ECE 598NSG/498NSU
Deep Learning in Hardware
Fall 2020

Low-complexity DNNs

Naresh Shanbhag

Department of Electrical and Computer Engineering

University of Illinois at Urbana-Champaign

http://shanbhag.ece.uiuc.edu

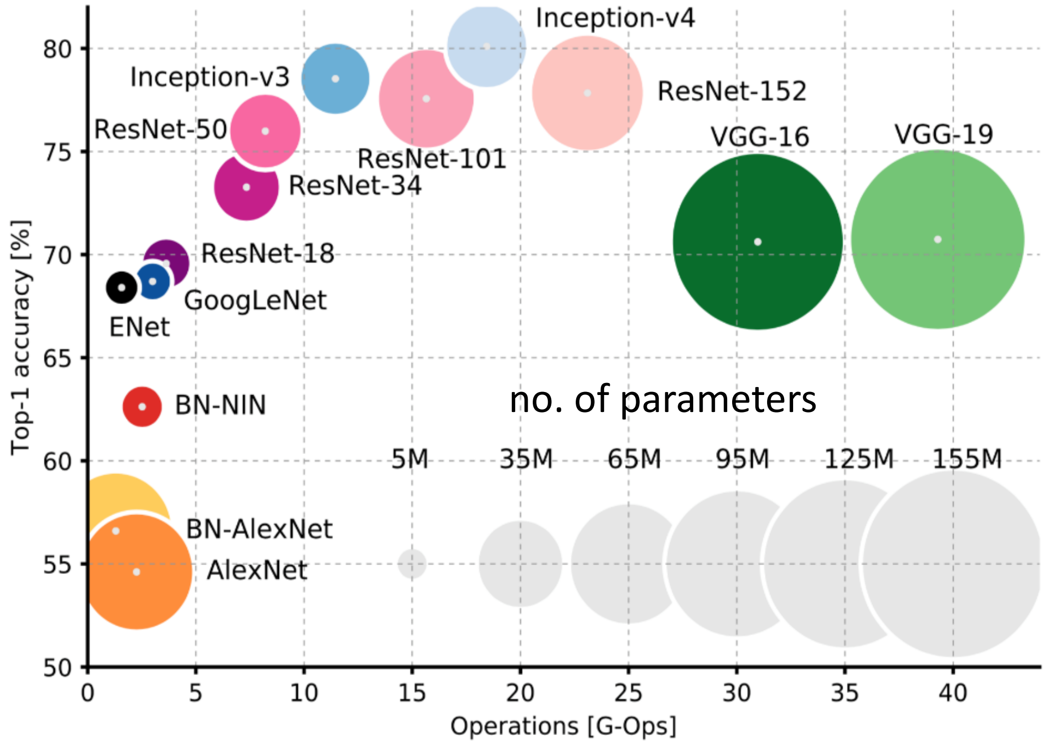
Today

- Motivation
- Low-complexity network - MobileNet
- Low-complexity network – SqueezeNet
- Low-complexity network – ShuffleNet

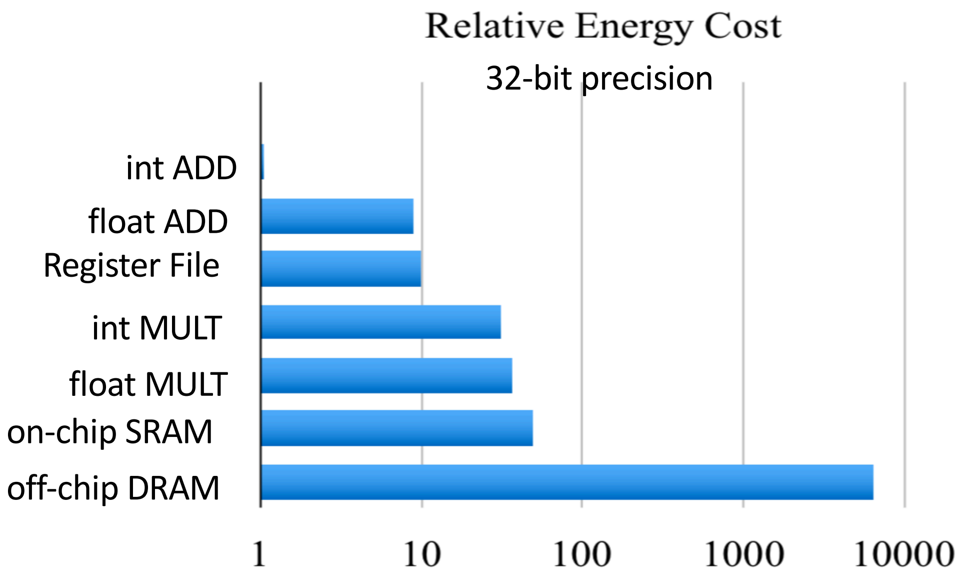
Motivation

- faster training
- easier '*over the air*' updates to Edge devices
- fewer off-chip accesses during inference – lower latency and energy costs of inference
  - e.g., FPGA on-chip memory < 10 MB

Memory Access Energy in DNNs



[Canziani-arXiv-2017]



[Horowitz-ISSCC-2014]

- Require large no. of parameters → don't fit within on-chip SRAM
- off-chip DRAM accesses are 100x more energy expensive

Low-Complexity DNNs

- Two approaches
- 1: design a low-complexity network from scratch
  – based on design intuitions (MobileNet, SqueezeNet)
- 2: reduce the complexity of a large network
  – model compression
  – factorization
  – distillation
  – binarization/ternarization
- need to address complexity vs. accuracy trade-off

Three Low-Complexity Networks

- MobileNet
- SqueezeNet
- ShuffleNet

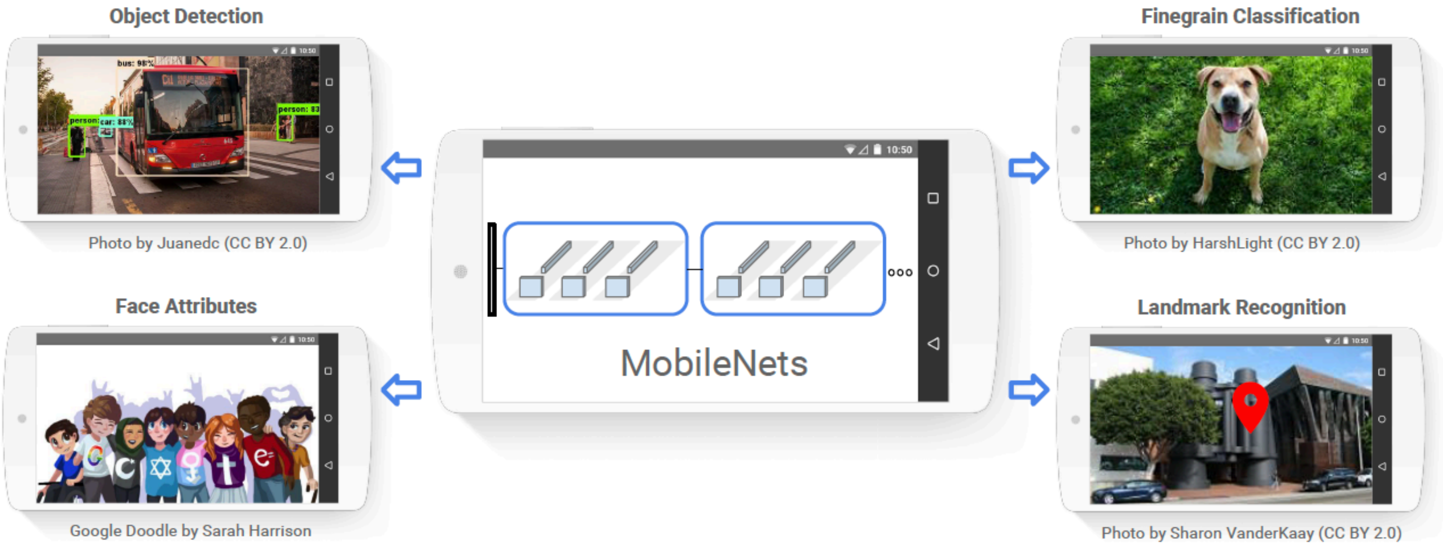
MobileNet

MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications

Andrew G. Howard   Menglong Zhu   Bo Chen   Dmitry Kalenichenko
Weijun Wang   Tobias Weyand   Marco Andreetto   Hartwig Adam

Google Inc.

intrinsically a low-complexity network → embedded vision apps.



