CSDS 451: Designing High Performant Systems for AI

Lecture 9

9/23/2025

Sanmukh Kuppannagari

sanmukh.kuppannagari@case.edu

https://sanmukh.research.st/

Case Western Reserve University

Outline

- Accelerating Convolutional Neural Network Models: Basics

Reading Materials

- Deep Learning, by Ian Goodfellow, Yoshua Bengio, Aaron Courville
  - Available Online: https://www.deeplearningbook.org/
  - CNNs: Chapter 9

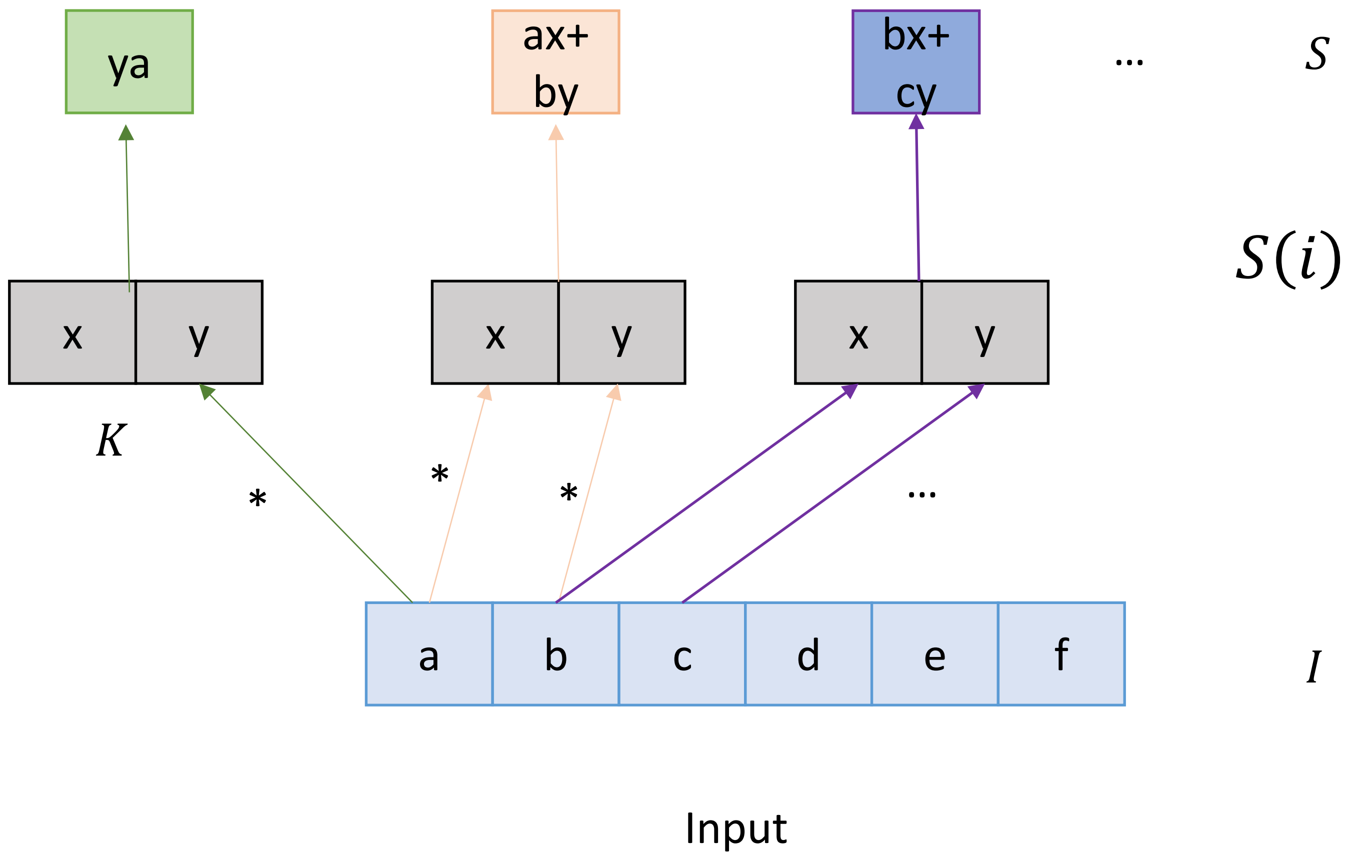
- A Guide to Convolution Arithmetic for Deep Learning
  - Available Online: https://arxiv.org/pdf/1603.07285.pdf

Convolutional Neural Networks

- Neural Network specialized for a grid-like topology
  - 1D: timeseries data, e.g., (12 pm, 63), (1pm, 65), (2 pm, 68),…
  - 2D: Images
  - 3D: Images with multiple channels, e.g., RGB images

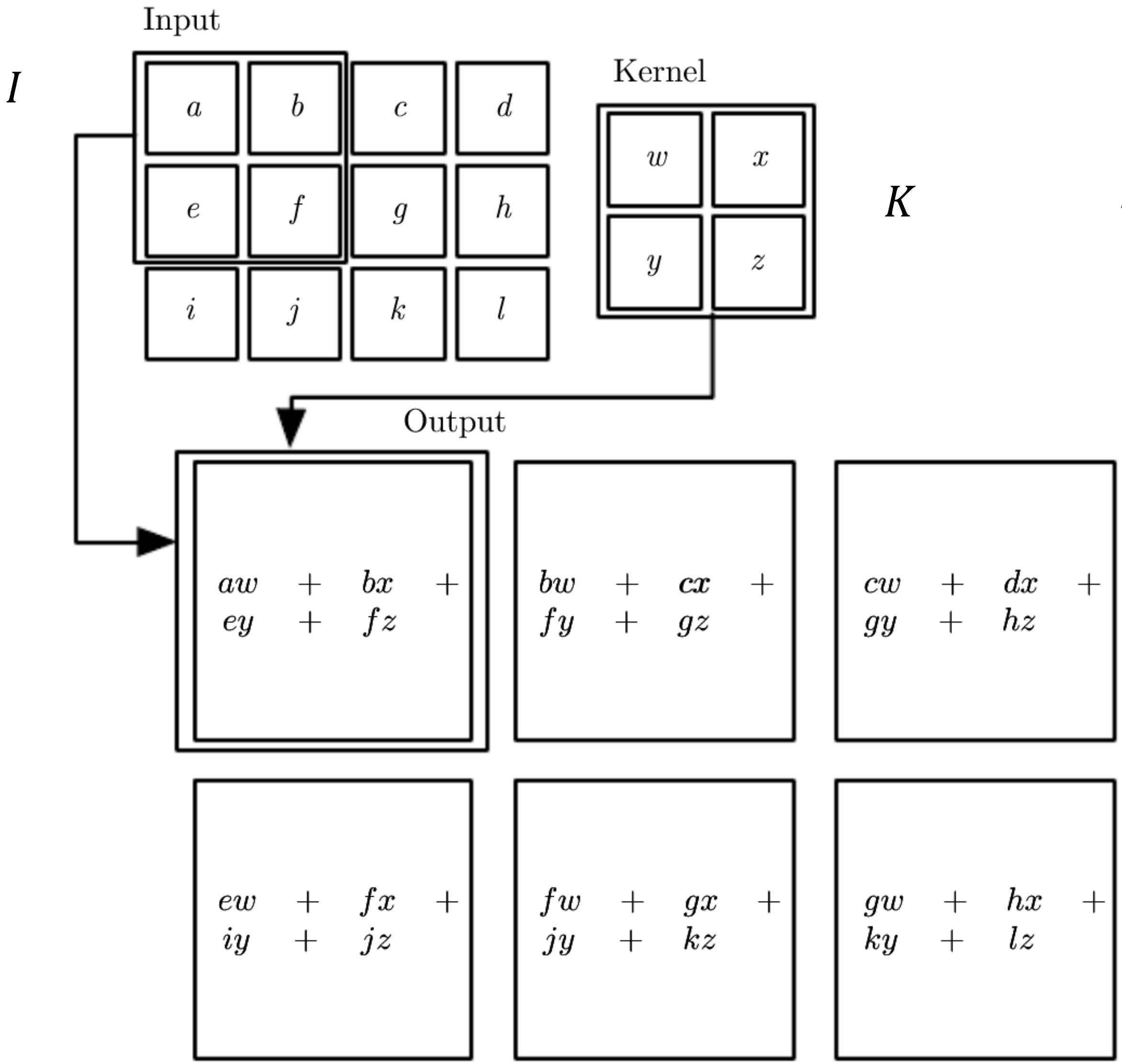
- Name derives from “Convolution” operations performed in the “Convolution” Layers

1D Convolution



$$S(i) = \sum_m I(i+m) * K(m)$$

2D Convolution



$$S(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n)$$

S

Source: Deep Learning Book

What happens for corner cases???

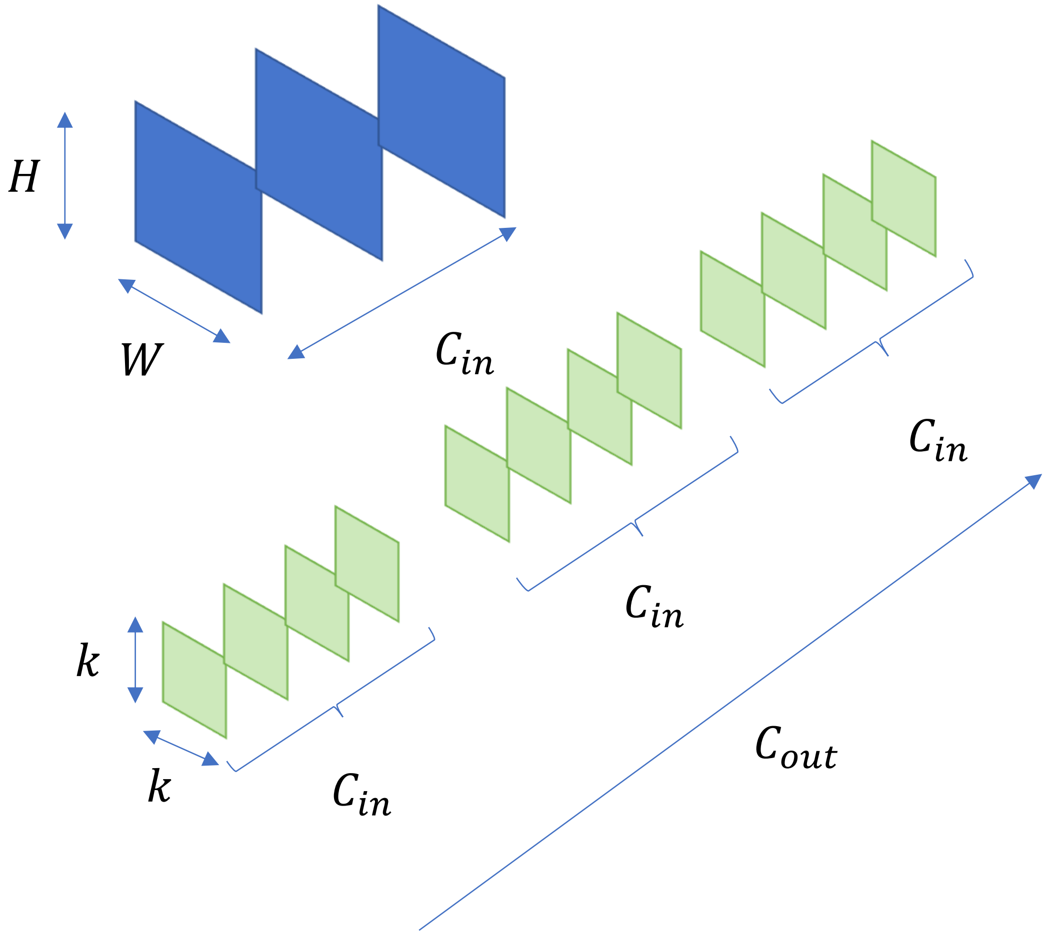
Convolutional Neural Network

- A Neural Network with:
  - Convolution Layers
  - Activations
  - Pooling Layers
  - …

Convolution Layer

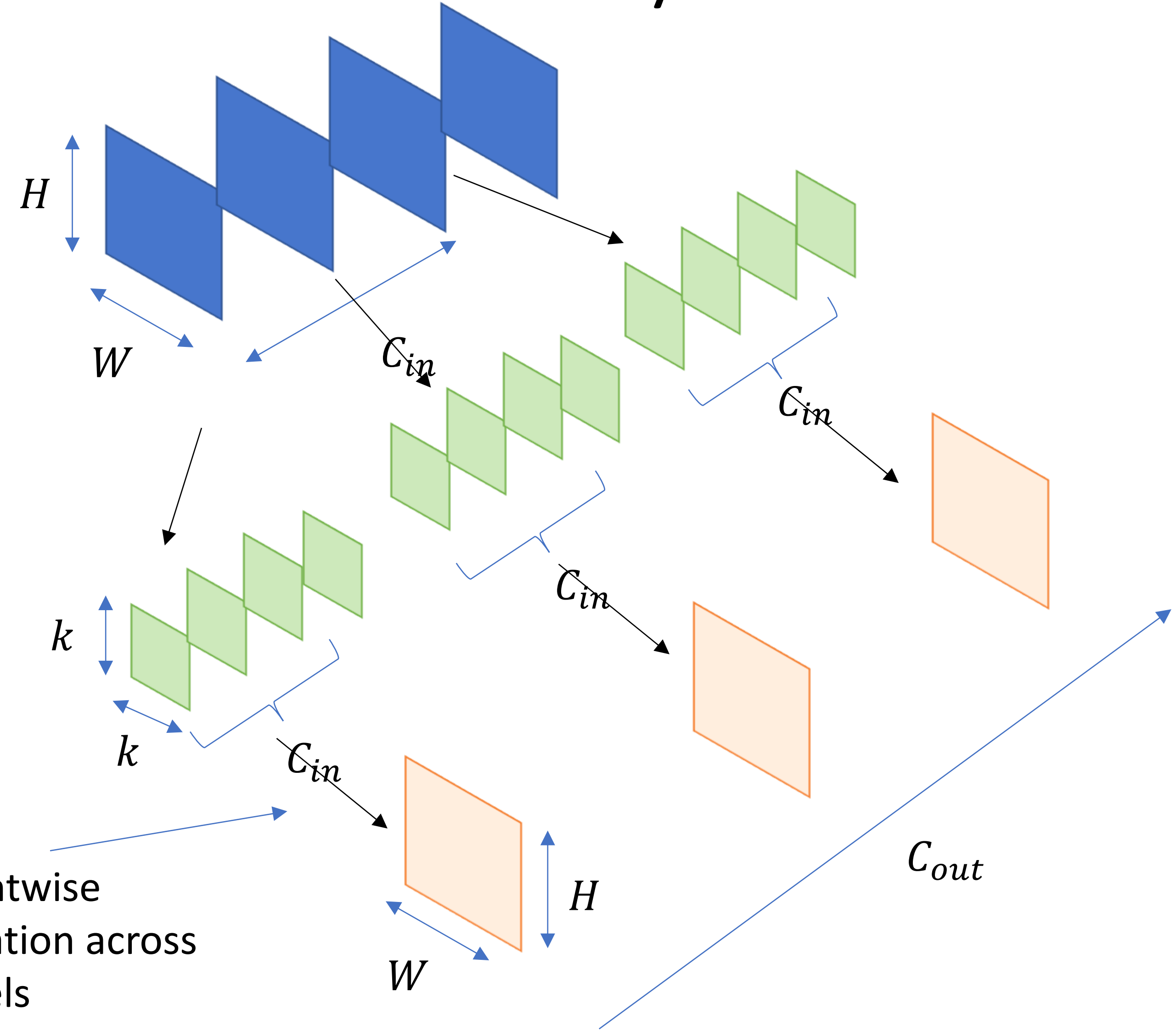
- Given an Input Feature Map
  - Size: $H \times W \times C_{in}$
  - H: Height of the input feature map
  - W: Width of the input feature map
  - C_{in}: Number of Channels in the feature map

