Neural Networks

Lecture 3 - Convolutional Neural Networks

Igor Janos

Agenda

1. Convolution Operation
    - Examples of hand-crafted convolutional filters
2. Convolutional Neural Network
    - Convolutional layer, assembling multiple layers together
    - Convolutional architectures
3. Siamese Networks
    - Image similarity, Face recognition

1 Convolution Operation

Convolution

- Integral transform

Reflected about y-axis and shifted

$$(f * g)(t) := \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau$$

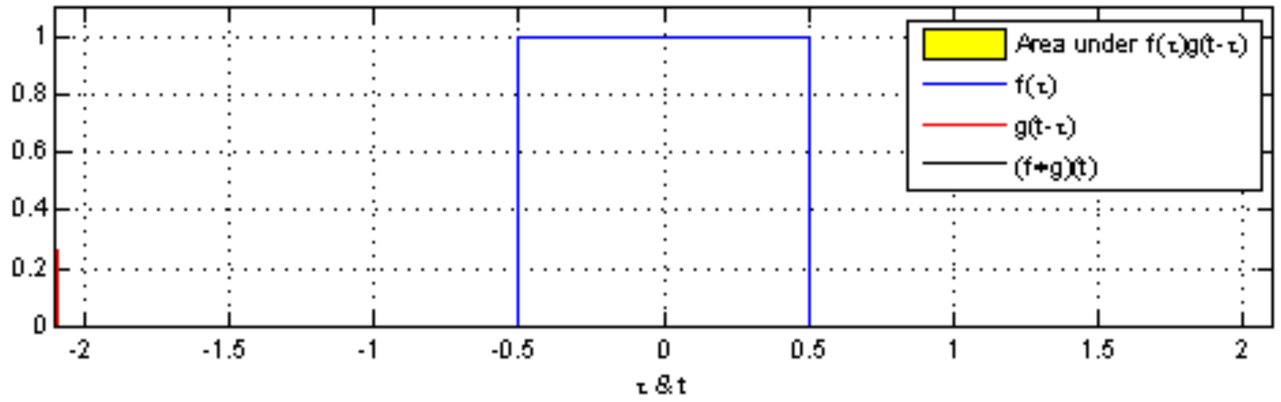


Image - https://en.wikipedia.org/wiki/Convolution

Discrete Convolution

Reflected about y-axis and shifted

$$(f * g)[n] := \sum_{m=-\infty}^{\infty} f[m]g[n-m]$$

Discrete Convolution

Commutativity

$$(f * g) = (g * f)$$

$$(f * g)(t) := \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau \qquad (f * g)[n] := \sum_{m=-\infty}^{\infty} f[m]g[n-m]$$

$$(f * g)(t) := \int_{-\infty}^{\infty} f(t-\tau)g(\tau)d\tau \qquad (f * g)[n] := \sum_{m=-\infty}^{\infty} f[n-m]g[m]$$

Discrete Convolution

- Let's define g only on a small interval $m \in \{-1,0,1\}$

$$(f * g)[n] := \sum_{m=-1}^{1} f[n-m]g[m]$$

Kernel with size 3

Discrete Convolution

Image Processing

- Extend for 2-dimensional data

$$(f * g)[x, y] := \sum_{i=-a}^{a} \sum_{j=-b}^{b} f[x - i, y - j] g[i, j]$$

Kernel size = $(2a + 1) \times (2b + 1)$

Discrete Convolution

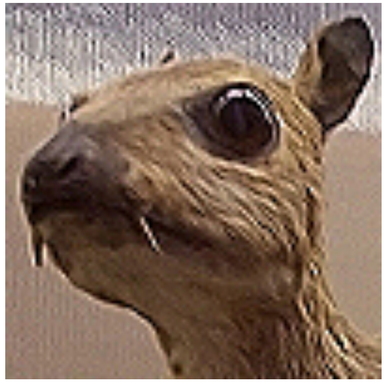
Image Processing - Examples

$$\frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}$$

Gaussian Blur

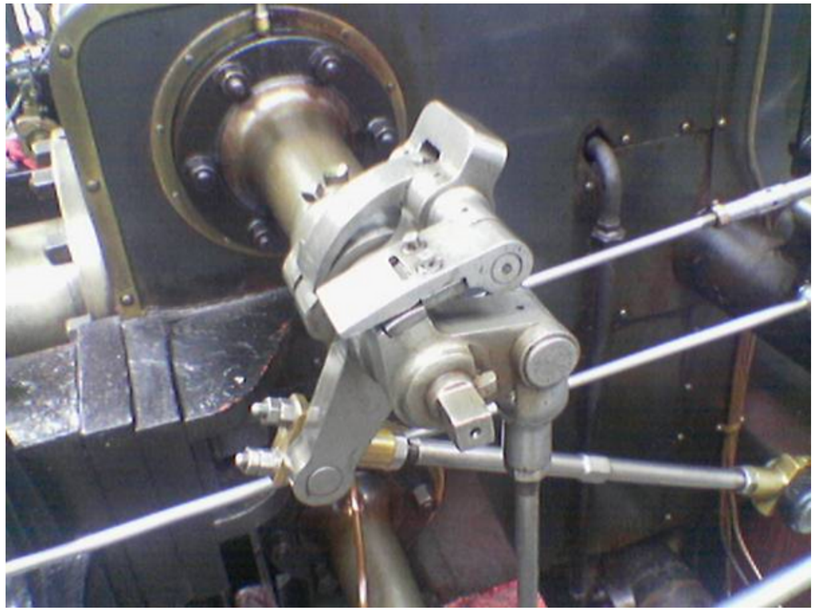
$$\begin{bmatrix}0 & -1 & 0\\ -1 & 5 & -1\\ 0 & -1 & 0\end{bmatrix}$$

Sharpen



$$\begin{bmatrix}1 & 0 & -1\\ 2 & 0 & -2\\ 1 & 0 & -1\end{bmatrix}$$

Sobel Operator



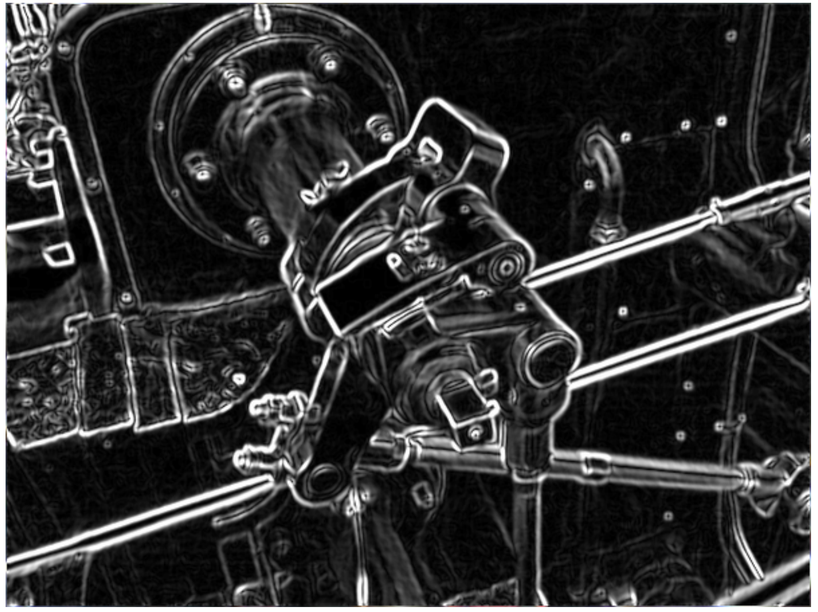


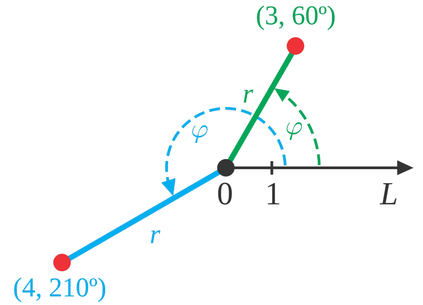
Image - https://en.wikipedia.org/wiki/Kernel_(image_processing)

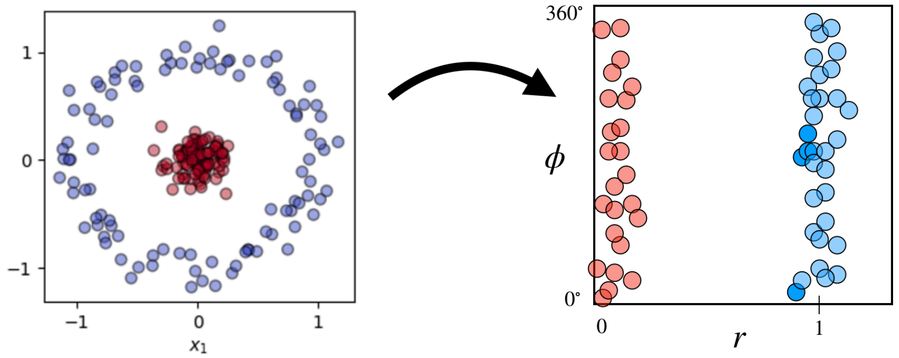
Image - https://en.wikipedia.org/wiki/Sobel_operator

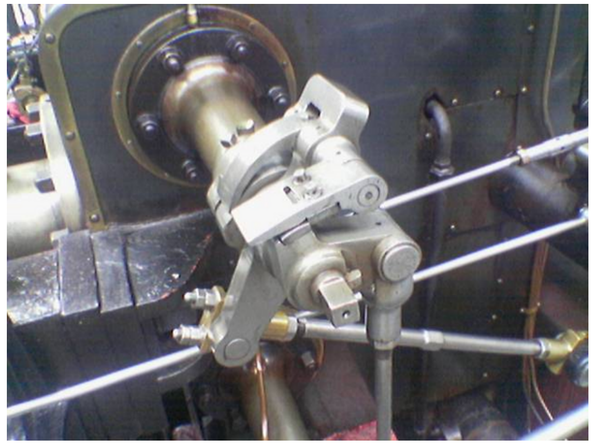
Discrete Convolution

Image Processing - Examples

- Convolution gives us **representations**
- Representations are specified by the **kernel matrix**
- Hand-crafted or **learned**







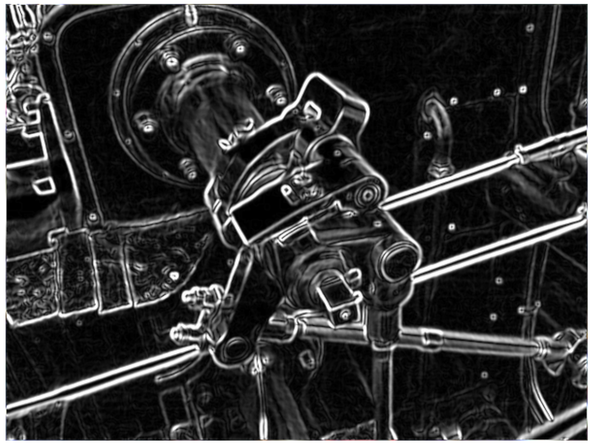


Image - https://en.wikipedia.org/wiki/Sobel_operator

2 Convolutional Neural Network

Convolutional Neural Network

- **Inspired** by biology
    - Visual cortex - topographical representation of the field of vision
    - Reception field of neighbouring cells corresponds to adjacent portions of the field of vision
    - Sequence of cells can perceive complex shapes

- **Uses convolution as building blocks**

Convolutional Neural Network

Convolution vs. Cross-Correlation

Convolution

$$(f * g)[n] := \sum_{m=-1}^{1} f[n-m]g[m]$$

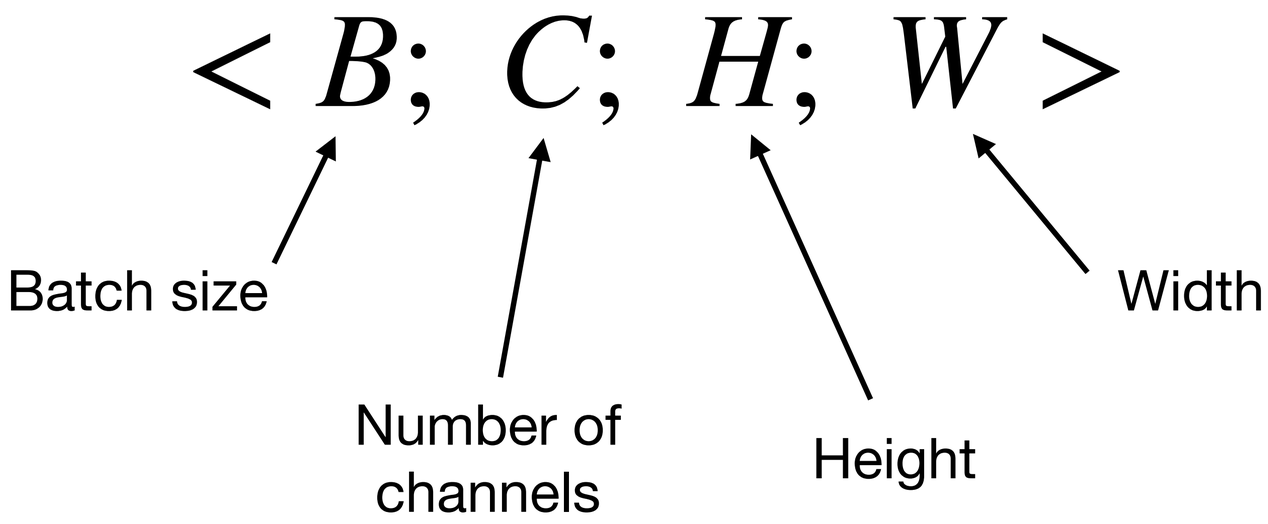
Cross-Correlation

$$(f \circ g)[n] := \sum_{m=-1}^{1} f[n+m]g[m]$$

- The **kernel is learned** - we do not flip the kernel
- Cross-correlation is easier to implement
- We call it ***"Convolution"***

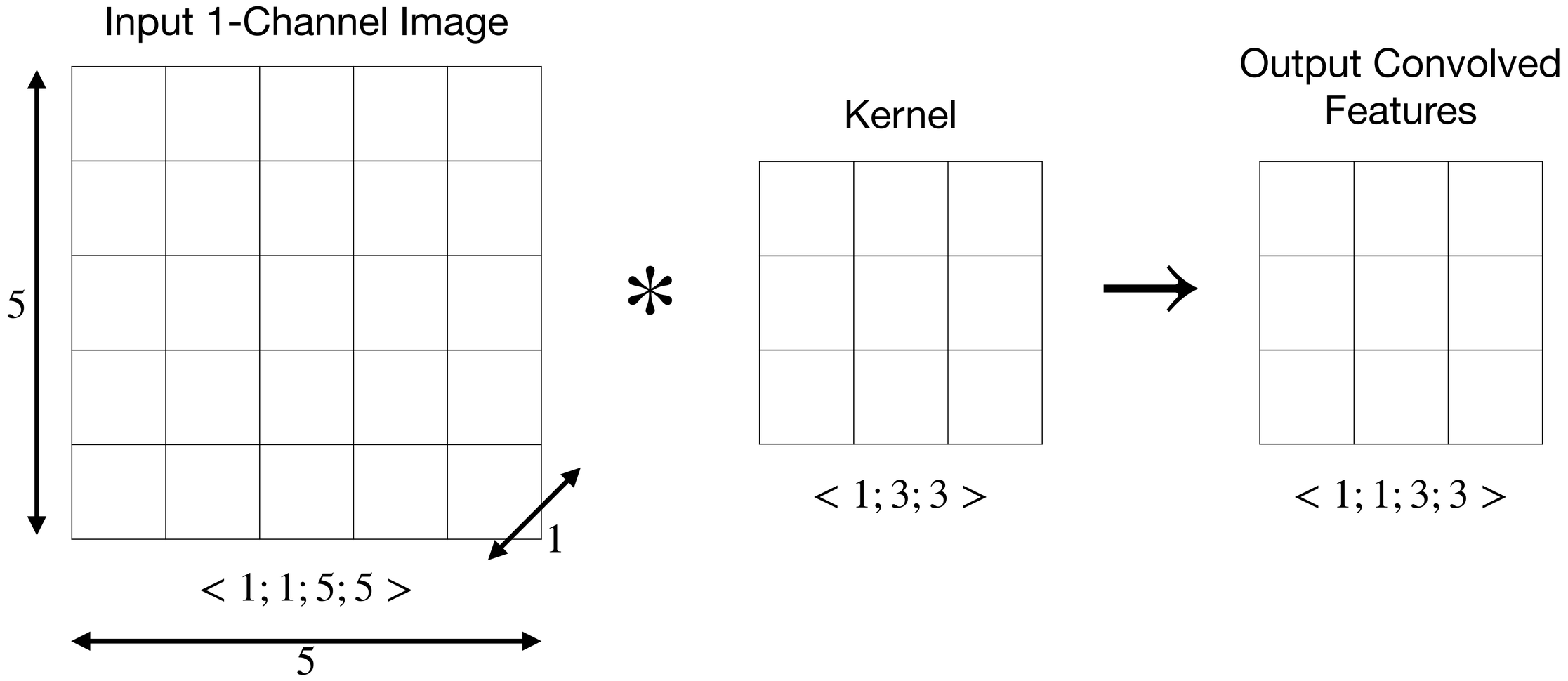
Convolutional Layer

- Assume we are working with 2D data (images)
- Our data (tensors) come in shape:



Convolutional Layer

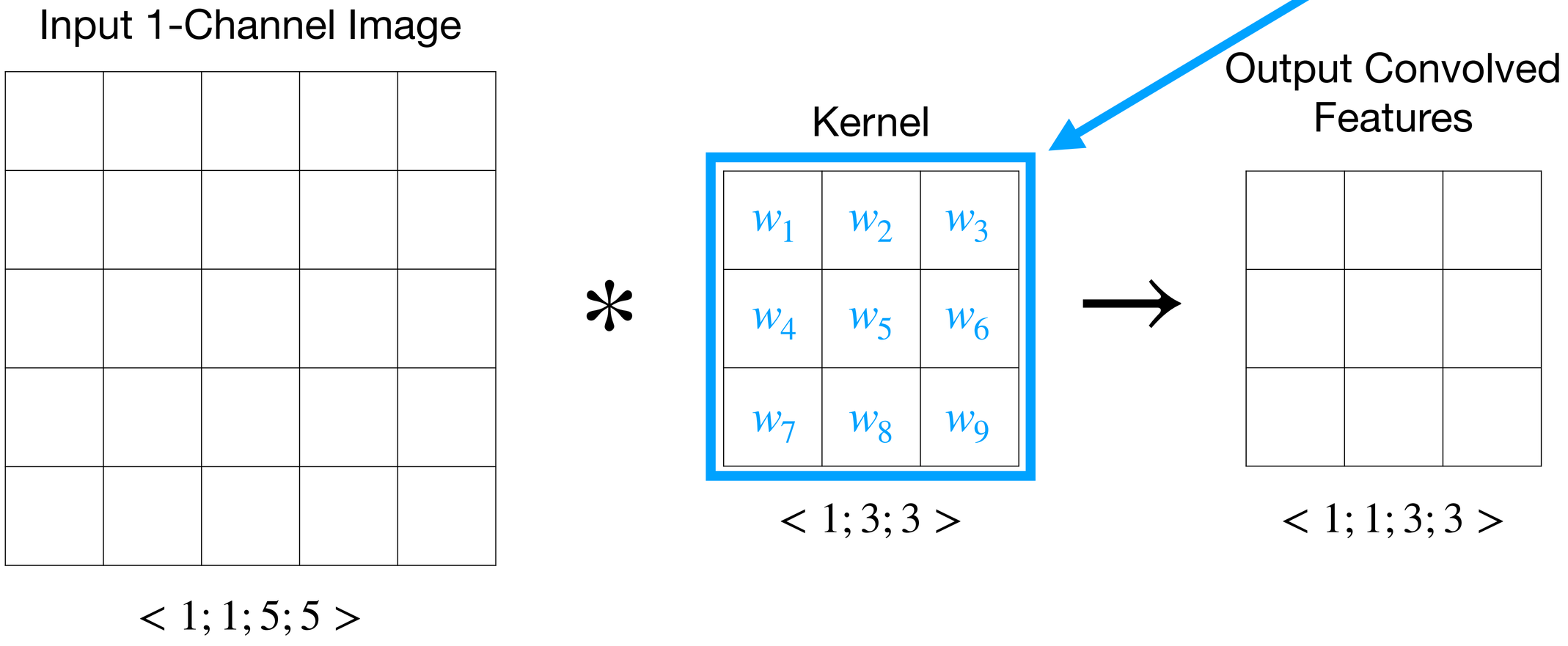
Example



Convolutional Layer

Example

$$(f * g)[x, y] := \sum_{i=-1}^{1} \sum_{j=-1}^{1} f[x+i, y+j] g[i, j]$$



Convolutional Layer

Example

$$(f * g)[x, y] := \sum_{i=-1}^{1} \sum_{j=-1}^{1} f[x+i, y+j] g[i, j]$$

Input 1-Channel Image

x_1	x_2	x_3		
x_4	x_5	x_6		
x_7	x_8	x_9		

$< 1; 1; 5; 5 >$

$*$

Kernel

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

$< 1; 3; 3 >$

$\rightarrow$

Output Convolved Features

$< 1; 1; 3; 3 >$

Convolutional Layer

Example

$$(f * g)[x, y] := \sum_{i=-1}^{1} \sum_{j=-1}^{1} f[x+i, y+j] g[i, j]$$

Input 1-Channel Image

x_1	x_2	x_3		
x_4	x_5	x_6		
x_7	x_8	x_9		

$< 1; 1; 5; 5 >$

$*$

Kernel

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

$< 1; 3; 3 >$

$\rightarrow$

Output Convolved Features

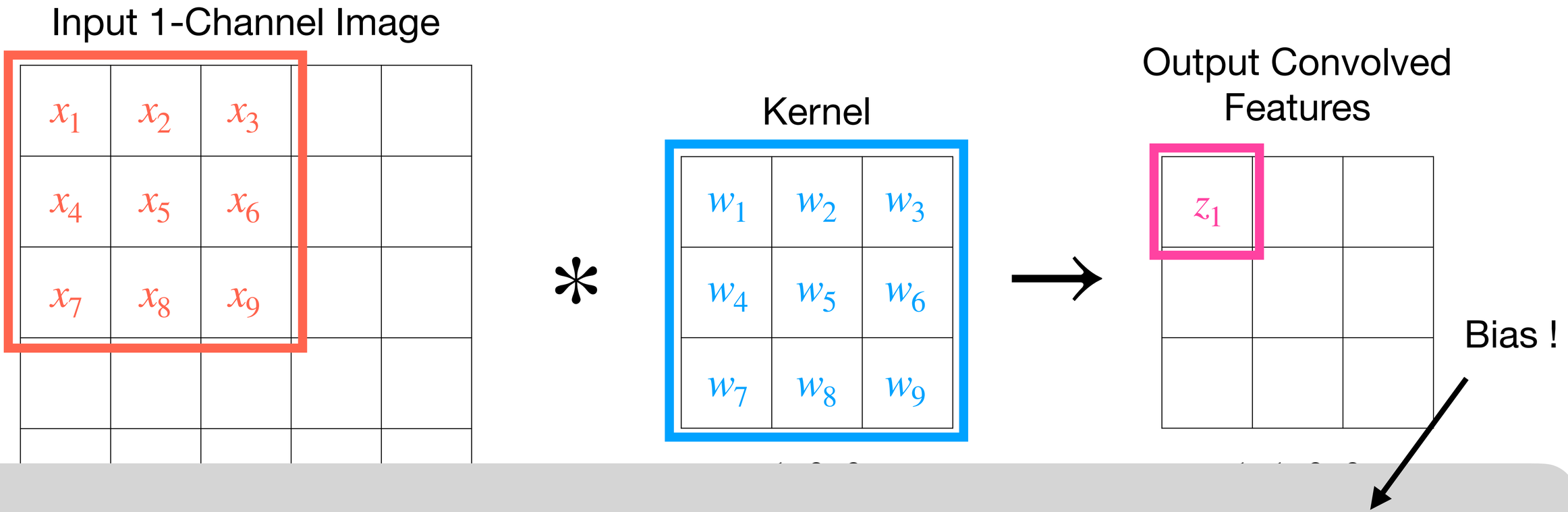
z_1		

$< 1; 1; 3; 3 >$

Convolutional Layer

Example

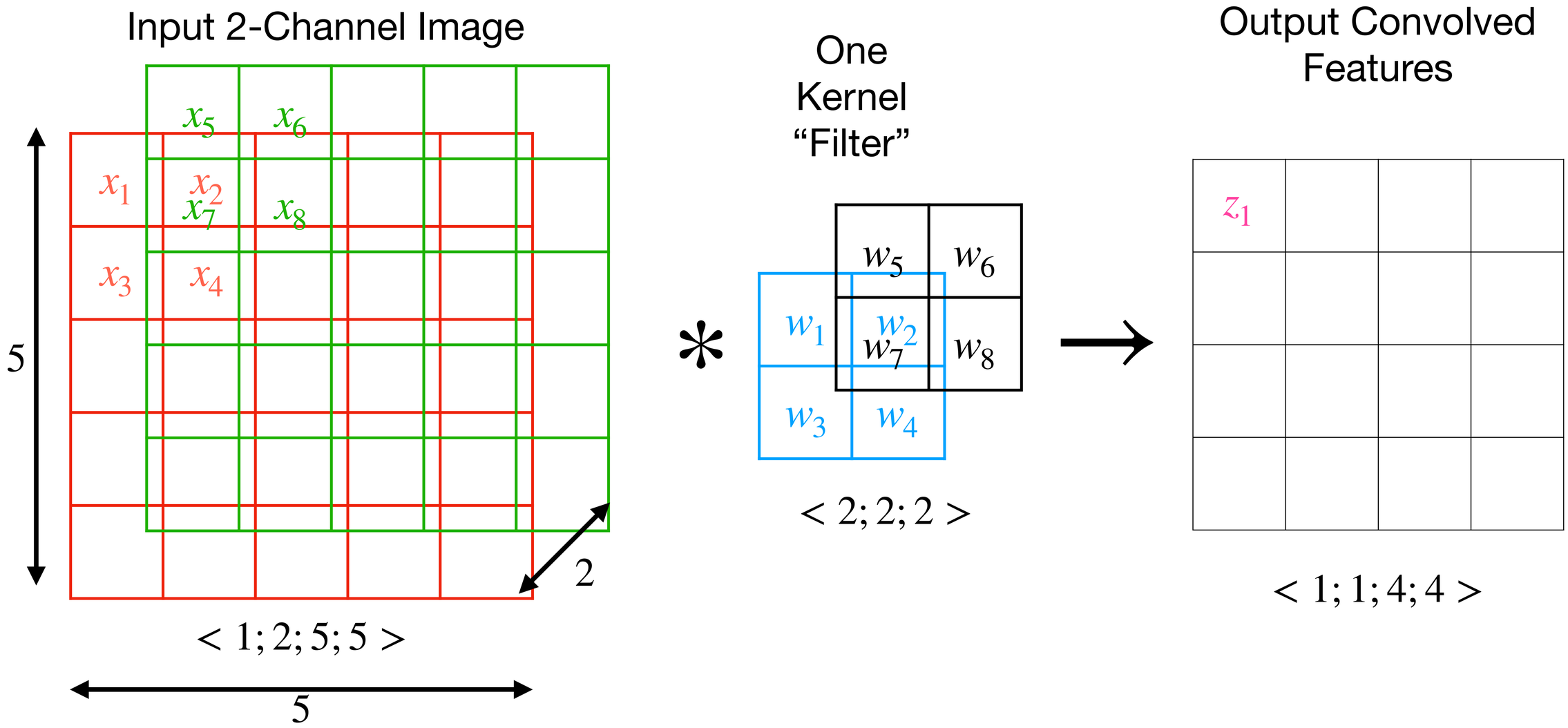
$$(f * g)[x, y] := \sum_{i=-1}^{1} \sum_{j=-1}^{1} f[x+i, y+j] g[i, j]$$



$$z_1 = w_1 x_1 + w_2 x_2 + \ldots + w_9 x_9 + b$$

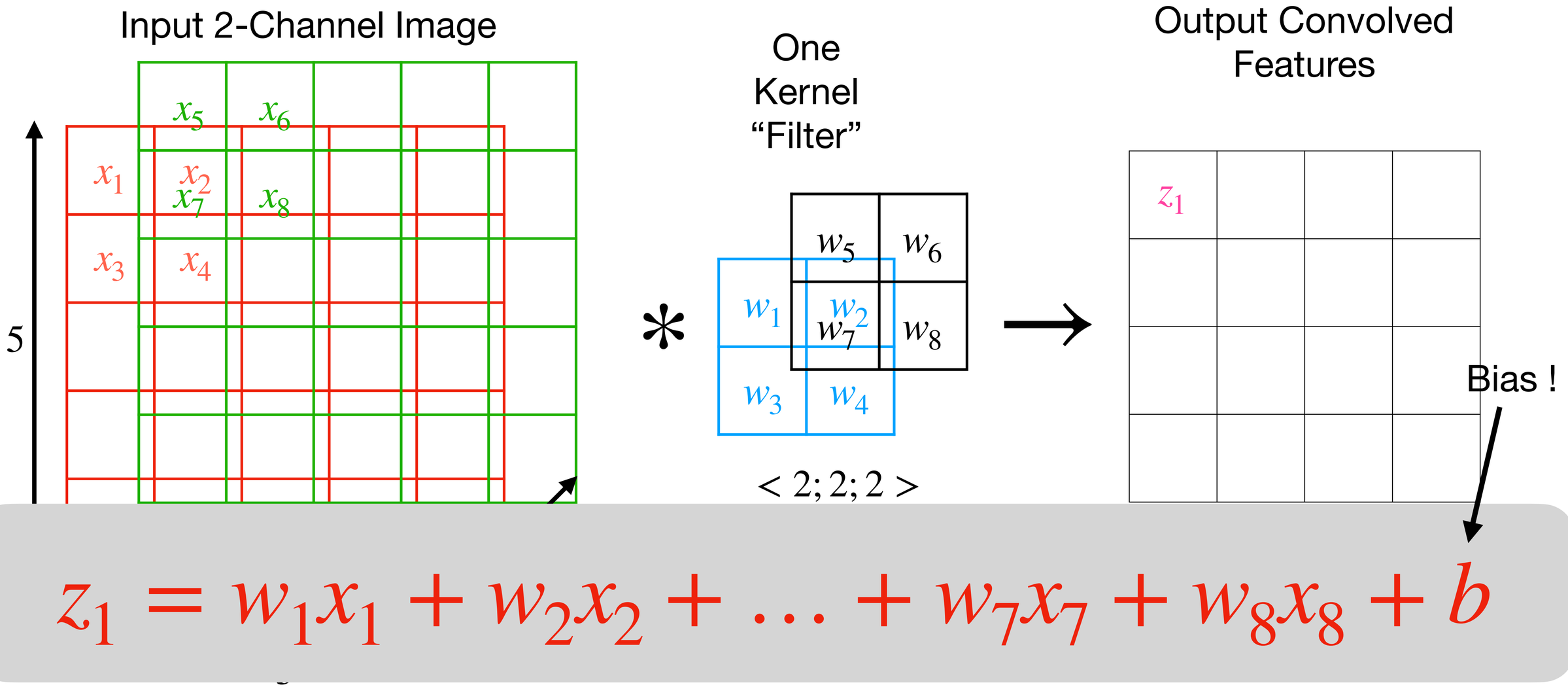
Convolutional Layer

Example



Convolutional Layer

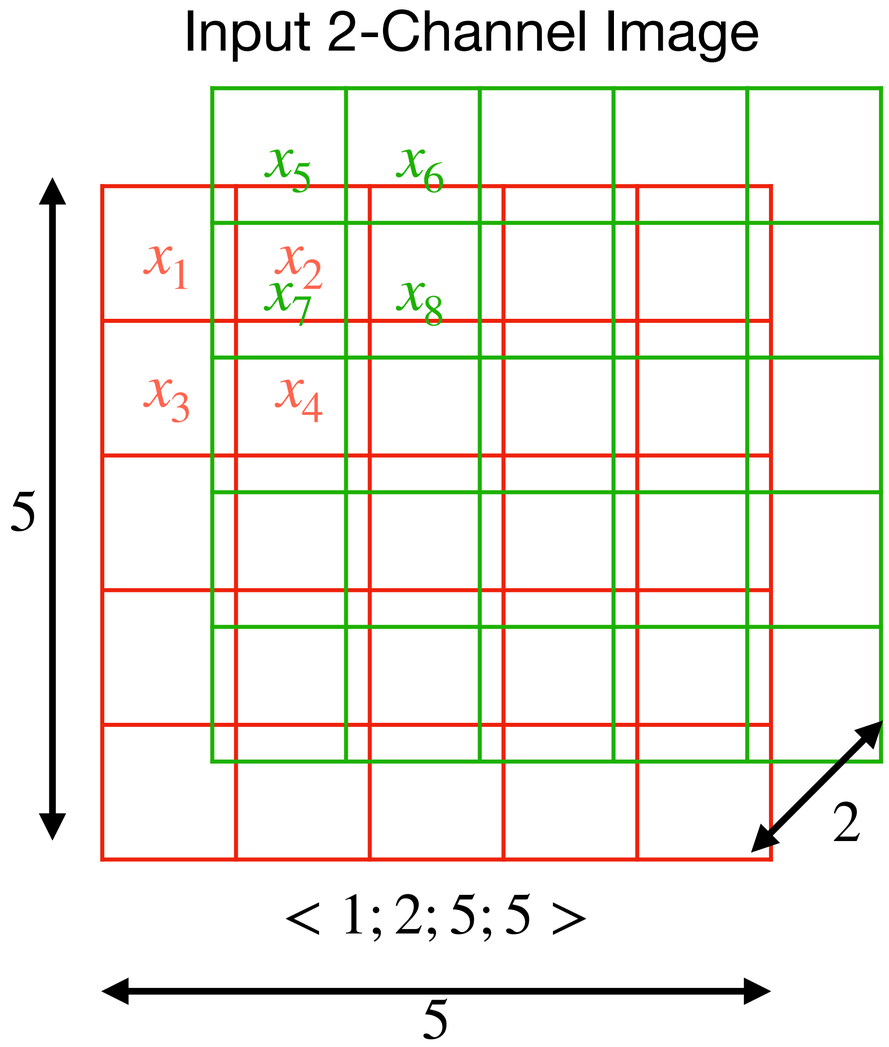
Example



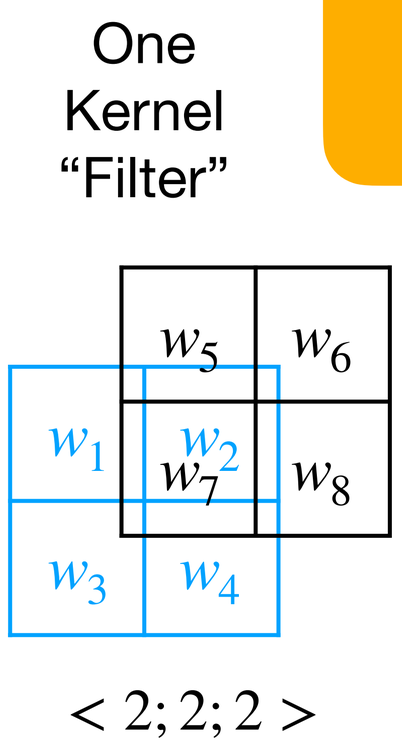
$$z_1 = w_1x_1 + w_2x_2 + \ldots + w_7x_7 + w_8x_8 + b$$

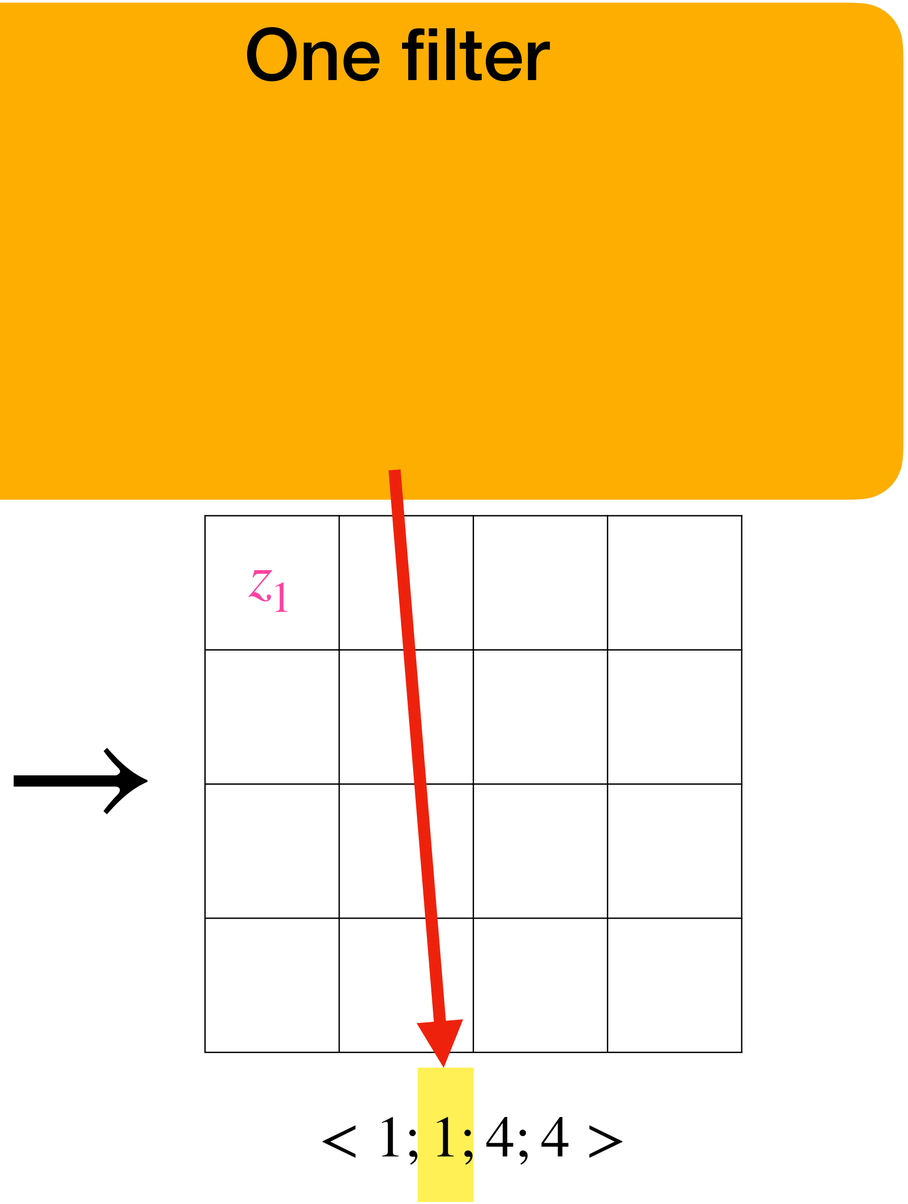
Convolutional Layer

Example



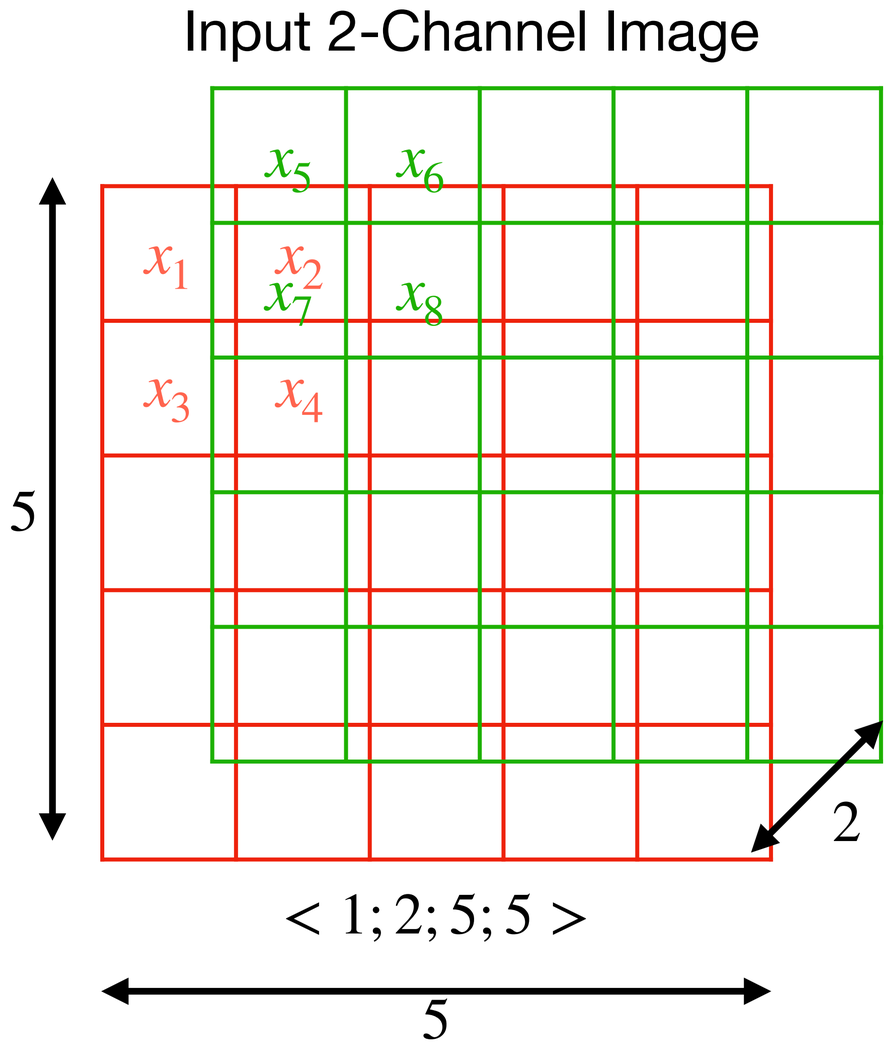
$*$



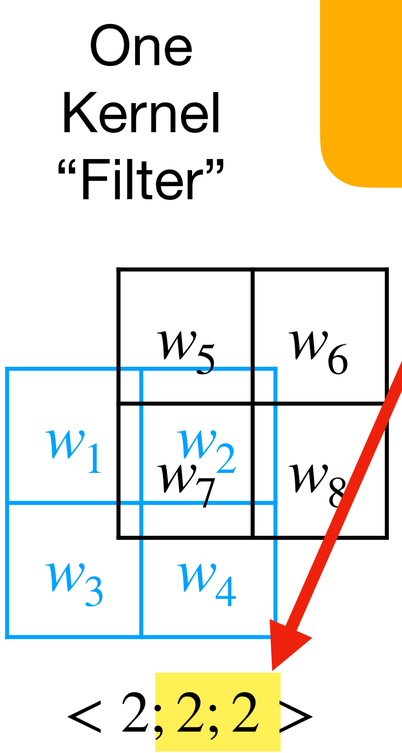


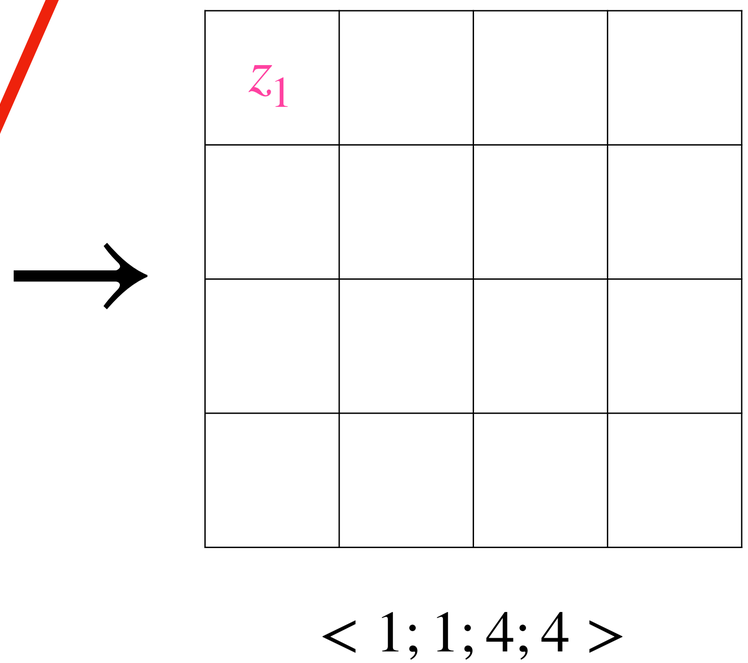
Convolutional Layer

Example



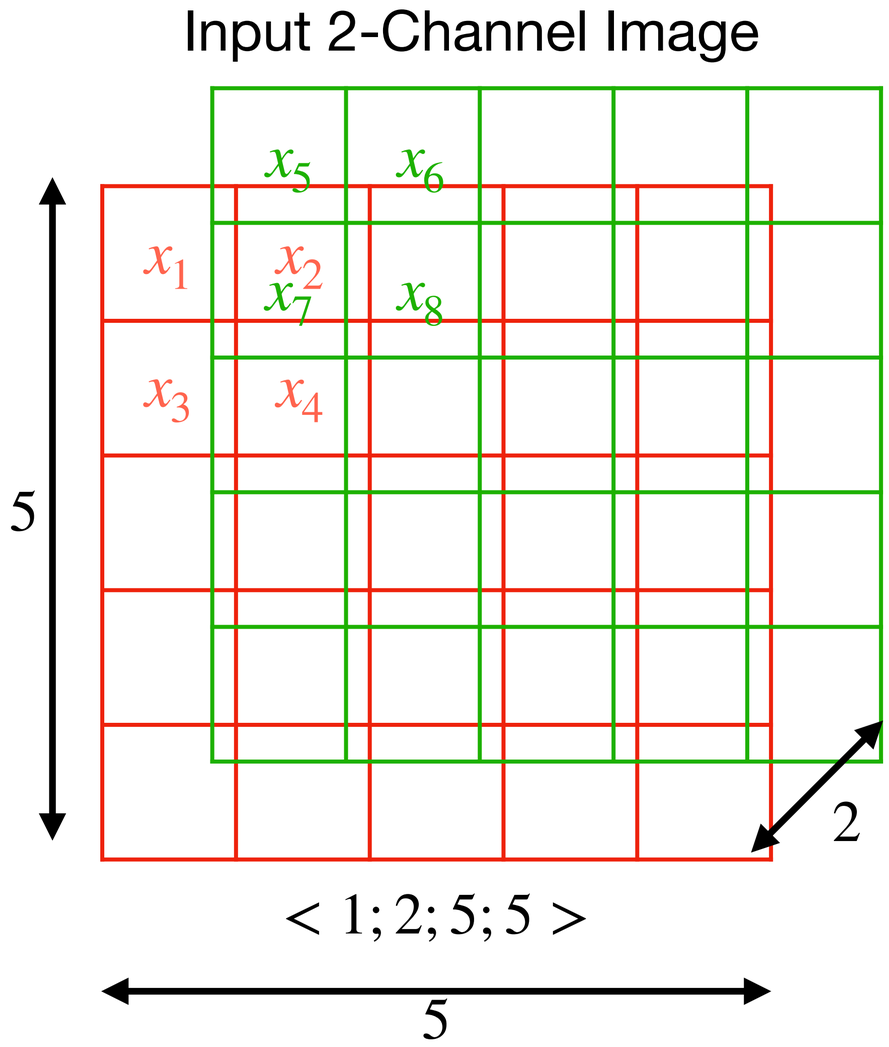
$*$



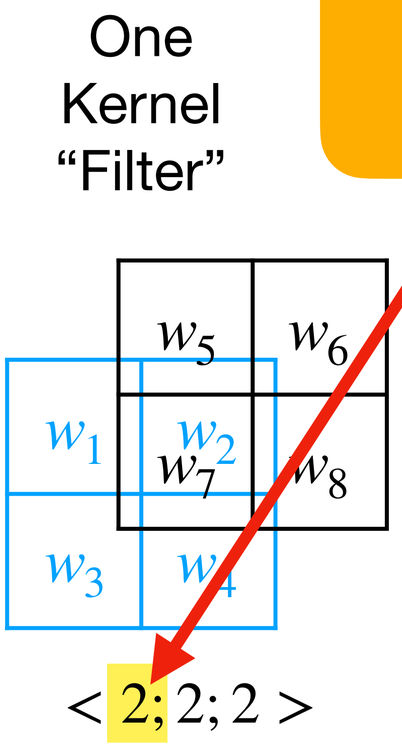


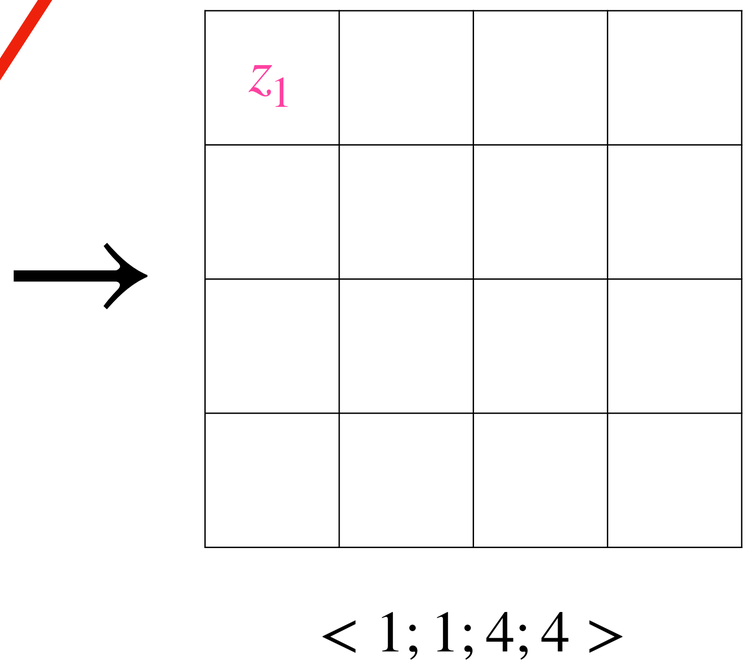
Convolutional Layer

Example



$*$

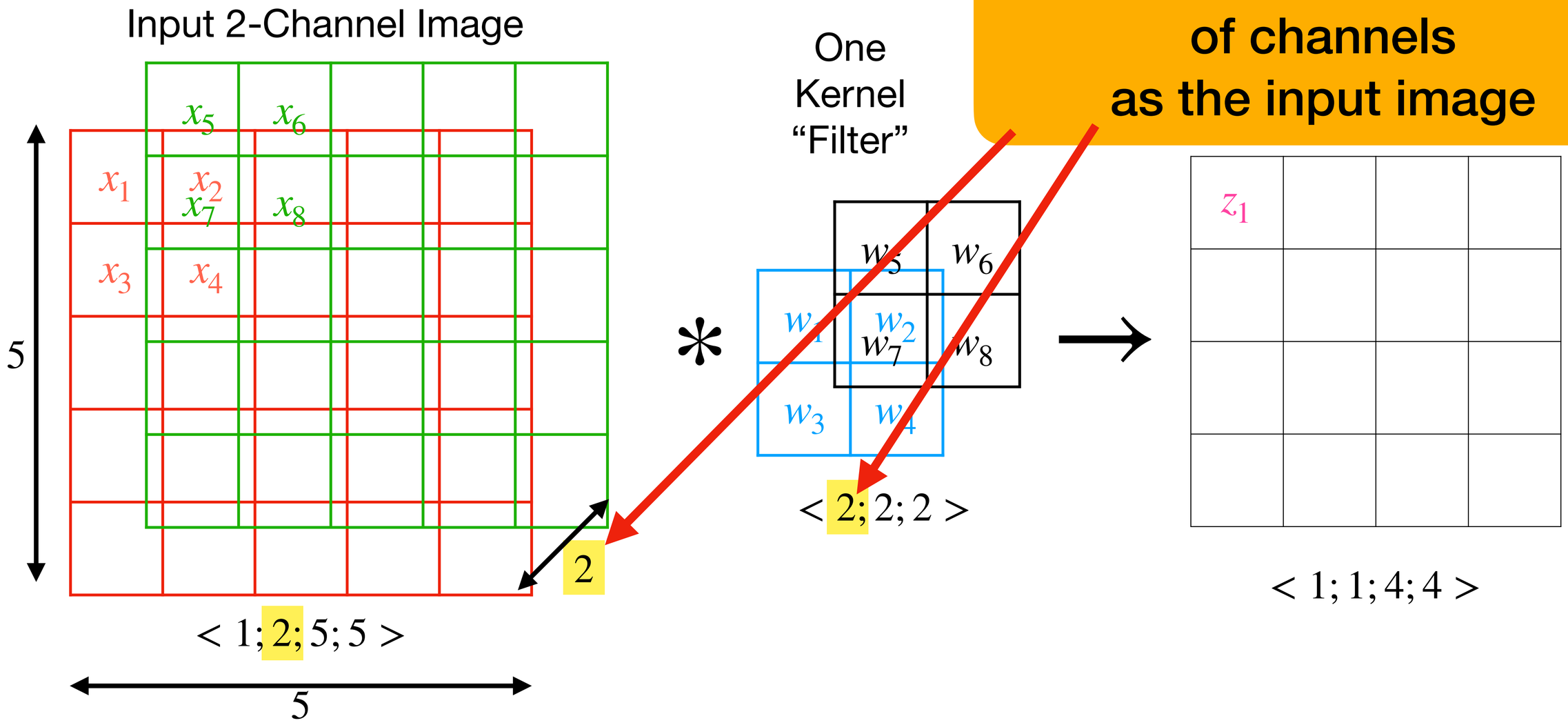




One filter
with kernel size 2×2
must have the same number
of channels

Convolutional Layer

Example



One filter
with kernel size 2×2
must have the same number
of channels
as the input image

