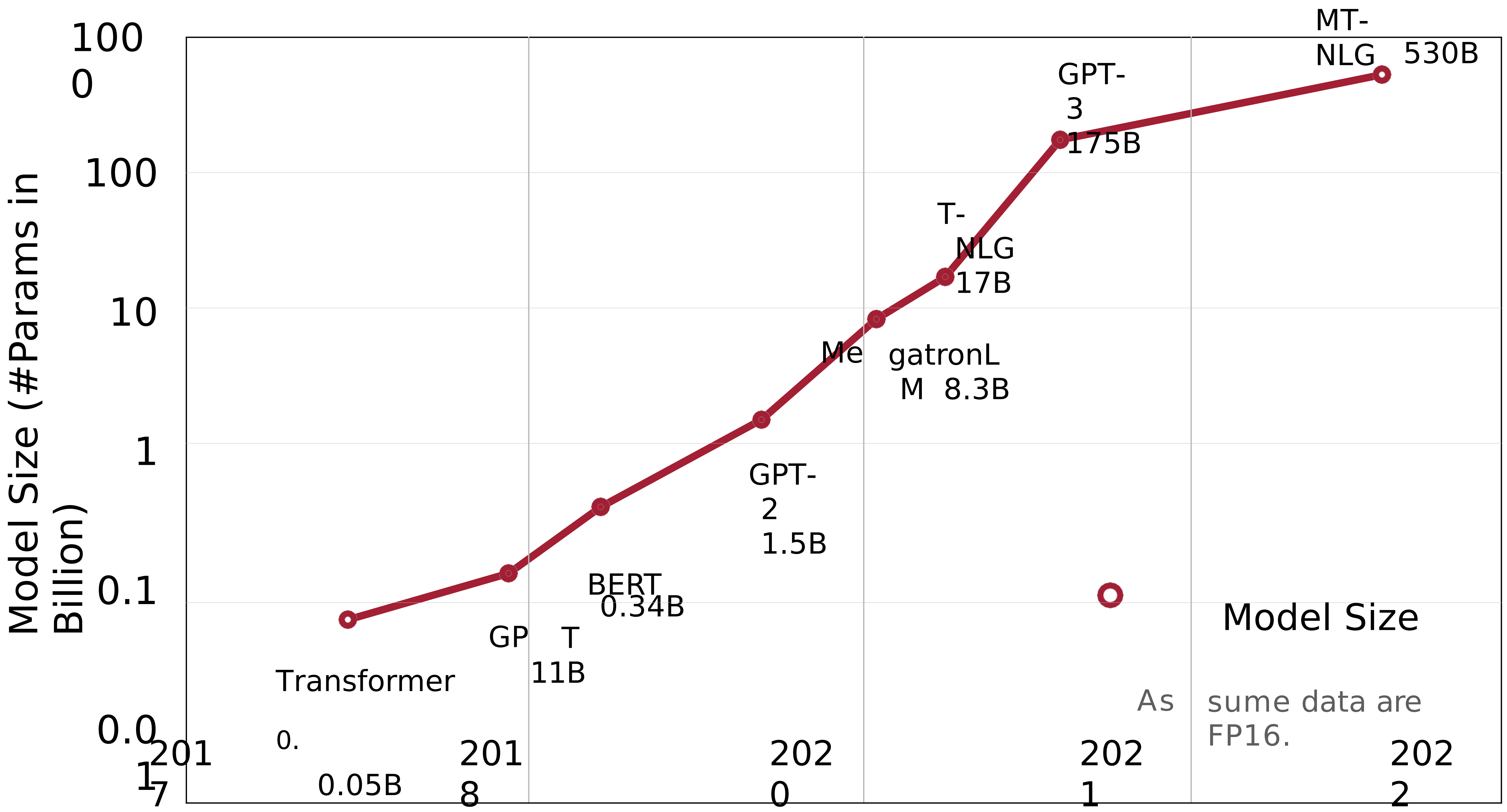
邊緣人工智慧
Edge AI

Basics of Artificial Intelligence (AI) and Deep Learning (DL)

吳凱強
Kai-Chiang Wu

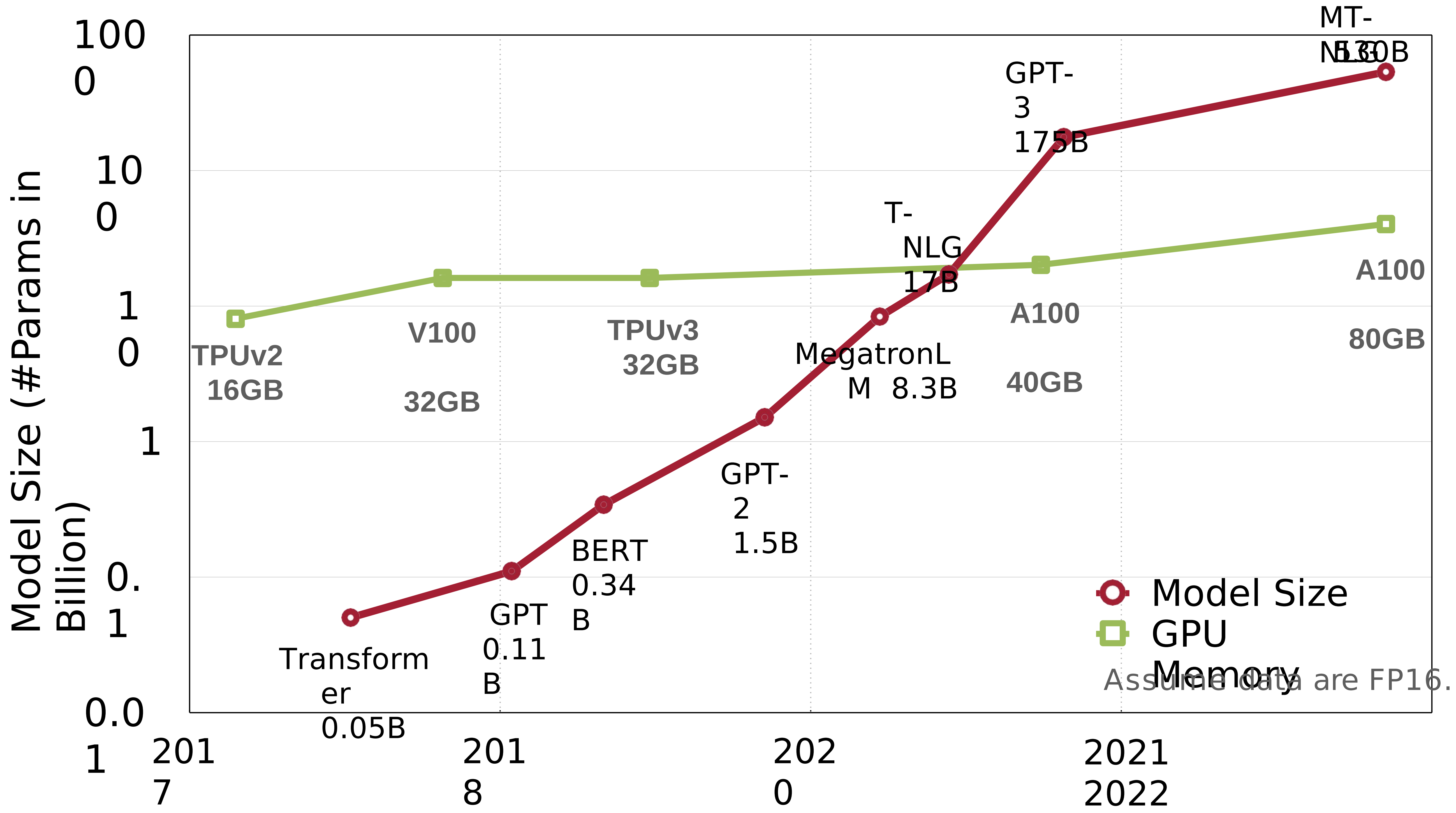
Deep Learning Continues to Scale

- **The demand of computation grows exponentially**



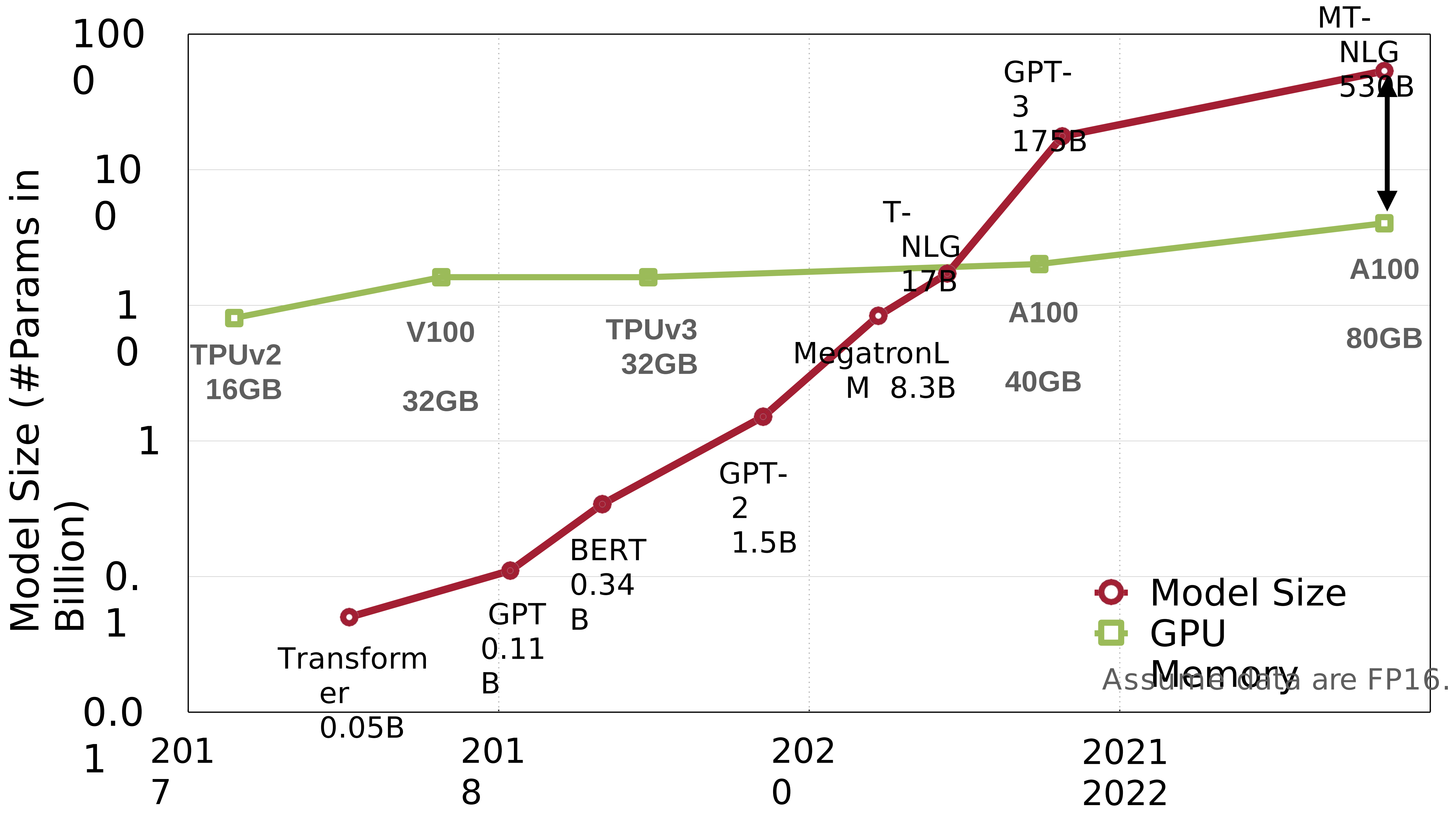
Problem: DL Models Outgrow Hardware

- Moore's Law: 2x every 2 years; DL models: 4x every 2 years



Efficient Deep Learning Techniques are Essential

- Bridges the gap between the supply and demand of computation

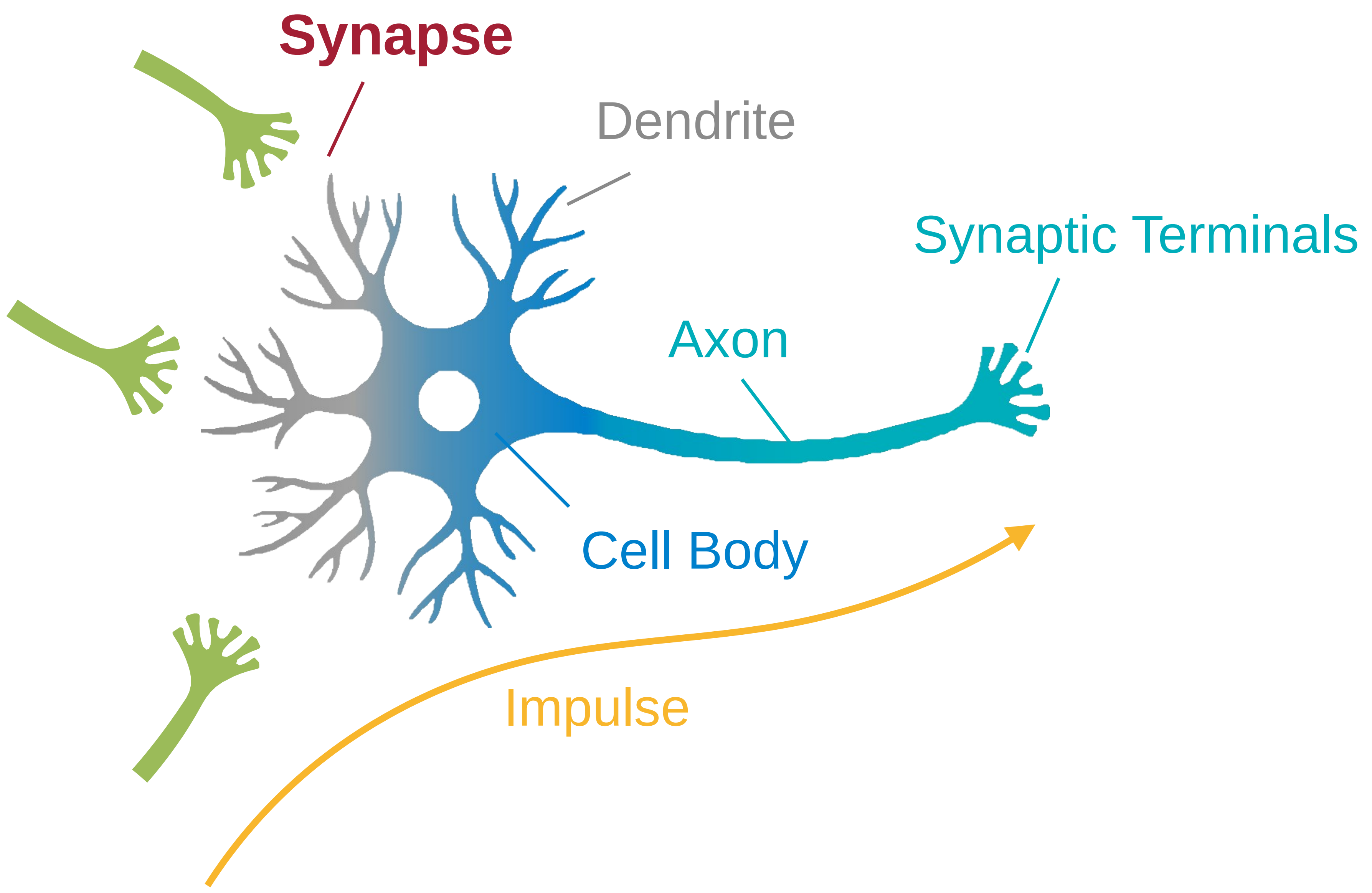


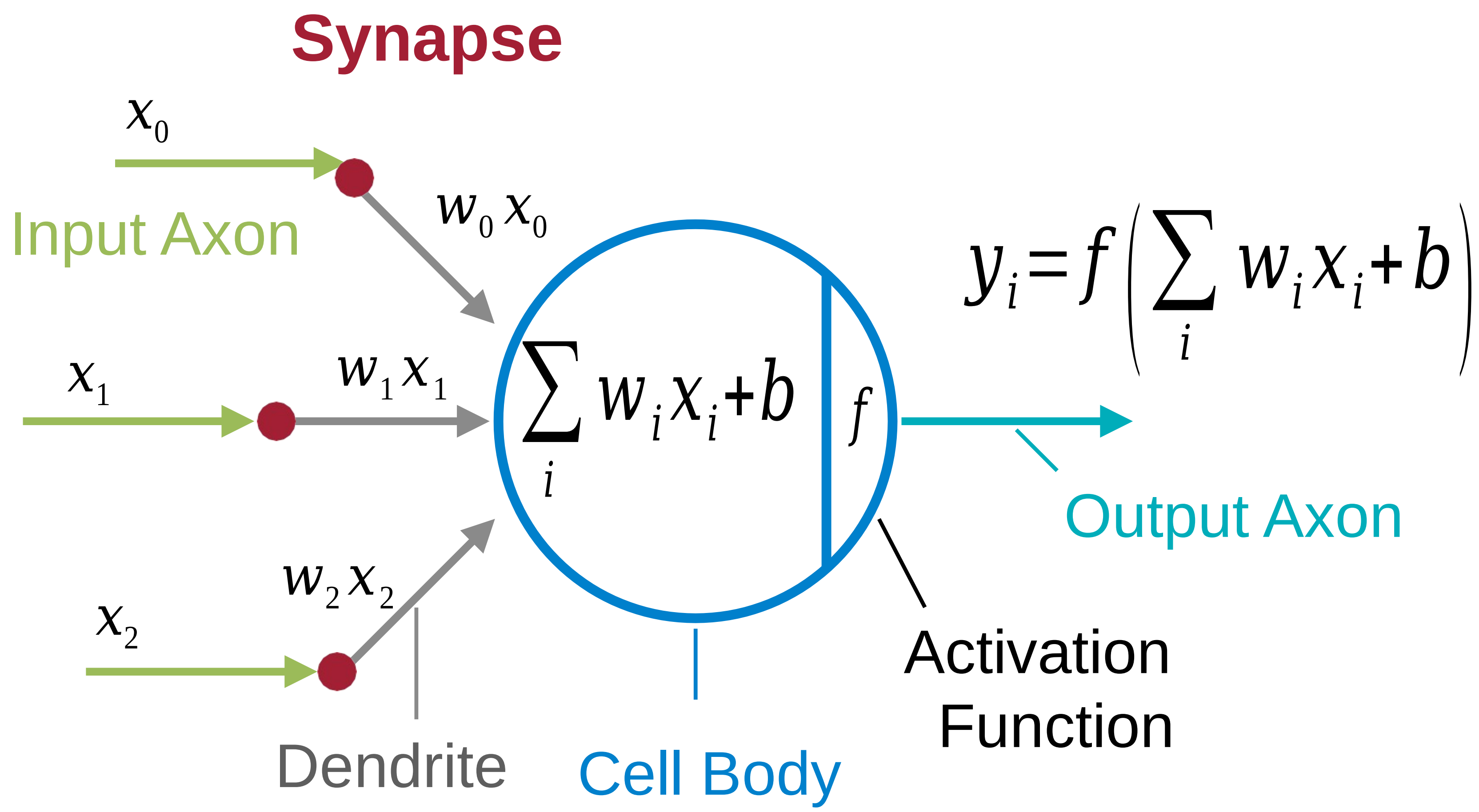
Model compression bridges the gap.

Lecture Plan

1. Review the terminology of neural networks
2. Review popular building blocks in a neural network
3. Review convolutional neural networks' architecture
4. Introduce popular efficiency metrics for neural netowrks

Neuron and Synapse





Deep Neural Network

3-Layer Neural Network With 2 Hidden Layers

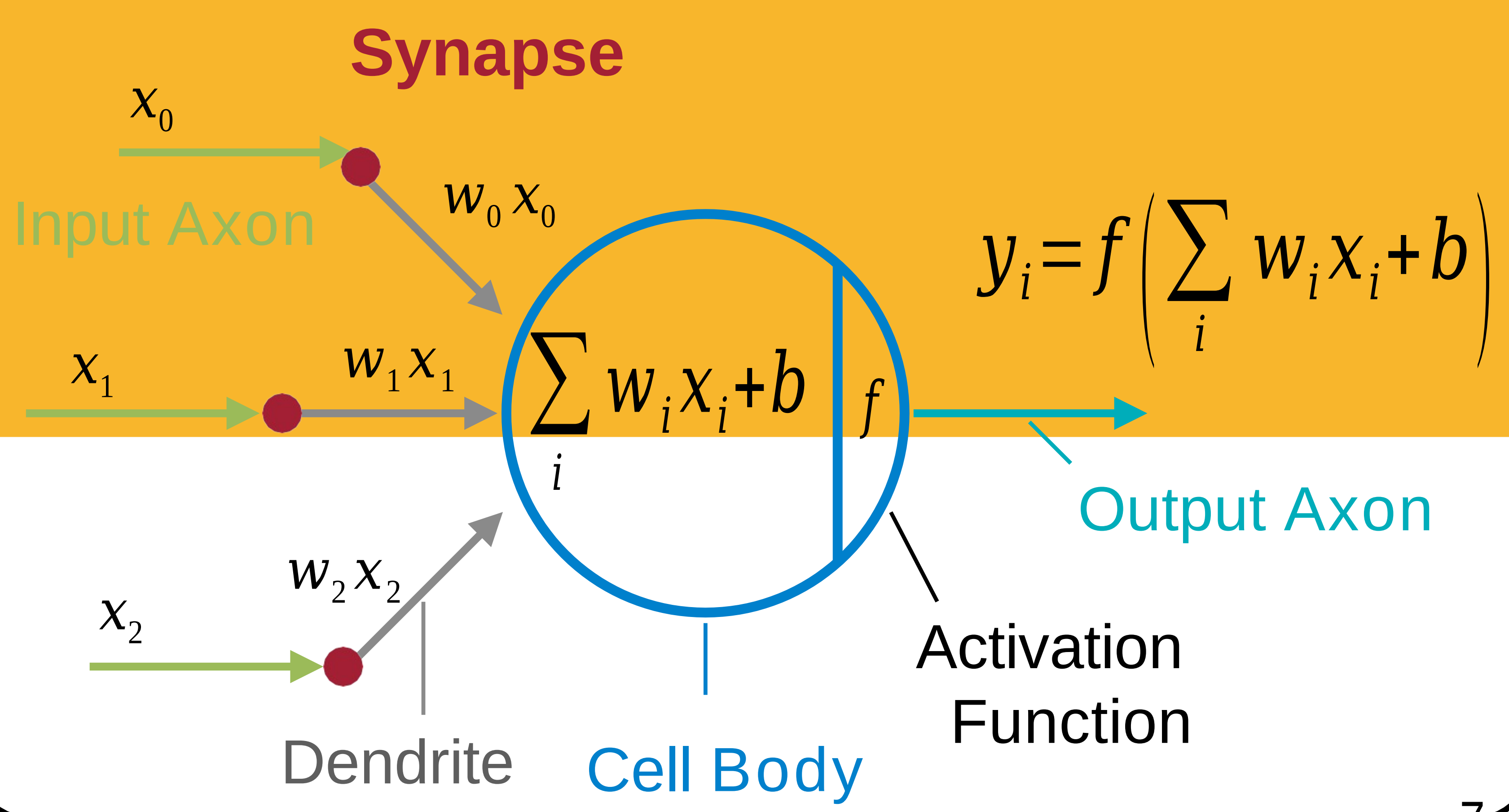
Inputs

Layer 0

Hidden Layer 1

Hidden Layer 2

Outputs

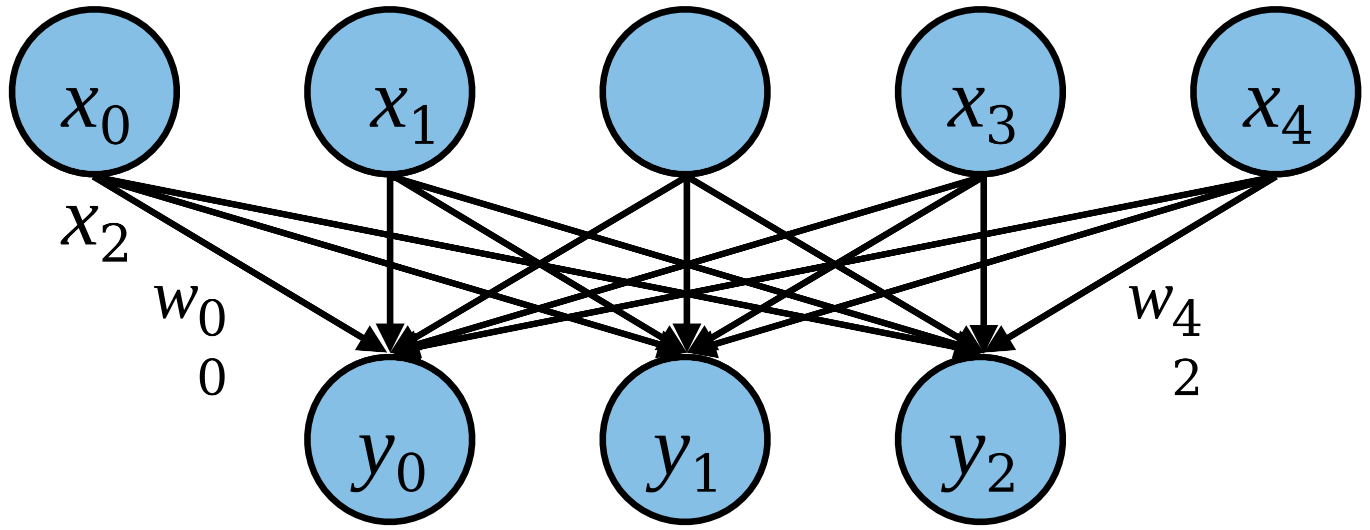


The dimensionality of these *hidden* layers determines the **width** of the model.

Popular Neural Network Layers

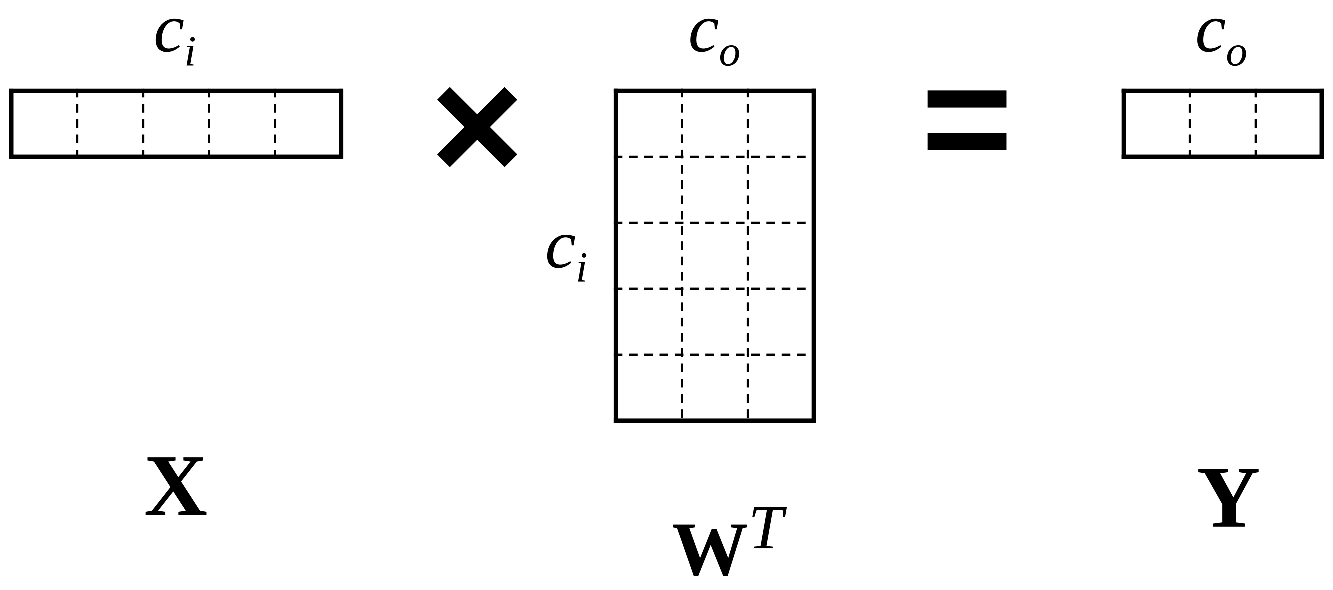
Fully-Connected Layer (Linear Layer)

- **The output neuron is connected to all input neurons.**
  - **Shape of Tensors:**
    - Input Features $\mathbf{X}$: (n, c_i)
    - Output Features $\mathbf{Y}$: (n, c_o)
    - Weights $\mathbf{W}$: (c_o, c_i)
    - Bias $\mathbf{b}$: (c_o)



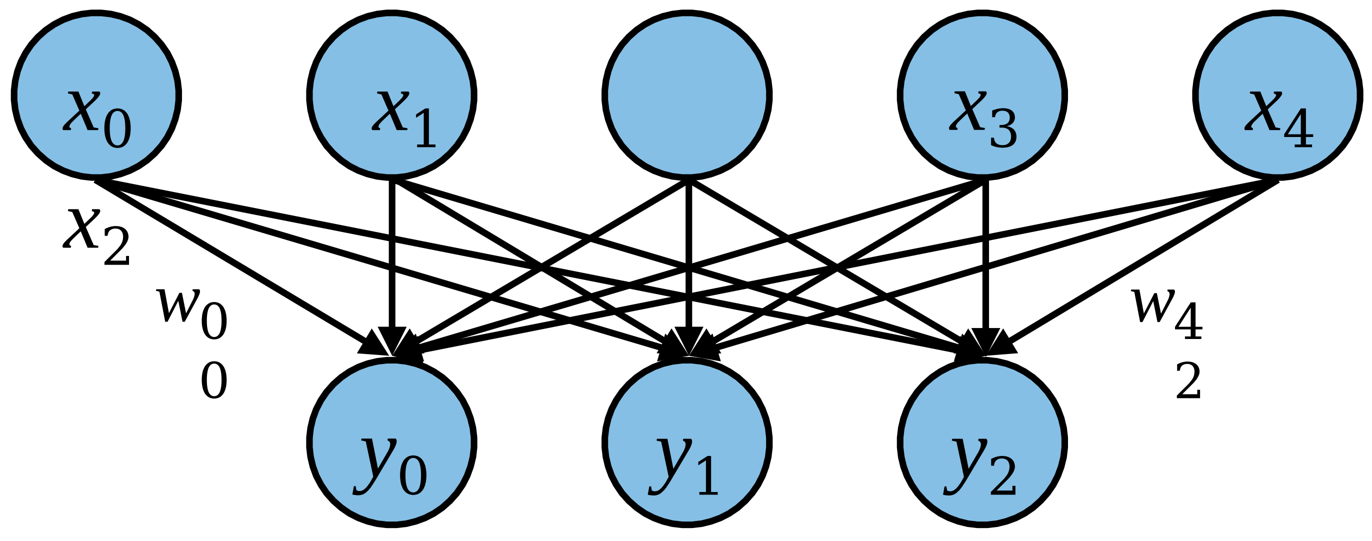
$$y_i = \sum_j w_{ij} x_j + b_i$$

Notations	
c_i	Input Channels
c_o	Output Channels



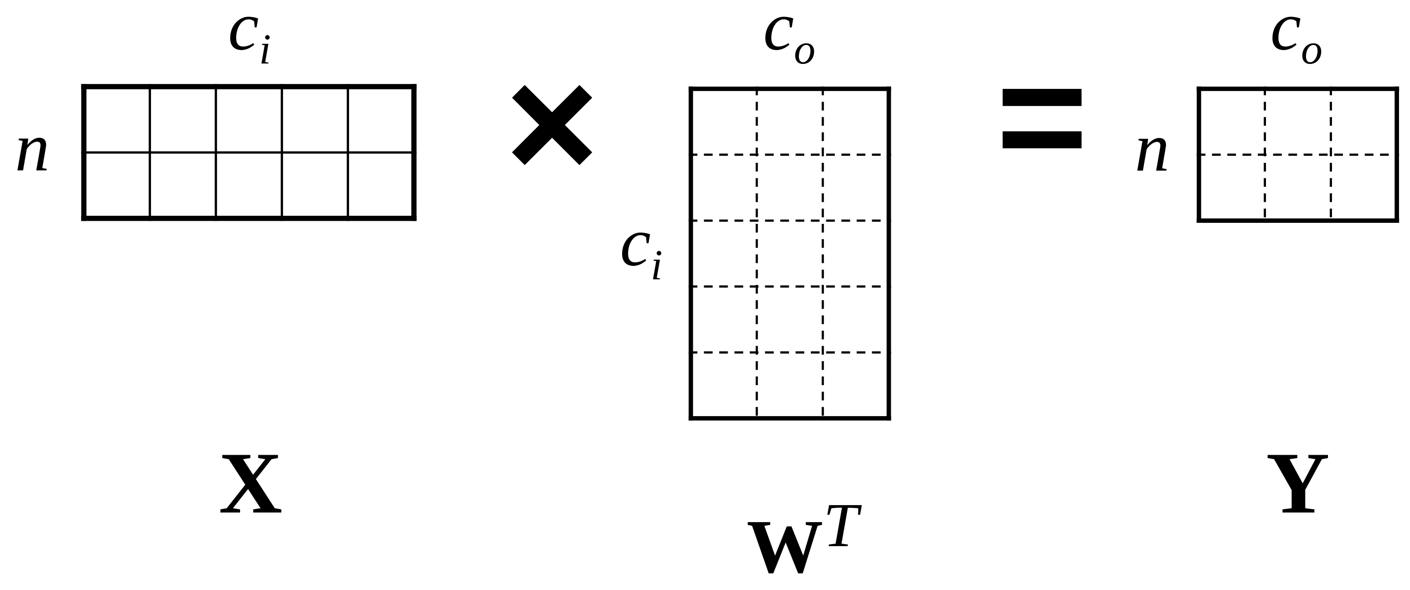
Fully-Connected Layer (Linear Layer)

- **The output neuron is connected to all input neurons.**
  - **Shape of Tensors:**
    - Input Features $\mathbf{X}$: (n, c_i)
    - Output Features $\mathbf{Y}$: (n, c_o)
    - Weights $\mathbf{W}$: (c_o, c_i)
    - Bias $\mathbf{b}$: (c_o)



$$y_i = \sum_j w_{ij} x_j + b_i$$

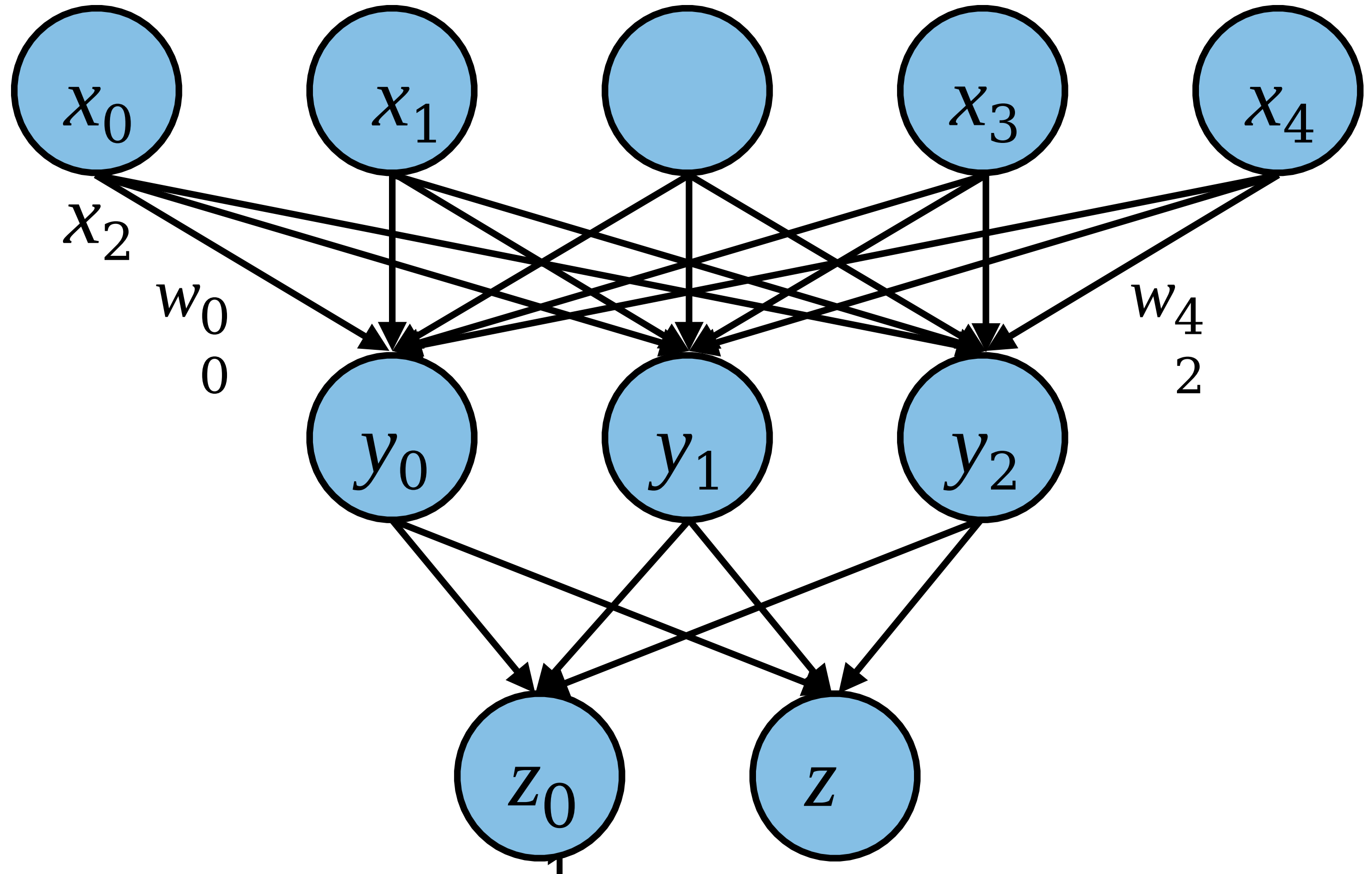
Notations	
n	Batch Size
c_i	Input Channels
c_o	Output Channels



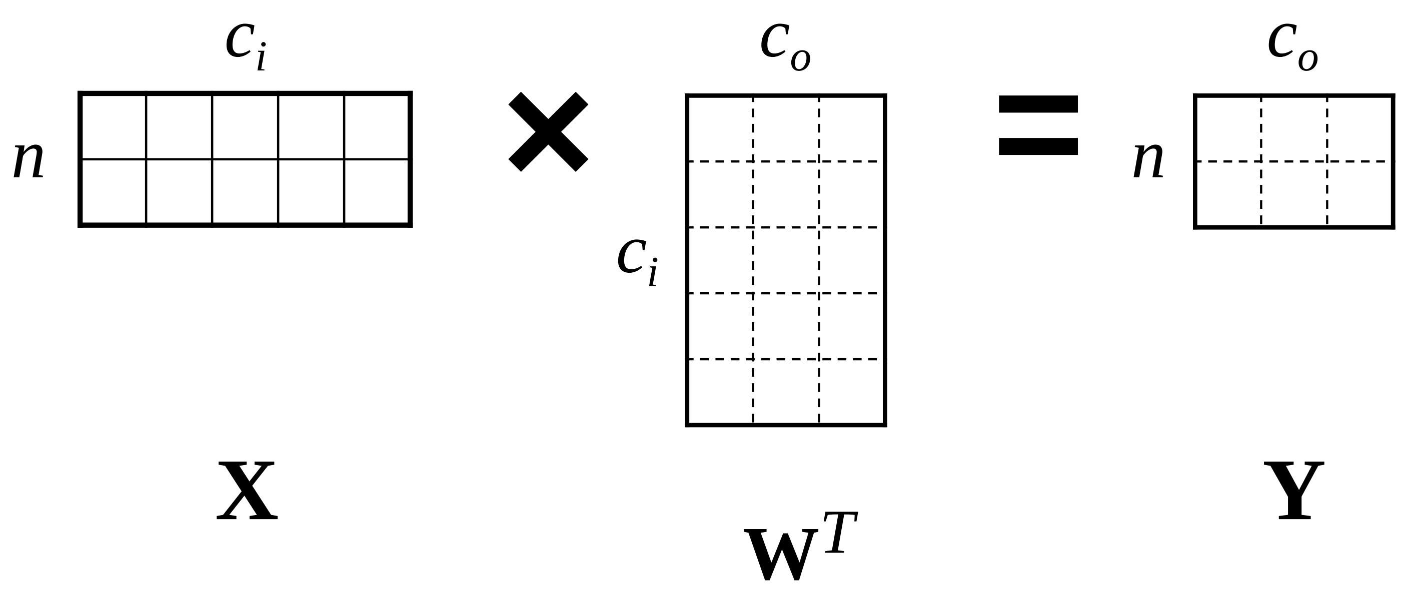
Fully-Connected Layer (Linear Layer)

- **The output neuron is connected to all input neurons.**
  - **Shape of Tensors:**
    - Input Features $\mathbf{X}$: (n, c_i)
    - Output Features $\mathbf{Y}$: (n, c_o)
    - Weights $\mathbf{W}$: (c_o, c_i)
    - Bias $\mathbf{b}$: (c_o)

