl	12 - 37 ppl
	Differentiable Quant.	0.26 - 306 ppl	12.06 - 38.44 ppl

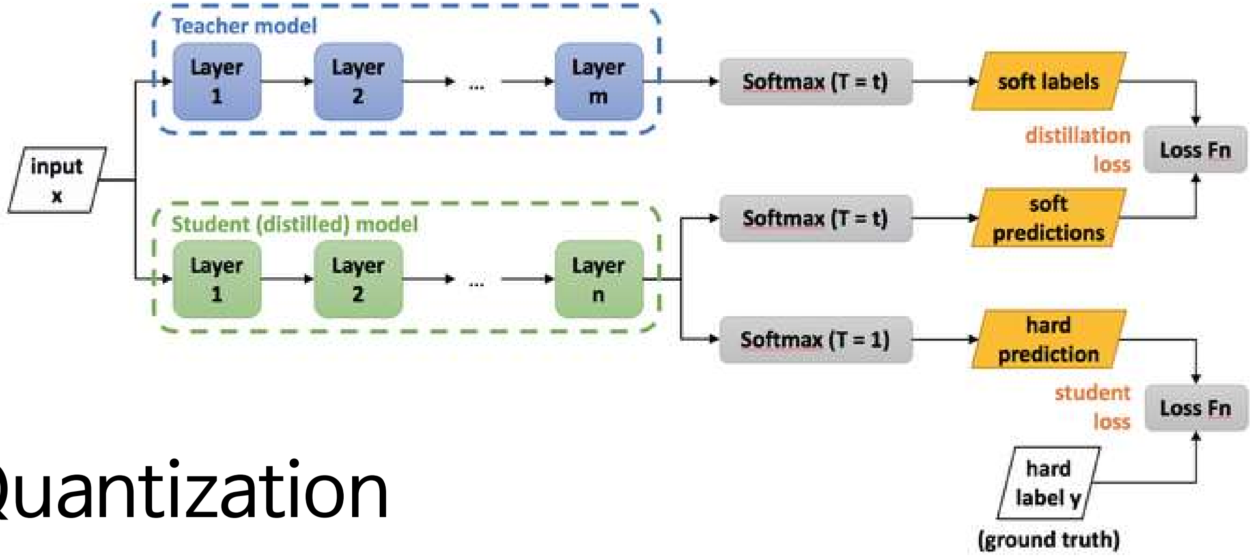
Teacher model : 84.8M param., 340MB, 26.1 ppl 15.88 BLUE

[기존 모델에의 quantization과 Distillation을 통한 모델 경량화]

II. 모델 경량화: Distillation (6)

[Kill-The-Bits'20]

- Knowledge Distillation



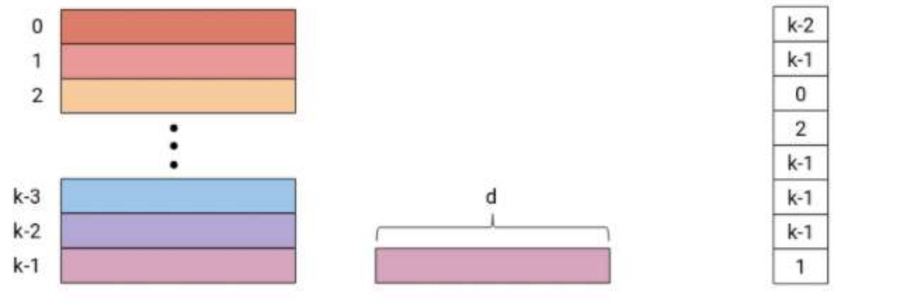
- Product Quantization
  - Was utilized to quantize the weights of convolutional and fully-connected layer

$$||y - \hat{y}||_2^2 = ||x(W - q(W))||_2^2$$

Objective function of quantization
for minimizing reconstruction error

VS

$$||\mathbf{W} - \widehat{\mathbf{W}}||_2^2 = \sum ||\mathbf{w}_j - \mathbf{q}(\mathbf{w}_j)||_2^2$$



(a) Codebook (b) Codeword (c) Indexes to codebook

II. 모델 경량화: Distillation (7)

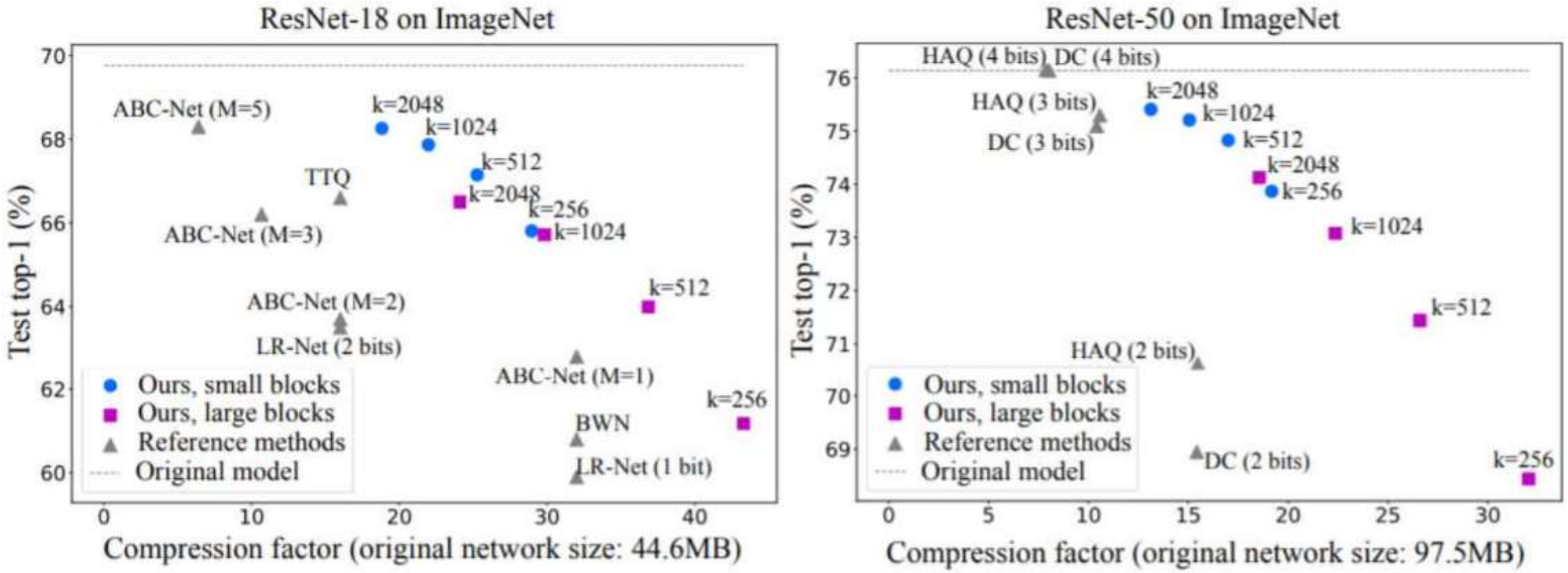


Figure 3: Compression results for ResNet-18 and ResNet-50 architectures. We explore two compression regimes as defined in Section 4.1: small block sizes (block sizes of $d=4$ and 9) and large block sizes (block sizes $d=8$ and 18). The results of our method for $k = 256$ centroids are of practical interest as they correspond to a byte-compatible compression scheme.

Table 1: Results for vanilla ResNet-18 and ResNet-50 architectures for $k = 256$ centroids.

Model (original top-1)	Compression	Size ratio	Model size	Top-1 (%)
ResNet-18 (69.76%)	Small blocks	29x	1.54 MB	**65.81** $\pm$0.04
	Large blocks	43x	1.03 MB	**61.10** $\pm$0.03
ResNet-50 (76.15%)	Small blocks	19x	5.09 MB	**73.79** $\pm$0.05
	Large blocks	31x	3.19 MB	**68.21** $\pm$0.04

III. Quantization: Overview

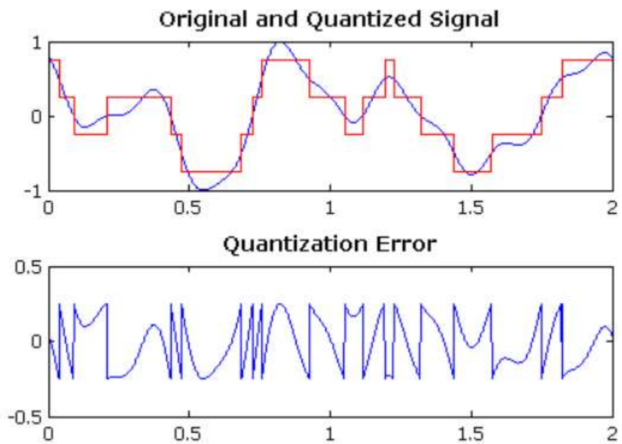
- **Main idea**
  - Quantize FP32 typed weights, activations, or gradients to values with lower bit-widths

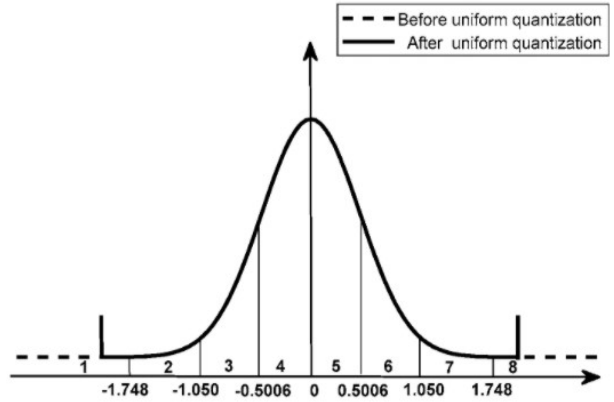
1.12	3.42	-1.5	-12
32	-1	-5	15
24	0.55	-54	0.24
-0.1	0.1	-0.2	2

→

1	1	-1	-1
1	-1	-1	1
1	1	-1	1
-1	1	-1	1

- **Challenges**
  - **Quantization errors** severely affect the precisions of DL models
  - Quantization makes **hard to converge** values of parameters
    - Specifically, values nearby boundaries of each quantization interval may be changed frequently
  - Quantization functions are **non-differentiable**
    - Use of alternative differentiable functions (e.g., STE)
    - Gradient mismatch problem





III. Quantization : Classification

- Methods
  - **Quantization during training**
    - BinaryConnect, Binary-weight, XNOR-Net, DoReFA-Net, LQ-Net, …
  - **Quantization after training**

* For more details, please refer to our survey paper,
E. Kim et al., Communications of the KIISE 38(8):18-29, Aug., 2020

