Designing Light-Weight Networks for Mobile Applications

Jianping Fan
Dept of Computer Science
UNC-Charlotte

Course Website:
http://webpages.uncc.edu/jfan/itcs5152.html

Outline

1. Background
2. SqueezeNet
3. MobileNet v1
4. MobileNet v2
5. ShuffleNet
6. Summary
7. References

Deep Learning on Mobile

Phones

Drones

Robots

Glasses

Self Driving Cars

Source:
http://isca2016.eecs.umich.edu/wp-content/uploads/2016/07/4A-1.pdf

Deep Learning on Mobile

Phones

Drones

Robots

Glasses

Self Driving Cars

Battery Constrained!

Source:
http://isca2016.eecs.umich.edu/wp-content/uploads/2016/07/4A-1.pdf

Special Requirements from Mobile Applications

- **Low Battery**→ small size of model, cost-efficient inference methods, short inference time & cost
- **Small Memory**→ small size of model, less parameters, less data-hungry for model updating
- **Quick Response**: →short inference time

⬇

Small Model

For Mobile Applications

What's the "Right" Neural Network?

- Sufficiently high accuracy
- Low computational complexity
- Low energy usage
- Small model size

Accuracy *vs.* Energy Cost & Model Size

Why smaller models?

Model	MACC	COMP	ADD	DIV	Activations	Params	SIZE(MB)
SimpleNet	1.9G	1.82M	1.5M	1.5M	6.38M	6.4M	24.4
SqueezeNet	861.34M	9.67M	226K	1.51M	12.58M	1.25M	4.7
Inception v4*	12.27G	21.87M	53.42M	15.09M	72.56M	42.71M	163
Inception v3*	5.72G	16.53M	25.94M	8.97M	41.33M	23.83M	91
Incep-Resv2*	13.18G	31.57M	38.81M	25.06M	117.8M	55.97M	214
ResNet-152	11.3G	22.33M	35.27M	22.03M	100.11M	60.19M	230
ResNet-50	3.87G	10.89M	16.21M	10.59M	46.72M	25.56M	97.70
AlexNet	7.27G	17.69M	4.78M	9.55M	20.81M	60.97M	217.00
GoogleNet	16.04G	161.07M	8.83M	16.64M	102.19M	7M	40
NIN	11.06G	28.93M	380K	20K	38.79M	7.6M	29
VGG16	154.7G	196.85M	10K	10K	288.03M	138.36M	512.2

*Inception v3, v4 did not have any Caffe model, so we reported their size related information from MXNet and Tensorflow respectively. Inception-ResNet-V2 would take 60 days of training with 2 Titan X to achieve the reported accuracy. Statistics are obtained using `http://dgschwend.github.io/netscope`

A.2 GENERALIZATION SAMPLES

Why smaller models?

Model	MACC	COMP	ADD	DIV	Activations	Params	SIZE(MB)
SimpleNet	1.9G	1.82M	1.5M	1.5M	6.38M	6.4M	24.4
SqueezeNet	861.34M	9.67M	226K	1.51M	12.58M	1.25M	4.7
Inception v4*	12.27G	21.87M	53.42M	15.09M	72.56M	42.71M	163
Inception v3*	5.72G	16.53M	25.94M	8.97M	41.33M	23.83M	91
Incep-Resv2*	13.18G	31.57M	38.81M	25.06M	117.8M	55.97M	214
ResNet-152	11.3G	22.33M	35.27M	22.03M	100.11M	60.19M	230
ResNet-50	3.87G	10.89M	16.21M	10.59M	46.72M	25.56M	97.70
AlexNet	7.27G	17.69M	4.78M	9.55M	20.81M	60.97M	217.00
GoogleNet	16.04G	161.07M	8.83M	16.64M	102.19M	7M	40
NIN	11.06G	28.93M	380K	20K	38.79M	7.6M	29
VGG16	154.7G	196.85M	10K	10K	288.03M	138.36M	512.2

*Inception v3, v4 did not have any Caffe model, so we reported their size related information from MXNet and Tensorflow respectively. Inception-ResNet-V2 would take 60 days of training with 2 Titan X to achieve the reported accuracy. Statistics are obtained using `http://dgschwend.github.io/netscope`

A.2 GENERALIZATION SAMPLES

Why smaller models?

Model	MACC	COMP	ADD	DIV	Activations	Params	SIZE(MB)
SimpleNet	1.9G	1.82M	1.5M	1.5M	6.38M	6.4M	24.4
SqueezeNet	861.34M	9.67M	226K	1.51M	12.58M	1.25M	4.7
Inception v4*	12.27G	21.87M	53.42M	15.09M	72.56M	42.71M	163
Inception v3*	5.72G	16.53M	25.94M	8.97M	41.33M	23.83M	91
Incep-Resv2*	13.18G	31.57M	38.81M	25.06M	117.8M	55.97M	214
ResNet-152	11.3G	22.33M	35.27M	22.03M	100.11M	60.19M	230
ResNet-50	3.87G	10.89M	16.21M	10.59M	46.72M	25.56M	97.70
AlexNet	7.27G	17.69M	4.78M	9.55M	20.81M	60.97M	217.00
GoogleNet	16.04G	161.07M	8.83M	16.64M	102.19M	7M	40
NIN	11.06G	28.93M	380K	20K	38.79M	7.6M	29
VGG16	154.7G	196.85M	10K	10K	288.03M	138.36M	512.2

*Inception v3, v4 did not have any Caffe model, so we reported their size related information from MXNet and Tensorflow respectively. Inception-ResNet-V2 would take 60 days of training with 2 Titan X to achieve the reported accuracy. Statistics are obtained using `http://dgschwend.github.io/netscope`

A.2 GENERALIZATION SAMPLES

Why smaller models?

Model Structure		Input Dimension (H × W × D)	Kernel Stride/Padding	Kernel Dimension (H × W × D)	Output Vectors Dimension (H × W × D)	Parameters	Operations	Data Size (Bytes), # of Fragmented Packets by PDU Size	Execution Time (ms) 216 MHz	Execution Time (ms) 80 MHz
Input		32 × 32 × 3						3072 B, 27 pkts	55	301
CNN Layer 1	Conv 1	32 × 32 × 3	1/2	5 × 5 × 32	32 × 32 × 32	2432	4.9 M	32,768 B, 283 pkts	55	301
	Pool 1	32 × 32 × 32	2/0	3 × 3	16 × 16 × 32		73.7 K	8192 B, 71 pkts	3	14
	Relu 1	16 × 16 × 32			16 × 16 × 32			8192 B, 71 pkts	<1	<1
CNN Layer 2	Conv 1	16 × 16 × 32	1/2	5 × 5 × 32	16 × 16 × 32	25,632	13.1 M	8192 B, 71 pkts	79	427
	Relu 1	16 × 16 × 32			16 × 16 × 32			8192 B, 71 pkts	< 1	1
	Pool 1	16 × 16 × 32	2/0	3 × 3	8 × 8 × 32		18.4 K	2048 B, 18 pkts	1	4
CNN Layer 3	Conv 1	8 × 8 × 32	1/2	5 × 5 × 32	8 × 8 × 64	51,264	6.6 M	4096 B, 36 pkts	39	212
	Relu 1	8 × 8 × 64			8 × 8 × 64			4096 B, 36 pkts	< 1	1
	Pool 1	8 × 8 × 64	2/0	3 × 3	4 × 4 × 64		9.2 K	1024 B, 9 pkts	< 1	1
Fully Connected & Softmax Output Layer 4		4 × 4 × 64			10	10,240	20 K	10 B, 1 pkts	<1 Total: 178	<1 Total: 962

method	# layers	# params	error (%)
Maxout [10]			9.38
NIN [25]			8.81
DSN [24]			8.22
FitNet [35]	19	2.5M	8.39
Highway [42, 43]	19	2.3M	7.54 (7.72±0.16)
Highway [42, 43]	32	1.25M	8.80
ResNet	20	0.27M	8.75
ResNet	32	0.46M	7.51
ResNet	44	0.66M	7.17
ResNet	56	0.85M	6.97
ResNet	110	1.7M	**6.43** (6.61±0.16)
ResNet	1202	19.4M	7.93

Model Name	Dataset	Number of parameters (MB)
E-Net	Cityscapes	1.5
ICNet	Cityscapes	30.1
PSPNet (ResNet-101)	Cityscapes	260.2
Dilated Frontend (VGG)	Cityscapes	512.4
FCN8s (VGG)	Cityscapes	512.5
Dilated Context (VGG)	Cityscapes	512.6
Segnet (VGG)	Pascal	112.4
Deeplab v2 (VGG)	Pascal	144.5
FCN8s (ResNet-101)	Pascal	162.9
Deeplab v2 (ResNet-101)	Pascal	168.4
PSPNet (ResNet-101)	Pascal	272.7
Dilated Frontend (VGG)	Pascal	512.4
FCN8s (VGG)	Pascal	513.0
CRF-RNN (VGG)	Pascal	513.0
Dilated Context (VGG)	Pascal	538.4

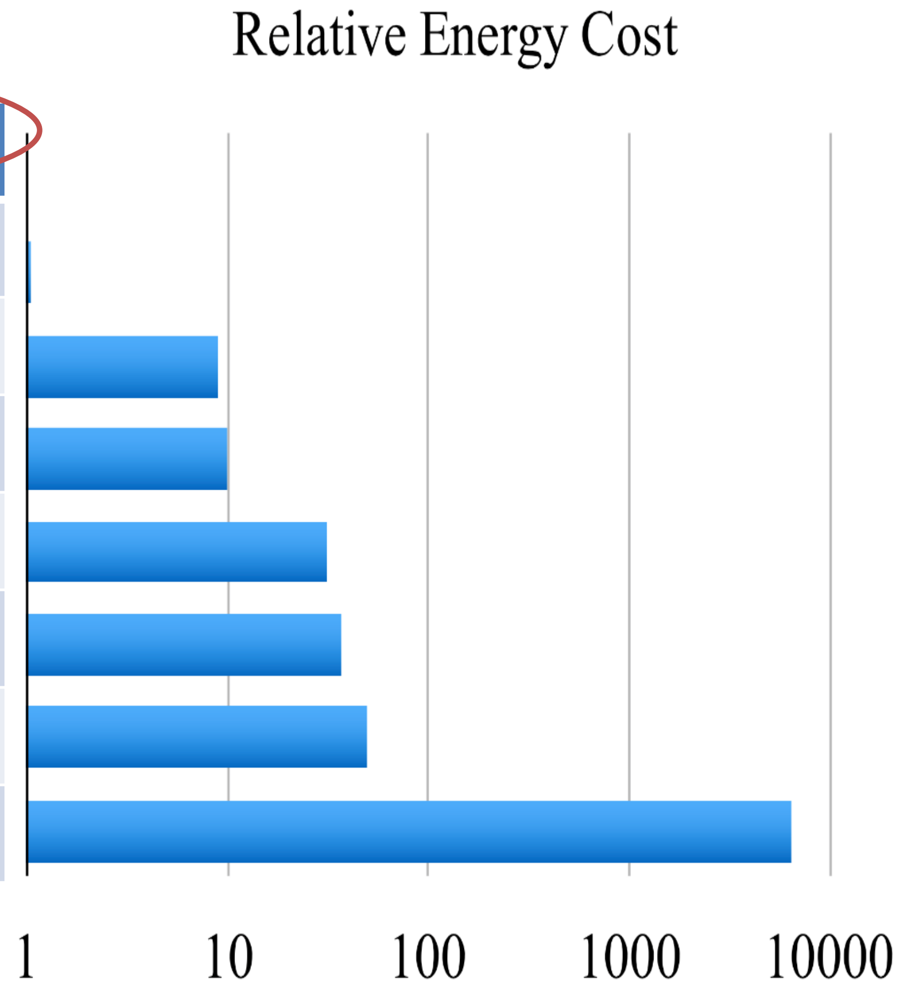
Why smaller models?

Networks		Input	Output	Layers	Parameters
FCN	FCN-5	26,752	26,752	5	55 millions
	FCN-8	26,752	26,752	8	58 millions
CNN	AlexNet	150,528	1,000	4	61 millions
	ResNet-50	150,528	1,000	50	3.8 billions
RNN	LSTM-32	10,000	10,000	2	13 millions
	LSTM-64	10,000	10,000	2	13 millions

Model	top-1 err, %	top-5 err, %	#params	time/batch 16
ResNet-50	24.01	7.02	25.6M	49
ResNet-101	22.44	6.21	44.5M	82
ResNet-152	22.16	6.16	60.2M	115
WRN-50-2-bottleneck	21.9	6.03	68.9M	93
pre-ResNet-200	21.66	5.79	64.7M	154

Why smaller models?

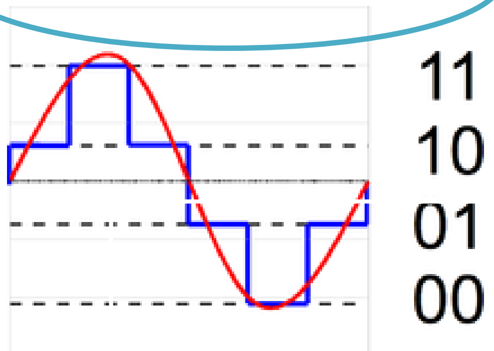
Operation	Energy [pJ]	Relative Cost
32 bit int ADD	0.1	1
32 bit float ADD	0.9	9
32 bit Register File	1	10
32 bit int MULT	3.1	31
32 bit float MULT	3.7	37
32 bit SRAM Cache	5	50
32 bit DRAM Memory	640	6400



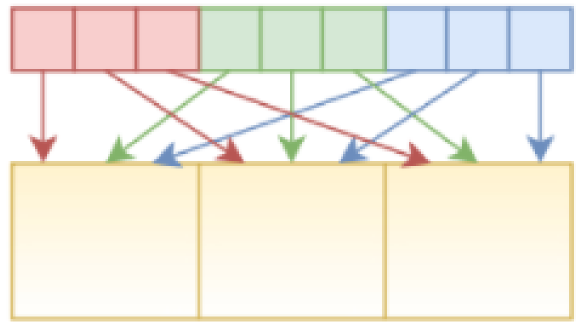
Source:
http://isca2016.eecs.umich.edu/wp-content/uploads/2016/07/4A-1.pdf

