

Computer Vision; Image Classification; Small Networks



[YouTube Playlist](#)

Maziar Raissi

Assistant Professor

Department of Applied Mathematics

University of Colorado Boulder

maziar.raissi@colorado.edu



Boulder



[YouTube Playlist](#)

Distilling the Knowledge in a Neural Network

$z_i \rightarrow$ logits computed for each class

$q_i \rightarrow$ probability (using softmax)

$$q_i = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}$$

$T \rightarrow$ temperature (e.g., $T = 1$)

Higher $T \Rightarrow$ softer prob. distribution over classes

Knowledge is transferred to the distilled model by training it on a transfer set and using a soft target distribution for each case in the transfer set that is produced by using the cumbersome model with a high temperature in its softmax. The same high temperature is used when training the distilled model, but after it has been trained it uses a temperature of 1.

Matching logits is a special case of distillation

$$C = -\sum_i p_i \log q_i \rightarrow$$
 cross-entropy loss

$v_i \rightarrow$ logits of the cumbersome model

$$\frac{\partial C}{\partial z_i} = \frac{1}{T}(q_i - p_i) = \frac{1}{T} \left(\frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)} - \frac{\exp(v_i/T)}{\sum_j \exp(v_j/T)} \right)$$

If the temperature is high compared with the magnitude of the logits

$$\frac{\partial C}{\partial z_i} \approx \frac{1}{T} \left(\frac{1 + z_i/T}{N + \sum_j z_j/T} - \frac{1 + v_i/T}{N + \sum_j v_j/T} \right)$$

$$\frac{\partial C}{\partial z_i} \approx \frac{1}{NT^2} (z_i - v_i)$$

assume $\sum_j z_j = \sum_j v_j = 0$
equivalent to minimizing $1/2(z_i - v_i)^2$

| Cumbersome Net | Smaller Net | Distilled Net |
|----------------|-------------|---------------|
| 67 | 146 | 74 |

MNIST: Number of Errors

Performing inference with ensembles of specialists

$$KL(\mathbf{p}^g, \mathbf{q}) + \sum_{m \in A_k} KL(\mathbf{p}^m, \mathbf{q})$$

$p^g \rightarrow$ generalist model

$p^m \rightarrow$ specialist model

$A_k \rightarrow$ active set of specialists

classes k : $n = 1$ most probable classes (for each test case) according to the generalist model

A_k : All specialist models m whose special subset of confusable classes S^m has a non-empty intersection with k

confusable subset of the classes (like different types of mushroom)

JFT is an internal Google dataset that has 100 million labeled images with 15,000 labels

| |
|---|
| JFT 1: Tea party; Easter; Bridal shower; Baby shower; Easter Bunny; ... |
| JFT 2: Bridge; Cable-stayed bridge; Suspension bridge; Viaduct; Chimney; ... |
| JFT 3: Toyota Corolla E100; Opel Signum; Opel Astra; Mazda Familia; ... |

| System | Conditional Test Accuracy | Test Accuracy |
|------------------------|---------------------------|---------------|
| Baseline | 43.1% | 25.0% |
| + 61 Specialist models | 45.9% | 26.1% |

Experiments on speech recognition

$$\boldsymbol{\theta} = \arg \max_{\boldsymbol{\theta}'} P(h_t | \mathbf{s}_t; \boldsymbol{\theta}')$$

$P \rightarrow$ acoustic model

$s_t \rightarrow$ acoustic observations

$P(h_t | \mathbf{s}_t; \boldsymbol{\theta}')$ \rightarrow prob. of the “correct” HMM state h_t

h_t is determined by a forced alignment

with the correct sequence of words

Word Error Rate (WER)

| System | Test Frame Accuracy | WER |
|------------------------|---------------------|-------|
| Baseline | 58.9% | 10.9% |
| 10xEnsemble | 61.1% | 10.7% |
| Distilled Single model | 60.8% | 10.7% |

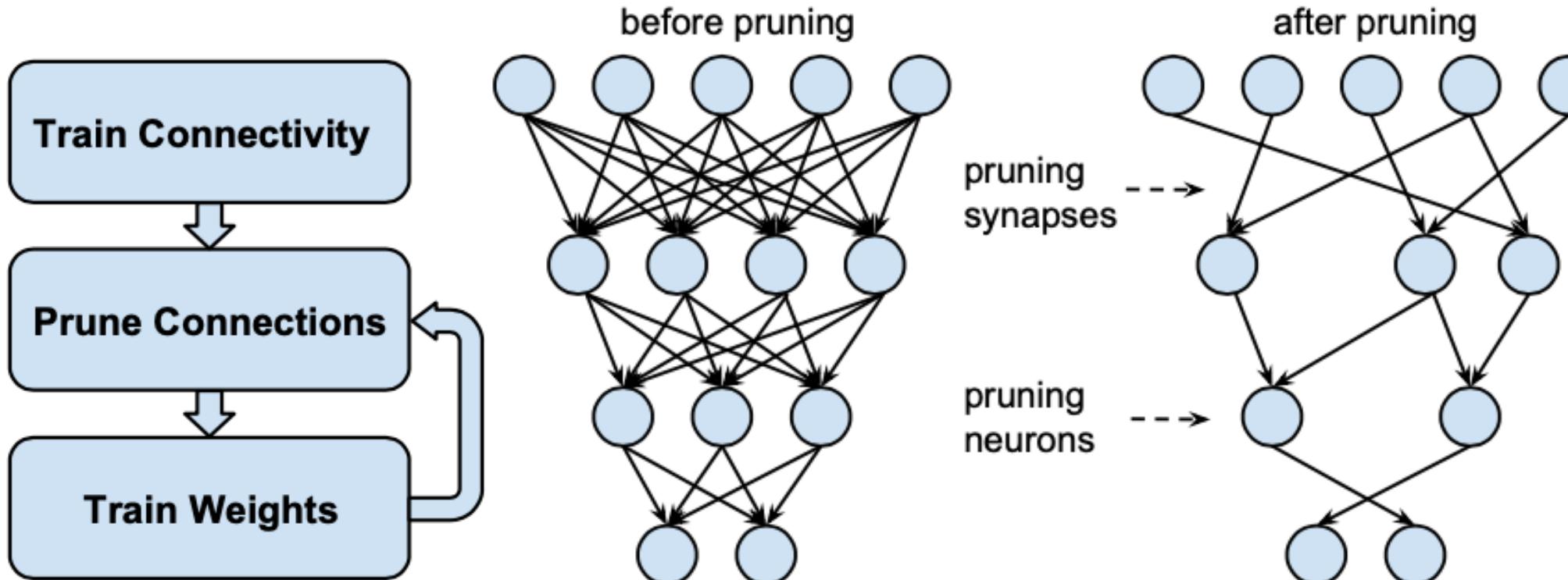


Boulder

Learning both Weights and Connections for Efficient Neural Networks



[YouTube Video](#)



Dropout Ratio Adjustment

$D_o \rightarrow$ original dropout rate

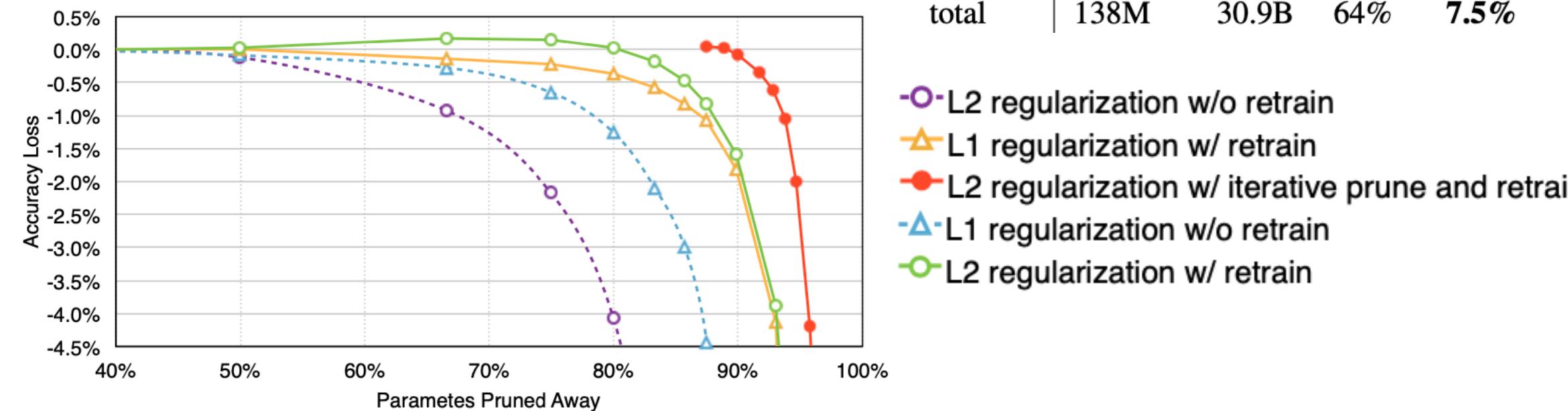
$D_r \rightarrow$ dropout rate during retraining

$C_{ir} \rightarrow$ number of connections in layer i for the network after retraining

$C_{io} \rightarrow$ number of connections in layer i for the original network

$$D_r = D_o \sqrt{\frac{C_{ir}}{C_{io}}}$$

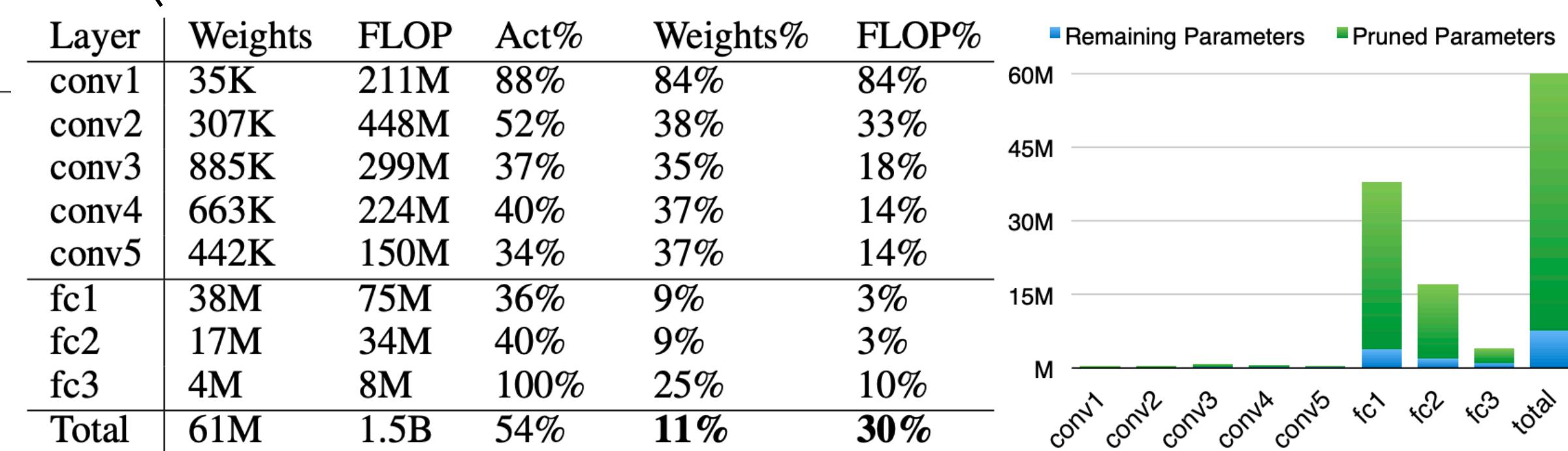
Since C_{io} & C_{ir} vary quadratically with the number of neurons!



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

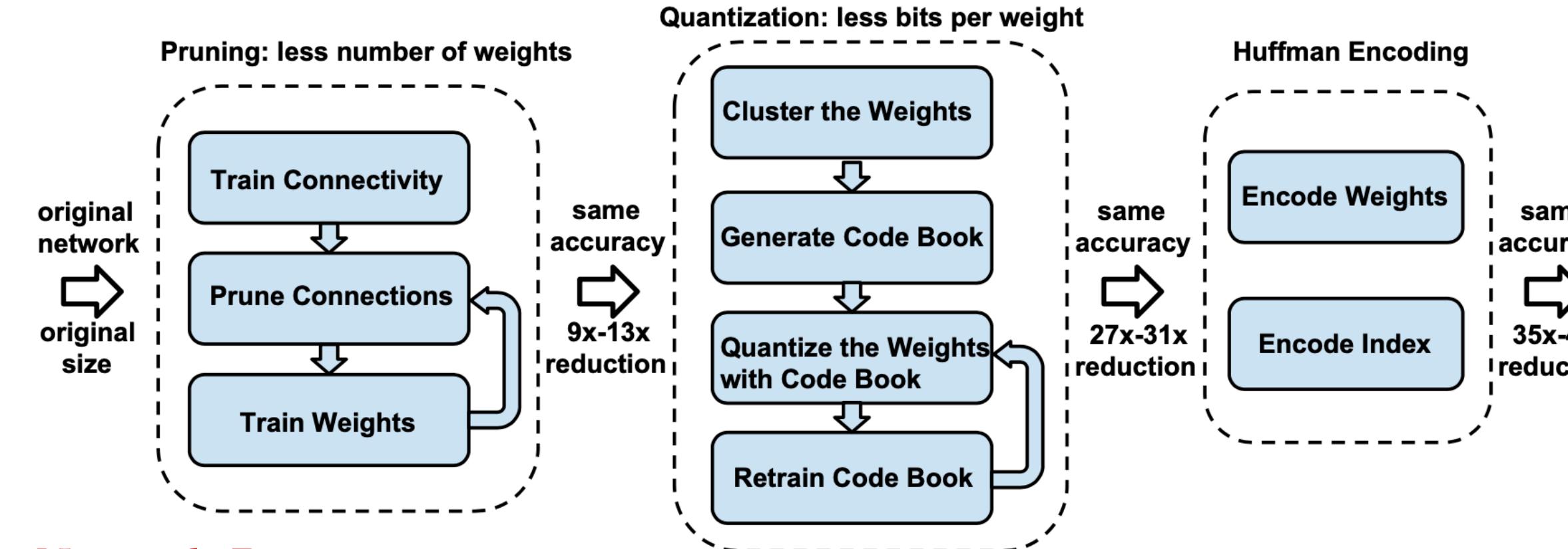
| Layer | Weights | FLOP | Act% | Weights% | FLOP% |
|-------|---------|------|------|----------|-------|
| fc1 | 235K | 470K | 38% | 8% | 8% |
| fc2 | 30K | 60K | 65% | 9% | 4% |
| fc3 | 1K | 2K | 100% | 26% | 17% |
| Total | 266K | 532K | 46% | 8% | 8% |

| 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× |





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


[YouTube Playlist](#)


Network Pruning

All connections with weights below a threshold are removed from the network

Compressed Sparse Row (CSR)
Compressed Sparse Column (CSC)