Notations	
n	Batch Size
c_i	Input Channels
c_o	Output Channels



Multilayer Perceptron (MLP)



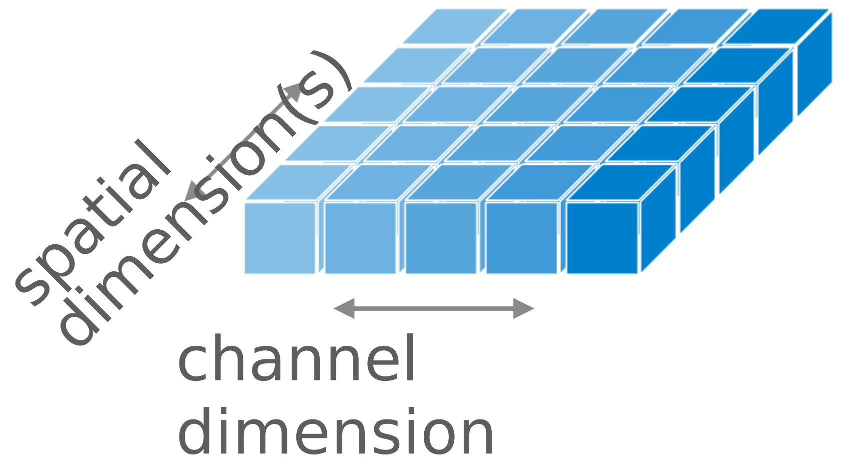
Convolution Layer

- **The output neuron is connected to input neurons in the receptive field**
  - **Shape of Tensors:**
    - Input Features $\mathbf{X}$: $\cancel{(n, c_i)}$
    - Output Features $\mathbf{Y}$: $\cancel{(n, c_o)}$
    - Weights $\mathbf{W}$: $\cancel{(c_o, c_i)}$
    - Bias $\mathbf{b}$: (c_o)

1D Conv

$(n=1, c_i, w_i)$

$(n=1, c_o, w_o)$



Notations	
n	Batch Size
c_i	Input Channels
c_o	Output Channels
w_i, w_o	Input/Output Width

Convolution Layer

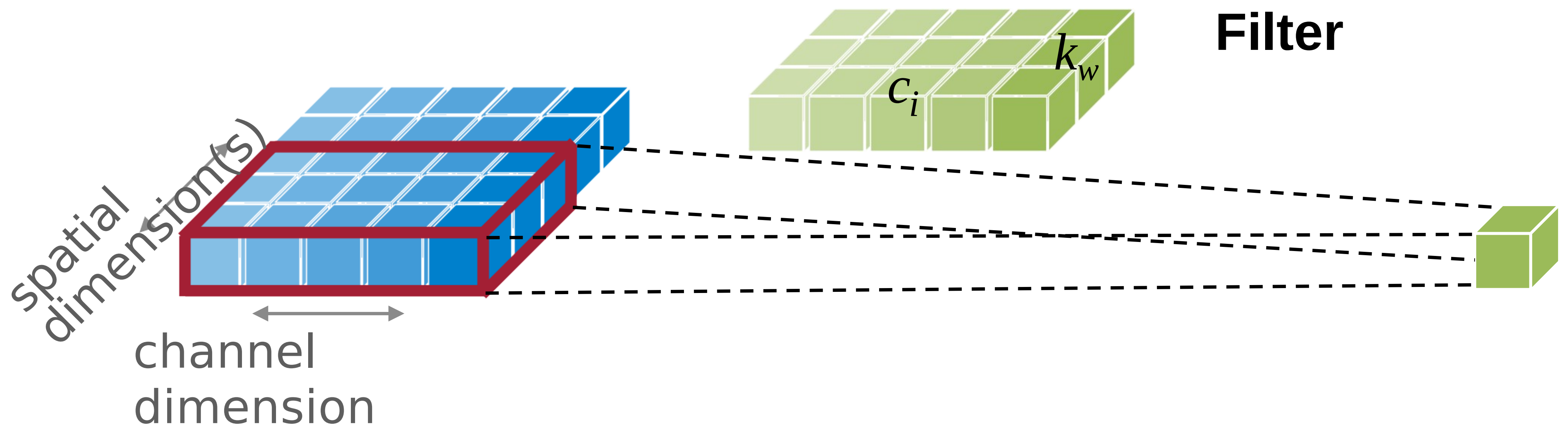
- The output neuron is connected to input neurons in the receptive field
  - Shape of Tensors:
    - Input Features **X** : $\cancel{(n, c_i)}$
    - Output Features **Y** : $\cancel{(n, c_o)}$
    - Weights **W** : $\cancel{(c_o, c_i)}$
    - Bias **b** : (c_o)

1D Conv

$(n\!=\!1, c_i, w_i)$

$(n\!=\!1, c_o, w_o)$

(c_o, c_i, k_w)



Notations	
n	Batch Size
c_i	Input Channels
c_o	Output Channels
w_i, w_o	Input/Output Width
k_w	Kernel Width

Convolution Layer

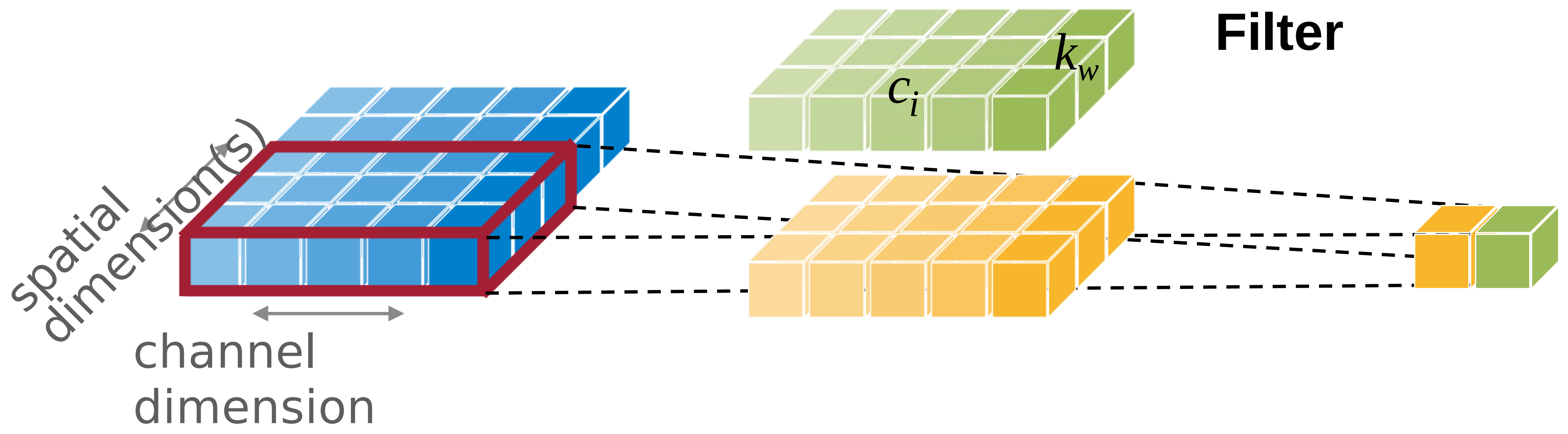
- **The output neuron is connected to input neurons in the receptive field**
  - **Shape of Tensors:**
    - Input Features $\mathbf{X}$: $\cancel{(n, c_i)}$
    - Output Features $\mathbf{Y}$: $\cancel{(n, c_o)}$
    - Weights $\mathbf{W}$: $\cancel{(c_o, c_i)}$
    - Bias $\mathbf{b}$: (c_o)

1D Conv

$(n{=}1, c_i, w_i)$

$(n{=}1, c_o, w_o)$

(c_o, c_i, k_w)



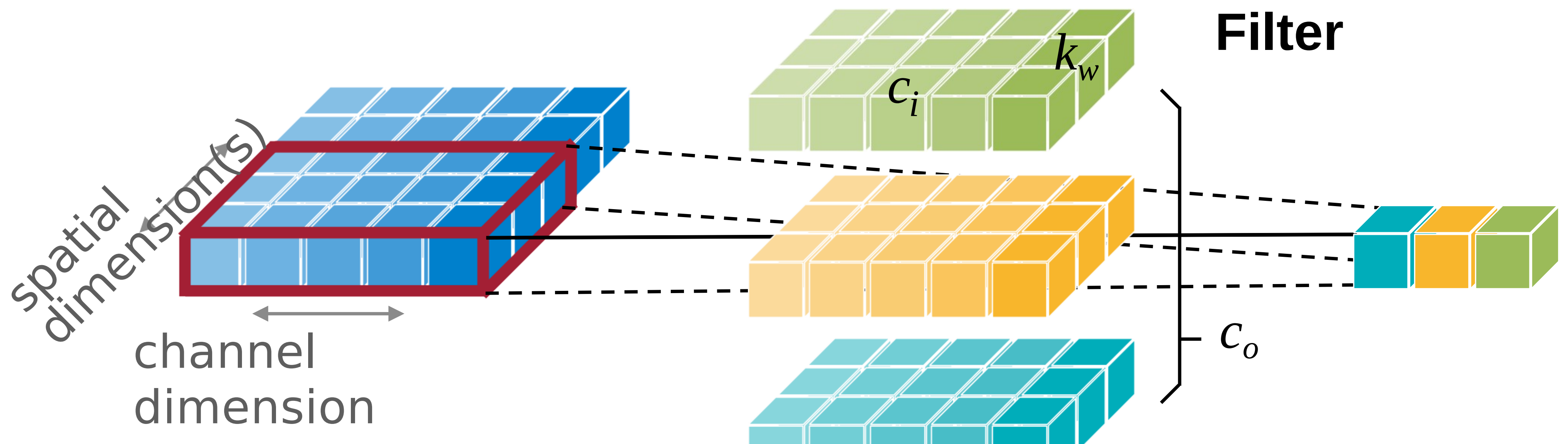
Notations	
n	Batch Size
c_i	Input Channels
c_o	Output Channels
w_i, w_o	Input/Output Width
k_w	Kernel Width

Convolution Layer

- **The output neuron is connected to input neurons in the receptive field**
  - **Shape of Tensors:**
    - Input Features $\mathbf{X}$: ~~(n, c_i)~~ $(n{=}1, c_i, w_i)$
    - Output Features $\mathbf{Y}$: ~~(n, c_o)~~ $(n{=}1, c_o, w_o)$
    - Weights $\mathbf{W}$: ~~(c_o, c_i)~~ (c_o, c_i, k_w)
    - Bias $\mathbf{b}$: (c_o)

1D Conv

Notations	
n	Batch Size
c_i	Input Channels
c_o	Output Channels
w_i, w_o	Input/Output Width
k_w	Kernel Width



Convolution Layer

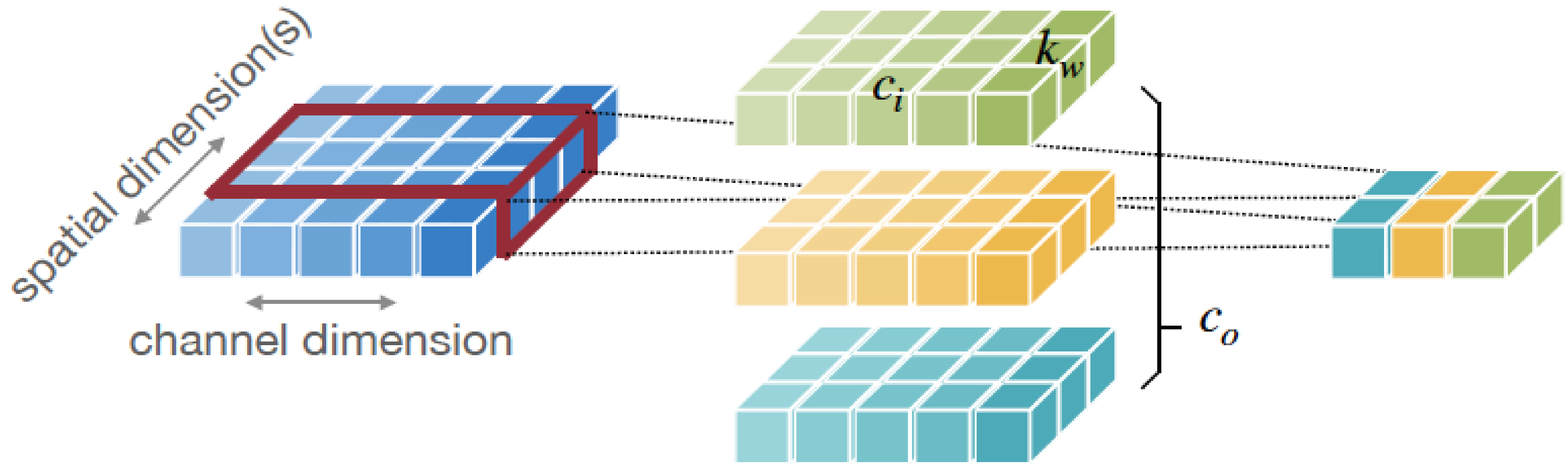
- The output neuron is connected to input neurons in the receptive field
  - Shape of Tensors:
    - Input Features $\mathbf{X}$: $\cancel{(n, c_i)}$
    - Output Features $\mathbf{Y}$: $\cancel{(n, c_o)}$
    - Weights $\mathbf{W}$: $\cancel{(c_o, c_i)}$
    - Bias $\mathbf{b}$: (c_o)

1D Conv

$(n\!=\!1, c_i, w_i)$

$(n\!=\!1, c_o, w_o)$

(c_o, c_i, k_w)



Notations	
n	Batch Size
c_i	Input Channels
c_o	Output Channels
w_i, w_o	Input/Output Width
k_w	Kernel Width

Convolution Layer

- **The output neuron is connected to input neurons in the receptive field**
  - **Shape of Tensors:**
    - Input Features $\mathbf{X}$: $\cancel{(n, c_i)}$
    - Output Features $\mathbf{Y}$: $\cancel{(n, c_o)}$
    - Weights $\mathbf{W}$: $\cancel{(c_o, c_i)}$
    - Bias $\mathbf{b}$: (c_o)

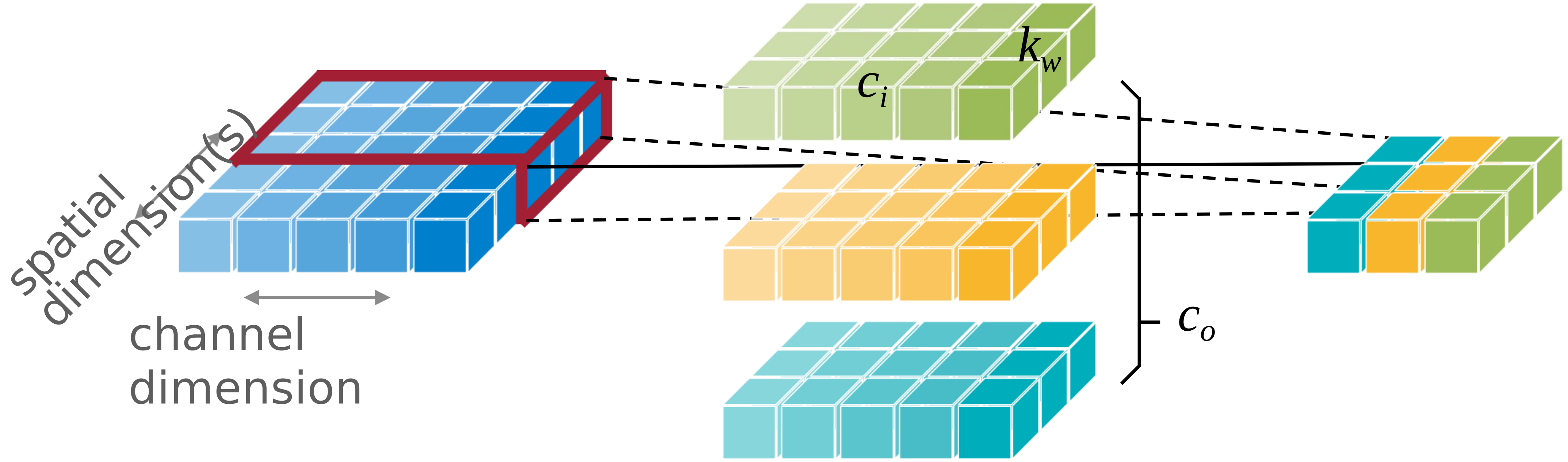
1D Conv

$(n{=}1, c_i, w_i)$

$(n{=}1, c_o, w_o)$

(c_o, c_i, k_w)

Notations	
n	Batch Size
c_i	Input Channels
c_o	Output Channels
w_i, w_o	Input/Output Width
k_w	Kernel Width

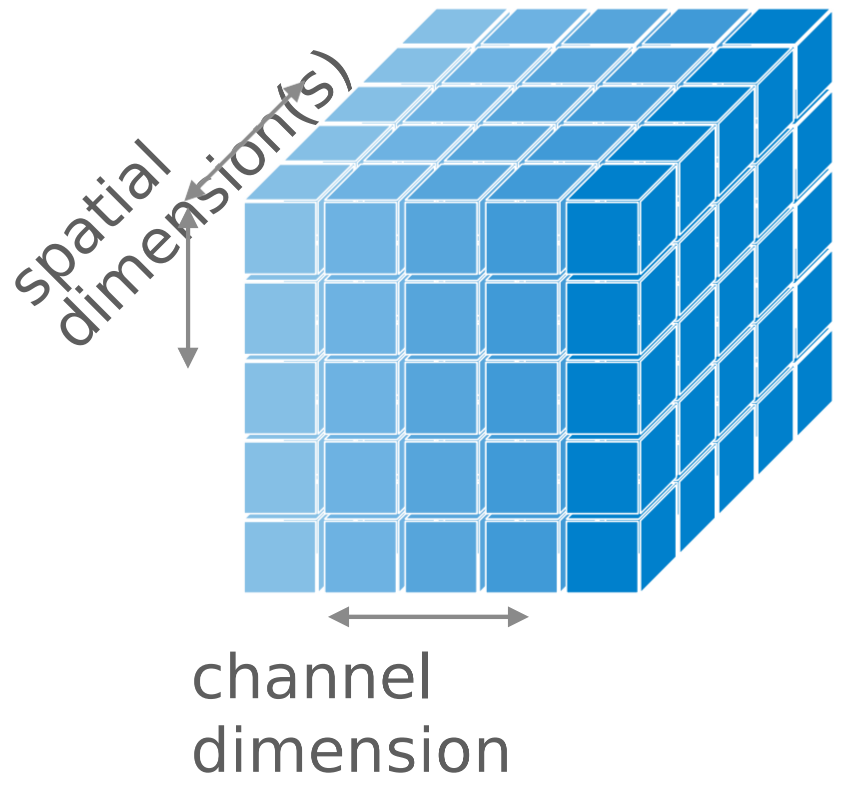


Convolution Layer

- **The output neuron is connected to input neurons in the receptive field**
  - **Shape of Tensors:**
    - Input Features $\mathbf{X}$: $\cancel{(n, c_i)}$
    - Output Features $\mathbf{Y}$: $\cancel{(n, c_o)}$
    - Weights $\mathbf{W}$: $\cancel{(c_o, c_i)}$
    - Bias $\mathbf{b}$: (c_o)

1D Conv	2D Conv
(n, c_i, w_i)	$(n{=}1, c_i, h_i, w_i)$
(n, c_o, w_o)	$(n{=}1, c_o, h_o, w_o)$
(c_o, c_i, k_w)	(c_o, c_i, k_h, k_w)

Notations	
n	Batch Size
c_i	Input Channels
c_o	Output Channels
w_i, w_o	Input/Output Width
h_i, h_o	Input/Output Height



Activation Map / Feature Map

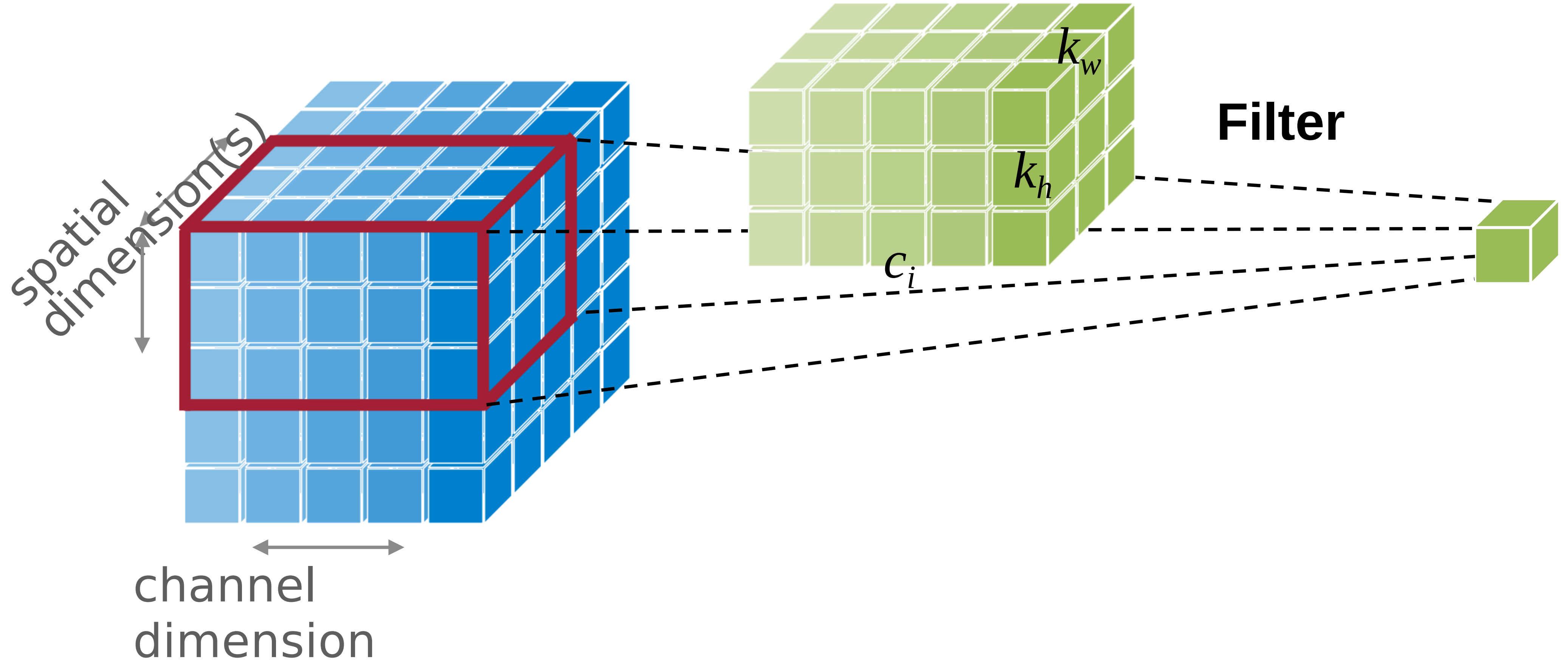
hw

Convolution Layer

- **The output neuron is connected to input neurons in the receptive field**
  - **Shape of Tensors:**
    - Input Features $\mathbf{X}$: $\cancel{(n, c_i)}$
    - Output Features $\mathbf{Y}$: $\cancel{(n, c_o)}$
    - Weights $\mathbf{W}$: $\cancel{(c_o, c_i)}$
    - Bias $\mathbf{b}$: (c_o)

1D Conv	2D Conv
(n, c_i, w_i)	$(n{=}1, c_i, h_i, w_i)$
(n, c_o, w_o)	$(n{=}1, c_o, h_o, w_o)$
(c_o, c_i, k_w)	(c_o, c_i, k_h, k_w)

Notations	
n	Batch Size
c_i	Input Channels
c_o	Output Channels
w_i, w_o	Input/Output Width
h_i, h_o	Input/Output Height
k_h	Kernel Height
k_w	Kernel Width

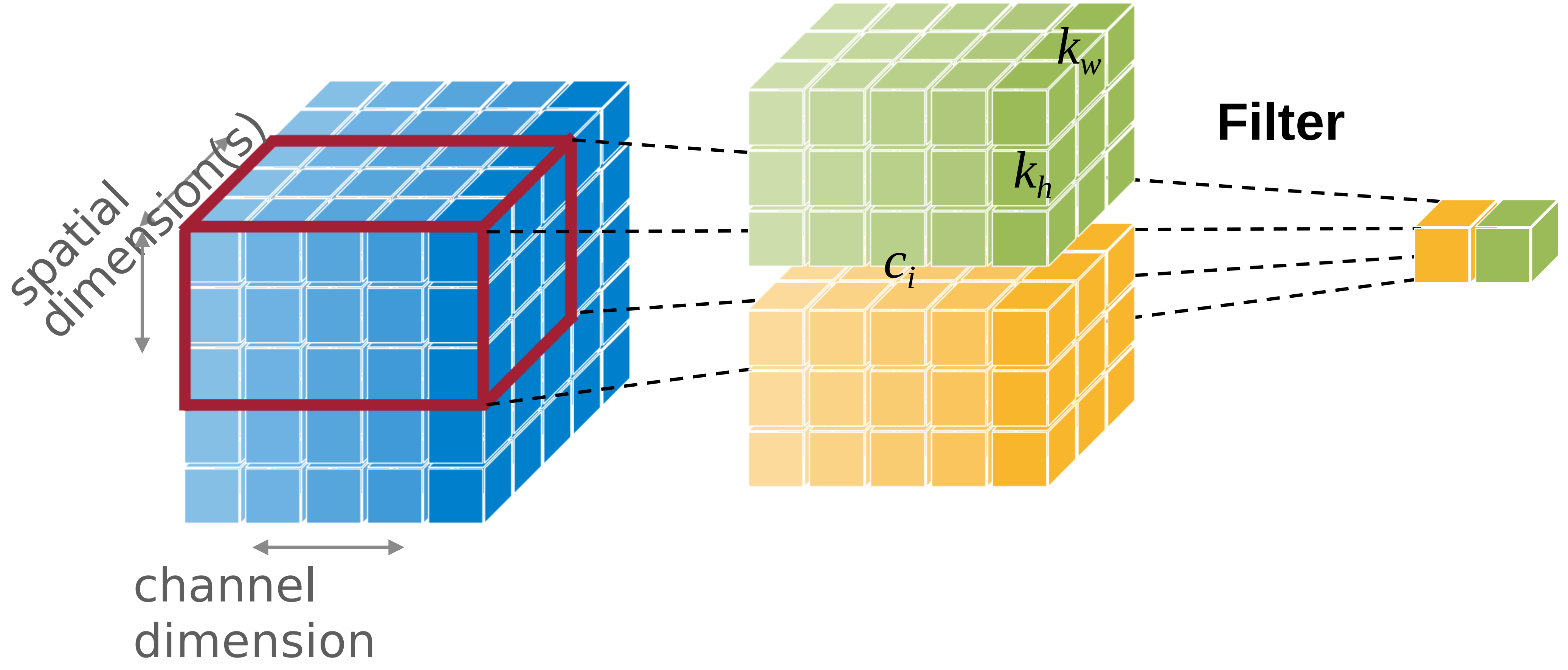


Convolution Layer

- **The output neuron is connected to input neurons in the receptive field**
  - **Shape of Tensors:**
    - Input Features $\mathbf{X}$: $\cancel{(n, c_i)}$
    - Output Features $\mathbf{Y}$: $\cancel{(n, c_o)}$
    - Weights $\mathbf{W}$: $\cancel{(c_o, c_i)}$
    - Bias $\mathbf{b}$: (c_o)

1D Conv	2D Conv
(n, c_i, w_i)	$(n{=}1, c_i, h_i, w_i)$
(n, c_o, w_o)	$(n{=}1, c_o, h_o, w_o)$
(c_o, c_i, k_w)	(c_o, c_i, k_h, k_w)

Notations	
n	Batch Size
c_i	Input Channels
c_o	Output Channels
w_i, w_o	Input/Output Width
h_i, h_o	Input/Output Height
k_h	Kernel Height
k_w	Kernel Width

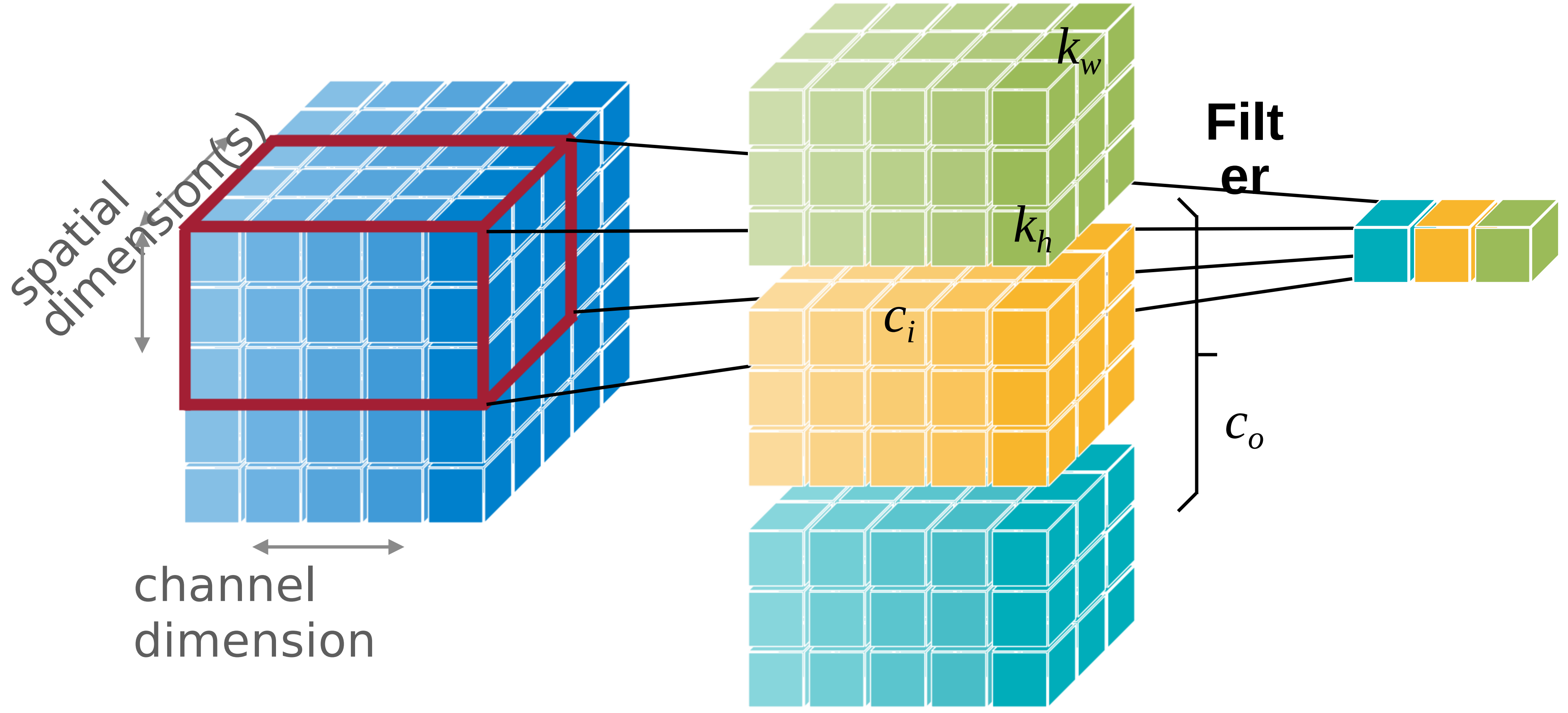


Convolution Layer

- **The output neuron is connected to input neurons in the receptive field**
  - **Shape of Tensors:**
    - Input Features $\mathbf{X}$: ~~(n, c_i)~~
    - Output Features $\mathbf{Y}$: ~~(n, c_o)~~
    - Weights $\mathbf{W}$: ~~(c_o, c_i)~~
    - Bias $\mathbf{b}$: (c_o)

1D Conv	2D Conv
(n, c_i, w_i)	$(n{=}1, c_i, h_i, w_i)$
(n, c_o, w_o)	$(n{=}1, c_o, h_o, w_o)$
(c_o, c_i, k_w)	(c_o, c_i, k_h, k_w)

Notations	
n	Batch Size
c_i	Input Channels
c_o	Output Channels
w_i, w_o	Input/Output Width
h_i, h_o	Input/Output Height
k_h	Kernel Height
k_w	Kernel Width

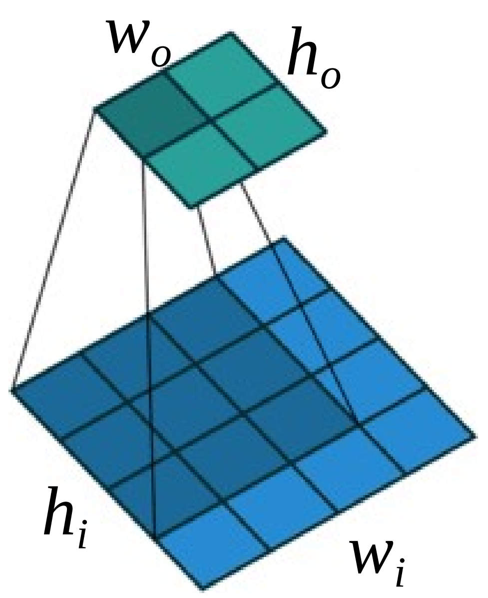


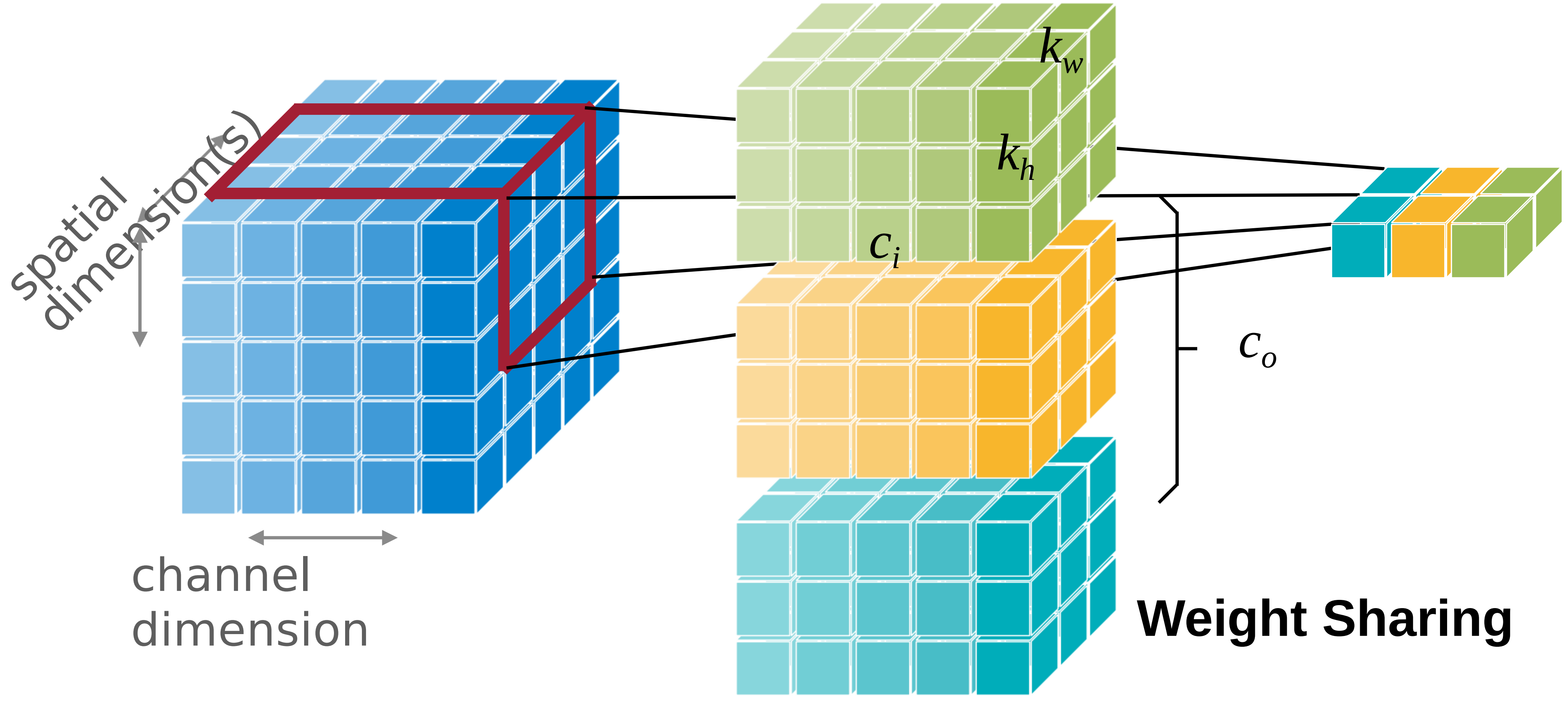
Convolution Layer

- **The output neuron is connected to input neurons in the receptive field**
  - **Shape of Tensors:**
    - Input Features $\mathbf{X}$: $\cancel{(n, c_i)}$
    - Output Features $\mathbf{Y}$: $\cancel{(n, c_o)}$
    - Weights $\mathbf{W}$: $\cancel{(c_o, c_i)}$
    - Bias $\mathbf{b}$: (c_o)

1D Conv	2D Conv
(n, c_i, w_i)	$(n{=}1, c_i, h_i, w_i)$
(n, c_o, w_o)	$(n{=}1, c_o, h_o, w_o)$
(c_o, c_i, k_w)	(c_o, c_i, k_h, k_w)

Notations	
n	Batch Size
c_i	Input Channels
c_o	Output Channels
w_i, w_o	Input/Output Width
h_i, h_o	Input/Output Height
k_h	Kernel Height
k_w	Kernel Width

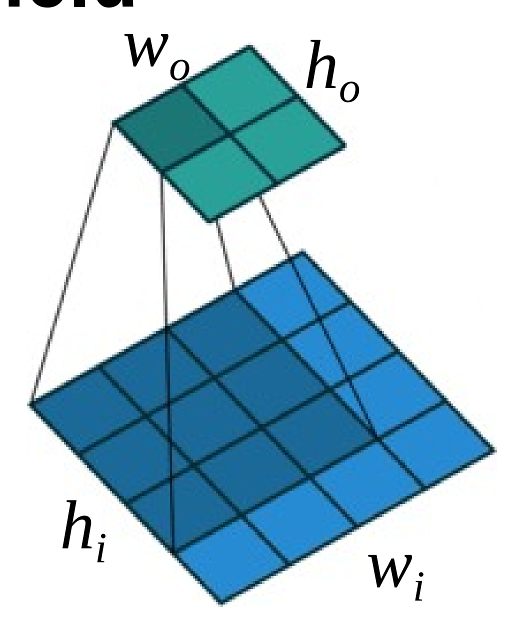




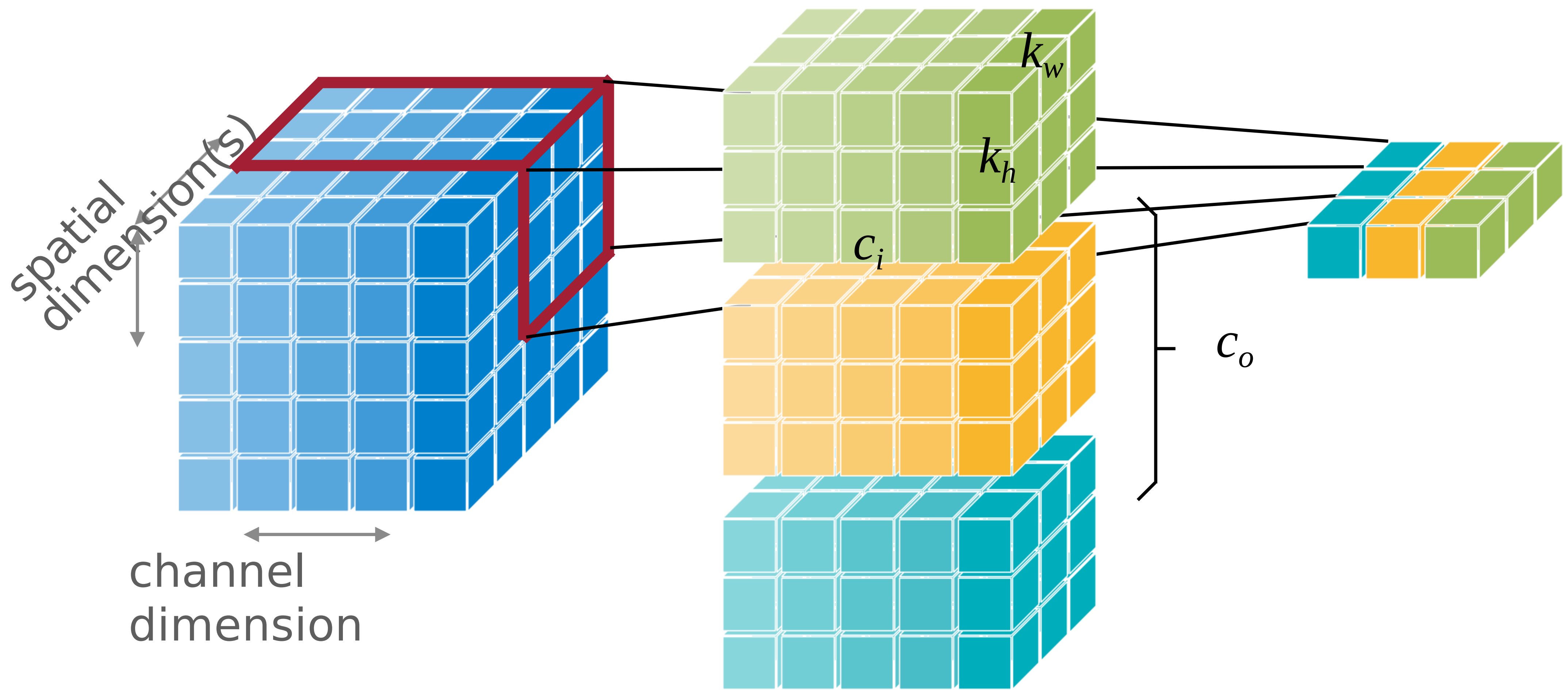
Convolution Layer

- The output neuron is connected to input neurons in the receptive field
  - Shape of Tensors:
    - Input Features $\mathbf{X}$: $\cancel{(n, c_i)}$
    - Output Features $\mathbf{Y}$: $\cancel{(n, c_o)}$
    - Weights $\mathbf{W}$: $\cancel{(c_o, c_i)}$
    - Bias $\mathbf{b}$: (c_o)