Techniques for Creating Fast & Energy-Efficient DNNs

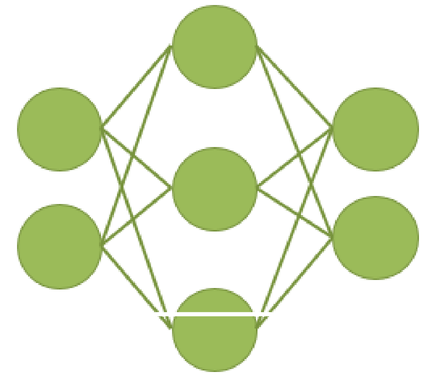
Model Compression



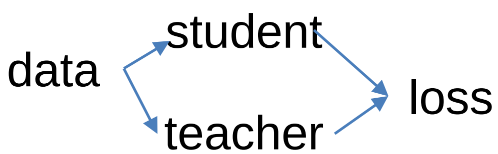
New Layer Types



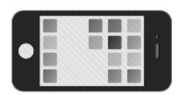
Original Net Design



Knowledge Distillation



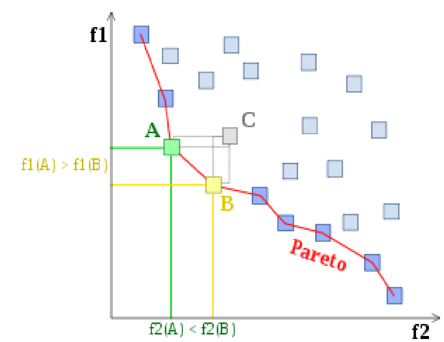
Efficient Implementation



cuDNN

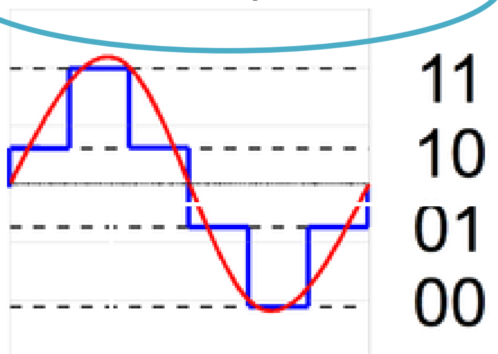
INTEL® MKL-DNN

Design Space Exploration

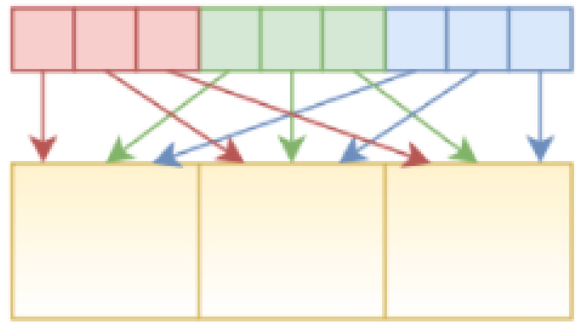


Techniques for Creating Fast & Energy-Efficient DNNs

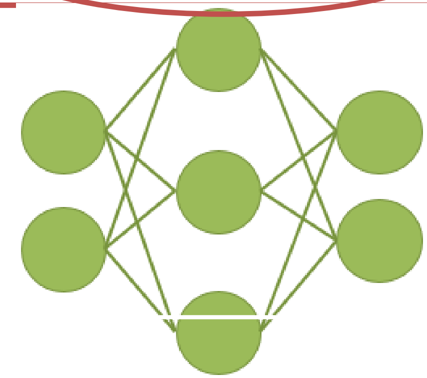
Model Compression



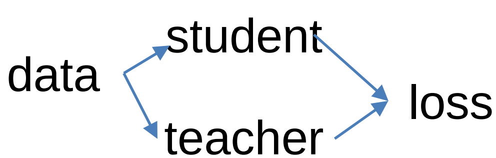
New Layer Types



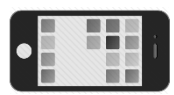
Original Net Design



Knowledge Distillation



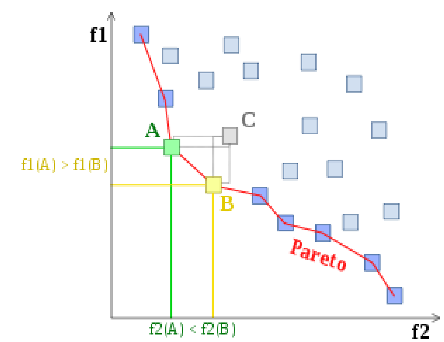
Efficient Implementation



cuDNN

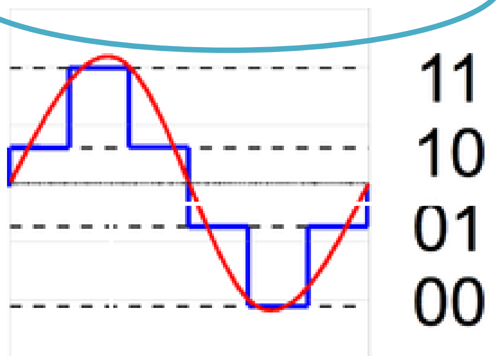
INTEL® MKL-DNN

Design Space Exploration

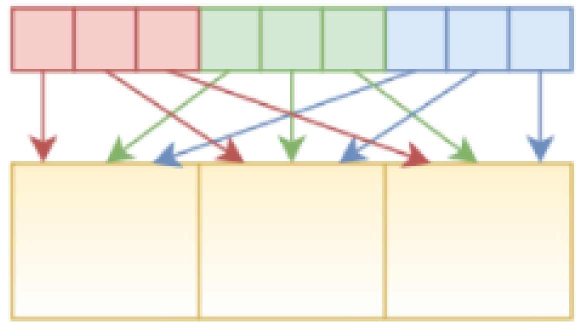


Techniques for Creating Fast & Energy-Efficient DNNs

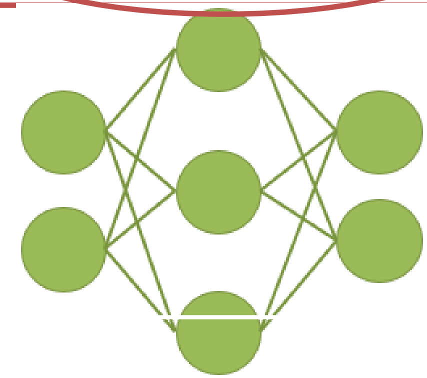
Model Compression



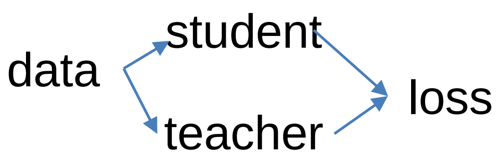
New Layer Types



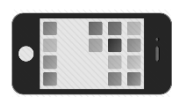
Original Net Design



Knowledge Distillation



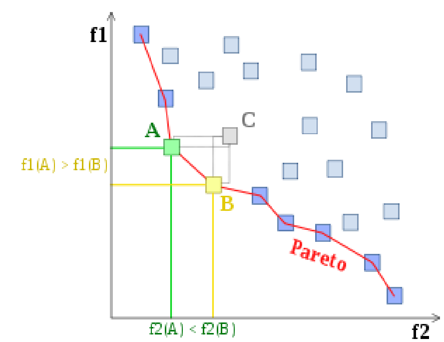
Efficient Implementation



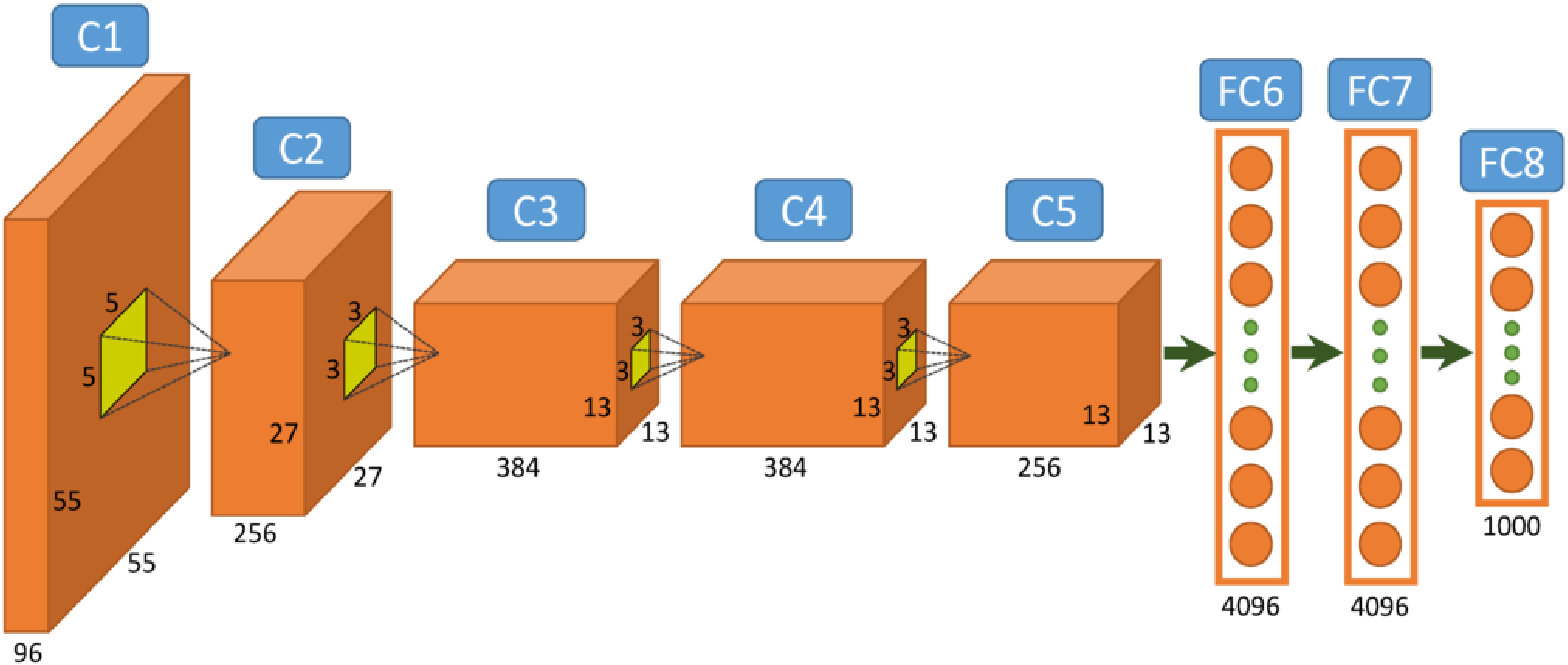
cuDNN

INTEL® MKL-DNN

Design Space Exploration



Where computational cost comes from?



Where **computational cost** comes from in deep networks?

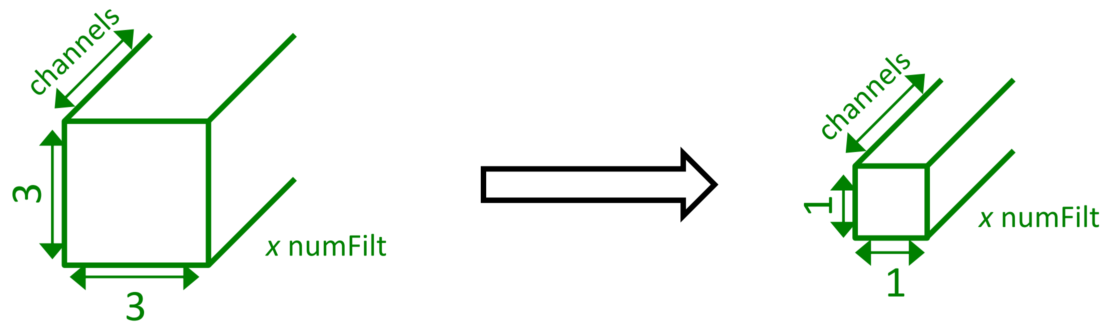
- **Kernel numbers** for convolution

- **Channel numbers** for image inputs or feature maps

- **Size of feature maps**

Kernel Reduction

REDUCING THE SIZE (HEIGHT AND WIDTH) OF FILTERS



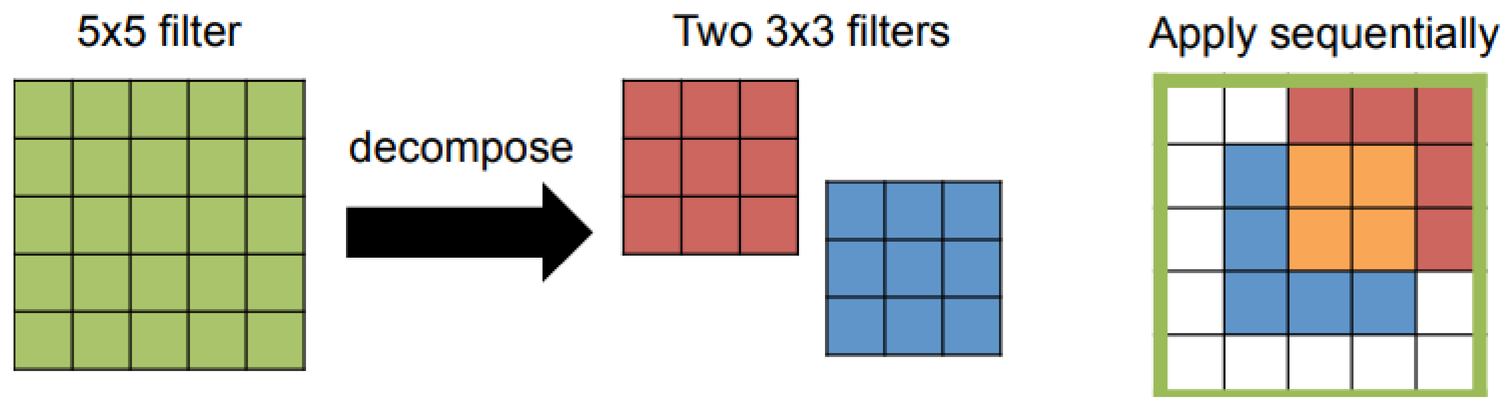
While **1x1 filters** cannot see outside of a 1-pixel radius, they retain the ability to combine and reorganize information across channels.

SqueezeNet (2016): we found that we could replace half the 3x3 filters with 1x1's without diminishing accuracy

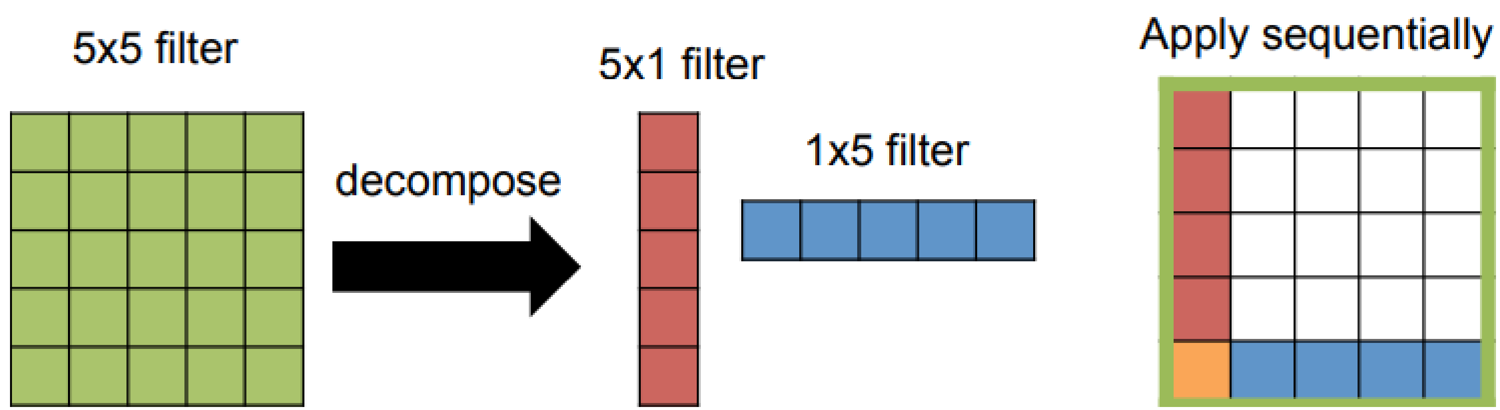
SqueezeNext (2018): eliminate most of the 3x3 filters – we use mix of 1x1, 3x1, and 1x3 filters (and still retain accuracy)

Kernel Reduction

Decomposing larger filter into smaller ones



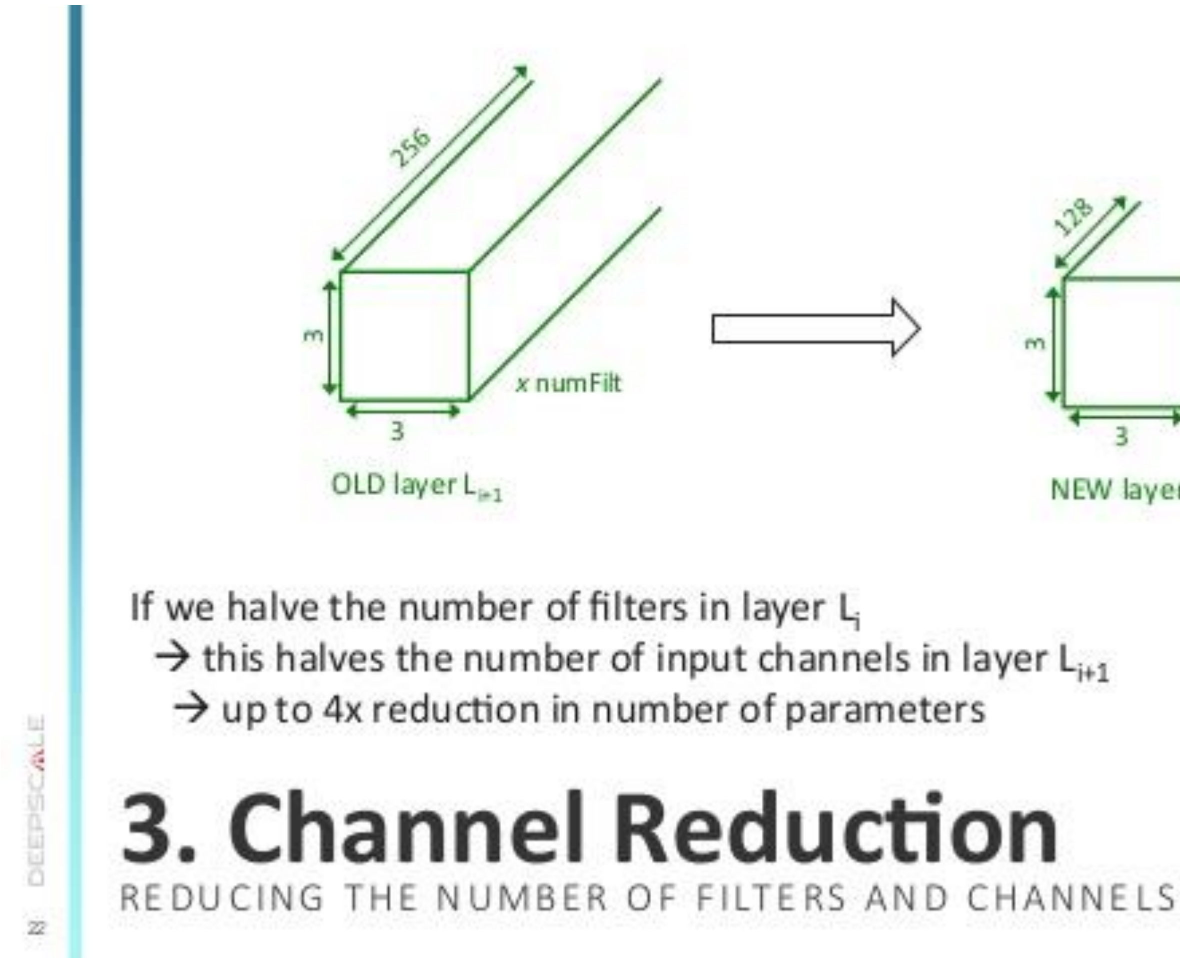
(a) Constructing a 5×5 support from 3×3 filters. Used in VGG-16.



(b) Constructing a 5×5 support from 1×5 and 5×1 filter. Used in GoogleNet/Inception v3 and v4.

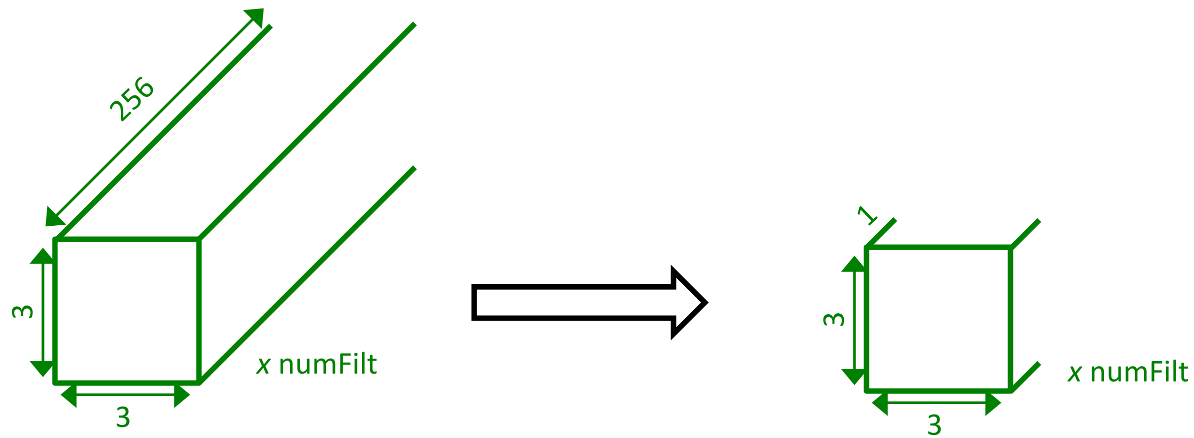
Channel Reduction

REDUCING THE NUMBER of CHANNELS



Depthwise Separable Convolution

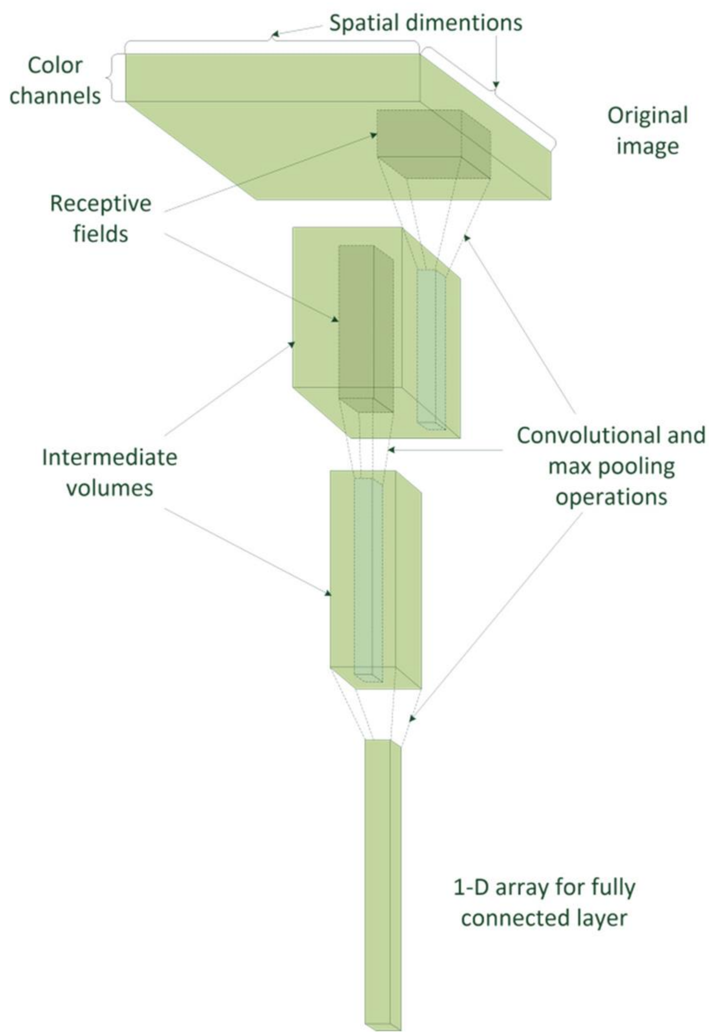
ALSO CALLED: "GROUP CONVOLUTIONS" or "CARDINALITY"



used in recent papers such as MobileNets and ResNeXt

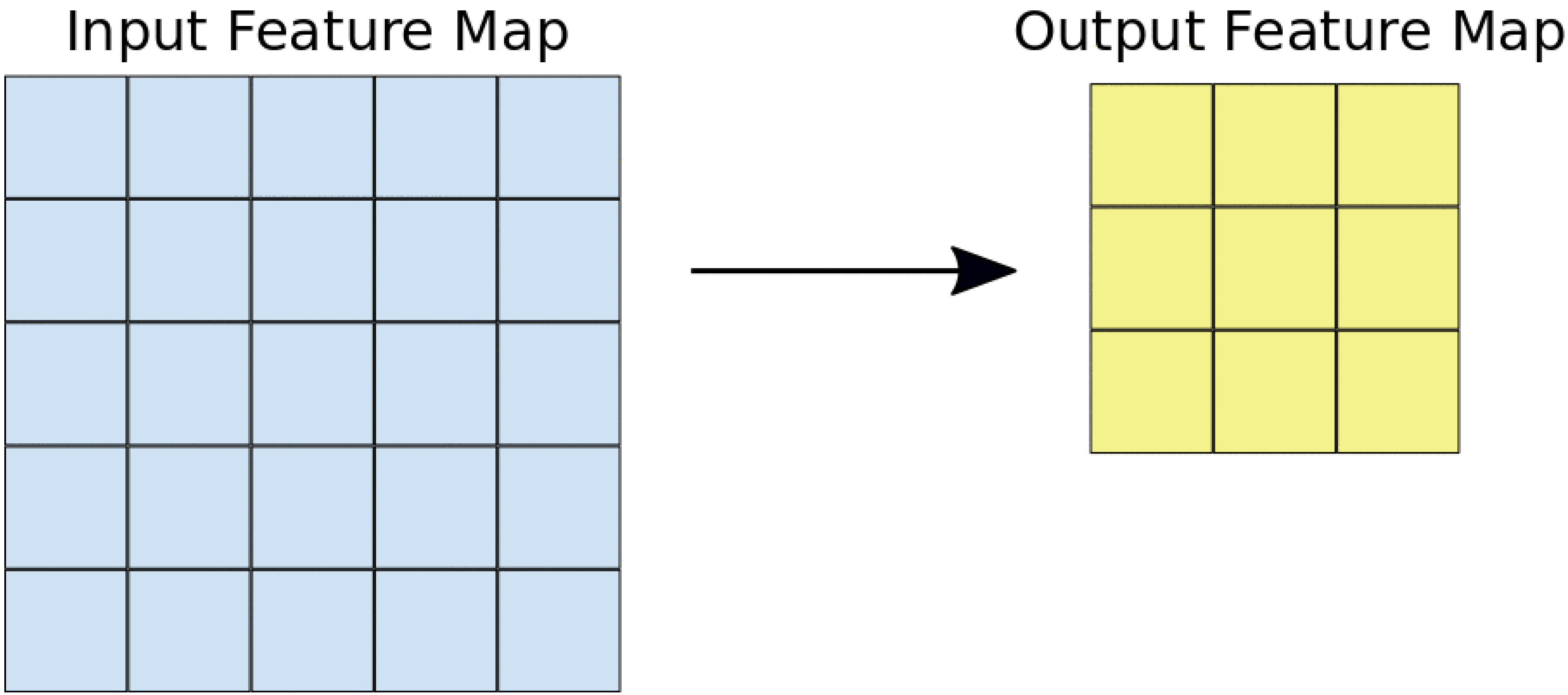
Channel Reduction

REDUCING THE NUMBER of CHANNELS



Feature Map Reduction

REDUCING THE SIZE of FEATURE MAPS



Four Advantages of Light-Weight (Smaller) Networks:

(1) Smaller CNNs require **less communication** across servers during distributed **training**.