} → $2a + n + 1$ numbers
 number of non-zero elements
 number of rows or columns

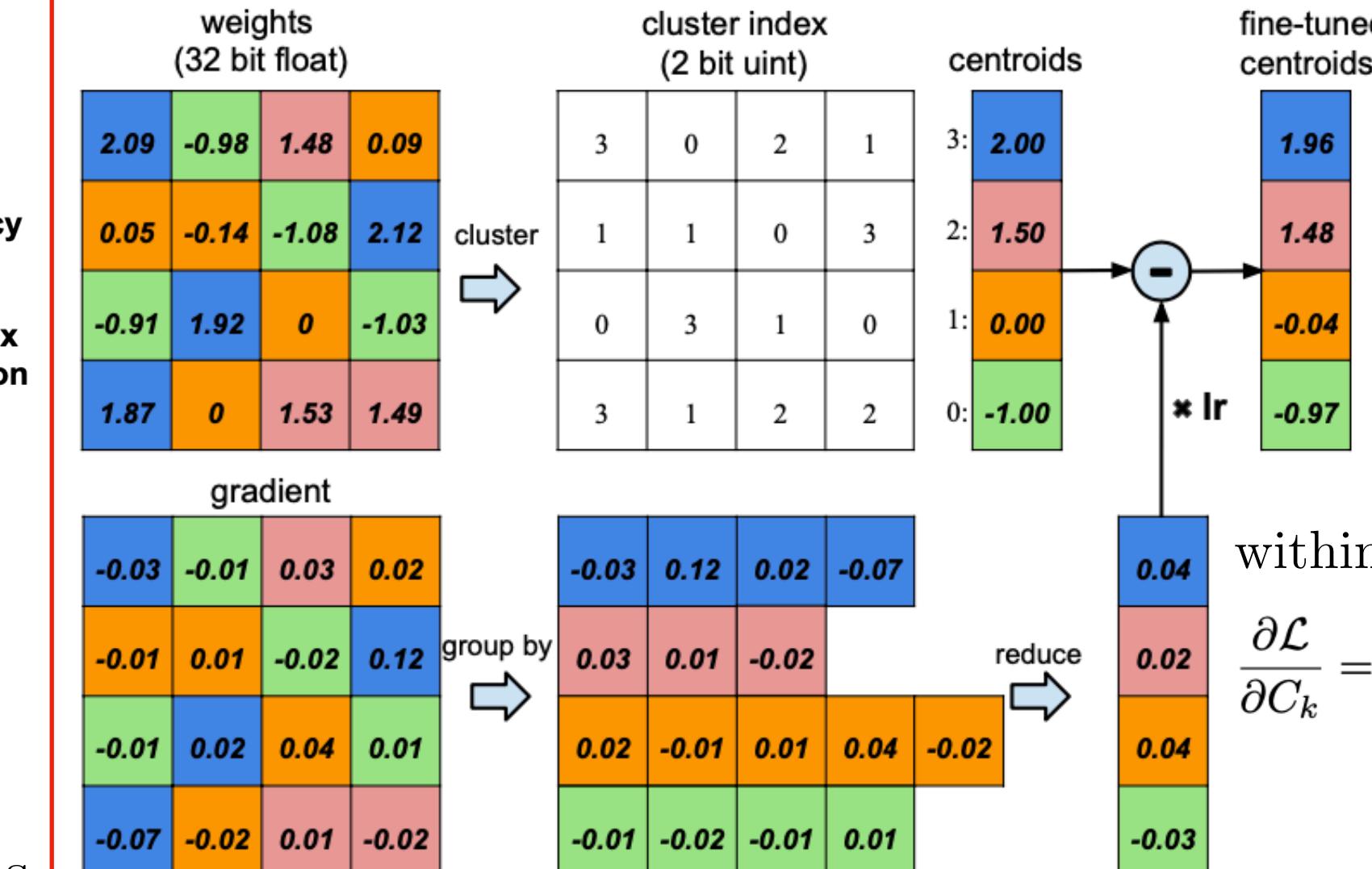
To compress further, store the index difference instead of the absolute position, and encode this difference in 8 bits for conv layer and 5 bits for fc layer.

| idx | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|-------|---|-----|---|---|-----|---|---|---|---|---|----|----|----|----|----|-----|
| diff | | 1 | | | 3 | | | | | | | | 8 | | | 3 |
| value | | 3.4 | | | 0.9 | | | | | | | | 0 | | | 1.7 |

Span Exceeds $8=2^3$

Filler Zero

Trained Quantization & Weight Sharing



Quantize pruned AlexNet to 8-bits (256 shared weights) for each CONV layers, and 5-bits (32 shared weights) for each FC layer.

$$r = \frac{nb}{n \log_2(k) + kb} = \frac{16 \times 32}{16 \times 2 + 4 \times 32} = 3.2$$

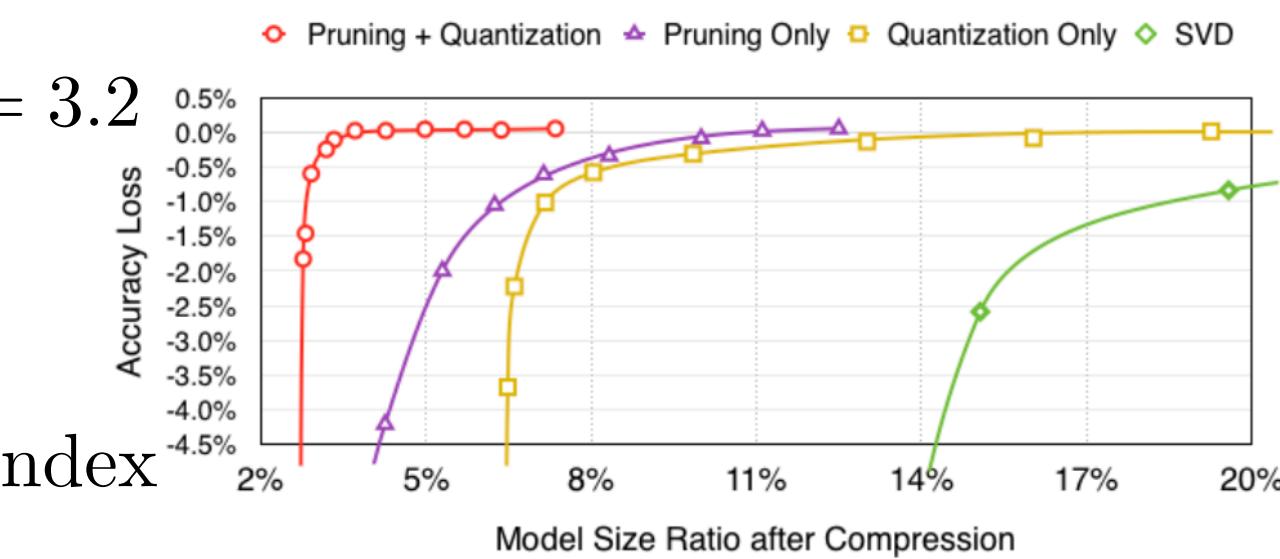
compression rate

$k \rightarrow$ number of clusters

$\log_2(k) \rightarrow$ required bits to encode the index

$n \rightarrow$ number of bits in a network

$b \rightarrow$ bits representing each connection



k-means clustering

$W = \{w_1, \dots, w_n\}$
 $\underbrace{\quad}_{n \text{ original weights}}$

$C = \{c_1, \dots, c_k\}$
 $\underbrace{\quad}_{k \ll n \text{ clusters}}$

$$\arg \min_C \sum_{i=1}^k \sum_{w \in c_i} |w - c_i|^2$$

within-cluster sum of squares (WCSS)

$$\frac{\partial \mathcal{L}}{\partial C_k} = \sum_{i,j} \frac{\partial \mathcal{L}}{\partial W_{ij}} \frac{\partial W_{ij}}{\partial C_k} = \sum_{i,j} \frac{\partial \mathcal{L}}{\partial W_{ij}} \mathbf{1}(I_{ij} = k)$$

Huffman Coding

More common symbols are represented with fewer bits

SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size


[YouTube Playlist](#)

$$X \in \mathbb{R}^{h \times w \times c} \quad Y \in \mathbb{R}^{h \times w \times f}$$

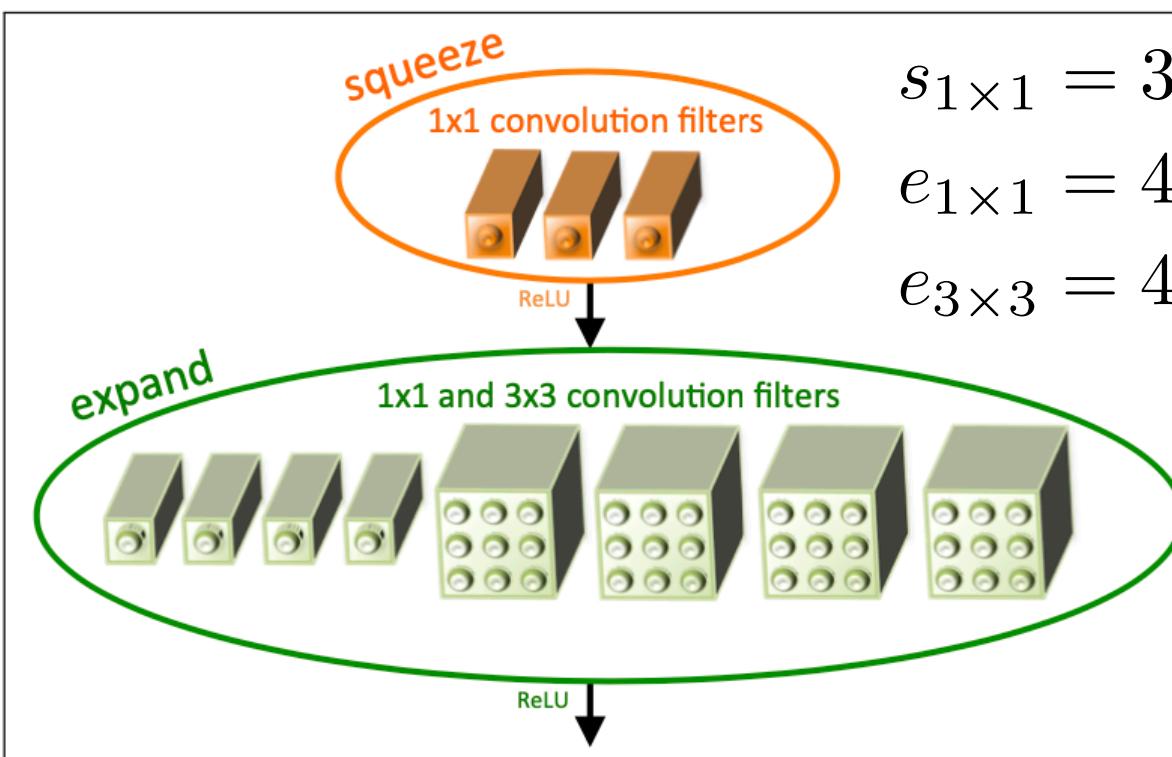
$\overbrace{\quad\quad\quad}^W \in \mathbb{R}^{k \times k \times c \times f}$

Strategy 1: Replace 3×3 filters with 1×1 filters

Strategy 2: Decrease the number of input channels to 3×3 filters

Strategy 3: Downsample late in the network so that convolution layers have large activation maps

The fire module



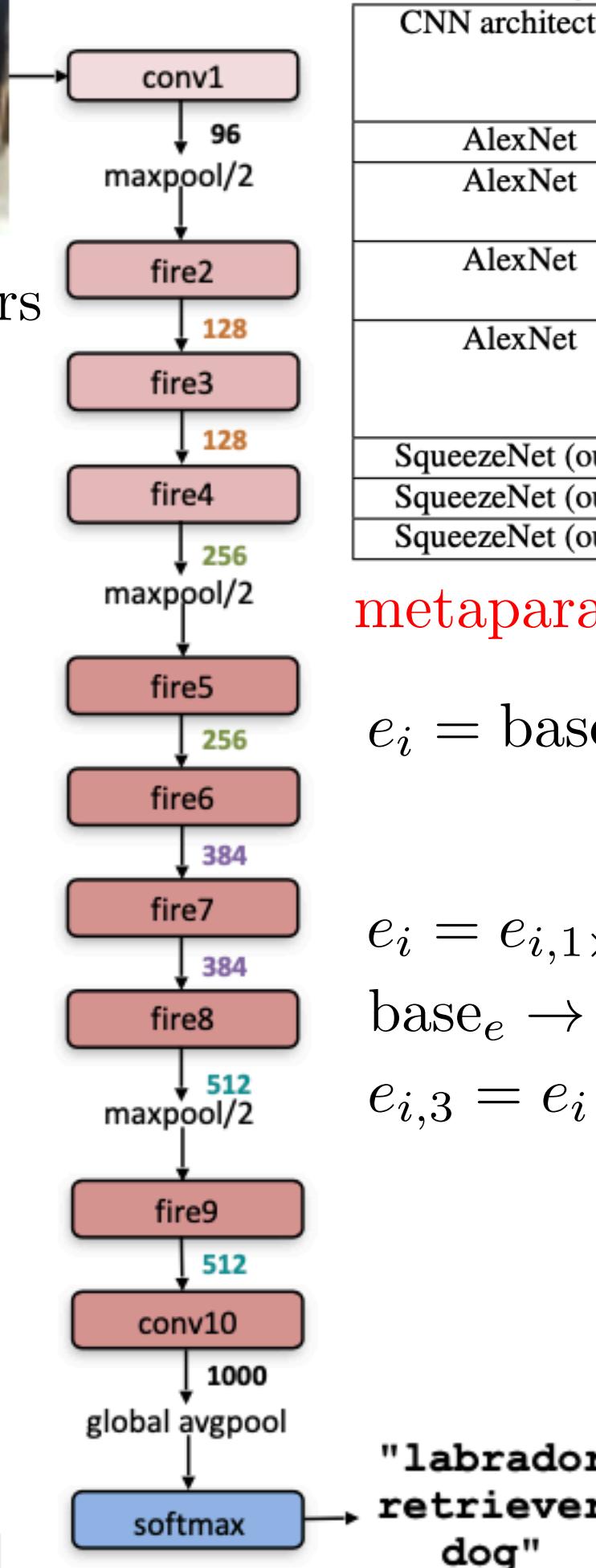
$s_{1 \times 1} \rightarrow$ number of 1×1 filters in the squeeze layer

$e_{1 \times 1} \rightarrow$ number of 1×1 filters in the expand layer

$e_{3 \times 3} \rightarrow$ number of 3×3 filters in the expand layer

$s_{1 \times 1} < e_{1 \times 1} + e_{3 \times 3}$ as per strategy 2

| Architecture | Top-1 Accuracy | Top-5 Accuracy | Model Size |
|-----------------------------|----------------|----------------|------------|
| Vanilla SqueezeNet | 57.5% | 80.3% | 4.8MB |
| SqueezeNet + Simple Bypass | 60.4% | 82.5% | 4.8MB |
| SqueezeNet + Complex Bypass | 58.8% | 82.0% | 7.7MB |



| CNN architecture | Compression Approach | Data Type | Original \rightarrow Compressed Model Size | Reduction in Model Size vs. AlexNet | Top-1 ImageNet Accuracy | Top-5 ImageNet Accuracy |
|-------------------|--------------------------------------|-----------|--|-------------------------------------|-------------------------|-------------------------|
| AlexNet | None (baseline) | 32 bit | 240MB | 1x | 57.2% | 80.3% |
| AlexNet | SVD (Denton et al., 2014) | 32 bit | 240MB \rightarrow 48MB | 5x | 56.0% | 79.4% |
| AlexNet | Network Pruning (Han et al., 2015b) | 32 bit | 240MB \rightarrow 27MB | 9x | 57.2% | 80.3% |
| AlexNet | Deep Compression (Han et al., 2015a) | 5-8 bit | 240MB \rightarrow 6.9MB | 35x | 57.2% | 80.3% |
| SqueezeNet (ours) | None | 32 bit | 4.8MB | 50x | 57.5% | 80.3% |
| SqueezeNet (ours) | Deep Compression | 8 bit | 4.8MB \rightarrow 0.66MB | 363x | 57.5% | 80.3% |
| SqueezeNet (ours) | Deep Compression | 6 bit | 4.8MB \rightarrow 0.47MB | 510x | 57.5% | 80.3% |