1D Conv	2D Conv
(n, c_i, w_i)	$(n{=}1, c_i, h_i, w_i)$
(n, c_o, w_o)	$(n{=}1, c_o, h_o, w_o)$
(c_o, c_i, k_w)	(c_o, c_i, k_h, k_w)



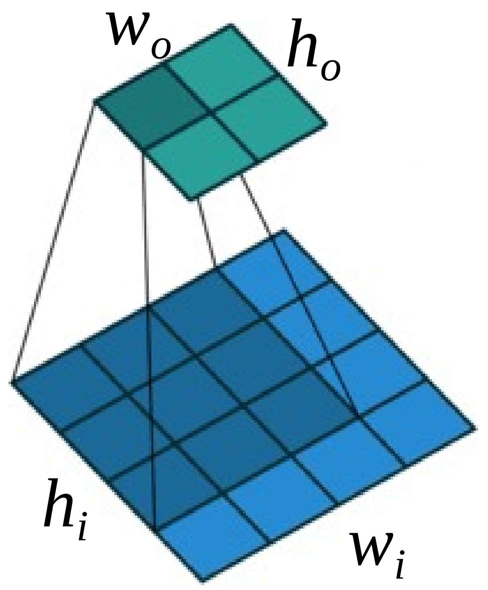
Notations	
n	Batch Size
c_i	Input Channels
c_o	Output Channels
w_i, w_o	Input/Output Width
h_i, h_o	Input/Output Height
k_h	Kernel Height
k_w	Kernel Width



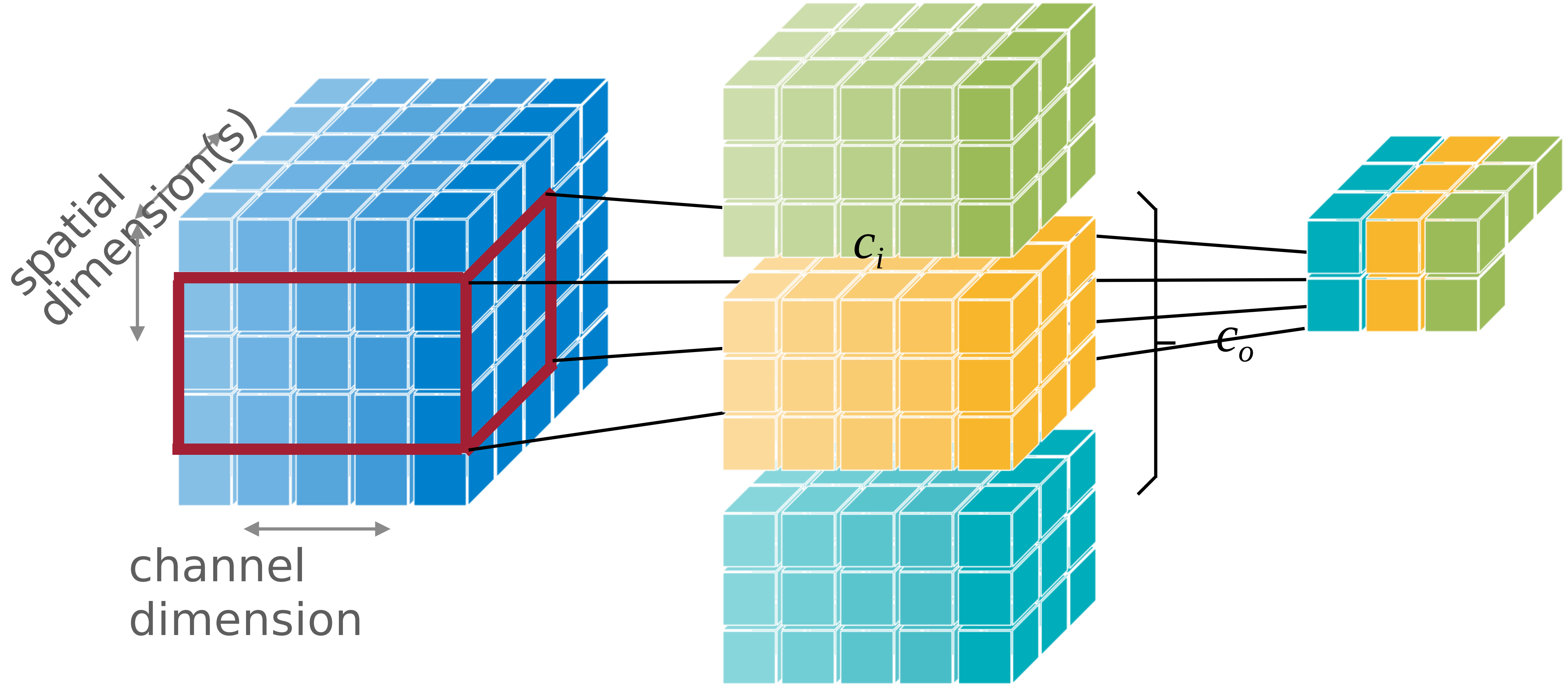
Convolution Layer

- **The output neuron is connected to input neurons in the receptive field**
  - **Shape of Tensors:**
    - Input Features $\mathbf{X}$: $\cancel{(n, c_i)}$
    - Output Features $\mathbf{Y}$: $\cancel{(n, c_o)}$
    - Weights $\mathbf{W}$: $\cancel{(c_o, c_i)}$
    - Bias $\mathbf{b}$: (c_o)

1D Conv	2D Conv
(n, c_i, w_i)	$(n{=}1, c_i, h_i, w_i)$
(n, c_o, w_o)	$(n{=}1, c_o, h_o, w_o)$
(c_o, c_i, k_w)	(c_o, c_i, k_h, k_w)



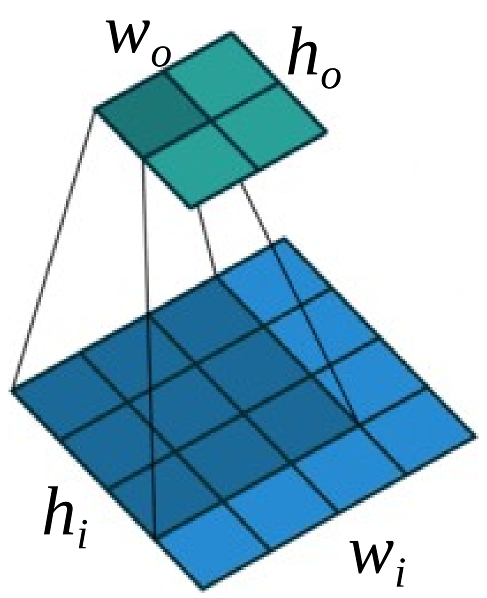
Notations	
n	Batch Size
c_i	Input Channels
c_o	Output Channels
w_i, w_o	Input/Output Width
h_i, h_o	Input/Output Height
k_h	Kernel Height
k_w	Kernel Width



Convolution Layer

- **The output neuron is connected to input neurons in the receptive field**
  - **Shape of Tensors:**
    - Input Features $\mathbf{X}$: $\cancel{(n, c_i)}$
    - Output Features $\mathbf{Y}$: $\cancel{(n, c_o)}$
    - Weights $\mathbf{W}$: $\cancel{(c_o, c_i)}$
    - Bias $\mathbf{b}$: (c_o)

1D Conv	2D Conv
(n, c_i, w_i)	$(n{=}1, c_i, h_i, w_i)$
(n, c_o, w_o)	$(n{=}1, c_o, h_o, w_o)$
(c_o, c_i, k_w)	(c_o, c_i, k_h, k_w)



Notations	
n	Batch Size
c_i	Input Channels
c_o	Output Channels
w_i, w_o	Input/Output Width
h_i, h_o	Input/Output Height
k_h	Kernel Height
k_w	Kernel Width

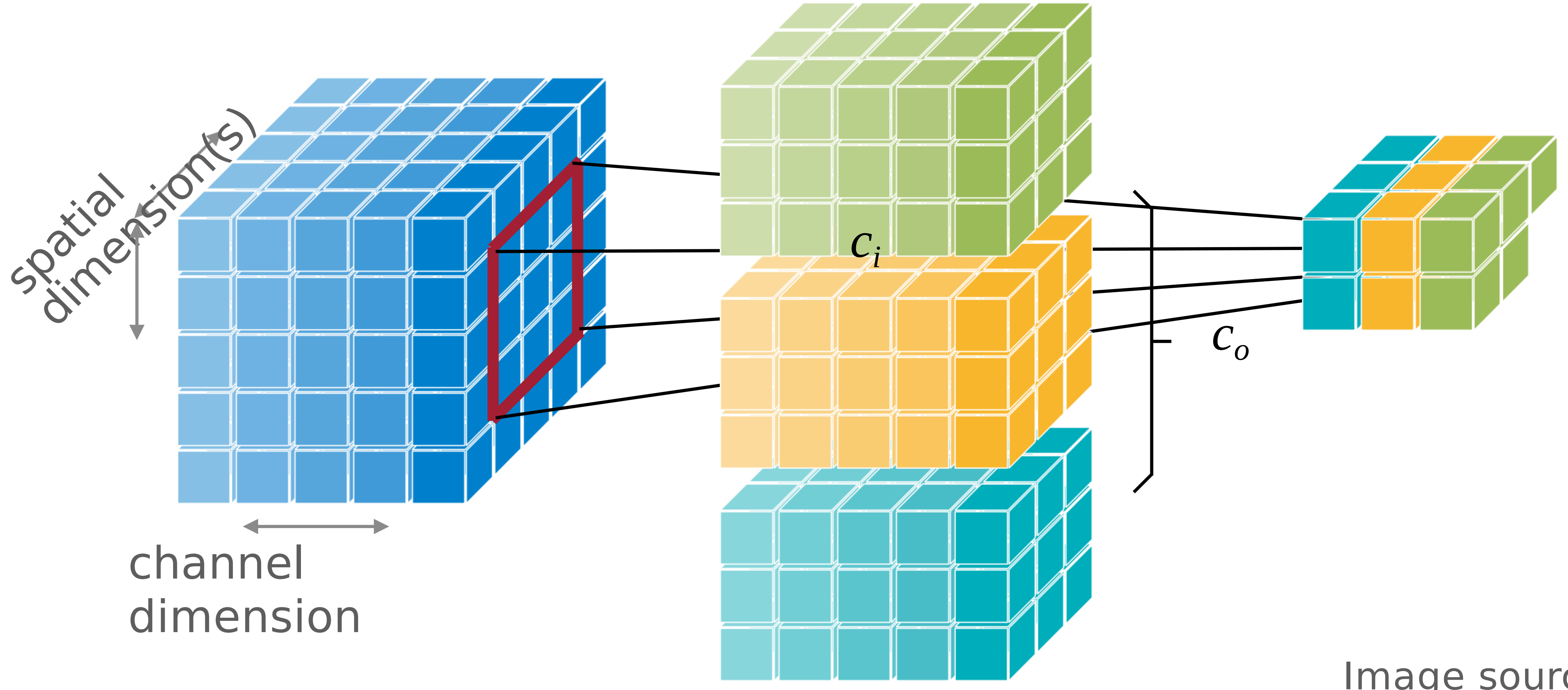
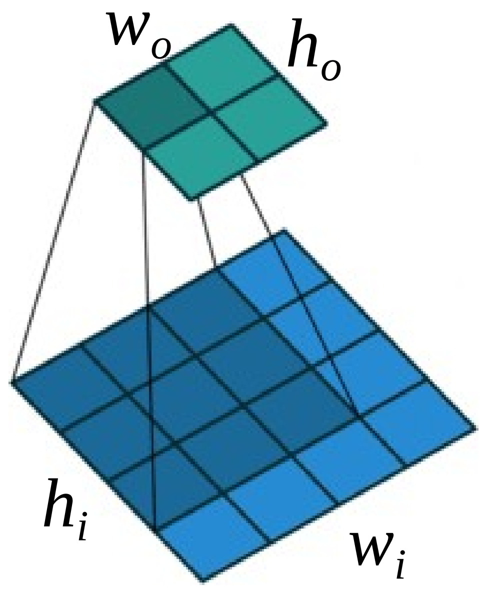


Image source: 1

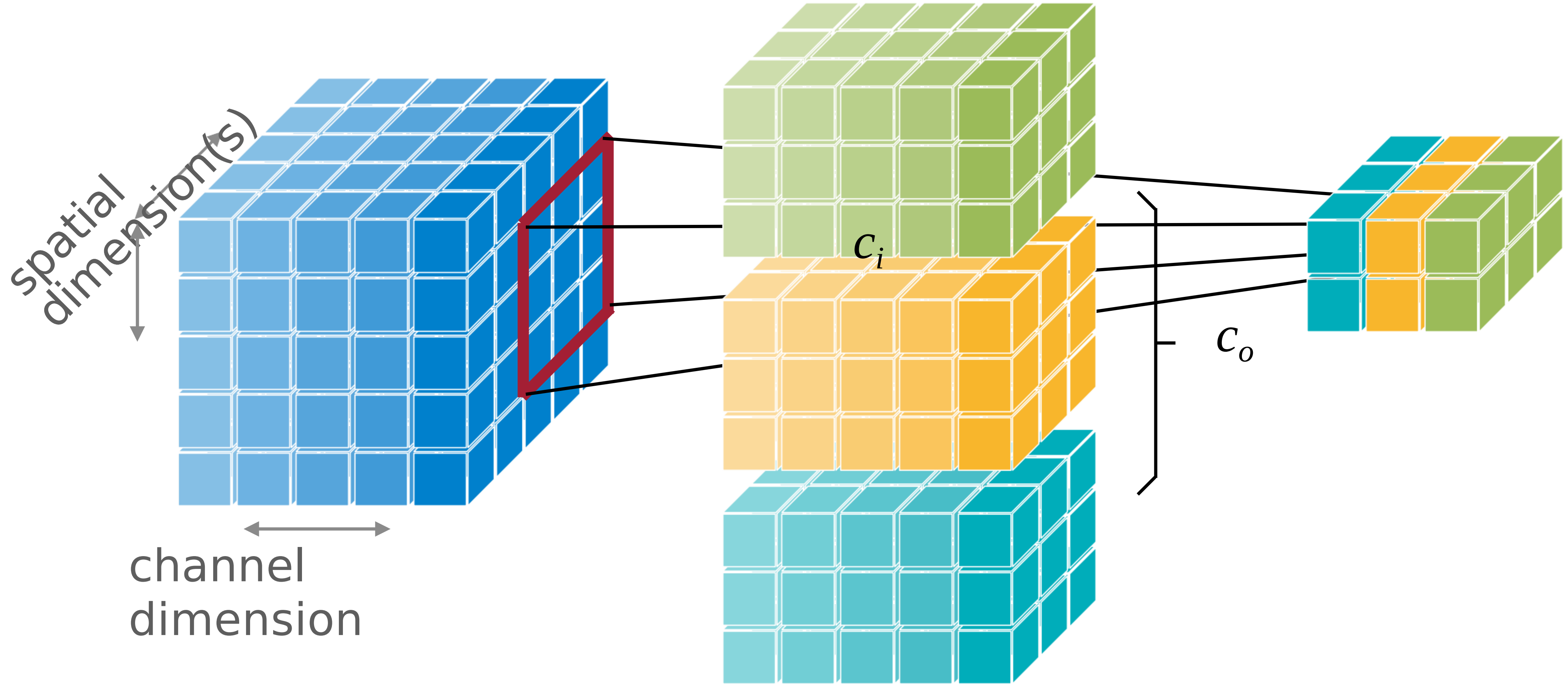
Convolution Layer

- **The output neuron is connected to input neurons in the receptive field**
  - **Shape of Tensors:**
    - Input Features $\mathbf{X}$: $\cancel{(n, c_i)}$
    - Output Features $\mathbf{Y}$: $\cancel{(n, c_o)}$
    - Weights $\mathbf{W}$: $\cancel{(c_o, c_i)}$
    - Bias $\mathbf{b}$: (c_o)

1D Conv	2D Conv
(n, c_i, w_i)	$(n\!=\!1, c_i, h_i, w_i)$
(n, c_o, w_o)	$(n\!=\!1, c_o, h_o, w_o)$
(c_o, c_i, k_w)	(c_o, c_i, k_h, k_w)



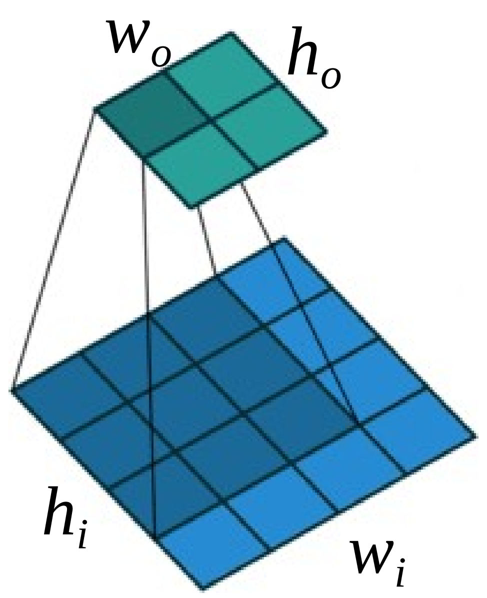
Notations	
n	Batch Size
c_i	Input Channels
c_o	Output Channels
w_i, w_o	Input/Output Width
h_i, h_o	Input/Output Height
k_h	Kernel Height
k_w	Kernel Width



Convolution Layer

- **The output neuron is connected to input neurons in the receptive field**
  - **Shape of Tensors:**
    - Input Features $\mathbf{X}$: $\cancel{(n, c_i)}$
    - Output Features $\mathbf{Y}$: $\cancel{(n, c_o)}$
    - Weights $\mathbf{W}$: $\cancel{(c_o, c_i)}$
    - Bias $\mathbf{b}$: (c_o)

1D Conv	**2D Conv**
(n, c_i, w_i)	$(n{=}1, c_i, h_{i,} w_i)$
(n, c_o, w_o)	$(n{=}1, c_o, h_o, w_o)$
(c_o, c_i, k_w)	(c_o, c_i, k_h, k_w)



Notations	
n	Batch Size
c_i	Input Channels
c_o	Output Channels
w_i, w_o	Input/Output Width
h_i, h_o	Input/Output Height
k_h	Kernel Height
k_w	Kernel Width

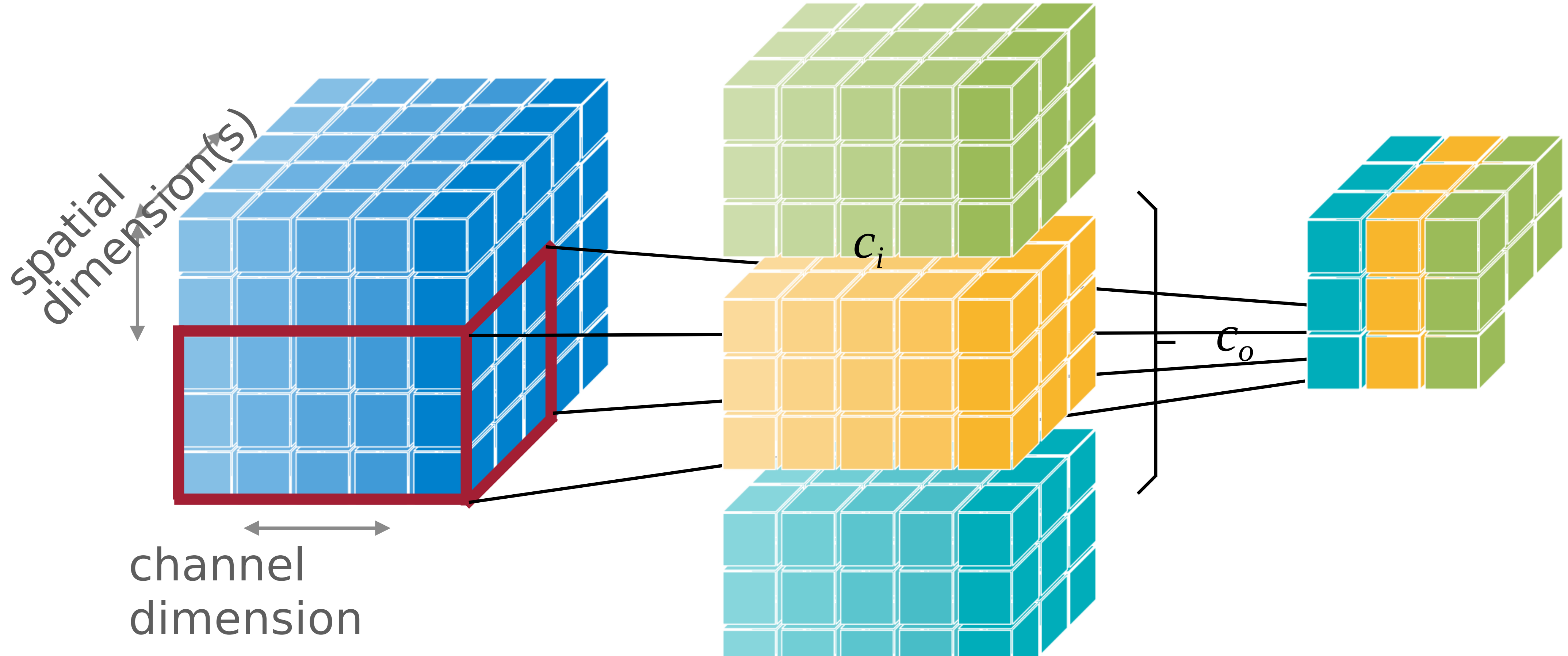
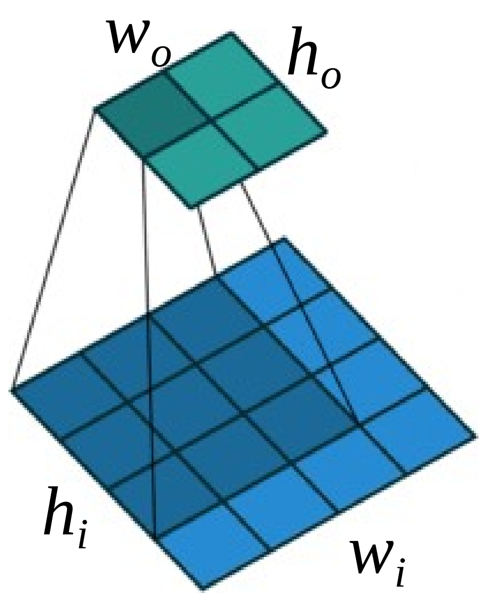


Image source: 1

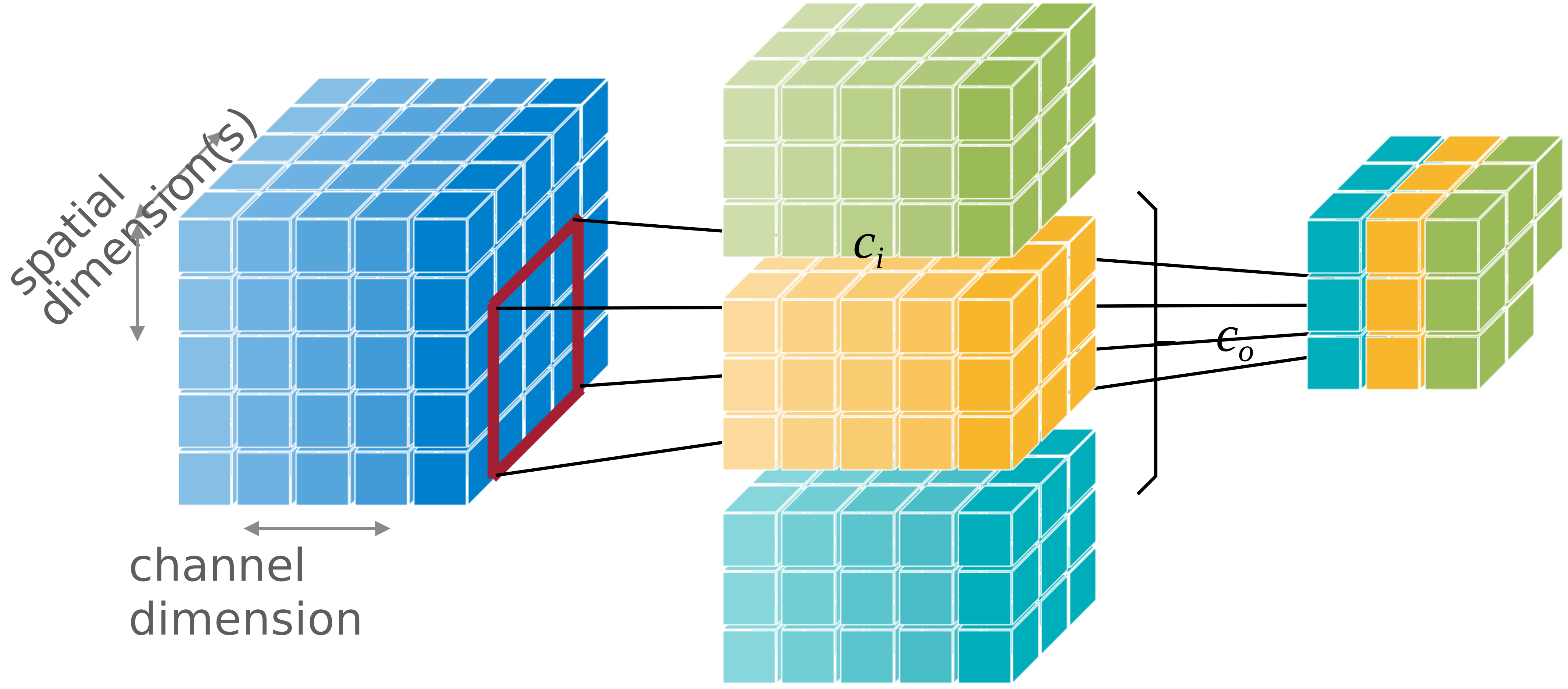
Convolution Layer

- **The output neuron is connected to input neurons in the receptive field**
  - **Shape of Tensors:**
    - Input Features $\mathbf{X}$: $\cancel{(n, c_i)}$
    - Output Features $\mathbf{Y}$: $\cancel{(n, c_o)}$
    - Weights $\mathbf{W}$: $\cancel{(c_o, c_i)}$
    - Bias $\mathbf{b}$: (c_o)

1D Conv	2D Conv
(n, c_i, w_i)	$(n{=}1, c_i, h_i, w_i)$
(n, c_o, w_o)	$(n{=}1, c_o, h_o, w_o)$
(c_o, c_i, k_w)	(c_o, c_i, k_h, k_w)



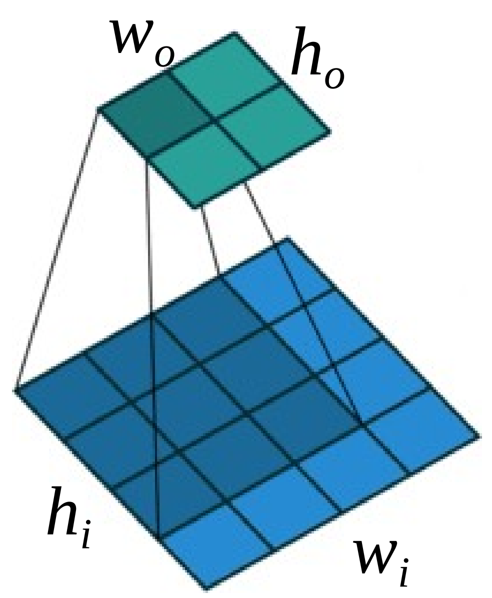
Notations	
n	Batch Size
c_i	Input Channels
c_o	Output Channels
w_i, w_o	Input/Output Width
h_i, h_o	Input/Output Height
k_h	Kernel Height
k_w	Kernel Width



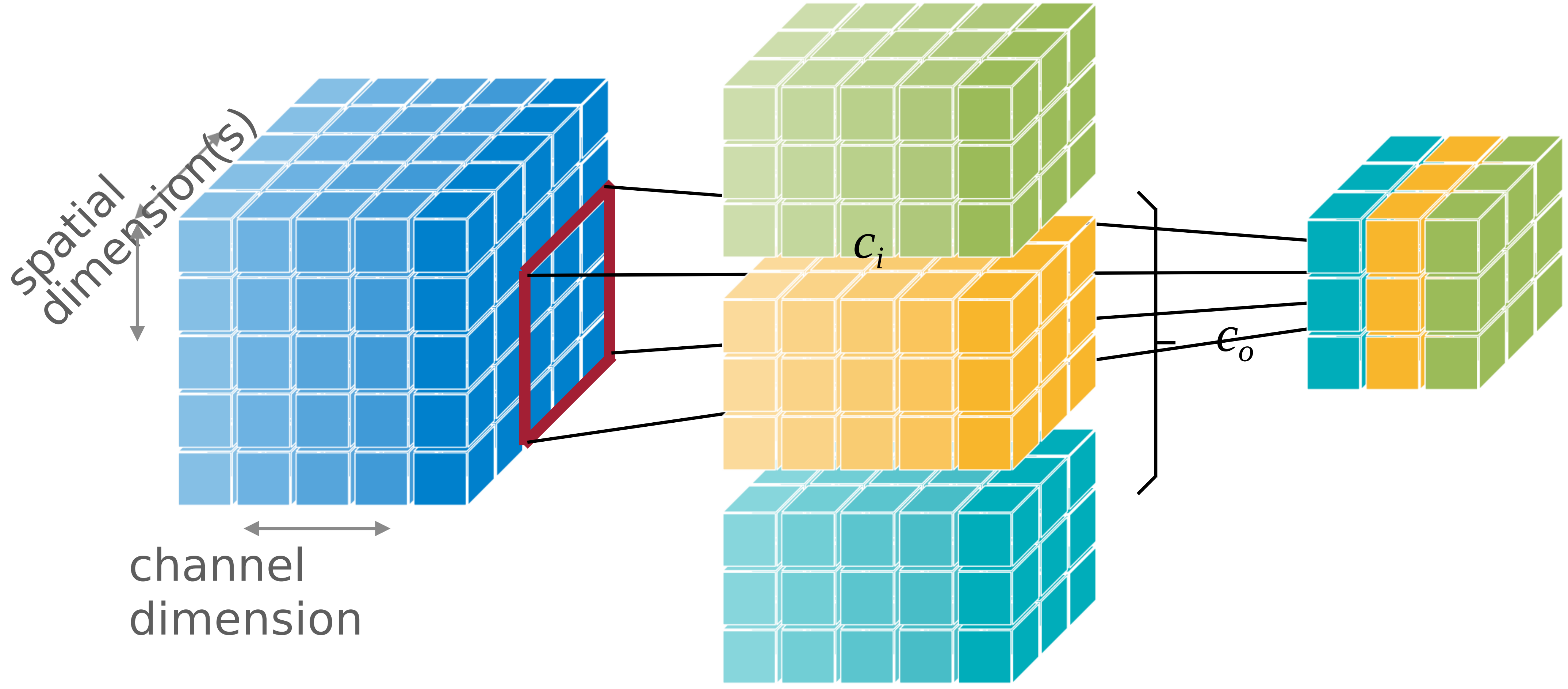
Convolution Layer

- **The output neuron is connected to input neurons in the receptive field**
  - **Shape of Tensors:**
    - Input Features $\mathbf{X}$: $\cancel{(n, c_i)}$
    - Output Features $\mathbf{Y}$: $\cancel{(n, c_o)}$
    - Weights $\mathbf{W}$: $\cancel{(c_o, c_i)}$
    - Bias $\mathbf{b}$: (c_o)

1D Conv	2D Conv
(n, c_i, w_i)	$(n{=}1, c_i, h_i, w_i)$
(n, c_o, w_o)	$(n{=}1, c_o, h_o, w_o)$
(c_o, c_i, k_w)	(c_o, c_i, k_h, k_w)



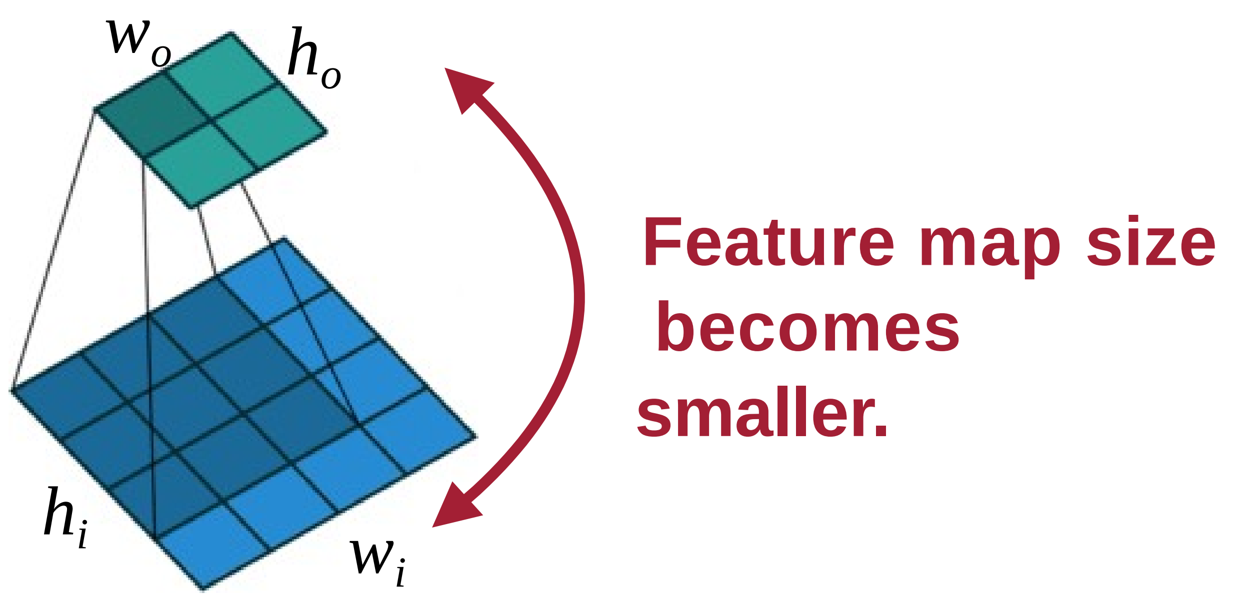
Notations	
n	Batch Size
c_i	Input Channels
c_o	Output Channels
w_i, w_o	Input/Output Width
h_i, h_o	Input/Output Height
k_h	Kernel Height
k_w	Kernel Width



Convolution Layer

- **The output neuron is connected to input neurons in the receptive field**
  - **Shape of Tensors:**
    - Input Features $\mathbf{X}$: $\cancel{(n, c_i)}$
    - Output Features $\mathbf{Y}$: $\cancel{(n, c_o)}$
    - Weights $\mathbf{W}$: $\cancel{(c_o, c_i)}$
    - Bias $\mathbf{b}$: (c_o)

1D Conv	2D Conv
(n, c_i, w_i)	$(n{=}1, c_i, h_i, w_i)$
(n, c_o, w_o)	$(n{=}1, c_o, h_o, w_o)$
(c_o, c_i, k_w)	(c_o, c_i, k_h, k_w)



Notations	
n	Batch Size
c_i	Input Channels
c_o	Output Channels
w_i, w_o	Input/Output Width
h_i, h_o	Input/Output Height
k_h	Kernel Height
k_w	Kernel Width

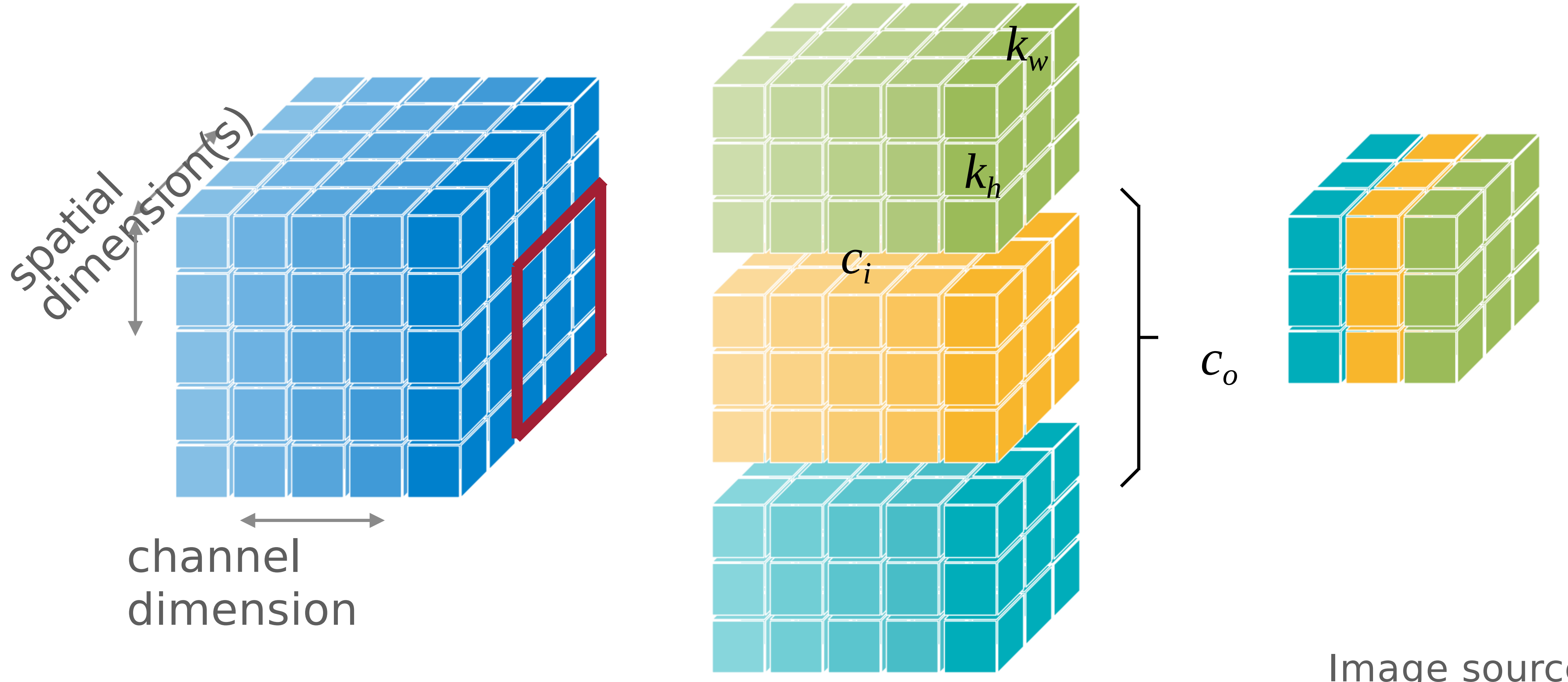
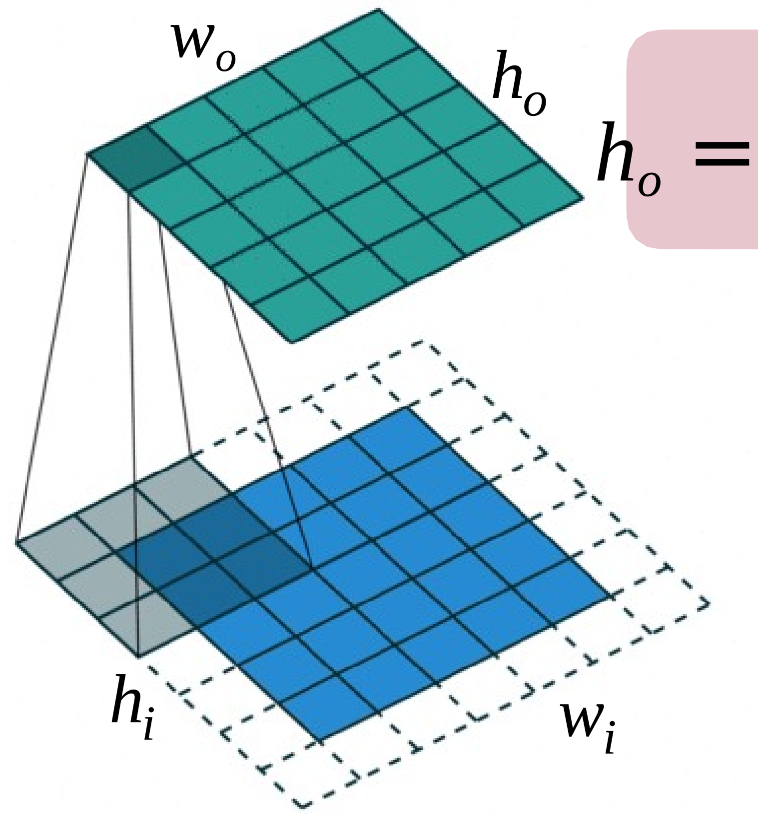