- Codebook

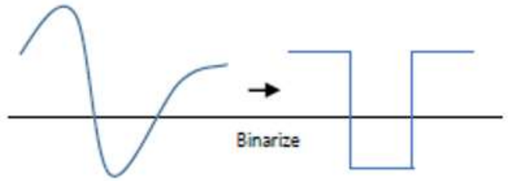
Approaches	Types	codebooks	Representative work
Fixed codebook	• Binarization • Scaled Binarization • Ternarization • Scaled Ternarization • Powers of two • K-bits	• {-1,1} • {-a, b} • {-1, 0, 1} • {-a, 0, b} • $\{0, \pm 1, \pm 2^{-1}, \dots, \pm 2^{-L}\}$ • $\{\pm v1, \{\pm v1 \pm v2\}, \cdots\}$	• BinaryConnect, Binary-weight • XNOR-Net • Ternary Net • TTQ • [Tang and Kwan'94] • DoReFa-Net, LQ-Net
Adaptive codebook	• Soft Quantization • Hard Quantization	• Learned from data • Learned from data	• Variational Network Quantization • Vector quantization, [Choi'2016]

- Targets for quantization

Components	Benefits	Challenges
Weights	• Smaller model size • Faster forward training & inference • Less energy	• Hard to converge with quantized weights • Require approximate gradients • Accuracy degradation
Activations	• Smaller memory foot print during training • Allows replacement of dot-products by bitwise operations • Less energy	• Gradient mismatch problem
Gradients	• Communication & memory savings	• Convergence requirement

- Main idea
  - Train DNNs with binary weights, while retraining precision of the stored weights in which gradients are accumulated to regularize all the parameters
- Binarization
  - Deterministic
    - $$w_b = \begin{cases} +1 & \text{if } w \geq 0, \\ -1 & \text{otherwise.} \end{cases}$$
  - Stochastic
    - $$w_b = \begin{cases} +1 & \text{with probability } p = \sigma(w), \\ -1 & \text{with probability } 1 - p. \end{cases}$$

  where $\sigma(x) = \text{clip}(\frac{x+1}{2}, 0, 1) = \max(0, \min(1, \frac{x+1}{2}))$ is a hard sigmoid function which is used to limit p into [0, 1].
- Propagations & Updates
  - w is binarized during forward and backward propagation
  - Full-precision w is used during parameter update

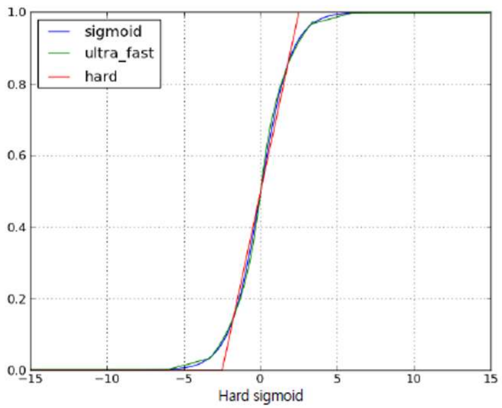


1.12	3.42	-1.5	-12
32	-1	-5	15
24	0.55	-54	0.24
-0.1	0.1	-0.2	2

→

1	1	-1	-1
1	-1	-1	1
1	1	-1	1
-1	1	-1	1

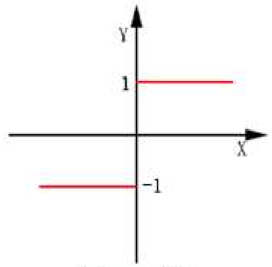
Binarization example



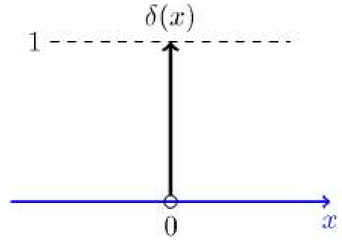
* Binarized Neural Networks: Training deep neural networks with weights and activations constrained to +1 or -1, [Courbariaux et al., 2016]

III. Quantization : Binarization (2)

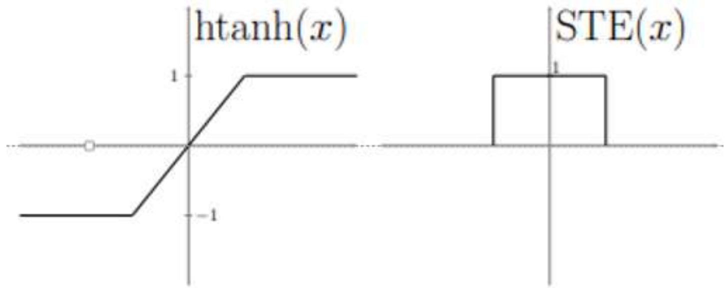
- Forward & Backward propagation
  - The derivative of the sign function is zero almost everywhere -〉 Gradient Vanishing
  - How to back-propagate gradient through ?
    - “straight-through estimator(STE)”, previously introduced by Hinton (2012)



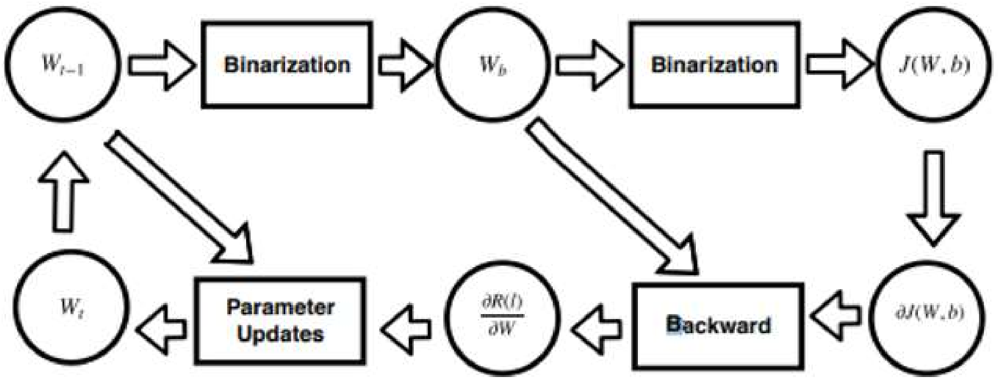
(a) y=sgn(x)

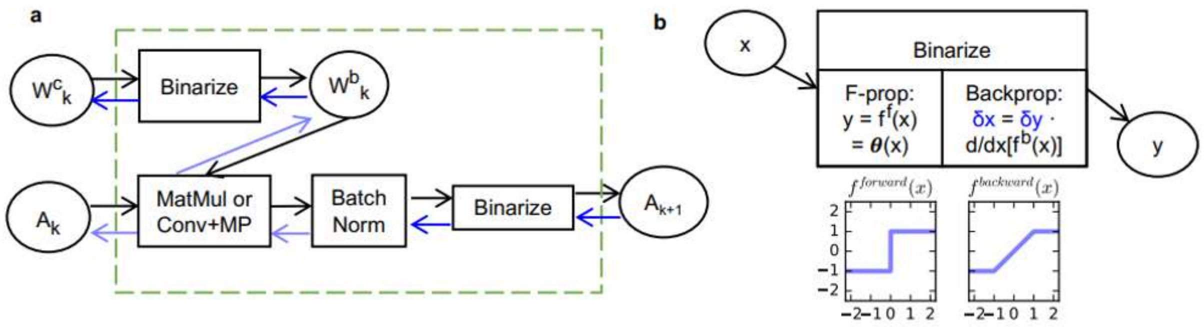


Derivative of Signum function
(form of dirac delta function)



- Procedure



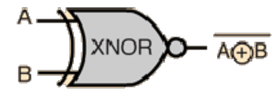


Binary convolution transformer(dashed green line)

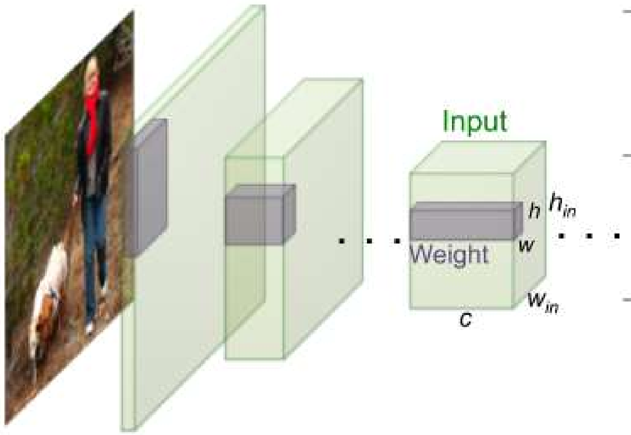
III. Quantization : XNOR-NET (1)

한국과학기술정보연구원
Korea Institute of Science and Technology Information

- Goal: Simple but efficient approximations to CNN by binarizing the weights and intermediate representations in CNN
  - Convolutions can be estimated by only addition and subtraction (X2 speedup)



A	B	Out
0	0	1
0	1	0
1	0	0
1	1	1



	Network Variations		Operations used in Convolution	Memory Saving (Inference)	Computation Saving (Inference)	Accuracy on ImageNet (AlexNet)
Standard Convolution	Real-Value Inputs 0.11 -0.21 ... -0.34 -0.25 0.61 ... 0.52	Real-Value Weights 0.12 -1.2 ... 0.41 -0.2 0.5 ... 0.68	+, −, ×	1x	1x	%56.7
Binary Weight	Real-Value Inputs 0.11 -0.21 ... -0.34 -0.25 0.61 ... 0.52	Binary Weights 1 -1 ... 1 -1 1 ... 1	+, −	~32x	~2x	%56.8
BinaryWeight Binary Input (**XNOR-Net**)	Binary Inputs 1 -1 ... -1 -1 1 ... 1	Binary Weights 1 -1 ... 1 -1 1 ... 1	XNOR, bitcount	~32x	~58x	%44.2