metaparameters

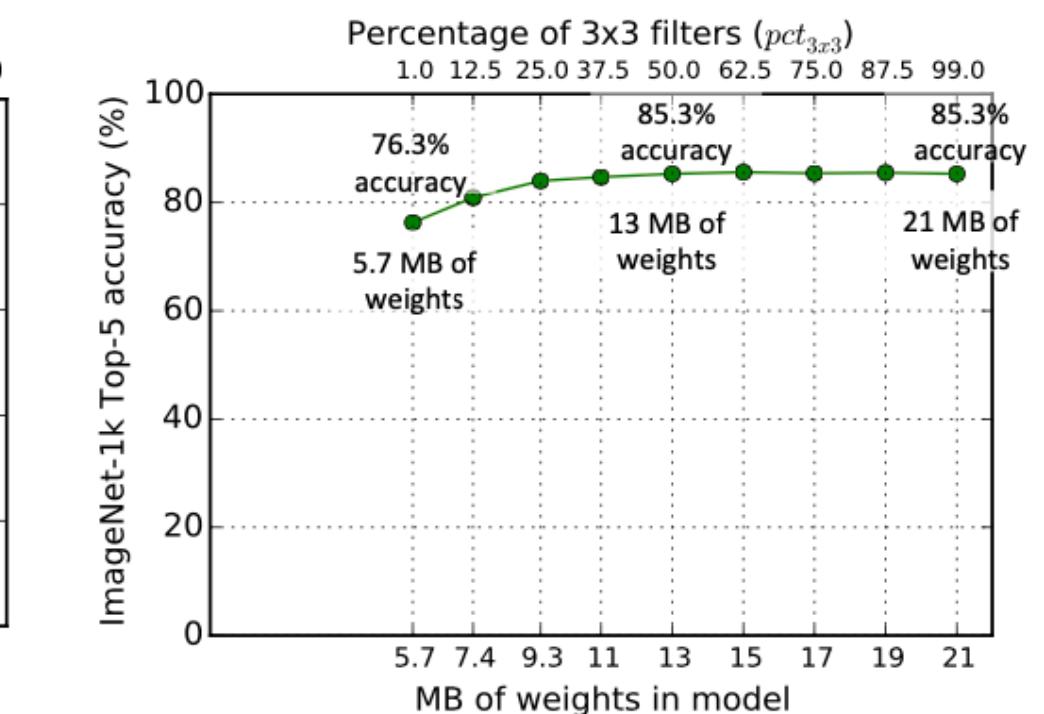
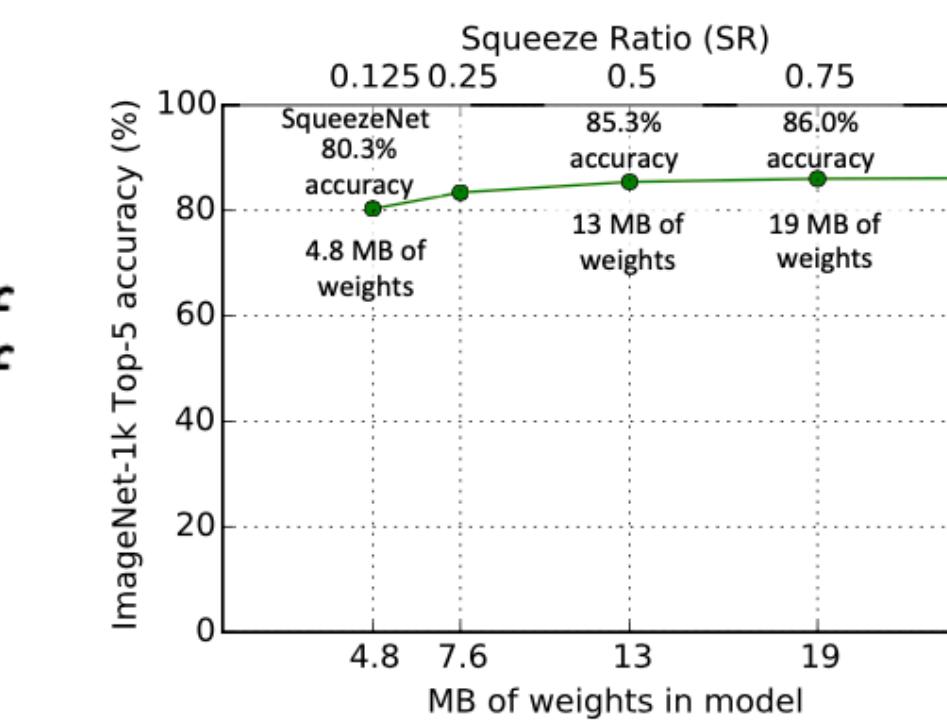
$e_i = \text{base}_e + \left(\text{incr}_e * \left\lfloor \frac{i}{\text{freq}} \right\rfloor \right)$ after every freq fire modules increase the number of expand filters by incr_e

$e_i = e_{i,1 \times 1} + e_{i,3 \times 3} \rightarrow$ number of expand filters for fire module *i*

base_e \rightarrow number of expand filters in the first fire module

$e_{i,3} = e_i * \text{pct}_{3 \times 3}, \quad e_{i,1} = e_i * (1 - \text{pct}_{3 \times 3}), \quad s_{i,1 \times 1} = \text{SR} * e_i$

percentage



XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks


[YouTube Playlist](#)

| | Network Variations | Operations used in Convolution | Memory Saving (Inference) | Computation Saving (Inference) | Accuracy on ImageNet (AlexNet) | |
|--------------------------------------|--|--------------------------------|---------------------------|--------------------------------|--------------------------------|-------|
| Standard Convolution | Real-Value Inputs Input Weight h_{in} w c | 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 | |

- virtual reality
- augmented reality
- smart wearable devices

$$B^* = \arg \max_B W^T B \text{ s.t. } B \in \{+1, -1\}^n$$

$$\begin{aligned} B_i &= +1 \text{ if } W_i \geq 0 \\ B_i &= -1 \text{ if } W_i < 0 \implies B^* = \text{sign}(W) \end{aligned}$$

$$I \in \mathbb{R}^{c \times w_{in} \times h_{in}}$$

$$W \in \mathbb{R}^{c \times w \times h}, w \leq w_{in}, h \leq h_{in}$$

$$W \approx \alpha B$$

$$B \in \{+1, -1\}^{c \times w \times h} \rightarrow \text{binary filter}$$

$$\alpha > 0 \rightarrow \text{scaling factor}$$

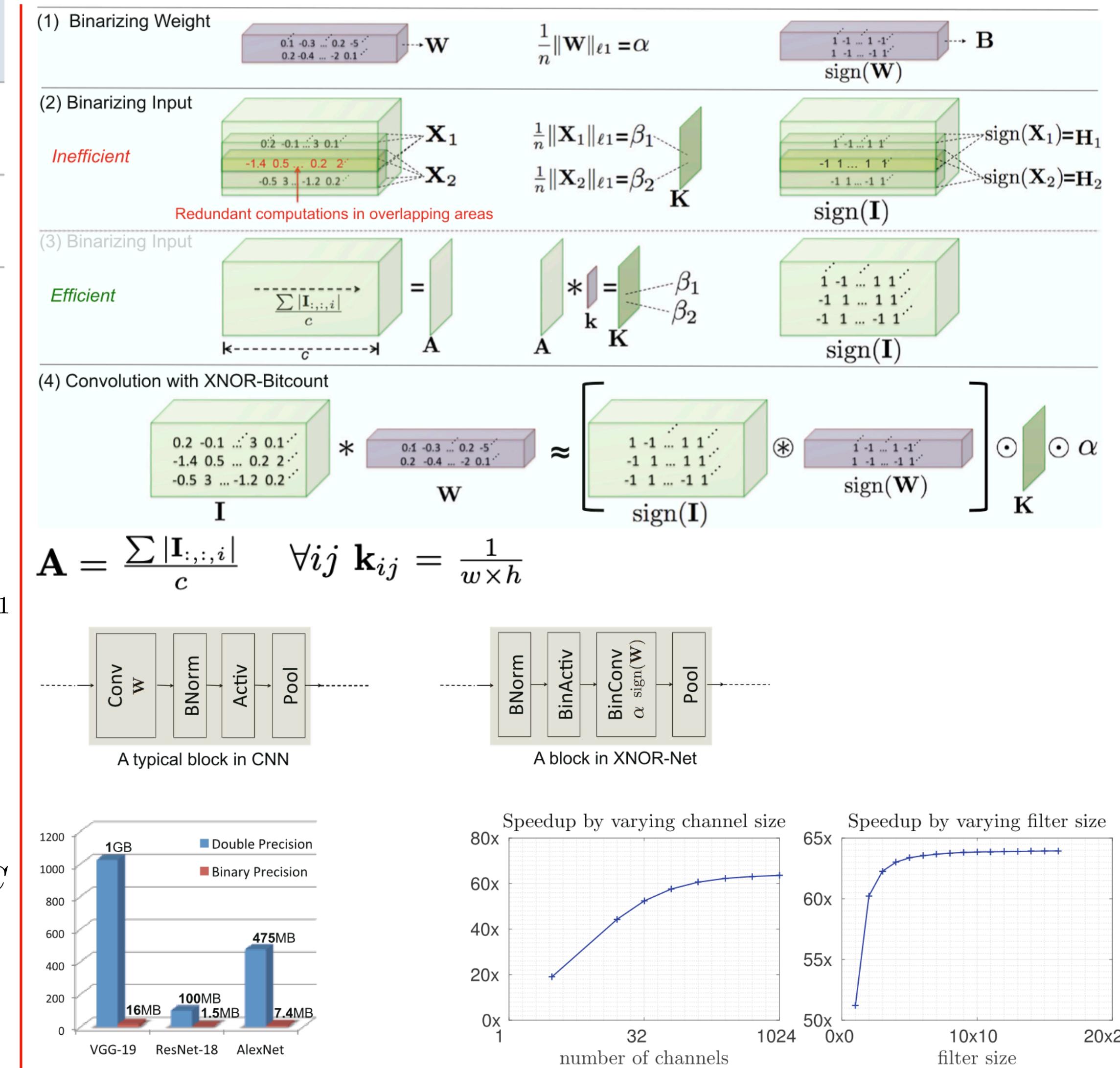
$$I * W \approx (I \oplus B)\alpha$$

convolution without any multiplications

$$W, B \in R^n \text{ where } n = cwh$$

$$J(B, \alpha) = \|W - \alpha B\|^2 = \alpha^2 \underbrace{B^T B}_{=n \text{ since } B \in \{+1, -1\}^n} - 2\alpha W^T B + \underbrace{W^T W}_{:=c \text{ is a constant since } W \text{ is known}}$$

$$B^*, \alpha^* = \arg \min_{B, \alpha} J(B, \alpha)$$



MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications

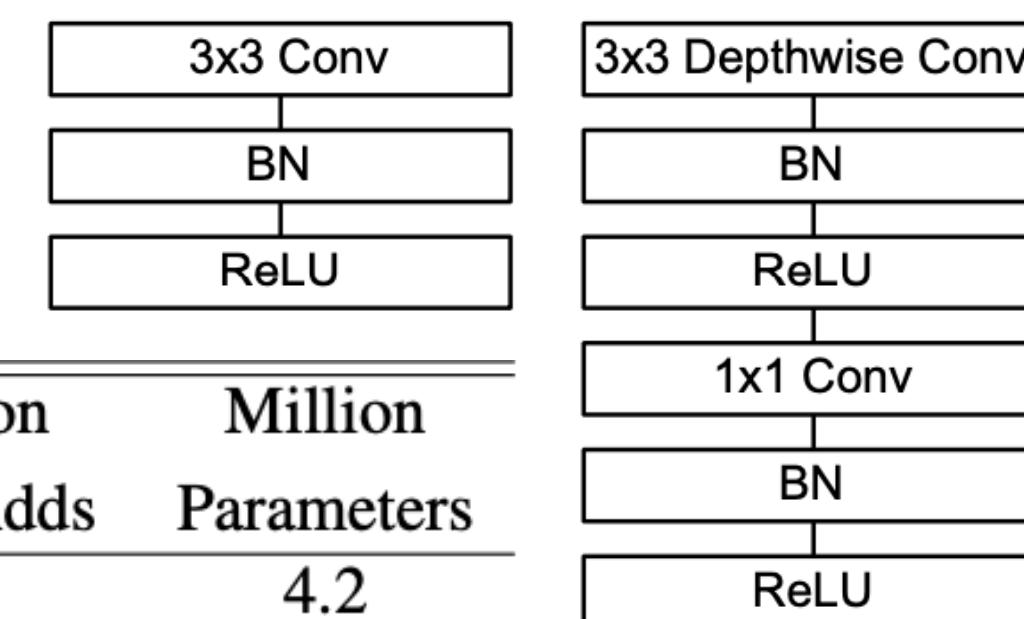

[YouTube Playlist](#)
 $F \in \mathbb{R}^{D_F \times D_F \times M} \rightarrow \text{input feature maps}$
 $G \in \mathbb{R}^{D_F \times D_F \times N} \rightarrow \text{output feature maps}$
 $K \in \mathbb{R}^{D_K \times D_K \times M \times N} \rightarrow \text{convolution kernel}$
 $G_{k,\ell,n} = \sum_{i,j,m} K_{i,j,m,n} F_{k+i-1,\ell+j-1,m} \rightarrow \text{stride one and padding}$

 Computational Cost: $D_K D_K M N D_F D_F$
 $\hat{G}_{k,\ell,m} = \sum_{i,j} \hat{K}_{i,j,m} F_{k+i-1,\ell+j-1,m}$
 $\hat{K} \in \mathbb{R}^{D_K \times D_K \times M} \rightarrow \text{depth-wise (channel-wise) convolution kernel}$

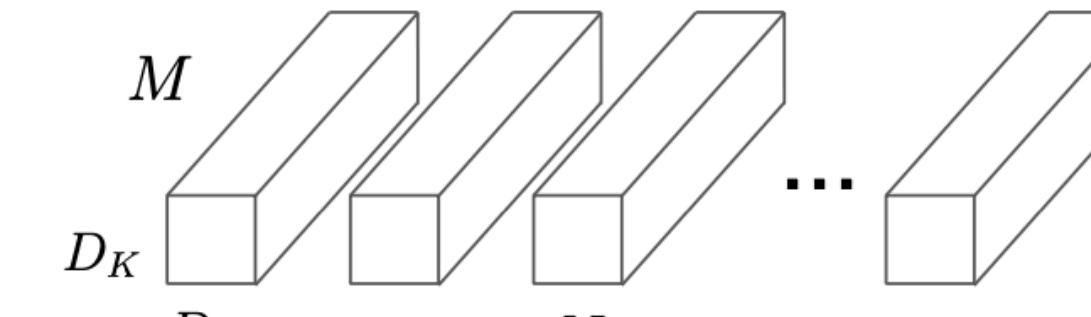
 Computational Cost: $D_K D_K M D_F D_F$

Depth-wise Separable Convolution Cost:

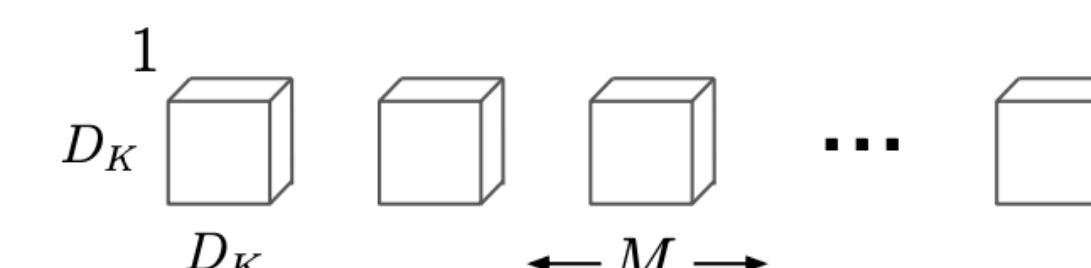
$$D_K D_K M D_F D_F + \underbrace{M N D_F D_F}_{1 \times 1 \text{ conv}}$$


 $\alpha \rightarrow \text{width multiplier}$
 $\alpha M, \alpha N$
 $\rho \rightarrow \text{resolution multiplier}$
 ρD_F