Convolutional Layer

- Convolution is a linear operator !
    - We need non-linear activations to capture complex patterns

Convolutional Layer

- Convolution is a linear operator !
    - We need non-linear activations to capture complex patterns
- **Parameters = Kernel matrices (tensors)**
    - One for each output channel (filter)

Convolutional Layer

- Convolution is a linear operator !
    - We need non-linear activations to capture complex patterns
- **Parameters = Kernel matrices (tensors)**
    - One for each output channel (filter)
- Total number of parameters does not depend on input spatial resolution
    - Can be applied to images of **any size**

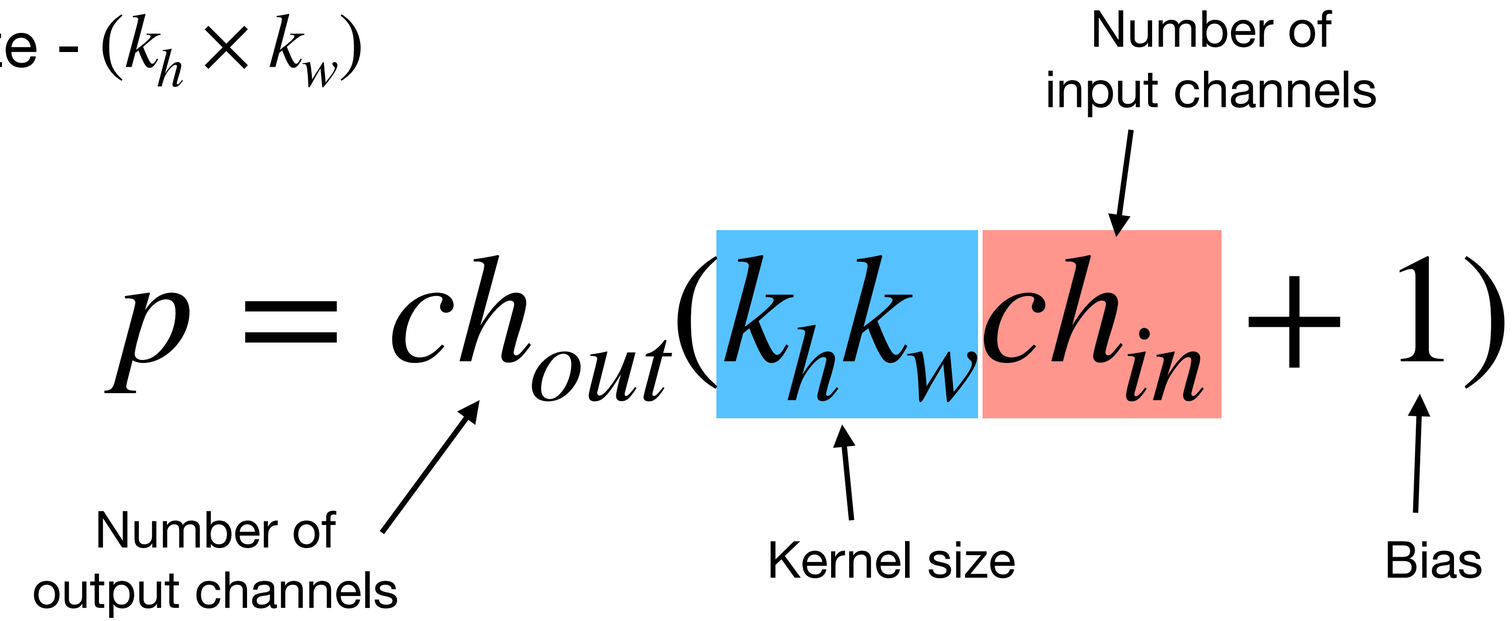
Convolutional Layer

- Convolution is a linear operator !
    - We need non-linear activations to capture complex patterns
- **Parameters = Kernel matrices (tensors)**
    - One for each output channel (filter)
- Total number of parameters does not depend on input spatial resolution
    - Can be applied to images of **any size**
- **Translation invariant**

Convolutional Layer

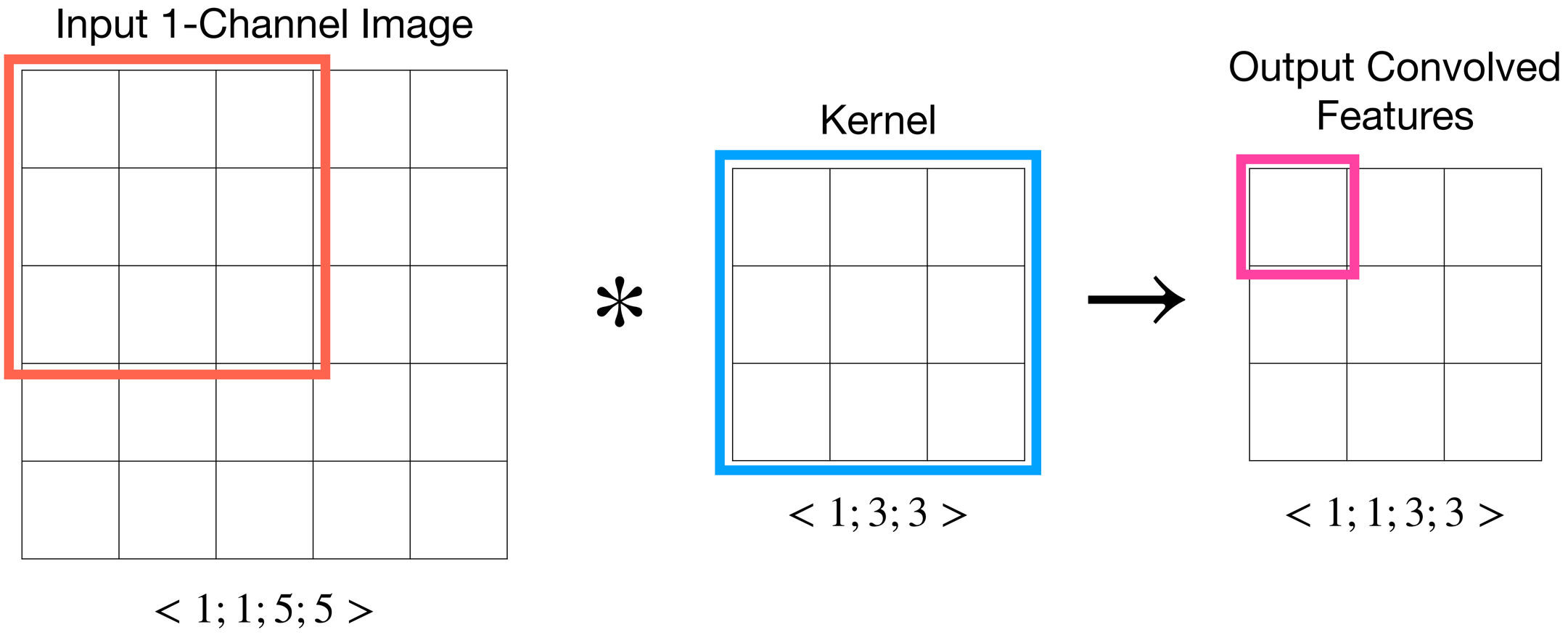
Number of Parameters

- Number of input channels - ch_{in}
- Number of output channels (filters) - ch_{out}
- Kernel size - $(k_h \times k_w)$



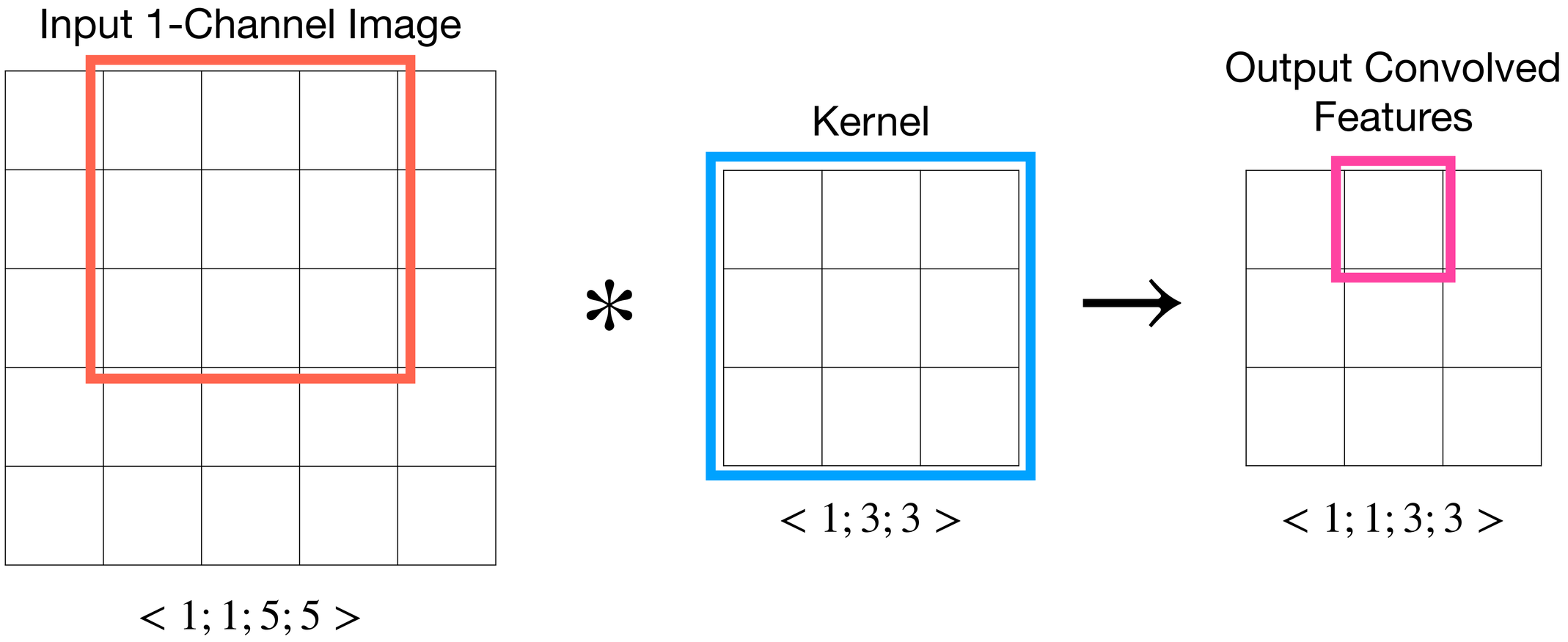
Convolutional Layer

Input & Output Shapes



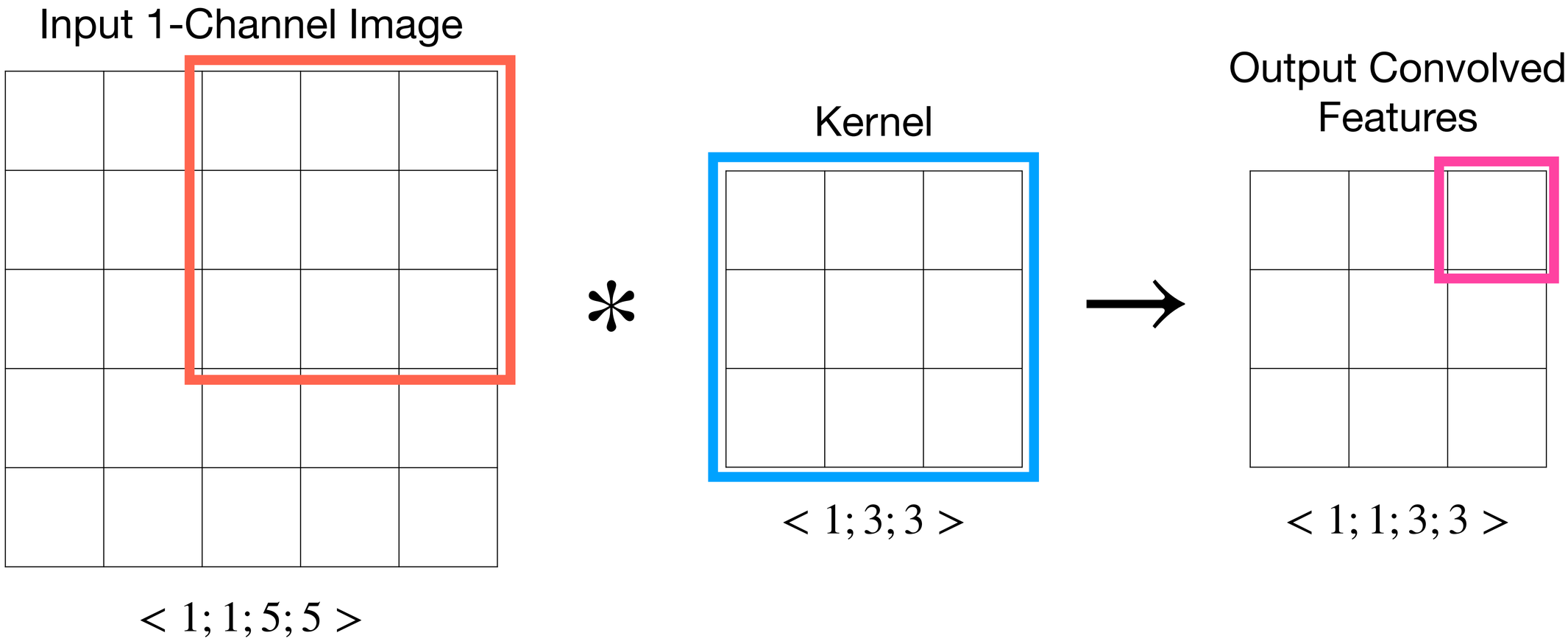
Convolutional Layer

Input & Output Shapes



Convolutional Layer

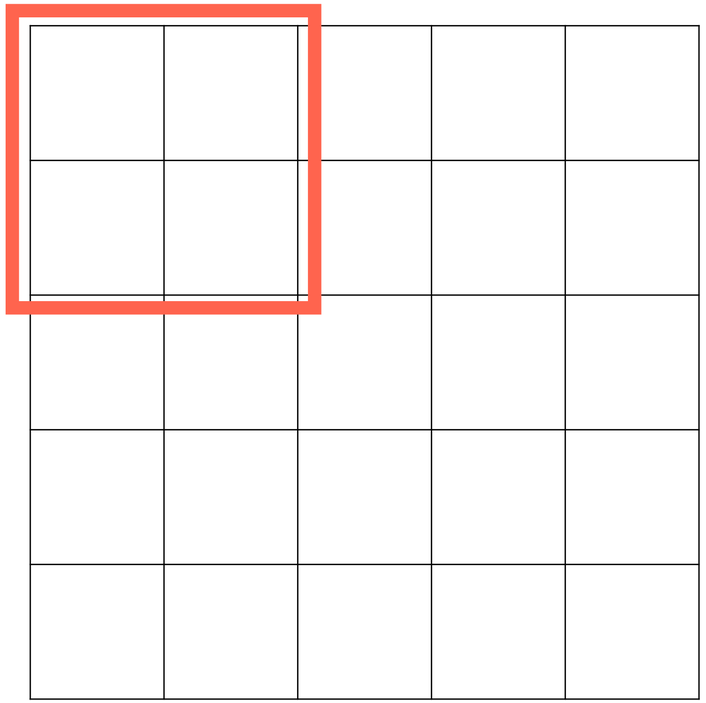
Input & Output Shapes



Convolutional Layer

Input & Output Shapes

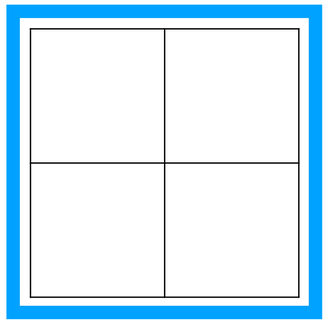
Input 1-Channel Image



$< 1; 1; 5; 5 >$

$*$

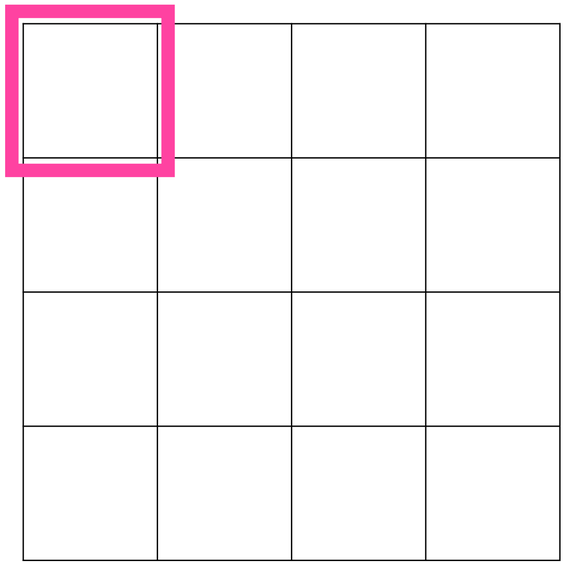
Kernel



$< 1; 2; 2 >$

$\rightarrow$

Output Convolved Features

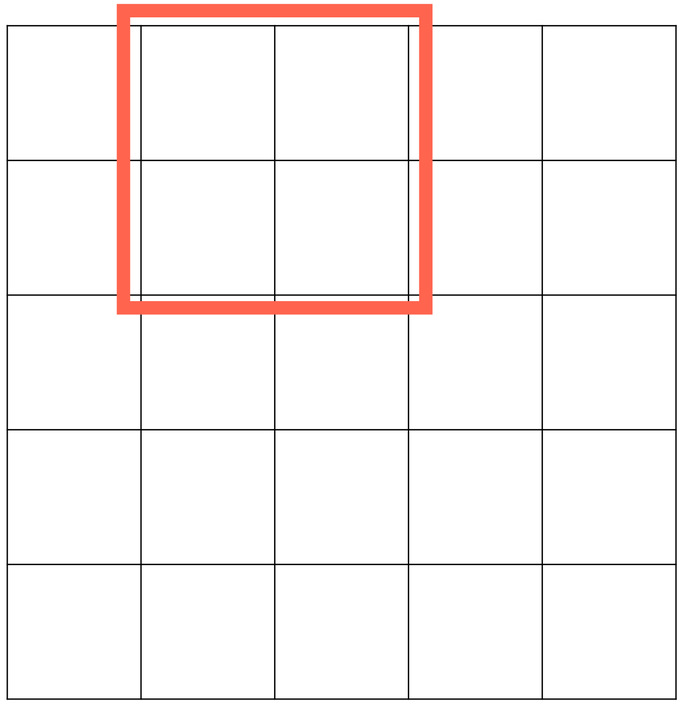


$< 1; 1; 4; 4 >$

Convolutional Layer

Input & Output Shapes

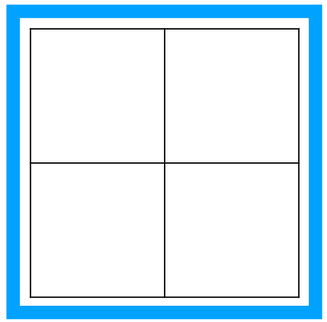
Input 1-Channel Image



$< 1; 1; 5; 5 >$

$*$

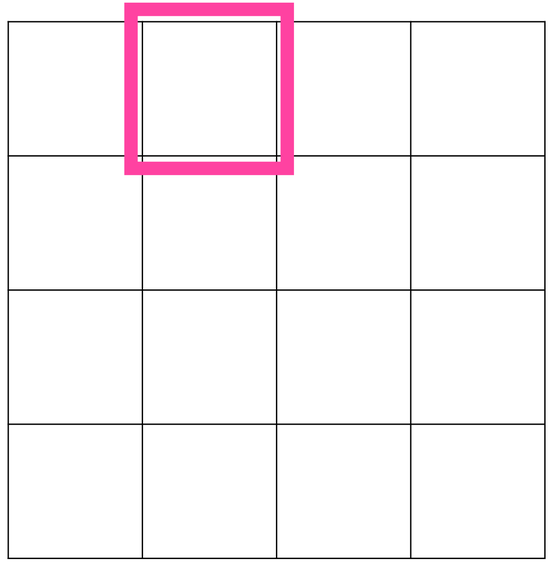
Kernel



$< 1; 2; 2 >$

$\rightarrow$

Output Convolved Features

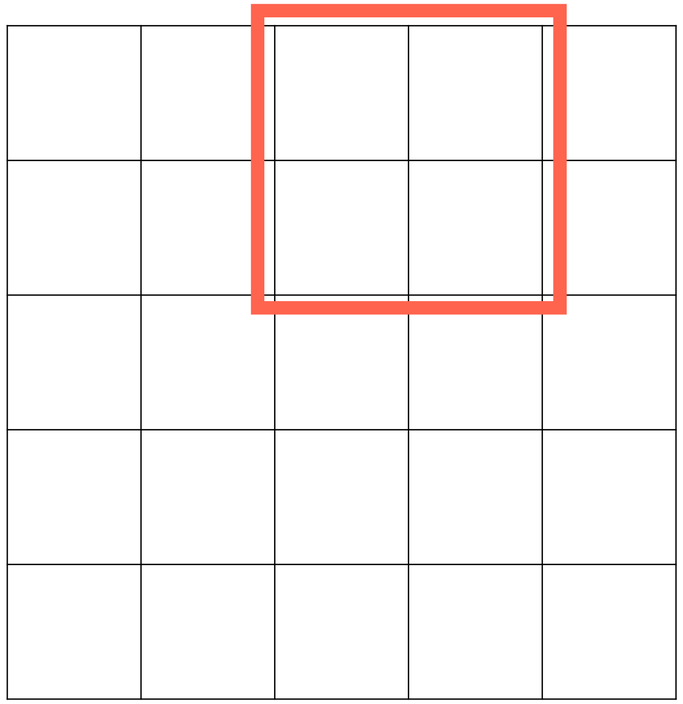


$< 1; 1; 4; 4 >$

Convolutional Layer

Input & Output Shapes

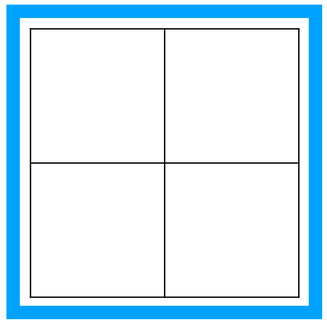
Input 1-Channel Image



$< 1; 1; 5; 5 >$

$*$

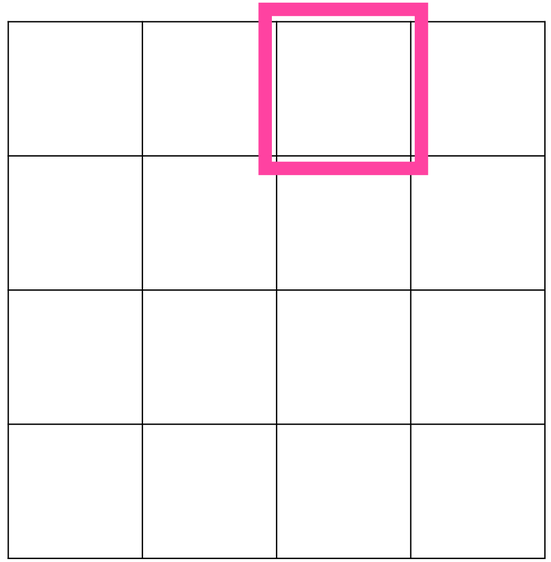
Kernel



$< 1; 2; 2 >$

$\rightarrow$

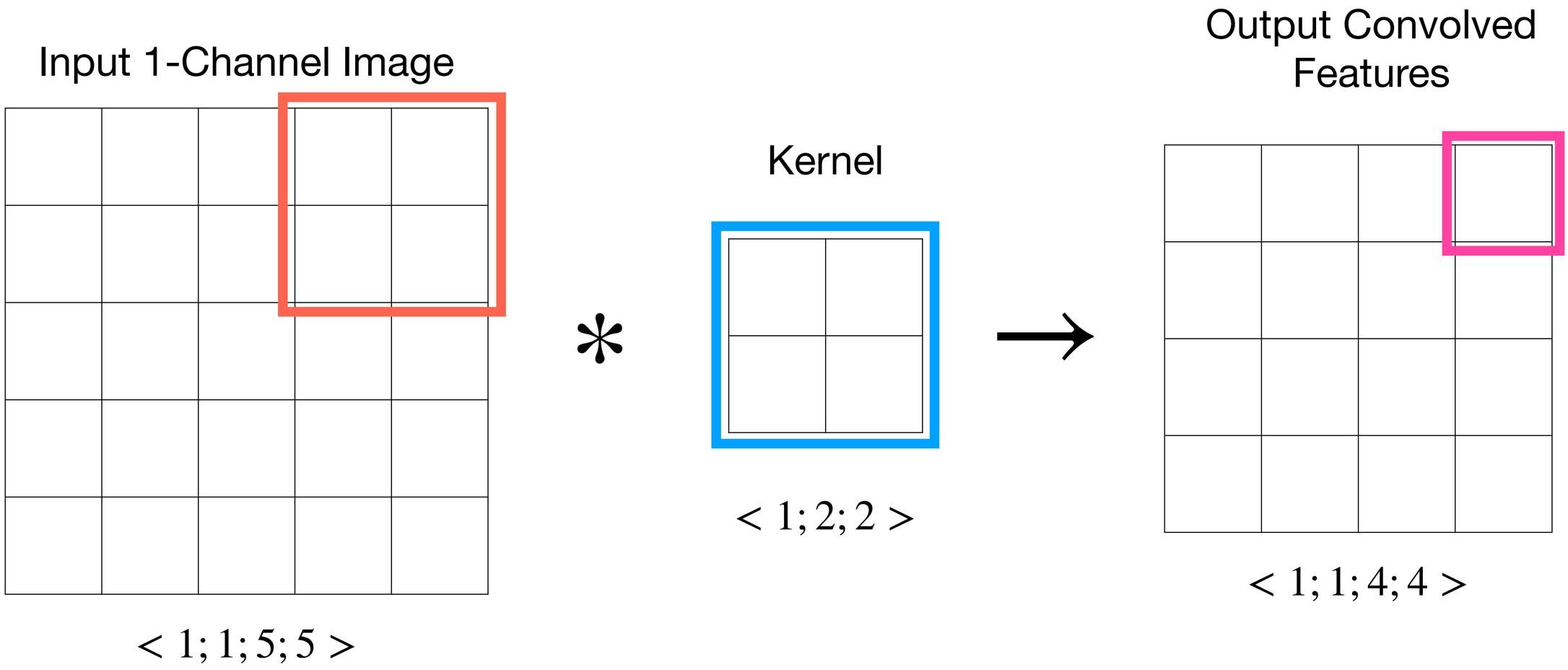
Output Convolved Features



$< 1; 1; 4; 4 >$

Convolutional Layer

Input & Output Shapes



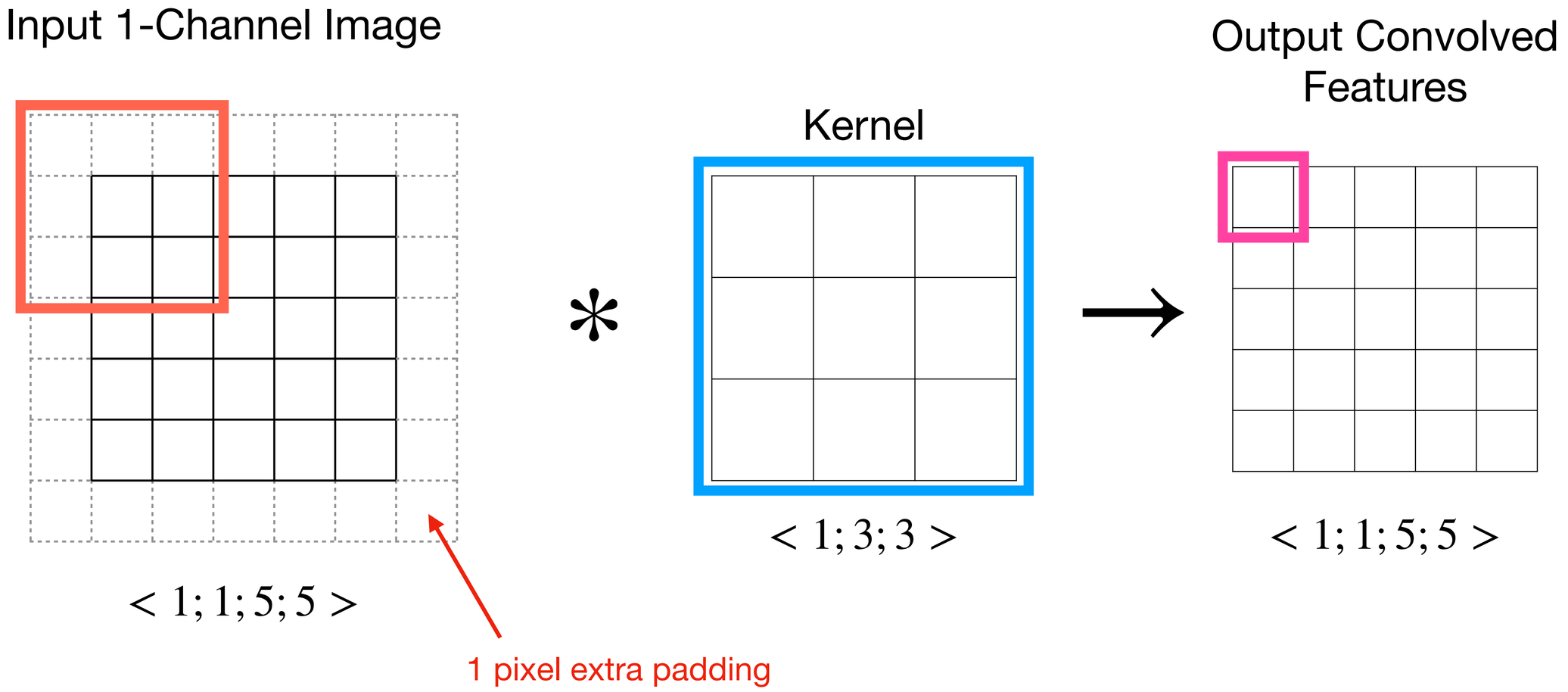
Convolutional Layer

Padding

- Add extra information to make convolution **preserve the original shape**
    - Zeros
    - Mirror
    - Constant
    - Copy edge pixels
    - …

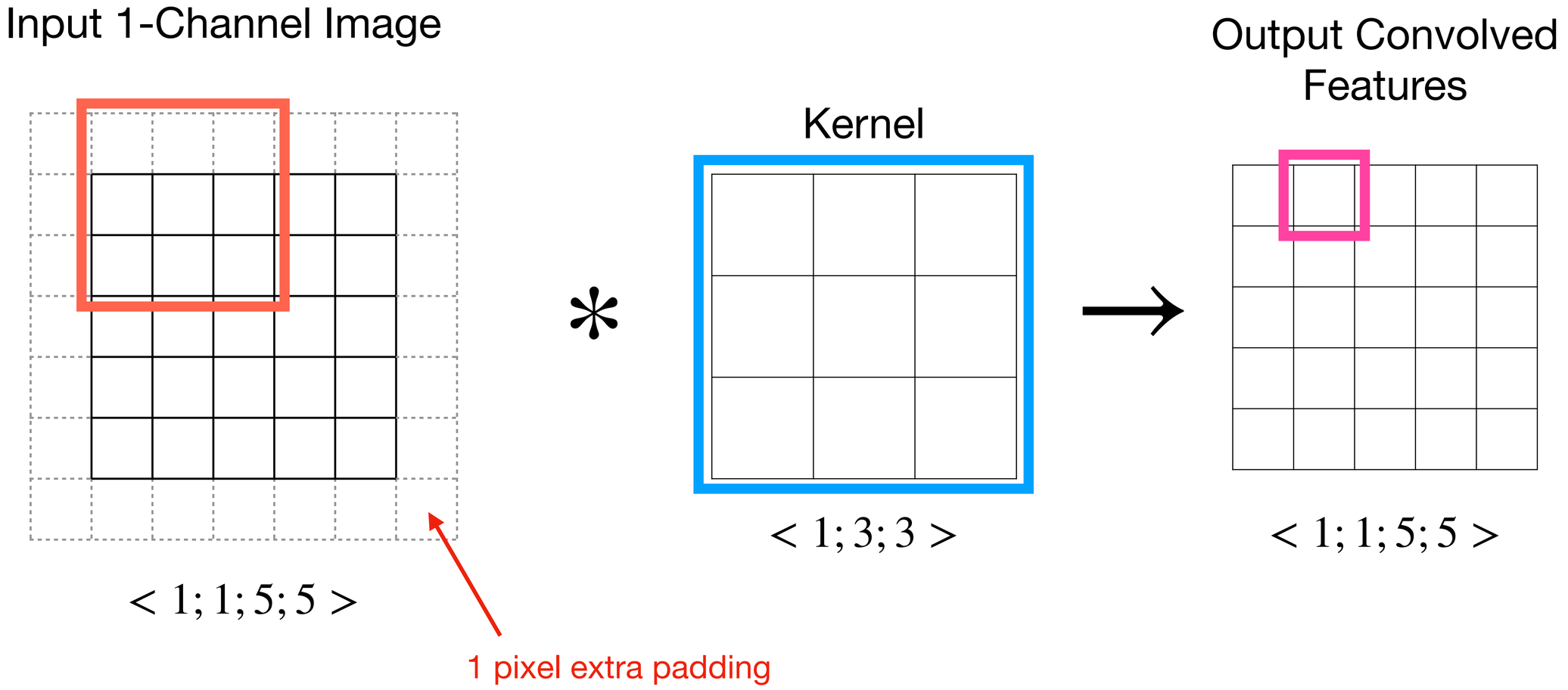
Convolutional Layer

Input & Output Shapes - Padding



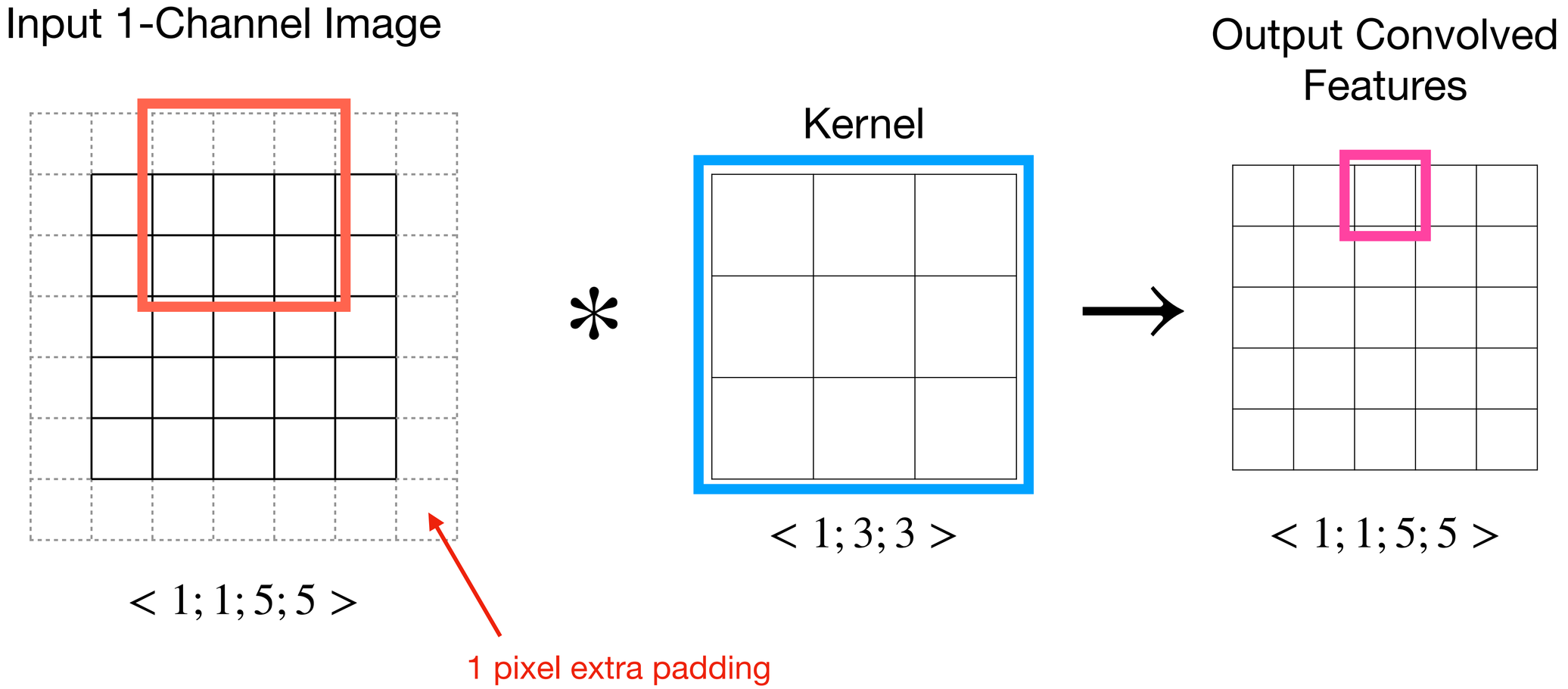
Convolutional Layer

Input & Output Shapes - Padding



Convolutional Layer

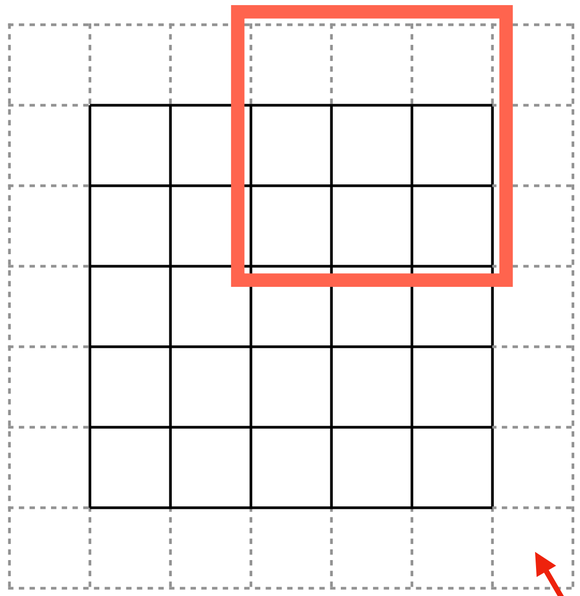
Input & Output Shapes - Padding



Convolutional Layer

Input & Output Shapes - Padding

Input 1-Channel Image

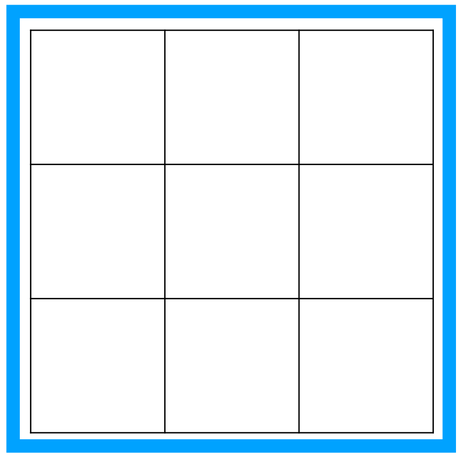


$< 1; 1; 5; 5 >$

1 pixel extra padding

$*$

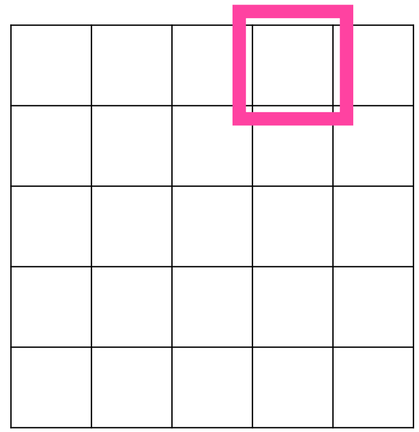
Kernel



$< 1; 3; 3 >$

$\rightarrow$

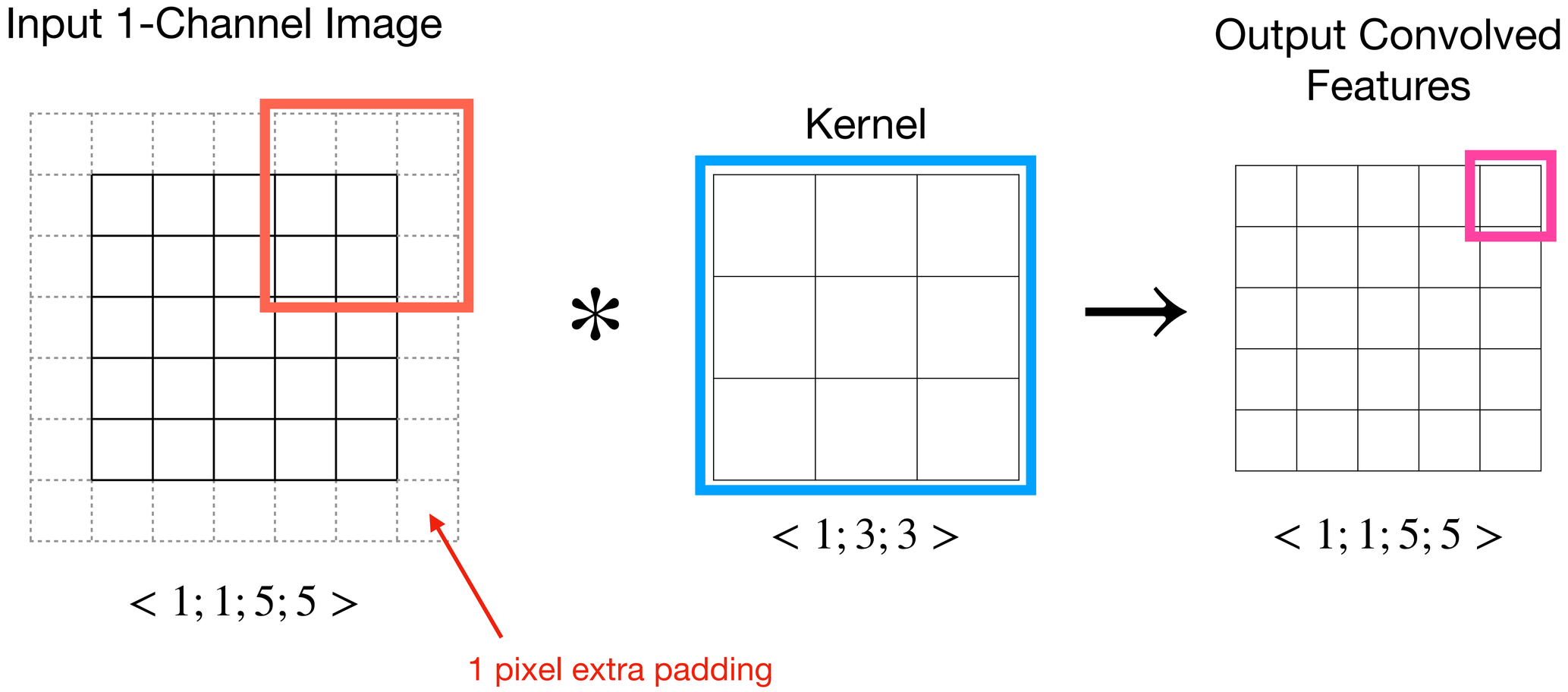
Output Convolved Features



$< 1; 1; 5; 5 >$

Convolutional Layer

Input & Output Shapes - Padding



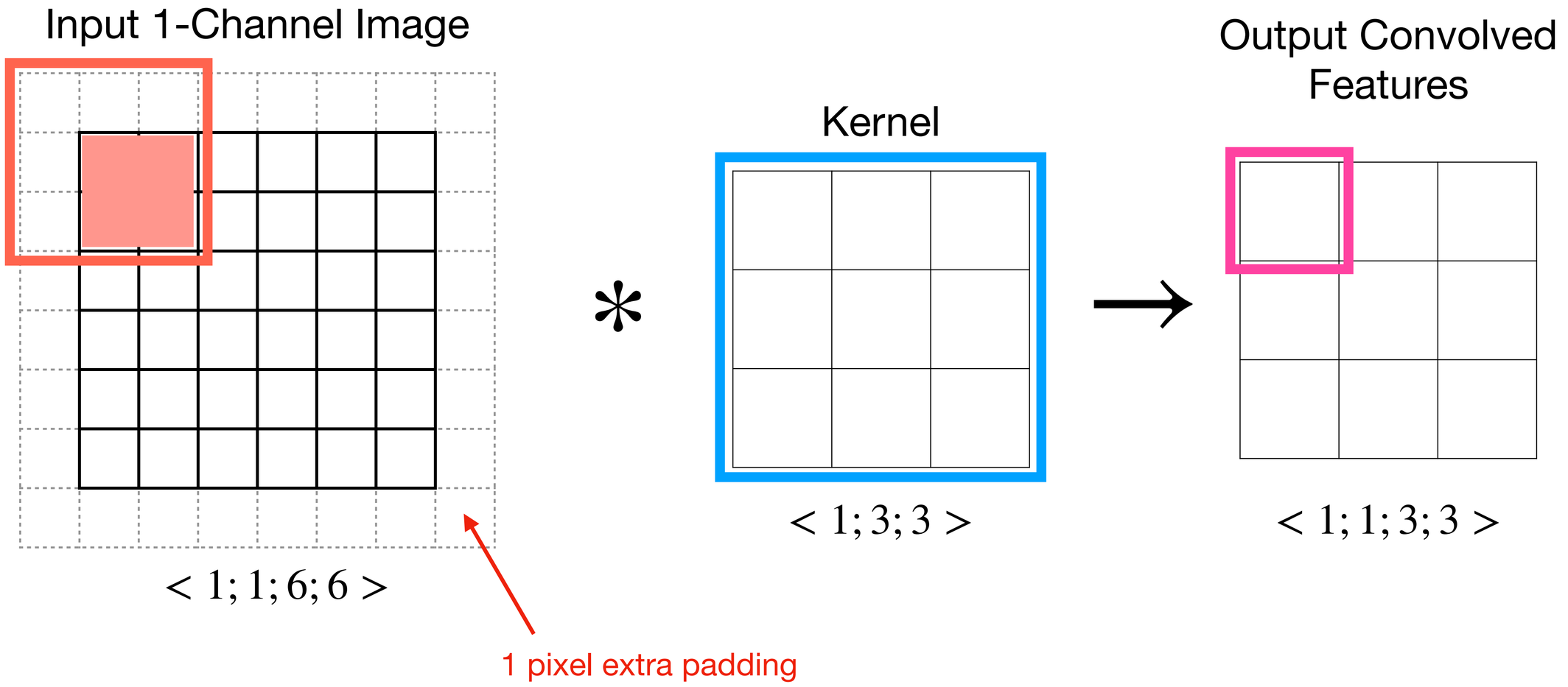
Convolutional Layer

Stride

- Convolution will produce tensors with **reduced spatial resolution**
- Alternative to pooling layers
    - “Smart” downsampling