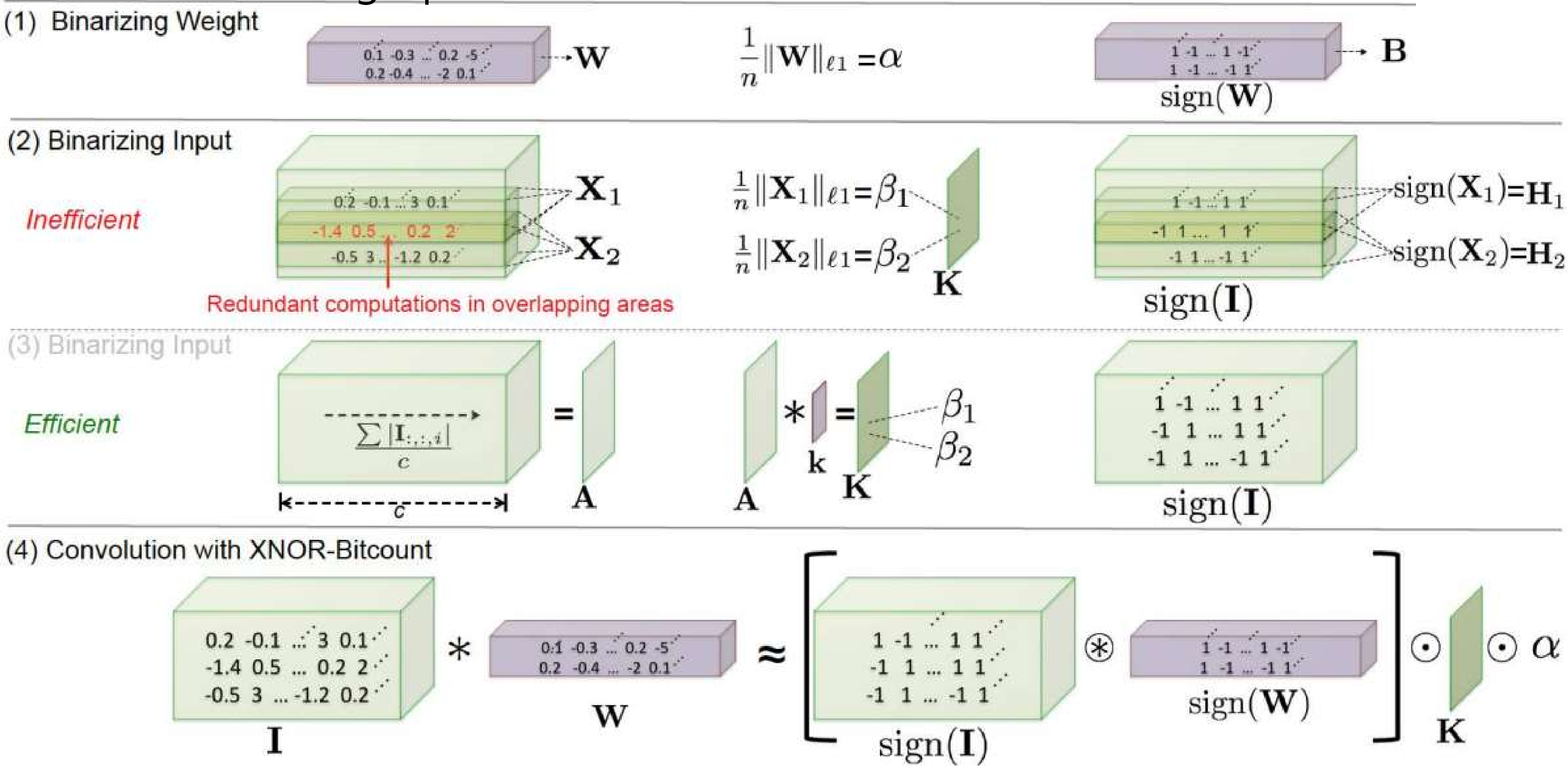
*XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks, Rastegari et al., ECCV 2016

Ⅲ. Quantization : XNOR-NET (2)

$$\mathbf{C}^* = \mathrm{sign}(\mathbf{Y}) = \mathrm{sign}(\mathbf{X}^\intercal)\,\mathrm{sign}(\mathbf{W}) = \mathbf{H}^{*\intercal}\mathbf{B}^*$$

- Binarize both weights and inputs (previous results)
- Convolution as **Binary dot product** implemented by XNOR-BitCounting operations

A	B	Out
-1	1	1
-1	1	-1
1	-1	-1
1	1	1



[기존 연산의 bitwise 연산자로 치환 가능성,Backward Prop.를 위한 Real weight 유지 필요]

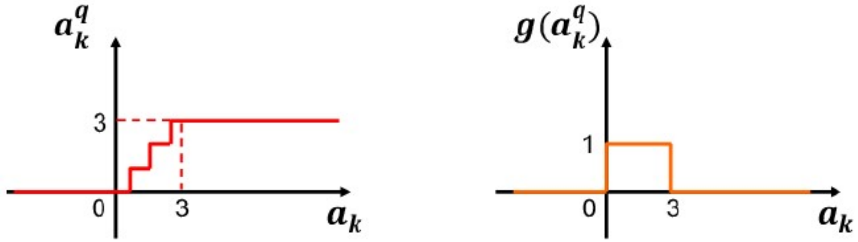
III. Quantization : DoReFa-Net

- Goal: Forward/backward passes operate on low bitwidth weights, activations and gradients
  – Gradients are stochastically quantized before being propagated to convolutions
- Multi-bit quantization

An STE we will use extensively in this work is **quantize$_k$** that quantizes a real number input $r_i \in [0, 1]$ to a k-bit number output $r_o \in [0, 1]$. This STE is defined as below:

$$\textbf{Forward: } r_o = \frac{1}{2^k - 1} \mathrm{round}((2^k - 1) r_i) \tag{5}$$

$$\textbf{Backward: } \frac{\partial c}{\partial r_i} = \frac{\partial c}{\partial r_o}. \tag{6}$$



* DoReFa-Net: Training Low Bitwidth Convolutional Neural Networks with Low Bitwidth Gradients, S. Zhou et al., arXiv:1606.06160

Learnable Quantizer (1/2)

[LQ-Nets'18]

- Existing methods often use simple, hand-crafted quantizers
  - e.g., uniform or logarithmic quantization
- Learnable quantizer
  - Optimal quantizer should yield minimal quantization error for the input data distribution

$$Q^*(x) = \arg\min_Q \int p(x)(Q(x) - x)^2 dx,$$

  - Distributions can be complex and different at different layers

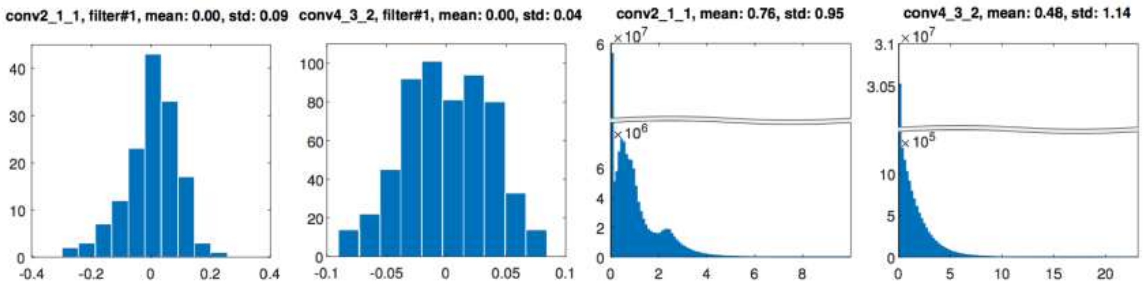
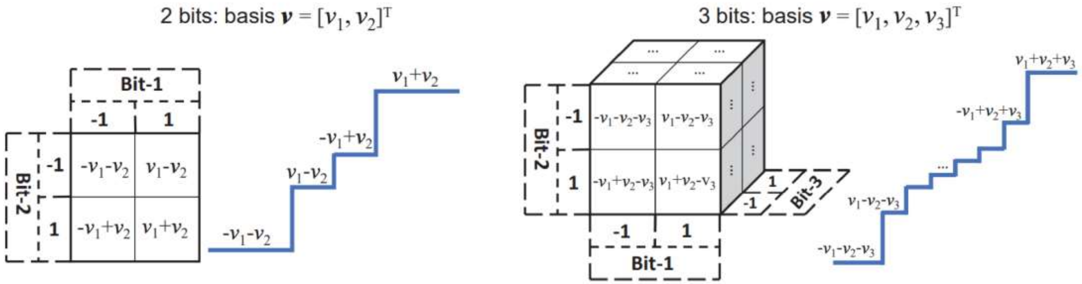


Fig. 1: Distributions of weights (left two columns) and activations (right two columns) at different layers of the ResNet-20 network trained on CIFAR-10. All the test-set images are used to get the activation statistics.

Learnable Quantizer (2/2)

- Bitwise operator compatibility
  - Training quantizers to optimize the quantization level $\{q_i\}$ hampers bitwise operation compatibility
  - Solution is to separate mappings between floating-point basis and bits and learn the basis
  - Confine quantizations into subspaces compatible with bitwise operations



  - An integer q represented by a K-bit binary encoding is the inner product btw. A basis vector and the binary coding vector $\boldsymbol{b}$

$$q = \left\langle \begin{bmatrix} 1 \\ 2 \\ \vdots \\ 2^{K-1} \end{bmatrix}, \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_K \end{bmatrix} \right\rangle, \ \boldsymbol{b} = [b_1, b_2, \ldots, b_K]^T \, where \ b_i \in \{0,1\}$$

Basis vector with K scalars

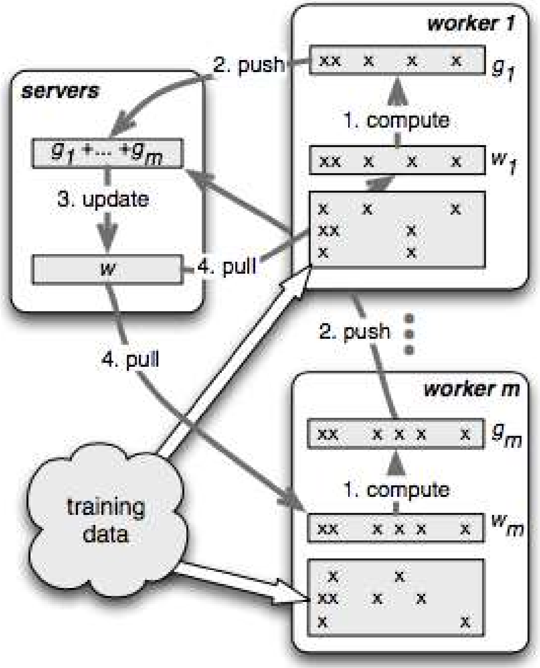
Table 4: Impact of bit-width on our LQ-Nets

ResNet-20 (CIFAR-10)		VGG-Small (CIFAR-10)		ResNet-18 (ImageNet)	
Bit-width (W/A)	Acc. (%)	Bit-width (W/A)	Acc. (%)	Bit-width (W/A)	Acc. (%)
32/32	92.1	32/32	93.8	32/32	70.3
1/32	90.1	1/32	93.5	2/32	68.0
2/32	91.8	2/32	93.8	3/32	69.3
3/32	92.0	3/32	93.8	4/32	70.0
1/2	88.4	1/2	93.4	1/2	62.6
2/2	90.2	2/2	93.5	2/2	64.9
2/3	91.1	2/3	93.8	3/3	68.2
3/3	91.6	3/3	93.8	4/4	69.3

