Tutorial: Convolutional Neural Networks

Karan Taneja

CS419 (M) TA

Autumn 2018

Quiz at 10:20 AM

YOLOv3 (You Only Look Once)

Initial Notes

- Convolutional Neural Network is abbreviated as **CNN**.
- Presentation shall focus **more on intuition and high-level understanding** rather than mathematical details.
- We mainly discuss CNNs for **image classification problem using supervised methods**.

Journey

- Convolution and Max-Pooling Layers
- Fully Connected Layers
- Architecture of CNN for Classification Problem
- Loss, Inference and Training

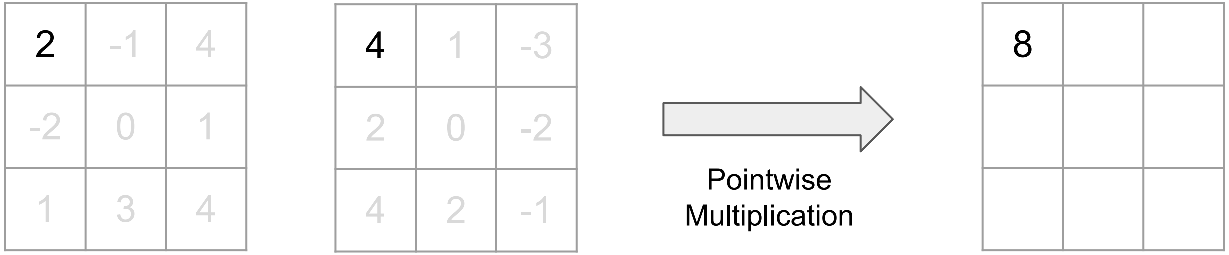
Convolution

Giving the **integral** of the **pointwise multiplication** of the two functions as a function of the amount that one of the original functions is **translated**.

- Wikipedia

Convolution

Giving the **integral** of the **pointwise multiplication** of the two functions as a function of the amount that one of the original functions is **translated**.



2	-1	4
-2	0	1
1	3	4

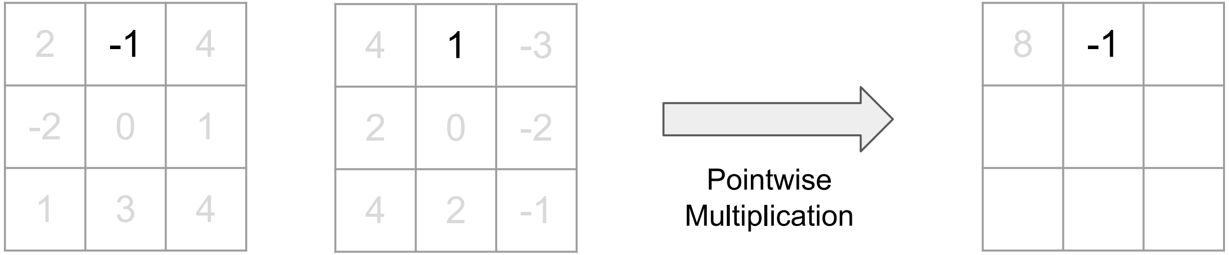
4	1	-3
2	0	-2
4	2	-1

Pointwise Multiplication

8		

Convolution

Giving the **integral** of the **pointwise multiplication** of the two functions as a function of the amount that one of the original functions is **translated**.



2	-1	4
-2	0	1
1	3	4

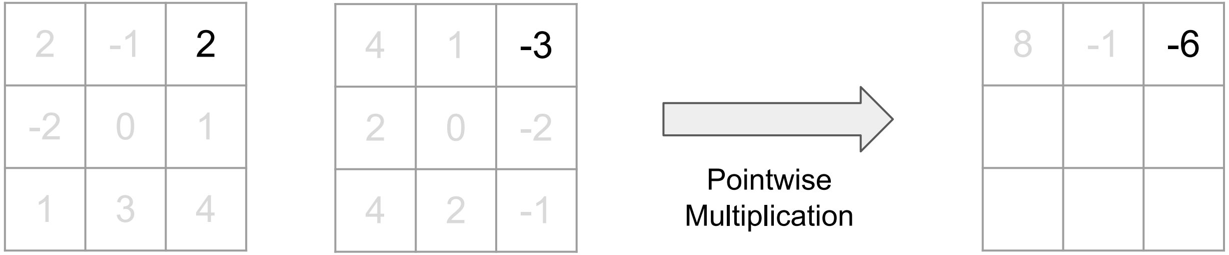
4	1	-3
2	0	-2
4	2	-1

Pointwise Multiplication

8	-1	

Convolution

Giving the **integral** of the **pointwise multiplication** of the two functions as a function of the amount that one of the original functions is **translated**.



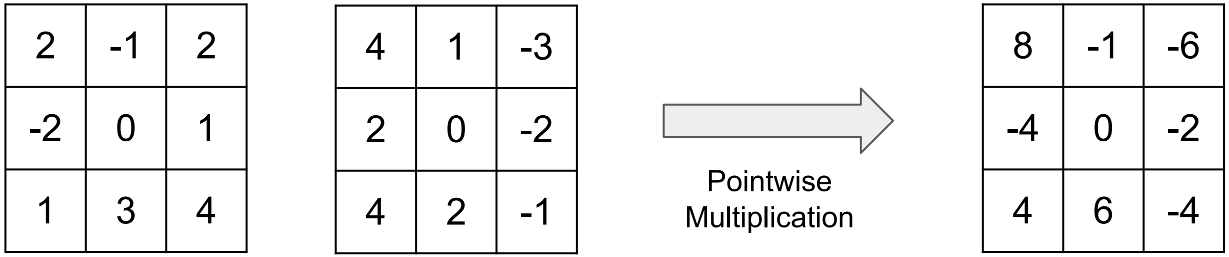
2	-1	2
-2	0	1
1	3	4

4	1	-3
2	0	-2
4	2	-1

8	-1	-6

Convolution

Giving the **integral** of the **pointwise multiplication** of the two functions as a function of the amount that one of the original functions is **translated**.



2	-1	2
-2	0	1
1	3	4

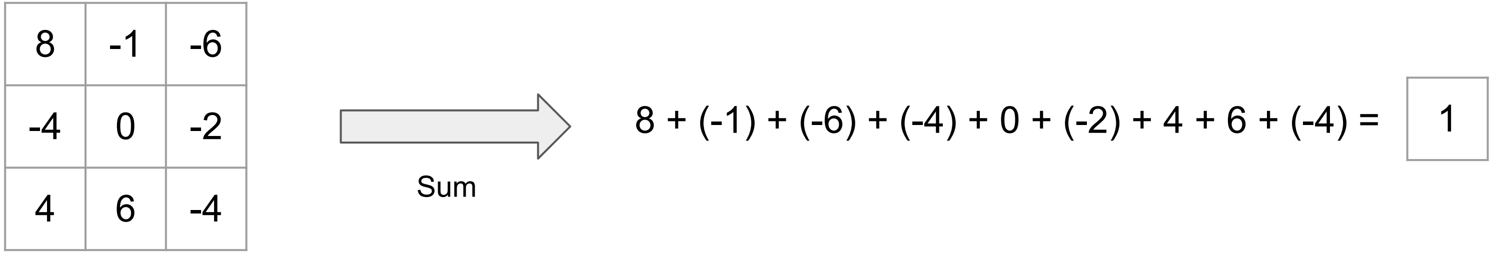
4	1	-3
2	0	-2
4	2	-1

Pointwise Multiplication

8	-1	-6
-4	0	-2
4	6	-4

Convolution

Giving the **integral** of the **pointwise multiplication** of the two functions as a function of the amount that one of the original functions is **translated**.



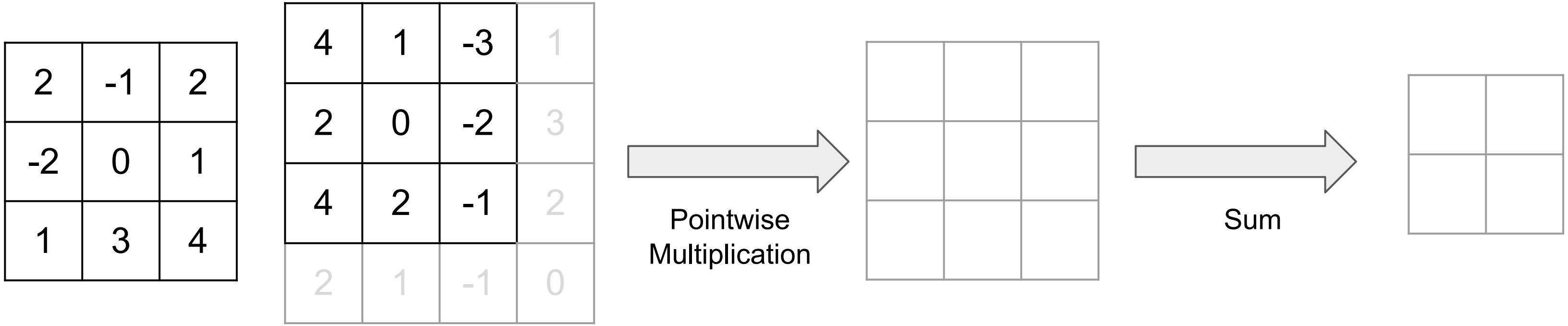
8	-1	-6
-4	0	-2
4	6	-4

Sum

$$8 + (-1) + (-6) + (-4) + 0 + (-2) + 4 + 6 + (-4) = \boxed{1}$$

Convolution

Giving the **integral** of the **pointwise multiplication** of the two functions as a function of the amount that one of the original functions is **translated**.

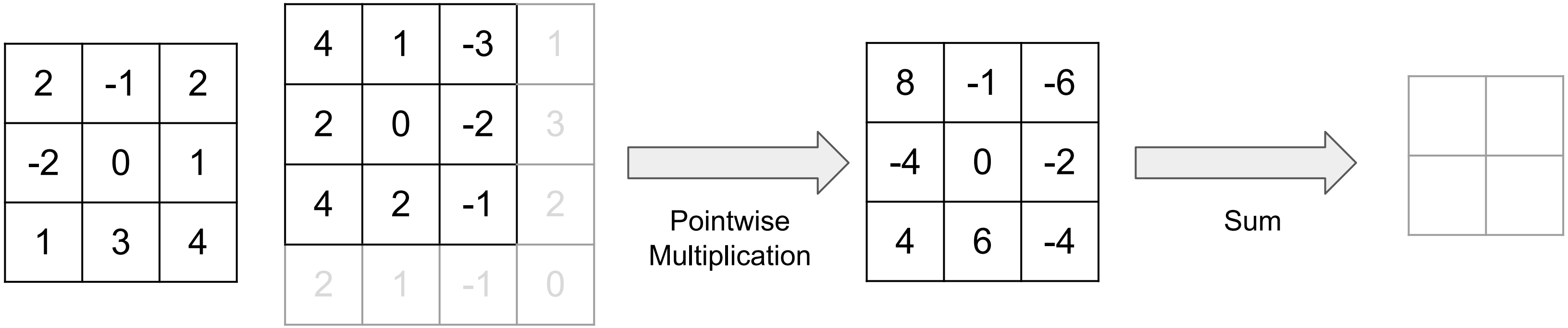


2	-1	2
-2	0	1
1	3	4

4	1	-3	1
2	0	-2	3
4	2	-1	2
2	1	-1	0

Convolution

Giving the **integral** of the **pointwise multiplication** of the two functions as a function of the amount that one of the original functions is **translated**.



