Model Compression

Joseph E. Gonzalez
Co-director of the RISE Lab
jegonzal@cs.berkeley.edu

What is the Problem Being Solved?

- Large neural networks are costly to deploy
  - Gigaflops of computation, hundreds of MB of storage
- Why are they costly?
  - **Added computation requirements** adversely affect
    - throughput/latency/energy
  - **Added memory requirements** adversely affect
    - download/storage of model parameters (OTA)
    - throughput and latency through caching
    - Energy! (5pJ for SRAM cache read, 640pj for DRAM vs 0.9pJ for a FLOP)

Approaches to “Compressing” Models

- Architectual Compression
  - **Layer Design →** Typically using factorization techniques to reduce storage and computation
  - **Pruning →** Eliminating weights, layers, or channels to reduce storage and computation from large pre-trained models
- Weight Compression
  - **Low Bit Precision Arithmetic →** Weights and activations are stored and computed using low bit precision
  - **Quantized Weight Encoding →** Weights are quantized and stored using dictionary encodings.

ShuffleNet:
An Extremely Efficient Convolutional Neural Network for Mobile Devices
(2017)

Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, Jian Sun

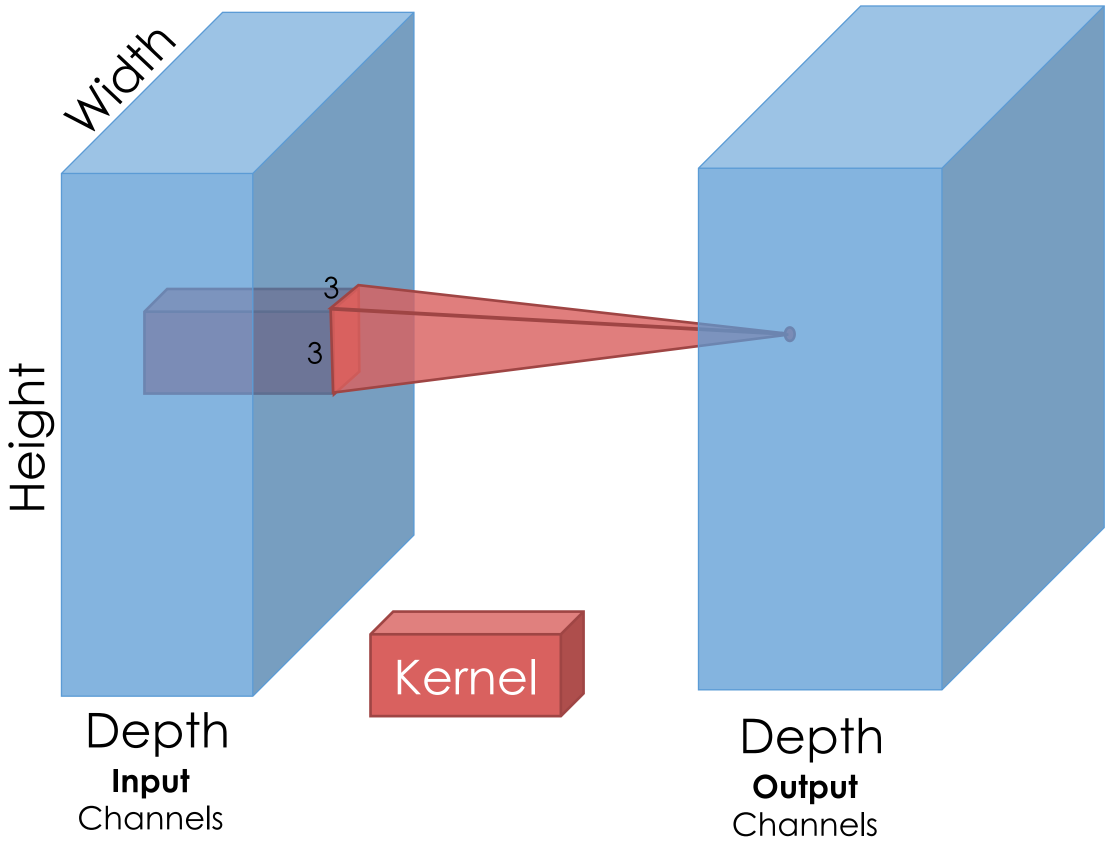
Megvii Inc. (Face++)

Related work (at the time)

- **SqueezeNet** (2016) – Aggressively leverage 1x1 (point-wise) convolution to reduce inputs to 3x3 convolutions.
  - 57.5% Acc (comparable to AlexNet)
  - 1.2M Parameters → compressed down to 0.47MB
- **MobileNetV1** (2017) – Aggressively leverage depth-wise separable convolutions to achieve
  - 70.6 acc on ImageNet
  - 569M – Mult-Adds
  - 4.2M -- Parameters

Background

Regular Convolution



Computation (for 3x3 kernel)