(2) Smaller CNNs require **less bandwidth** to export a new model from the cloud to a mobile device.

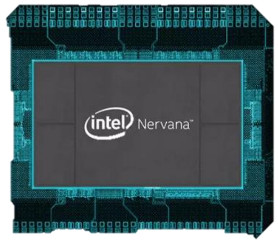
(3) Smaller CNNs are **more feasible** to deploy on **FPGAs** and other **hardware with limited memory**.

(4) Smaller CNNs result in **less inference time and storage space**.

DNN Challenges in Training

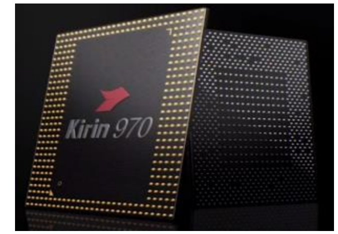
TFLite

Nervana



Apple AI

Hawaii NPU



Don't support full training due to energy inefficiency

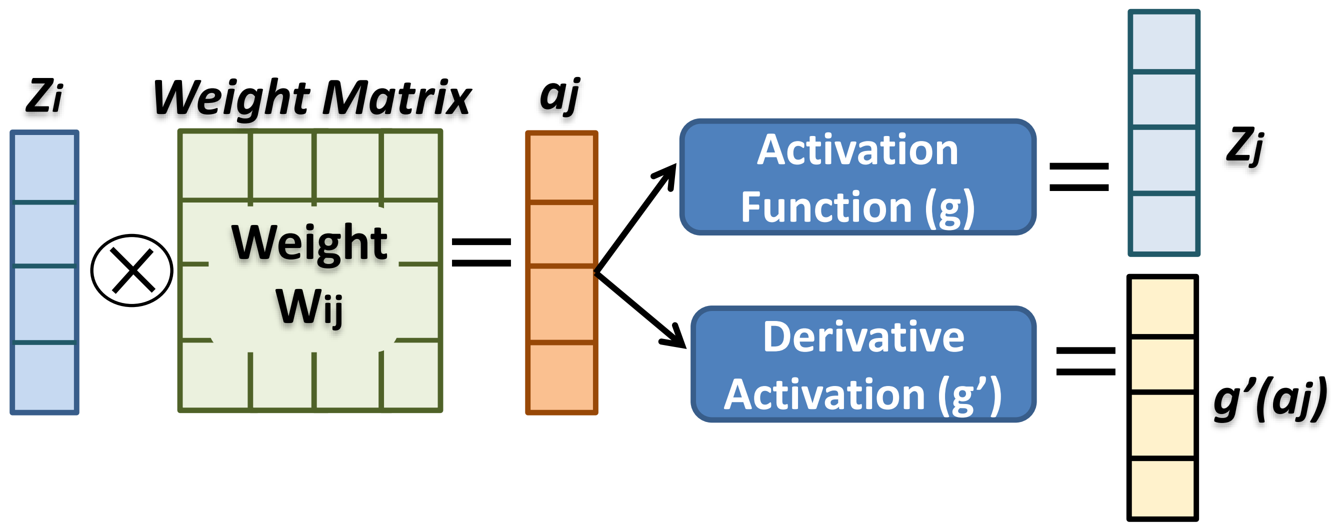
How about using existing PIM architectures?

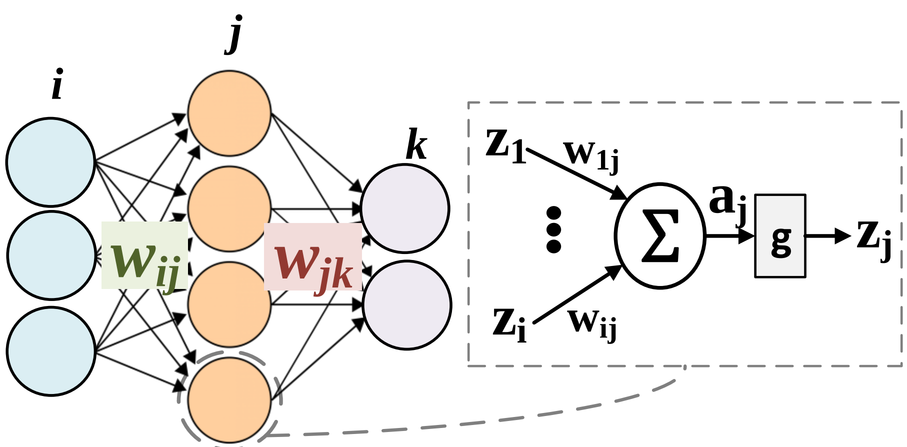
DNN/CNN Training

1. Highly Parallel Architecture ✓
2. High Precision Computation ✗
3. Large Data Movement ✗

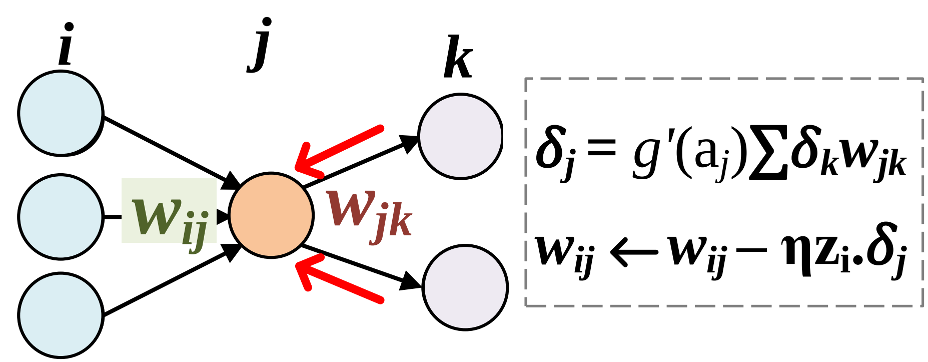
Mohsen Imani, Saransh Gupta, Yeseo, Kim, Tajana Rosing:University of California San Diego