	# of Layers	# of parameters CONV	# of parameters FC	# of MACs CONV	# of MACs FC	Top-1 error %	Top-5 error %
AlexNet	5C – 3F	3.7M	58.6M	1,077M	58.6M	42.9	15.3
VGGNet	13/16/19C – 3F	14.7M	123.6M	15,360M	123.6M	28.07	9.33
ResNet	18/34/50/101/152C – 1F	21M	512K	3,643M	512K	21.53	5.6
DenseNet	120/160/168/200C – 1F	~ 52M	1.152M	**> 7B**	1.152M	20.85	5.3
MobileNet	27C – 1F	3.1M	1M	532M	1M	29.4	11.022

[source: Dbouk]

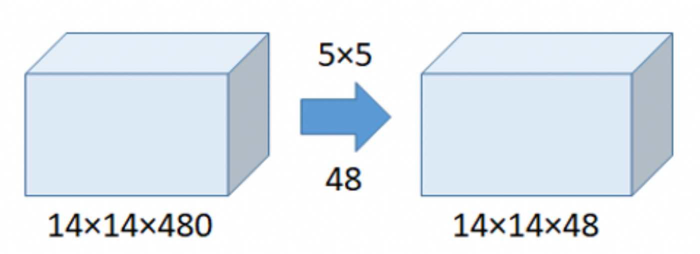
- Objective: to design small, low-latency models for embedded platforms
- 5X smaller model complexity than ResNet but higher error rates

- optimizes size and latency
- parametrized network -> can design many versions
- two parameters: width and resolution multipliers
- built from

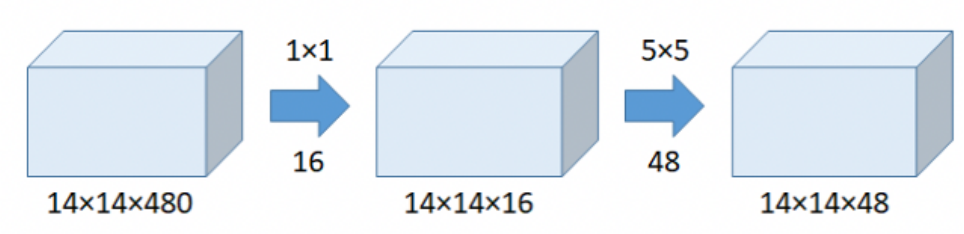
depth-wise separable convolutions →

standard convolution = depth-wise separable x point-wise convolutions

- depth-wise separable convolutions – first proposed in (Sifre, Ph.D. thesis, 2014) – use pointwise convolutions to reduce input dimension of larger filters
- Used by Inception modules in GoogleNet



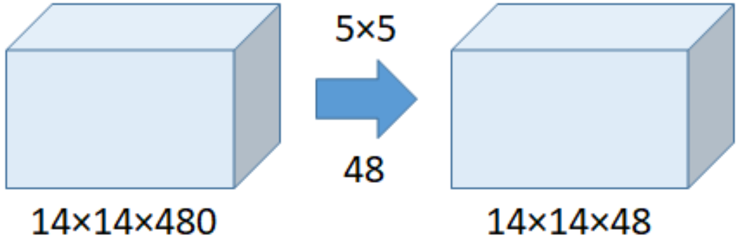
total #MACs = 112.9M



total #MACs = 5.3M

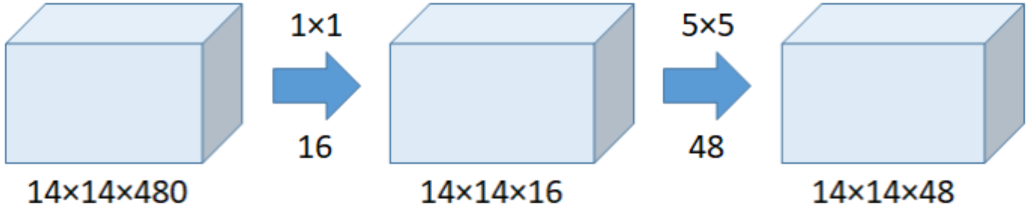
- depth-wise separable convolutions – first proposed in (Sifre, Ph.D. thesis, 2014) → factorize convolutions to save complexity
- better than reducing size of filters for accuracy
- Used by Inception modules in GoogleNet

