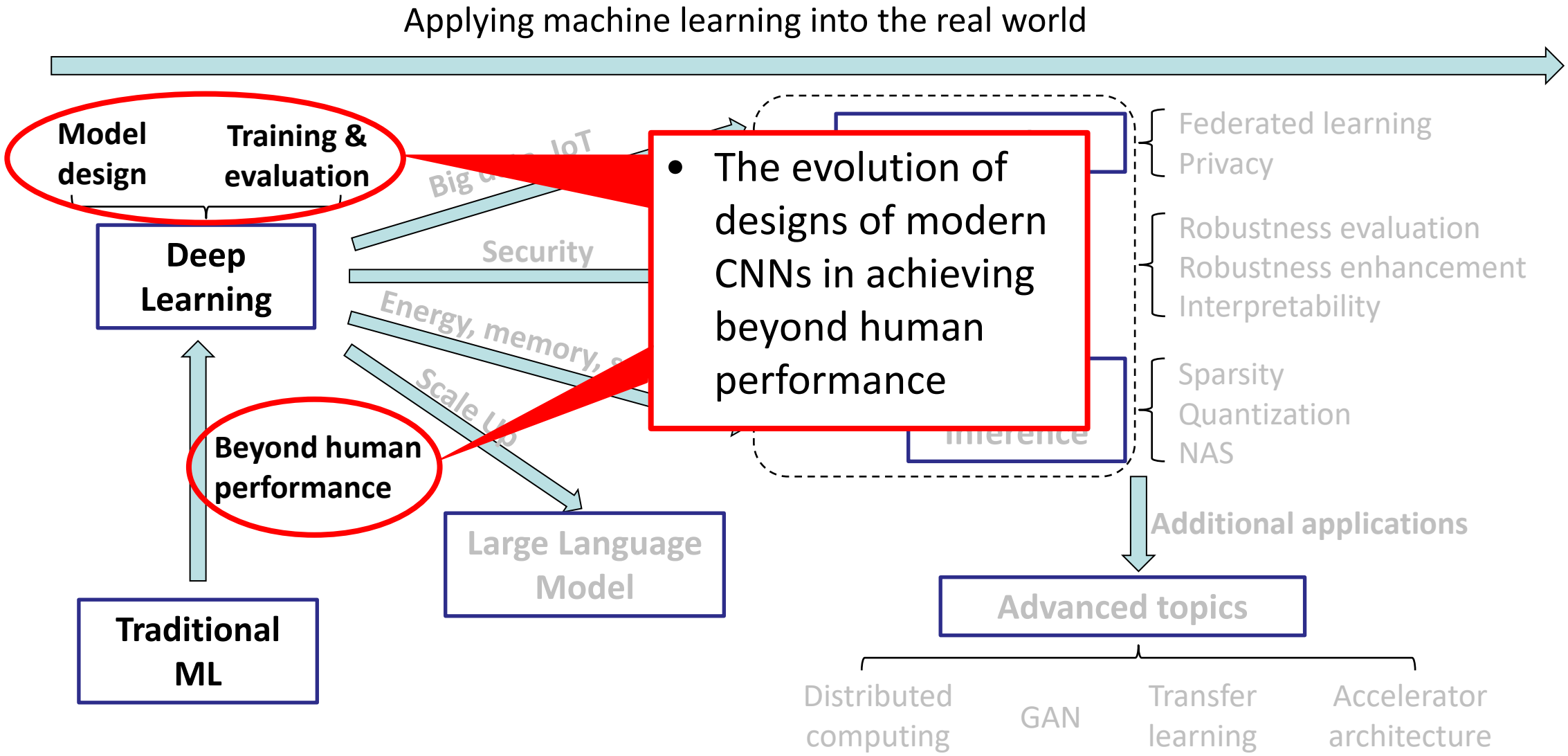
ECE 661 COMP ENG ML & DEEP NEURAL NETS

8. COMPACT NEURAL ARCHITECTURE DESIGN

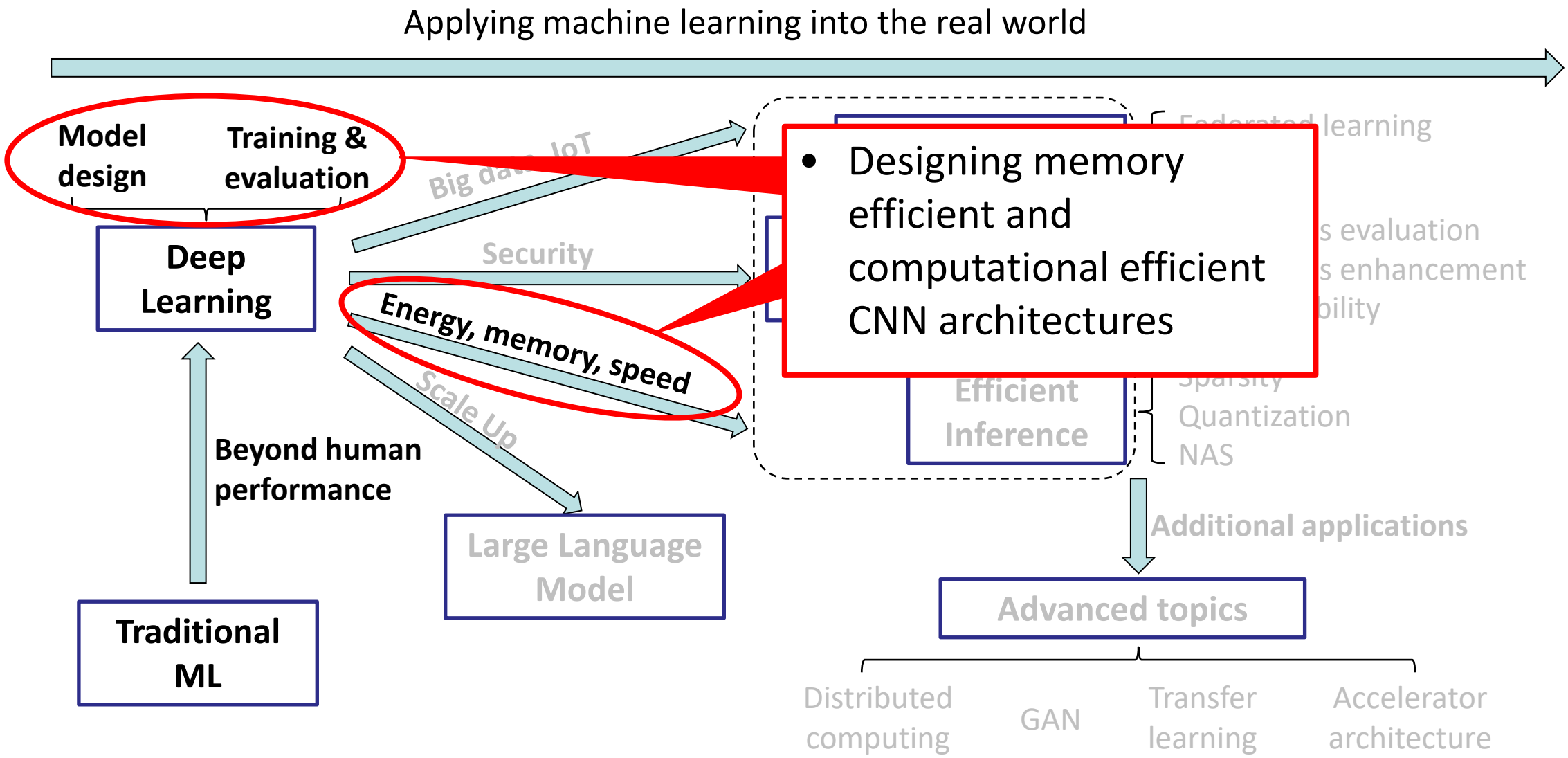
HAI "HELEN" LI, SPRING 2025

The previous lecture



- The evolution of designs of modern CNNs in achieving beyond human performance

This lecture



Resource consumption of ConvNets

How much resource does a ConvNet usually cost?