- $C_{in} \times C_{out}$ Kernels, each of size $k \times k$



Convolution Layer

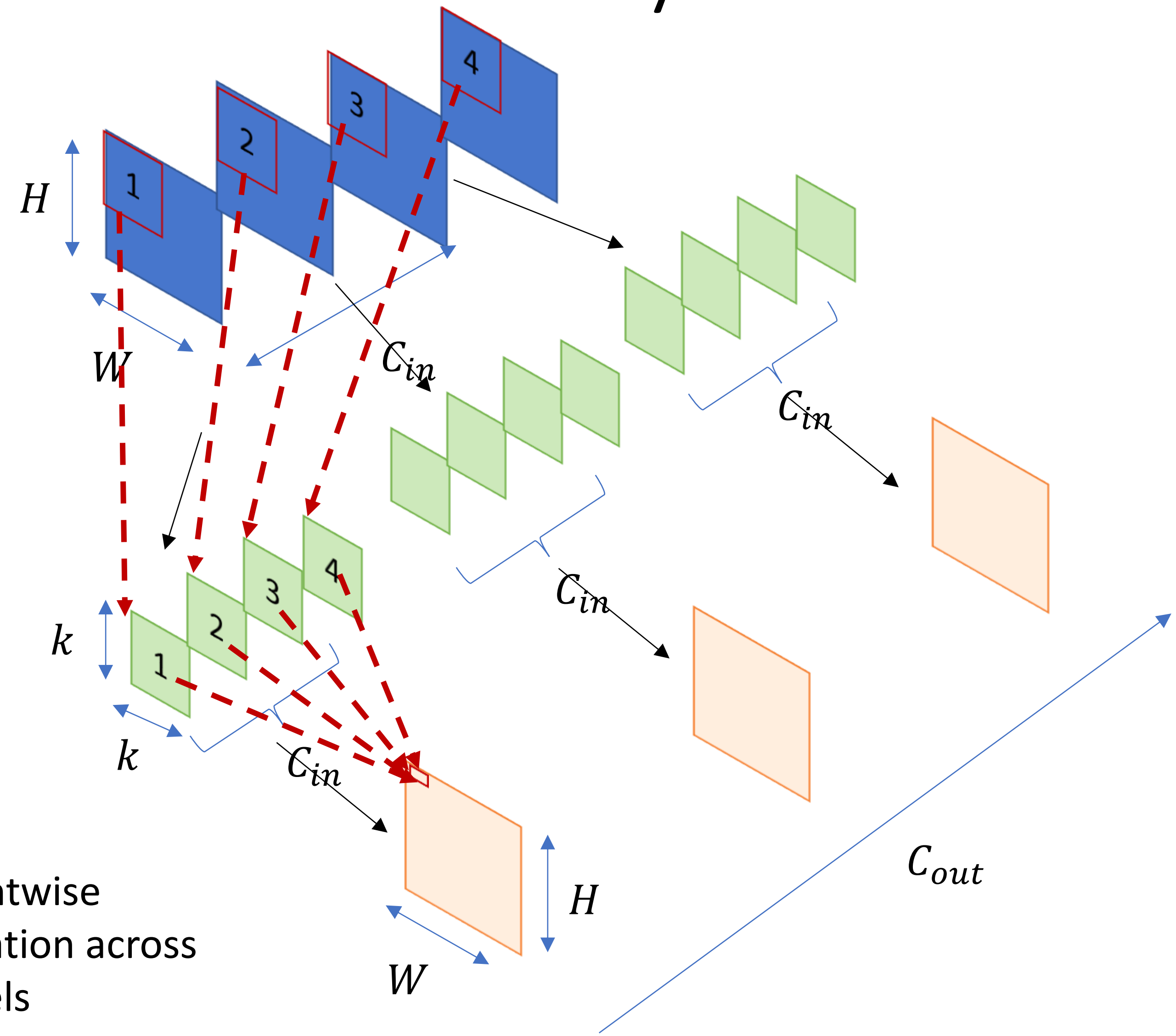
- Produce Output
  - Size: $H \times W \times C_{out}$



Elementwise summation across channels

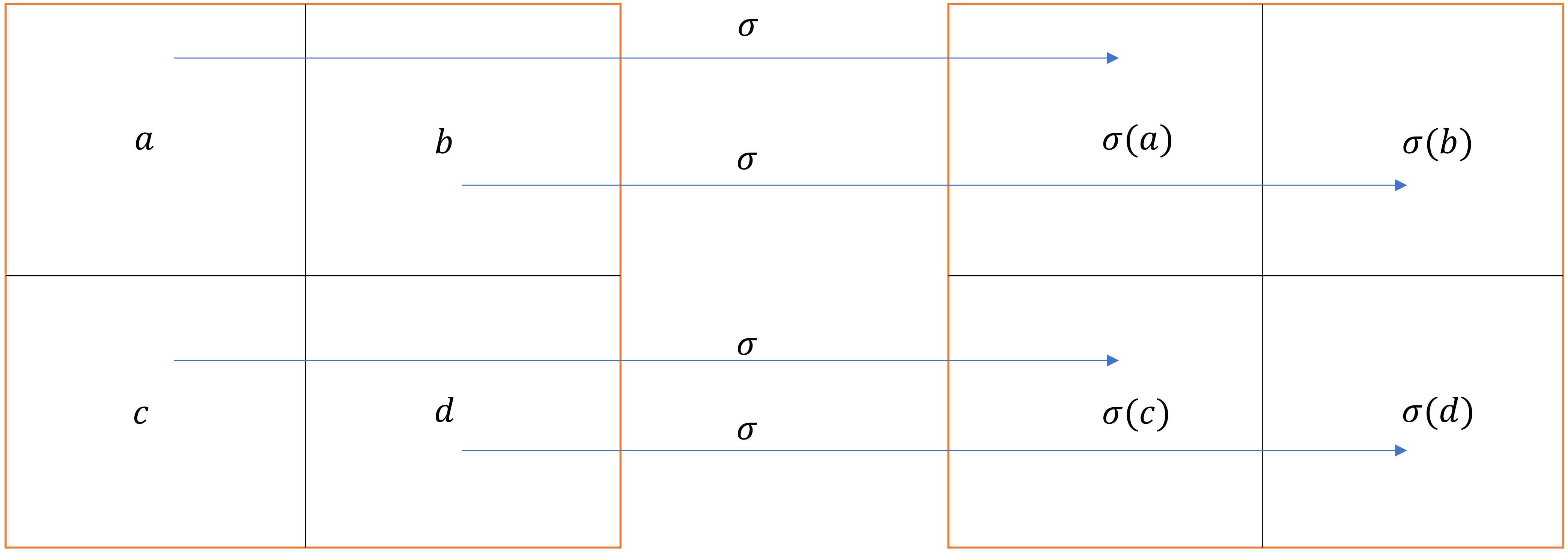
Convolution Layer

- Produce Output
  - Size: $H \times W \times C_{out}$



Elementwise summation across channels

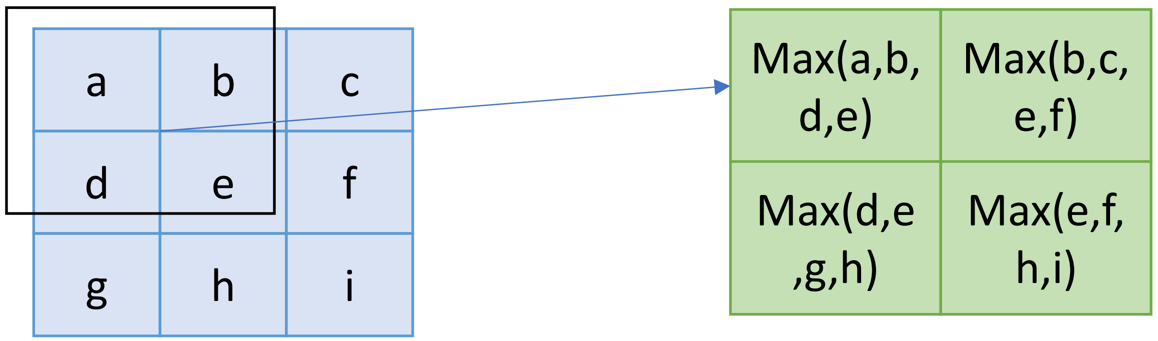
Activation Layer



σ : Activation Function, Sigmoid, ReLU, Softmax

Pooling Layer

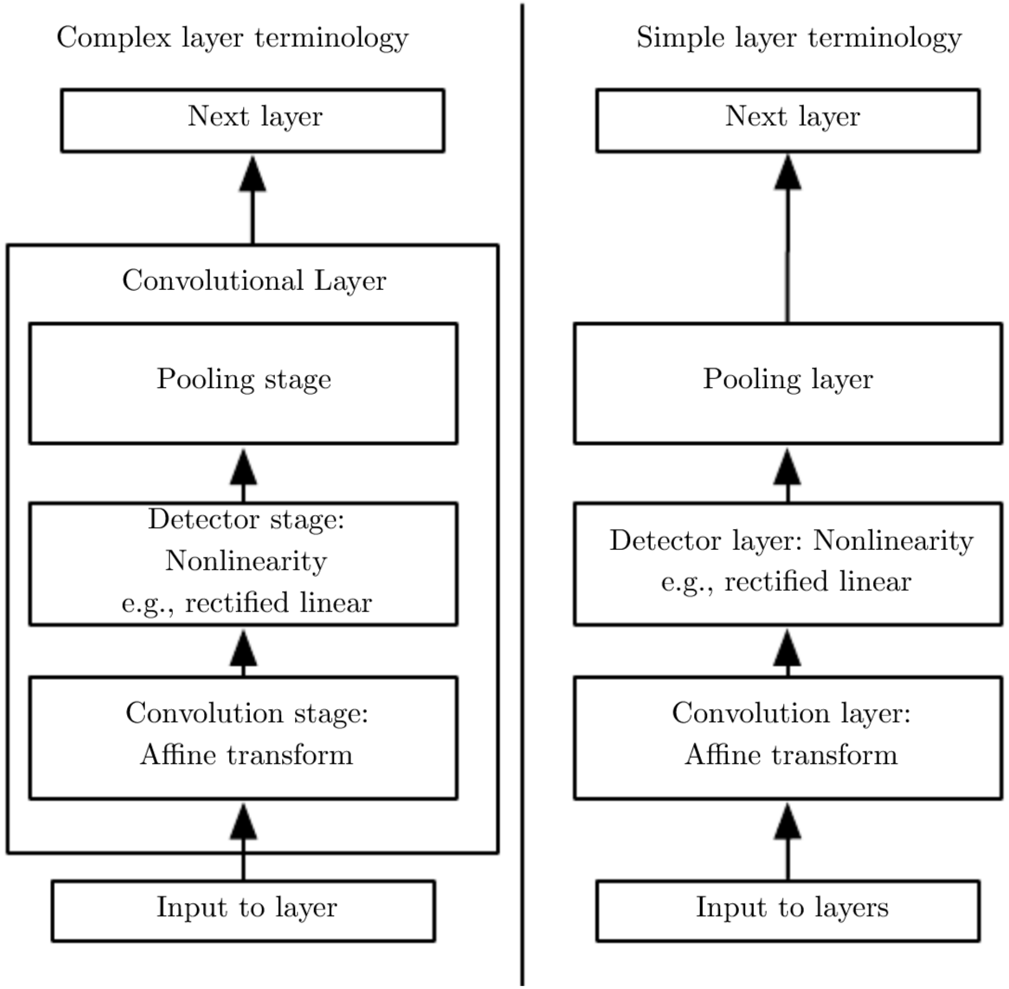
- Replaces the output of a grid with a summary of the grid



Max Pooling: Size
2 X 2

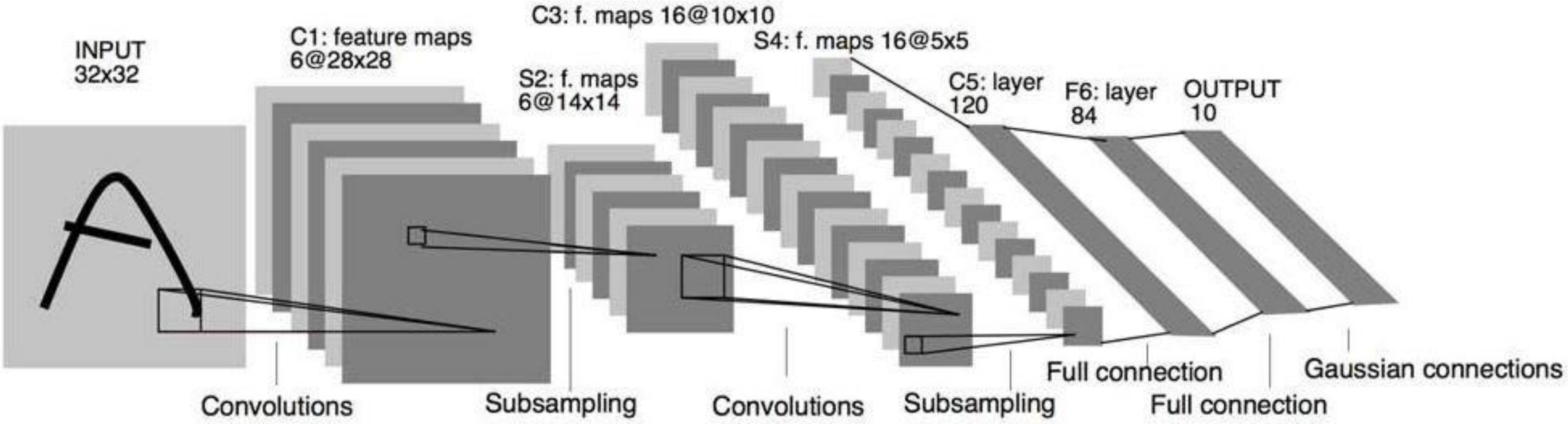
Pooling Types:
Average Pooling
Max Pooling

Convolution Neural Network



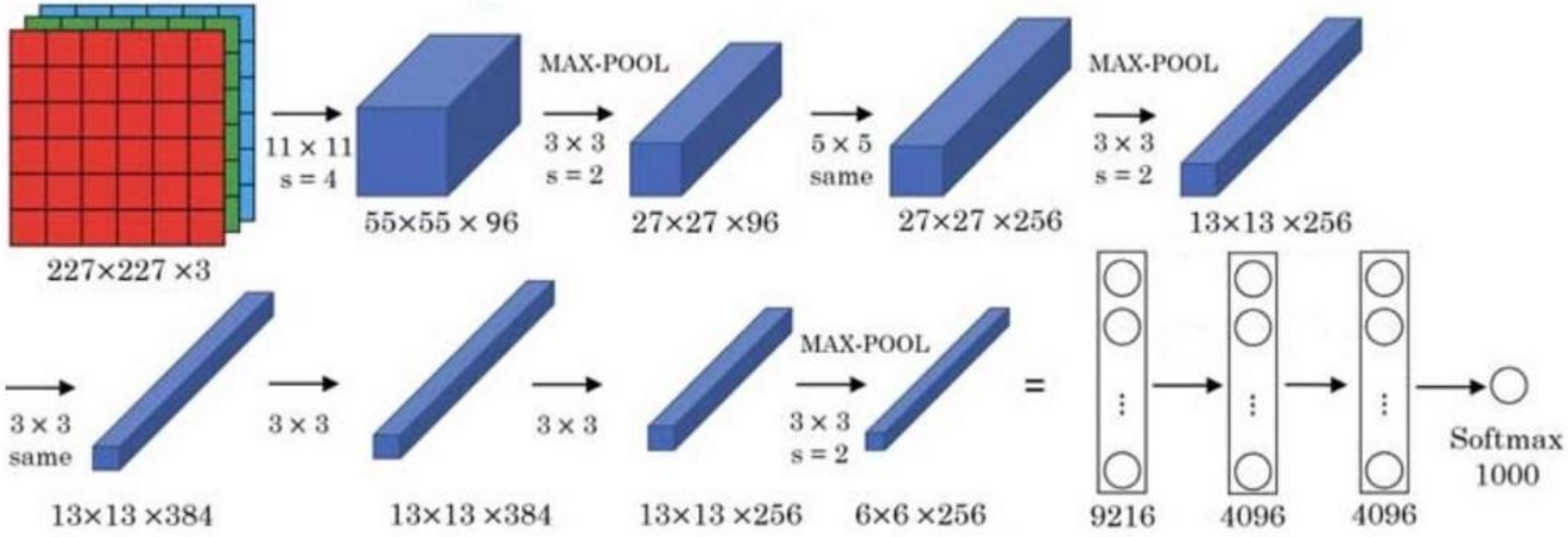
Source: Deep Learning Book

Example CNN Networks



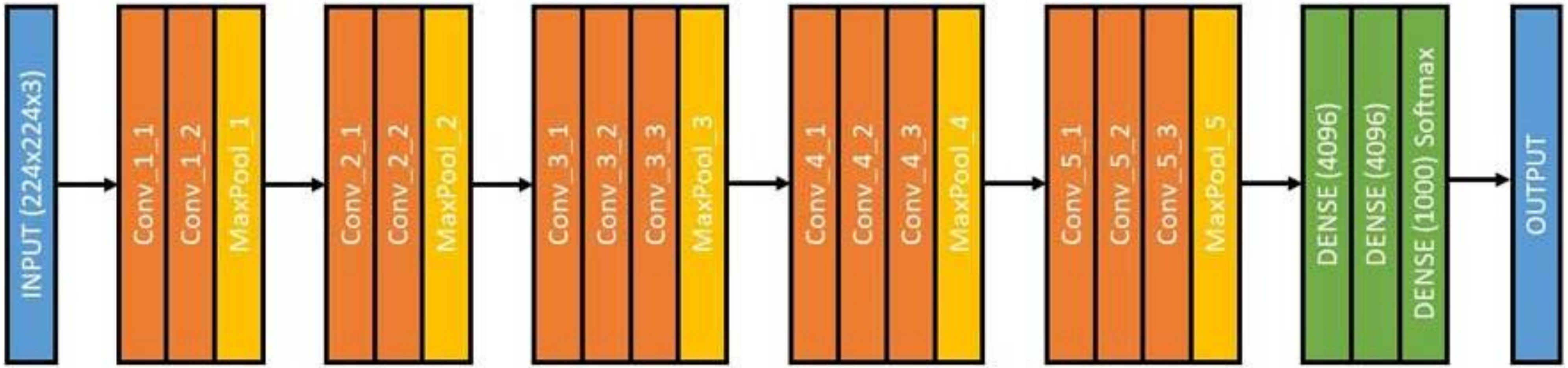
LeNet (1998): Gradient Based Learning Applied to Document Recognitions,
http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf

Example CNN Networks



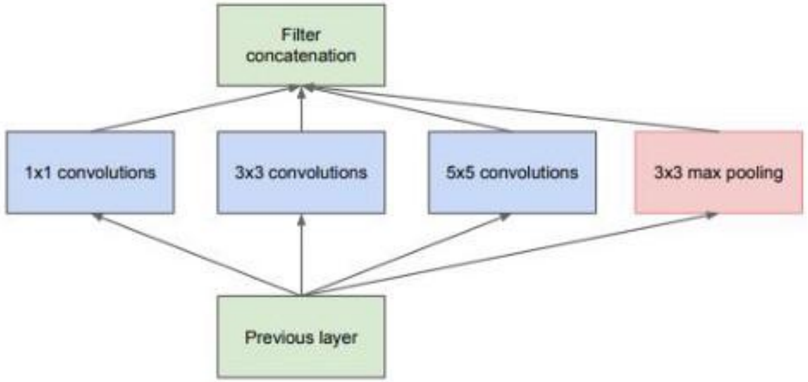
AlexNet (2012): ImageNet Classification with Deep Convolutional Neural Networks, https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