Neural Networks



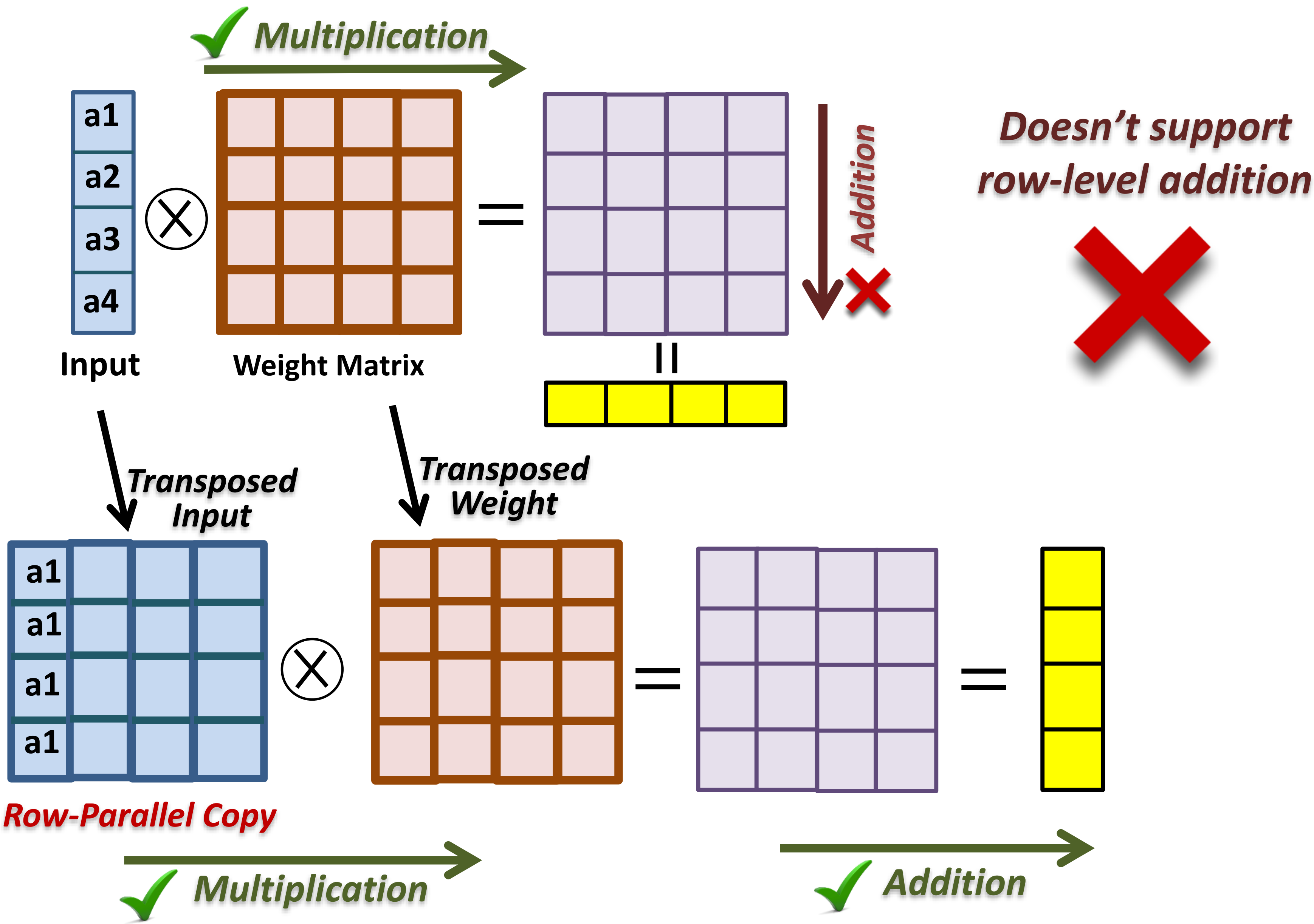


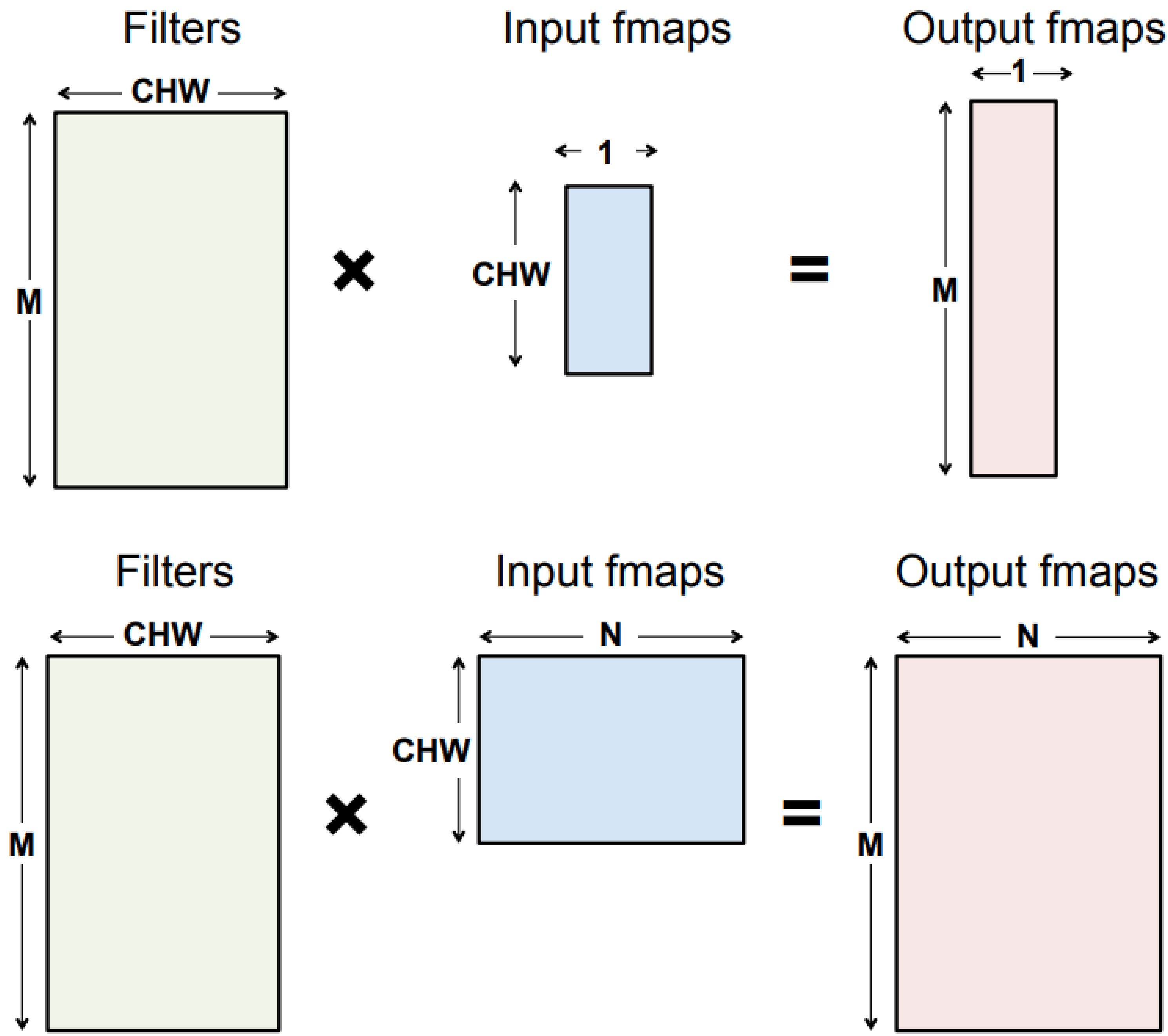
Feed Forward



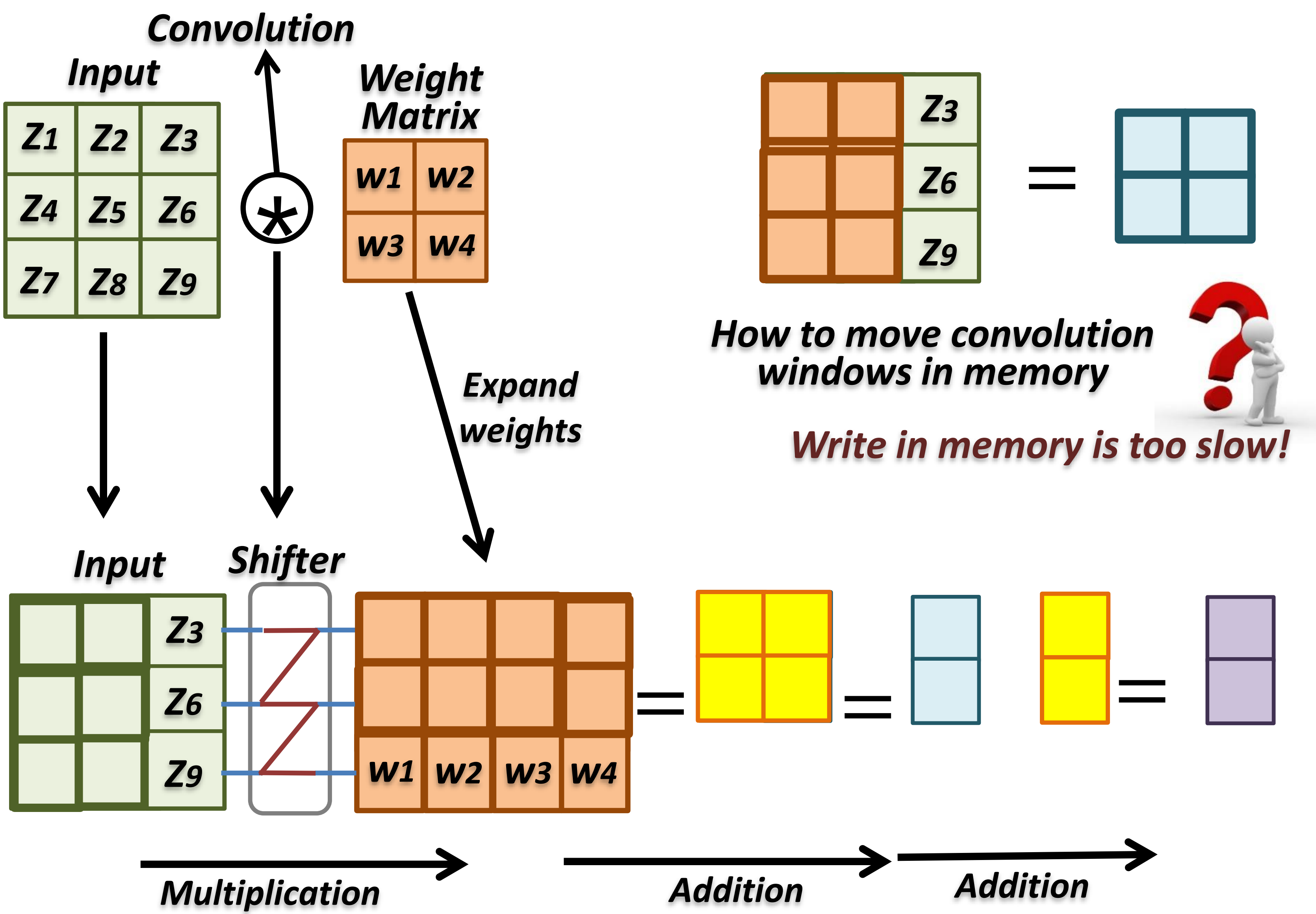
Back Propagation

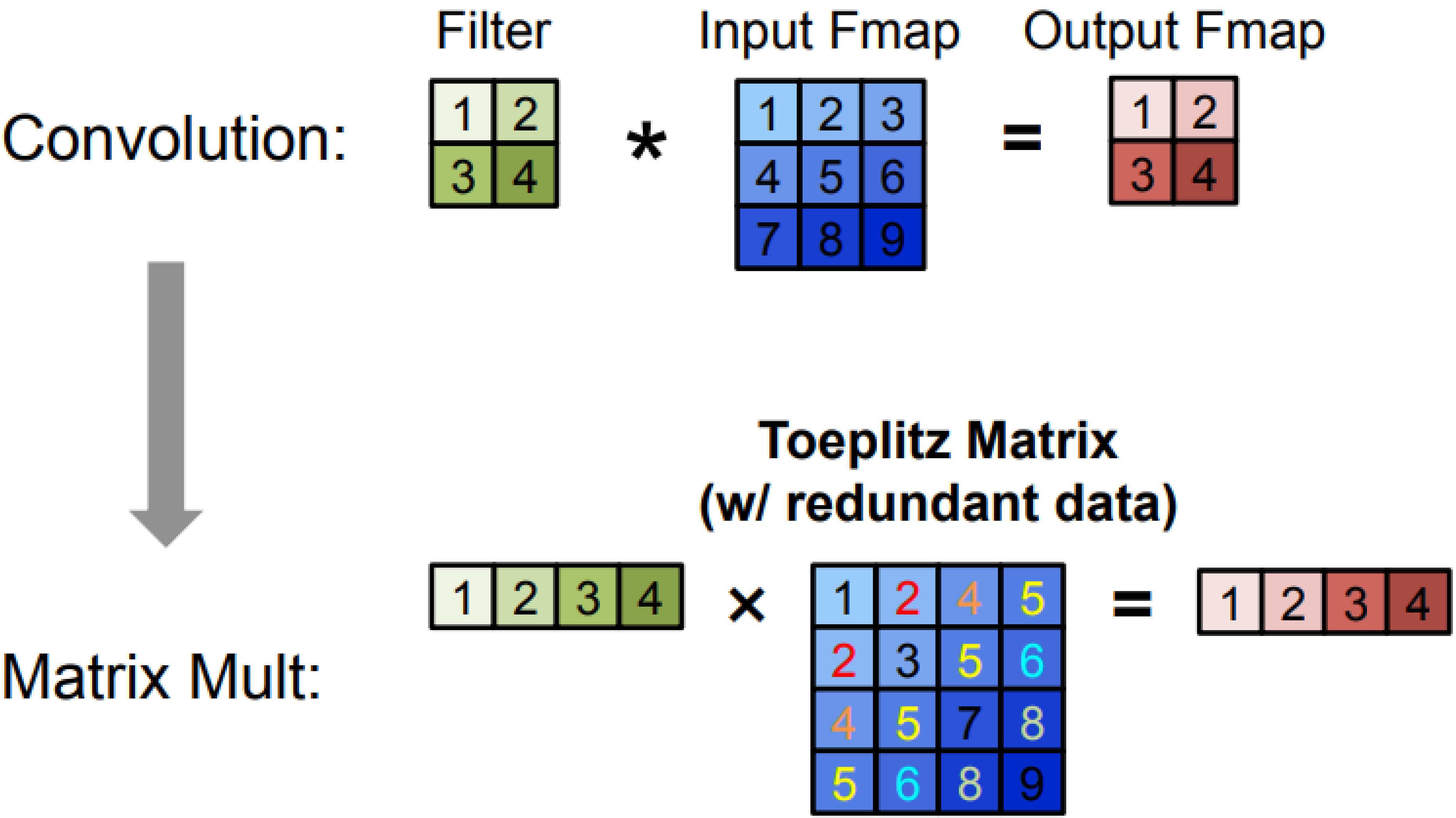
Vector-Matrix Multiplication



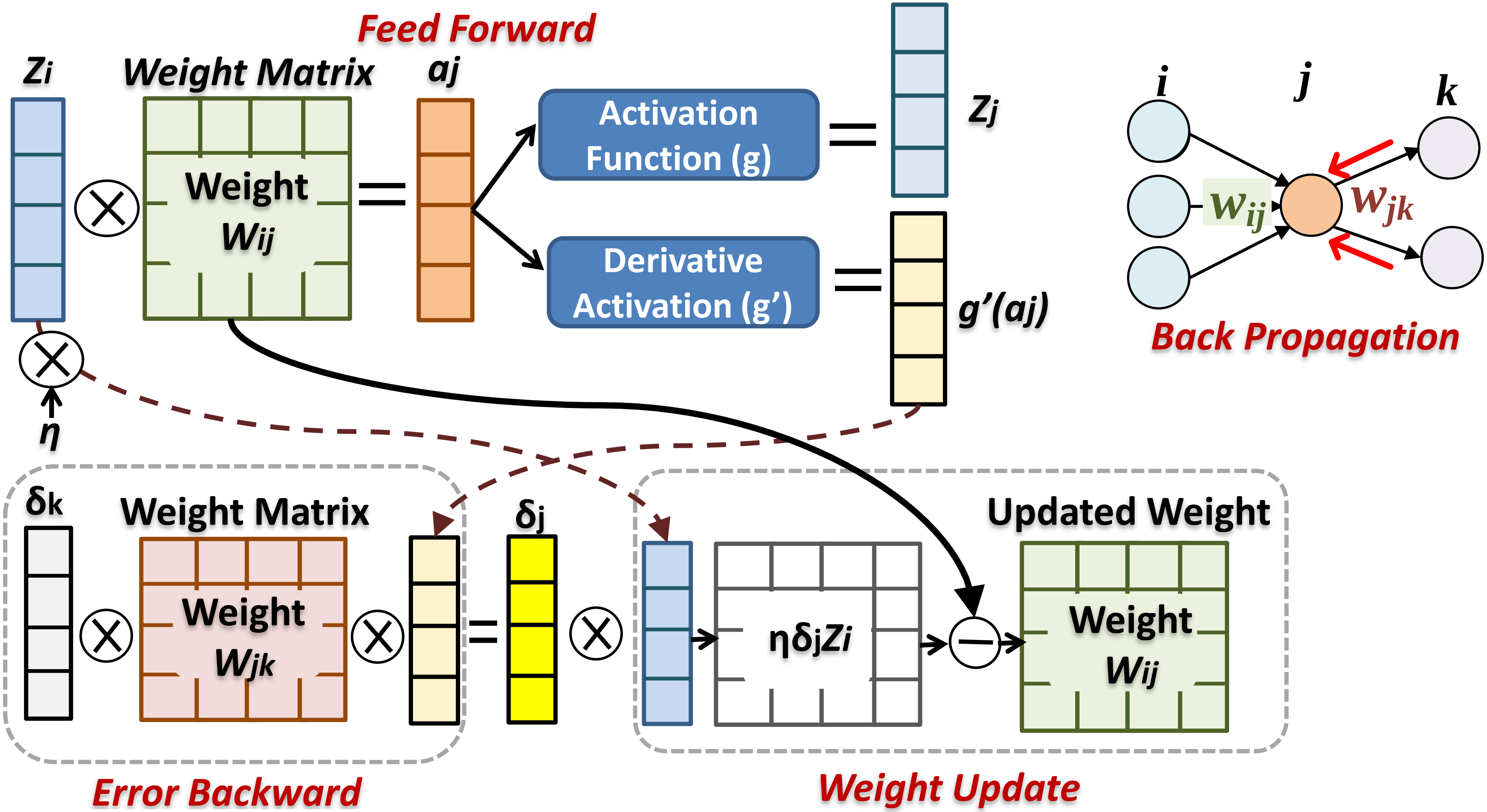

Filters
CHW
M
Input fmaps
1
CHW
Output fmaps
1
M
Filters
CHW
M
Input fmaps
N
CHW
Output fmaps
N
M

Neural Network: Convolution Layer




Filter
Input Fmap
Output Fmap
Convolution:
1 2
3 4
*
1 2 3
4 5 6
7 8 9
=
1 2
3 4
Toeplitz Matrix
(w/ redundant data)
Matrix Mult:
1 2 3 4
×
1 2 4 5
2 3 5 6
4 5 7 8
5 6 8 9
=
1 2 3 4

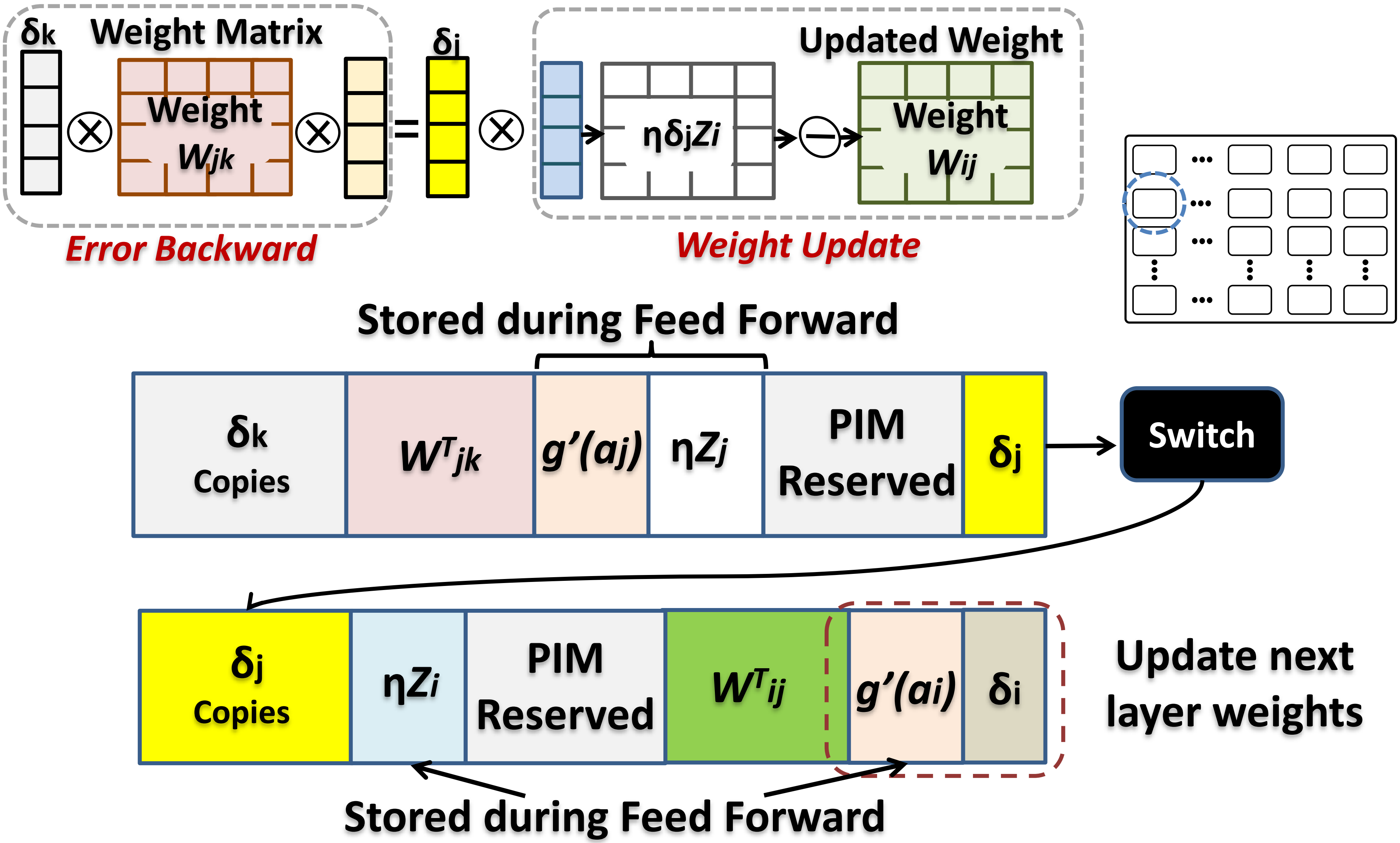
Neural Network: Back Propagation



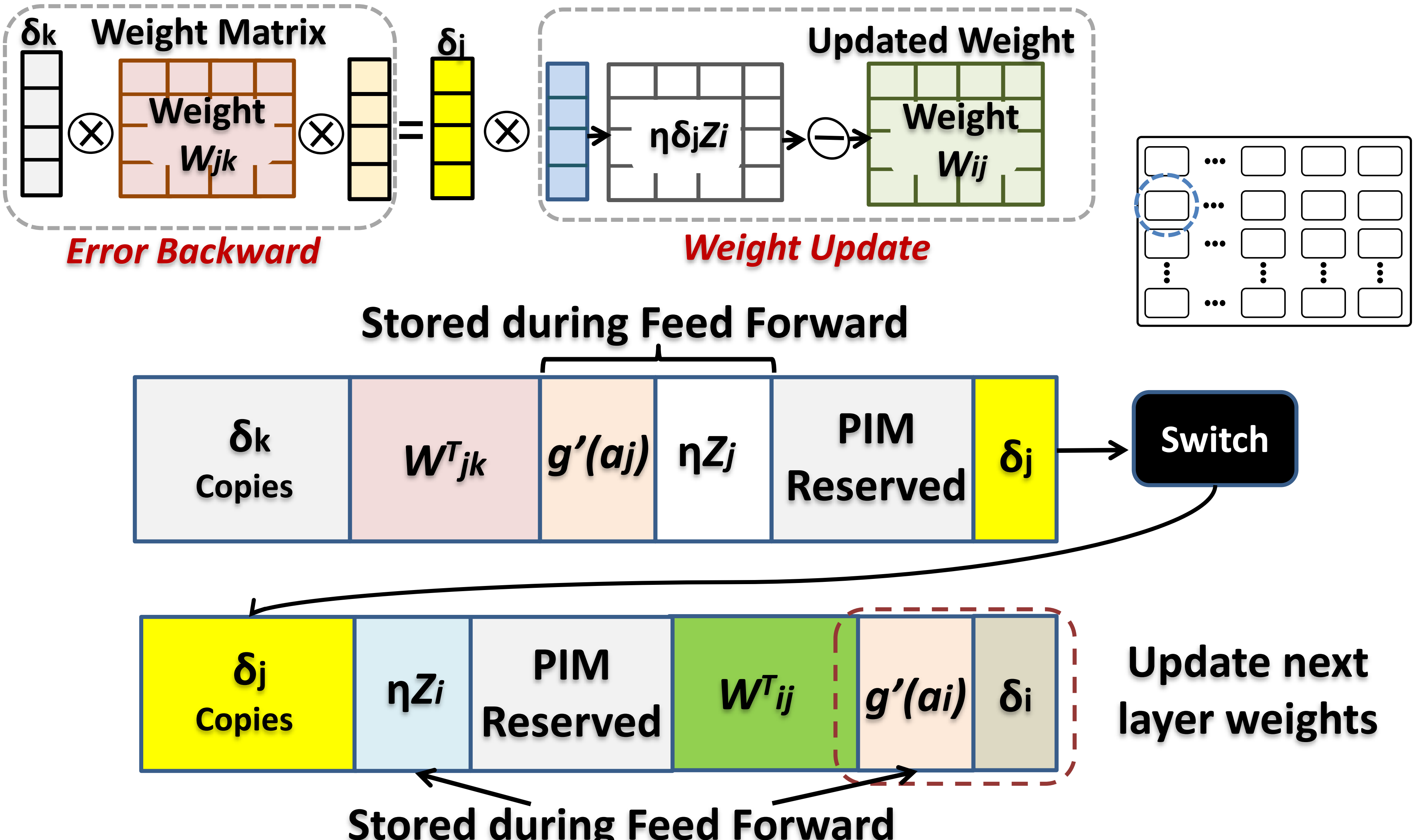
$$\delta_j = g'(a_j)\sum \delta_k w_{jk}$$

$$w_{ij} \leftarrow w_{ij} - \eta z_i . \delta_j$$

Memory Layout: Back Propagation



Memory Layout: Back Propagation



Approaches for Applying Deep Networks on Mobile Devices

a. **Designing Light-Weight Deep Networks** directly towards mobile applications

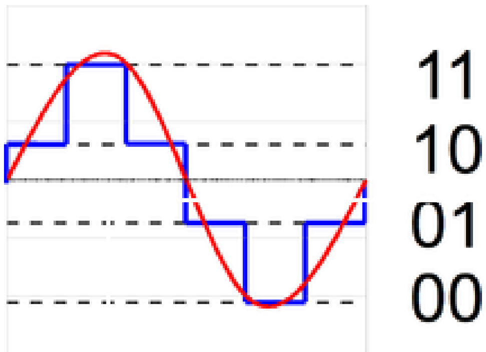
b. Performing **Model Compression** over Heavy but Accurate Network

c. **Knowledge Distillation** from a Heavy Large Teacher Network to a Light-Weight Small Student Network

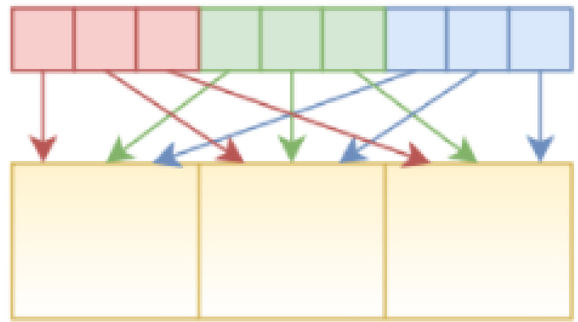
d. **Searching Light-Weight Network** according to Pre-defined Constraints

Techniques for Creating Fast & Energy-Efficient DNNs

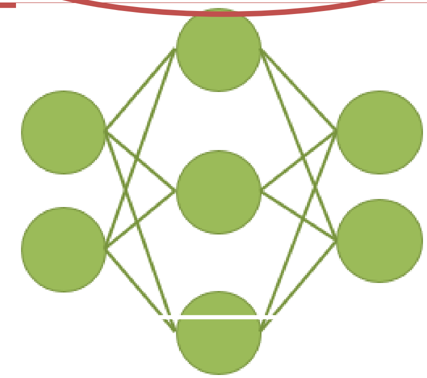
Model Compression



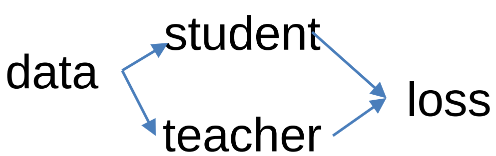
New Layer Types



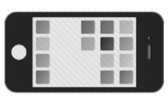
Original Net Design



Knowledge Distillation



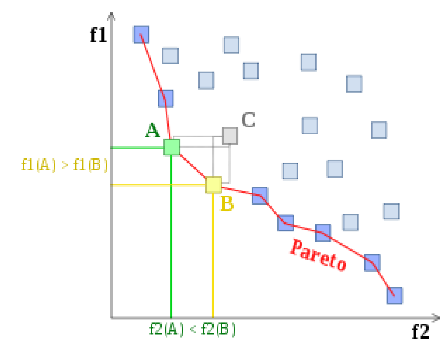
Efficient Implementation



cuDNN

INTEL® MKL-DNN

Design Space Exploration



SqueezeNet

Three issues to define **size of neural networks**:

- **Kernel numbers** for convolution
- **Channel numbers** for image input or feature maps
- **Size of feature maps**

Techniques for Small Deep Neural Networks

- Remove Fully-Connected Layers
- Kernel Reduction (3x3 -> 1x1) **SqueezeNet**
- Channel Reduction
- Depthwise Separable Convolutions

Key ideas

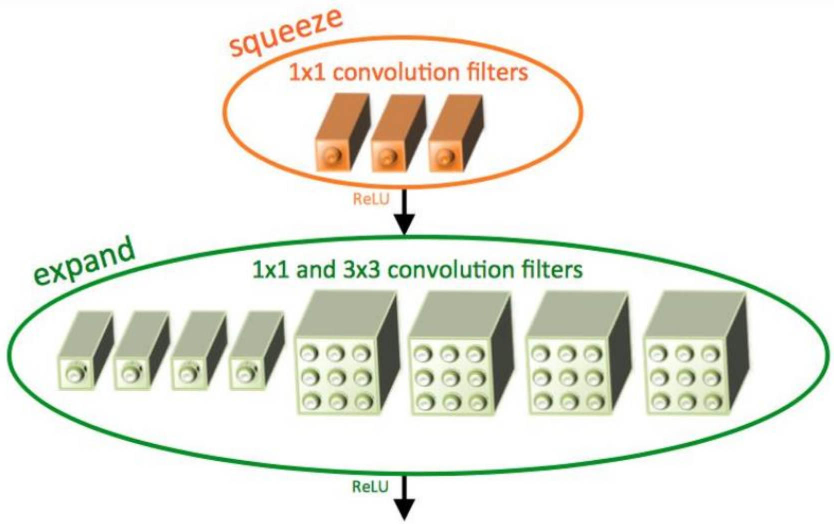
- ***Strategy 1.*** Replace 3×3 filters with 1x1 filters
  - Parameters per filter: (3×3 filter) = 9 * (1x1 filter)

- ***Strategy 2.*** Decrease the number of **input channels** to 3×3 filters by using **squeeze layers**
  - Total # of parameters: (# of input channels) * (# of filters) * (# of parameters per filter)

- ***Strategy 3.* Down-sample *late in the network*** so that convolution layers have large activation maps
  - Size of activation maps: the size of input data, the choice of layers in which to down-sample in the CNN architecture

Iandola F N, Han S, Moskewicz M W, et al. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size[J]. arXiv preprint arXiv:1602.07360, 2016.

SqueezeNet



- **Fire module** is consist of:
  - A **squeeze** convolution layer
    - full of $S_{1\times1}$# of 1×1 filters
  - An **expand** layer
    - mixture of $e_{1\times1}$ # of 1×1 and $e_{3\times3}$ # of 3×3 filters

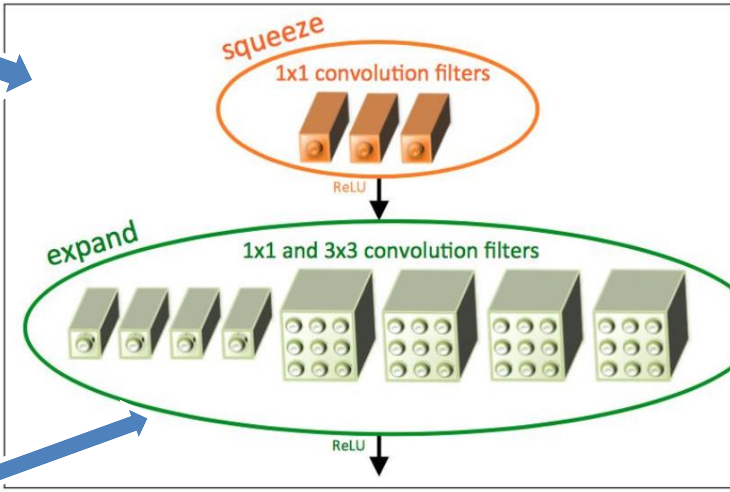
A Fire module is comprised of: a squeeze convolution layer (which has only 1x1 filters), feeding into an expand layer that has a mix of 1x1 and 3x3 convolution filters

Iandola F N, Han S, Moskewicz M W, et al. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size[J]. arXiv preprint arXiv:1602.07360, 2016.

SqueezeNet

- ***Strategy 2.*** Decrease the number of **input channels** to 3×3 filters
  - Total # of parameters: (# of input channels) * (# of filters) * (# of parameters per filter)

A Fire module is comprised of: a squeeze convolution layer (which has only 1x1 filters), feeding into an expand layer that has a mix of 1x1 and 3x3 convolution filters



How much can we limit $S_{1\times1}$?

Squeeze Layer

Set $S_{1\times1} < (e_{1\times1} + e_{3\times3})$

limits the # of input channels to 3×3 filters

- ***Strategy 1.*** Replace 3×3 filters with 1x1 filters
  - Parameters per filter: (3×3 filter) = 9 * (1×1 filter)

How much can we replace 3×3 with 1×1?