Image source: 1

Convolution Layer: Padding

- **Padding can be used to keep the output feature map size is the same as input feature map size**
  - **Zero Padding pads the input boundaries with zero.**
  - **<u>Other Paddings</u>: Reflection Padding, Replication Padding, Constant Padding**



$$h_o = h_i + 2p - k_h + 1$$

p is padding

$h_i = w_i = 5$

$k_h = k_w = 3$

$h_o = w_o$

$= 5 + 2 \times 1 - 3 + 1 = 5$

Notations	
n	Batch Size
c_i	Input Channels
c_o	Output Channels
w_i, w_o	Input/Output Width
h_i, h_o	Input/Output Height
k_h	Kernel Height
k_w	Kernel Width

Zero Padding

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	1	2	3	0	0
0	0	4	5	6	0	0
0	0	7	8	9	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Reflection Padding

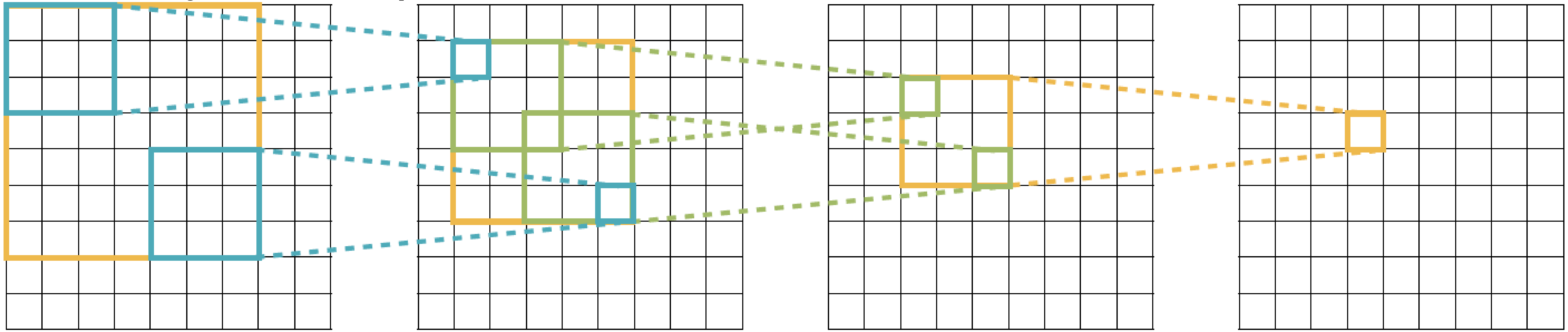
9	8	7	8	9	8	7
6	5	4	5	6	4	5
3	2	1	2	3	2	1
6	5	4	5	6	5	4
9	8	7	8	9	8	7
6	5	4	5	6	5	4
3	2	1	2	3	2	1

Replication Padding

1	1	1	2	3	3	3
1	1	1	2	3	3	3
1	1	1	2	3	3	3
4	4	4	5	6	6	6
7	7	7	8	9	9	9
7	7	7	8	9	9	9
7	7	7	8	9	9	9

Convolution Layer: Receptive Field

- In convolution, each output element depends on $k_h k_w$ receptive field in the input
- Each successive convolution adds k1 to the receptive field size
- With L layers, the receptive field size is



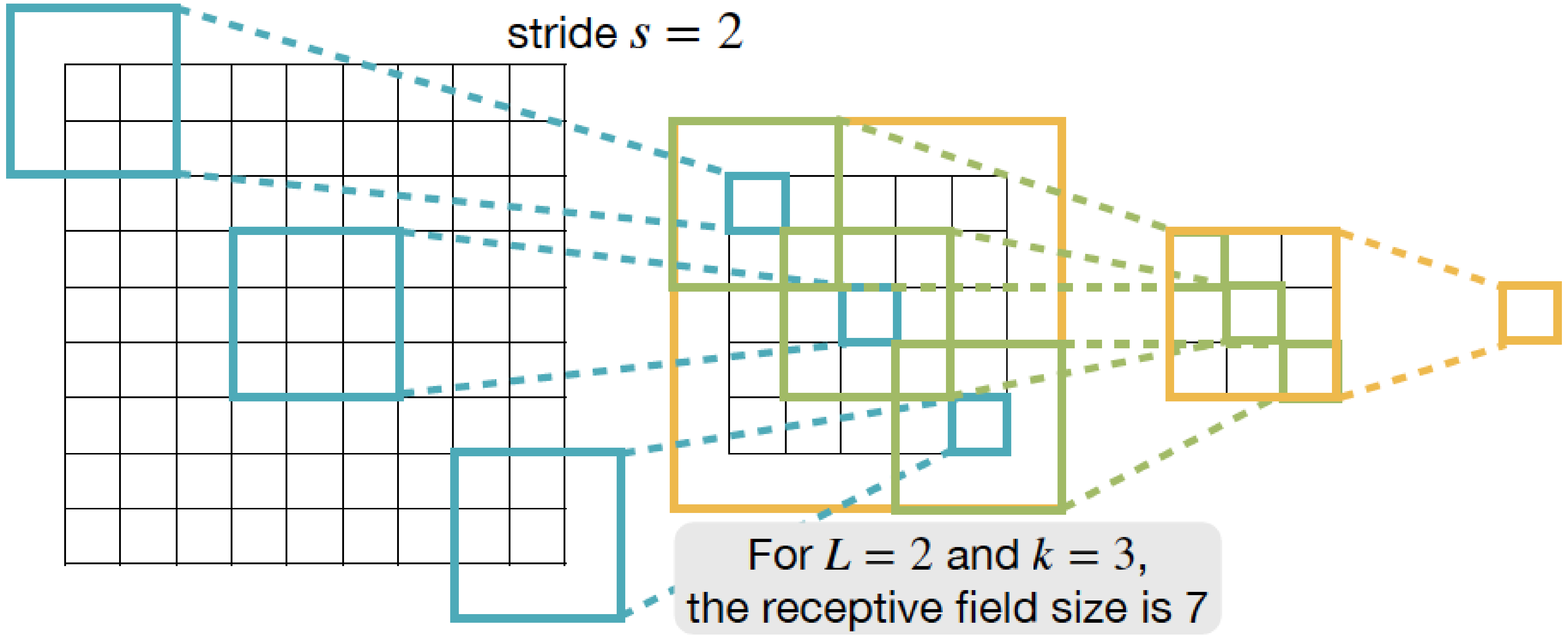
For $L = 2$ and $k = 3$, the receptive field size is **5**

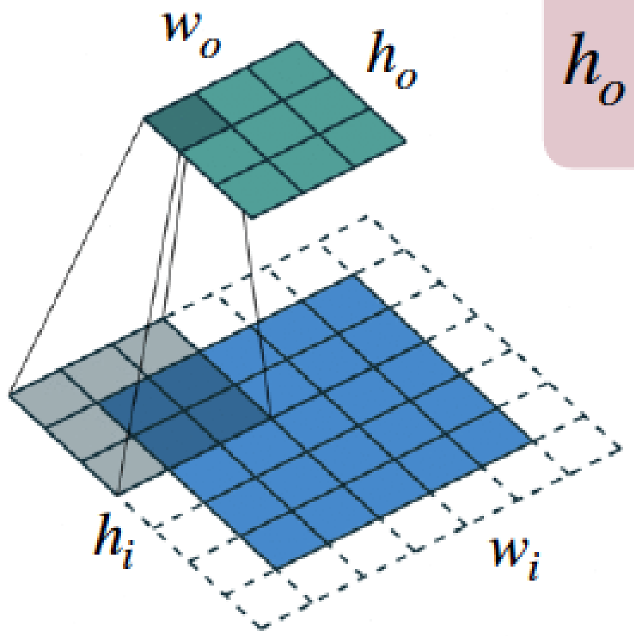
For $L = 3$ and $k = 3$, the receptive field size is **7**

Problem: For large images, we need many layers for each output to “see” the whole image

Solution: Downsample inside the neural network

Strided Convolution Layer

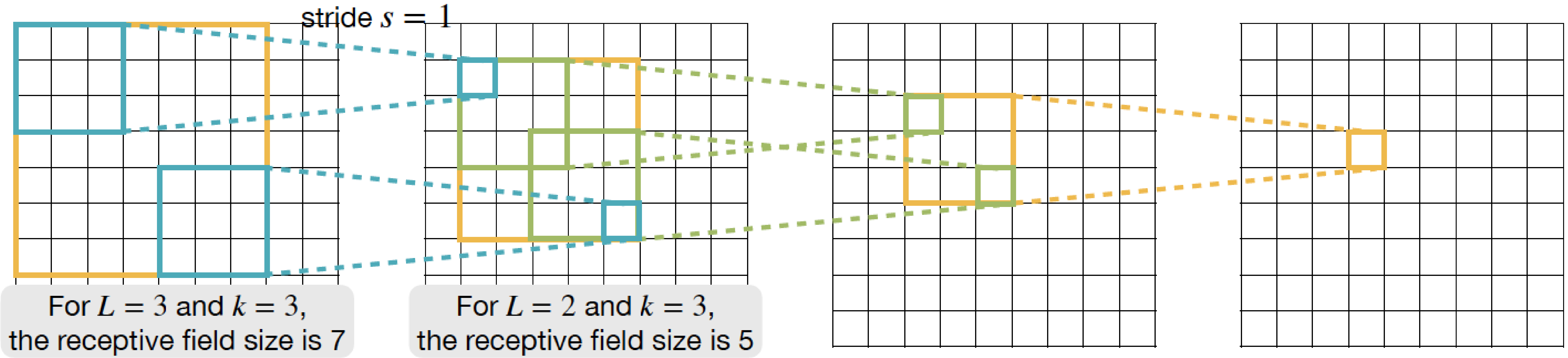




$$h_o = \frac{h_i + 2p - k_h}{s} + 1$$

p is padding

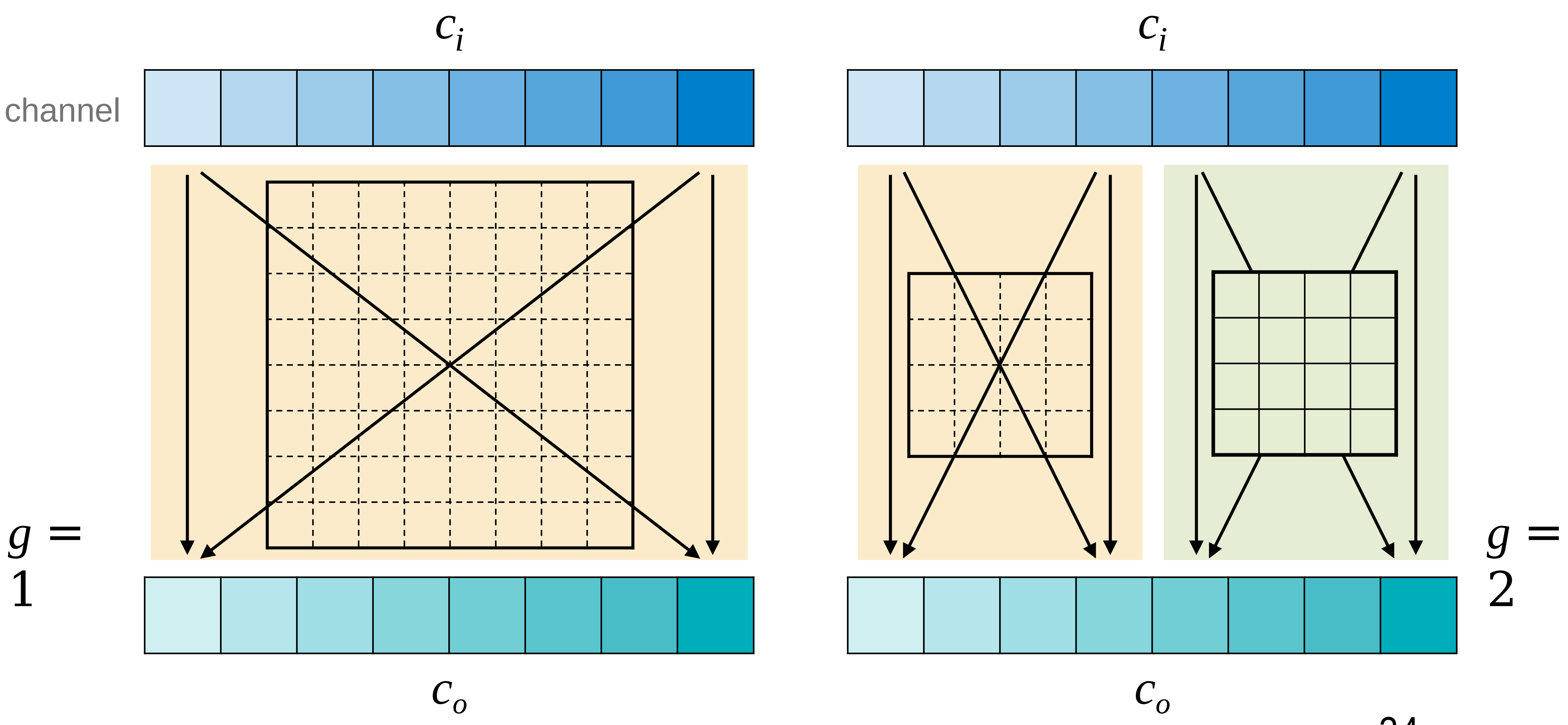
s is stride



Grouped Convolution Layer

- **A group of narrower convolutions**
  - **Shape of Tensors:**
    - Input Features $\mathbf{X}$: (n, c_i, h_i, w_i)
    - Output Features $\mathbf{Y}$: (n, c_o, h_o, w_o)
    - Weights $\mathbf{W}$: $\cancel{(c_o, c_i, k_h, k_w)}$ $(g \cdot c_o/g, c_i/g, k_h, k_w)$
    - Bias $\mathbf{b}$: (c_o)

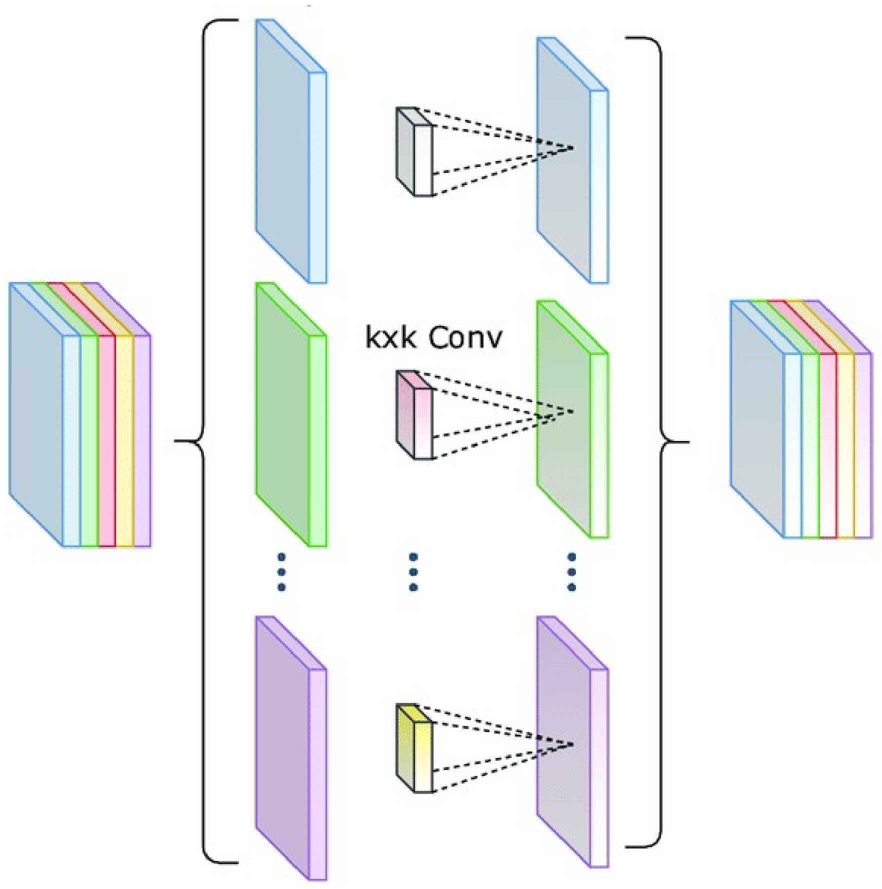
Notations	
n	Batch Size
c_i	Input Channels
c_o	Output Channels
w_i, w_o	Input/Output Width
h_i, h_o	Input/Output Height
k_h	Kernel Height
k_w	Kernel Width
g	Groups

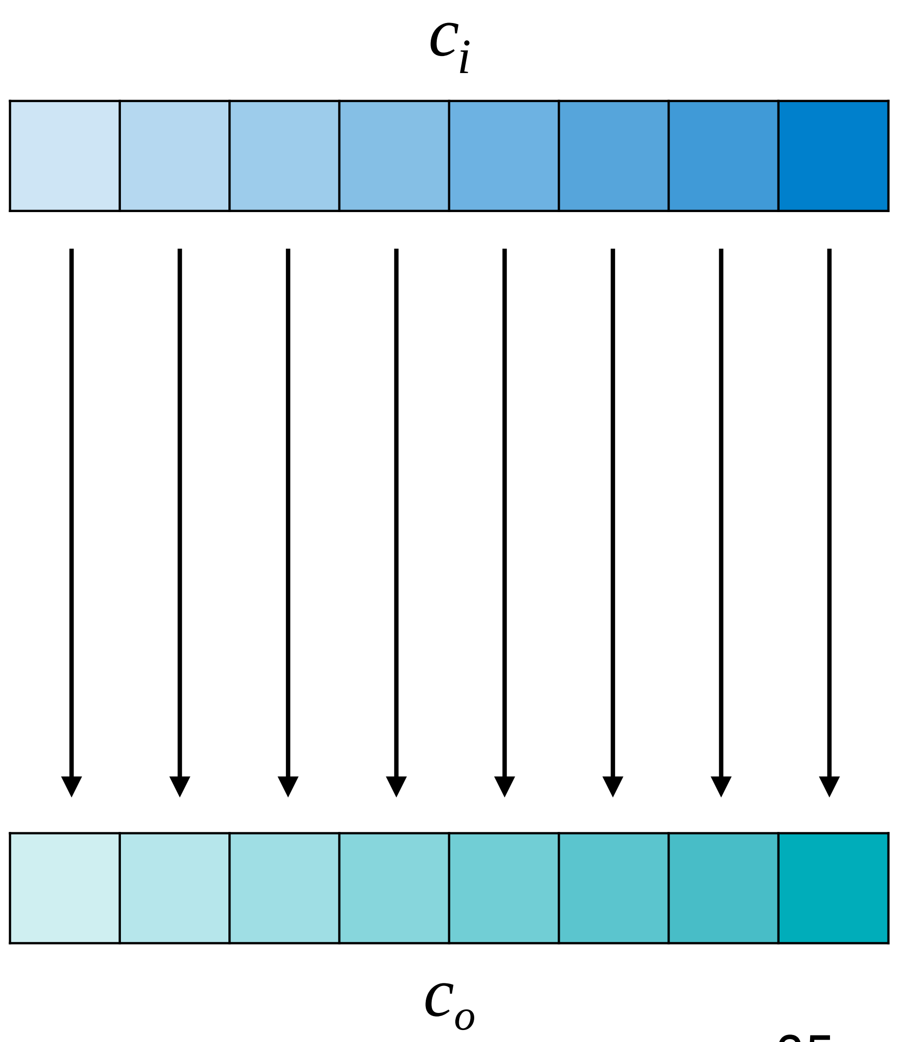


Depthwise Convolution Layer

- **Independent filter for each channel: $g = c_i = c_o$ in grouped convolution**
  - **Shape of Tensors:**
    - Input Features $\mathbf{X}$: $(n, c_i, h_{i,} w_i)$
    - Output Features $\mathbf{Y}$: (n, c_o, h_o, w_o)
    - Weights $\mathbf{W}$: $\cancel{(c_o, c_i, k_h, k_w)}$ (c_i, k_h, k_w)
    - Bias $\mathbf{b}$: (c_o)

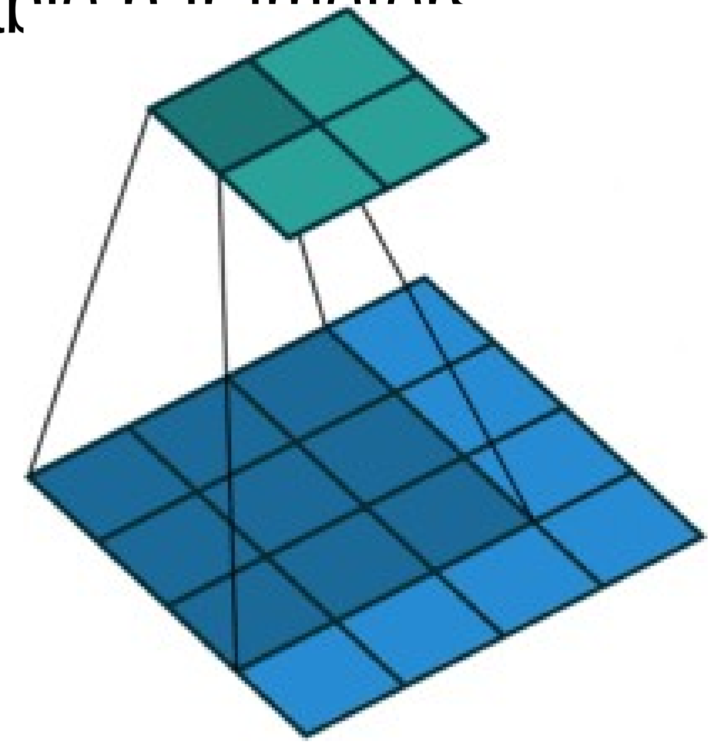
Notations	
n	Batch Size
c_i	Input Channels
c_o	Output Channels
w_i, w_o	Input/Output Width
h_i, h_o	Input/Output Height
k_h	Kernel Height
k_w	Kernel Width
g	Groups





Pooling Layer

- **Downsample the feature map to a smaller size**
  - **The output neuron pools the features In the receptive field, similar to convolution.**
    - Usually, the stride is the same as the kernel size: $s = k$
  - **Pooling operates over each channel independently**
    - No learnable parameters



stride = kernel size = 2

5	0	1	7
2	1	3	5
0	2	3	1
2	8	1	3

Input Feature Map

Max Pooling

5	7
8	3

Average Pooling

2	4
3	2

Output Feature Map

Normalization Layer

- **Normalizing the features makes optimization faster.**
  - **Normalization layer normalizes the features as follows,**

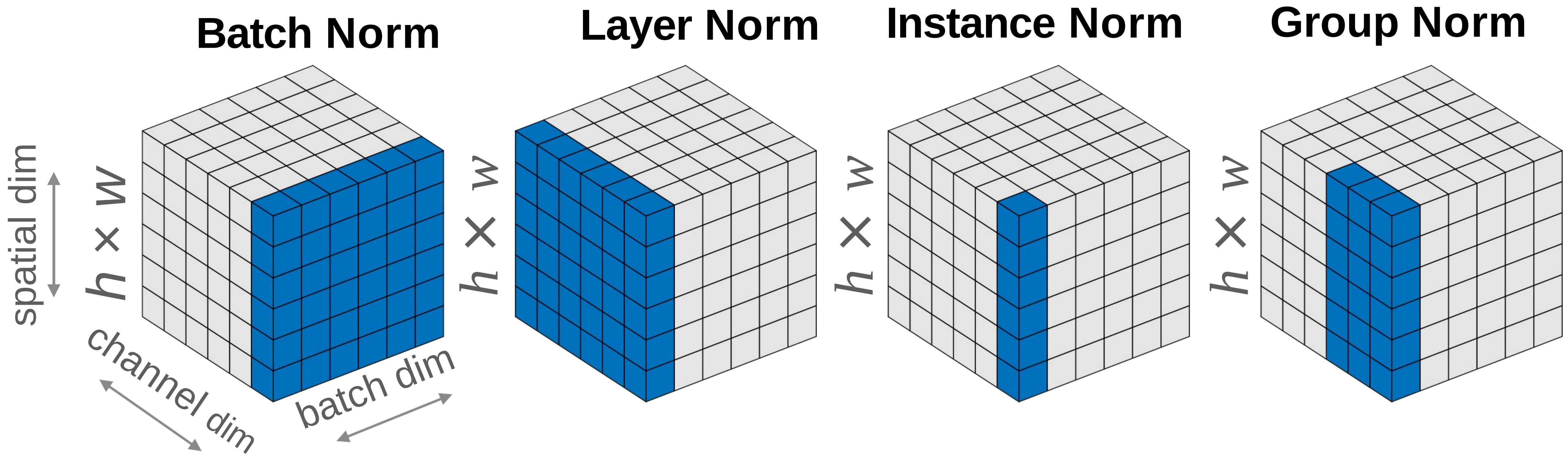
$$\mu_i = \frac{1}{m} \sum_{k \in \mathcal{S}_i} x_k$$

$$\sigma_i = \sqrt{\frac{1}{m} \sum_{k \in \mathcal{S}_i} (x_k - \mu_i)^2 + \epsilon}$$

$$\hat{x}_i = \frac{1}{\sigma}(x_i - \mu_i)$$

    - is the mean, and is the standard deviation (std) over the set of pixels $\mathcal{S}_I$
  - **Then learns a per-channel linear transform to compensate for the possible lost of representational ability**

$$y = \gamma_{i_c} \hat{x}_i + \beta_{i_c}$$

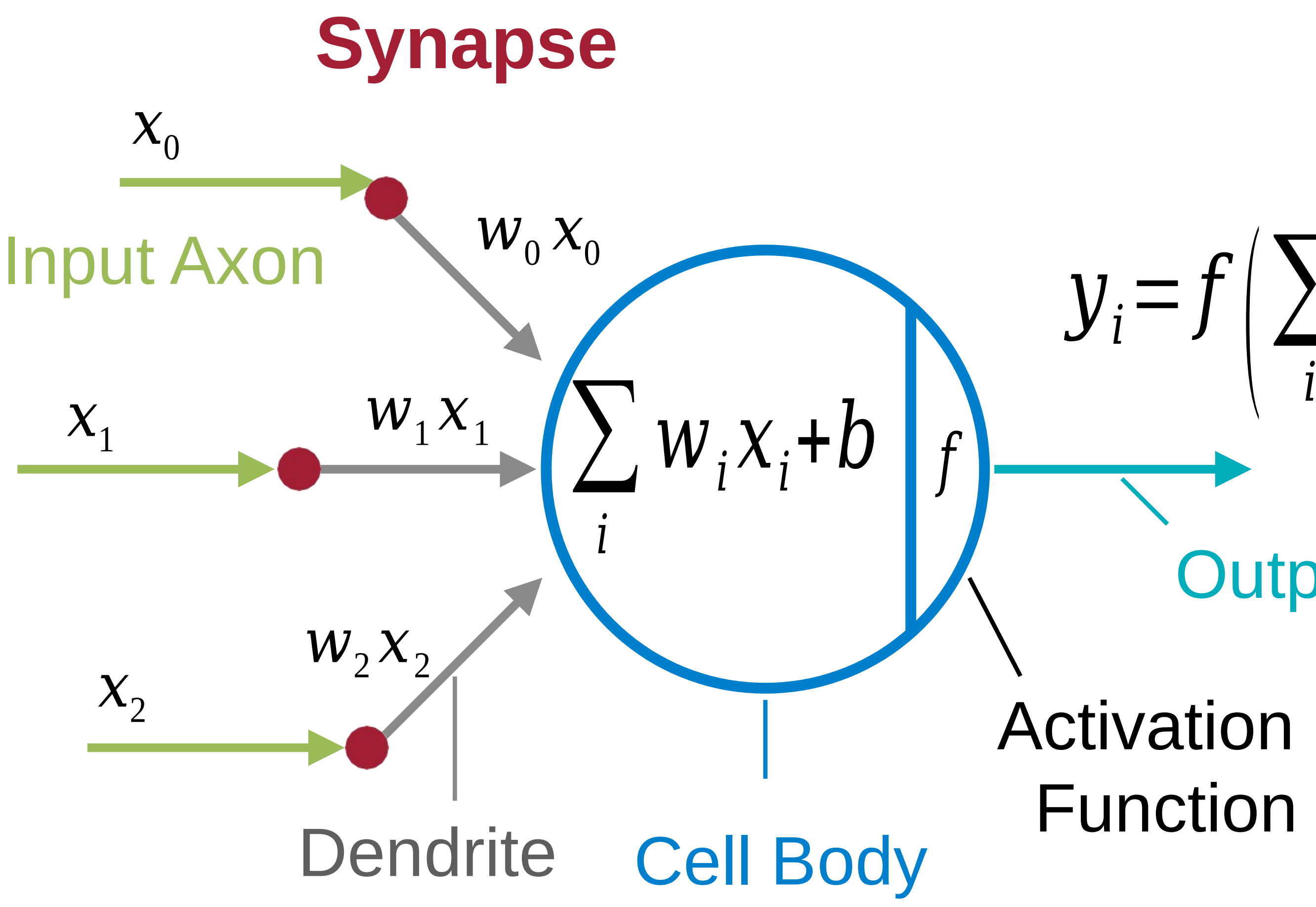


Different normalizations use different definitions of the set $\mathcal{S}_i$

(color in the blue)

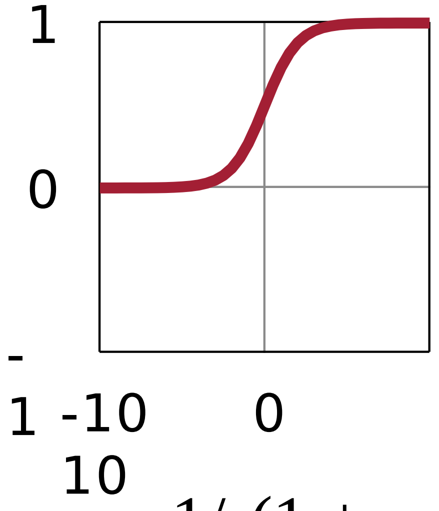
Activation Function

- **Activation functions are typically non-linear functions**



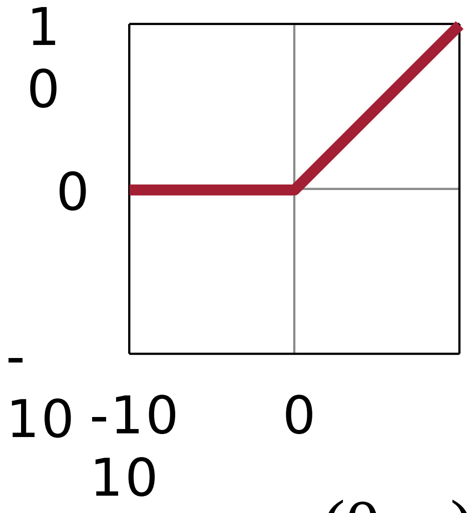
$$y_i = f\left(\sum_i w_i x_i + b\right)$$

Sigmoid



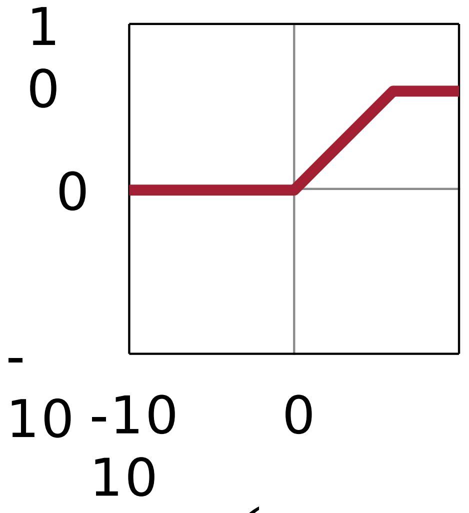
$y = 1/(1 + e^{-x})$

ReLU



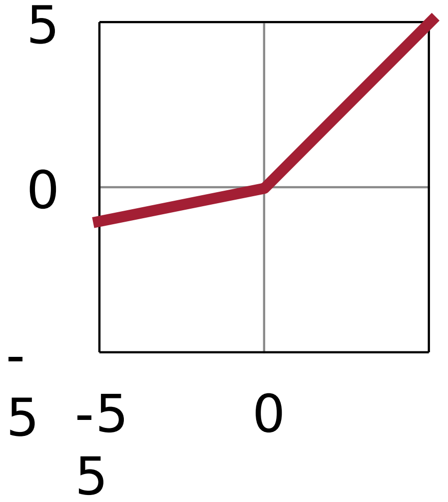
$y = \max(0, x)$

ReLU6



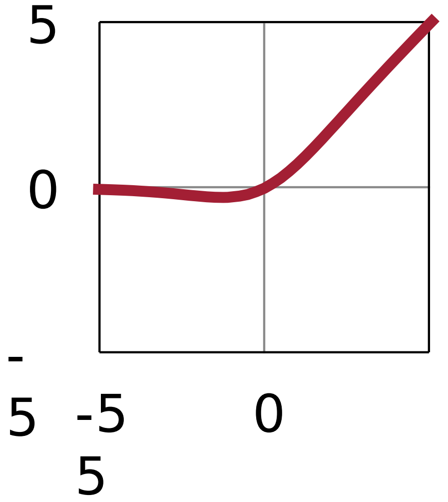
$y = \min(\max(0, x), 6)$

Leaky ReLU



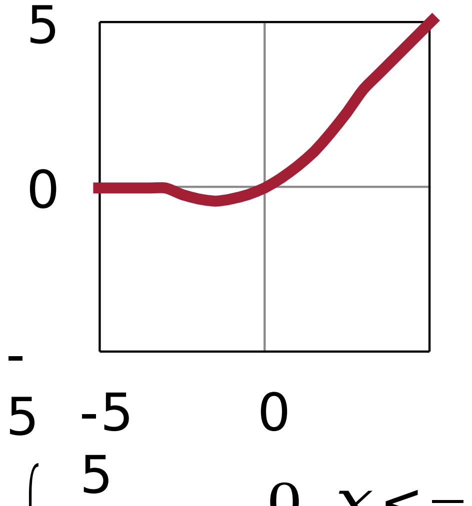
$y = \max(x, x)$

Swish



$y = x/(1 + e^{-x})$

Hard Swish



$$y = \begin{cases} 0, x < -3 \\ x, x \geq 3 \\ x \cdot (x+3)/6, otherwise \end{cases}$$

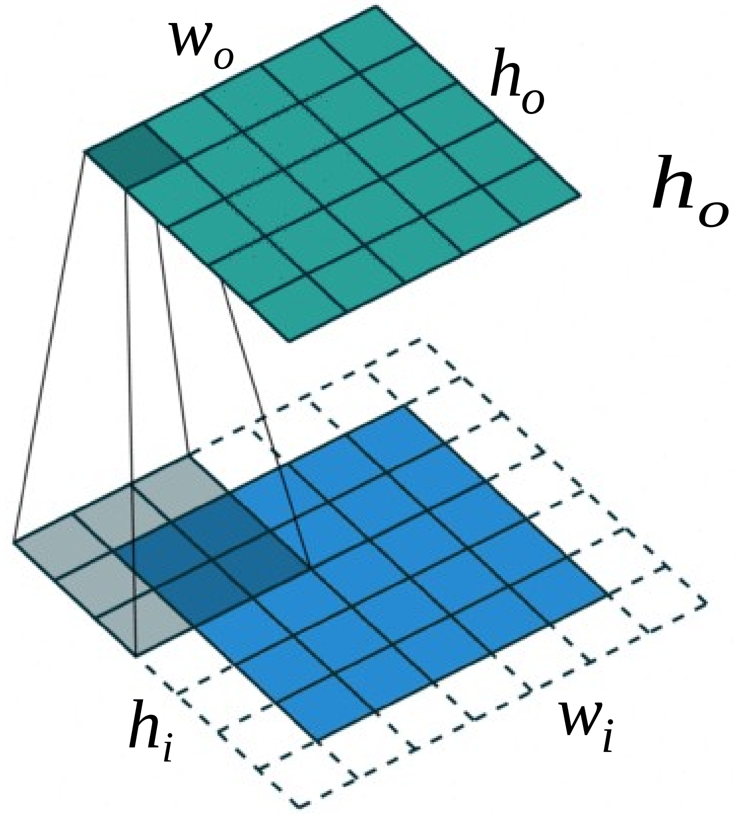
Other Activation Functions: Tanh, GELU, ELU, Mish…

Popular Neural Network

AlexNet

AlexNet	C × H × W	H, W
Image (3×224×224)	3×224×224	
11×11 Conv, channel 96, stride 4, pad 2	96×55×55	$\frac{224+2\times2-11}{4}+1=55$
3x3 MaxPool, stride 2	96×27×27	$\frac{55+0-3}{2}+1=27$
5x5 Conv, channel 256, pad 2, groups 2	256×27×27	$\frac{27+2\times2-5}{1}+1=27$
3x3 MaxPool, stride 2	256×13×13	$\frac{27+0-3}{2}+1=13$
3x3 Conv, channel 384, pad 1	384×13×13	$\frac{13+2\times1-3}{1}+1=13$
3x3 Conv, channel 384, pad 1, groups 2	384×13×13	$\frac{13+2\times1-3}{1}+1=13$
3x3 Conv, channel 256, pad 1, groups 2	256×13×13	$\frac{13+2\times1-3}{1}+1=13$
3x3 MaxPool, stride 2	256x6x6	$\frac{13+0-3}{2}+1=6$
Linear, channel 4096	4096	
Linear, channel 4096	4096	
Linear, channel 1000	1000	

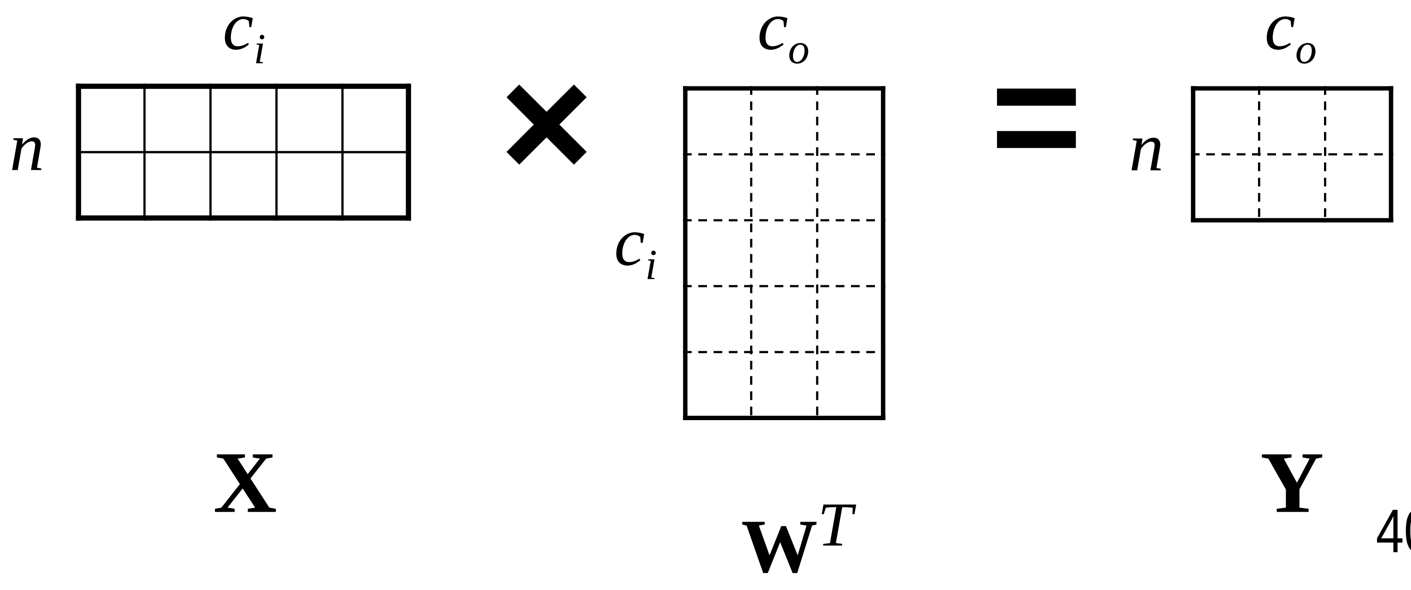
Convolution Layer / Pooling Layer



$$h_o=\frac{h_i+2p-k_h}{s}+1$$

p is padding
s is stride

Linear Layer



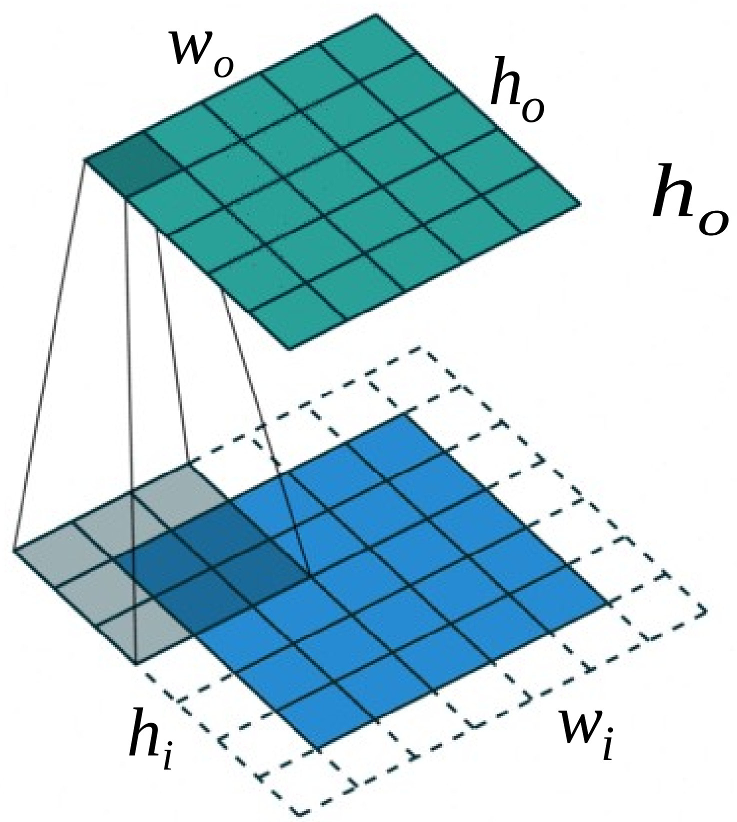
VGG 16

AlexNet
Image (3×224×224)
11×11 Conv, channel 96, stride 4, pad 2
3x3 MaxPool, stride 2
5x5 Conv, channel 256, pad 2, groups 2
3x3 MaxPool, stride 2
3x3 Conv, channel 384, pad 1
3x3 Conv, channel 384, pad 1, groups 2
3x3 Conv, channel 256, pad 1, groups 2
3x3 MaxPool, stride 2
Linear, channel 4096
Linear, channel 4096
Linear, channel 1000

