



# *Lightweighting DL Models*

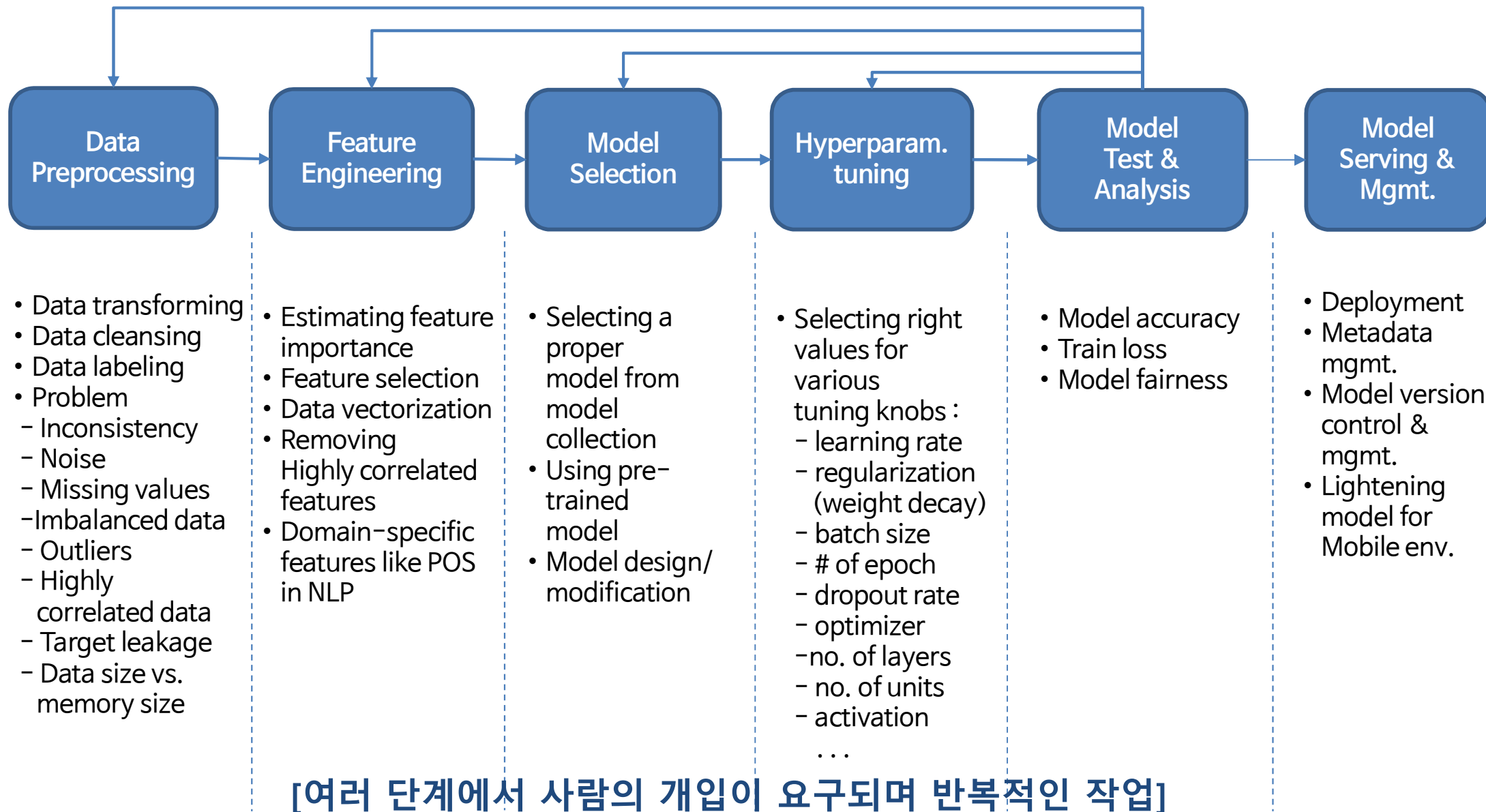
June 04-11, 2021  
Kyong-Ha Lee  
[bart7449@gmail.com](mailto:bart7449@gmail.com)

# 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





# Challenge 3: Time & Costs

## 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

## Common carbon footprint benchmarks

in lbs of CO2 equivalent

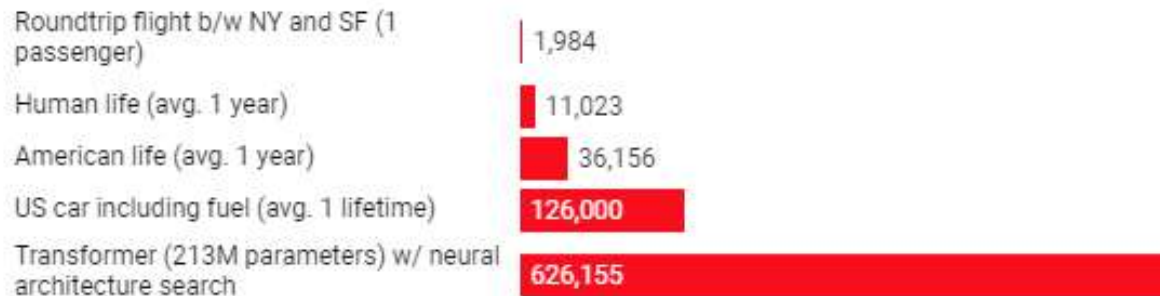


Chart: MIT Technology Review • Source: Strubell et al. • Created with Datawrapper

## 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



- **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: 개요

- 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

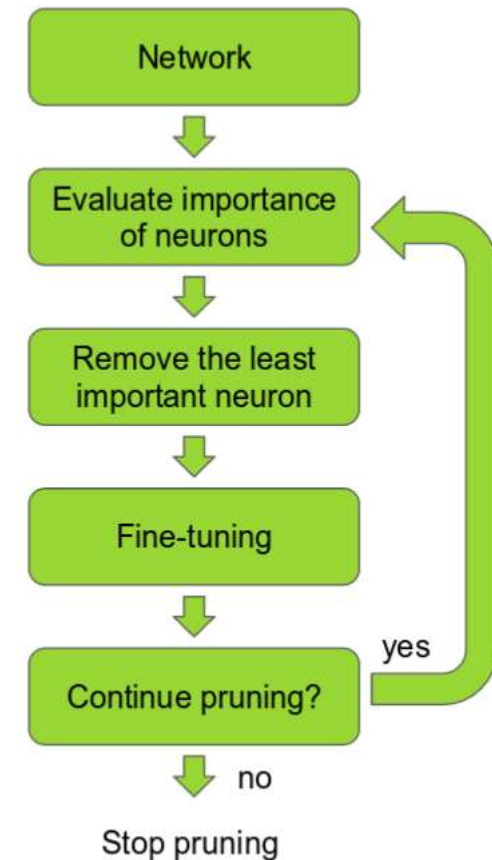
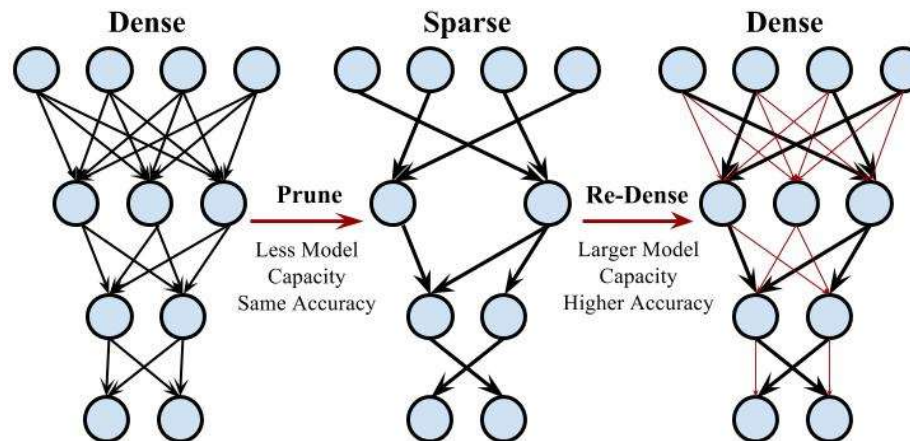
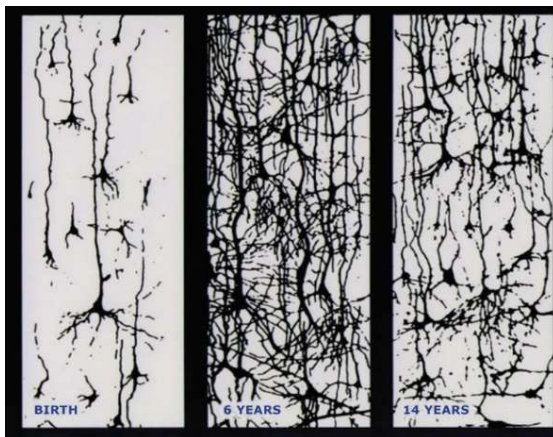
Approaches	Technique	Approach
<b>Pruning</b>	• Remove weights or neurons under threshold	• From pre-trained model
<b>Weight sharing</b>	• Cluster weights and encode centroid of the weights	• From pre-trained model
<b>Quantization</b>	• Substitute weight, activations or gradients with lower bit-widths	• From pre-trained model • From scratch
<b>Distillation</b>	• 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
<b>Compact Network Design</b>	• 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



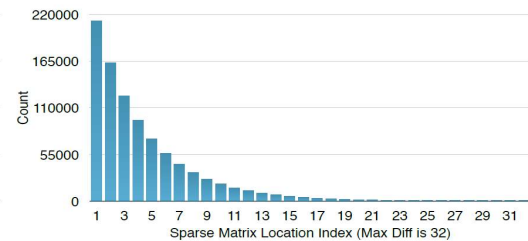
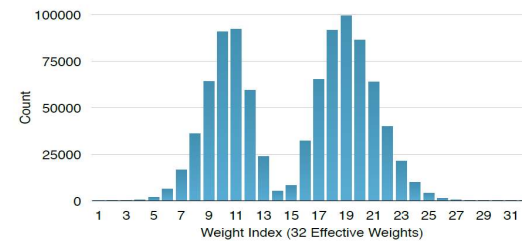
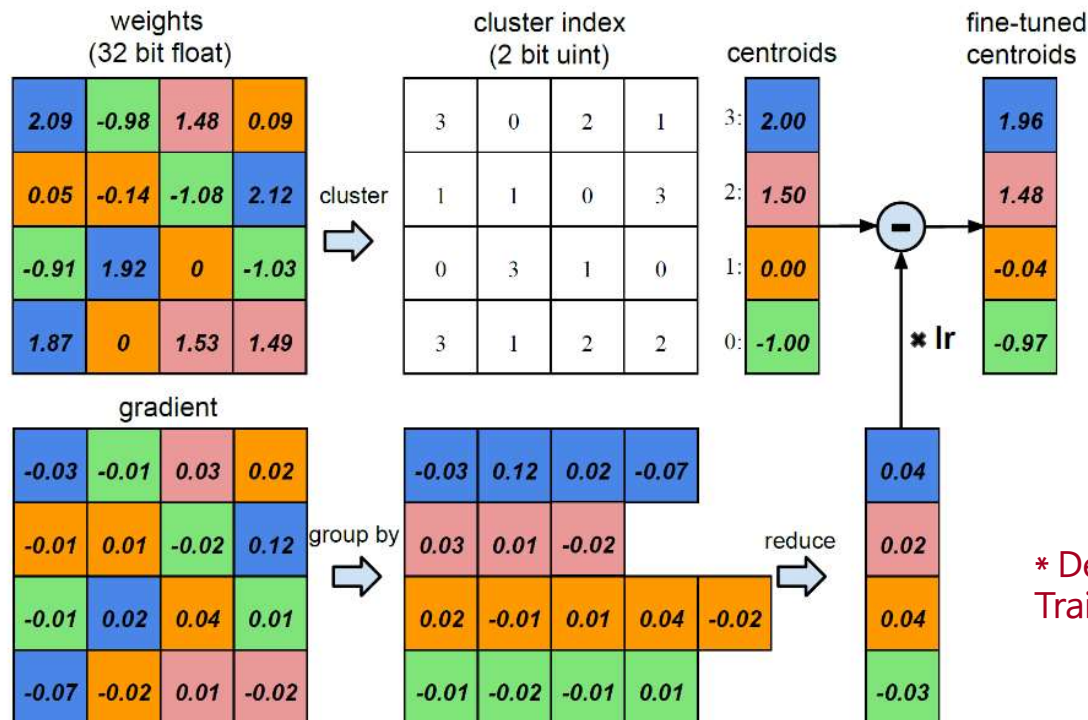
[기존 모델의 가공 및 재학습을 통한 모델 경량화]