standard convolution



total #MACs = 112.9M

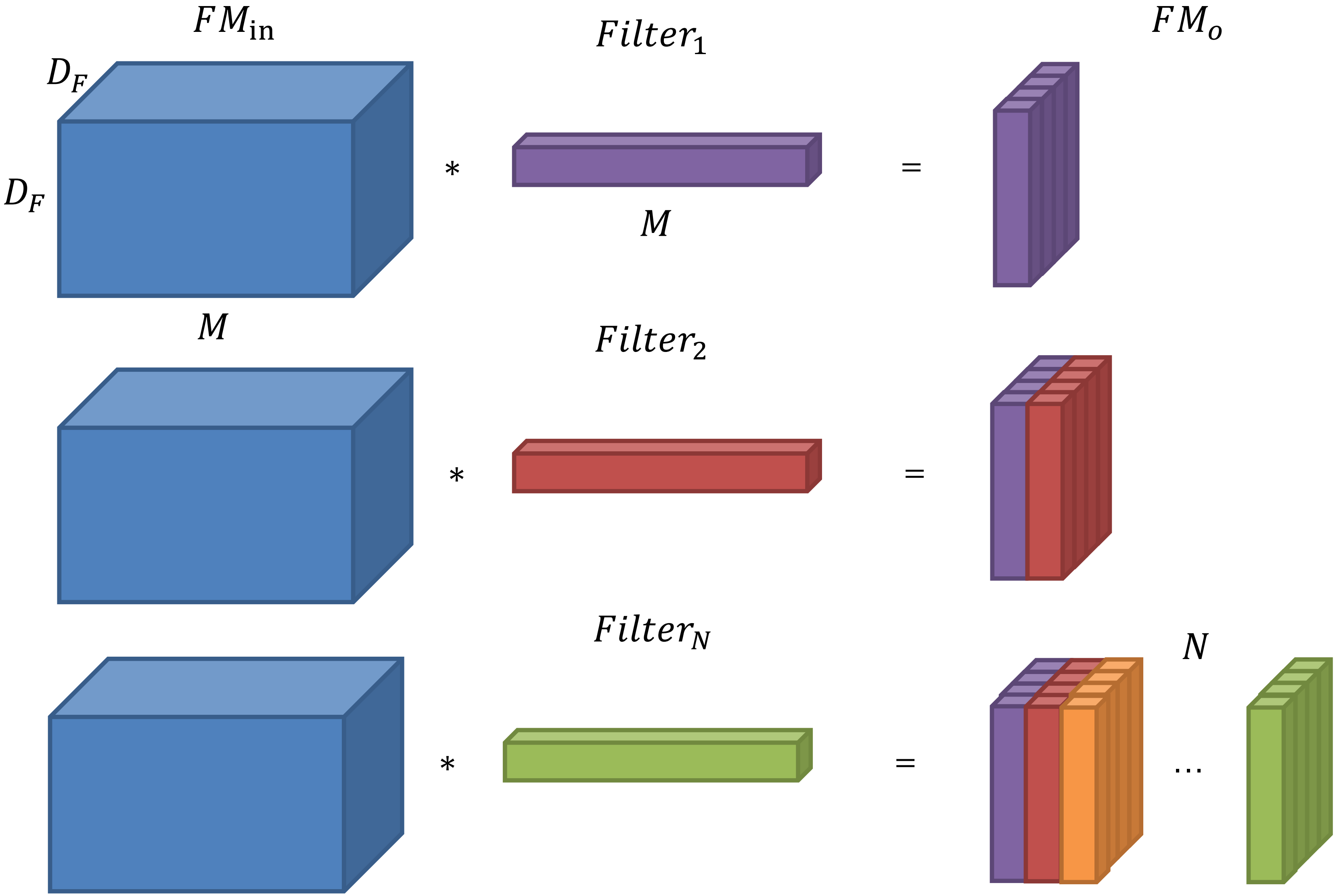
factorized convolution



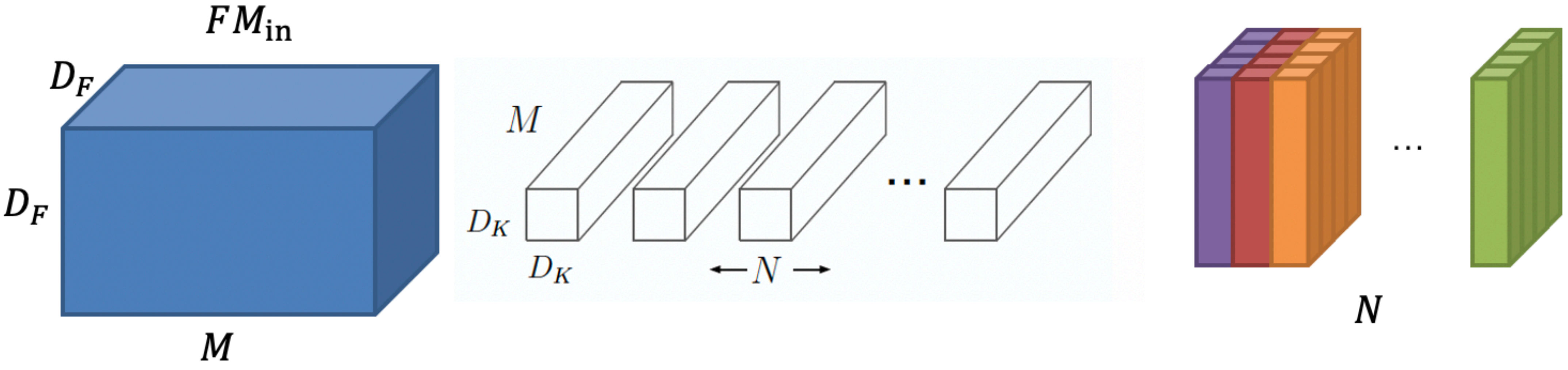
total #MACs = 5.3M

input and output FM sizes in both are the same → reduces impact on accuracy

Standard Convolution



Standard Convolution Filters

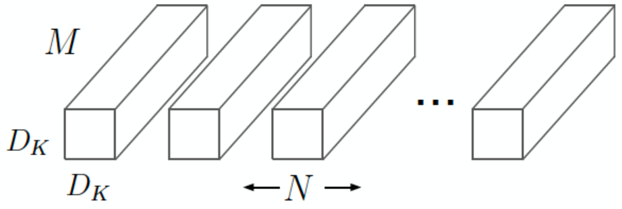


- Kernel size: $D_K^2 \times M \times N$
- Computational cost: $D_K^2 \times M \times N \times D_F^2$

Depth-wise Separable Convolutions

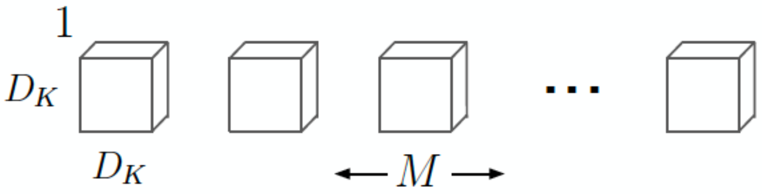
- breaks the relationship between N and kernel size
- depth-wise separable convolution = depth-wise convolution $\times$ point-wise convolution

standard



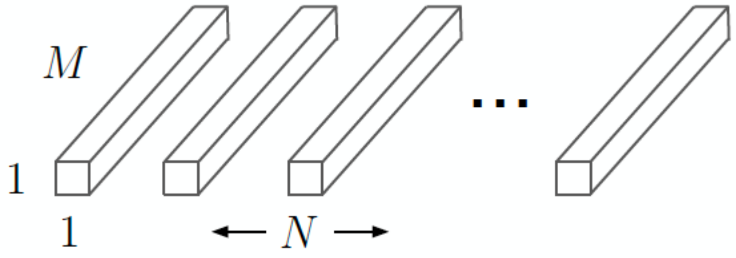
=

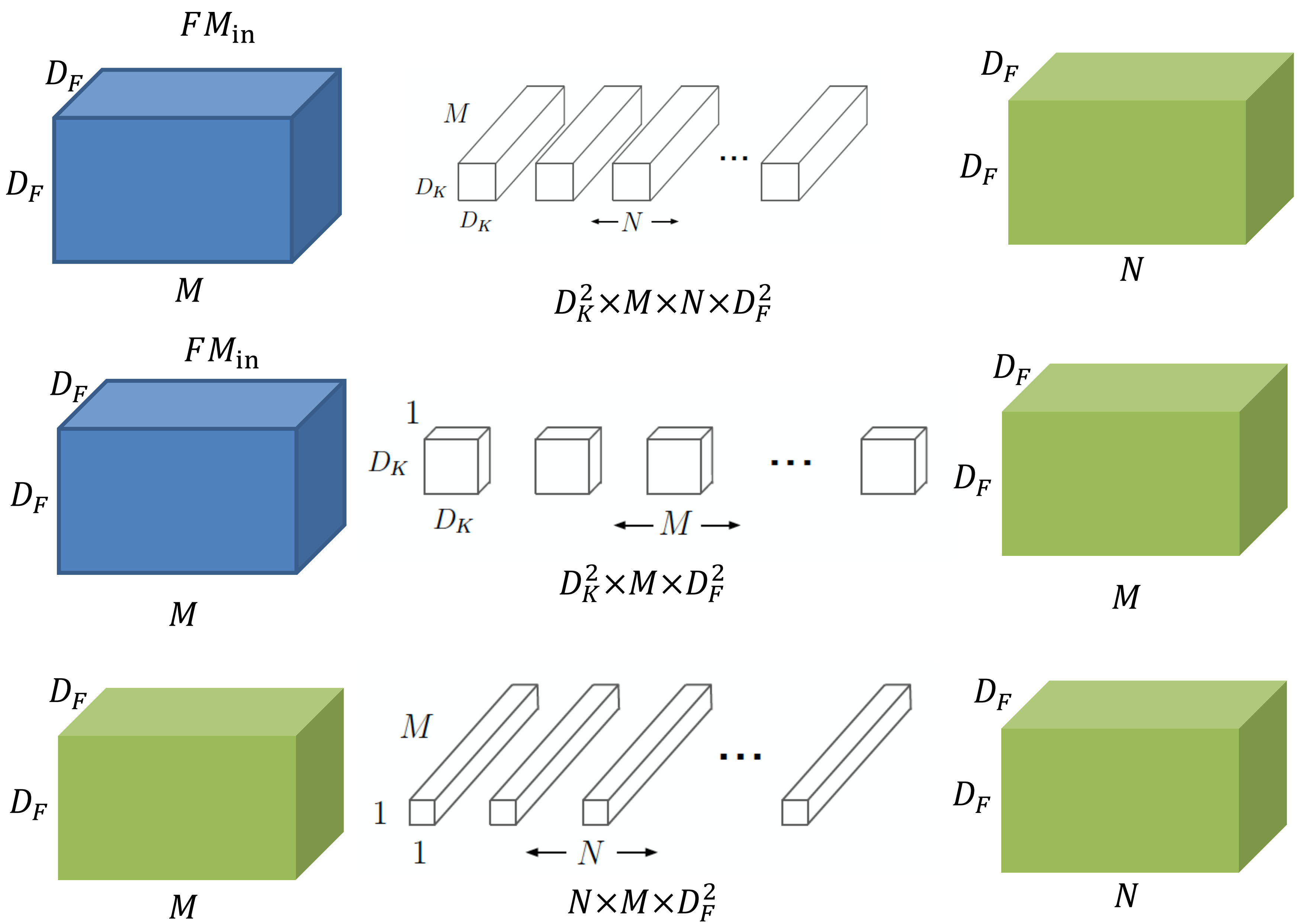
depth-wise (in-plane conv)



X

point-wise (cross-plane conv)