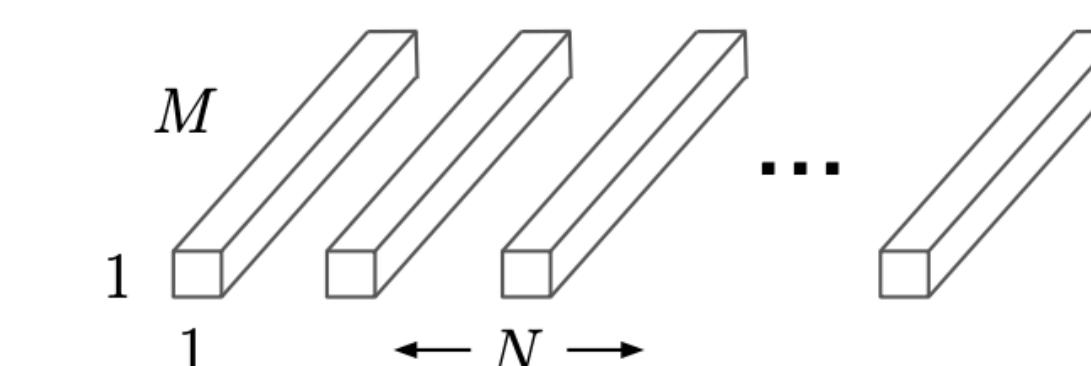
| Model | ImageNet Accuracy | Million Mult-Adds | Million Parameters |
|-------------------|-------------------|-------------------|--------------------|
| 1.0 MobileNet-224 | 70.6% | 569 | 4.2 |
| GoogleNet | 69.8% | 1550 | 6.8 |
| VGG 16 | 71.5% | 15300 | 138 |



(a) Standard Convolution Filters



(b) Depthwise Convolutional Filters



(c) 1x1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

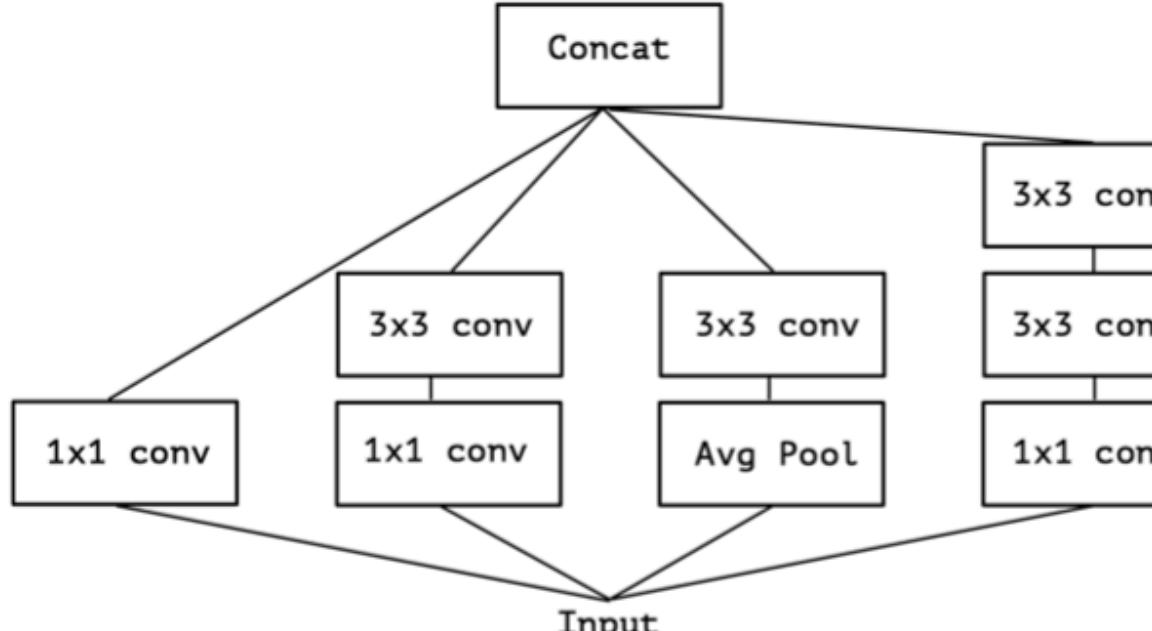
| Type / Stride | Filter Shape | Input Size |
|-----------------|--------------------------------------|----------------------------|
| Conv / s2 | $3 \times 3 \times 3 \times 32$ | $224 \times 224 \times 3$ |
| Conv dw / s1 | $3 \times 3 \times 32 \text{ dw}$ | $112 \times 112 \times 32$ |
| Conv / s1 | $1 \times 1 \times 32 \times 64$ | $112 \times 112 \times 32$ |
| Conv dw / s2 | $3 \times 3 \times 64 \text{ dw}$ | $112 \times 112 \times 64$ |
| Conv / s1 | $1 \times 1 \times 64 \times 128$ | $56 \times 56 \times 64$ |
| Conv dw / s1 | $3 \times 3 \times 128 \text{ dw}$ | $56 \times 56 \times 128$ |
| Conv / s1 | $1 \times 1 \times 128 \times 128$ | $56 \times 56 \times 128$ |
| Conv dw / s2 | $3 \times 3 \times 128 \text{ dw}$ | $56 \times 56 \times 128$ |
| Conv / s1 | $1 \times 1 \times 128 \times 256$ | $28 \times 28 \times 128$ |
| Conv dw / s1 | $3 \times 3 \times 256 \text{ dw}$ | $28 \times 28 \times 256$ |
| Conv / s1 | $1 \times 1 \times 256 \times 256$ | $28 \times 28 \times 256$ |
| Conv dw / s2 | $3 \times 3 \times 256 \text{ dw}$ | $28 \times 28 \times 256$ |
| Conv / s1 | $1 \times 1 \times 256 \times 512$ | $14 \times 14 \times 256$ |
| 5× Conv dw / s1 | $3 \times 3 \times 512 \text{ dw}$ | $14 \times 14 \times 512$ |
| Conv / s1 | $1 \times 1 \times 512 \times 512$ | $14 \times 14 \times 512$ |
| Conv dw / s2 | $3 \times 3 \times 512 \text{ dw}$ | $14 \times 14 \times 512$ |
| Conv / s1 | $1 \times 1 \times 512 \times 1024$ | $7 \times 7 \times 512$ |
| Conv dw / s2 | $3 \times 3 \times 1024 \text{ dw}$ | $7 \times 7 \times 1024$ |
| Conv / s1 | $1 \times 1 \times 1024 \times 1024$ | $7 \times 7 \times 1024$ |
| Avg Pool / s1 | Pool 7×7 | $7 \times 7 \times 1024$ |
| FC / s1 | 1024×1000 | $1 \times 1 \times 1024$ |
| Softmax / s1 | Classifier | $1 \times 1 \times 1000$ |

| Width Multiplier | ImageNet Accuracy | Million | | Resolution | ImageNet Accuracy | Million | |
|--------------------|-------------------|-----------|------------|-------------------|-------------------|-----------|------------|
| | | Mult-Adds | Parameters | | | Mult-Adds | Parameters |
| 1.0 MobileNet-224 | 70.6% | 569 | 4.2 | 1.0 MobileNet-224 | 70.6% | 569 | 4.2 |
| 0.75 MobileNet-224 | 68.4% | 325 | 2.6 | 1.0 MobileNet-192 | 69.1% | 418 | 4.2 |
| 0.5 MobileNet-224 | 63.7% | 149 | 1.3 | 1.0 MobileNet-160 | 67.2% | 290 | 4.2 |
| 0.25 MobileNet-224 | 50.6% | 41 | 0.5 | 1.0 MobileNet-128 | 64.4% | 186 | 4.2 |

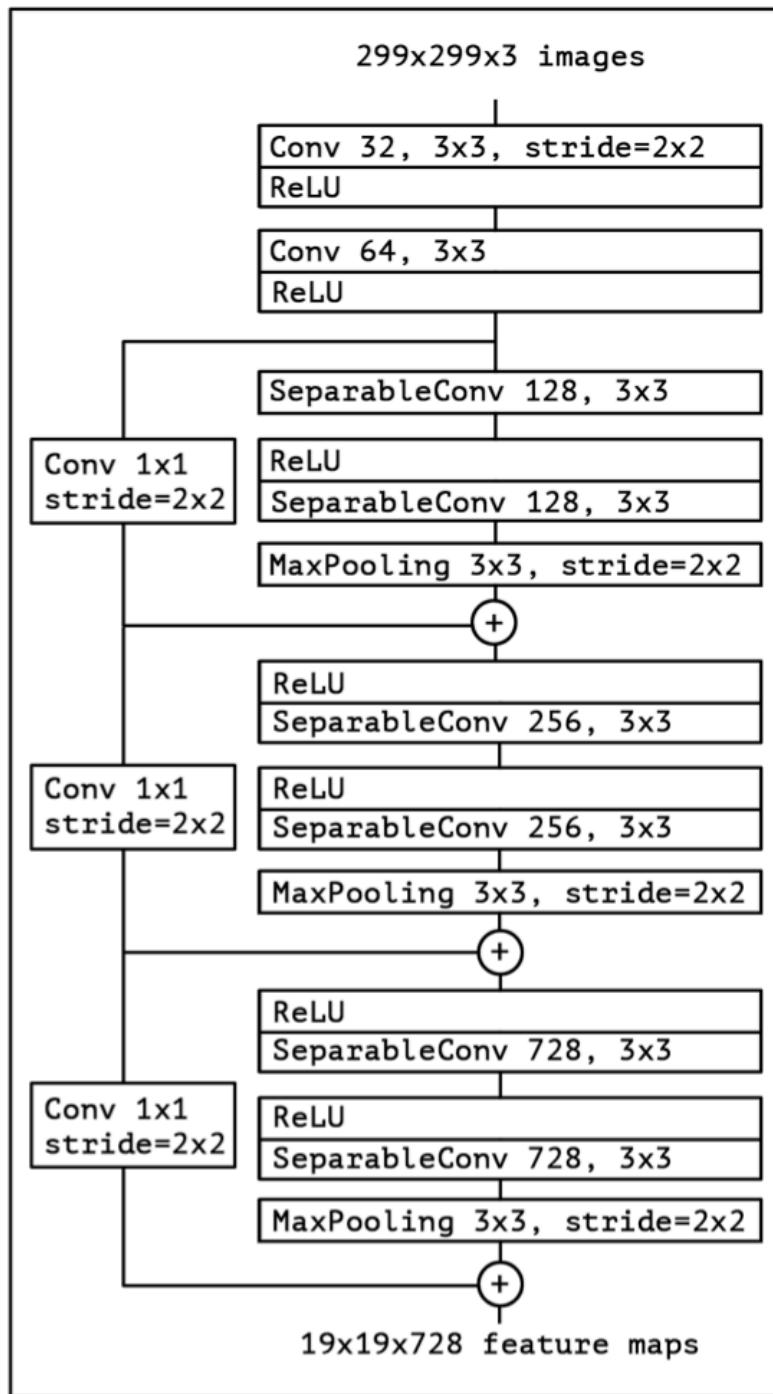
Xception: Deep Learning with Depthwise Separable Convolutions

[YouTube Playlist](#)

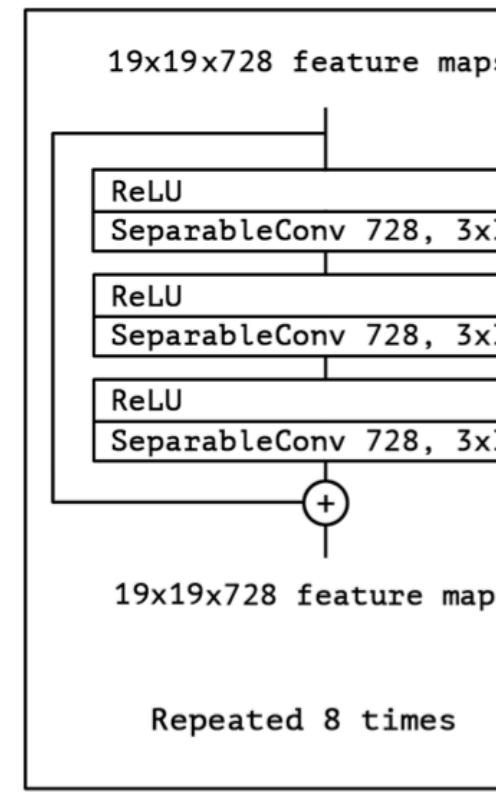
A canonical Inception module (Inception V3)



