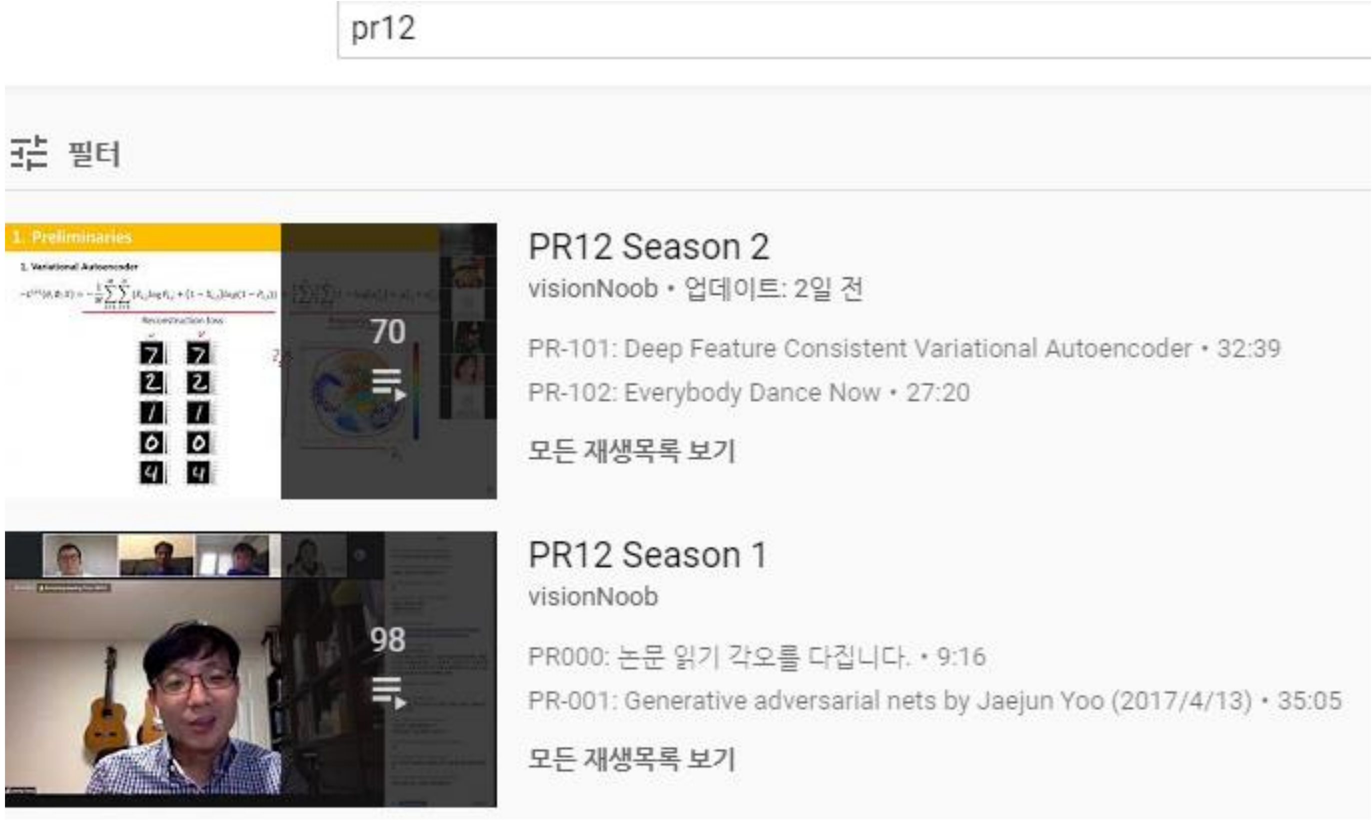
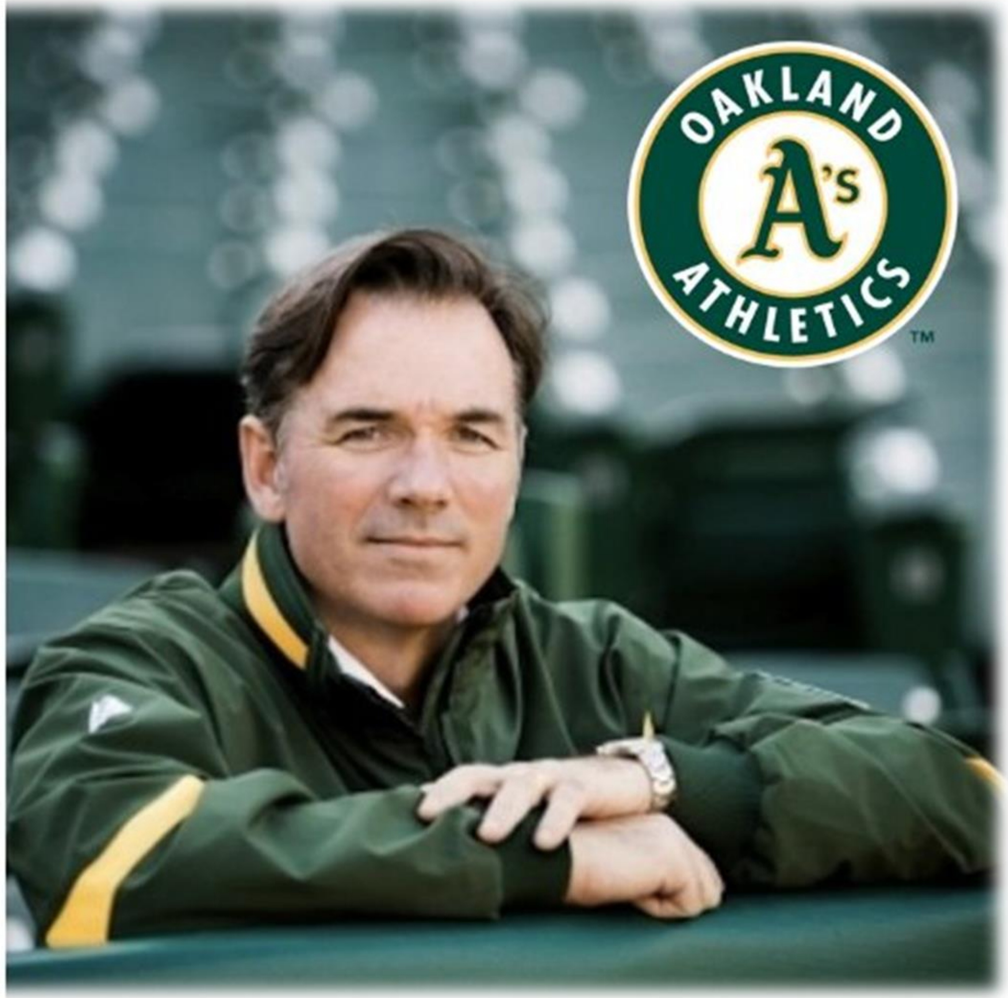
Efficient CNN 톺아보기

Samsung Electronics
Jinwon Lee

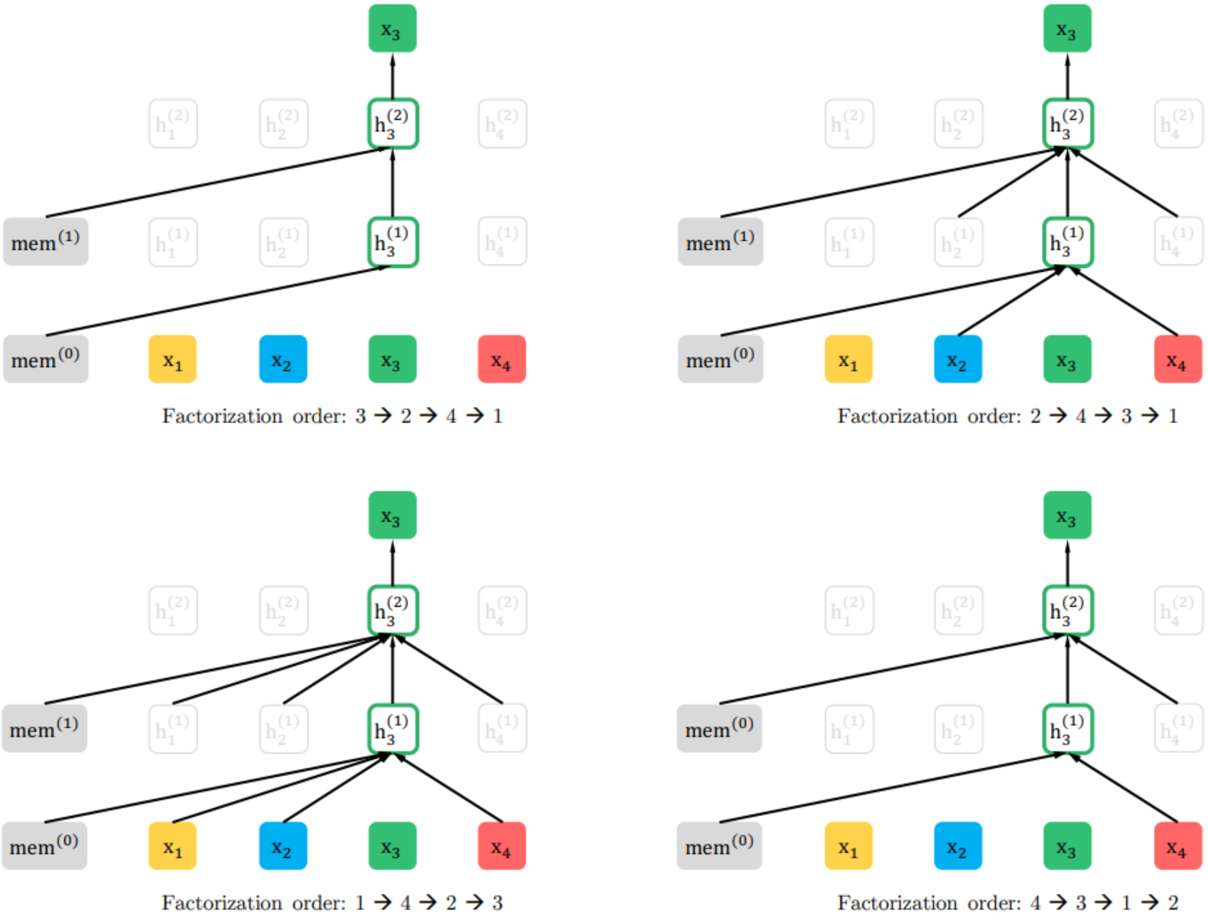

TensorFlow
Korea
가입함
알림
공유하기
더 보기
멤버 44,579명
멤버 찾기
관리자 및 댓글 관리자 9
이진원
관리자
최근 게시물 2개 보기
Samsung Electronics Staff Engineer
이지민
관리자
최근 게시물 1개 보기
서울대학교
메시지 보내기
Hwalsuk Lee
관리자
최근 게시물 1개 보기
Naver Corporation Research Engineer
메시지 보내기


pr12
필터
PR12 Season 2
visionNoob • 업데이트: 2일 전
70
PR-101: Deep Feature Consistent Variational Autoencoder • 32:39
PR-102: Everybody Dance Now • 27:20
모든 재생목록 보기
PR12 Season 1
visionNoob
98
PR000: 논문 읽기 각오를 다집니다. • 9:16
PR-001: Generative adversarial nets by Jaejun Yoo (2017/4/13) • 35:05
모든 재생목록 보기

MoneyBall



XLNet(Yang, arxiv 19 Jun 2019)