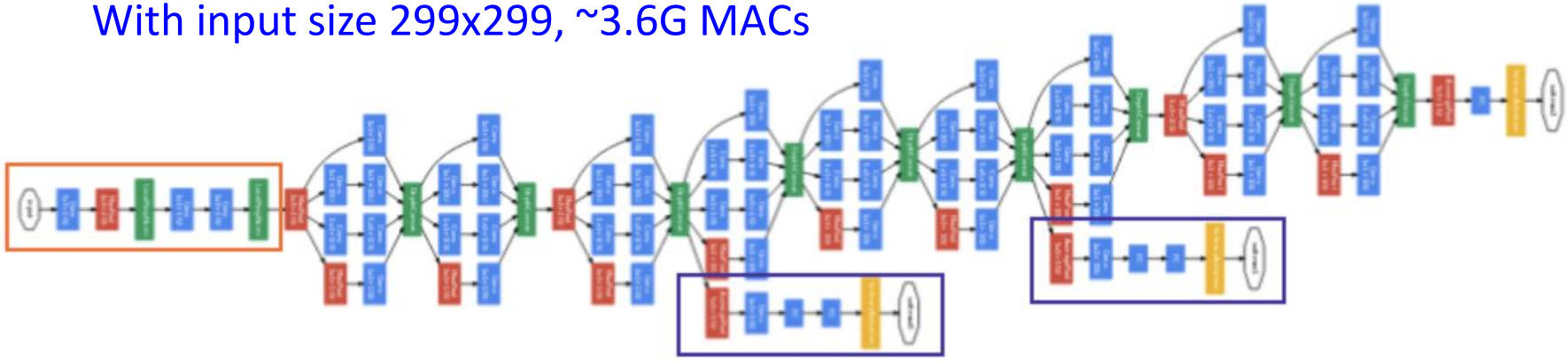
Architecture	Input Size	Parameter Memory	MACs
AlexNet	224x224	233 MB	727 M
VGG-16/19	224x224	528 MB / 548 MB	16 G /20 G
GoogLeNet	224x224	51 MB	2 G
ResNet-18/34	224x224	45 MB / 83 MB	2 G/ 4 G
DenseNet-121	224x224	31 MB	3 G
SqueezeNet	224x224	**4.8 MB**	727 M
MobileNet	224x224	**16.8 MB**	575 M
ShuffleNet	224x224	**13.6 MB**	292 M
MobileNet-V2	224x224	**13.6 MB**	300 M

Mobile convnets are usually resource-friendly!

Recap: GoogLeNet (Inception)

- GoogLeNet targets on **high-end** architecture design with **high accuracy**. This is quite popular in early CNN designs.
- However, this approach does not consider **model efficiency**. Thus, it is difficult to deploy GoogLeNet on mobile devices with limited computational power.

With input size 224x224, ~2G MACs
With input size 299x299, ~3.6G MACs



We need compact CNN models for efficient training and inference.

Szegedy, Christian, et al. "Going deeper with convolutions." (2015)

Recap: GoogLeNet (Inception)

The cost for training a GoogLeNet

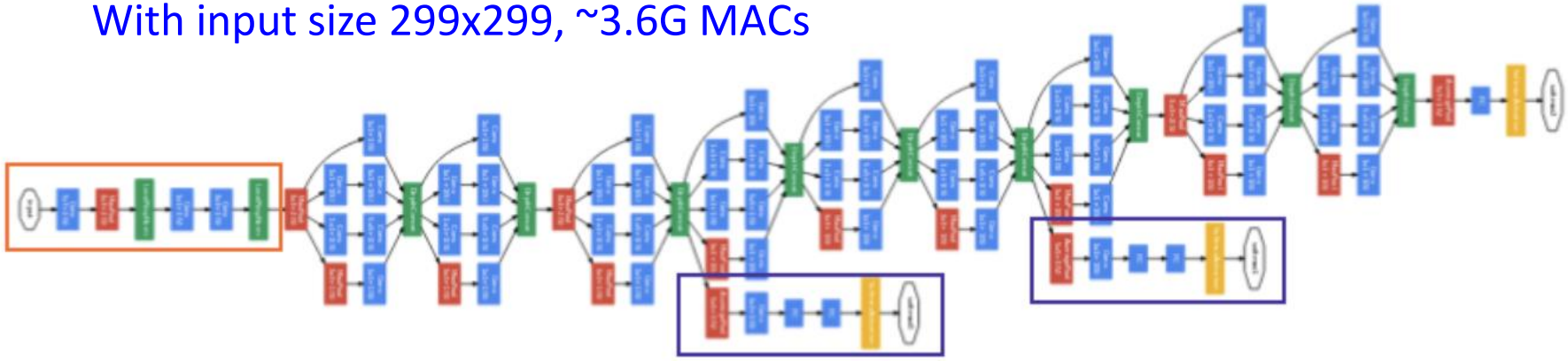
Training on ImageNet for 1 epoch takes **~35** K40 GPU hours.

Figure: Training GoogLeNet on a Tesla K40

```
2016-03-07 12:26:28.905231: step 20, loss = 14.81 (9.5 examples/sec; 3.380 sec/batch)
2016-03-07 12:27:02.699719: step 30, loss = 14.45 (9.5 examples/sec; 3.378 sec/batch)
2016-03-07 12:27:36.515699: step 40, loss = 13.98 (9.5 examples/sec; 3.376 sec/batch)
2016-03-07 12:28:10.220956: step 50, loss = 13.92 (9.6 examples/sec; 3.327 sec/batch)
```

With input size 224x224, ~2G MACs

With input size 299x299, ~3.6G MACs



We need compact CNN models for efficient training and inference.

Szegedy, Christian, et al. "Going deeper with convolutions." (2015)

Compact CNN Architectures

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

MobileNet: Efficient CNN for mobile vision applications

ShuffleNet: An extremely efficient CNN for mobile devices

MobileNetV2: Inverted residuals and linear bottlenecks

SqueezeNet

SqueezeNet is designed to **reduce memory consumption** for large CNN models (e.g., AlexNet).

Strategies:

- Replace some of 3x3 filters with 1x1 filters. Decrease the number of input channels that are input to 3x3 filters using **squeeze layers**.
- **Down-sample late** in the network to spend more computation budgets (MACs) on larger activation maps. This indicates that SqueezeNet stacks more convolutional layers at the early stage of the CNN architecture.