III. Quantization: SketchML (1)

- **Goal** : Build a compression method that can efficiently handle a sparse & nonuniform gradients consisting of key-value pairs in distributed ML

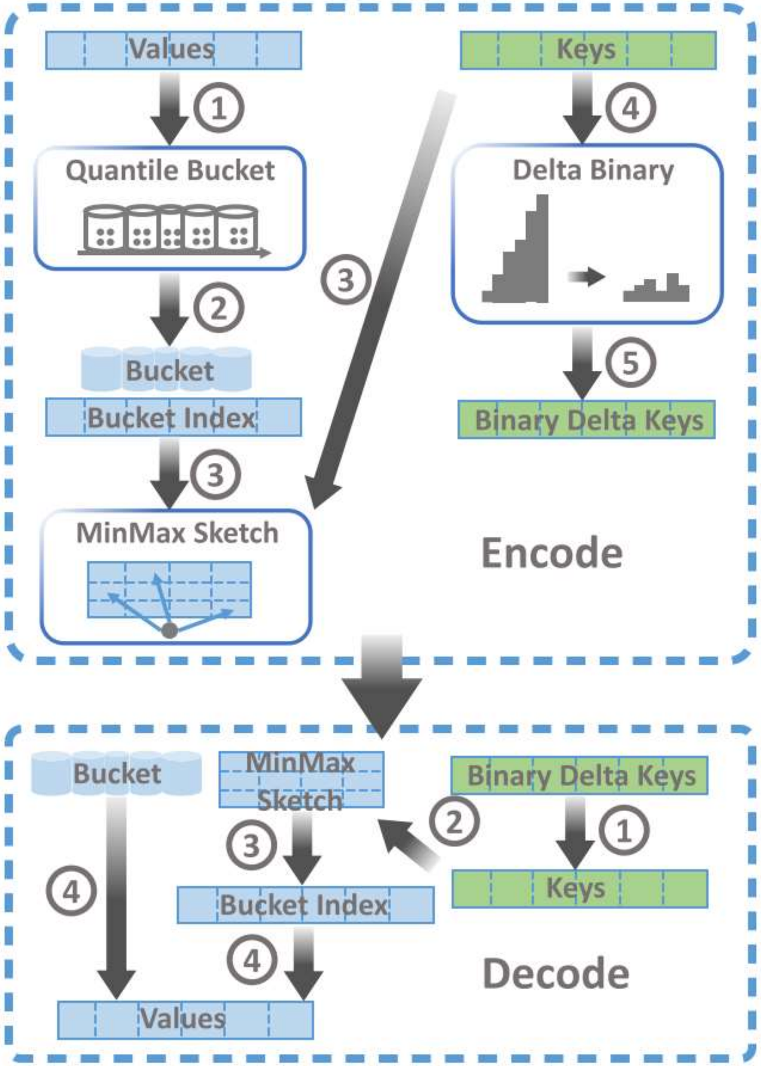
- Distributed ML
  - Data-parallelism
    - Data split over multiple machines
    - Model replicas train over different parts of data & communicate model information periodically
    - Averaging gradients from workers
  - Model-parallelism
    - Models split over multiple machines
    - A single training iteration spans multiple machines
  - Reducing the size of gradients to be communicated is a major challenge



Parameter server [OSDI'14]

* SketchML: Accelerating distributed machine learning with data sketches J. Jiang et al., SIGMOD 2018

Ⅲ. Quantization: SketchML (2)



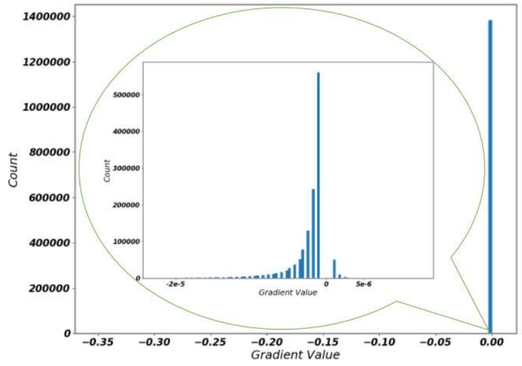
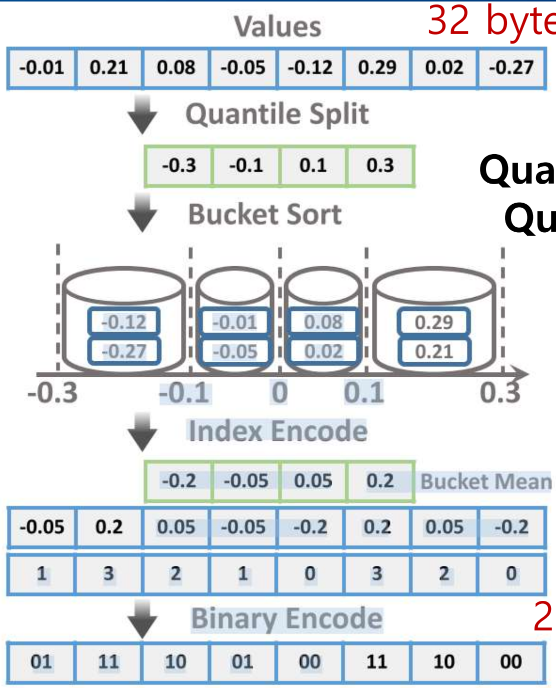


Figure 4: Nonuniform Gradient Values

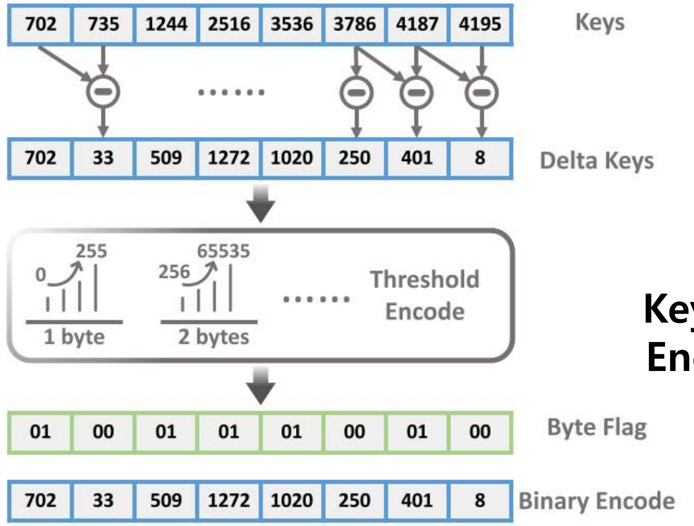
32 bytes (8* 4bytes)



Quantile-Bucket Quantification

2bytes(2bit *8)

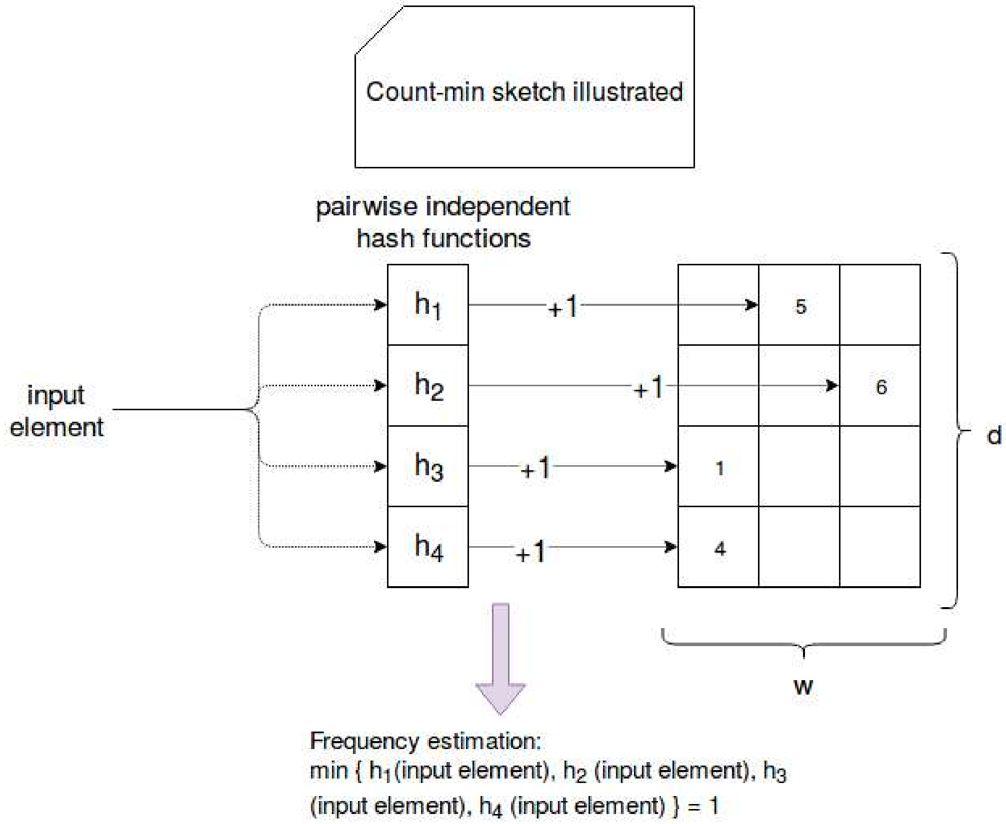
~ X8배 감소



Key Delta Encoding

Appendix: Sketch algorithm overview

- Idea:
  - Summarize your data with a probabilistic data structure
- *Heavy hitters* problem
  - focuses on the retrieval of all elements appearing at least *x%* times in a given data stream
- We can also classify the use of Count-min sketch in the following categories of queries:
  - **Point query** : retrieves the estimated number of occurrences for one particular event
  - **Inner product query** : computes the inner product of 2 vectors.; useful to estimate the join size in relational query processing
  - **Range query** : counts the sum of elements between 2 range values

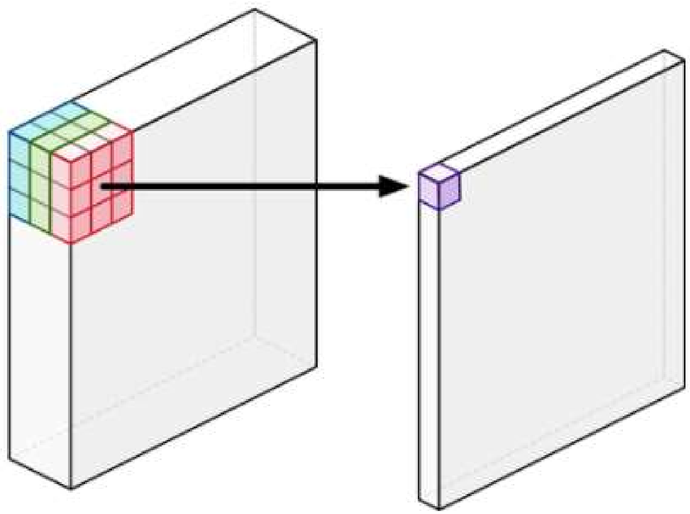


Ⅳ. Compact Network Design (1)