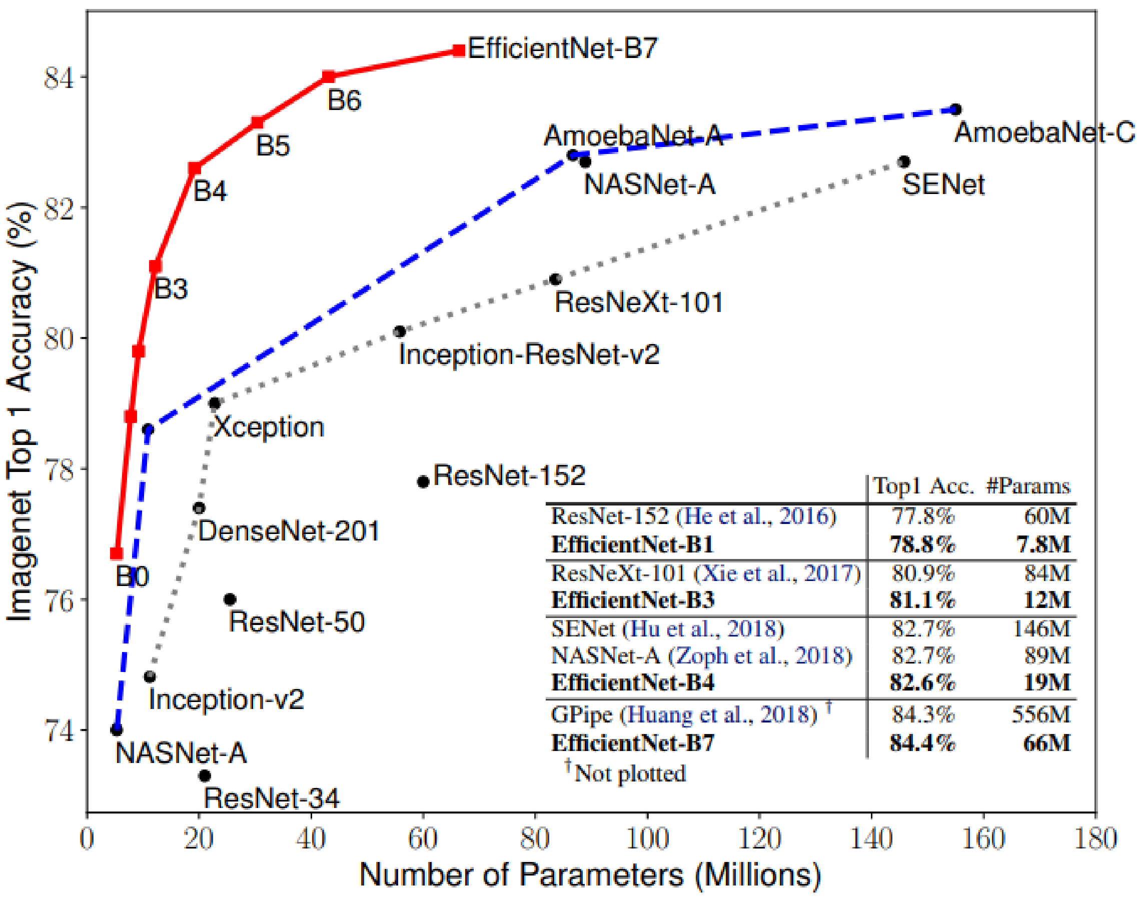


- Parameters
  - 340 million parameters
- Training
  - 512 TPU v3 chips for 500K steps
  - 2.5 days
- 512 TPU x 2.5 days x $8

  = $245,000

Slide Credit : 윤승현's presentation @NVIDIA KR Conference 2019

Convolutional Neural Networks



- GPipe
  - 556 million parameters
- AmoebaNet
  - 450 K40 GPU, 7 days training
- NAS
  - 800 GPU, 28 days training

Toward More Accurate Models

- The impressive results of the current object category recognition systems are obtained on a limited number(20~1000) of basic-level object categories

Dog

Pomeranian

Bicycle

Road Bike

Car

Sports Car

Bag

Shoulder bag

System	ImageNet 22,000 Top-1 Accuracy
[Le12]	13.6%
[Chilimbi14]	**29.8%**

- With large scale dataset with 22,000 categories, the performance drops to 30%

Slide Credit : Prof. Sung Ju Hwang

Efficient Models for On-device AI

그래서 오늘은...

Convolutional Neural Network

그 중에서도
Efficient CNN에는 어떤 model들이 있는지...
논문의 핵심 idea들 위주로
살펴보겠습니다

그리고 몇가지 bonus insight도...

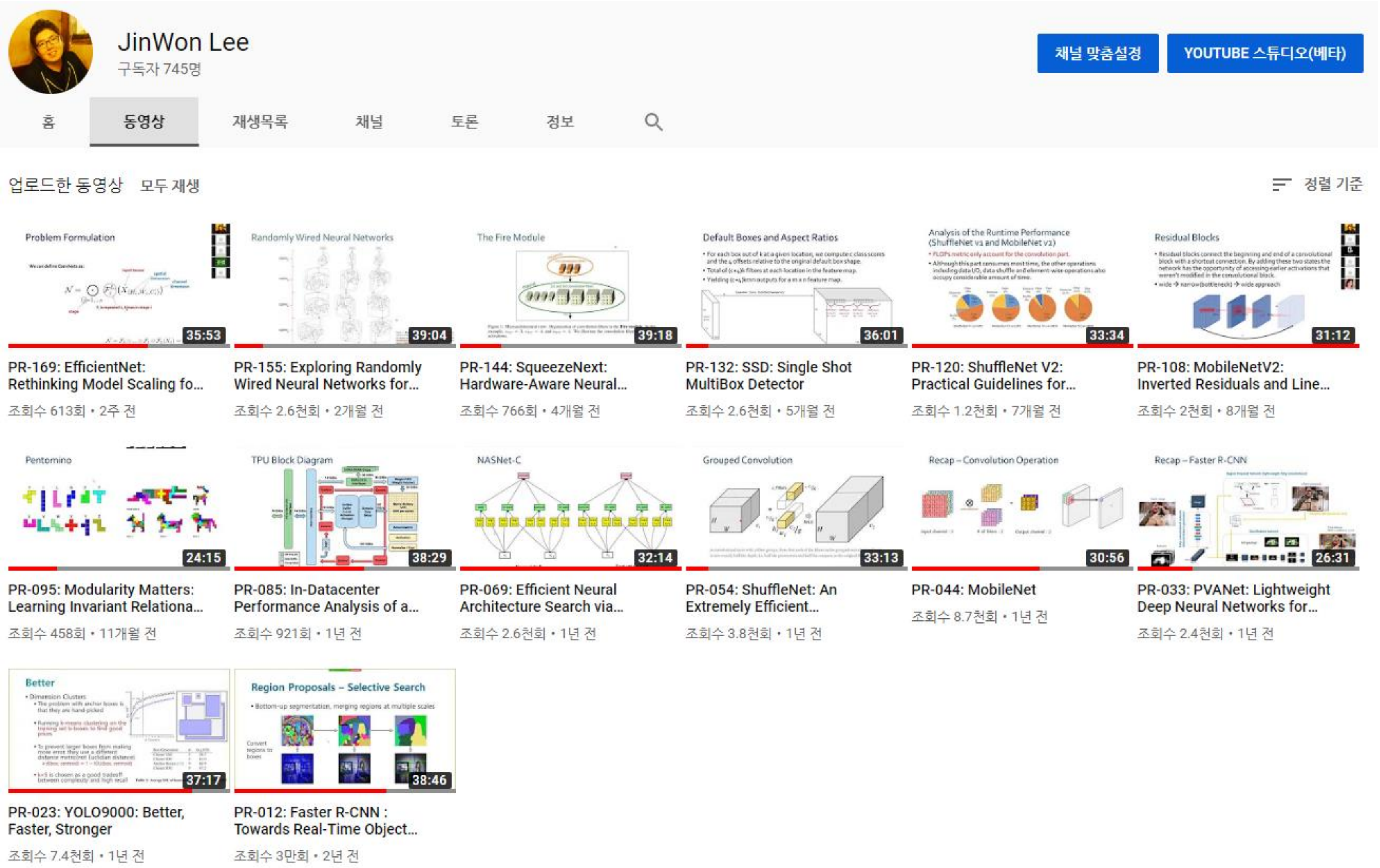
오늘 다루지 않는 것들

- AutoML (ㅠㅠ)
- Pruning
- Knowledge Distillation
- 오늘 소개할 논문들의 Experimental Results

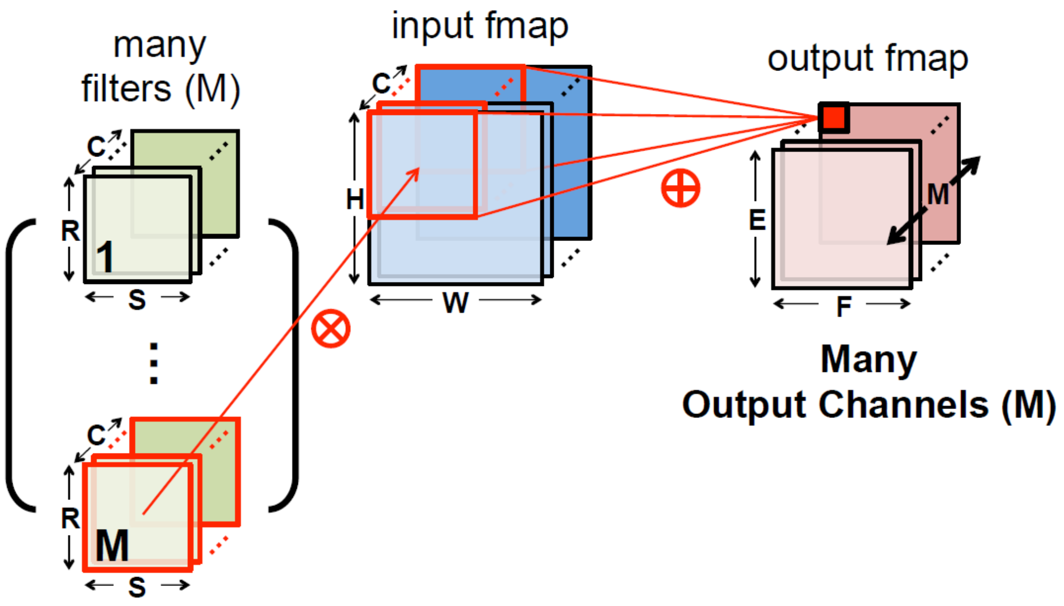
오늘 살펴볼 논문들…

1. Going Deeper with Convolutions
2. Very Deep Convolutional Networks for Large-Scale Image Recognition
3. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and …
4. SqueezeNext: Hardware-Aware Neural Network Design
5. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision …
6. MobileNetV2: Inverted Residuals and Linear Bottlenecks
7. Searching for MobileNetV3
8. ShuffleNet: An Extremely Efficient Convolutional Neural Network for …
9. ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design
10. Shift: A Zero FLOP, Zero Parameter Alternative to Spatial Convolutions
11. CondenseNet: An Efficient DenseNet using Learned Group …
12. All You Need is a Few Shifts: Designing Efficient Convolutional Neural …
13. Fully Learnable Group Convolution for Acceleration of Deep Neural …

깨알홍보 – 12PR

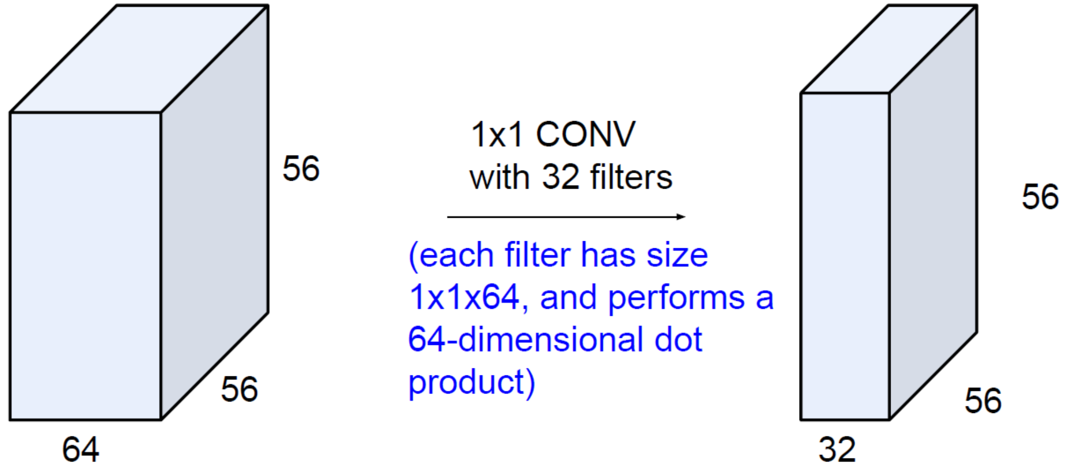

JinWon Lee
구독자 745명
채널 맞춤설정
YOUTUBE 스튜디오(베타)
홈
동영상
재생목록
채널
토론
정보
업로드한 동영상
모두 재생
정렬 기준
35:53
PR-169: EfficientNet: Rethinking Model Scaling fo...
조회수 613회 • 2주 전
39:04
PR-155: Exploring Randomly Wired Neural Networks for...
조회수 2.6천회 • 2개월 전
39:18
PR-144: SqueezeNext: Hardware-Aware Neural...
조회수 766회 • 4개월 전
36:01
PR-132: SSD: Single Shot MultiBox Detector
조회수 2.6천회 • 5개월 전
33:34
PR-120: ShuffleNet V2: Practical Guidelines for...
조회수 1.2천회 • 7개월 전
31:12
PR-108: MobileNetV2: Inverted Residuals and Line...
조회수 2천회 • 8개월 전
24:15
PR-095: Modularity Matters: Learning Invariant Relationa...
조회수 458회 • 11개월 전
38:29
PR-085: In-Datacenter Performance Analysis of a...
조회수 921회 • 1년 전
32:14
PR-069: Efficient Neural Architecture Search via...
조회수 2.6천회 • 1년 전
33:13
PR-054: ShuffleNet: An Extremely Efficient...
조회수 3.8천회 • 1년 전
30:56
PR-044: MobileNet
조회수 8.7천회 • 1년 전
26:31
PR-033: PVANet: Lightweight Deep Neural Networks for...
조회수 2.4천회 • 1년 전
37:17
PR-023: YOLO9000: Better, Faster, Stronger
조회수 7.4천회 • 1년 전
38:46
PR-012: Faster R-CNN : Towards Real-Time Object...
조회수 3만회 • 2년 전