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

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

## 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



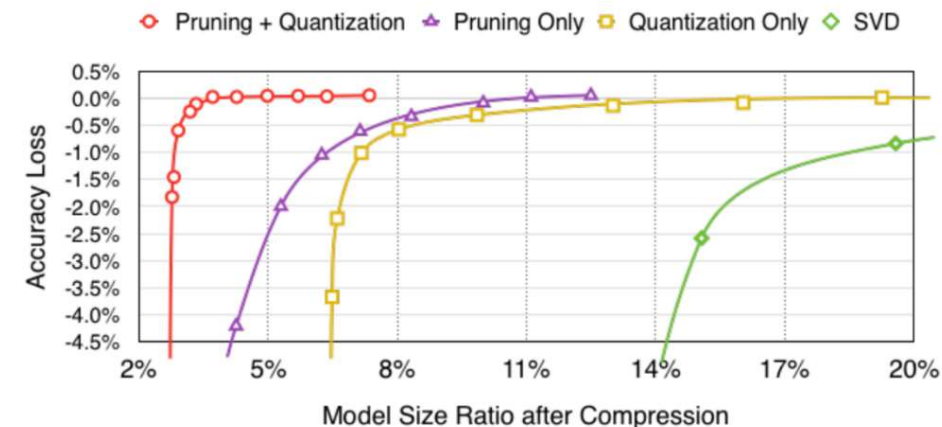
Biased weights

\* 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

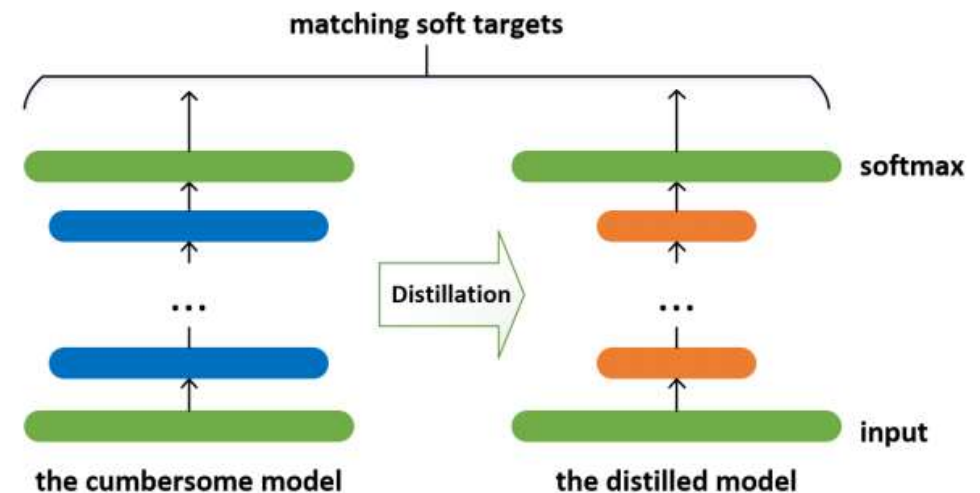
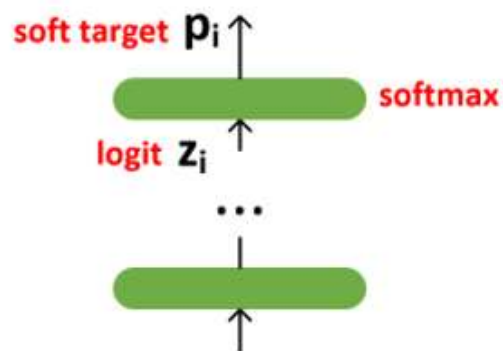
Network	Top-1 Error	Top-5 Error	Parameters	Compress Rate
LeNet-300-100 Ref	1.64%	-	1070 KB	
LeNet-300-100 Compressed	1.58%	-	<b>27 KB</b>	40×
LeNet-5 Ref	0.80%	-	1720 KB	
LeNet-5 Compressed	0.74%	-	<b>44 KB</b>	39×
AlexNet Ref	42.78%	19.73%	240 MB	
AlexNet Compressed	42.78%	19.70%	<b>6.9 MB</b>	35×
VGG-16 Ref	31.50%	11.32%	552 MB	
VGG-16 Compressed	31.17%	10.91%	<b>11.3 MB</b>	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  $T < T_1$

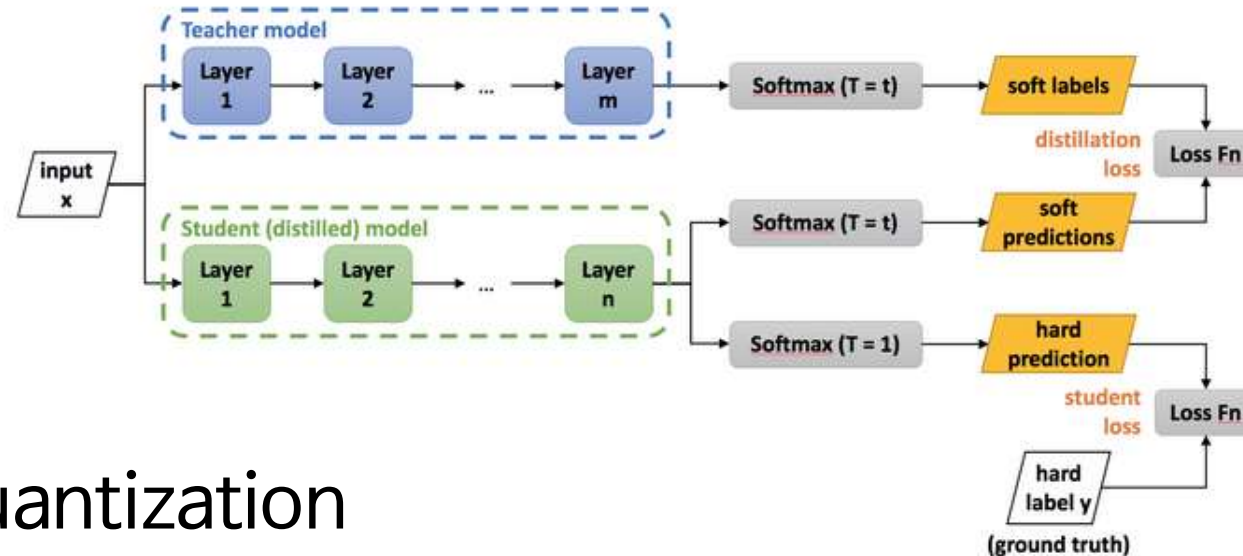




## II. 모델 경량화: Distillation (2)

- Knowledge Distillation

[Kill-The-Bits'20]

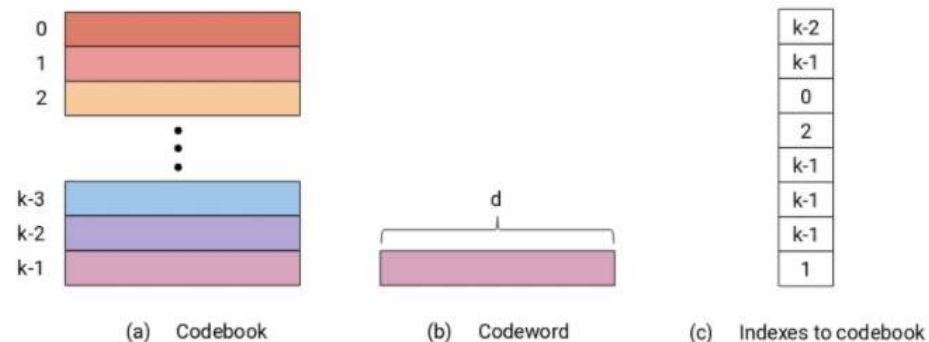


- Product Quantization

- 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



## II. 모델 경량화: Distillation (2)

CIFAR10 accuracy

		2 bits	4 bits	8 bits
Student model 1 1M param - 4 MB 84.5% - 88.8% <b>X5.25</b>	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% <b>X16.5</b>	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% <b>X46.66</b>	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]

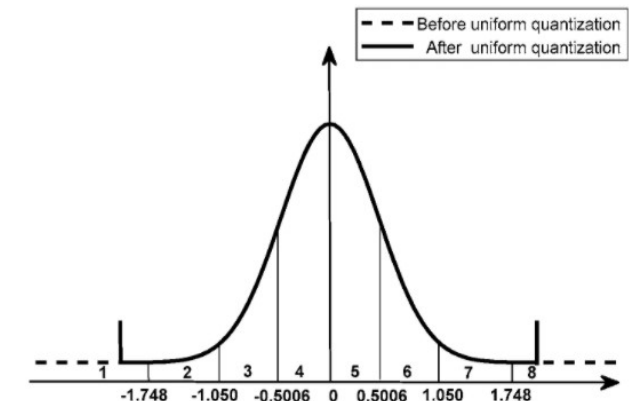
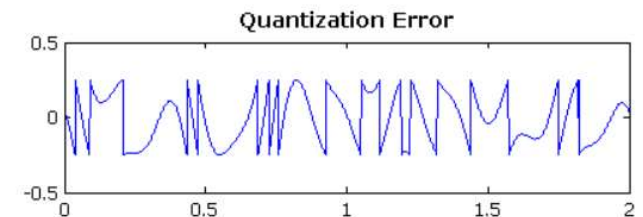
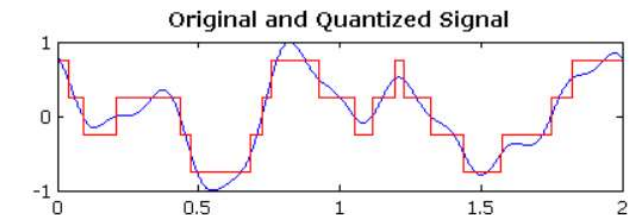
# III. Quantization: Overview

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

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





# 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 KIIE* 38(8):18-29, Aug., 2020

- Codebook

Approaches	Types	codebooks	Representative work
Fixed codebook	<ul style="list-style-type: none"><li>• Binarization</li><li>• Scaled Binarization</li><li>• Ternarization</li><li>• Scaled Ternarization</li><li>• Powers of two</li><li>• K-bits</li></ul>	<ul style="list-style-type: none"><li>• <math>\{-1, 1\}</math></li><li>• <math>\{-a, b\}</math></li><li>• <math>\{-1, 0, 1\}</math></li><li>• <math>\{-a, 0, b\}</math></li><li>• <math>\{0, \pm 1, \pm 2^{-1}, \dots, \pm 2^{-L}\}</math></li><li>• <math>\{\pm v1, \{\pm v1 \pm v2\}, \dots\}</math></li></ul>	<ul style="list-style-type: none"><li>• BinaryConnect, Binary-weight</li><li>• XNOR-Net</li><li>• Ternary Net</li><li>• TTQ</li><li>• [Tang and Kwan'94]</li><li>• DoReFa-Net, LQ-Net</li></ul>
Adaptive codebook	<ul style="list-style-type: none"><li>• Soft Quantization</li><li>• Hard Quantization</li></ul>	<ul style="list-style-type: none"><li>• Learned from data</li><li>• Learned from data</li></ul>	<ul style="list-style-type: none"><li>• Variational Network Quantization</li><li>• Vector quantization, [Choi'2016]</li></ul>

- Targets for quantization

Components	Benefits	Challenges
Weights	<ul style="list-style-type: none"><li>• Smaller model size</li><li>• Faster forward training &amp; inference</li><li>• Less energy</li></ul>	<ul style="list-style-type: none"><li>• Hard to converge with quantized weights</li><li>• Require approximate gradients</li><li>• Accuracy degradation</li></ul>
Activations	<ul style="list-style-type: none"><li>• Smaller memory foot print during training</li><li>• Allows replacement of dot-products by bitwise operations</li><li>• Less energy</li></ul>	<ul style="list-style-type: none"><li>• Gradient mismatch problem</li></ul>
Gradients	<ul style="list-style-type: none"><li>• Communication &amp; memory savings</li></ul>	<ul style="list-style-type: none"><li>• Convergence requirement</li></ul>

# III. Quantization : Binarization (1)

- 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

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

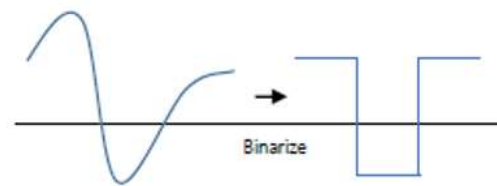
- Stochastic

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

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  $[0, 1]$ .

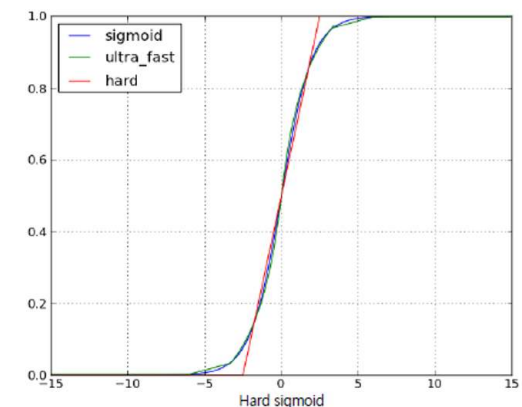
- Propagations & Updates

- $w$  is binarized during forward and backward propagation
- Full-precision  $w$  is used during parameter update



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

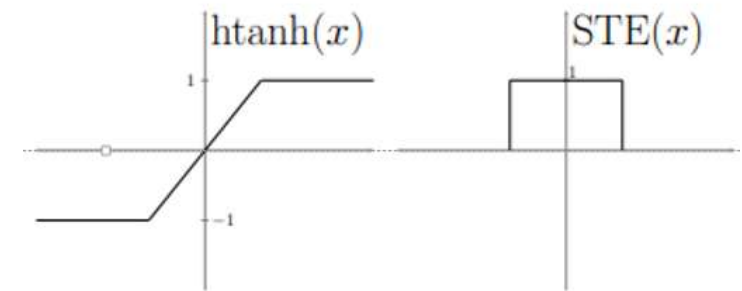
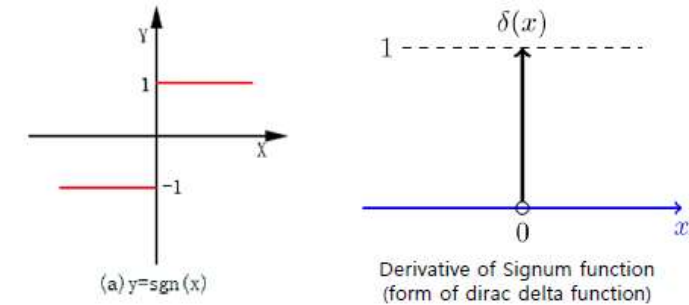
Binarization example