- Problem #1 : Extensive cost
  - With more channels, we can learn more filters
  - However, # of parameters increases as we have more channels.
- Problem #2 : Dead channels
  - Some channels do not affects the outputs at all.
- Problem #3 : Low correlation btw. Channels

Ⅳ. Compact Network Design (1)

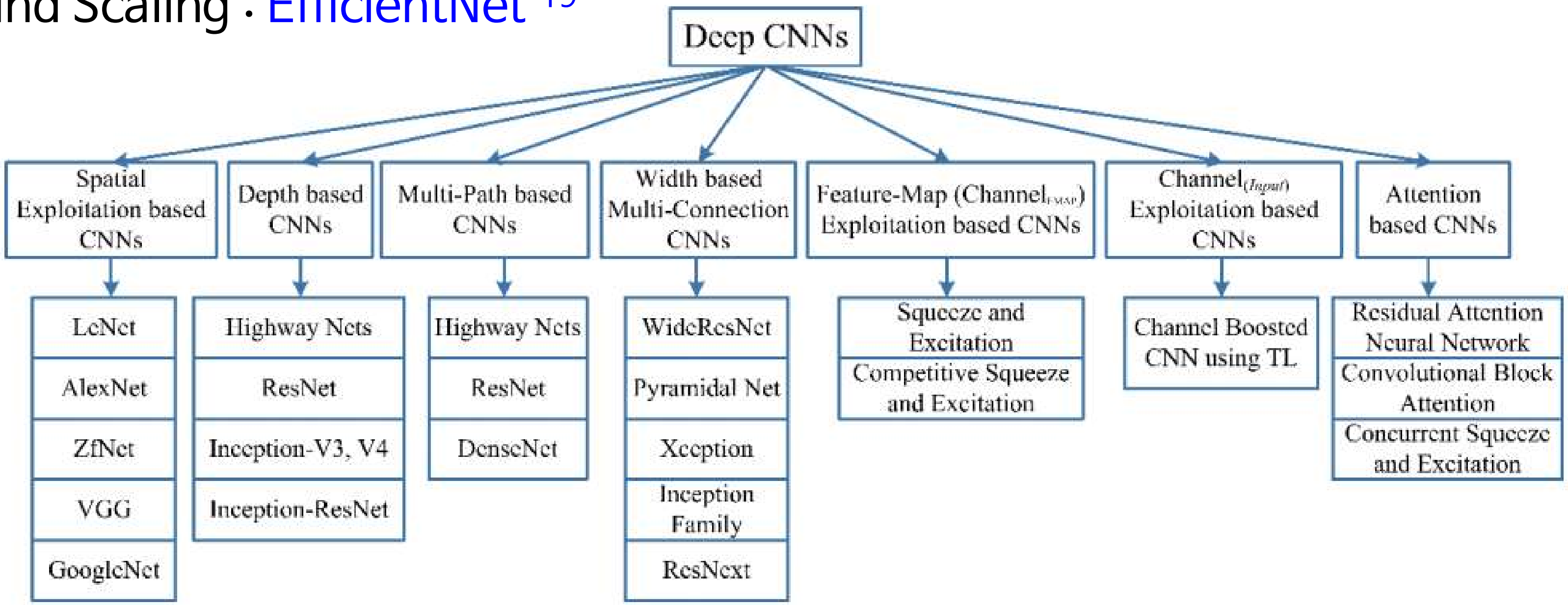
- Standard Convolution
  - # of Parameters
    - K : filter size, C: $\#$of Input channels, M : # of output channels
    - One filter size : K^2C
    - Total # of parameters : K^2CM
  - Computational cost
    - Input size : $H \times W$, output size : $H \times W$
    - K^2CMHW



- CNN 특징 : Full connectivity가 아닌 local connectivity로 계산량 감소
- 최적화 주요 요소 : Activation, Pooling, Architecture, Regularization, Optimization 등

Ⅳ. Compact Network Design (2)

- Design optimized neural network architecture
  - Grouped Convolution : AlexNet'12, ShuffleNet'17
  - Residual connection, Bottleneck : Inception'15,' ResNet'16
  - Depth-wise Conv., Point-wise conv.: Inception'15, MobileNet'17
  - Dilated Convolution: DeepLab'18, ShiftNet'17~'19
  - Shift Convolution: ShiftNet'17~ '19
  - Compound Scaling : EfficientNet'19

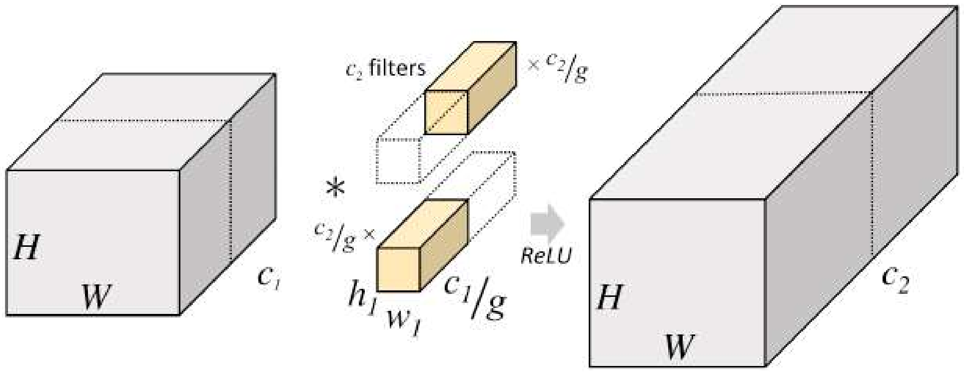


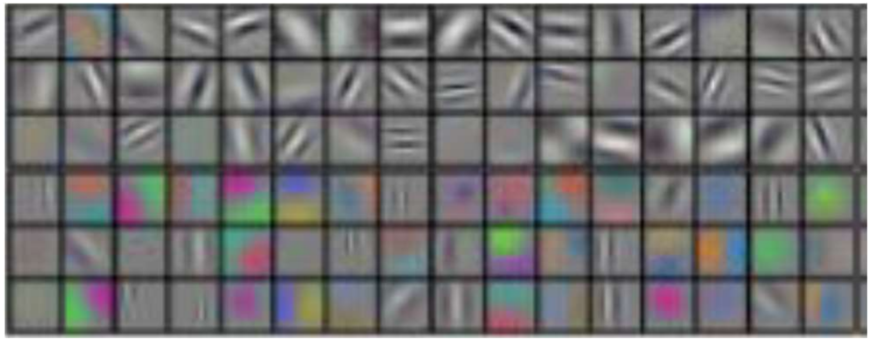
* Khan, Asifullah, et al. "A survey of the recent architectures of deep convolutional neural networks." *Artificial Intelligence Review* (2020): 1-62.

Grouped Convolution

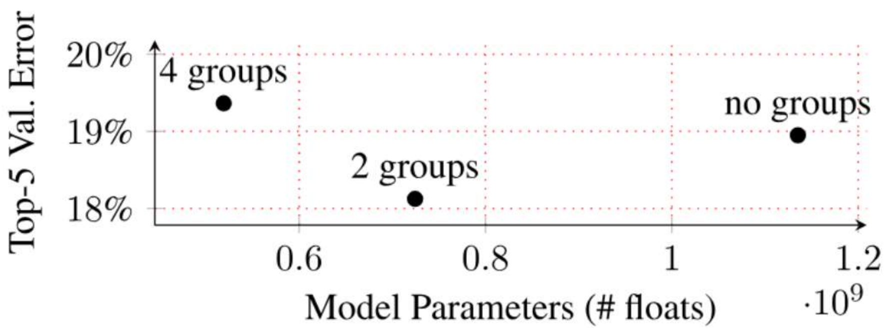
[AlexNet, ShuffleNet]

- Group filters to separate channel information and learn (benefit to parallel processing)
- Sparse by learning highly correlated information for each filter group (fewer parameters)
- # of Parameters
  - $(K^2 CM)/g$; g: # of groups
- Computational cost
  - $(K^2 CMHW)/g$



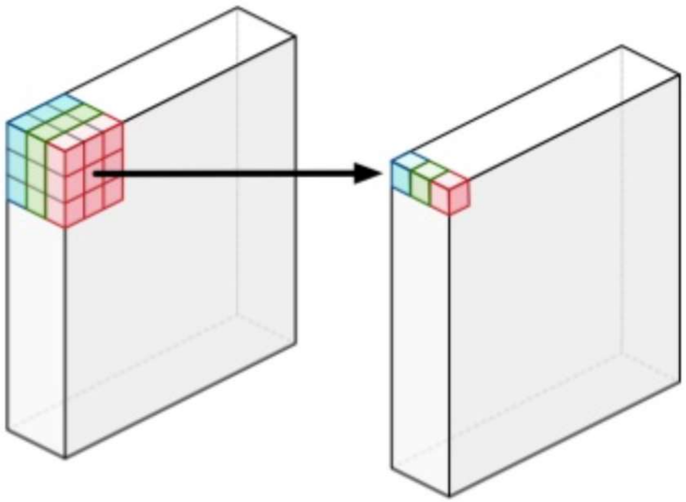


- (Top) Shading-originated kernel learning group
- (Bottom) Kernel learning group focused on colors & patterns



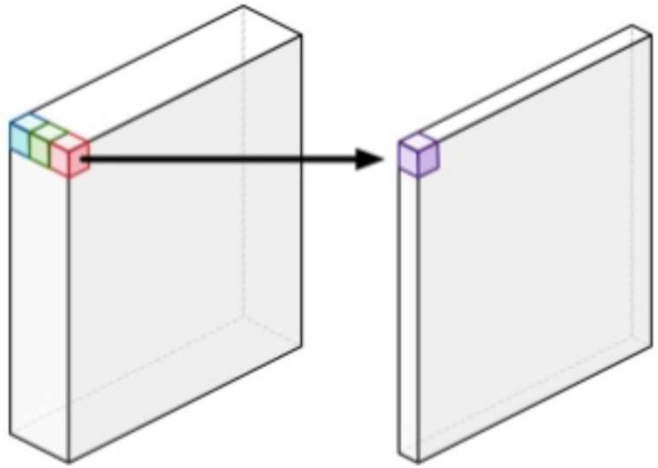
Depthwise Convolution

- Channel-independent convolution
  - Standard conv. Is impossible to spatial features for each channel
  - Each filter works for each channel
- # of parameters
  - K^2C
- Computational cost
  - K^2CMW