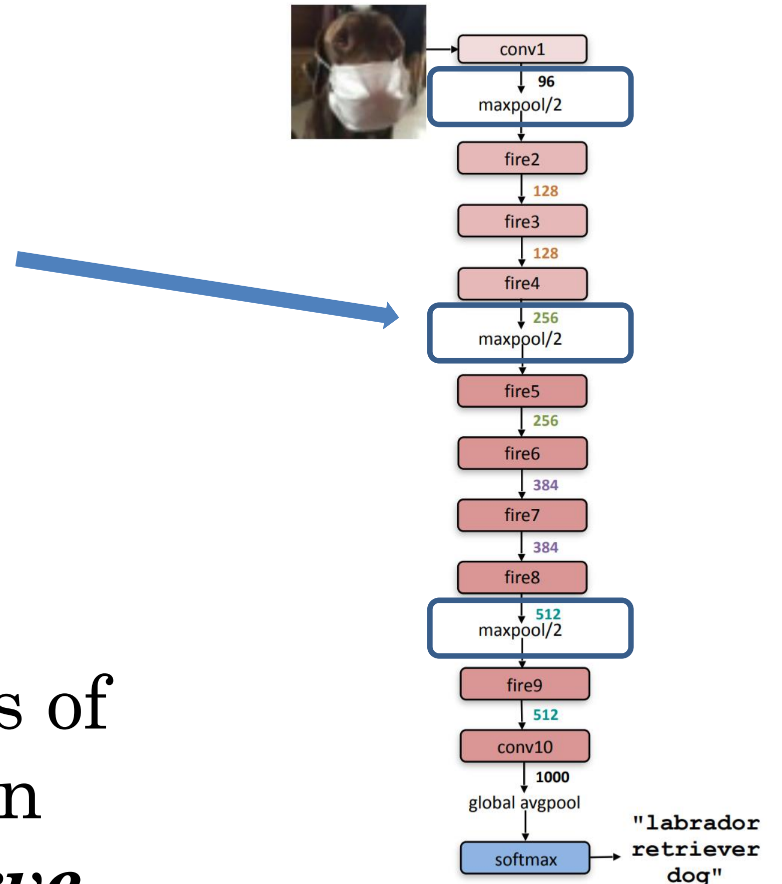
($e_{1\times1}$ *vs* $e_{3\times3}$)?

Iandola F N, Han S, Moskewicz M W, et al. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size[J]. arXiv preprint arXiv:1602.07360, 2016.

SqueezeNet

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

◦ Size of activation maps: the size of input data, the choice of layers in which to downsample in the CNN architecture



These relative late placements of pooling concentrates activation maps at later phase to ***preserve higher accuracy*** .

Iandola F N, Han S, Moskewicz M W, et al. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size[J]. arXiv preprint arXiv:1602.07360, 2016.

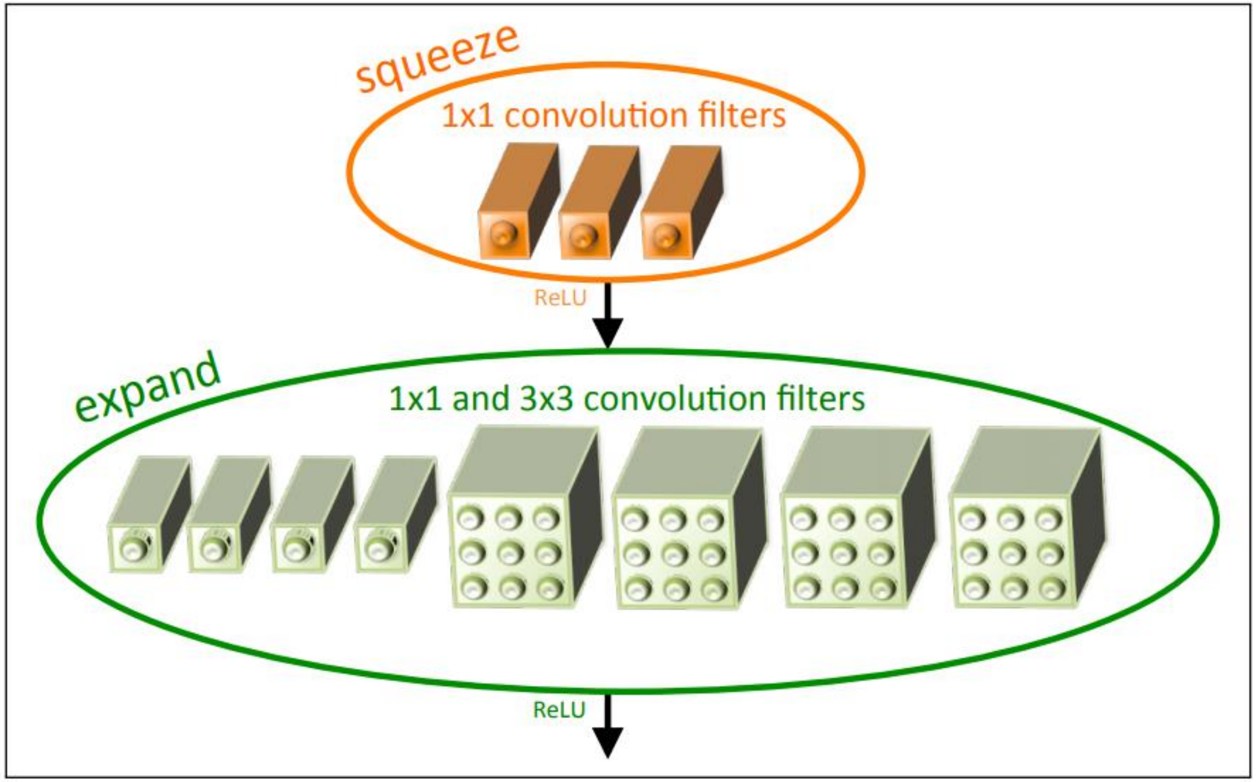


Figure 1: Microarchitectural view: Organization of convolution filters in the **Fire module**. In this example, $s_{1x1} = 3$, $e_{1x1} = 4$, and $e_{3x3} = 4$. We illustrate the convolution filters but not the activations.

A Fire module is comprised of: a squeeze convolution layer (which has only 1x1 filters), feeding into an expand layer that has a mix of 1x1 and 3x3 convolution filters

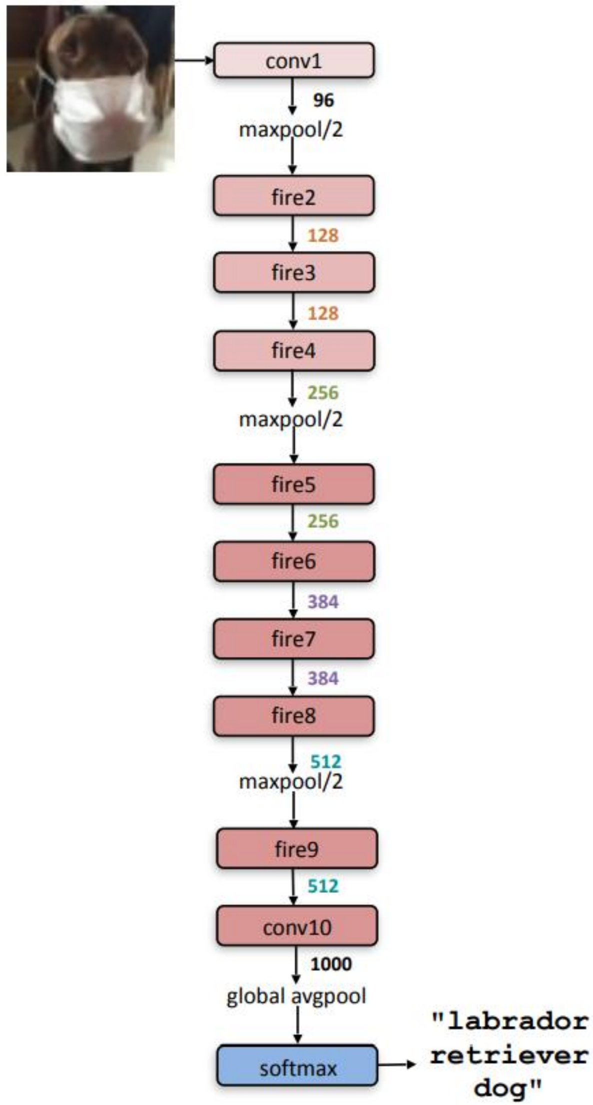


Figure 2: Macroarchitectural view of our SqueezeNet architecture. Left: SqueezeNet (Section 3.3); Middle: SqueezeNet with simple bypass (Section 6); Right: SqueezeNet with complex bypass

SqueezeNet Design Strategies

- ***Strategy 1.*** Replace 3x3 filters with 1x1 filters
  – Parameters per filter: (3x3 filter) = 9 * (1x1 filter)

- ***Strategy 2.*** Decrease the number of **input channels** to 3x3 filters
  – Total # of parameters: (# of input channels) * (# of filters) * (# of parameters per filter)

- ***Strategy 3.*** Down-sample late in the network so that convolution layers have large activation maps
  – Size of activation maps: the size of input data, the choice of layers in which to down-sample in the CNN architecture

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

Microarchitecture – **Fire Module**

- **Fire module** is consist of:
  - A *squeeze* convolution layer
    - full of s_{1x1} # of 1x1 filters
  - An *expand* layer
    - mixture of e_{1x1} # of 1x1 and e_{3x3} # of 3x3 filters

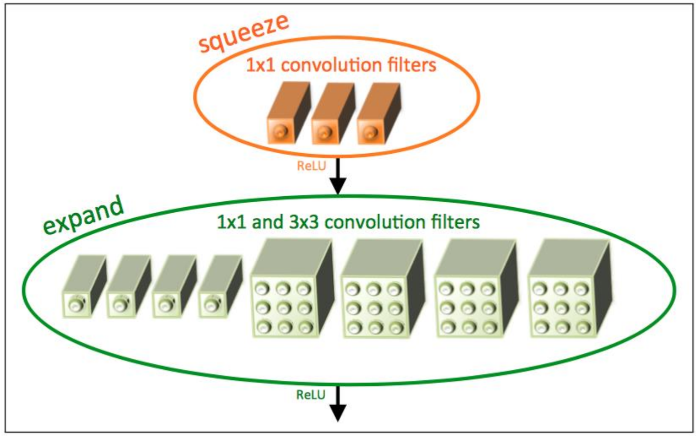


Figure 1: Microarchitectural view: Organization of convolution filters in the **Fire module**. In this example, $s_{1x1} = 3$, $e_{1x1} = 4$, and $e_{3x3} = 4$. We illustrate the convolution filters but not the activations.

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

Microarchitecture – Fire Module

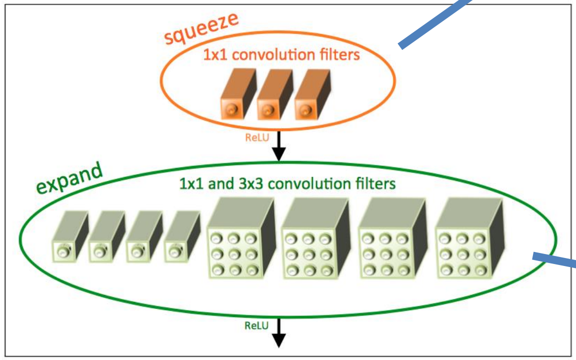


Figure 1: Microarchitectural view: Organization of convolution filters in the **Fire module**. In this example, $s_{1x1} = 3$, $e_{1x1} = 4$, and $e_{3x3} = 4$. We illustrate the convolution filters but not the activations.

Strategy 2. Decrease the number of input channels to 3x3 filters

Total # of parameters: (# of input channels) * (# of filters) * (# of parameters per filter)

Squeeze Layer

How much can we limit s_{1x1}?

Set $s_{1x1} < (e_{1x1} + e_{3x3})$,

limits the # of input channels to 3*3 filters

Strategy 1. Replace 3*3 filters with 1*1 filters

Parameters per filter: (3*3 filter) = 9 (1*1 filter)

How much can we replace 3*3 with 1*1?

(e_{1x1} vs e_{3x3})?

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

Parameters in Fire Module

The # of expanded filter(e_i)

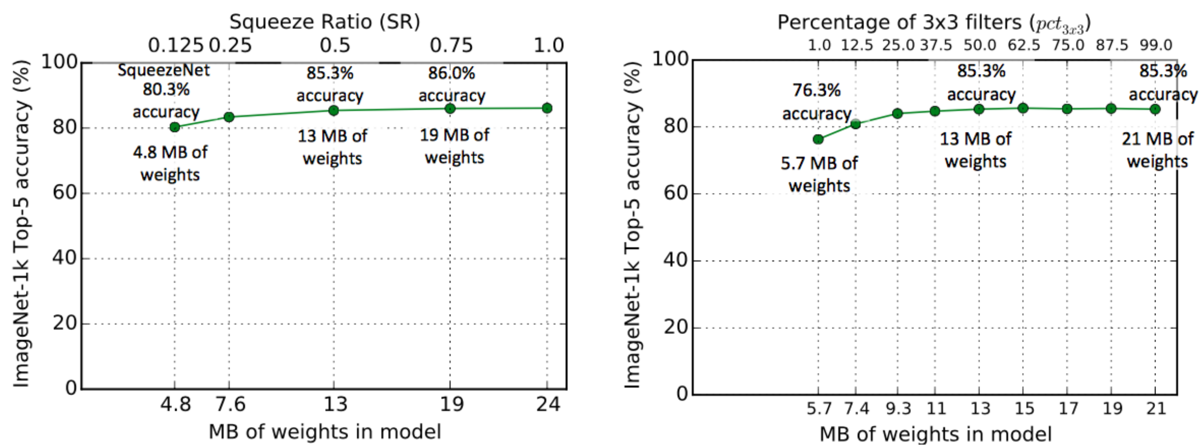
$$e_i = e_{i,1x1} + e_{i,3x3}$$

The % of 3x3 filter in expanded layer(pct_{3x3})

$$e_{i,3x3} = \boldsymbol{pct_{3x3}} * e_i$$

The Squeeze Ratio(SR)

$$s_{i,1x1} = \boldsymbol{SR} * e_i$$



(a) Exploring the impact of the squeeze ratio (SR) on model size and accuracy.

(b) Exploring the impact of the ratio of 3x3 filters in expand layers (pct_{3x3}) on model size and accuracy.

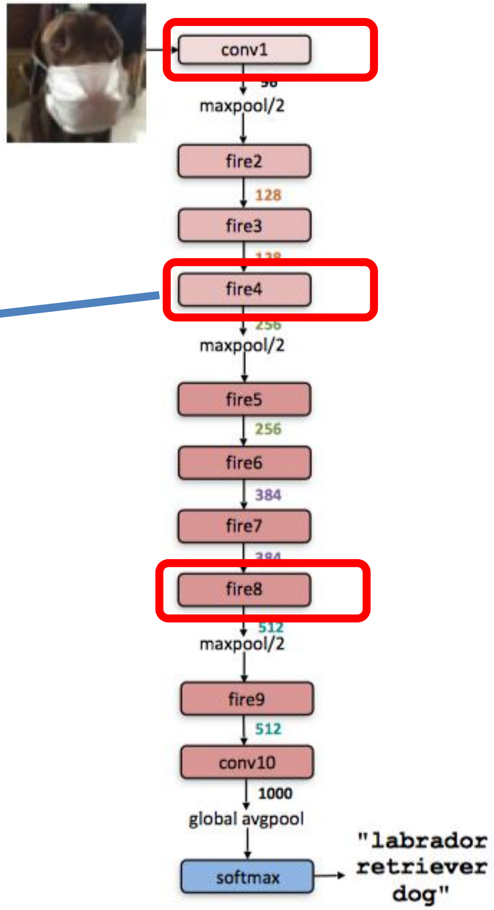
Figure 3: Microarchitectural design space exploration.

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

Macroarchitecture

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

Size of activation maps: the size of input data, the choice of layers in which to downsample in the CNN architecture



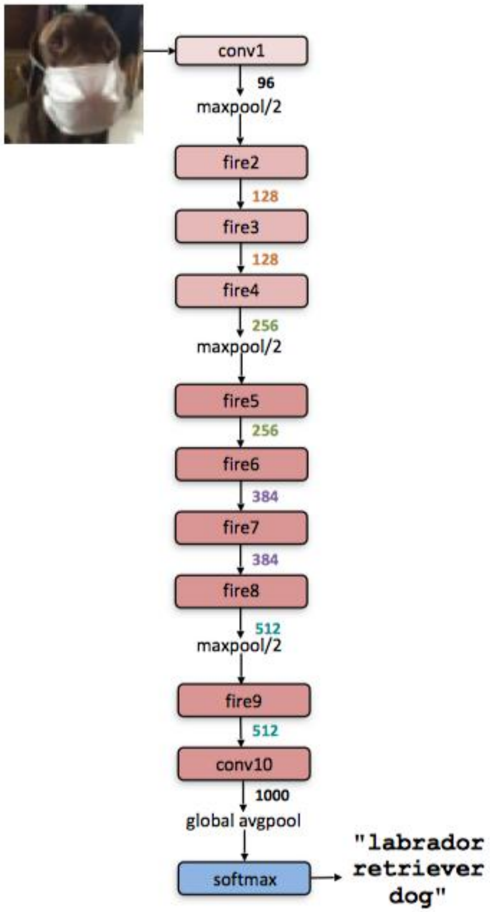
These relative late placements of pooling concentrates activation maps at later phase to **preserve higher accuracy**

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

Macroarchitecture

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



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

Summary of SqueezeNet

- **Comparing with AlexNet**: with close accuracy rate, the ratio of their parameter sizes is **1:50**
- If model compression is performed, comparing with AlexNet, the ratio of their parameter sizes is **1:510**
- **Three strategies**: (a) using 1x1filter to replace 3x3filter; (b) using squeeze layer to reduce the channels; © performing down-sampling late in the network to gain larger activation map; all these are implemented in **fire module**

MobileNet V1

What’s the “Right” Neural Network?

- Sufficiently high accuracy
- Low computational complexity
- Low energy usage
- Small model size

Three issues to define **size of neural networks**:

- **Kernel numbers** for convolution
- **Channel numbers** for image input or feature maps
- **Size of feature maps**

Related Work

- Quantization, pruning, decomposition and distillation
- Small network, Squeezenet, Xception network

Techniques for Small Deep Neural Networks

- Remove Fully-Connected Layers
- Kernel Reduction (3x3 -> 1x1)
- Channel Reduction
- Depthwise Separable Convolutions

Techniques for Small Deep Neural Networks

- Remove Fully-Connected Layers
- $\boxed{\text{Kernel Reduction (3x3 -> 1x1)}}$ ➡ **SqueezeNet**
- Channel Reduction
- Depthwise Separable Convolutions

Techniques for Small Deep Neural Networks

- Remove Fully-Connected Layers
- Kernel Reduction (3x3 -> 1x1) ➡ **SqueezeNet**
- Channel Reduction
- Depthwise Separable Convolutions

MobileNet V1

MobileNet v1

Key Idea : Depthwise Separable Convolution!

- The MobileNet model is based on **depthwise separable convolutions** which is a form of factorized convolutions which factorize a *standard convolution* into a *depthwise convolution* and a 1×1 convolution called a *pointwise convolution*.

Howard A G, Zhu M, Chen B, et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications[J]. arXiv preprint arXiv:1704.04861, 2017.