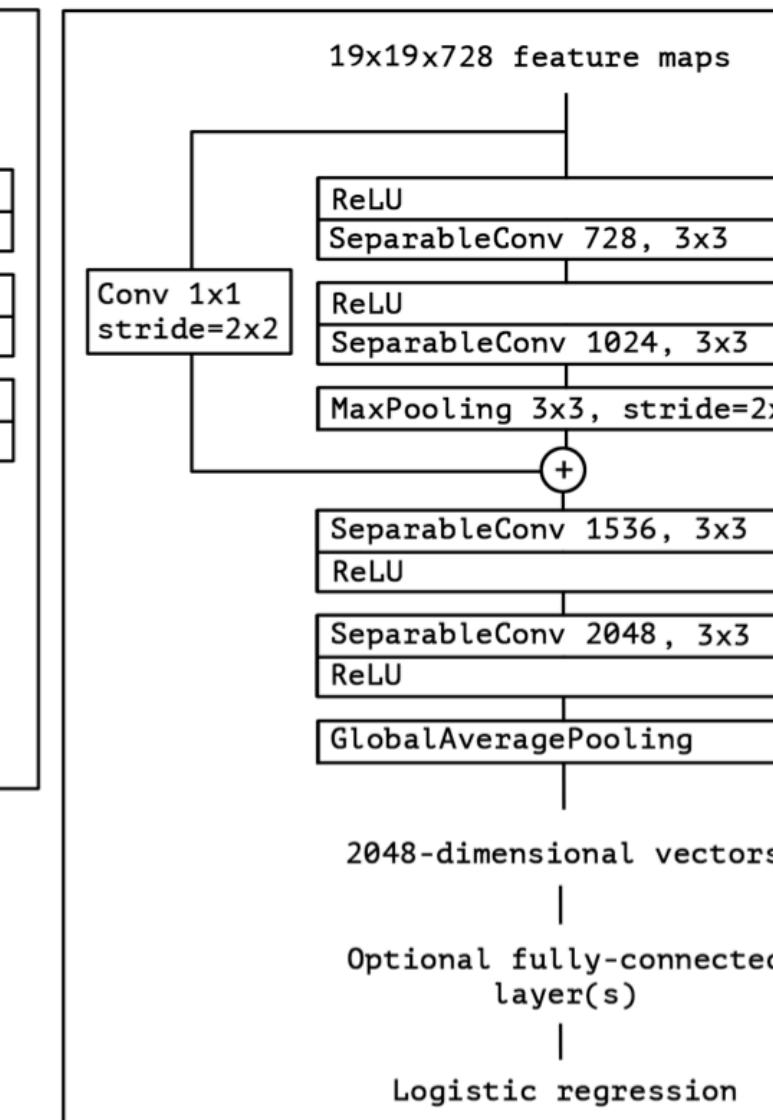
Entry flow



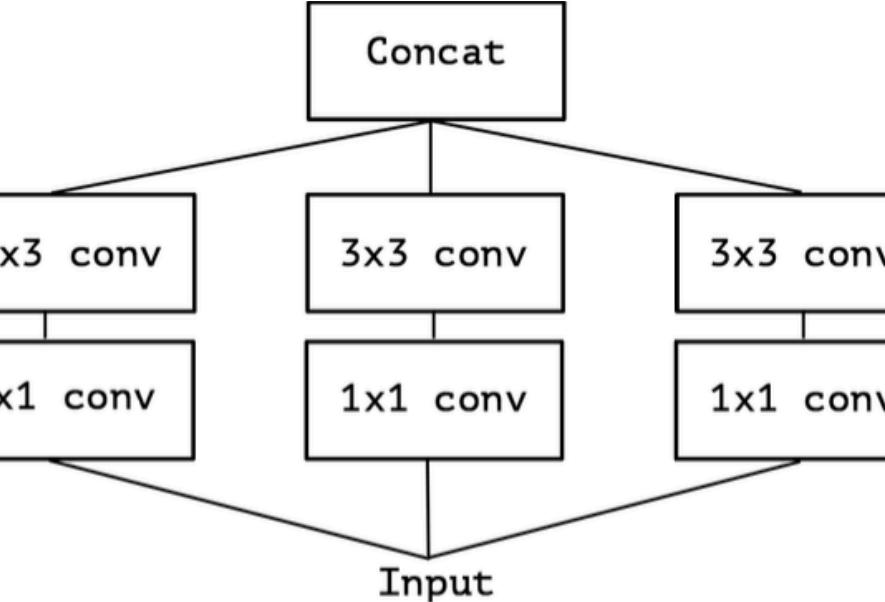
Middle flow



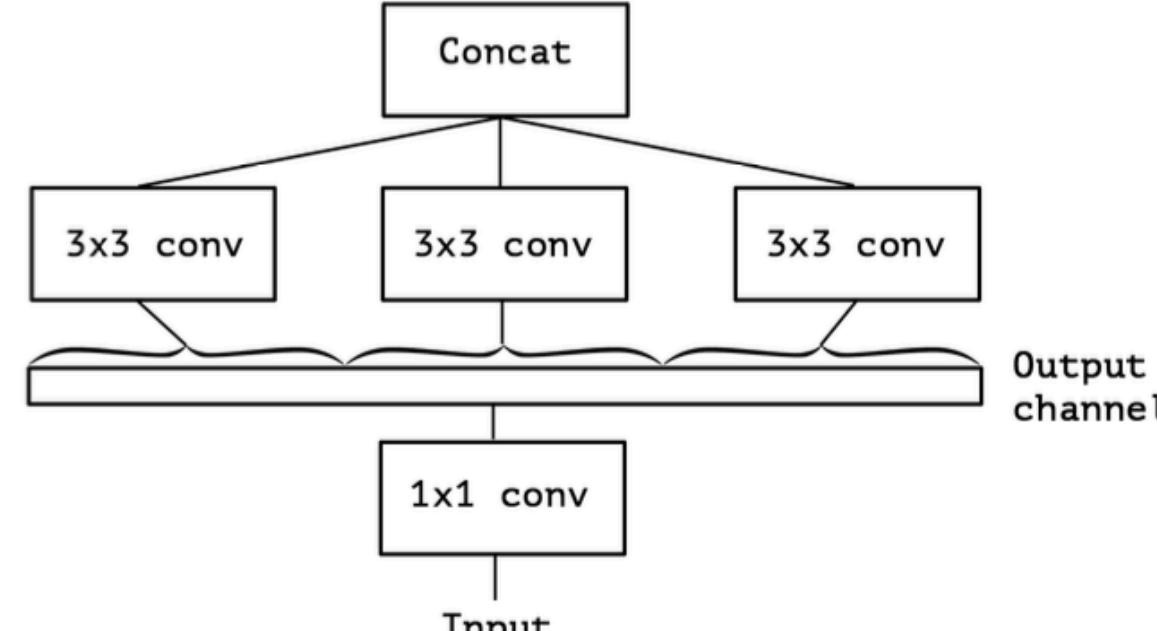
Exit flow



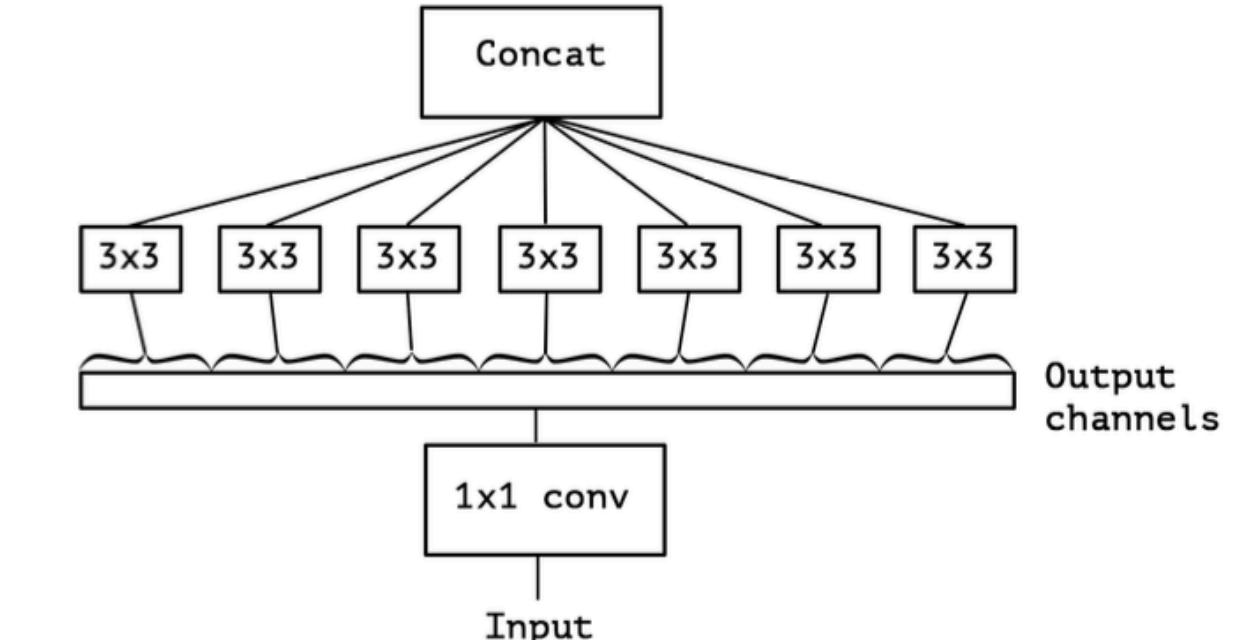
A simplified Inception module



A strictly equivalent reformulation of the simplified Inception module

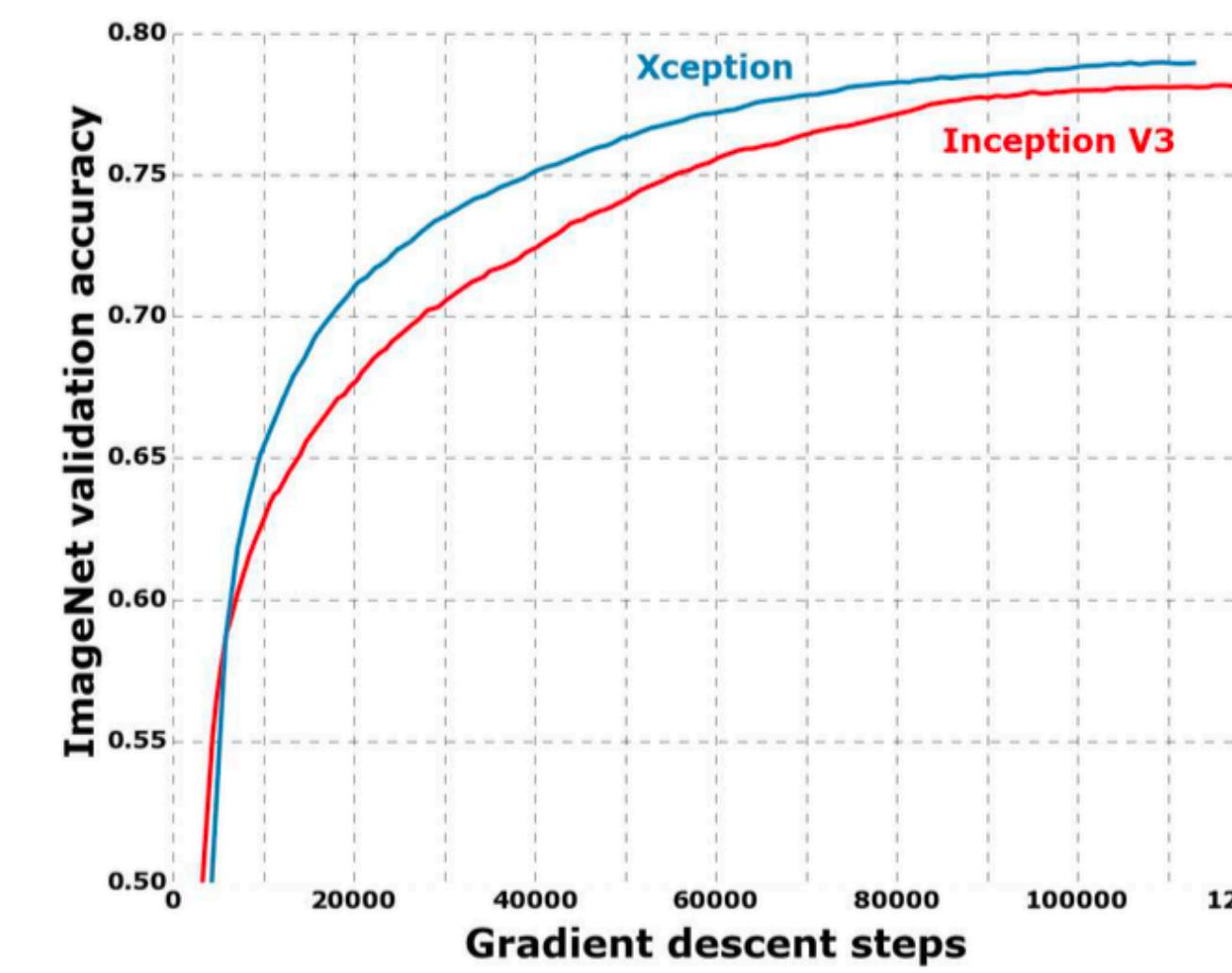


An “extreme” version of our Inception module, with one spatial convolution per output channel of the 1x1 convolution.



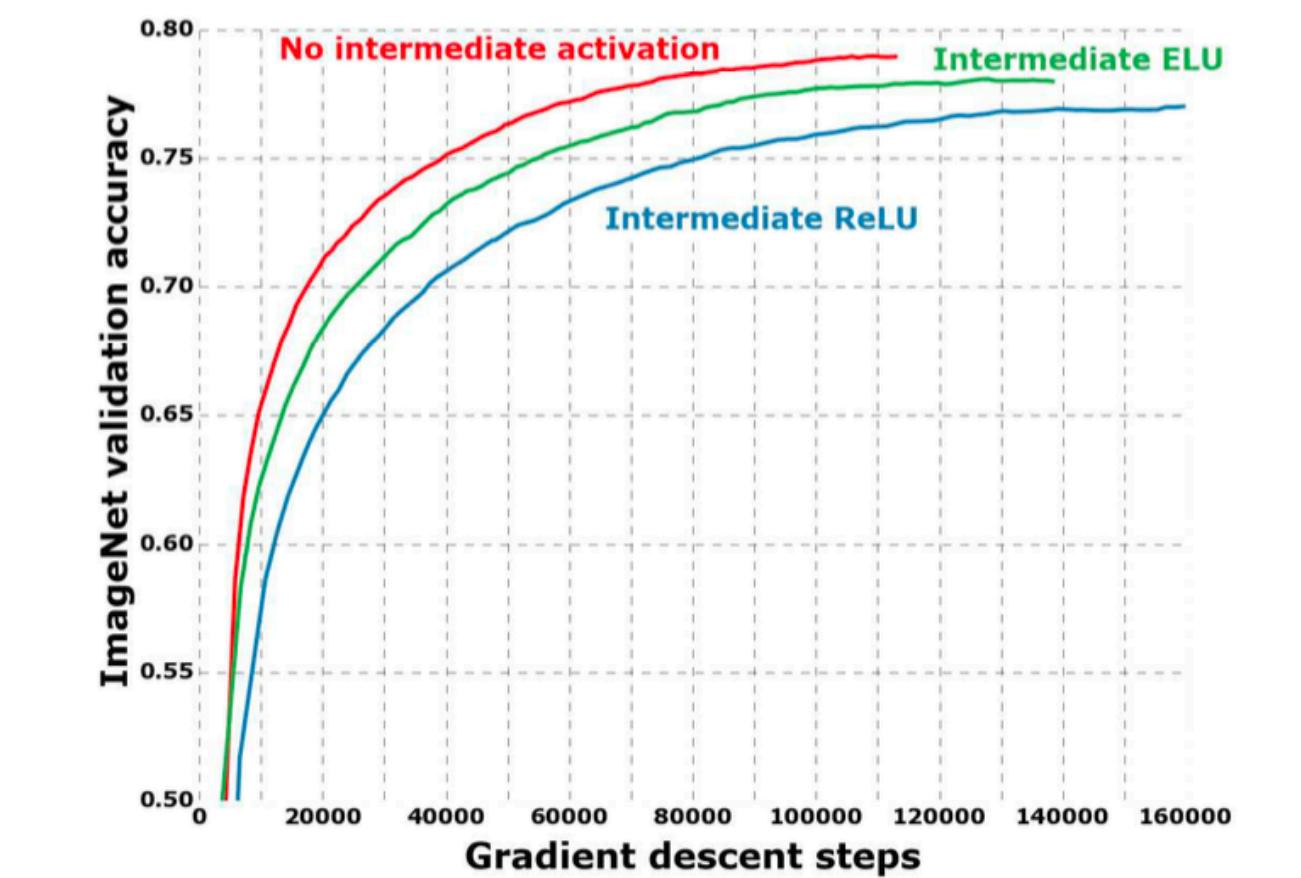
depthwise separable convolutions
depthwise convolution
pointwise convolution

| | Top-1 accuracy | Top-5 accuracy |
|-----------------|----------------|----------------|
| VGG-16 | 0.715 | 0.901 |
| ResNet-152 | 0.770 | 0.933 |
| Inception V3 | 0.782 | 0.941 |
| Xception | 0.790 | 0.945 |



Parameter count Steps/second

| | Parameter count | Steps/second |
|-----------------|-----------------|--------------|
| Inception V3 | 23,626,728 | 31 |
| Xception | 22,855,952 | 28 |



