PART III:CONVOLUTIONAL NEURAL NETWORKS

THE BASIC BUILDING BLOCKS OF CNNs

Discrete Convolution

1D: [Spectra]

$$f(x) * g(x) = \sum_{k=-\infty}^{k=+\infty} f(k).g(k-x)$$

2D: [Images]

$$f(x,y) * g(x,y) = \sum_{k=-\infty}^{k=+\infty} \sum_{l=-\infty}^{l=+\infty} f(k,l).g(x-k,y-l)$$

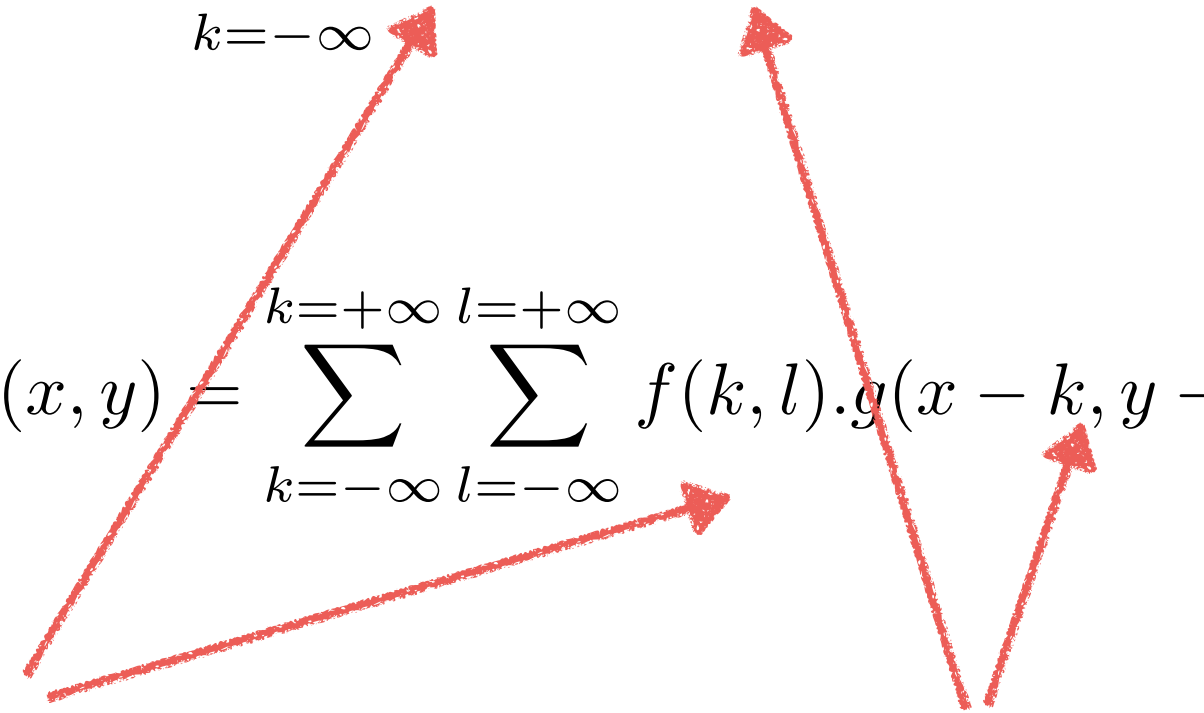
DISCRETE CONVOLUTION

1D: [Spectra]

$$f(x) * g(x) = \sum_{k=-\infty}^{k=+\infty} f(k).g(k-x)$$

2D: [Images]