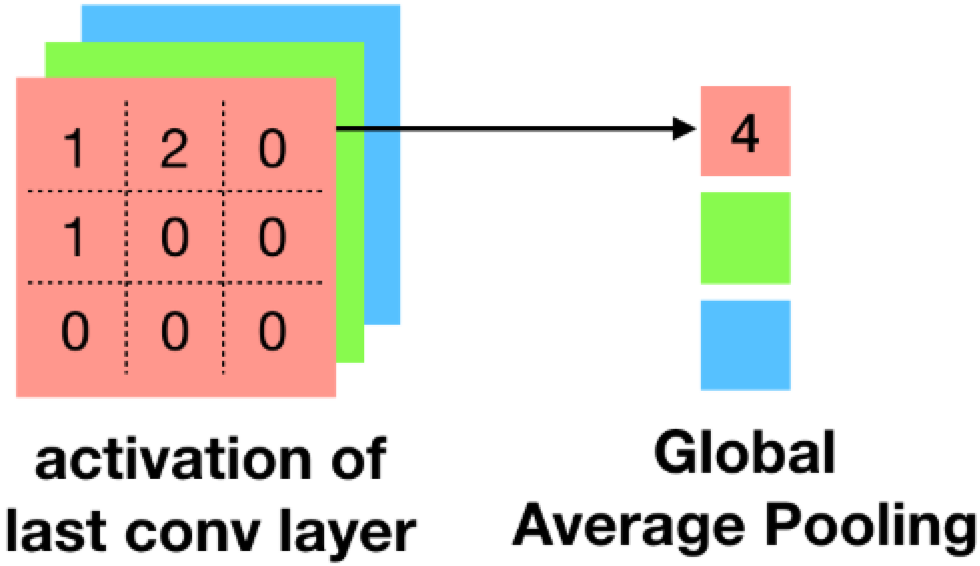
Convolution

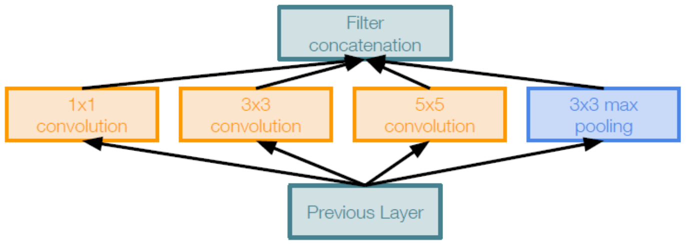


- H=E, W=F인 경우,
  - Parameter 수 : R x S x C x M
  - 연산량 : R x S x C x H x W x M
  - Memory Access Cost : R x S x C x M + (H x W) x (C + M)

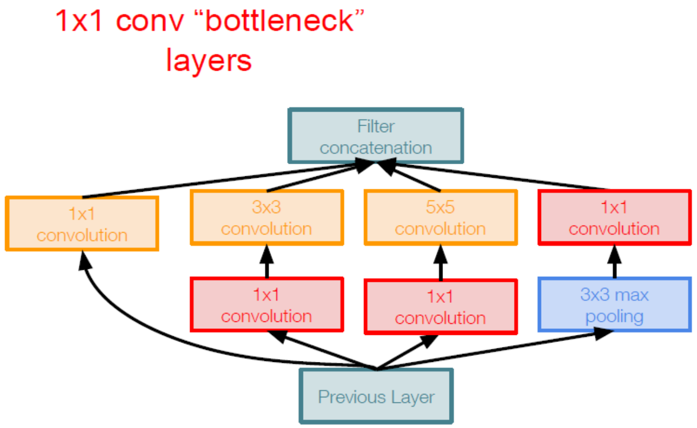
1. Bottleneck Layer, Global Average Pooling