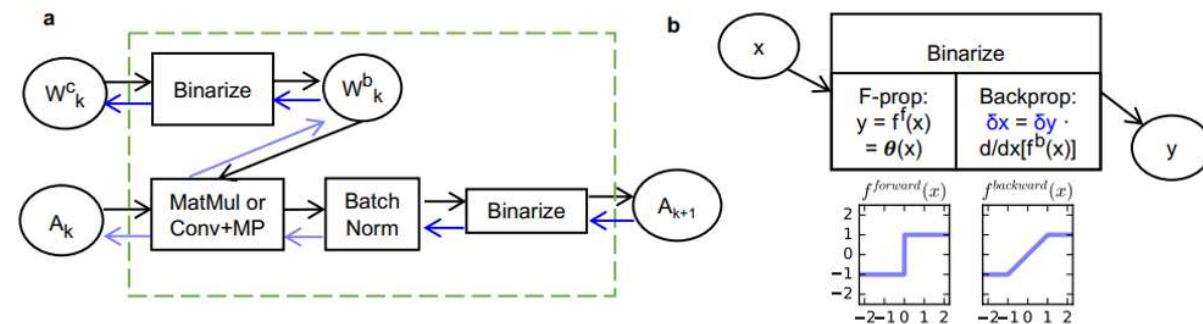
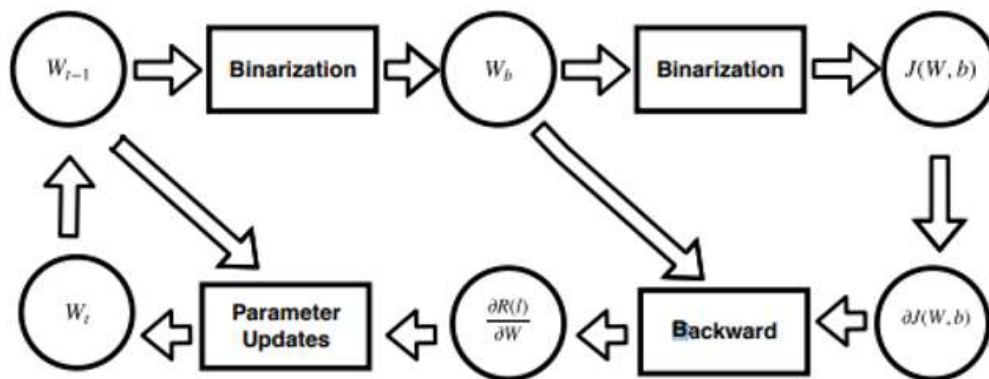
\* Binarized Neural Networks: Training deep neural networks with weights and activations constrained to +1 or -1, [Courbariaux et al., 2016]

# III. Quantization : Binarization (2)

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



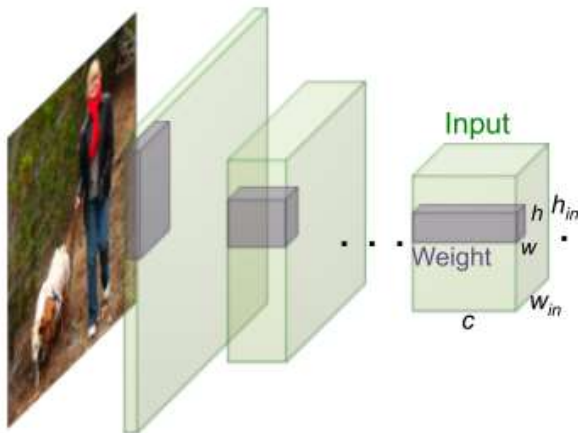
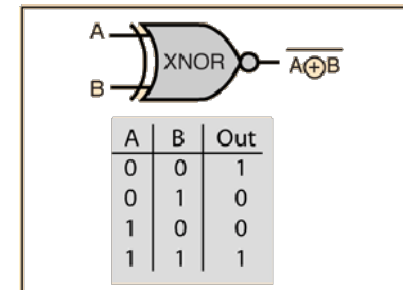
## • Procedure



Binary convolution transformer(dashed green line)

# III. Quantization : XNOR-NET (1)

- 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)

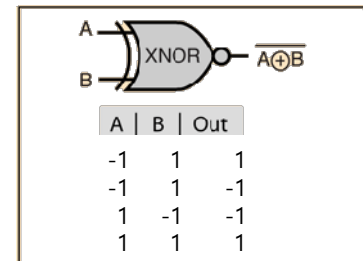


	Network Variations	Operations used in Convolution	Memory Saving (Inference)	Computation Saving (Inference)	Accuracy on ImageNet (AlexNet)
Standard Convolution	Real-Value Inputs Real-Value Weights 	$+, -, \times$	1x	1x	%56.7
Binary Weight	Real-Value Inputs Binary Weights 	$+, -$	$\sim 32x$	$\sim 2x$	%56.8
BinaryWeight Binary Input (XNOR-Net)	Binary Inputs Binary Weights 	XNOR , bitcount	$\sim 32x$	$\sim 58x$	%44.2

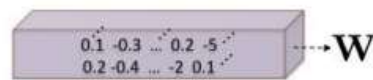
# III. Quantization : XNOR-NET (2)

$$\mathbf{C}^* = \text{sign}(\mathbf{Y}) = \text{sign}(\mathbf{X}^T) \text{sign}(\mathbf{W}) = \mathbf{H}^{*\top} \mathbf{B}^*$$

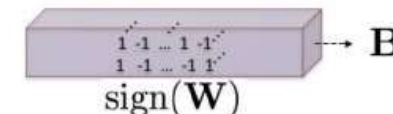
- Binarize both weights and inputs (previous results)
- Convolution as **Binary dot product** implemented by XNOR-BitCounting operations



(1) Binarizing Weight



$$\frac{1}{n} \|\mathbf{W}\|_{\ell_1} = \alpha$$



(2) Binarizing Input

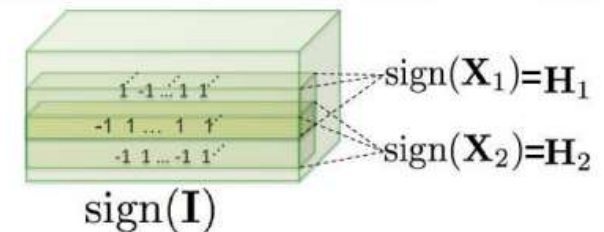
Inefficient



Redundant computations in overlapping areas

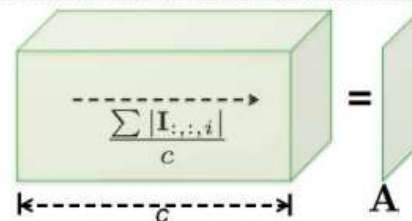
$$\frac{1}{n} \|\mathbf{X}_1\|_{\ell_1} = \beta_1$$

$$\frac{1}{n} \|\mathbf{X}_2\|_{\ell_1} = \beta_2$$

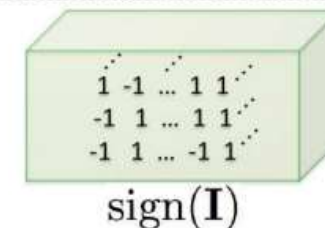


(3) Binarizing Input

Efficient



$$\mathbf{A} * \mathbf{k} = \mathbf{K}$$



(4) Convolution with XNOR-Bitcount

$$\mathbf{I} * \mathbf{W} \approx \left[ \mathbf{sign}(\mathbf{I}) * \mathbf{sign}(\mathbf{W}) \right] \odot \mathbf{K} \odot \alpha$$

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



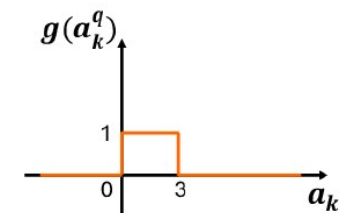
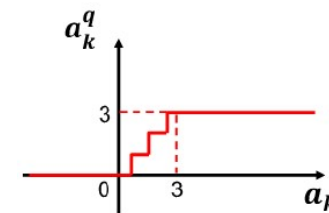
### III. Quantization : DoReFa-Net

- 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<sub>k</sub>** 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:

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

$$\textbf{Backward: } \frac{\partial c}{\partial r_i} = \frac{\partial c}{\partial r_o}. \quad (6)$$



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

# III. Quantization: 실험

## • 개발 환경

### – [계산과학플랫폼 지원]

- Edison Server 4대 (Tesla P100 7개)  
(P100 ~ 5.3 TFLOPS, FP16 지원)
- Intel® Xeon CPU E4-2640 core 32개,  
RAM 99GB (Server 각각  
GPU(1,2,2,2))



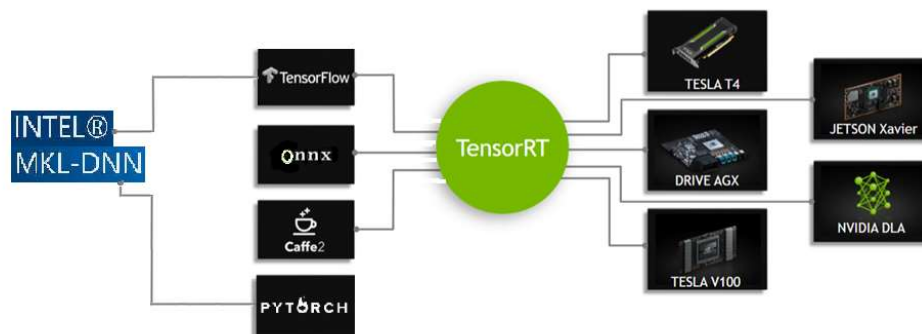
### – [개발 GPU Server]

- RTX 2080 TI GPU 2대
- (RTX 2080 TI ~ 0.42 TFLOPS)
- Intel i9 CPU core 36개, 64GB



### – Low-bitwidth 지원 플랫폼

- Specified precision FP32, FP16,  
Int8bit 연산 지원 GPU
- CUDA 10.1, 10.0, 9.0, CUDNN 7.6.3



## • 실험 데이터 IMAGENET

### – ImageNet data

### – Training dataset

- 1,000-class classification
- 전체 training image 개수  
1,281,273, total 140GB (JPEG)
- Avg. image resolution: 400 x 350
- 1,000개의 class 별 약 10,000장  
image로 구성

### – Testing dataset

- 전체 validation image 개수  
50,000장
- 1,000개 클래스 별 약 500장  
image



# III. Quantization : 실험

Bitwidth /method	model	Thruput. (Training)**	Thruput. (Test)**	Precision@1	Precision@5	Footprint***
FP (32)	AlexNet	1.26 it/s	1.38 it/s	50.19%	74.38%	488Mb
FxP (16) w/ LS*	AlexNet	1.13 it/s	1.38 it/s	51.41%	74.96%	366Mb
Fxp (16) wo/ LS*	AlexNet	1.17 it/s	1.59 it/s	41.26%	65.78%	366Mb
XNOR-Net (1,1,32)	AlexNet	0.45 it/s	0.47 it/s	47.63%	69.46%	(x2.2 압축)
DoReFa-Net(1,2,6)	AlexNet	0.46 it/s	1.04 it/s	52.41%	72.18%	-
FP (32)	ResNet50	1.58 it/s	1.55 it/s	76.35%	93.09%	204Mb
FxP (16) w/ LS	ResNet50	1.30 it/s	1.23 it/s	76.19%	93.00%	204Mb
FxP (16) wo/LS	ResNet50	1.24 it/s	1.35 it/s	48.0%	72.0%	153Mb

- nVidia제공 Apex library의 FP32 연산에 비해 16bit MP연산이 Alex(ResNet) 모델 기준 memory footprint, accuracy, 속도 측면에서 성능 향상 확인
- XNOR-Net (W, A, G) = (1, 1, 32) 알고리즘 Memory footprint는 x2.2압축 및 3배 정도의 처리 속도 향상 확인. Full Precision 대비 6% accuracy 감소
- DoReFa-Net(W, A, G) = (1, 2, 6) 알고리즘은 Memory footprint 변화는 없으나, 3 배 정도의 처리 속도와 Full Precision 알고리즘과 비슷한 accuracy확인

\* LS은 Loss에 대한 Scale

\*\* Train/Test Time은 256 batch (244\*244\*256 px) 처리 속도 sec단위 (minibatch별 처리 속도 )

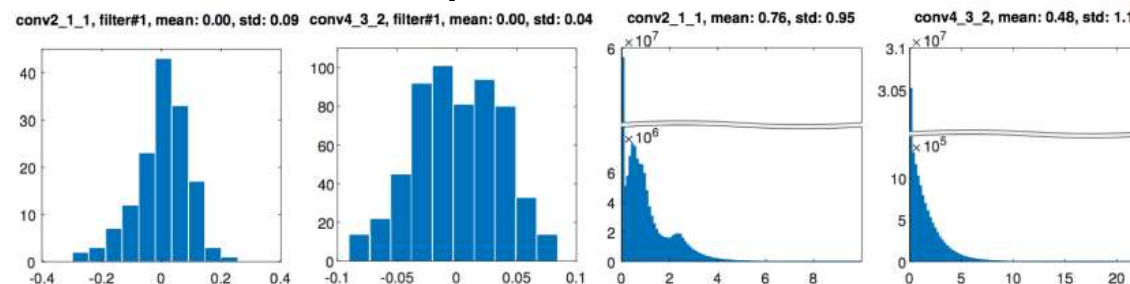
\*\*\* Model footprint는 checkpoint data의 크기를 기준으로 함

[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

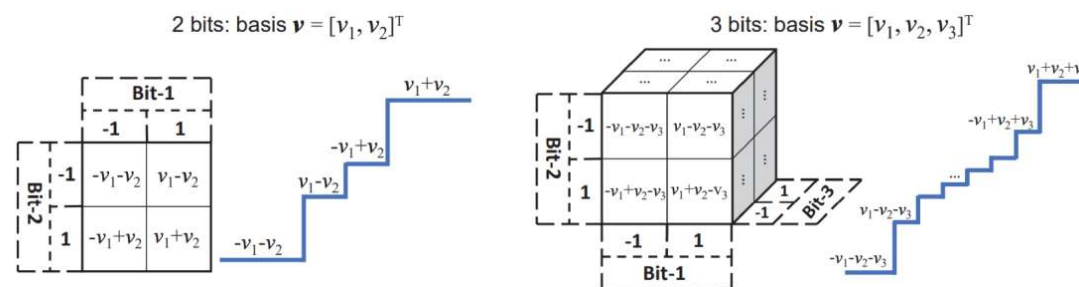
$$Q^*(x) = \arg \min_Q \int p(x)(Q(x) - x)^2 dx,$$

- 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.