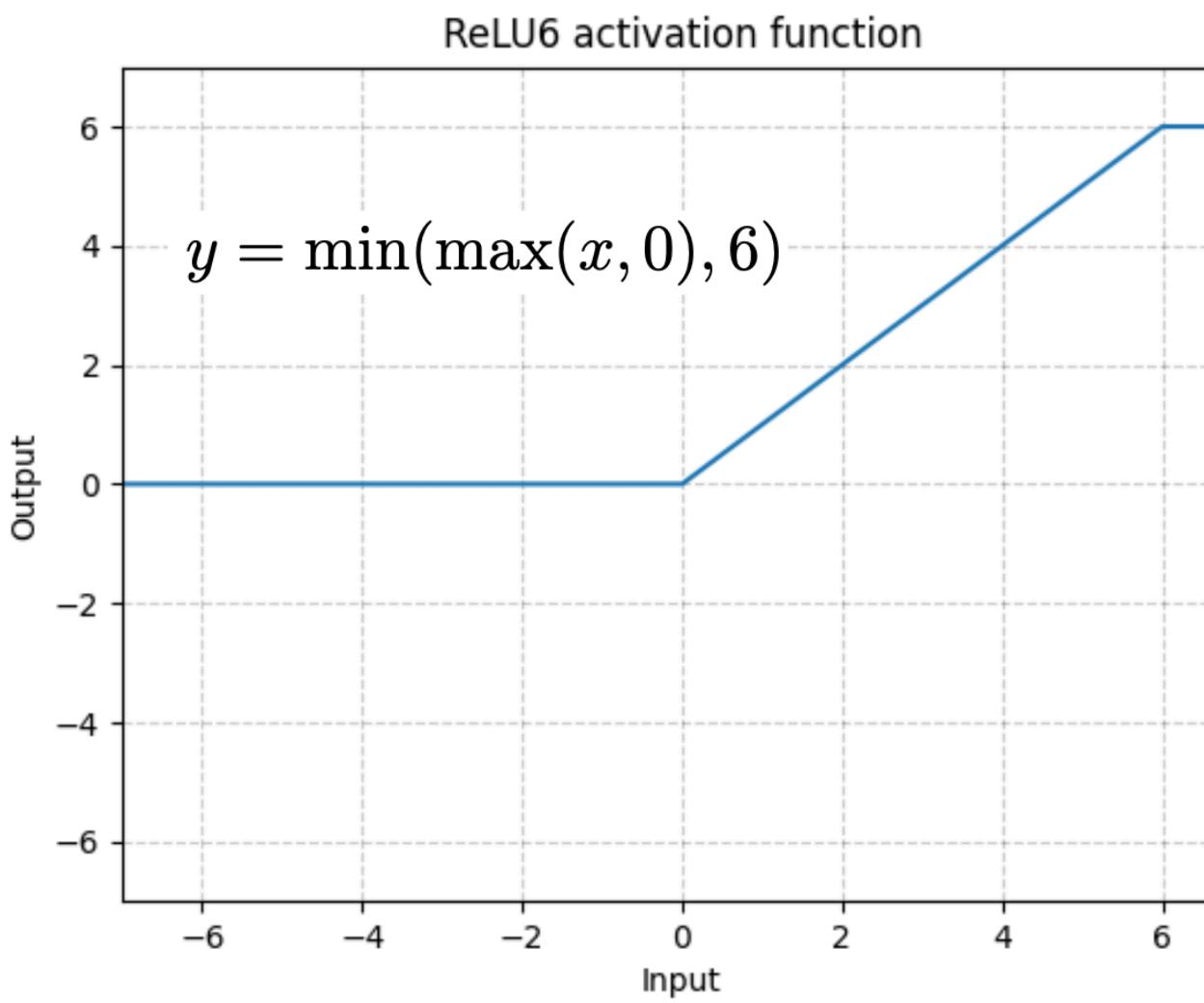
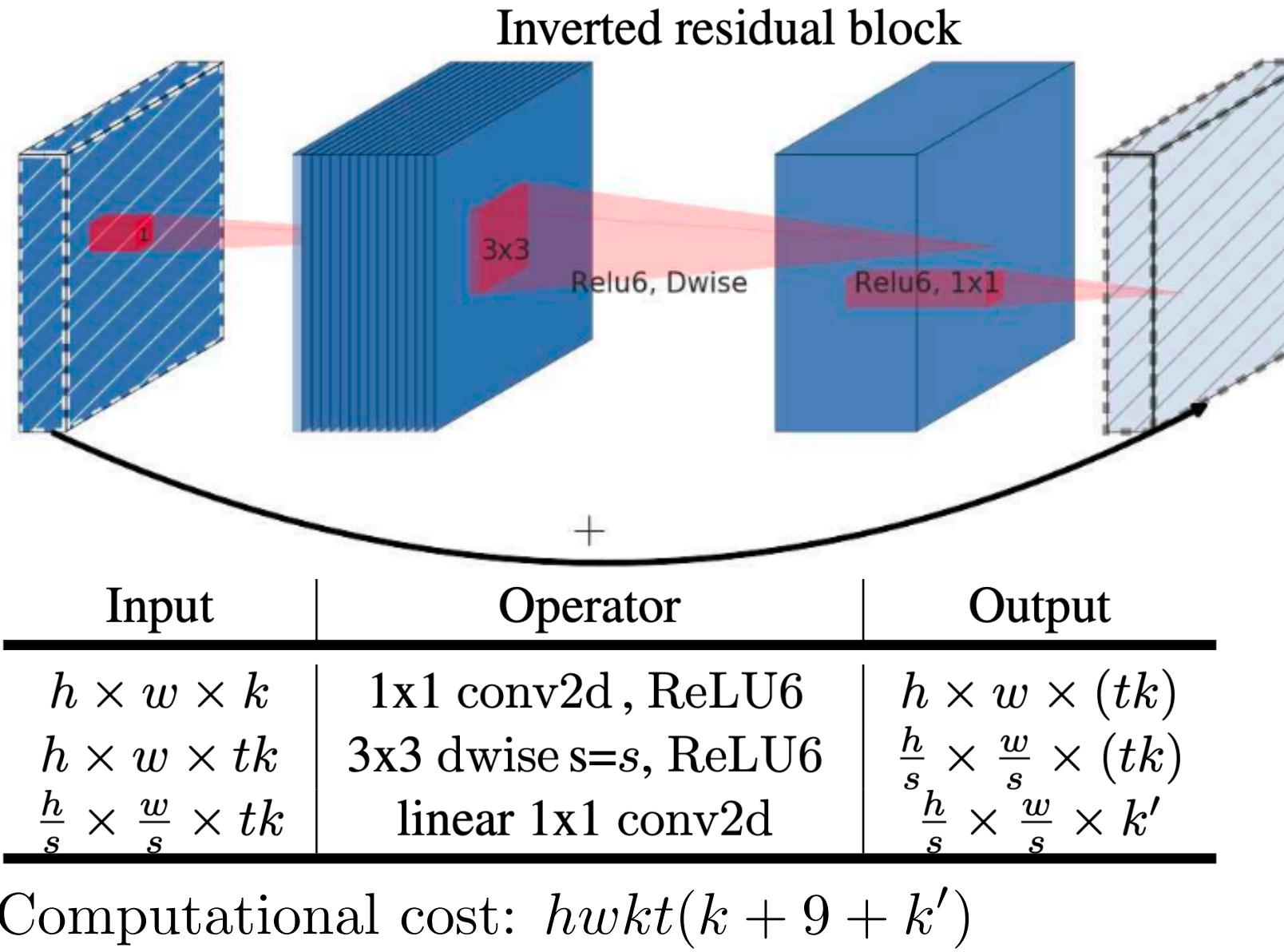


Boulder

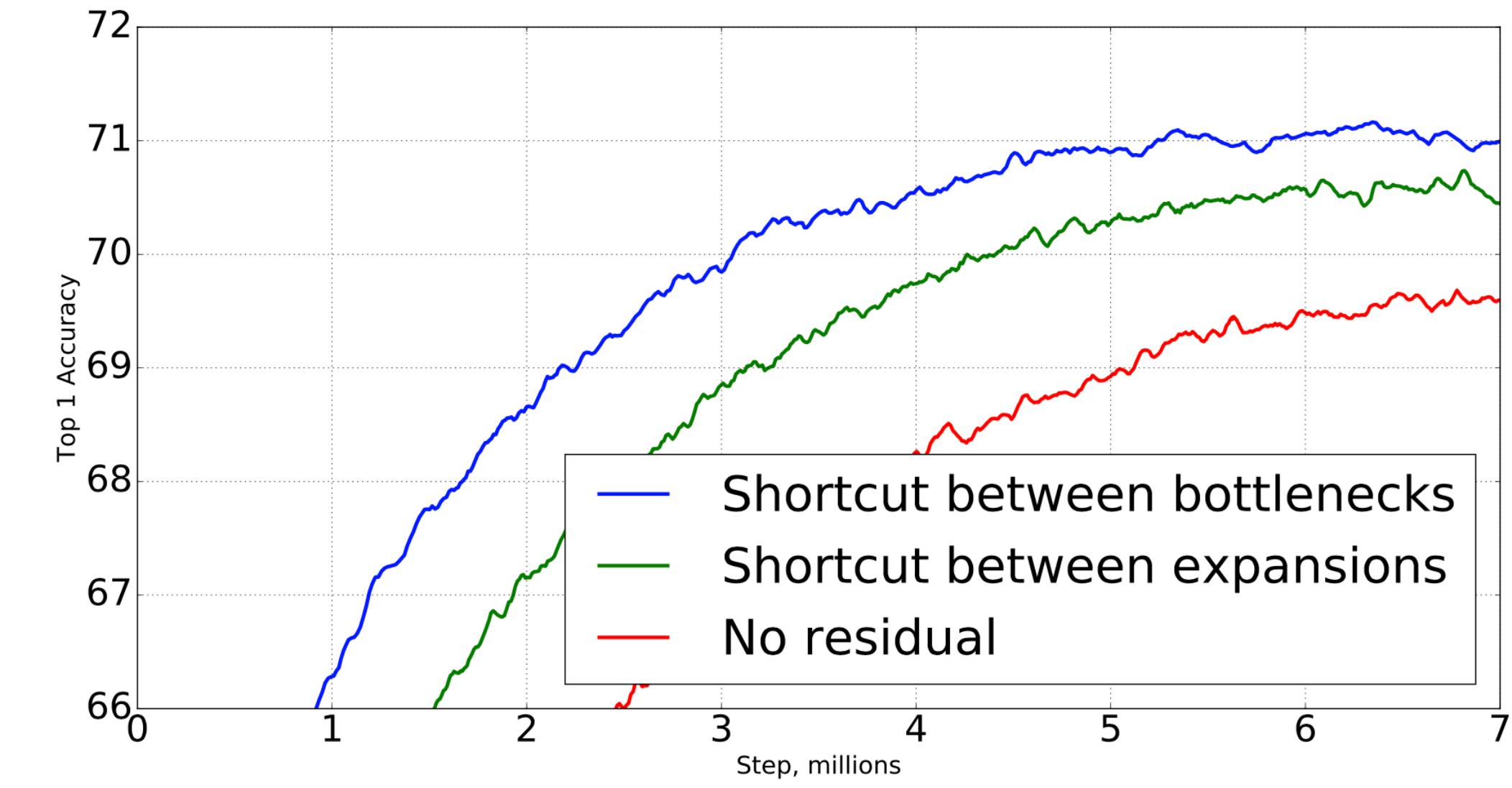
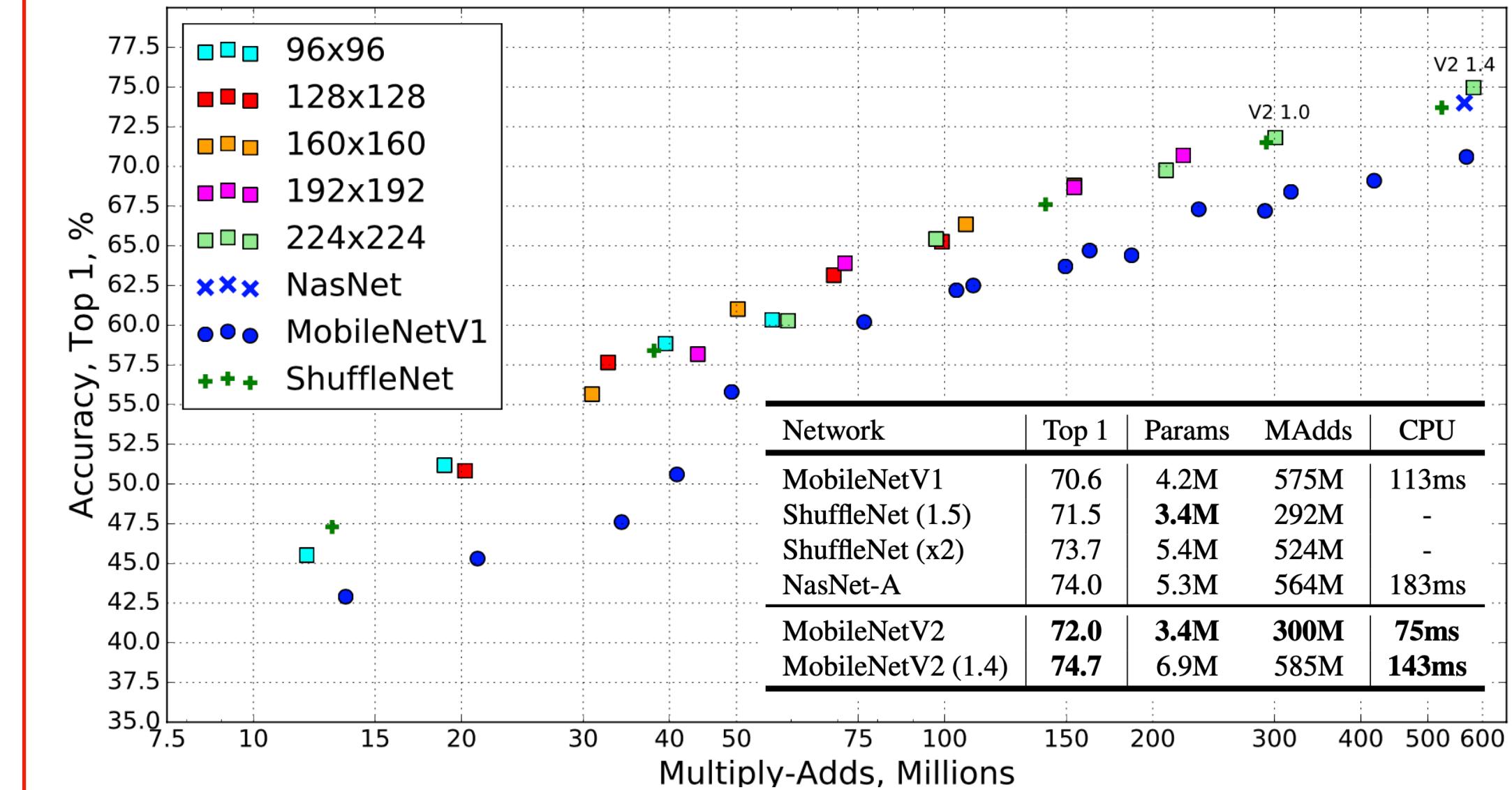
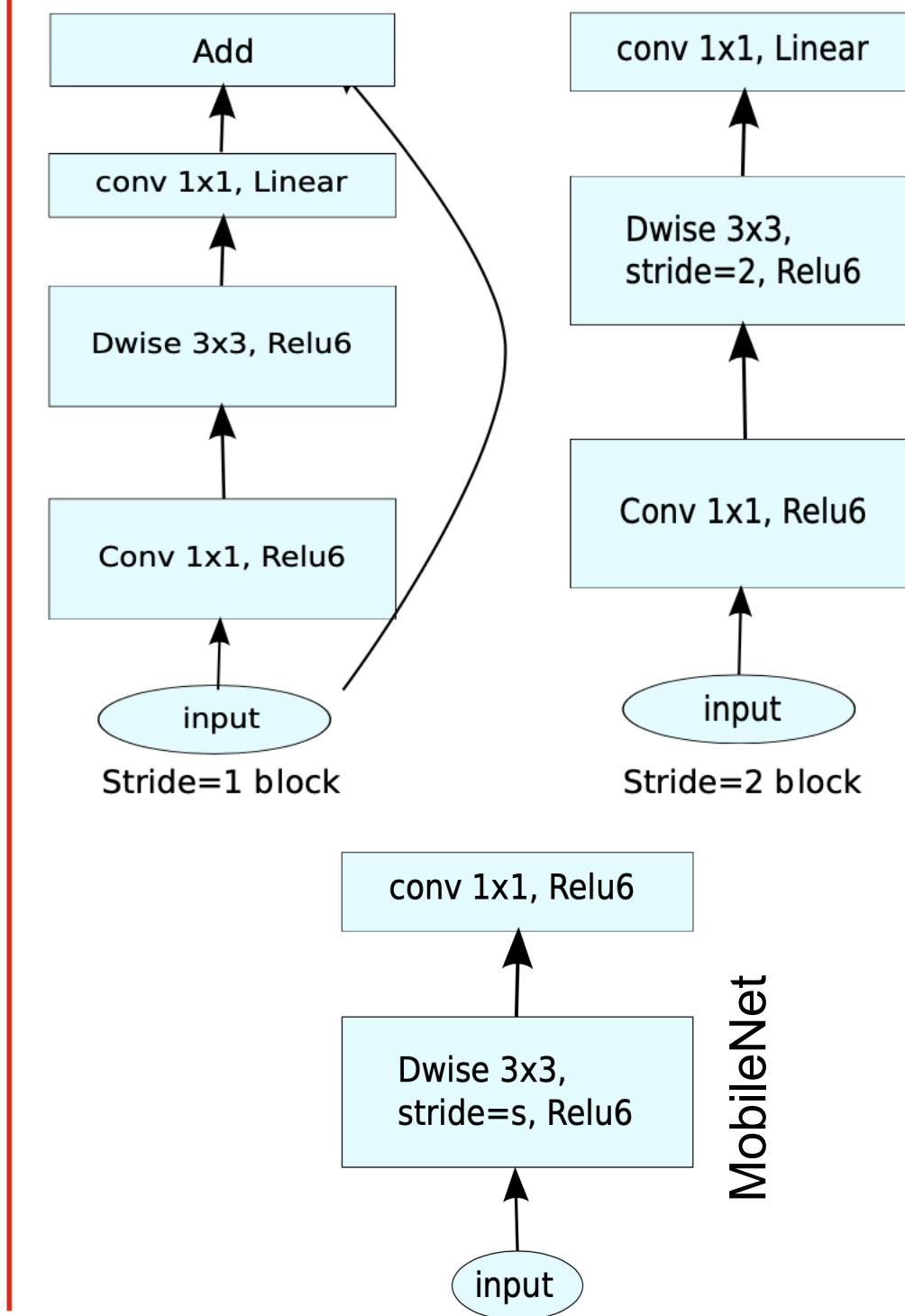