FM_{in}
D_F
D_F
M
M
D_K
D_K
N
$D_K^2 \times M \times N \times D_F^2$
D_F
D_F
N
FM_{in}
D_F
D_F
M
1
D_K
D_K
M
$D_K^2 \times M \times D_F^2$
D_F
D_F
M
D_F
D_F
M
M
1
1
N
$N \times M \times D_F^2$
D_F
D_F
N

Computational Cost

- Standard: $D_K^2 \times M \times N \times D_F^2$

- Depth-wise separable: $D_K^2 \times M \times D_F^2 + N \times M \times D_F^2$
- second term usually dominates

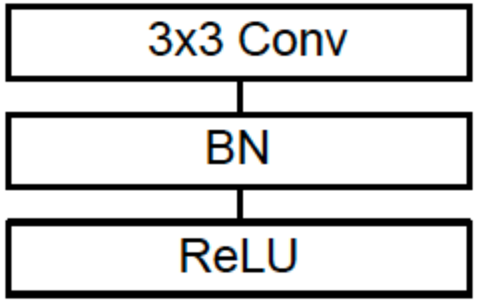
- Savings:

$$\frac{D_K^2 \times M \times D_F^2 + N \times M \times D_F^2}{D_K^2 \times M \times N \times D_F^2} = \frac{1}{N} + \frac{1}{D_K^2}$$

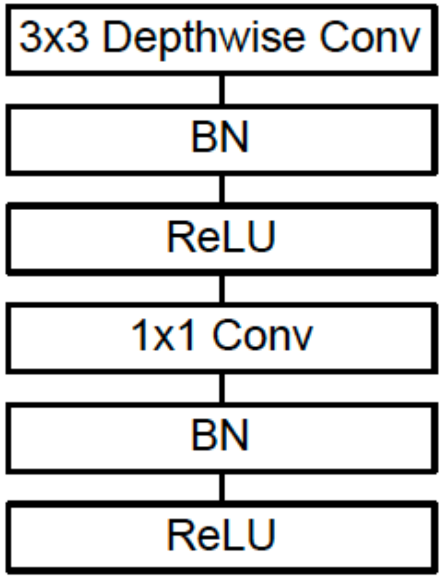
- MobileNet: $D_K = 3 \rightarrow 8\times$-to-$9\times$ savings in complexity

Network Architecture

Standard CONV layer



MobileNet CONV layer

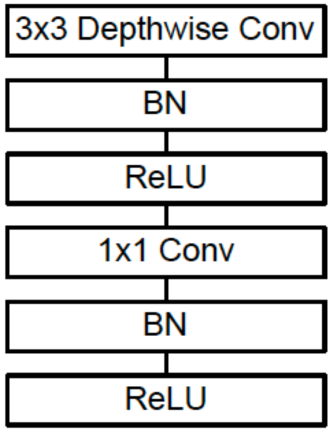


- MobileNet Layer 1 is standard CONV

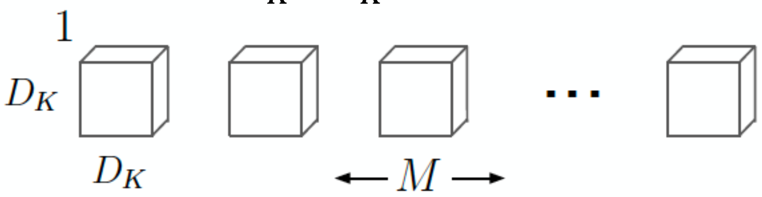
MobileNet Architecture – 28 Layers

Table 1. MobileNet Body Architecture

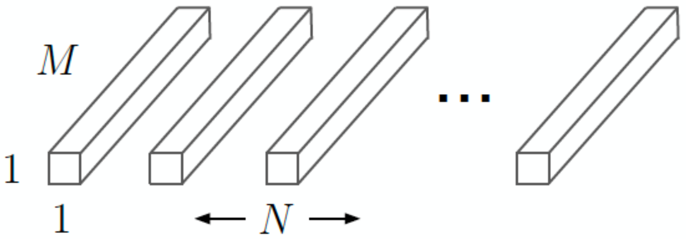
Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
5× Conv dw / s1 Conv / s1	$3 \times 3 \times 512$ dw $1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$ $14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ 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	$1 \times 1 \times 1000$



$D_K \times D_K \times M$



$1 \times 1 \times M \times N$



Computational Costs by Layer Type

Table 2. Resource Per Layer Type

Type	Mult-Adds	Parameters
Conv 1×1	94.86%	74.59%
Conv DW 3×3	3.06%	1.06%
Conv 3×3	1.19%	0.02%
Fully Connected	0.18%	24.33%

- most of the cost in point-wise convolutions → 95% computation and 75% of storage
- point-wise convolutions = dot-products

Parameterizing MobileNet

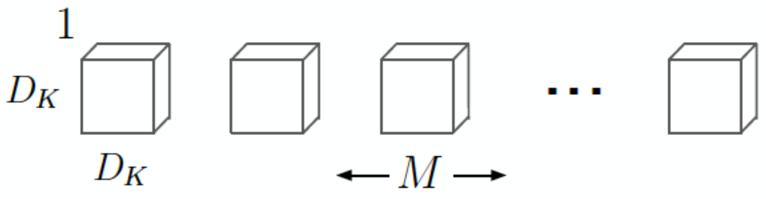
- how to generate MobileNets with different size, accuracy, latency trade-offs?

- parameterize the network with:
  - width multiplier – scales the number of channels
  - resolution multiplier – scales the 2D size of the FMs

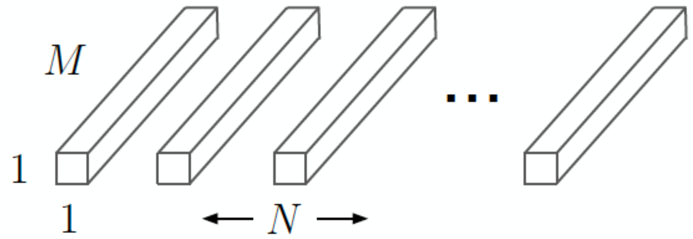
Width Multiplier

- width multiplier ($\alpha \in (0,1]$): $N \rightarrow \alpha N$; $M \rightarrow \alpha M$ ($\alpha = 1$: baseline)

depth-wise



point-wise

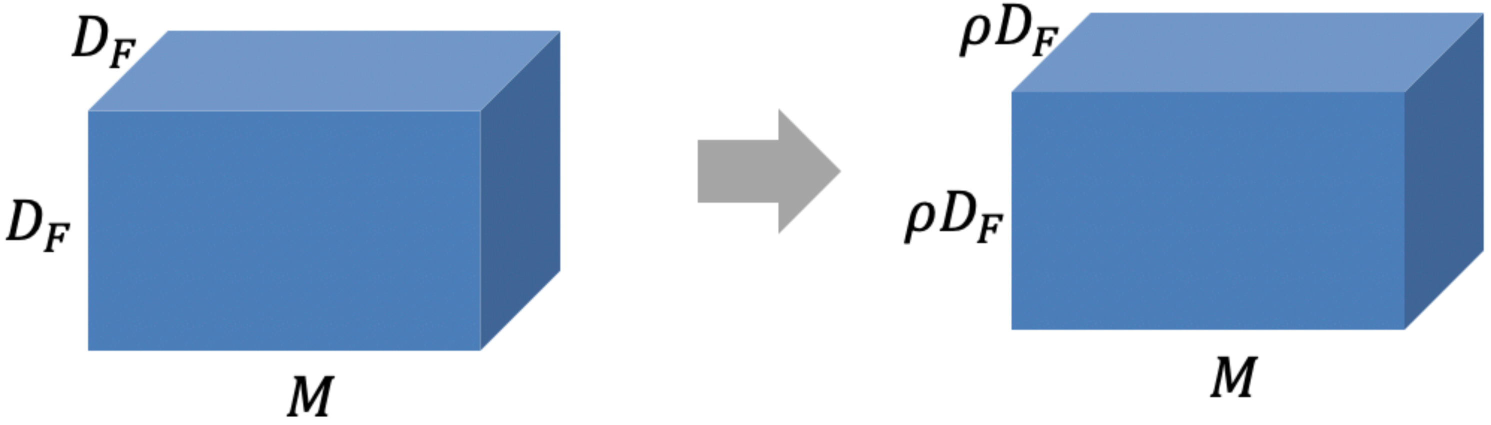


- computational cost reduced roughly by α^2

$$D_K^2 \times M \times D_F^2 + N \times M \times D_F^2 \rightarrow D_K^2 \times \alpha M \times D_F^2 + \alpha N \times \alpha M \times D_F^2$$

Resolution Multiplier

- resolution multiplier ($\rho \in (0,1]$): $D_F \rightarrow \rho D_F$; ($\rho = 1$: baseline) – implicitly set via assigning input resolution



- computational cost: reduced roughly by ρ^2

$$D_K^2 \times M \times D_F^2 + N \times M \times D_F^2 \rightarrow D_K^2 \times M \times \rho^2 D_F^2 + N \times M \times \rho^2 D_F^2$$

Total Cost with Both Multipliers

Table 3. Resource usage for modifications to standard convolution. Note that each row is a cumulative effect adding on top of the previous row. This example is for an internal MobileNet layer with $D_K = 3$, $M = 512$, $N = 512$, $D_F = 14$.