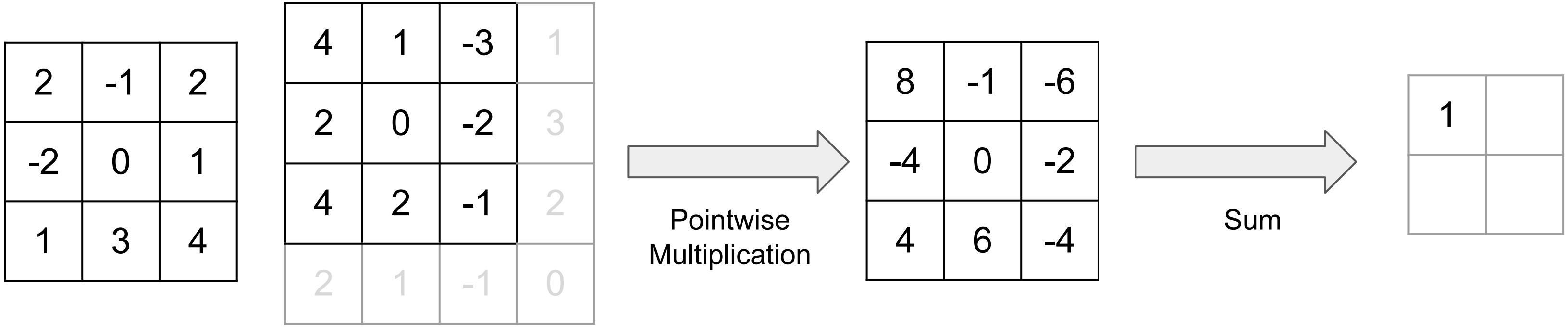
2	-1	2
-2	0	1
1	3	4

4	1	-3	1
2	0	-2	3
4	2	-1	2
2	1	-1	0

8	-1	-6
-4	0	-2
4	6	-4

Convolution

Giving the **integral** of the **pointwise multiplication** of the two functions as a function of the amount that one of the original functions is **translated**.



2	-1	2
-2	0	1
1	3	4

4	1	-3	1
2	0	-2	3
4	2	-1	2
2	1	-1	0

Pointwise Multiplication

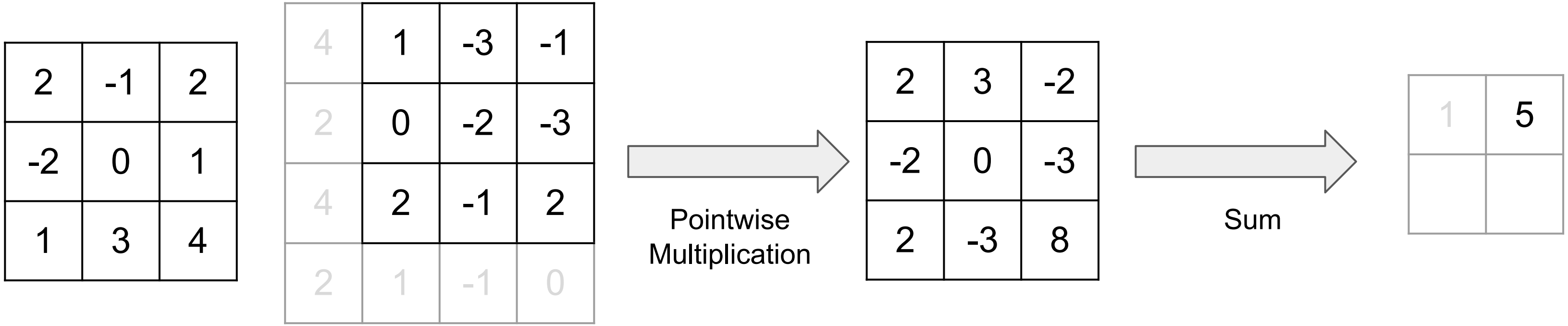
8	-1	-6
-4	0	-2
4	6	-4

Sum

1	

Convolution

Giving the **integral** of the **pointwise multiplication** of the two functions as a function of the amount that one of the original functions is **translated**.



2	-1	2
-2	0	1
1	3	4

4	1	-3	-1
2	0	-2	-3
4	2	-1	2
2	1	-1	0

Pointwise Multiplication

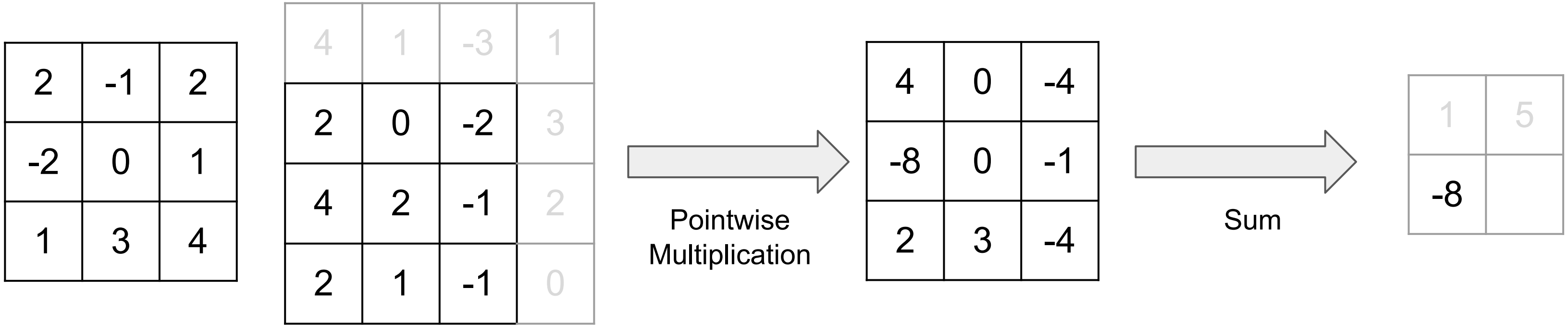
2	3	-2
-2	0	-3
2	-3	8

Sum

1	5

Convolution

Giving the **integral** of the **pointwise multiplication** of the two functions as a function of the amount that one of the original functions is **translated**.



2	-1	2
-2	0	1
1	3	4

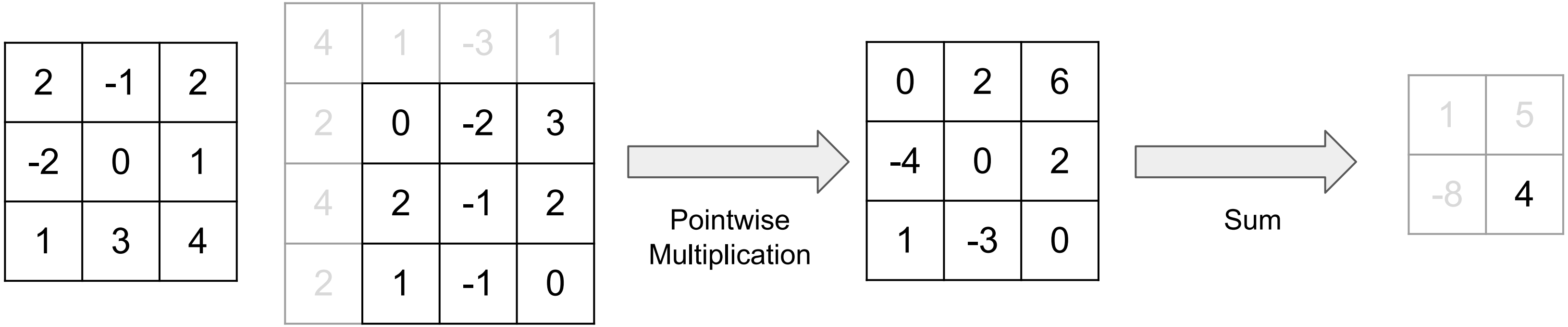
4	1	-3	1
2	0	-2	3
4	2	-1	2
2	1	-1	0

4	0	-4
-8	0	-1
2	3	-4

1	5
-8	

Convolution

Giving the **integral** of the **pointwise multiplication** of the two functions as a function of the amount that one of the original functions is **translated**.



2	-1	2
-2	0	1
1	3	4

4	1	-3	1
2	0	-2	3
4	2	-1	2
2	1	-1	0

Pointwise Multiplication

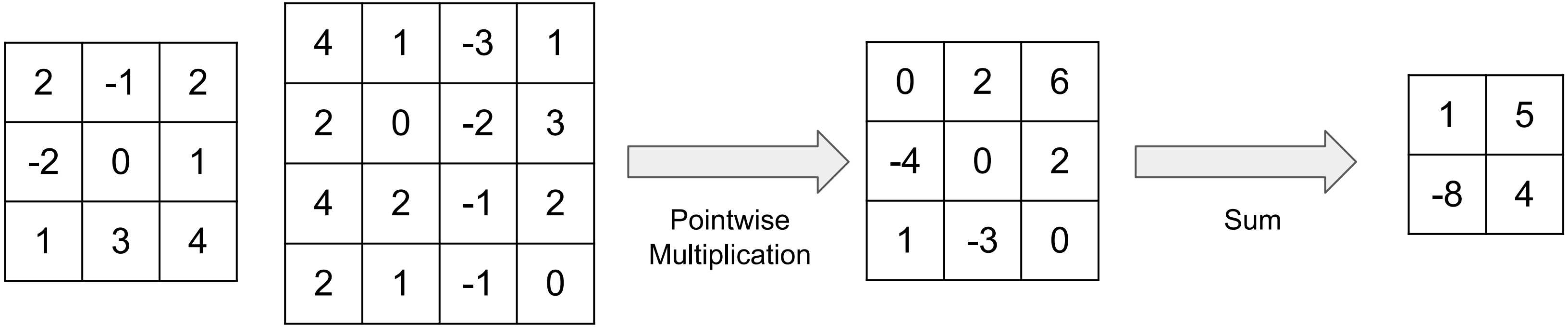
0	2	6
-4	0	2
1	-3	0

Sum

1	5
-8	4

Convolution

Stride 1 x 1   Padding “Valid”



2	-1	2
-2	0	1
1	3	4

4	1	-3	1
2	0	-2	3
4	2	-1	2
2	1	-1	0

Pointwise Multiplication

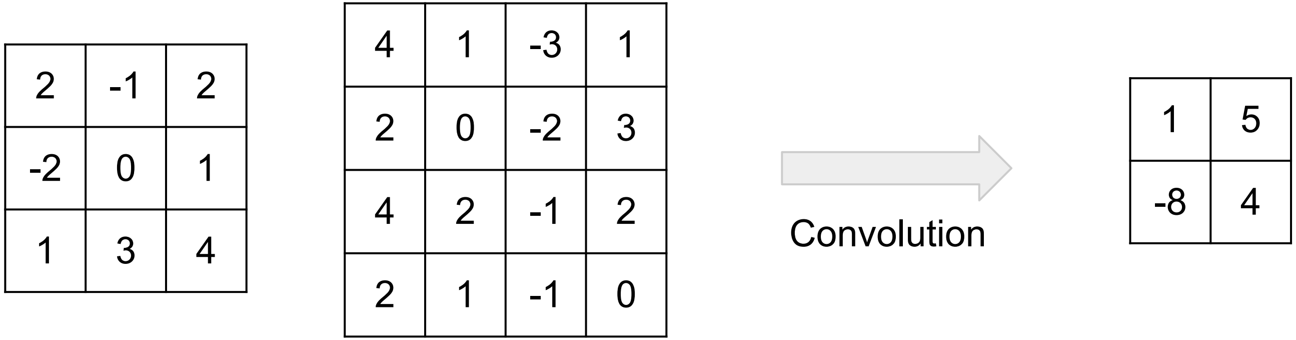
0	2	6
-4	0	2
1	-3	0

Sum

1	5
-8	4

Convolution

Stride 1 x 1   Padding “Valid”