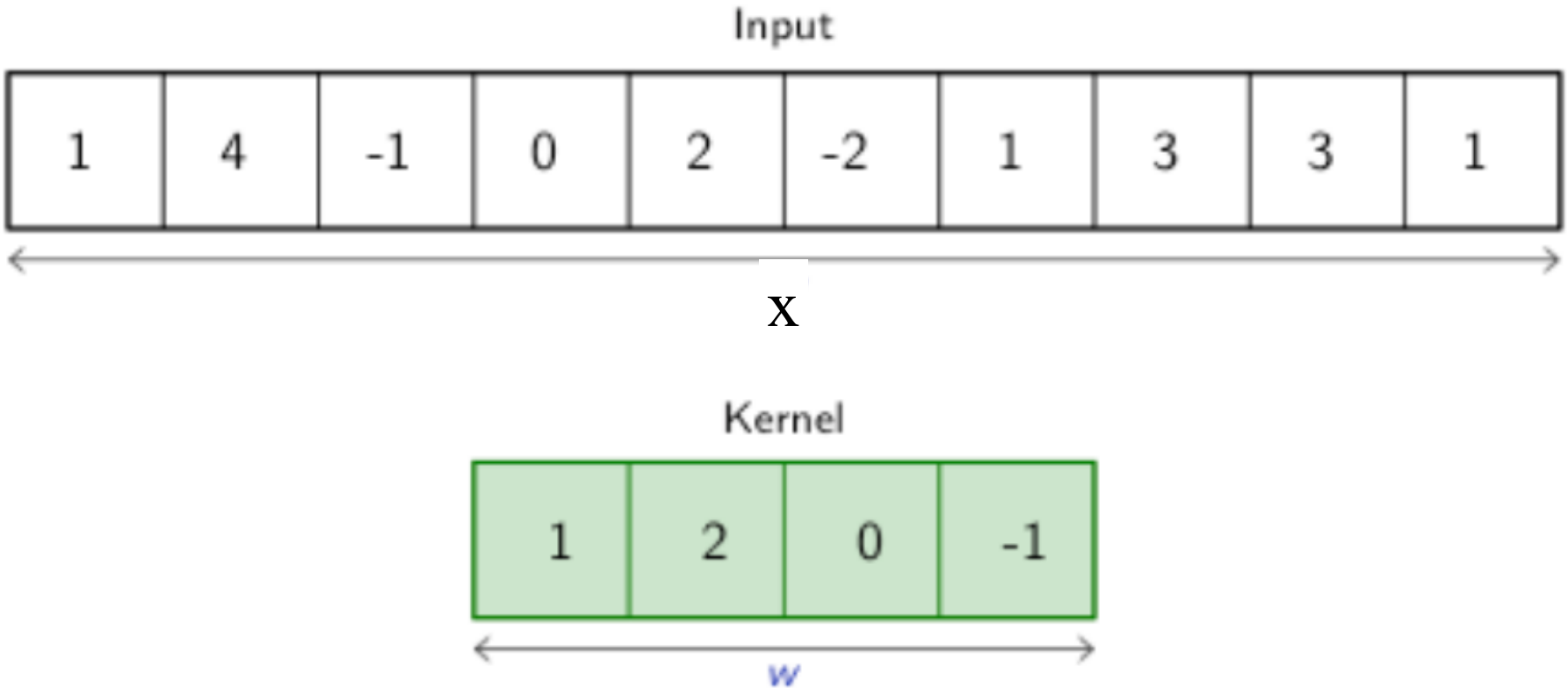
$$f(x,y) * g(x,y) = \sum_{k=-\infty}^{k=+\infty} \sum_{l=-\infty}^{l=+\infty} f(k,l).g(x-k,y-l)$$