Example CNN Networks

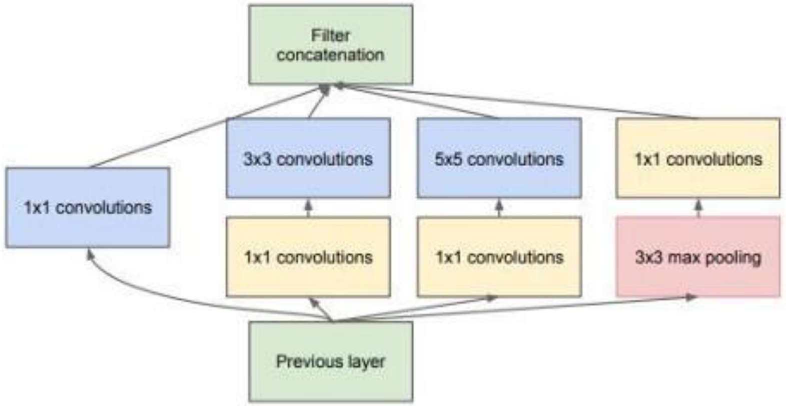


VGG16 (2014): VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION, https://arxiv.org/pdf/1409.1556.pdf

Example CNN Networks

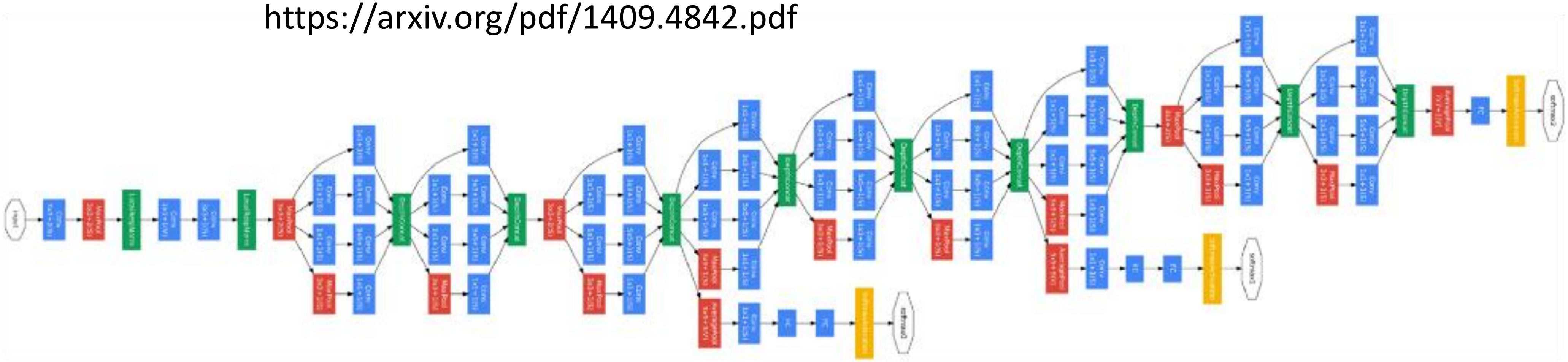


(a) Inception module, naïve version

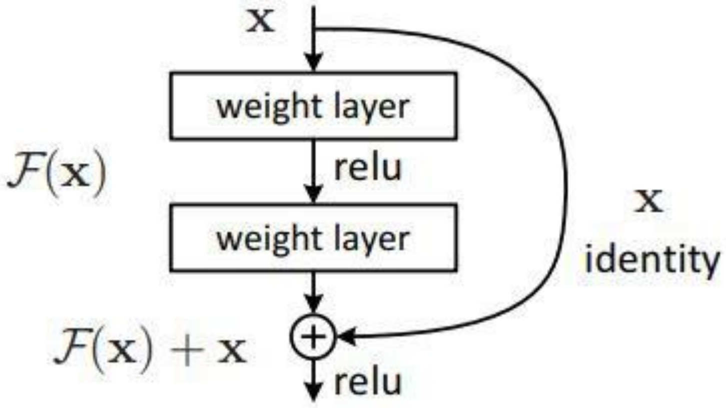


(b) Inception module with dimension reductions

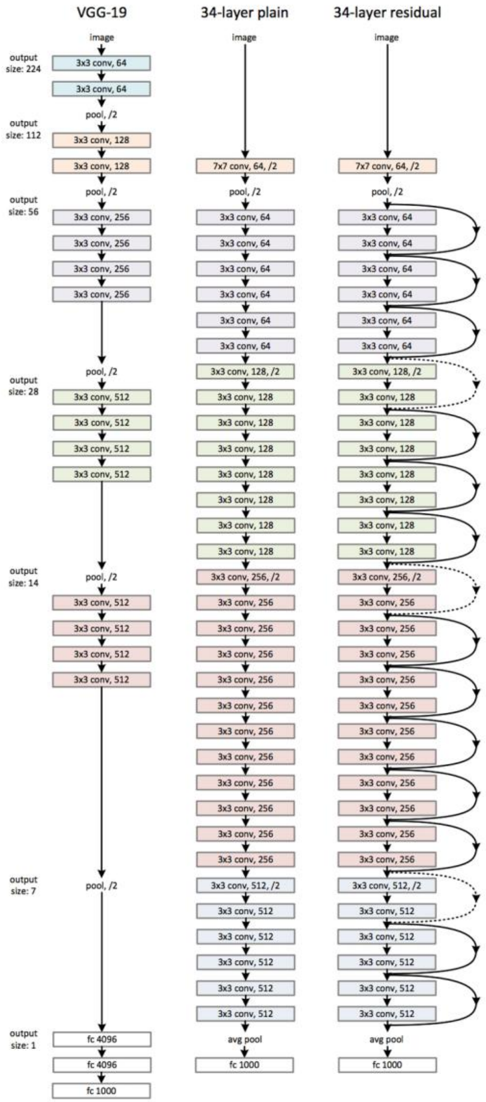
GoogLeNet/InceptionNet (2014): Going deeper with convolutions, https://arxiv.org/pdf/1409.4842.pdf



Example CNN Networks

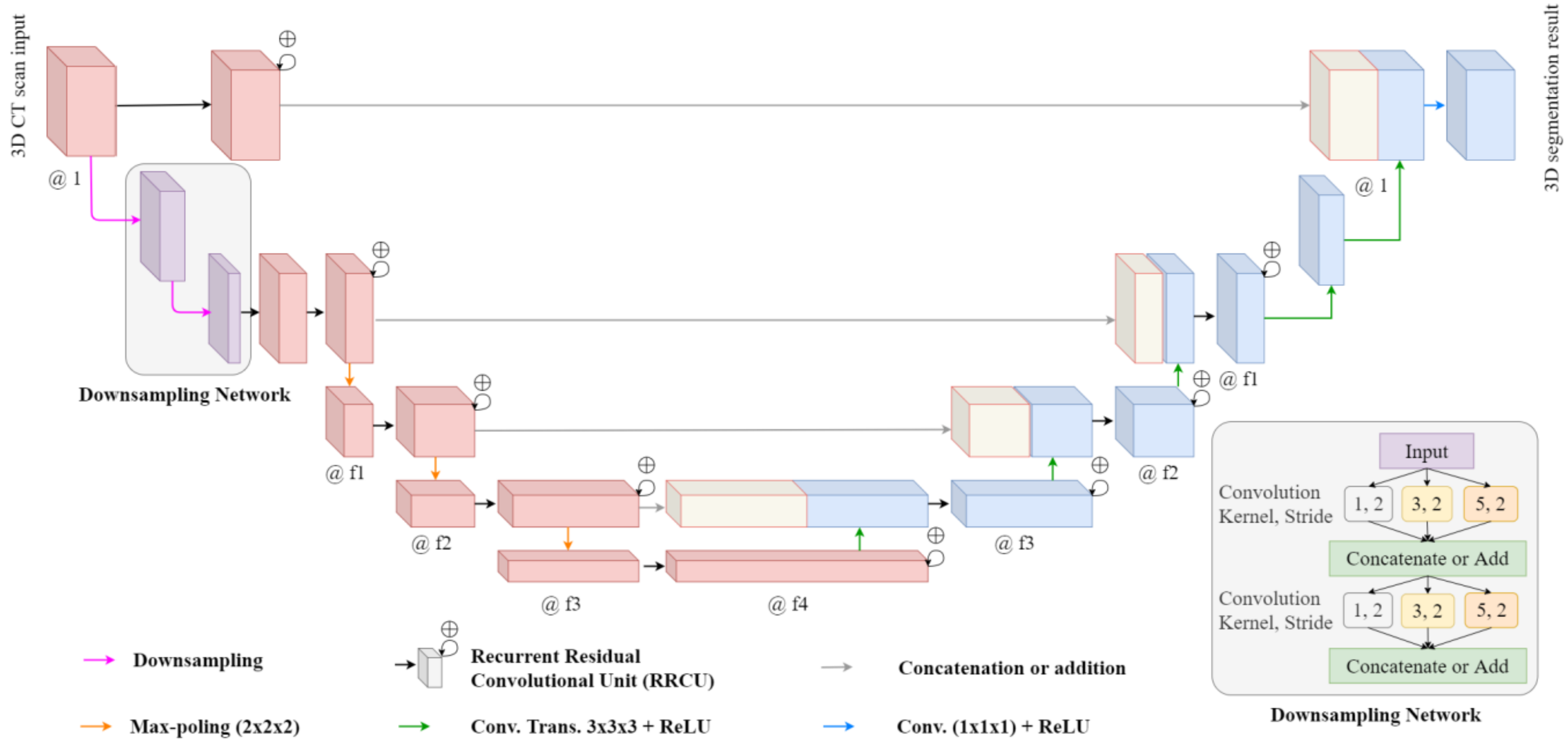


ResNet (2015): Deep Residual Learning for Image Recognition,
https://arxiv.org/pdf/1512.03385.pdf



CASE SCHOOL OF ENGINEERING
CASE WESTERN RESERVE UNIVERSITY

Example CNN Networks



U-net: State of the art in image segmentation, especially in biomedical domain

Example CNN Networks

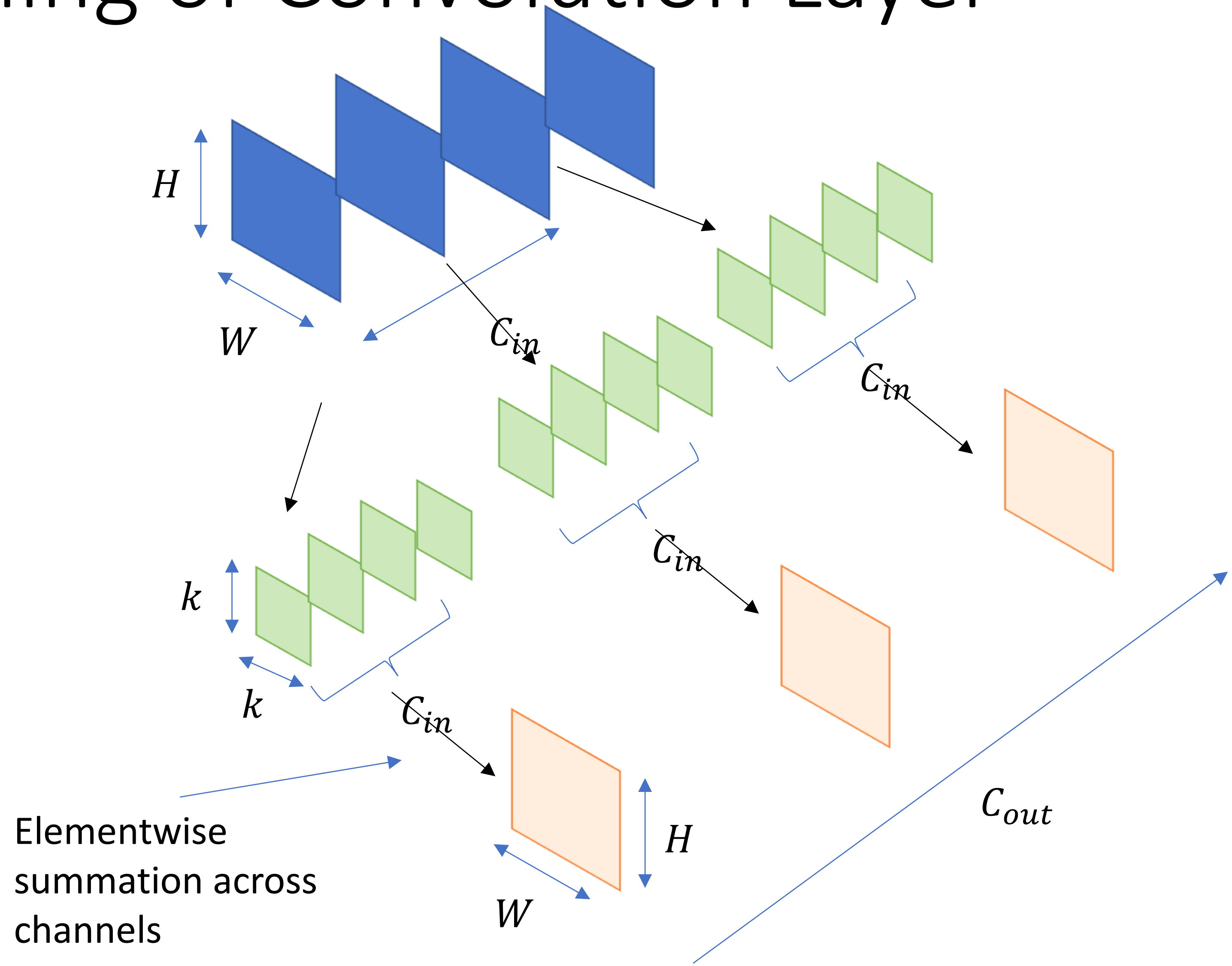
- EfficientNet (2019): Tan, M., & Le, Q. V. (2020). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *ArXiv:1905.11946 [Cs, Stat]*. http://arxiv.org/abs/1905.11946

- FixEfficientNets (2022): Touvron, H., Vedaldi, A., Douze, M., & Jégou, H. (2020b). Fixing the train-test resolution discrepancy: FixEfficientNet. ArXiv:2003.08237 [Cs]. http://arxiv.org/abs/2003.08237

- AmoebaNets (2019): Regularized Evolution for Image Classifier Architecture Search, https://ojs.aaai.org/index.php/AAAI/article/view/4405

Performance Modeling of Convolution Layer

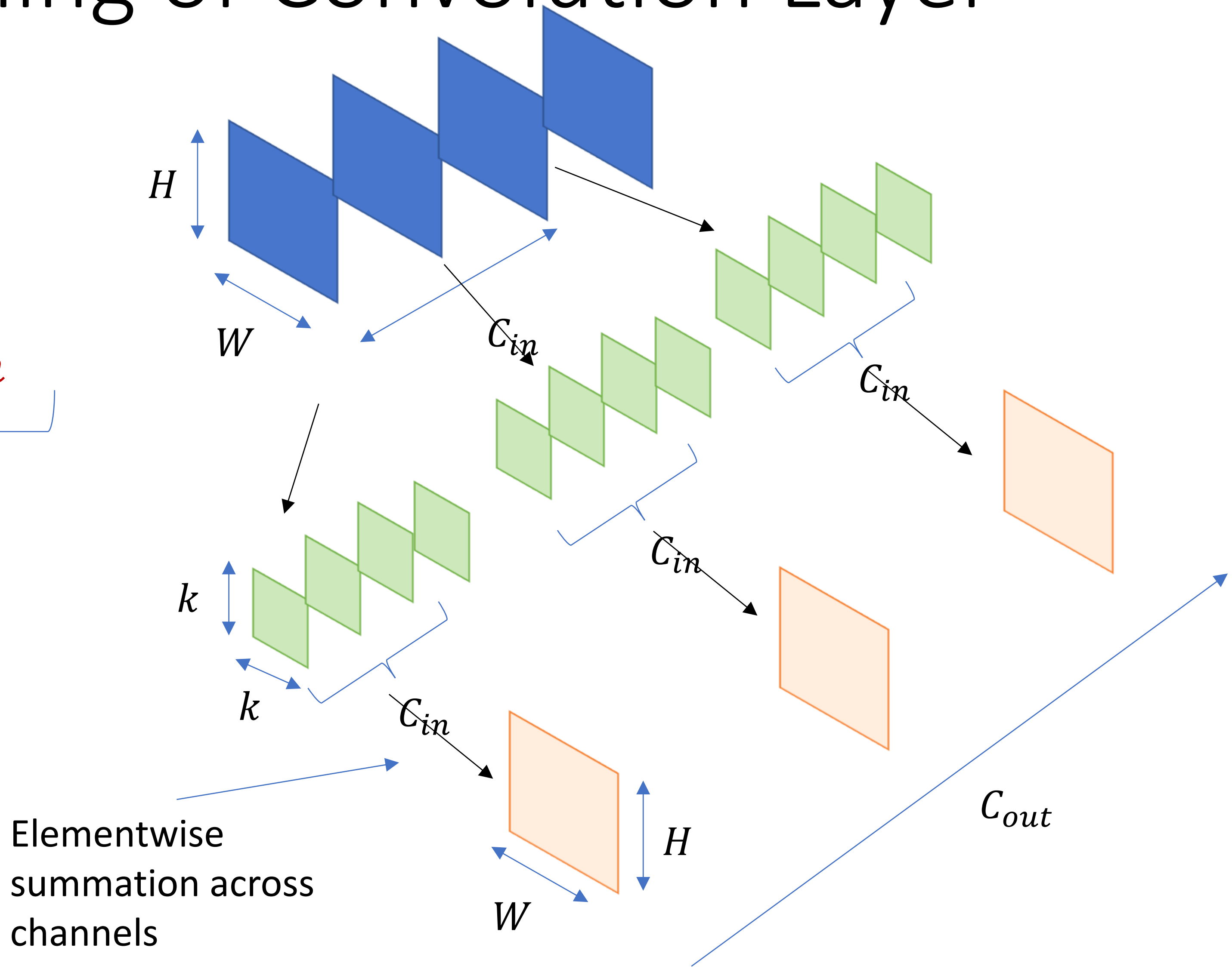
- Number of Operations: ??