- 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  $\mathbf{b}$

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

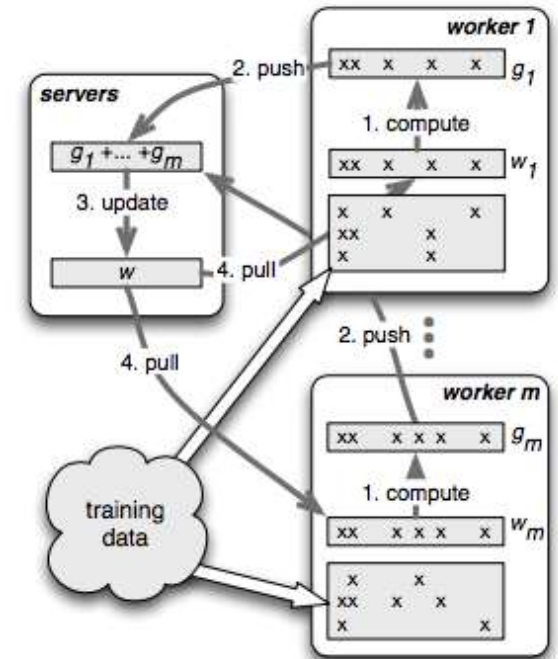
└ Basis vector with K scalars

**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]



# III. Quantization: SketchML (2)

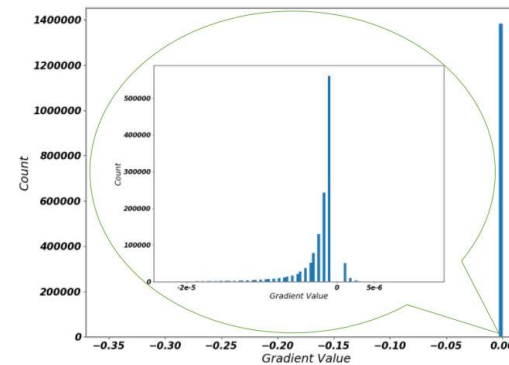
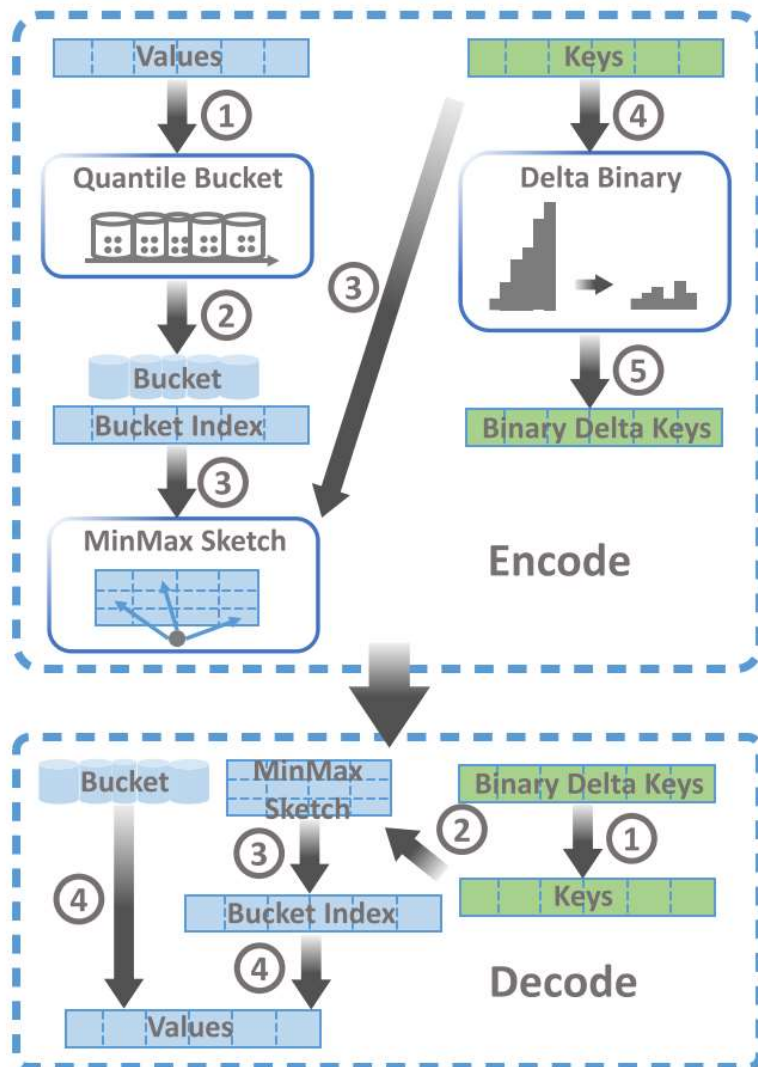
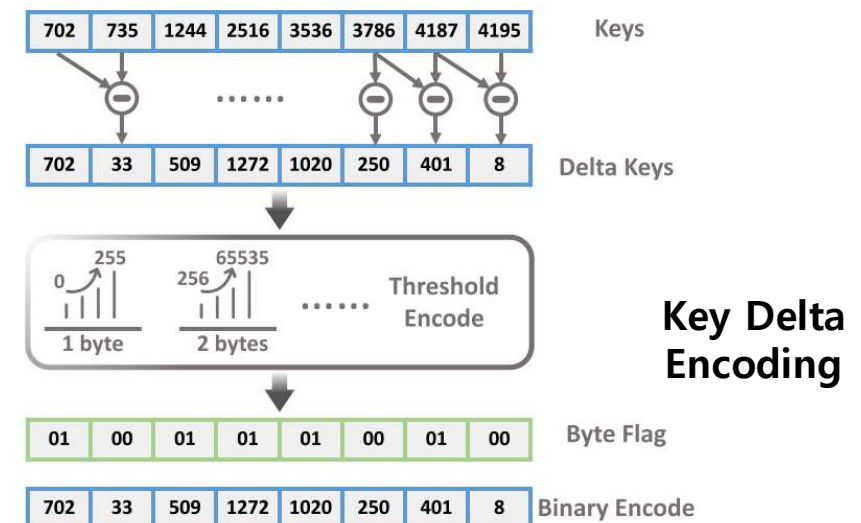
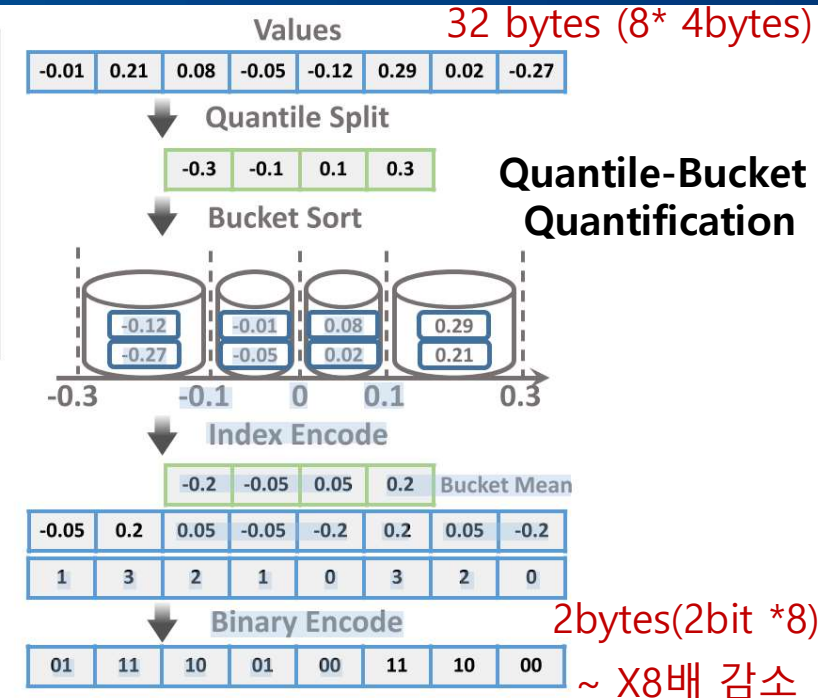


Figure 4: Nonuniform Gradient Values



[전달하는 gradient 값의 극단적인 감소가 가능]

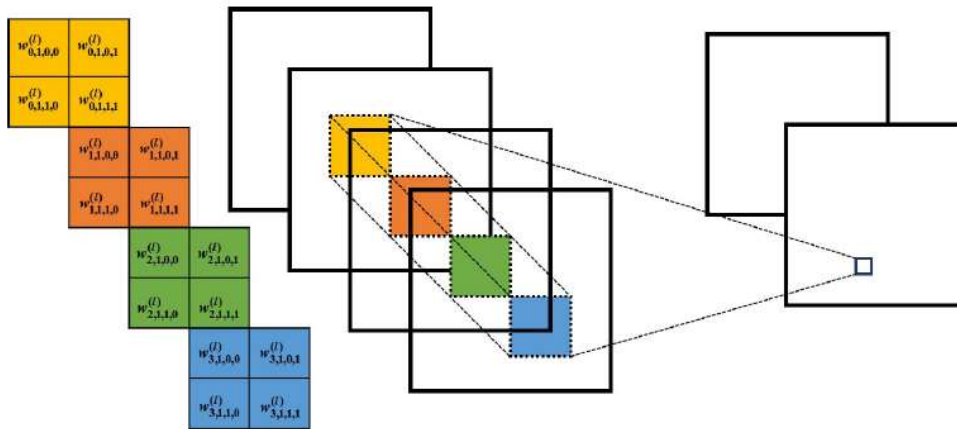
### III. Quantization : Summary

- + 모델 크기의 큰 단축 ( $\sim \times 32$ ; 32bit  $\rightarrow$  1bit)
- + 추론 시간의 개선
- 정확도의 저하
  - 1b& 2b activation Resnet50 achieves 64.6% top-1 accuracy on ImageNet (full-precision 75.3%)
- 학습 시간에서의 큰 개선이 없음
  - Backward Prop. 시의 real weight 유지 필요 등



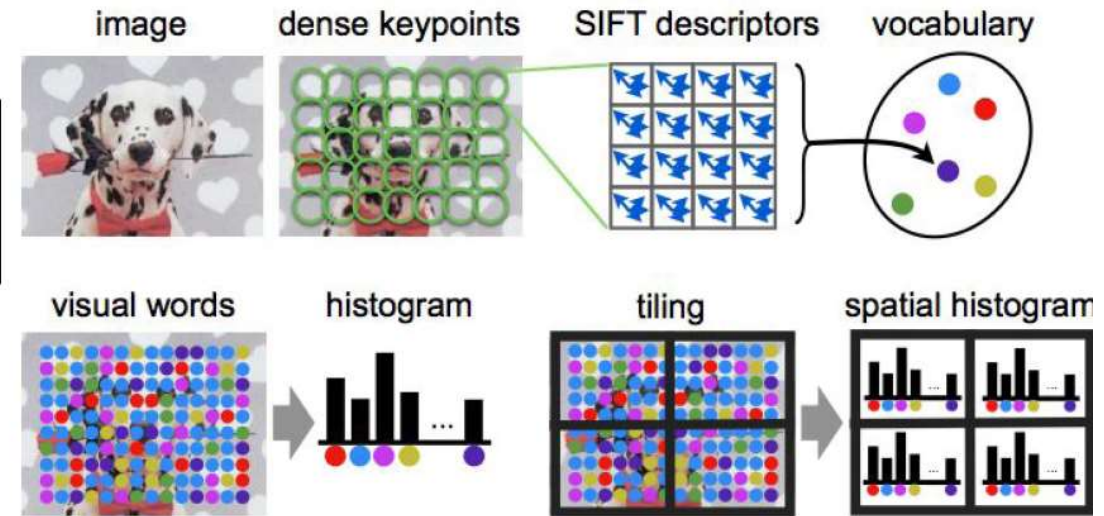
# IV. Compact Network Design (1)

## • Convolutional Neural Network Basics



Convolution Operation  
- feature extraction

$$Output[b, i, j, k] = \sum_{d_i, d_j, q} Input[b, strides[1] * i + d_i, strides[2] * j + d_j, q] \times filter[d_i, d_j, q, k]$$