MobileNet v1

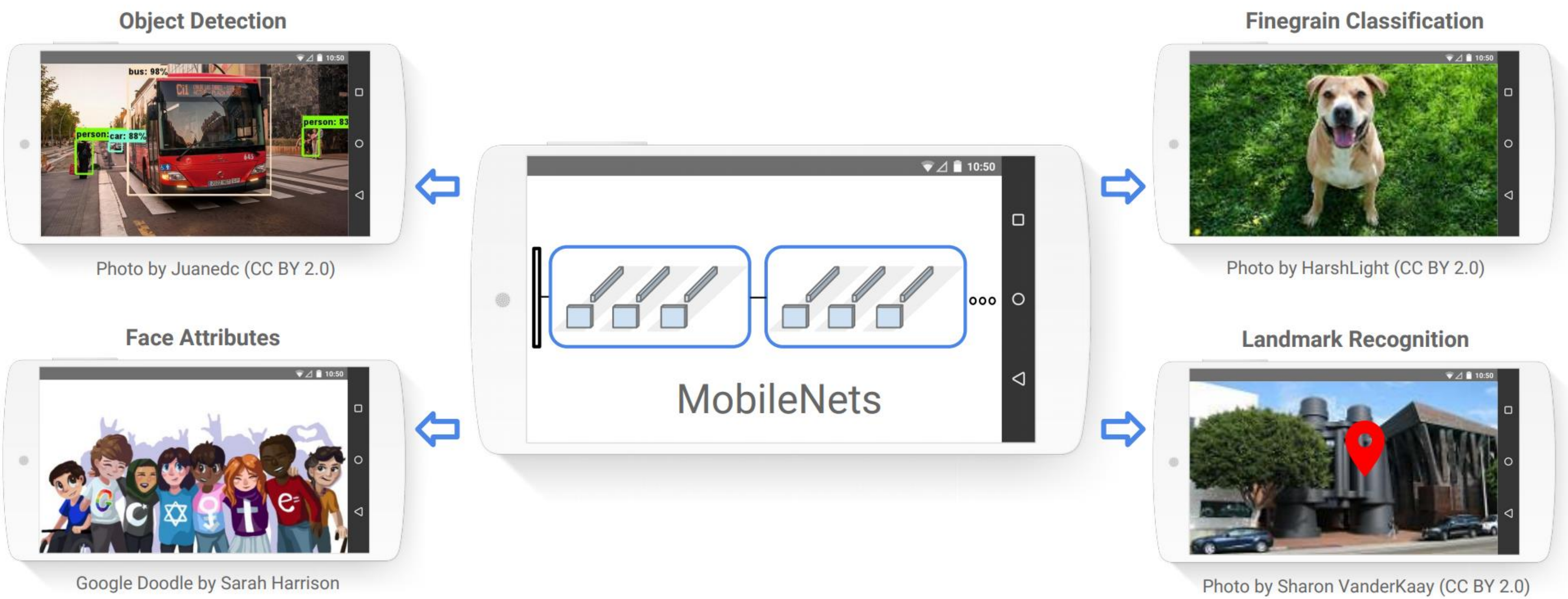
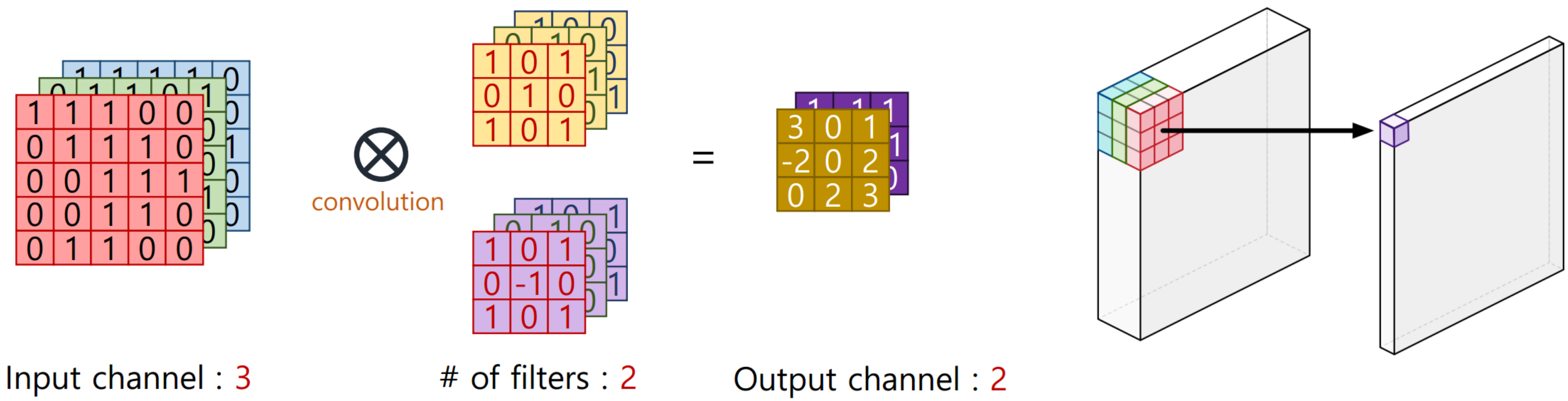


Figure 1. MobileNet models can be applied to various recognition tasks for efficient on device intelligence.

Howard A G, Zhu M, Chen B, et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications[J]. arXiv preprint arXiv:1704.04861, 2017.

MobileNet v1

Standard Convolution Operation

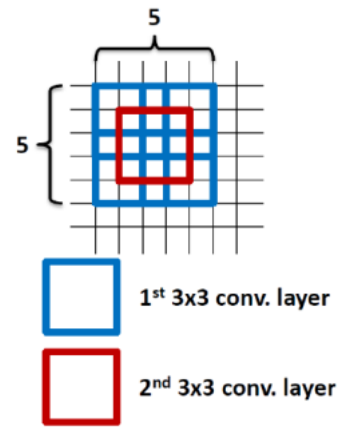


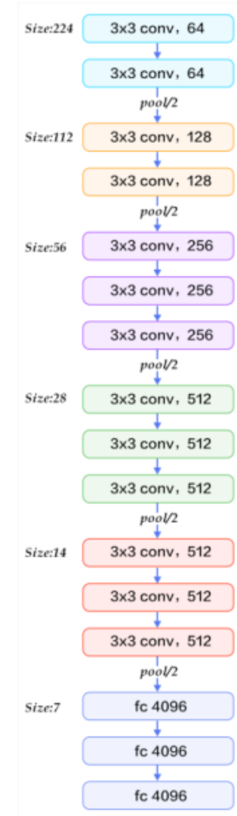
Howard A G, Zhu M, Chen B, et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications[J]. arXiv preprint arXiv:1704.04861, 2017.

MobileNet v1

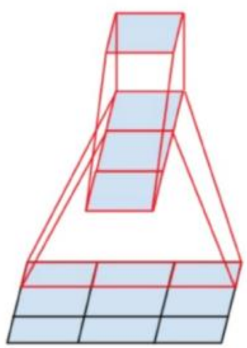
VGG, Inception-v3

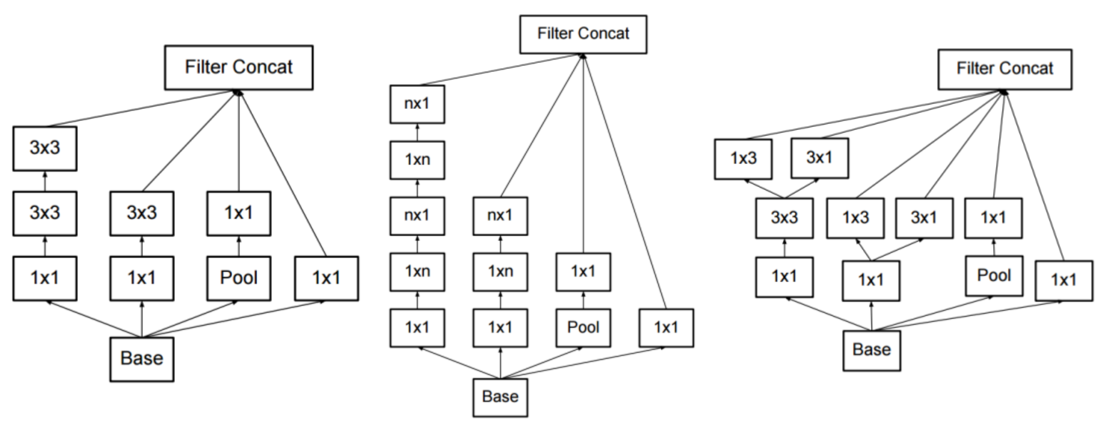
- **VGG** – use only 3x3 convolution
  - Stack of 3x3 conv layers has same effective receptive field as 5x5 or 7x7 conv layer
  - Deeper means more non-linearities
  - Fewer parameters: $2 \times (3 \times 3 \times C)$ vs $(5 \times 5 \times C)$





- **Inception-v3**
  - Factorization of filters





Howard A G, Zhu M, Chen B, et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications[J]. arXiv preprint arXiv:1704.04861, 2017.

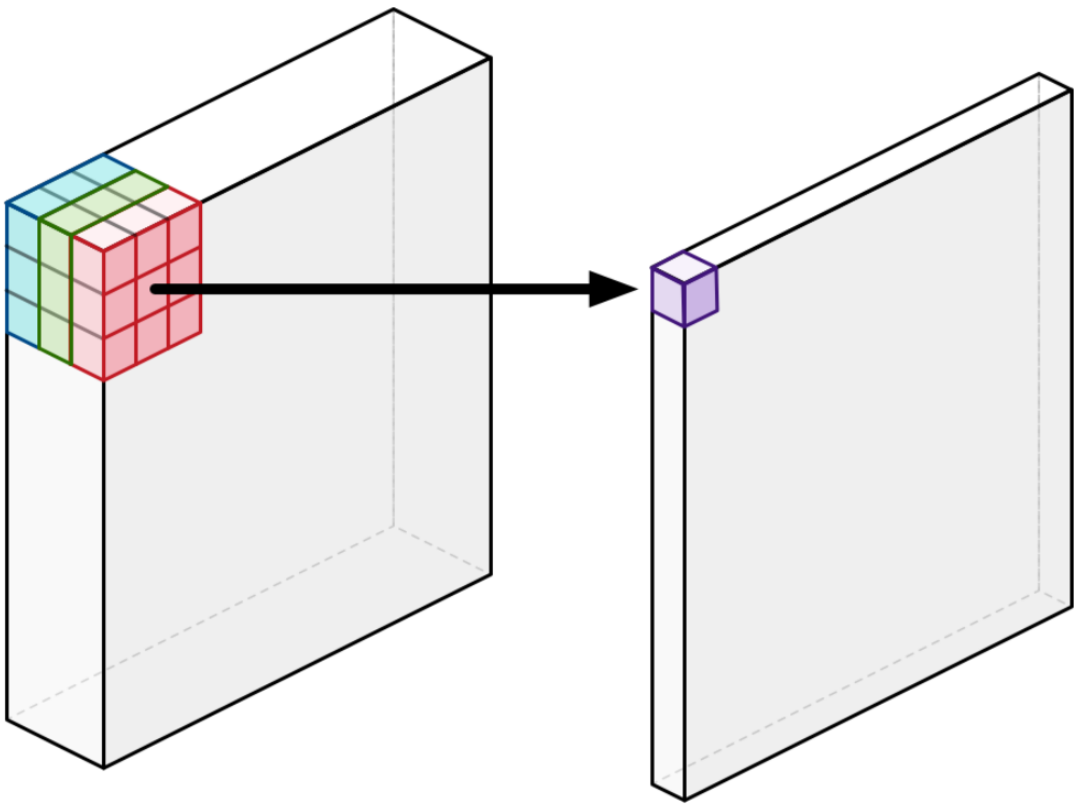
MobileNet v1

Why should we always consider **all channels**?

Howard A G, Zhu M, Chen B, et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications[J]. arXiv preprint arXiv:1704.04861, 2017.

MobileNet v1

Standard Convolution



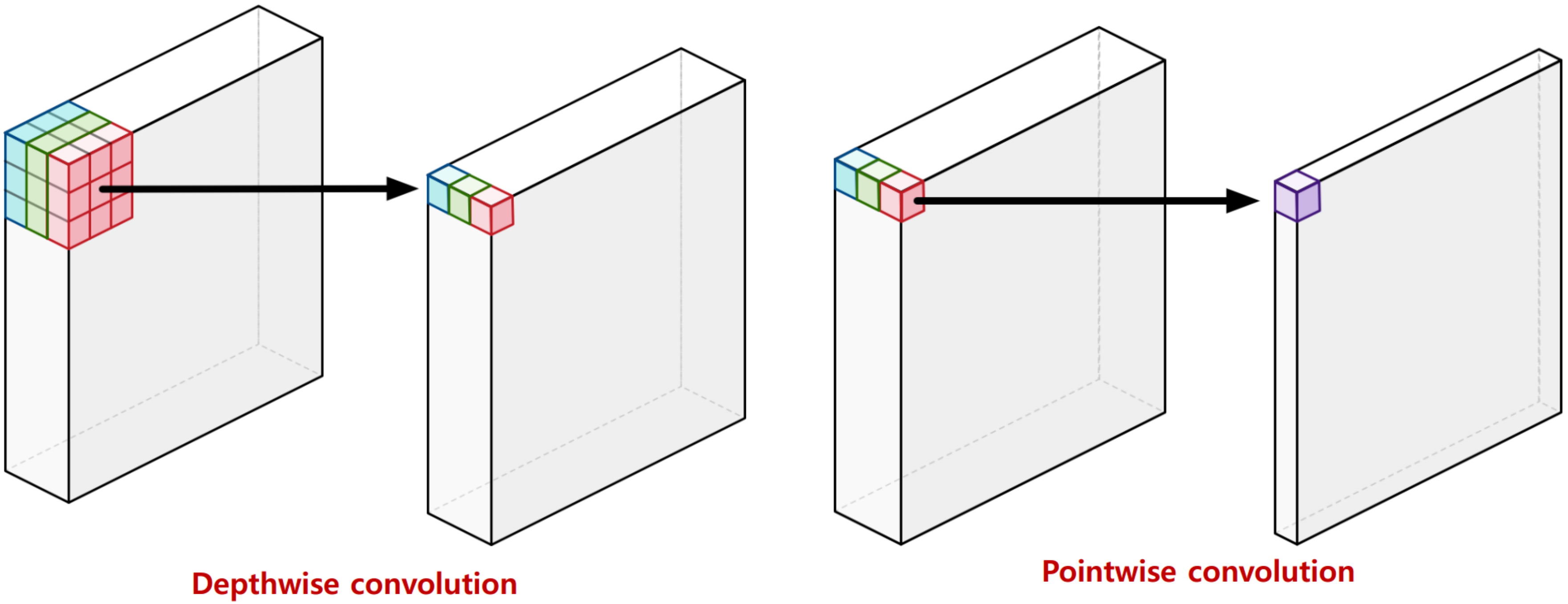
Depthwise convolution

Howard A G, Zhu M, Chen B, et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications[J]. arXiv preprint arXiv:1704.04861, 2017.

MobileNet v1

Depthwise Separable Convolution

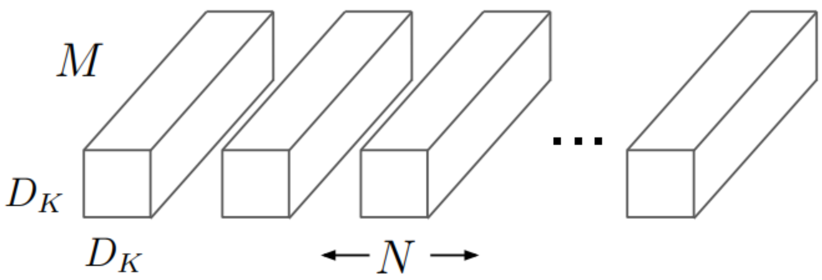
- Depthwise Convolution + Pointwise Convolution(1x1 convolution)



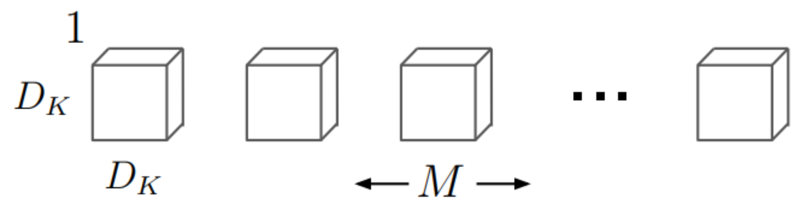
Howard A G, Zhu M, Chen B, et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications[J]. arXiv preprint arXiv:1704.04861, 2017.

MobileNet v1

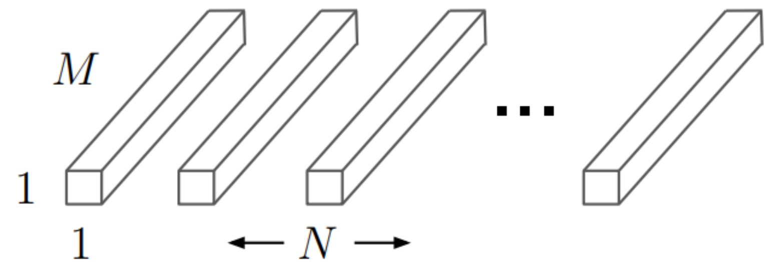
Standard Convolution vs Depthwise Separable Convolution



(a) Standard Convolution Filters



(b) Depthwise Convolutional Filters



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

Howard A G, Zhu M, Chen B, et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications[J]. arXiv preprint arXiv:1704.04861, 2017.

MobileNet v1

Standard Convolution vs Depthwise Separable Convolution

- Standard convolutions have the computational cost of
  - $D_K \times D_K \times \mathrm{M} \times \mathrm{N} \times D_F \times D_F$
- Depthwise separable convolutions cost
  - $D_K \times D_K \times \mathrm{M} \times D_F \times D_F + \mathrm{M} \times \mathrm{N} \times D_F \times D_F$
- Reduction in computations
  - $1/\mathrm{N} + 1/D_K^2$
  - If we use 3x3 depthwise separable convolutions, we get between 8 to 9 times

D_K: width/height of filters
D_F: width/height of feature maps
M : number of input channels
N : number of output channels(number of filters)

Howard A G, Zhu M, Chen B, et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications[J]. arXiv preprint arXiv:1704.04861, 2017.

MobileNet v1

Depthwise Separable Convolution

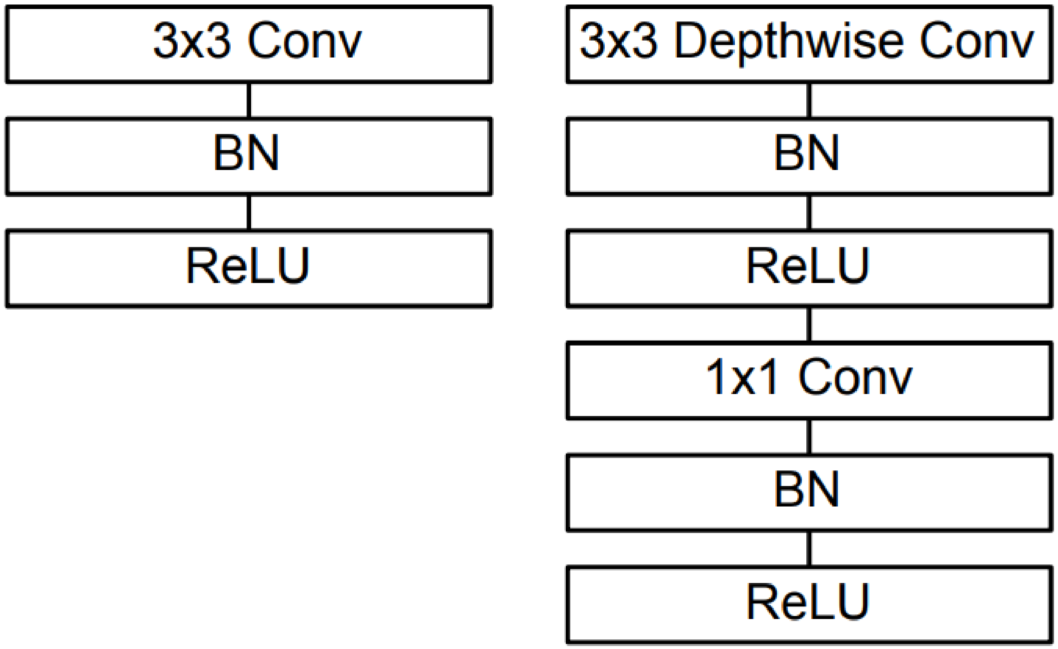


Figure 3. Left: Standard convolutional layer with batchnorm and ReLU. Right: Depthwise Separable convolutions with Depthwise and Pointwise layers followed by batchnorm and ReLU.

Howard A G, Zhu M, Chen B, et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications[J]. arXiv preprint arXiv:1704.04861, 2017.

MobileNet v1

Model Structure

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		$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$	Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
	Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2		$3 \times 3 \times 512$ 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	$1 \times 1 \times 1000$

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%

Howard A G, Zhu M, Chen B, et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications[J]. arXiv preprint arXiv:1704.04861, 2017.

MobileNet v1

Width Multiplier & Resolution Multiplier

- Width Multiplier – Thinner Models
  - For a given layer and width multiplier α, the number of input channels M becomes αM and the number of output channels N becomes αN – where α with typical settings of 1, 0.75, 0.6 and 0.25
- Resolution Multiplier – Reduced Representation
  - The second hyper-parameter to reduce the computational cost of a neural network is a resolution multiplier ρ
  - $0<\rho\leq 1$, which is typically set of implicitly so that input resolution of network is 224, 192, 160 or 128(ρ = 1, 0.857, 0.714, 0.571)
- Computational cost:
  - $D_K \times D_K \times \alpha M \times \rho D_F \times \rho D_F + \alpha M \times \alpha N \times \rho D_F \times \rho D_F$

Howard A G, Zhu M, Chen B, et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications[J]. arXiv preprint arXiv:1704.04861, 2017.

MobileNet v1

Width Multiplier & Resolution Multiplier

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

Howard A G, Zhu M, Chen B, et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications[J]. arXiv preprint arXiv:1704.04861, 2017.

MobileNet v1

Experiments – Model Choices

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

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

Howard A G, Zhu M, Chen B, et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications[J]. arXiv preprint arXiv:1704.04861, 2017.