CONVOLUTION KERNEL

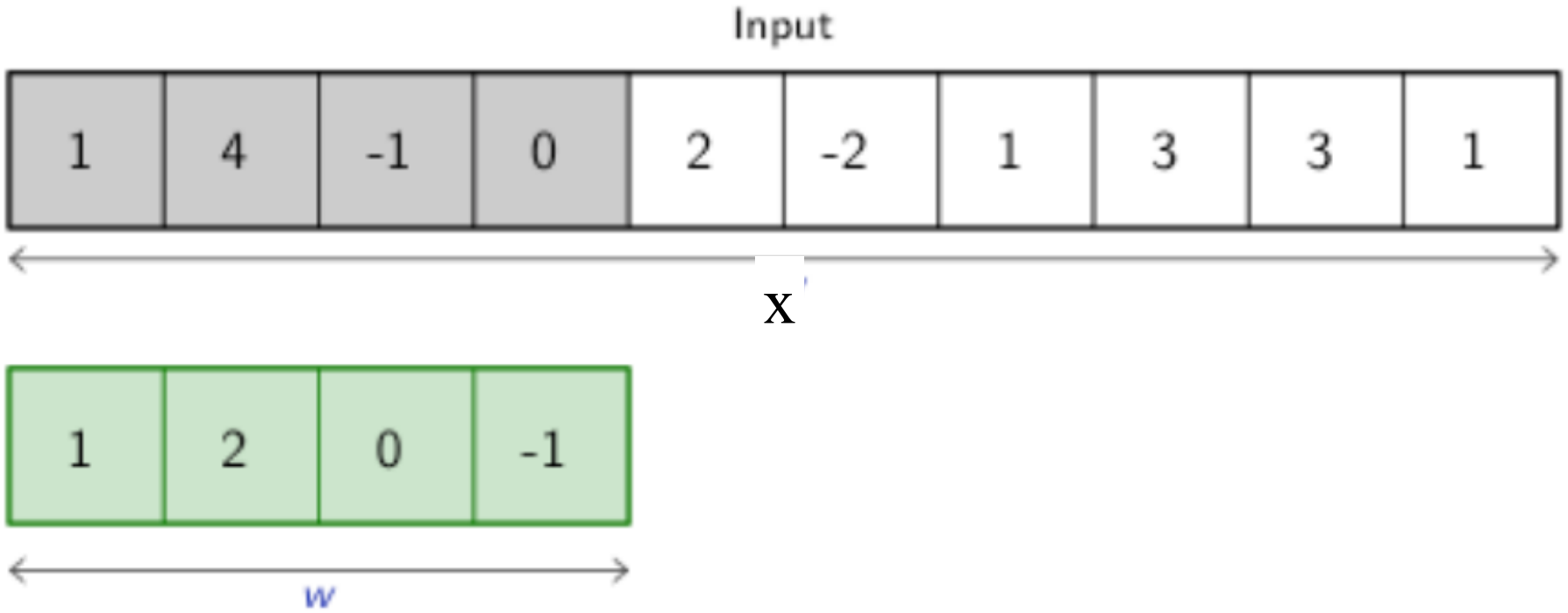
INPUT DATA

1-D CONVOLUTION