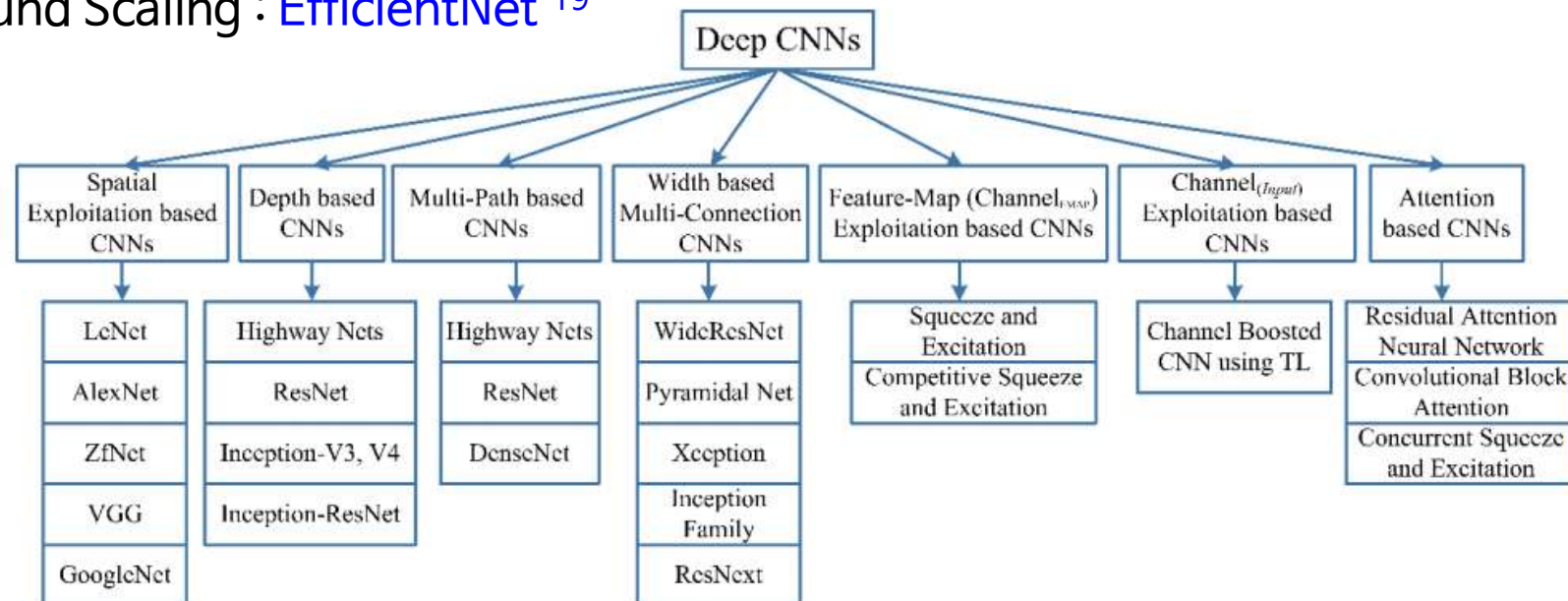


www.robots.ox.ac.uk

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

## IV. Compact Network Design (2)

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

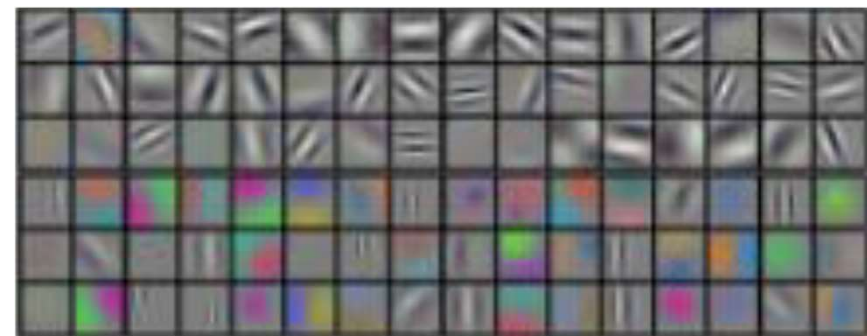
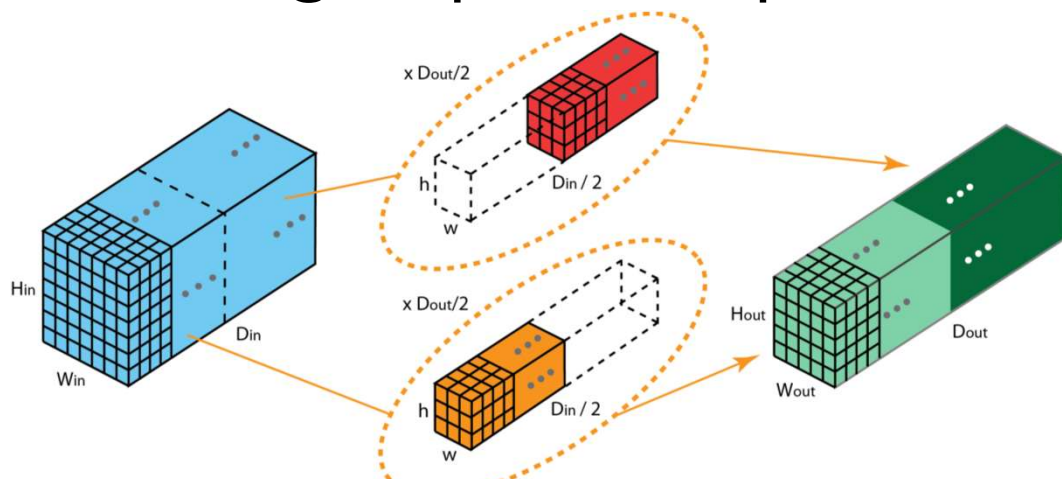


\* Khan, Asifullah, et al. "A survey of the recent architectures of deep convolutional neural networks." *Artificial Intelligence Review* (2020): 1-62.

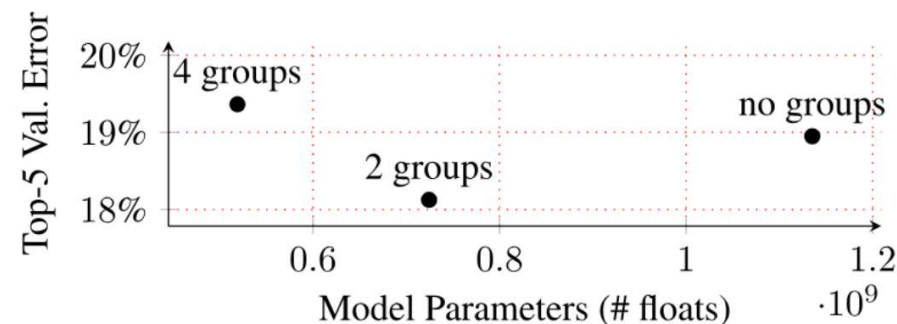
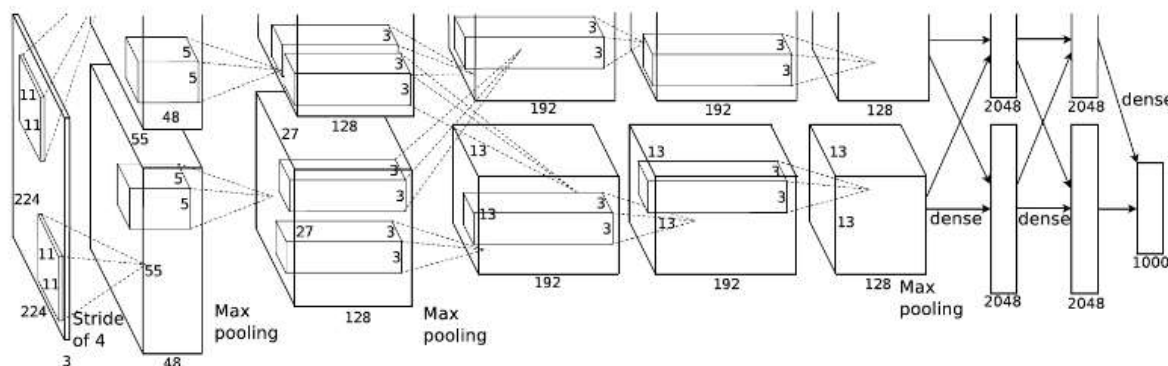
# Grouped Convolution

- Group filters to separate channel information and learn (benefit to parallel processing)
- Sparse by learning highly correlated information for each filter group (fewer parameters)

[ AlexNet, ShuffleNet ]



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



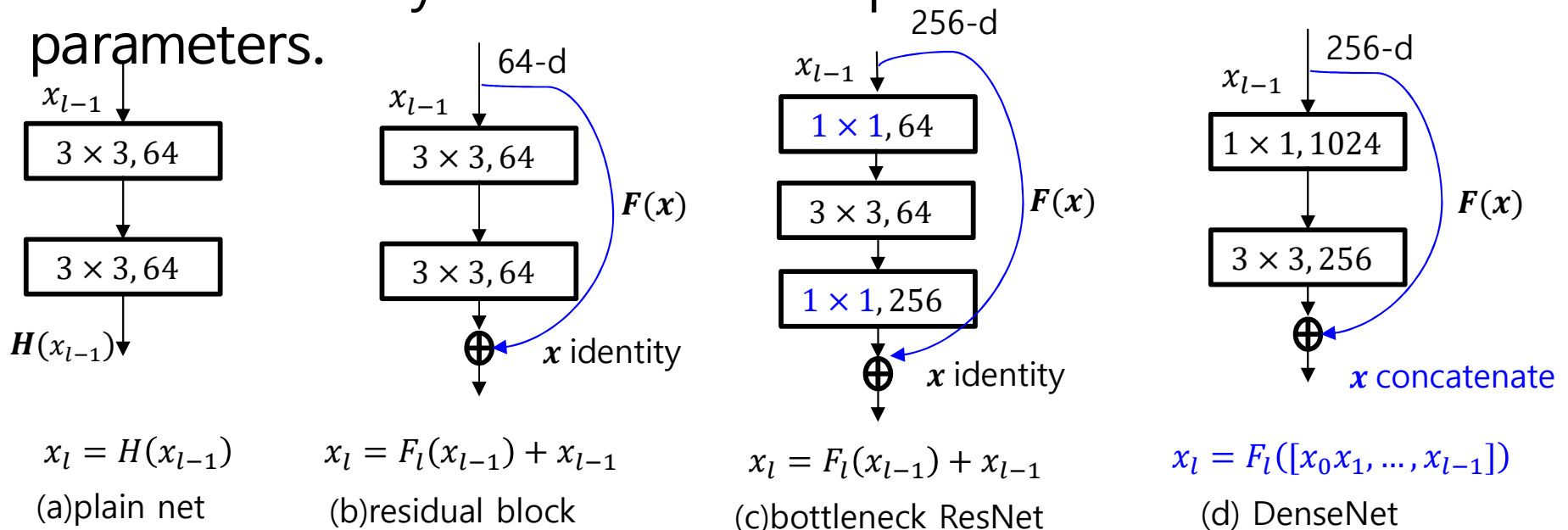
[ ResNet, DenseNet ]

## • Residual Connection

- With the same number of parameters, there is little additional computation other than addition, so deep networks are well optimized.

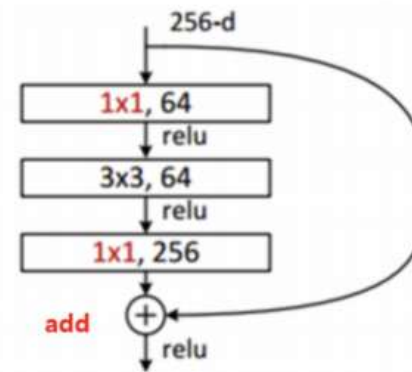
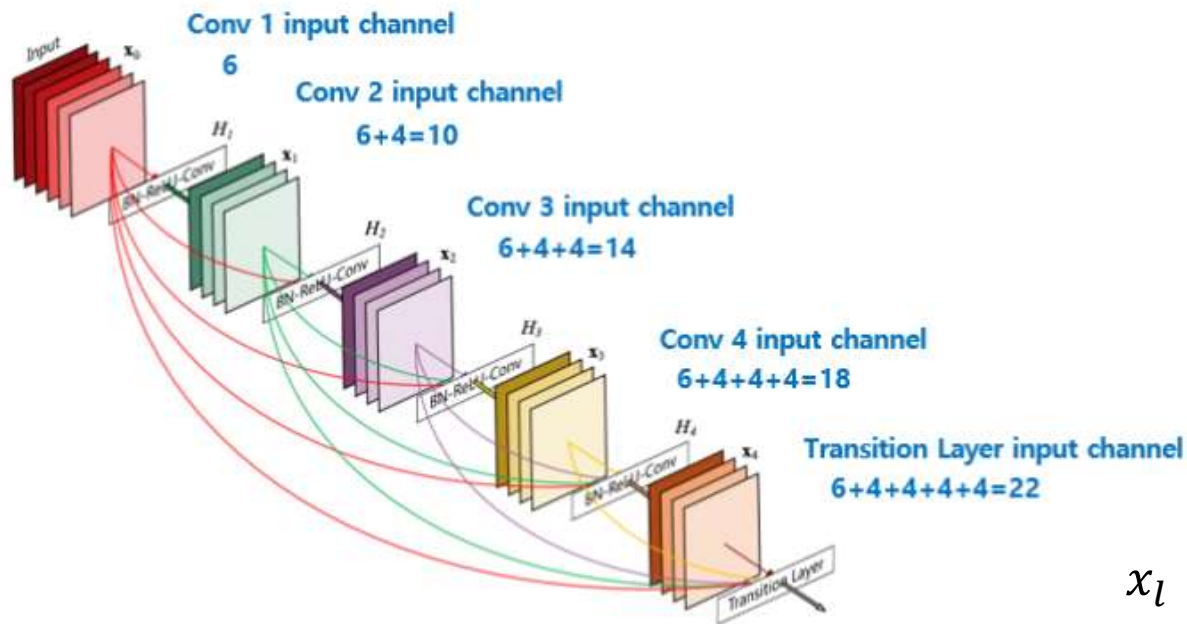
## • Dense Net

- While collecting feature maps as concatenation, compression at the transition layer ensures better performance with fewer parameters.