MobileNet v1

Experiments – Results

Table 8. MobileNet Comparison to Popular Models

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

Table 9. Smaller MobileNet Comparison to Popular Models

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
0.50 MobileNet-160	60.2%	76	1.32
Squeezenet	57.5%	1700	1.25
AlexNet	57.2%	720	60

Table 13. COCO object detection results comparison using different frameworks and network architectures. mAP is reported with COCO primary challenge metric (AP at IoU=0.50:0.05:0.95)

Framework Resolution	Model	mAP	Billion Mult-Adds	Million Parameters
SSD 300	deeplab-VGG	21.1%	34.9	33.1
	Inception V2	22.0%	3.8	13.7
	MobileNet	19.3%	1.2	6.8
Faster-RCNN 300	VGG	22.9%	64.3	138.5
	Inception V2	15.4%	118.2	13.3
	MobileNet	16.4%	25.2	6.1
Faster-RCNN 600	VGG	25.7%	149.6	138.5
	Inception V2	21.9%	129.6	13.3
	Mobilenet	19.8%	30.5	6.1

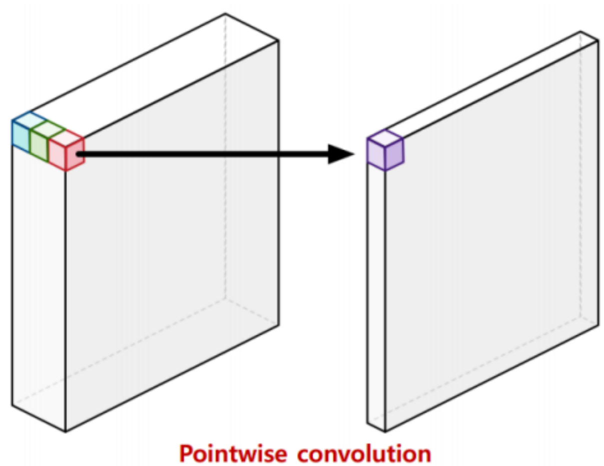
Figure 6. Example objection detection results using MobileNet SSD.

Howard A G, Zhu M, Chen B, et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications[J]. arXiv preprint arXiv:1704.04861, 2017.

MobileNet V2

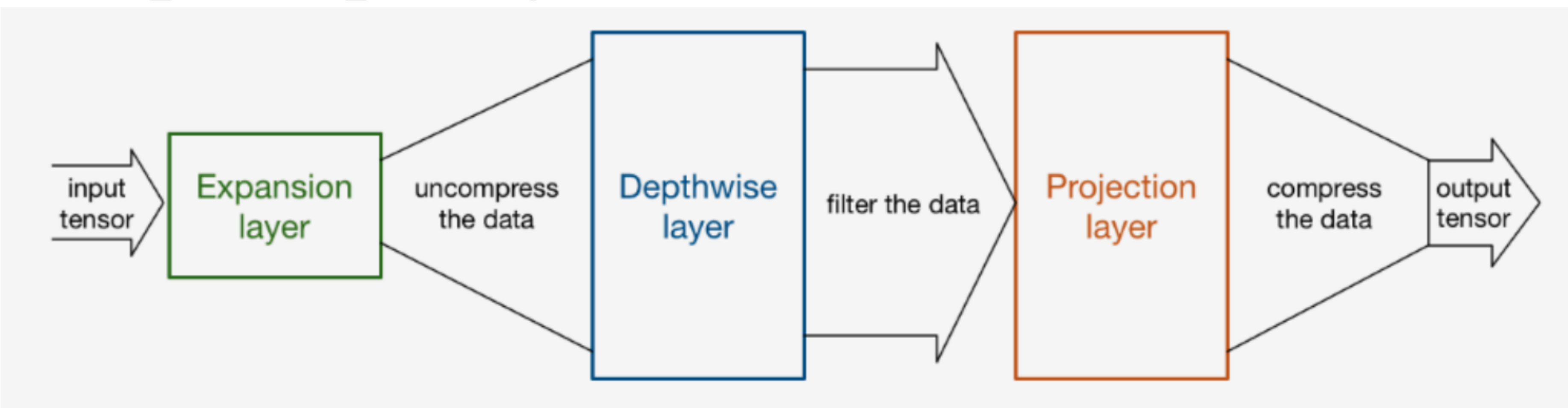
Motivation

- Projects data with a high dimensions (channels) into a tensor with a much low dimensions.
- For example, the depthwise layer may work on a tensor with 144 channels, which the projection layer will then shrink down to only 24 channels.



Main idea

- This module takes as an input a low dim compressed representation which is first expanded to high dim and filtered with a lightweight depthwise convolution.



Key ideas

- ***Strategy 1*. Linear Bottleneck**
  - using depthwise separable convolution as efficient building blocks. However, V2 introduces two new features to the architecture: 1) **linear bottlenecks between the layers**

- ***Strategy 2*. Inverted Residual Blocks**
  - shortcut connections between the bottlenecks

Sandler M, Howard A, Zhu M, et al. Mobilenetv2: Inverted residuals and linear bottlenecks[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 4510-4520.

Linear Bottleneck

- Informally, for an input set of real images, we say that the set of layer activations forms a “manifold of interest” .

- It has been long assumed that manifolds of interest in neural networks could be embedded in low-dimensional subspaces.

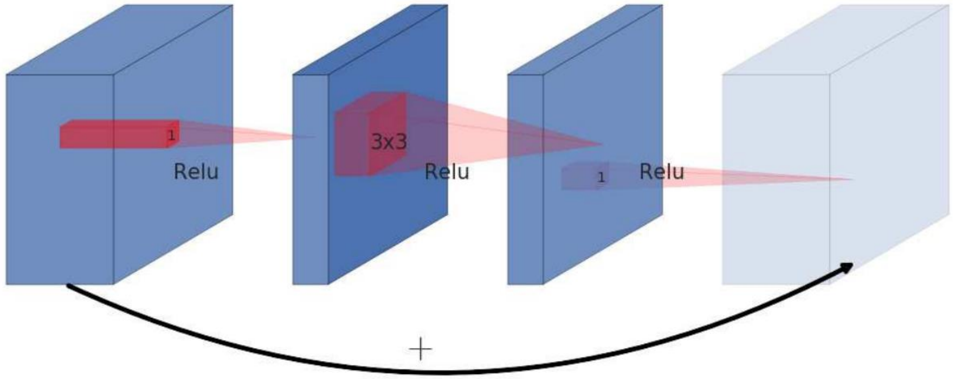
Sandler M, Howard A, Zhu M, et al. Mobilenetv2: Inverted residuals and linear bottlenecks[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 4510-4520.

MobileNet v2

Residual Blocks

- Residual blocks connect the beginning and end of a convolutional block with a shortcut connection. By adding these two states the network has the opportunity of accessing earlier activations that weren’t modified in the convolutional block.

- Wide → narrow(bottleneck) → wide approach

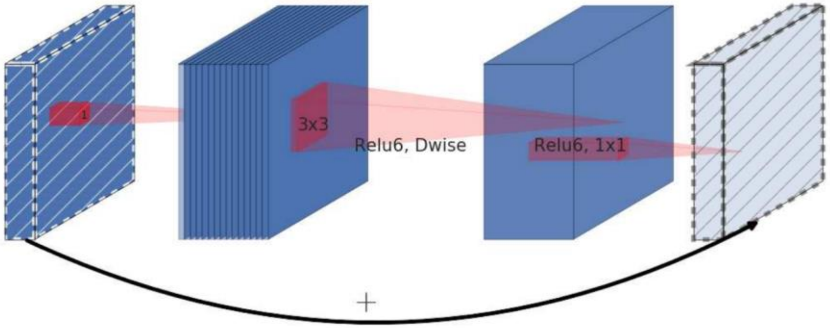


Sandler M, Howard A, Zhu M, et al. Mobilenetv2: Inverted residuals and linear bottlenecks[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 4510-4520.

MobileNet v2

Inverted Residuals

- Inspired by the intuition that the bottlenecks actually contain all the necessary information, while an expansion layer acts merely as an implementation detail that accompanies a non-linear transformation of the tensor, the authors use shortcuts directly between the bottlenecks.
- narrow → wide → narrow approach

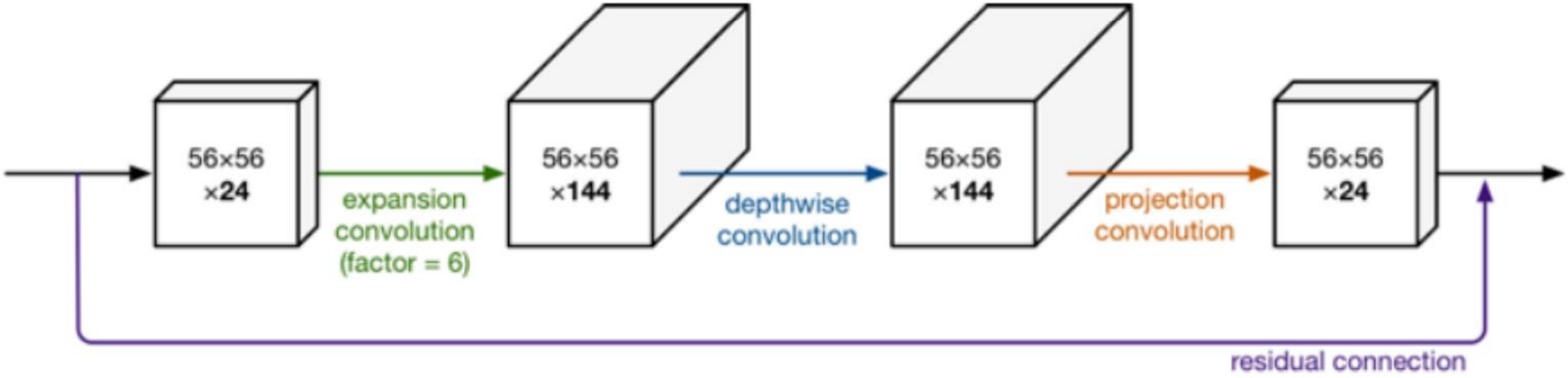


Sandler M, Howard A, Zhu M, et al. Mobilenetv2: Inverted residuals and linear bottlenecks[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 4510-4520.

MobileNet v2

Information Flow Interpretation

- The proposed convolutional block has a unique property that allows to separate the network expressiveness (encoded by expansion layers) from its capacity (encoded by bottleneck inputs).



Sandler M, Howard A, Zhu M, et al. Mobilenetv2: Inverted residuals and linear bottlenecks[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 4510-4520.

The Architecture of MobileNetV2

- The architecture of MobileNetV2 contains the initial fully convolution layer with 32 filters, followed by 19 residual bottleneck layers described in the Table 2.

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 2: MobileNetV2 : Each line describes a sequence of 1 or more identical (modulo stride) layers, repeated n times. All layers in the same sequence have the same number c of output channels. The first layer of each sequence has a stride s and all others use stride 1. All spatial convolutions use 3×3 kernels. The expansion factor t is always applied to the input size as described in Table 1.

Sandler M, Howard A, Zhu M, et al. Mobilenetv2: Inverted residuals and linear bottlenecks[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 4510-4520.

MobileNet v2

Memory Efficient Inference

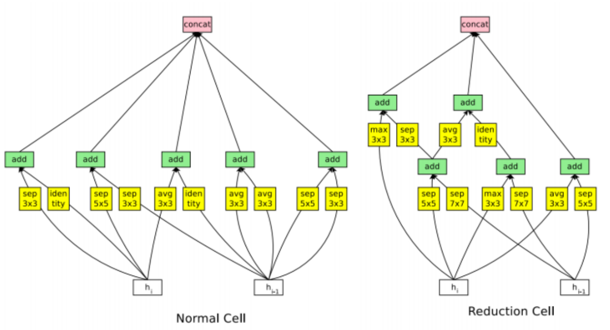
- The amount of memory is simply the maximum total size of combined inputs and outputs across all operations.

- If we treat a bottleneck residual block as a single operation (and treat inner convolution as a disposable tensor), the total amount of memory would be dominated by the size of bottleneck tensors, rather than the size of tensors that are internal to bottleneck (and much larger)

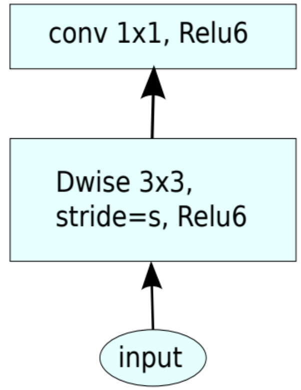
Sandler M, Howard A, Zhu M, et al. Mobilenetv2: Inverted residuals and linear bottlenecks[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 4510-4520.

MobileNet v2

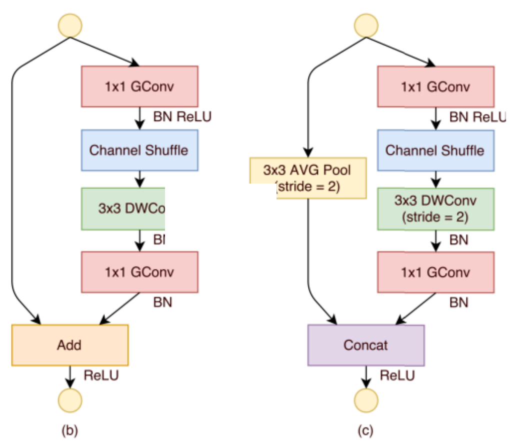
Comparison of Convolutional Blocks for Different Architectures



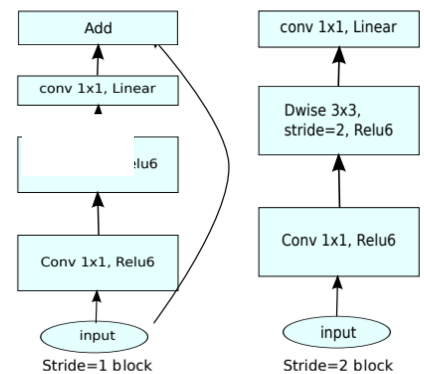
(a) NasNet[23]



(b) MobileNet[27]



(c) ShuffleNet [20]



(d) Mobilenet V2

Sandler M, Howard A, Zhu M, et al. Mobilenetv2: Inverted residuals and linear bottlenecks[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 4510-4520.

MobileNet v2

The Max Number of Channels/Memory(in Kb)

Size	MobileNetV1	MobileNetV2	ShuffleNet (2x,g=3)
112x112	1/O(1)	1/O(1)	1/O(1)
56x56	128/800	32/200	48/300
28x28	256/400	64/100	400/600K
14x14	512/200	160/62	800/310
7x7	1024/199	320/32	1600/156
1x1	1024/2	1280/2	1600/3
max	800K	**200K**	600K

Sandler M, Howard A, Zhu M, et al. Mobilenetv2: Inverted residuals and linear bottlenecks[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 4510-4520.

MobileNet v2

ImageNet Classification Results

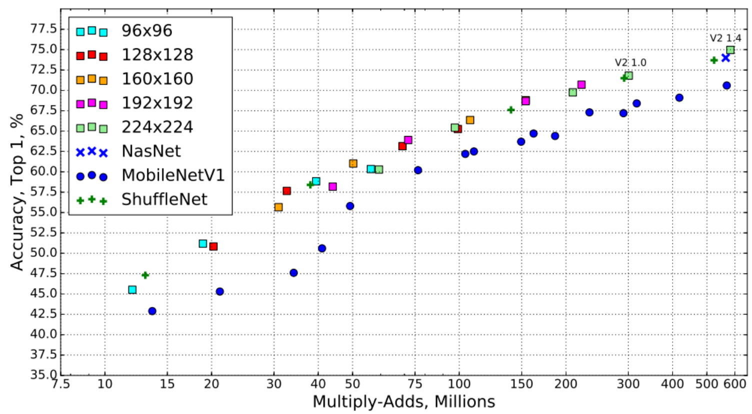


Figure 5: Performance curve of MobileNetV2 vs MobileNetV1, ShuffleNet, NAS. For our networks we use multipliers 0.35, 0.5, 0.75, 1.0 for all resolutions, and additional 1.4 for for 224. Best viewed in color.

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 4: Performance on ImageNet, comparison for different networks. As is common practice for ops, we count the total number of Multiply-Adds. In the last column we report running time in milliseconds (ms) for a single large core of the Google Pixel 1 phone (using TF-Lite). We do not report ShuffleNet numbers as efficient group convolutions and shuffling are not yet supported.

Sandler M, Howard A, Zhu M, et al. Mobilenetv2: Inverted residuals and linear bottlenecks[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 4510-4520.

MobileNet v2

MobileNet v1 ***Vs*** MobileNet v2

Version	MACs (millions)	Parameters (millions)
MobileNet V1	569	4.24
MobileNet V2	300	3.47

Version	iPhone 7	iPhone X	iPad Pro 10.5
MobileNet V1	118	162	204
MobileNet V2	145	233	220

Version	Top-1 Accuracy	Top-5 Accuracy
MobileNet V1	70.9	89.9
MobileNet V2	71.8	91.0

Sandler M, Howard A, Zhu M, et al. Mobilenetv2: Inverted residuals and linear bottlenecks[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 4510-4520.

MobileNet v2

Object Detection & Semantic Segmentation Results

Network	mAP	Params	MAdd	CPU
SSD300	23.2	36.1M	35.2B	-
SSD512	26.8	36.1M	99.5B	-
YOLOv2	21.6	50.7M	17.5B	-
MNet V1 + SSDLite	22.2	5.1M	1.3B	270ms
MNet V2 + SSDLite	22.1	**4.3M**	**0.8B**	200ms

Table 6: Performance comparison of MobileNetV2 + SSDLite and other realtime detectors on the COCO dataset object detection task.

Network	OS	ASPP	MF	mIOU	Params	MAdds
MNet V1	16	✓		75.29	11.15M	14.25B
	8	✓	✓	78.56	11.15M	941.9B
MNet V2*	16	✓		75.70	4.52M	5.8B
	8	✓	✓	78.42	4.52M	387B
MNet V2*	16			**75.32**	**2.11M**	**2.75B**
	8		✓	77.33	2.11M	152.6B
ResNet-101	16	✓		80.49	58.16M	81.0B
	8	✓	✓	82.70	58.16M	4870.6B

Table 7: MobileNet + DeepLabv3 inference strategy on the PASCAL VOC 2012 *validation* set.

Object Detection

Semantic Segmentation

Sandler M, Howard A, Zhu M, et al. Mobilenetv2: Inverted residuals and linear bottlenecks[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 4510-4520.

ShuffleNet

Three issues to define **size of neural networks**:

- **Kernel numbers** for convolution
- **Channel numbers** for image input or feature maps
- **Size of feature maps**

Techniques for Small Deep Neural Networks

- Remove Fully-Connected Layers
- Kernel Reduction (3x3 -> 1x1) → **SqueezeNet**
- Channel Reduction
- Depthwise Separable Convolutions

MobileNet V1

Techniques for Small Deep Neural Networks

- Remove Fully-Connected Layers
- Kernel Reduction (3x3 -> 1x1) → **SqueezeNet**
- Channel Reduction
- Depthwise Separable Convolutions

→ **ShuffleNet**

MobileNet V1

ShuffleNet

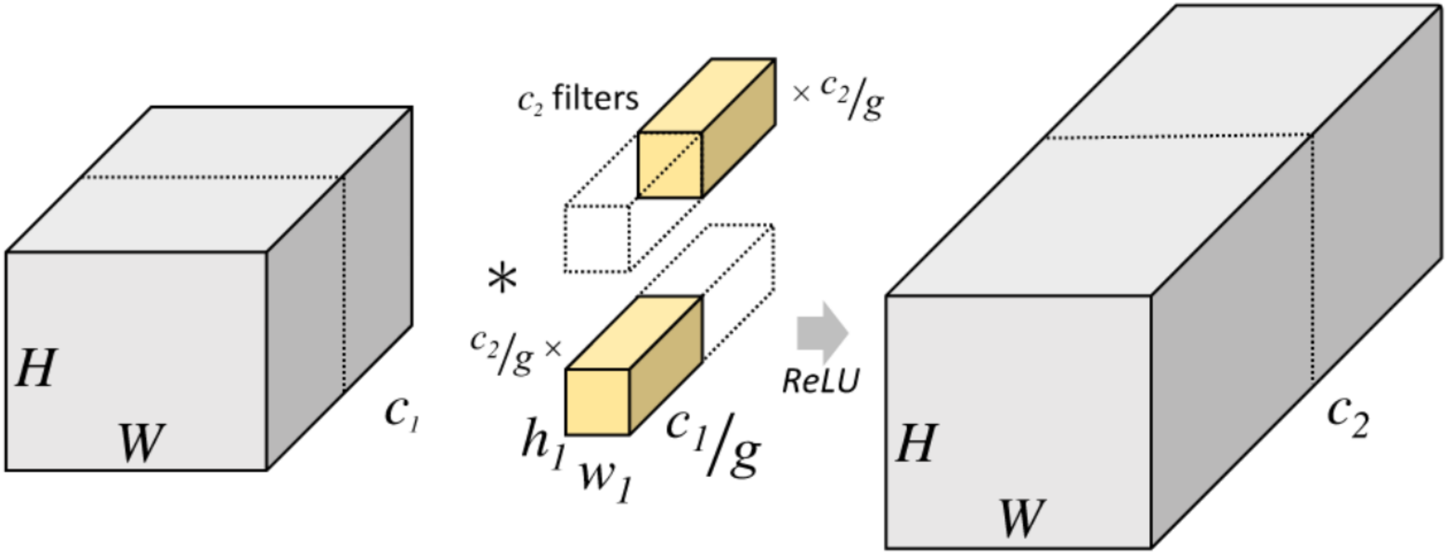
Key ideas

- ***Strategy 1.*** Use **depthwise separable convolution**
- ***Strategy 2.*** **Grouped convolution** on 1x1 convolution layers – pointwise group convolution
- ***Strategy 3.*** **Channel shuffle operation** after pointwise group convolution

Zhang X, Zhou X, Lin M, et al. Shufflenet: An extremely efficient convolutional neural network for mobile devices[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 6848-6856.