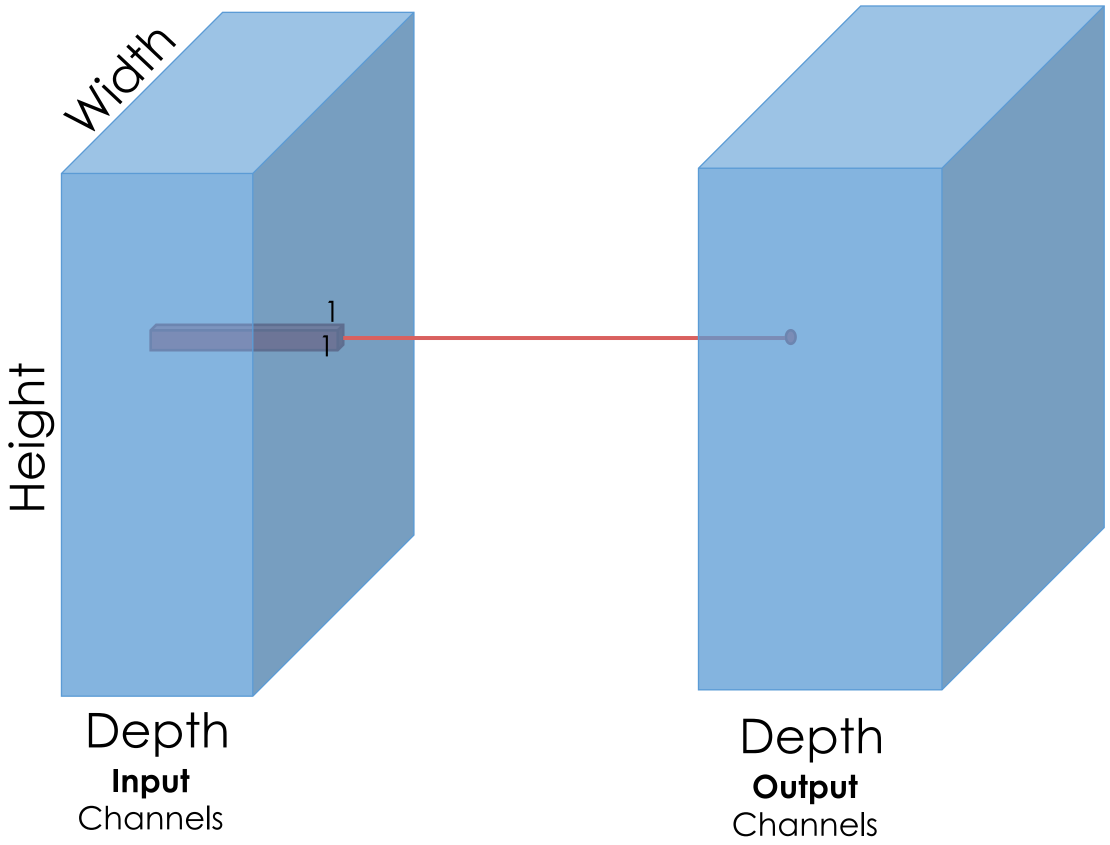
$$Y_{w_o,h_o,c_o} = \sum_{c_i,u,v} K_{c_i,u,v} * X_{w_o+u-1,h_o+v-1,c_i}$$

Computational Complexity:
width, **h**eight, **c**hannel **o**ut, **c**hannel **i**n, **filter size**

$$(w * h * c_o) * (c_i * 3 * 3)$$

Combines information **across space** and **across channels.**

1x1 Convolution (Point Convolution)



Computation

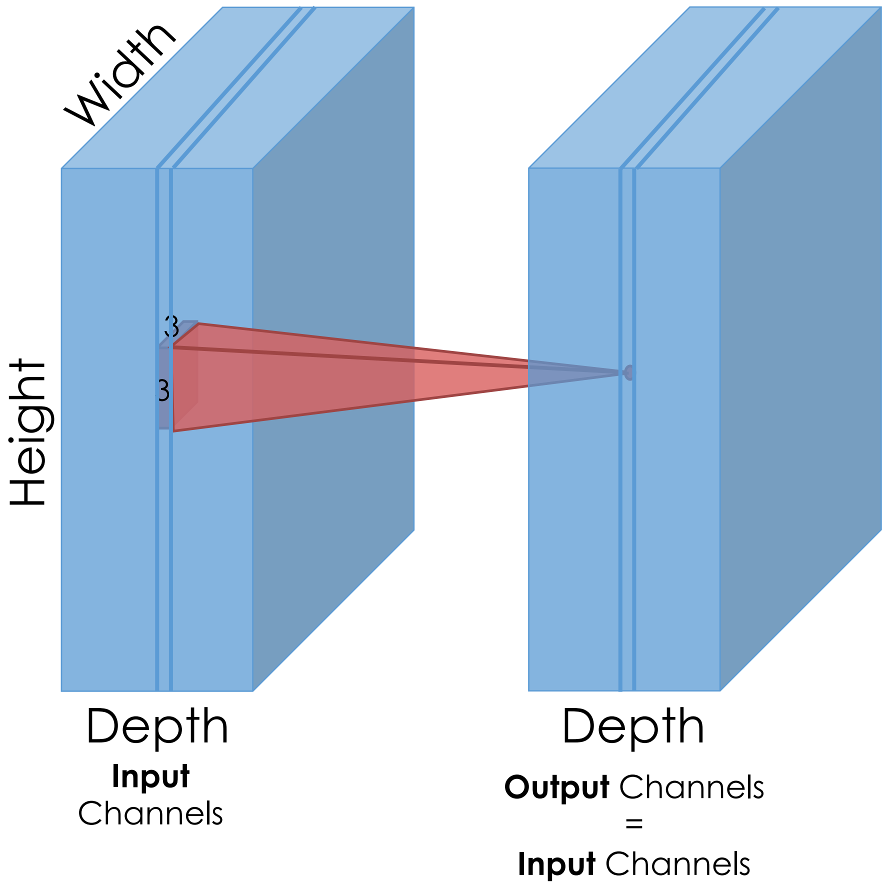
$$Y_{w_o,h_o,c_o} = \sum_{c_i} K_{c_i} * X_{w_o,h_o,c_i}$$