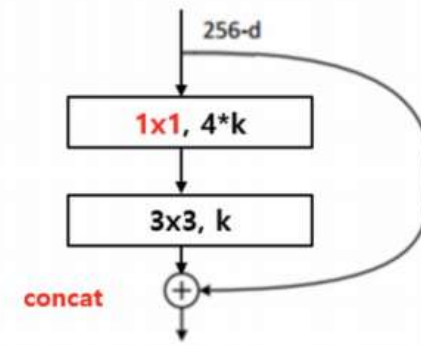




- Dense Convolution (DenseNet\*)
  - 기존의 feature map을 additive 형태가 아닌 concat 형태로 취합
  - feature-maps 압축으로 적은 parameter로 더 나은 성능 확인



bottleneck  
(for ResNet)



bottleneck  
(for DenseNet)

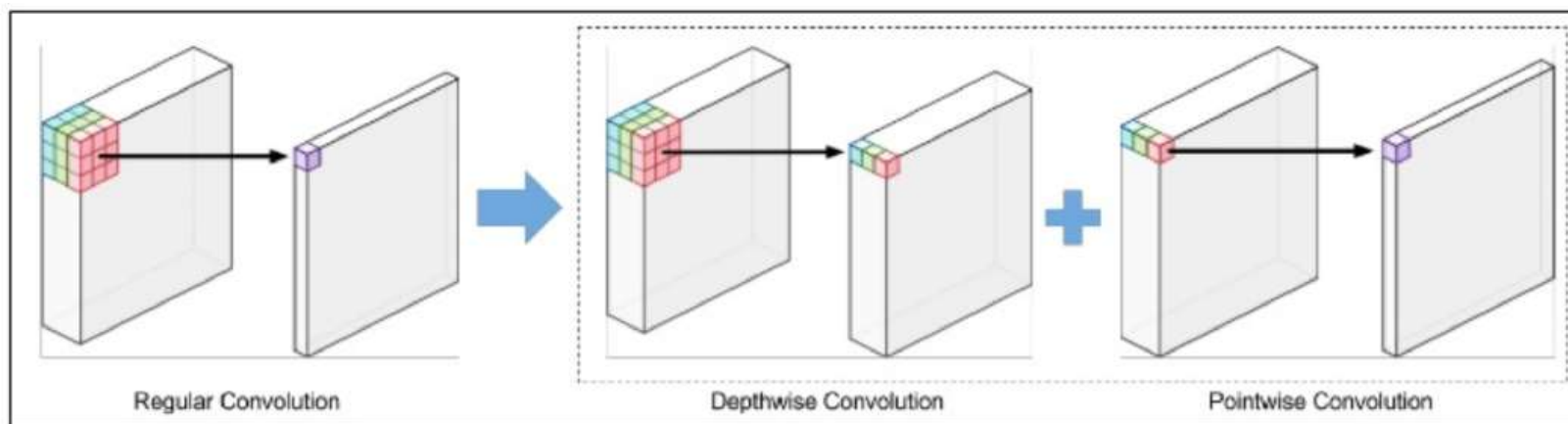
$$x_l = H_l(x_{l-1}) + x_{l-1} \quad x_l = H_l([x_0, x_1, \dots, x_{l-1}])$$

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

# 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,  $N$  필터 출력 채널 수

	Spatial Convolution	Depthwise Separable Conv.
operation 수	$N(CK^2 + 1)$ $= NCK^2 + N$	$C(K^2 + 1) + N(C + 1)$ $= CK^2 + NC + C + N$
Parameter 수	$CXYK^2N$	$CXYK^2 + CXYN$ $= CXY(K^2 + N)$



\* Francois Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions," CVPR 2017

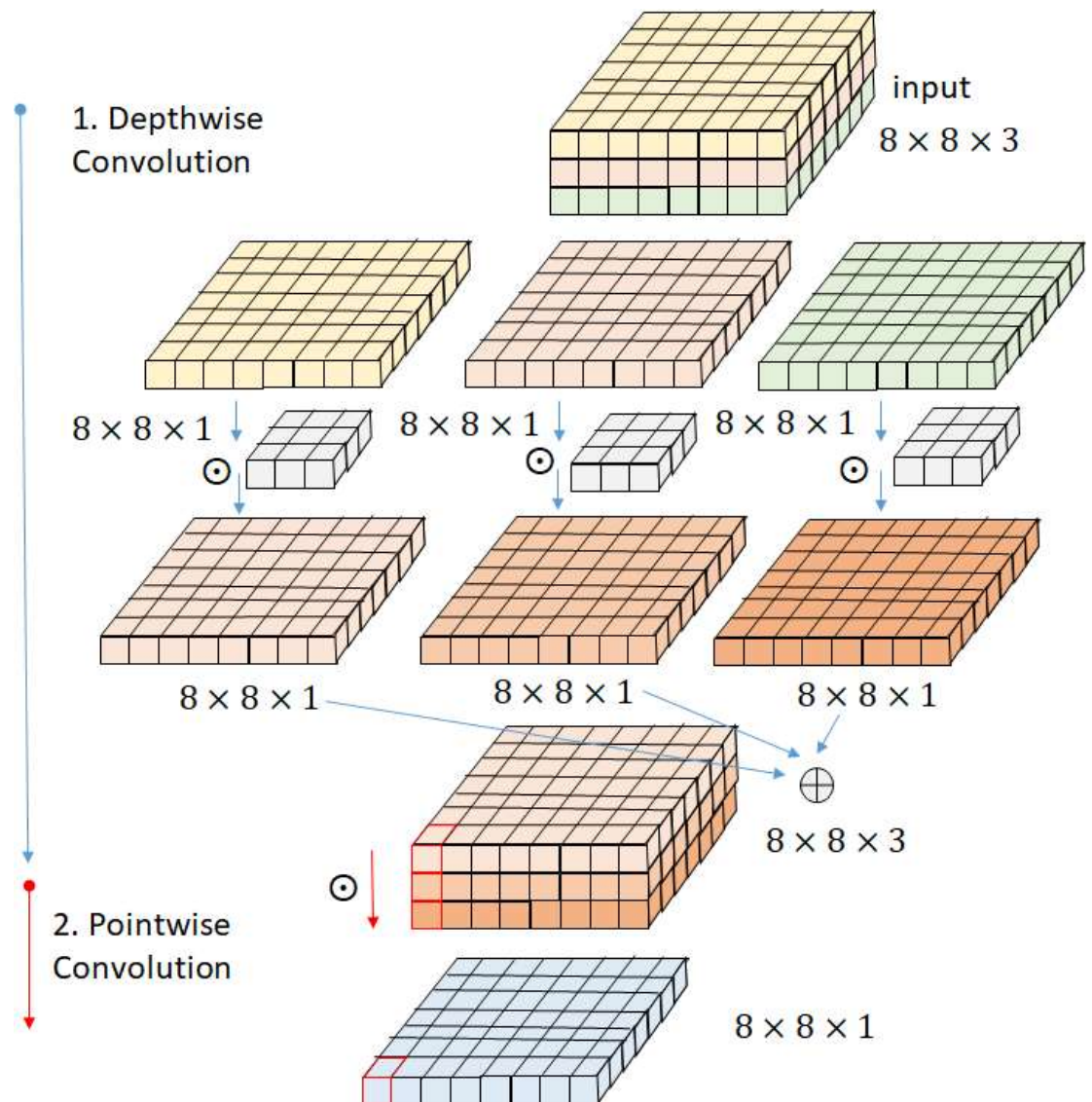
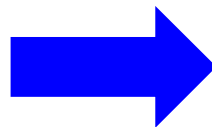
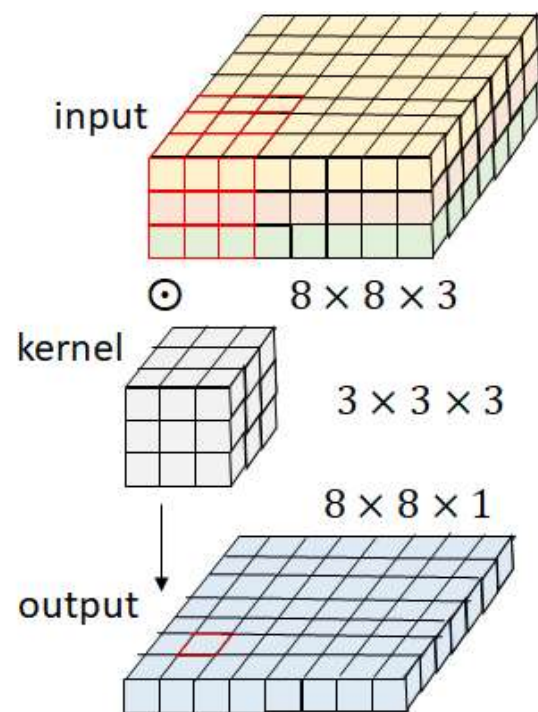
\*\* Andrew G. Howard et al., "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," CVPR 2017

# Reinterpretation of Convolutions

[ MobileNet, Inception ]

## Depth-Wise Separable Point-wise Convolution

### General Convolution

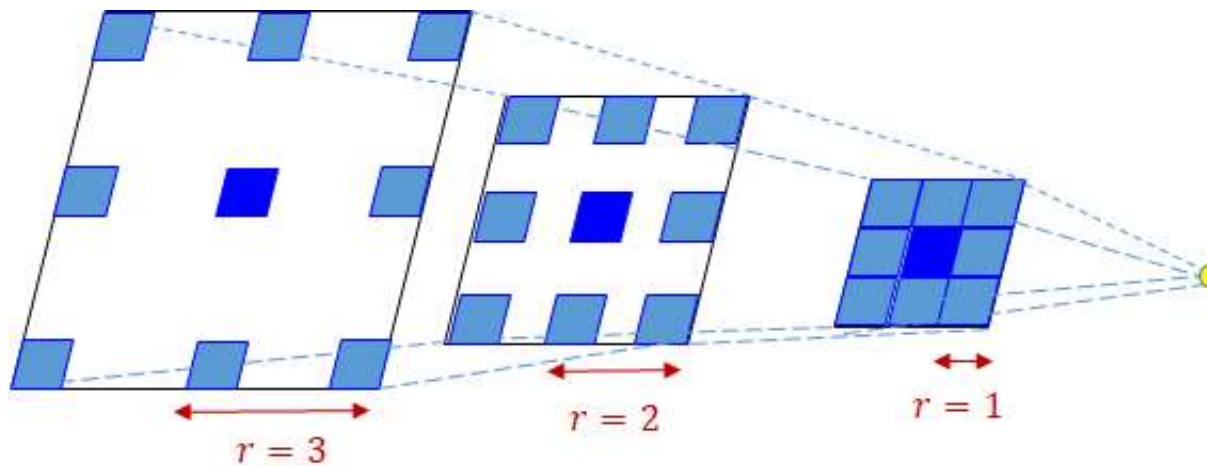




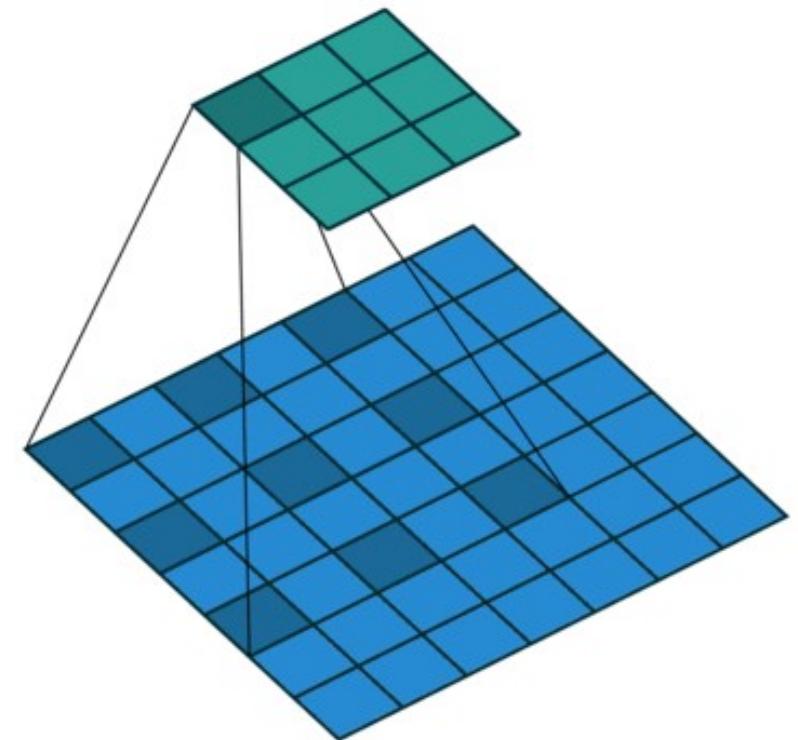
- Overview

[ DeepLab, ShiftNet ]

- Receptive field area is widened, but by maintaining the kernel size, the amount of convolution operation is maintained while obtaining a pooling effect to improve accuracy



$r$  : dilation rate, the spacing between kernels



- Shift (ShiftNet\*, \*\*)

- Spatial convolution을 shift operation으로 대체하여 연산을 줄임

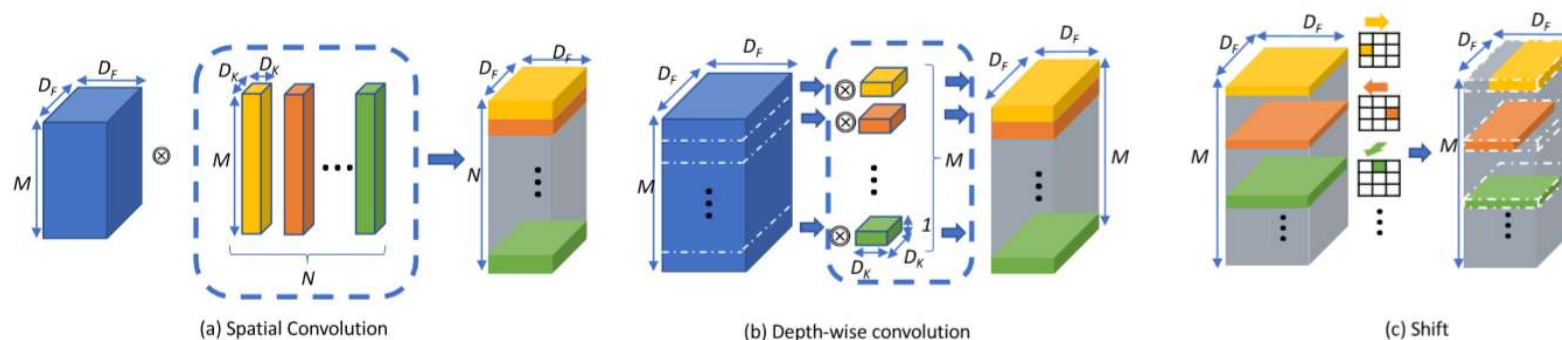


Figure 2: Illustration of (a) spatial convolutions, (b) depth-wise convolutions and (c) shift. In (c), the 3x3 grids denote a shift matrix with a kernel size of 3. The lighted cell denotes a 1 at that position and white cells denote 0s.