MobileNetV2: Inverted Residuals and Linear Bottlenecks

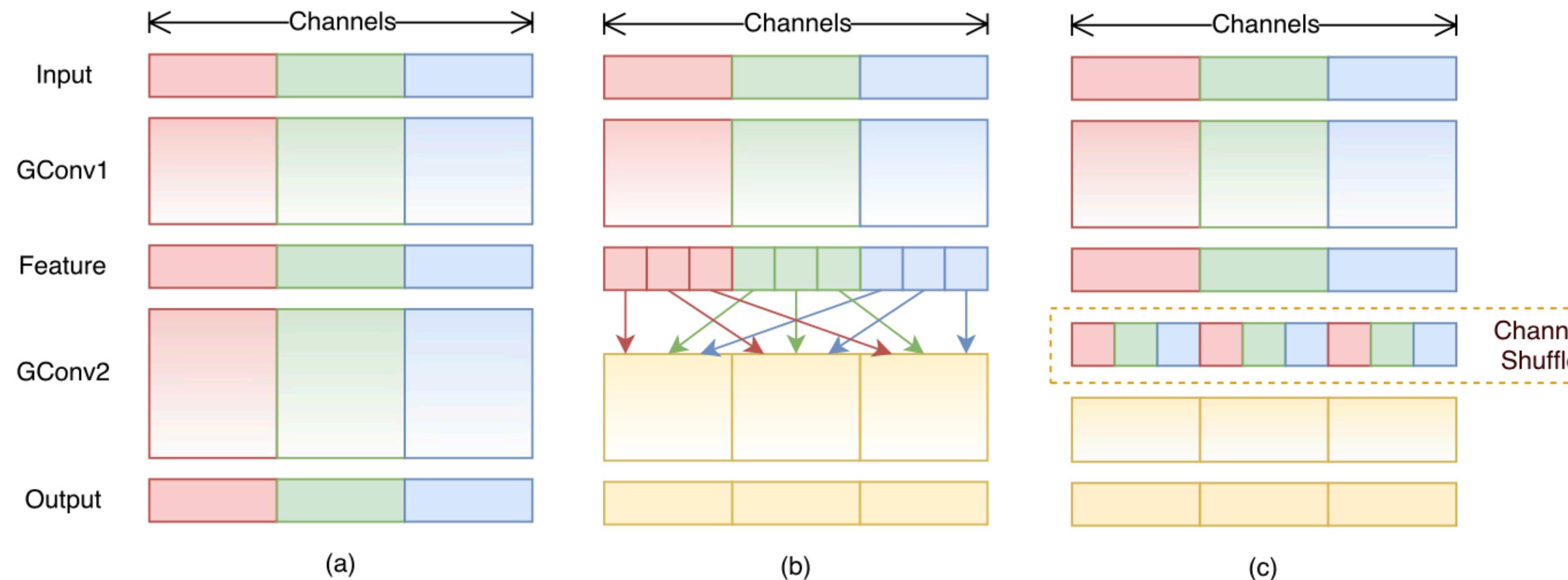
[YouTube Playlist](#)



| Input | Operator | t | c | n | s |
|--------------------------|-------------|-----|------|-----|-----|
| $224^2 \times 3$ | conv2d | - | 32 | 1 | 2 |
| $112^2 \times 32$ | bottleneck | 1 | 16 | 1 | 1 |
| $112^2 \times 16$ | bottleneck | 6 | 24 | 2 | 2 |
| $56^2 \times 24$ | bottleneck | 6 | 32 | 3 | 2 |
| $28^2 \times 32$ | bottleneck | 6 | 64 | 4 | 2 |
| $14^2 \times 64$ | bottleneck | 6 | 96 | 3 | 1 |
| $14^2 \times 96$ | bottleneck | 6 | 160 | 3 | 2 |
| $7^2 \times 160$ | bottleneck | 6 | 320 | 1 | 1 |
| $7^2 \times 320$ | conv2d 1x1 | - | 1280 | 1 | 1 |
| $7^2 \times 1280$ | avgpool 7x7 | - | - | 1 | - |
| $1 \times 1 \times 1280$ | conv2d 1x1 | - | k | - | |



ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices


[YouTube Playlist](#)


– drones – robots – smartphones

channel shuffle

$g \rightarrow$ number of groups

$n \rightarrow$ number of channels per group

$gn \rightarrow$ number of channels

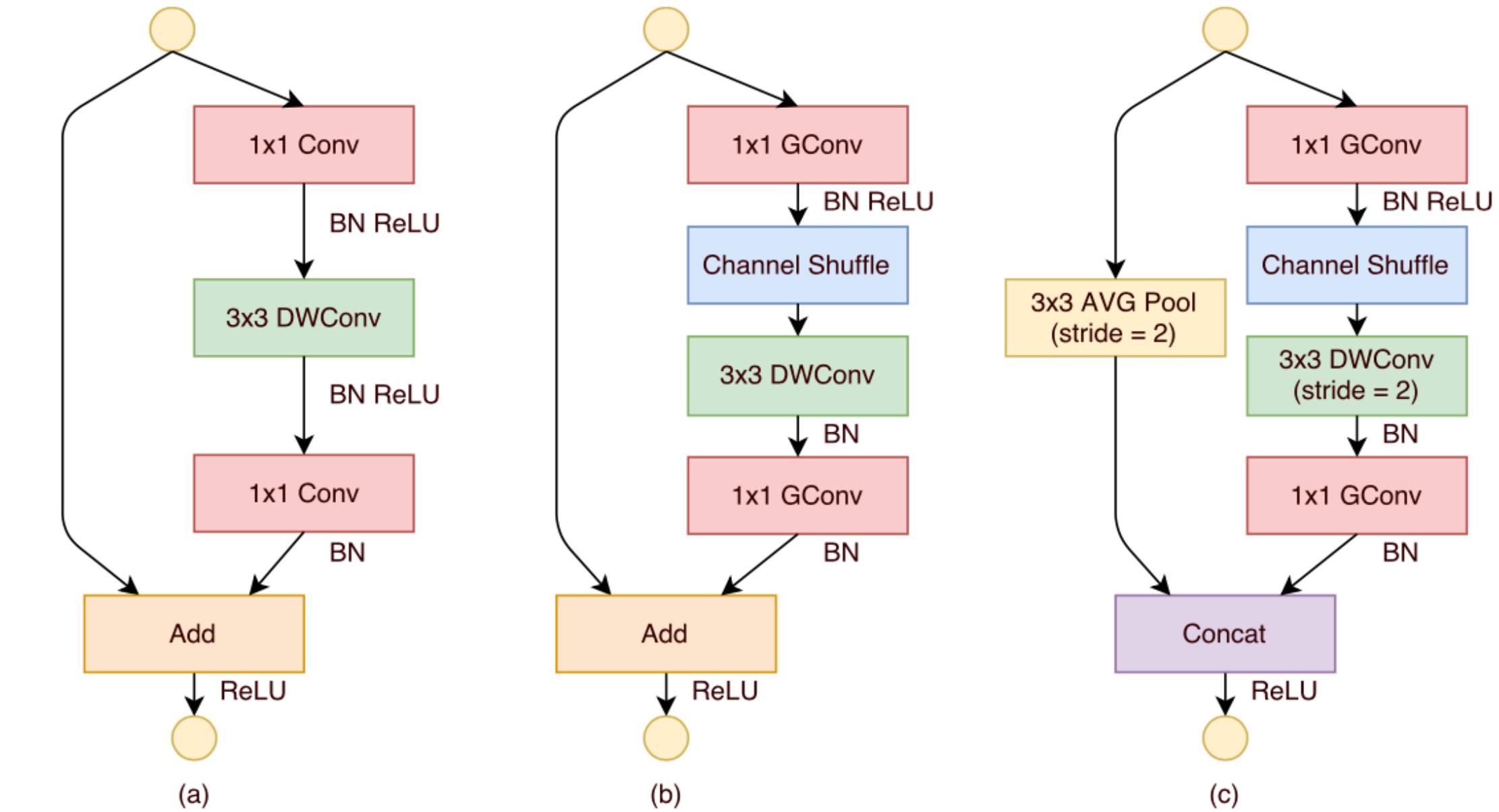
reshape to (g, n)

transpose

flatten

| Model | Cls err. (%) | Complexity (MFLOPs) |
|-----------------------------|--------------|---------------------|
| VGG-16 [31] | 28.5 | 15300 |
| ShuffleNet 2× ($g = 3$) | 26.3 | 524 |
| GoogleNet [34]* | 31.3 | 1500 |
| ShuffleNet 1× ($g = 8$) | 32.4 | 140 |
| AlexNet [22] | 42.8 | 720 |
| SqueezeNet [14] | 42.5 | 833 |
| ShuffleNet 0.5× ($g = 4$) | 41.6 | 38 |

| Model | Cls err. (%) | FLOPs | 224 × 224 | 480 × 640 | 720 × 1280 |
|-----------------------------|--------------|-------|-----------|-----------|------------|
| ShuffleNet 0.5× ($g = 3$) | 43.2 | 38M | 15.2ms | 87.4ms | 260.1ms |
| ShuffleNet 1× ($g = 3$) | 32.6 | 140M | 37.8ms | 222.2ms | 684.5ms |
| ShuffleNet 2× ($g = 3$) | 26.3 | 524M | 108.8ms | 617.0ms | 1857.6ms |
| AlexNet [22] | 42.8 | 720M | 184.0ms | 1156.7ms | 3633.9ms |
| 1.0 MobileNet-224 [12] | 29.4 | 569M | 110.0ms | 612.0ms | 1879.2ms |

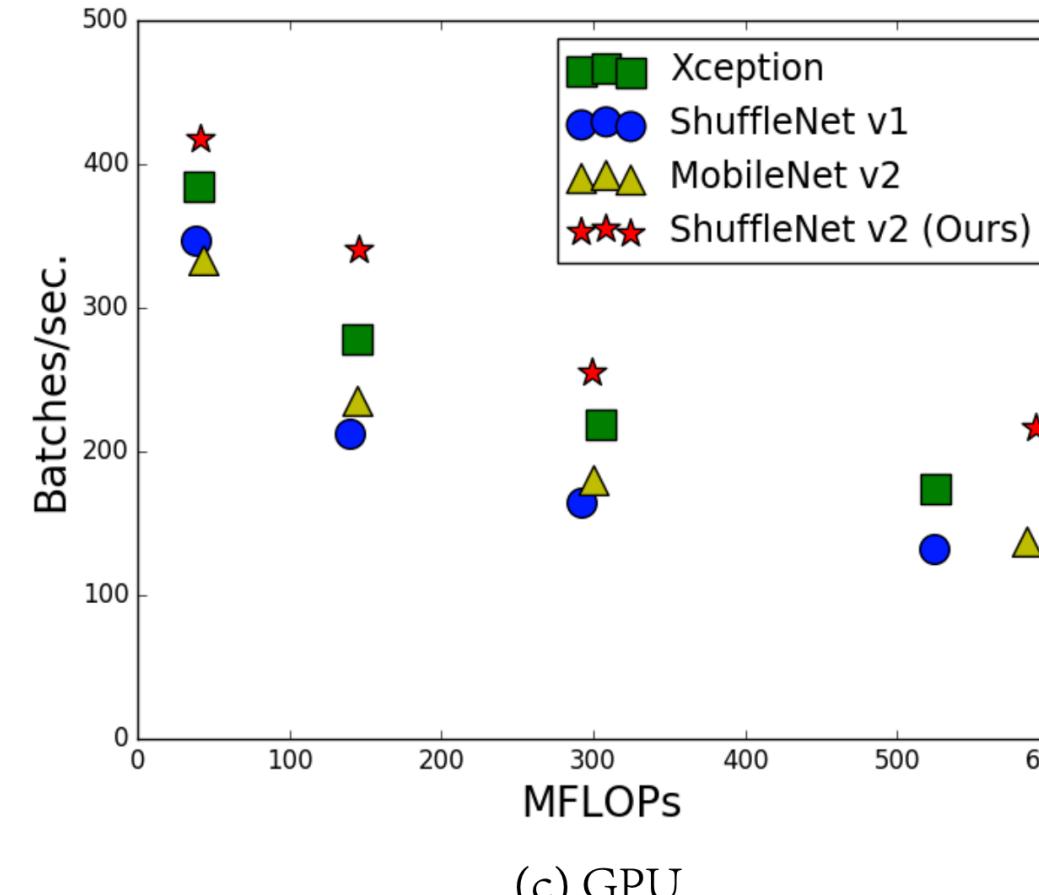


| Layer | Output size | KSize | Stride | Repeat | Output channels (g groups) | | | | |
|------------|-------------|-------|--------|--------|-------------------------------|---------|---------|---------|---------|
| | | | | | $g = 1$ | $g = 2$ | $g = 3$ | $g = 4$ | $g = 8$ |
| Image | 224 × 224 | | | | 3 | 3 | 3 | 3 | 3 |
| Conv1 | 112 × 112 | 3 × 3 | 2 | 1 | 24 | 24 | 24 | 24 | 24 |
| MaxPool | 56 × 56 | 3 × 3 | 2 | | | | | | |
| Stage2 | 28 × 28 | | 2 | 1 | 144 | 200 | 240 | 272 | 384 |
| | 28 × 28 | | 1 | 3 | 144 | 200 | 240 | 272 | 384 |
| Stage3 | 14 × 14 | | 2 | 1 | 288 | 400 | 480 | 544 | 768 |
| | 14 × 14 | | 1 | 7 | 288 | 400 | 480 | 544 | 768 |
| Stage4 | 7 × 7 | | 2 | 1 | 576 | 800 | 960 | 1088 | 1536 |
| | 7 × 7 | | 1 | 3 | 576 | 800 | 960 | 1088 | 1536 |
| GlobalPool | 1 × 1 | 7 × 7 | | | | | | | |
| FC | | | | | 1000 | 1000 | 1000 | 1000 | 1000 |
| Complexity | | | | | 143M | 140M | 137M | 133M | 137M |

ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design


[YouTube Video](#)

Networks with similar FLOPs have different speeds!