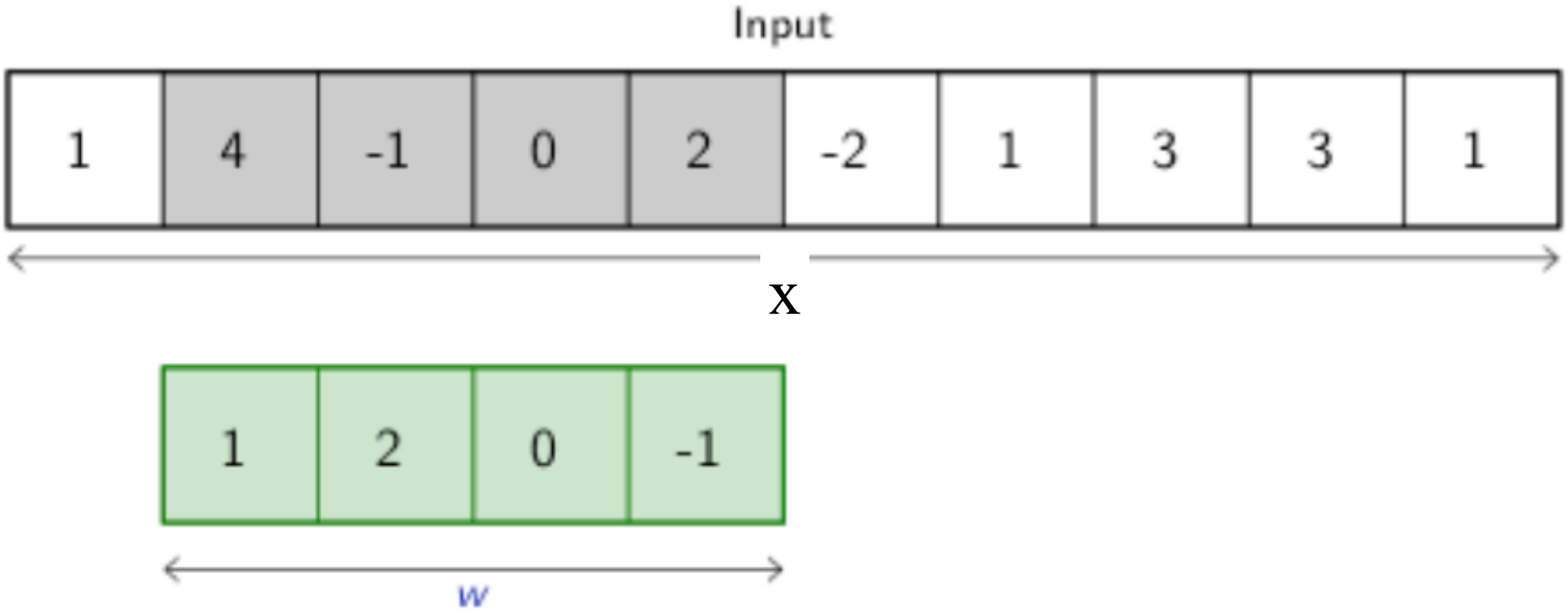
credit

1-D CONVOLUTION



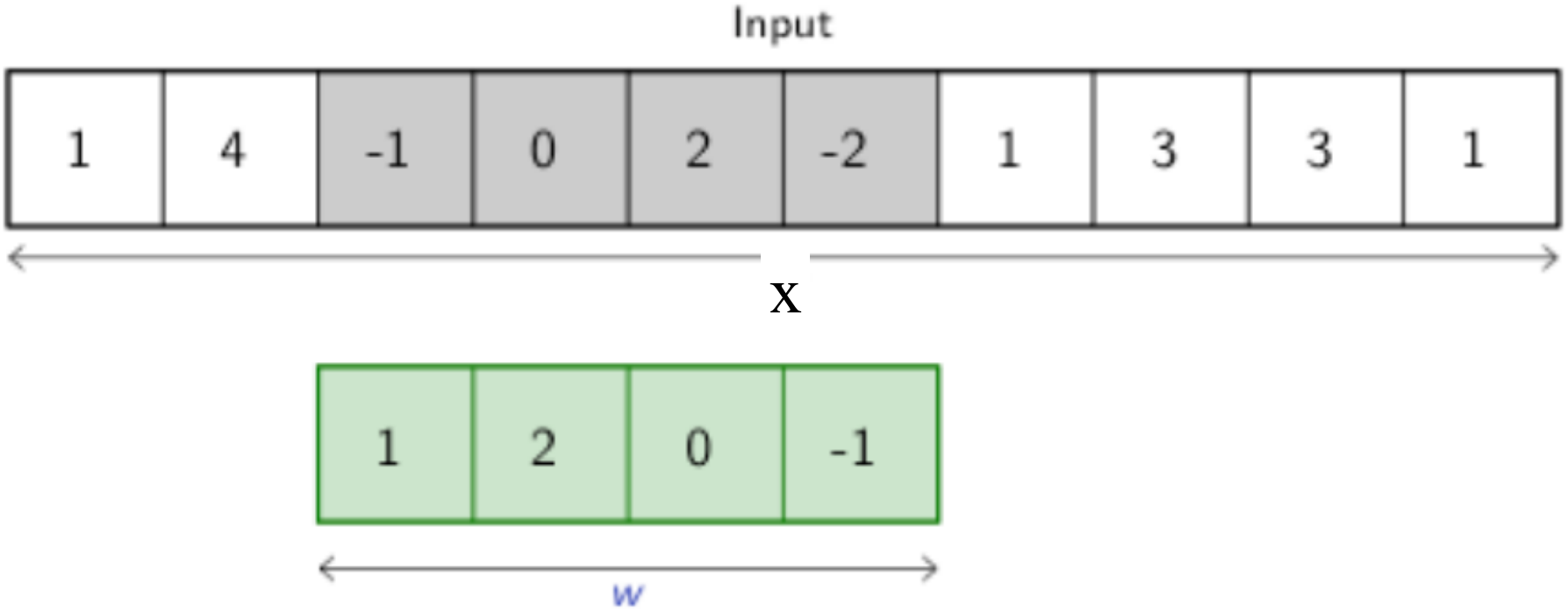