2	-1	2
-2	0	1
1	3	4

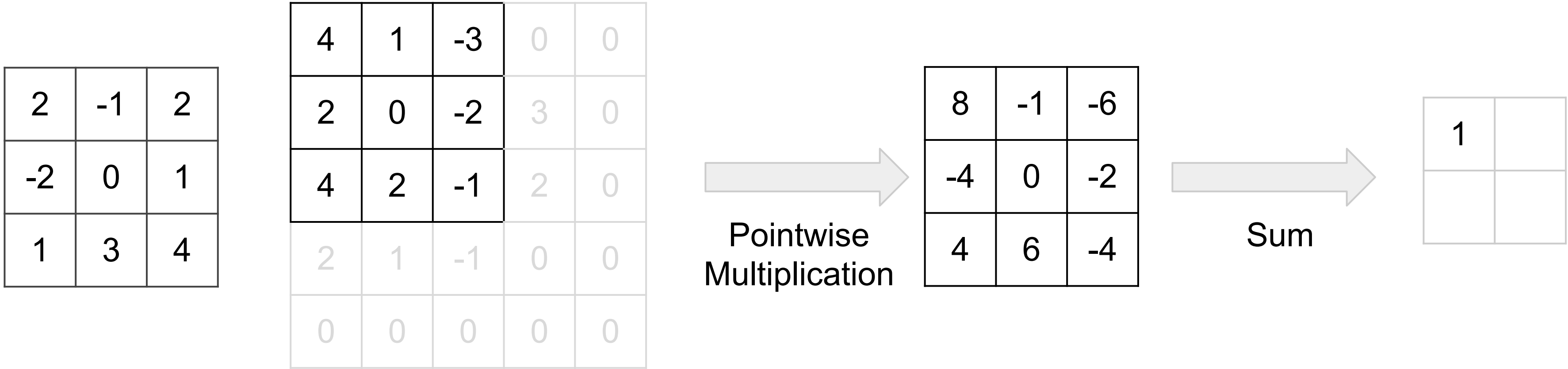
4	1	-3	1
2	0	-2	3
4	2	-1	2
2	1	-1	0

Convolution

1	5
-8	4

Convolution

Stride 2 x 2   Padding “Same”



2	-1	2
-2	0	1
1	3	4

4	1	-3	0	0
2	0	-2	3	0
4	2	-1	2	0
2	1	-1	0	0
0	0	0	0	0

Pointwise Multiplication

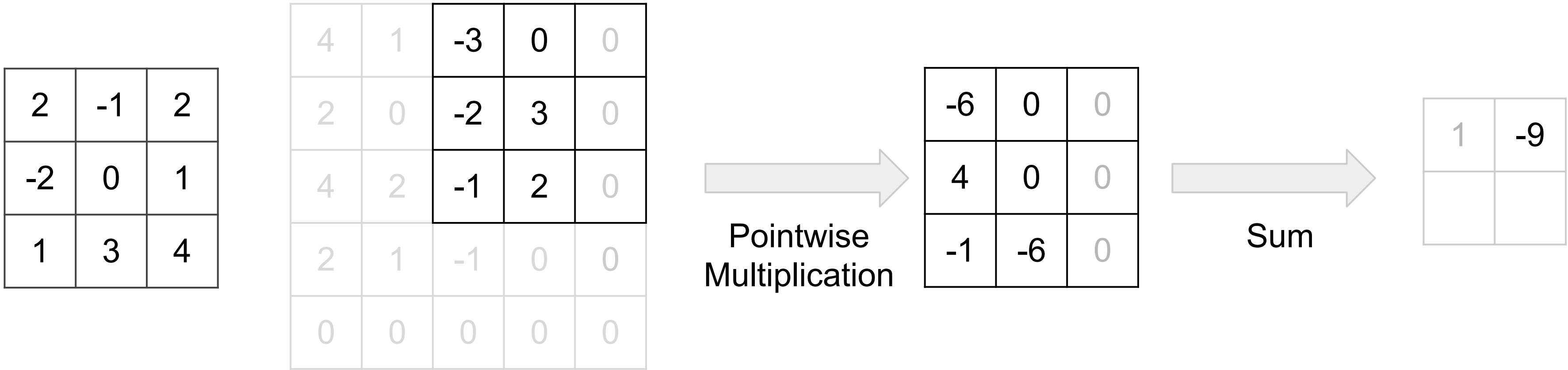
8	-1	-6
-4	0	-2
4	6	-4

Sum

1	

Convolution

Stride 2 x 2   Padding “Same”



2	-1	2
-2	0	1
1	3	4

4	1	-3	0	0
2	0	-2	3	0
4	2	-1	2	0
2	1	-1	0	0
0	0	0	0	0

Pointwise Multiplication

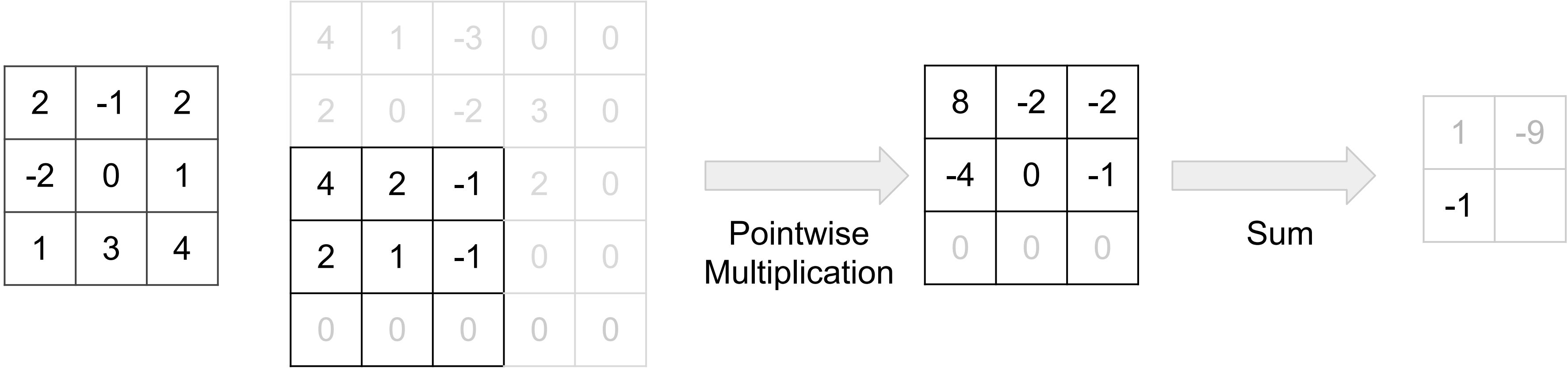
-6	0	0
4	0	0
-1	-6	0

Sum

1	-9

Convolution

Stride 2 x 2   Padding “Same”



2	-1	2
-2	0	1
1	3	4

4	1	-3	0	0
2	0	-2	3	0
4	2	-1	2	0
2	1	-1	0	0
0	0	0	0	0

Pointwise Multiplication

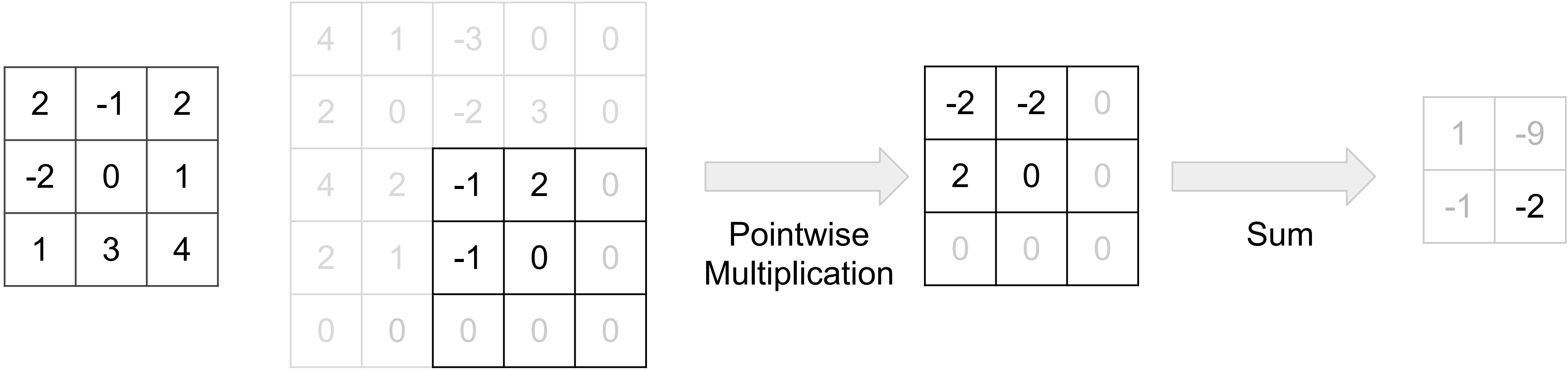
8	-2	-2
-4	0	-1
0	0	0

Sum

1	-9
-1	

Convolution

Stride 2 x 2   Padding “Same”



2	-1	2
-2	0	1
1	3	4

4	1	-3	0	0
2	0	-2	3	0
4	2	-1	2	0
2	1	-1	0	0
0	0	0	0	0

Pointwise Multiplication

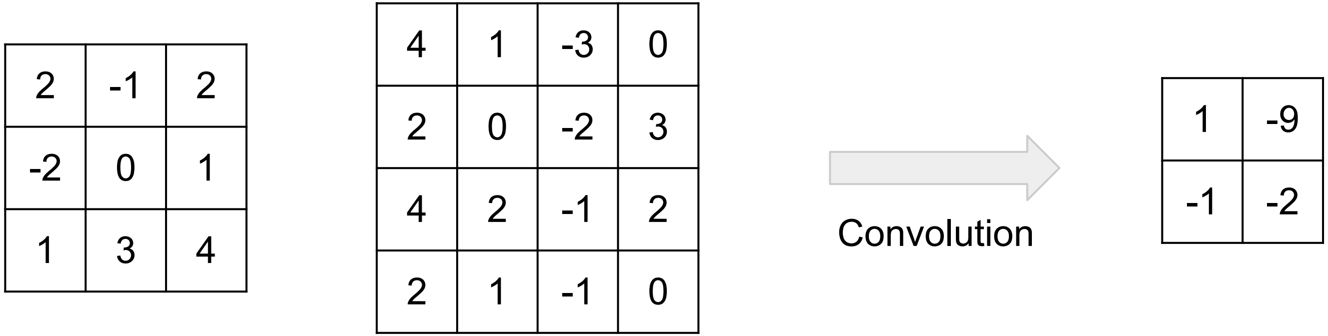
-2	-2	0
2	0	0
0	0	0

Sum

1	-9
-1	-2

Convolution

Stride 2 x 2   Padding “Same”