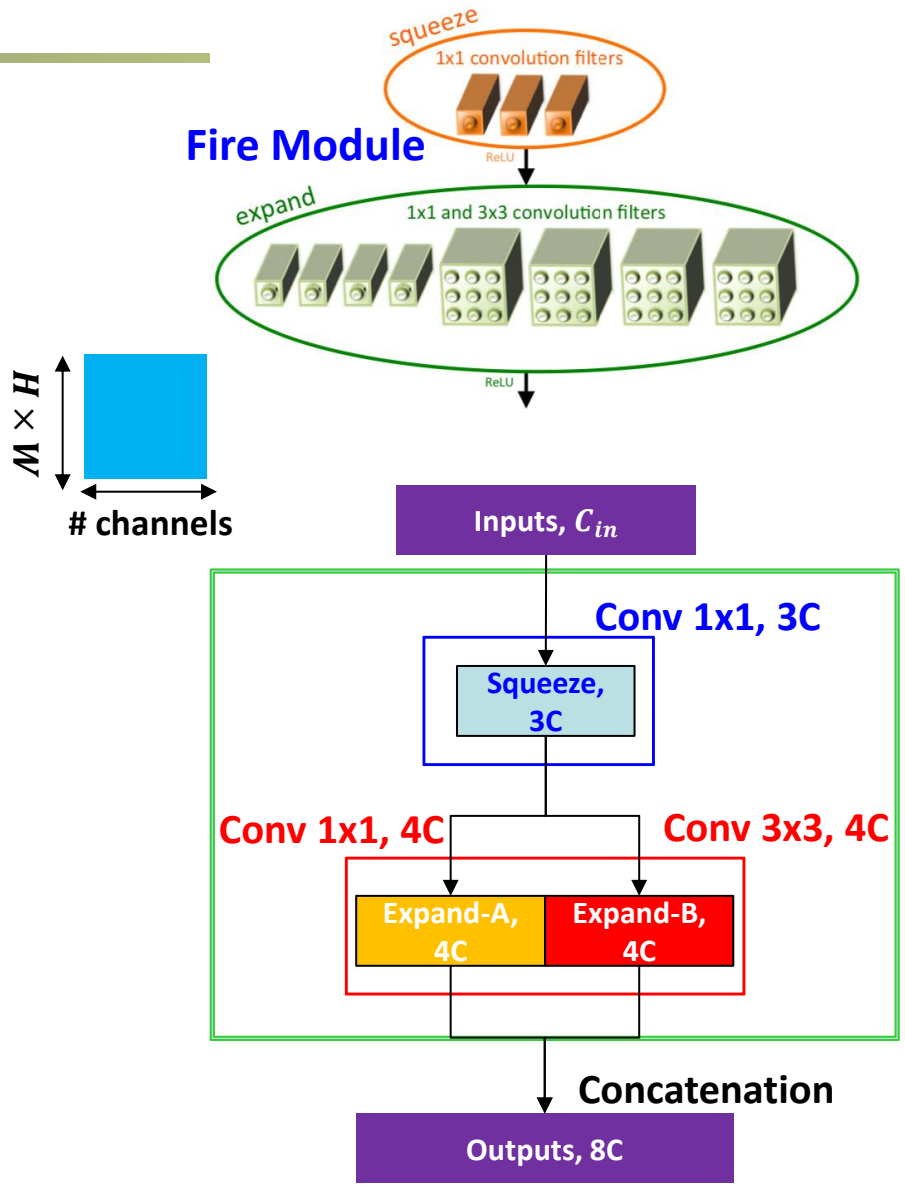
Iandola, Forrest N., et al. "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size." (2016)

SqueezeNet: “Fire” module

A “fire” module has **“squeeze layers”** and “**expand layers**”.

- **Squeeze layers**: make feature map ‘*narrower*’ (similar to bottleneck structure) → efficiency
- **Expand layers**: use a mixture of 1×1 and 3×3 filters to make the feature map ‘*wider*’ → performance

The combination of Squeeze and Expand layers leads to parameter and MAC saving.



Iandola, Forrest N., et al. "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size.” (2016)

SqueezeNet: late downsampling

What does SqueezeNet look like?

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									
	activations		parameters				compression info				1,248,424 (total)	**421,098** (total)

SqueezeNet uses **fire modules** and removes redundant FC layers.

Iandola, Forrest N., et al. "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size." (2016)

SqueezeNet: results

- As a result, SqueezeNet achieves similar accuracy compared to AlexNet, with 50× parameter saving.
- With model compression/quantization (we will cover it in later lectures), the size of SqueezeNet can be further reduced.

Table 2: Comparing SqueezeNet to model compression approaches. By *model size*, we mean the number of bytes required to store all of the parameters in the trained model.

CNN architecture	Compression Approach	Data Type	Original → 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 → 48MB	5x	56.0%	79.4%
AlexNet	Network Pruning (Han et al., 2015b)	32 bit	240MB → 27MB	9x	57.2%	80.3%
AlexNet	Deep Compression (Han et al., 2015a)	5-8 bit	240MB → 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 → 0.66MB	**363x**	57.5%	80.3%
SqueezeNet (ours)	Deep Compression	6 bit	4.8MB → 0.47MB	**510x**	57.5%	80.3%

Later works find the latency of SqueezeNet be sub-optimal, due to **parallel branches** in "fire module" → **call for a streamlined architecture!**

MobileNet

Motivation:

- Build **a streamlined, scalable** architecture for **lightweight** neural architectures.
- Reach **state-of-the-art performance and resource tradeoff** on image classification tasks.
- Achieve **adaptable structures** for **transfer learning** on a wide range of vision tasks (e.g., object detection).

Main approach:

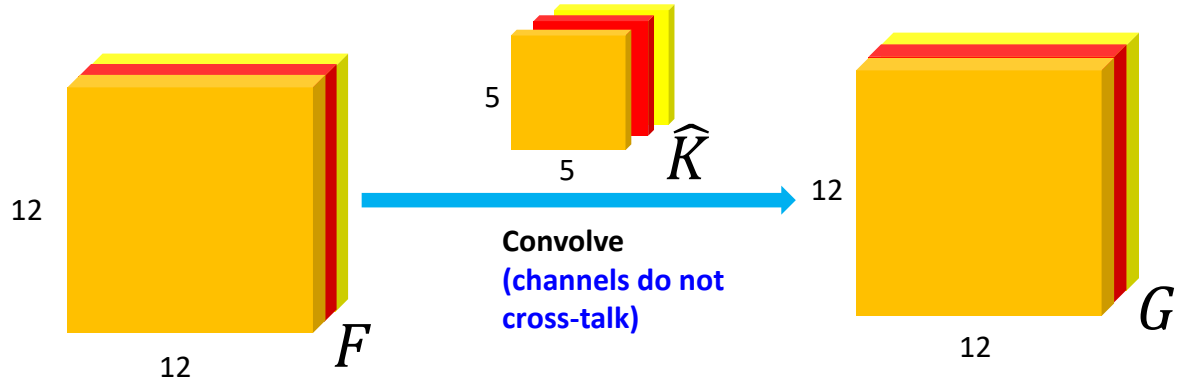
- Introduce **depthwise separable convolution** to reduce computational overhead within operations.
- Introduce two more hyperparameters (resolution & width multiplier) to further reduce the model size of MobileNet.

Howard, Andrew G., et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications." arXiv preprint arXiv:1704.04861 (2017).

MobileNet: depthwise convolution

Is there a faster way to do convolution?

- Use **depthwise convolution** to extract spatial information for each independent input channel. Channel-wise information interactions are not considered.
- Here we give an example of a 5×5 depthwise convolution on a 12×12 feature map with padding enabled.



$$G_{k,l,m} = \sum_{i,j} \widehat{K}_{i,j,m} F_{k+i-1,l+j-1,m}$$

$\widehat{K}$: Depthwise Convolution kernel; F: Feature Map

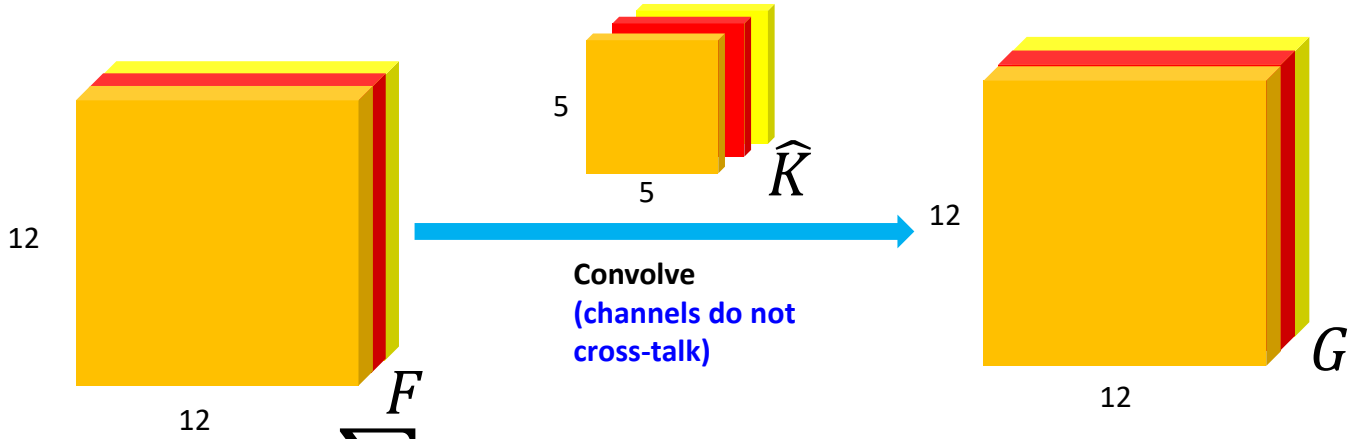
G: Output feature map; i, j, m: index

- We do not sum over channel dimension!
- Each channel has its own depthwise convolution outputs.
- Depthwise convolution does not change the number of channels.