Performance Modeling of Convolution Layer

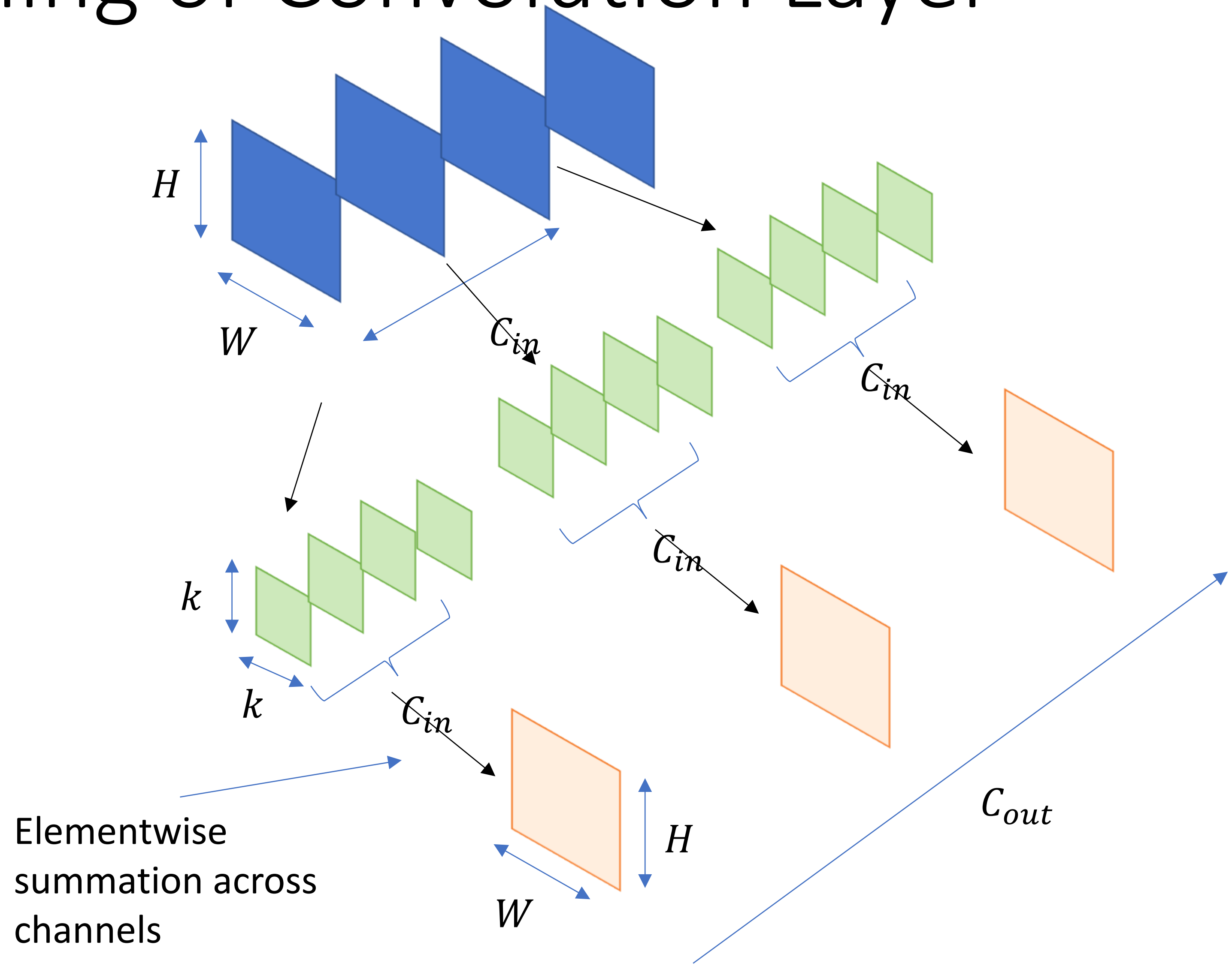
- Number of Operations:

- $$\underbrace{H \times W \times C_{out}}_{\text{Output Pixels}} \times \underbrace{k \times k \times C_{in}}_{\text{Operations per Pixels}}$$



Performance Modeling of Convolution Layer

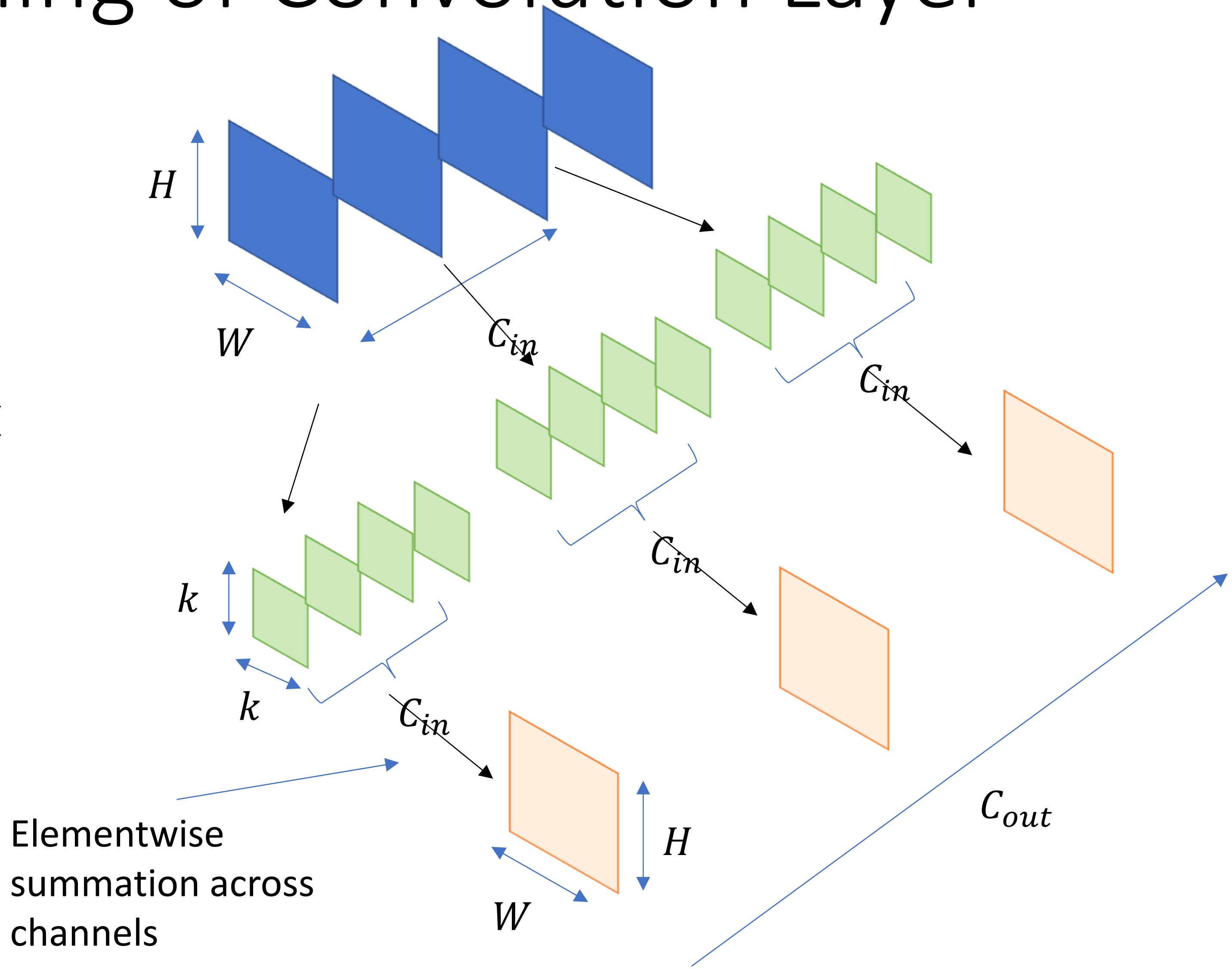
- Memory Transfers (infinite cache): ??



Performance Modeling of Convolution Layer

- Memory Transfers (infinite cache): ??

- $H \times W \times C_{in} + C_{in} \times C_{out} \times k \times k + H \times W \times C_{out}$
- $\approx H \times W \times (C_{in} + C_{out})$



Performance Modeling of Convolution Layer

- Theoretical Arithmetic Intensity

$$\frac{k^2 C_{in} C_{out}}{C_{in} + C_{out}}$$

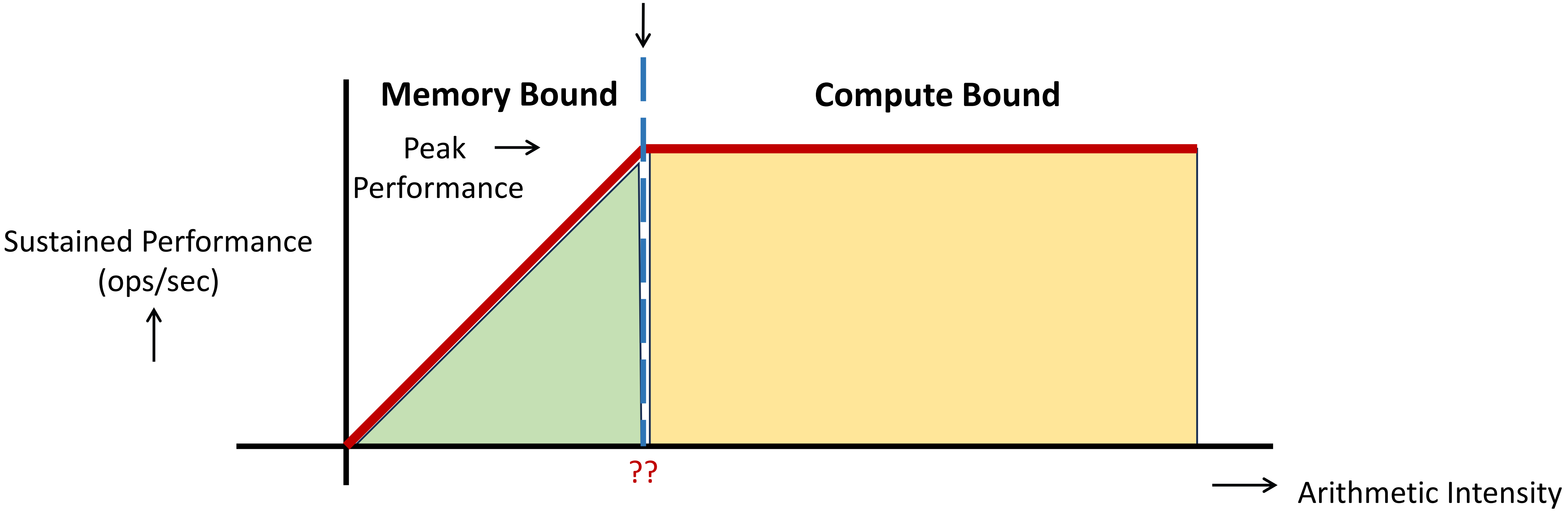
Example value for $k = 9, C_{in} = 3, C_{out} = 64$

232

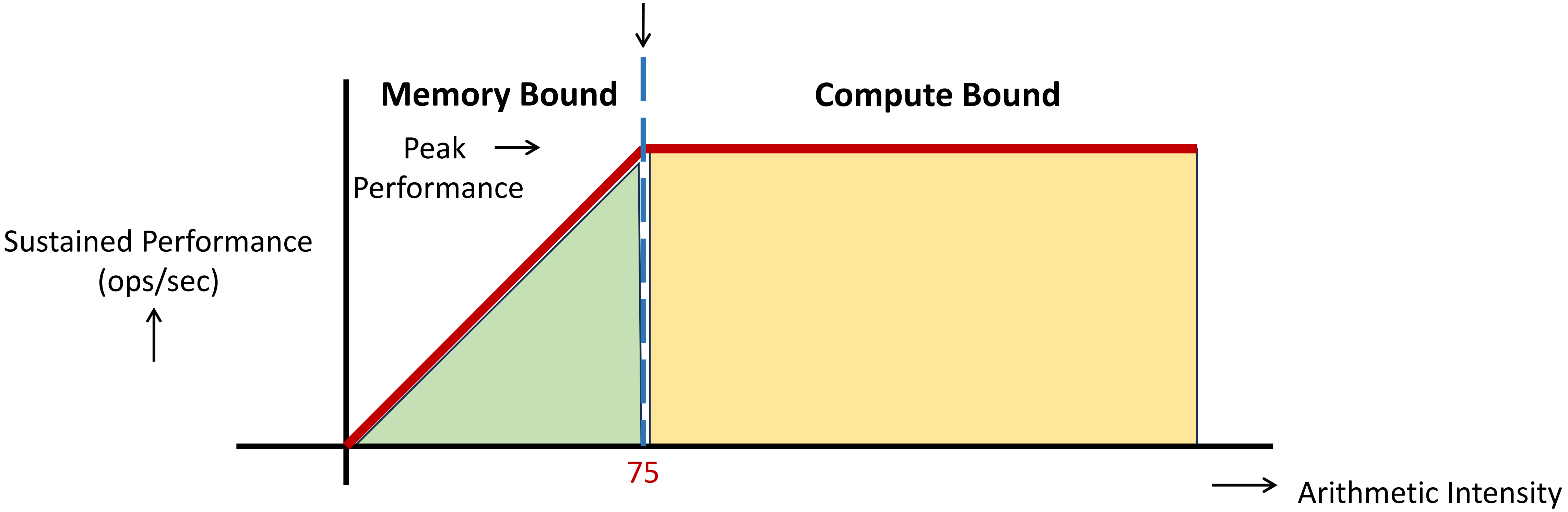
Example Machine Intensity = 75 (Nvidia A100*) how???

*https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/a100/pdf/nvidia-a100-datasheet-nvidia-us-2188504-web.pdf

CNN on A100 Roofline Plot (assuming infinite cache)

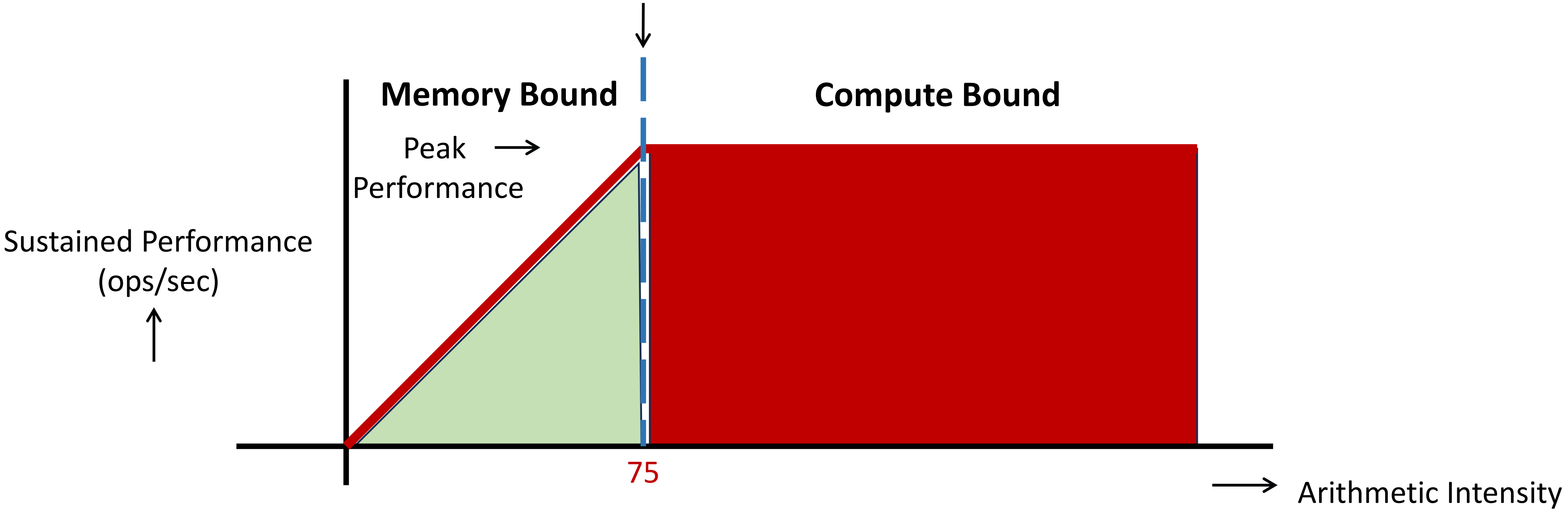


CNN on A100 Roofline Plot (assuming infinite cache)



Which area will the CNN Performance lie in?