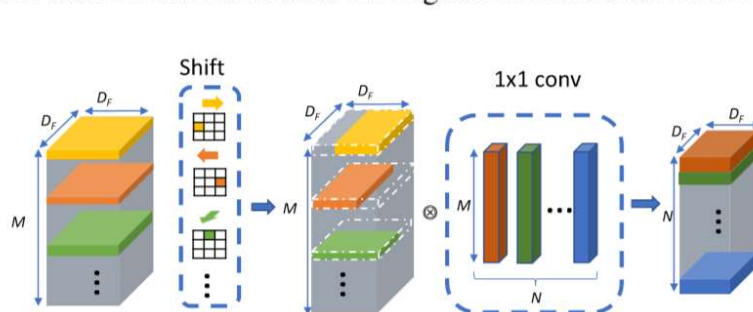
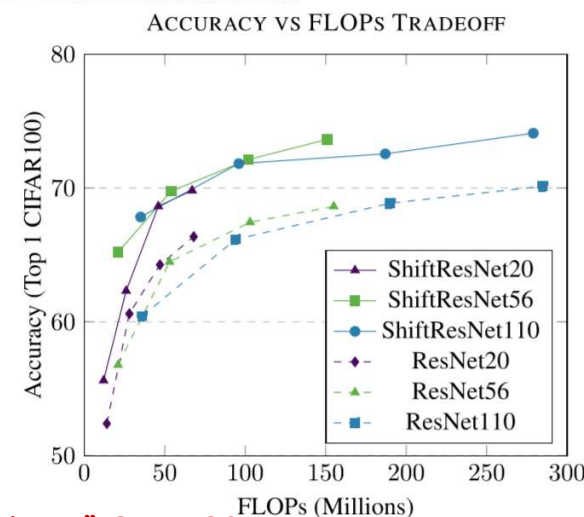


Figure 1: Illustration of a shift operation followed by a 1x1 convolution. The shift operation adjusts data spatially and the 1x1 convolution mixes information across channels.

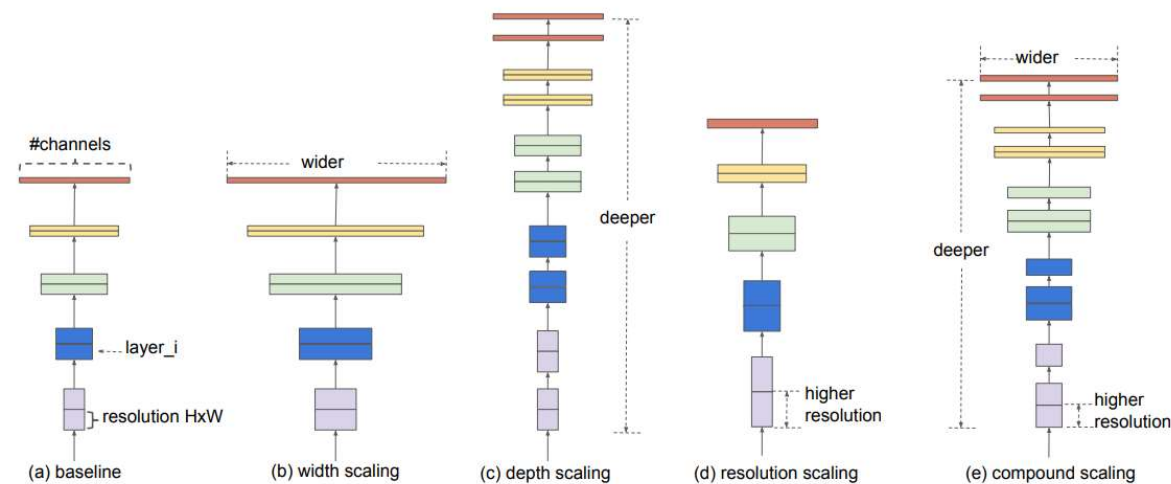


\* B. Wu et al., "Shift: A zero FLOP, Zero Parameter Alternative to Spatial Convolutions," CVPR 2017

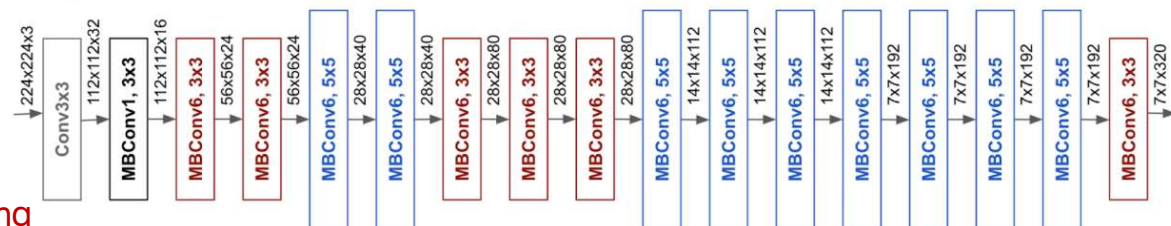
\*\* W. Chen et al., "All You Need is a Few Shifts: Designing Efficient Convolutional Neural Networks for Image Classification," CVPR 2019

## • Compound Scaling (EfficientNet\*)

- CNN model의 3요소 (Depth  $\alpha$ , Width  $\beta$ , Resolution  $\gamma$ )의 Efficient한 Scaling 방법에 대한 연구
- mNasNet\*\*의 결과 모델을 base model로 찾음
- $\alpha = 1.2$ ,  $\beta = 1.1$ ,  $\gamma = 1.15$ 를 고정한 채,  $\phi$ 를 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	$224 \times 224$	32	1
2	MBConv1, k3x3	$112 \times 112$	16	1
3	MBConv6, k3x3	$112 \times 112$	24	2
4	MBConv6, k5x5	$56 \times 56$	40	2
5	MBConv6, k3x3	$28 \times 28$	80	3
6	MBConv6, k5x5	$28 \times 28$	112	3
7	MBConv6, k5x5	$14 \times 14$	192	4
8	MBConv6, k3x3	$7 \times 7$	320	1
9	Conv1x1 & Pooling & FC	$7 \times 7$	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

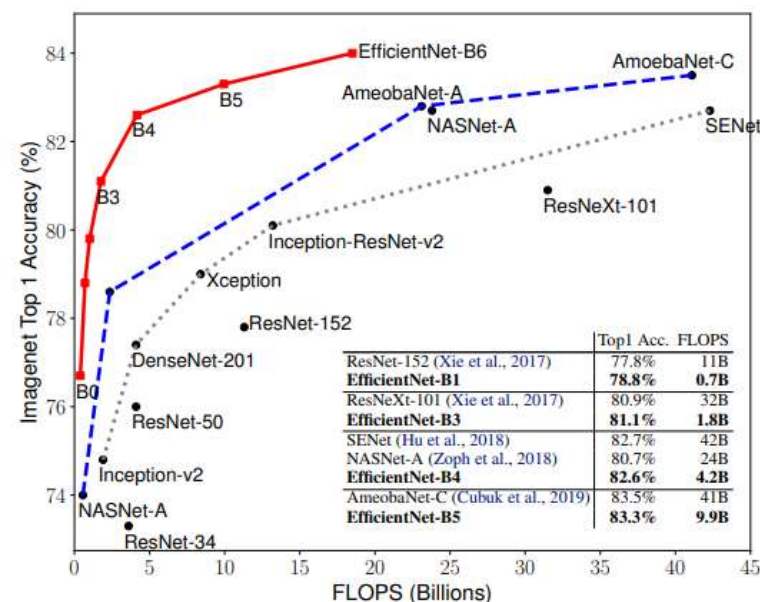
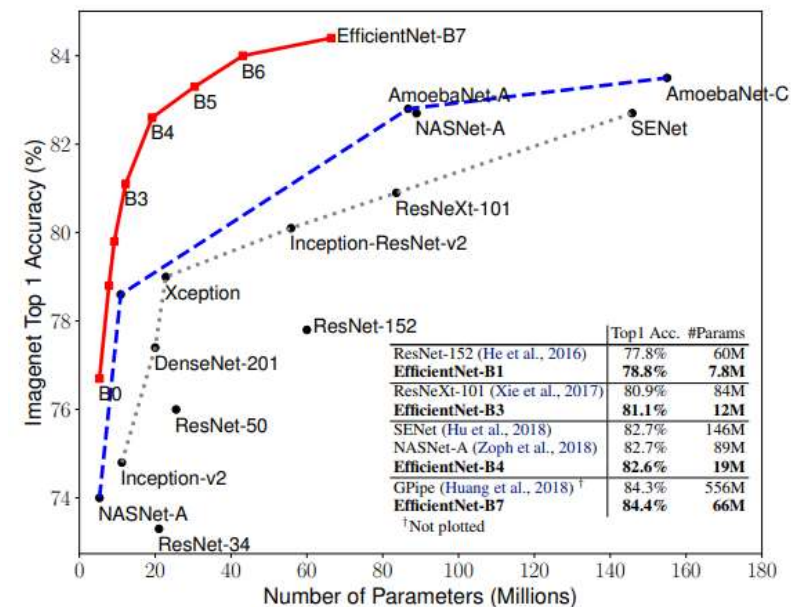


Table 4. Inference Latency Comparison

Acc. @ Latency	
ResNet-152	77.8% @ 0.554s
EfficientNet-B1	78.8% @ 0.098s
<b>Speedup</b>	<b>5.7x</b>
Acc. @ Latency	
GPipe	84.3% @ 19.0s
EfficientNet-B7	84.4% @ 3.1s
<b>Speedup</b>	<b>6.1x</b>

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

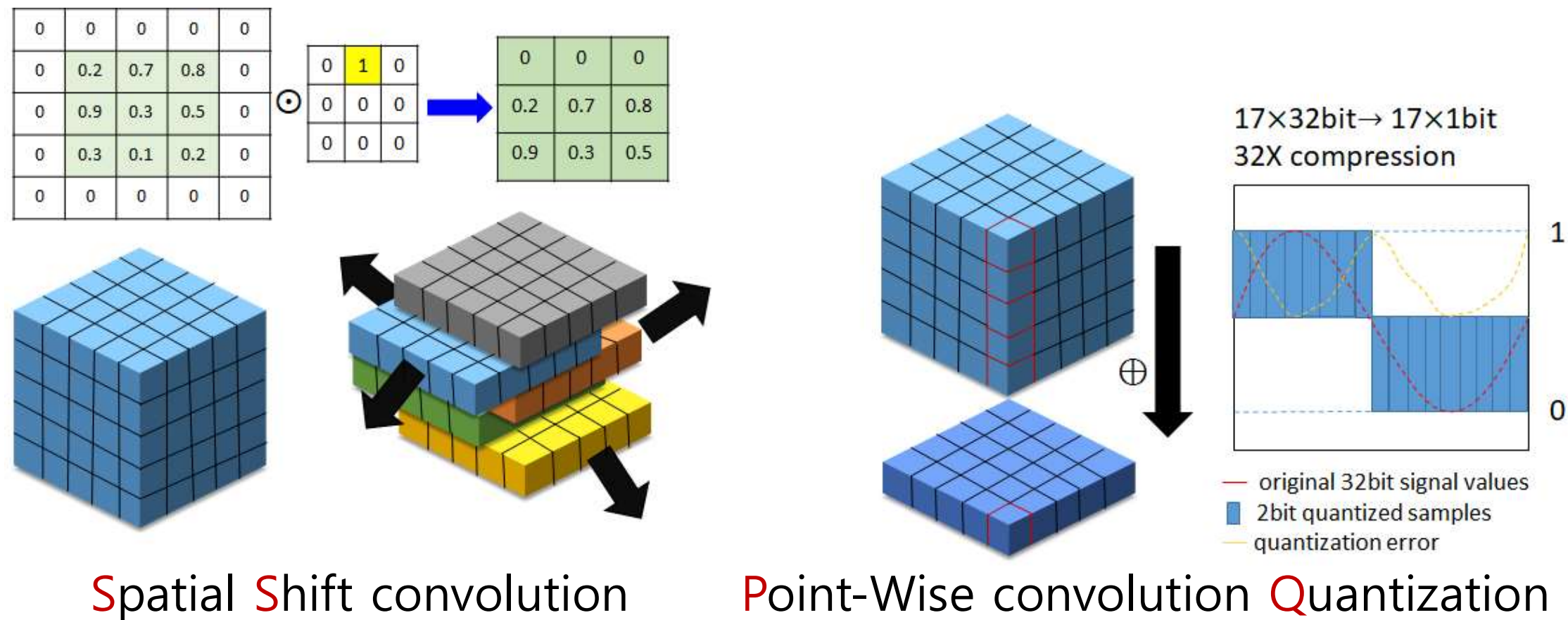
Figure 5. FLOPS vs. ImageNet Accuracy – Similar to Figure 1

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



# III. Studies: Compact Network Design

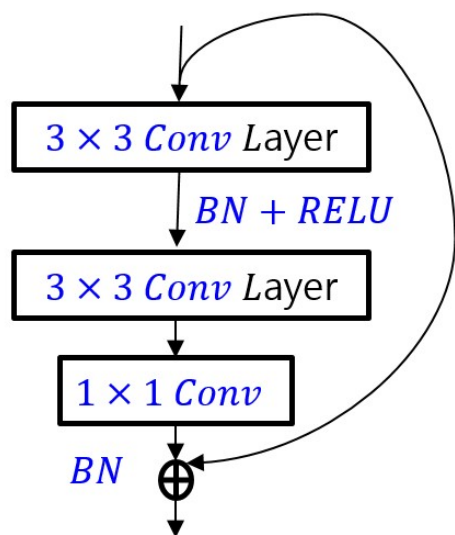
- Spatial Shift Point-wise Quantization Network Design
  - Spatial operation is performed by shift operation and quantization is applied to the Point-Wise convolution to reduce 90MB storage.