Computational Complexity:

$$(w * h * c_o) * c_i$$

Combines information **across channels *only*.**

Depthwise Convolution



Computation (for 3x3 kernel)

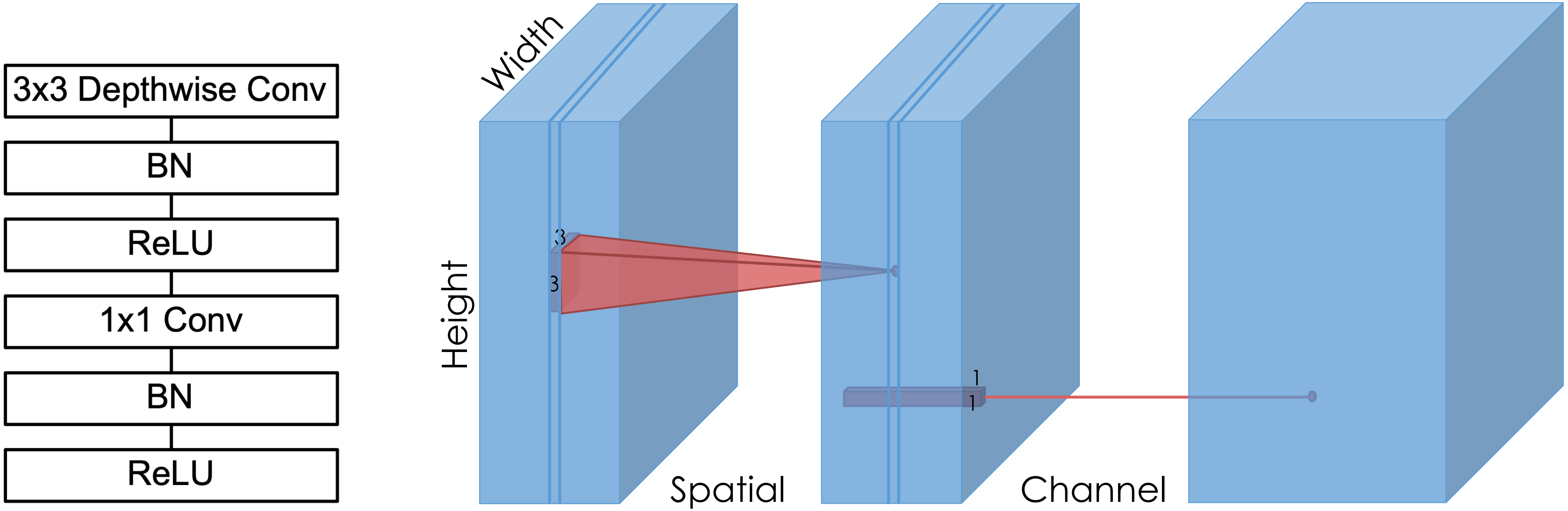
$$Y_{w_o,h_o,c_o} = \sum_{u,v} K_{c_o,u,v} * X_{w_o+u-1,h_o+v-1,c_o}$$

Computational Complexity:

$$(w * h * c_i) * (3 * 3)$$

Combines information **across space *only***

MobileNet Layer Architecture



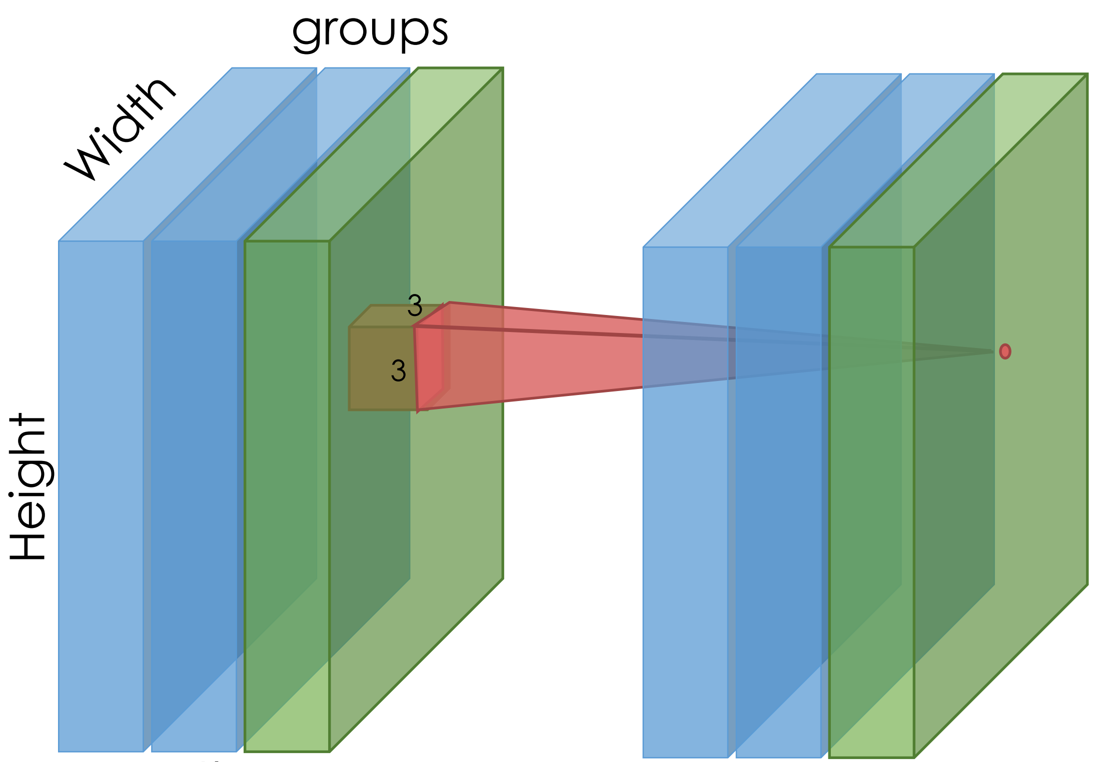
Computational Cost: $(w * h * c_i)(\overbrace{3 * 3}^{\text{Spatial Aggregation}} + \overbrace{c_o}^{\text{Channel Aggregation}})$

Observation from MobileNet Paper

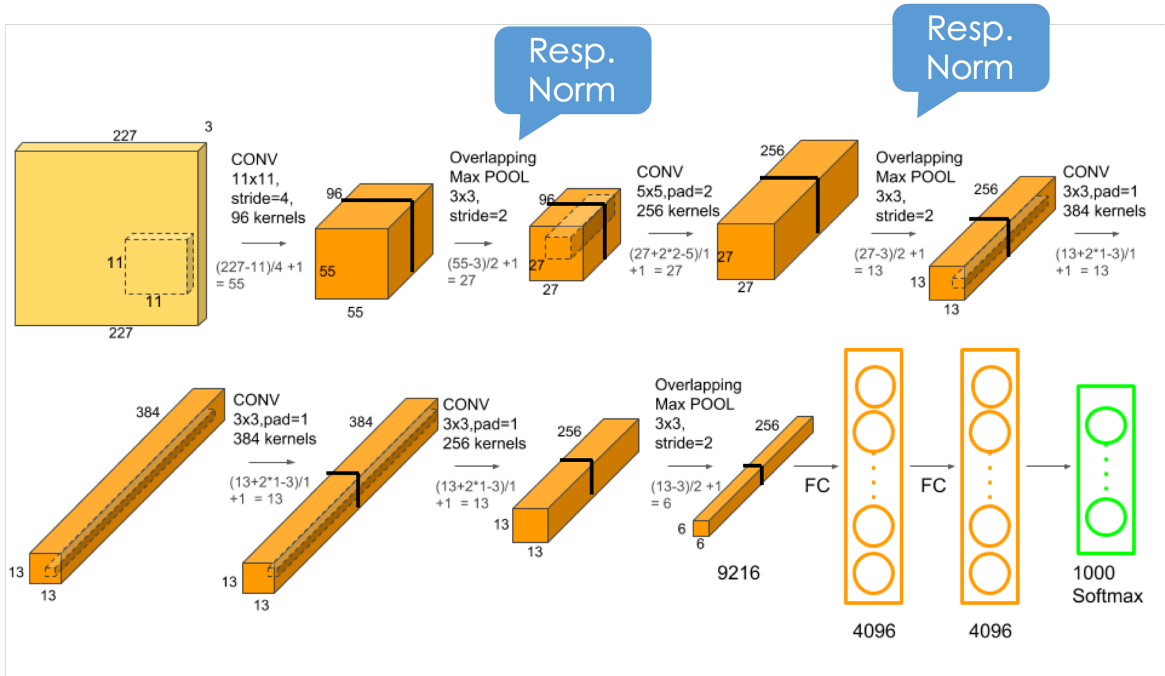
"MobileNet spends 95% of it's computation time in 1x1 convolutions which also has 75% of the parameters as can be seen in Table 2."