CNN on A100 Roofline Plot (assuming infinite cache)



Which area will the CNN Performance lie in? Compute Bound

Implications

- Essentially a Compute Bound Problem, assuming large cache

- Objective: How to ensure efficiency is $O(1)$, i.e., all compute units are fully utilized

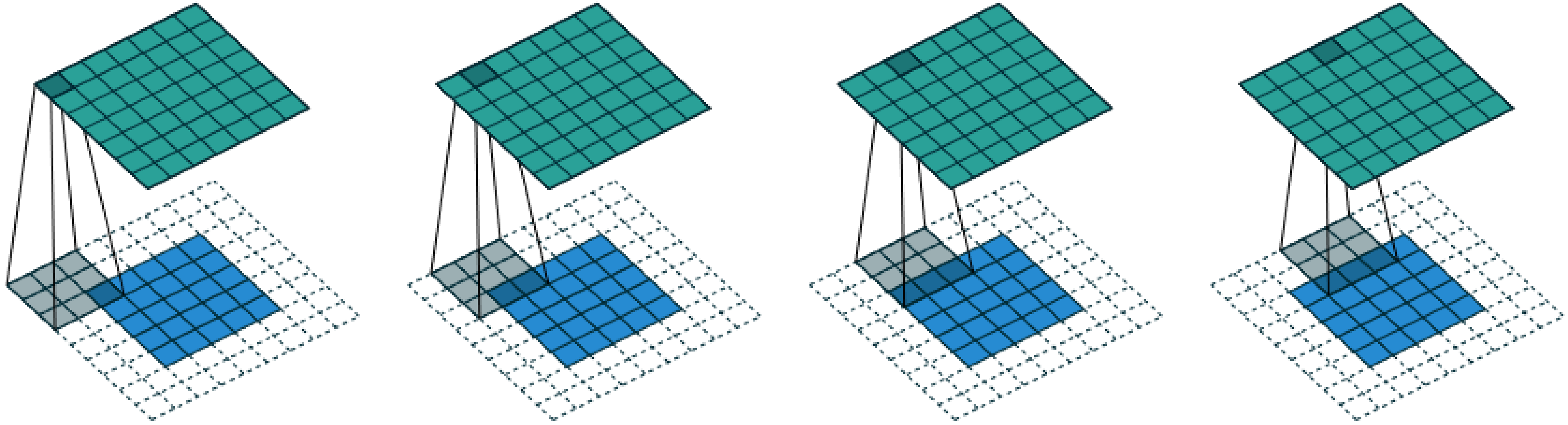
- If cache is limited
  - Objective 1: How to improve data reuse so that compute units have enough to work on?
  - Objective 2: How to ensure efficiency is $O(1)$, i.e., all compute units are fully utilized

- Another approach: increase Number of Devices
  - + Larger On-chip memory
  - + Higher bandwidth
  - - Communication/Synchronization overheads

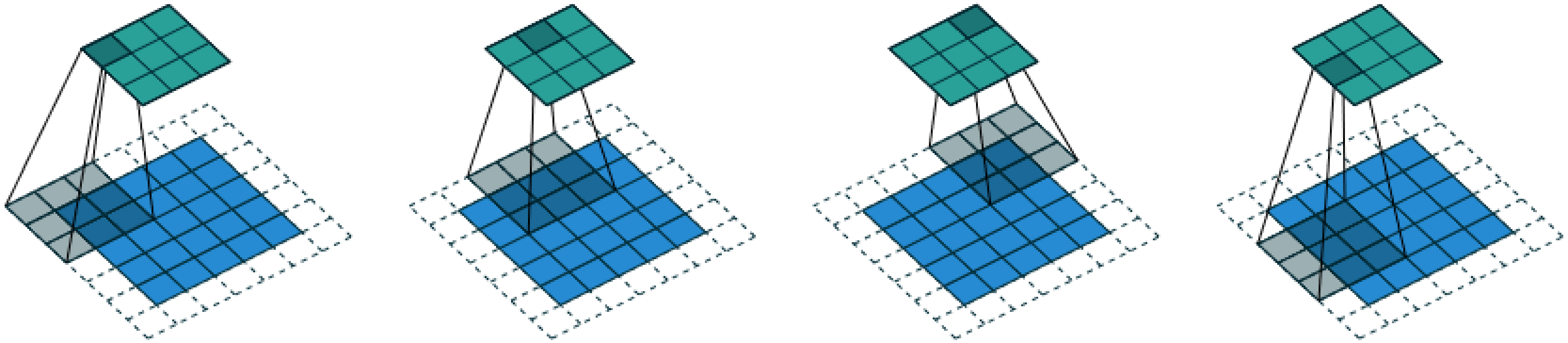
Variations

Reading Homework:

A guide to convolutional arithmetic for deep learning



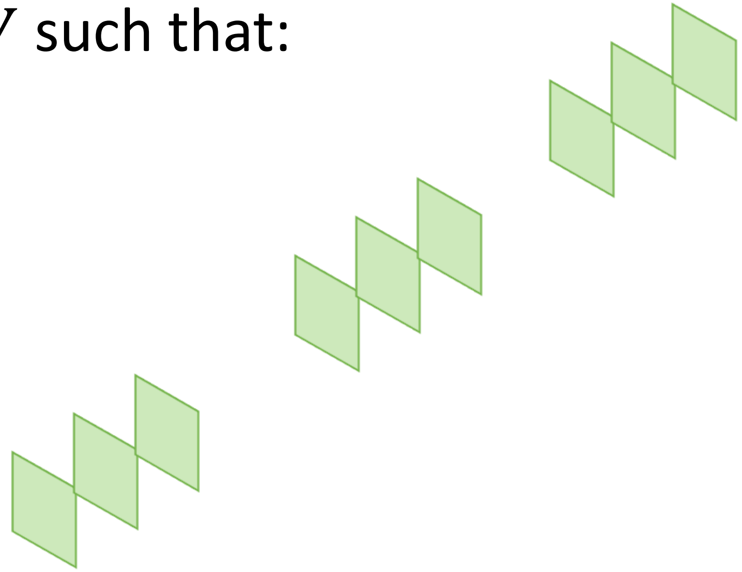
- Padding



- Strides

Convolutional Neural Network

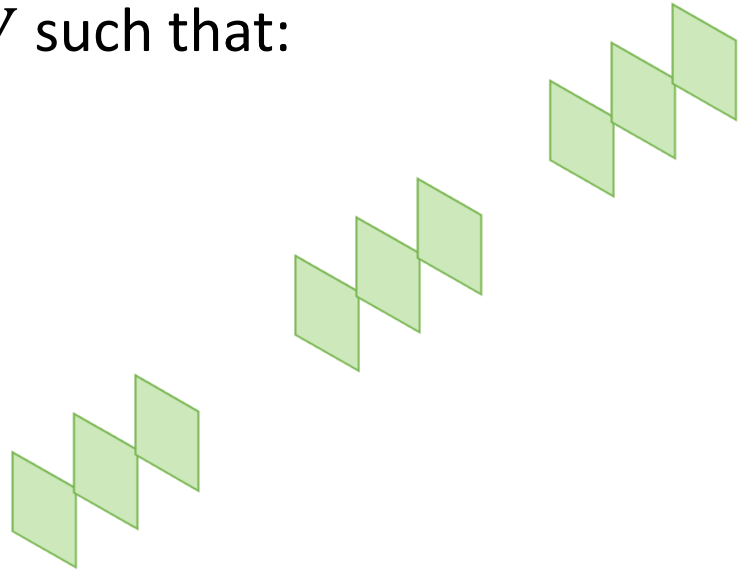
- Training
  - Given $\mathrm{x_i} \in X$ (image from a database), y_i: Label for the image
  - “Learn” a model F, defined by the filters W such that:
    - $\sum(y_i - F(x_i\colon W))$^2 is minimized



- Inference
  - Given a new image x'
  - Predict a label $y' = W(x')$
  - No change in the filters

Convolutional Neural Network

- **Training**
  - Given $\mathrm{x_i} \in X$ (image from a database), y_i: Label for the image
  - "Learn" a model F, defined by the filters W such that:
    - $\sum(y_i - F(x_i\!: W))^\wedge 2$ is minimized



- Inference
  - Given a new image x'
  - Predict a label $y' = W(x')$
  - No change in the filters

Training

$$E(W)\colon \min_{W} \sum (y_i - F(x_i : W))\hat{}2$$

$$W_{t+1} \leftarrow W_t - \alpha \frac{\delta E(W)}{\delta W}$$

Iteratively

$$\frac{\delta E(W)}{\delta W} = 2\sum_i \underbrace{\left(y_i - F(x_i : W)\right)}_{\text{Error}} \times \frac{\delta F(x_i : W)}{\delta W}$$

For all samples, in each iteration

Training

L layers in CNN

$$W_1, W_2, W_3, \dots, W_L$$
$$O_1, O_2, O_3, \dots, O_L$$

Weights (Filters/Kernels)

Output of layer

$$2 * Error \; * \; \frac{\delta F(x_i : W_{1:L})}{\delta W_1}, \frac{\delta F(x_i : W_{1:L})}{\delta W_2}, \dots, \frac{\delta F(x_i : W_{1:L})}{\delta W_L}$$