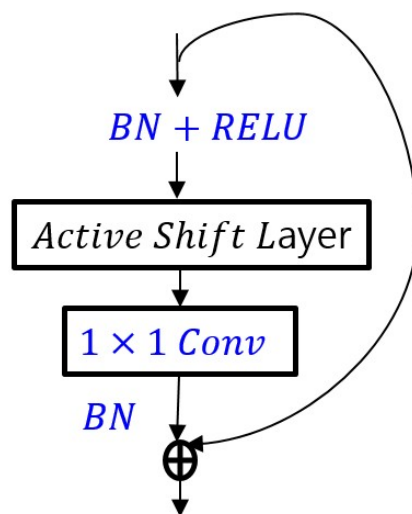


# SSPQ Neural Network Structure

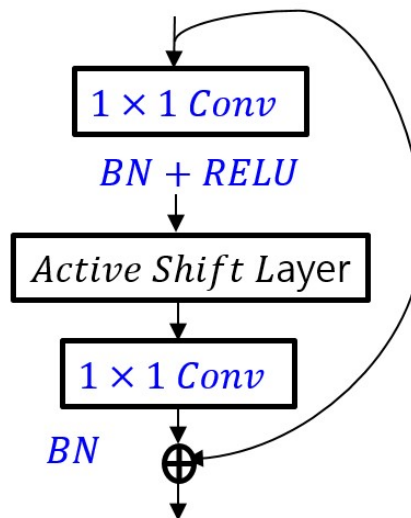
- Inverted residual block & DoReFa-Net quantization techniques are applied



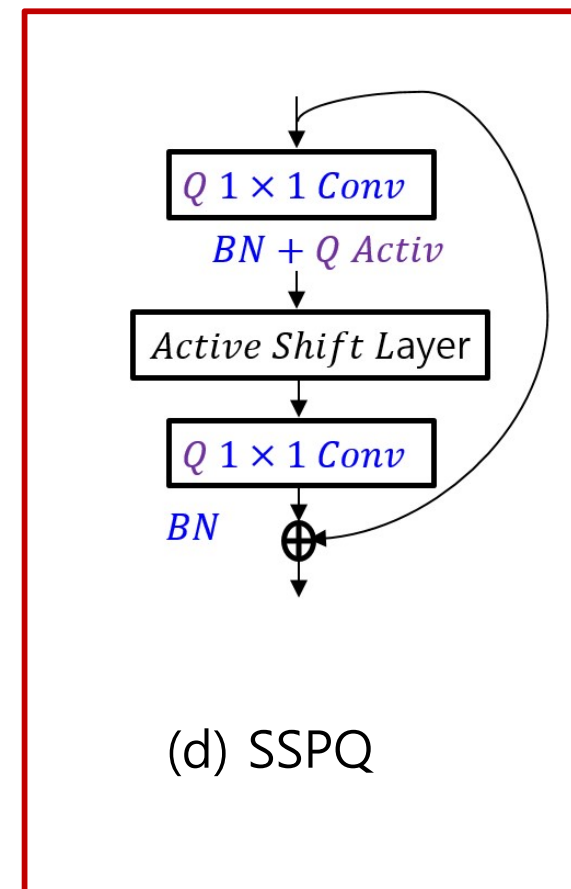
(a) Resnet Basic



(b) Active Shift



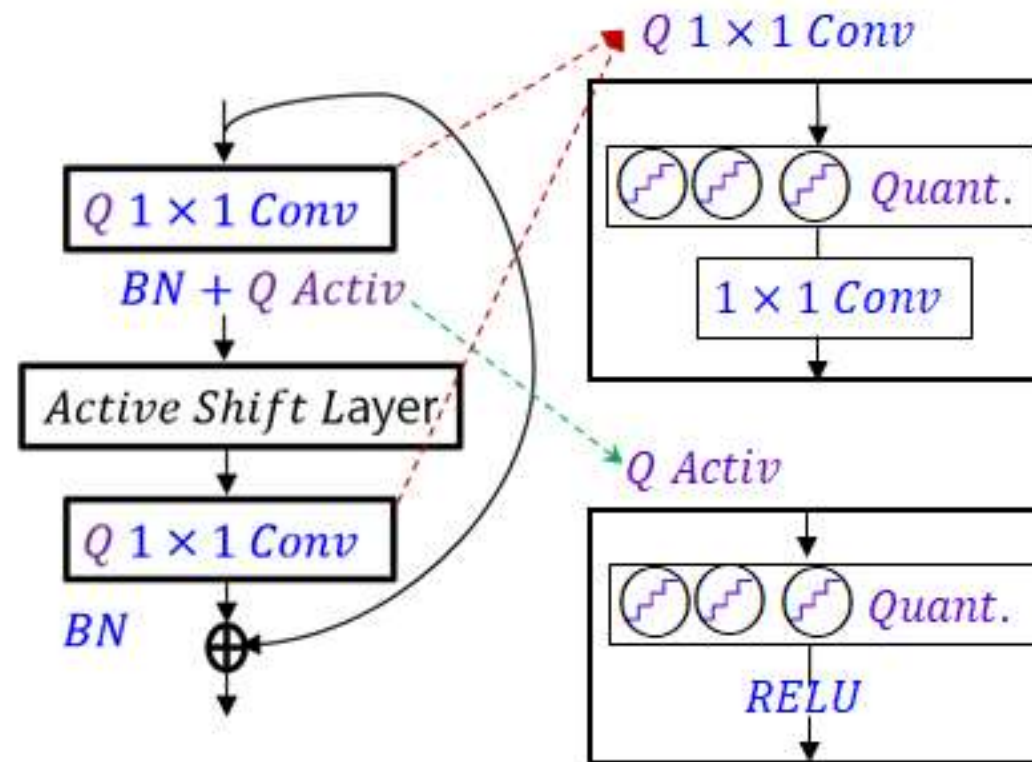
(c) Inverted Residual



(d) SSPQ

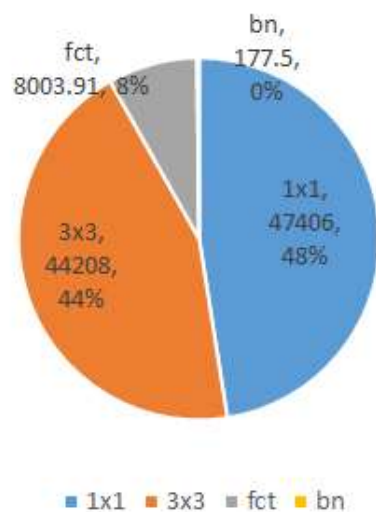
# SSPQ Neural Network Structure

- Inverted residual block & DoReFa-Net quantization techniques are applied
  - Applying Batch Normalization after quantization to reduce quantization error

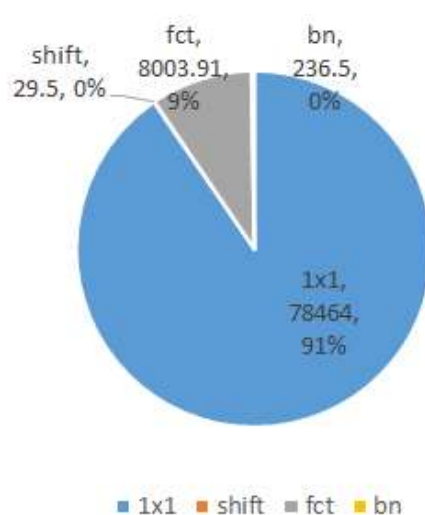


- 10.2X reduction in size
  - Compared to ResNet50, with 73.66% accuracy (2.68% accuracy degradation)
  - Reduced 90MB storages in 1x1 convolutions by quantization

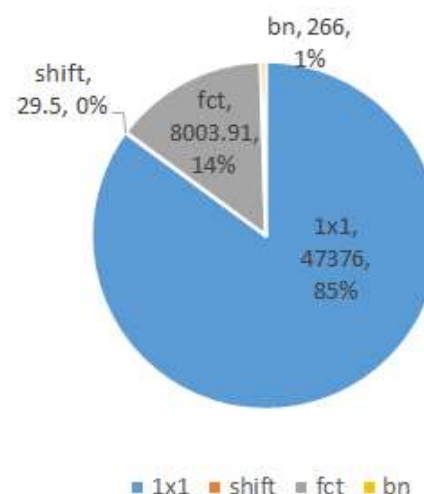
97.49 MB → 84.74 MB → 54.41 MB → 9.79\* MB



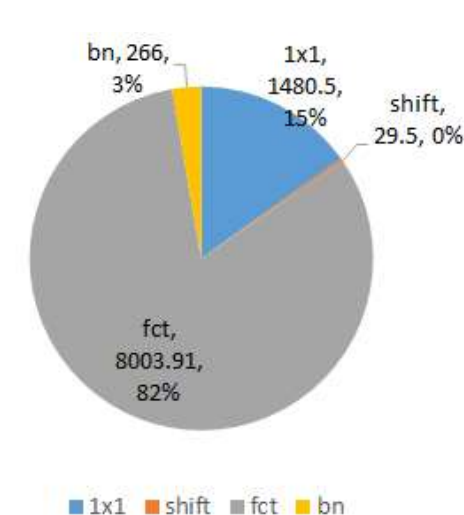
(a) ResNet Basic



(b) Active Shift

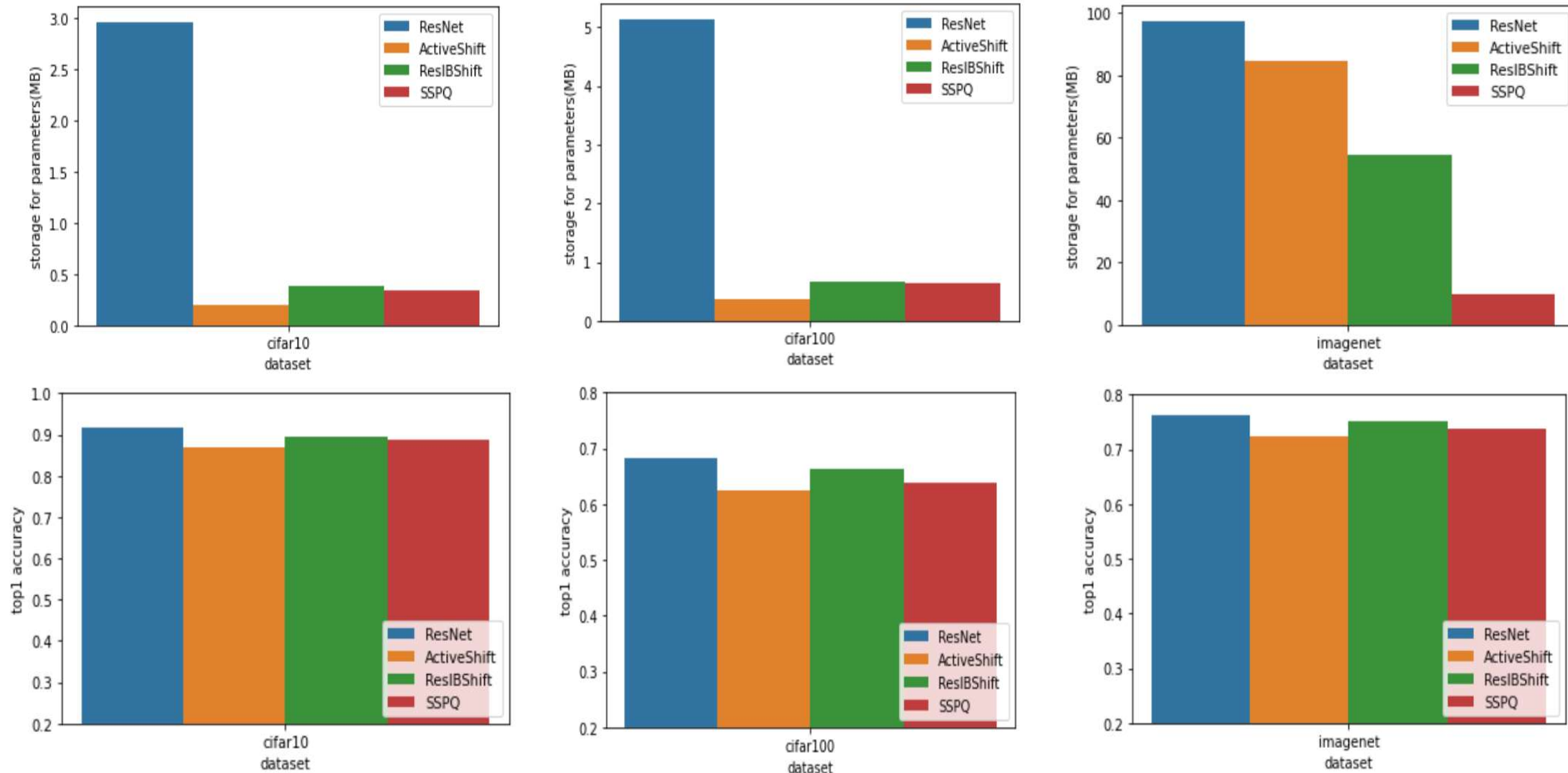


(c) Inverted Residual



(d) SSPQ

- Experimental Results for CIFAR10, CIFAR100 and ImageNet



# SSPQ Neural Network Analysis

- compared to ResNet50, with 73.66% accuracy (2.68% accuracy degradation) in ImageNet dataset
- Better than MobileNet\_v2, comparable to kill-the-bits