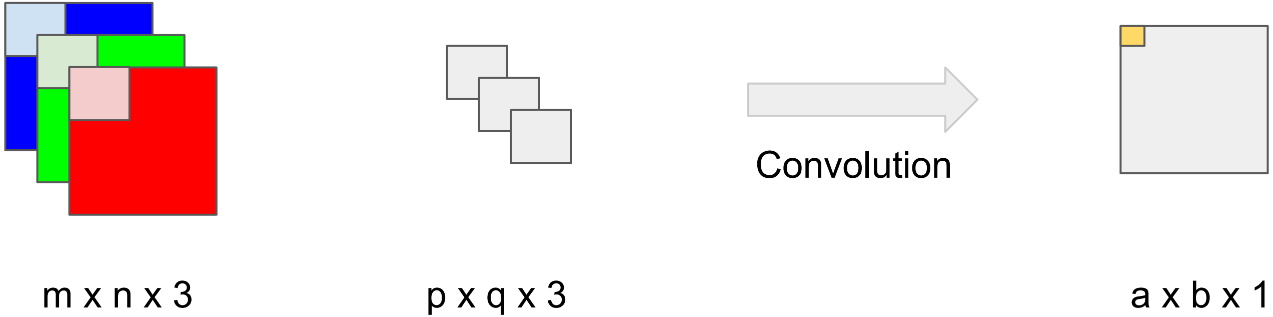
2	-1	2
-2	0	1
1	3	4

4	1	-3	0
2	0	-2	3
4	2	-1	2
2	1	-1	0

Convolution

1	-9
-1	-2

2D Convolution



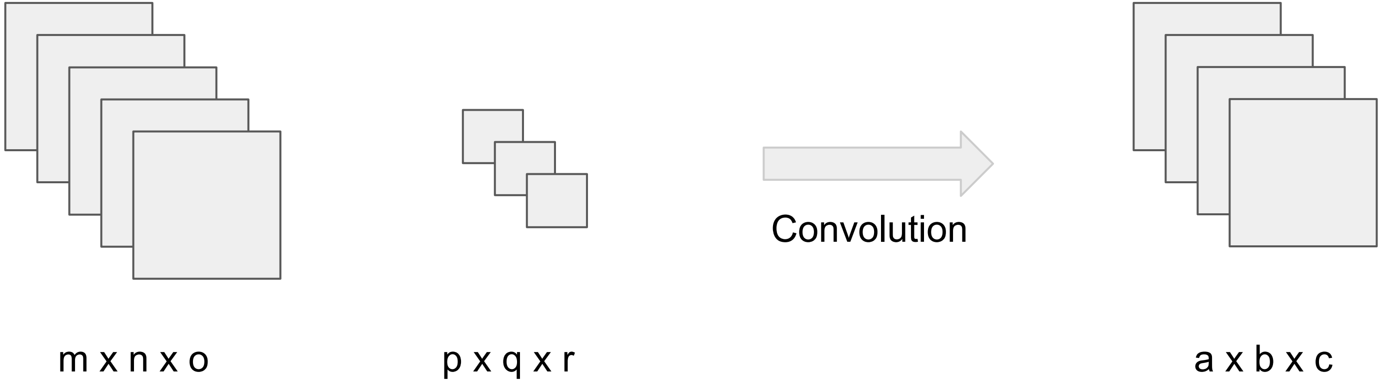
m x n x 3

p x q x 3

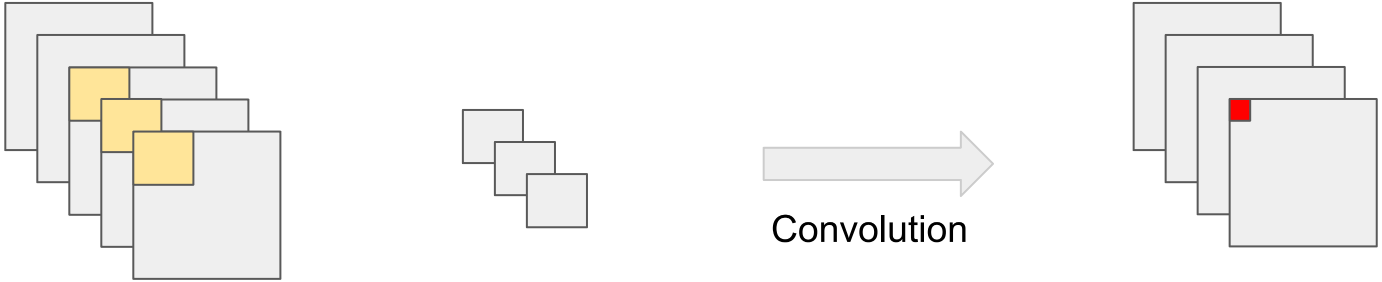
Convolution

a x b x 1

3D Convolution



3D Convolution



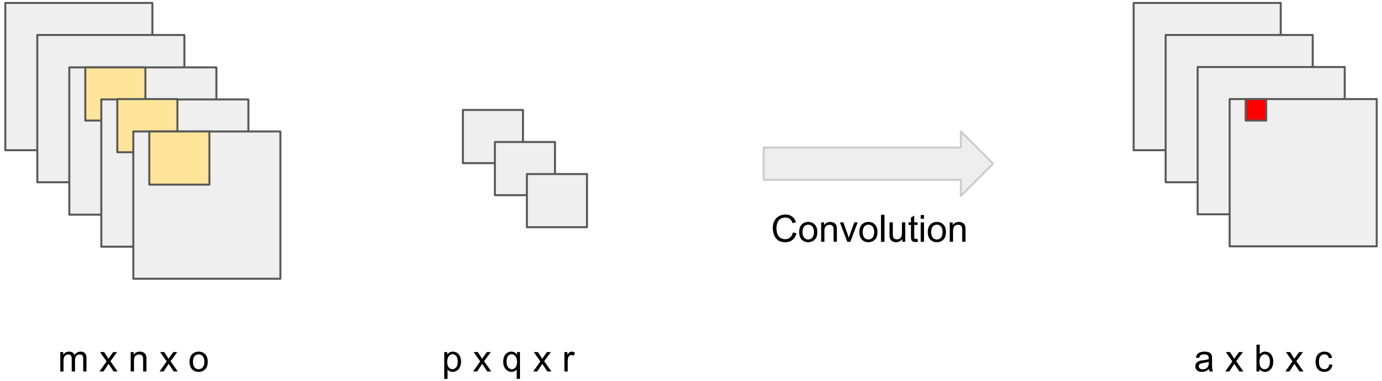
Convolution

m x n x o

p x q x r

a x b x c

3D Convolution



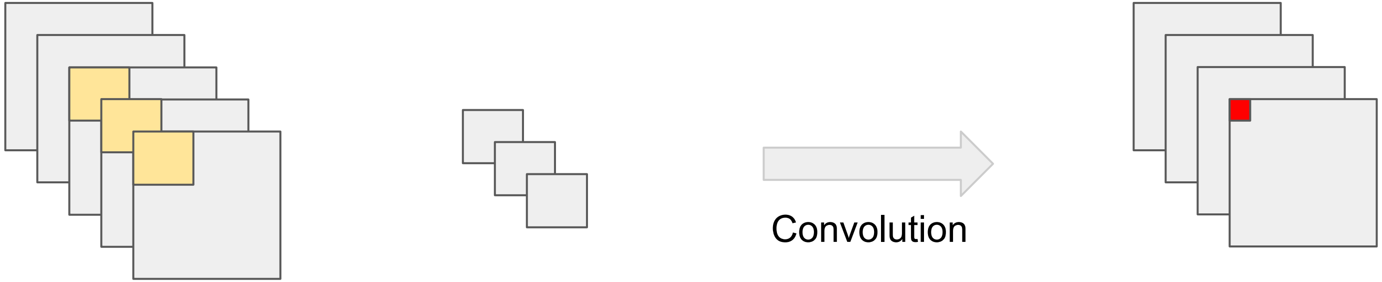
m x n x o

p x q x r

Convolution

a x b x c

3D Convolution



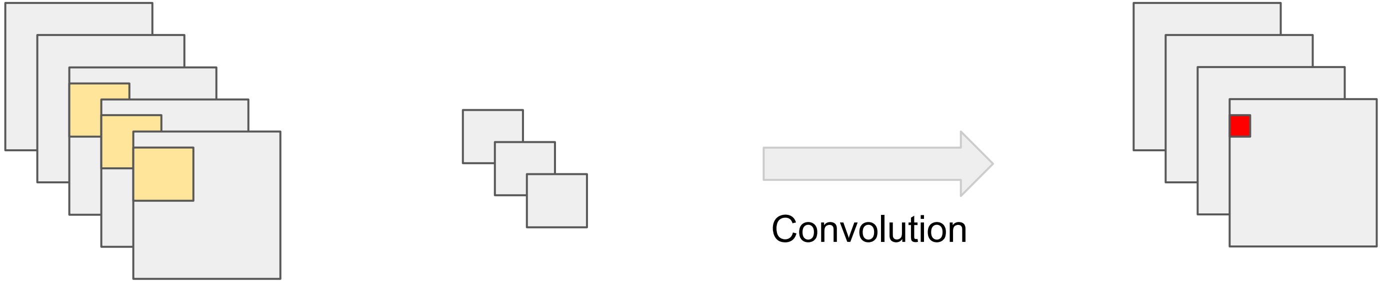
m x n x o

p x q x r

Convolution

a x b x c

3D Convolution



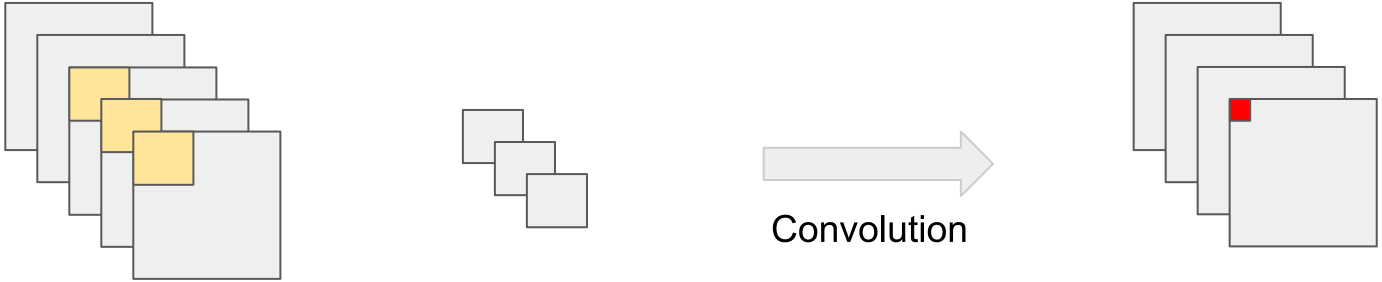
Convolution

m x n x o

p x q x r

a x b x c

3D Convolution



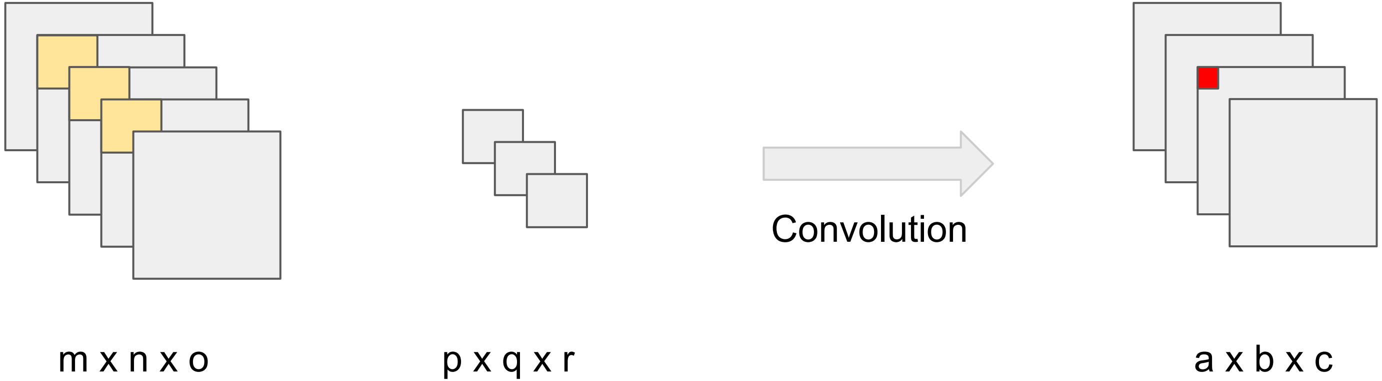
Convolution

m x n x o

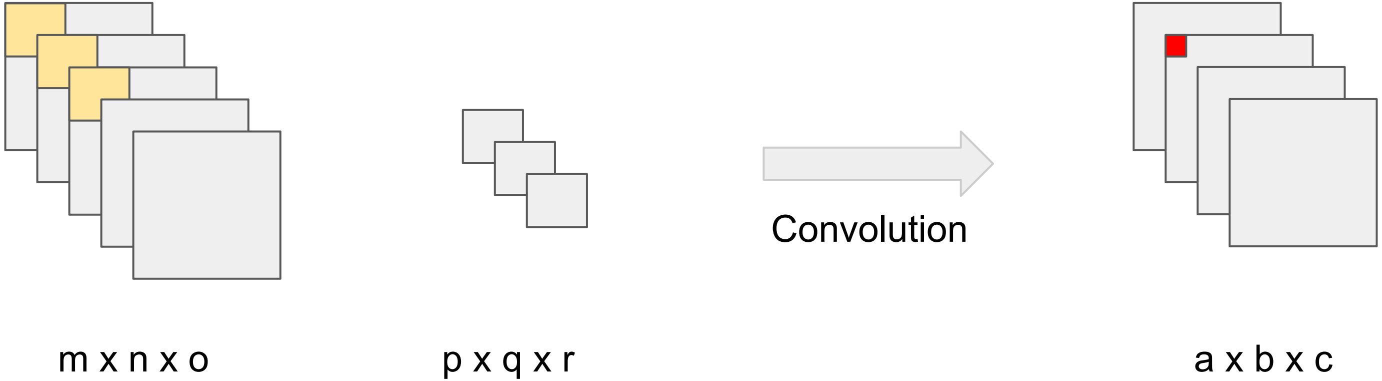
p x q x r

a x b x c

3D Convolution



3D Convolution



m x n x o

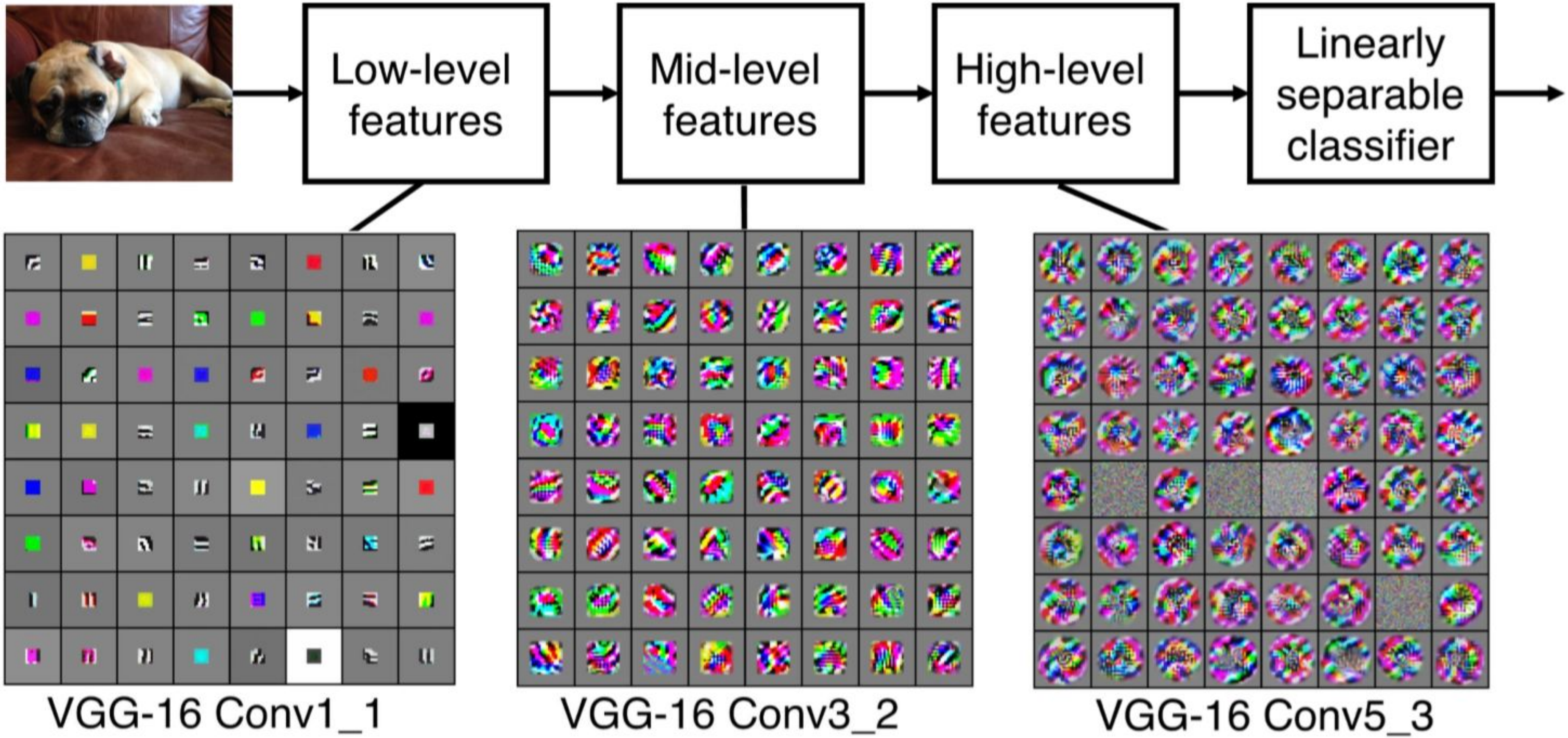
p x q x r