MobileNet: depthwise convolution

Question: What is the parameter count and MACs for the following depthwise convolution?

Answer: Parameters: $3 \times 5 \times 5 = 75$
Multi-Adds: $3 \times 5 \times 5 \times 12 \times 12 = 10800$



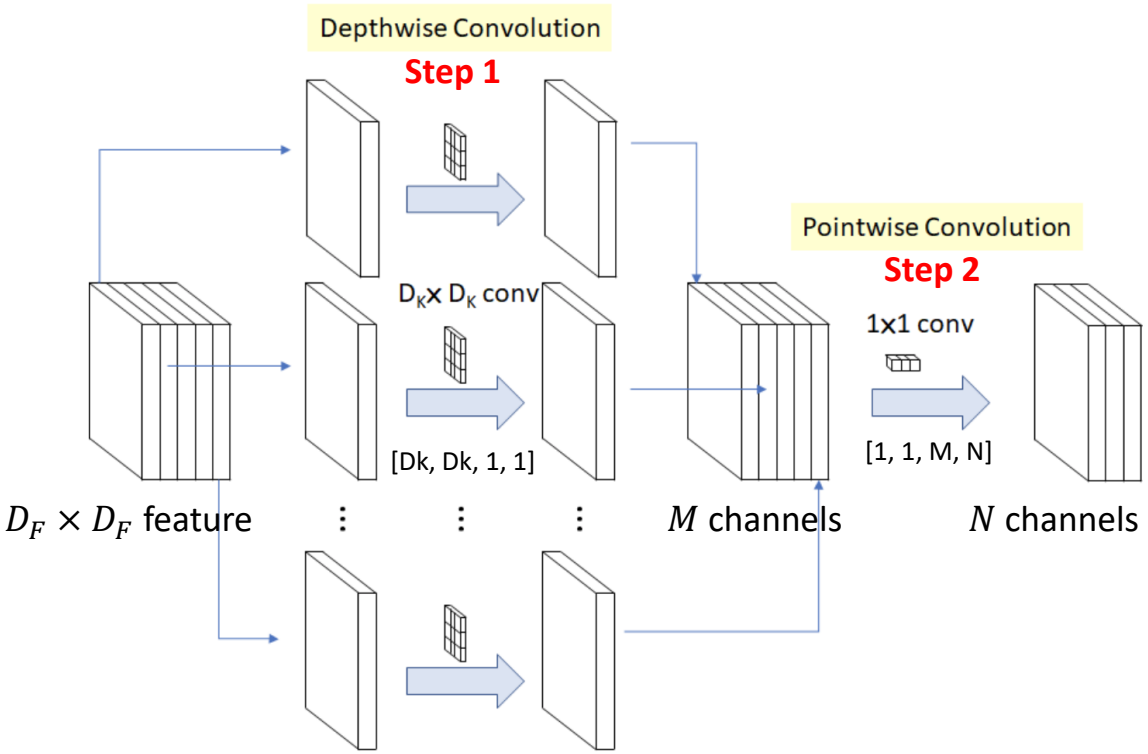
$$G_{k,l,m} = \sum_{i,j} \widehat{K}_{i,j,m} F_{k+i-1,l+j-1,m}$$

$\widehat{K}$: Depthwise Convolution kernel; F: Feature Map
G: Output feature map; i, j, m: index

- We do not sum over channel dimension!
- Each channel has its own depthwise convolution outputs.
- Depthwise convolution does not change the number of channels.

MobileNet: depthwise separable convolution

- A regular convolution extracts spatial features and channel-wise features within only 1 single convolution step.
- A $\mathrm{D}_k \times D_k$ **depthwise separable convolution** is composed of a $\mathrm{D}_k \times D_k$ **depthwise convolution** and a 1×1 **pointwise convolution**.

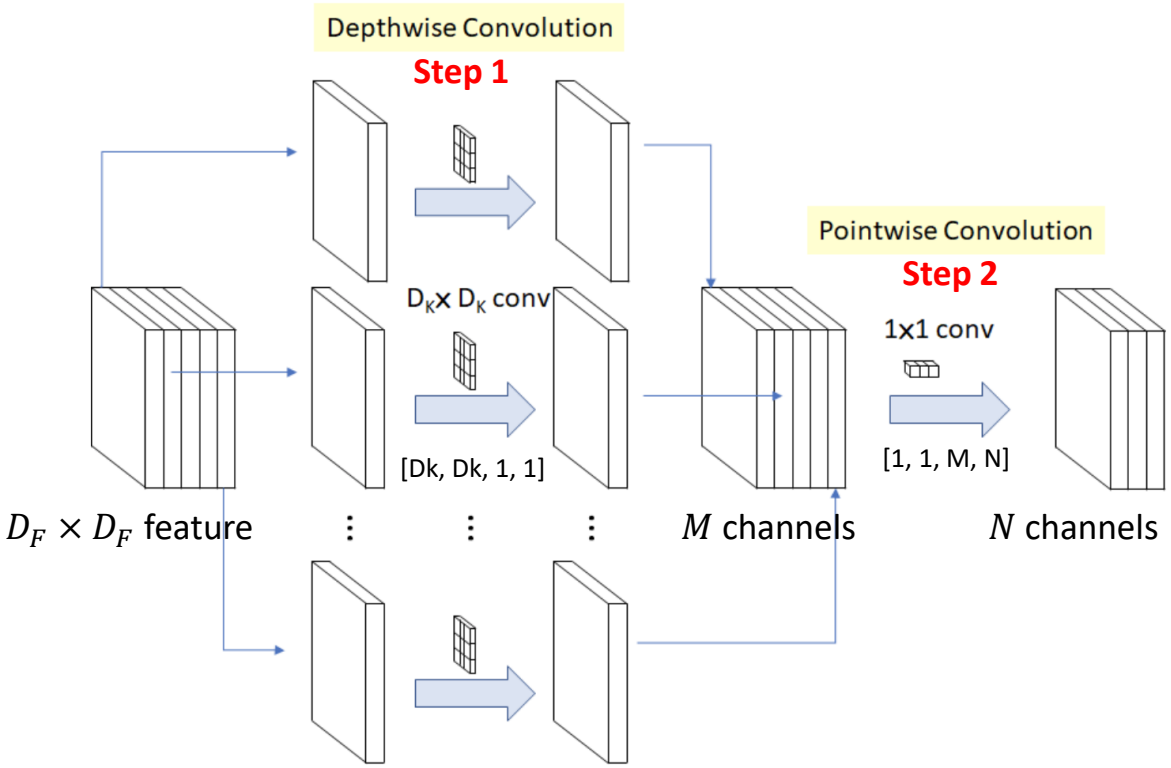


Step 1: Depthwise convolution *only* extracts **spatial features** for each independent channel.

Step 2: Pointwise convolution extracts **channel-wise information** by mapping features to a different projection.

MobileNet: depthwise separable convolution

- **What is the number of parameters and multiply-accumulates (MACs) for each step in a depthwise-separable convolution?**
- Assume this depthwise separable convolution has a stride of 1, and padding is used to keep the size of input and output feature map to be consistent



Step 1: Depthwise convolution

$D_K \times D_K \times M$ parameters

$D_K \times D_K \times M \times D_F \times D_F$ MACs

Step 2: Pointwise convolution.