Point-wise convolution

- Effects of dimensional reduction by reducing # of channels
- Use of 1x1 convolutional filter

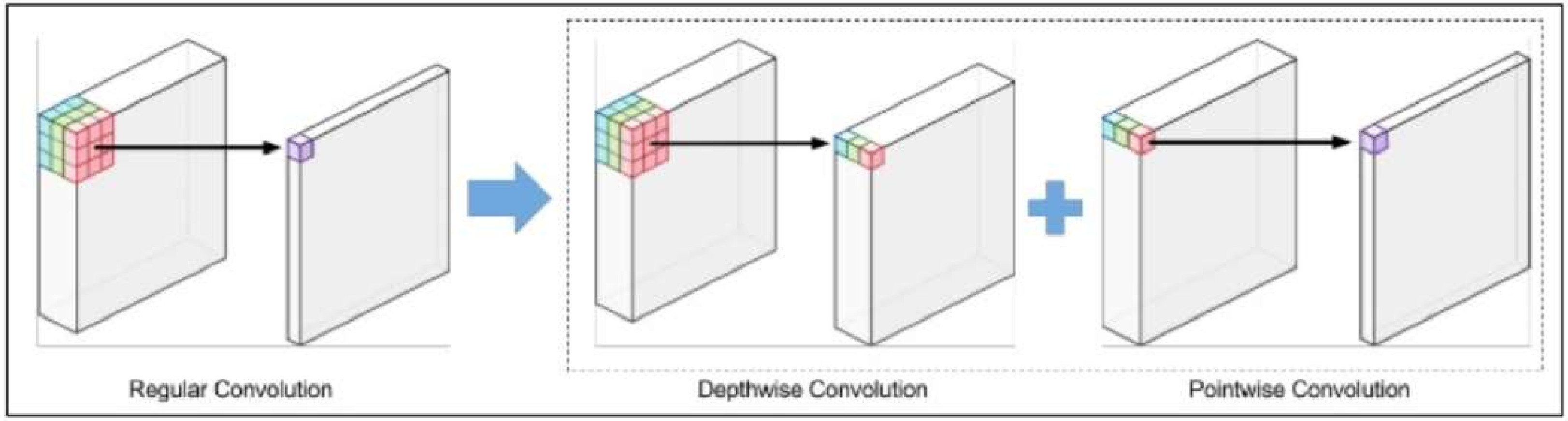


- # of parameters
  - CM since $K = 1$
- Computational cost
  - $CMHW$ since $K = 1$

DepthWise Separable PointWise Conv.

- Xception*,MobileNet**
  - CNN 계산량 및 매개변수 수를 줄임으로써 Neural Network 경량화
  - C 채널수, (X,Y) 입력 영상 크기, K 커널 필터 크기, M 필터 출력 채널 수

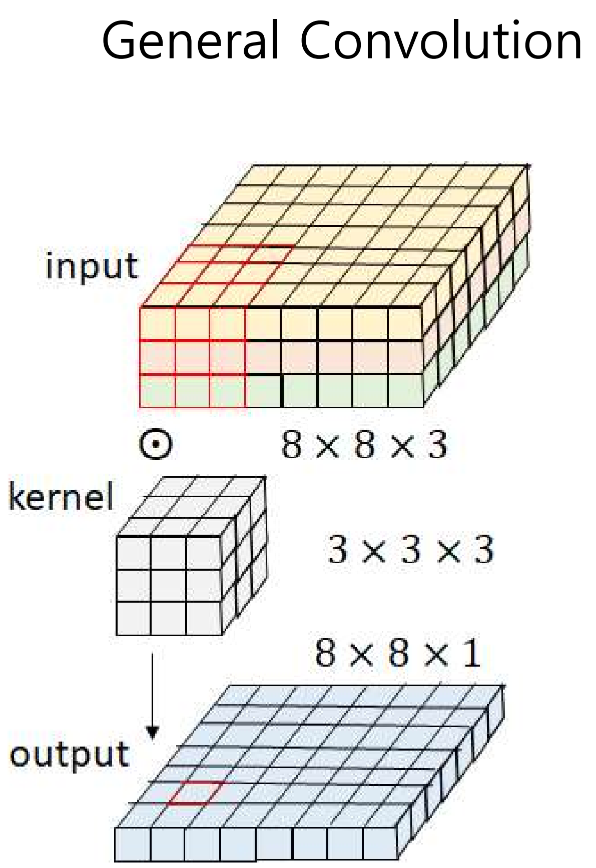
	Spatial Convolution	Depthwise Separable Conv.
operation 수	MCK^2	$C(K^2+M)$
Parameter수	$CHWK^2M$	$CHWK^2+CHW\mathrm{M}$ $=CHW(K^2+M)$

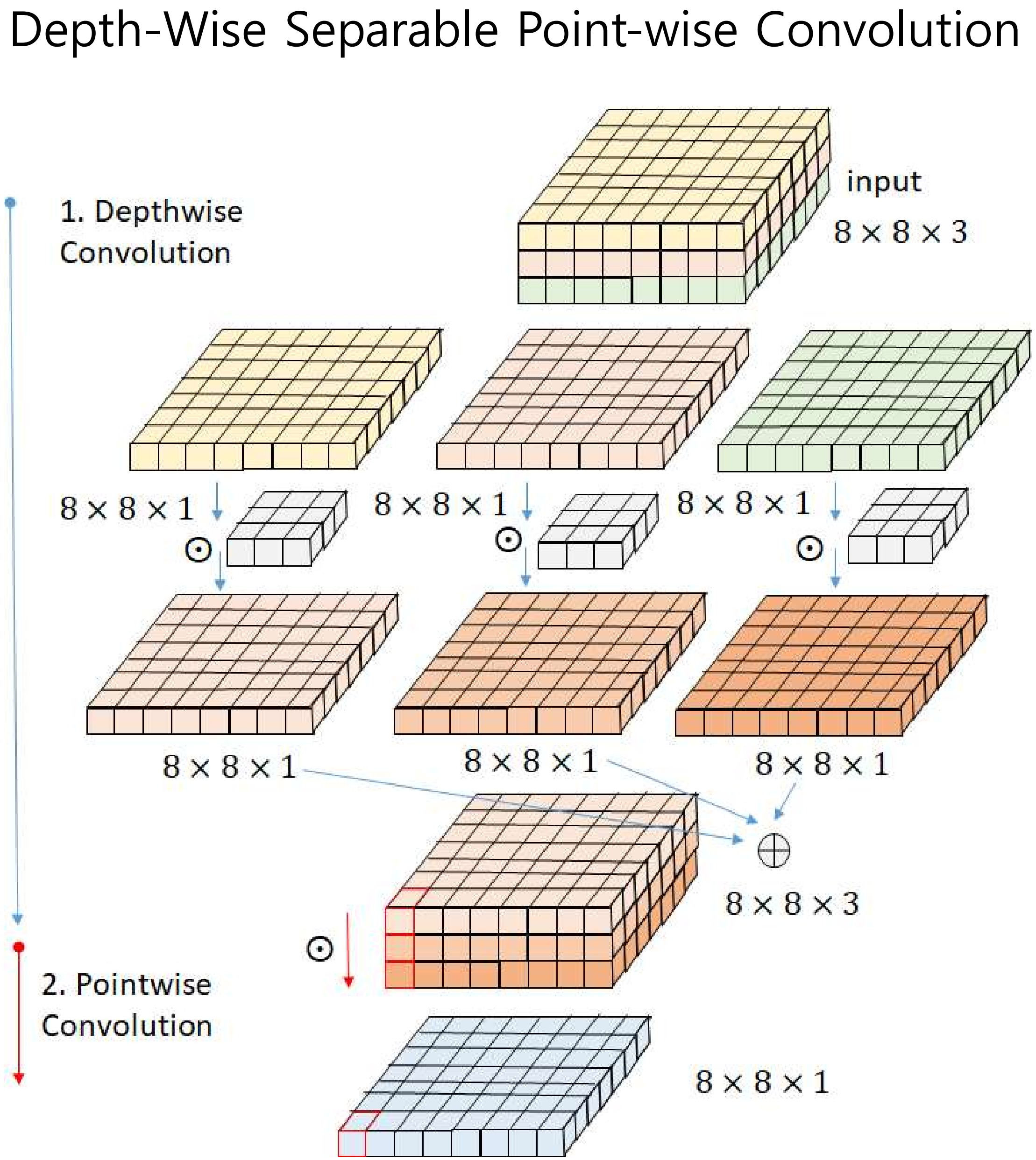


* Francois Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions," CVPR 2017
** Andrew G. Howard et al., "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," CVPR 2017

Reinterpretation of Convolutions

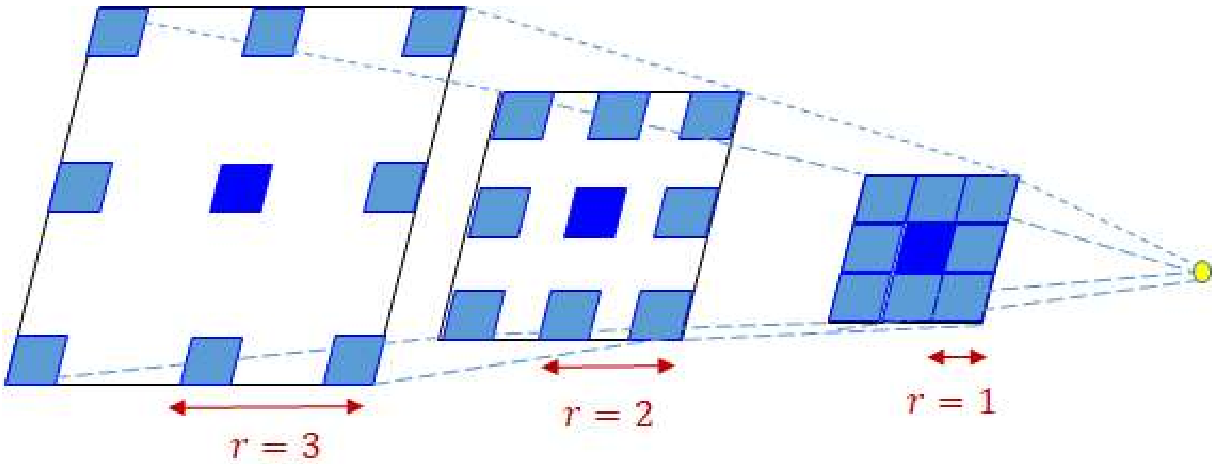
[MobileNet, Inception]