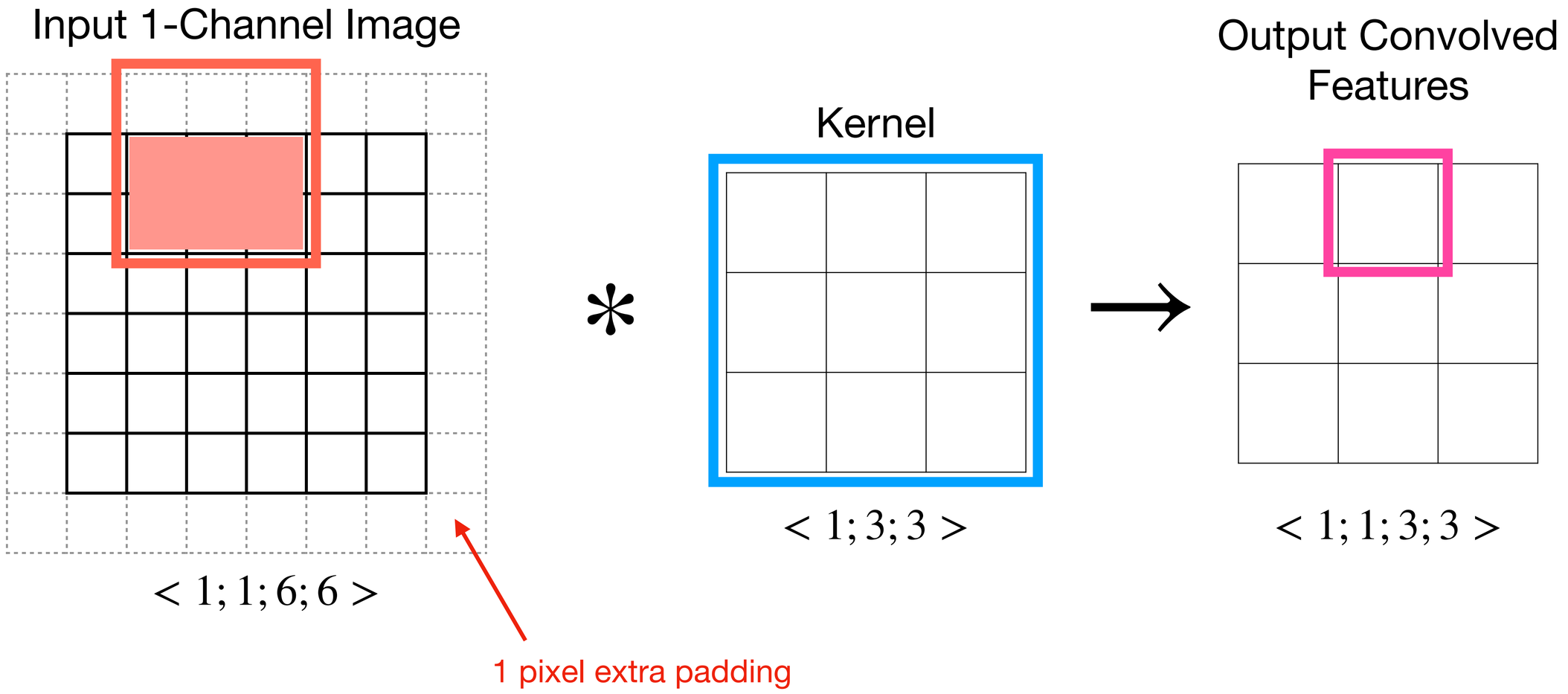
Convolutional Layer

Input & Output Shapes - Stride & Padding



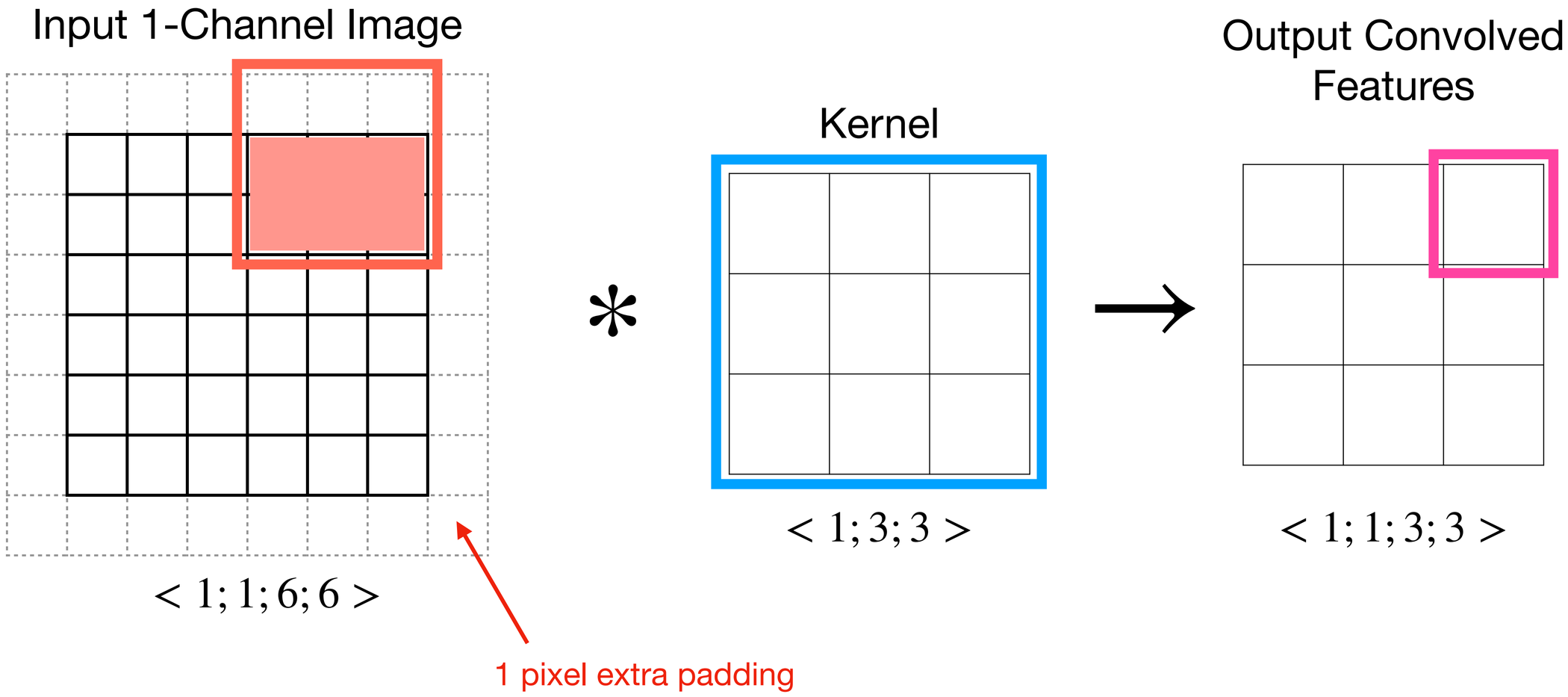
Convolutional Layer

Input & Output Shapes - Stride & Padding



Convolutional Layer

Input & Output Shapes - Stride & Padding



Convolutional Layer

Input & Output Shapes

- Input spatial resolution - h_{in}, w_{in}
- Kernel size - $k_h \times k_w$
- Stride - s_h, s_w
- Padding - p_h, p_w
- Output spatial resolution - h_{out}, w_{out}

$$h_{out} = \left\lfloor \frac{h_{in} + 2p_h - (k_h - 1) - 1}{s_h} + 1 \right\rfloor$$

$$w_{out} = \left\lfloor \frac{w_{in} + 2p_w - (k_w - 1) - 1}{s_w} + 1 \right\rfloor$$

Convolutional Layer

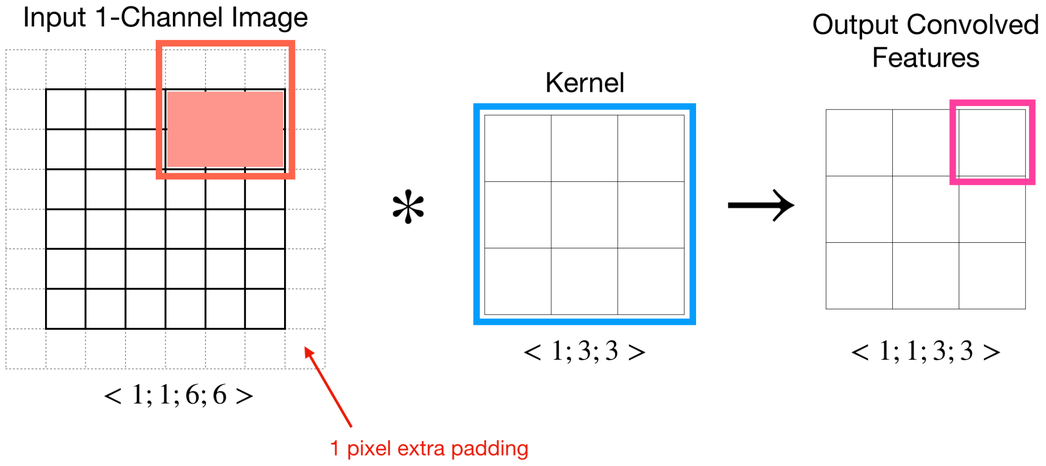
Input & Output Shapes

- Input spatial resolution - h_{in}, w_{in}
- Kernel size - $k_h \times k_w$
- Stride - s_h, s_w
- Padding - p_h, p_w
- Output spatial resolution - h_{out}, w_{out}

$$h_{out} = \left\lfloor \frac{h_{in} + 2p_h - (k_h - 1) - 1}{s_h} + 1 \right\rfloor$$

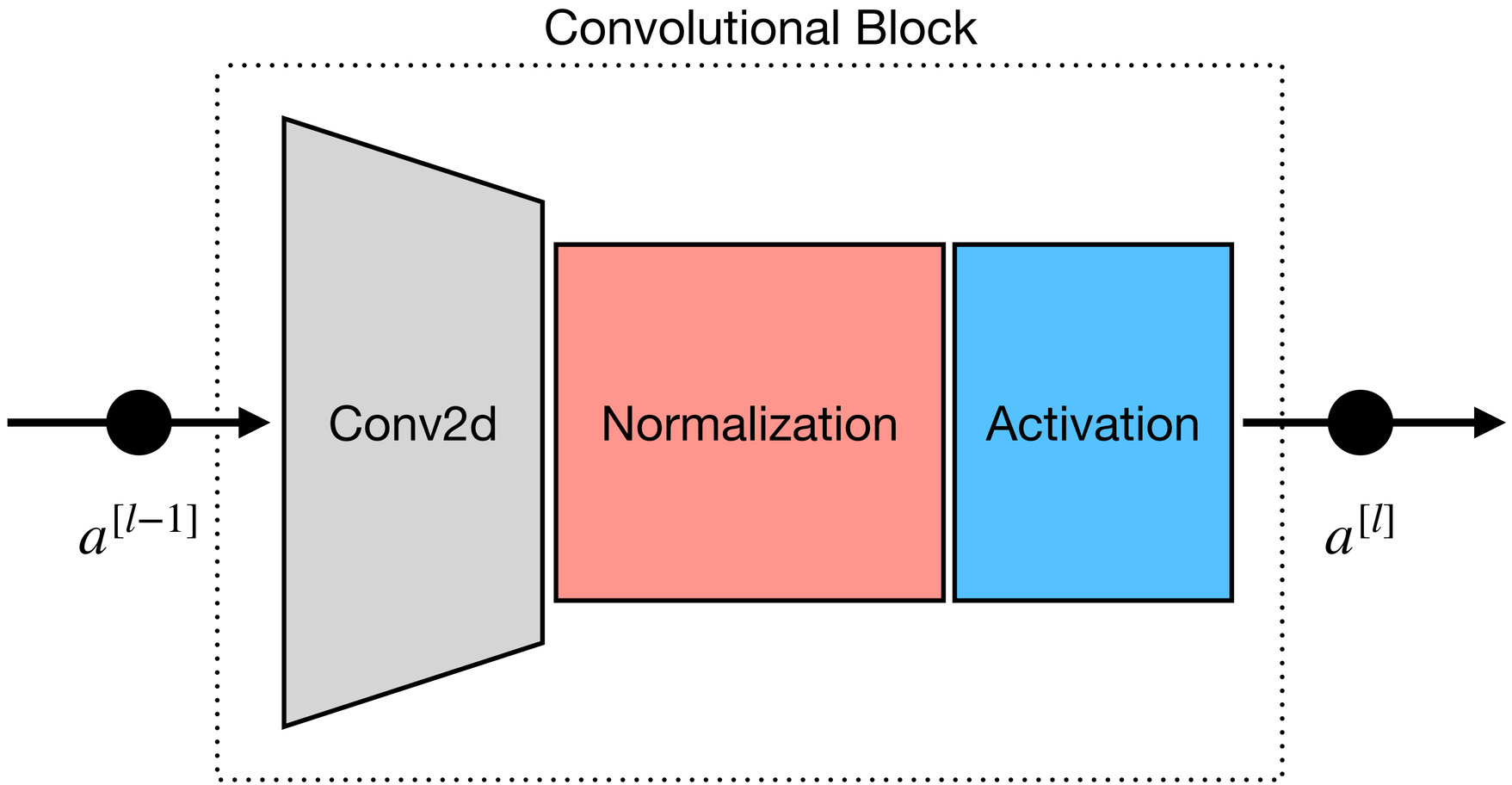
$$w_{out} = \left\lfloor \frac{w_{in} + 2p_w - (k_w - 1) - 1}{s_w} + 1 \right\rfloor$$

$$h_{out} = \left\lfloor \frac{6 + 2 - (3 - 1) - 1}{2} + 1 \right\rfloor = 3$$



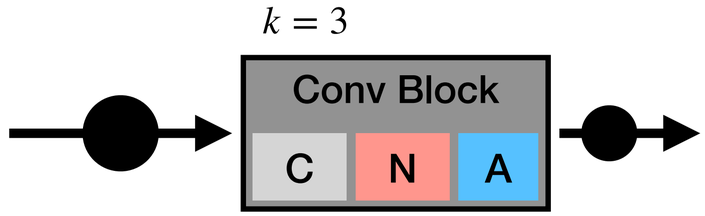
Multiple Convolutional Layers

Convolutional Block



Convolutional Layers

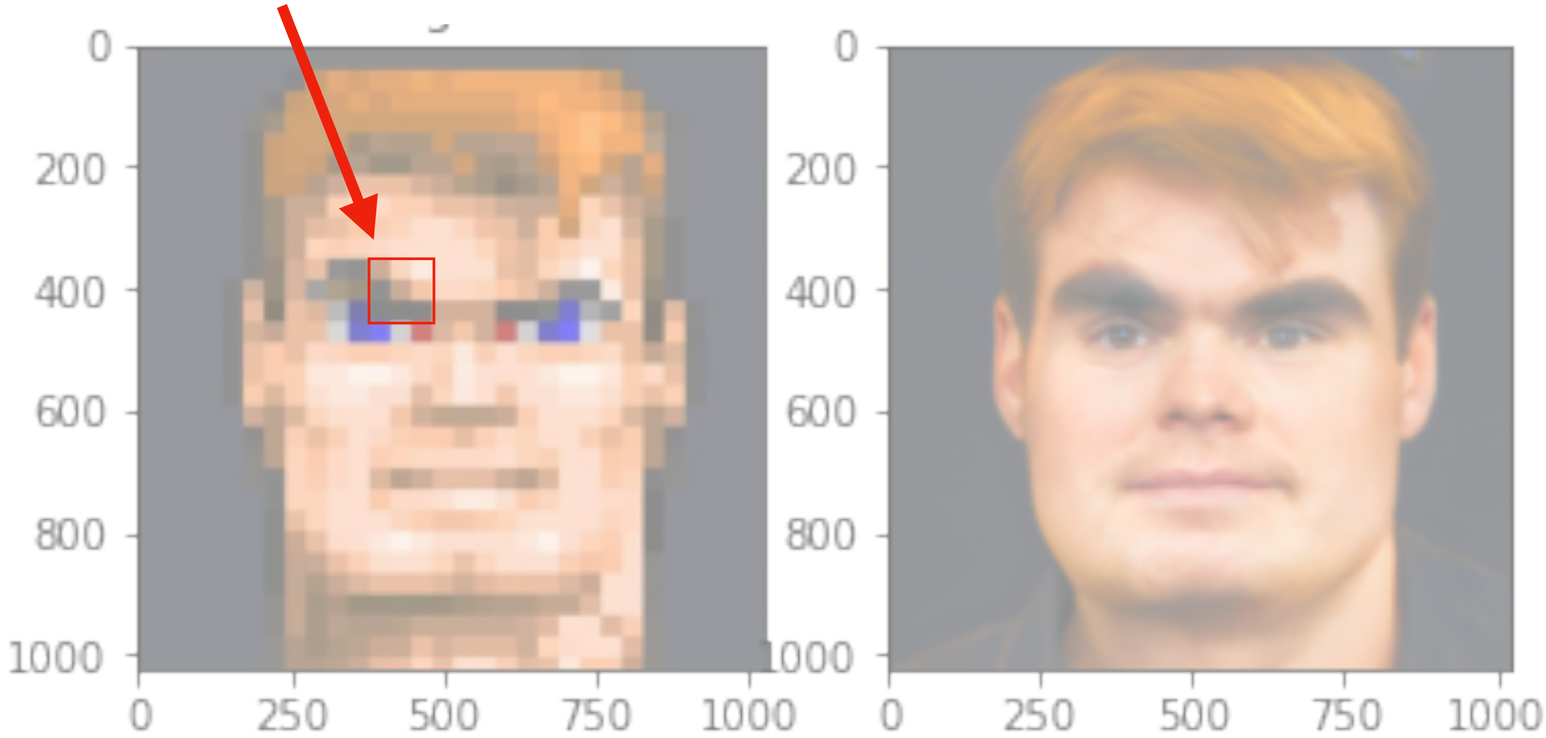
Receptive Field



$< 1; 3; 32; 32 >$ $< 1; ?; 30; 30 >$

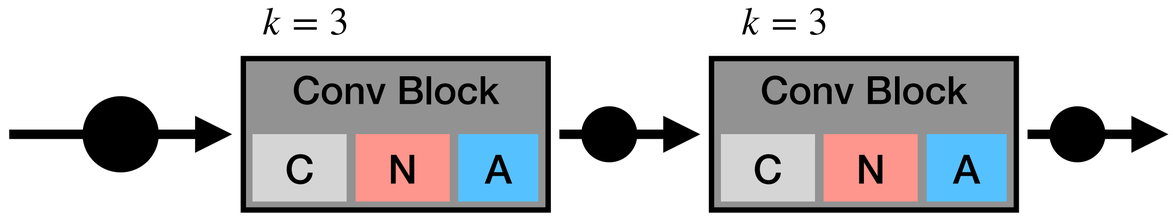
Receptive field

3×3



Convolutional Layers

Receptive Field



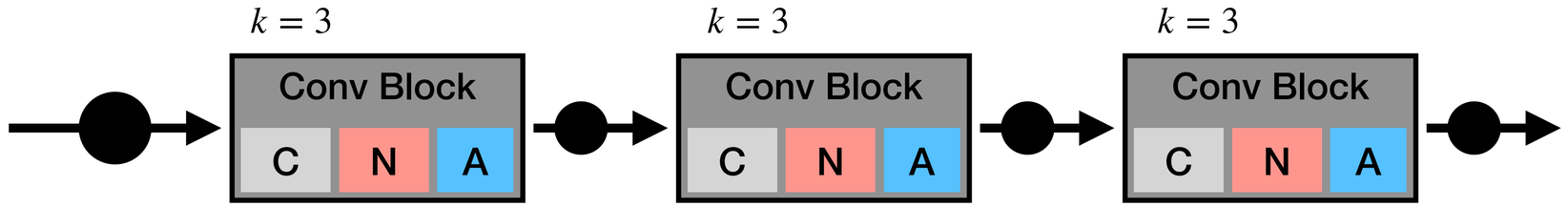
$< 1; 3; 32; 32 >$ $< 1; ?; 30; 30 >$ $< 1; ?; 28; 28 >$

Receptive field

5×5

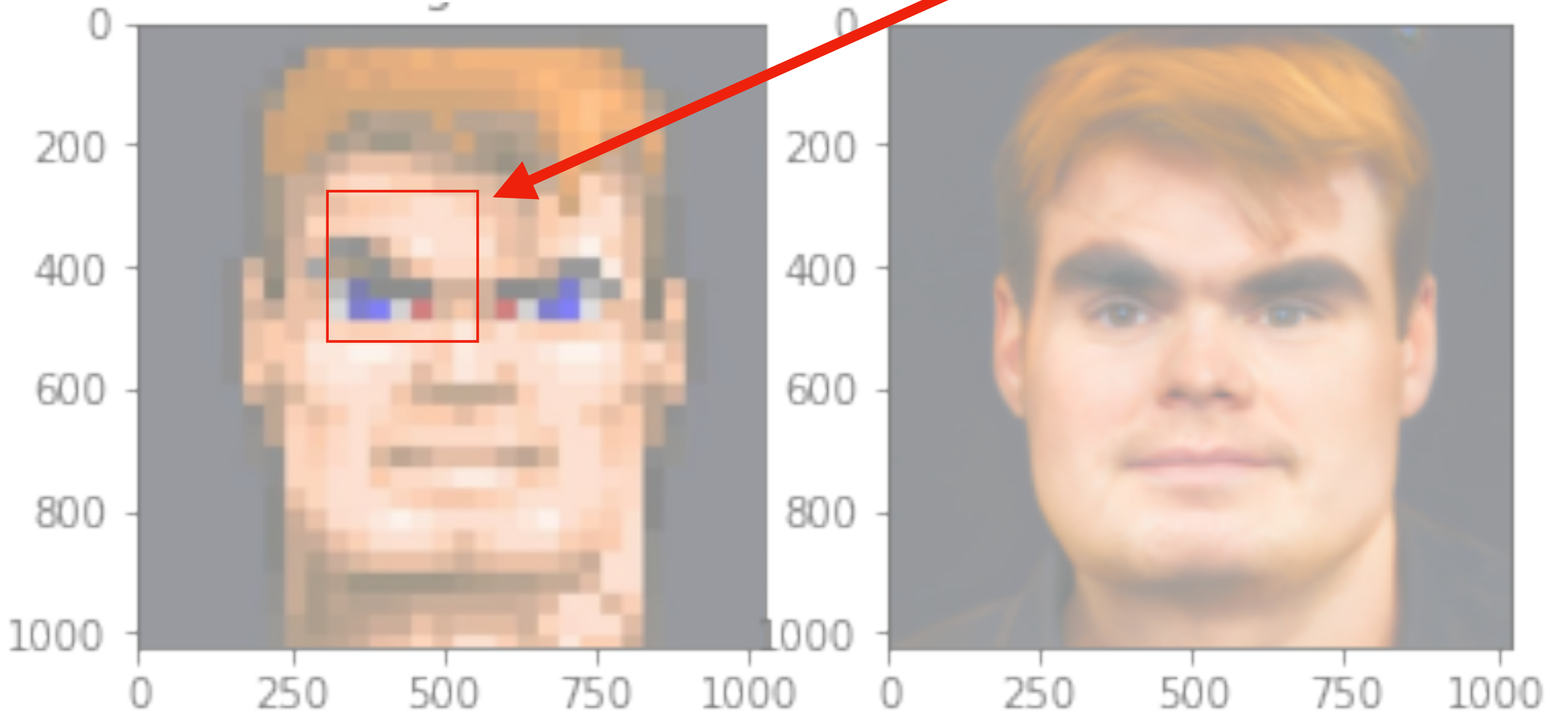
Convolutional Layers

Receptive Field



Receptive field

7×7



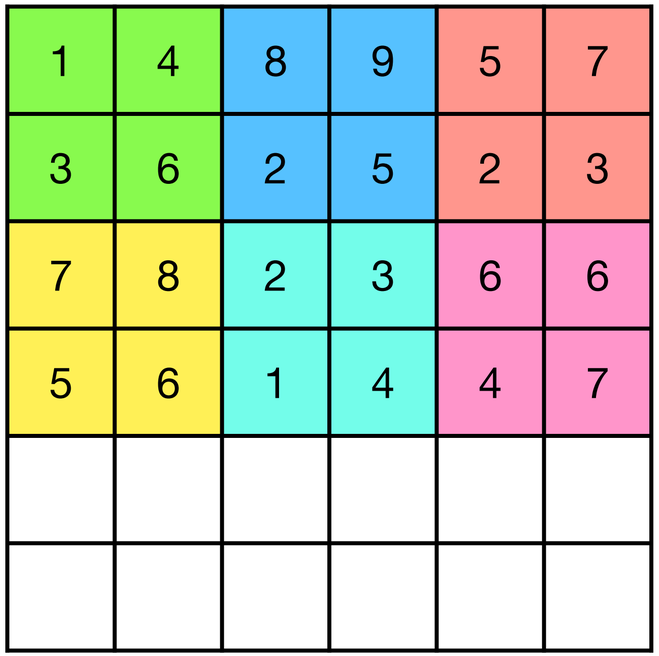
Pooling Layers

- Convolution lowers spatial dimensions of activation maps (without padding)
- Convolution operates on a **small local neighborhood**
    - Multiple blocks are necessary for the receptive field to grow