Layer/Modification	Million Mult-Adds	Million Parameters
Convolution	462	2.36
Depthwise Separable Conv	52.3	0.27
$\alpha = 0.75$	29.6	0.15
$\rho = 0.714$	15.1	0.15

- Baseline: $D_K^2 \times M \times D_F^2 + N \times M \times D_F^2$
- Reduced: $D_K^2 \times \alpha M \times \rho^2 D_F^2 + \alpha N \times \alpha M \times \rho^2 D_F^2$
- overall: computational cost reduced by $(\alpha\rho)^2$; storage cost by $(\alpha)^2$

Accuracy vs. Size

Table 4. Depthwise Separable vs Full Convolution MobileNet

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
Conv MobileNet	71.7%	4866	29.3
MobileNet	70.6%	569	4.2

Table 5. Narrow vs Shallow MobileNet

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
0.75 MobileNet	68.4%	325	2.6
Shallow MobileNet	65.3%	307	2.9

- Shallow MobileNet – remove 5 layers with $14\times14\times512$ FMs
- narrow network preserves depth $\rightarrow$ good for accuracy

Accuracy vs. Size

Table 6. MobileNet Width Multiplier

Width Multiplier	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
0.75 MobileNet-224	68.4%	325	2.6
0.5 MobileNet-224	63.7%	149	1.3
0.25 MobileNet-224	50.6%	41	0.5

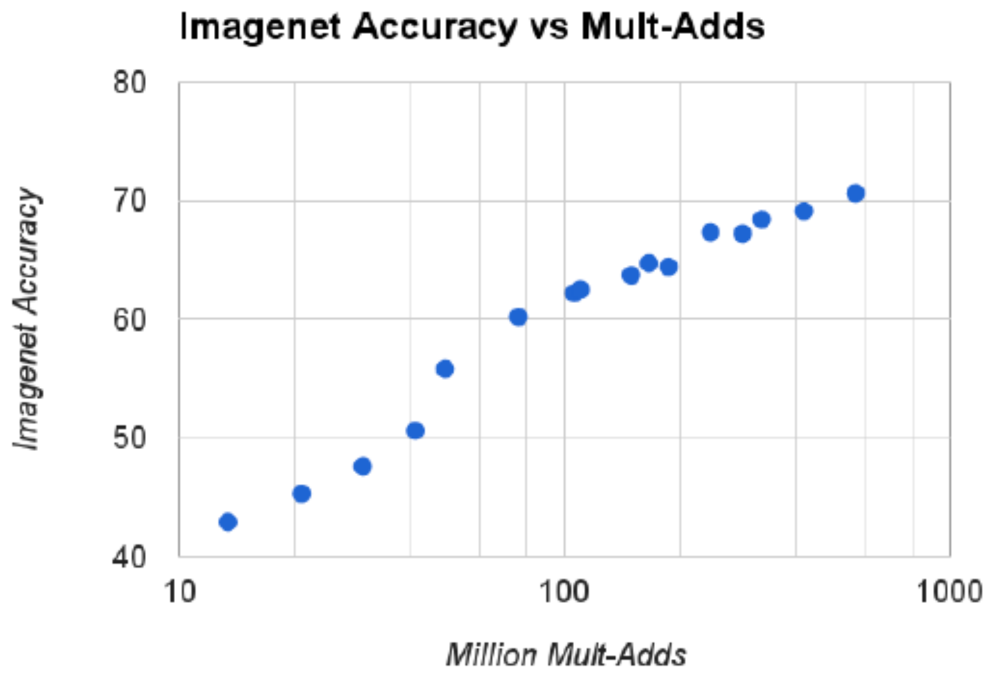
sharp drop

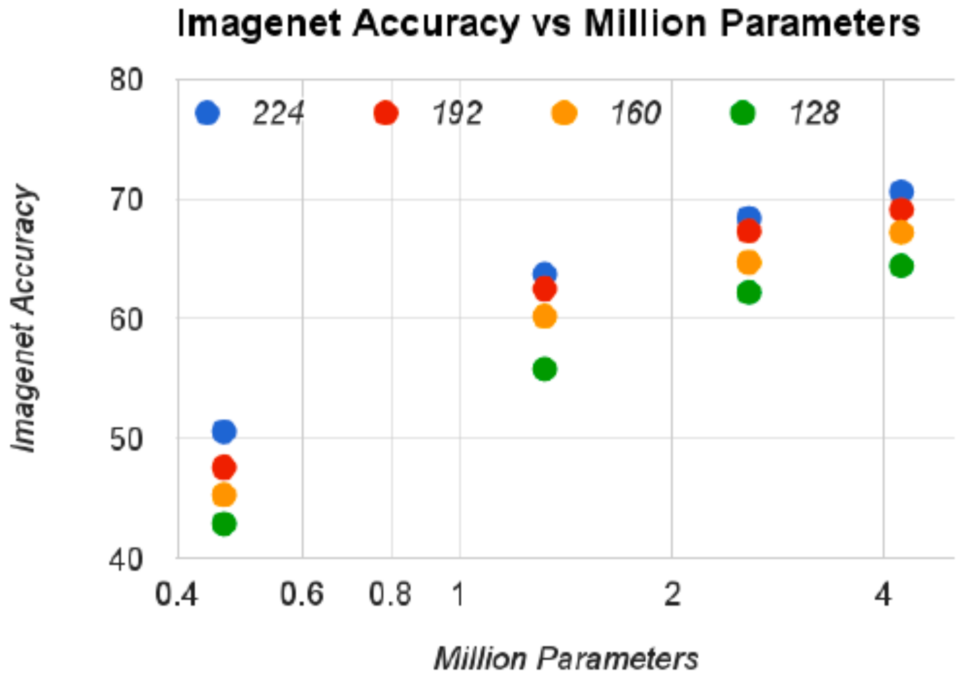
Table 7. MobileNet Resolution

Resolution	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
1.0 MobileNet-192	69.1%	418	4.2
1.0 MobileNet-160	67.2%	290	4.2
1.0 MobileNet-128	64.4%	186	4.2

smooth roll-off

Accuracy vs. Storage/Complexity Trade-off





log-linear dependence **log-linear dependence**

- Networks generated as a cross-product:

$$\alpha \in \{1, 0.75, 0.5, 0.25\} \times \rho \in \{224, 192, 160, 128\}$$

SqueezeNet

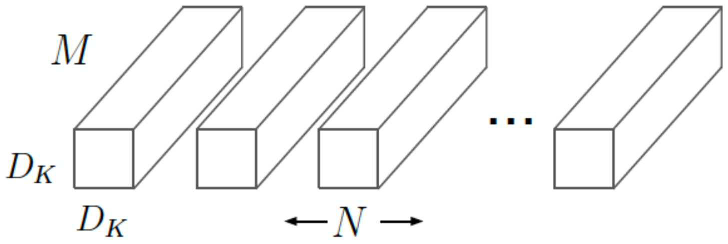
SQUEEZENET: ALEXNET-LEVEL ACCURACY WITH 50X FEWER PARAMETERS AND <0.5MB MODEL SIZE