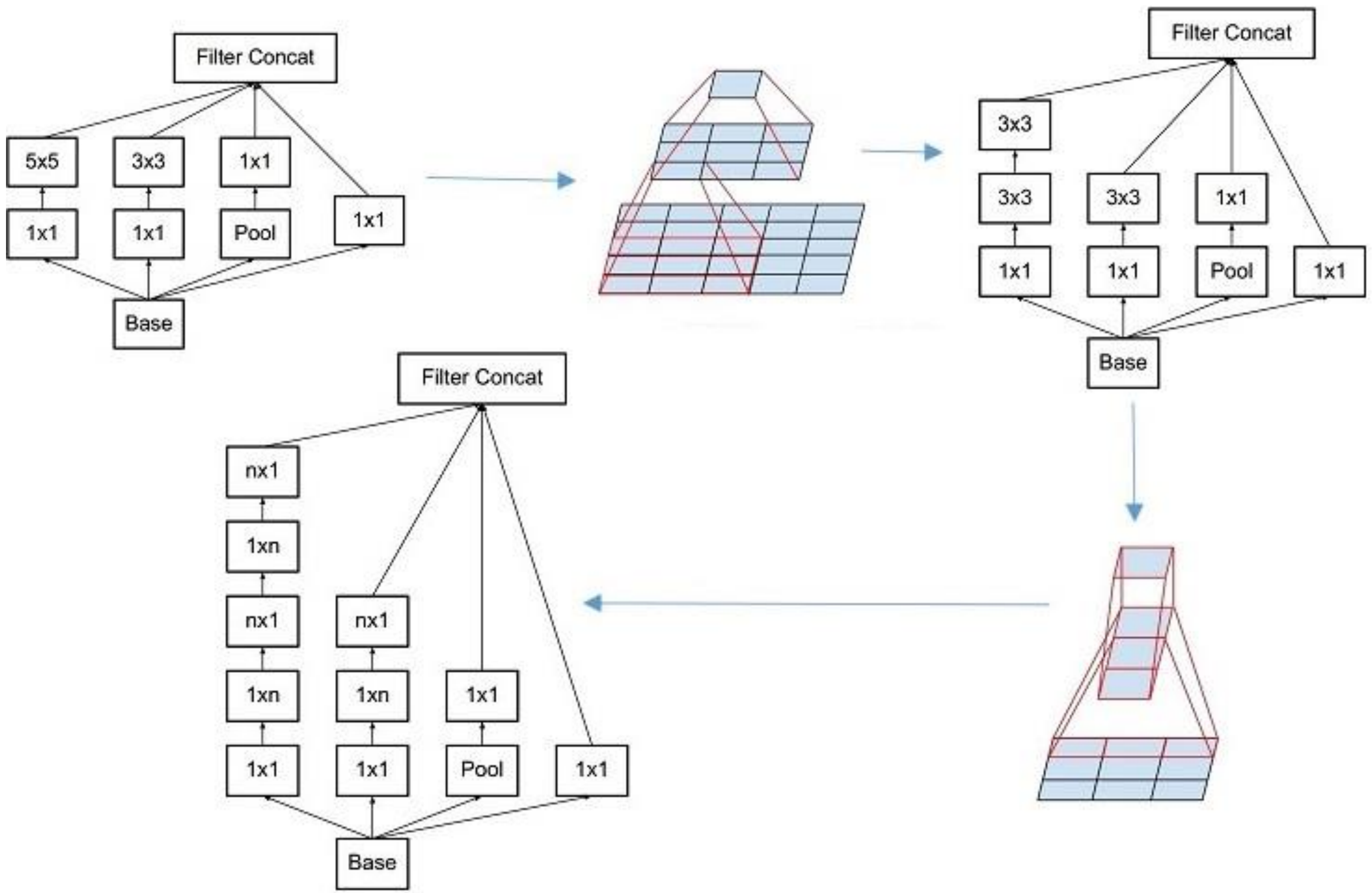
Naive Inception module

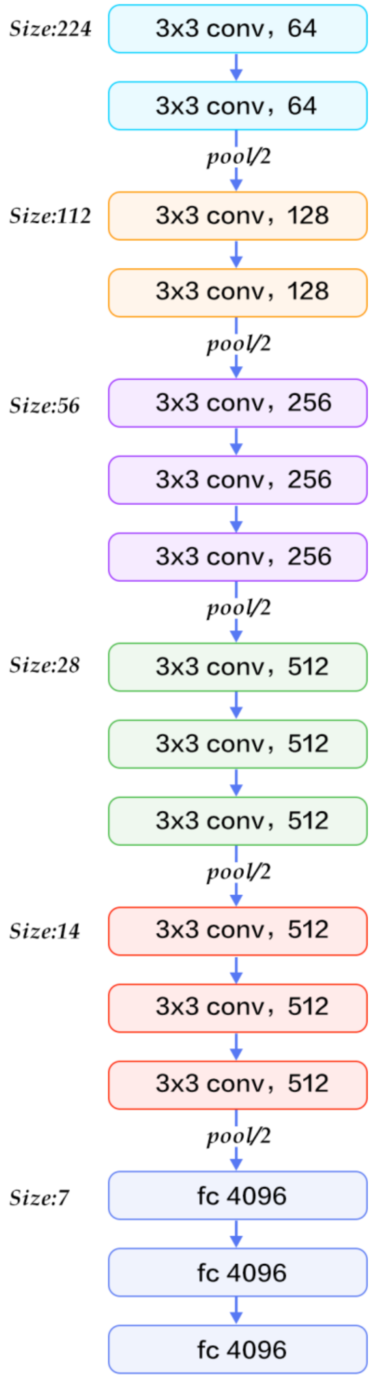


Inception module with dimension reduction

2. Filter Factorization

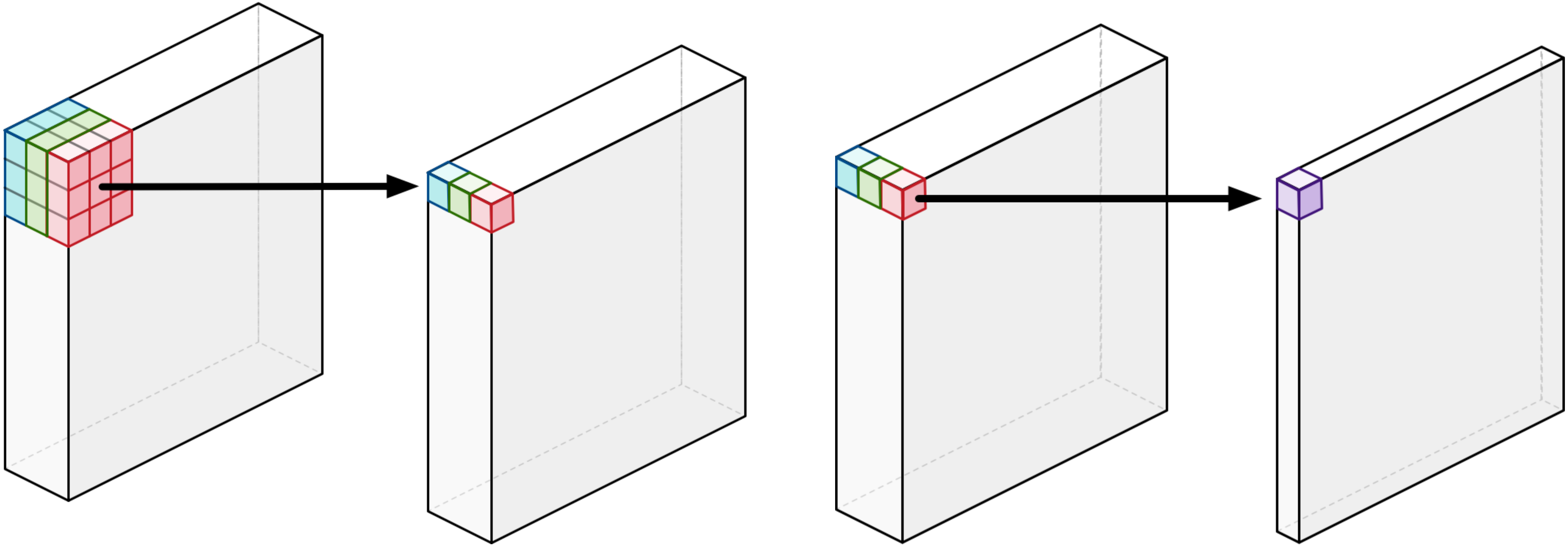
2. Filter Factorization





3. Depthwise Separable Convolution

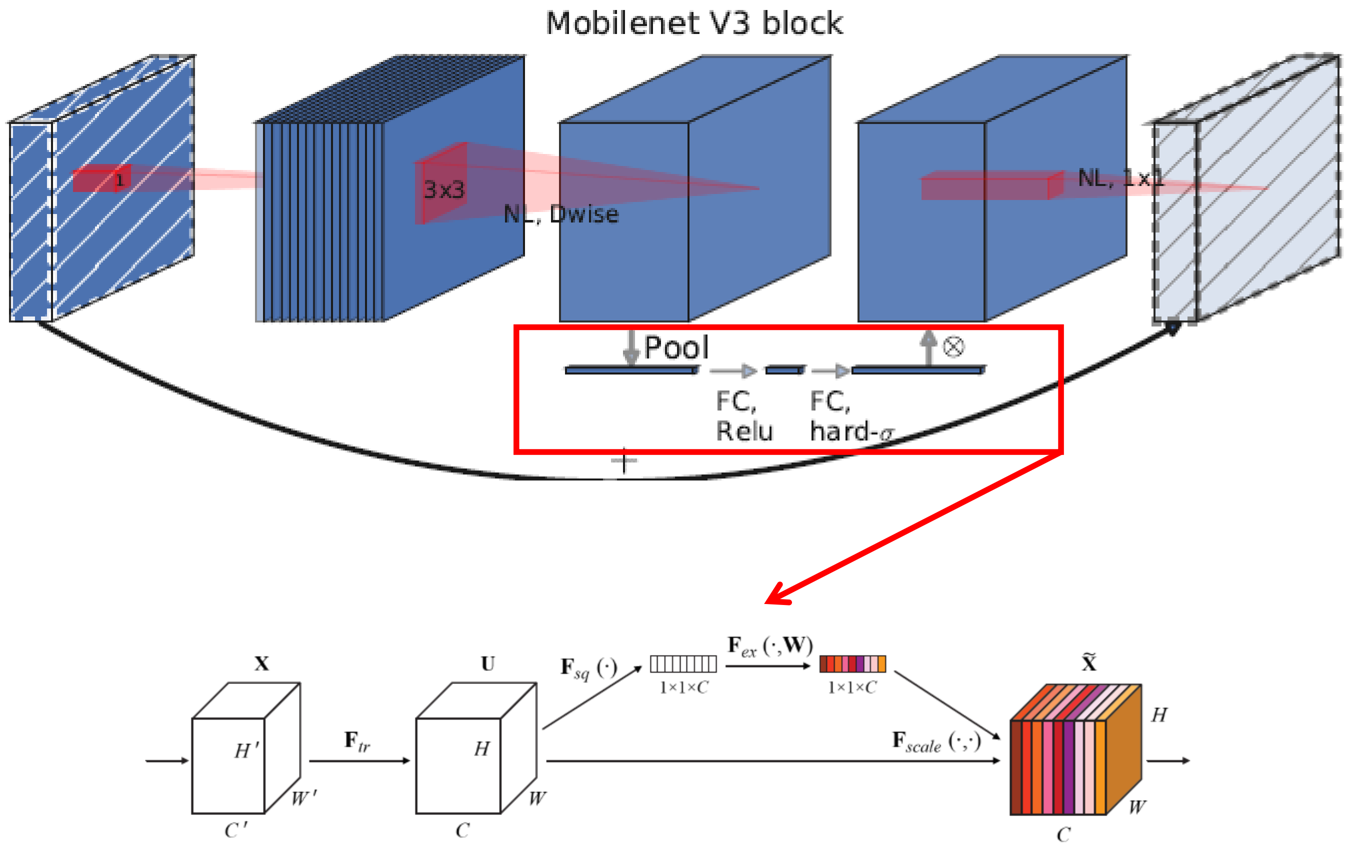
3. Depthwise Separable Convolution



Depthwise convolution

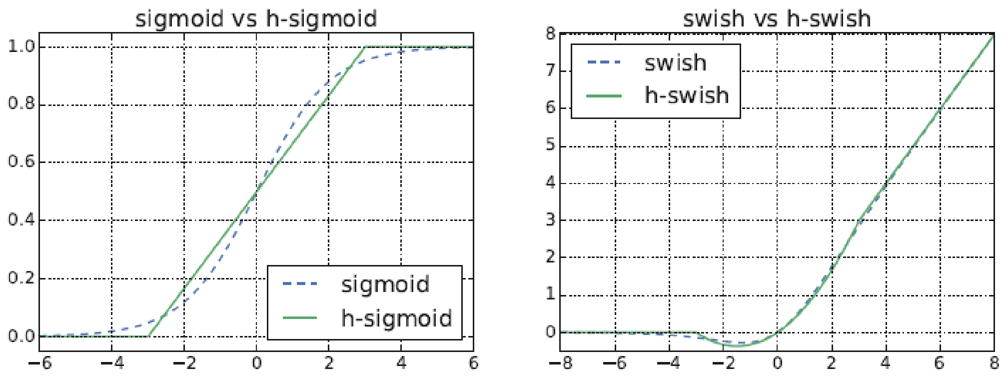
Pointwise convolution

MobileNetV3



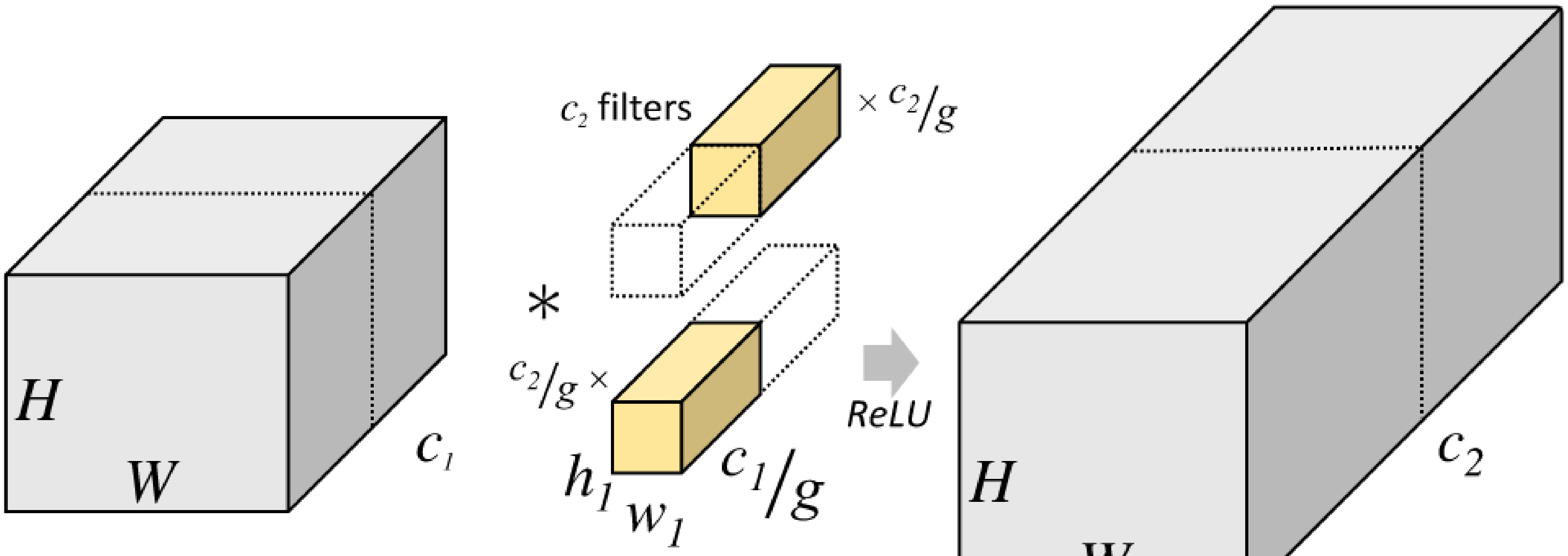
$$\text{swish}\, x = x \cdot \sigma(x)$$

$$\text{h-swish}[x] = x\frac{\mathrm{ReLU6}(x+3)}{6}$$



4. Group Convolution

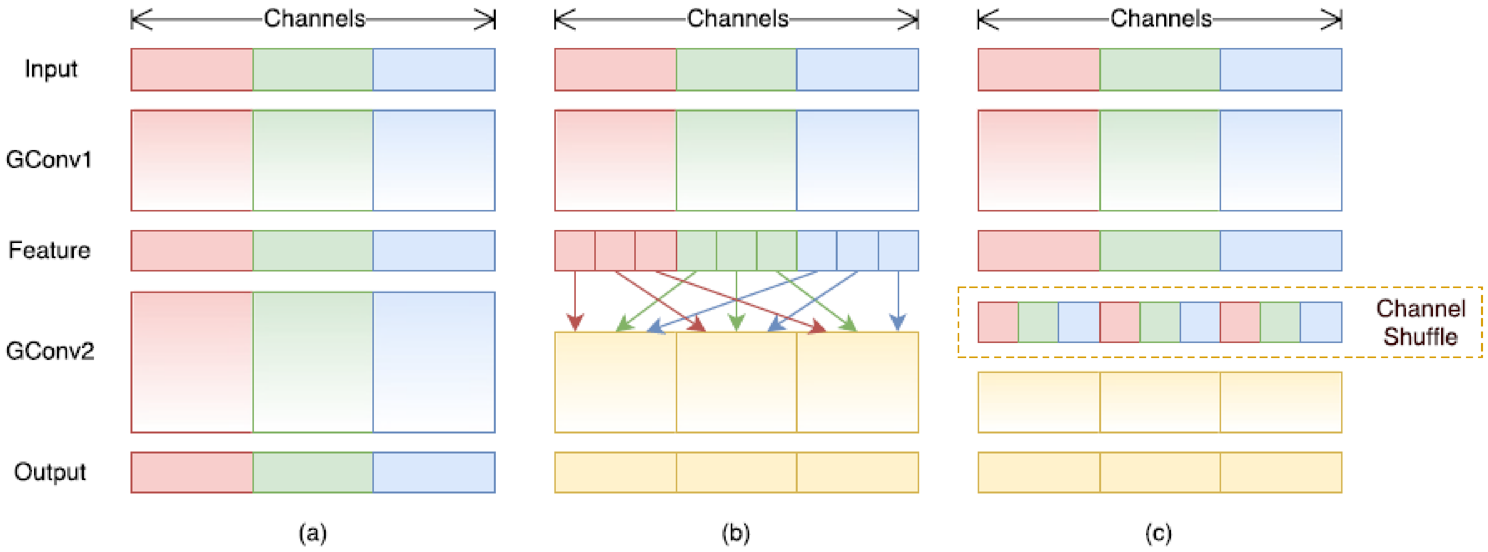
Grouped Convolution



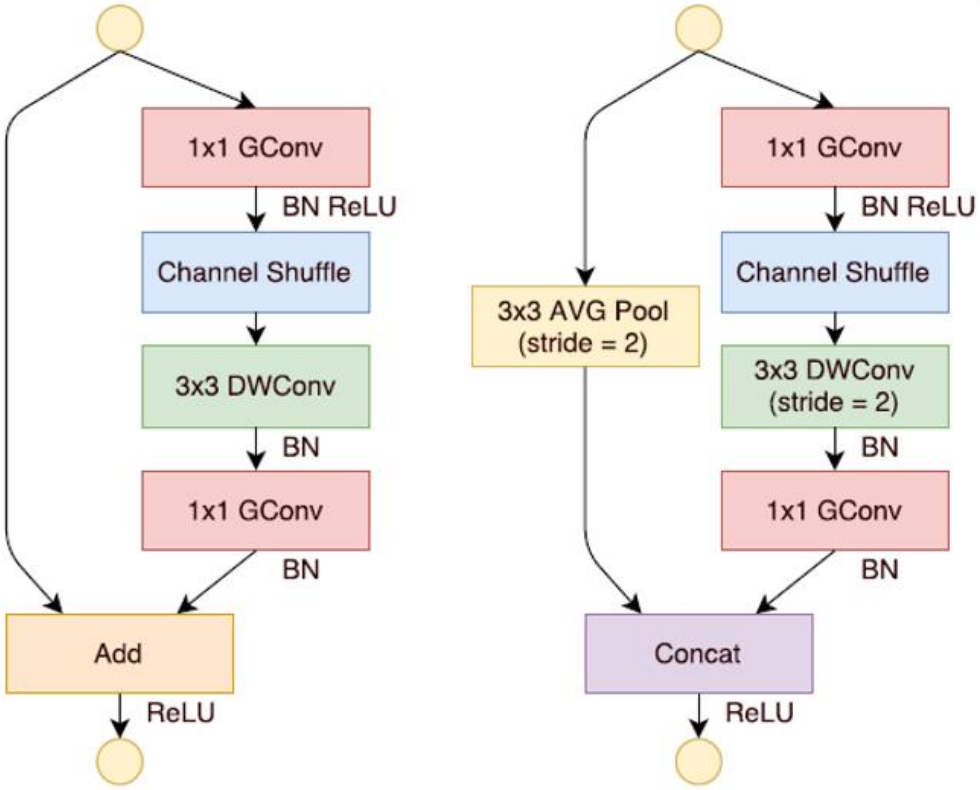
A convolutional layer with 2 filter groups. Note that each of the filters in the grouped convolutional layer is now exactly half the depth, i.e. half the parameters and half the compute as the original filter.