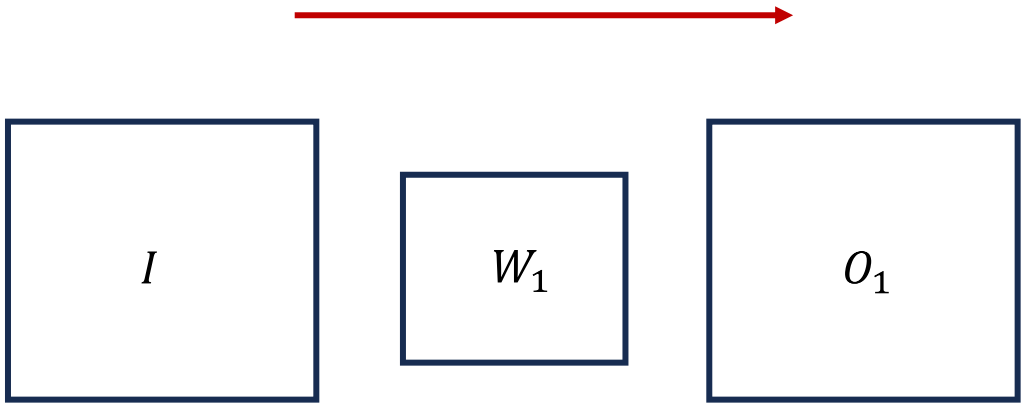
Compute these for all the samples

CNN Training: Forward Propagation

I

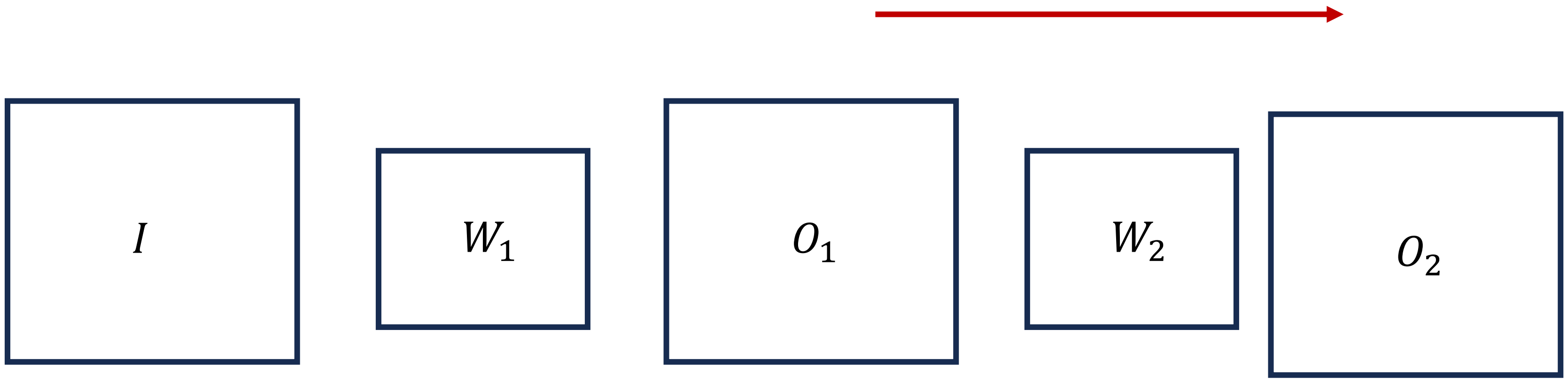
For a single image

CNN Training: Forward Propagation



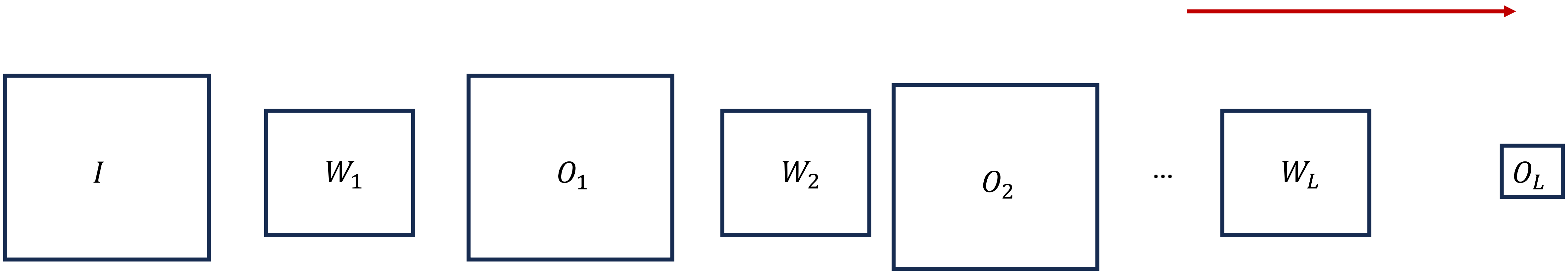
For a single image

CNN Training: Forward Propagation



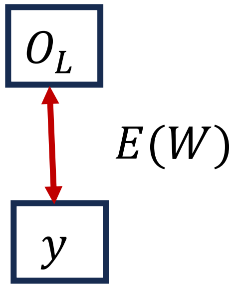
For a single image

CNN Training: Forward Propagation



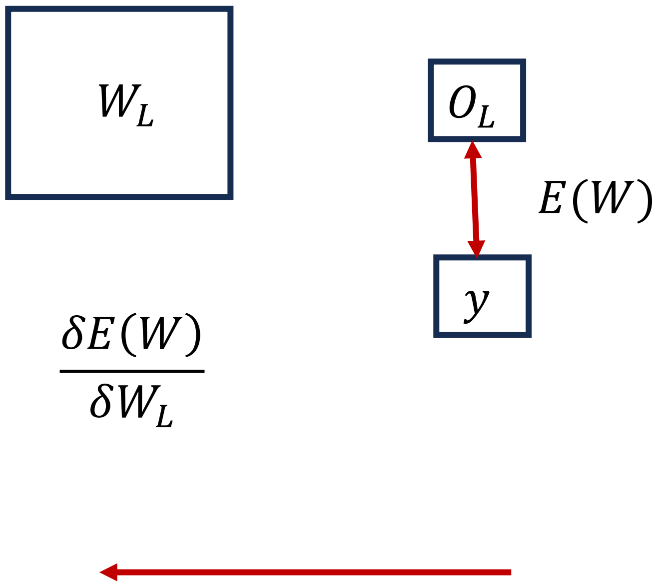
For a single image

CNN Training: Error Calculation



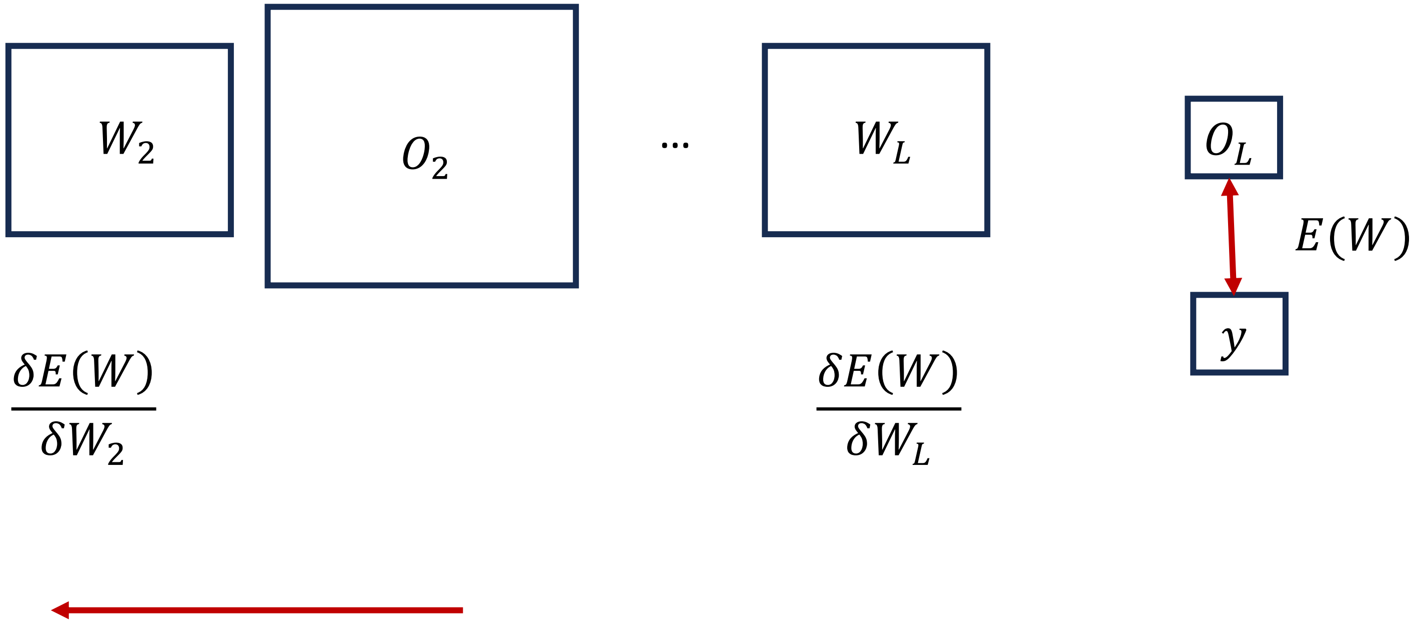
For a single image

CNN Training: Backward Propagation



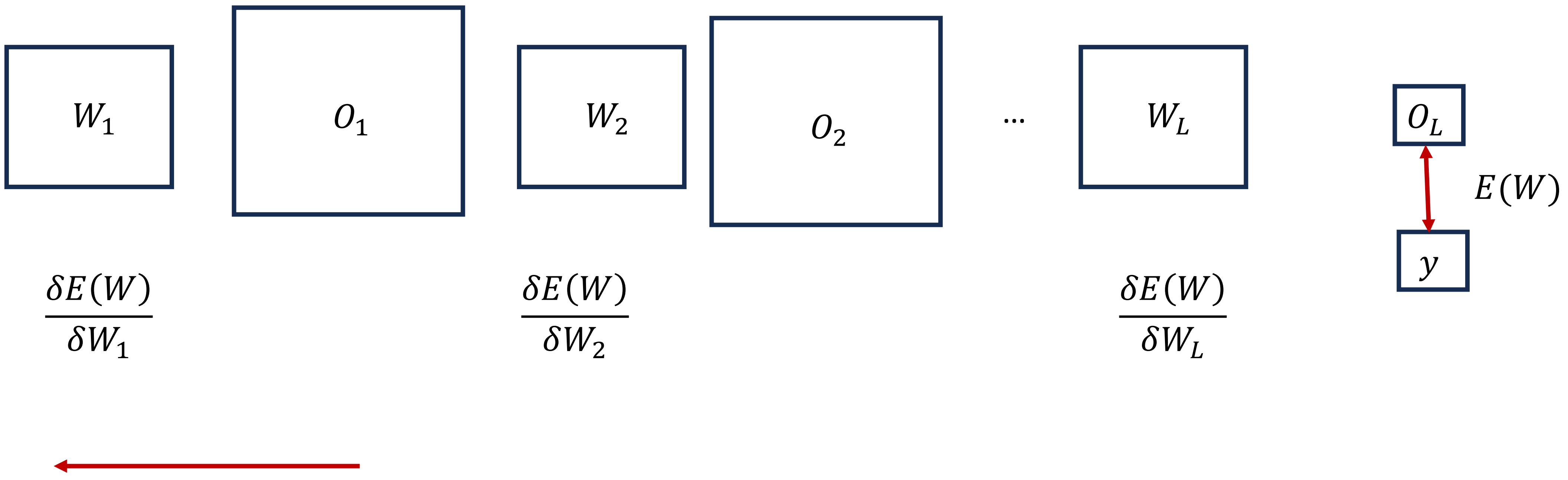
For a single image

CNN Training: Backward Propagation



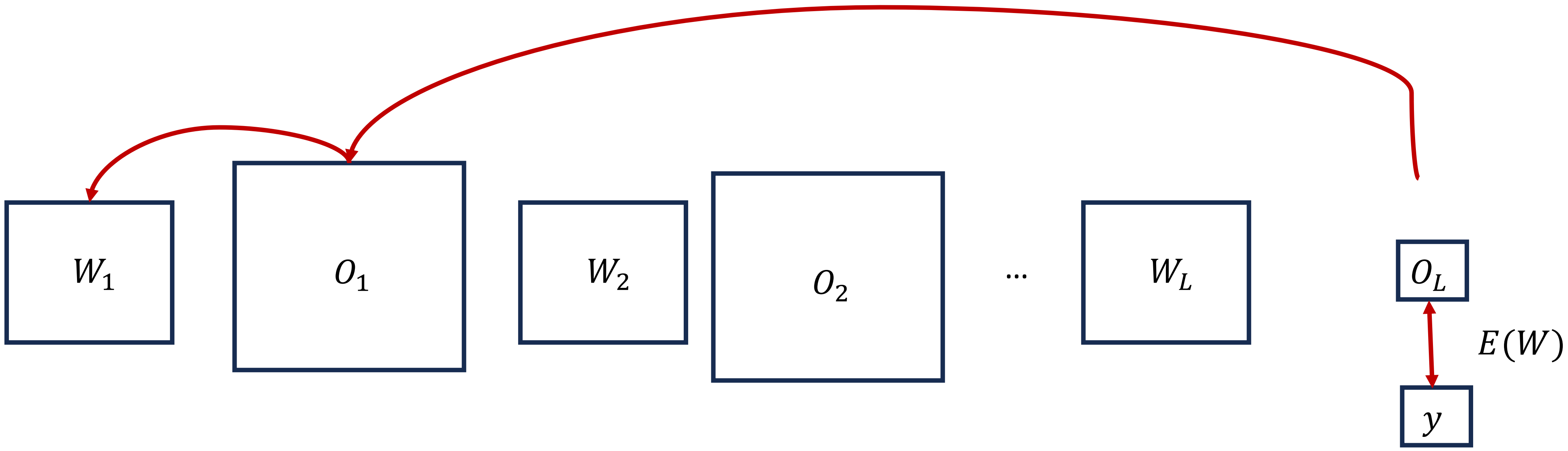
For a single image

CNN Training: Backward Propagation



For a single image

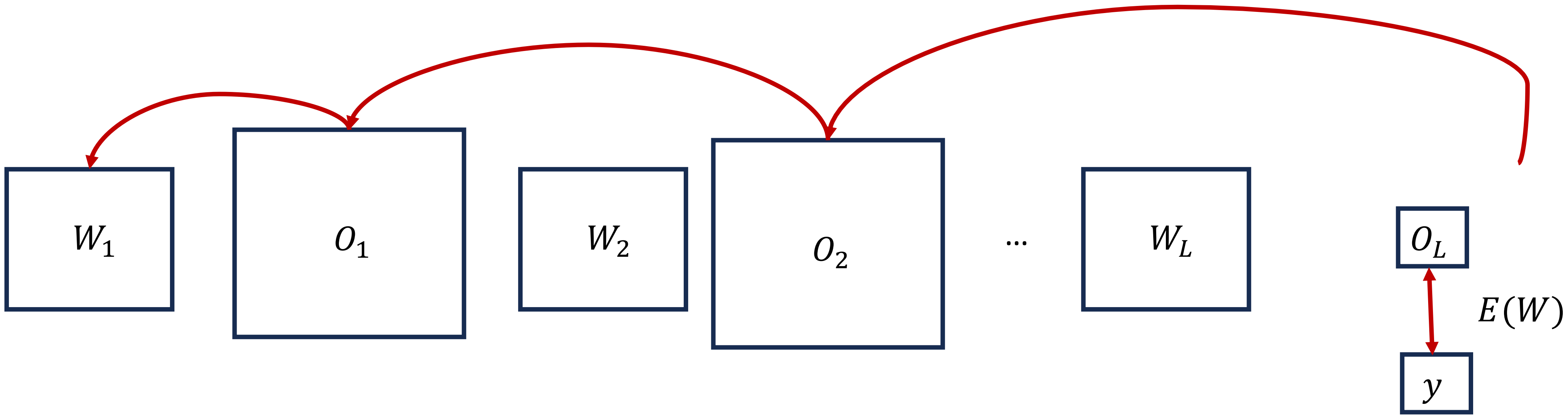
CNN Training: Backward Propagation



$$\frac{\delta E(W)}{\delta W_1} = \frac{\delta O_1}{\delta W_1} \times \frac{\delta E(W)}{\delta O_1}$$

For a single image

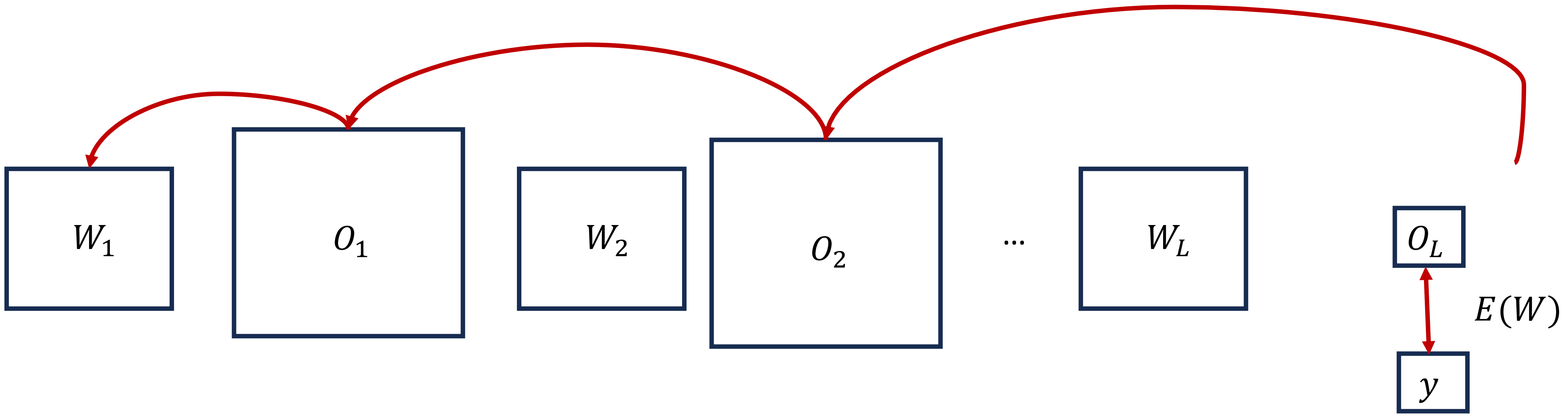
CNN Training: Backward Propagation



$$\frac{\delta E(W)}{\delta W_1} = \frac{\delta O_1}{\delta W_1} \times \frac{\delta O_2}{\delta O_1} \times \frac{\delta E(W)}{\delta O_2}$$

For a single image

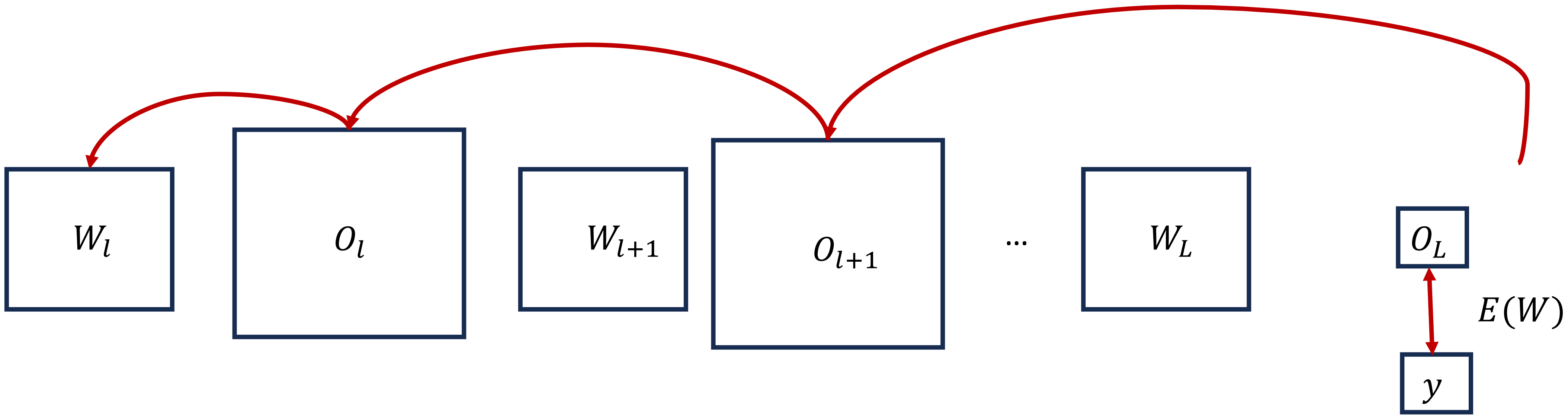
CNN Training: Backward Propagation



$$\frac{\delta E(W)}{\delta W_1} = \frac{\delta O_1}{\delta W_1} \times \frac{\delta O_2}{\delta O_1} \times \frac{\delta O_3}{\delta O_2} \times \cdots \times \frac{\delta O_L}{\delta O_{L-1}} \times \frac{\delta E(W)}{\delta O_L}$$

For a single image

CNN Training: Backward Propagation



$$\frac{\delta E(W)}{\delta W_l} = \frac{\delta O_l}{\delta W_l} \times \frac{\delta O_{l+1}}{\delta O_l} \times \frac{\delta O_{l+2}}{\delta O_{l+1}} \times \cdots \times \frac{\delta O_L}{\delta O_{L-1}} \times \frac{\delta E(W)}{\delta O_L}$$

For a single image

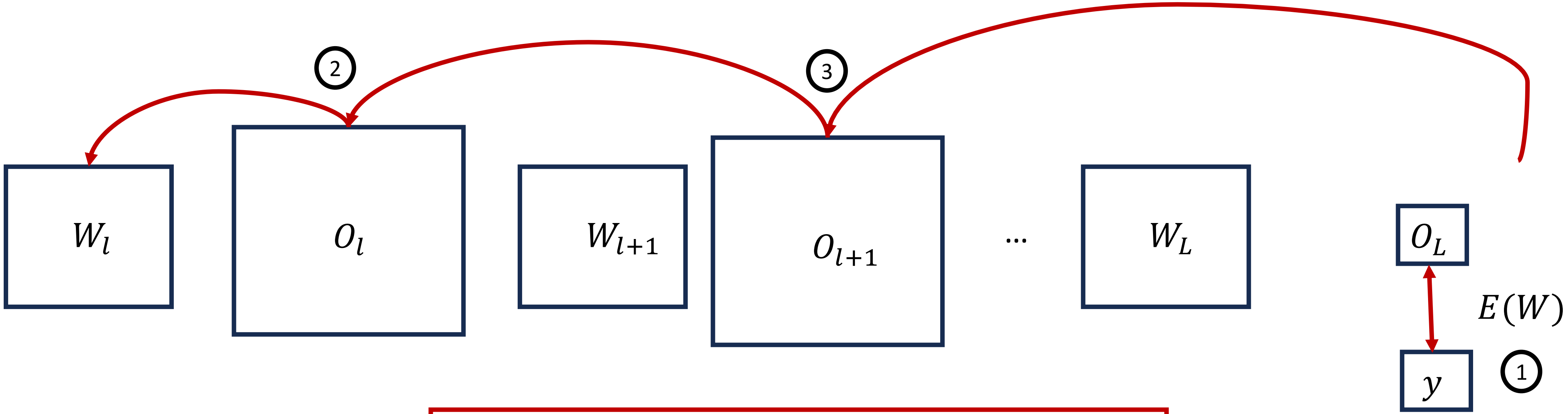
CNN Training: Backward Propagation

$$\frac{\delta E(W)}{\delta W_1} = \frac{\delta O_1}{\delta W_1} \times \frac{\delta O_2}{\delta O_1} \times \frac{\delta O_3}{\delta O_2} \times \cdots \times \frac{\delta O_L}{\delta O_{L-1}} \times \frac{\delta E(W)}{\delta O_L}$$

Reuse calculations performed for later layer
In earlier layer

$$\frac{\delta E(W)}{\delta W_l} = \frac{\delta O_l}{\delta W_l} \times \overbrace{\frac{\delta O_{l+1}}{\delta O_l} \times \frac{\delta O_{l+2}}{\delta O_{l+1}} \times \cdots \times \frac{\delta O_L}{\delta O_{L-1}} \times \frac{\delta E(W)}{\delta O_L}}$$

CNN Training: Backward Propagation



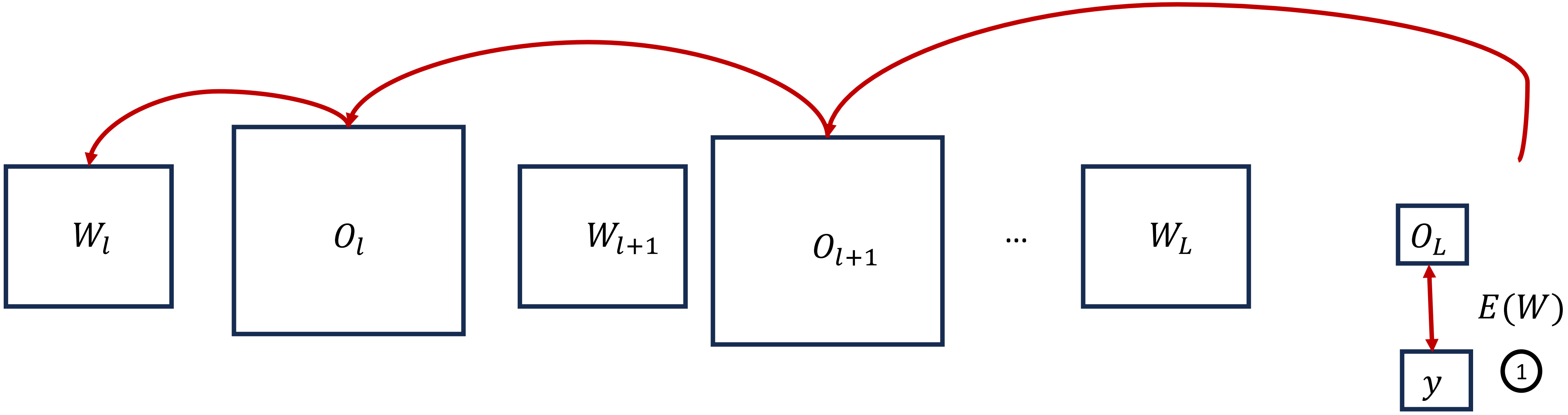
$$\frac{\delta E(W)}{\delta W_l} = \frac{\delta O_l}{\delta W_l} \times \frac{\delta O_{l+1}}{\delta O_l} \times \frac{\delta O_{l+2}}{\delta O_{l+1}} \times \cdots \times \frac{\delta O_L}{\delta O_{L-1}} \times \boxed{\frac{\delta E(W)}{\delta O_L}}$$

2 3 1

We need to compute 3 types of terms

For a single image

CNN Training: Backward Propagation



$$\frac{\delta E(W)}{\delta W_l} = \frac{\delta O_l}{\delta W_l} \times \frac{\delta O_{l+1}}{\delta O_l} \times \frac{\delta O_{l+2}}{\delta O_{l+1}} \times \cdots \times \frac{\delta O_L}{\delta O_{L-1}} \times \boxed{\frac{\delta E(W)}{\delta O_L}}$$

1

We need to compute 3 types of terms

For a single image

CNN Training: Backward Propagation

- $\frac{\delta E(W)}{\delta O_L}$???