Forrest N. Iandola[1], Song Han[2], Matthew W. Moskewicz[1], Khalid Ashraf[1], William J. Dally[2], Kurt Keutzer[1]
[1]DeepScale* & UC Berkeley [2]Stanford University

- another low-complexity network
- based on a set of design intuitions to reduce network size

Design Strategies

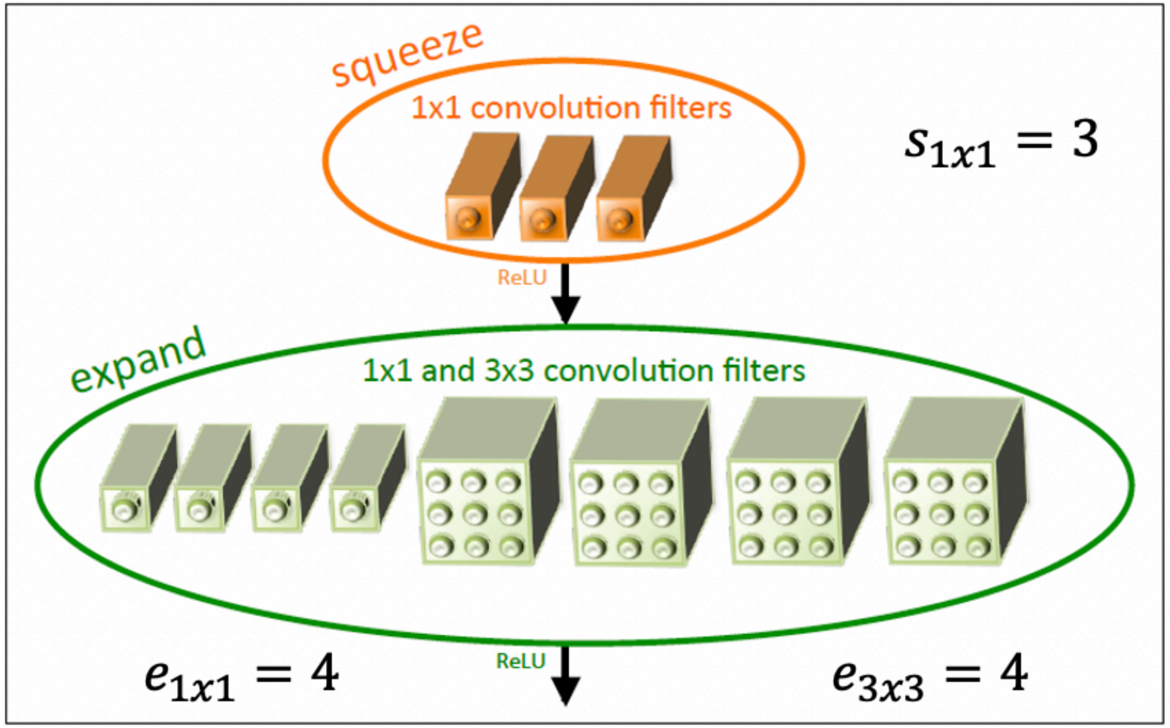
- **S1**: replace $3{\times}3$ filters with $1{\times}1$ filters → $9{\times}$ fewer parameters
- **S2**: decrease the number of input channels to $3{\times}3$ filters → reduces the number of parameters (width multiplier)



reduce M when $D_K = 3$

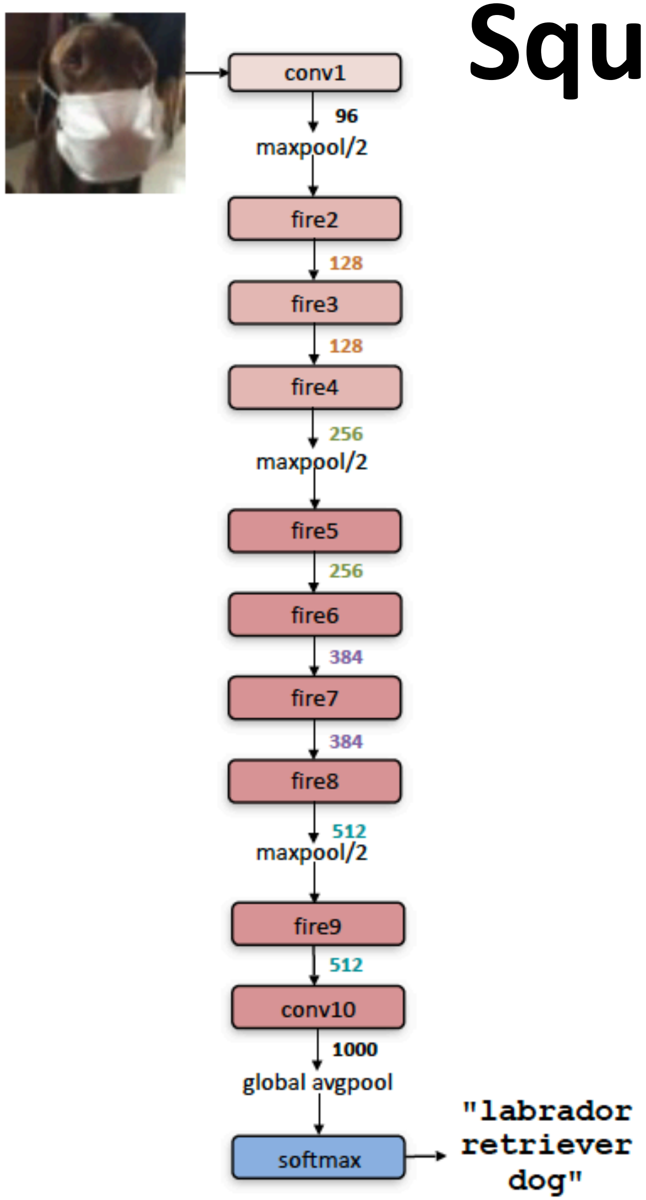
- **S3**: downsample late in network → conv layers have large activation maps → better accuracy

Fire Module



- concatenation of squeeze (**S1**) and expand layers
- parameterized by 3 variables: $s_{1x1}, e_{1x1}, e_{3x3}$
- Set: $s_{1x1} < e_{1x1} + e_{3x3}$ per **S2**

SqueezeNet Architecture



- Conv1: standard; no FC layer
- 8 Fire modules (fire2-9)
- max pooling after conv1, fire4, fire8, conv10 (per **S3**)

Table 1: SqueezeNet architectural dimensions. (The formatting of this table was inspired by the Inception2 paper (Ioffe & Szegedy, 2015).)

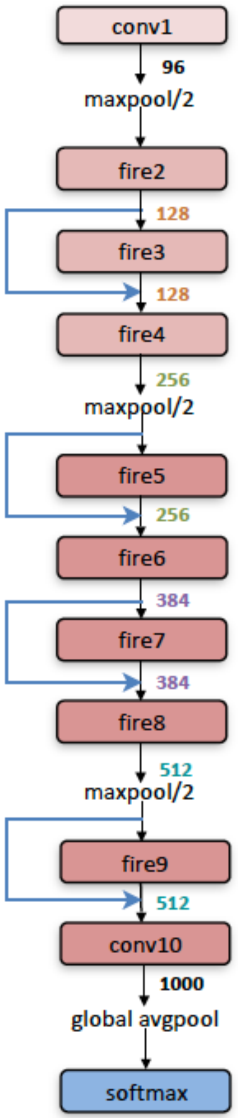
layer name/type	output size	filter size / stride (if not a fire layer)	depth	s_{1x1} (#1x1 squeeze)	e_{1x1} (#1x1 expand)	e_{3x3} (#3x3 expand)	s_{1x1} sparsity	e_{1x1} sparsity	e_{3x3} sparsity	# bits	#parameter before pruning	#parameter after pruning
input image	224x224x3										-	-
conv1	111x111x96	7x7/2 (x96)	1				100% (7x7)			6bit	14,208	14,208
maxpool1	55x55x96	3x3/2	0									
fire2	55x55x128		2	16	64	64	100%	100%	**33%**	6bit	11,920	5,746
fire3	55x55x128		2	16	64	64	100%	100%	**33%**	6bit	12,432	6,258
fire4	55x55x256		2	32	128	128	100%	100%	**33%**	6bit	45,344	20,646
maxpool4	27x27x256	3x3/2	0									
fire5	27x27x256		2	32	128	128	100%	100%	**33%**	6bit	49,440	24,742
fire6	27x27x384		2	48	192	192	100%	**50%**	**33%**	6bit	104,880	44,700
fire7	27x27x384		2	48	192	192	**50%**	100%	**33%**	6bit	111,024	46,236
fire8	27x27x512		2	64	256	256	100%	**50%**	**33%**	6bit	188,992	77,581
maxpool8	13x12x512	3x3/2	0									
fire9	13x13x512		2	64	256	256	**50%**	100%	**30%**	6bit	197,184	77,581
conv10	13x13x1000	1x1/1 (x1000)	1				**20%** (3x3)			6bit	513,000	103,400
avgpool10	1x1x1000	13x13/1	0									
	$\underbrace{}_{\text{activations}}$	$\underbrace{}_{\text{parameters}}$					$\underbrace{}_{\text{compression info}}$				1,248,424 (total)	**421,098** (total)