	VGG-16
	Image (3×224×224)
1	3x3 Conv, channel 64, pad 1
2	3x3 Conv, channel 64, pad 1
	2x2 MaxPool, stride 2
3	3x3 Conv, channel 128, pad 1
4	3x3 Conv, channel 128, pad 1
	2x2 MaxPool, stride 2
5	3x3 Conv, channel 256, pad 1
6	3x3 Conv, channel 256, pad 1
7	3x3 Conv, channel 256, pad 1
	2x2 MaxPool, stride 2
8	3x3 Conv, channel 512, pad 1
9	3x3 Conv, channel 512, pad 1
10	3x3 Conv, channel 512, pad 1
	2x2 MaxPool, stride 2
11	3x3 Conv, channel 512, pad 1
12	3x3 Conv, channel 512, pad 1
13	3x3 Conv, channel 512, pad 1
	2x2 MaxPool, stride 2
14	Linear, channel 4096
15	Linear, channel 4096
16	Linear, channel 1000

Conv
BatchNorm
ReLU

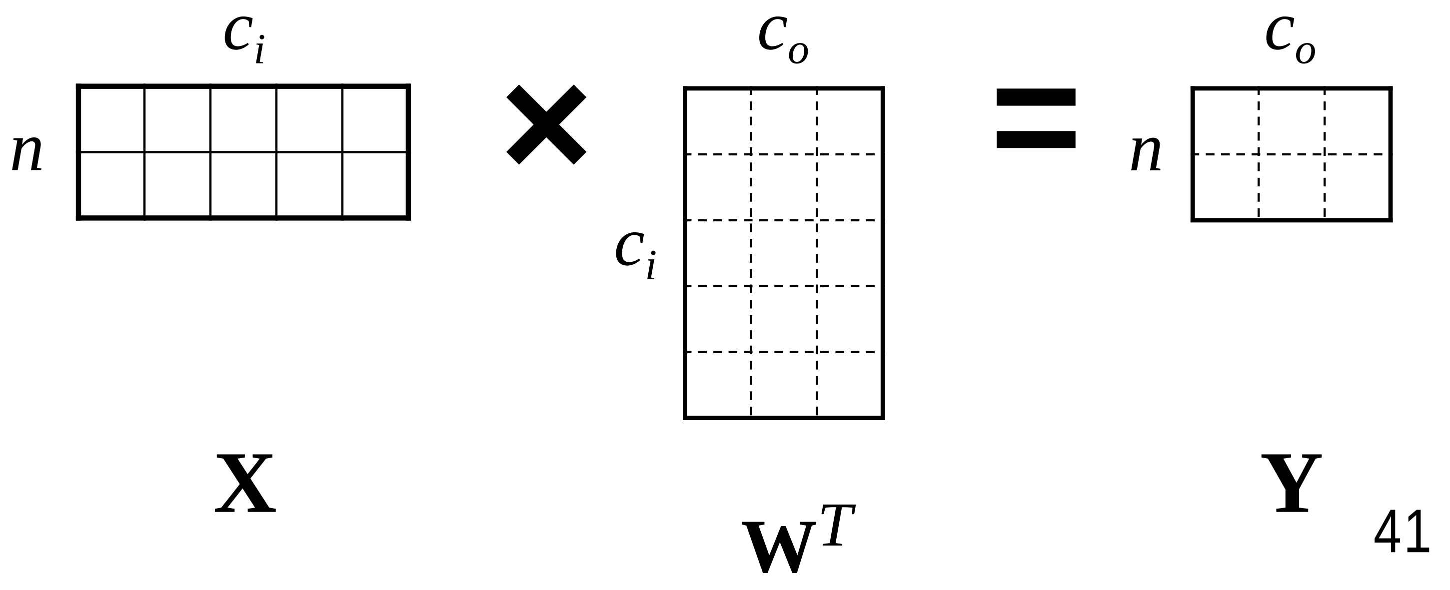
Convolution Layer / Pooling Layer



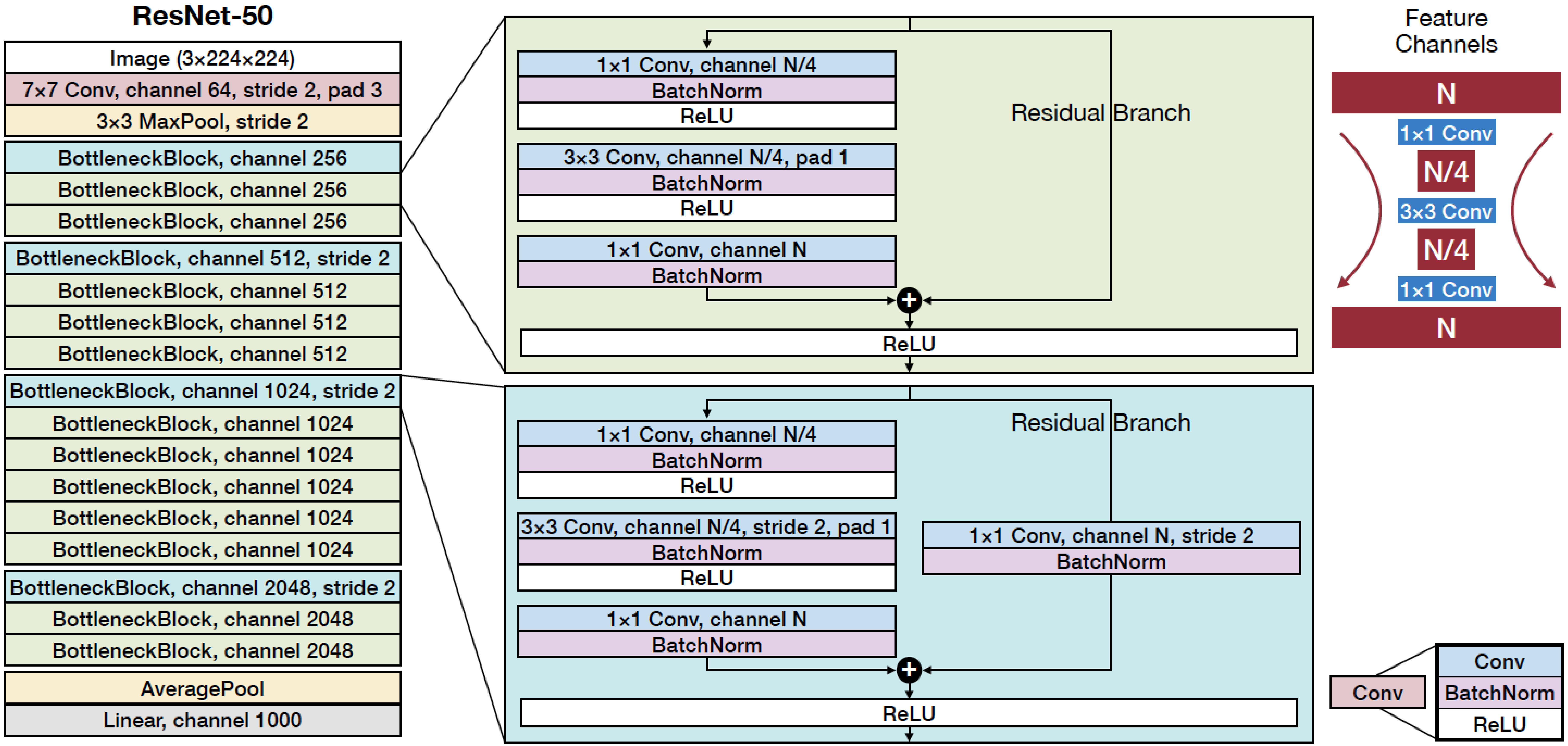
$$h_o = \frac{h_i + 2p - k_h}{s} + 1$$

p is padding
s is stride

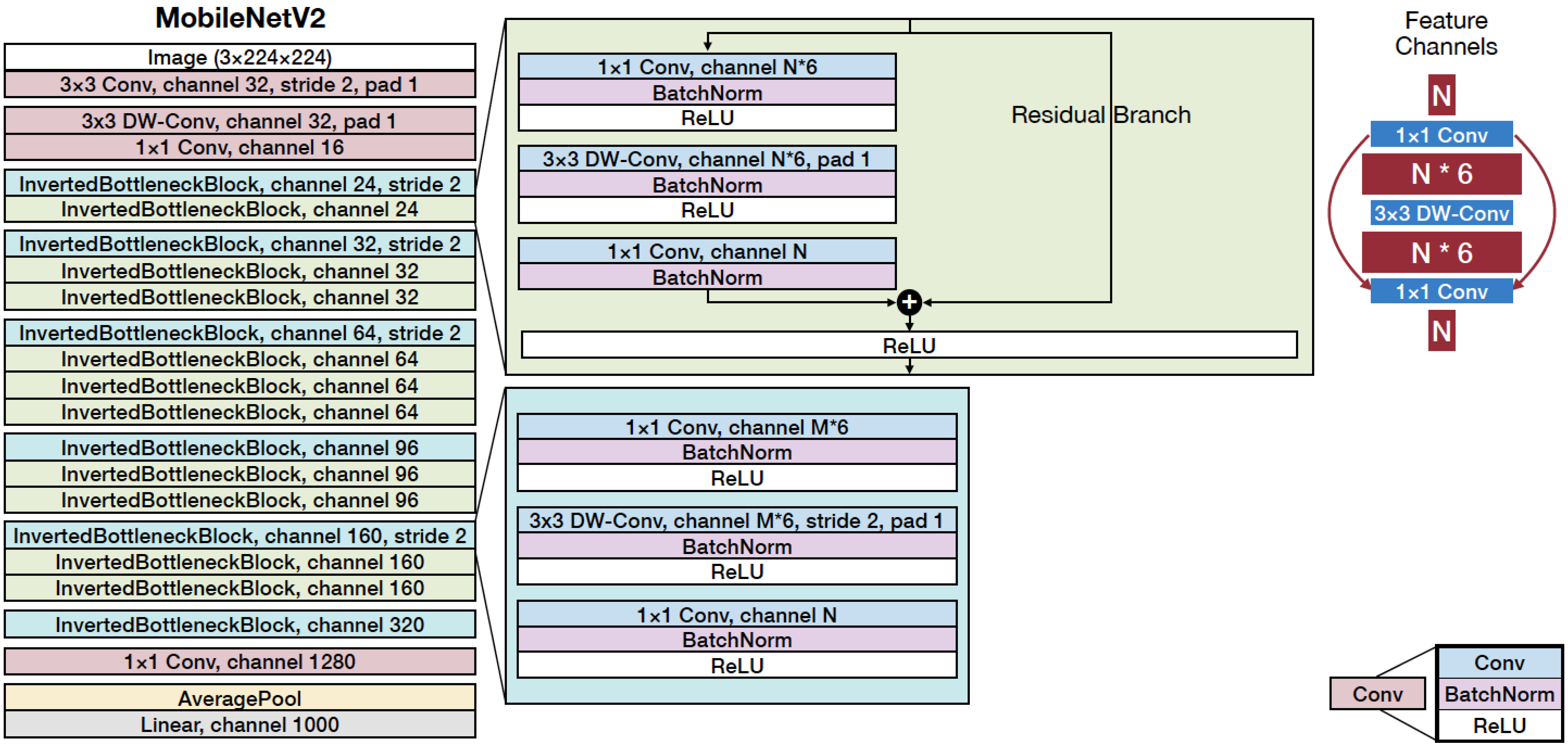
Linear Layer



ResNet-50


ResNet-50
Image (3×224×224)
7×7 Conv, channel 64, stride 2, pad 3
3×3 MaxPool, stride 2
BottleneckBlock, channel 256
BottleneckBlock, channel 256
BottleneckBlock, channel 256
BottleneckBlock, channel 512, stride 2
BottleneckBlock, channel 512
BottleneckBlock, channel 512
BottleneckBlock, channel 512
BottleneckBlock, channel 1024, stride 2
BottleneckBlock, channel 1024
BottleneckBlock, channel 1024
BottleneckBlock, channel 1024
BottleneckBlock, channel 1024
BottleneckBlock, channel 1024
BottleneckBlock, channel 2048, stride 2
BottleneckBlock, channel 2048
BottleneckBlock, channel 2048
AveragePool
Linear, channel 1000
1×1 Conv, channel N/4
BatchNorm
ReLU
3×3 Conv, channel N/4, pad 1
BatchNorm
ReLU
1×1 Conv, channel N
BatchNorm
Residual Branch
ReLU
1×1 Conv, channel N/4
BatchNorm
ReLU
3×3 Conv, channel N/4, stride 2, pad 1
BatchNorm
ReLU
1×1 Conv, channel N
BatchNorm
Residual Branch
1×1 Conv, channel N, stride 2
BatchNorm
ReLU
Feature Channels
N
1×1 Conv
N/4
3×3 Conv
N/4
1×1 Conv
N
Conv
Conv
BatchNorm
ReLU

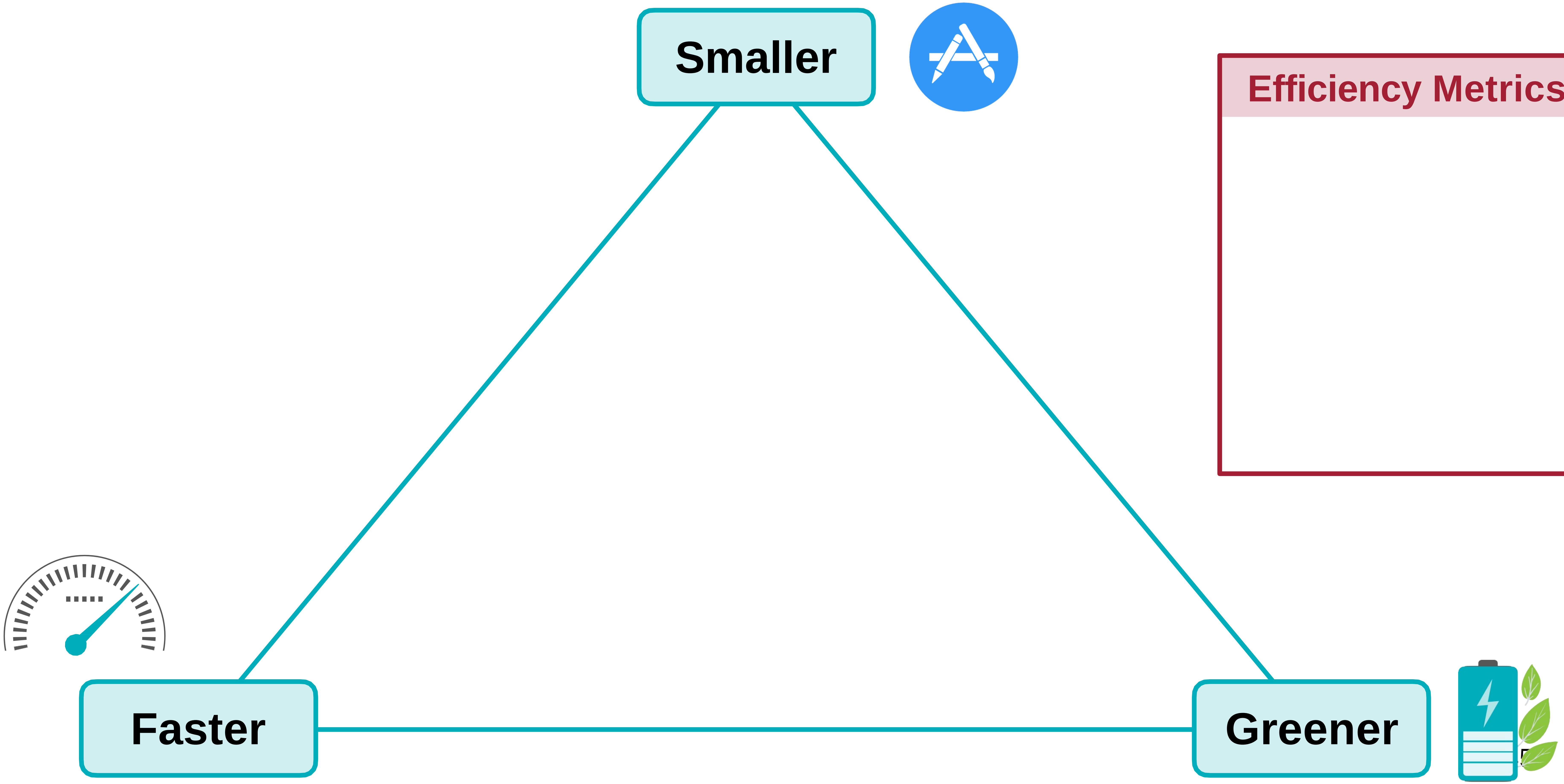
MobileNetV2



Efficiency Metrics

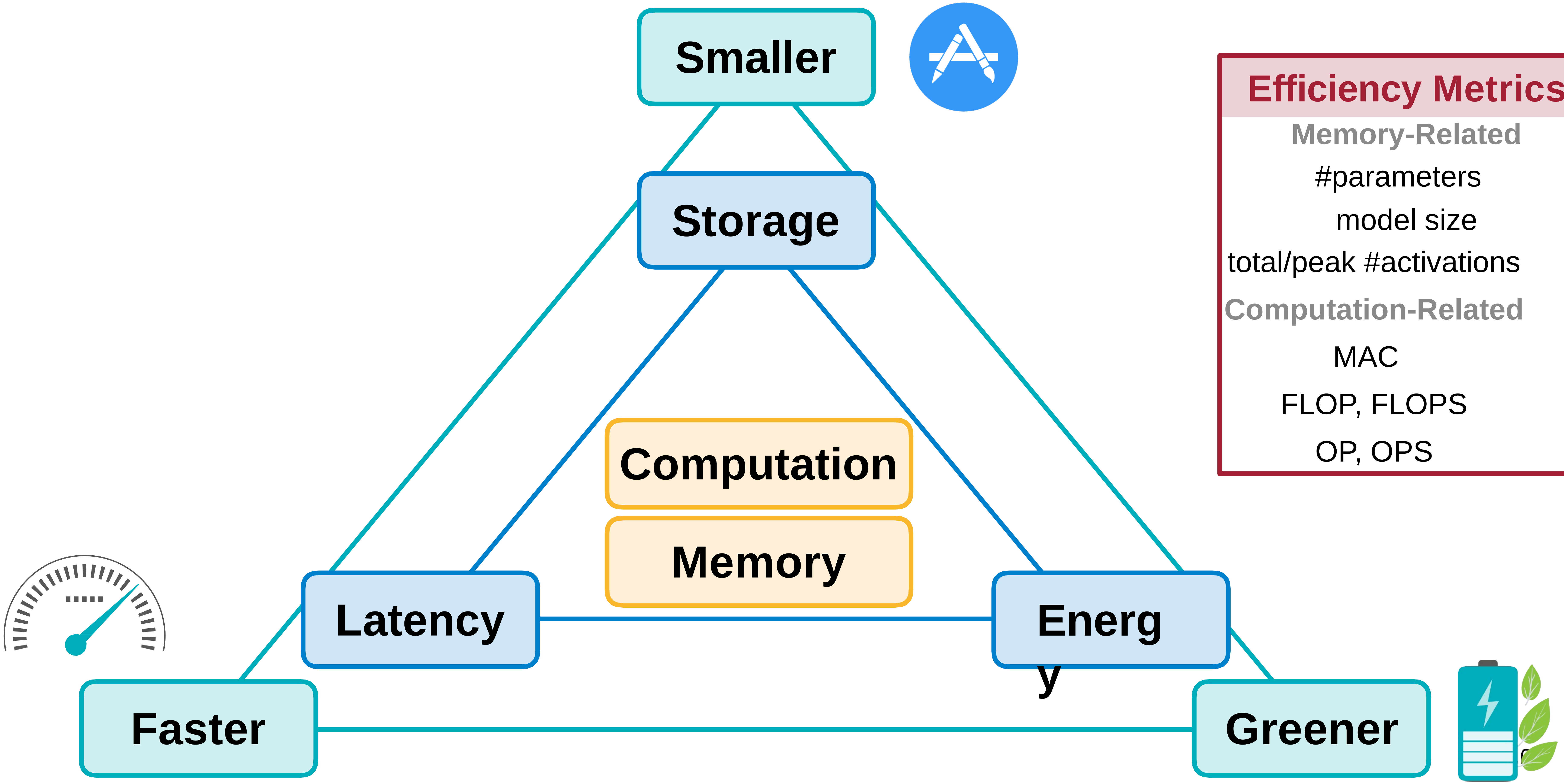
How should we measure the efficiency of neural networks?

Efficiency of Neural Networks



Efficiency Metrics

Efficiency of Neural Networks



Efficiency Metrics

Memory-Related

#parameters

model size

total/peak #activations

Computation-Related

MAC

FLOP, FLOPS

OP, OPS

Latency

- Measures the delay for a specific task

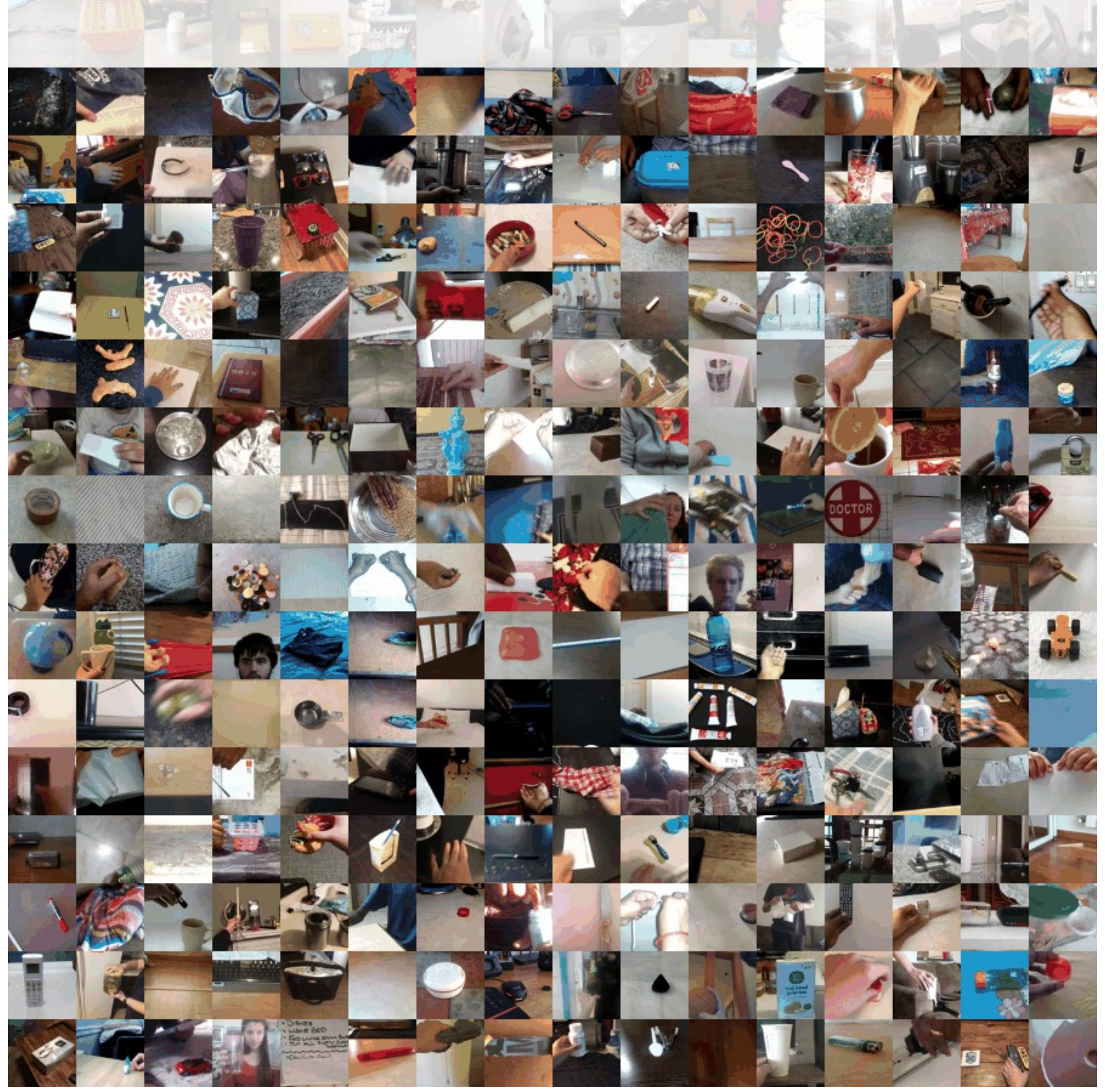
High Latency
638ms

Low Latency
46ms

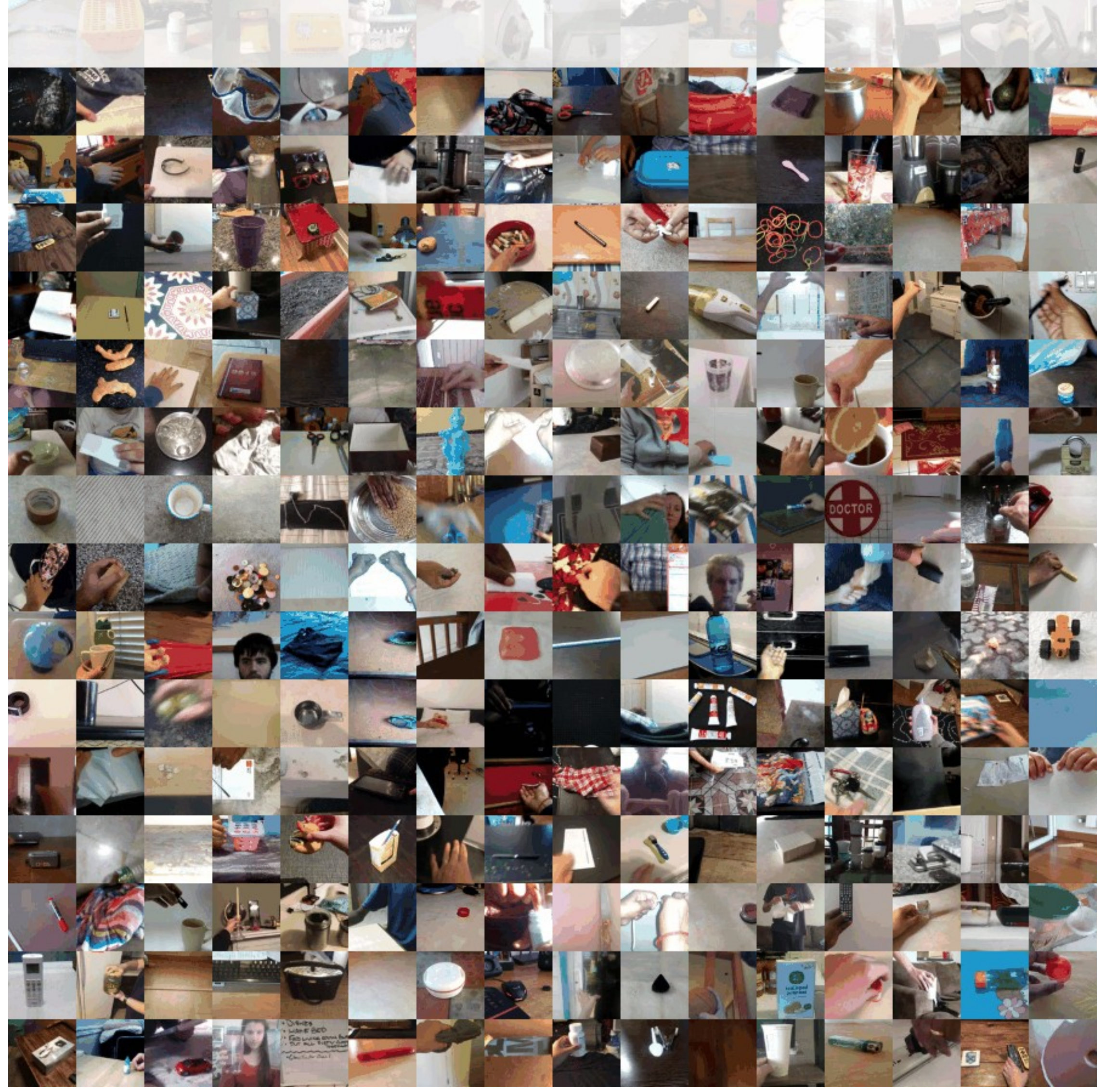
Speed is measured on Nvidia Jetson AGX Orin with TensorRT, fp16, batch size 1.

Throughput

- Measures the rate at which data is processed



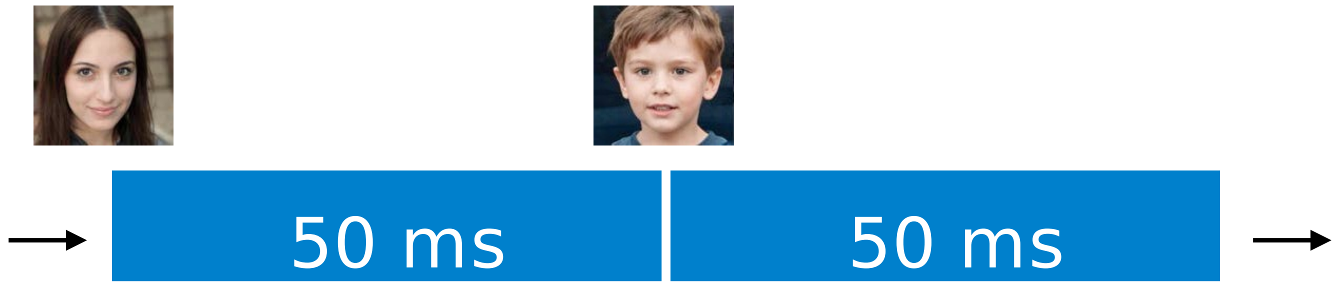
Low Throughput = 6.1 video/s



High Throughput = 77.4 video/s

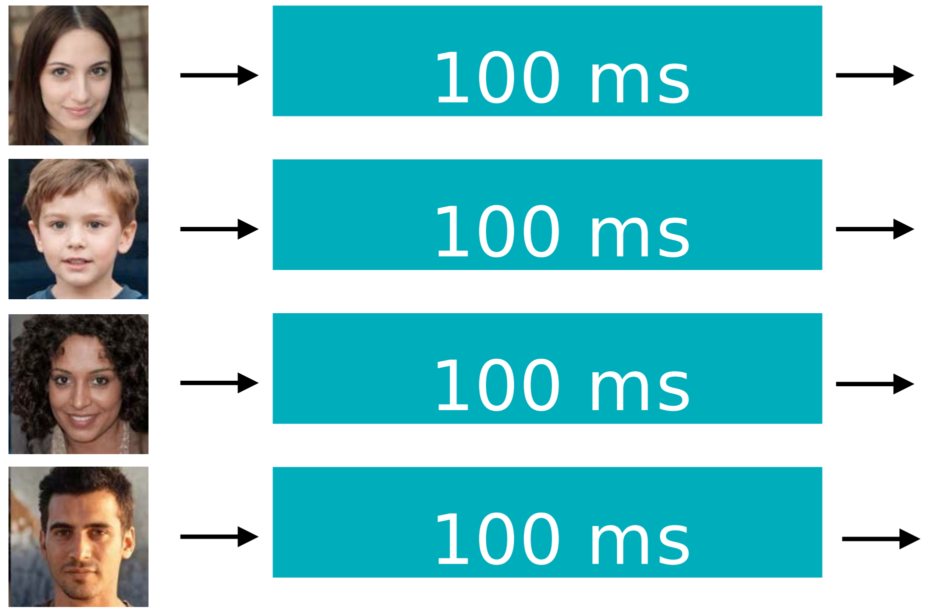
Latency vs. Throughput

- Does higher throughput translate to lower latency? Why?
- Does lower latency translate to higher throughput? Why?



Design 1

Latency: 50 ms
Throughput: 20 image/s

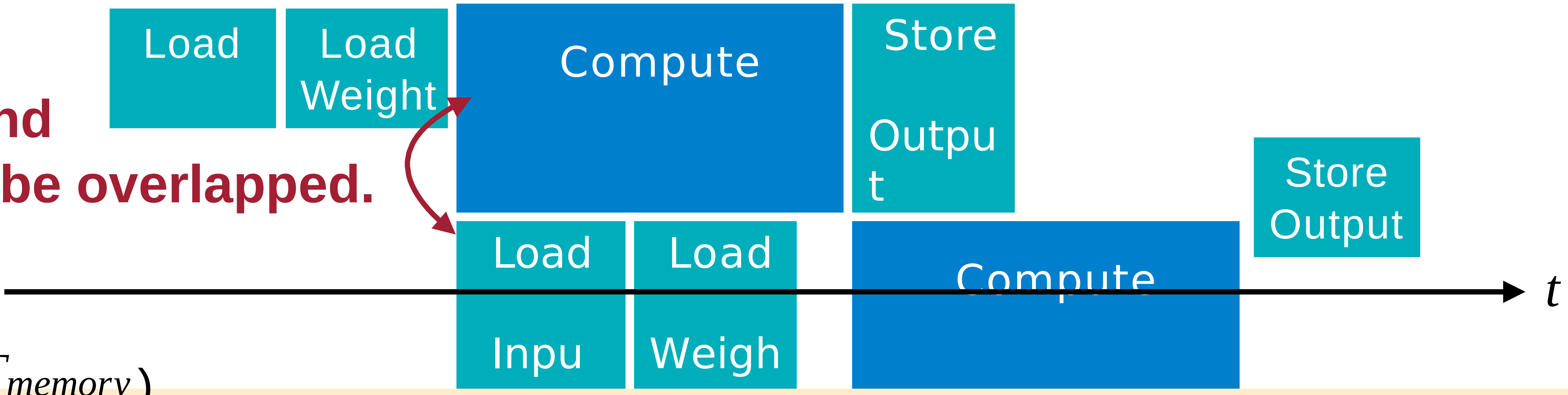


Design 2

Latency: 100 ms
Throughput: 40 image/s

Latency

Data Movement and Computation can be overlapped.



$$Latency \approx \max\left(T_{computation}, T_{memory}\right)$$

$$T_{\text{computation}} \approx \frac{\text{Number of Operations in Neural Network Model}}{\text{Number of Operations that Processor can Process Per Second}}$$

NN Specification

Hardware Specification

$$T_{\text{memory}} \approx T_{\text{data movement of activations}} + T_{\text{data movement of weights}}$$

$$T_{\text{data movement of weights}} \approx \frac{\text{Neural Network Model Size}}{\text{Memory Bandwidth of Processor}}$$

NN Specification

Hardware Specification

$$T_{\text{data movement of activations}} \approx \frac{\text{Input Activation Size + Output Activation Size}}{\text{Memory Bandwidth of Processor}}$$

NN Specification

Hardware Specification

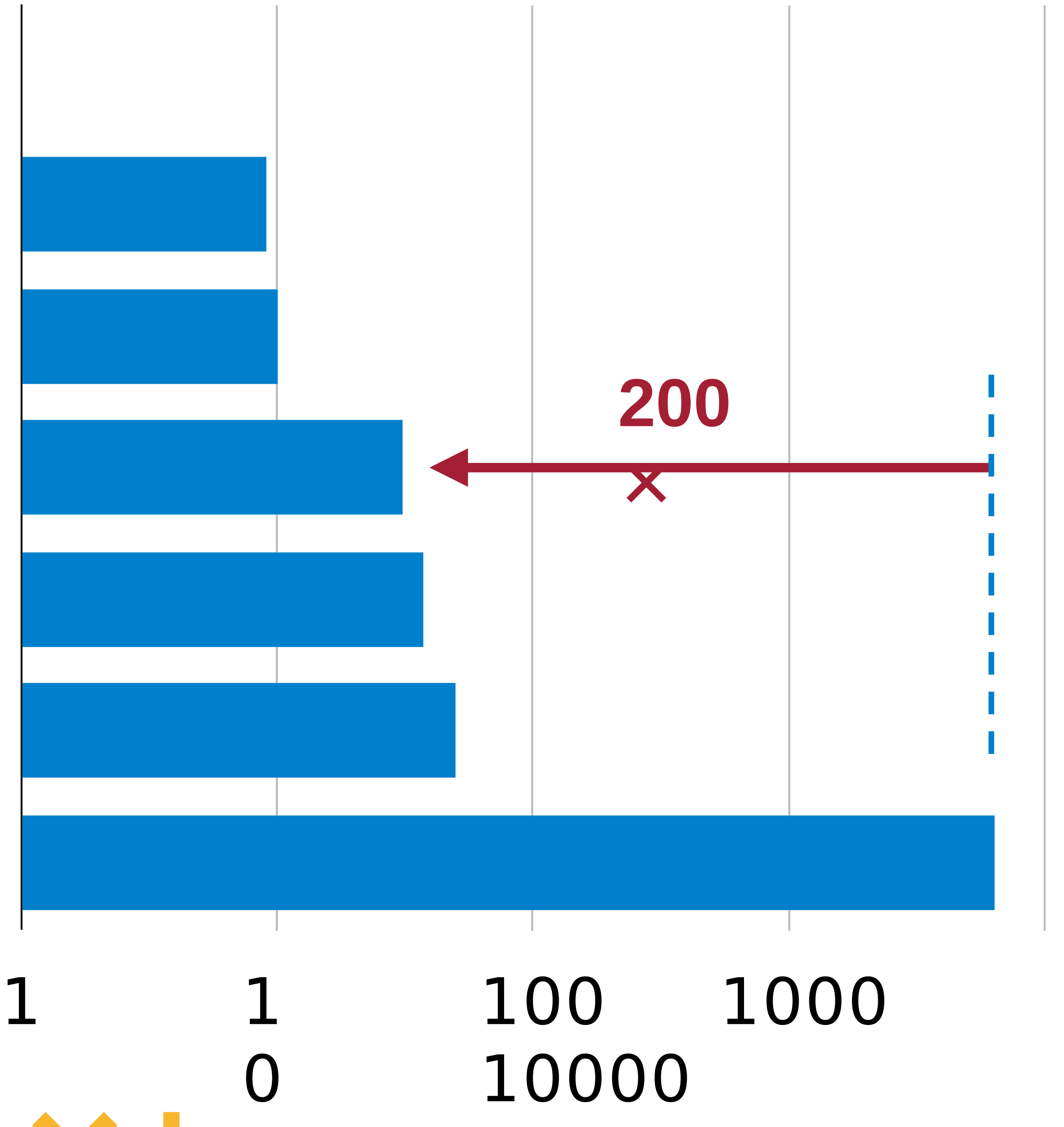
Energy Consumption

- **Data movement → more memory reference → more energy**

Operation	Energy [pJ]
32 bit int ADD	0.1
32 bit float ADD	0.9
32 bit Register File	1
32 bit int MULT	3.1
32 bit float MULT	3.7
32 bit SRAM Cache	5
32 bit DRAM Memory	640