Convolution

a x b x c

Why is convolution important?

Hint: When is the inner product maximum?



Low-level features

Mid-level features

High-level features

Linearly separable classifier

VGG-16 Conv1_1

VGG-16 Conv3_2

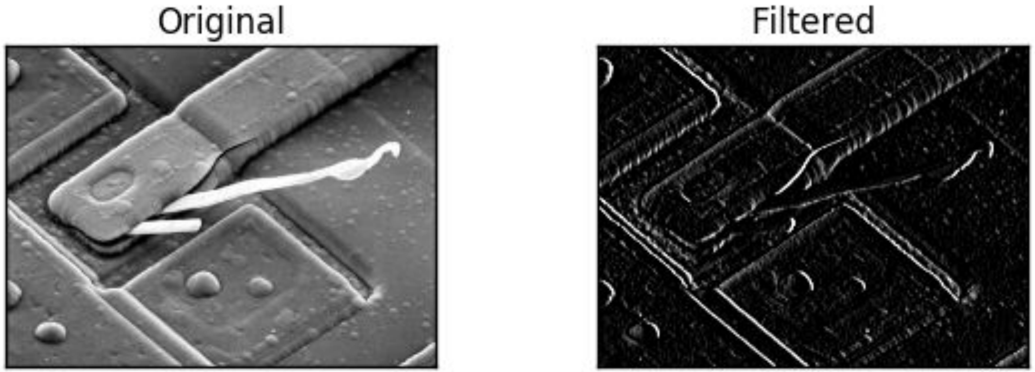
VGG-16 Conv5_3

Sobel X

1	0	-1
2	0	-2
1	0	-1

Taken from http://nbviewer.jupyter.org/github/akagrecha/ImFilters/blob/master/ImFilters.ipynb

Sobel X



Original

Filtered

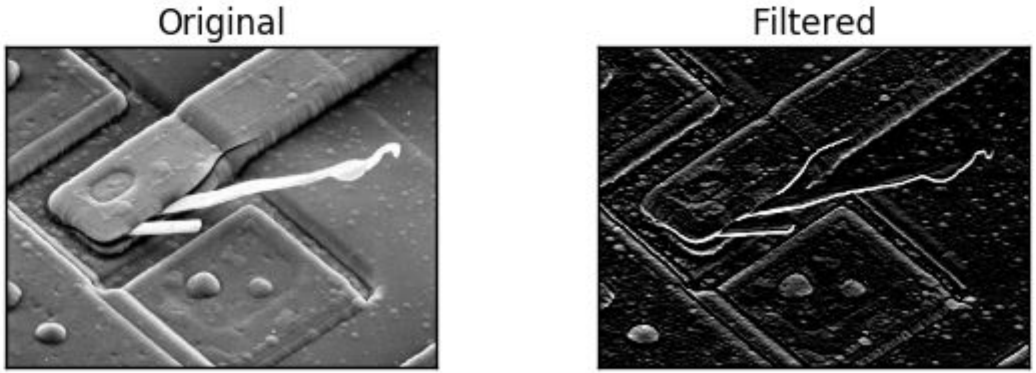
Taken from http://nbviewer.jupyter.org/github/akagrecha/ImFilters/blob/master/ImFilters.ipynb

Sobel Y

1	2	1
0	0	0
-1	-2	-1

Taken from http://nbviewer.jupyter.org/github/akagrecha/ImFilters/blob/master/ImFilters.ipynb

Sobel Y



Original

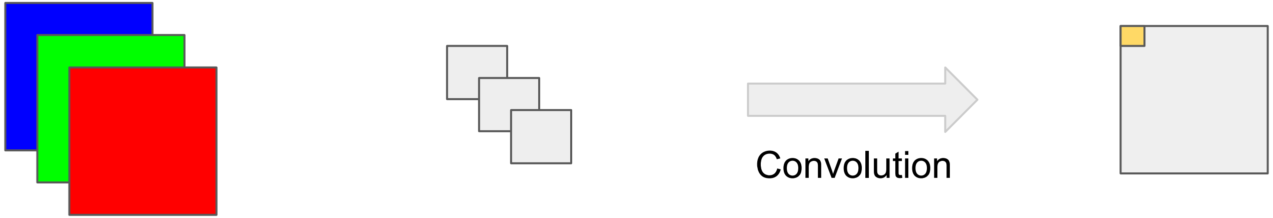
Filtered

Taken from http://nbviewer.jupyter.org/github/akagrecha/ImFilters/blob/master/ImFilters.ipynb

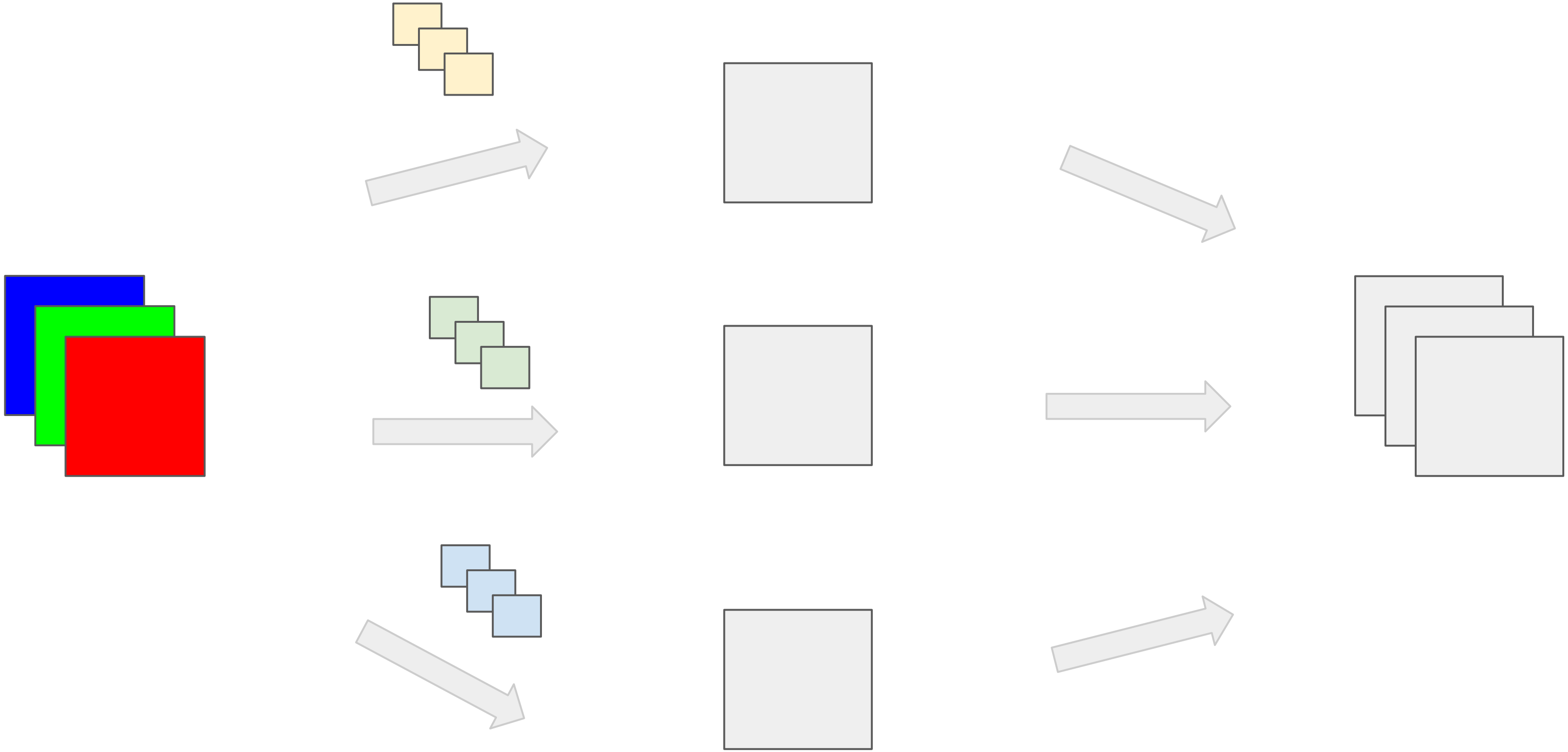
How to decide the kernel size and stride in CNN layers?