Design Intuitions

- add 1 pixel zero-padding to input of 3×3 expand filters → 3×3 and 1×1 filters output activations have identical dimensions
- ReLU is applied to activations from squeeze and expand layers (Nair & Hinton, 2010)
- dropout (Srivastava et. al., 2014) with ratio 50% after fire9
- no FC layer (inspired by NiN Lin et al., 2013)
- linearly decrease learning rate from 0.04 down (Mishkin et al., 2016)
- expand layer implemented in Caffe as concatenation of 1×1 filters and 3×3 filters

SqueezeNet Variants

with simple bypass



- similar to ResNet
- simple bypass applies to fire modules with same # of input and output dimensions
- complex bypass applied to fire modules with differing input/output dimensions (adds extra parameters)
- Squeeze layers limit information; bypass layers compensate for it

with complex bypass

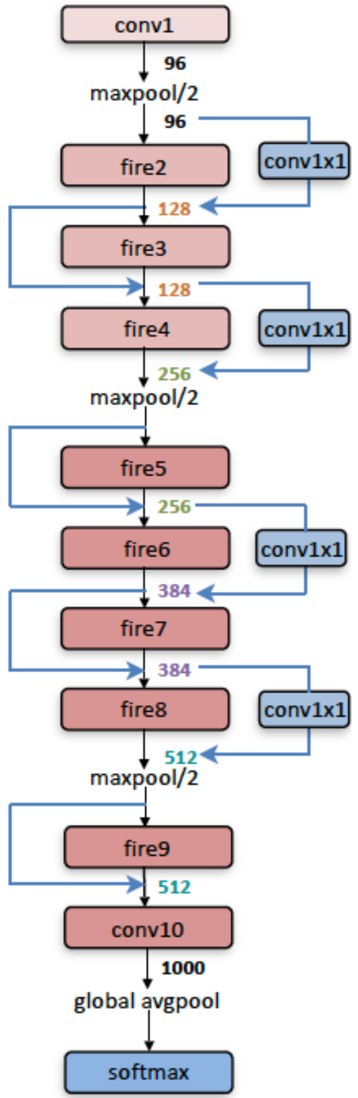


Table 3: SqueezeNet accuracy and model size using different macroarchitecture configurations

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

- also applied model compression -> low-complexity model is further compressible

ShuffleNet

ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices

Xiangyu Zhang* Xinyu Zhou* Mengxiao Lin Jian Sun
Megvii Inc (Face++)
{zhangxiangyu,zxy,linmengxiao,sunjian}@megvii.com

- Another low complexity network for mobile devices
- Channel shuffle and ShuffleNet Unit

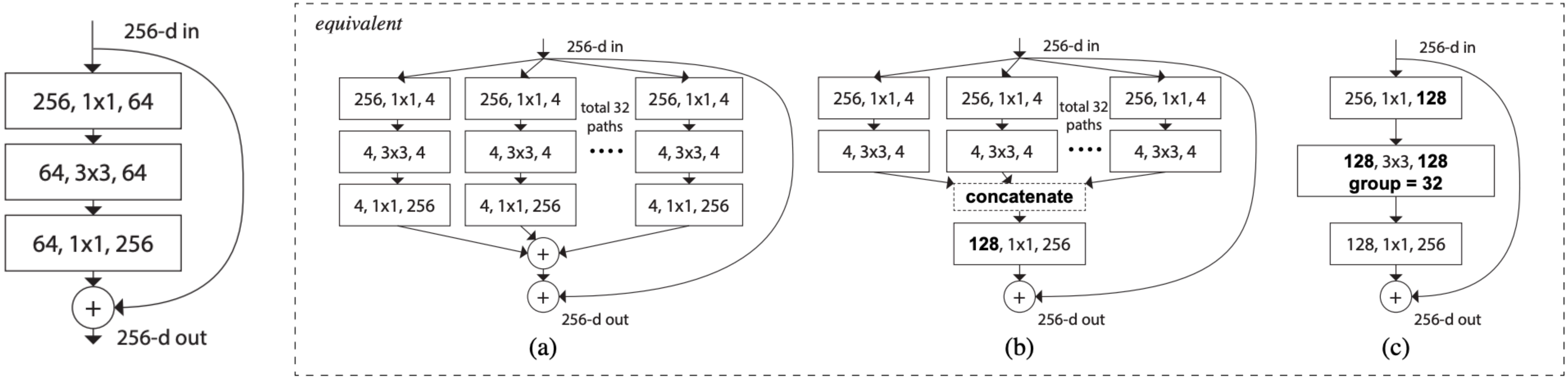
Design Strategy

- Repurpose two principles:
  - Use of repeated blocks of same shape (VGG and ResNet)
  - Use of split-transform-merge strategy (Inception models – GoogleLeNet, SqueezeNet)
- Channel Shuffle: related input and output channels by cross talking
- Replace standard convolution with 1x1 Point-wise **Group** convolution and 3x3 Depth-wise Separable convolution

Group Convolution

[Xie et al., ‘17, ResNext]

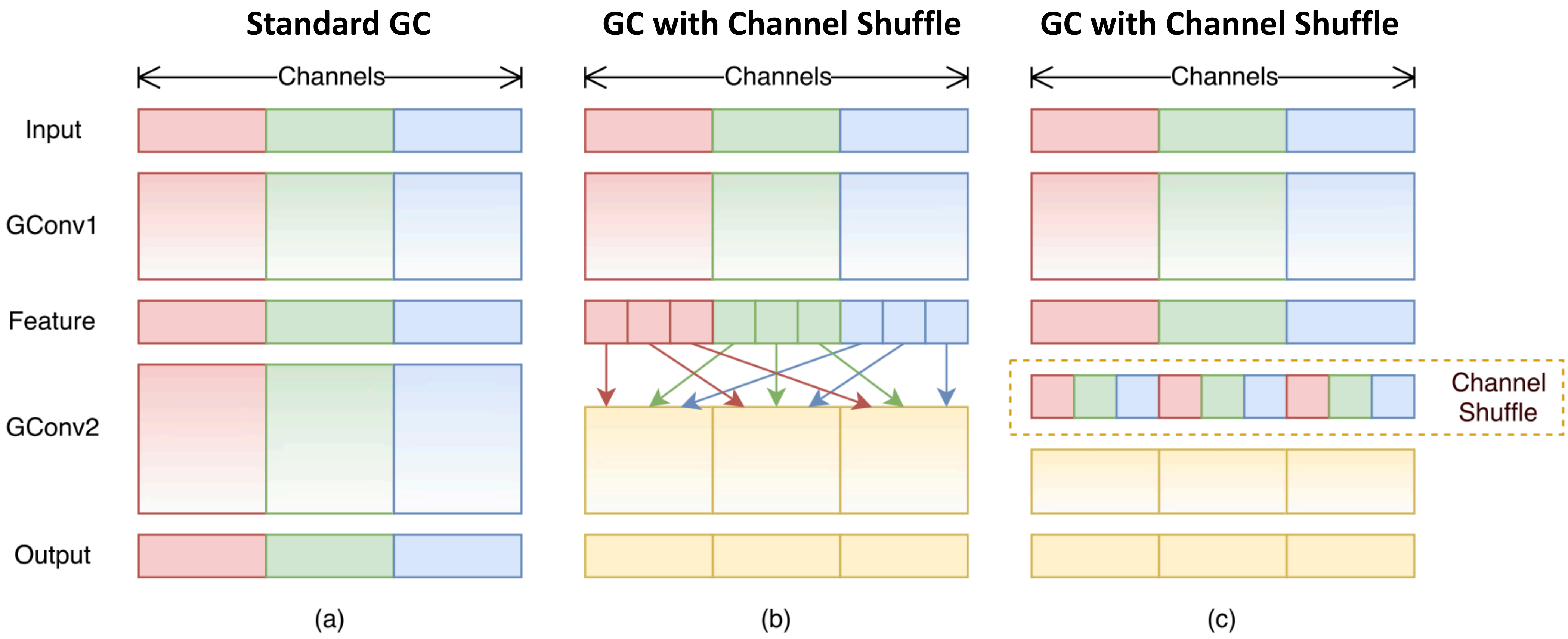
Standard ResNet block **ResNext block with $C = 32$**