- Idea, eliminate the 1x1 conv but still achieve mixing of channel information?
  - Pointwise (1x1) **Group** Convolution
  - **Channel Shuffle**

Group Convolution



Used in AlexNet to partition model across machines.



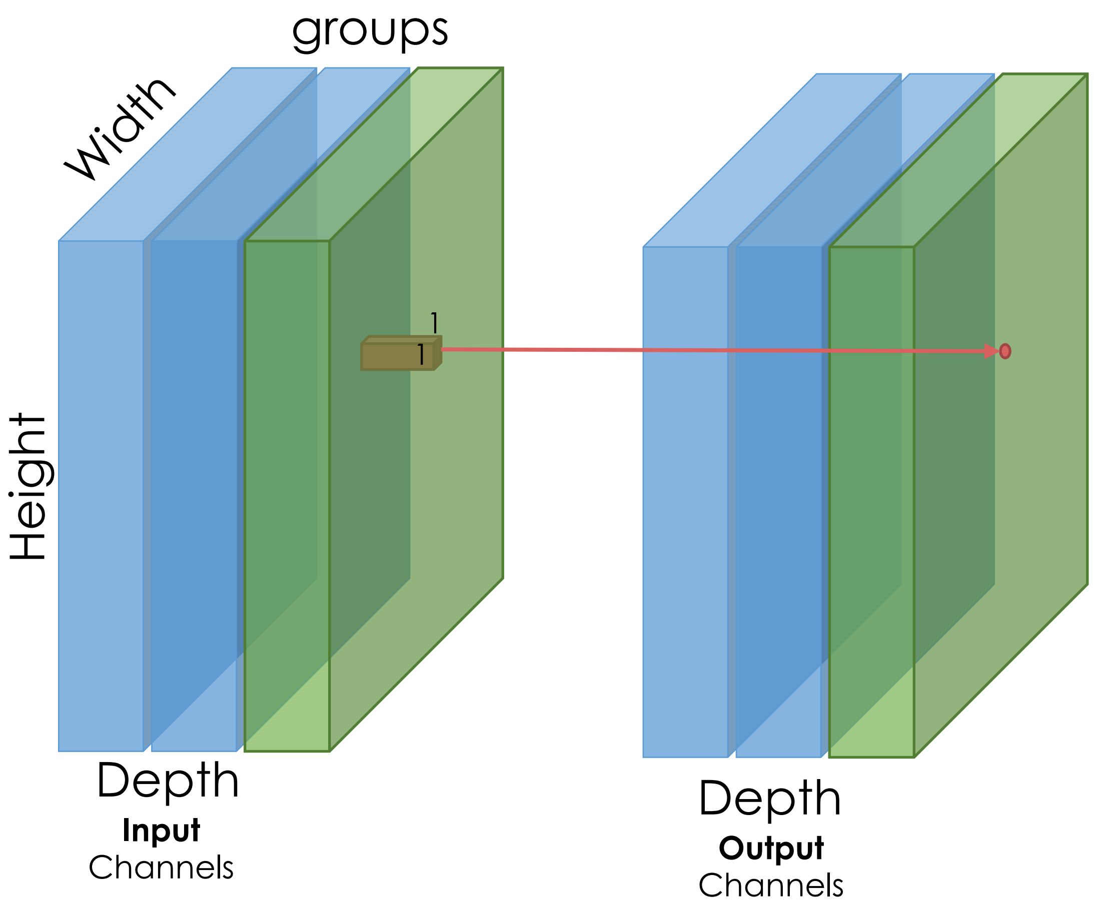
Computational Complexity:

$$(w * h * c_o)(c_i/g * 3 * 3)$$

Combines **some** information **across space** and **across channels.**

Can we apply to 1x1 (pointwise) convolution?

Pointwise (1x1) Group Convolution

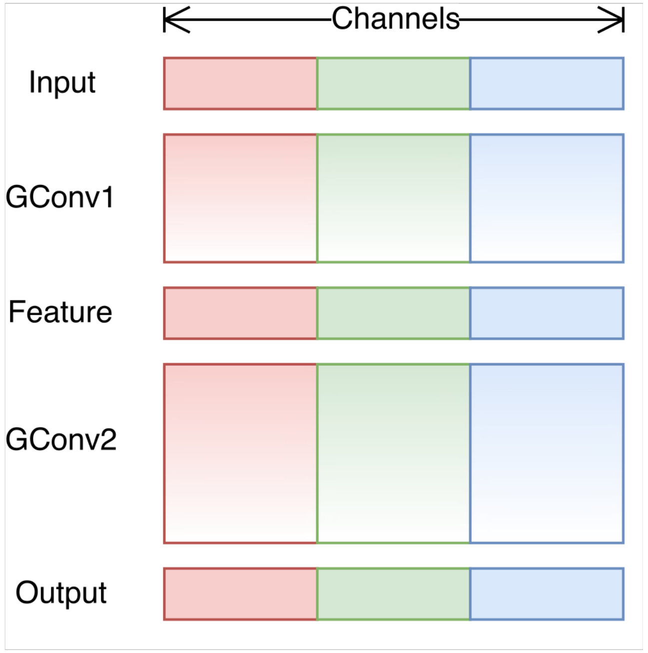


Computational Complexity:

$$(w * h * c_o) * c_i / g$$

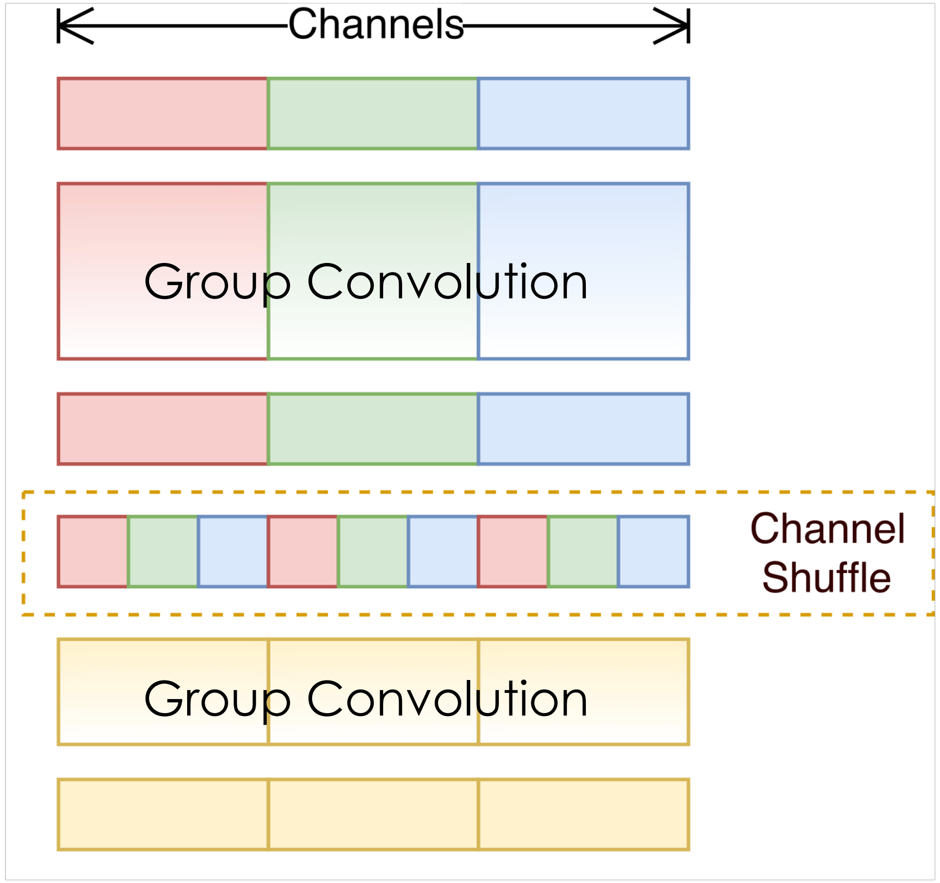
Combines **some** information **across channels.**

Issue: If applied repeatedly channel remain independent.