Rough Energy Cost For Various Operations in 45nm 0.9V

Relative Energy Cost



1  **= 200 ✕+**

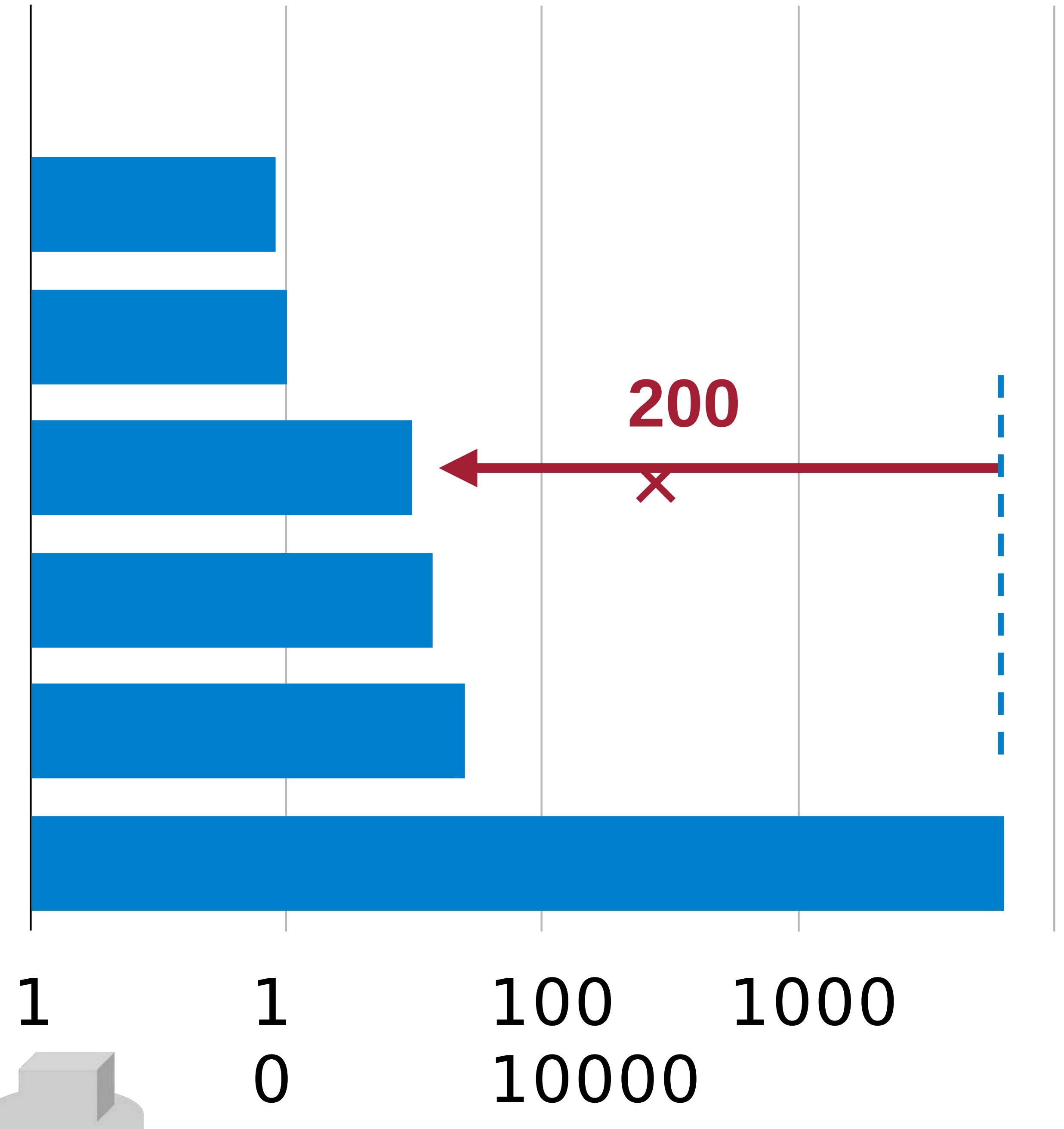
Energy Consumption

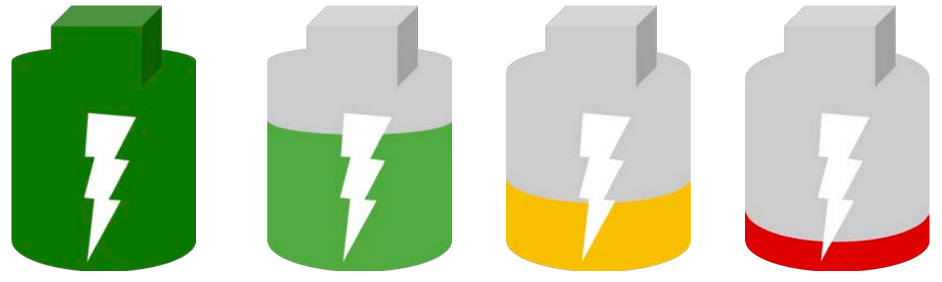
- **Data movement → more memory reference → more energy**

Operation	Energy [pJ]
32 bit int ADD	0.1
32 bit float ADD	0.9
32 bit Register File	1
32 bit int MULT	3.1
32 bit float MULT	3.7
32 bit SRAM Cache	5
32 bit DRAM Memory	640

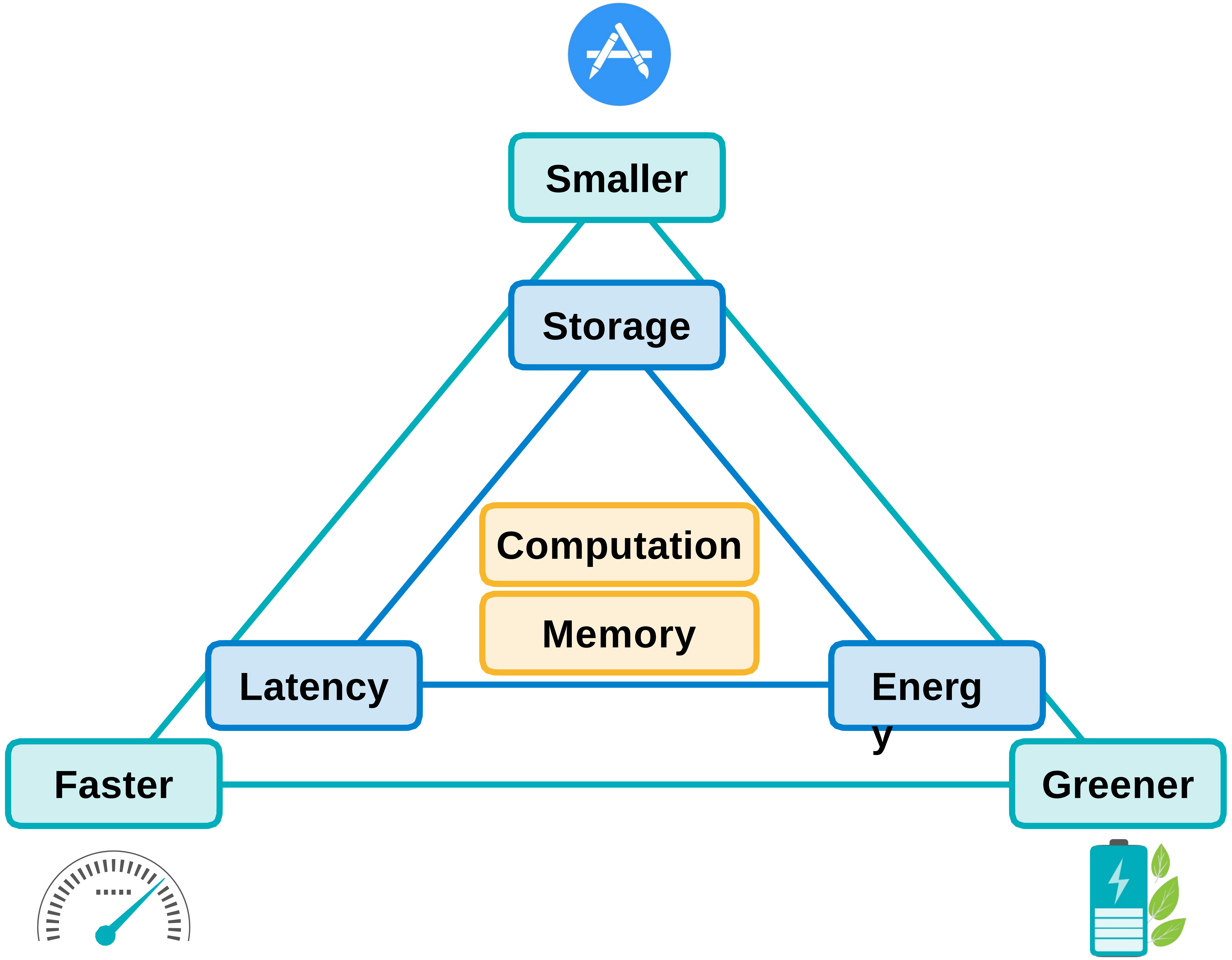
Rough Energy Cost For Various Operations in 45nm 0.9V

Relative Energy Cost





Efficiency of Neural Networks



Efficiency Metrics

Memory-Related

#parameters

model size

total/peak #activations

Computation-Related

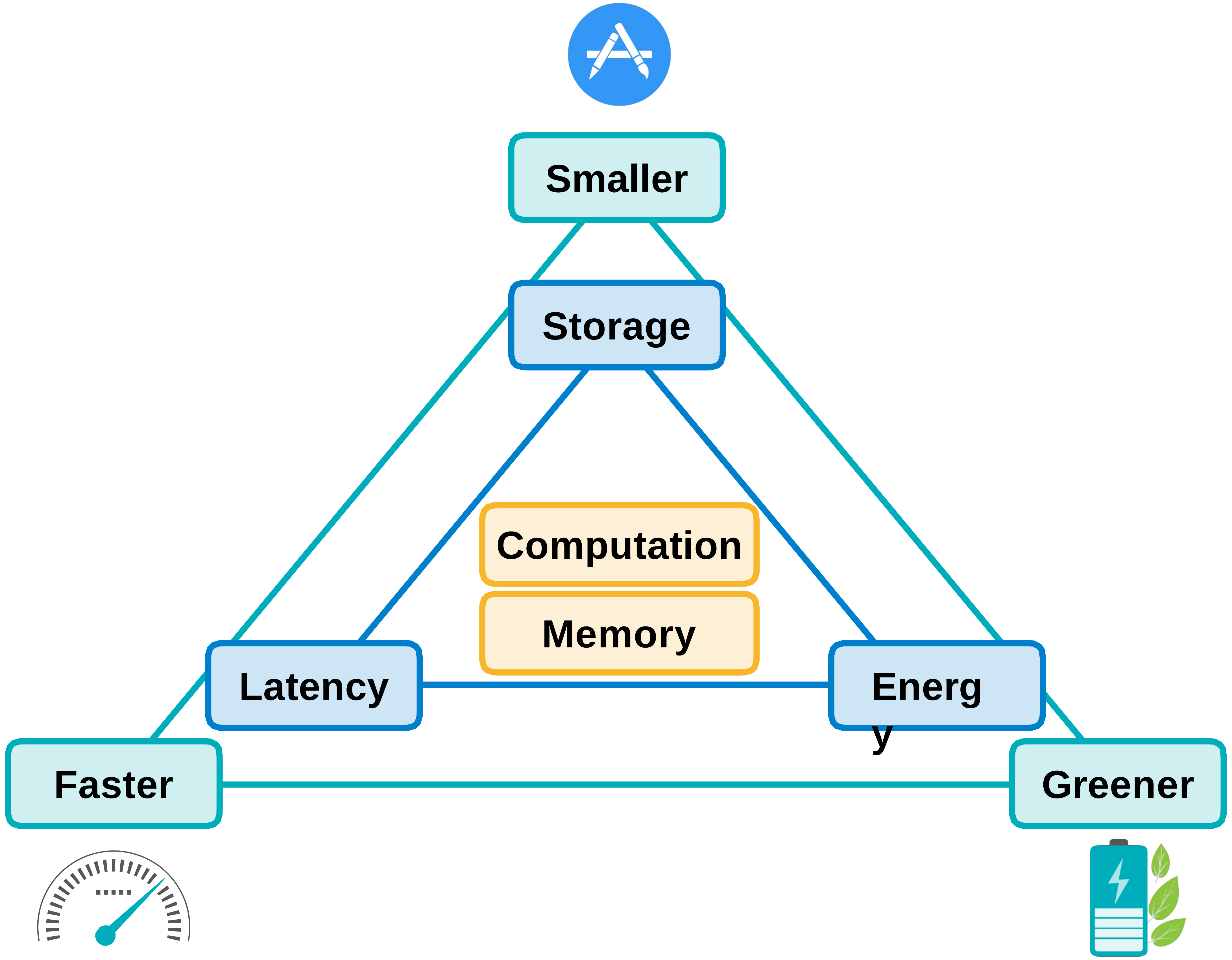
MAC

FLOP, FLOPS

OP, OPS

Image source: 1

Efficiency of Neural Networks



Efficiency Metrics

Memory-Related

#parameters

model size

total/peak #activations

Computation-Related

MAC

FLOP, FLOPS

OP, OPS

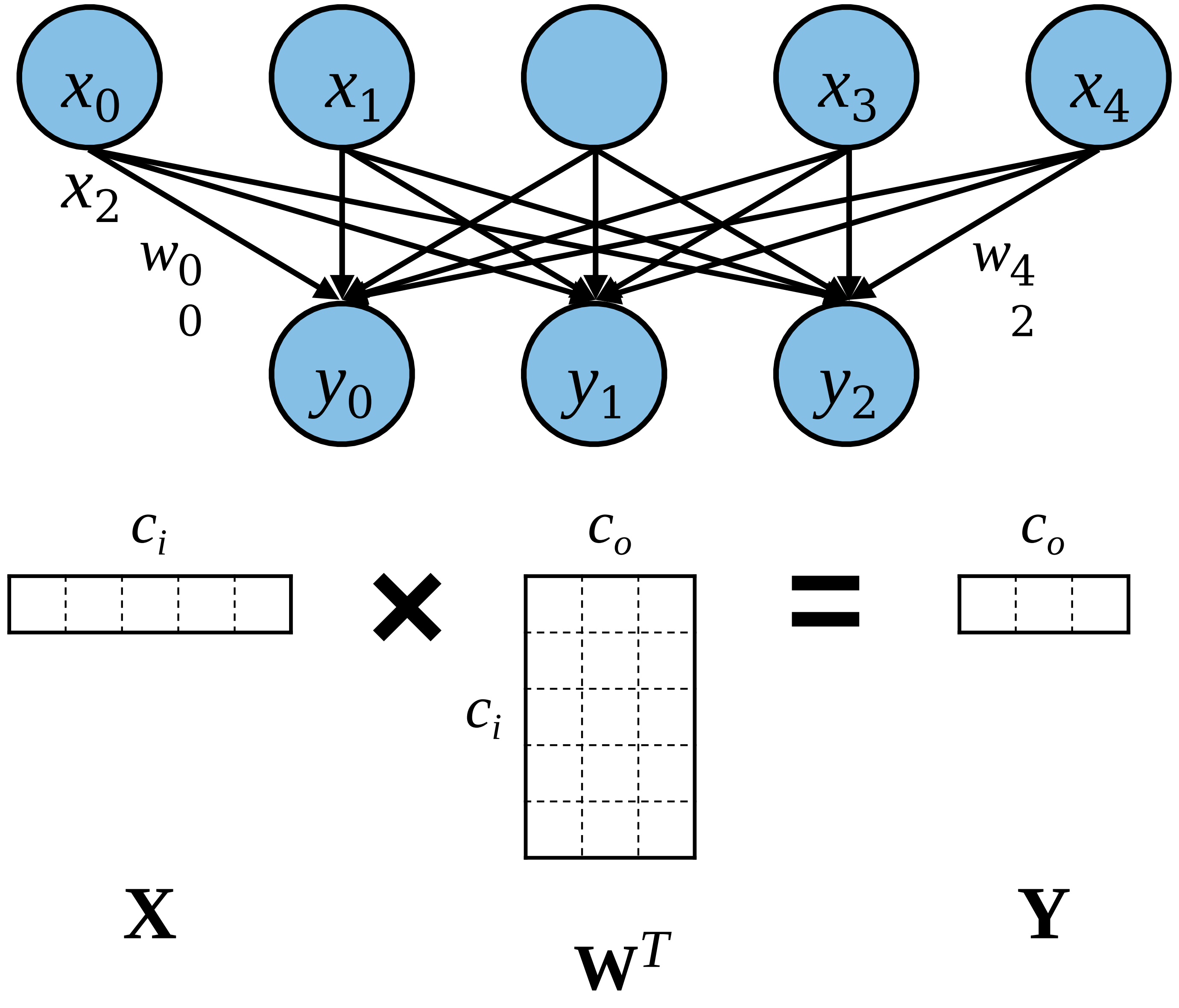
Image source: 1

Number of Parameters (#Parameters)

- #Parameters is the parameter (synapse/weight) count of the given neural network, *i.e.*, the number of elements in the weight tensors.

Number of Parameters (#Parameters)

Layer	#Parameters
Linear Layer	
Convolution	
Grouped Convolution	
Depthwise Convolution	

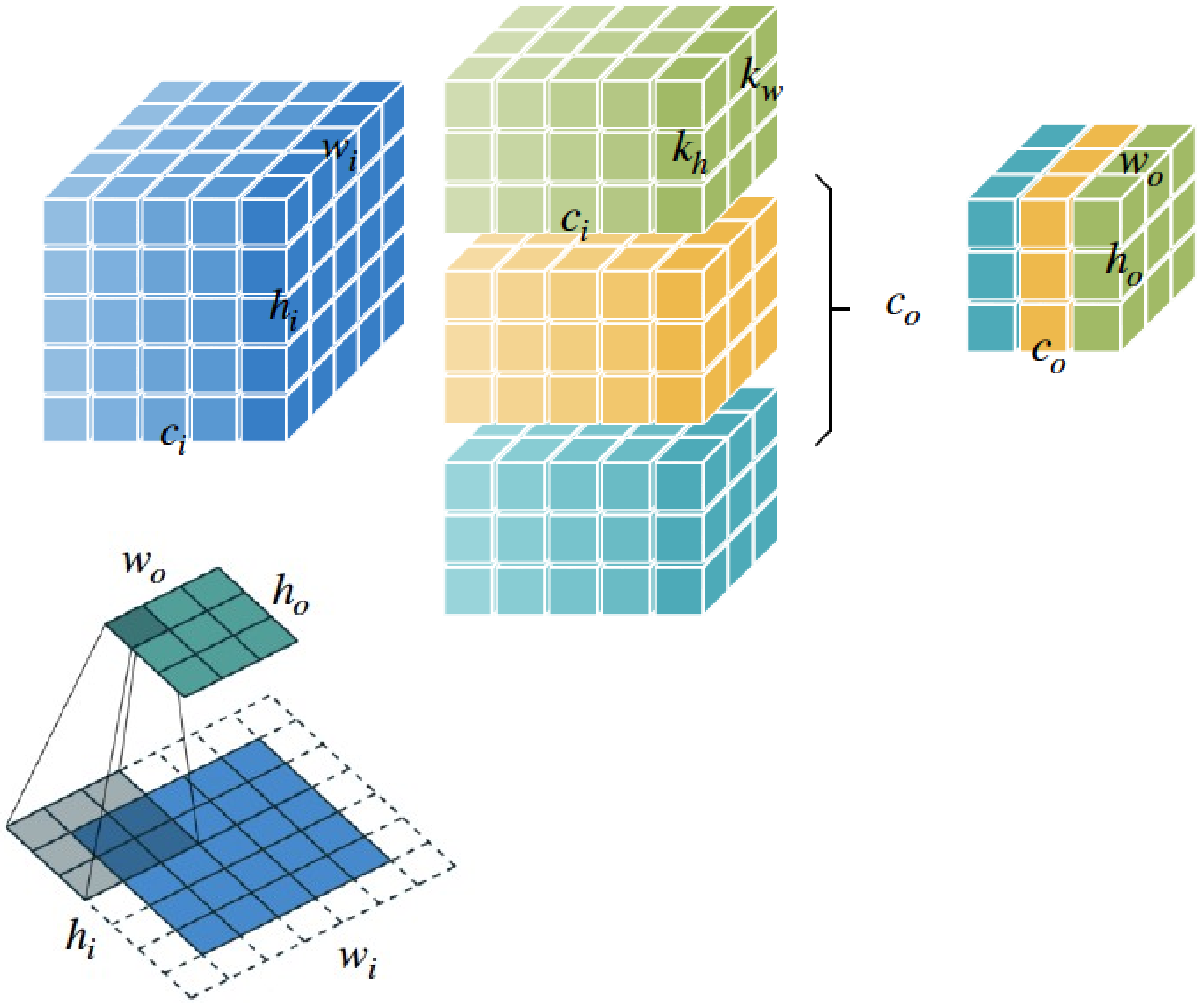


Notations	
n	Batch Size
c_i	Input Channels
c_o	Output Channels
w_i, w_o	Input/Output Width
h_i, h_o	Input/Output Height
k_h, k_w	Kernel Height/Width
g	Groups

Number of Parameters (#Parameters)

Layer	#Parameters
Linear Layer	
Convolution	
Grouped Convolution	
Depthwise Convolution	

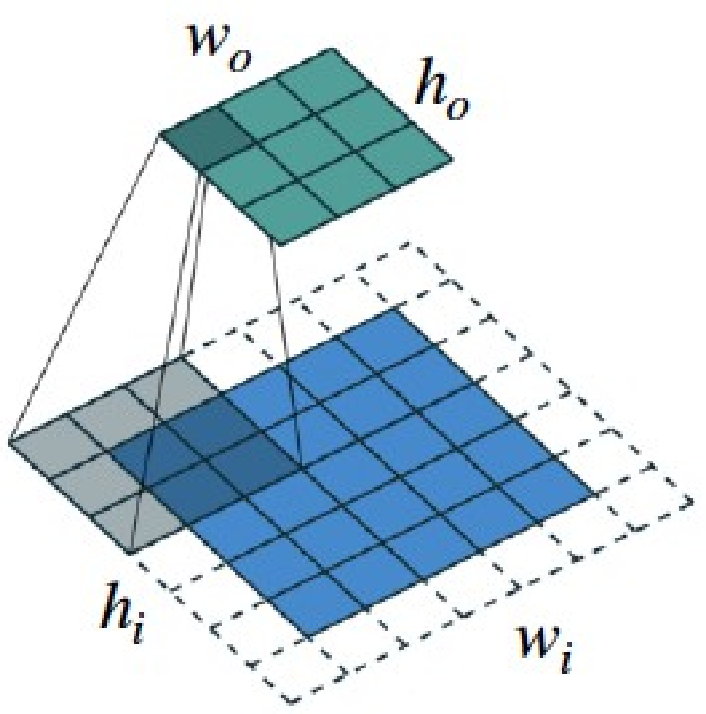
Notations	
n	Batch Size
c_i	Input Channels
c_o	Output Channels
w_i, w_o	Input/Output Width
h_i, h_o	Input/Output Height
k_h, k_w	Kernel Height/Width
g	Groups

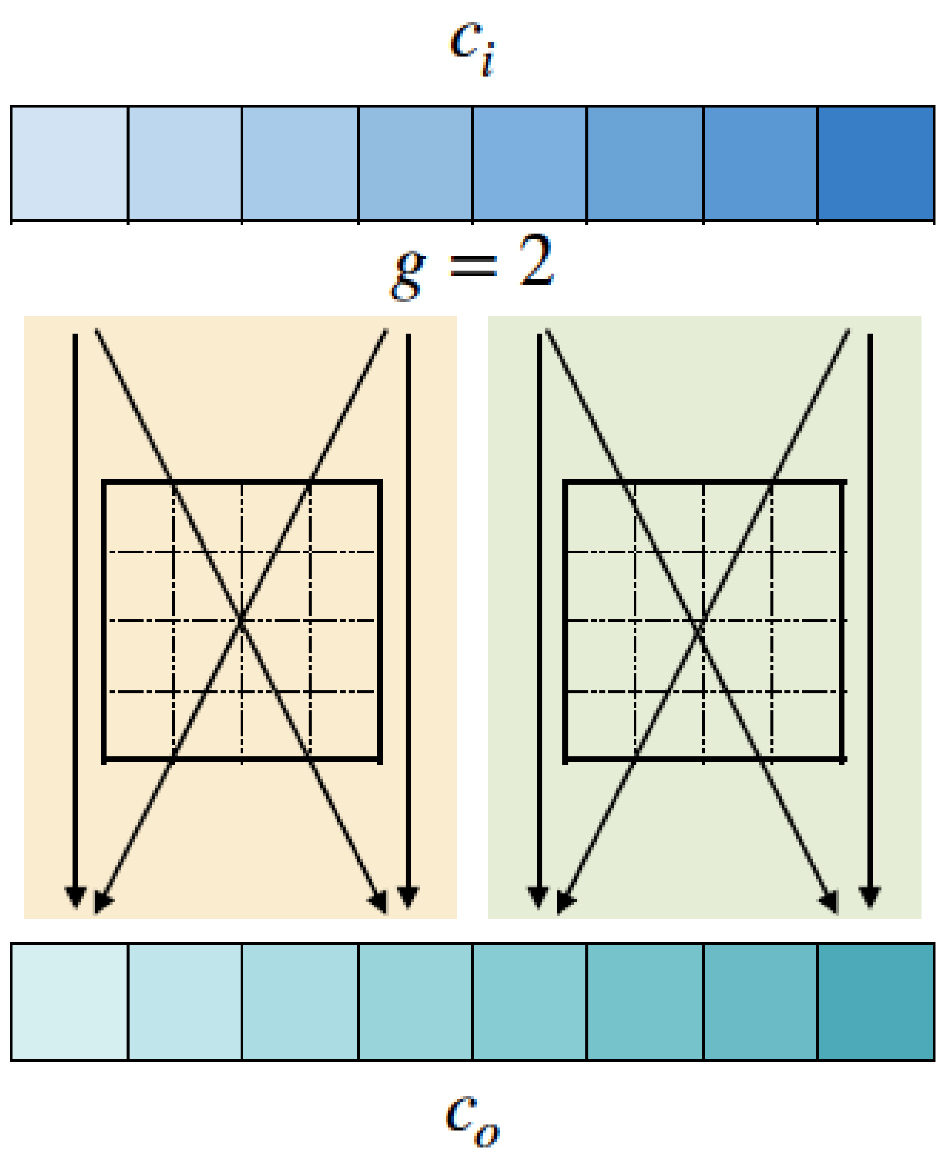


Number of Parameters (#Parameters)

Layer	#Parameters
Linear Layer	
Convolution	
Grouped Convolution	g g
Depthwise Convolution	

Notations	
n	Batch Size
c_i	Input Channels
c_o	Output Channels
w_i, w_o	Input/Output Width
h_i, h_o	Input/Output Height
k_h, k_w	Kernel Height/Width
g	Groups

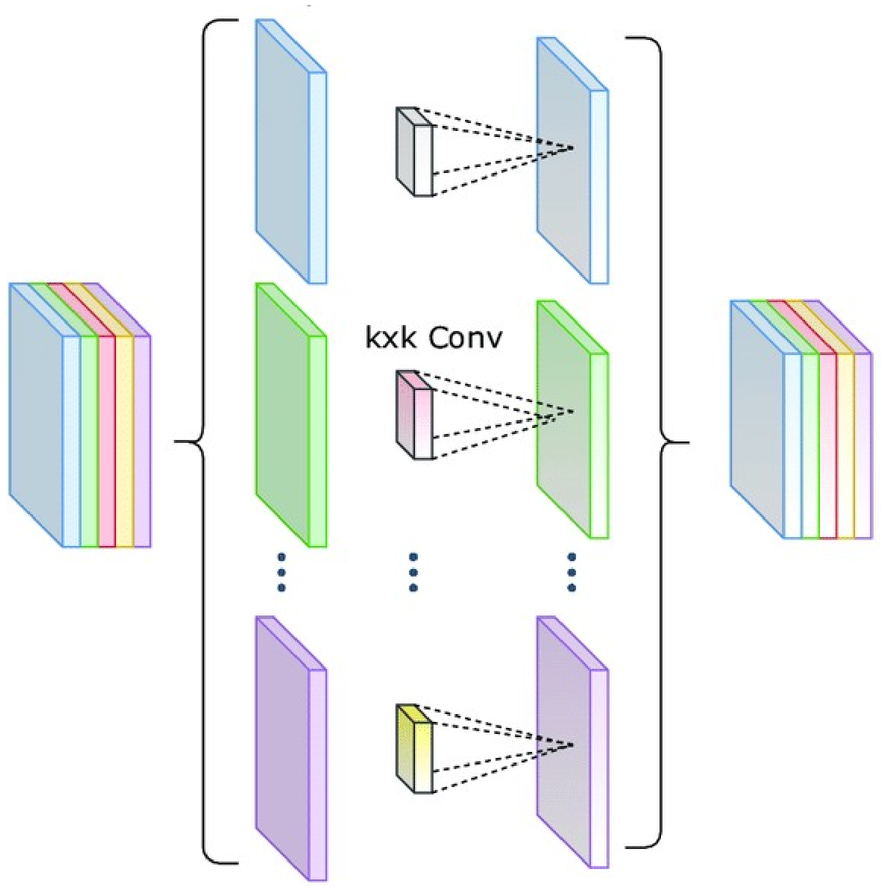


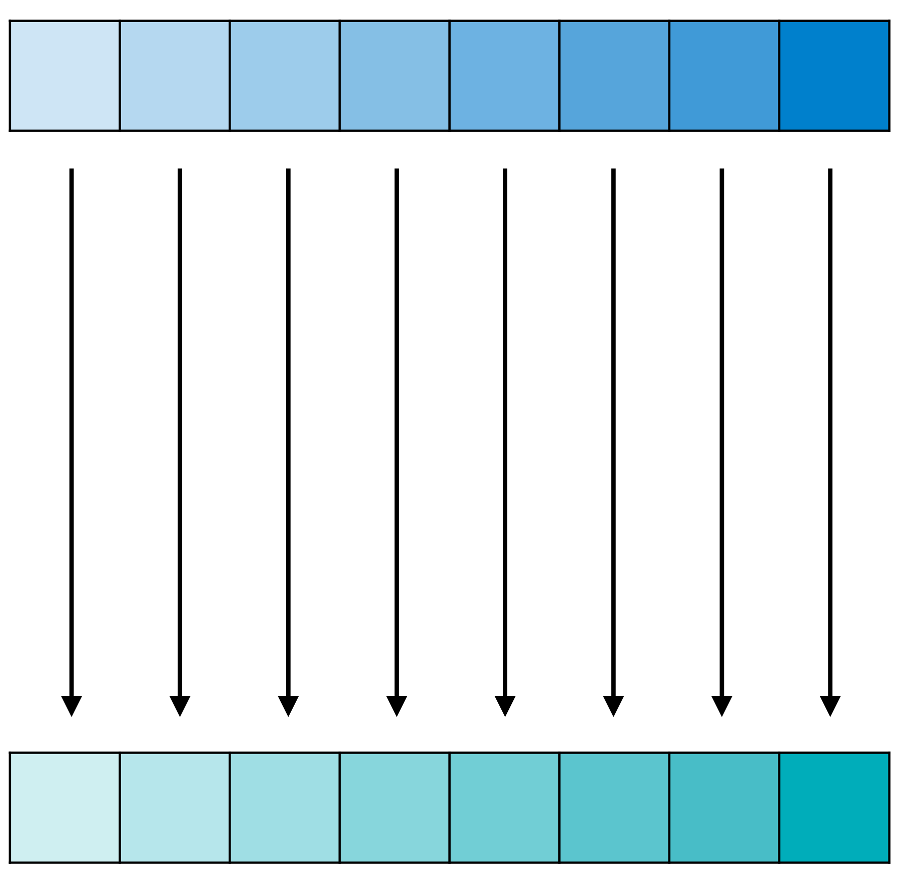


Number of Parameters (#Parameters)

Layer	#Parameters
Linear Layer	
Convolution	
Grouped Convolution	g
Depthwise Convolution	

Notations	
n	Batch Size
c_i	Input Channels
c_o	Output Channels
w_i, w_o	Input/Output Width
h_i, h_o	Input/Output Height
k_h, k_w	Kernel Height/Width
g	Groups





AlexNet: #Parameters

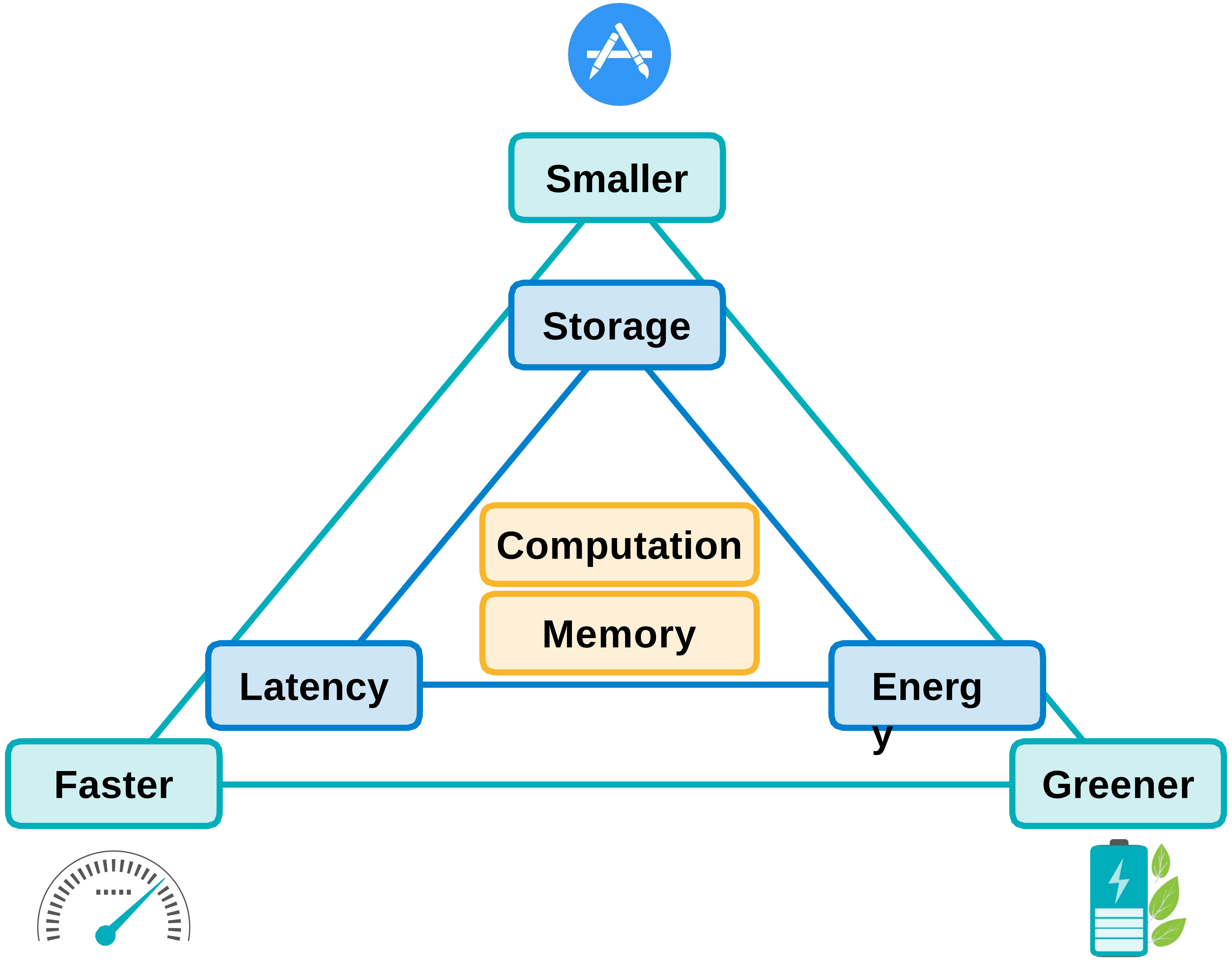
AlexNet	C × H × W	#Parameters (bias is ignored)
Image (3×224×224)	3×224×224	
11×11 Conv, channel 96, stride 4, pad 2	96×55×55	96×3×11×11 = 24, 848
3x3 MaxPool, stride 2	96×27×27	
5x5 Conv, channel 256, pad 2, groups 2	256×27×27	256×96×5×5 / 2 = 307, 200
3x3 MaxPool, stride 2	256×13×13	
3x3 Conv, channel 384, pad 1	384×13×13	384×256×3×3 = 884, 736
3x3 Conv, channel 384, pad 1, groups 2	384×13×13	384×384×3×3 / 2 = 663, 552
3x3 Conv, channel 256, pad 1, groups 2	256×13×13	256[illegible]×3 / 2 = 442, 368
3x3 MaxPool, stride 2	256×6×6	
Linear, channel 4096	4096	4096×(256×6×6) = 37, 748, 736
Linear, channel 4096	4096	4096×4096 = 16, 777, 216
Linear, channel 1000	1000	1000×4096 = 4, 096, 000

?