Figure from https://blog.yani.io/filter-group-tutorial/

ShuffleNet

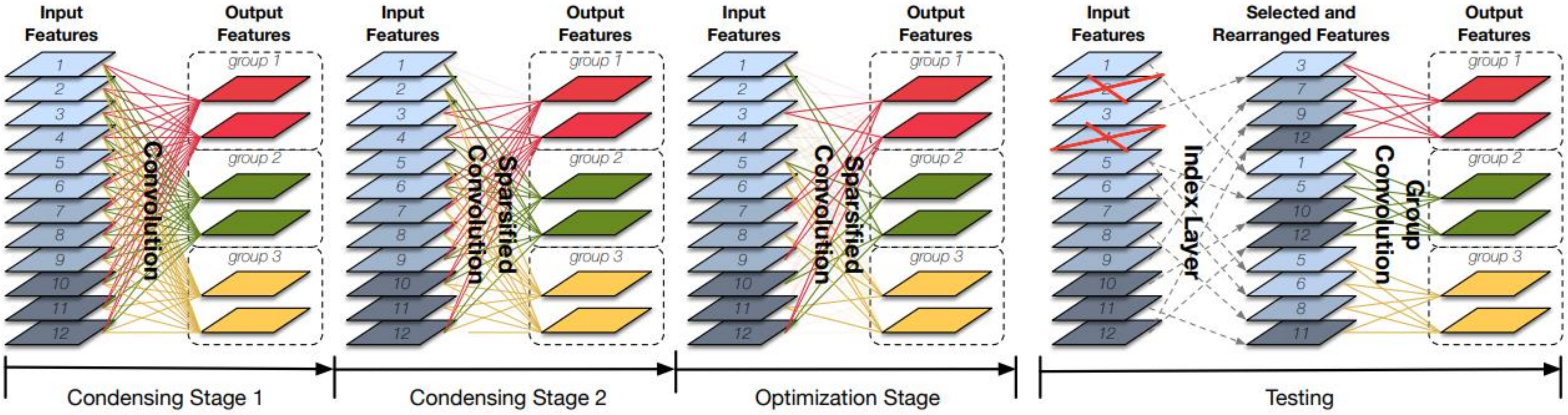


Channels
Input
GConv1
Feature
GConv2
Output
Channel
Shuffle
(a)
(b)
(c)

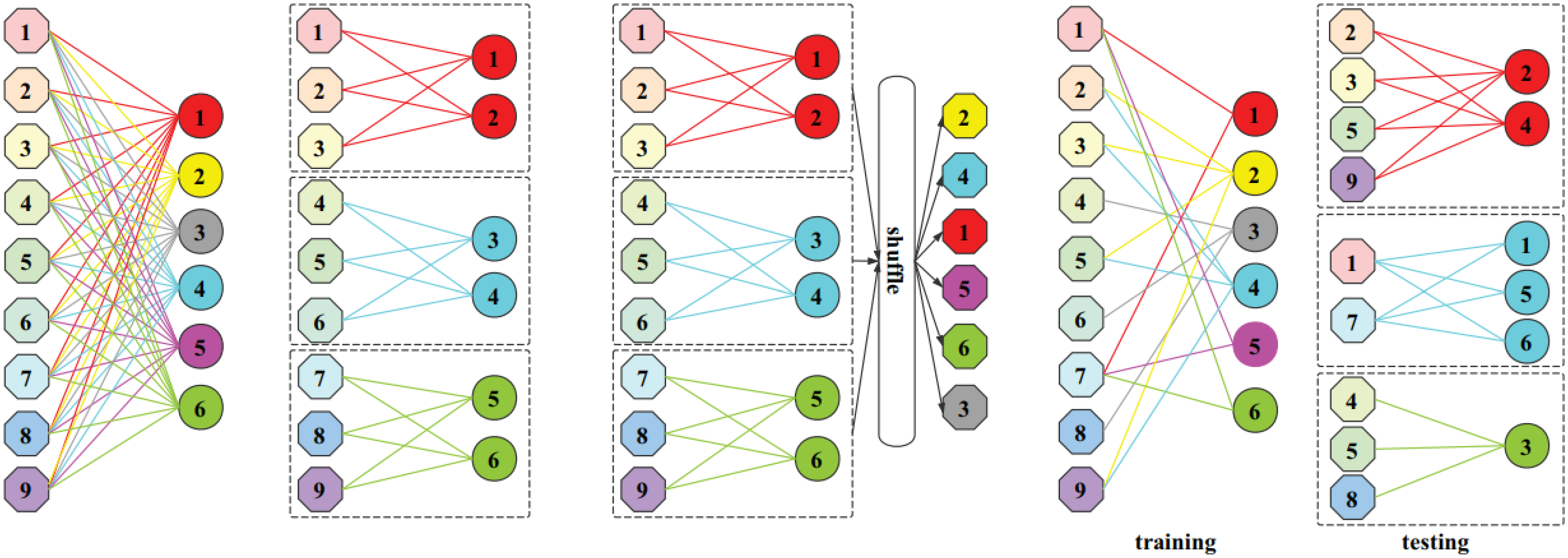


1x1 GConv
BN ReLU
Channel Shuffle
3x3 DWConv
BN
1x1 GConv
BN
Add
ReLU
3x3 AVG Pool
(stride = 2)
1x1 GConv
BN ReLU
Channel Shuffle
3x3 DWConv
(stride = 2)
BN
1x1 GConv
BN
Concat
ReLU

CondenseNet



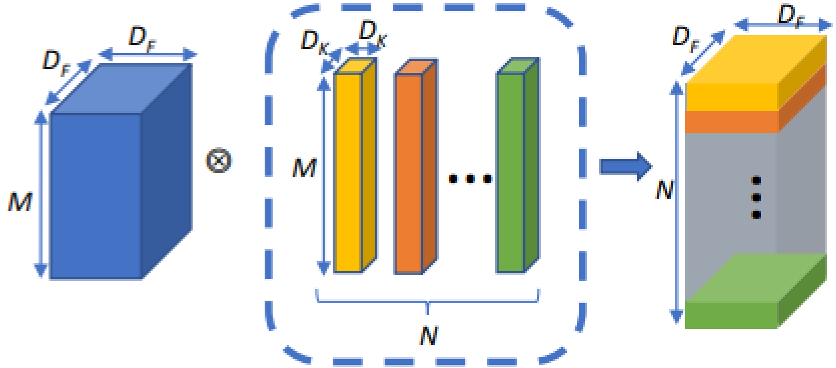
FLGC – CVPR 2019



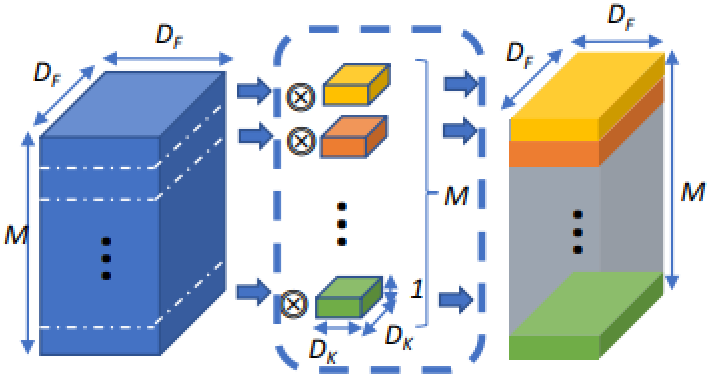
(a) Normal conv (b) Standard group conv (c) ShuffleNet group conv unit (d) FLGC