(c) GPU

- Memory Access Cost (MAC)
- degree of parallelism
- platform

G1) Equal channel width minimizes memory access cost (MAC).

$$B = hwc_1c_2 \rightarrow \text{FLOPs of a } 1 \times 1 \text{ convolution}$$

$$\text{MAC} = \underbrace{hw(c_1 + c_2)}_{\text{input/output feature maps}} + \underbrace{c_1c_2}_{\text{kernel weights}}$$

$$\text{MAC} \geq 2\sqrt{hwB} + \frac{B}{hw} \text{ (equality when } c_1 = c_2\text{)}$$

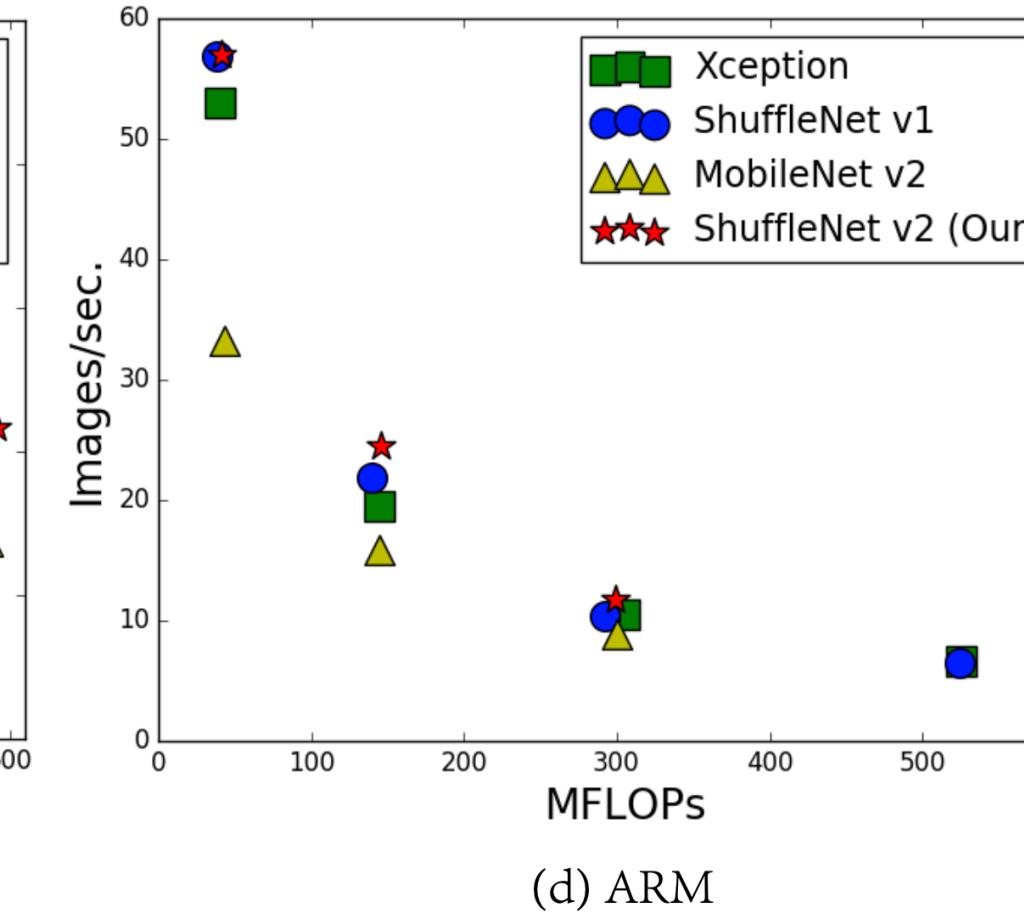
G2) Excessive group convolution increases MAC.

$g \rightarrow$ number of groups

$$B = hwc_1c_2/g$$

$$\text{MAC} = hw(c_1 + c_2) + c_1c_2/g$$

$$\text{MAC} = hwc_1 + Bg/c_1 + B/(hw) \implies \text{MAC increases by } g$$



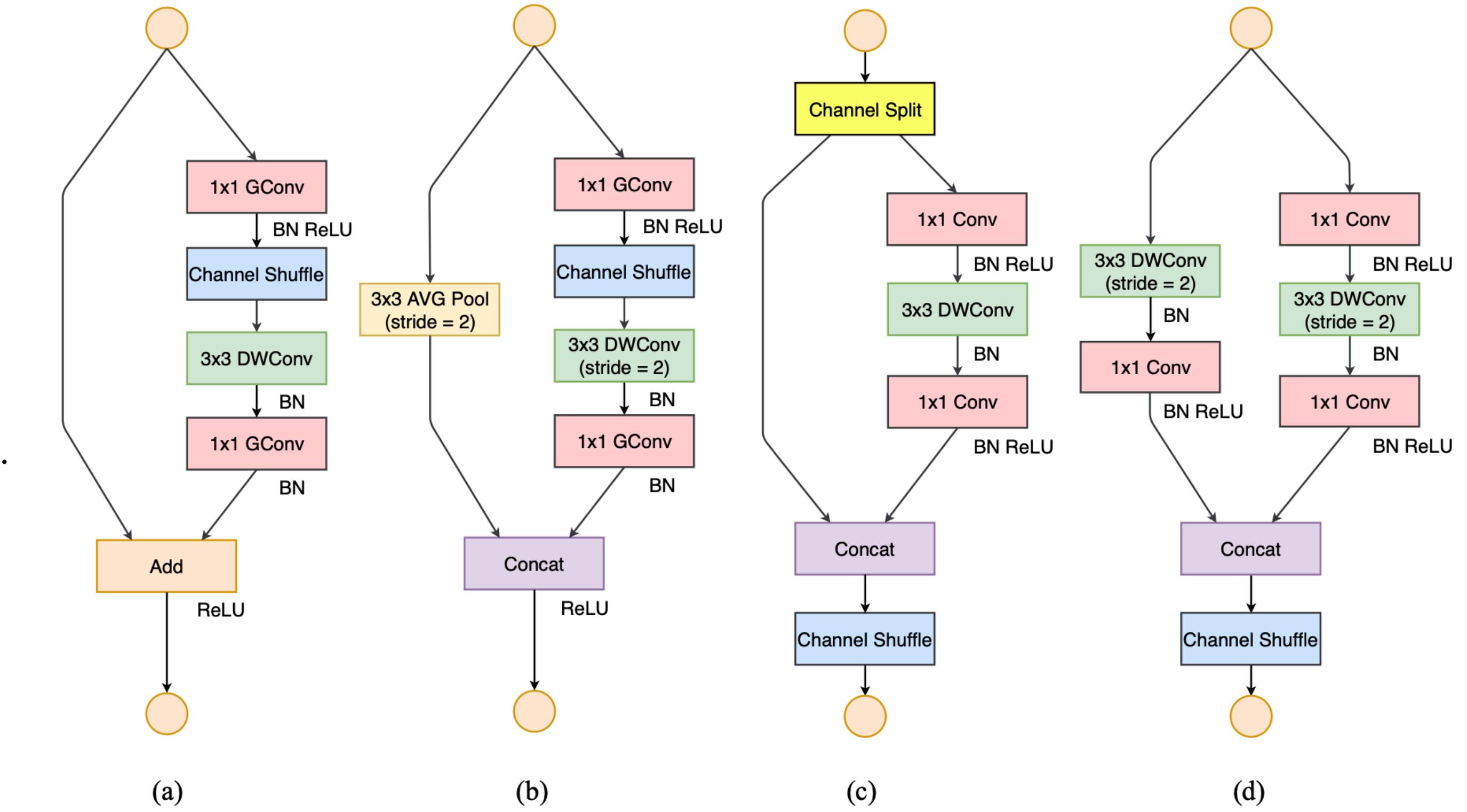
(d) ARM

G3) Network fragmentation (e.g., Inception modules) reduces degree of parallelism.

G4) Element-wise operations are non-negligible (high MAC/FLOPs ratio).

- ShuffleNet v1 violates G2 (group convolutions) & G1 (bottleneck-like building blocks)
- MobileNet v2 violates G1 (inverted bottleneck structure) & G4 (depth-wise convolutions)
- auto-generated structures violate G3

high MAC/FLOPs ratio



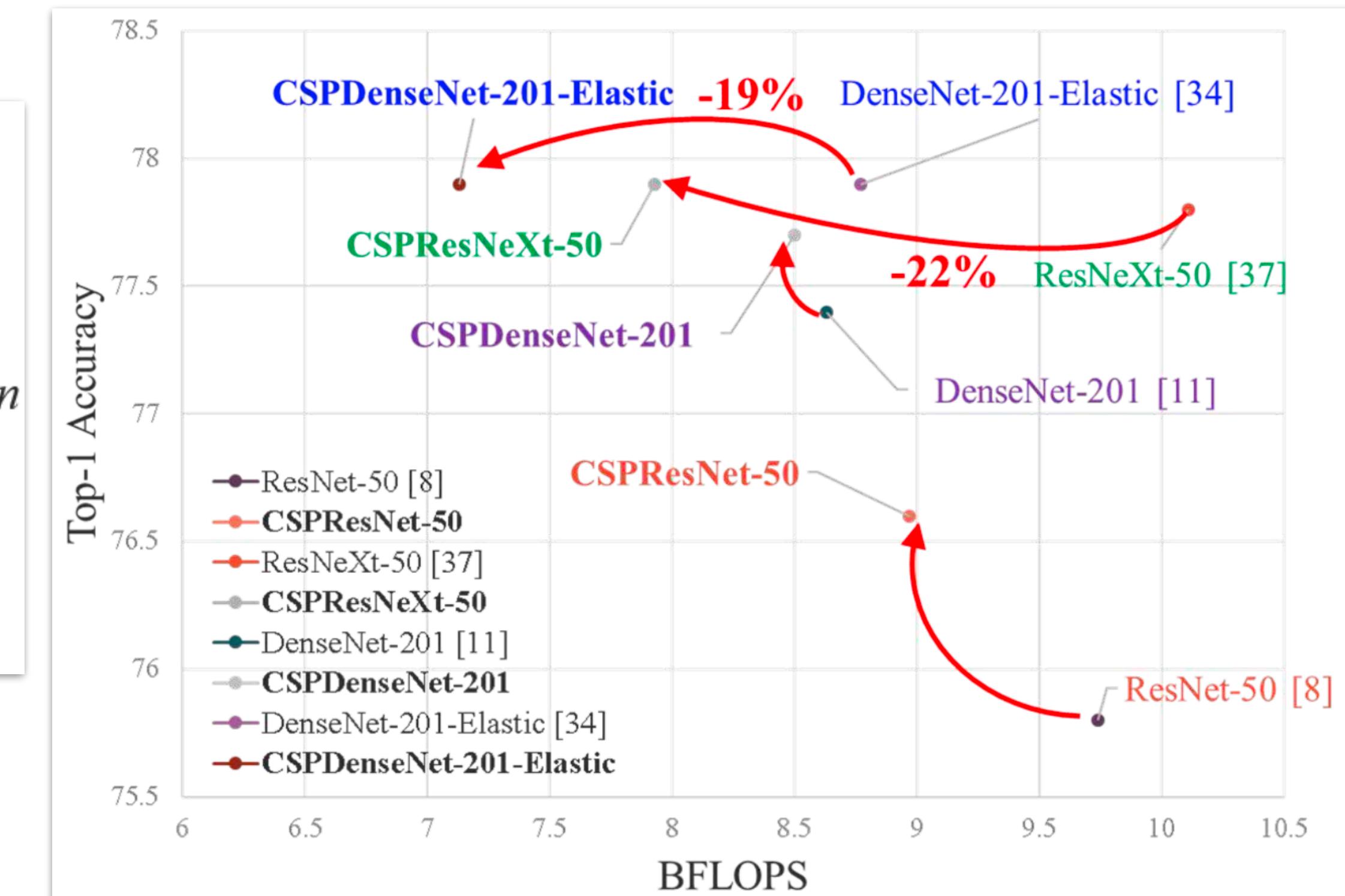
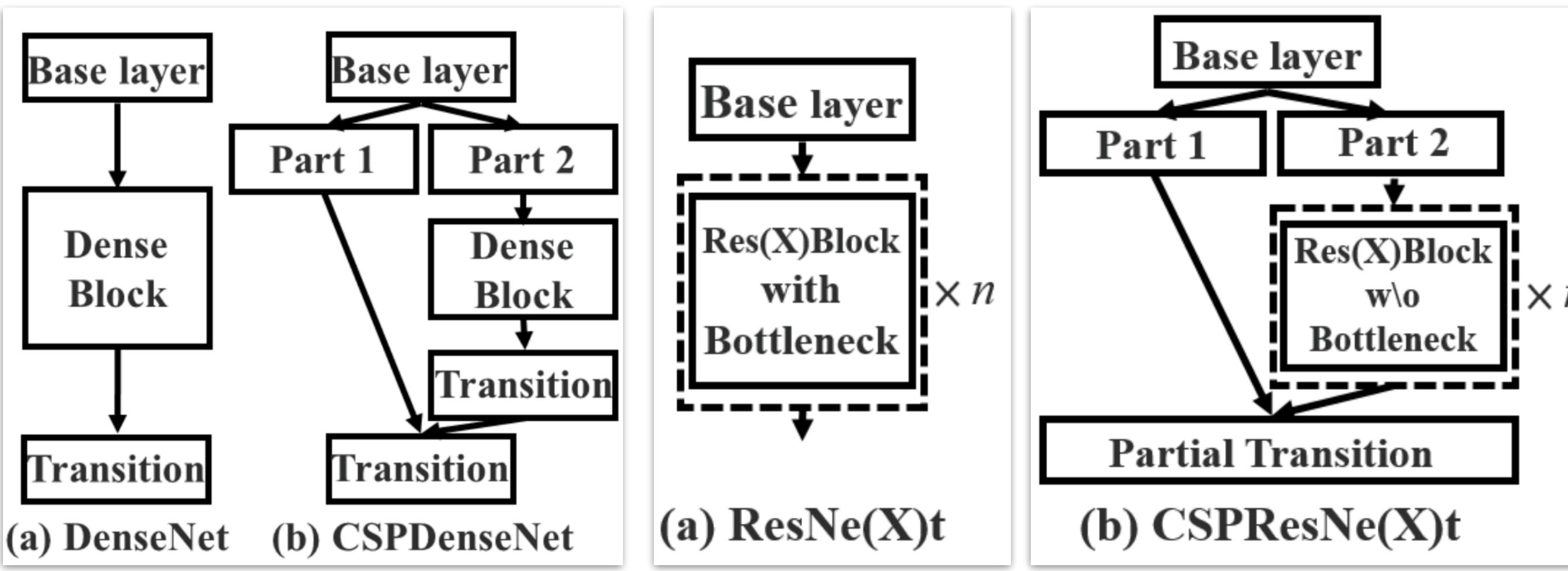
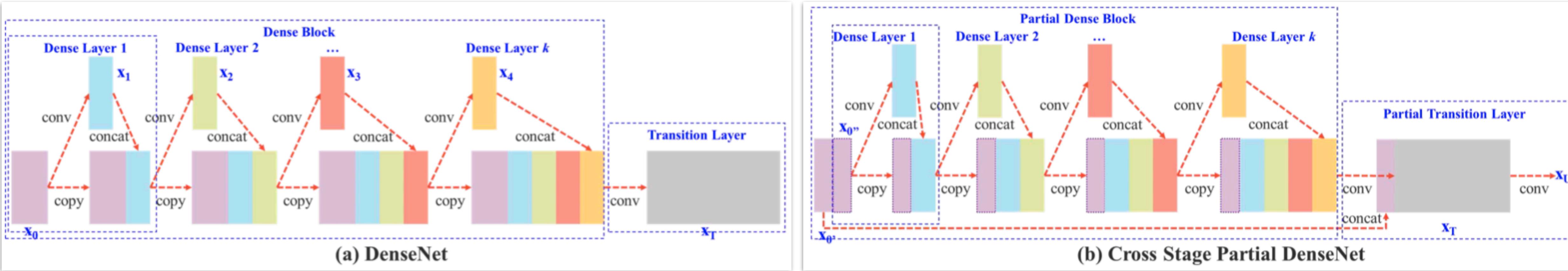
Channel Split

The input of c feature channels are split into two branches with $c - c'$ and c' channels ($c' = c/2$).



Boulder

CSPNet: A New Backbone that can Enhance Learning Capability of CNN





Boulder



Questions?

[YouTube Playlist](#)