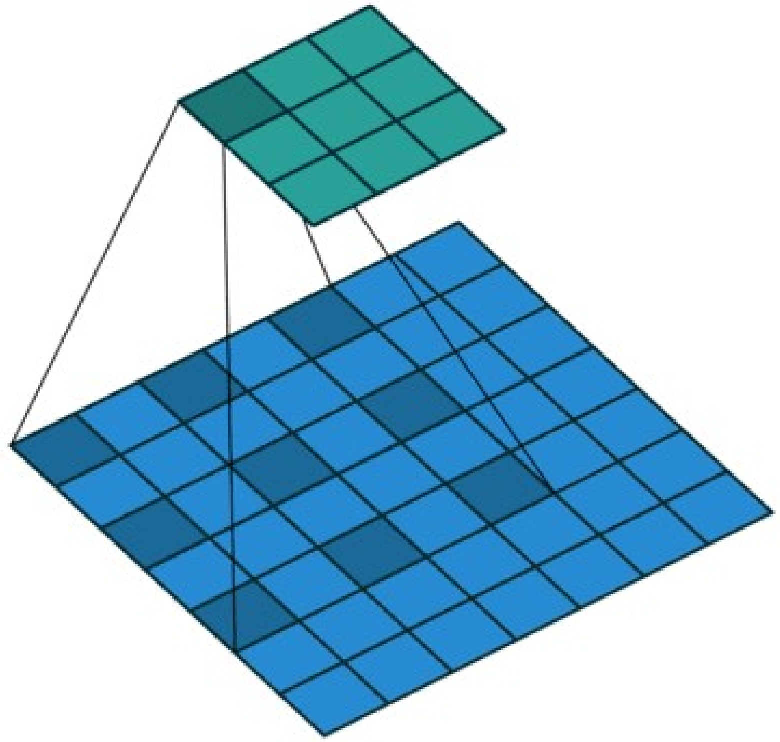


Dilated Convolution

- **Overview** [DeepLab, ShiftNet]
  - Receptive filed area is widened, but by maintaining the kernel size
  - \# of convolution operations is maintained while obtaining a pooling effect to improve accuracy



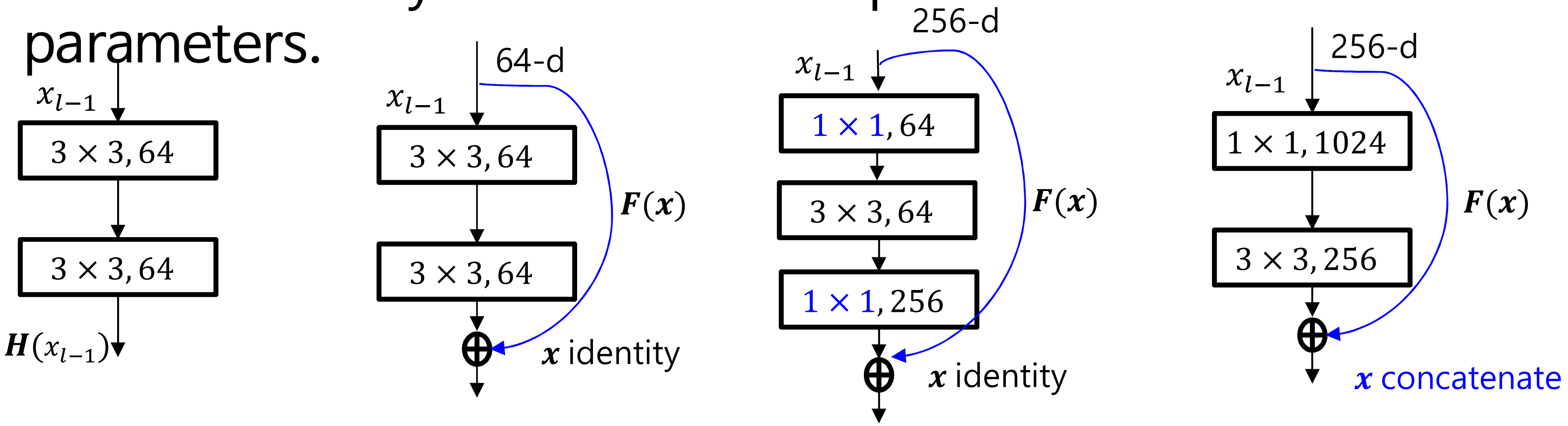
r : dilation rate, the spacing between kernels



Residual Connection, Bottleneck, & DenseNet

[ResNet, DenseNet]

- **Residual Connection**
  - With the same number of parameters, there is little additional computation other than addition, so deep networks are well optimized.
- **Dense Net**
  - While collecting feature maps as concatenation, compression at the transition layer ensures better performance with fewer parameters.



$x_l = H(x_{l-1})$

(a)plain net

$x_l = F_l(x_{l-1}) + x_{l-1}$

(b)residual block

$x_l = F_l(x_{l-1}) + x_{l-1}$

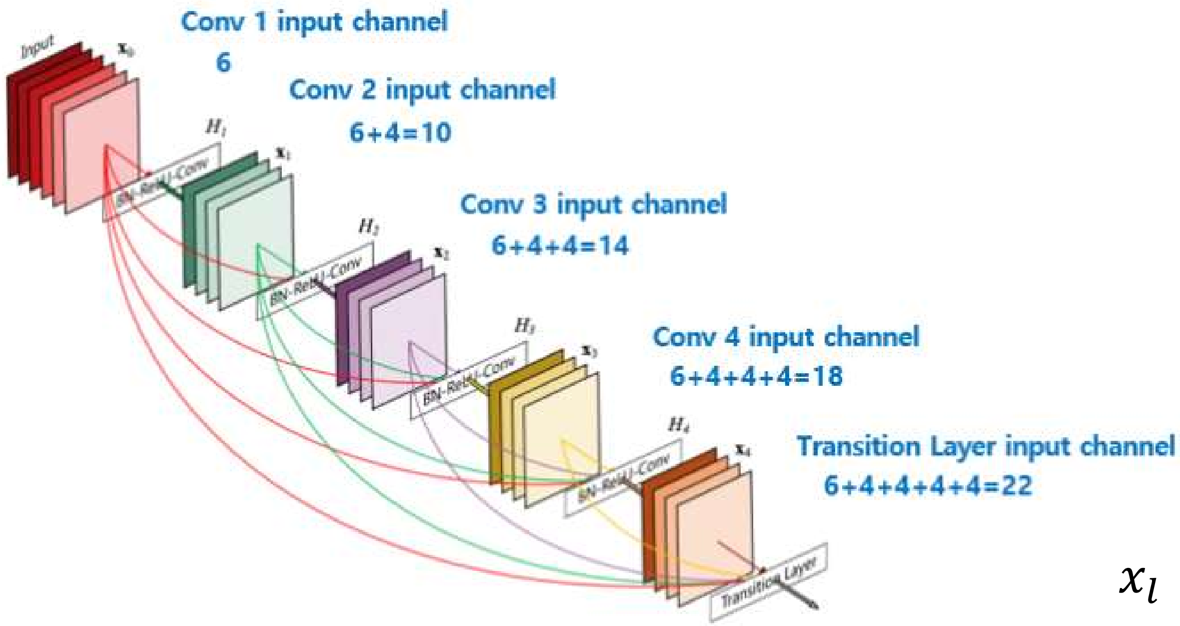
(c)bottleneck ResNet

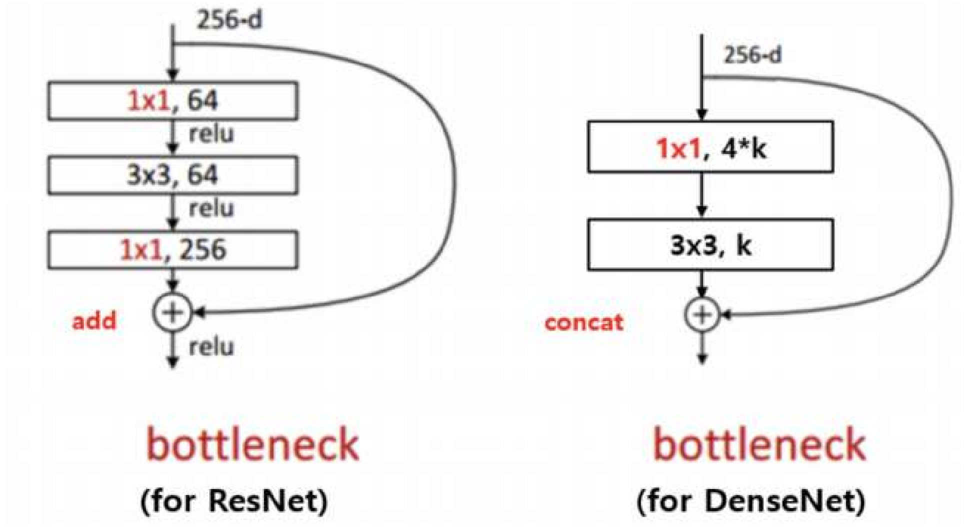
$x_l = F_l([x_0 x_1, \dots, x_{l-1}])$

(d) DenseNet

Dense Convolution

- Dense Convolution (DenseNet*)
  - 기존의 feature map을 additive형태가 아닌 concat 형태로 취합
  - feature-maps 압축으로 적은 parameter로 더 나은 성능 확인





$$x_l = H_l(x_{l-1}) + x_{l-1} \qquad x_l = H_l([x_0, x_1, \dots, x_{l-1}])$$

* G. Huang et al., "Densely Connected Convolutional Networks," CVPR 2017

Shift Convolution

- Shift (ShiftNet*, **)
  - Spatial convolution을 shift operation으로 대체하여 연산을 줄임

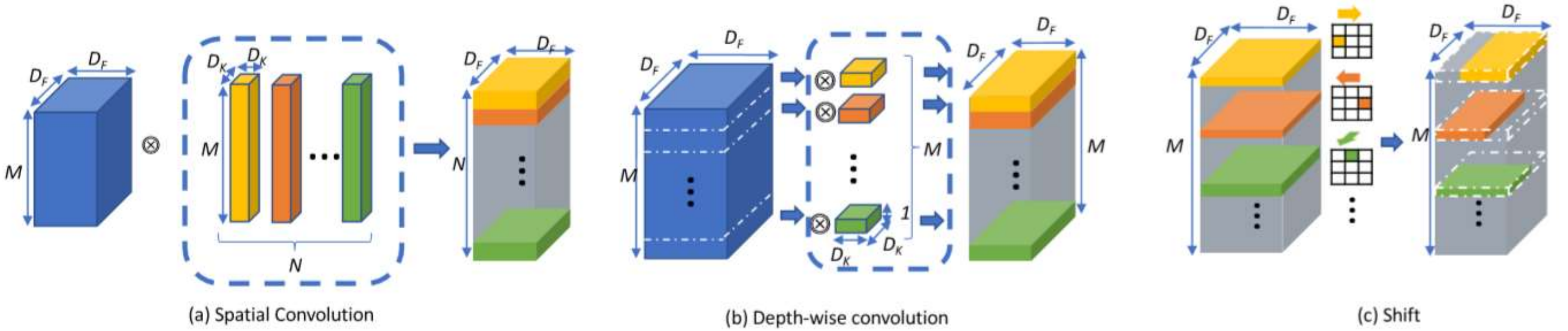


Figure 2: Illustration of (a) spatial convolutions, (b) depth-wise convolutions and (c) shift. In (c), the 3x3 grids denote a shift matrix with a kernel size of 3. The lighted cell denotes a 1 at that position and white cells denote 0s.