5. Shift

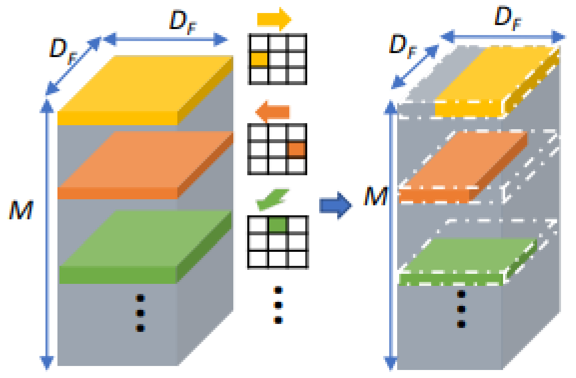
ShiftNet



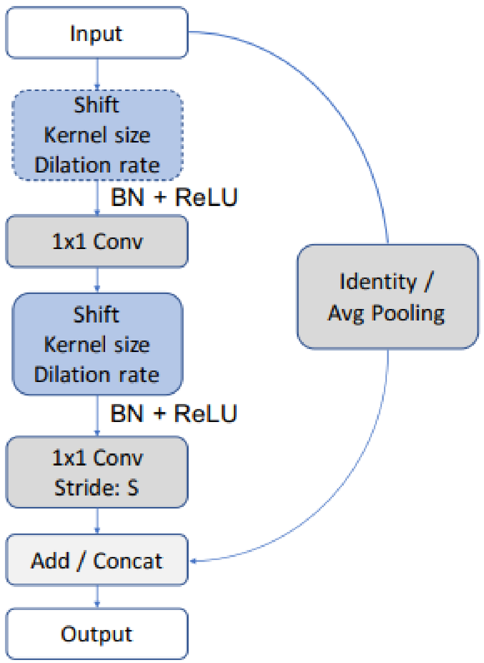
(a) Spatial Convolution



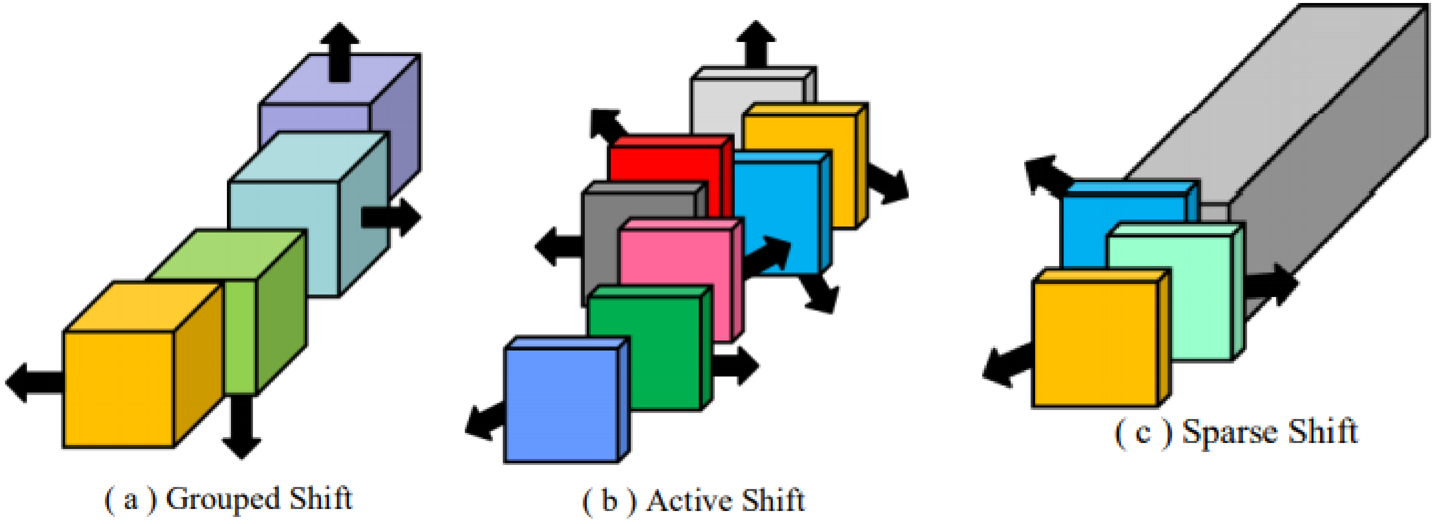
(b) Depth-wise convolution



(c) Shift



All You Need is a Few Shifts – CVPR 2019



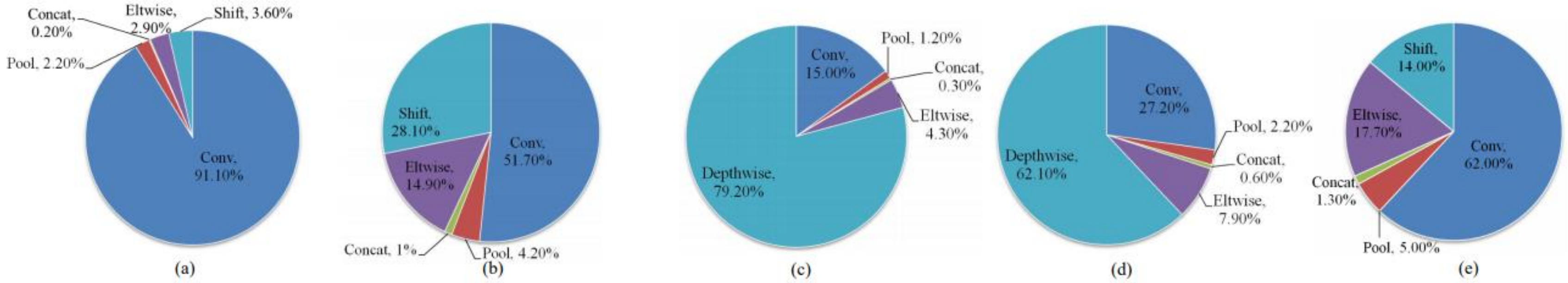
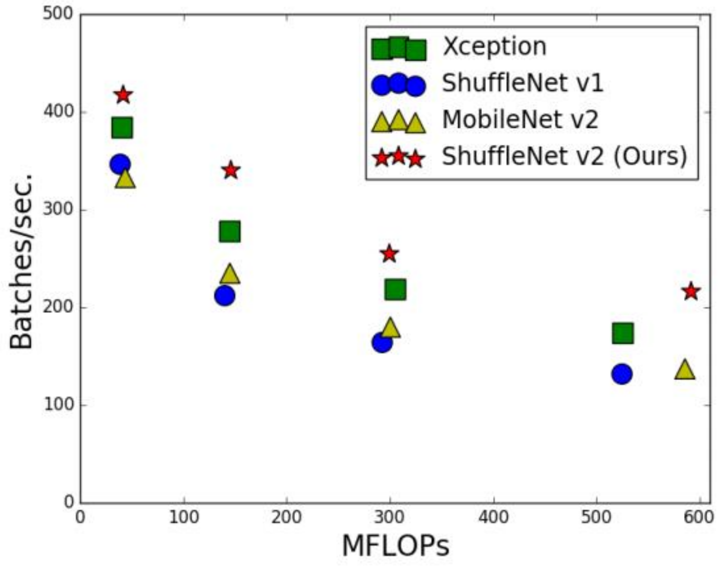
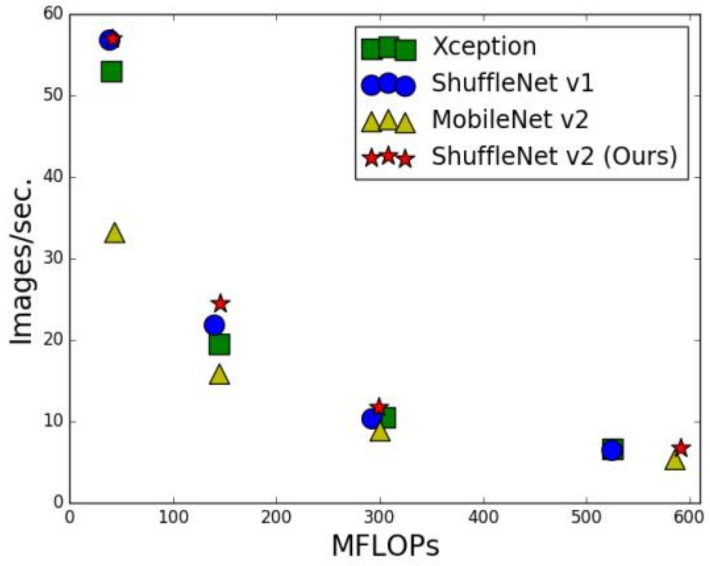


Figure 2. The practical runtime analysis. For clear comparison, both batch-normalization and ReLU layers are neglected since they can be merged into convolutional layer for inference. Also data feeding and preprocessing time are not considered here. Results are achieved under Caffe with mini-batch 32. They are averaged from 100 runs. (a) ShiftNet-A [37] on CPU (Intel Xeon E5-2650, atlas). (b) ShiftNet-A on GPU (TITAN X Pascal, CUDA8 and cuDNN5). (c) Shift layers in ShiftNet-A are replaced by depthwise separable convolution layers. (d) Depthwise separable convolution layers with kernel size 5 are replaced by the ones with kernel size 3. (e) ShiftNet-A with 80% shift sparsity on GPU (Shift sparsity denotes the ratio of unshifted feature maps).

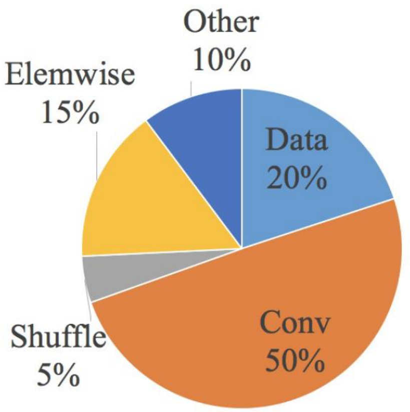
6. Use Direct Metric



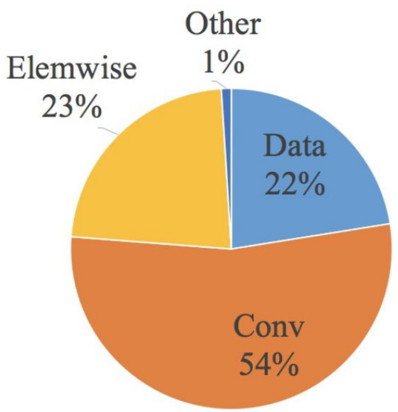
(c) GPU



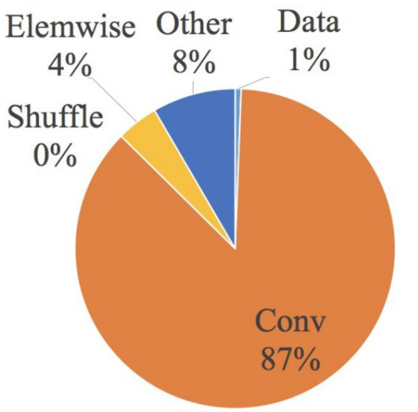
(d) ARM



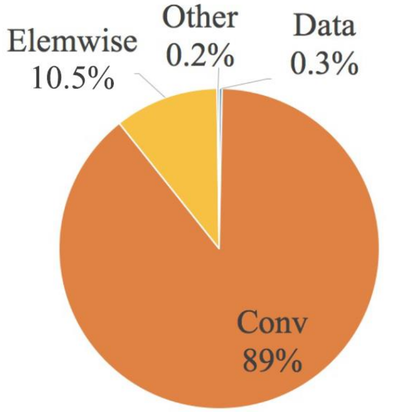
ShuffleNet V1 on GPU



MobileNet V2 on GPU



ShuffleNet V1 on ARM



MobileNet V2 on ARM

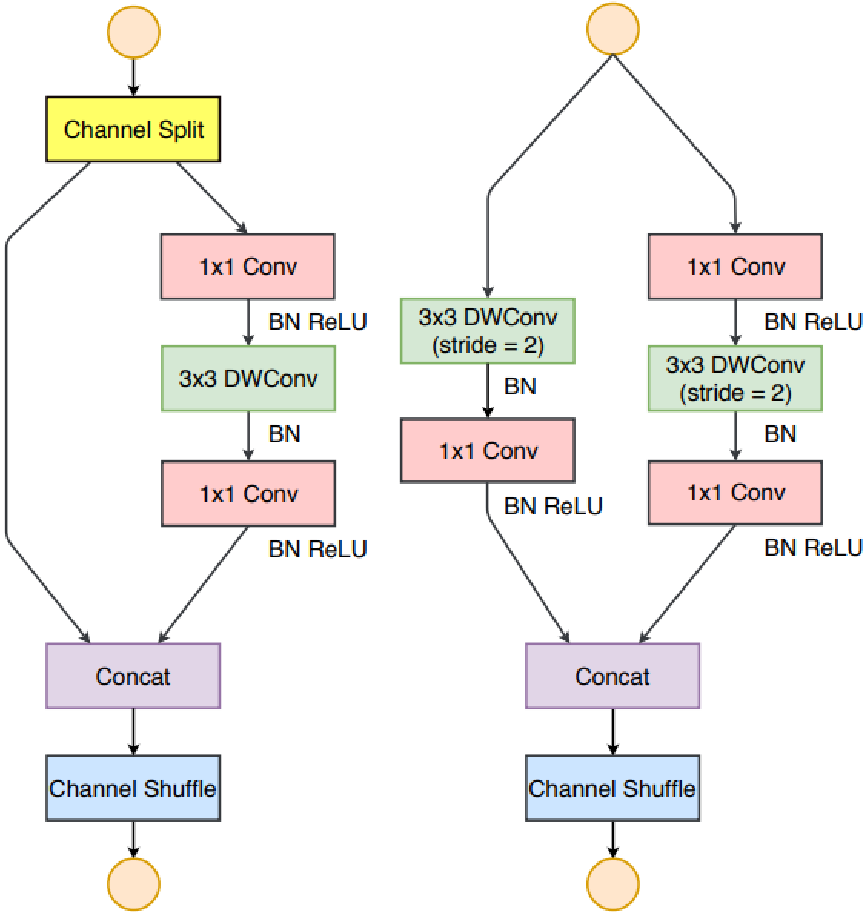
ShuffleNet V2 – Guide 1

- **Equal channel width minimizes memory access cost (MAC).**
- Let h and w be the spatial size of the feature map, the FLOPs of the 1 x 1 convolution is $B = hwc_1c_2$.
- The memory access cost (MAC), or the number of memory access operations, is $MAC = hw(c_1 + c_2) + c_1c_2$.
- MAC has a lower bound given by FLOPs. It reaches the lower bound when the numbers of input and output channels are equal.