$M \times N$ parameters

$D_F \times D_F \times M \times N$ MACs

MobileNet: depthwise separable convolution

Compare 3×3 regular convolution vs. 3×3 depthwise-separable convolution. What can we expect for the theoretical speedup?

Recall that a 3×3 regular convolution has $3 \times 3 \times M \times N \times D_F \times D_F$ MACs and $3 \times 3 \times M \times N$ parameters.

A depthwise separable convolution has $3 \times 3 \times M \times D_F \times D_F + D_F \times D_F \times M \times N$ MACs and $3 \times 3 \times M + M \times N$ parameters.

When N, M is large, the theoretical speedup is calculated as:

$$\frac{3 \times 3 \times M \times N \times D_F \times D_F}{3 \times 3 \times M \times D_F \times D_F + D_F \times D_F \times M \times N} \approx 9$$

MobileNet: depthwise separable convolution

How does depthwise separable convolutions work in MobileNetV1?

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

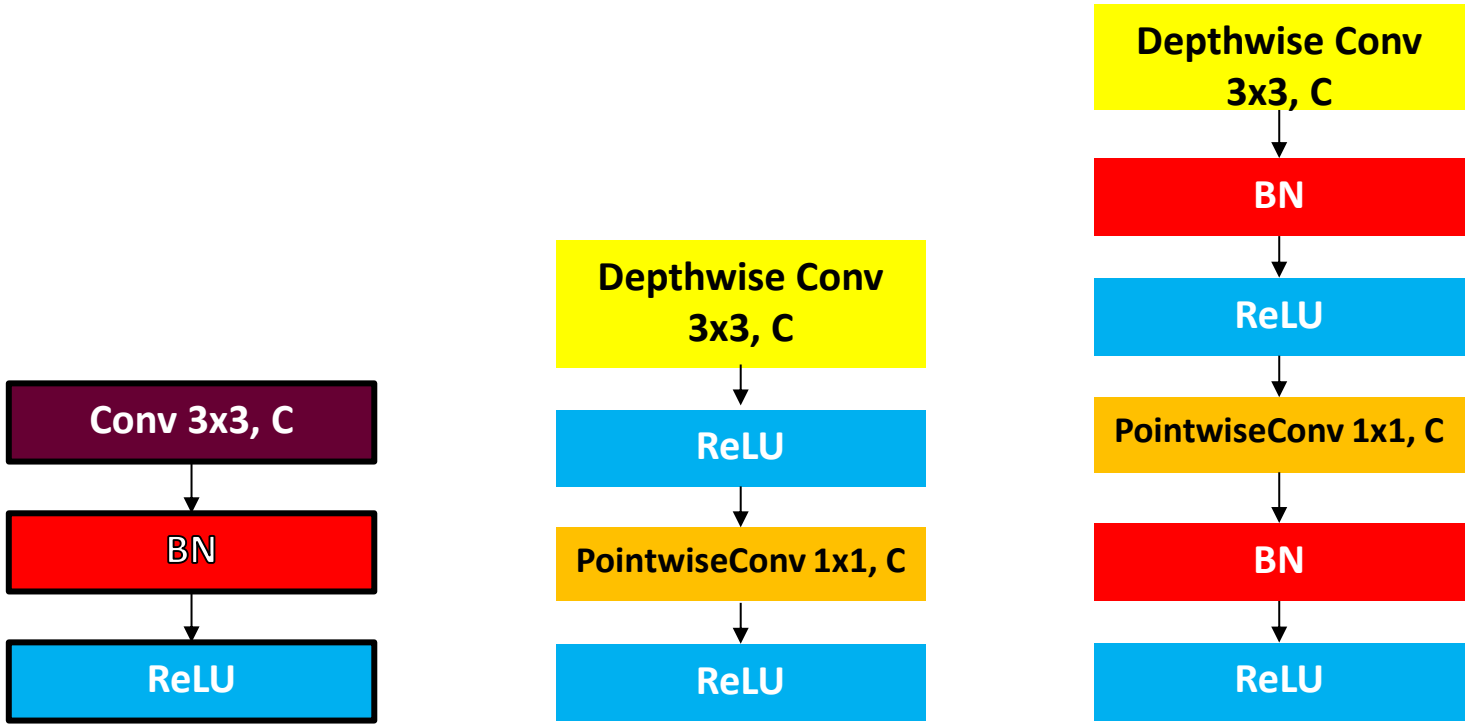
Conv MobileNet: the architecture obtained by replacing all depthwise separable convolution (3×3) in MobileNet with regular convolution (3×3).

- Depthwise separable convolution gives about **9x MAC reduction** while preserving similar accuracy, compared to Conv MobileNet.
- MobileNet architecture only has **1% top-1 accuracy drop** on ImageNet-1K classification task, compared to Conv MobileNet.

MobileNet: position of non-linearity and BN

How to insert nonlinearity and batch normalization in depthwise separable convolution?

Follow the same rule as ConvNet designs. BN and nonlinearity need to be inserted for both depthwise convolutions and pointwise convolutions for best performance.



Standard convolution with BN

Depthwise separable convolution without BN

Depthwise separable convolution with BN

MobileNet: architecture

MobileNet is **deeper** than previous architectures (e.g., VGG-16).

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	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
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	$1 \times 1 \times 1000$

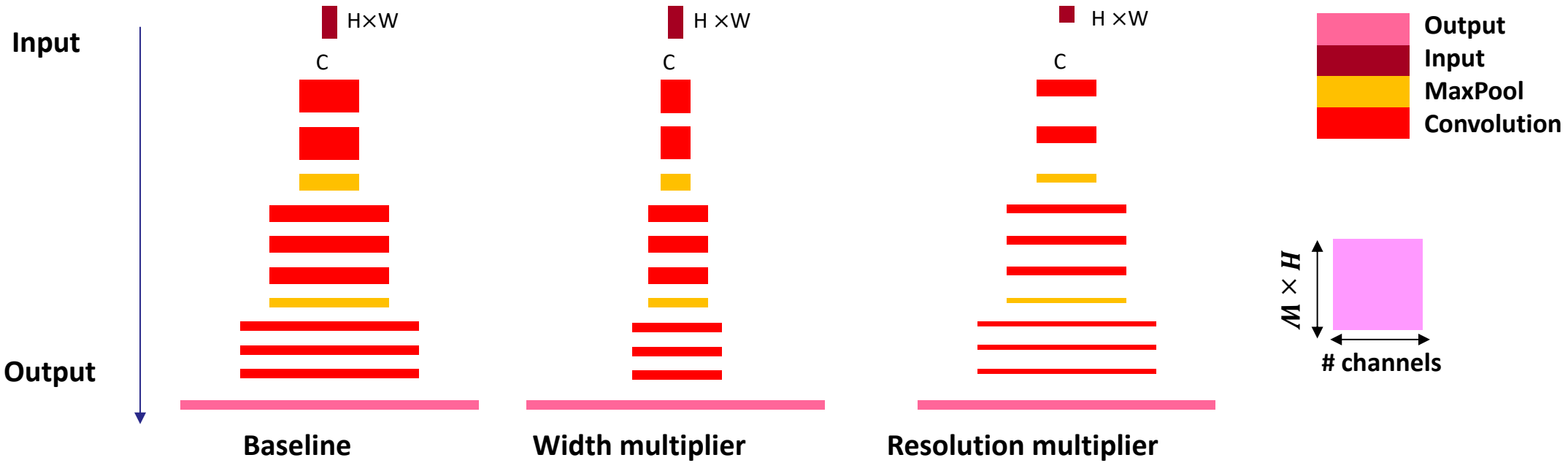
dw: Depthwise convolution
Conv: Regular convolution
s: stride patterns.
s2 means stride=2.

Computation breakdown:
Conv 1x1 consumes 94.86% of total MAC and 74.59% of total parameters.
Thus, Conv 1x1 operation has become the new computation bottleneck.

MobileNet: construct scalable neural architectures

MobileNet can scale down the architecture by introducing two hyperparameters in each layer:

- **Width multiplier:** Scale down the number of **channels**.
- **Resolution multiplier:** Scale down the **input resolution**. The size of feature maps for consequent layers will also be reduced.