ShuffleNet

Grouped Convolution

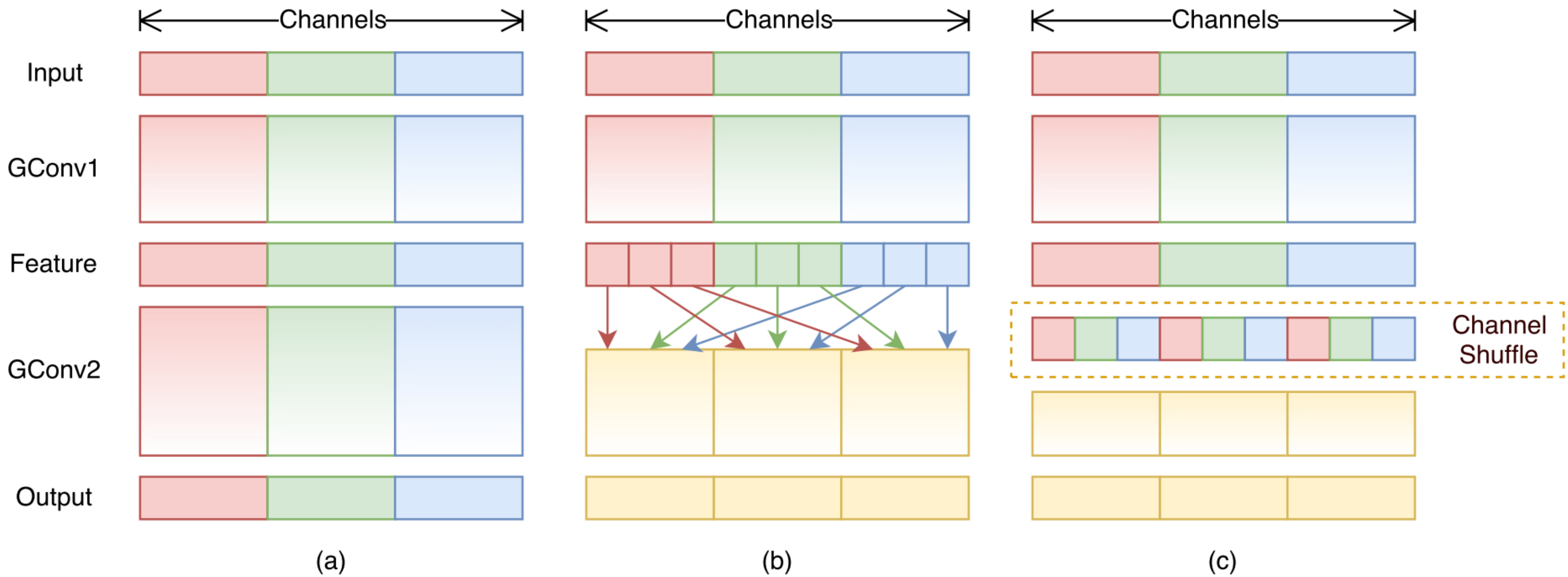


A convolutional layer with 2 filter groups. Note that each of the filters in the grouped convolutional layer is now exactly half the depth, i.e. half the parameters and half the compute as the original filter.

Zhang X, Zhou X, Lin M, et al. Shufflenet: An extremely efficient convolutional neural network for mobile devices[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 6848-6856.

ShuffleNet

1x1 Grouped Convolution with Channel Shuffling



- If multiple group convolutions stack together, there is one side effect(a)
  - Outputs from a certain channel are only derived from a small fraction of input channels
- If we allow group convolution to obtain input data from different groups, the input and outputchannels will be fully related

Zhang X, Zhou X, Lin M, et al. Shufflenet: An extremely efficient convolutional neural network for mobile devices[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 6848-6856.

Channel Shuffle Operation

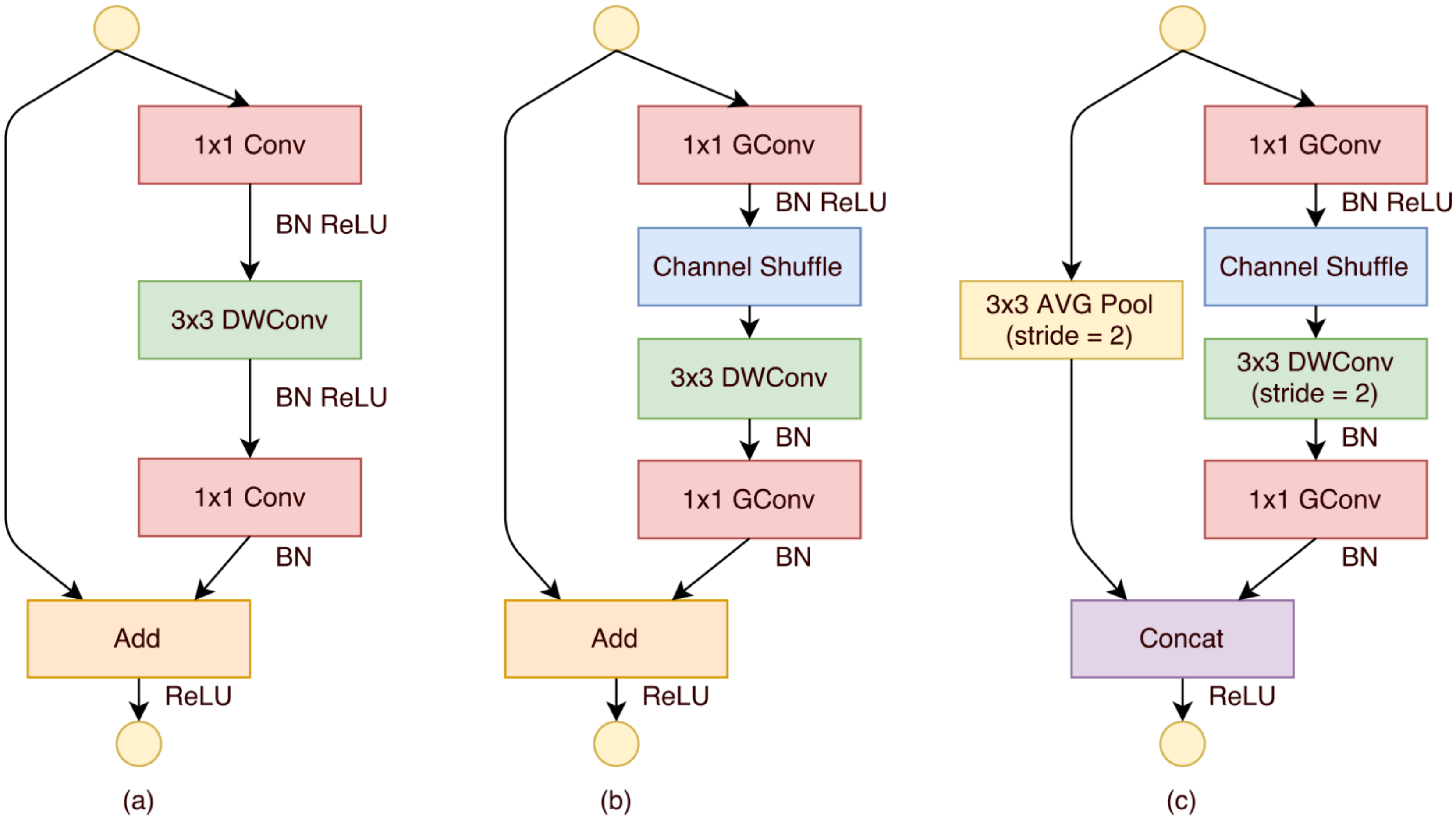
- Suppose a convolutional layer with g groups whose output has g×n channels; we first reshape the output channel dimension into (g, n), transposing and then flattening it back as the input of next layer.

- Channel shuffle operation is also differentiable

Zhang X, Zhou X, Lin M, et al. Shufflenet: An extremely efficient convolutional neural network for mobile devices[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 6848-6856.

ShuffleNet

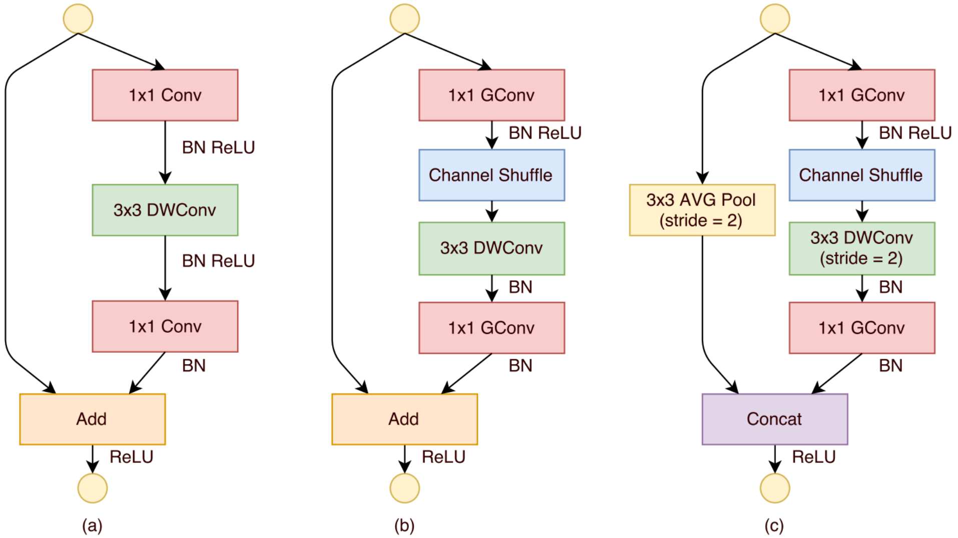
ShuffleNet Units



Zhang X, Zhou X, Lin M, et al. Shufflenet: An extremely efficient convolutional neural network for mobile devices[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 6848-6856.

ShuffleNet

ShuffleNet Units

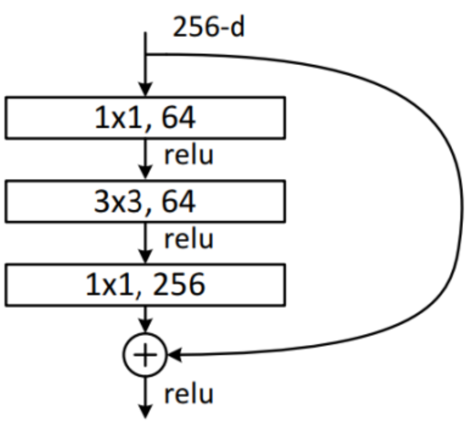


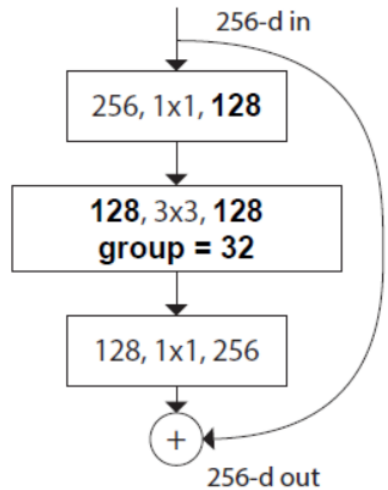
- From (a), replace the first 1x1 layer with pointwise group convolution followed by a channel shuffle operation
- ReLU is not applied to 3x3 DWConv
- As for the case where ShuffleNet is applied with stride, simply make to modifications
  - Add 3x3 average pooling on the shortcut path
  - Replace element-wise addition with channel concatenation to enlarge channel dimension with little extra computation

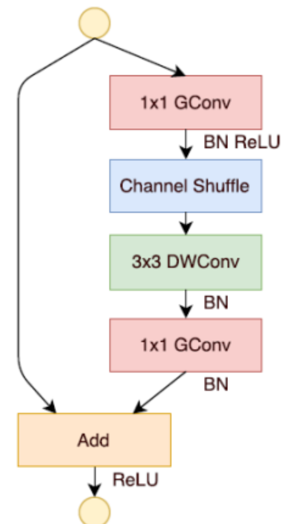
Zhang X, Zhou X, Lin M, et al. Shufflenet: An extremely efficient convolutional neural network for mobile devices[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 6848-6856.

ShuffleNet
Complexity

- For example, given the input size c×h×w and the bottleneck channel m







ResNet	ResNeXt	ShuffleNet
$hw(2cm + 9m^2)$	$hw(2cm + 9m^2/g)$	$hw(2cm/g + 9m)$

<Number of Operations>

Zhang X, Zhou X, Lin M, et al. Shufflenet: An extremely efficient convolutional neural network for mobile devices[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 6848-6856.

ShuffleNet

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

Table 1. ShuffleNet architecture. The complexity is evaluated with FLOPs, i.e. the number of floating-point multiplication-adds. Note that for Stage 2, we do not apply group convolution on the first pointwise layer because the number of input channels is relatively small.

Zhang X, Zhou X, Lin M, et al. Shufflenet: An extremely efficient convolutional neural network for mobile devices[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 6848-6856.

ShuffleNet

Experimental Results

- It is clear that channel shuffle consistently boosts classification scores for different settings

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

Table 3. ShuffleNet with/without channel shuffle (*smaller number represents better performance*)

Zhang X, Zhou X, Lin M, et al. Shufflenet: An extremely efficient convolutional neural network for mobile devices[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 6848-6856.

ShuffleNet

Experimental Results

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** ($0.25\times$, $g = 8$)

Table 4. Classification error vs. various structures (%, *smaller number represents better performance*). We do not report VGG-like structure on smaller networks because the accuracy is significantly worse.

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

Table 5. ShuffleNet vs. MobileNet [12] on ImageNet Classification

Zhang X, Zhou X, Lin M, et al. Shufflenet: An extremely efficient convolutional neural network for mobile devices[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 6848-6856.

ShuffleNet

Experimental Results

- Results show that with similar accuracy ShuffleNet is much more efficient than others

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

Table 6. Complexity comparison. **Implemented by BVLC (https://github.com/BVLC/caffe/tree/master/models/bvlc_googlenet)*

Zhang X, Zhou X, Lin M, et al. Shufflenet: An extremely efficient convolutional neural network for mobile devices[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 6848-6856.

ShuffleNet

Experimental Results

- Due to memory access and other overheads, every 4x theoretical complexity reduction usually result in ~2.6x actual speedup. Compared with AlexNet achieving ~13x actual speedup(the theoretical speedup is 18x)

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%

Table 7. Object detection results on MS COCO (*larger numbers represents better performance*). For MobileNets we compare two results: 1) COCO detection scores reported by [12]; 2) finetuning from our reimplemented MobileNets, whose training and finetuning settings are exactly the same as that for ShuffleNets.

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

Table 8. Actual inference time on mobile device (*smaller number represents better performance*). The platform is based on a single Qualcomm Snapdragon 820 processor. All results are evaluated with **single thread**.

Zhang X, Zhou X, Lin M, et al. Shufflenet: An extremely efficient convolutional neural network for mobile devices[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 6848-6856.

Except Designing Light-Weight Networks, Other Approaches to enable Mobile Applications include:

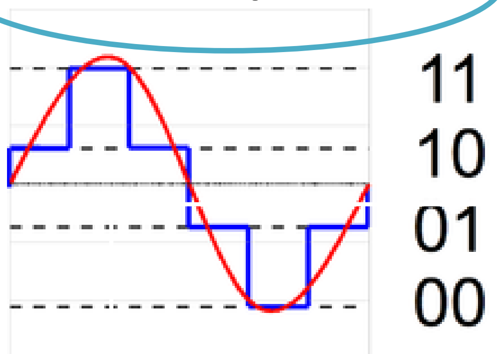
a. **Network Compression such as singular value decomposition (SVD), network pruning, quantization, binarization, …..**

b. **Knowledge Distillation from heavy large teacher network to light-weight small student network**

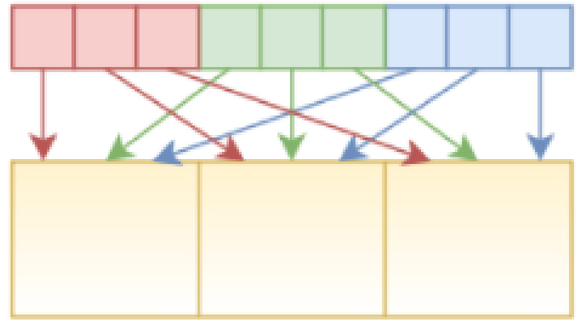
c. **Automatic Network Searching**

Techniques for Creating Fast & Energy-Efficient DNNs

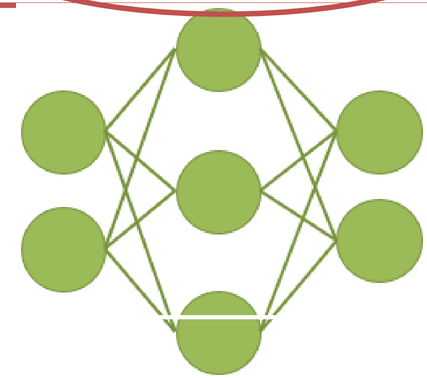
Model Compression



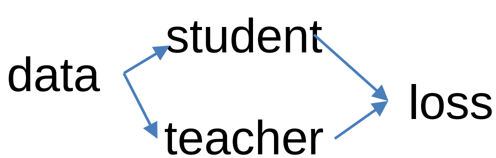
New Layer Types



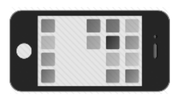
Original Net Design



Knowledge Distillation



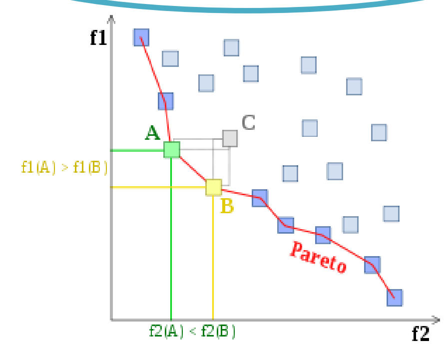
Efficient Implementation



cuDNN

INTEL® MKL-DNN

Design Space Exploration



Automatic Network Search

Summary

Model	
SqueezeNet	1×1 filters , input channels , Down-sample late in network
MobileNet v1	Pruning ,Weight Sharing , Groupwise Conv.
MobileNet v2	Depthwise conv + Pconv, Width Multiplier, Resolution Multiple
ShuffleNet	Depthwise convolution with Channel Shuffle

Summary

Model	Caffe	Tensorflow	Keras	PyTorch	Migration network	Recommendation level
SqueezeNet	1★	4★	2★	2★	AlexNet, faster-rcnn	3★
MobileNet v1	4★	5★	3★	3★	Mobilenet-SSD, MXNet, faster-rcnn	3★
MobileNet v2	3★	5★	2★	4★	MobileNetv2-SSDLite	5★
ShuffleNet	5★	4★	2★	5★	Shufflenet-SSD	4★

Tips: according to the number of people used

References

- Iandola F N, Han S, Moskewicz M W, et al. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size[J]. arXiv preprint arXiv:1602.07360, 2016.
- Howard A G, Zhu M, Chen B, et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications[J]. arXiv preprint arXiv:1704.04861, 2017.
- Sandler M, Howard A, Zhu M, et al. Mobilenetv2: Inverted residuals and linear bottlenecks[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 4510-4520.
- Zhang X, Zhou X, Lin M, et al. Shufflenet: An extremely efficient convolutional neural network for mobile devices[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 6848-6856.
- Han S, Mao H, Dally W J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding[J]. arXiv preprint arXiv:1510.00149, 2015.
- Chollet F. Xception: Deep learning with depthwise separable convolutions[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 1251-1258.
- http://slazebni.cs.illinois.edu/spring17/lec06_compression.pdf
- https://www.slideshare.net/JinwonLee9/mobilenet-pr044
- https://www.slideshare.net/JinwonLee9/pr108-mobilenetv2-inverted-residuals-and-linear-bottlenecks
- https://www.slideshare.net/JinwonLee9/shufflenet-pr054