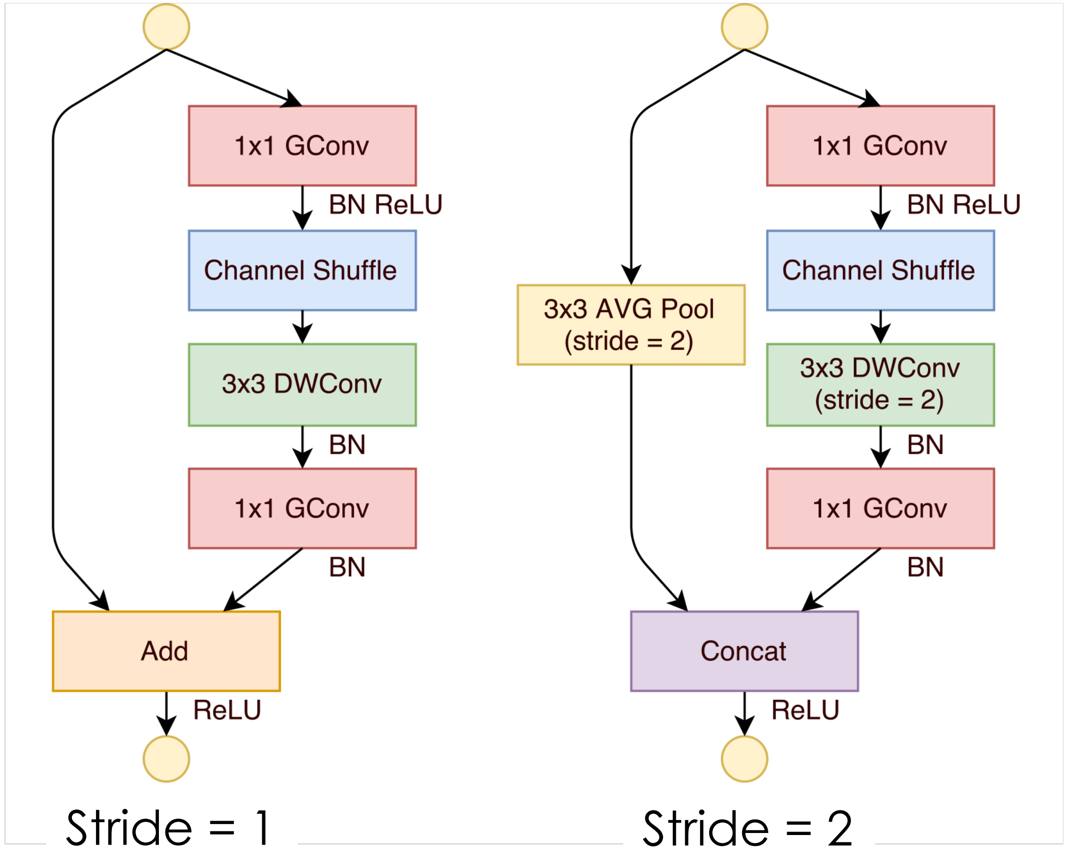
Channel Shuffle

- Permute channels between group convolution stages
  - Each group should get a channel from each of the previous groups.
- No arithmetic operations but does require data movement
  - Good or bad for hardware?