credit

1-D CONVOLUTION



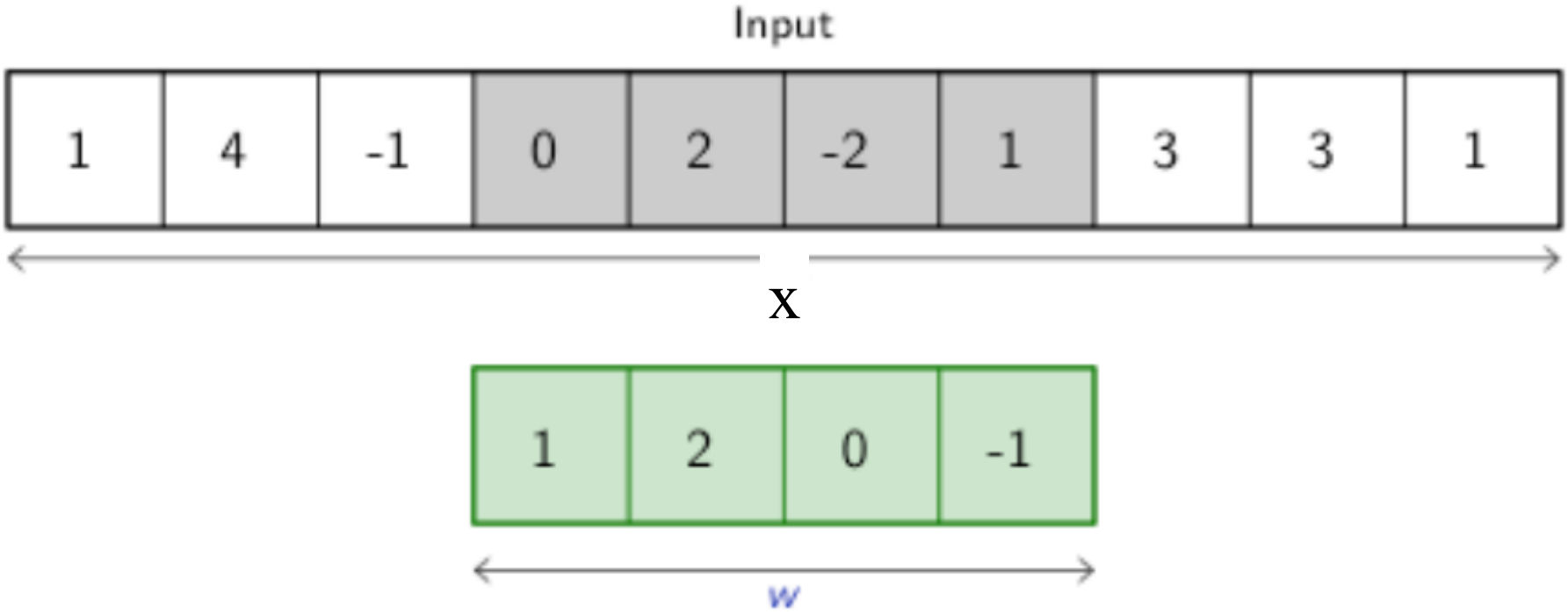
credit

1-D CONVOLUTION