Layer	#Parameters
Linear Layer	
Convolution	
Grouped Convolution	g
Depthwise Convolution	

61M in total

Efficiency of Neural Networks



Efficiency Metrics

Memory-Related

#parameters

model size

total/peak #activations

Computation-Related

MAC

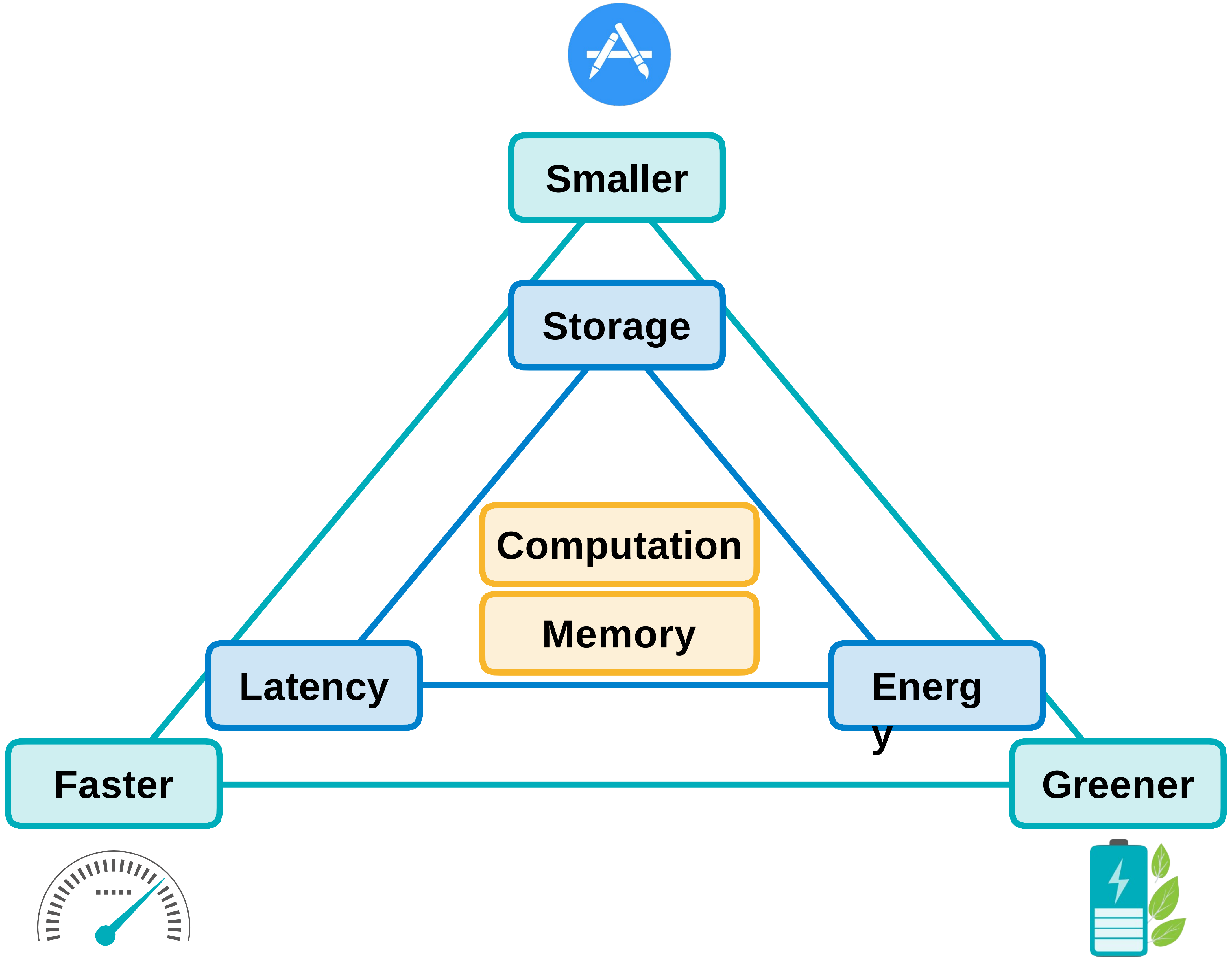
FLOP, FLOPS

OP, OPS

Model Size

- Model size measures the storage for the weights of the given neural network.
  - The common units for model size are: MB (megabyte), KB (kilobyte), bits.
  - In general, if the whole neural network uses the same data type (e.g., floating-point),
- Model Size = #Parameters · Bit Width
  - Example: AlexNet has 61M parameters.
- If all weights are stored with 32-bit numbers, total storage will be about
  - 61M × 4 Bytes (32 bits) = 228 MB
- If all weights are stored with 8-bit numbers, total storage will be about
  - 61M × 1 Bytes (32 bits) = 61 MB

Efficiency of Neural Networks



Efficiency Metrics

Memory-Related

#parameters

model size

total/peak #activations

Computation-Related

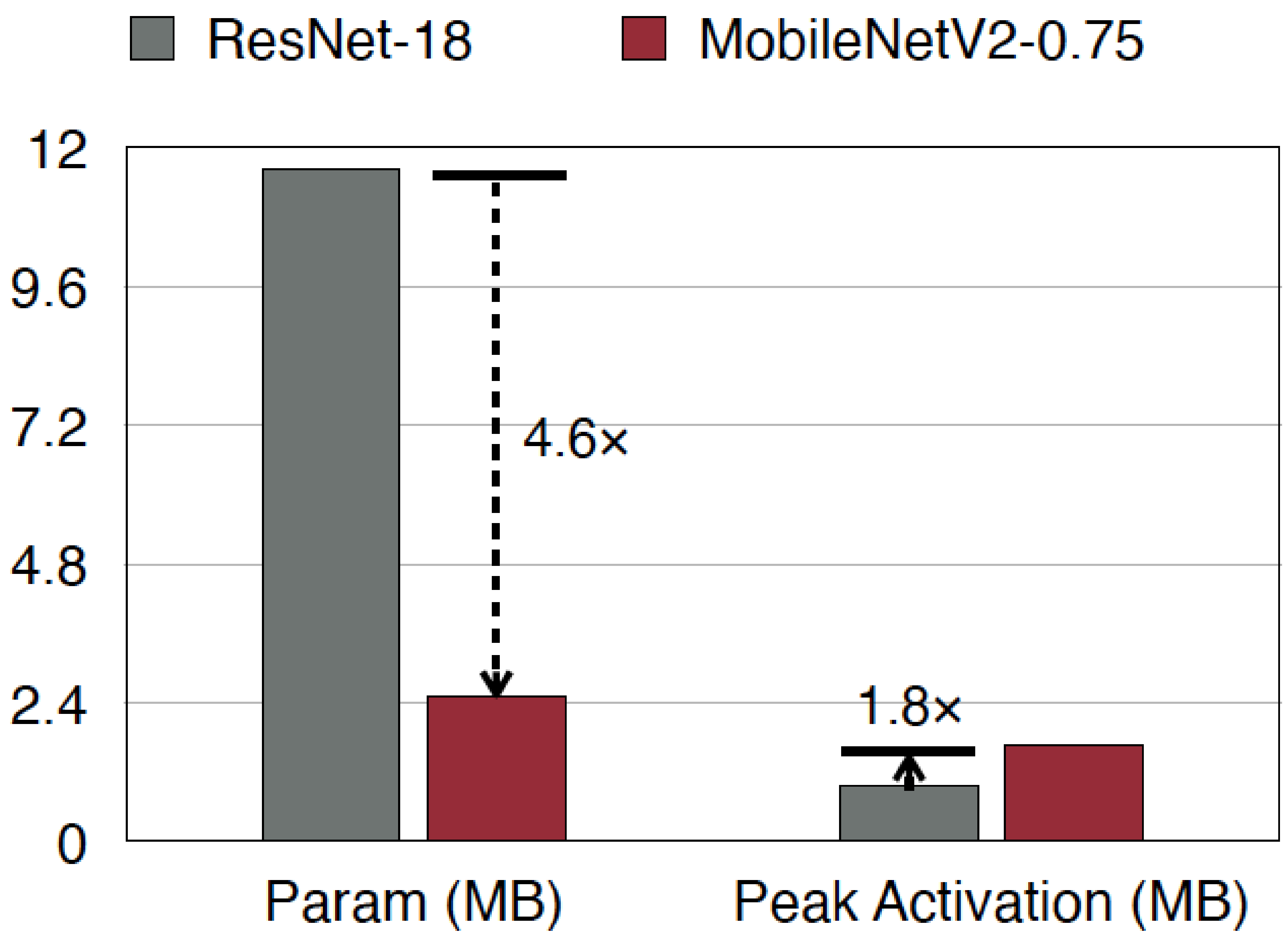
MAC

FLOP, FLOPS

OP, OPS

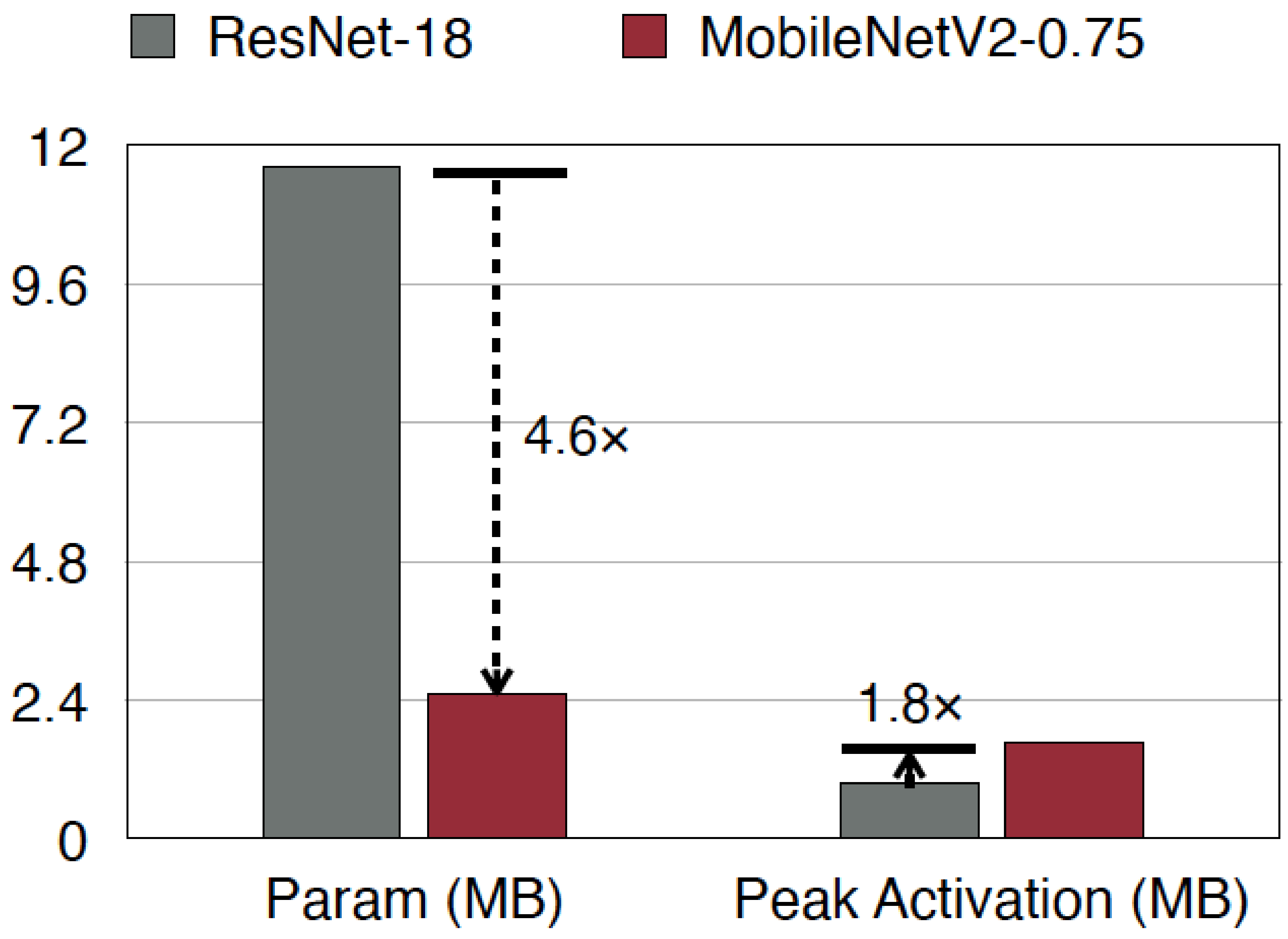
Number of Activations (#Activations)

- #Activation is the memory bottleneck in inference on IoT, not #Parameters



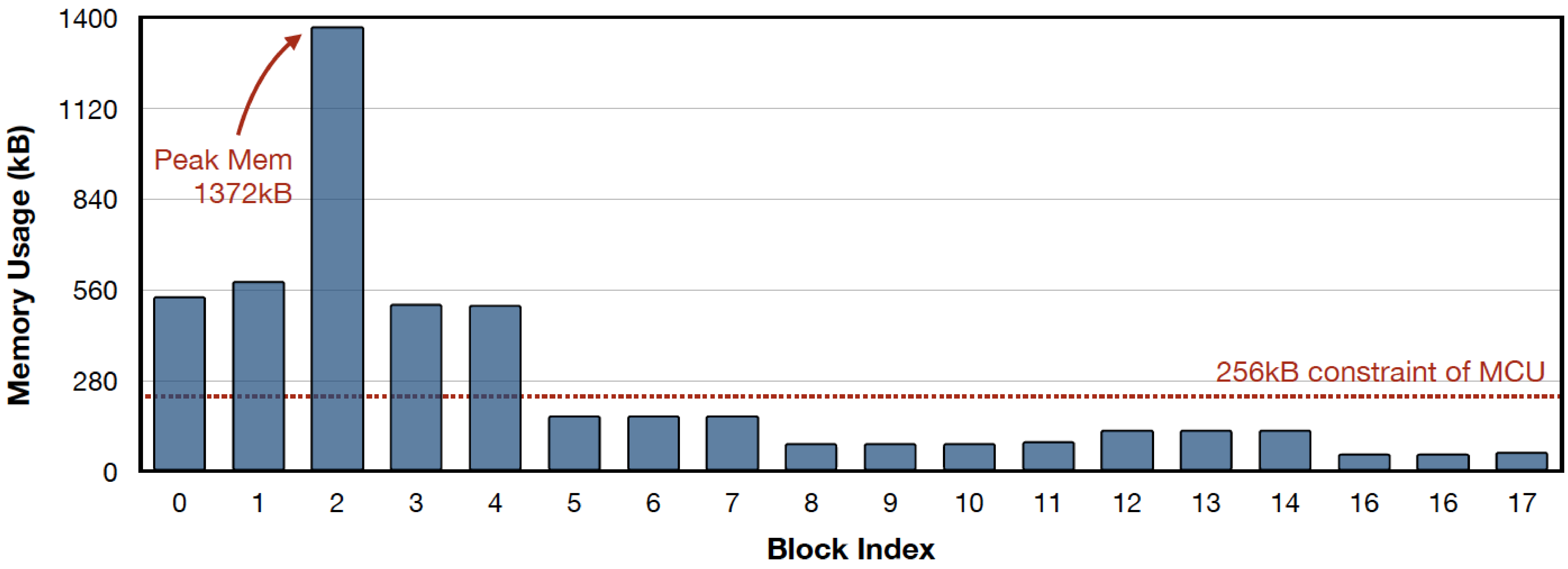
Number of Activations (#Activations)

- #Activation didn't improve from ResNet to MobileNet-v2



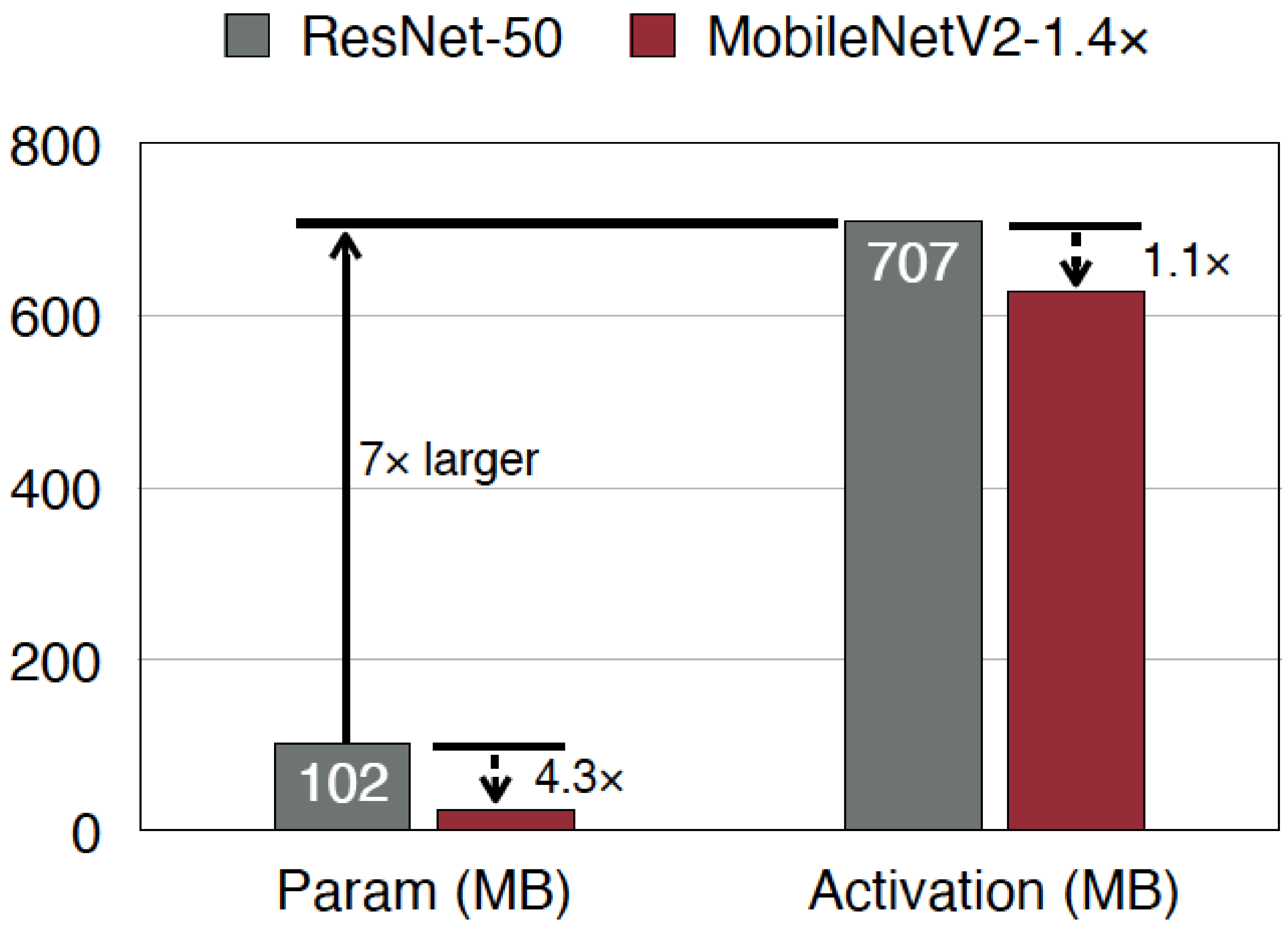
Number of Activations (#Activations)

- Imbalanced memory distribution of MobileNetV2



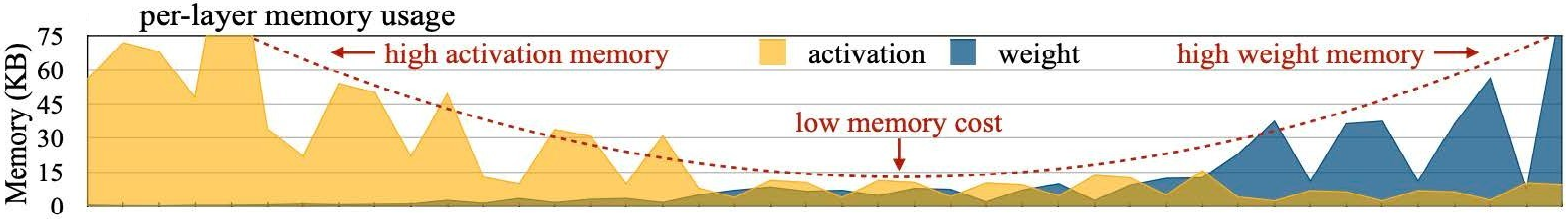
Number of Activations (#Activations)

- #Activation is the memory bottleneck in training, not #Parameters.



Number of Activations (#Activations)

- **Activation and weight memory distribution of MCUNet**



AlexNet: #Activations

AlexNet	C × H × W	
Image (3×224×224)	3×224×224	=150,528
11×11 Conv, channel 96, stride 4, pad 2	96×55×55	=290,400
3x3 MaxPool, stride 2	96×27×27	=69,984
5x5 Conv, channel 256, pad 2, groups 2	256×27×27	=186,624
3x3 MaxPool, stride 2	256×13×13	=43,264
3x3 Conv, channel 384, pad 1	384×13×13	=64,896
3x3 Conv, channel 384, pad 1, groups 2	384×13×13	=64,896
3x3 Conv, channel 256, pad 1, groups 2	256×13×13	=43,264
3x3 MaxPool, stride 2	256×6×6	=9,216
Linear, channel 4096	4096	=4,096
Linear, channel 4096	4096	=4,096
Linear, channel 1000	1000	=1,000

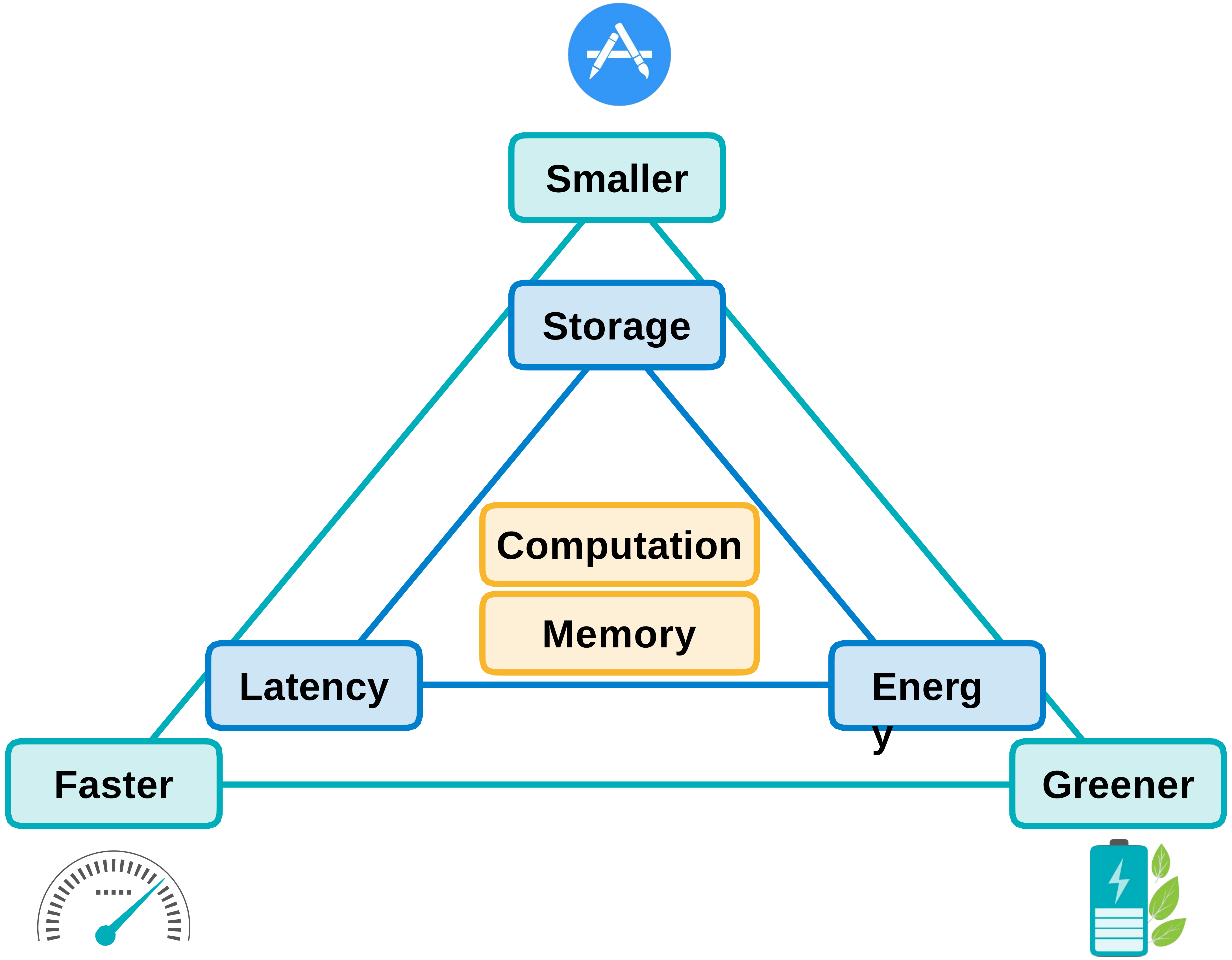
Total #Activation: 932,264

Peak #Activation:

≈ #input activation + #output activation

= 150,528 + 290,400 = 440,928

Efficiency of Neural Networks



Efficiency Metrics

Memory-Related

#parameters

model size

total/peak #activations

Computation-Related

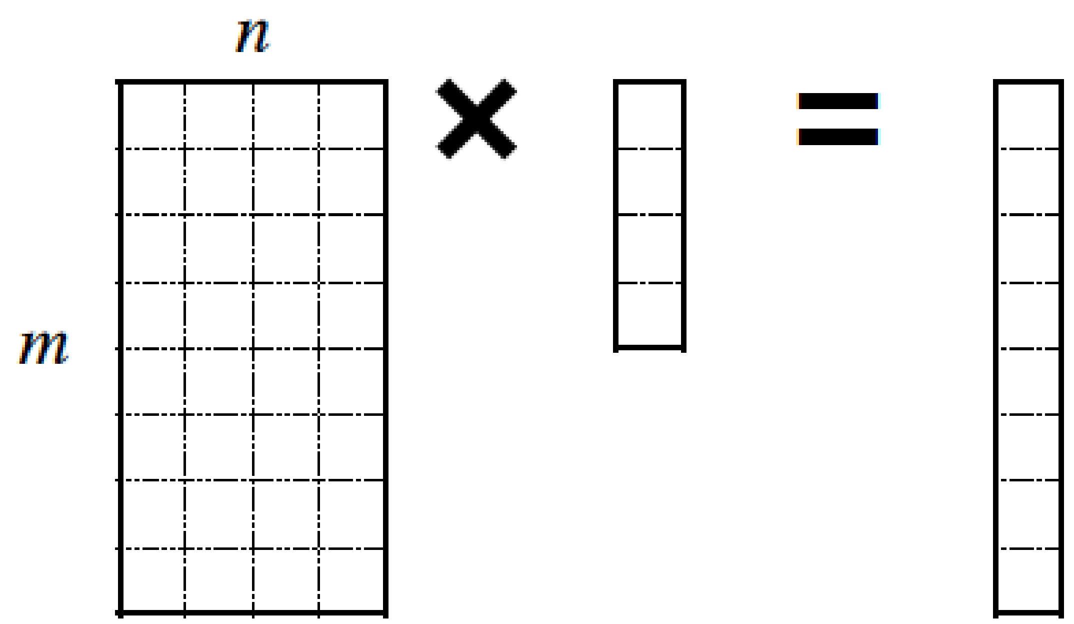
MAC

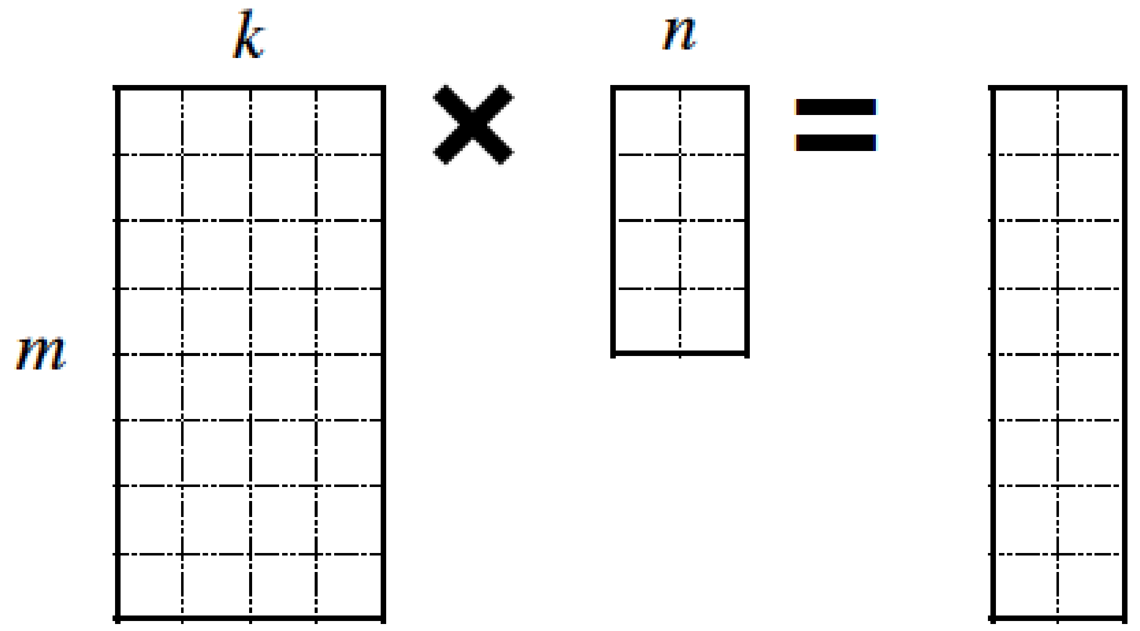
FLOP, FLOPS

OP, OPS

Number of Multiply-Accumulate Operations

- **Multiply-Accumulate Operation (MAC)**
  - $a \leftarrow a + b \cdot c$
- **Matrix-Vector Multiplication (MV)**
  - $MACs = m \cdot n$
- **General Matrix-Matrix Multiplication (GEMM)**
  - $MACs = m \cdot n \cdot k$

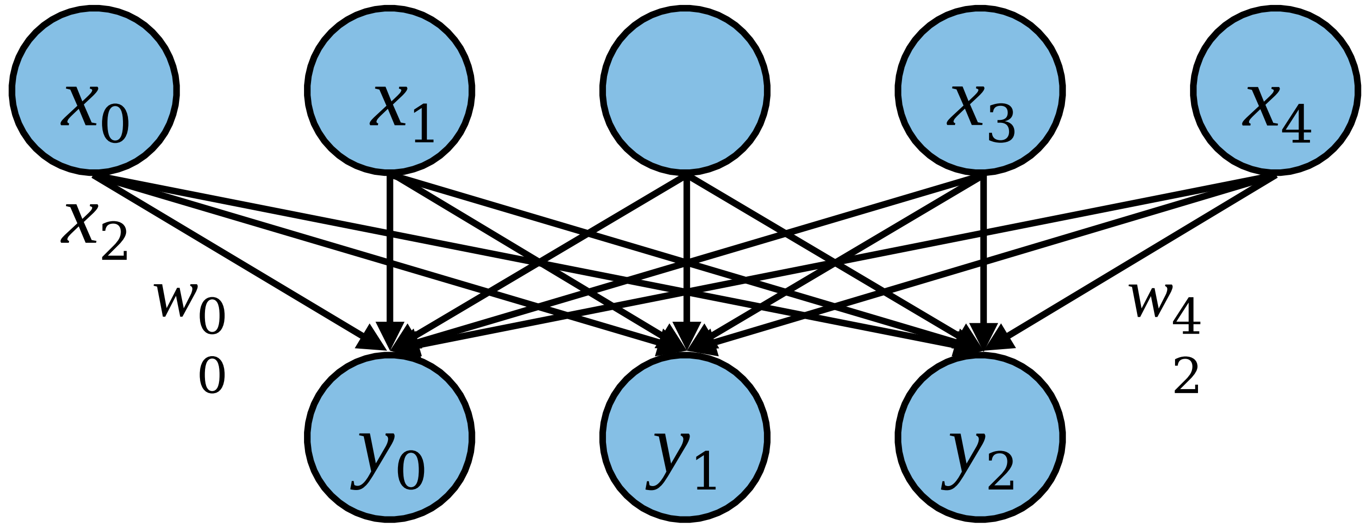


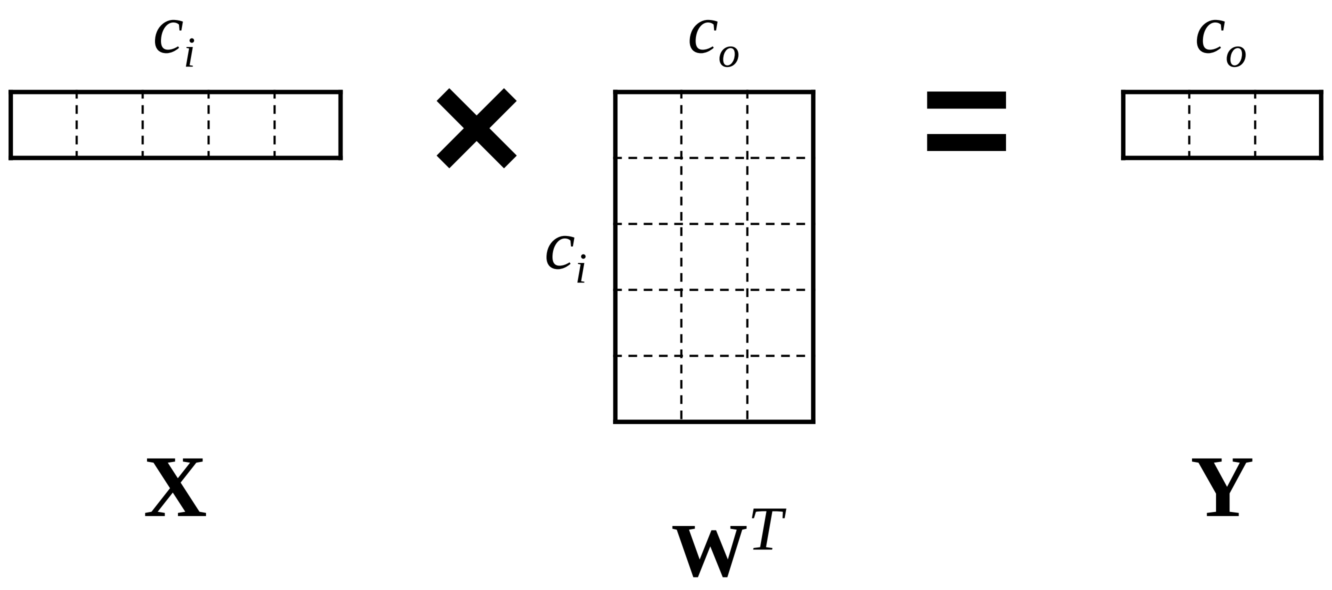


Number of Multiply-Accumulate Operations

Layer	#Parameters
Linear Layer	
Convolution	
Grouped Convolution	g
Depthwise Convolution	

Notations	
n	Batch Size
c_i	Input Channels
c_o	Output Channels
w_i, w_o	Input/Output Width
h_i, h_o	Input/Output Height
k_h, k_w	Kernel Height/Width
g	Groups

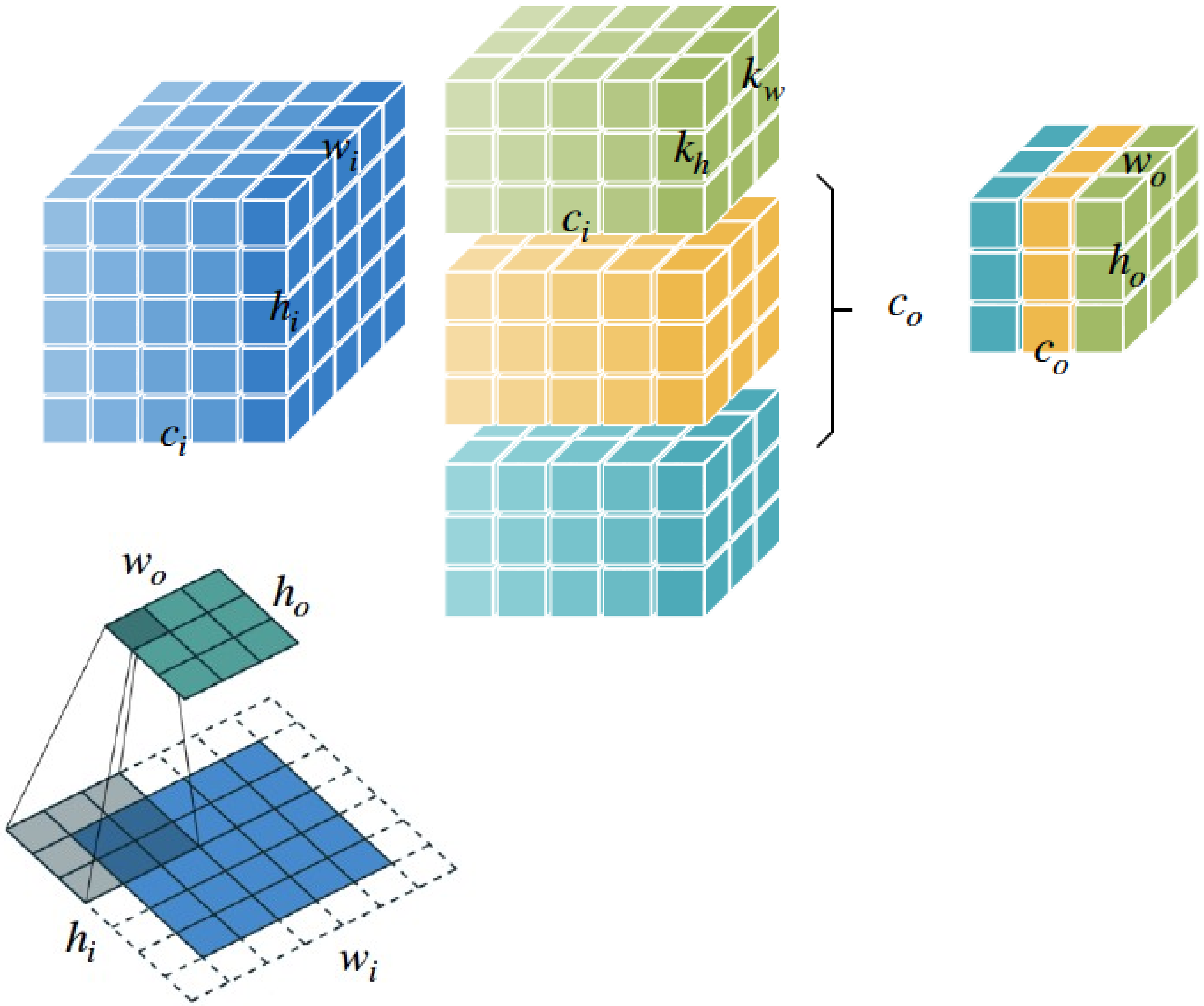




Number of Multiply-Accumulate Operations

Layer	#MACs
Linear Layer	
Convolution	
Grouped Convolution	
Depthwise Convolution	

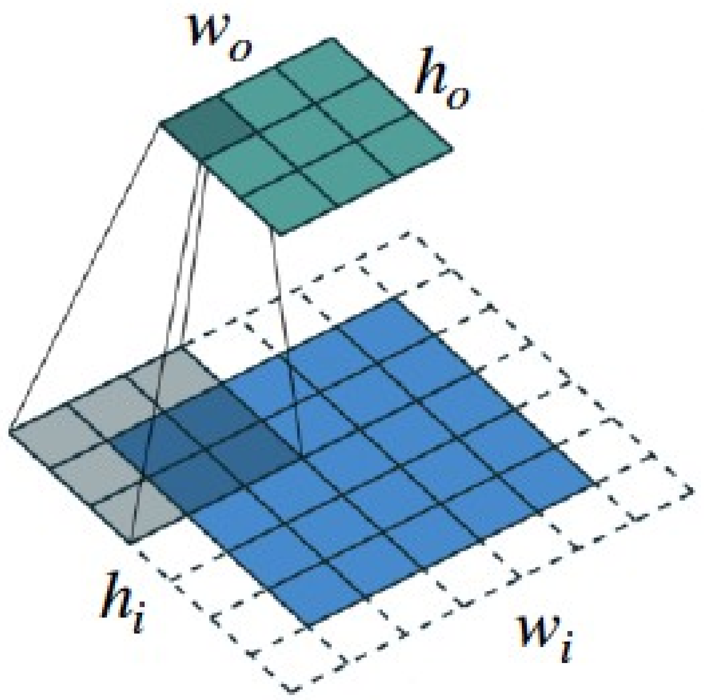
Notations	
n	Batch Size
c_i	Input Channels
c_o	Output Channels
w_i, w_o	Input/Output Width
h_i, h_o	Input/Output Height
k_h, k_w	Kernel Height/Width
g	Groups

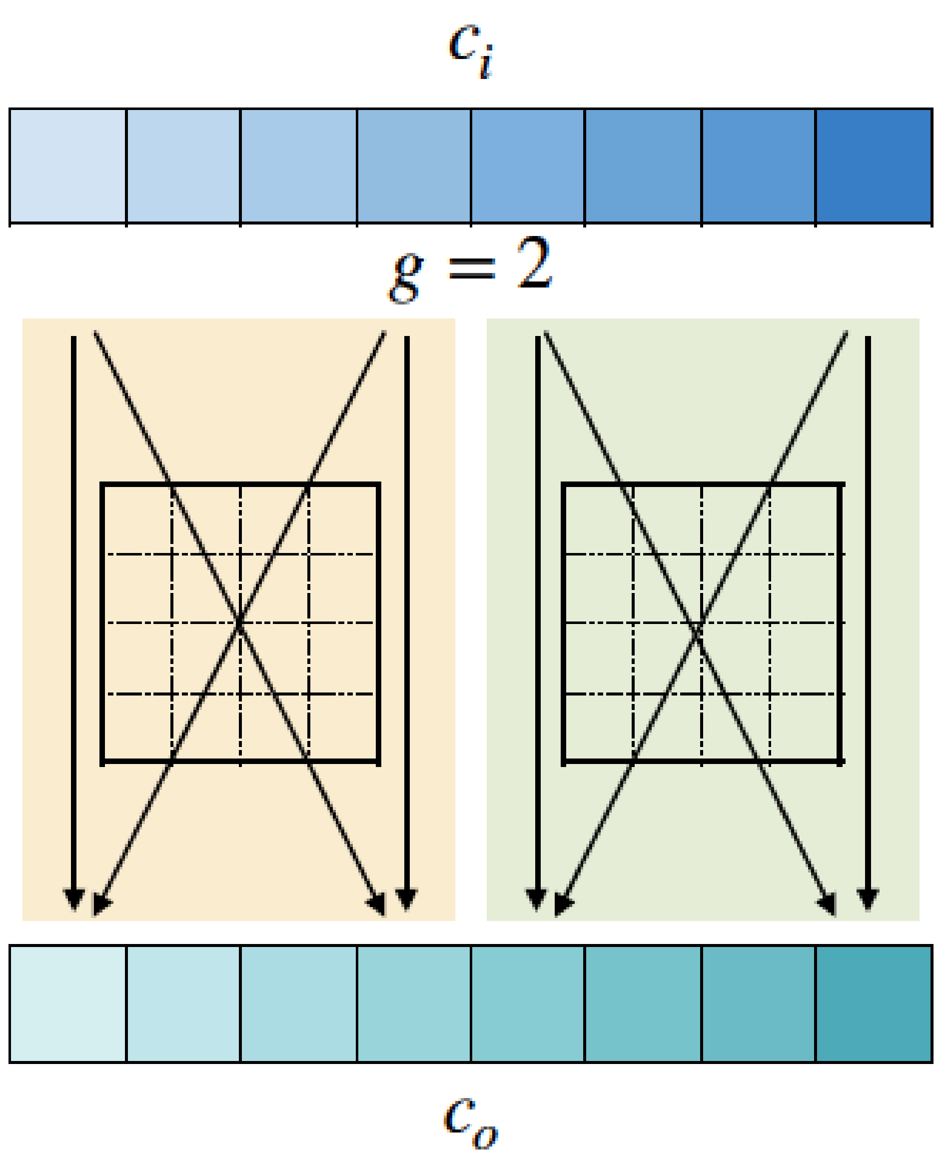


Number of Multiply-Accumulate Operations

Layer	#MACs
Linear Layer	
Convolution	
Grouped Convolution	
Depthwise Convolution	

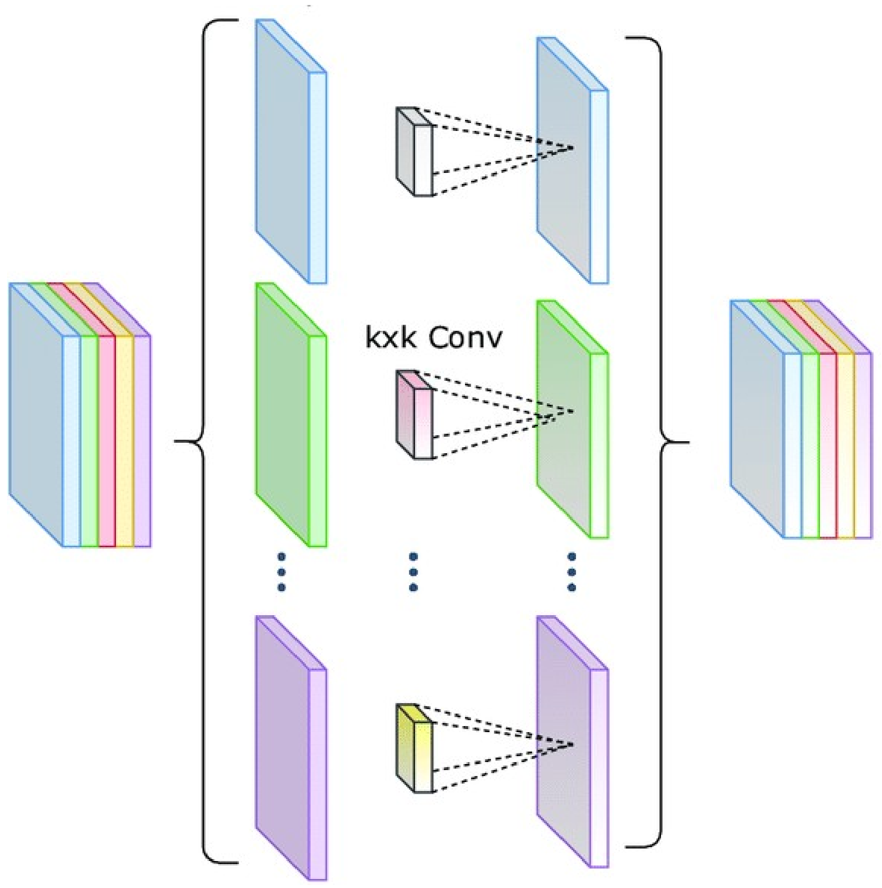
Notations	
n	Batch Size
c_i	Input Channels
c_o	Output Channels
w_i, w_o	Input/Output Width
h_i, h_o	Input/Output Height
k_h, k_w	Kernel Height/Width
g	Groups