CNN Training: Backward Propagation

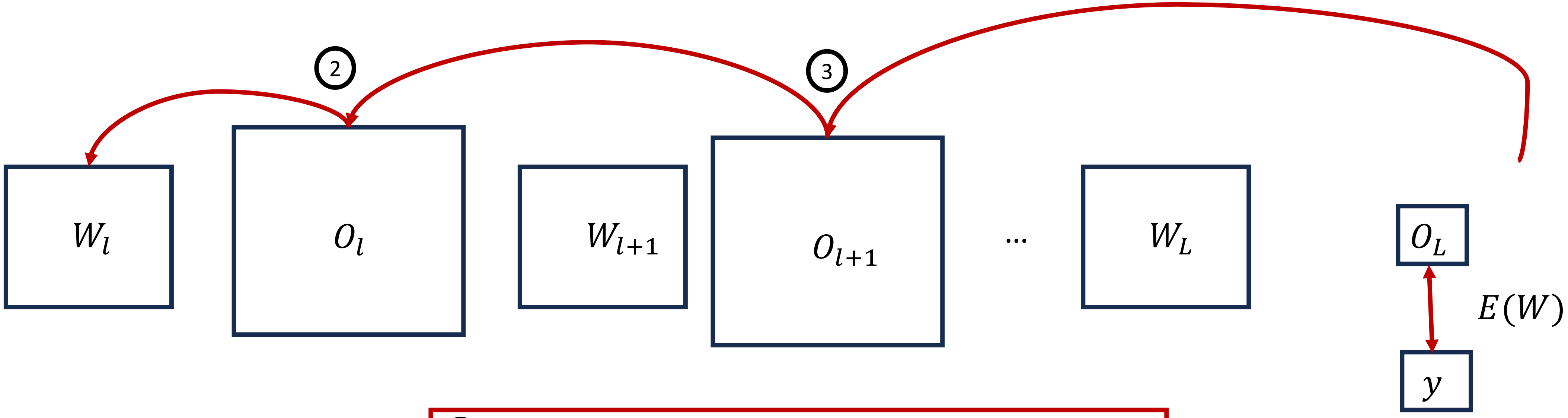
- $\frac{\delta E(W)}{\delta O_L} = \frac{\delta}{\delta O_L}(y_i \ - O_L)^2 = ??$

CNN Training: Backward Propagation

- $\frac{\delta E(W)}{\delta O_L} = \frac{\delta}{\delta O_L}(y_i \ - O_L)^2 = -2 \ * Error_i$

- For i image

CNN Training: Backward Propagation

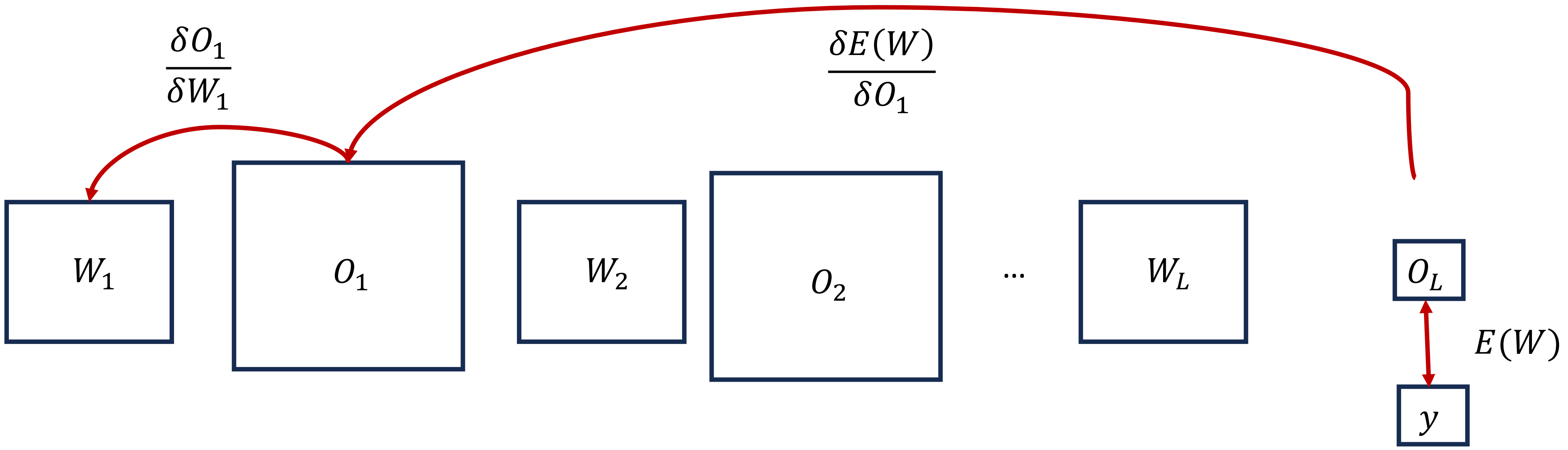


$$\frac{\delta E(W)}{\delta W_l} = \boxed{② \frac{\delta O_l}{\delta W_l} \times \boxed{③ \frac{\delta O_{l+1}}{\delta O_l} \times \frac{\delta O_{l+2}}{\delta O_{l+1}} \times \cdots \times \boxed{\frac{\delta O_L}{\delta O_{L-1}} \times \frac{\delta E(W)}{\delta O_L}}}}$$

We need to compute 3 types of terms

For a single image

CNN Training: Gradient Calculations



$$\frac{\delta E(W)}{\delta W_1} = \frac{\delta O_1}{\delta W_1} \times \frac{\delta E(W)}{\delta O_1}$$

CNN Training: Gradient Calculations

$$\begin{bmatrix} I_1 & I_2 & I_3 & I_4 \\ I_5 & I_6 & I_7 & I_8 \\ I_9 & I_{10} & I_{11} & I_{12} \\ I_{13} & I_{14} & I_{15} & I_{16} \end{bmatrix} \quad \begin{bmatrix} W_1 & W_2 \\ W_3 & W_4 \end{bmatrix} \xleftarrow{\frac{\delta O}{\delta W}} \begin{bmatrix} O_1 & O_2 & O_3 \\ O_4 & O_5 & O_6 \\ O_7 & O_8 & O_9 \end{bmatrix} \xleftarrow{\frac{\delta E(W)}{\delta O}}$$

Gradients produced till now in backward propagation

$$\frac{\delta E(W)}{\delta W} = \frac{\delta O}{\delta W} \times \frac{\delta E(W)}{\delta O}$$

For a layer l, removing the subscript l for clarity

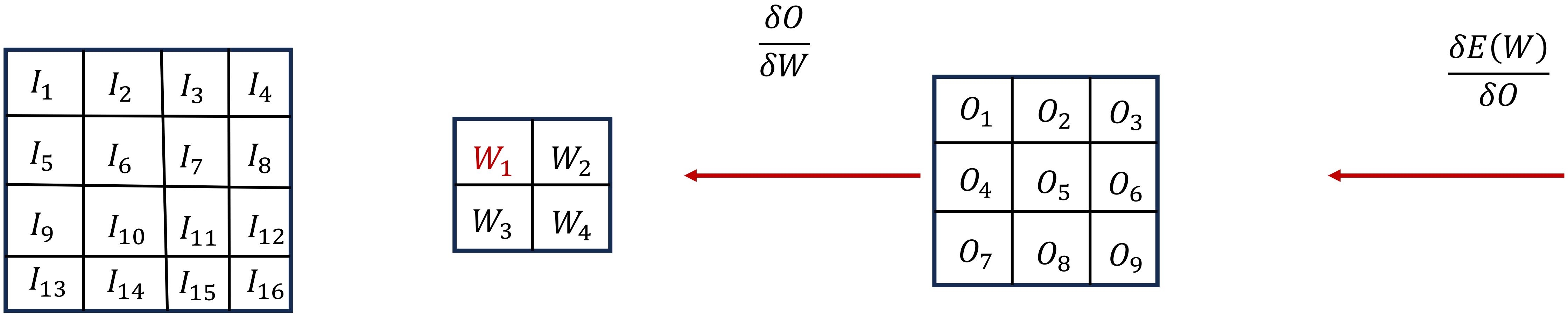
Outputs of the previous layer that we produced in Forward Propagation

CNN Training: Gradient Calculations

$$
\begin{bmatrix} I_1 & I_2 & I_3 & I_4 \\ I_5 & I_6 & I_7 & I_8 \\ I_9 & I_{10} & I_{11} & I_{12} \\ I_{13} & I_{14} & I_{15} & I_{16} \end{bmatrix}
\quad
\begin{bmatrix} \color{red}{W_1} & W_2 \\ W_3 & W_4 \end{bmatrix}
\xleftarrow{\frac{\delta O}{\delta W}}
\begin{bmatrix} O_1 & O_2 & O_3 \\ O_4 & O_5 & O_6 \\ O_7 & O_8 & O_9 \end{bmatrix}
\xleftarrow{\frac{\delta E(W)}{\delta O}}
$$

$$\frac{\delta E(W)}{\delta \color{red}{W_1}} = \frac{\delta O}{\delta \color{red}{W_1}} \times \frac{\delta E(W)}{\delta O}$$

CNN Training: Gradient Calculations



$$\frac{\delta E(W)}{\delta W_1} = \frac{\delta O_1}{\delta W_1} \times \frac{\delta E(W)}{\delta O_1} + \frac{\delta O_2}{\delta W_1} \times \frac{\delta E(W)}{\delta O_2} + \frac{\delta O_3}{\delta W_1} \times \frac{\delta E(W)}{\delta O_3} \ldots + \frac{\delta O_9}{\delta W_1} \times \frac{\delta E(W)}{\delta O_9}$$

CNN Training: Gradient Calculations

I_1	I_2	I_3	I_4
I_5	I_6	I_7	I_8
I_9	I_{10}	I_{11}	I_{12}
I_{13}	I_{14}	I_{15}	I_{16}

W_1	W_2
W_3	W_4

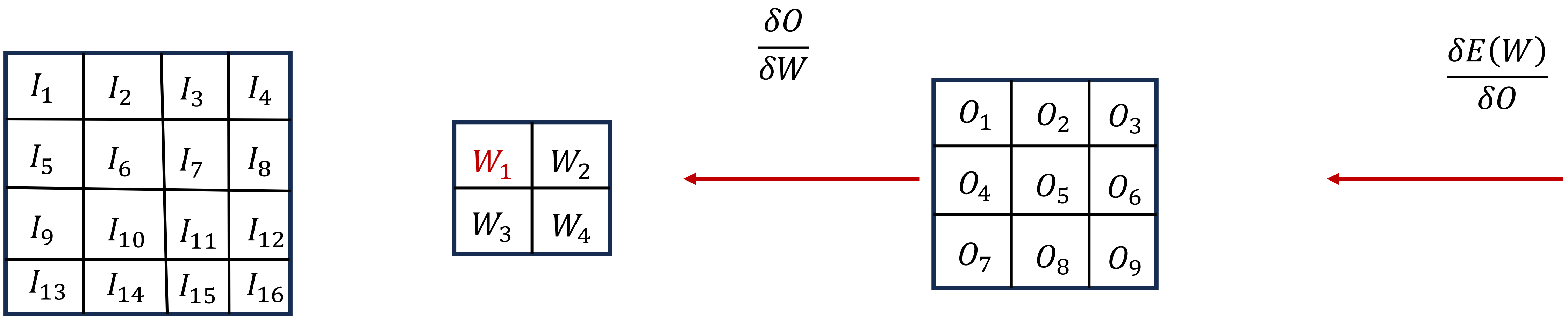
$$\frac{\delta O}{\delta W}$$

O_1	O_2	O_3
O_4	O_5	O_6
O_7	O_8	O_9

$$\frac{\delta E(W)}{\delta O}$$

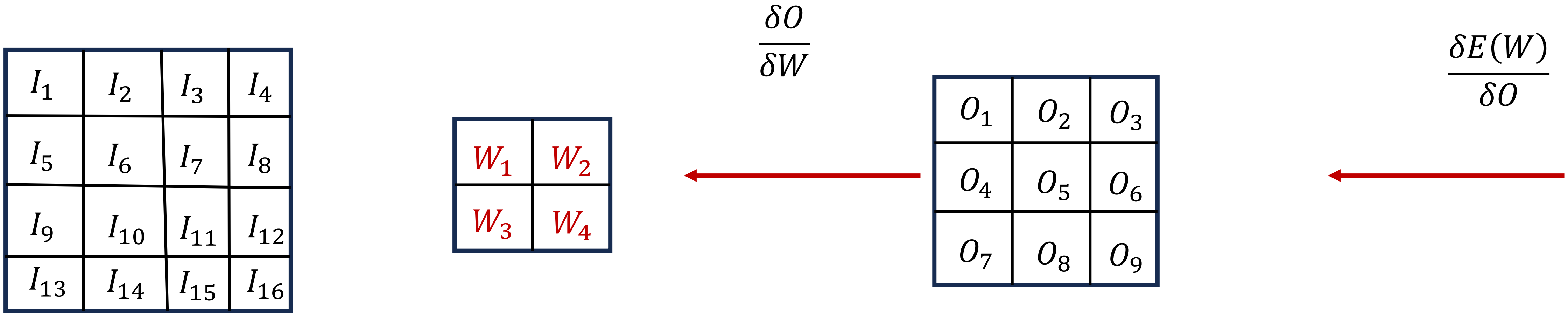
$$\frac{\delta E(W)}{\delta W_1} = \frac{\delta O_1}{\delta W_1} \times \frac{\delta E(W)}{\delta O_1} + \frac{\delta O_2}{\delta W_1} \times \frac{\delta E(W)}{\delta O_2} + \frac{\delta O_3}{\delta W_1} \times \frac{\delta E(W)}{\delta O_3} \ldots + \frac{\delta O_9}{\delta W_1} \times \frac{\delta E(W)}{\delta O_9}$$

CNN Training: Gradient Calculations



$$\frac{\delta E(W)}{\delta W_1} = I_1 \times \frac{\delta E(W)}{\delta O_1} + I_2 \times \frac{\delta E(W)}{\delta O_2} + I_3 \times \frac{\delta E(W)}{\delta O_3} \ldots + I_{11} \times \frac{\delta E(W)}{\delta O_9}$$

CNN Training: Gradient Calculations



$$\frac{\delta E(W)}{\delta W_1} = I_1 \times \frac{\delta E(W)}{\delta O_1} + I_2 \times \frac{\delta E(W)}{\delta O_2} + I_3 \times \frac{\delta E(W)}{\delta O_3} \ldots + I_{11} \times \frac{\delta E(W)}{\delta O_9}$$

$$\frac{\delta E(W)}{\delta W_2} = I_2 \times \frac{\delta E(W)}{\delta O_1} + I_3 \times \frac{\delta E(W)}{\delta O_2} + I_6 \times \frac{\delta E(W)}{\delta O_3} \ldots + I_{12} \times \frac{\delta E(W)}{\delta O_9}$$

$$\frac{\delta E(W)}{\delta W_3} = I_5 \times \frac{\delta E(W)}{\delta O_1} + I_6 \times \frac{\delta E(W)}{\delta O_2} + I_7 \times \frac{\delta E(W)}{\delta O_3} \ldots + I_{13} \times \frac{\delta E(W)}{\delta O_9}$$

...

CNN Training: Gradient Calculations

I_1	I_2	I_3	I_4
I_5	I_6	I_7	I_8
I_9	I_{10}	I_{11}	I_{12}
I_{13}	I_{14}	I_{15}	I_{16}

W_1	W_2
W_3	W_4

$$\frac{\delta O}{\delta W} \longleftarrow \frac{\delta E(W)}{\delta O}$$

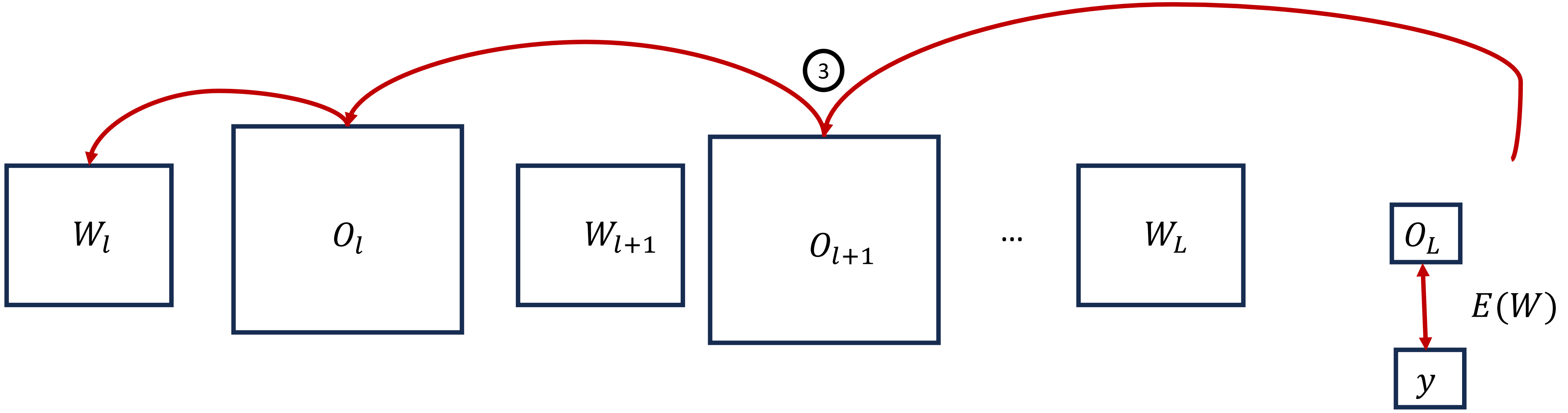
$\frac{\delta E}{\delta O_1}$	$\frac{\delta E}{\delta O_2}$	$\frac{\delta E}{\delta O_3}$
$\frac{\delta E}{\delta O_4}$	$\frac{\delta E}{\delta O_5}$	$\frac{\delta E}{\delta O_6}$
$\frac{\delta E}{\delta O_7}$	$\frac{\delta E}{\delta O_8}$	$\frac{\delta E}{\delta O_9}$

$$\frac{\delta E(W)}{\delta W} = Conv(I, \frac{\delta E(W)}{\delta O})$$

Note: Only inputs I and gradients $\frac{\delta E(W)}{\delta O}$ are needed. We don't need the output O for this layer. Why?