MobileNet: construct scalable neural architectures

Results: accuracy-resource trade-off for scalable MobileNets.

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

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

Width multiplier simultaneously reduces **MACs** and **parameter count**.

Resolution multiplier only reduces **MACs**. **(Why?)**

ShuffleNet

Research for designing mobile architectures becomes popular after MobileNet. ShuffleNet is one of the attractive design ideas.

Motivation:

- ShuffleNet is designed for mobile devices with much lower computational power (100-150M MACs).
- ShuffleNet addresses the problem of costly dense 1×1 convolution to improve model efficiency.

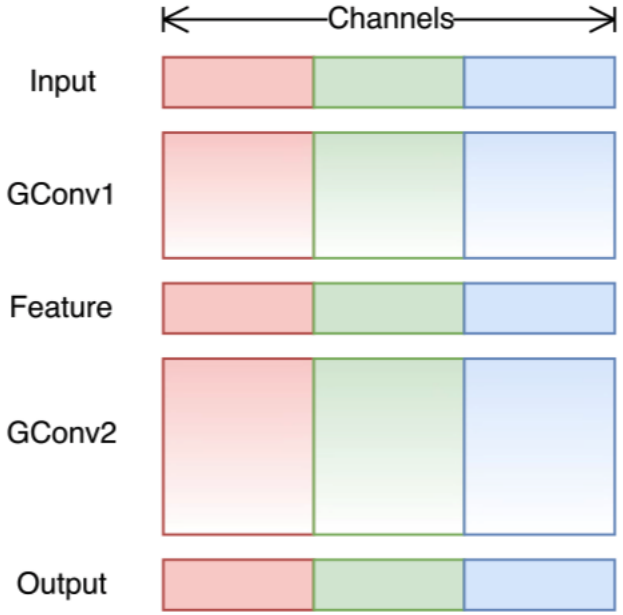
Features:

- Use 1×1 **group convolution** to reduce the computation overhead from 1×1 convolution.
- Introduce **channel shuffle** to enable cross-talks between different channels in group convolution. Channel shuffle cooperates well with group convolution.

Zhang, Xiangyu, et al. "Shufflenet: An extremely efficient convolutional neural network for mobile devices."(2018)

ShuffleNet: group convolution

- **Group Convolution (GConv)** splits the input into different groups, processes each group with regular convolution, and finally concatenates the outputs.
- Depthwise convolution is a special case of group convolution as every input channel is split as a separated group.



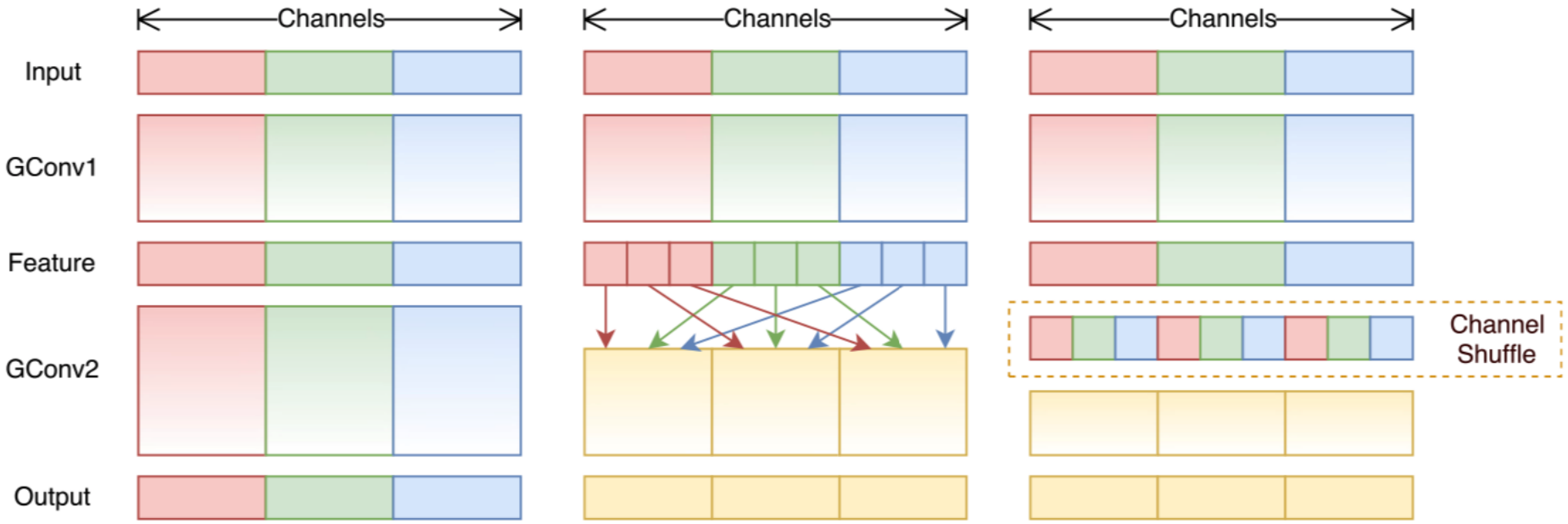
- Input channels with the same color are 'grouped' together to produce the outputs.
- Inputs from different groups can never **cross-talk**. This saves computation compared to the regular convolution. However, the lack of cross-talking also raises problems.

3 groups saves 3× MACs

Zhang, Xiangyu, et al. "Shufflenet: An extremely efficient convolutional neural network for mobile devices."(2018)

ShuffleNet: channel shuffle

- ShuffleNet enhances the ability of group convolution by incorporating channel shuffle into group convolution.
- **Channel shuffle** grabs the inputs from different groups, thus channels from different groups can cross-talk.



Pure random channel shuffle. Not recommended due to intractable inference process.

Channel shuffle of ShuffleNet. Consistent shuffle for both training and inference.

Zhang, Xiangyu, et al. "Shufflenet: An extremely efficient convolutional neural network for mobile devices."(2018)

ShuffleNet: channel shuffle

Channel shuffle leads to significant performance gain in various ShuffleNet models.

Table: Performance of ShuffleNet with and without channel shuffling

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

G=3: use 3 groups for group convolution.
0.5x: apply a width multiplier of 0.5.