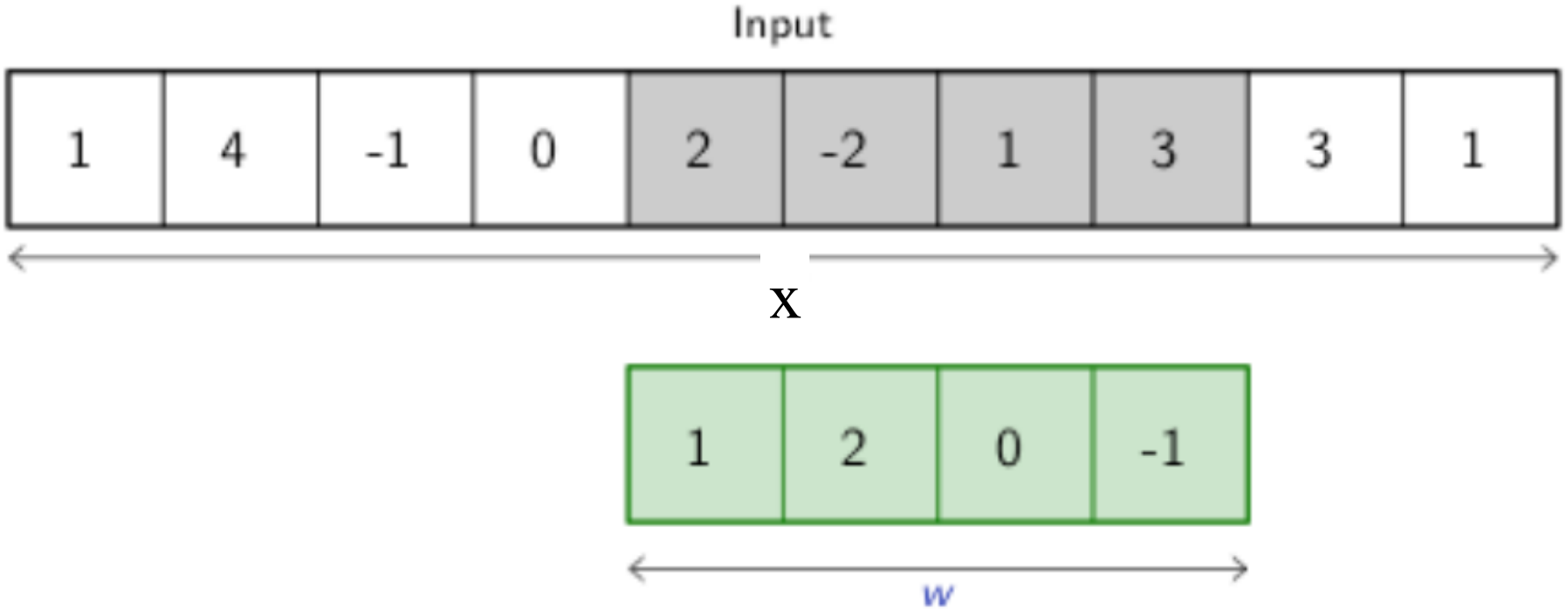
credit

1-D CONVOLUTION



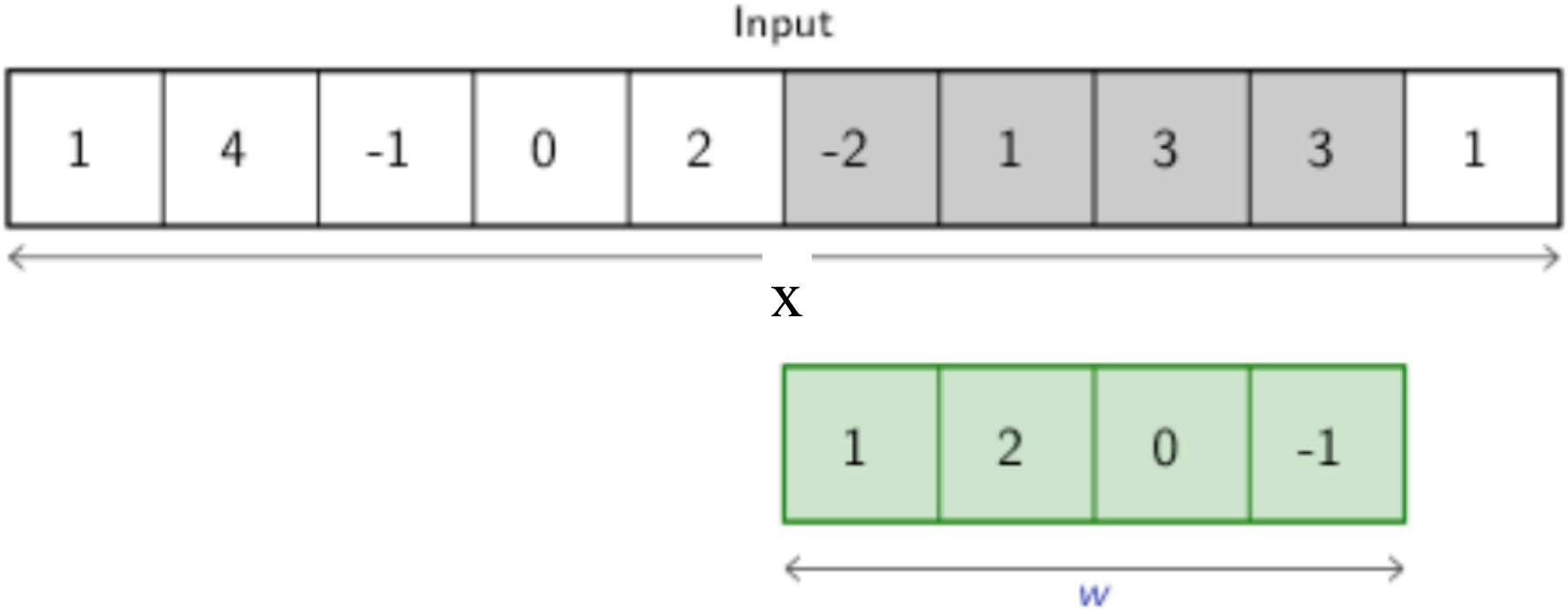