- **Pooling - allows the receptive field to grow faster**

Pooling Layers

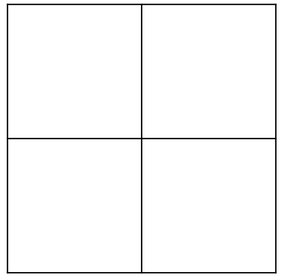
Input n-Channel Image



$< 1; n; 6; 6 >$

Pool operation

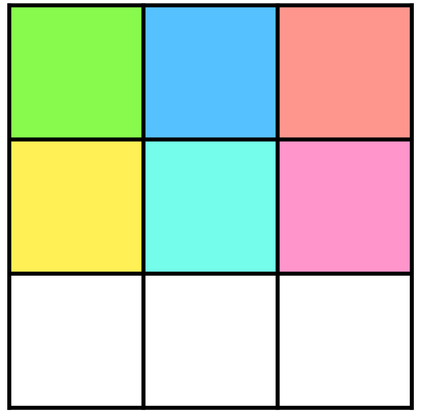
Kernel



$< 2; 2 >$

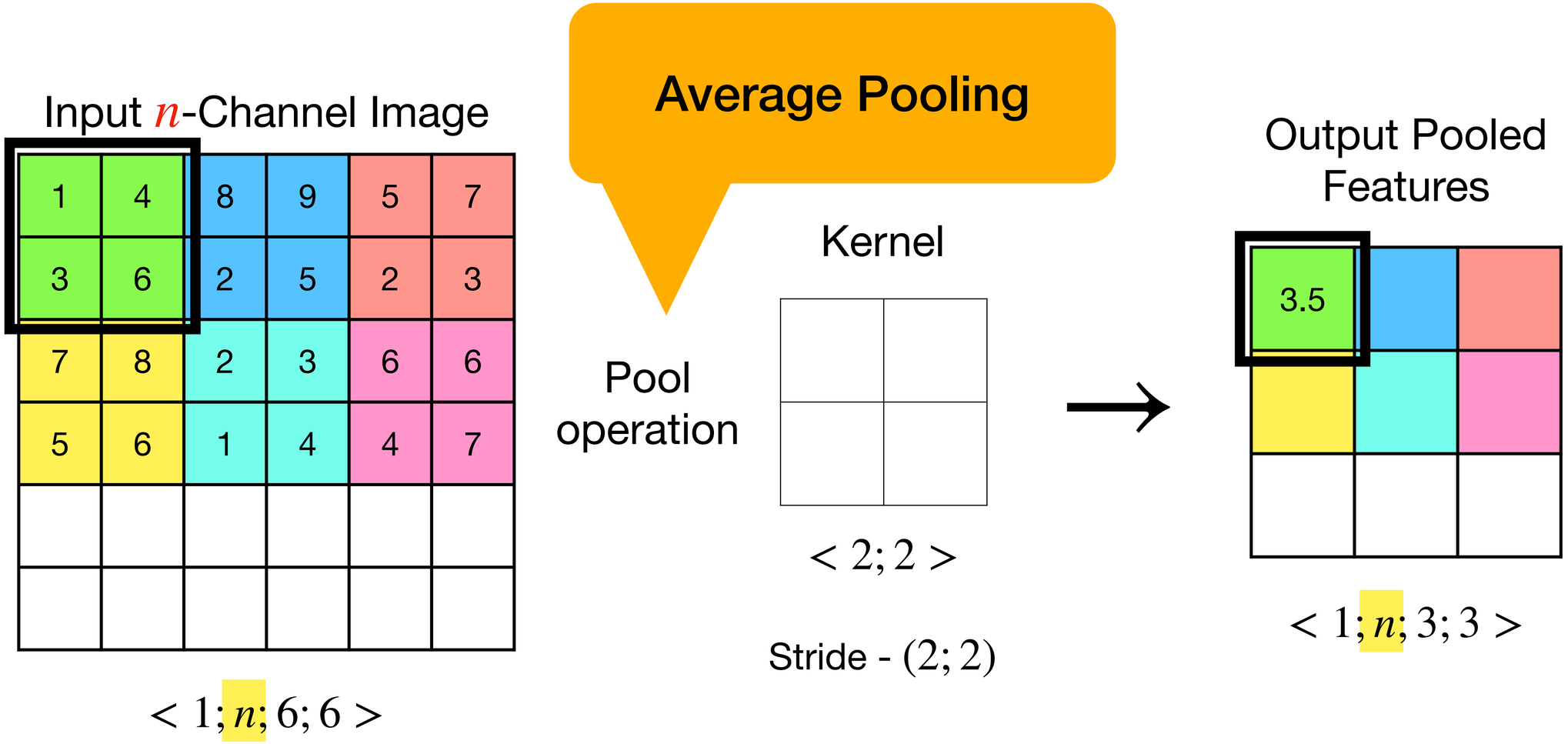
Stride - $(2; 2)$

Output Pooled Features

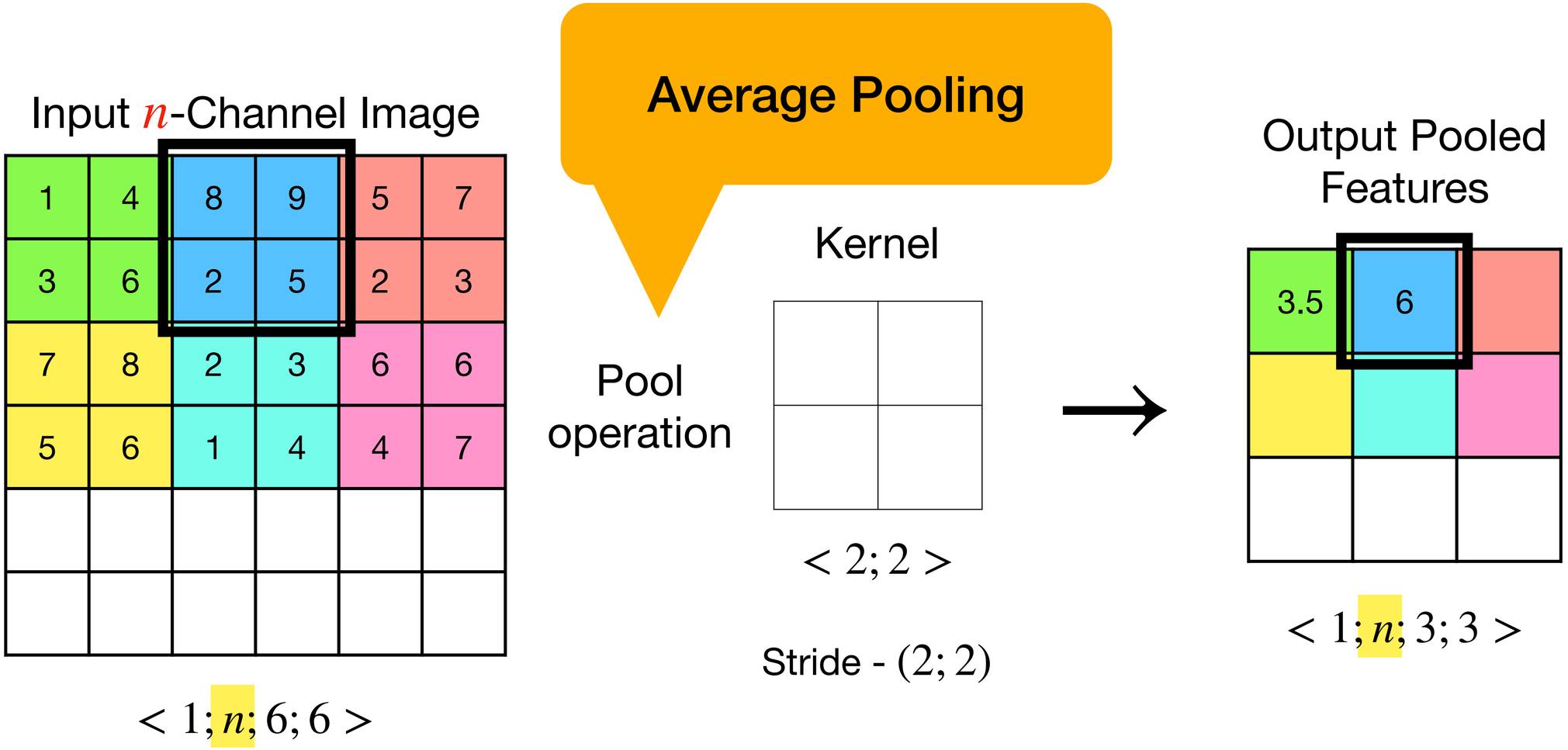


$< 1; n; 3; 3 >$

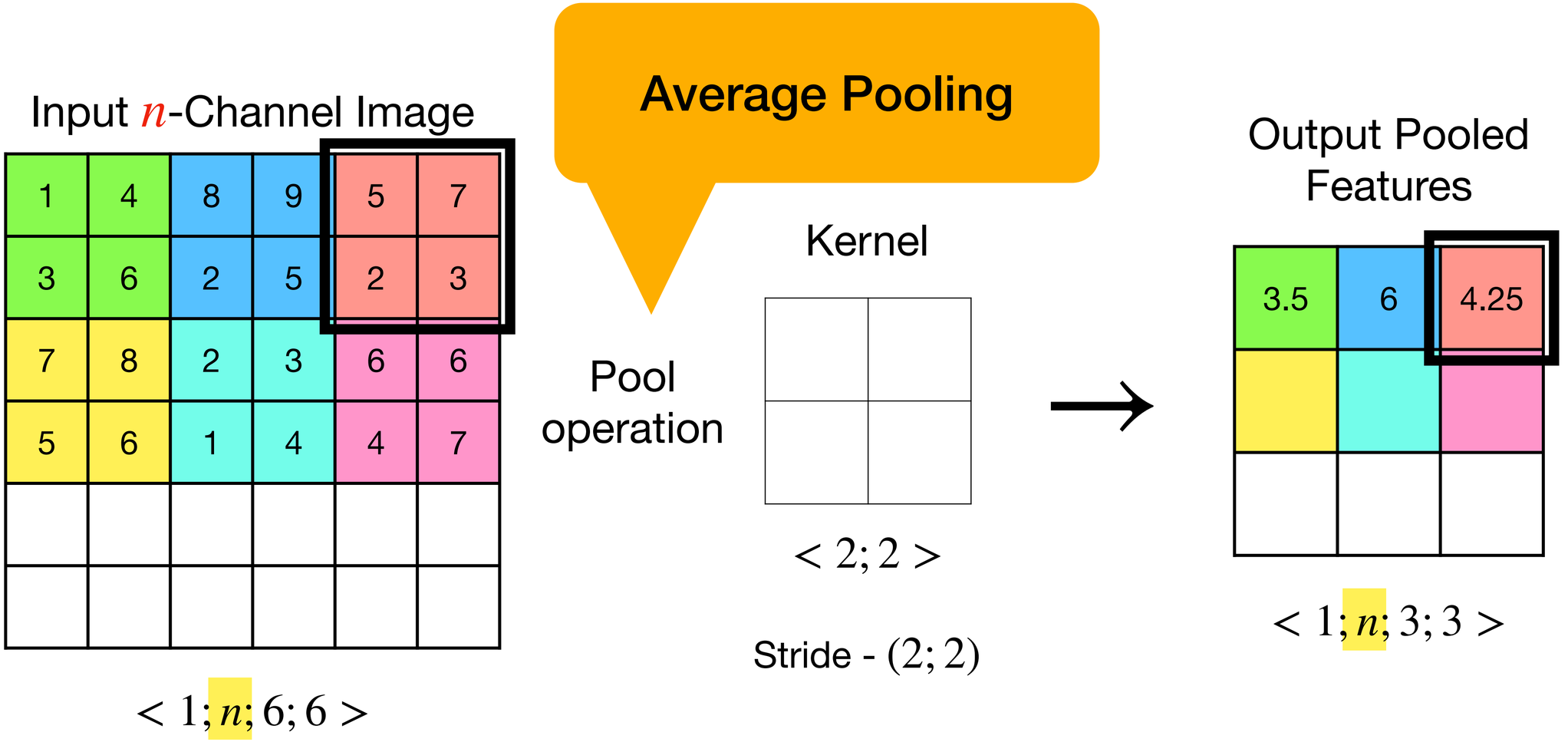
Pooling Layers



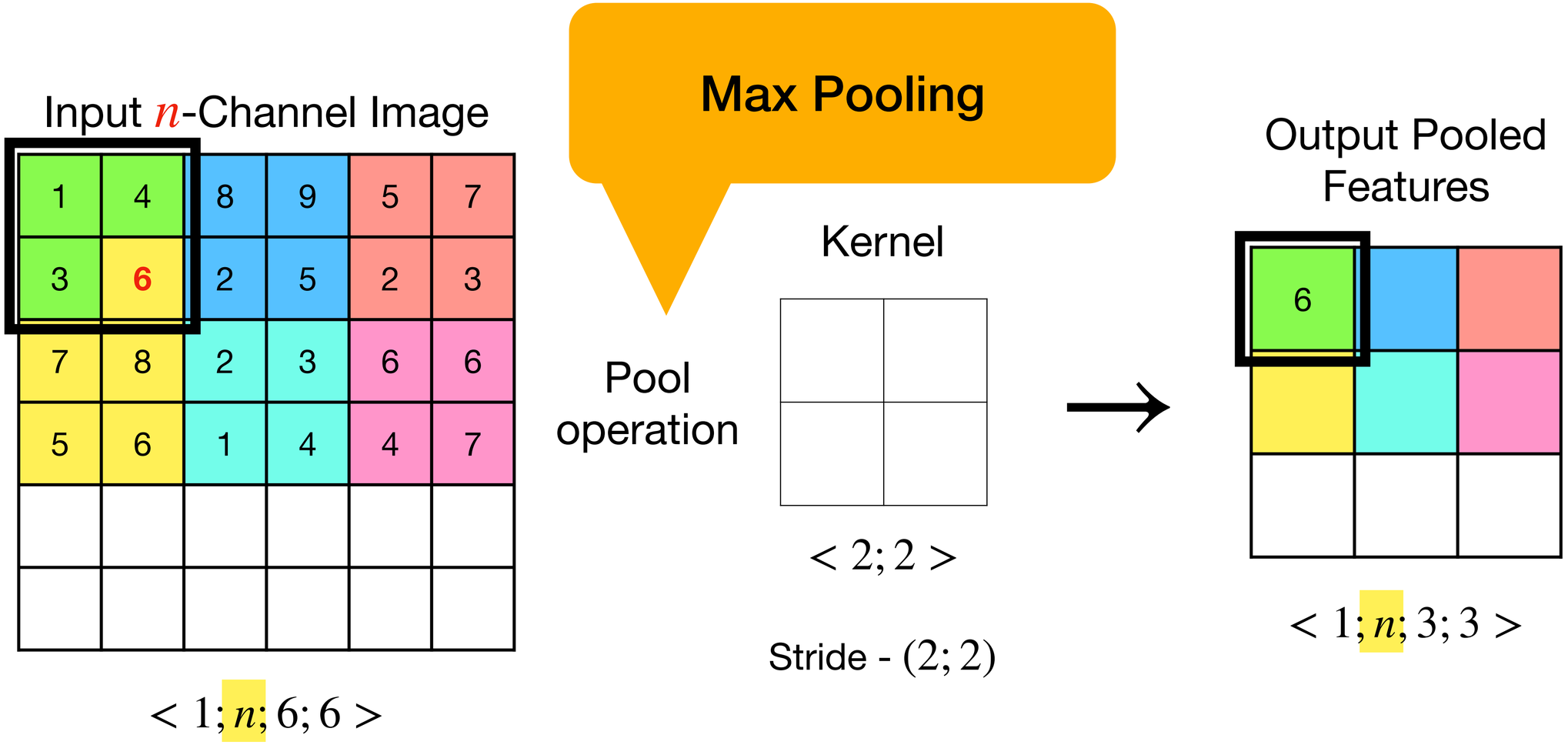
Pooling Layers



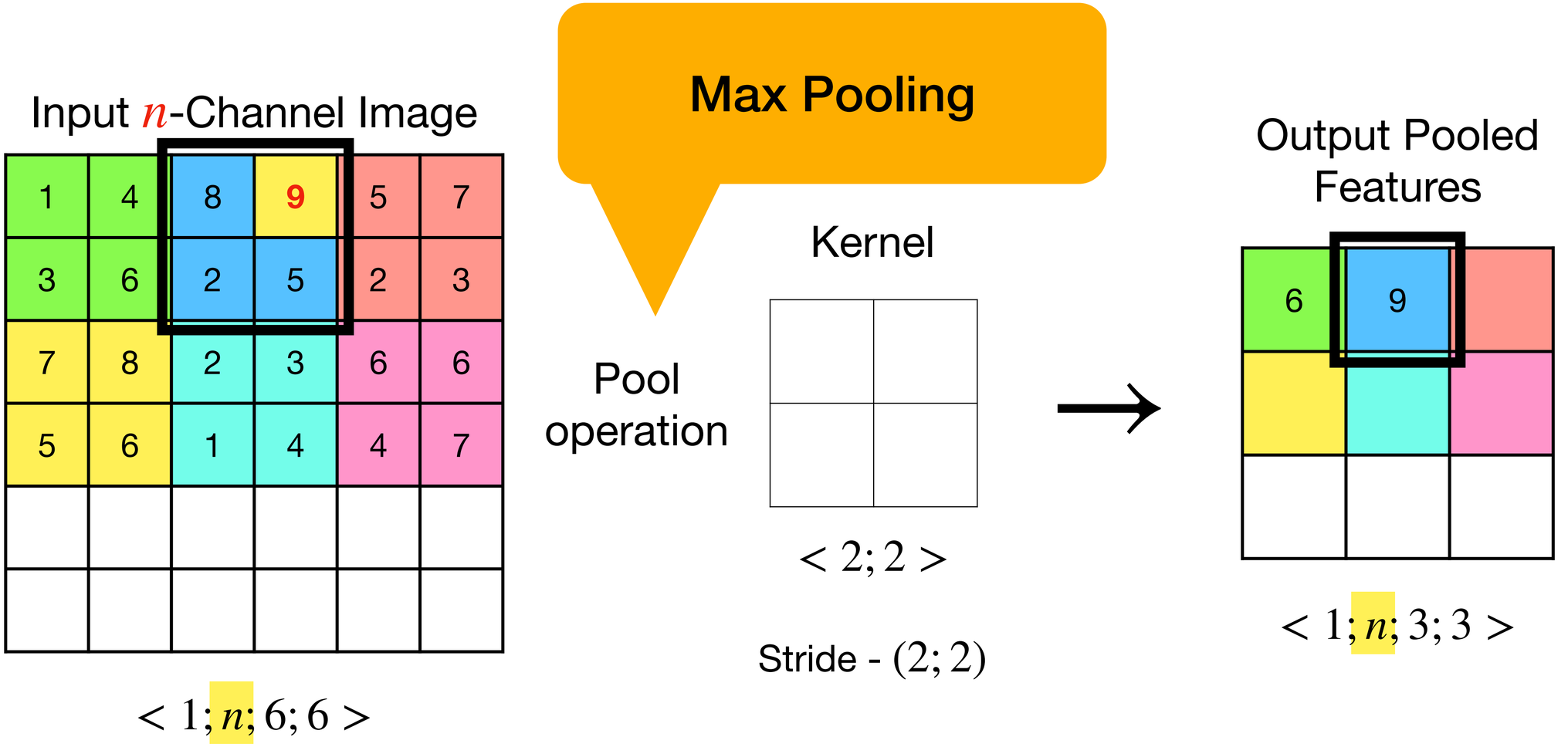
Pooling Layers



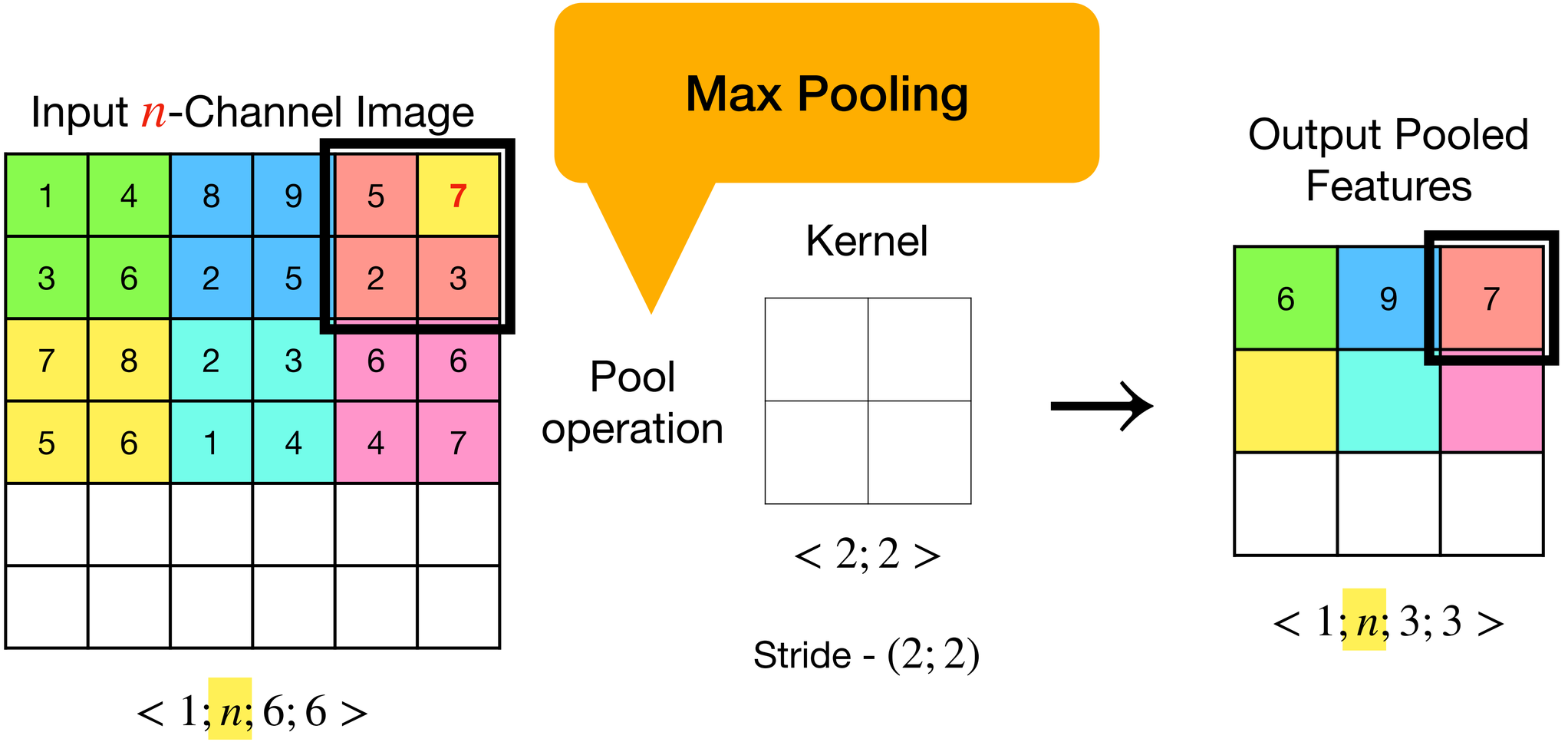
Pooling Layers



Pooling Layers

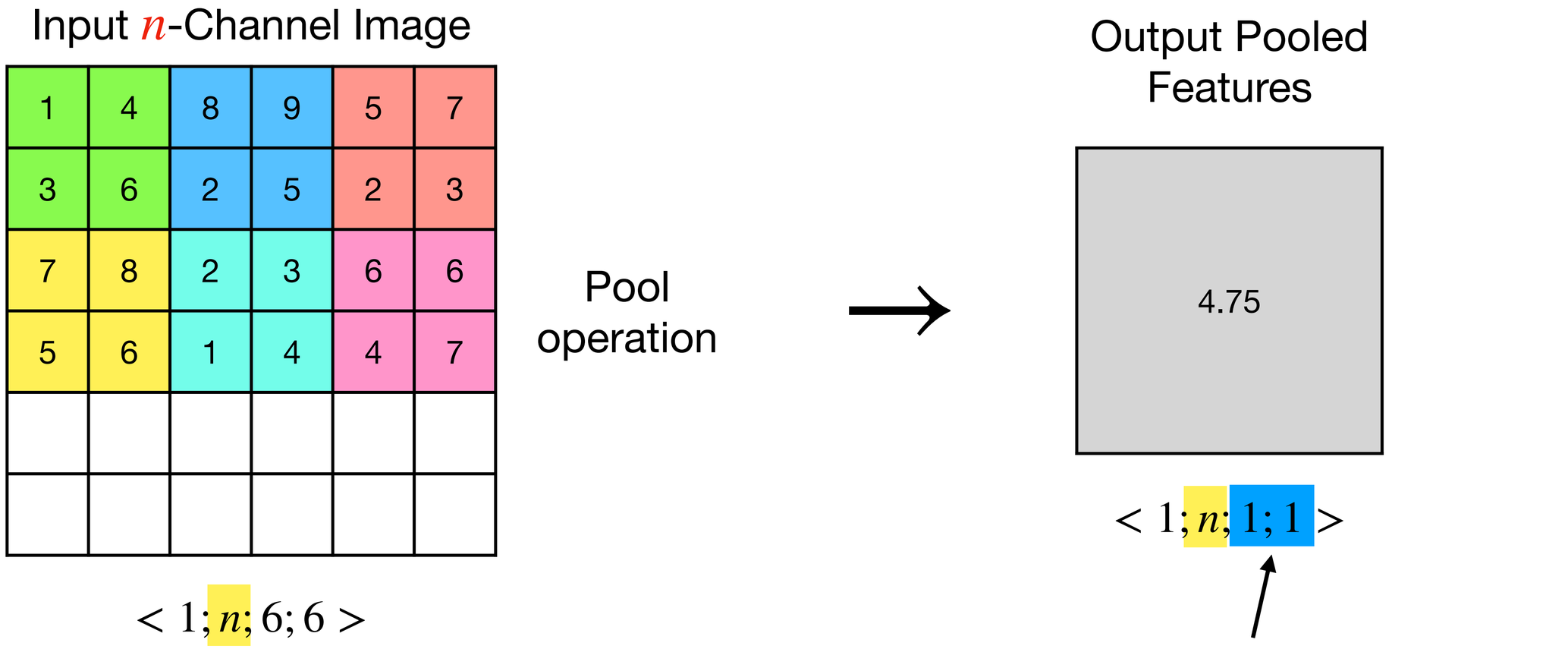


Pooling Layers



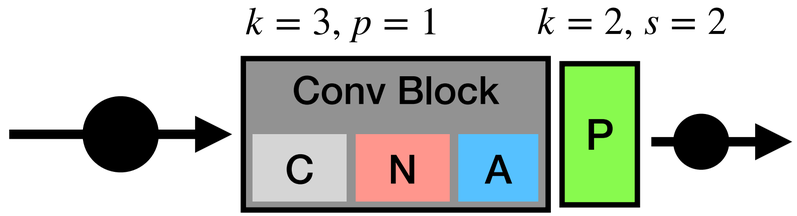
Pooling Layers

Global Average Pooling (Adaptive Pooling)



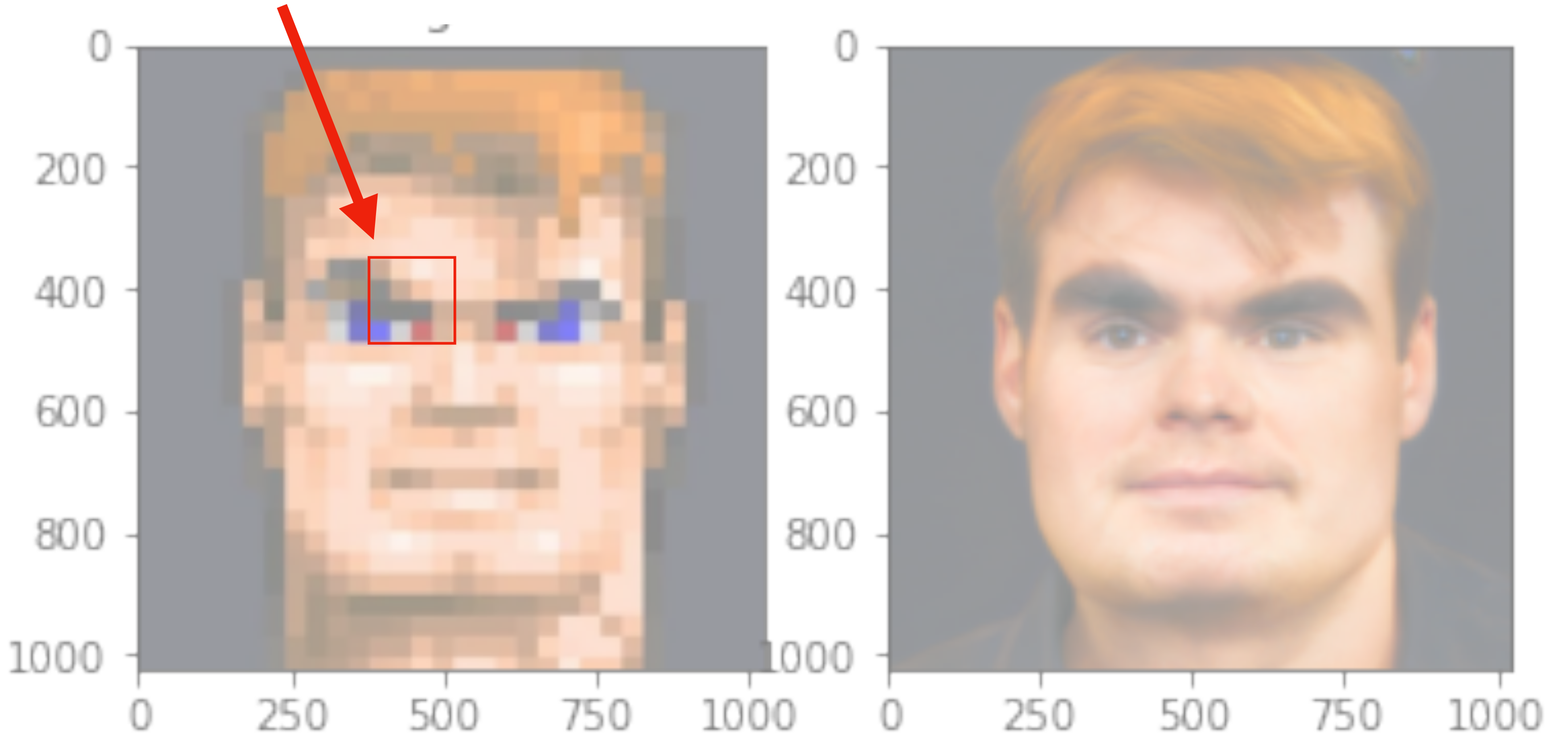
Convolutional Layers

Receptive Field



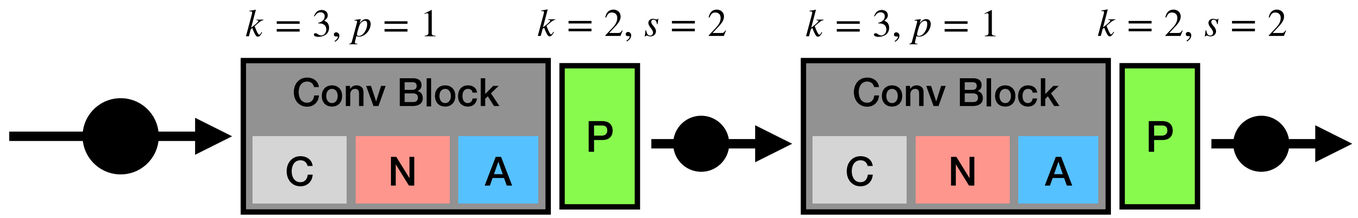
Receptive field

4×4



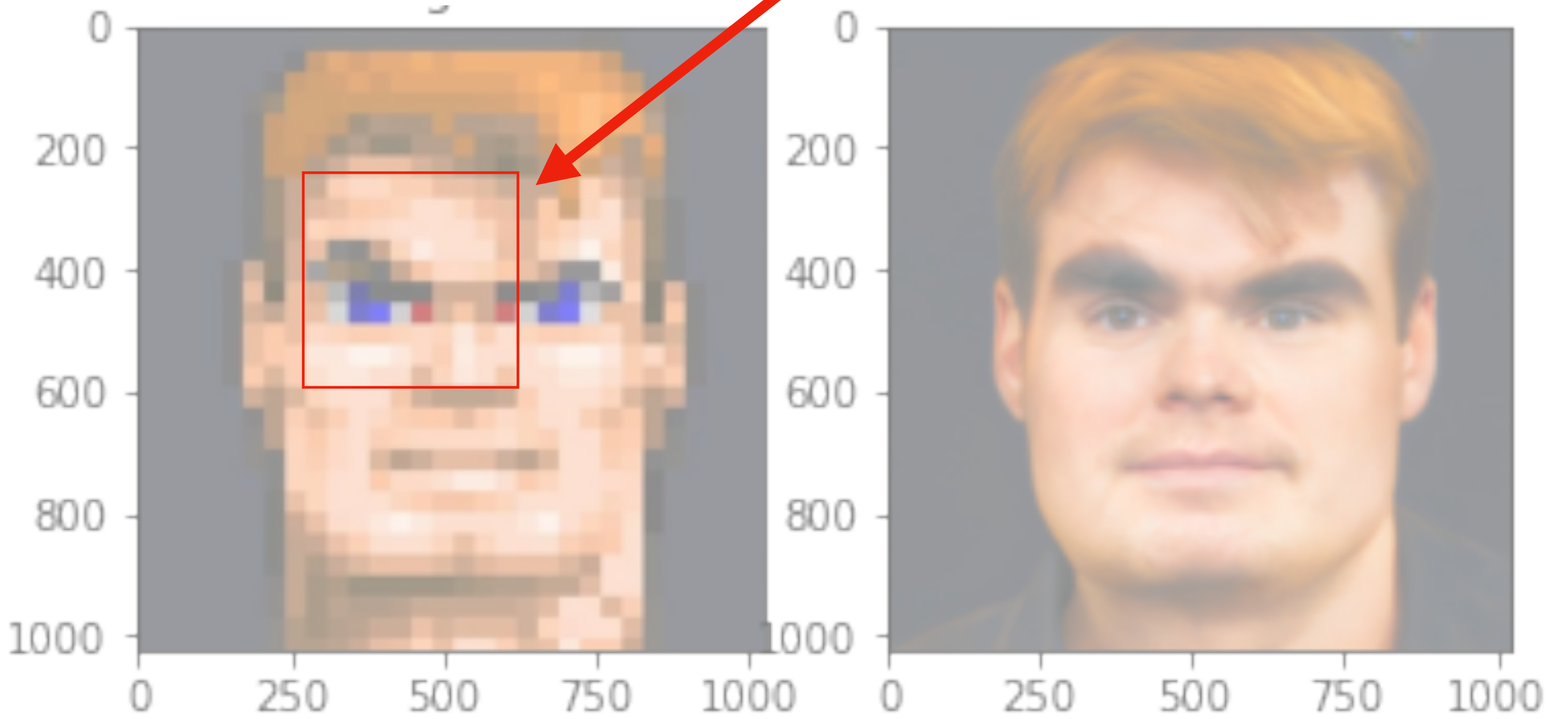
Convolutional Layers

Receptive Field



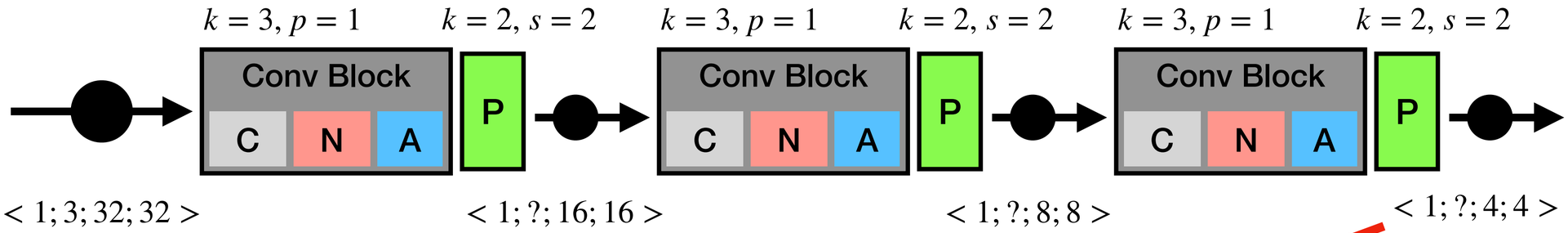
Receptive field

10×10



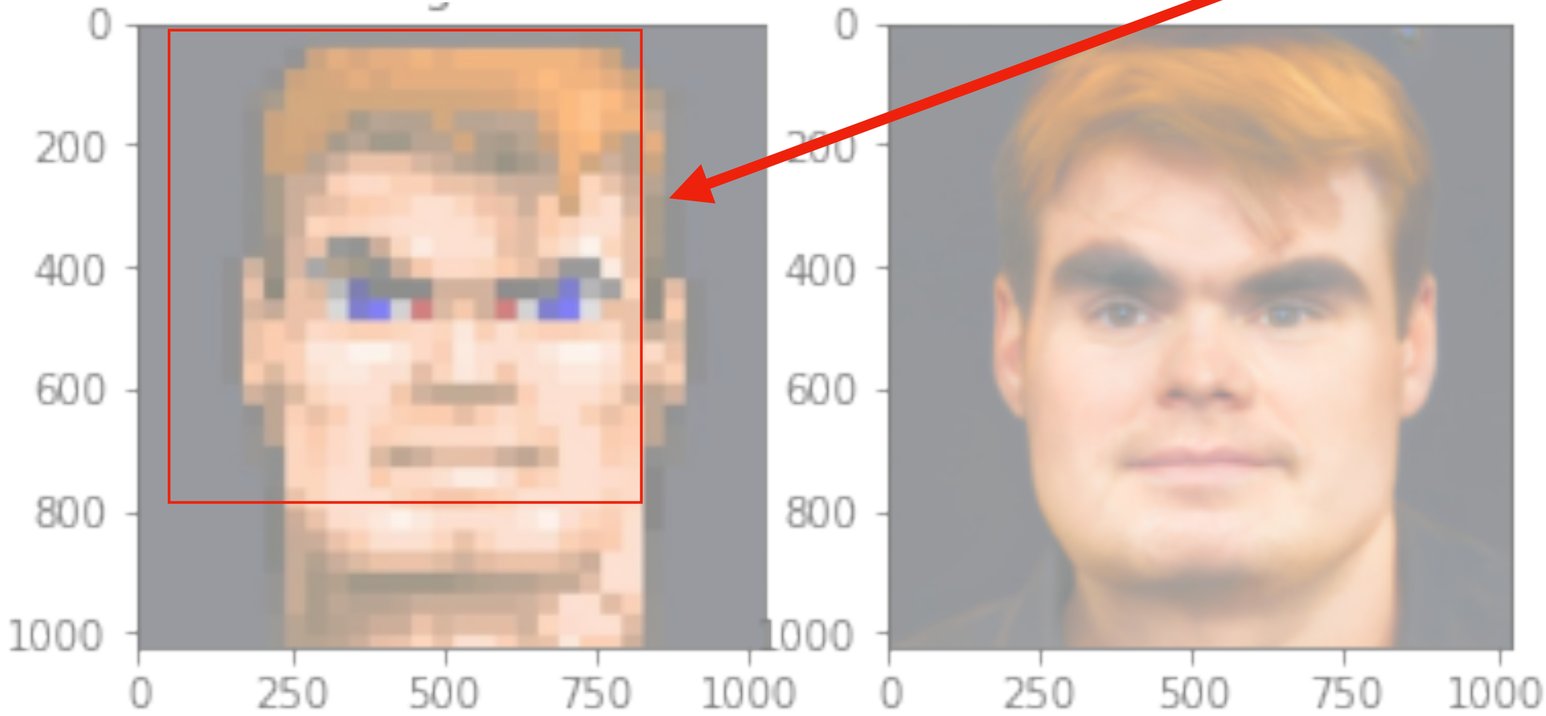
Convolutional Layers

Receptive Field



Receptive field

22×22



Convolutional Layers

Receptive Field - How to Compute ?

- Go backward, ignore padding

$$h_{out} = \left\lfloor \frac{h_{in} + 2p_h - (k_h - 1) - 1}{s_h} + 1 \right\rfloor$$

$$h_{out} = \left\lfloor \frac{h_{in} - (k_h - 1) - 1}{s_h} + 1 \right\rfloor$$

$$s_h(h_{out} - 1) = h_{in} - (k_h - 1) - 1$$

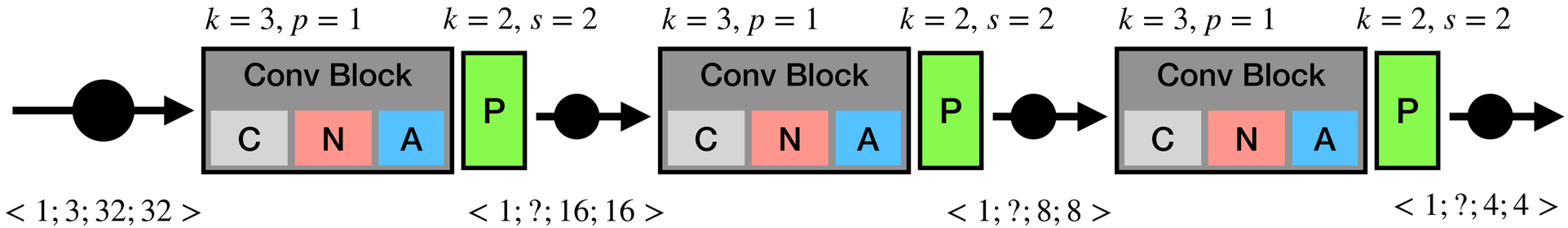
$$s_h(h_{out} - 1) + 1 + (k_h - 1) = h_{in}$$

$$s_h(h_{out} - 1) + k_h = h_{in}$$

Convolutional Layers

Receptive Field

$$s_h(h_{out} - 1) + k_h = h_{in}$$



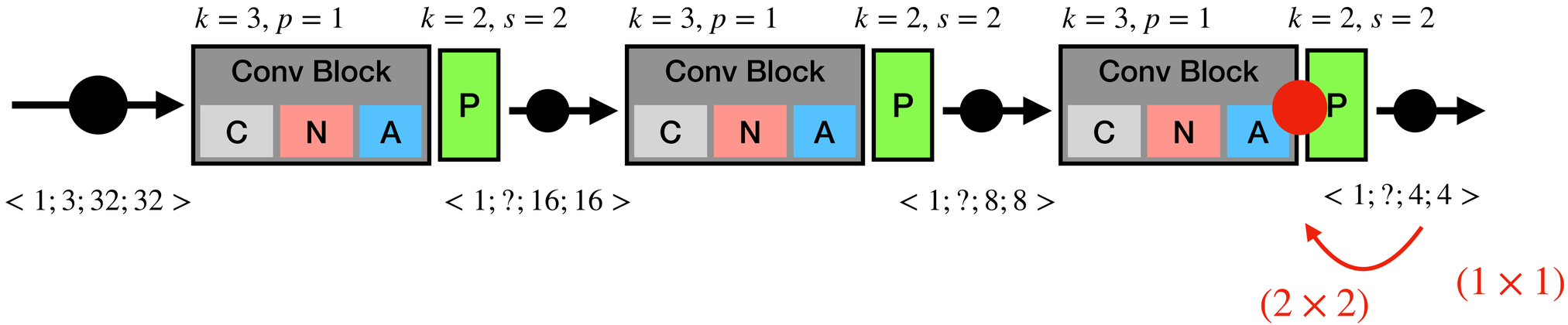
(1×1)

Start with 1x1
we are looking for the reception field of each element of the final activation map

Convolutional Layers

Receptive Field

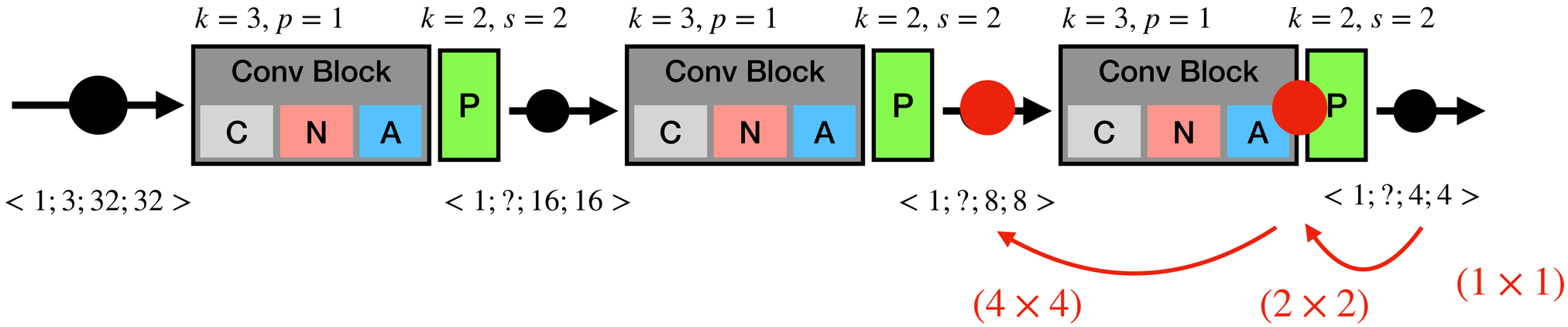
$$s_h(h_{out} - 1) + k_h = h_{in}$$



Convolutional Layers

Receptive Field

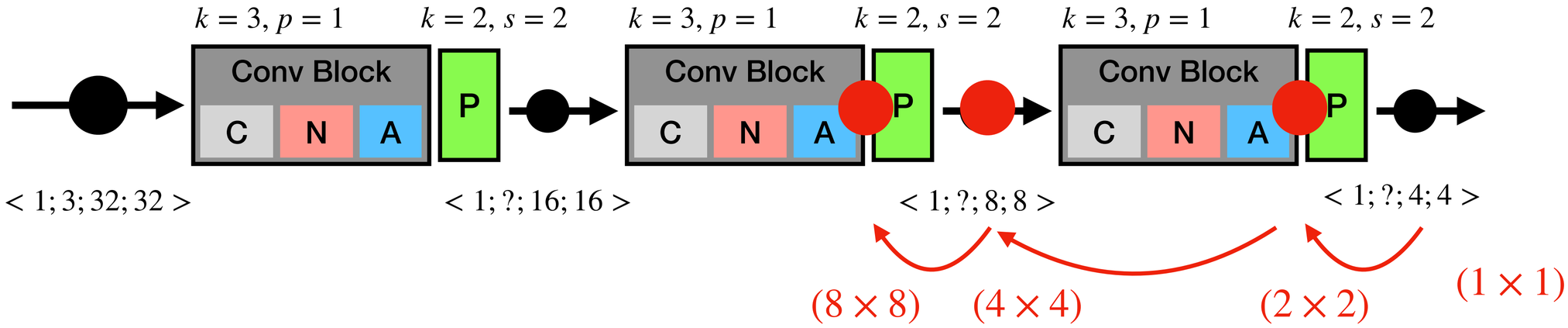
$$s_h(h_{out} - 1) + k_h = h_{in}$$



Convolutional Layers

Receptive Field

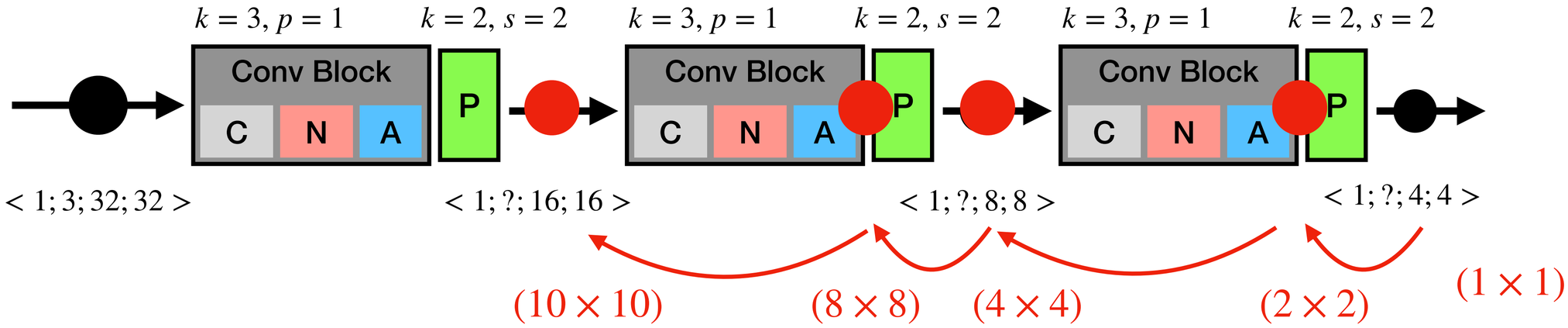
$$s_h(h_{out} - 1) + k_h = h_{in}$$



Convolutional Layers

Receptive Field

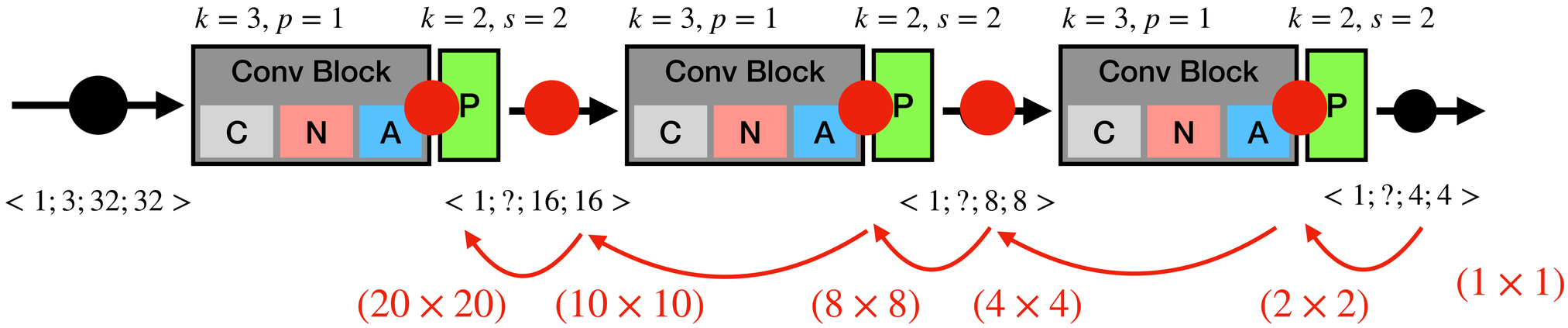
$$s_h(h_{out} - 1) + k_h = h_{in}$$



Convolutional Layers

Receptive Field

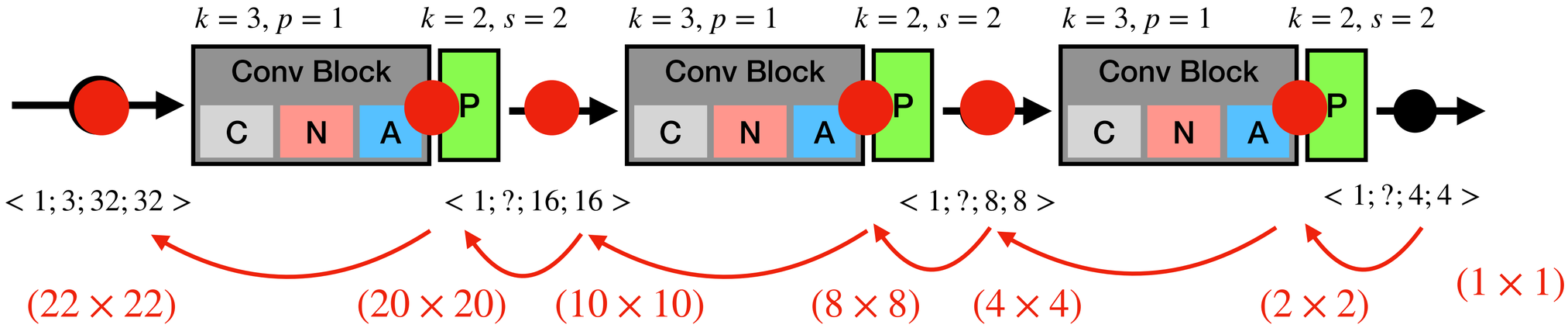
$$s_h(h_{out} - 1) + k_h = h_{in}$$



Convolutional Layers

Receptive Field

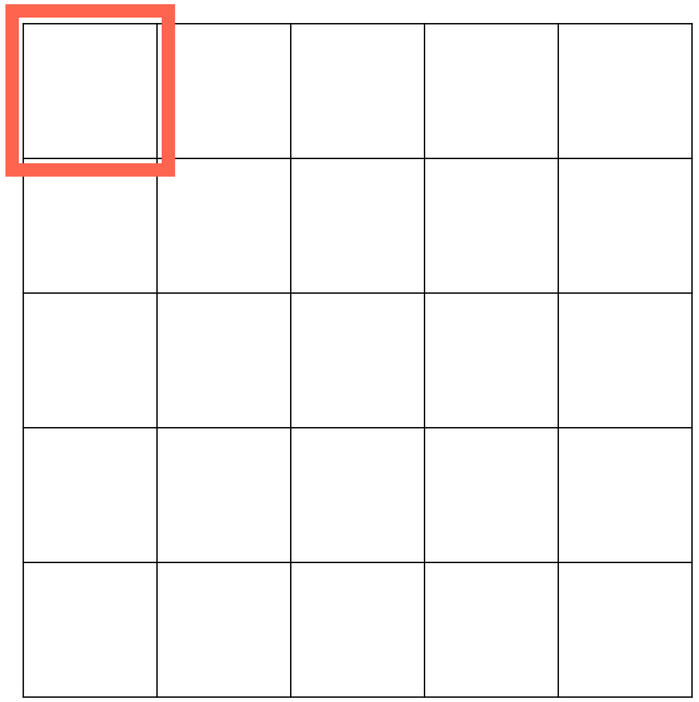
$$s_h(h_{out} - 1) + k_h = h_{in}$$



Convolutional Layer

1x1 Convolution

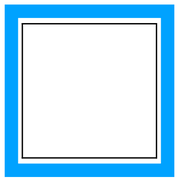
Input n-Channel Image



$< 1; n; 5; 5 >$

$*$

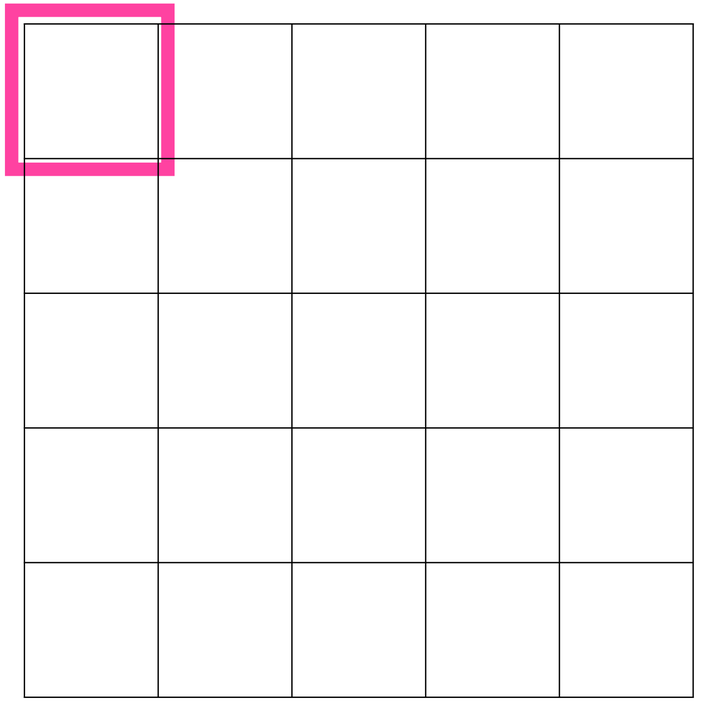
Kernel



$< n; 1; 1 >$

$\rightarrow$

Output Convolved Features



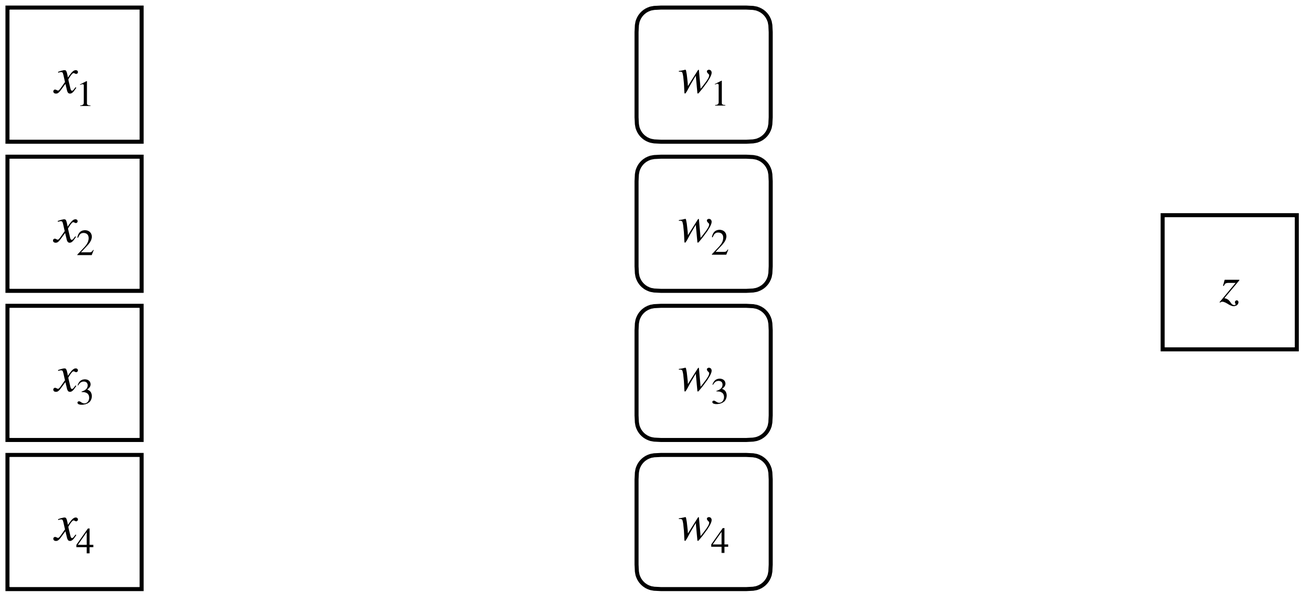
$< 1; 1; 5; 5 >$

Convolutional Layer

1x1 Convolution

Single Artificial Neuron

$$z = w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b$$



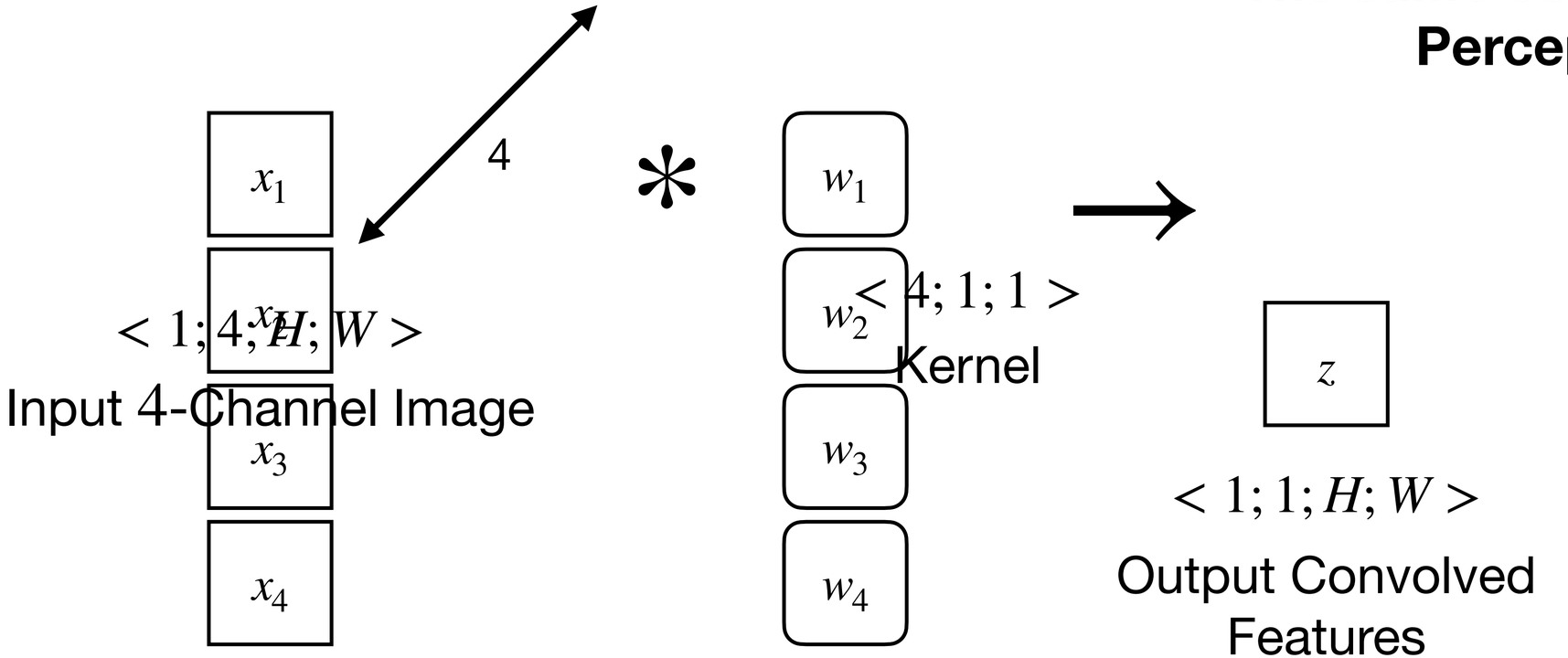
Convolutional Layer

1x1 Convolution

1x1 Convolution

$$z = w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b$$

The same computation as Perceptron !