Table: Performance of ShuffleNet vs. various structures

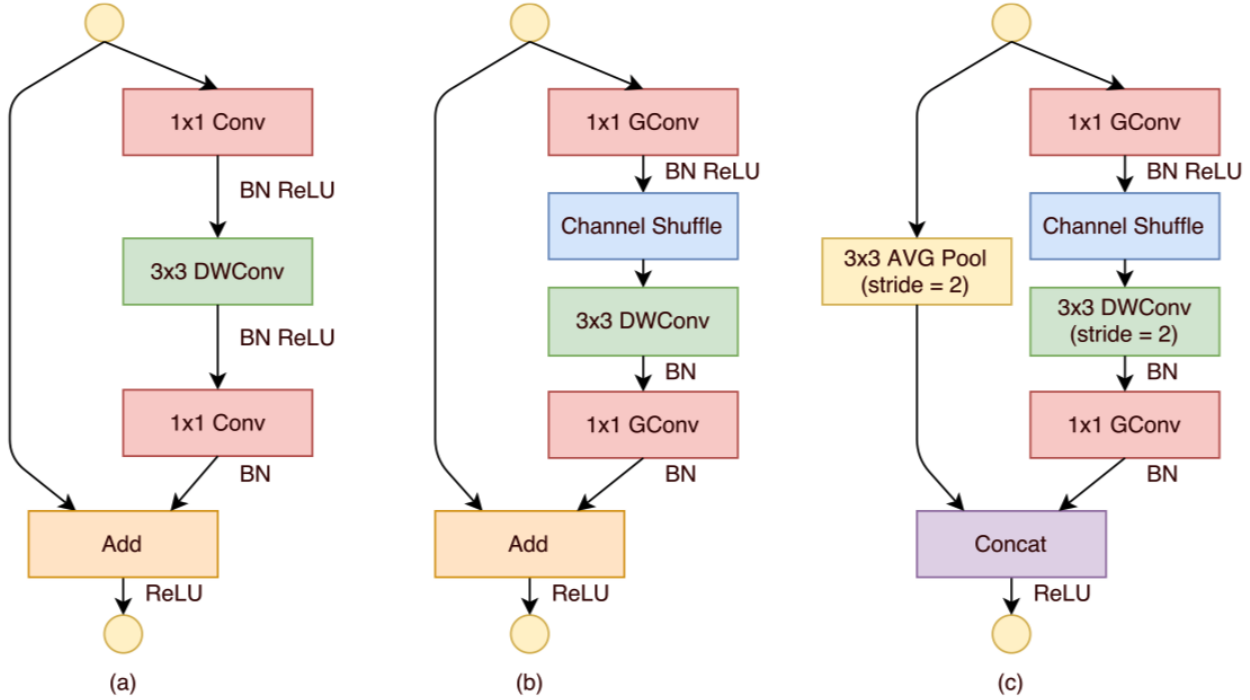
Complexity (MFLOPs)	VGG-like	ResNet	Xception-like	ResNeXt	ShuffleNet (ours)
140	50.7	37.3	33.6	33.3	**32.4** ($1\times$, $g = 8$)
38	-	48.8	45.1	46.0	**41.6** ($0.5\times$, $g = 4$)
13	-	63.7	57.1	65.2	**52.7** ($0.25\times$, $g = 8$)

Note: smaller number represents better performance.

Zhang, Xiangyu, et al. "Shufflenet: An extremely efficient convolutional neural network for mobile devices."(2018)

ShuffleNet unit

- ShuffleNet unit is the **building block** for ShuffleNet model.
- Note that operation following the 3x3 depthwise convolution has been changed to 1x1 group convolution. This is also called **group pointwise convolution** in ShuffleNet.



Bottleneck unit w/ depthwise convolution

ShuffleNet unit w/ strides=1

ShuffleNet unit w/ strides=2

Number of channels in feature map directly doubles after going through a stride=2 block.
(Why is it good?)

Zhang, Xiangyu, et al. "Shufflenet: An extremely efficient convolutional neural network for mobile devices."(2018)

ShuffleNet: architecture

- ShuffleNet is mainly composed of ShuffleNet units.
- The MACs of ShuffleNet ranges from 133M to 143M. This is much smaller than the state-of-the-art mobile CNN architectures (e.g., MobileNet) at that time.

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

Zhang, Xiangyu, et al. "Shufflenet: An extremely efficient convolutional neural network for mobile devices."(2018)

ShuffleNet: results

Results on ImageNet-1K dataset

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

- Compared to MobileNet, ShuffleNet reduces resource and computation overhead via the usage of pointwise group convolution.
- Channel shuffling mechanism further improves the performance.

Zhang, Xiangyu, et al. "Shufflenet: An extremely efficient convolutional neural network for mobile devices."(2018)

ShuffleNet: Latency

- ShuffleNet is 12.10x faster than AlexNet with a similar accuracy!

Table: Inference latency on a Qualcomm Snapdragon 820 processor

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

- Even on larger object detection tasks (e.g., MS-COCO), ShuffleNet can achieve a higher mean average precision (mAP) with a lower latency.

Model	mAP [.5, .95] (300× image)	mAP [.5, .95] (600× image)
ShuffleNet 2× ($g = 3$)	**18.7%**	**25.0%**
ShuffleNet 1× ($g = 3$)	14.5%	19.8%
1.0 MobileNet-224 [12]	16.4%	19.8%
1.0 MobileNet-224 (our impl.)	14.9%	19.3%

Zhang, Xiangyu, et al. "Shufflenet: An extremely efficient convolutional neural network for mobile devices."(2018)

MobileNetV2

MobileNetV2 pushes the boundary of state-of-the-art tailored computer vision models across a wide spectrum of model sizes.

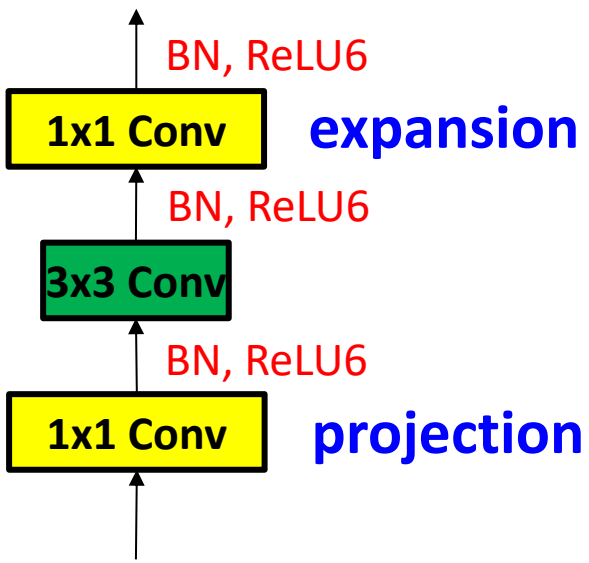
Features:

- MobileNetV2 uses **Inverted Residual Bottleneck (IRB) structure** to increase representational power.
- Like MobileNet, width multiplier and resolution multiplier are still prevalent in constructing scalable MobileNetV2 models.
- The design of **IRB** structure in MobileNetV2 is used as a good starting point for conducting neural architecture engineering.
  - Most neural architecture search works are built upon MobileNetV2 inverted bottleneck structures.
  - We will cover them in the later lectures.

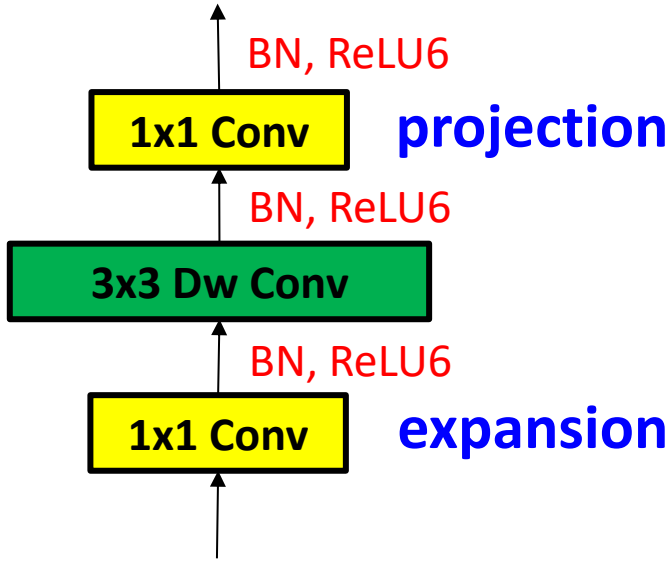
Sandler, Mark, et al. "Mobilenetv2: Inverted residuals and linear bottlenecks." (2018)

MobileNetV2: Inverted Bottleneck

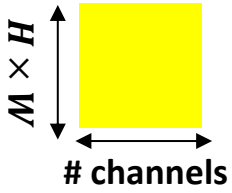
- Traditional bottleneck structure (e.g., ResNet-50) has a narrower feature map in-between and wider feature maps on the ends. This saves the computation of a **costly** 3x3 convolution.
- Inverted bottleneck (e.g., MobileNetV2) has wider feature maps in-between and narrower feature maps on the ends. This enhances the power of **lightweight** depthwise convolution.



Narrower feature map in-between, and **wider** feature map on the ends.