How do we exploit filters?

2D Convolutional Layer

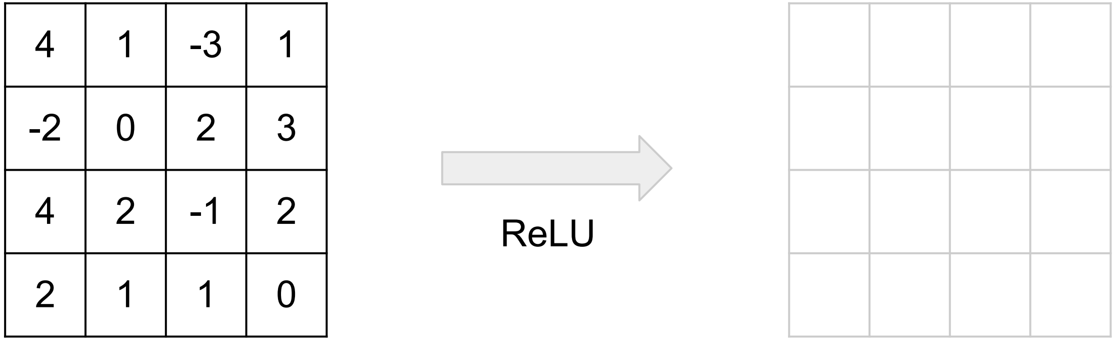


2D Convolutional Layer



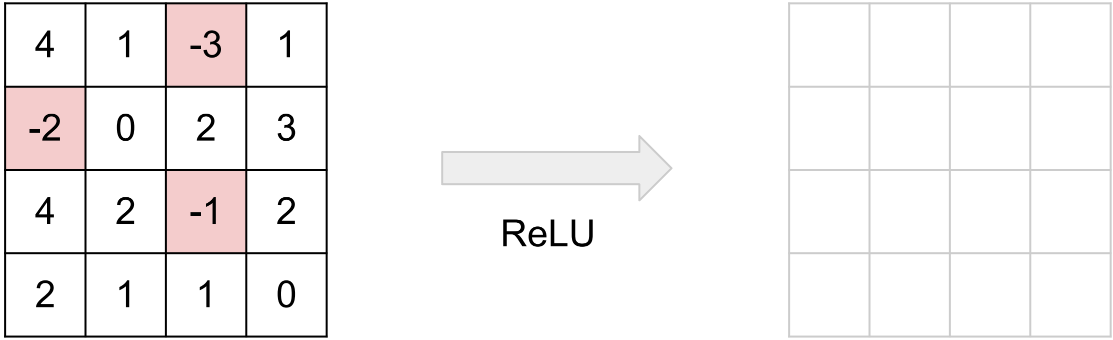
How do we decide the number of kernels in a layer?

Activation Function



4	1	-3	1
-2	0	2	3
4	2	-1	2
2	1	1	0

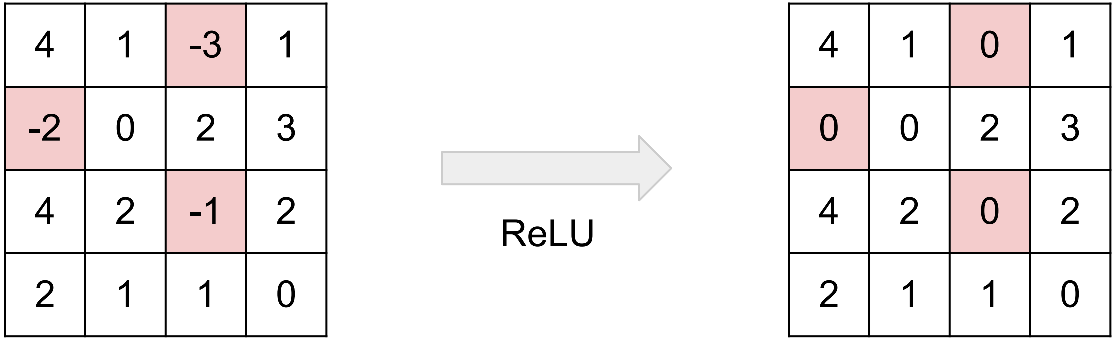
Activation Function



4	1	-3	1
-2	0	2	3
4	2	-1	2
2	1	1	0

ReLU

Activation Function

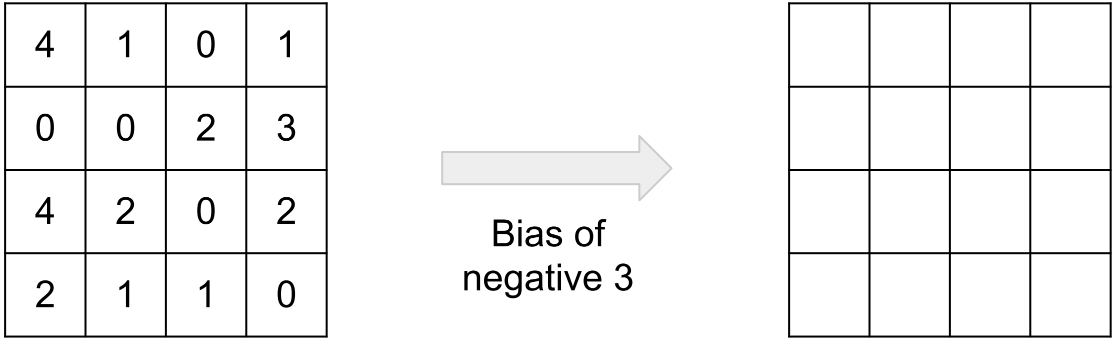


4	1	-3	1
-2	0	2	3
4	2	-1	2
2	1	1	0

ReLU

4	1	0	1
0	0	2	3
4	2	0	2
2	1	1	0

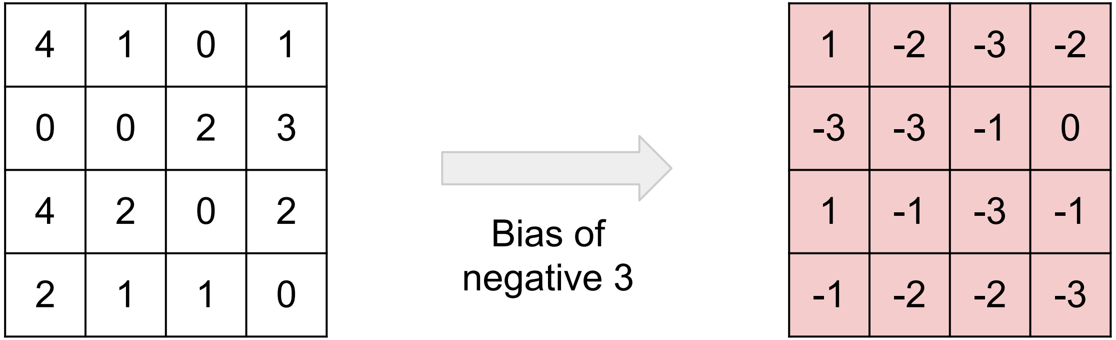
Activation Function



4	1	0	1
0	0	2	3
4	2	0	2
2	1	1	0

Bias of negative 3

Activation Function

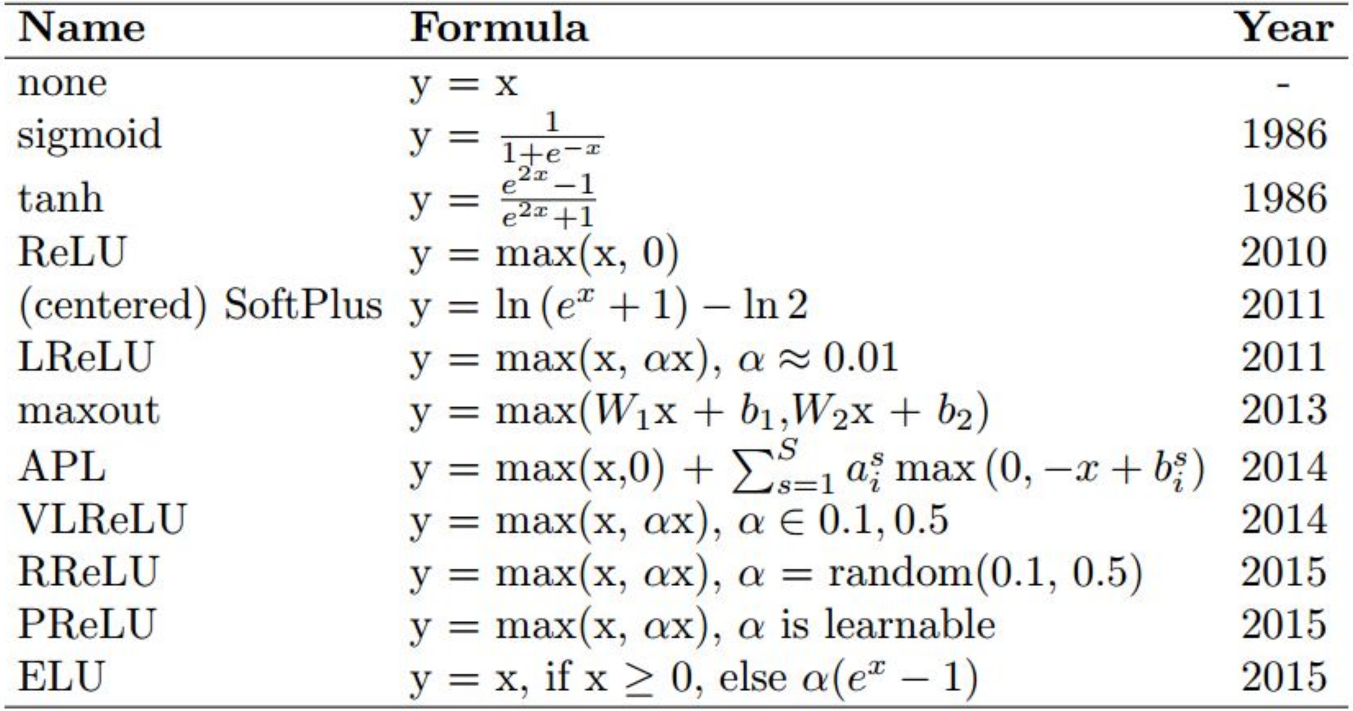


4	1	0	1
0	0	2	3
4	2	0	2
2	1	1	0

Bias of
negative 3

1	-2	-3	-2
-3	-3	-1	0
1	-1	-3	-1
-1	-2	-2	-3

Activation Functions



Name	Formula	Year
none	$y = x$	-
sigmoid	$y = \frac{1}{1+e^{-x}}$	1986
tanh	$y = \frac{e^{2x}-1}{e^{2x}+1}$	1986
ReLU	$y = \max(x, 0)$	2010
(centered) SoftPlus	$y = \ln\left(e^{x} + 1\right) - \ln 2$	2011
LReLU	$y = \max(x, \alpha x),\ \alpha \approx 0.01$	2011
maxout	$y = \max(W_1 x + b_1, W_2 x + b_2)$	2013
APL	$y = \max(x,0) + \sum_{s=1}^{S} a_i^s \max\left(0, -x + b_i^s\right)$	2014
VLReLU	$y = \max(x, \alpha x),\ \alpha \in 0.1, 0.5$	2014
RReLU	$y = \max(x, \alpha x),\ \alpha = \text{random}(0.1, 0.5)$	2015
PReLU	$y = \max(x, \alpha x),\ \alpha$ is learnable	2015
ELU	$y = x$, if $x \geq 0$, else $\alpha(e^{x} - 1)$	2015

Table taken from https://arxiv.org/pdf/1606.02228v2.pdf

Why is activation function important?