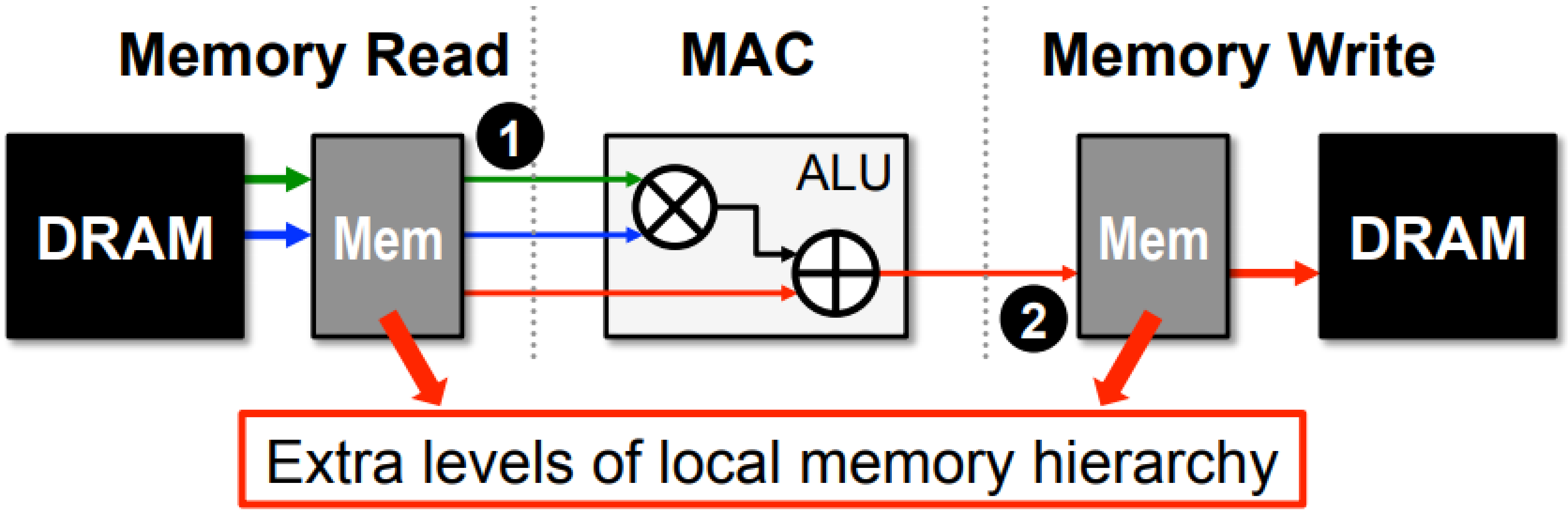
$$MAC \geq 2\sqrt{hwB} + \frac{B}{hw}$$

ShuffleNet V2

1. Use "balanced" convolutions (equal channel width).
2. Be aware of the cost of using group convolution.
3. Reduce the degree of fragmentation.
4. Reduce element-wise operations.



Most DNN Accelerators

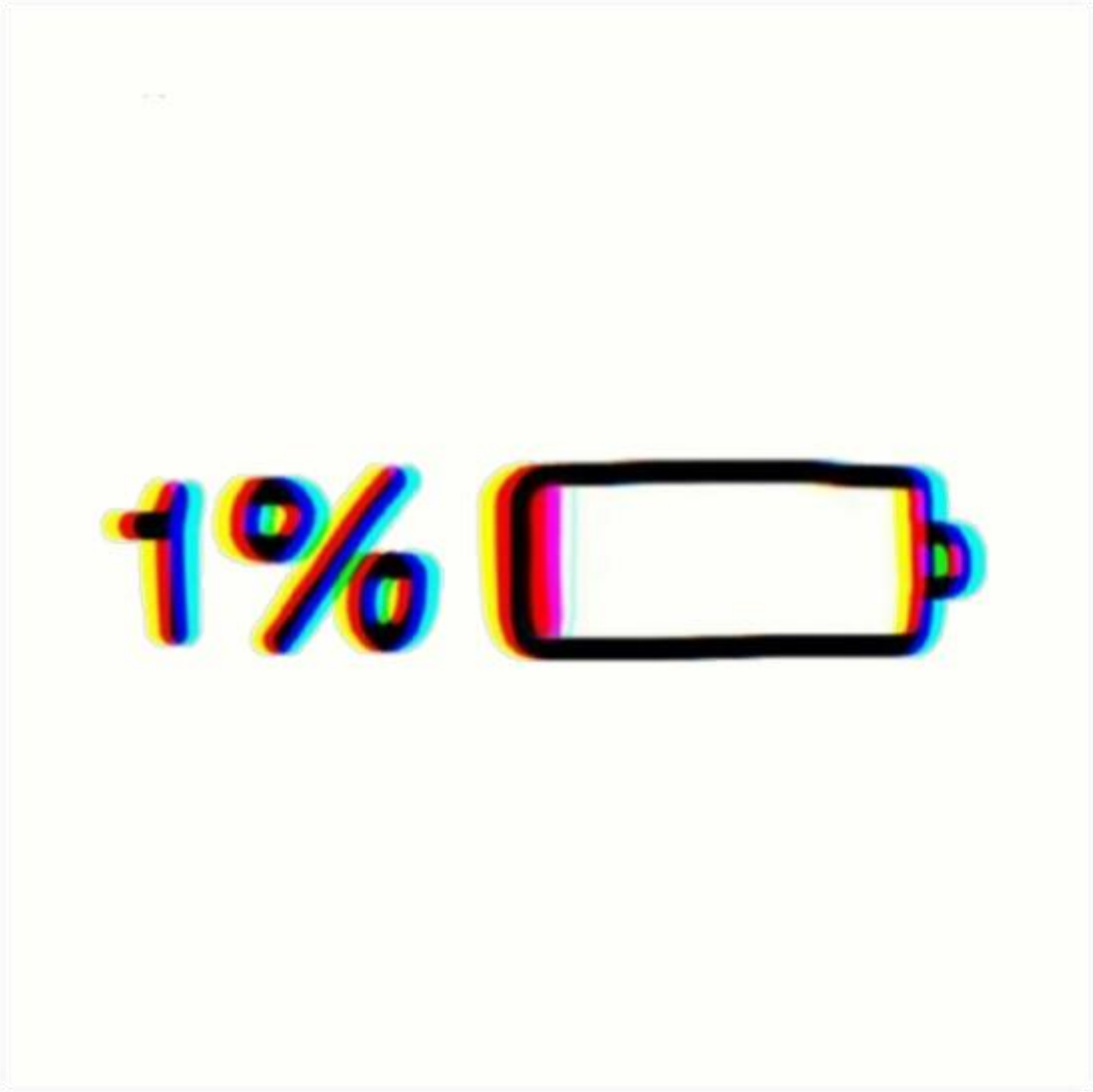


Opportunities: 1 **data reuse** 2 **local accumulation**

1 Can reduce DRAM reads of filter/fmap by up to **500×**
2 Partial sum accumulation does **NOT** have to access DRAM

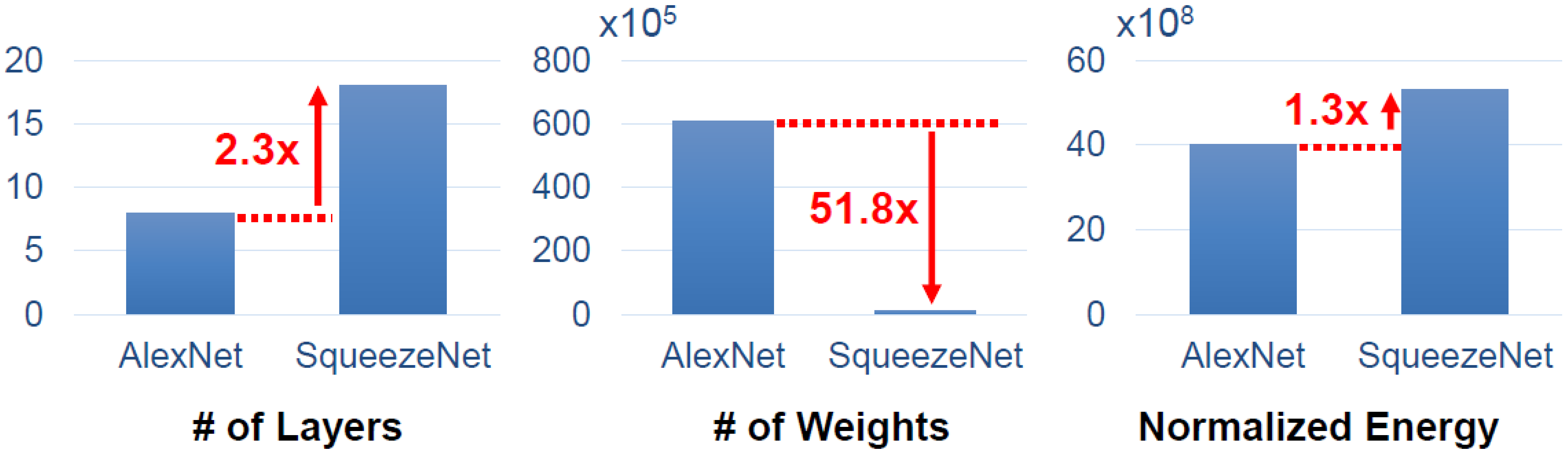
- Example: DRAM access in AlexNet can be reduced from **2896M** to **61M** (best case)

Energy is very Important



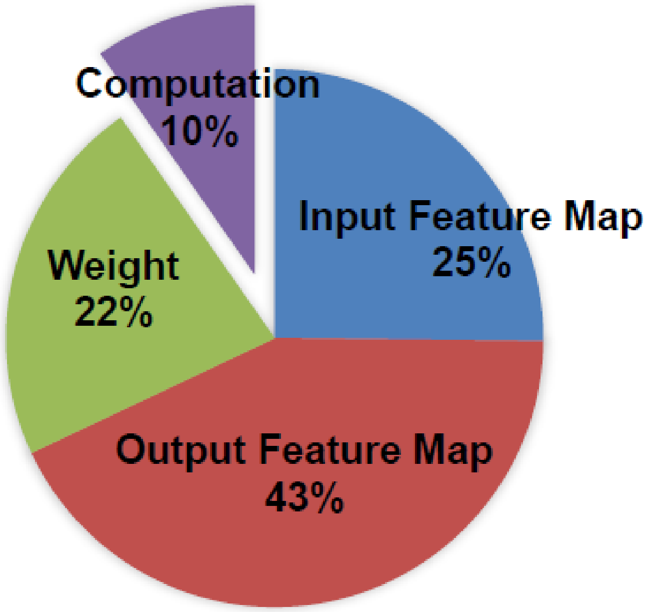
Energy?!

Deeper CNNs with fewer weights do not necessarily consume less energy than shallower CNNs with more weights

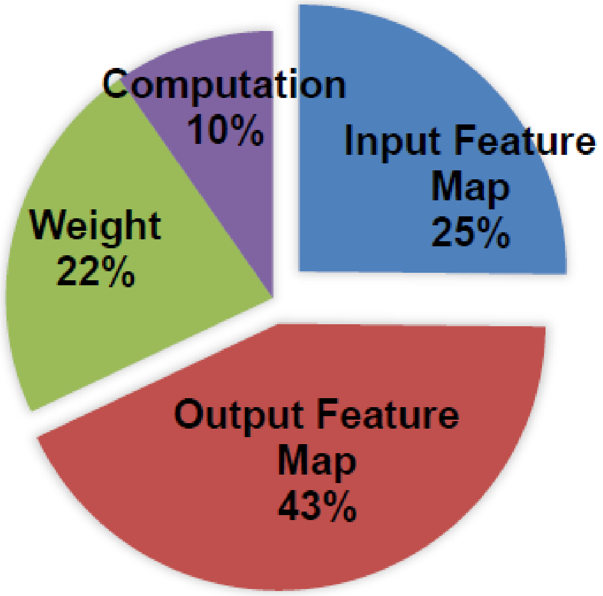


Key Insights

- Data movement is more expensive than computation
- Feature maps need to be taken into account



Computation 10%

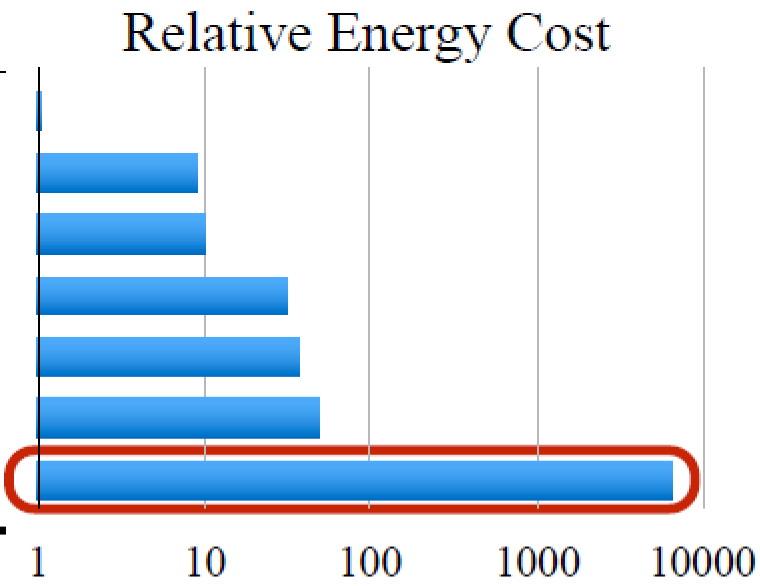


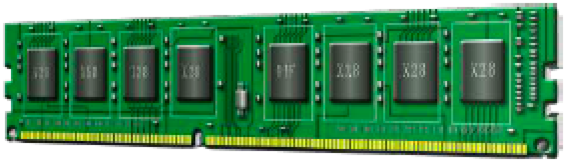
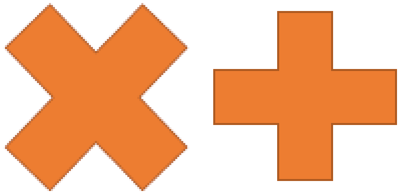
Feature Map 68%

GoogLeNet Energy Breakdown

Where is the Energy Consumed?