Convolutional Layer

1x1 Convolution

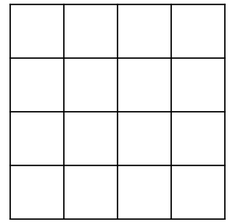
- **Is equivalent to “Linear” for flat vectors**
    - Computed channel-wise **for each element** of activation map
    - Can be applied to input of any size

- Can be used for dimensionality reduction

Implementation of Convolution

$< B;\ C;\ H;\ W >$

$k = 3$

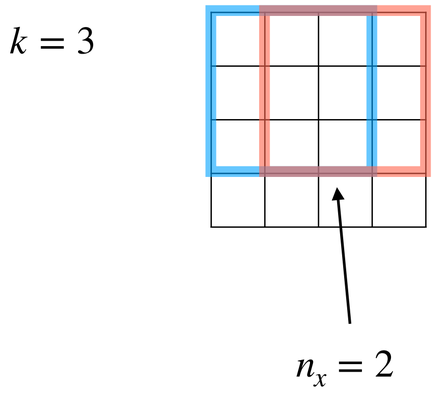


1. Unfold operation
    - Extracts $(k_h \times k_w)$ patches from input according to stride and padding rules

Implementation of Convolution

1. Unfold operation
    - Extracts $(k_h \times k_w)$ patches from input according to stride and padding rules

$< B;\ C;\ H;\ W >$

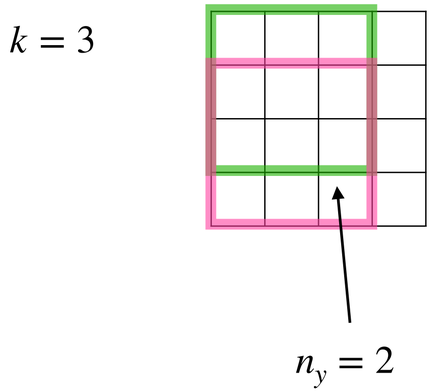


Two possible positions of kernel in horizontal direction

Implementation of Convolution

1. Unfold operation
    - Extracts $(k_h \times k_w)$ patches from input according to stride and padding rules

$< B;\ C;\ H;\ W >$



Two possible positions of kernel in vertical direction

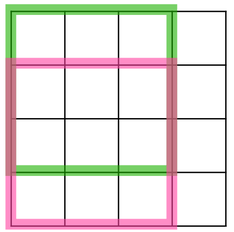
Implementation of Convolution

$< B;\ C;\ H;\ W >$

1. Unfold operation

   - Extracts $(k_h \times k_w)$ patches from input according to stride and padding rules

$k = 3$



Total number of patches
(output activation map size)

$$l = n_x \,.\, n_y$$

Implementation of Convolution

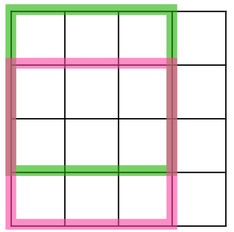
$< B;\ C;\ H;\ W >$

1. Unfold operation

   - Extracts $(k_h \times k_w)$ patches from input according to stride and padding rules

   - Each patch with C features (channel depth) is flattened into a simple feature vector

$k = 3$



Total number of patches
(output activation map size)

$$l = n_x \,.\, n_y$$

$$n_f = C \,.\, k_h \,.\, k_w$$

$$< B;\ n_f;\ l >$$

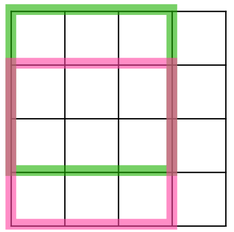
Implementation of Convolution

$< B;\ C;\ H;\ W >$

1. Unfold operation

   - Extracts $(k_h \times k_w)$ patches from input according to stride and padding rules
   - Each patch with C features (channel depth) is flattened into a simple feature vector
   - Transpose 2^{nd} and 3^{rd} dimension

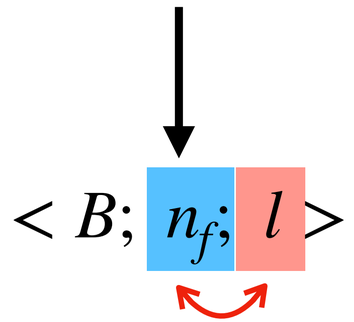
$k = 3$



Total number of patches
(output activation map size)

$$l = n_x \,.\, n_y$$

$$n_f = C \,.\, k_h \,.\, k_w$$

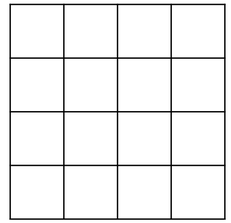


$< B;\ l;\ n_f >$

Implementation of Convolution

$< B;\ C;\ H;\ W >$

$k = 3$



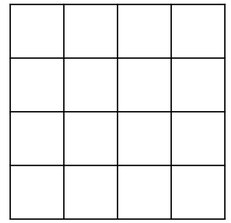
2. Matrix multiplication
    - Output channels - C_{out}
    - Weights matrix W has shape $< n_f;\ C_{out} >$

$$< B;\ l;\ n_f > \times < n_f;\ C_{out} > = < B;\ l;\ C_{out} >$$

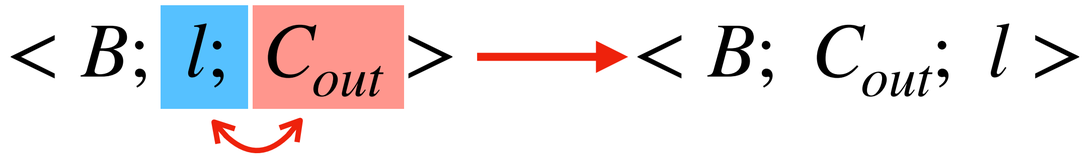
Implementation of Convolution

$< B;\ C;\ H;\ W >$

$k = 3$



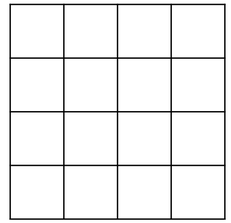
3. Transpose 2^{nd} and 3^{rd} dimension again



Implementation of Convolution

$< B;\ C;\ H;\ W >$

$k = 3$



4. Reshape into $(n_y \times n_x)$

$< B;\ C_{out};\ l > \rightarrow < B;\ C_{out};\ n_y;\ n_x >$

Convolutional Architectures

- 1989 - LeNet-1
- 2012 - AlexNet
- 2014 - Inception
- 2015 - ResNet
- 2016 - DenseNet
- 2019 - EfficientNet
- 2022 - ConvNext
- …

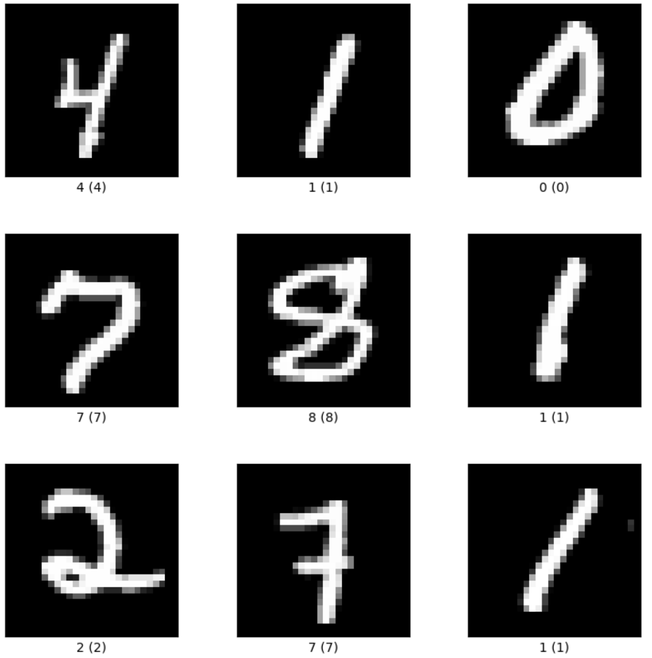
LeNet-5

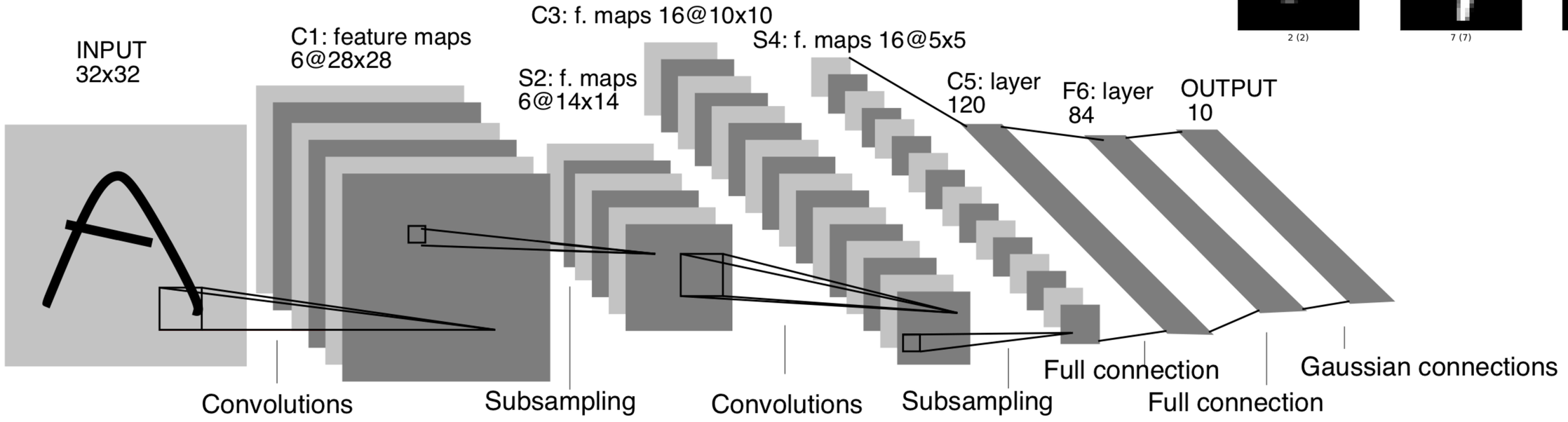
1998

Sigmoid Activation

2x (Conv + Pooling)

3x Fully Connected ($120 \rightarrow 84 \rightarrow 10$)

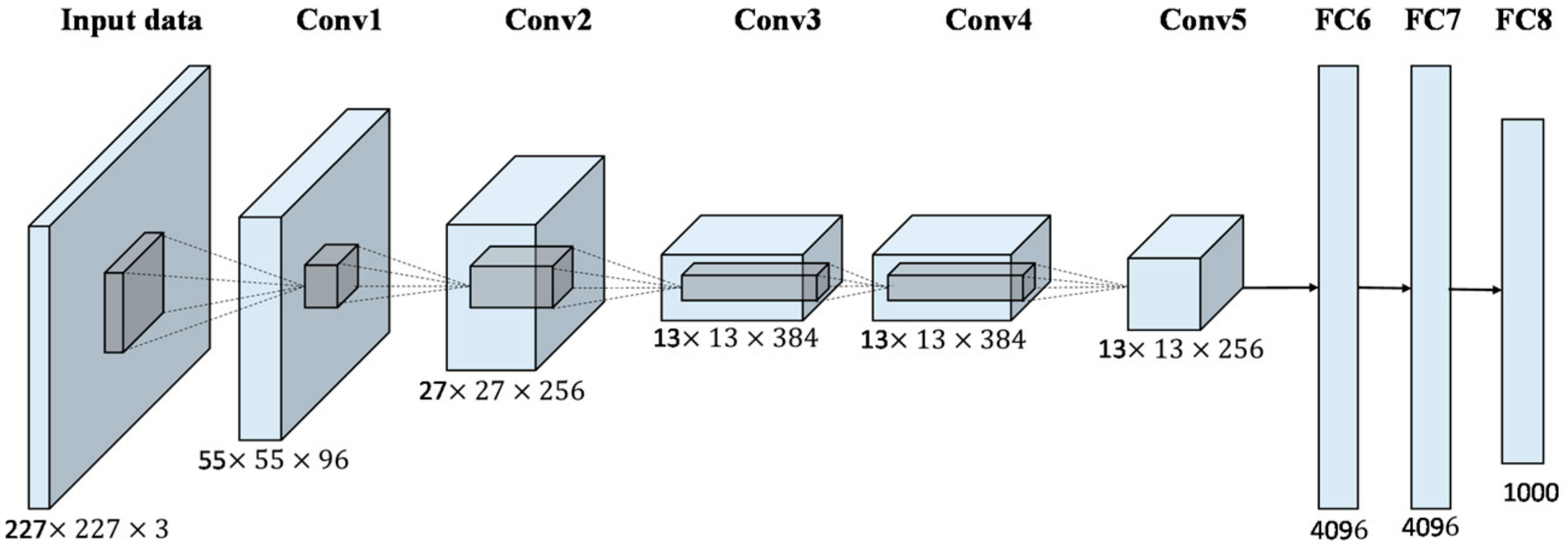




LeCun, Yann, et al. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE* 86.11 (1998): 2278-2324.

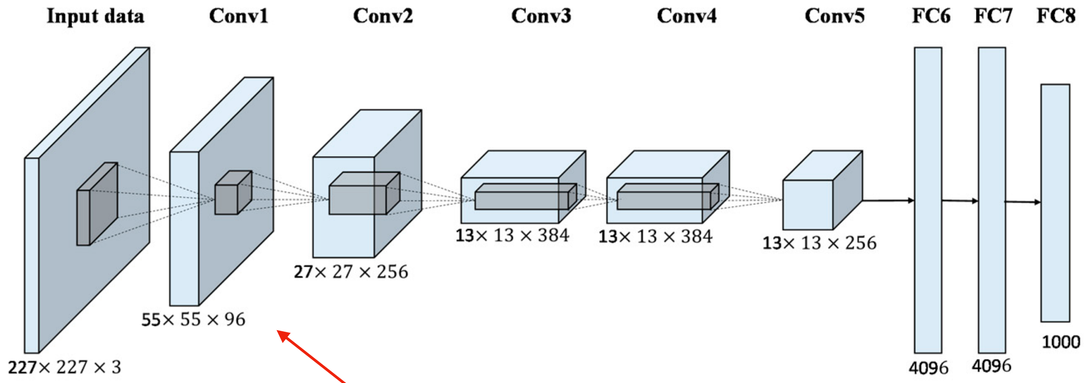
AlexNet

2012



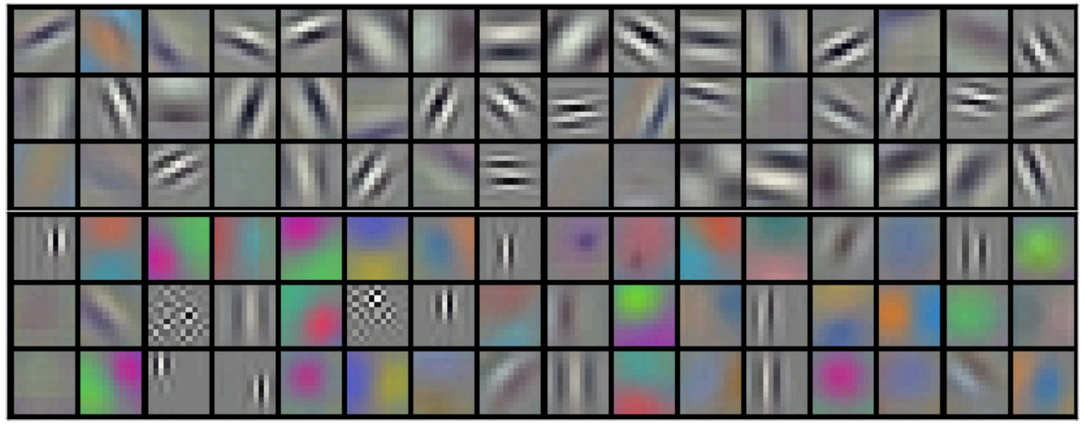
Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems* 25 (2012).

AlexNet
2012



- ImageNet Classification - 1000 classes
- Deep Network by today's standard
- **Trained on 2 GeForce GTX 580 GPUs (3GB each)**
- ReLU activation
- DropOut
- Normalization
- **60 million parameters**

96 filters learned in Conv1

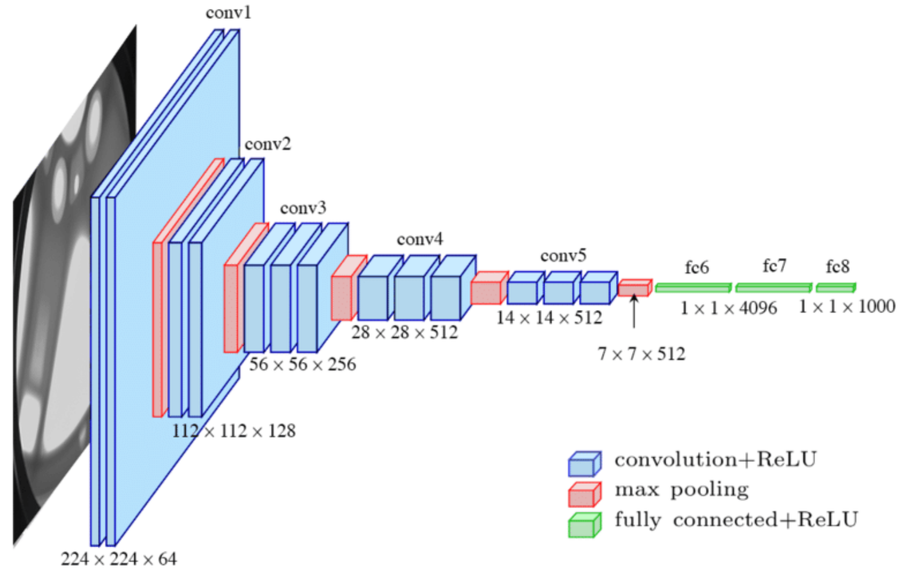


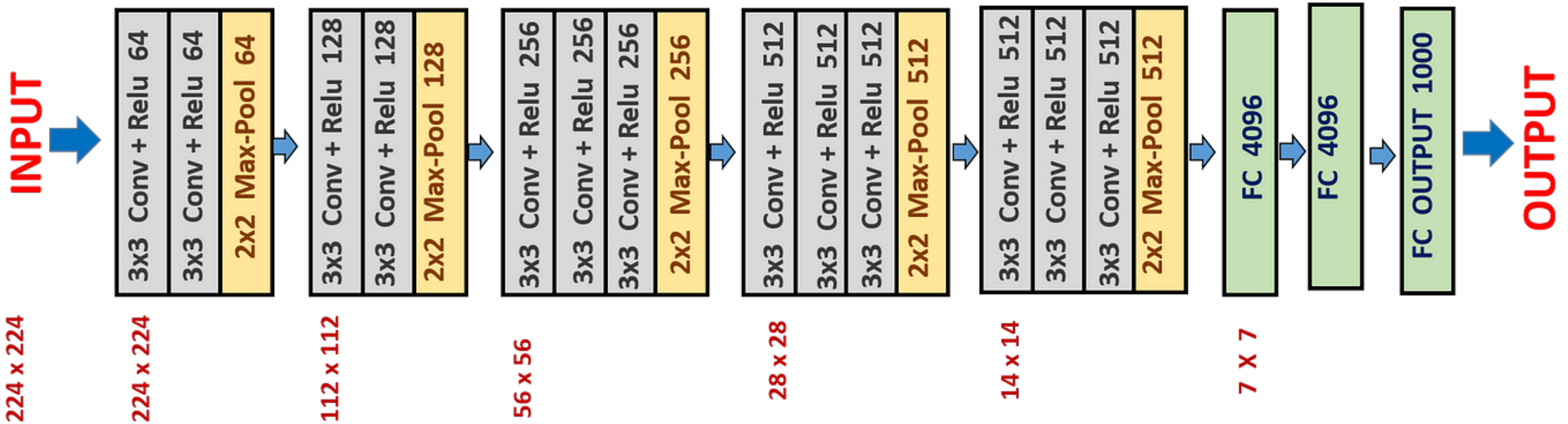
Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems* 25 (2012).

VGGNet

2014

138 millions of parameters



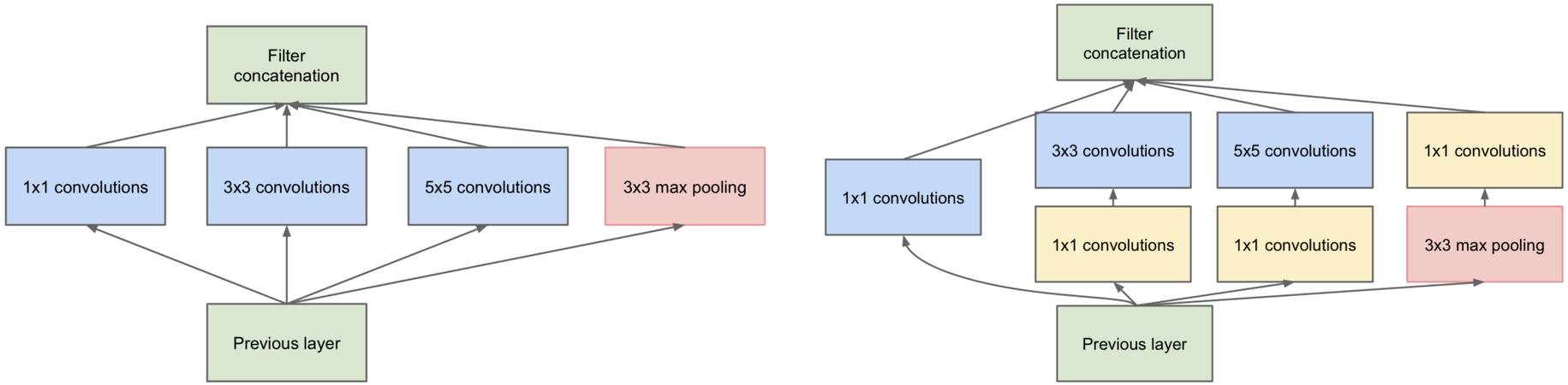


Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556* (2014).

Images - https://medium.com/@siddheshb008/vgg-net-architecture-explained-71179310050f

Inception

2014



(a) Inception module, naïve version (b) Inception module with dimension reductions

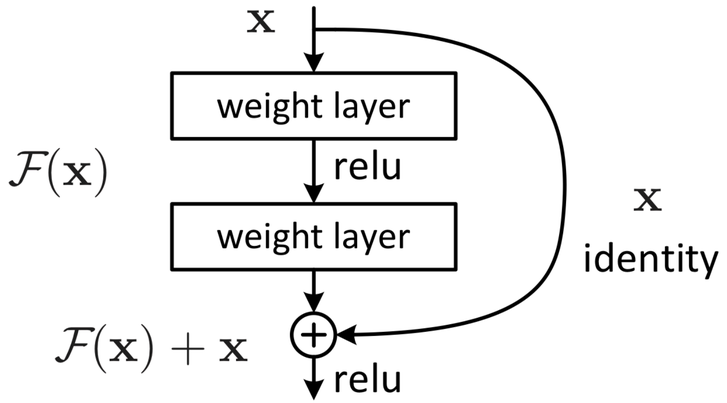
Figure 2: Inception module

Szegedy, Christian, et al. "Going deeper with convolutions." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

ResNet

2015

- Even with ReLU, normalization, initialization, …
  - **Deep networks are difficult to train**
- Proposed **residual block**
  - Skip connection improves gradient propagation

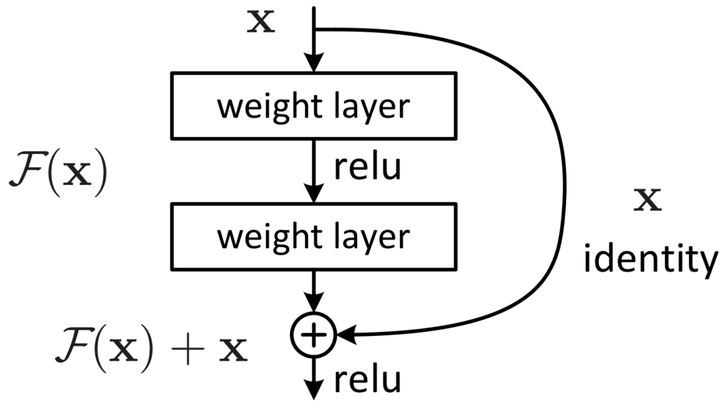


He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

ResNet

2015

- Even with ReLU, normalization, initialization, …
  - **Deep networks are difficult to train**
- Proposed **residual block**
  - Skip connection improves gradient propagation



Convolutional Block:

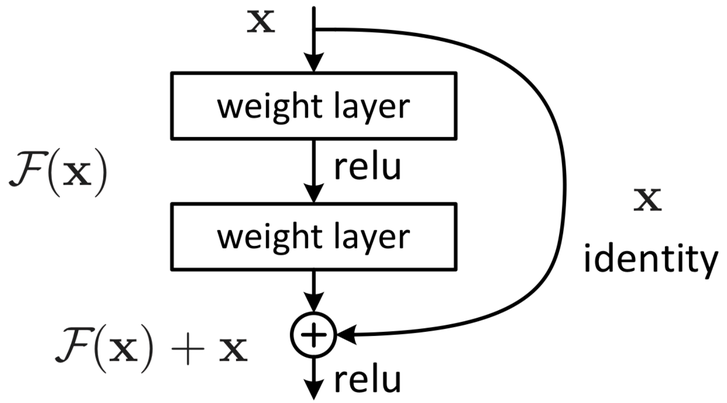
“What representation of x can be useful ?”

He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

ResNet

2015

- Even with ReLU, normalization, initialization, …
  - **Deep networks are difficult to train**
- Proposed **residual block**
  - Skip connection improves gradient propagation



Residual Block:

“How much does x need to be changed to be useful ?”

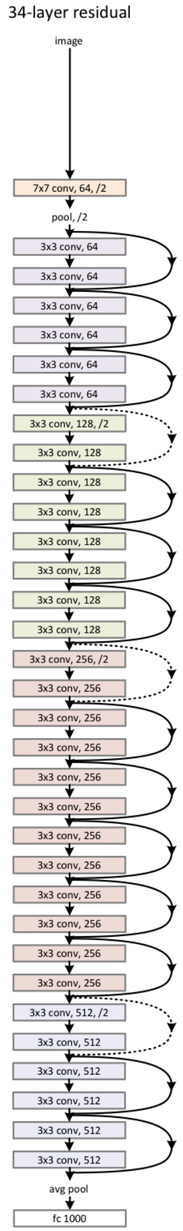
He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

ResNet

2015

- ResNet has become very popular
    - Comes in different sizes - 18, 34, 54, 101, 152 layers

- **Residual Block has become an integral part of many architectures**
- Skip connection has proven to be very effective tool
    - Authors have successfuly trained a model with 1202 layers !



He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

DenseNet

2016

- Direct connections between any two layers with the same feature-map size

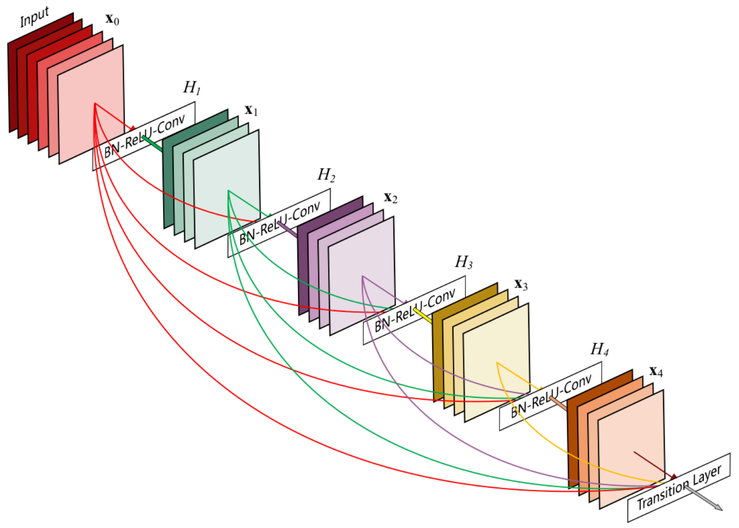
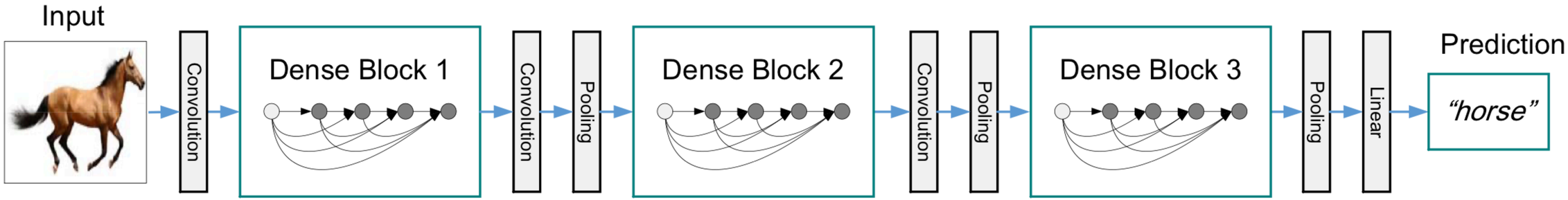


Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

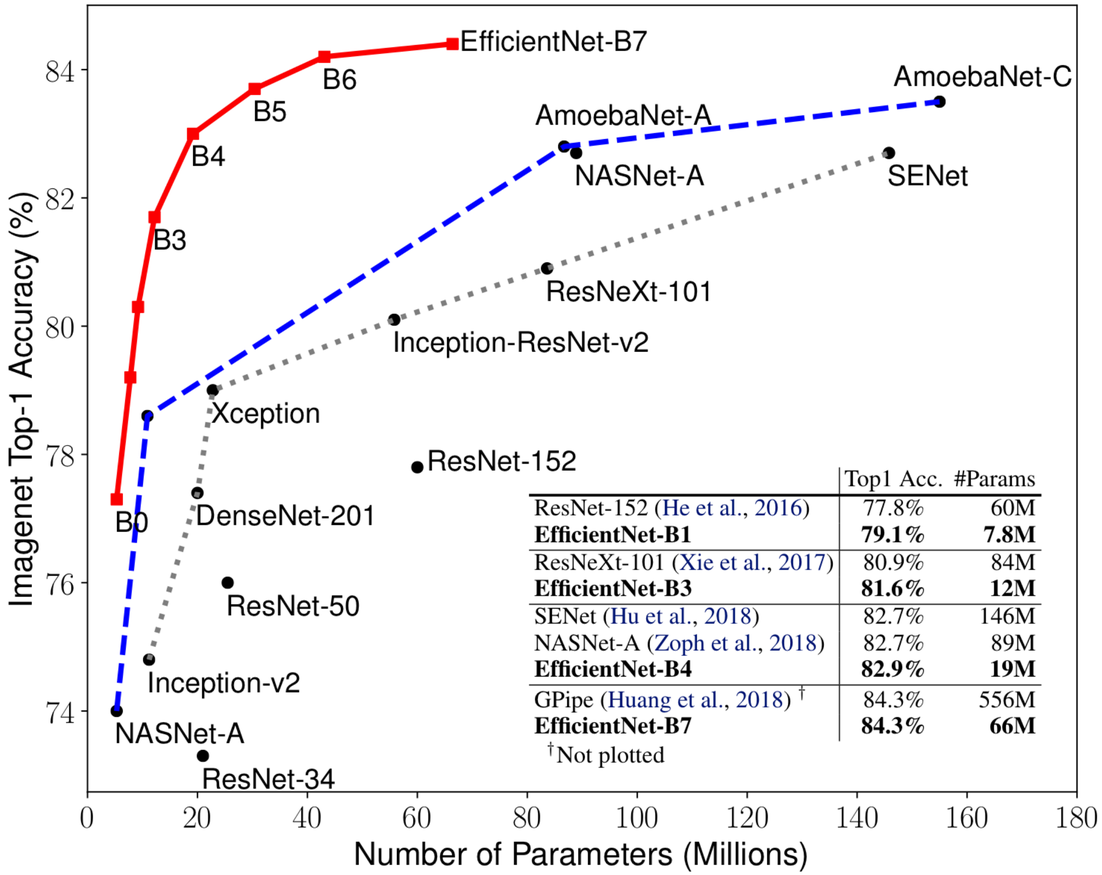


Huang, Gao, et al. "Densely connected convolutional networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

EfficientNet

2019

- Systematic study of how to scale up convolutional models properly
- Demonstrated on scaling up **MobileNets and ResNets**



	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**79.1%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.6%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**82.9%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.3%**	**66M**

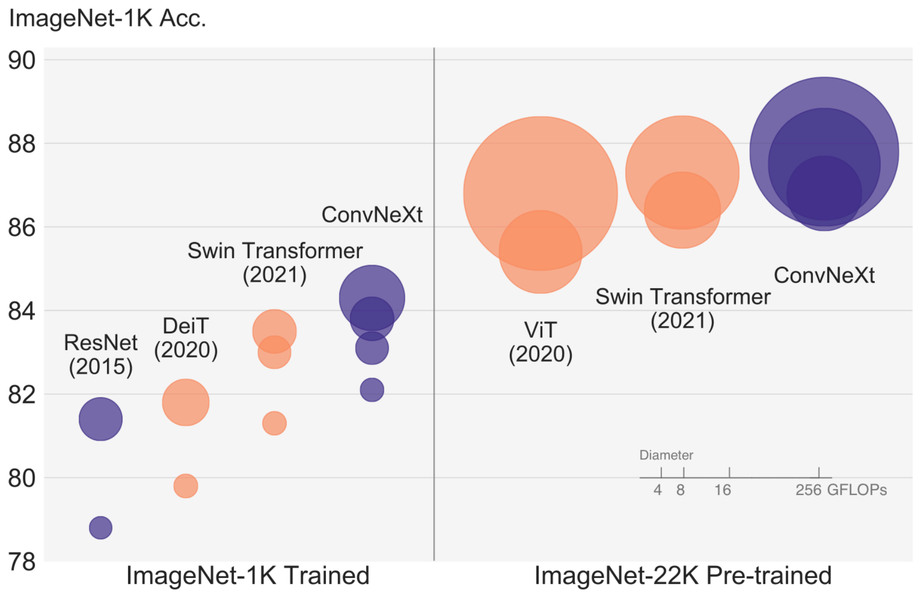
†Not plotted

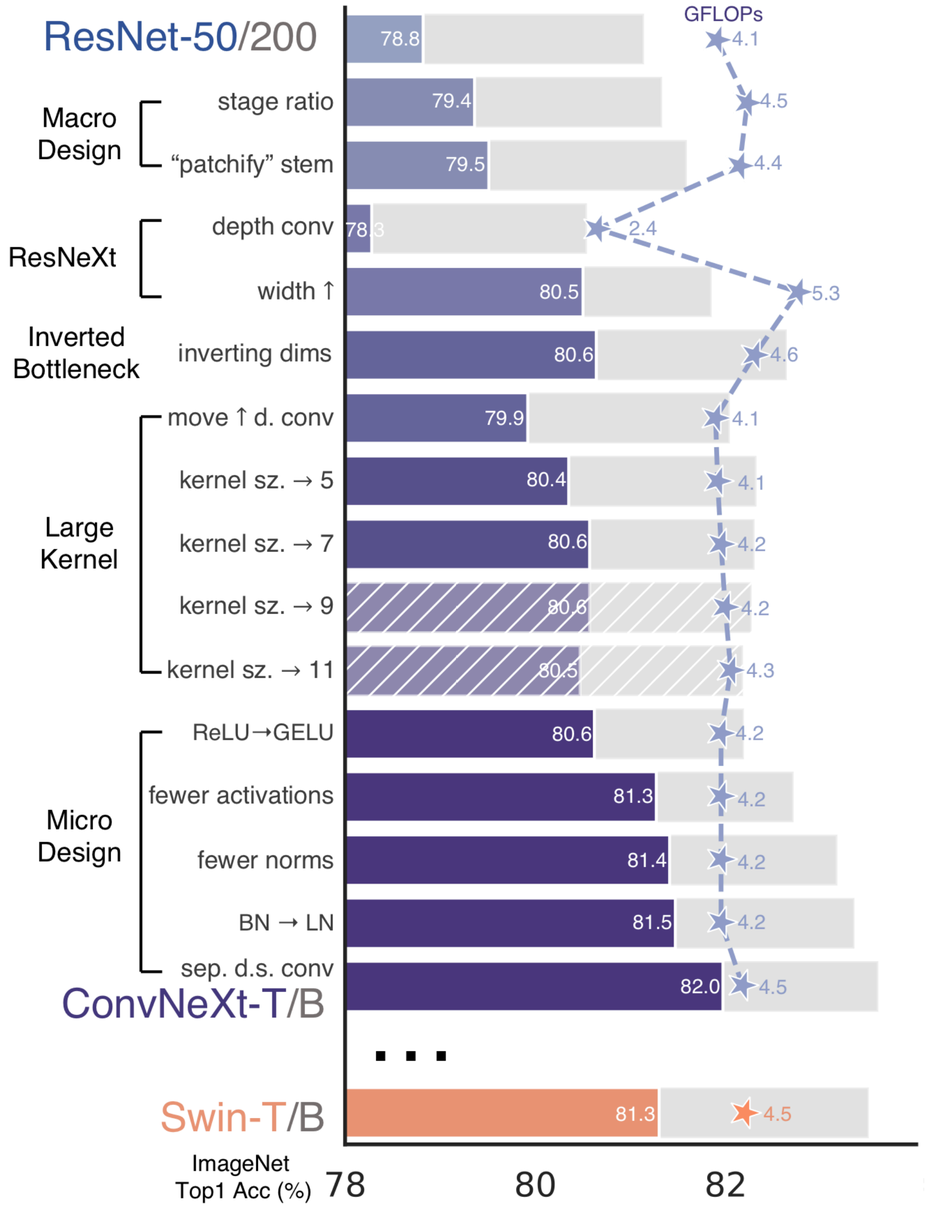
Tan, Mingxing, and Quoc Le. "Efficientnet: Rethinking model scaling for convolutional neural networks." *International conference on machine learning*. PMLR, 2019.

ConvNeXt

2022

- Modernized ResNet toward the design of Vision Transformer





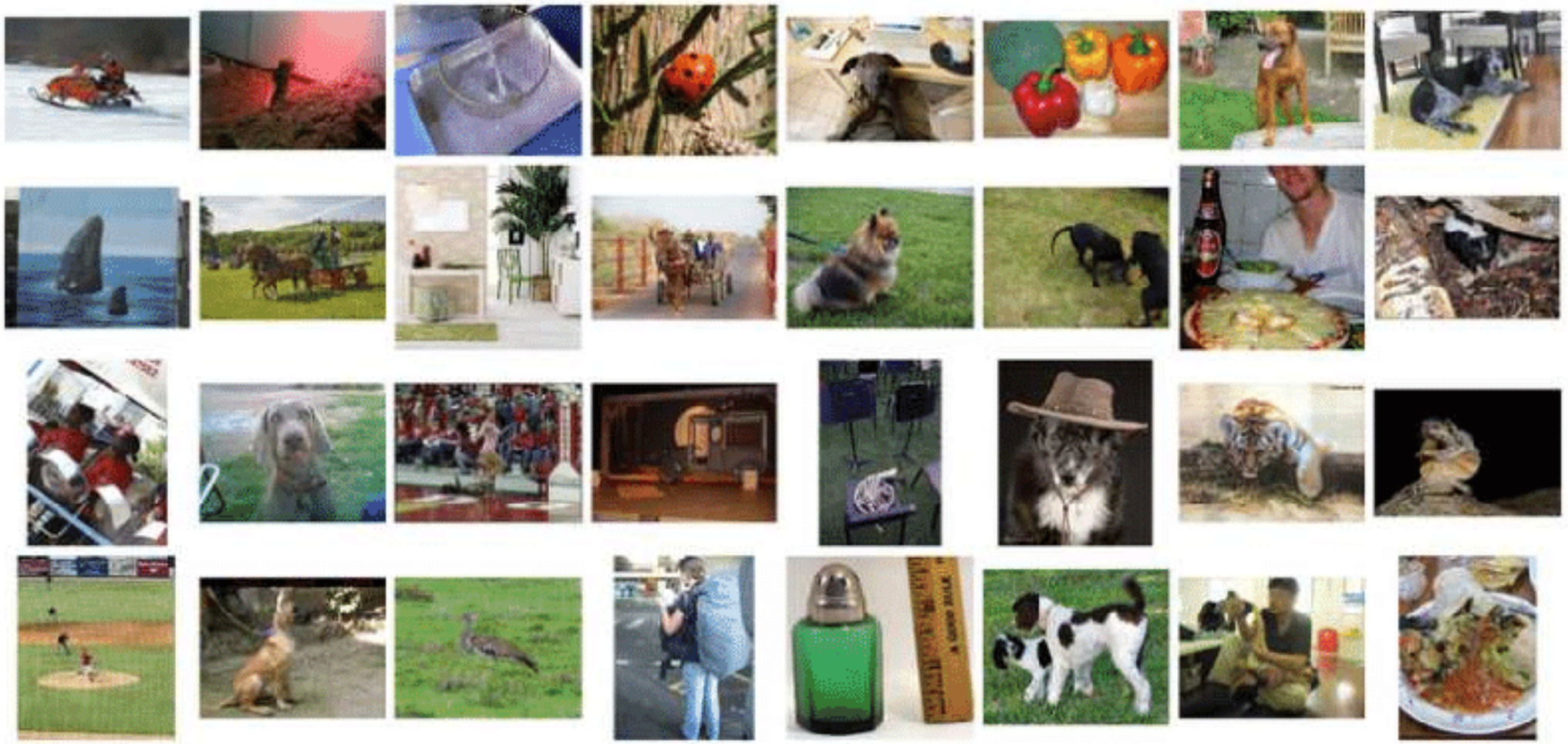
Liu, Zhuang, et al. "A convnet for the 2020s." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2022.

Using a Pre-Trained Model

- Decide which architecture is most suitable for you
    - Can you use big model ?
    - Do you need fast inference ?
    - Do you have CPU/GPU/memory limitations ?