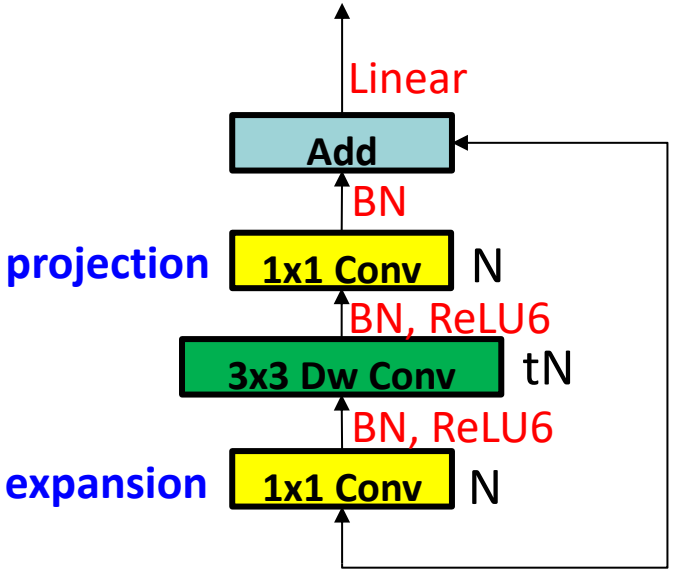


Wider feature map in-between and **narrower** feature map on the ends.

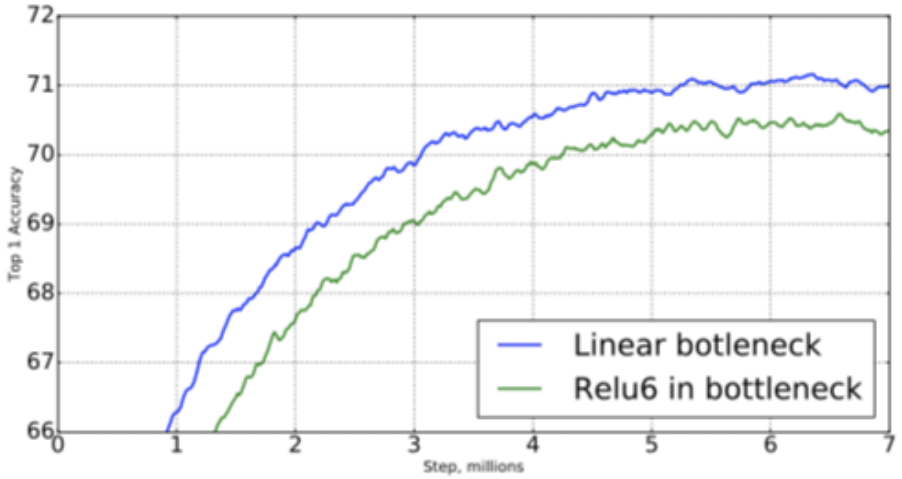


Inverted Residual Bottleneck (IRB)

- MobileNetV2 proposes to remove the final activation function in each inverted bottleneck. This can increase the representational power of MobileNetV2 models.
- A combination of an Inverted Residual with a linear bottleneck makes an Inverted Residual Bottleneck (IRB) block. The expansion factor (t) is usually set to 6 to coordinate expansion and projection layers.



IRB block



Linear bottleneck vs. ReLU bottleneck

MobileNetV2: architecture

- MobileNetV2 architecture is composed of multiple **IRB** blocks.
- Compared to MobileNet, each **IRB** block produces an output feature map with fewer channel numbers.

Table: MobileNetV2

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	-	

Table: MobileNet

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	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
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	$1 \times 1 \times 1000$

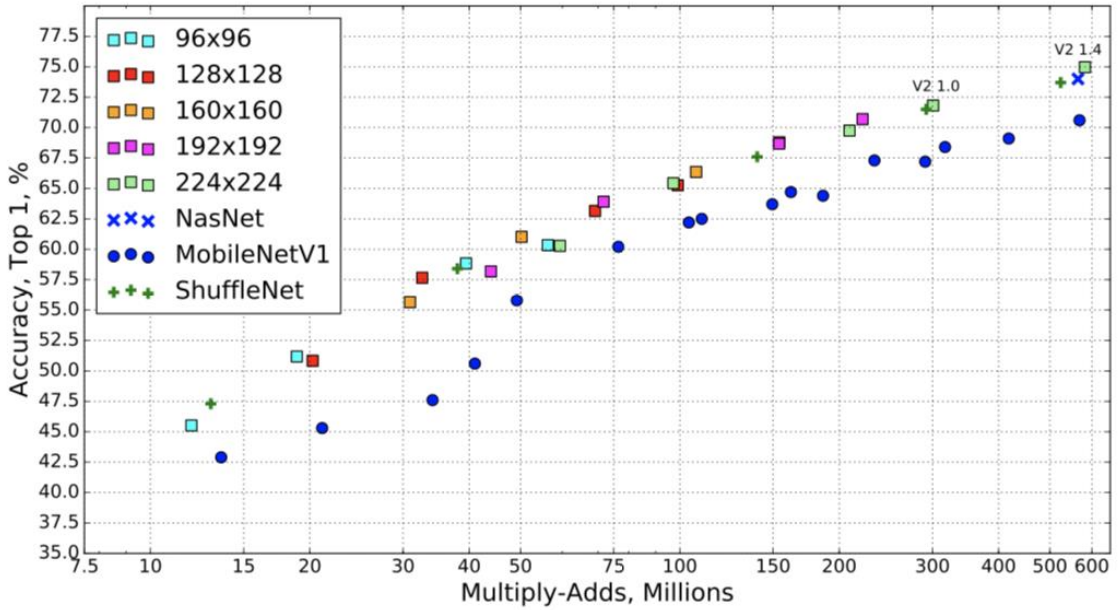
MobileNetV2: results

- MobileNetV2 has a significant improvement over MobileNetV1, thanks to the efficiency of the **IRB structure**.
- It also reaches a better accuracy-MAC tradeoff compared to other architectures.
- Till now, MobileNetV2 is still a very competitive and useful neural architecture for efficient deployment.

Table: Top-1 accuracy on ImageNet

Network	Top 1	Params	MAdds	CPU
MobileNetV1	70.6	4.2M	575M	113ms
ShuffleNet (1.5)	71.5	**3.4M**	292M	-
ShuffleNet (x2)	73.7	5.4M	524M	-
NasNet-A	74.0	5.3M	564M	183ms
MobileNetV2	**72.0**	**3.4M**	**300M**	**75ms**
MobileNetV2 (1.4)	**74.7**	6.9M	585M	**143ms**

Table: Accuracy vs. MAC tradeoff of various architectures



Resource consumption of ConvNets

How much resource does a ConvNet usually cost?

Architecture	Input Size	Parameter Memory	MACs
AlexNet	224x224	233 MB	727 M
VGG-16/19	224x224	528 MB / 548 MB	16 G /20 G
GoogLeNet	224x224	51 MB	2 G
ResNet-18/34	224x224	45 MB / 83 MB	2 G/ 4 G
DenseNet-121	224x224	31 MB	3 G
SqueezeNet	224x224	**4.8 MB**	727 M
MobileNet	224x224	**16.8 MB**	575 M
ShuffleNet	224x224	**13.6 MB**	292 M
MobileNet-V2	224x224	**13.6 MB**	300 M

Mobile convnets are usually resource-friendly!

In this lecture, we learned:

- **Why do we need compact neural architectures**
  - Limited resource on edge devices
  - Faster inference speed

- **Compact architectures**
  - SqueezeNet (Large parameter reduction)
  - ShuffleNet (Channel shuffle + Group convolution)
  - MobileNetV1 (Depthwise separable convolution)
  - MobileNetV2 (Inverted bottleneck)

Reading materials

SqueezeNet	Iandola, Forrest N., et al. "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size." arXiv preprint arXiv:1602.07360 (2016).
MobileNet	Howard, Andrew G., et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications." arXiv preprint arXiv:1704.04861 (2017).
ShuffleNet	Zhang, Xiangyu, et al. "Shufflenet: An extremely efficient convolutional neural network for mobile devices." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.
MobileNetV2	Sandler, Mark, et al. "Mobilenetv2: Inverted residuals and linear bottlenecks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.