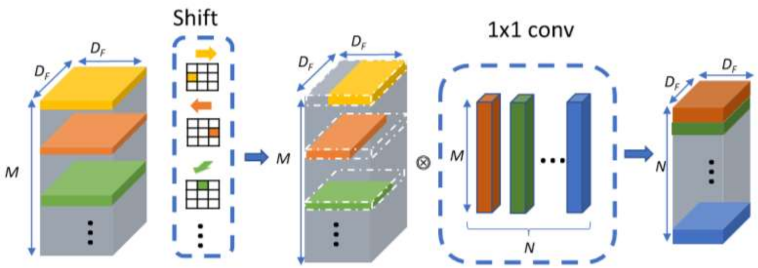
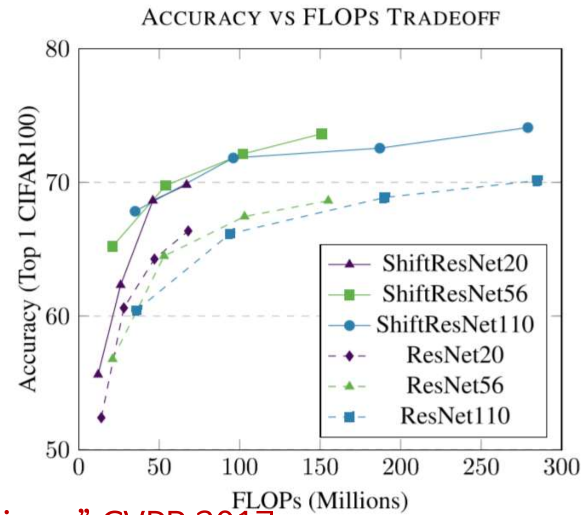


Figure 1: Illustration of a shift operation followed by a 1x1 convolution. The shift operation adjusts data spatially and the 1x1 convolution mixes information across channels.

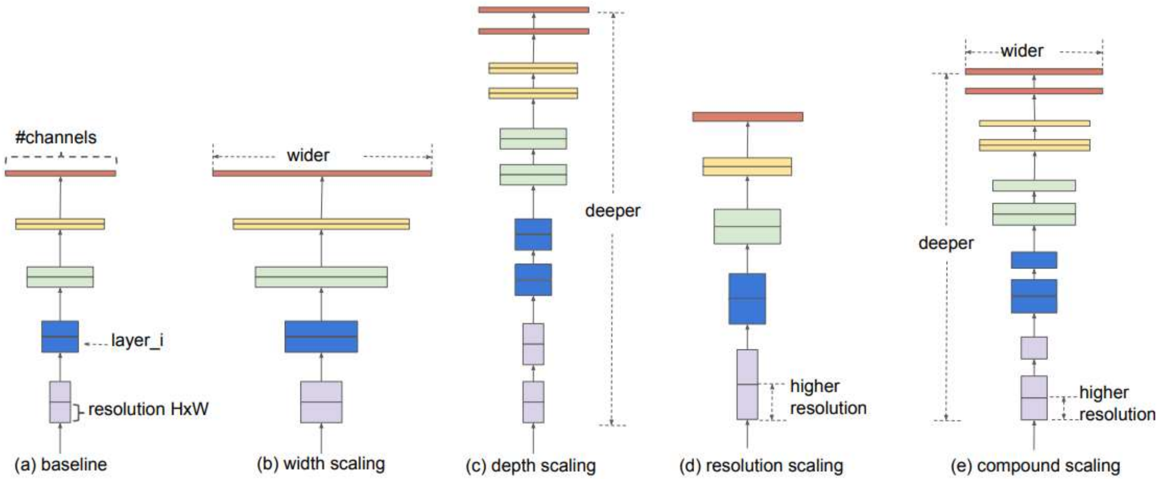


* B. Wu et al., “Shift: A zero FLOP, Zero Parameter Alternative to Spatial Convolutions,” CVPR 2017

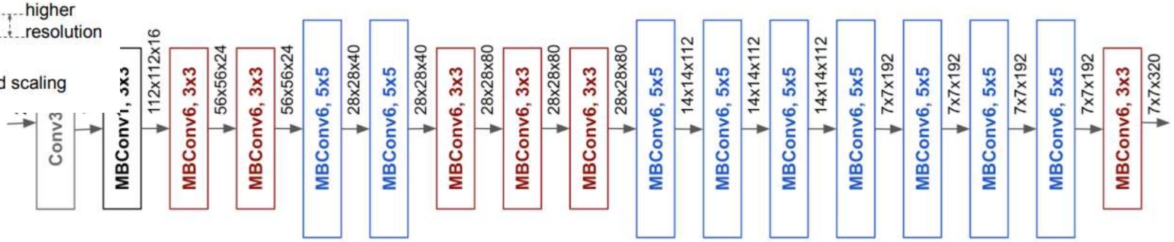
** W. Chen et al., “All You Need is a Few Shifts: Designing Efficient Convolutional Neural Networks for Image Classification,” CVPR 2019

Compound Scaling

- Compound Scaling (EfficientNet*)
  - CNN model의 3요소 (Depth α, Width β, Resolution γ)의 Efficient한 Scaling 방법에 대한 연구
  - mNasNet**의 결과 모델을 base model로 찾음
  - 최종적으로 $\alpha = 1.2,\ \beta = 1.1,\ \gamma = 1.15$ 를 고정한 채, ϕ를 늘려 $B_1 \sim B_7$모델 탐색



Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	28×28	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1



* Mingxing Tan, Quoc V. Le, "Efficient Net: Rethinking Model Scaling for Convolutional Neural Networks," ICML 2019
** Mingxing Tan et. al., "MnasNet: Platform-Aware Neural Architecture Search for Mobile," CVPR 2018

- Compound Scaling (EfficientNet)
  - 타모델 대비 Efficient Net은 parameter수를 8~9배 줄이면서 정확도 향상
  - Inference 속도 성능은 6배 정도 향상

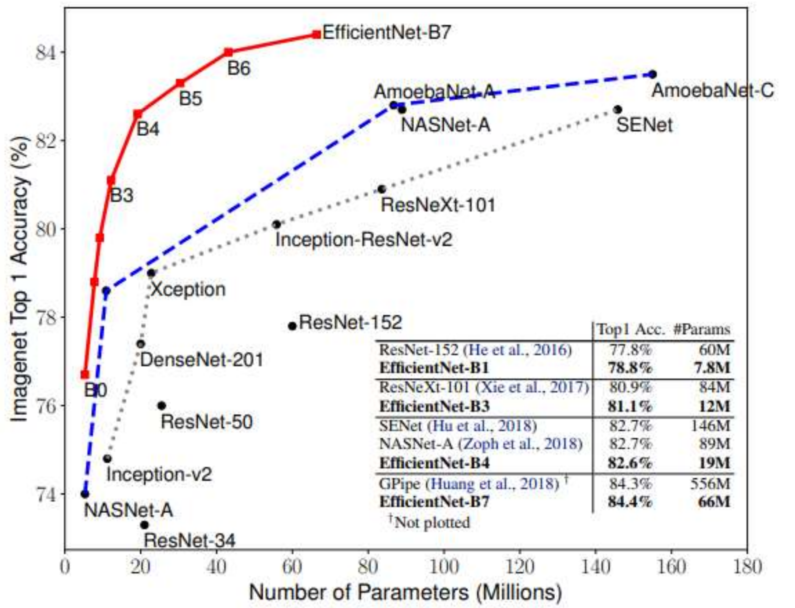


Figure 1. **Model Size vs. ImageNet Accuracy.** All numbers are

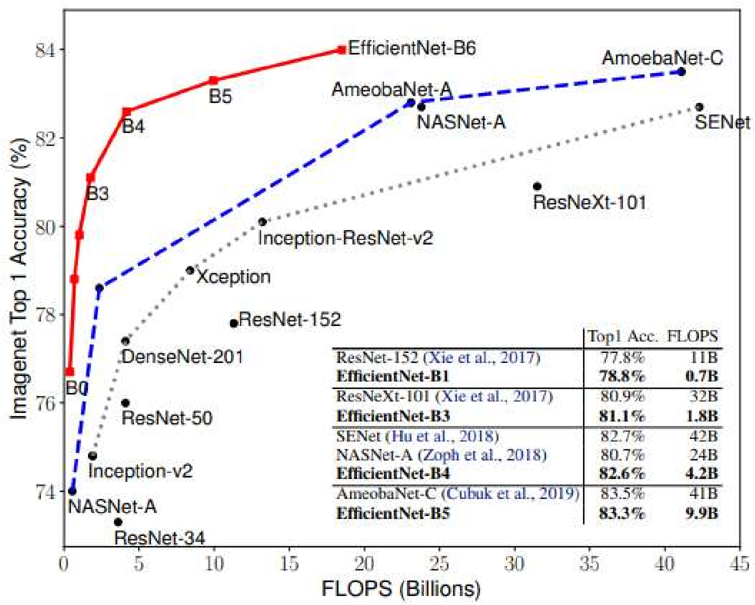


Figure 5. **FLOPS vs. ImageNet Accuracy** – Similar to Figure 1

Table 4. **Inference Latency Comparison**

	Acc. @ Latency
ResNet-152	77.8% @ 0.554s
EfficientNet-B1	78.8% @ 0.098s
Speedup	**5.7x**

	Acc. @ Latency
GPipe	84.3% @ 19.0s
EfficientNet-B7	84.4% @ 3.1s
Speedup	**6.1x**

* Mingxing Tan, Quoc V. Le, “Efficient Net: Rethinking Model Scaling for Convolutional Neural Networks,” ICML 2019