Using a Pre-Trained Model

Training Dataset - ImageNet



Using a Pre-Trained Model

Training Dataset - COCO - Common Objects in Context

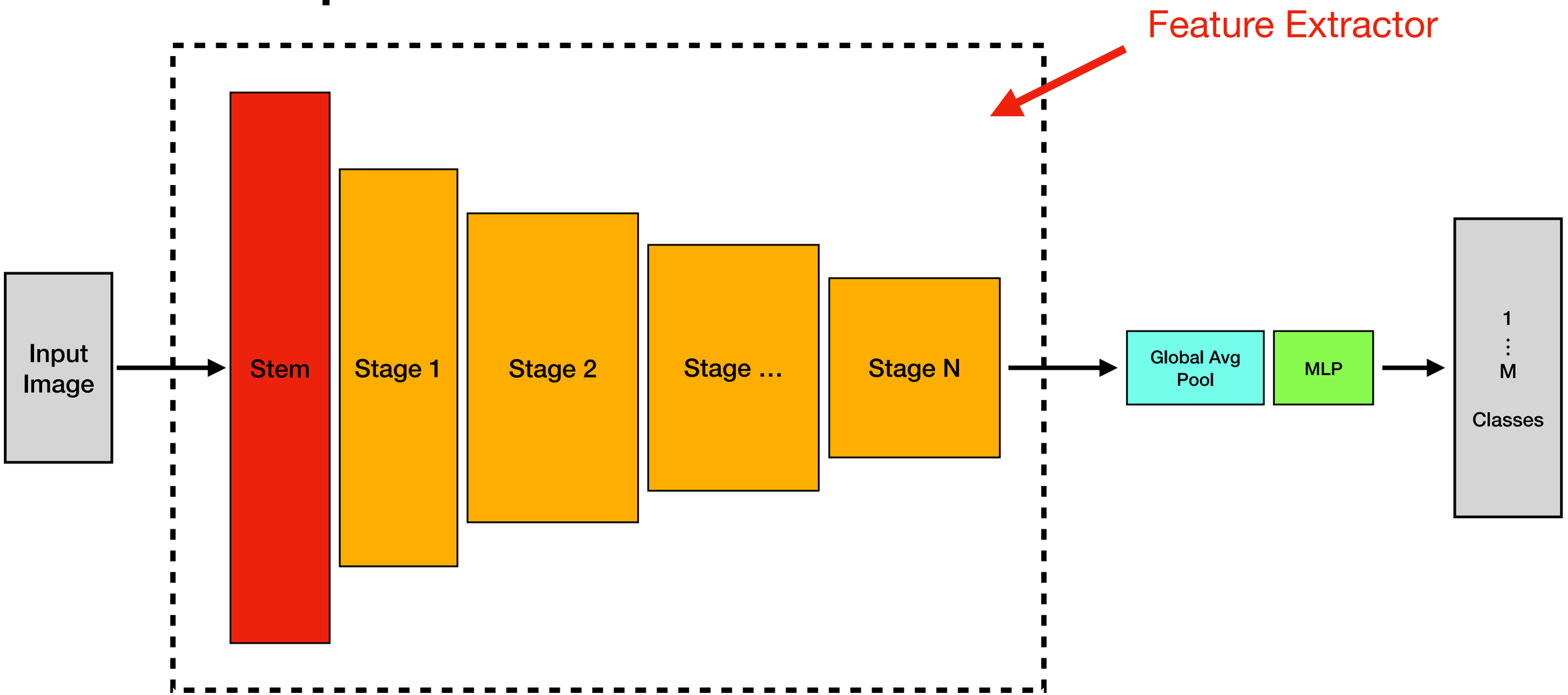






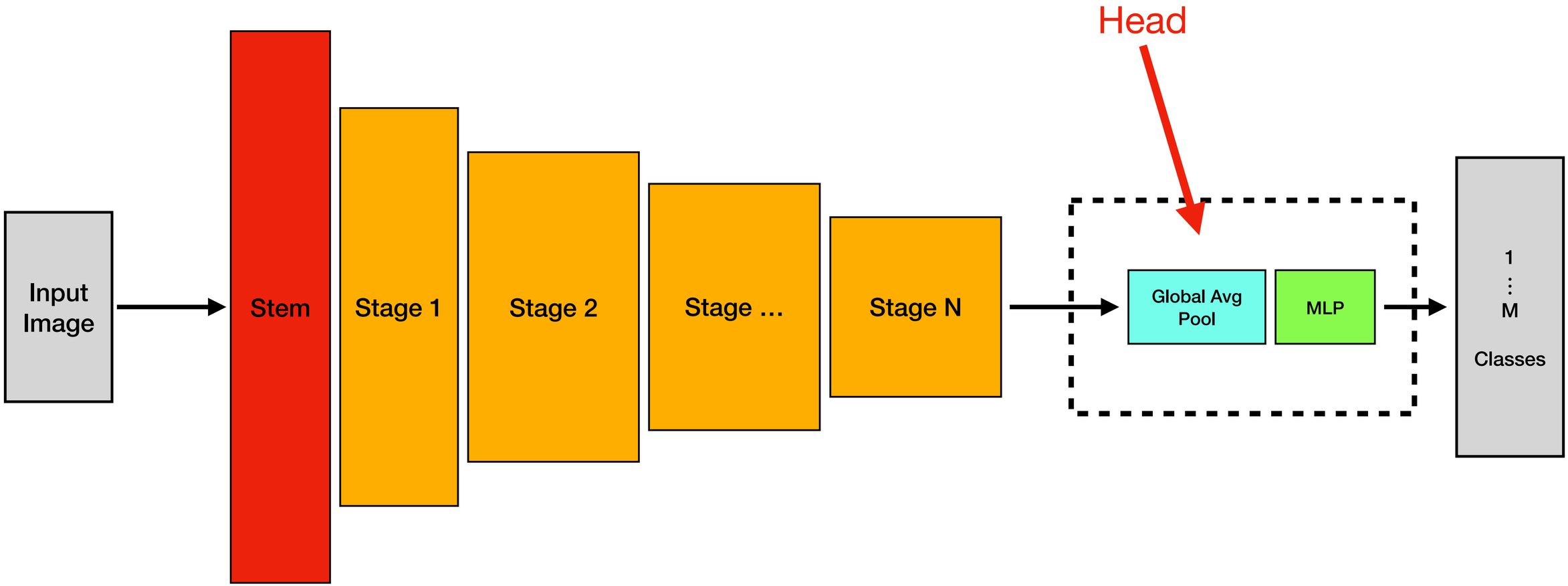

Using a Pre-Trained Model

Generic Components



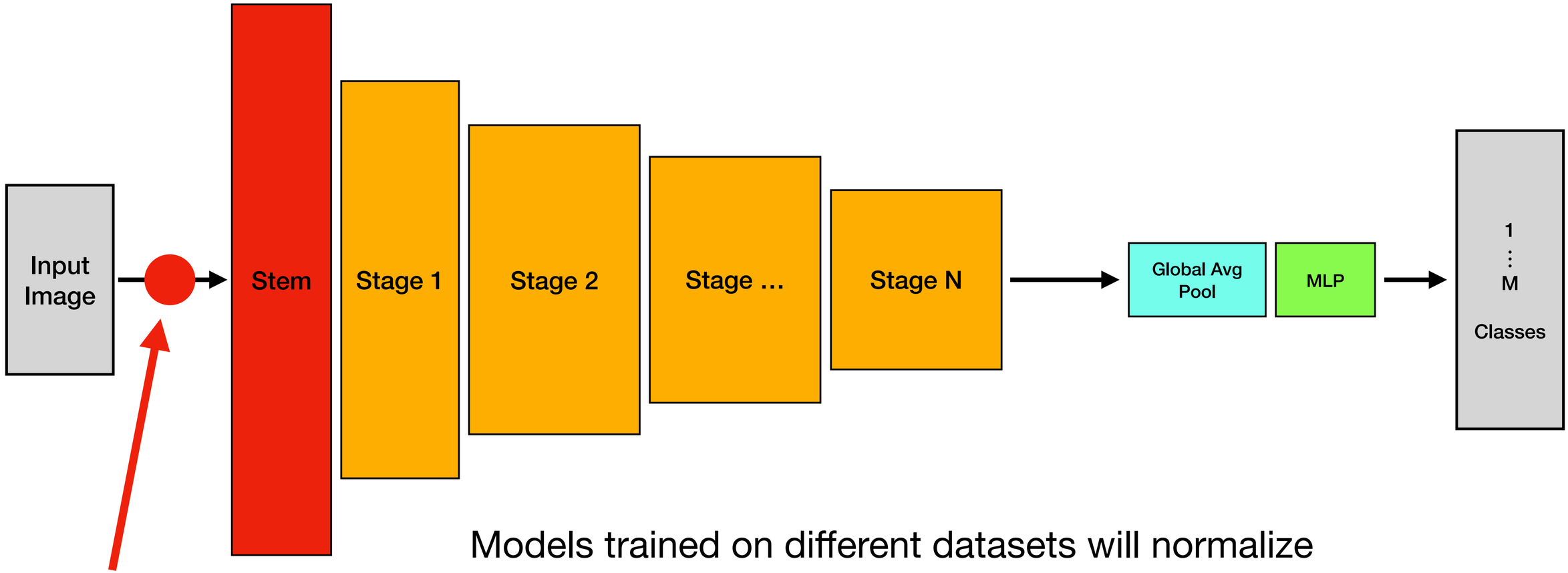
Using a Pre-Trained Model

Generic Components



Using a Pre-Trained Model

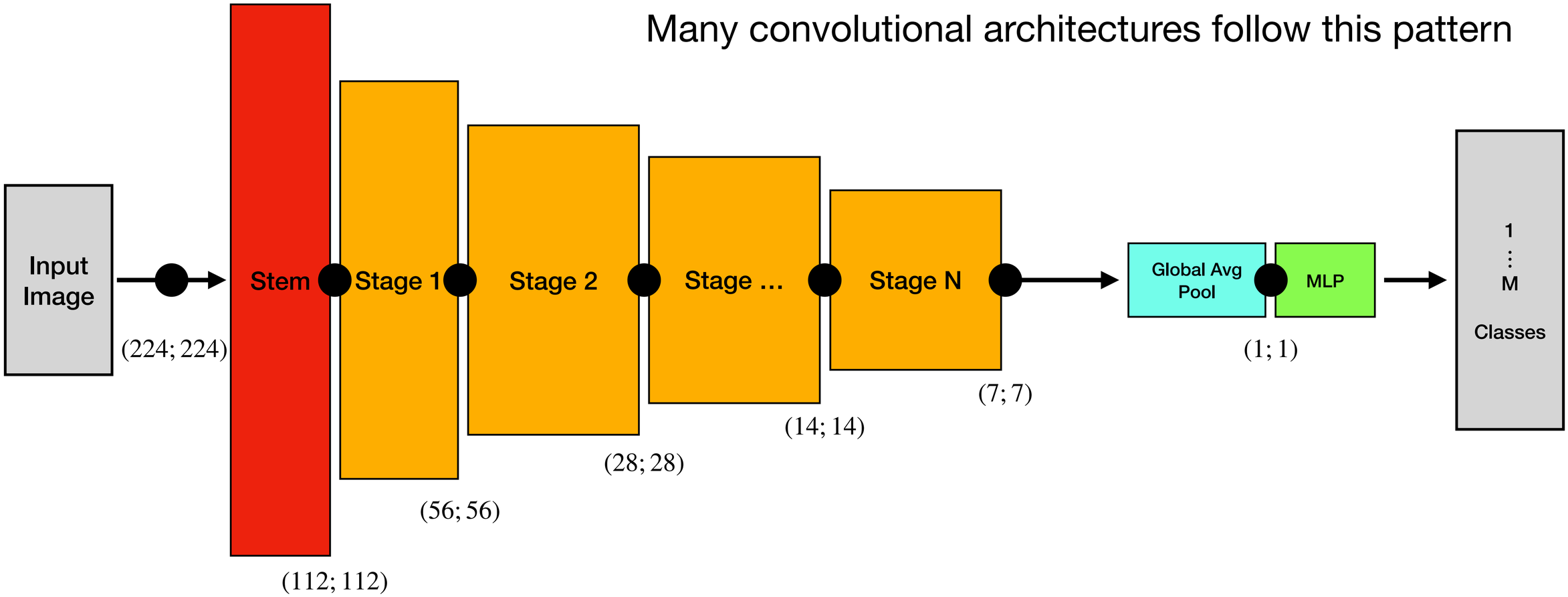
Generic Components



Models trained on different datasets will normalize inputs with different mean and standard deviation !

Using a Pre-Trained Model

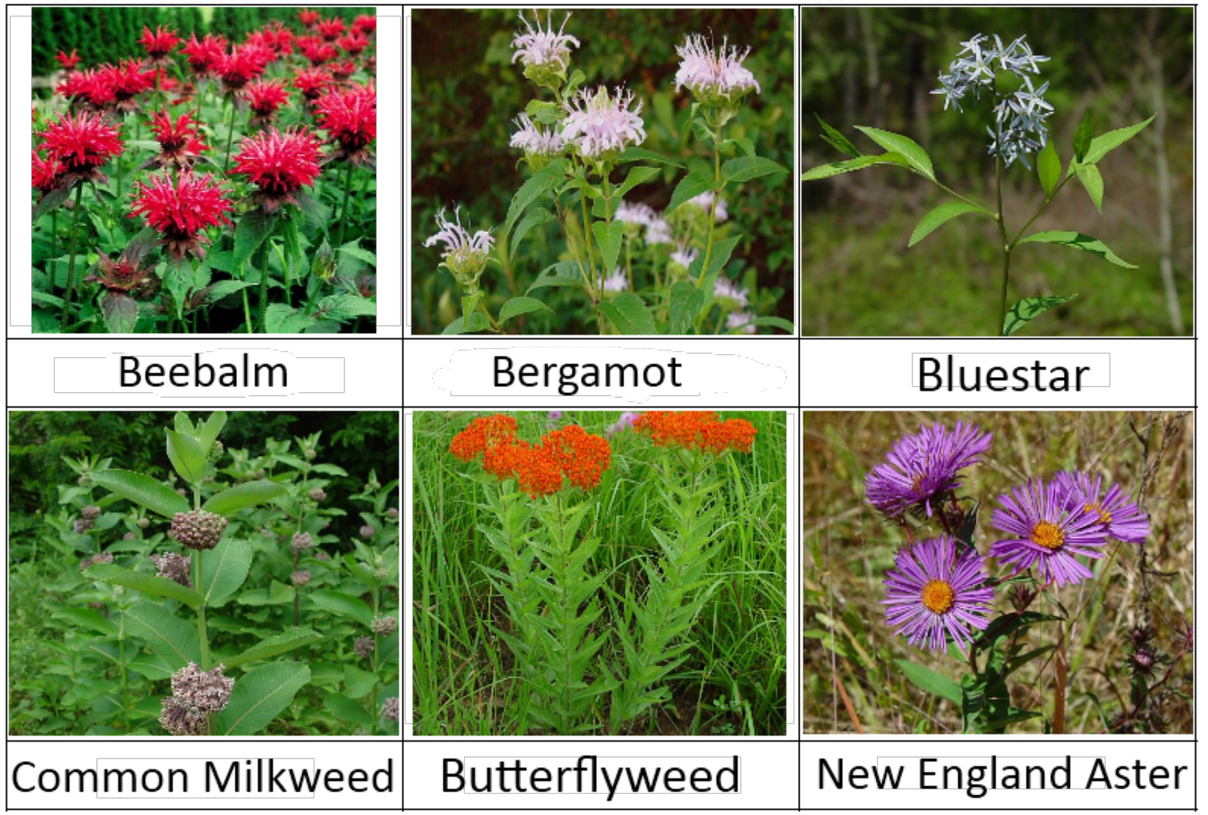
Generic Components - Spatial Dimensions



Using a Pre-Trained Model

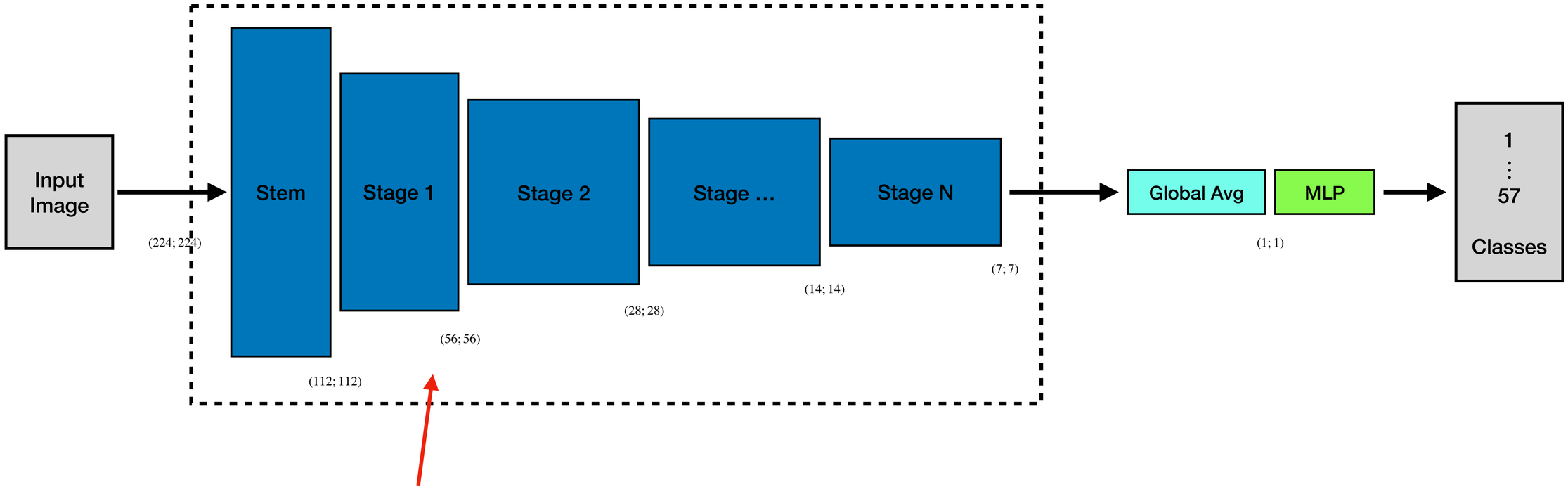
Transfer Learning - Identify Plant Species

- Goal
  - Identify plant species
- Dataset
  - 57 classes
  - Few thousands images



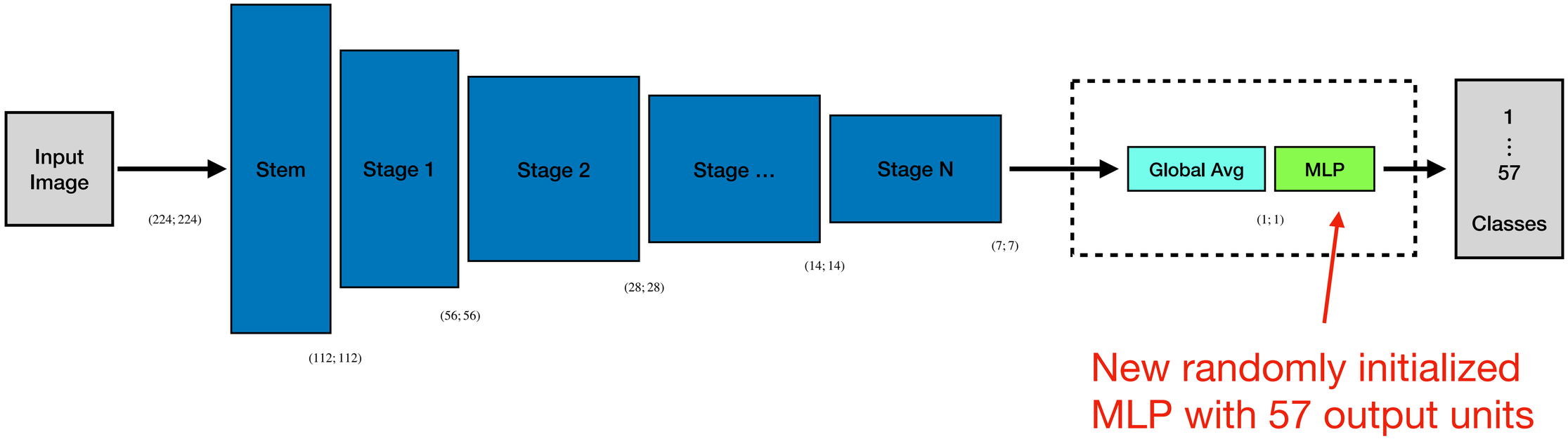
Using a Pre-Trained Model

Transfer Learning - Identify Plant Species



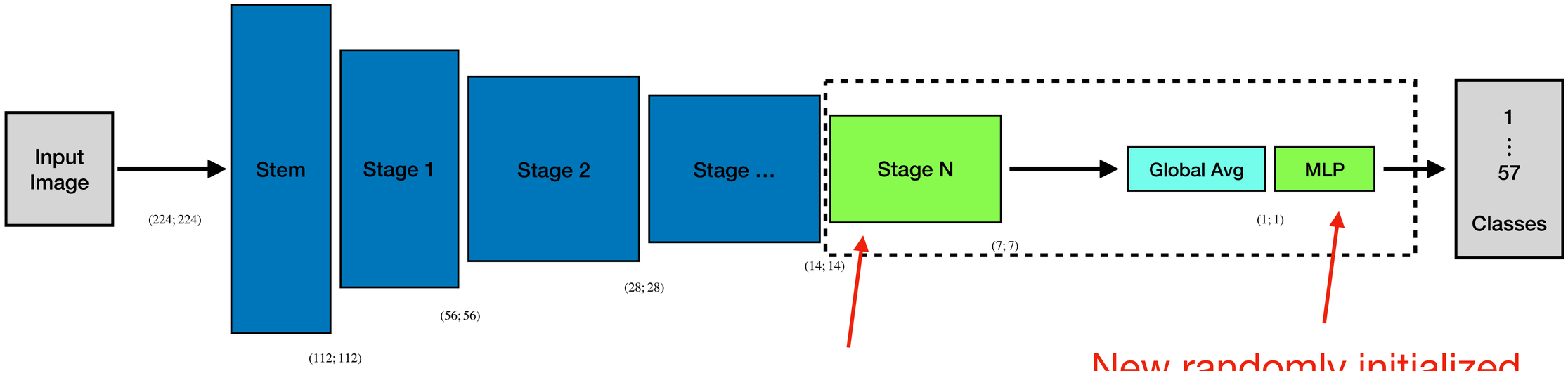
Using a Pre-Trained Model

Transfer Learning - Identify Plant Species



Using a Pre-Trained Model

Transfer Learning - Identify Plant Species



Augmentations

- **Always consider** the possibility of augmenting your data
  - Make **training set** bigger, add more variability
  - Cover the edge cases

- **Synthetic data !**
  - We will talk about them later

Augmentations

Convolution

- Is translation-invariant
- **Is not** rotation-invariant
- **Is not** scale-invariant

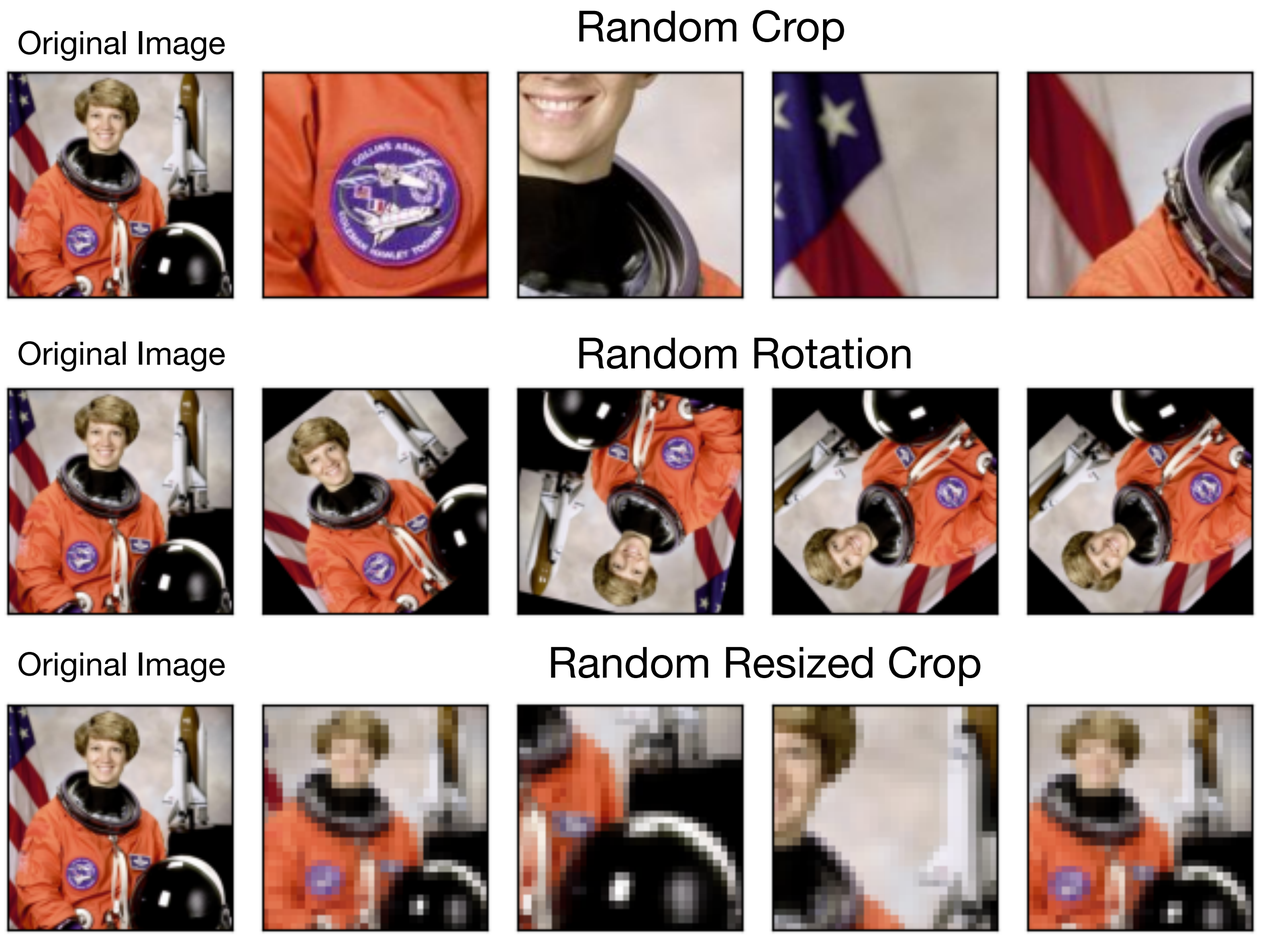


Image - https://pytorch.org/vision/stable/auto_examples/transforms/plot_transforms_illustrations.html

Augmentations

- Perspective
- Image deformations

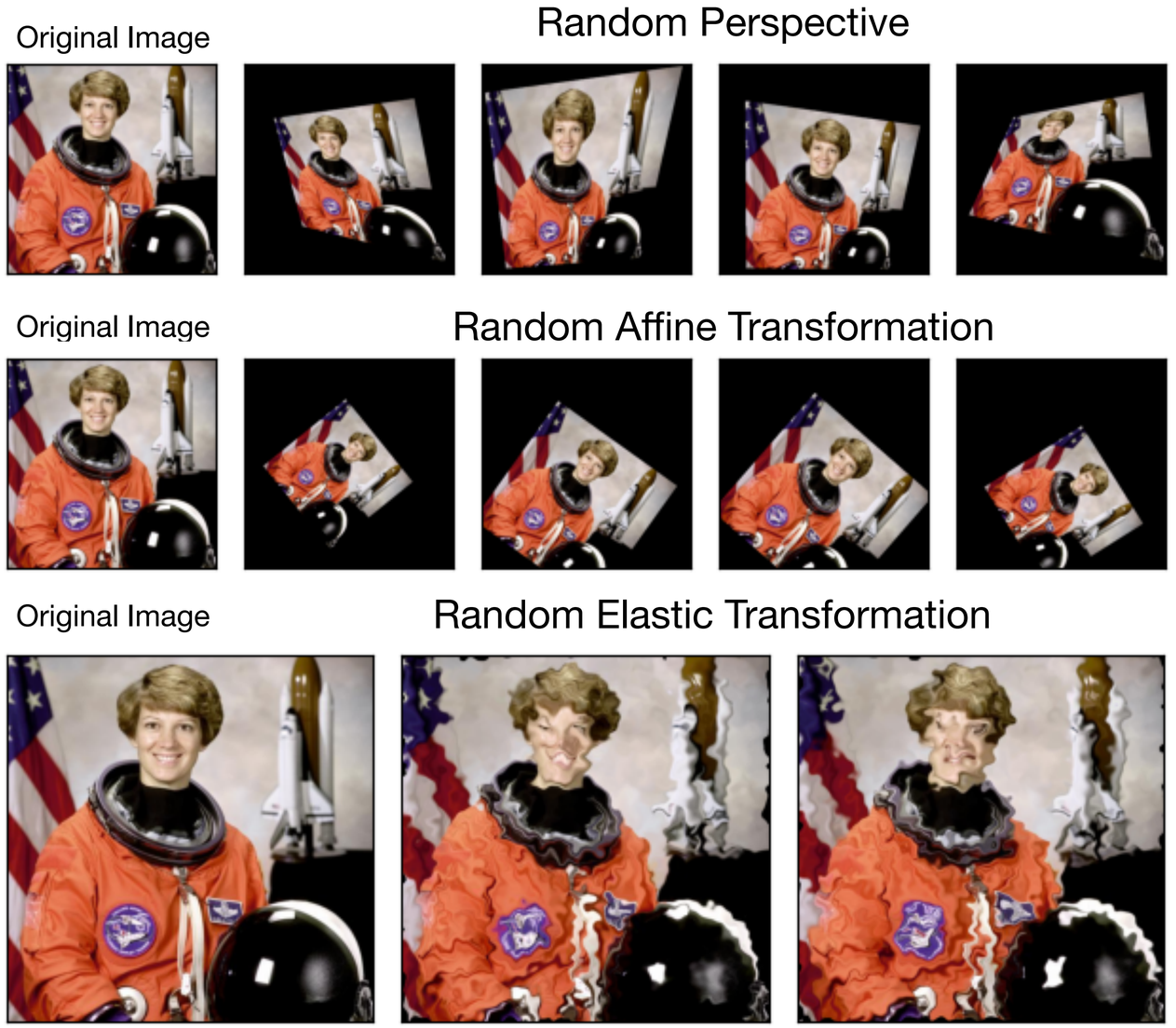


Image - https://pytorch.org/vision/stable/auto_examples/transforms/plot_transforms_illustrations.html

Augmentations

- Color
- Contrast
- Saturation
- Brightness
- Color channel bit-depth

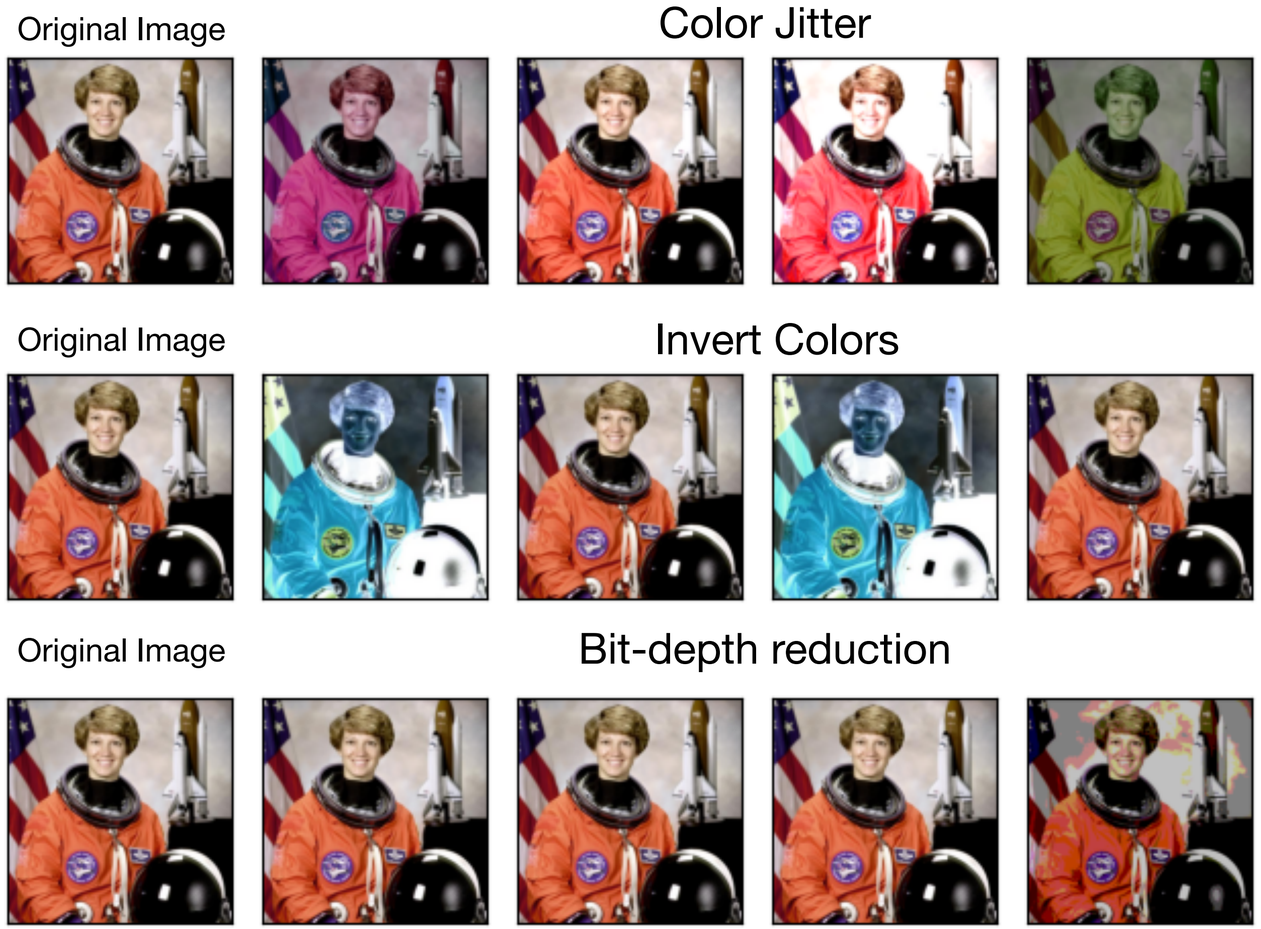


Image - https://pytorch.org/vision/stable/auto_examples/transforms/plot_transforms_illustrations.html

Augmentations

Corrupt Information

- Noise
- Blur
- Cut-out
- Compression

Image - Augmented from CelebA Spoof Dataset - https://mmlab.ie.cuhk.edu.hk/projects/CelebA/CelebA_Spoof.html

Augmentations

Histogram Matching

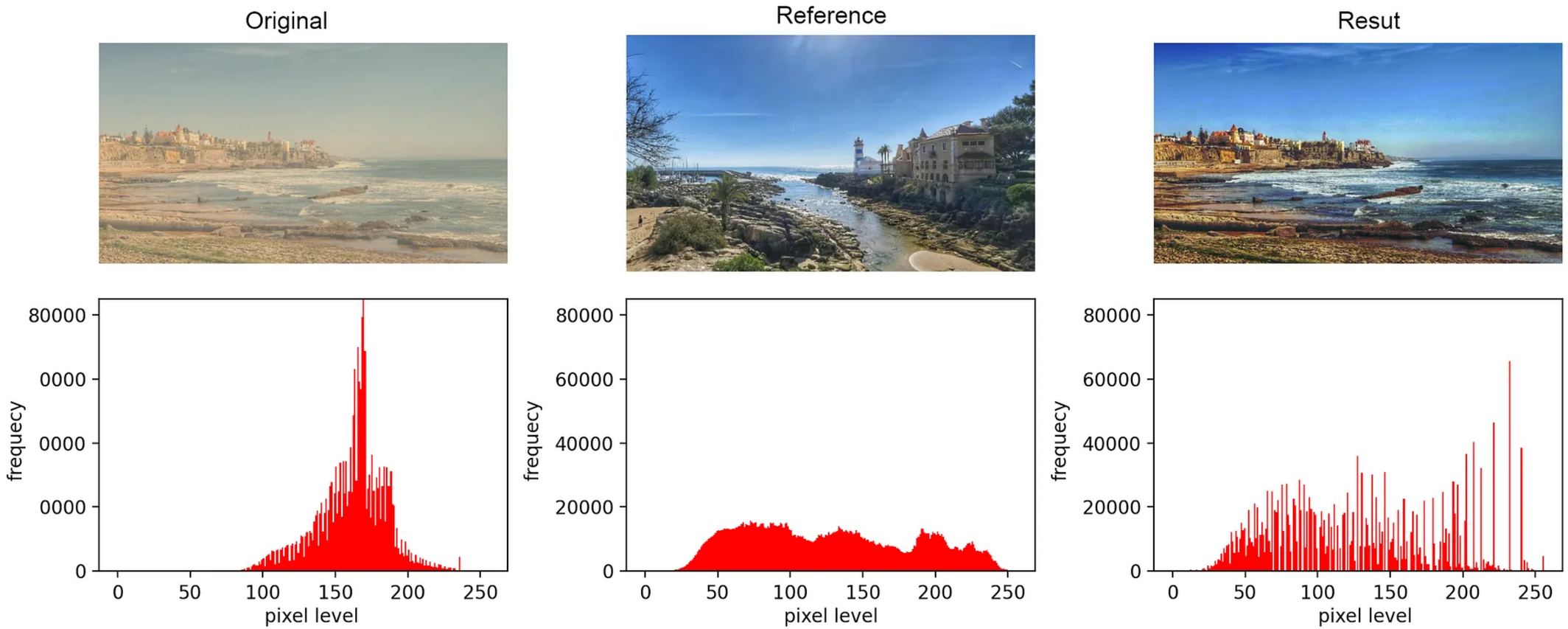


Image -https://medium.com/@er_95882/improving-images-using-equalisation-and-histogram-matching-with-python-c68ca78316e9

3 Siamese Networks

Siamese Networks

- Image similarity
- Written signature verification
- Face recognition / verification
- …

Image Similarity

Image - https://ioannotator.com/image-similarity-search

Image Similarity

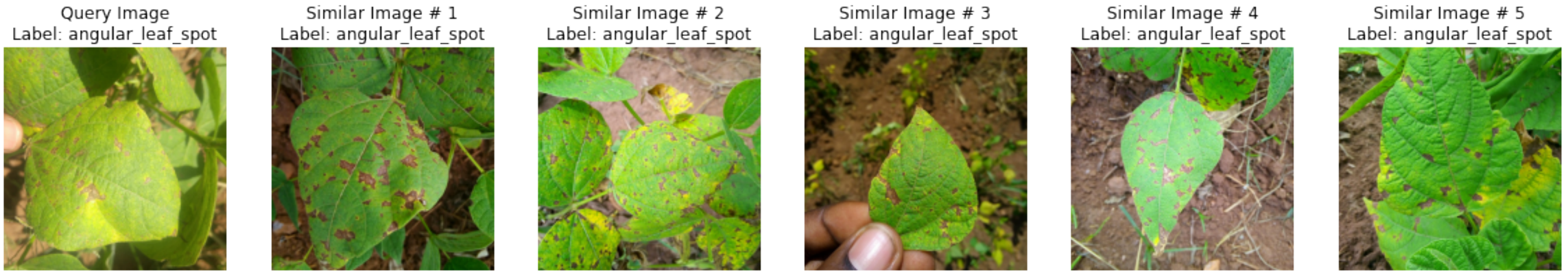


Image - https://huggingface.co/blog/image-similarity

Image Similarity

Which Images Are Similar ?

Images - Google

Image Similarity

Vector Similarity Metrics

- **Euclidean distance**

  In multi-dimensional space all vectors are often “far away” from each other

$$d(a, b) = \|a - b\|$$

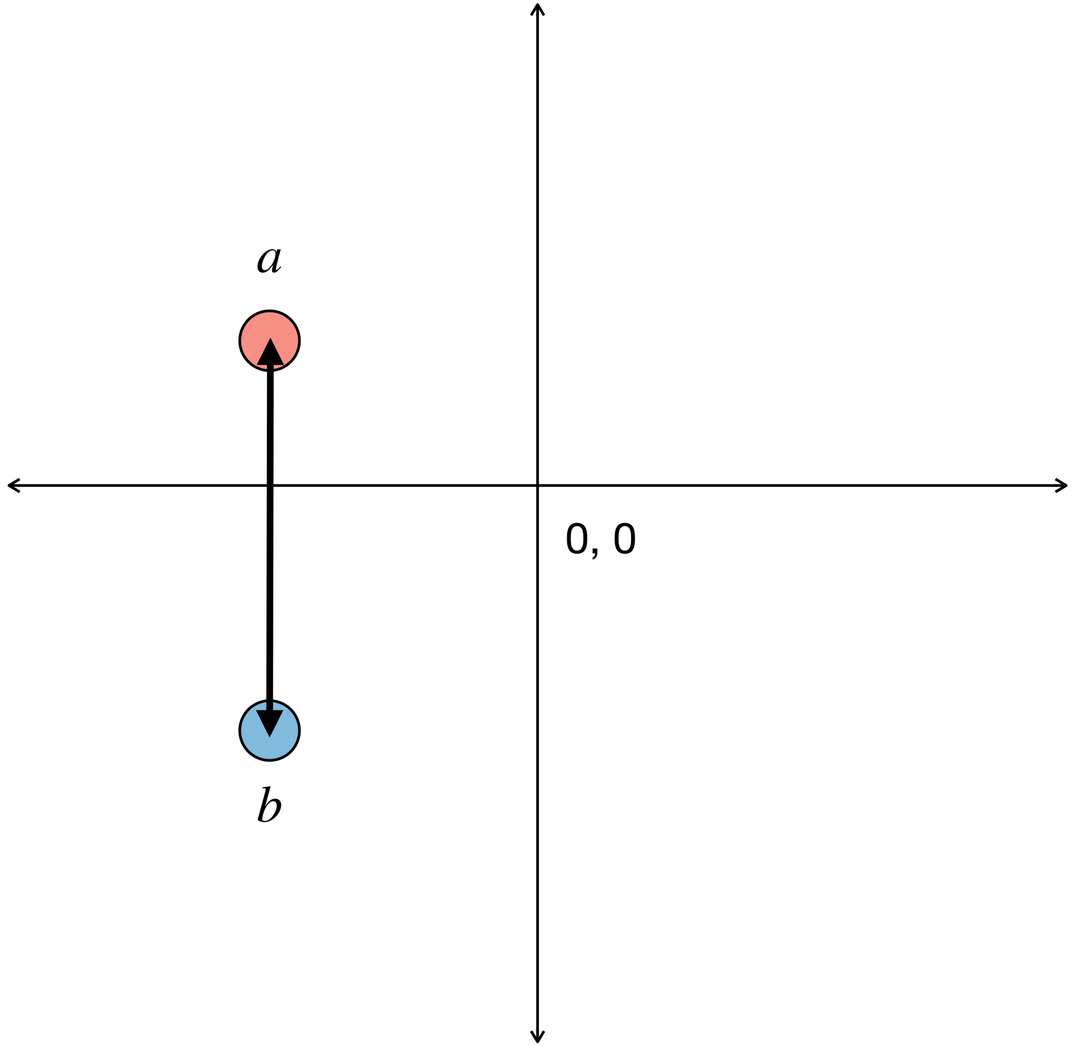
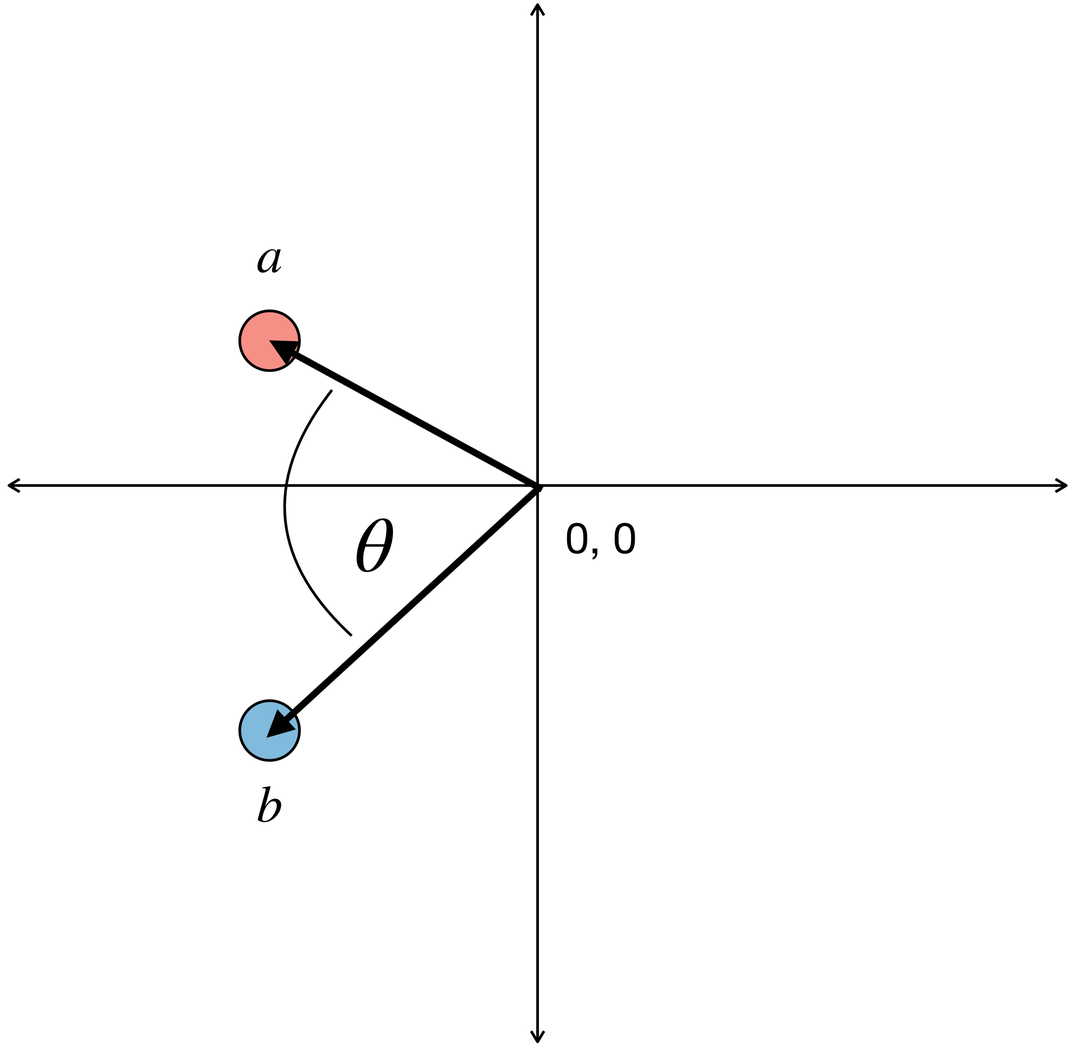


Image Similarity

Vector Similarity Metrics

- Euclidean distance
- **Cosine similarity**



$$S_C(a,b) = \cos\theta = \frac{a \,.\, b}{\|a\|\,\|b\|}$$

$$D_C(a,b) = 1 - S_C(a,b)$$ ← **Cosine distance**

Image Similarity

Vector Similarity Metrics

- Euclidean distanc
- **Cosine similarity**

For normalized vectors

$$S_C(a,b) = a \,.\, b$$

$$S_C(a,b) = \cos\theta = \frac{a \,.\, b}{\|a\|\,\|b\|}$$

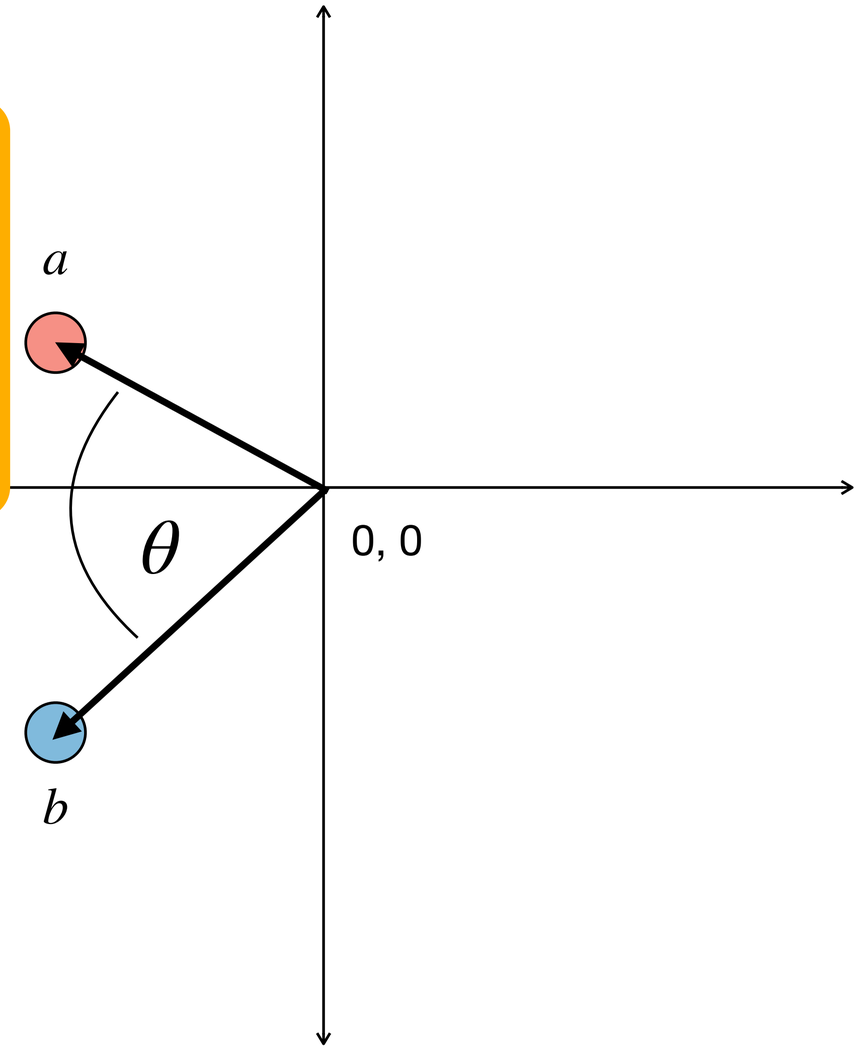
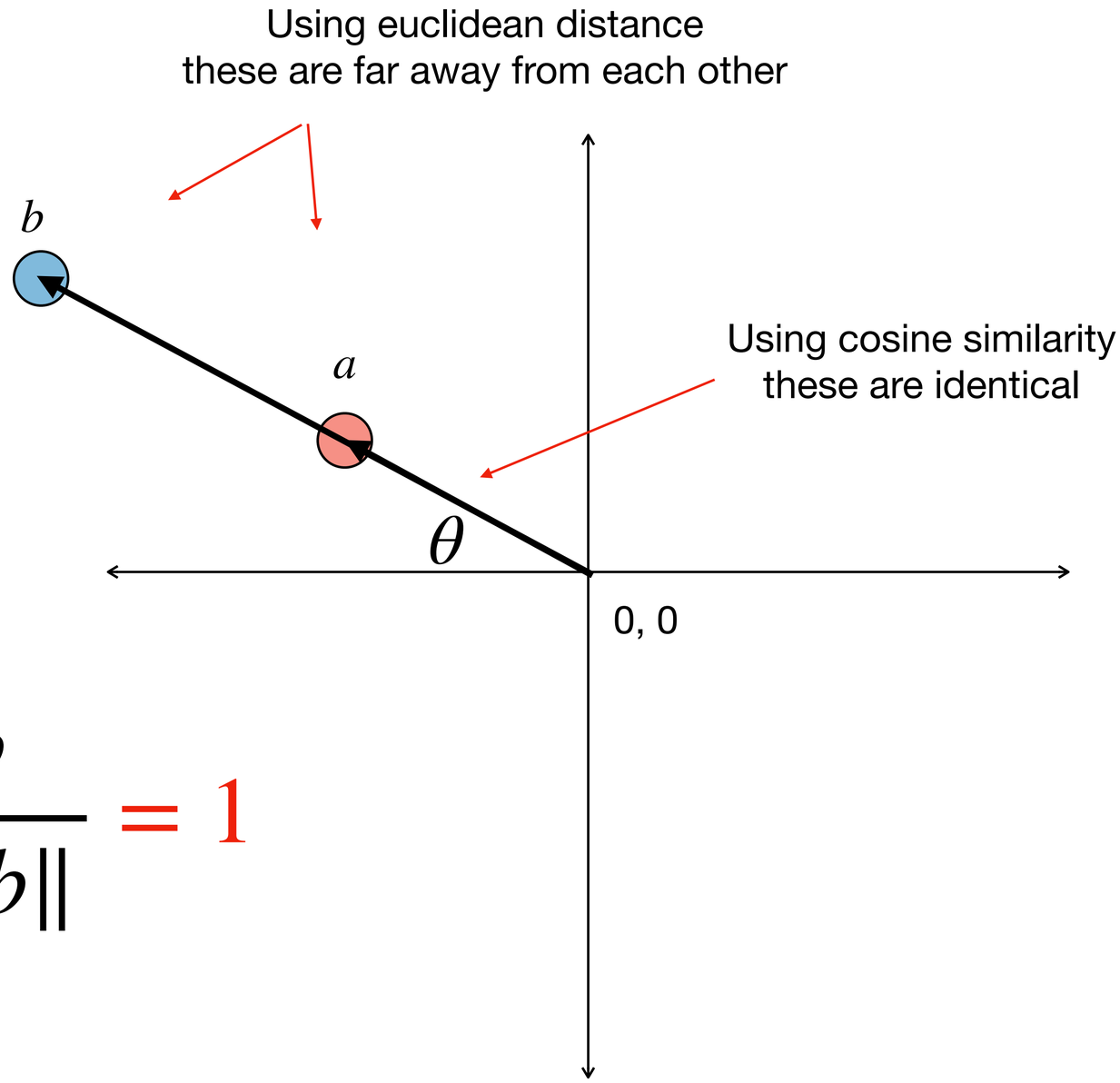