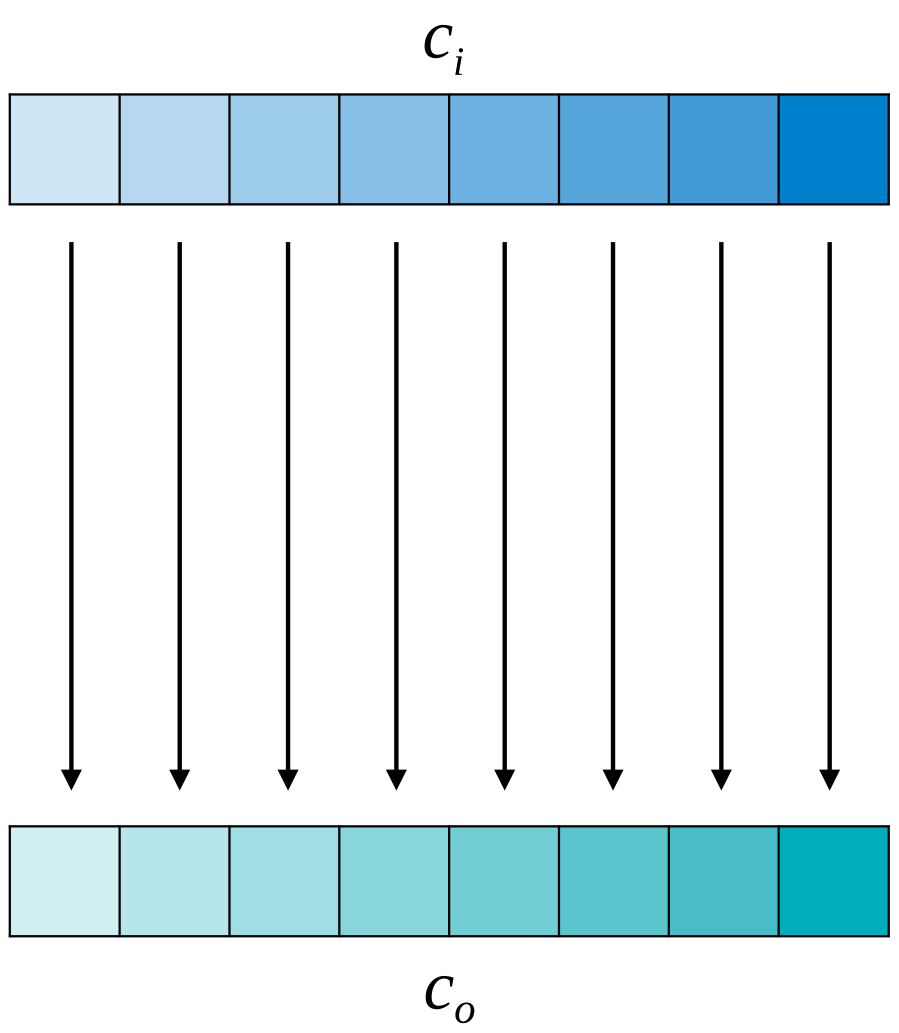




Number of Multiply-Accumulate Operations

Layer	#MACs
Linear Layer	
Convolution	
Grouped Convolution	
Depthwise Convolution	





Notations	
n	Batch Size
c_i	Input Channels
c_o	Output Channels
w_i, w_o	Input/Output Width
h_i, h_o	Input/Output Height
k_h, k_w	Kernel Height/Width
g	Groups

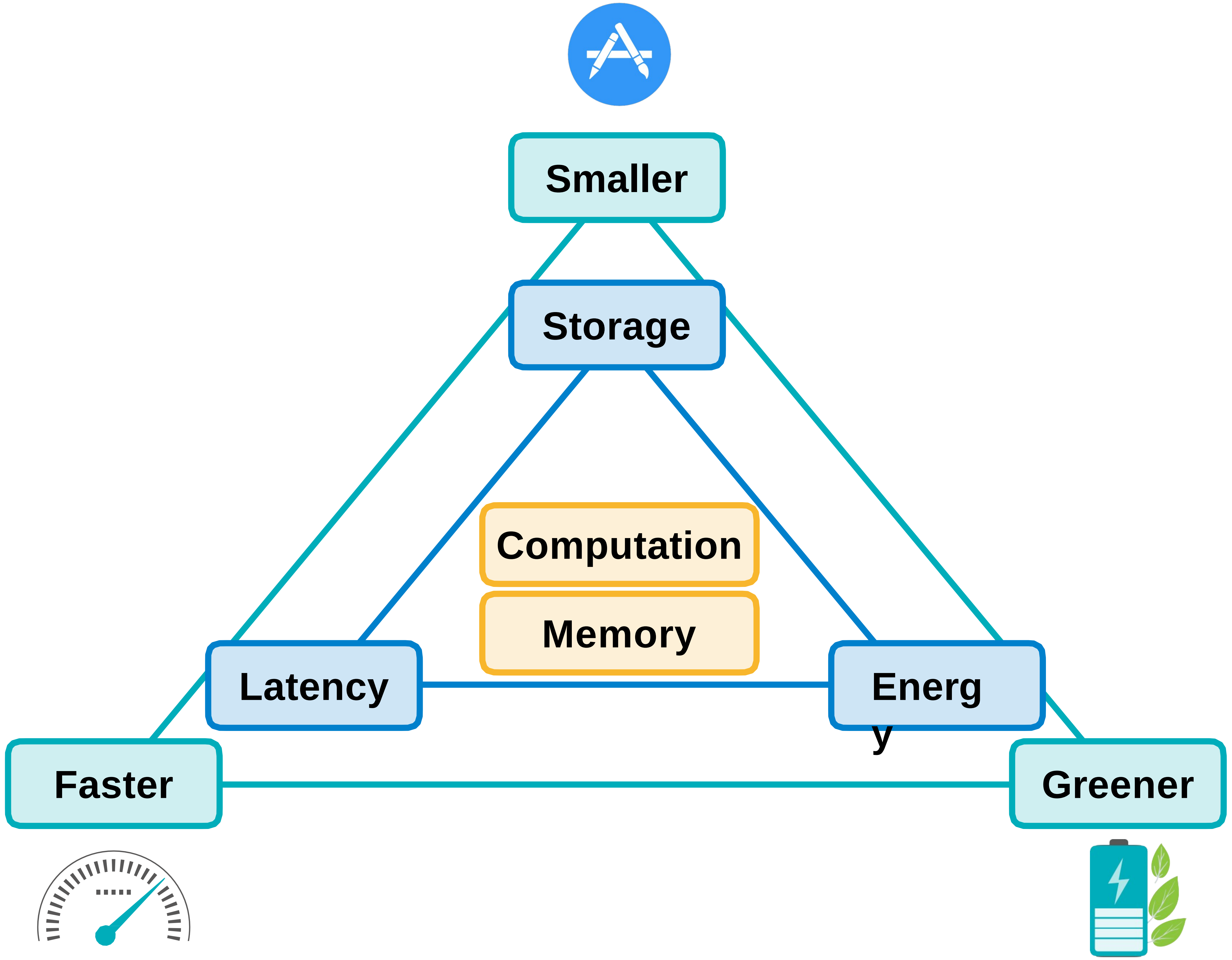
AlexNet: #MACs

AlexNet	C × H × W	MACs
Image (3×224×224)	3×224×224	
11×11 Conv, channel 96, stride 4, pad 2	96×55×55	96×3×11×11×55×55 = 105,415,200
3×3 MaxPool, stride 2	96×27×27	
5×5 Conv, channel 256, pad 2, groups 2	256×27×27	256×96×5×5×27×27 / 2 = 223,948,800
3×3 MaxPool, stride 2	256×13×13	
3×3 Conv, channel 384, pad 1	384×13×13	384×256×3×3×13×13 = 149,520,384
3×3 Conv, channel 384, pad 1, groups 2	384×13×13	384×384×3×3×13×13 / 2 = 112,140,288
3×3 Conv, channel 256, pad 1, groups 2	256×13×13	256[illegible]×3×13×13 / 2 = 74,760,1[illegible]
3×3 MaxPool, stride 2	256×6×6	
Linear, channel 4096	4096	4096×(256×6×6) = 37,748,736
Linear, channel 4096	4096	4096×4096 = 16,777,216
Linear, channel 1000	1000	1000×4096 = 4,096,000

Layer	#Parameters
Linear Layer	
Convolution	
Grouped Convolution	
Depthwise Convolution	

724M in total !!

Efficiency of Neural Networks



Efficiency Metrics

Memory-Related

#parameters

model size

total/peak #activations

Computation-Related

MAC

FLOP, FLOPS

OP, OPS

Number of Floating Point Operations (FLOP)

- A multiply is a floating-point operation
- An add is a floating-point operation
- One multiply-accumulate operation is two floating-point operations (FLOPs)
  - Example: AlexNet has 724M MACs, total number of floating-point operations will be
    - 724M x 2 = 1.4G FLOPs
- Floating-Point Operations Per Second (FLOPS)

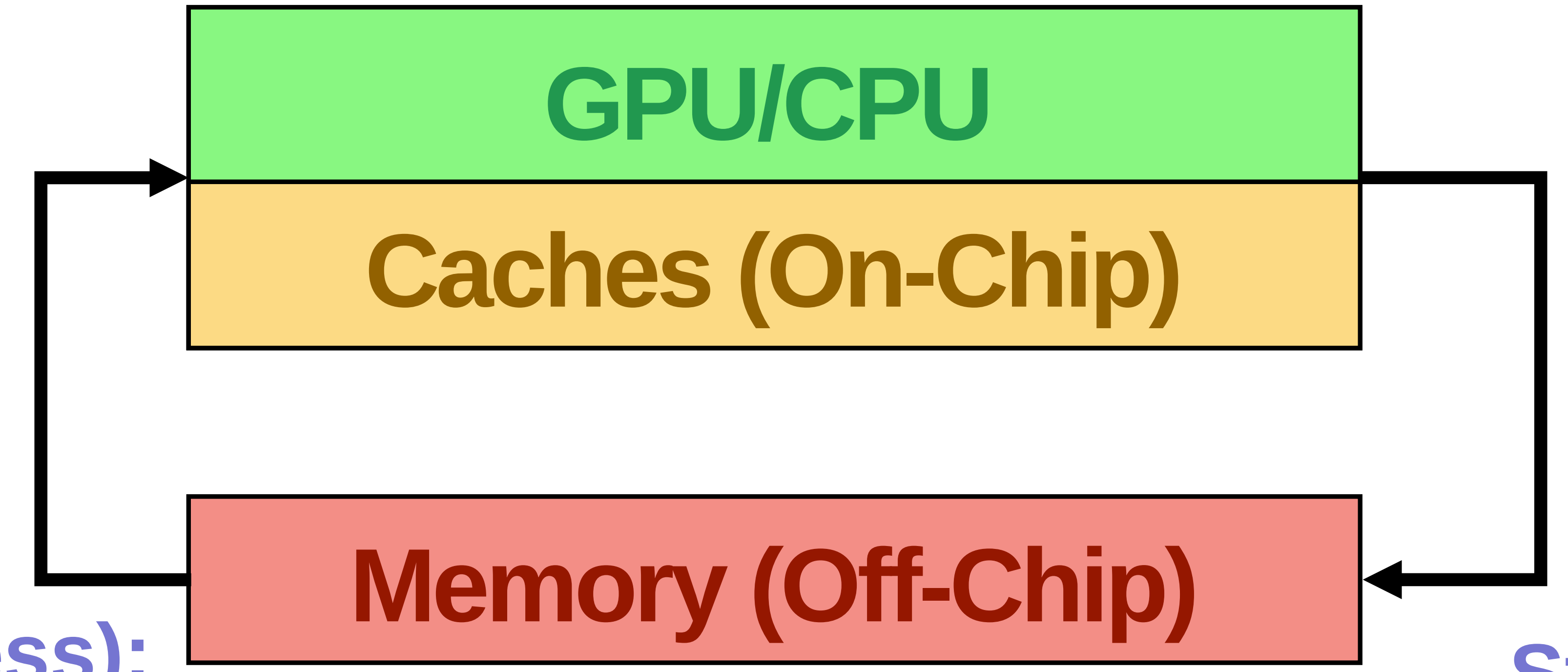
Number of Operations (OP)

- Activations/weights in neural network computing are not always floating-point
- To generalize, number of operations is used to measure the computation amount
  - Example: AlexNet has 724M MACs, total number of floating-point operations will be
    - 724M x 2 = 1.4G OPs
- Similarly, Operations Per Second (OPS)

Roofline Model for Performance Analysis

Process of DNN Inference on Hardware

Step2 (Compute):
Perform the operations
(e.g. Linear, Convolution…etc)



Step1 (Memory Access):
Moving Input/Weights
from memory onto GPU/CPU

Step3 (Memory Access):
Writing the computed
activation back to memory

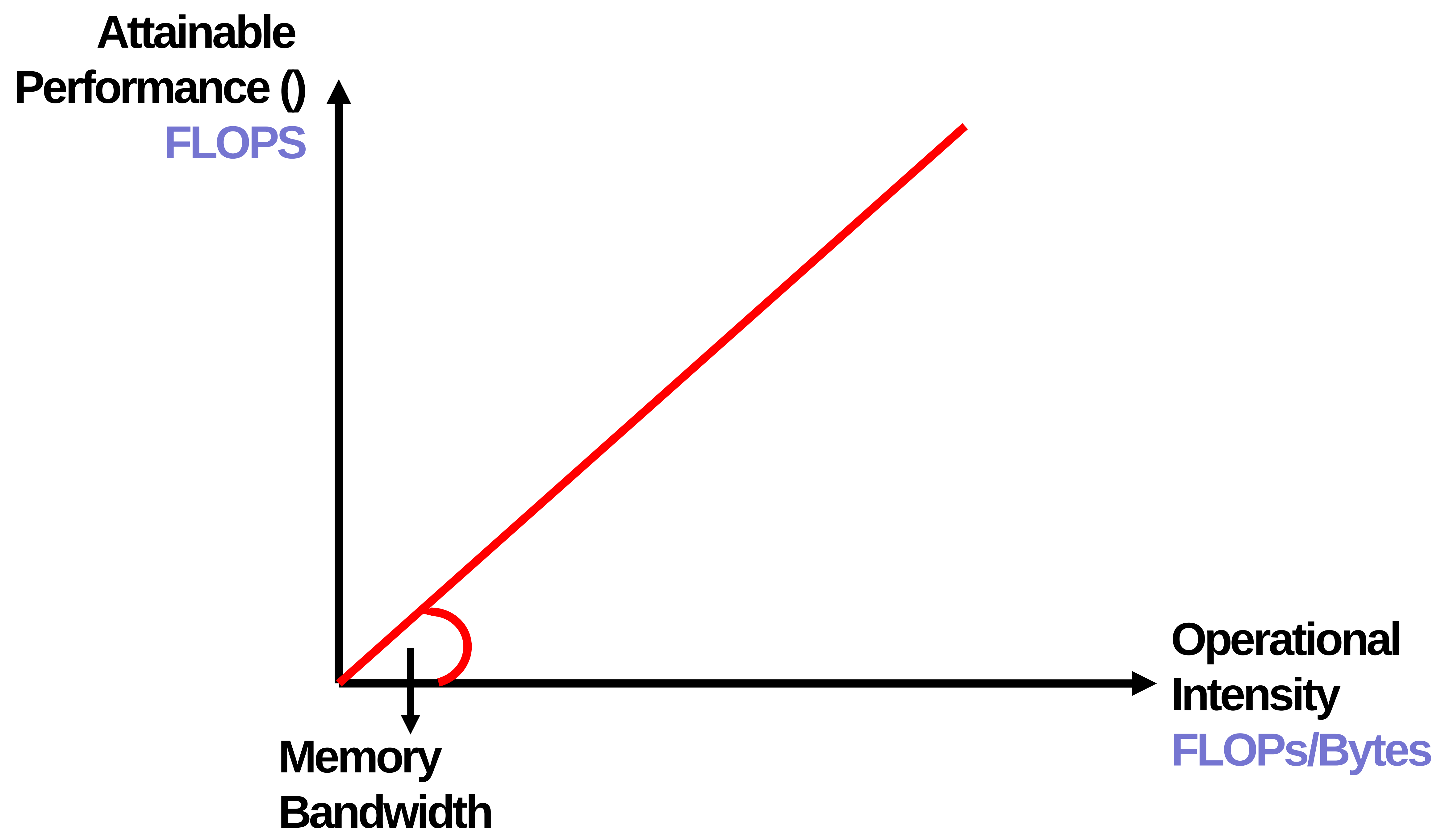
Operational Intensity

- Measure of data locality (data reuse)
- Ratio of Total FLOPs performed to Total Bytes moved

- Compute Intensive Operation
  - High computations (usually high operational intensity)
  - Example: Convolution, Fully-Connected
- Memory Intensive Operation
  - High memory access (usually low operational intensity)
  - Example: Activation

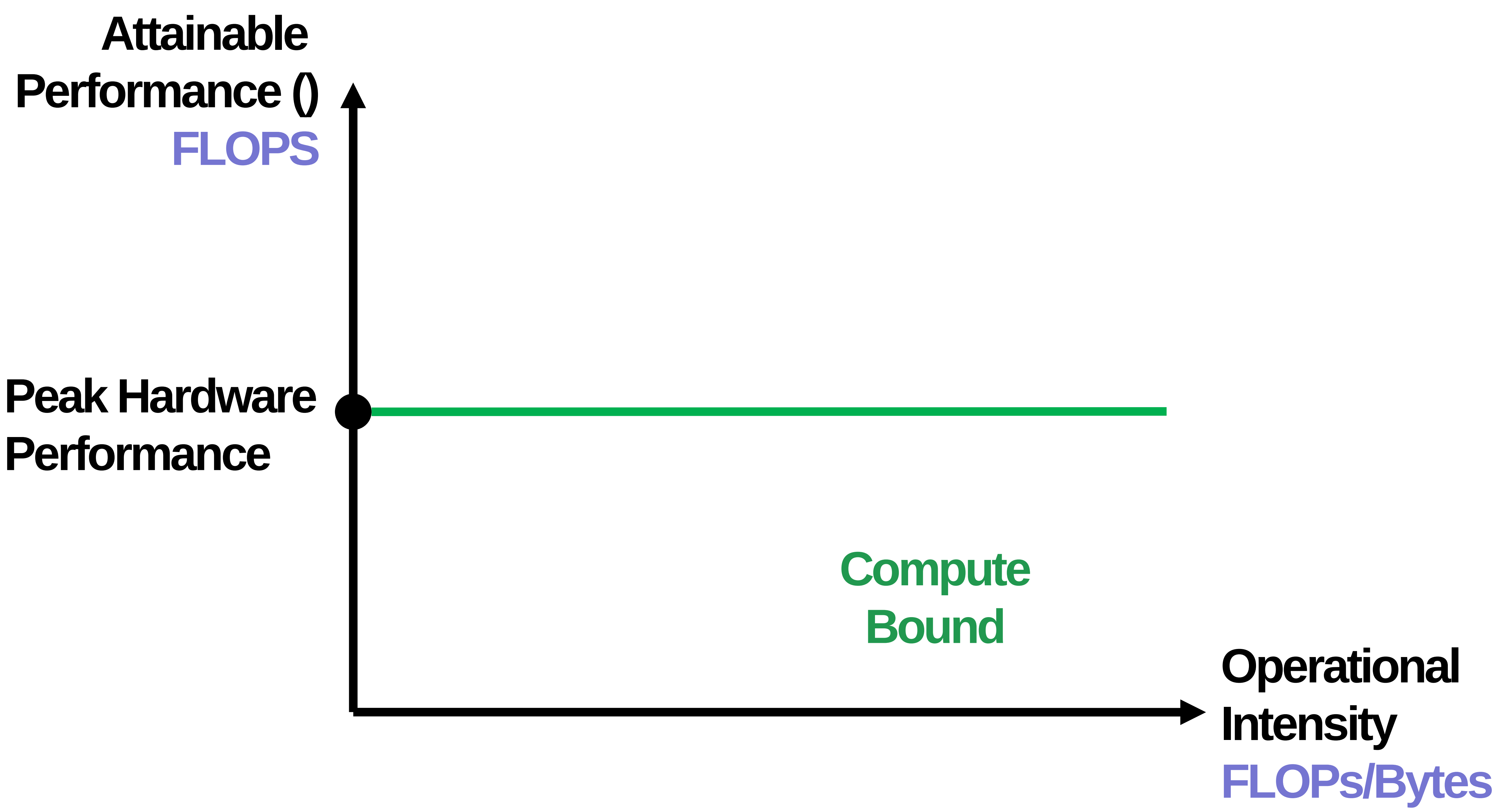
Performance Roofline: Memory Bandwidth

- Performance is bounded by the bandwidth of memory



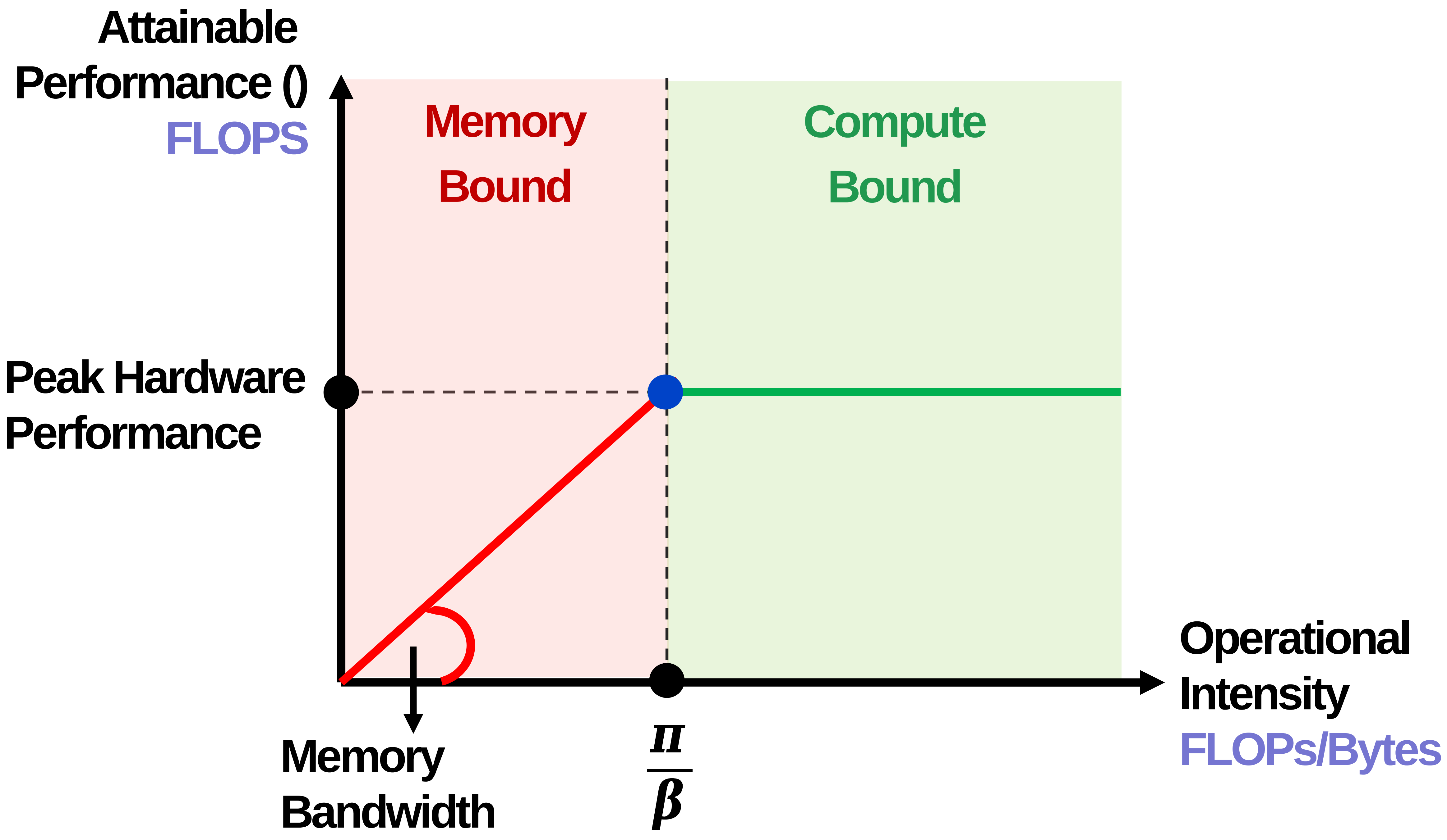
Performance Roofline: Computation Capability

- Performance is bounded by the computation capability of hardware



Roofline Model

- Performance roofline is bounded by "memory" or "compute"

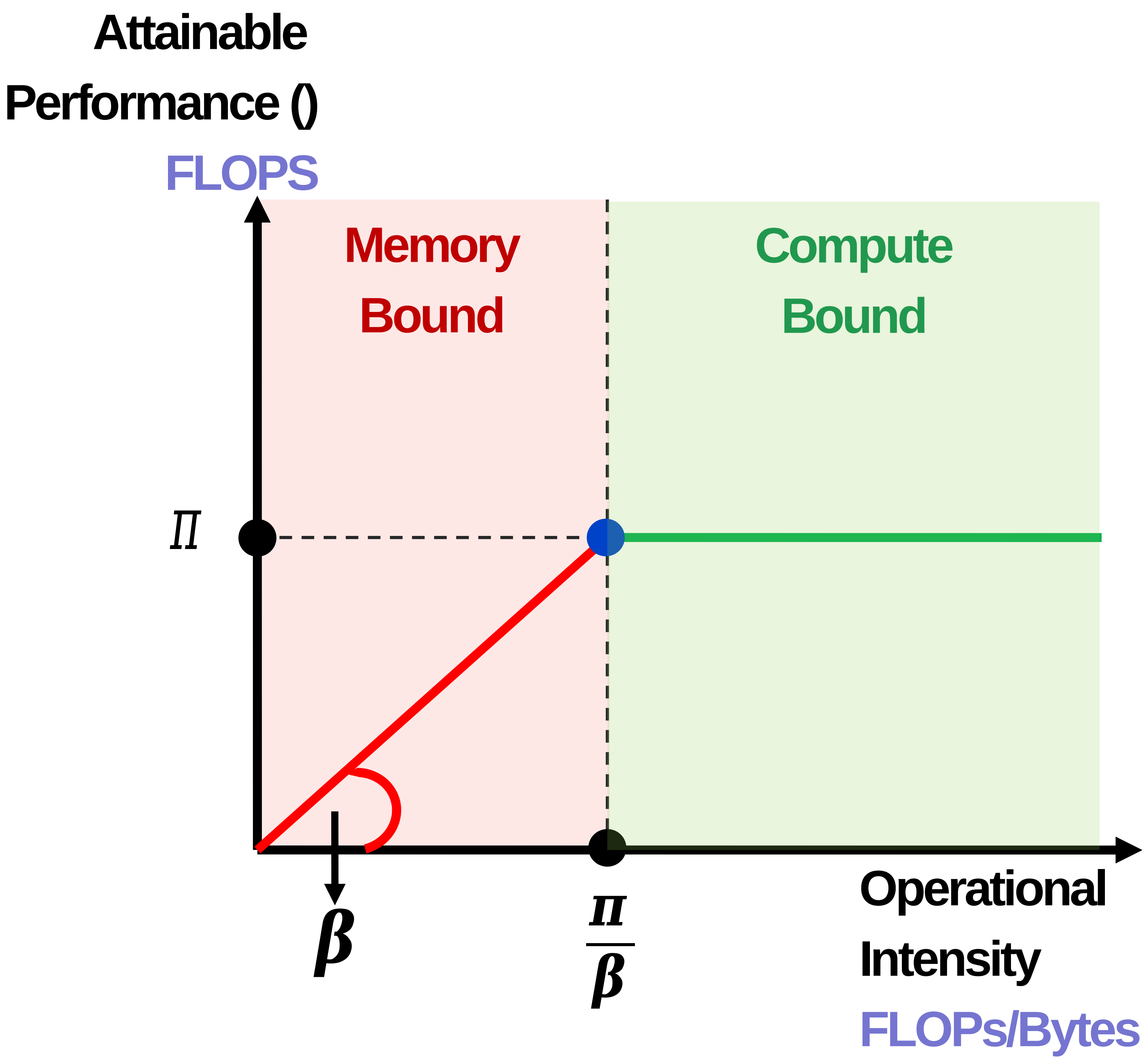


Estimated Execution Time

- The execution time is the number of operations divided by performance

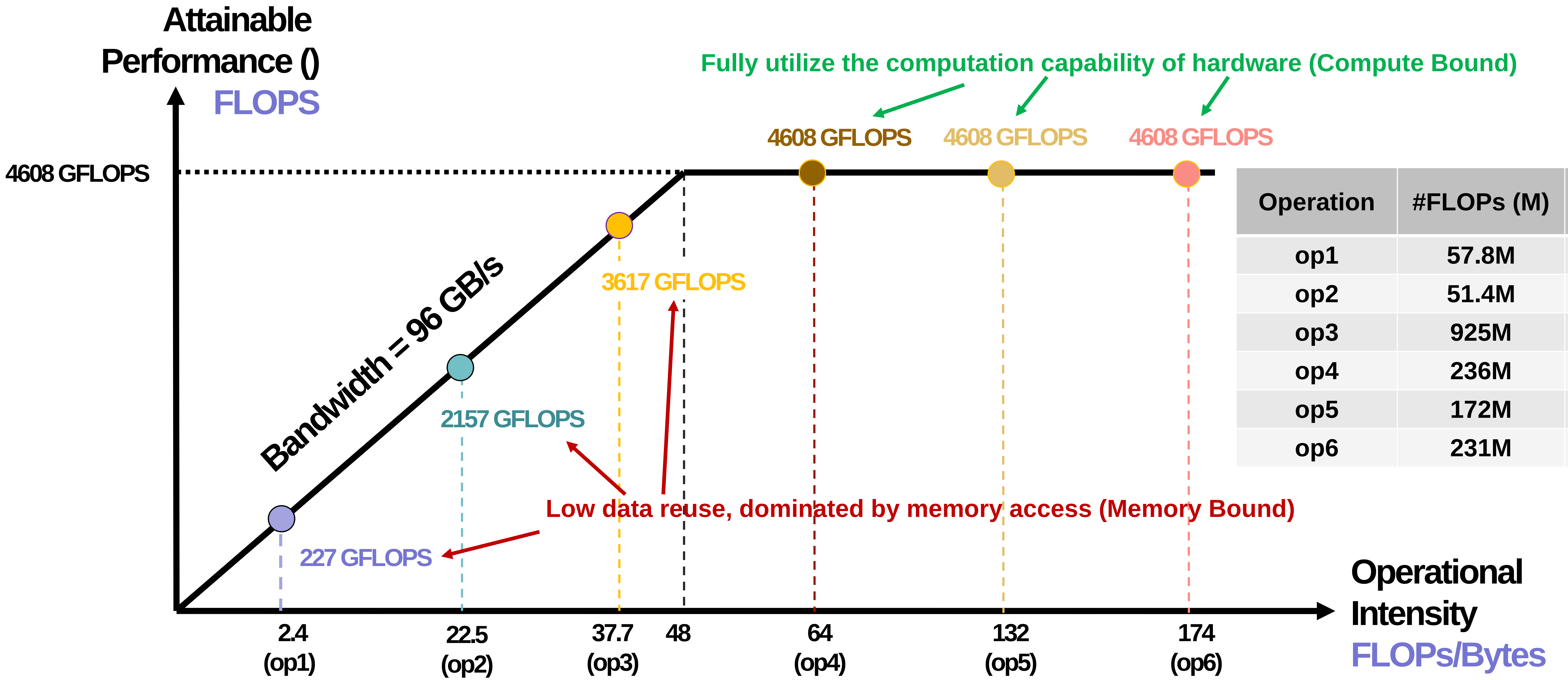
- Memory Bound

- Compute Bound



Example Usage of Roofline Model

- Higher Operational Intensity ✉ Higher Hardware Utilization



Operation	#FLOPs (M)	#Memory Access (MB)
op1	57.8M	25MB
op2	51.4M	2.3MB
op3	925M	25MB
op4	236M	3.7MB
op5	172M	1.3MB
op6	231M	1.3MB

Insight from Roofline Model

- Reduce #Memory Access when it’s memory bound (op1 vs. op2)
  - Reducing #FLOPs results in no speedup (op1 vs. op3)
  - Smaller #FLOPs does not imply faster execution

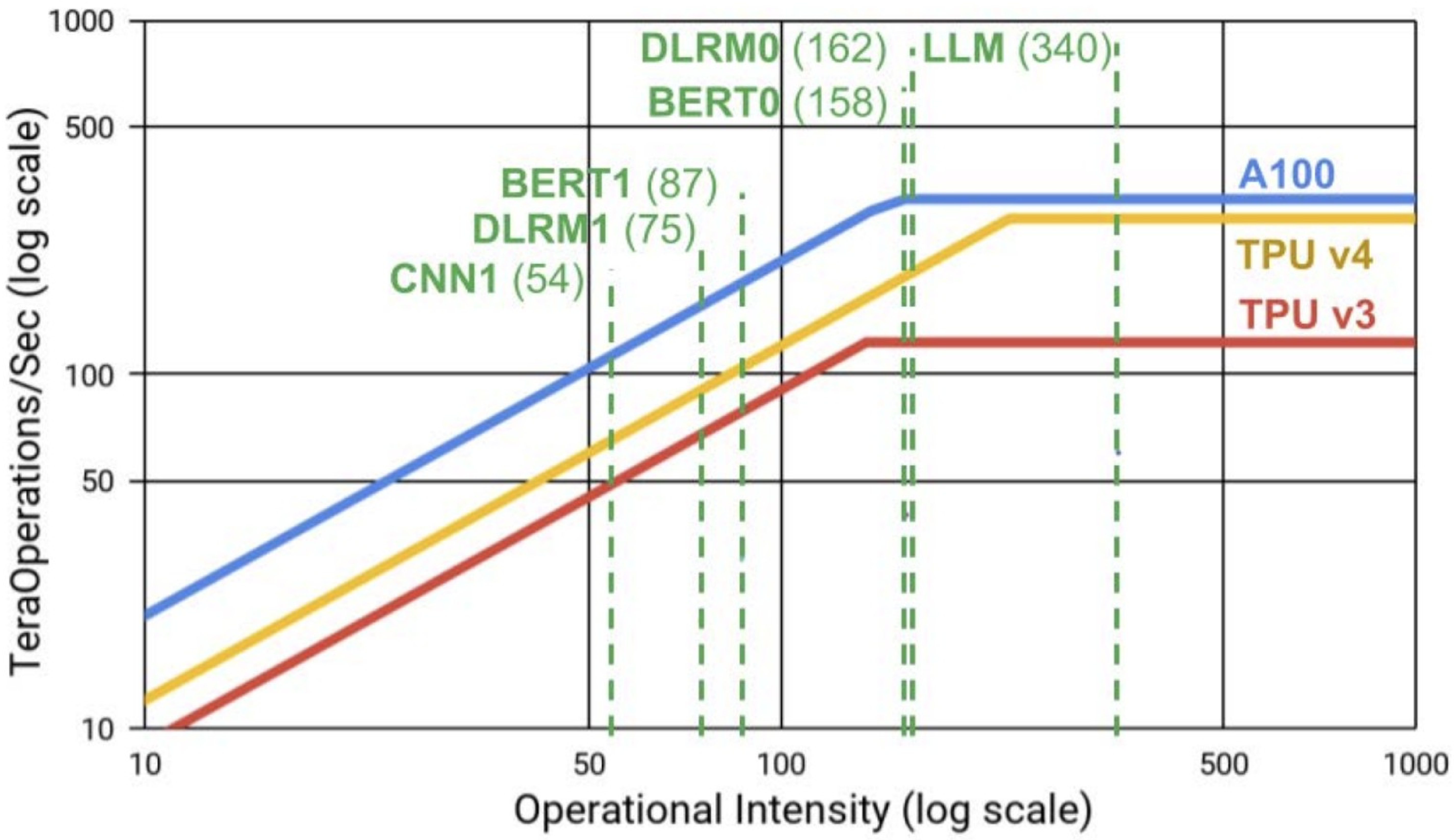
Operation	#FLOPs (M)	#Memory Access (MB)	Operational Intensity	Attainable Performance (GFLOPS)	Theoretical Inference Time
op1	57.8M	24.5MB	2.4	226.5	255
op2	51.4M	2.3MB	22.5	2156.7	23.8
op3	925M	24.5MB	37.7	3622.6	255

Insight from Roofline Model

- Reduce #FLOPs when it's compute bound (op5 vs. op6)
  - Reducing #Memory Access results in no speedup (op4 vs. op6)
  - Smaller #Memory Access does not imply faster execution

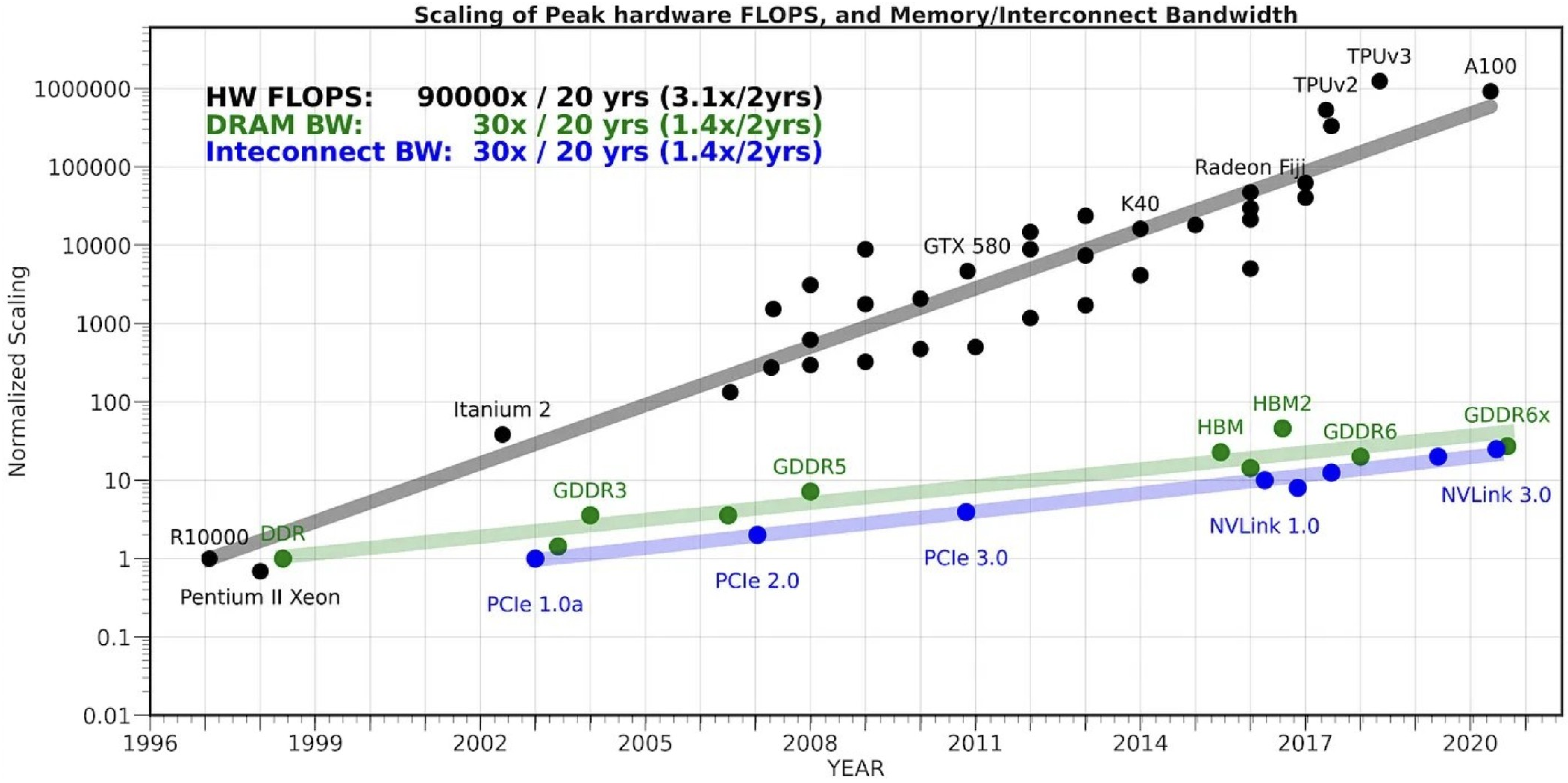
Operation	#FLOPs (M)	#Memory Access (MB)	Operational Intensity	Attainable Performance (GFLOPS)	Theoretical Inference Time
op4	236M	3.7MB	64	4608	51.2
op5	172M	1.3MB	132	4608	37.3
op6	231M	1.3MB	174	4608	50.3

Different Hardware Platforms, Different Rooflines



TPUv4: https://dl.acm.org/doi/pdf/10.1145/3579371.3589350

Memory Wall



Memory Wall and AI: https://medium.com/riselab/ai-and-memory-wall-2cb4265cb0b8