Note: This operation where the output dimension is greater than input dimension is call deconvolution or transposed convolution

CNN Training: Backward Propagation



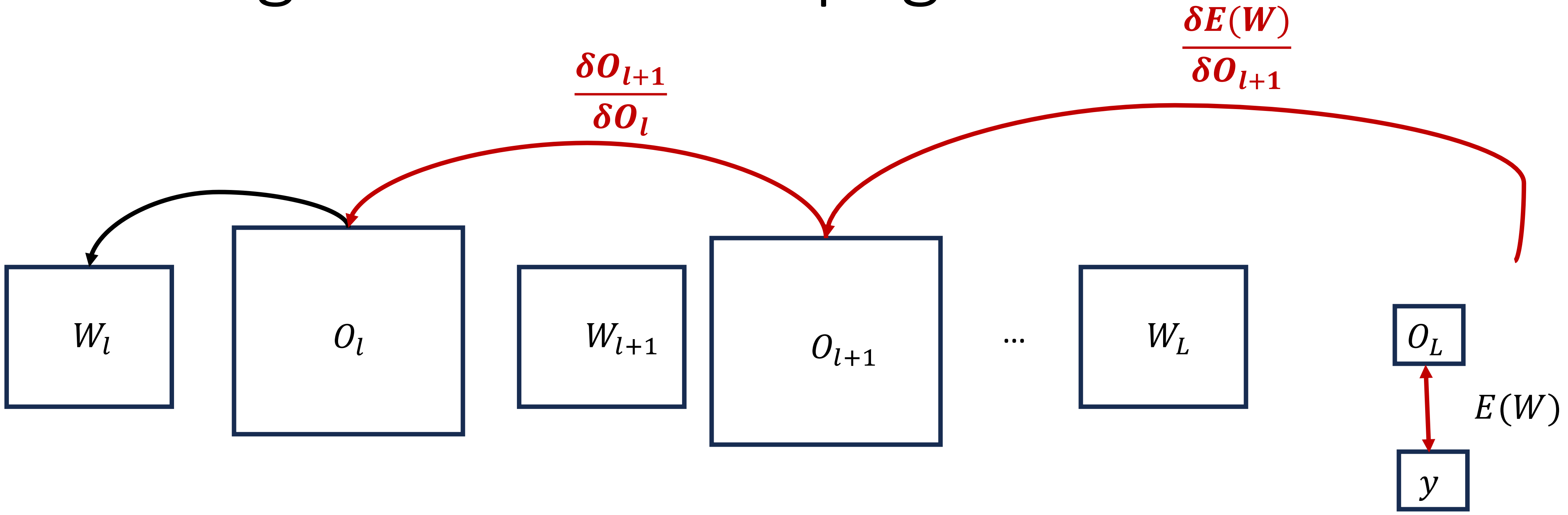
3

$$\frac{\delta E(W)}{\delta W_l} = \frac{\delta O_l}{\delta W_l} \times \frac{\delta O_{l+1}}{\delta O_l} \times \frac{\delta O_{l+2}}{\delta O_{l+1}} \times \cdots \times \frac{\delta O_L}{\delta O_{L-1}} \times \frac{\delta E(W)}{\delta O_L}$$

We need to compute 3 types of terms

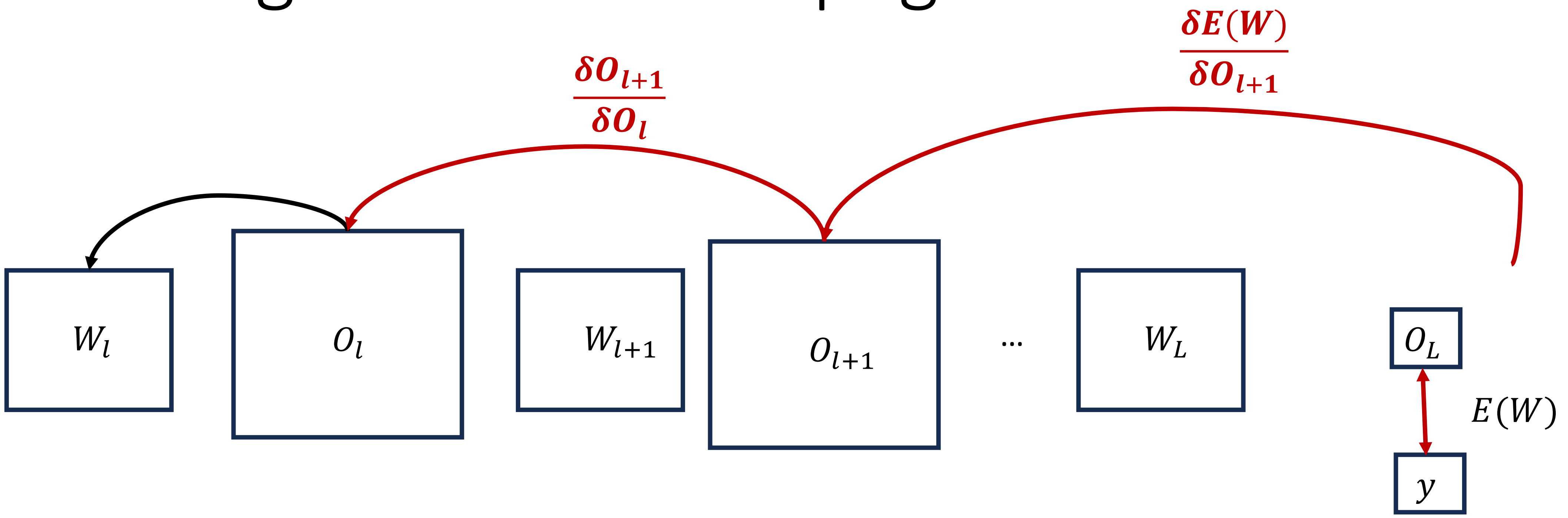
For a single image

CNN Training: Backward Propagation



$$\frac{\delta E(W)}{\delta O_l} = \frac{\delta O_{l+1}}{\delta O_l} \times \frac{\delta O_{l+2}}{\delta O_{l+1}} \times \cdots \times \frac{\delta O_L}{\delta O_{L-1}} \times \frac{\delta E(W)}{\delta O_L}$$

CNN Training: Backward Propagation



$$\frac{\delta E(W)}{\delta O_l} = \frac{\delta O_{l+1}}{\delta O_l} \times \frac{\delta E(W)}{\delta O_{l+1}}$$

CNN Training: Backward Propagation

O_l

O_1	O_2	O_3	O_4
O_5	O_6	O_7	O_8
O_9	O_{10}	O_{11}	O_{12}
O_{13}	O_{14}	O_{15}	O_{16}

W_1	W_2
W_3	W_4

O_{l+1}

$\frac{\delta E}{\delta O_1}$	$\frac{\delta E}{\delta O_2}$	$\frac{\delta E}{\delta O_3}$
$\frac{\delta E}{\delta O_4}$	$\frac{\delta E}{\delta O_5}$	$\frac{\delta E}{\delta O_6}$
$\frac{\delta E}{\delta O_7}$	$\frac{\delta E}{\delta O_8}$	$\frac{\delta E}{\delta O_9}$

What all do we need for the computation? Do we need inputs or outputs to the layer?

$$\frac{\delta E(W)}{\delta O_l} = Conv(W_{180}, \frac{\delta E(W)}{\delta O_{l+1}})$$

HW: W_{180} $- W$ rotated by 180 degrees

Key Operation in CNN

- Forward Propagation:

$$O_l = Conv(W_l, I_l)$$

- Backward Propagation

$$\frac{\delta E(W)}{\delta W_l} = Conv(I_l, \frac{\delta E(W)}{\delta O_l})$$

$$\frac{\delta E(W)}{\delta O_l} = Conv(W_{180}, \frac{\delta E(W)}{\delta O_{l+1}})$$

Training

$$E(W):\ \min_{W} \sum (y_i - F(x_i : W))^{\wedge}2$$

$$W_{t+1} \leftarrow W_t - \alpha \frac{\delta E(W)}{\delta W}$$

Iteratively

$$\frac{\delta E(W)}{\delta W} = 2\sum \underbrace{\left(y_i - F(x_i : W)\right)} \times \frac{\delta F(x_i : W)}{\delta W}$$

For all samples, in each iteration

Error

For multiple images in a batch: Simply sum up the gradients

$$\frac{\delta E(W)}{\delta W_l} = \boxed{\frac{\delta O_l}{\delta W_l} \times \frac{\delta O_{l+1}}{\delta O_l} \times \frac{\delta O_{l+2}}{\delta O_{l+1}} \times \cdots \times \frac{\delta O_L}{\delta O_{L-1}}} \times \boxed{\frac{\delta E(W)}{\delta O_L}}$$

Training

L layers in CNN

$$W_1, W_2, W_3, \ldots, W_L$$
$$O_1, O_2, O_3, \ldots, O_L$$

Weights (Filters/Kernels)

Output of layer

$$2 * Error \; * \; \frac{\delta F(x_i : W_{1:L})}{\delta W_1}, \frac{\delta F(x_i : W_{1:L})}{\delta W_2}, \ldots, \frac{\delta F(x_i : W_{1:L})}{\delta W_L}$$

Training – Key Performance Objective

- Accuracy – We want to train a model that can predict with low error on unseen data

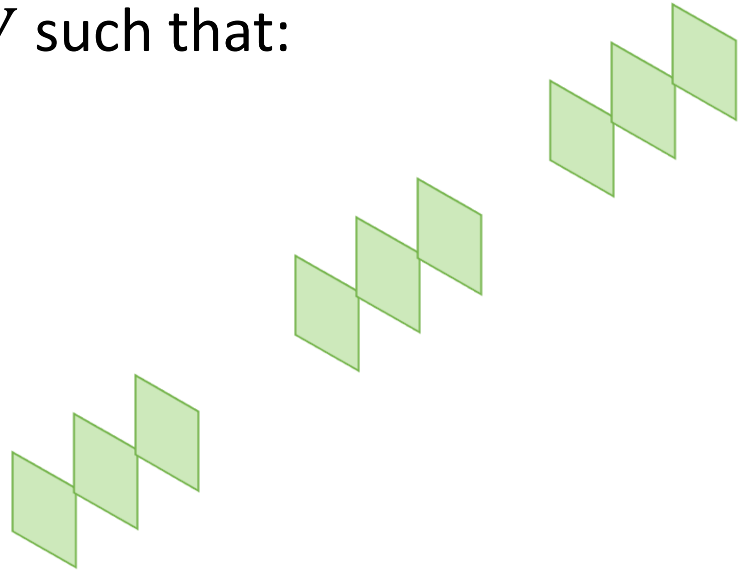
- Throughput – Given a large dataset, we want to train as quickly as possible

Training – Key Requirement

- For each image,
- Need to store the inputs of each layer
  - Note: Input I_l of layer l is output (activation) O_{l-1} of layer $l-1$
- Need to have gradients $\frac{\delta E(W)}{\delta W_l}, \frac{\delta E(W)}{\delta O_l}, \frac{\delta E(W)}{\delta O_{l+1}}$ for each layer l
- Ungraded HW assignment: Can you calculate the difference in storage requirements between computing gradients from left to right vs right to left?

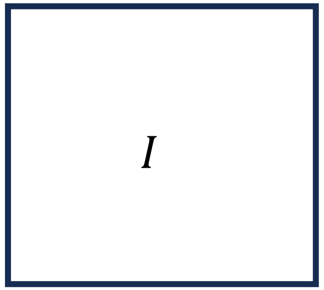
Convolutional Neural Network

- Training
  - Given $\mathrm{x_i} \in X$ (image from a database), y_i: Label for the image
  - "Learn" a model F, defined by the filters W such that:
    - $\sum(y_i - F(x_i : W))$^2 is minimized



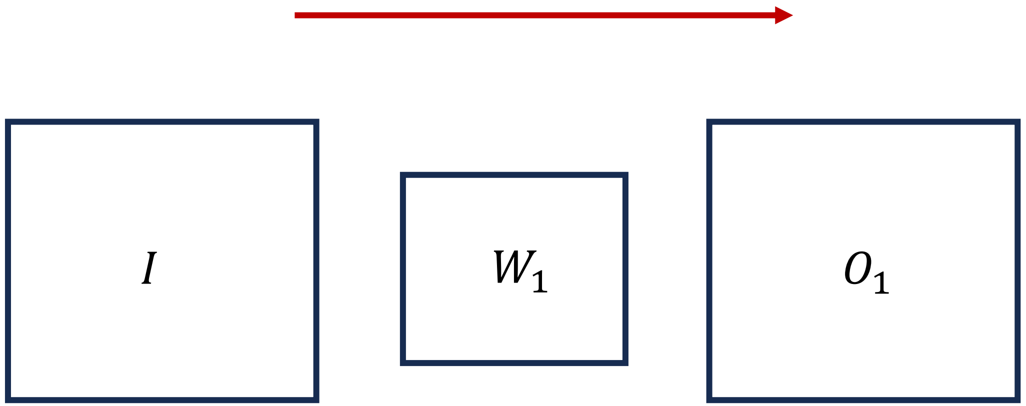
- **Inference**
  - Given a new image x'
  - Predict a label $y' = W(x')$
  - No change in the filters

CNN Training: Forward Propagation



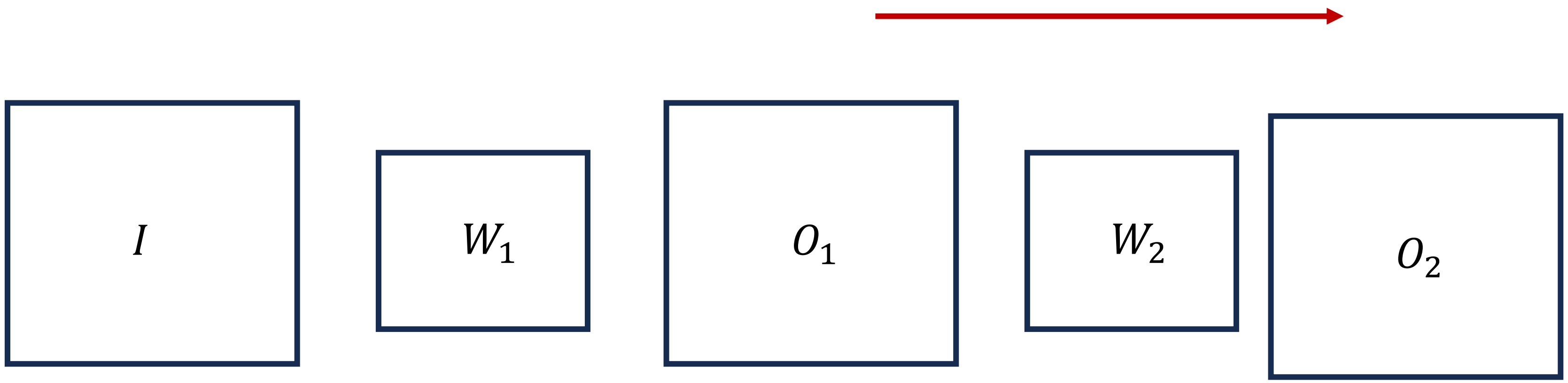
For a single image

CNN Training: Forward Propagation



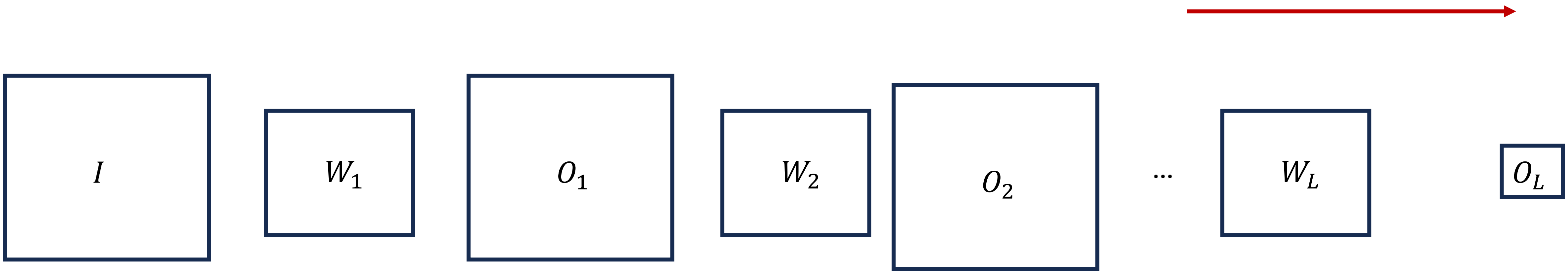
For a single image

CNN Training: Forward Propagation



For a single image

CNN Training: Forward Propagation



For a single image

Inference

- Simply the forward propagation portion
- No error calculation or backpropagation as ground truth doesn’t exist
- Performance Objectives:
  - Accuracy – Although ground truth is not known
  - Latency – Perform inference as fast as possible on single or a batch of images

Next Class

- 9/25 Lecture 10
  - Accelerating Convolutional Neural Networks: Convolution as Matrix Multiplication

Thank You

- Questions?

- Email: sanmukh.kuppannagari@case.edu