Image Similarity

Vector Similarity Metrics

- Euclidean distance
- **Cosine similarity**



$$S_C(a,b) = \cos\theta = \frac{a \,.\, b}{\|a\|\,\|b\|} = 1$$

Image Similarity

Vector Similarity Metrics

- Euclidean distance
- **Cosine similarity**

$$S_C(a, b) = \cos\theta = \frac{a \,.\, b}{\|a\| \, \|b\|} = 0$$

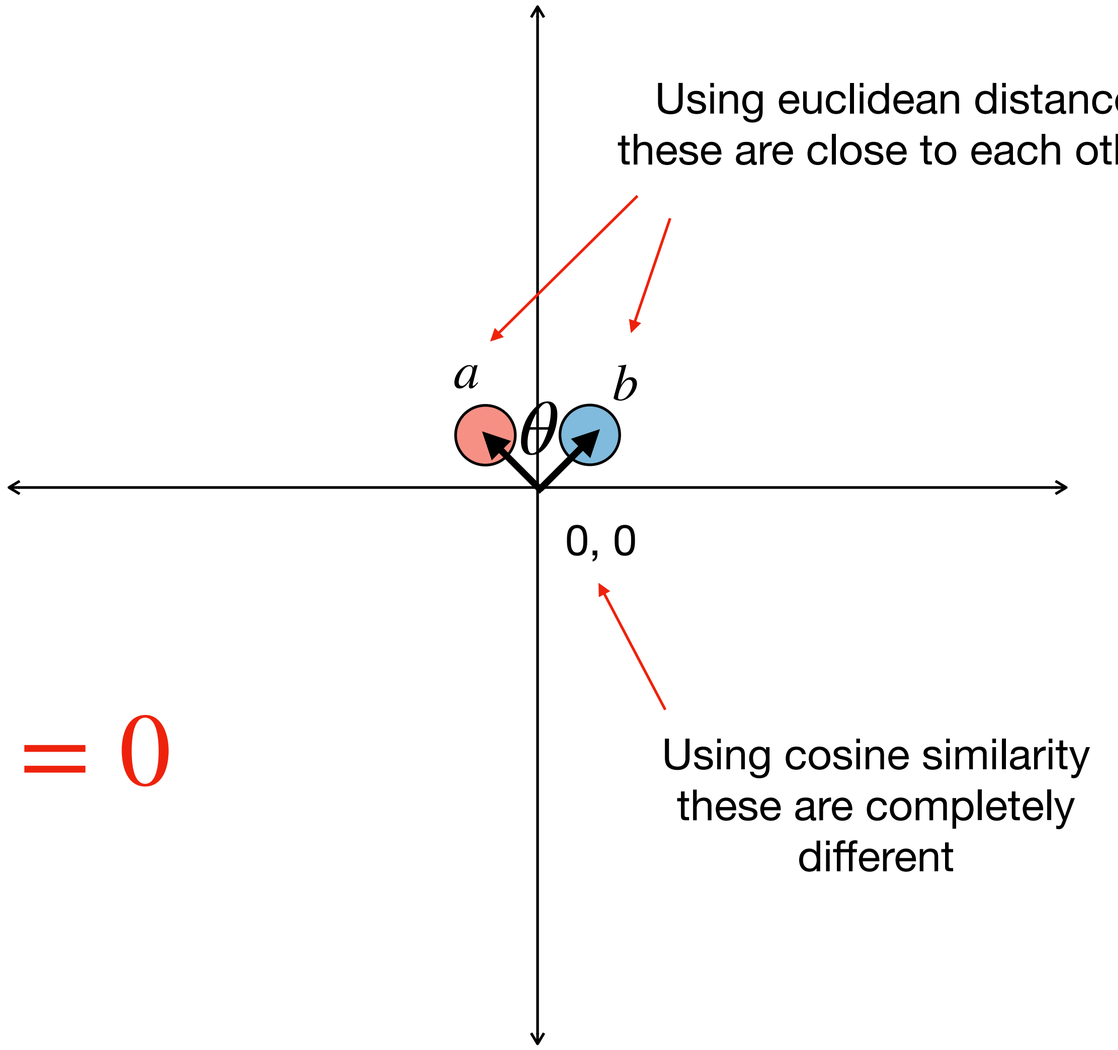


Image Similarity

Cosine Similarity Loss

- If we want two vectors to be **similar**

$$\mathscr{L}(a,b) = 1 - S_C(a,b)$$

$$= 1 - \frac{a \,.\, b}{\|a\|\|b\|}$$

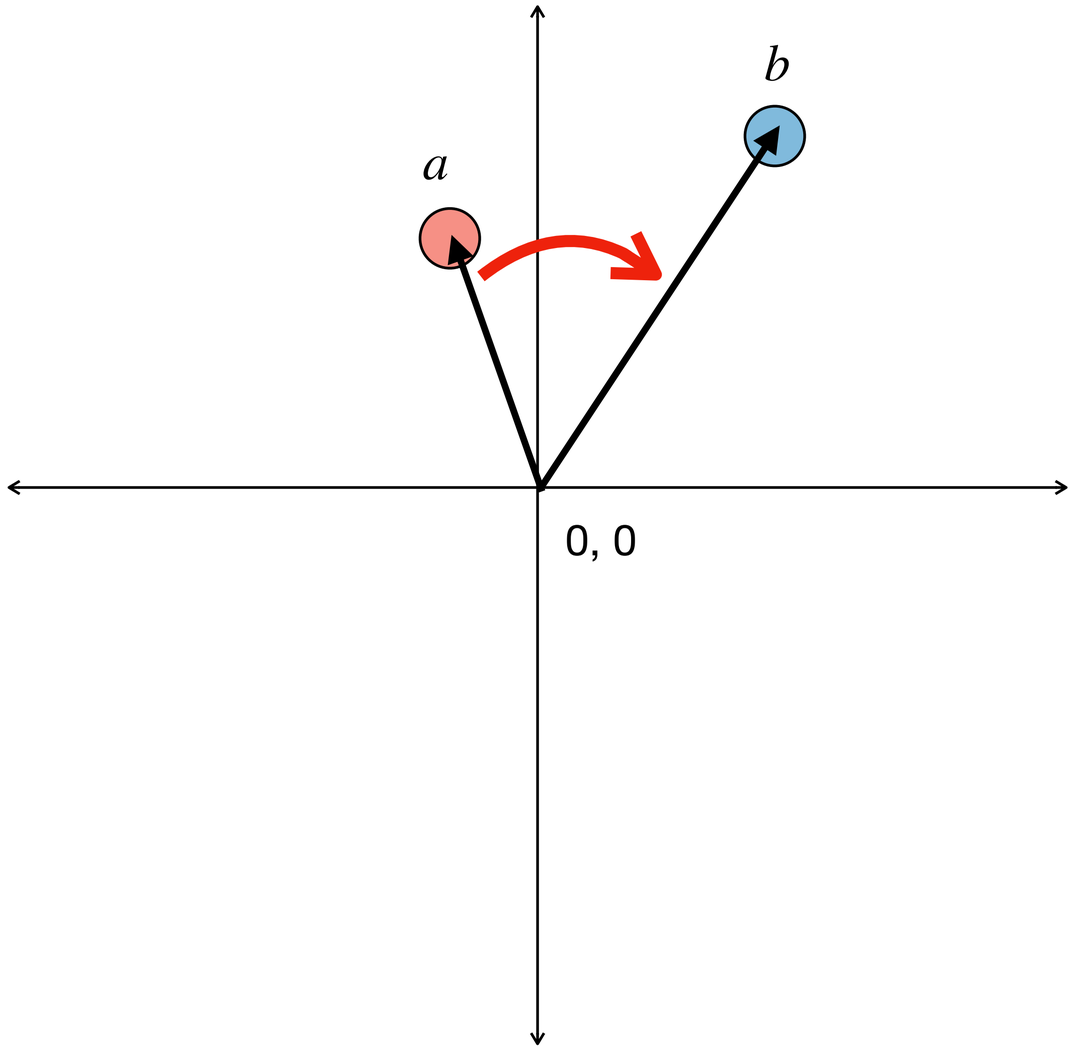
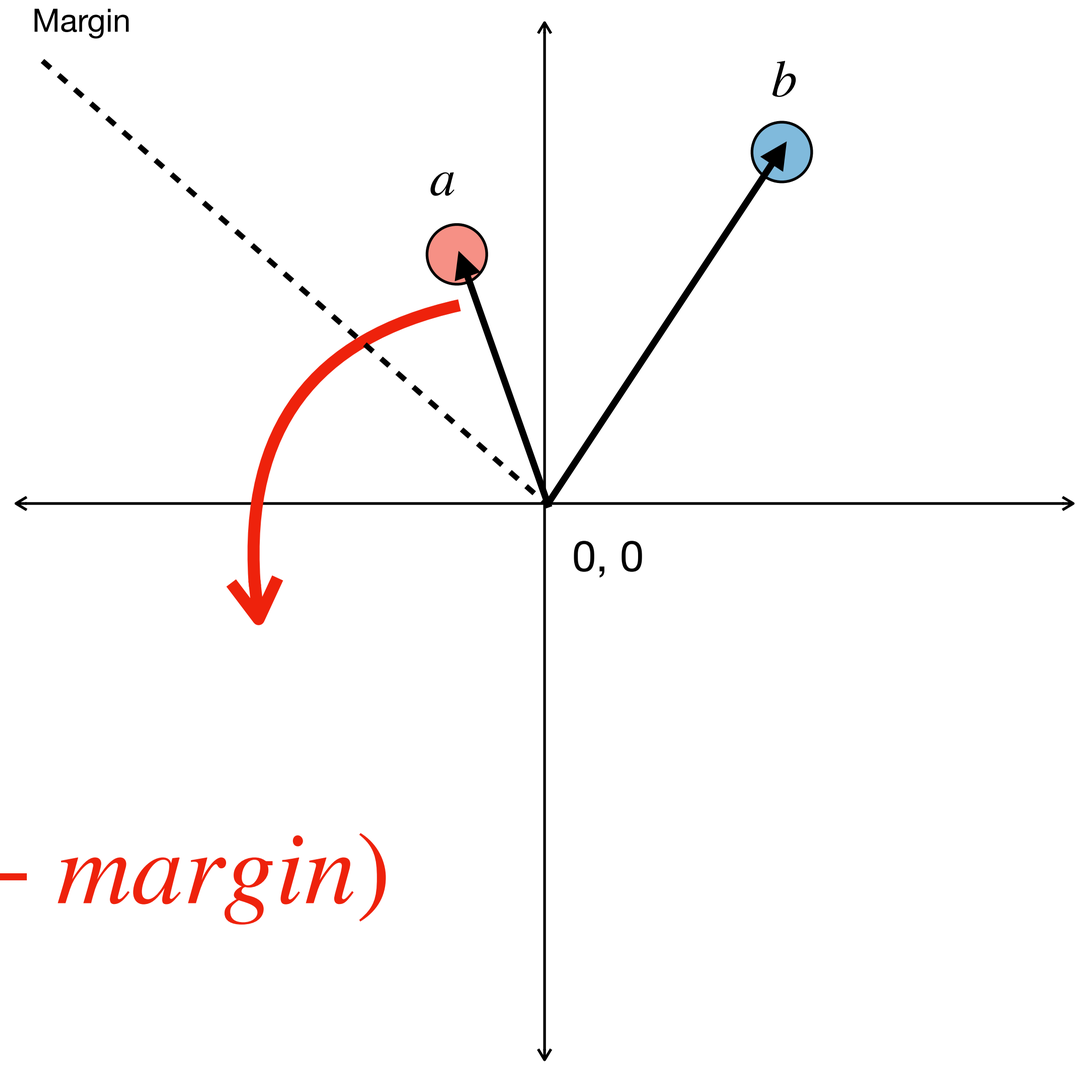


Image Similarity

Cosine Similarity Loss

- If we want two vectors to be **different**



$$\mathscr{L}(a, b) = max(0, S_C(a, b) - margin)$$

Image Similarity

Siamese Networks

CNN
Feature Extractor

Shared
Weights

CNN
Feature Extractor

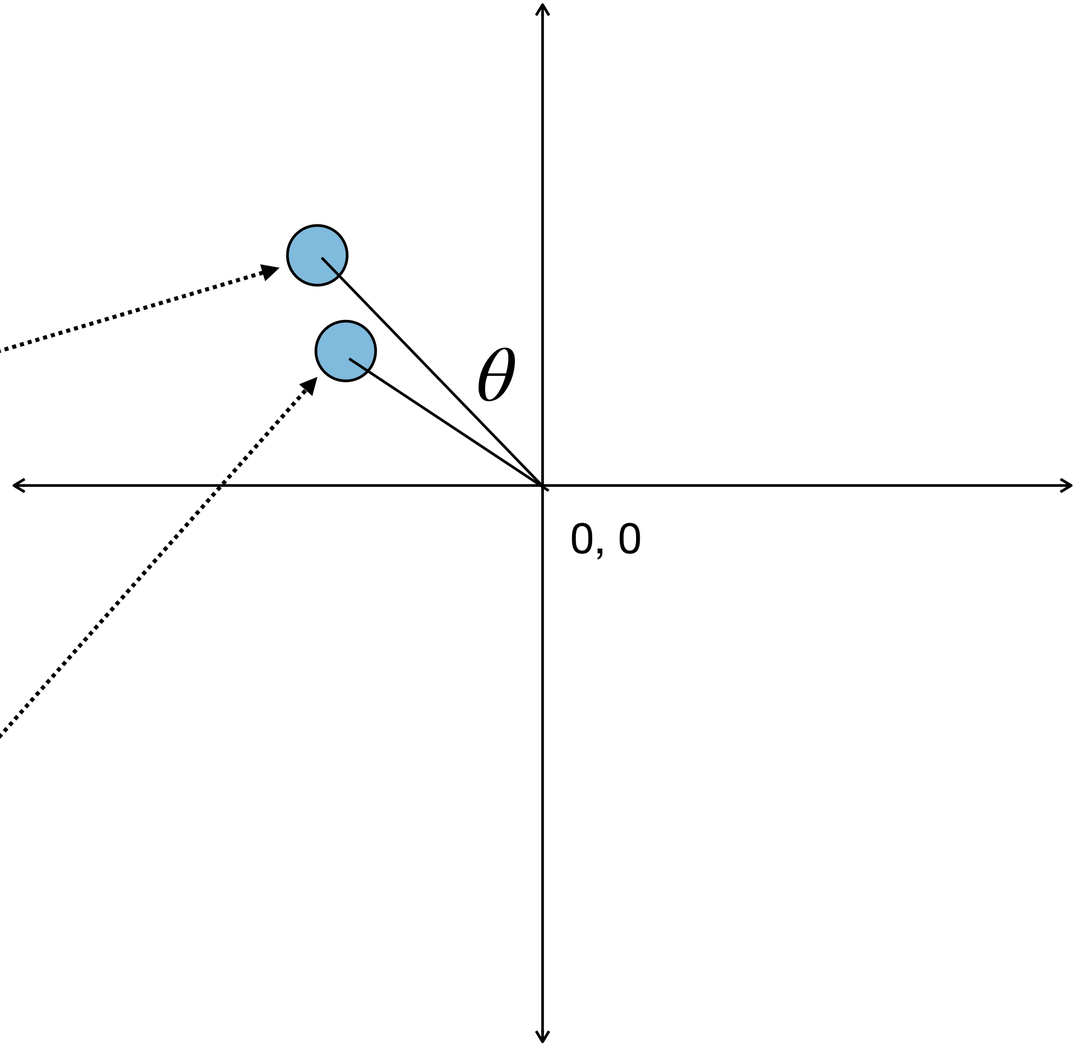


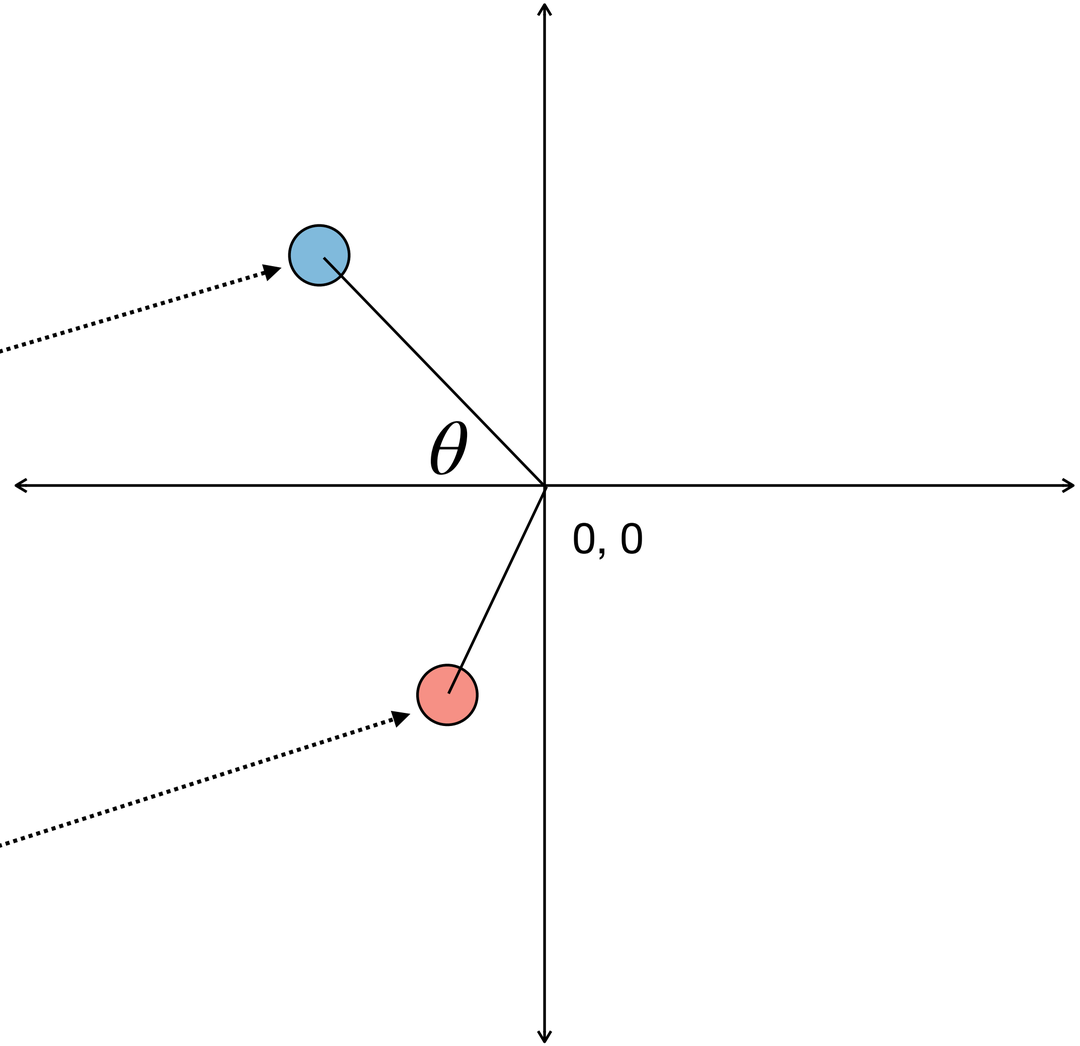
Image Similarity

Siamese Networks

CNN
Feature Extractor

Shared
Weights

CNN
Feature Extractor



Contrastive Loss

Siamese Networks

Hadsell, Raia, Sumit Chopra, and Yann LeCun. "Dimensionality reduction by learning an invariant mapping." *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*. Vol. 2. IEEE, 2006.

$$x = \{ \quad \}$$

Big bag of images

Contrastive Loss

Siamese Networks

Hadsell, Raia, Sumit Chopra, and Yann LeCun. "Dimensionality reduction by learning an invariant mapping." *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*. Vol. 2. IEEE, 2006.

$$\mathcal{X} = \{ \quad \}$$

x_1, x_2	y
	0
	1
	1
	0
…	

Combinations of images from $\mathcal{X}$

We must be able to tell if any two images from $\mathcal{X}$ are similar or not

Contrastive Loss

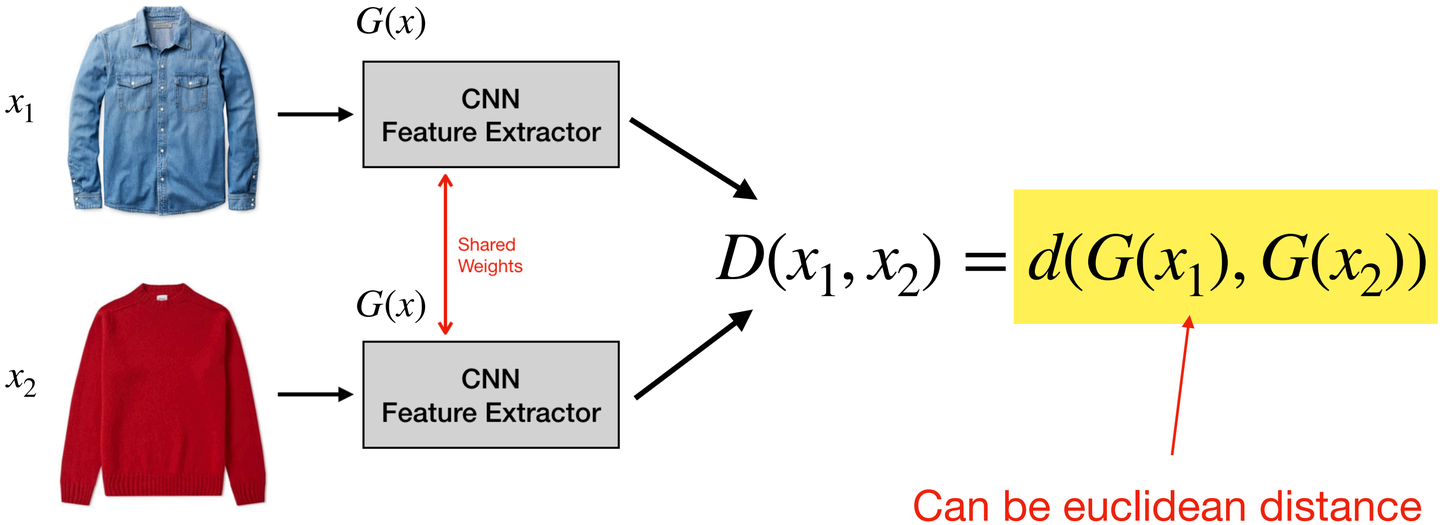
Siamese Networks

Hadsell, Raia, Sumit Chopra, and Yann LeCun. "Dimensionality reduction by learning an invariant mapping." *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*. Vol. 2. IEEE, 2006.

$\mathcal{X} = \{$  $\}$

x_1, x_2	y
	0
	1
	1
	0

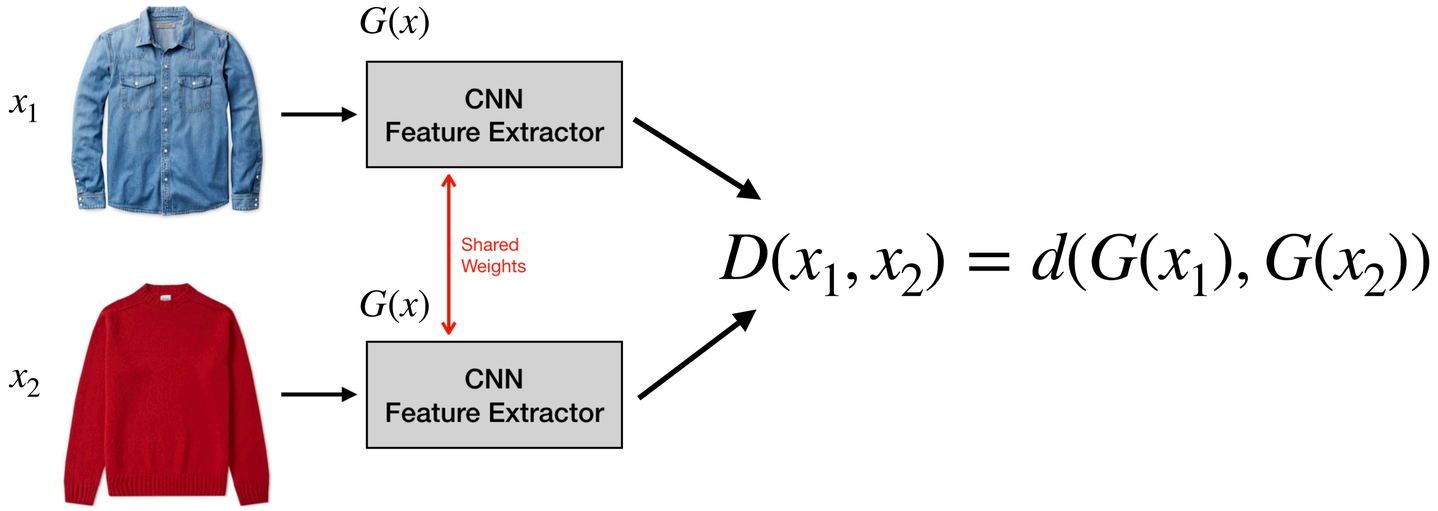
...

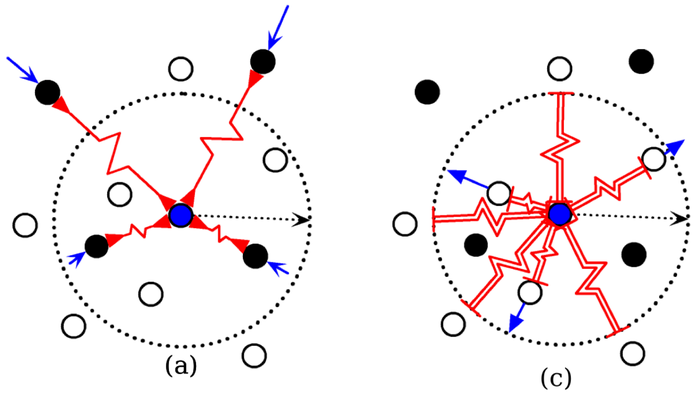


Contrastive Loss

Siamese Networks

Hadsell, Raia, Sumit Chopra, and Yann LeCun. "Dimensionality reduction by learning an invariant mapping." *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*. Vol. 2. IEEE, 2006.





$$\mathscr{L}(x_1, x_2, y) = (1 - y)L_S(x_1, x_2) + (y)L_D(x_1, x_2)$$

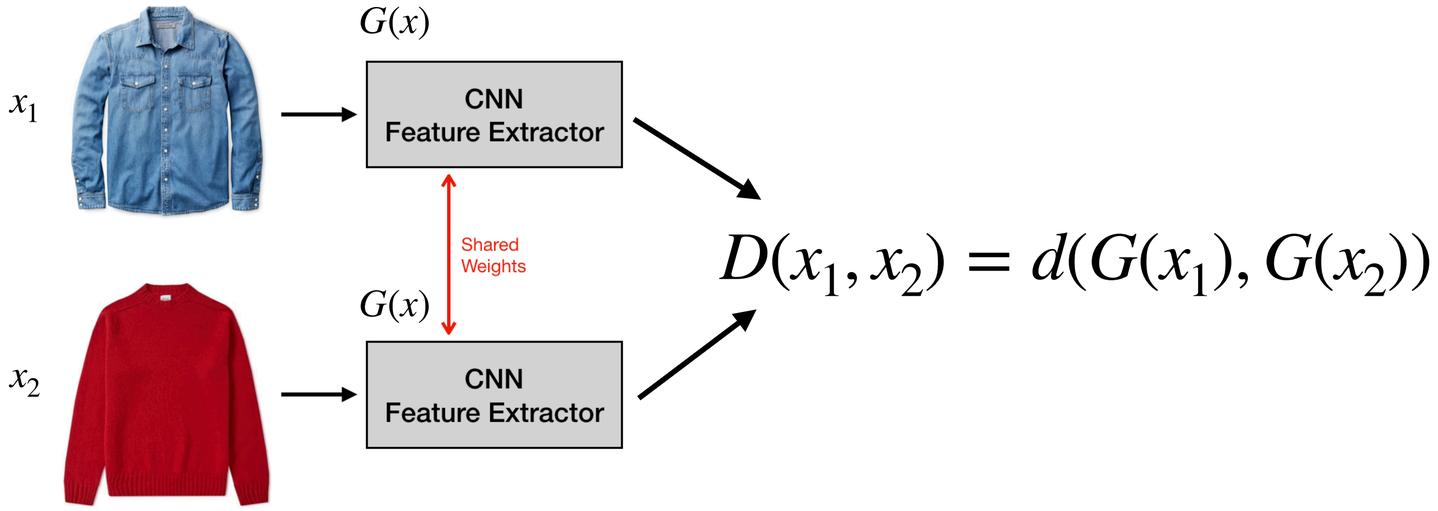
Loss component for similar images

Loss component for dissimilar images

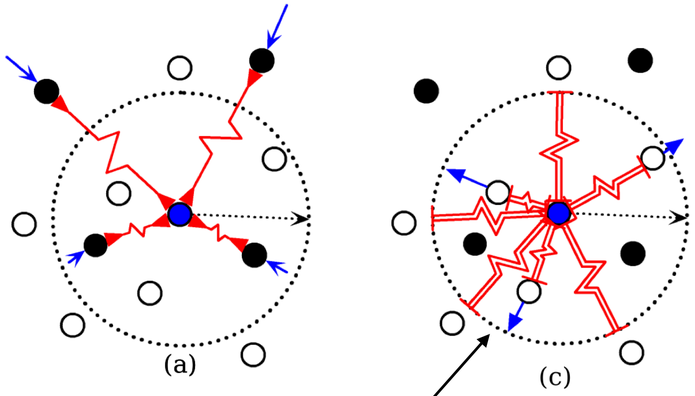
Contrastive Loss

Siamese Networks

Hadsell, Raia, Sumit Chopra, and Yann LeCun. "Dimensionality reduction by learning an invariant mapping." *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*. Vol. 2. IEEE, 2006.



$$D(x_1, x_2) = d(G(x_1), G(x_2))$$



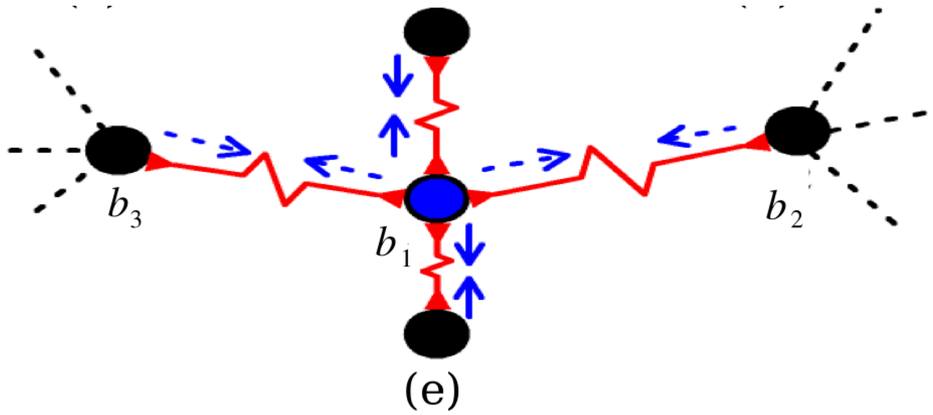
max radius of the repulsive force

$$\mathscr{L}(x_1, x_2, y) = (1-y)L_S(x_1, x_2) + (y)L_D(x_1, x_2)$$

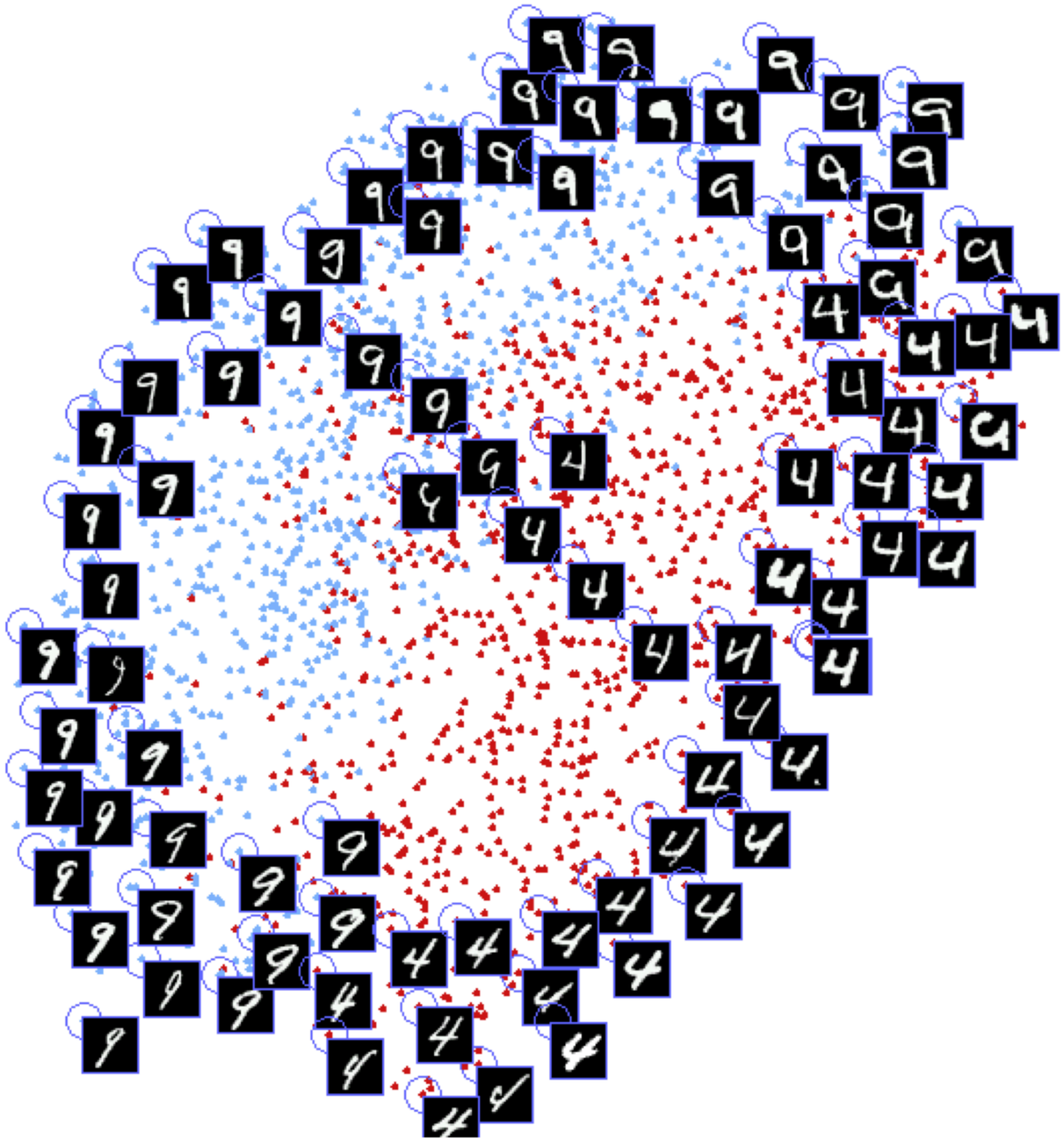
$$\mathscr{L}(x_1, x_2, y) = (1-y)\frac{1}{2}D(x_1, x_2)^2 + (y)\frac{1}{2}\{\max(0, m - D(x_1, x_2))\}^2$$

Contrastive Loss

Siamese Networks



Equilibrium was reached



Hadsell, Raia, Sumit Chopra, and Yann LeCun. "Dimensionality reduction by learning an invariant mapping." *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*. Vol. 2. IEEE, 2006.

Face Recognition

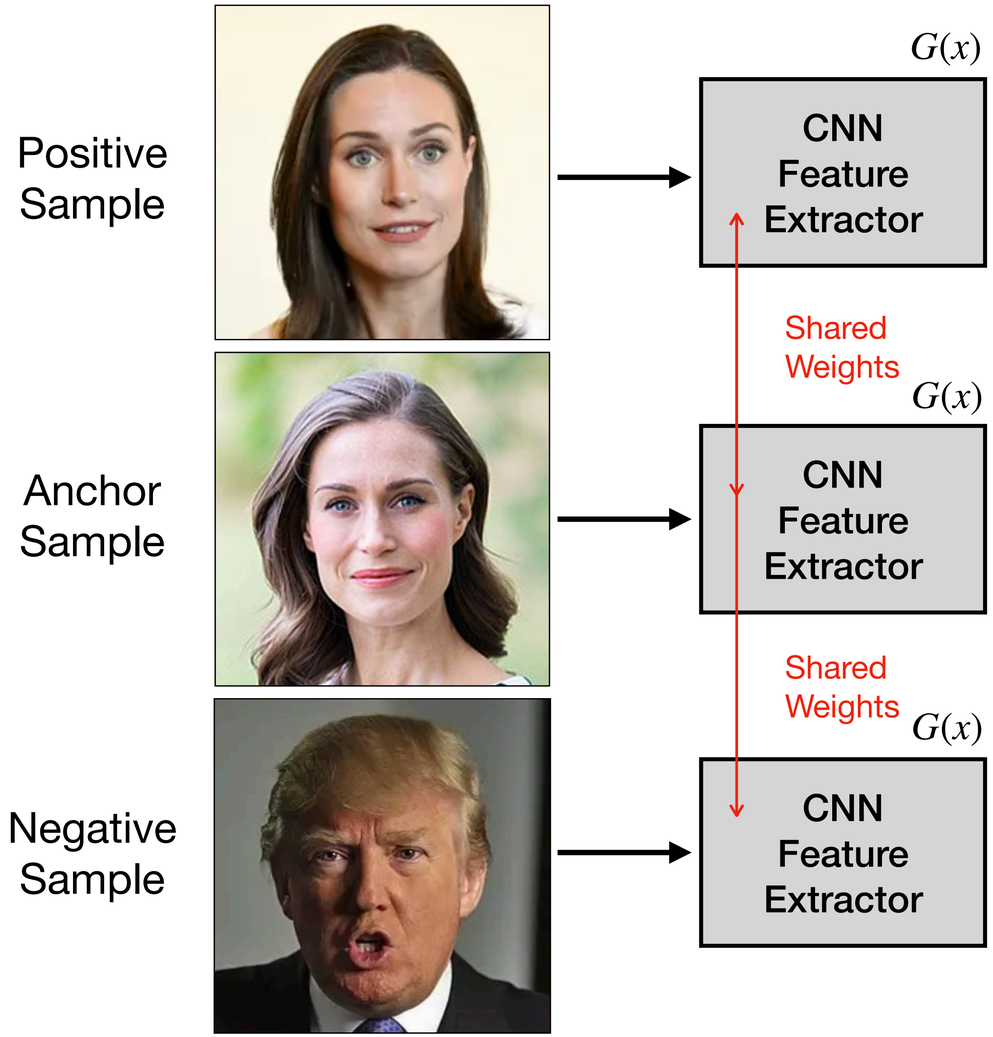
- A case of image similarity
- Large number of classes (Open-set)
    - few samples / class
- **Samples belonging to the same class should be isolated from other classes**

Image - Cao, Qiong, et al. "Vggface2: A dataset for recognising faces across pose and age." 2018 13th IEEE international conference on automatic face & gesture recognition (FG 2018). IEEE, 2018.