Why is ReLU so widely used?

Why not some other function like sigmoid or hyperbolic tangent?

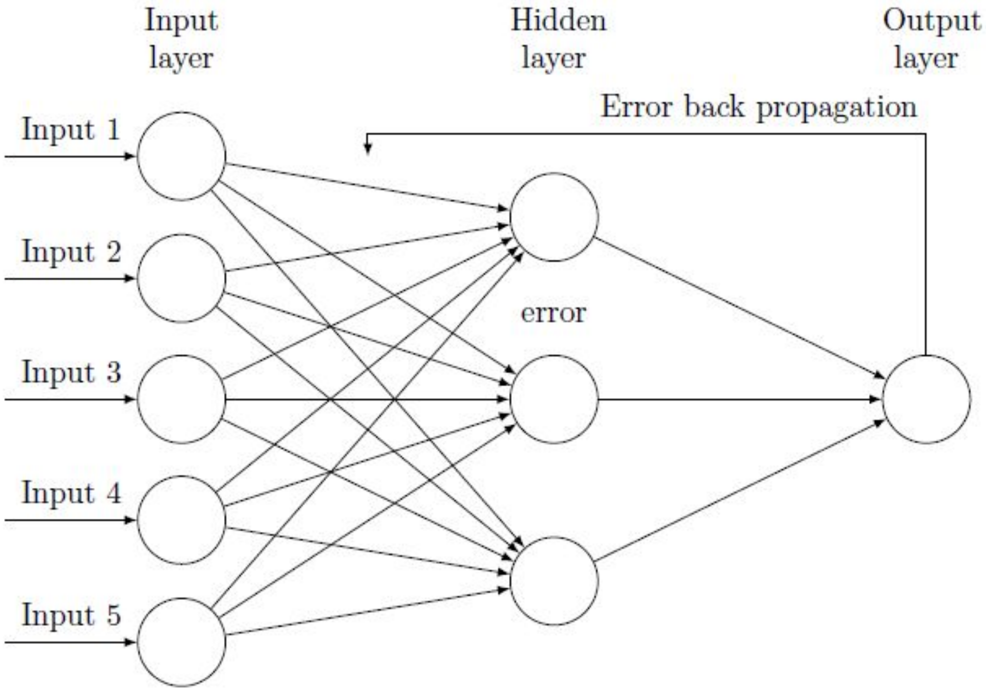
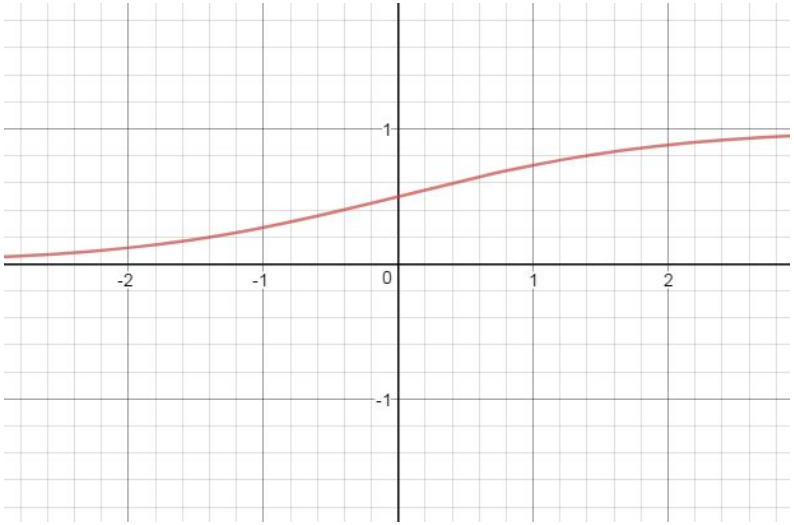
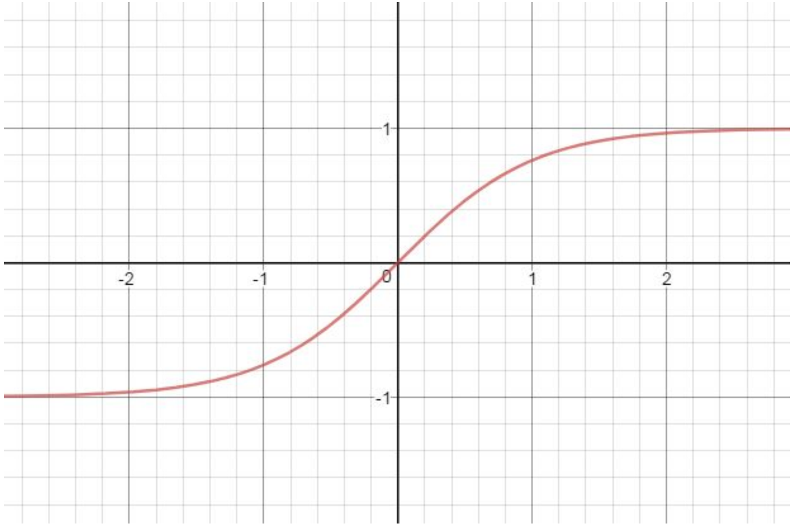


Image from TEX StackExchange (https://tex.stackexchange.com/questions/162326/drawing-back-propagation-neural-network)



Sigmoid



Hyperbolic Tangent

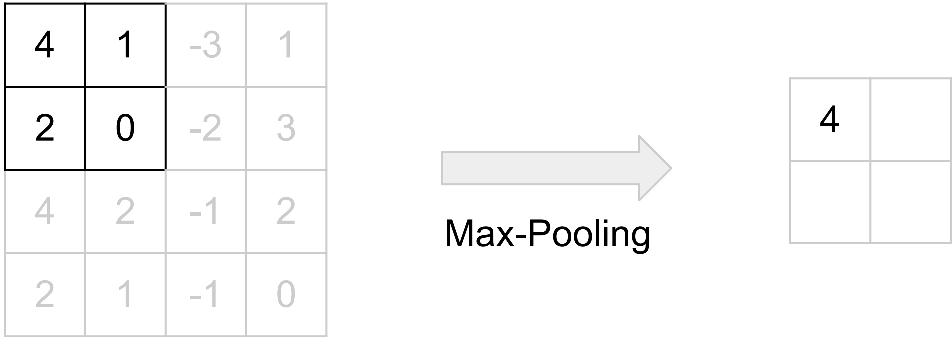
Drawn using Desmos Graphing Calculator (https://www.desmos.com/calculator)

Max-Pooling

Kernel Size 2x2 Stride 2 x 2 Padding “Valid”/”Same”

Max-Pooling

Kernel Size 2x2 Stride 2 x 2 Padding “Valid”/”Same”

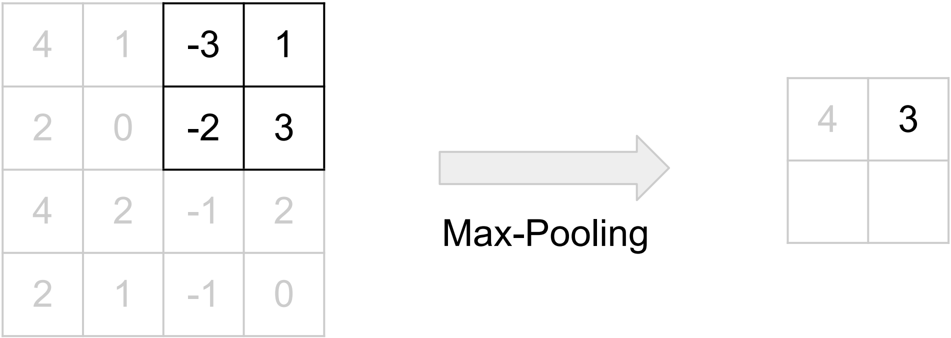


4	1	-3	1
2	0	-2	3
4	2	-1	2
2	1	-1	0

4	

Max-Pooling

Kernel Size 2x2 Stride 2 x 2 Padding “Valid”/”Same”

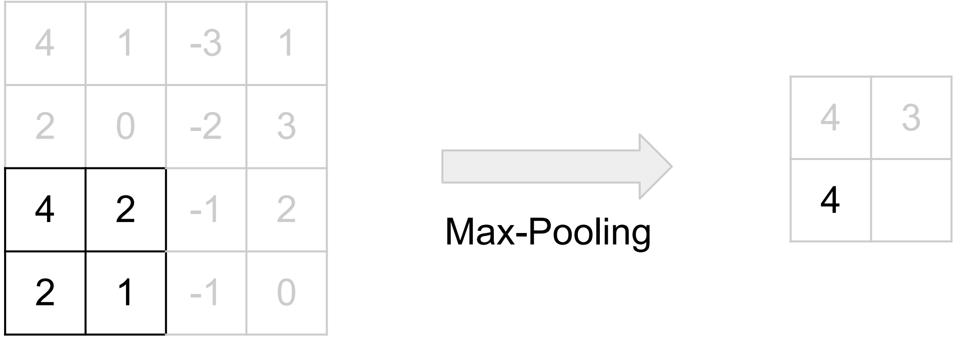


4	1	-3	1
2	0	-2	3
4	2	-1	2
2	1	-1	0

4	3

Max-Pooling

Kernel Size 2x2 Stride 2 x 2 Padding “Valid”/”Same”



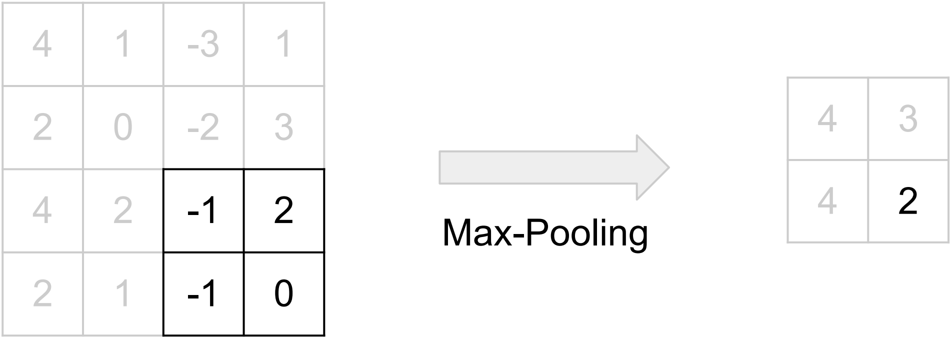
4	1	-3	1
2	0	-2	3
4	2	-1	2
2	1	-1	0

Max-Pooling

4	3
4	

Max-Pooling

Kernel Size 2x2   Stride 2 x 2   Padding “Valid”/”Same”



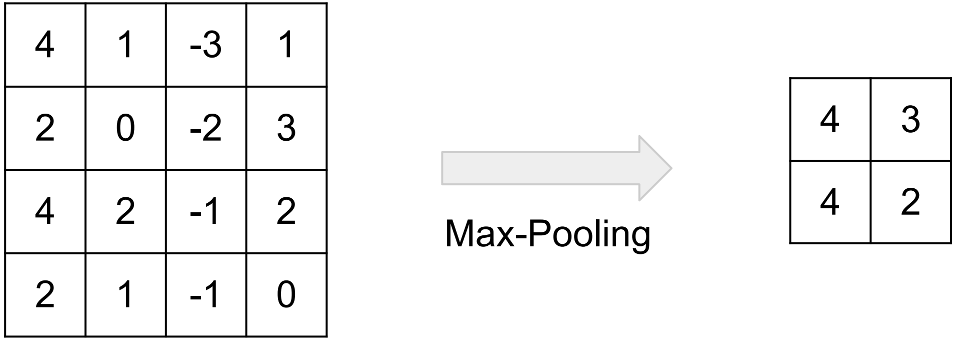
4	1	-3	1
2	0	-2	3
4	2	-1	2
2	1	-1	0

Max-Pooling

4	3
4	2

Max-Pooling

Kernel Size 2x2 Stride 2 x 2 Padding “Valid”/”Same”



4	1	-3	1
2	0	-2	3
4	2	-1	2
2	1	-1	0

4	3
4	2

What about 2D (for RGB) and 3D Max-Pooling?