block equivalent ***GC***

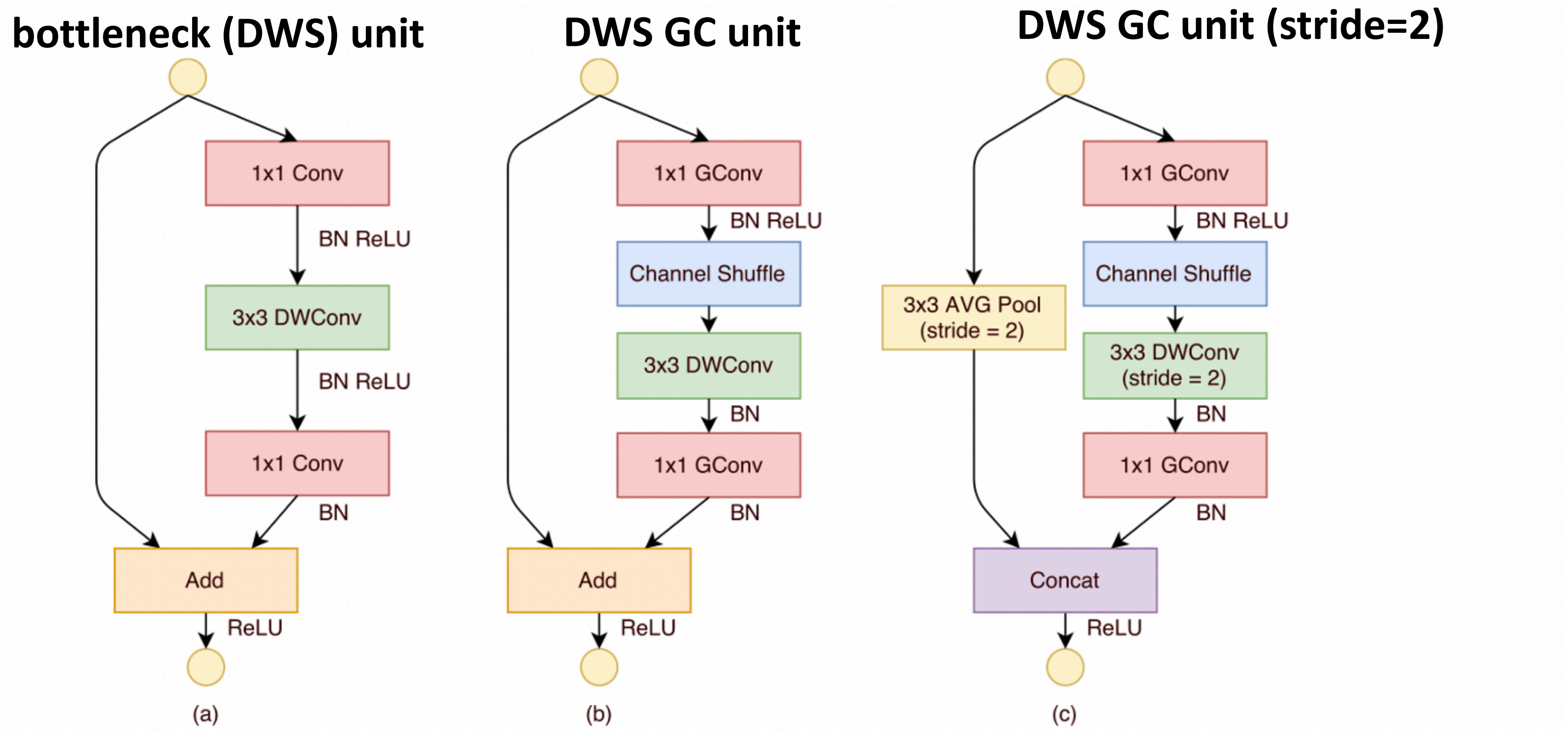
- used by AlexNet to distribute computations across GPUs; used by ResNext to enhance accuracy without increasing computational cost (use of cardinality)
- GC: split input FM into C (cardinality) groups; each group responsible for certain depth; concatenate (sum) each group at the end

Channel Shuffle



- allow information flow between channel groups and strengthen representation
- differentiable operation, meaning end-to-end training is available

ShuffleNet Units



- Consist of 1x1 group convolution followed by channel shuffle, 3x3 depthwise separable Convolution, and 1x1 group convolution
- Batch Normalization and ReLU after each convolution except the last 1x1 GConv
- Element-wise addition or concatenation to inputs at the final step

ShuffleNet Architecture

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

- Conv1: standard
- 3 stages of ShuffleNet unit groups
  - Each contains ShuffleNet units with stride = 1 and stride = 2
  - The number of units in each stage is specified by the number of repeats

Accuracy vs. Use of Channel Shuffle

Model	Cls err. (%, no shuffle)	Cls err. (%, shuffle)	Δ err. (%)
ShuffleNet 1x ($g = 3$)	34.5	**32.6**	1.9
ShuffleNet 1x ($g = 8$)	37.6	**32.4**	5.2
ShuffleNet 0.5x ($g = 3$)	45.7	**43.2**	2.5
ShuffleNet 0.5x ($g = 8$)	48.1	**42.3**	5.8
ShuffleNet 0.25x ($g = 3$)	56.3	**55.0**	1.3
ShuffleNet 0.25x ($g = 8$)	56.5	**52.7**	3.8

- Small number represents better accuracy
- Channel Shuffle provide 6% decrease in classification error

Accuracy vs. Architecture

Complexity (MFLOPs)	VGG-like	ResNet	Xception-like	ResNeXt	ShuffleNet (ours)
140	50.7	37.3	33.6	33.3	**32.4** ($1\times$, $g = 8$)
38	-	48.8	45.1	46.0	**41.6** ($0.5\times$, $g = 4$)
13	-	63.7	57.1	65.2	**52.7** ($0.25\times$, $g = 8$)

Model	Complexity (MFLOPs)	Cls err. (%)	Δ err. (%)
1.0 MobileNet-224	569	29.4	-
ShuffleNet $2\times$ ($g = 3$)	524	**26.3**	3.1
ShuffleNet $2\times$ (with *SE*[13], $g = 3$)	527	**24.7**	4.7
0.75 MobileNet-224	325	31.6	-
ShuffleNet $1.5\times$ ($g = 3$)	292	**28.5**	3.1
0.5 MobileNet-224	149	36.3	-
ShuffleNet $1\times$ ($g = 8$)	140	**32.4**	3.9
0.25 MobileNet-224	41	49.4	-
ShuffleNet $0.5\times$ ($g = 4$)	38	**41.6**	7.8
ShuffleNet $0.5\times$ (shallow, $g = 3$)	40	42.8	6.6

- achieves best performance across all network architectures

Summary

- Trade-off between complexity and accuracy
- Based on design intuitions to preserve accuracy
- separable convolutions
- use of point-wise convolutions
- delaying max-pooling
- others

Course Web Page

https://courses.grainger.illinois.edu/ece598nsg/fa2020/
https://courses.grainger.illinois.edu/ece498nsu/fa2020/

http://shanbhag.ece.uiuc.edu