Schroff, Florian, Dmitry Kalenichenko, and James Philbin. "Facenet: A unified embedding for face recognition and clustering." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

Face Recognition

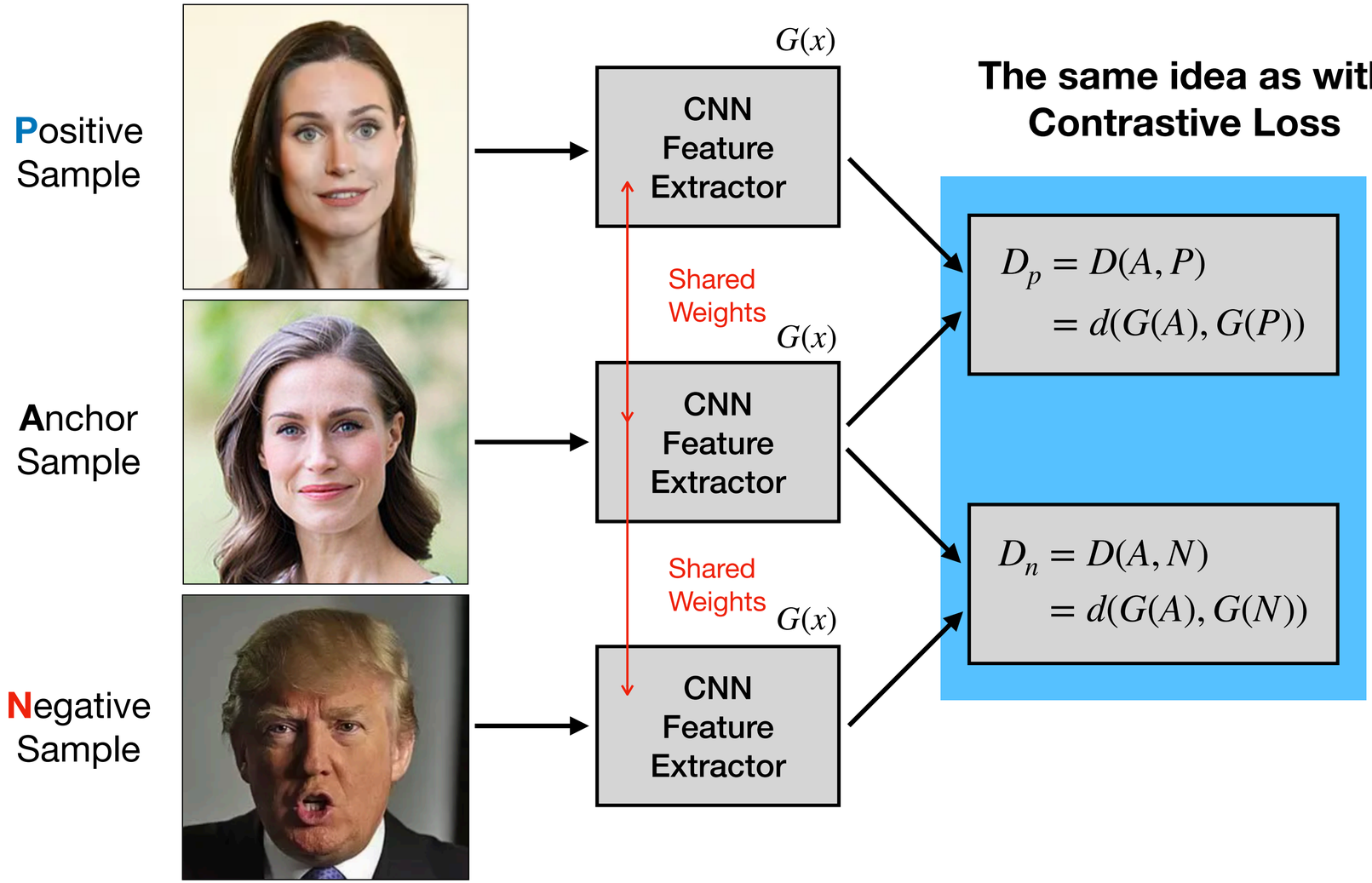
Triplet Loss



Schroff, Florian, Dmitry Kalenichenko, and James Philbin. "Facenet: A unified embedding for face recognition and clustering." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

Face Recognition

Triplet Loss



Schroff, Florian, Dmitry Kalenichenko, and James Philbin. "Facenet: A unified embedding for face recognition and clustering." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

Face Recognition

Triplet Loss

Positive Sample

Anchor Sample

Negative Sample

$G(x)$

CNN Feature Extractor

Shared Weights

$G(x)$

CNN Feature Extractor

Shared Weights

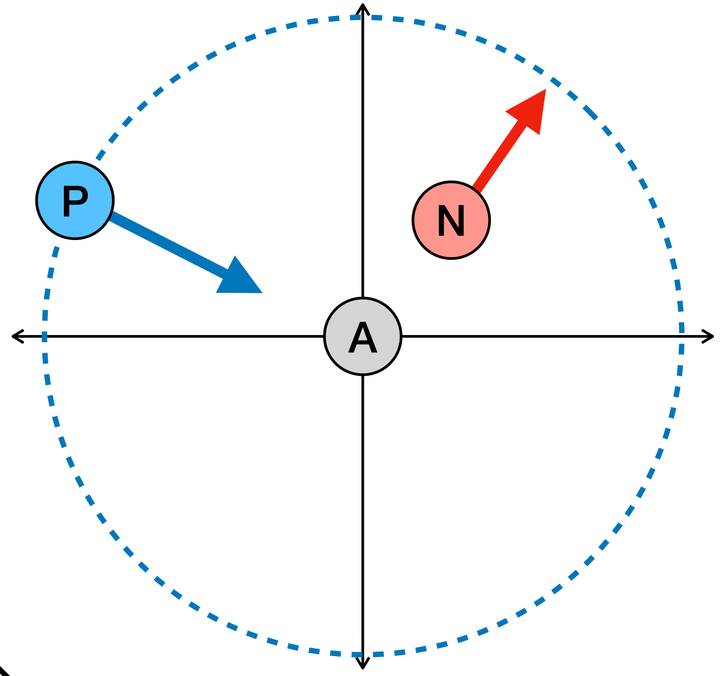
$G(x)$

CNN Feature Extractor

$$D_p = D(A, P) = d(G(A), G(P))$$

$$D_n = D(A, N) = d(G(A), G(N))$$

Triplet Loss



Schroff, Florian, Dmitry Kalenichenko, and James Philbin. "Facenet: A unified embedding for face recognition and clustering." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

Face Recognition

Triplet Loss

Positive Sample

Anchor Sample

Negative Sample

$G(x)$ CNN Feature Extractor

Shared Weights

$G(x)$ CNN Feature Extractor

Shared Weights

$G(x)$ CNN Feature Extractor

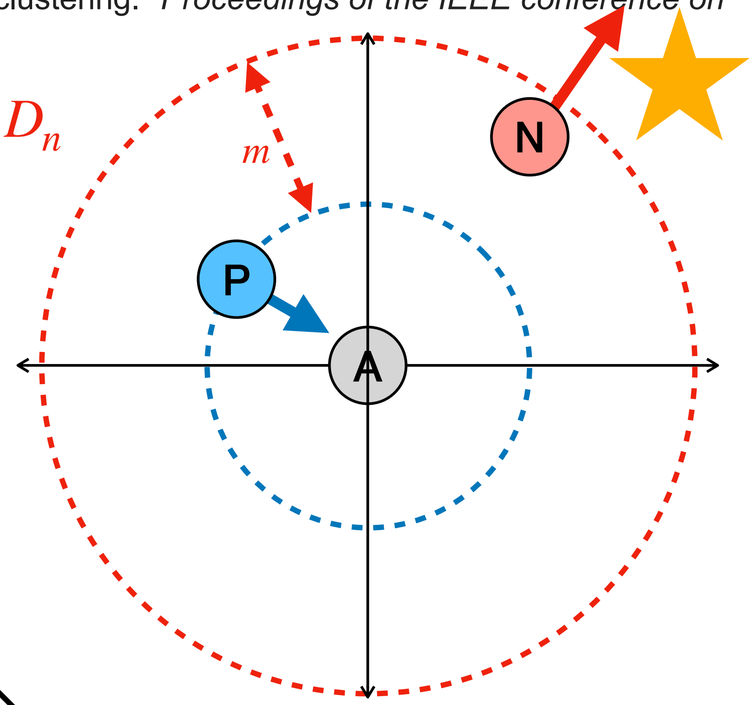
$$D_p = D(A,P) = d(G(A), G(P))$$

$$D_n = D(A,N) = d(G(A), G(N))$$

Triplet Loss

$$\mathscr{L}(A,P,N) = D(A,P) - D(A,N) + m$$

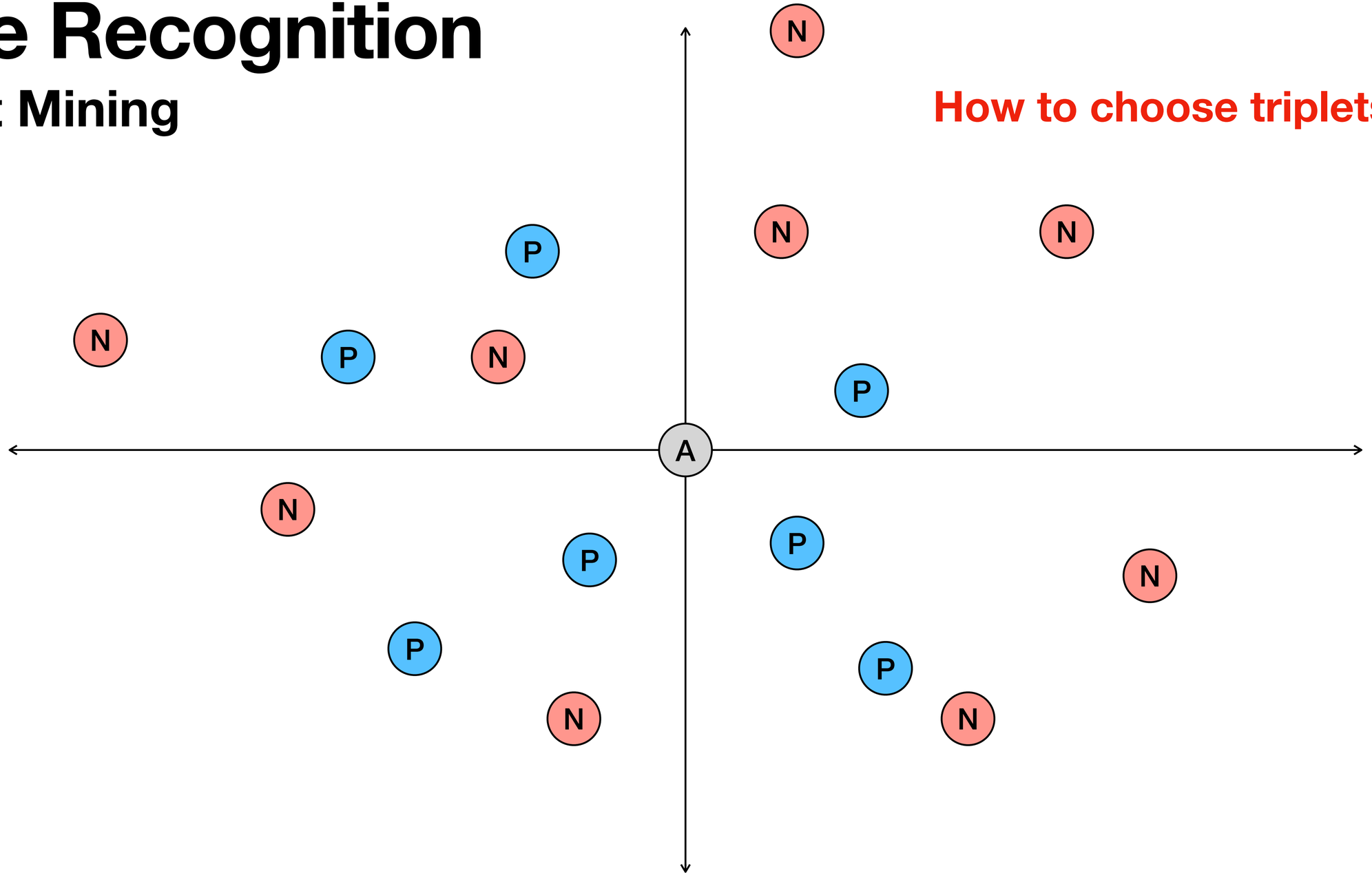
$D_p + m < D_n$



Face Recognition

Triplet Mining

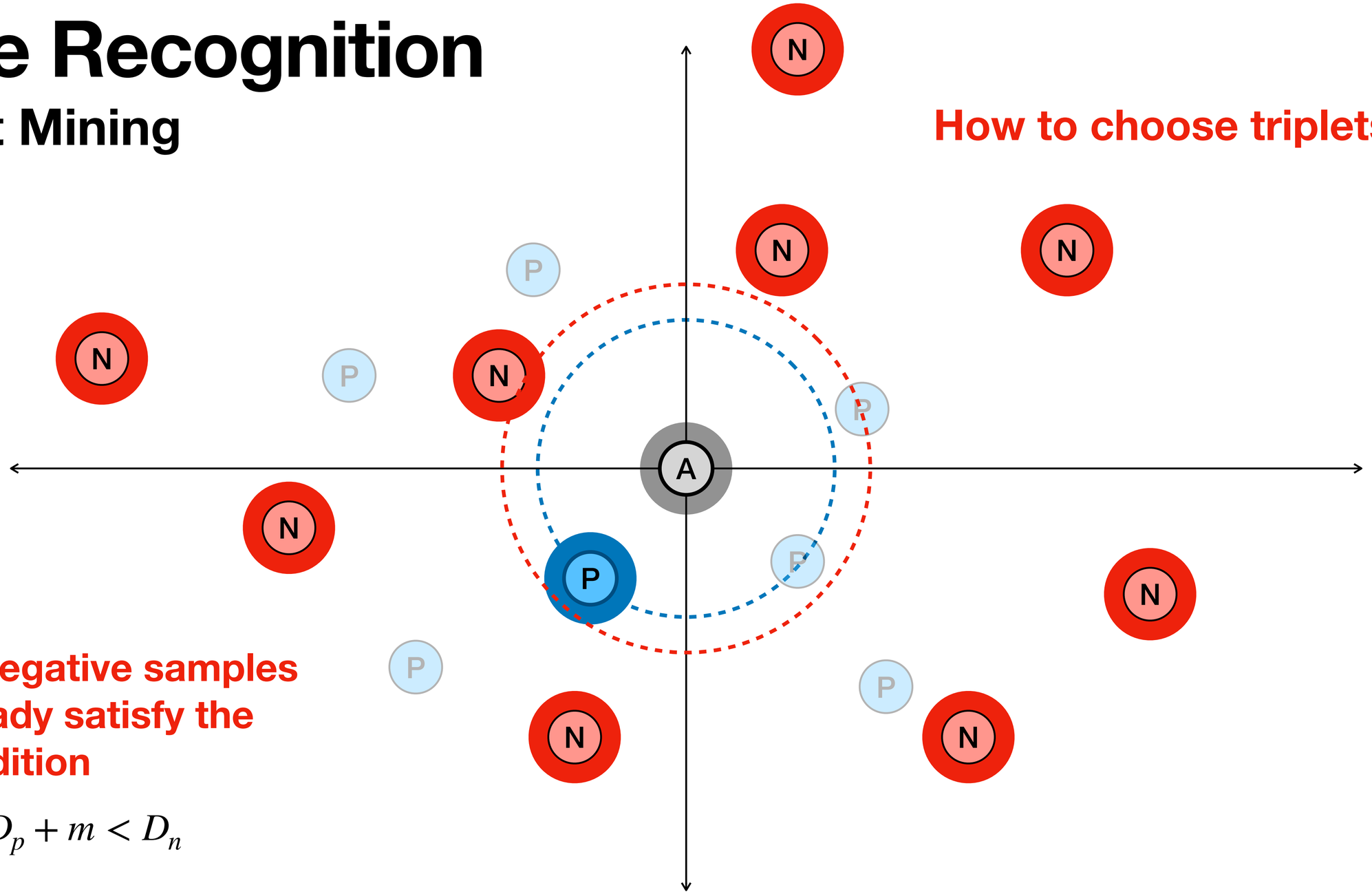
How to choose triplets ?



Face Recognition

Triplet Mining

How to choose triplets ?



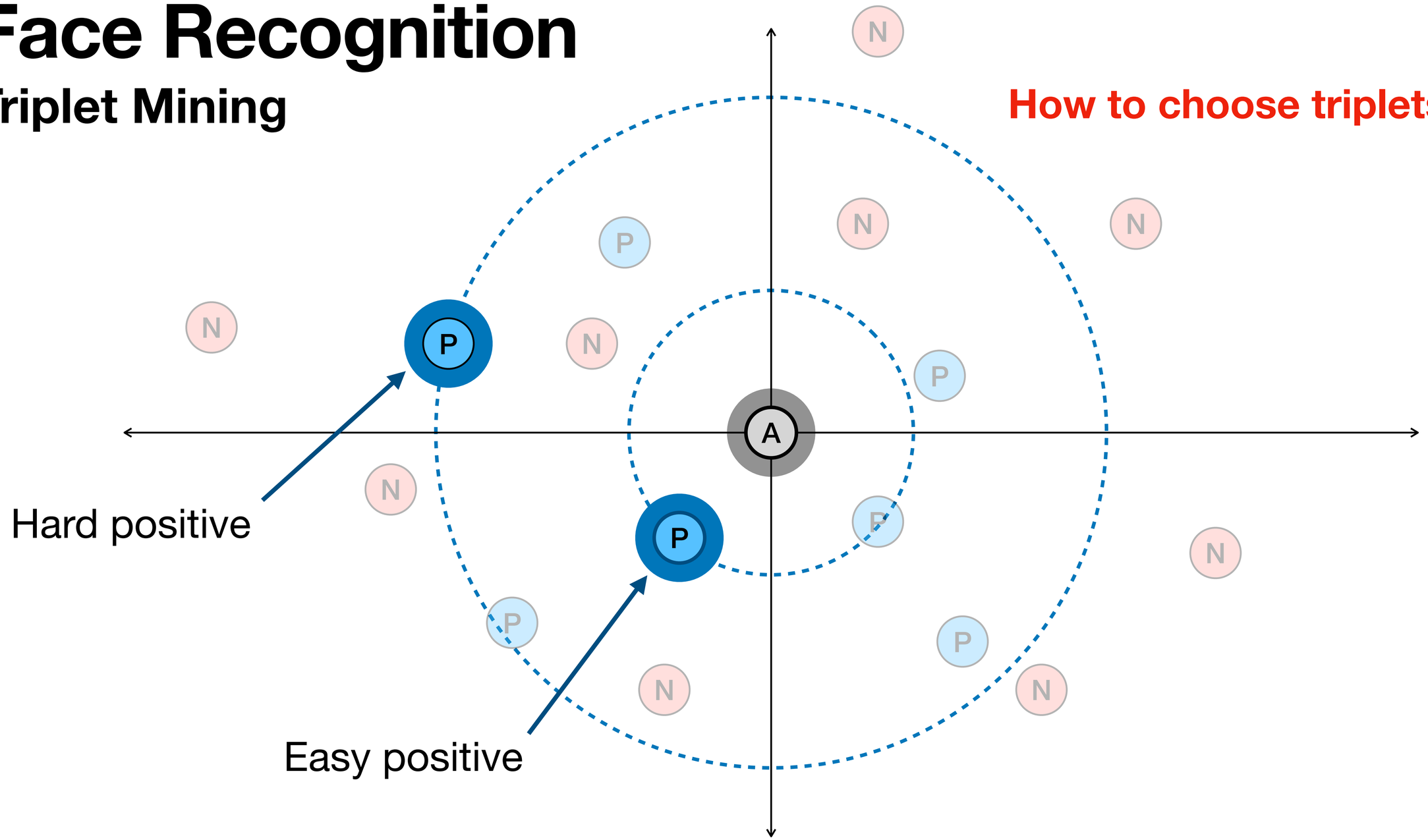
All negative samples already satisfy the condition

$$D_p + m < D_n$$

Face Recognition

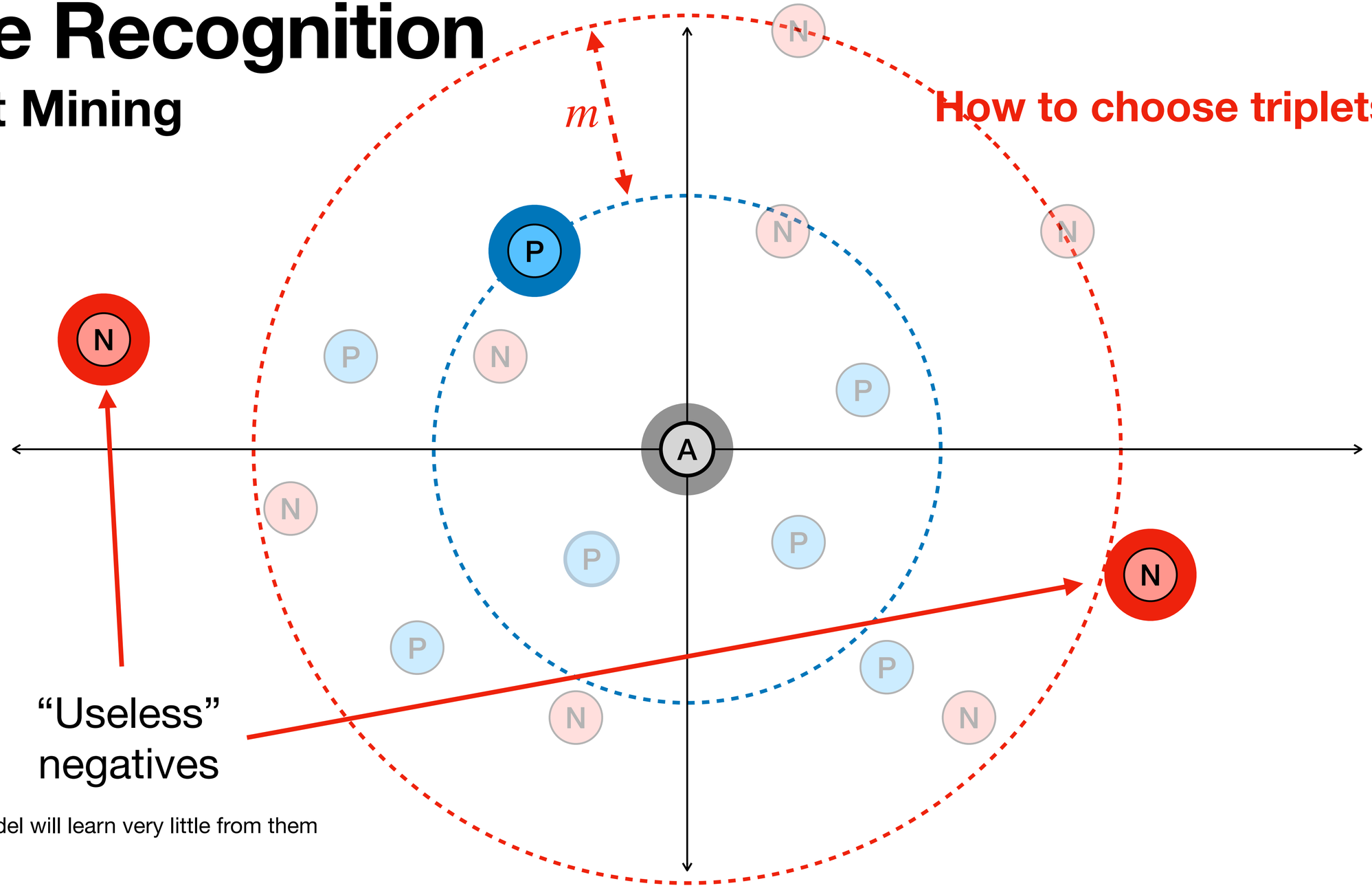
Triplet Mining

How to choose triplets ?



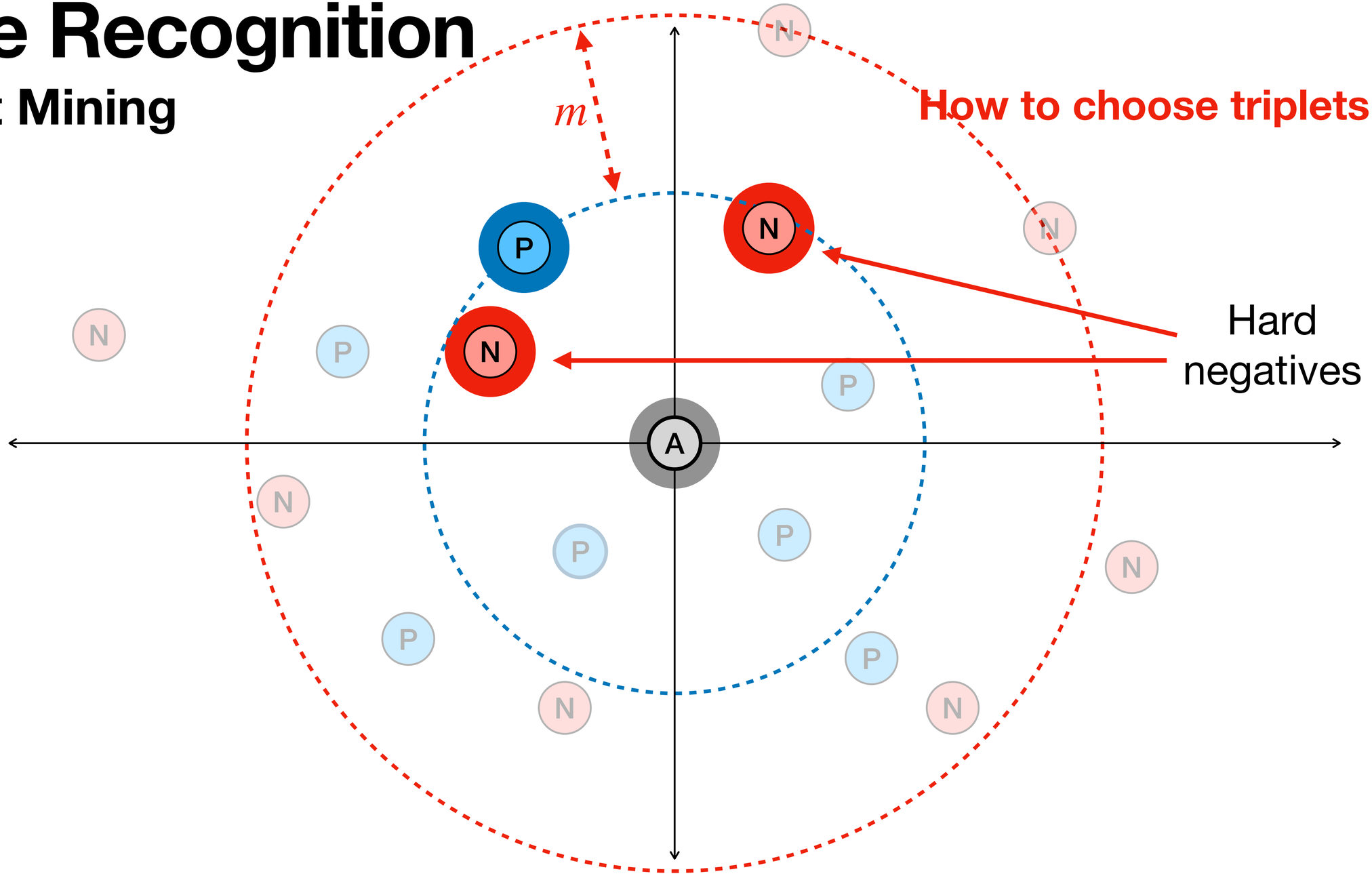
Face Recognition

Triplet Mining



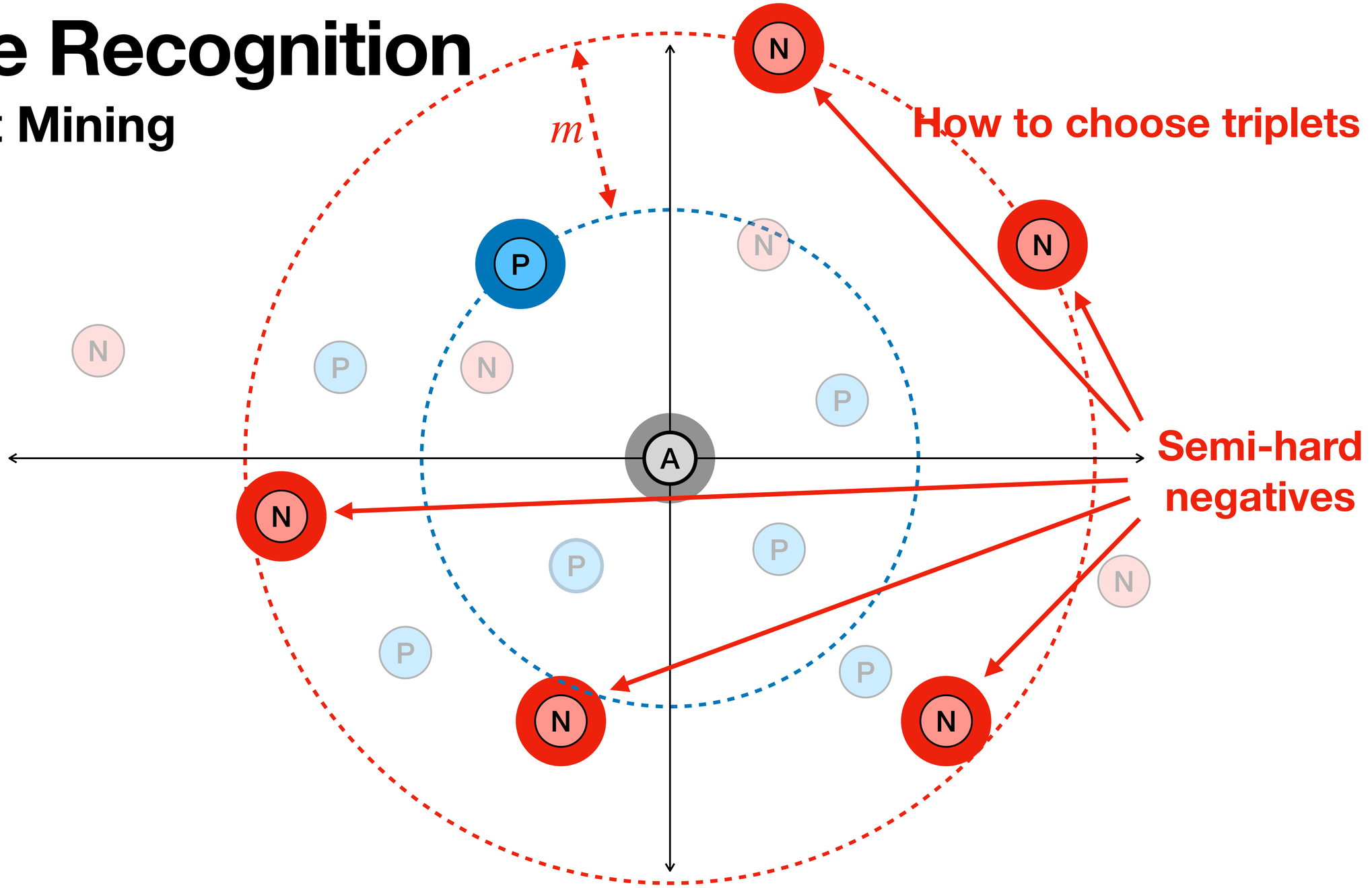
Face Recognition

Triplet Mining



Face Recognition

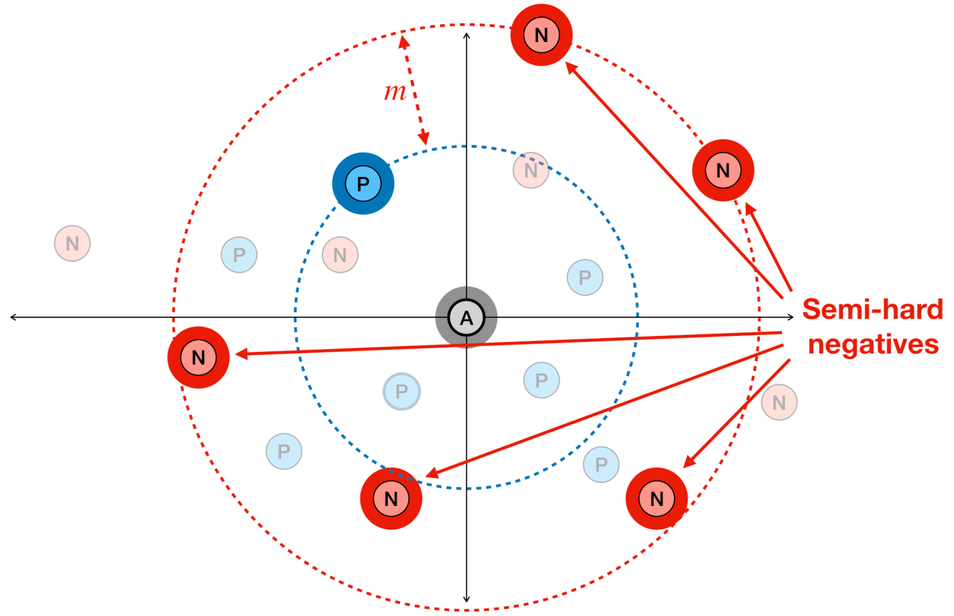
Triplet Mining



Face Recognition

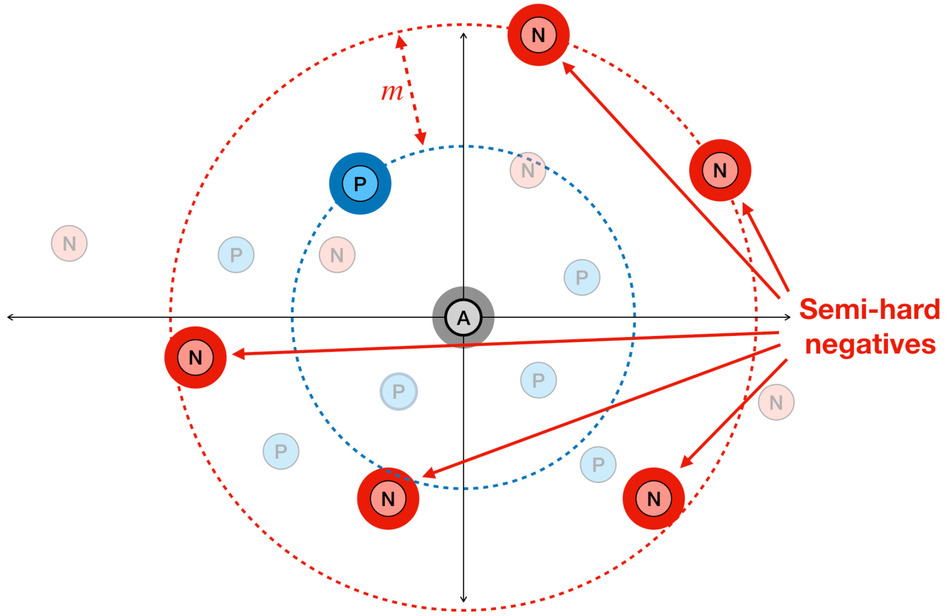
Triplet Mining

- **Offline**
  - Difficult with large datasets
  - Representations change as training progresses



Face Recognition

Triplet Mining



- Offline
  - Difficult with large datasets
  - Representations change as training progresses
- **Online**
  - Randomly sample a larger “mini-batch”
  - Compute distance matrix
  - Select useful triplets

Face Recognition

Image Batch

1. Randomly select identity

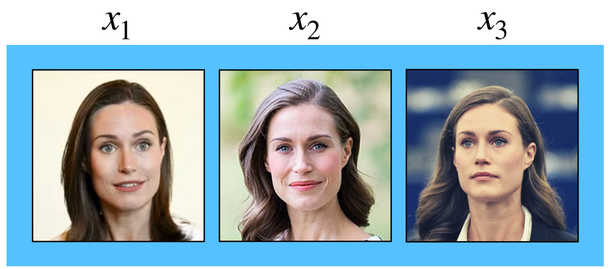
x_1

Face Recognition

Image Batch

1. Randomly select identity

2. Select more samples of the same identity (positives)

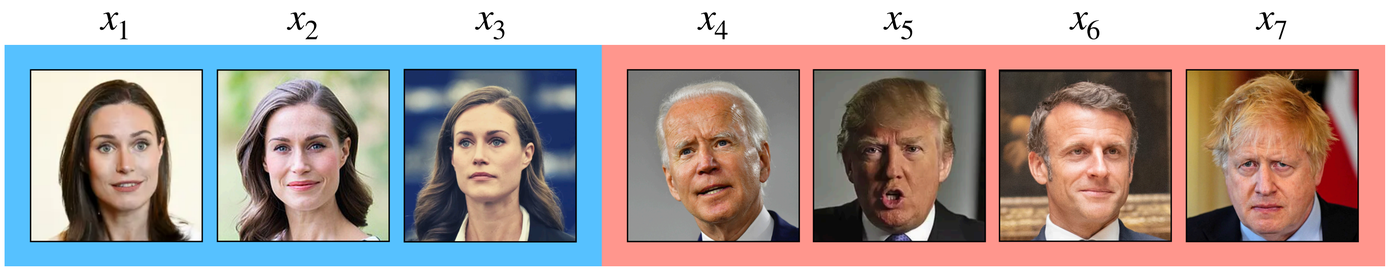


Face Recognition

Image Batch

1. Randomly select identity
2. Select more samples of the same identity (positives)
3. Fill the rest of the batch with random negative samples

$< 7;\ 3;\ 112;\ 112 >$



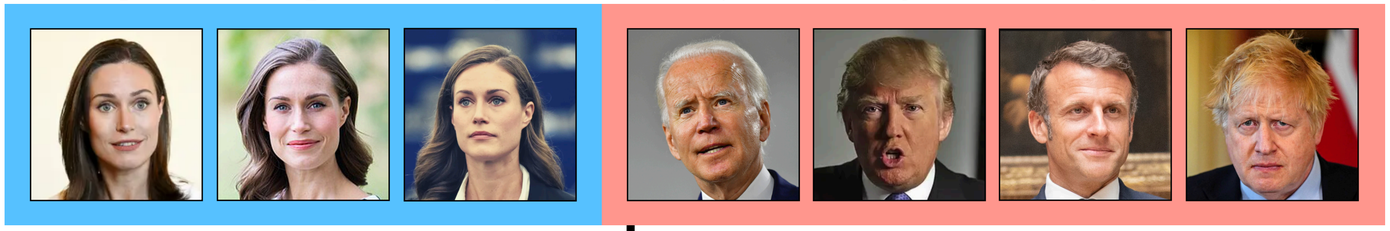
Face Recognition

Image Batch

1. Randomly select identity
2. Select more samples of the same identity (positives)
3. Fill the rest of the batch with random negative samples
4. Project into lower-dimension representation

$< 7;\ 3;\ 112;\ 112 >$

x_1 x_2 x_3 x_4 x_5 x_6 x_7



CNN Feature Extractor

$G(x)$

$$P^T = (p_1 \quad p_2 \quad p_3 \quad p_4 \quad p_5 \quad p_6 \quad p_7)$$

$< 7;\ 384 >$

Projected samples live on the surface of a hyper-sphere $\|p_i\| = 1$

Face Recognition

Distance Matrix

$$\|p_i\| = 1$$

$$< 7;\ 384 >$$

$$P^T = (p_1 \quad p_2 \quad p_3 \quad p_4 \quad p_5 \quad p_6 \quad p_7)$$

1. Compute 2D matrix with mutual distances of all elements

$$D = \begin{bmatrix} d(p_1,p_1) & d(p_1,p_2) & d(p_1,p_3) & \dots & d(p_1,p_7) \\ d(p_2,p_1) & d(p_2,p_2) & d(p_2,p_3) & \dots & d(p_2,p_7) \\ d(p_3,p_1) & d(p_3,p_2) & d(p_3,p_3) & \dots & d(p_3,p_7) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ d(p_7,p_1) & d(p_7,p_2) & d(p_7,p_3) & \dots & d(p_7,p_7) \end{bmatrix}$$

Distances between positive samples

Face Recognition

Distance Matrix

$$\|p_i\| = 1$$

$$< 7;\ 384 >$$

$$P^T = (p_1 \quad p_2 \quad p_3 \quad p_4 \quad p_5 \quad p_6 \quad p_7)$$

1. Compute 2D matrix with mutual distances of all elements

$$D = \begin{bmatrix} d(p_1,p_1) & d(p_1,p_2) & d(p_1,p_3) & \dots & d(p_1,p_7) \\ d(p_2,p_1) & d(p_2,p_2) & d(p_2,p_3) & \dots & d(p_2,p_7) \\ d(p_3,p_1) & d(p_3,p_2) & d(p_3,p_3) & \dots & d(p_3,p_7) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ d(p_7,p_1) & d(p_7,p_2) & d(p_7,p_3) & \dots & d(p_7,p_7) \end{bmatrix}$$

If we are using cosine distance

$$d(a,b) = 1 - a \,.\, b$$

Face Recognition

Distance Matrix

$$\|p_i\| = 1$$

$$< 7;\ 384 >$$

$$P^T = (p_1 \quad p_2 \quad p_3 \quad p_4 \quad p_5 \quad p_6 \quad p_7)$$

1. Compute 2D matrix with mutual distances of all elements

$$D = 1 - \begin{bmatrix} p_1p_1 & p_1p_2 & p_1p_3 & \cdots & p_1p_7 \\ p_2p_1 & p_2p_2 & p_2p_3 & \cdots & p_2p_7 \\ p_3p_1 & p_3p_2 & p_3p_3 & \cdots & p_3p_7 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ p_7p_1 & p_7p_2 & p_7p_3 & \cdots & p_7p_7 \end{bmatrix}$$

Face Recognition

Distance Matrix

$$\|p_i\| = 1$$

$$< 7;\ 384 >$$

$$P^T = (p_1 \quad p_2 \quad p_3 \quad p_4 \quad p_5 \quad p_6 \quad p_7)$$

1. Compute 2D matrix with mutual distances of all elements

Cosine similarity matrix

$$D = 1 - \begin{bmatrix} p_1p_1 & p_1p_2 & p_1p_3 & \cdots & p_1p_7 \\ p_2p_1 & p_2p_2 & p_2p_3 & \cdots & p_2p_7 \\ p_3p_1 & p_3p_2 & p_3p_3 & \cdots & p_3p_7 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ p_7p_1 & p_7p_2 & p_7p_3 & \cdots & p_7p_7 \end{bmatrix} = 1 - PP^T$$

$< 7;\ 7 >$

$< 7;\ 384 >$ $< 384;\ 7 >$

Outer product

Cosine distance matrix

Face Recognition

Distance Matrix

$$\|p_i\| = 1$$

$$< 7;\ 384 >$$

$$P^T = (p_1 \quad p_2 \quad p_3 \quad p_4 \quad p_5 \quad p_6 \quad p_7)$$

1. Compute 2D matrix with mutual distances of all elements

Expresses how good each of the p_i vectors is aligned with all of the others

Cosine similarity matrix

$$D = 1 - \begin{bmatrix} p_1p_1 & p_1p_2 & p_1p_3 & \cdots & p_1p_7 \\ p_2p_1 & p_2p_2 & p_2p_3 & \cdots & p_2p_7 \\ p_3p_1 & p_3p_2 & p_3p_3 & \cdots & p_3p_7 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ p_7p_1 & p_7p_2 & p_7p_3 & \cdots & p_7p_7 \end{bmatrix} = 1 - PP^T$$

$< 7;\ 7 >$

$< 7;\ 384 >$ $< 384;\ 7 >$

Outer product

Cosine distance matrix

Face Recognition

Triplet Selection

$$m = 0.3$$

1. Select Anchor

D	p_1	p_2	p_3	p_4	p_5	p_6	p_7
p_1		0.3	0.9	0.4	0.6	0.7	1.4
p_2	0.3		0.8	0.2	0.1	0.7	0.9
p_3	0.9	0.8		0.1	0.4	0.6	0.8
p_4	0.4	0.2	0.1		0.8	0.2	0.3
p_5	0.6	0.1	0.4	0.8		0.3	0.4
p_6	0.7	0.7	0.6	0.2	0.3		0.6
p_7	1.4	0.9	0.8	0.3	0.4	0.6	

Face Recognition

Triplet Selection

$$m = 0.3$$

1. Select Anchor

D	p_1	p_2	p_3	p_4	p_5	p_6	p_7
p_1		0.3	0.9	0.4	0.6	0.7	1.4
p_2	0.3		0.8	0.2	0.1	0.7	0.9
p_3	0.9	0.8		0.1	0.4	0.6	0.8
p_4	0.4	0.2	0.1		0.8	0.2	0.3
p_5	0.6	0.1	0.4	0.8		0.3	0.4
p_6	0.7	0.7	0.6	0.2	0.3		0.6
p_7	1.4	0.9	0.8	0.3	0.4	0.6	

Face Recognition

Triplet Selection

1. Select Anchor
2. Combinations of Positives

$$m = 0.3$$

D	p_1	p_2	p_3	p_4	p_5	p_6	p_7
p_1		0.3	0.9	0.4	0.6	0.7	1.4
p_2	0.3		0.8	0.2	0.1	0.7	0.9
p_3	0.9	0.8		0.1	0.4	0.6	0.8
p_4	0.4	0.2	0.1		0.8	0.2	0.3
p_5	0.6	0.1	0.4	0.8		0.3	0.4
p_6	0.7	0.7	0.6	0.2	0.3		0.6
p_7	1.4	0.9	0.8	0.3	0.4	0.6	

Face Recognition

Triplet Selection

1. Select Anchor
2. Combinations of Positives
3. Decide if proper Negatives are available

These are semi-hard

$m = 0.3$

D	p_1	p_2	p_3	p_4	p_5	p_6	p_7
p_1		0.3	0.9	0.4	0.6	0.7	1.4
p_2	0.3		0.8	0.2	0.1	0.7	0.9
p_3	0.9	0.8		0.1	0.4	0.6	0.8
p_4	0.4	0.2	0.1		0.8	0.2	0.3
p_5	0.6	0.1	0.4	0.8		0.3	0.4
p_6	0.7	0.7	0.6	0.2	0.3		0.6
p_7	1.4	0.9	0.8	0.3	0.4	0.6	

$(p_1; p_2; p_4)$

$(p_1; p_2; p_5)$

Face Recognition

Triplet Selection

1. Select Anchor
2. Combinations of Positives
3. Decide if proper Negatives are available

$$m = 0.3$$

D	p_1	p_2	p_3	p_4	p_5	p_6	p_7
p_1		0.3	0.9	0.4	0.6	0.7	1.4
p_2	0.3		0.8	0.2	0.1	0.7	0.9
p_3	0.9	0.8		0.1	0.4	0.6	0.8
p_4	0.4	0.2	0.1		0.8	0.2	0.3
p_5	0.6	0.1	0.4	0.8		0.3	0.4
p_6	0.7	0.7	0.6	0.2	0.3		0.6
p_7	1.4	0.9	0.8	0.3	0.4	0.6	

$(p_1; p_2; p_4)$
$(p_1; p_2; p_5)$
$(p_1; p_3; p_4)$
$(p_1; p_3; p_5)$
$(p_1; p_3; p_6)$

Face Recognition

Triplet Selection

1. Select Anchor
2. Combinations of Positives
3. Decide if proper Negatives are available

$$m = 0.3$$

D	p_1	p_2	p_3	p_4	p_5	p_6	p_7
p_1		0.3	0.9	0.4	0.6	0.7	1.4
p_2	0.3		0.8	0.2	0.1	0.7	0.9
p_3	0.9	0.8		0.1	0.4	0.6	0.8
p_4	0.4	0.2	0.1		0.8	0.2	0.3
p_5	0.6	0.1	0.4	0.8		0.3	0.4
p_6	0.7	0.7	0.6	0.2	0.3		0.6
p_7	1.4	0.9	0.8	0.3	0.4	0.6	

$(p_1; p_2; p_4)$
$(p_1; p_2; p_5)$
$(p_1; p_3; p_4)$
$(p_1; p_3; p_5)$
$(p_1; p_3; p_6)$
$(p_2; p_1; p_4)$
$(p_2; p_1; p_5)$

Face Recognition

Triplet Selection

1. Select Anchor
2. Combinations of Positives
3. Decide if proper Negatives are available

$$m = 0.3$$

D	p_1	p_2	p_3	p_4	p_5	p_6	p_7
p_1		0.3	0.9	0.4	0.6	0.7	1.4
p_2	0.3		0.8	0.2	0.1	0.7	0.9
p_3	0.9	0.8		0.1	0.4	0.6	0.8
p_4	0.4	0.2	0.1		0.8	0.2	0.3
p_5	0.6	0.1	0.4	0.8		0.3	0.4
p_6	0.7	0.7	0.6	0.2	0.3		0.6
p_7	1.4	0.9	0.8	0.3	0.4	0.6	

$(p_1; p_2; p_4)$
$(p_1; p_2; p_5)$
$(p_1; p_3; p_4)$
$(p_1; p_3; p_5)$
$(p_1; p_3; p_6)$
$(p_2; p_1; p_4)$
$(p_2; p_1; p_5)$
$(p_2; p_3; p_4)$
$(p_2; p_3; p_5)$
$(p_2; p_3; p_6)$
$(p_2; p_3; p_7)$

Face Recognition

Triplet Selection

1. Select Anchor
2. Combinations of Positives
3. Decide if proper Negatives are available

$$m = 0.3$$

D	p_1	p_2	p_3	p_4	p_5	p_6	p_7
p_1		0.3	0.9	0.4	0.6	0.7	1.4
p_2	0.3		0.8	0.2	0.1	0.7	0.9
p_3	0.9	0.8		0.1	0.4	0.6	0.8
p_4	0.4	0.2	0.1		0.8	0.2	0.3
p_5	0.6	0.1	0.4	0.8		0.3	0.4
p_6	0.7	0.7	0.6	0.2	0.3		0.6
p_7	1.4	0.9	0.8	0.3	0.4	0.6	

$(p_1; p_2; p_4)$
$(p_1; p_2; p_5)$
$(p_1; p_3; p_4)$
$(p_1; p_3; p_5)$
$(p_1; p_3; p_6)$
$(p_2; p_1; p_4)$
$(p_2; p_1; p_5)$
$(p_2; p_3; p_4)$
$(p_2; p_3; p_5)$
$(p_2; p_3; p_6)$
$(p_2; p_3; p_7)$

...

Face Recognition

Triplet Selection

1. Select Anchor
2. Combinations of Positives
3. Decide if proper Negatives are available

$$m = 0.3$$

D	p_1	p_2	p_3	p_4	p_5	p_6	p_7
p_1		0.3	0.9	0.4	0.6	0.7	1.4
p_2	0.3		0.8	0.2	0.1	0.7	0.9
p_3	0.9	0.8		0.1	0.4	0.6	0.8
p_4	0.4	0.2	0.1		0.8	0.2	0.3
p_5	0.6	0.1	0.4	0.8		0.3	0.4
p_6	0.7	0.7	0.6	0.2	0.3		0.6
p_7	1.4	0.9	0.8	0.3	0.4	0.6	

$(p_1; p_2; p_4)$
$(p_1; p_2; p_5)$
$(p_1; p_3; p_4)$
$(p_1; p_3; p_5)$
$(p_1; p_3; p_6)$
$(p_2; p_1; p_4)$
$(p_2; p_1; p_5)$
$(p_2; p_3; p_4)$
$(p_2; p_3; p_5)$
$(p_2; p_3; p_6)$
$(p_2; p_3; p_7)$

...

Are there any questions ?