Other Types of Pooling

- Average pooling
- L2-norm pooling
- Max + Average

Why is pooling important?

What is difference between different kinds of pooling?

Dense/ Fully connected/ Perceptron Layers

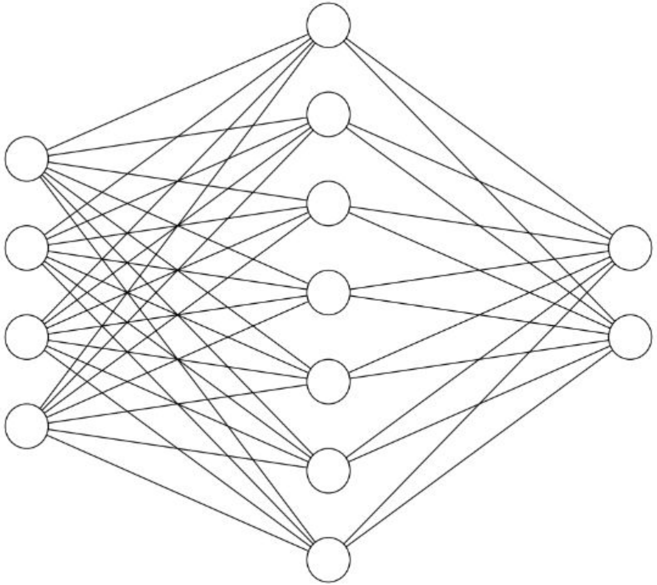
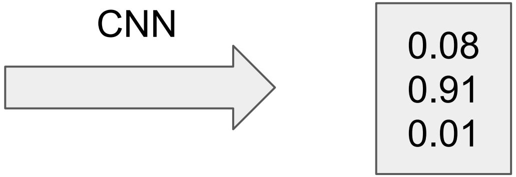


Image taken from https://gist.github.com/craffel/2d727968c3aaebd10359

Why is multi-layer perceptron structure important in CNNs?

Input and Output for Classification Problem

Image (RGB values)



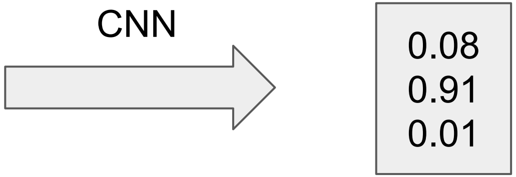
CNN

0.08
0.91
0.01

Probability
Vector

Image taken from Google Images

Input and Output for Classification Problem



CNN

0.08
0.91
0.01

Prob (Dog | Image) = 0.08
Prob (Cat | Image) = 0.91
Prob (Rat | Image) = 0.01

Image taken from Google Images

Architecture for CNNs

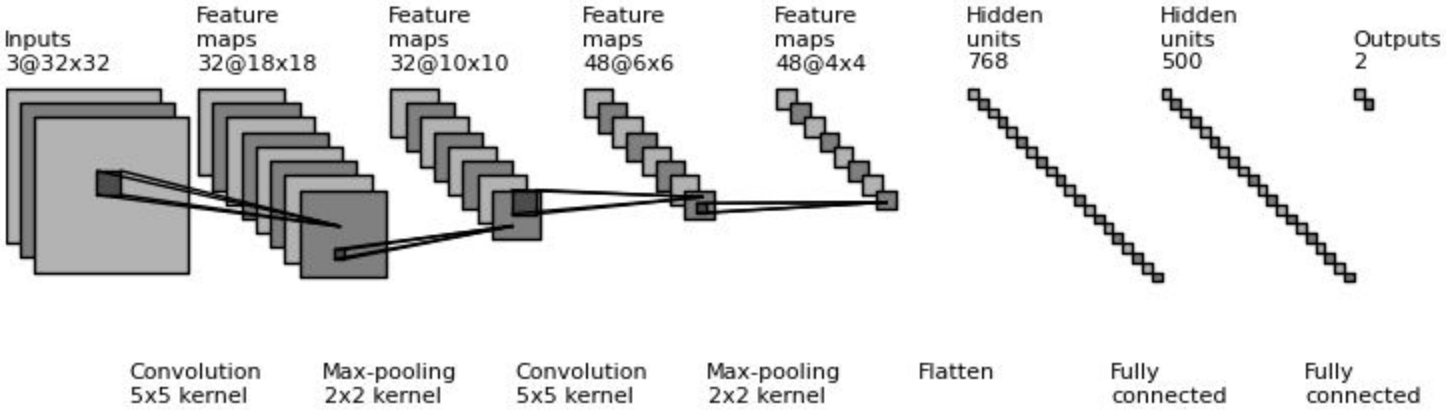
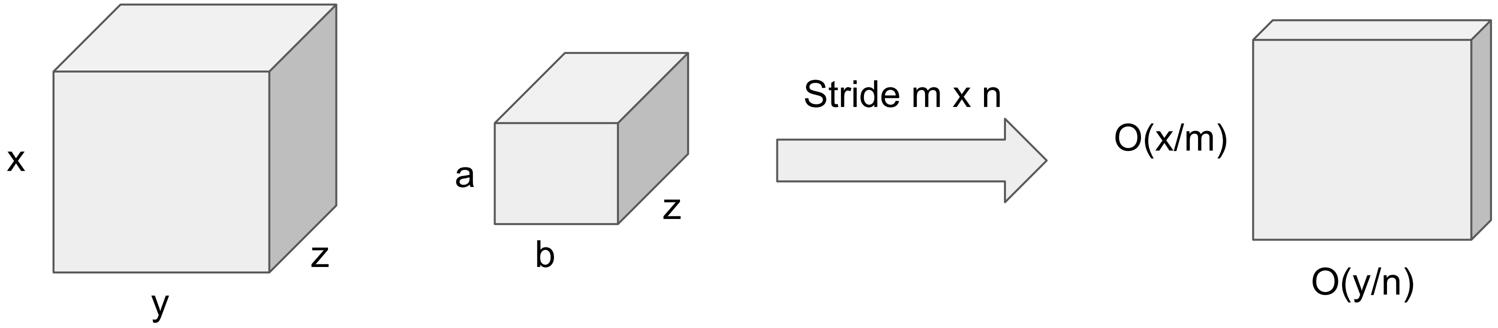


Image taken from https://github.com/gwding/draw_convnet/blob/master/convnet_fig.png`

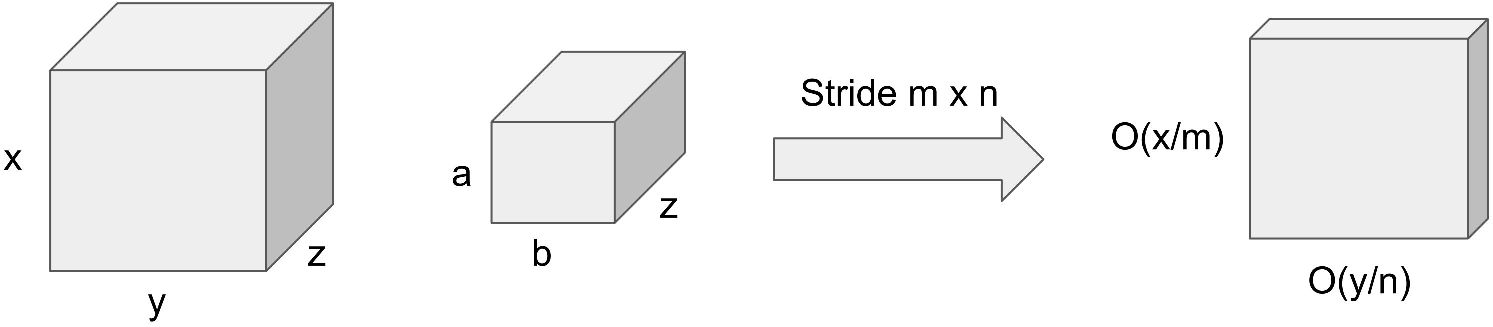
How to decide network depth in CNNs?

Space Complexity of Forward Pass



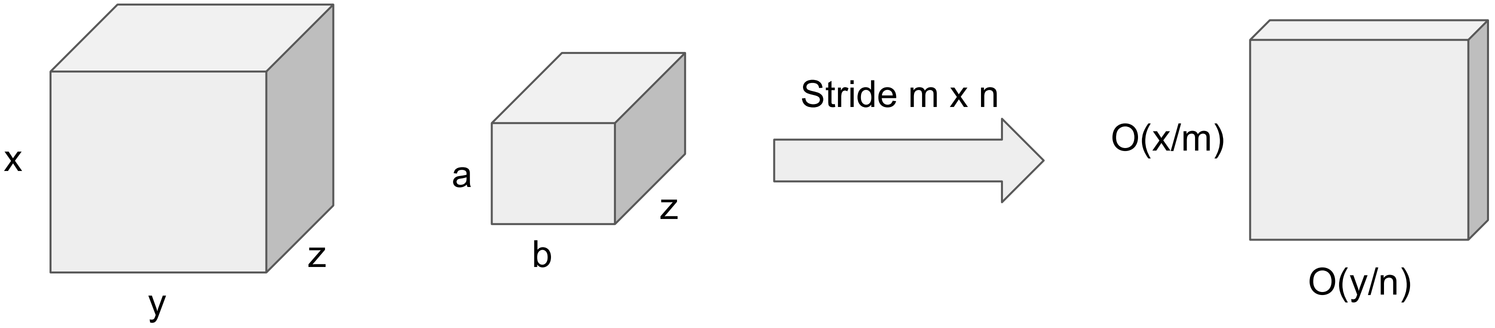
$$O\left(\frac{x}{m} \cdot \frac{y}{n}\right)$$

Space Complexity of Backpropagation



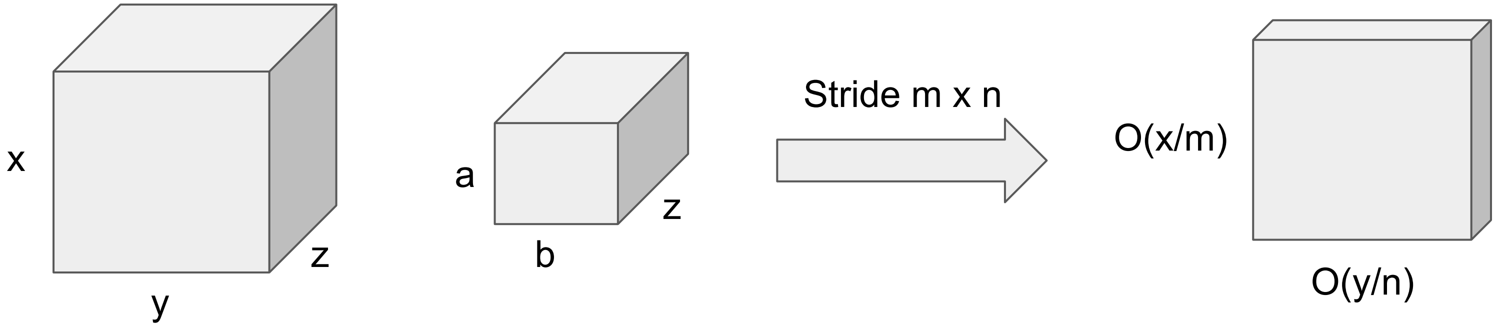
$$O(abc)$$

Time Complexity of Forward Pass



$$O\left(abz \cdot \frac{x}{m} \cdot \frac{y}{n}\right)$$

Space Complexity of Backpropagation



$$O(abc)$$

How to model probabilities?

?

How to model probabilities?

Softmax function!

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

How to calculate error of model?

Cross Entropy Loss

$$L = -\frac{1}{M}\sum_{i=0}^{M} log(p(c_i|X = x_i))$$

How to train model?

THE END

Thank you.