Operation	Energy [pJ]
32 bit int ADD	0.1
32 bit float ADD	0.9
32 bit Register File	1
32 bit int MULT	3.1
32 bit float MULT	3.7
32 bit SRAM Cache	5
32 bit DRAM Memory	**640**



1  = 1000 

This image is in the public domain

EfficientNet

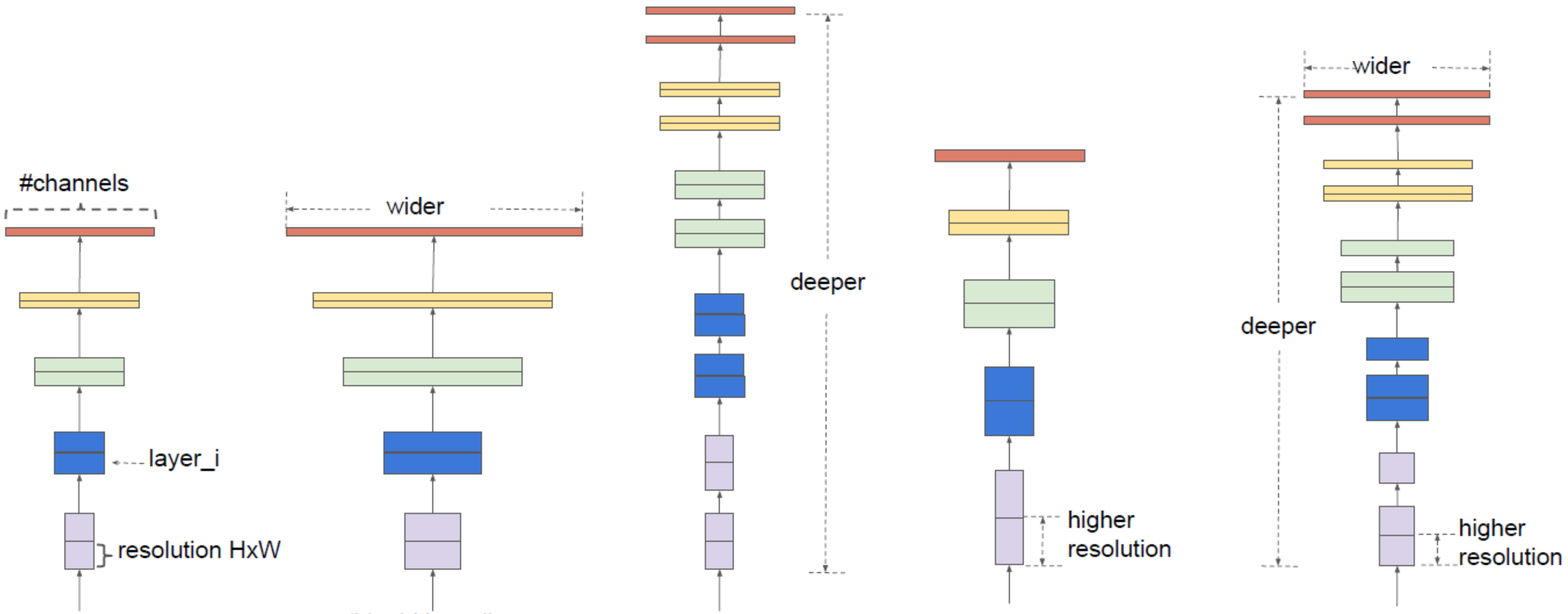
$$\text{depth: } d = \alpha^{\phi}$$

$$\text{width: } w = \beta^{\phi}$$

$$\text{resolution: } r = \gamma^{\phi}$$

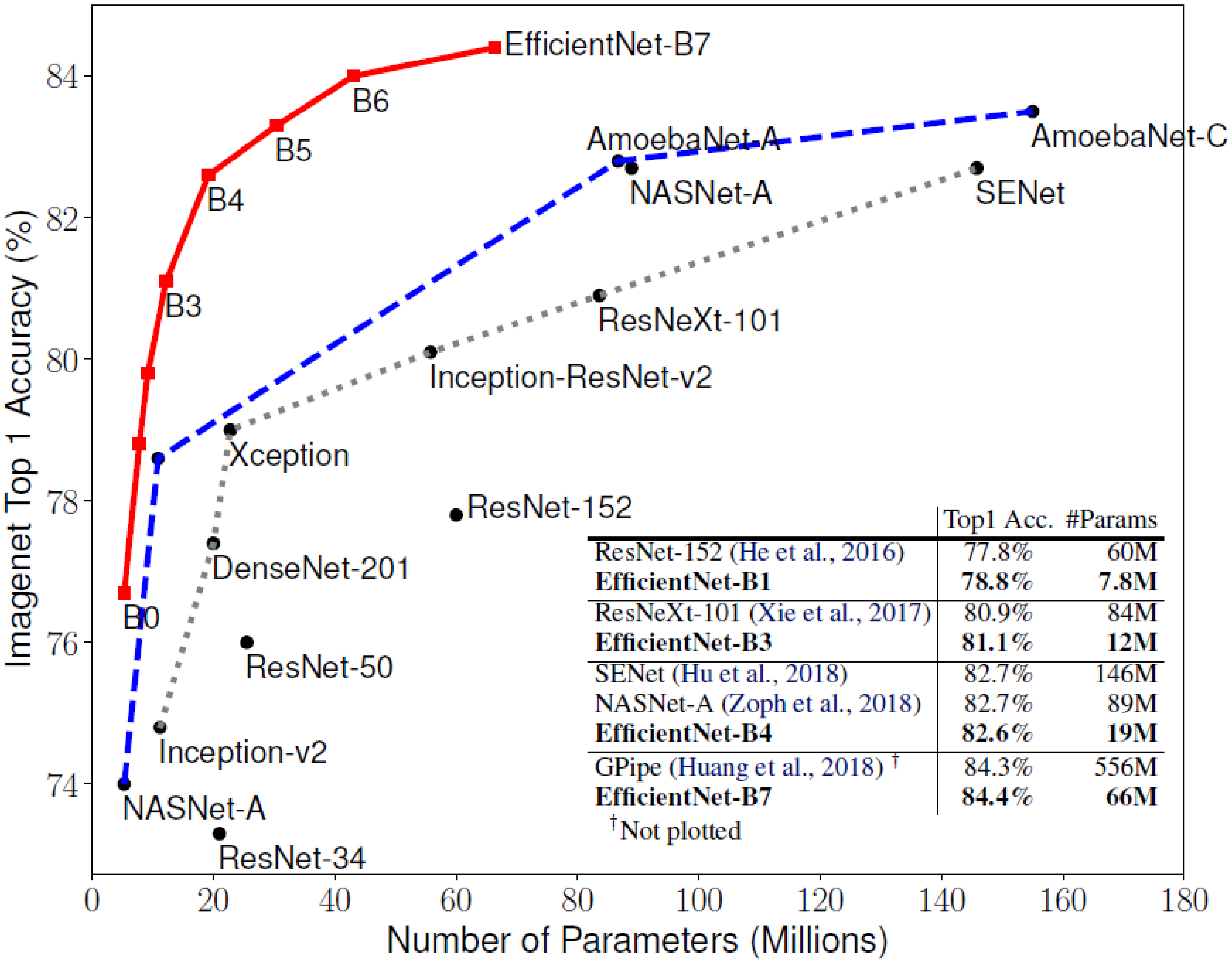
$$\text{s.t. } \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$$

$$\alpha \geq 1, \beta \geq 1, \gamma \geq 1$$



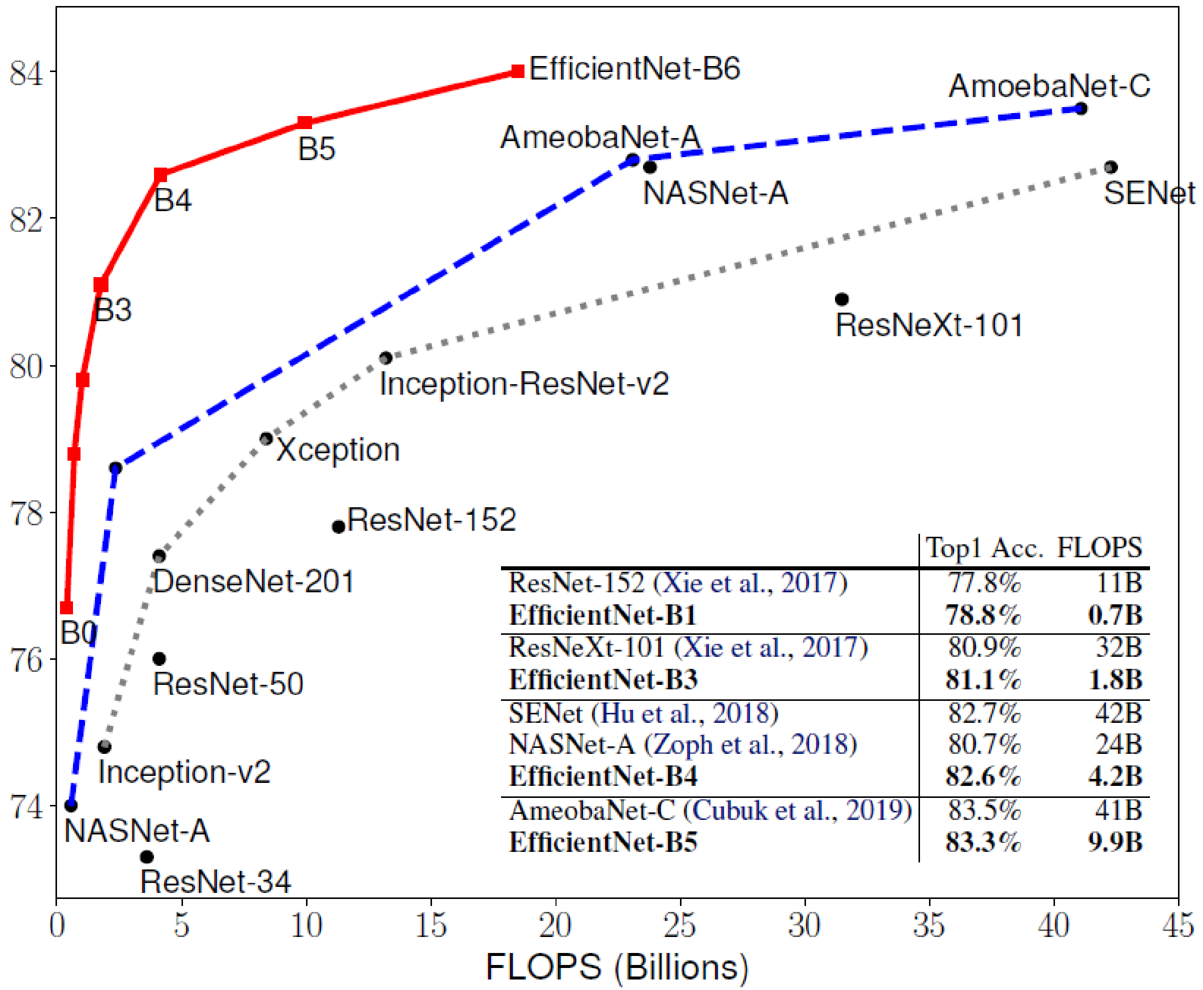
(a) baseline (b) width scaling (c) depth scaling (d) resolution scaling (e) compound scaling

EfficientNet



	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**78.8%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.1%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**82.6%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.4%**	**66M**

†Not plotted



	Top1 Acc.	FLOPS
ResNet-152 (Xie et al., 2017)	77.8%	11B
EfficientNet-B1	**78.8%**	**0.7B**
ResNeXt-101 (Xie et al., 2017)	80.9%	32B
EfficientNet-B3	**81.1%**	**1.8B**
SENet (Hu et al., 2018)	82.7%	42B
NASNet-A (Zoph et al., 2018)	80.7%	24B
EfficientNet-B4	**82.6%**	**4.2B**
AmeobaNet-C (Cubuk et al., 2019)	83.5%	41B
EfficientNet-B5	**83.3%**	**9.9B**

Thank you