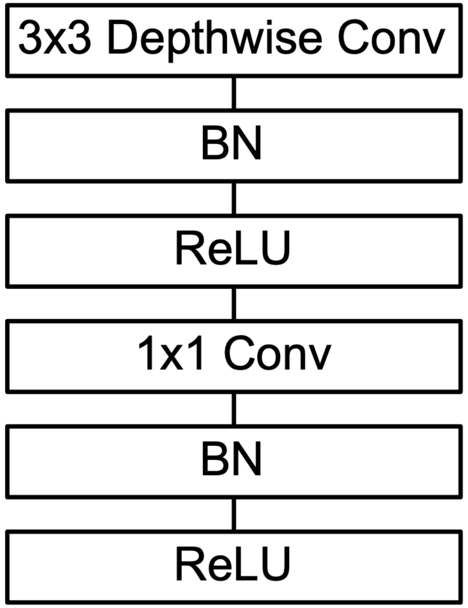
ShuffleNet Architecture

ShuffleNet Unit

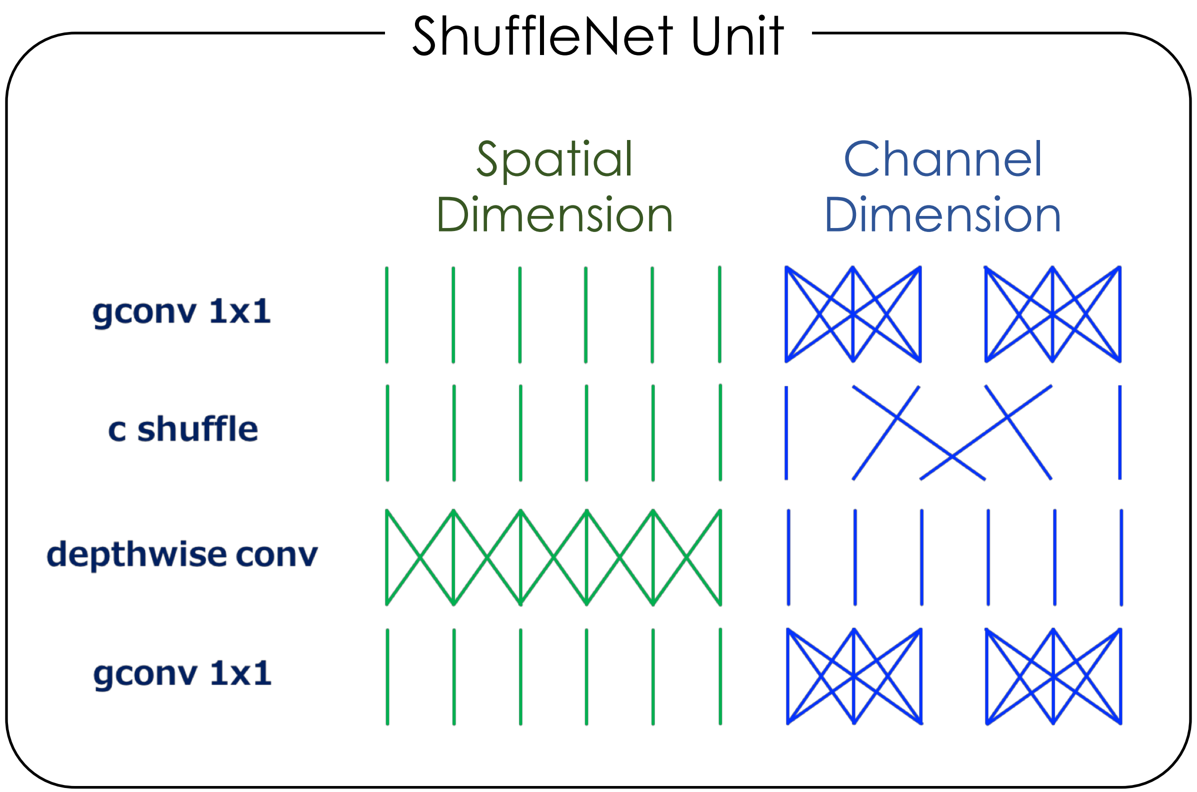


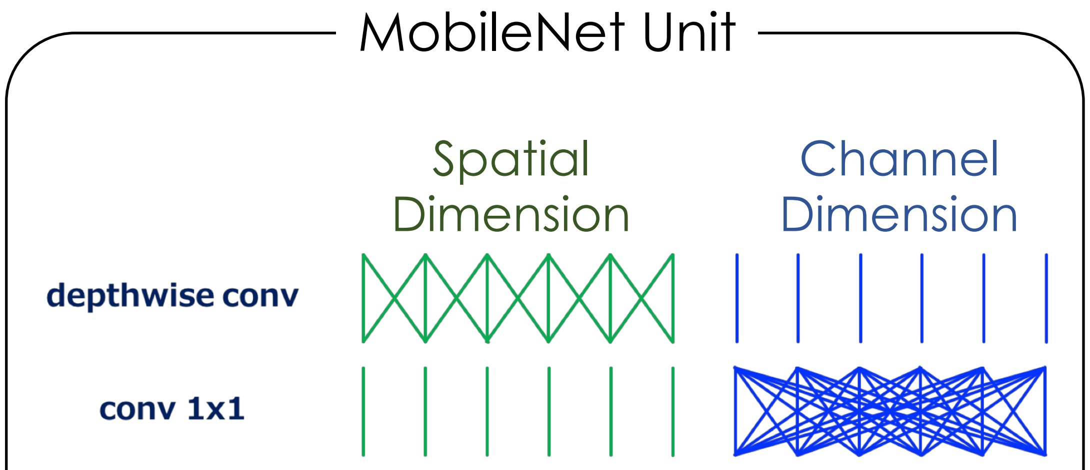
Based on Residual Networks

MobileNet Unit



Alternative Visualization





Notice the cost of the 1x1 convolution

Images from https://medium.com/@yu4u/why-mobilenet-and-its-variants-e-g-shufflenet-are-fast-1c7048b9618d

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

Increased width when increasing number of groups → Constant FLOPs

Observation:

Shallow networks need more channels (width) to maintain accuracy.

What are the Metrics of Success?

- Reduction in network size (parameters)
- Reduction in computation (FLOPS)
- Accuracy
- Runtime (Latency)

Comparisons to Other Architectures

- Less computation and more accurate than MobileNet

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

- Can be configured to **match accuracy** of other models while using **less compute**.

Model	Cls err. (%)	Complexity (MFLOPs)
VGG-16 [30]	28.5	15300
ShuffleNet $2\times$ ($g = 3$)	26.3	**524**
GoogleNet [33]*	31.3	1500
ShuffleNet $1\times$ ($g = 8$)	32.4	**140**
AlexNet [21]	42.8	720
SqueezeNet [14]	42.5	833
ShuffleNet $0.5\times$ ($g = 4$)	41.6	**38**

Runtime Performance on a Mobile Processor
(Qualcomm Snapdragon 820)

Model	Cls err. (%)	FLOPs	224×224	480×640	720×1280
ShuffleNet $0.5\times$ ($g = 3$)	43.2	38M	15.2ms	87.4ms	260.1ms
ShuffleNet $1\times$ ($g = 3$)	32.6	140M	37.8ms	222.2ms	684.5ms
ShuffleNet $2\times$ ($g = 3$)	26.3	524M	108.8ms	617.0ms	1857.6ms
AlexNet [21]	42.8	720M	184.0ms	1156.7ms	3633.9ms
1.0 MobileNet-224 [12]	29.4	569M	110.0ms	612.0ms	1879.2ms

- **Faster** and **more accurate** than MobileNet
- Caveats
  - Evaluated using single thread
  - Unclear how this would perform on a GPU … no numbers reported

Model Size

- They don't report memory footprint of model
  - Onnx implementation is 5.6MB → ~1.4M parameters
- MobileNet reports model size
  - 4.2M Parameters → ~16MB
- Generally relatively small

Limitations and Future Impact

- Limitations
  - Decreases arithmetic intensity
  - They disable Pointwise Group Convolution on smaller input layers due to "performance issues"
- Future Impact
  - Not yet a widely used as MobileNet
- Discussion:
  - Could potentially benefit from hardware optimization?