credit

1-D CONVOLUTION



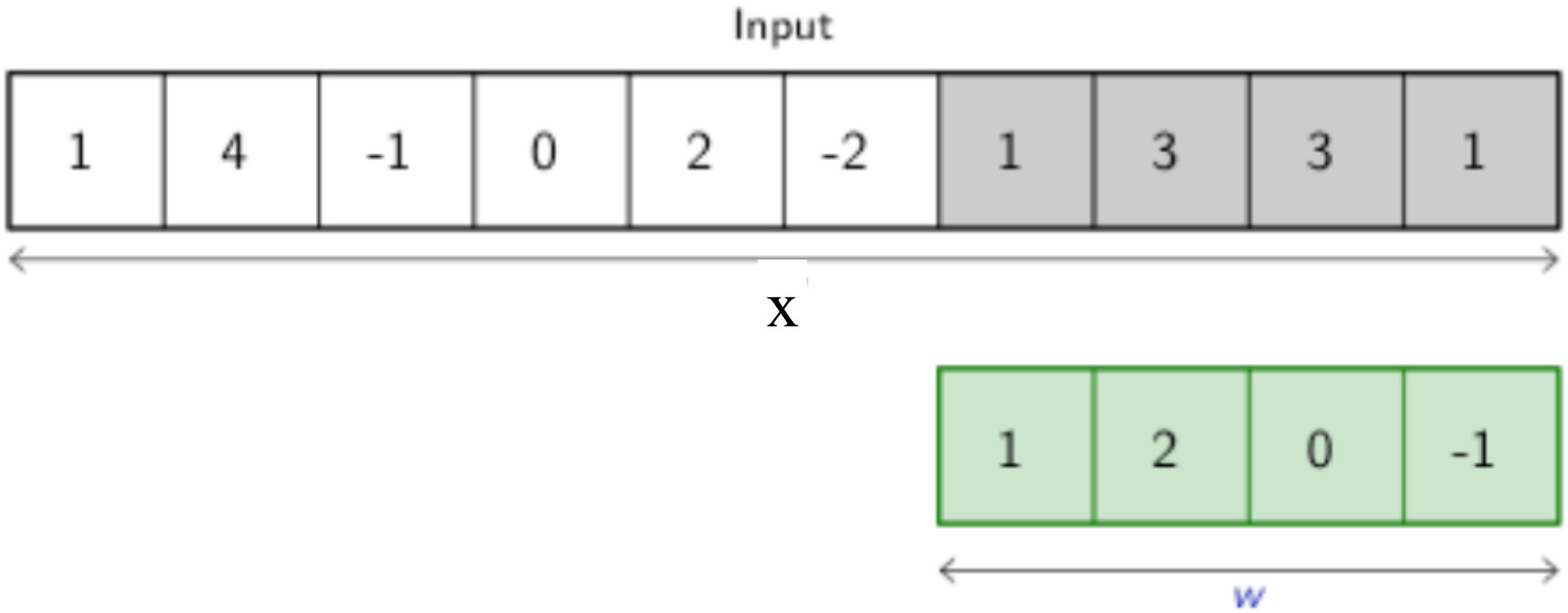
credit

1-D CONVOLUTION



credit

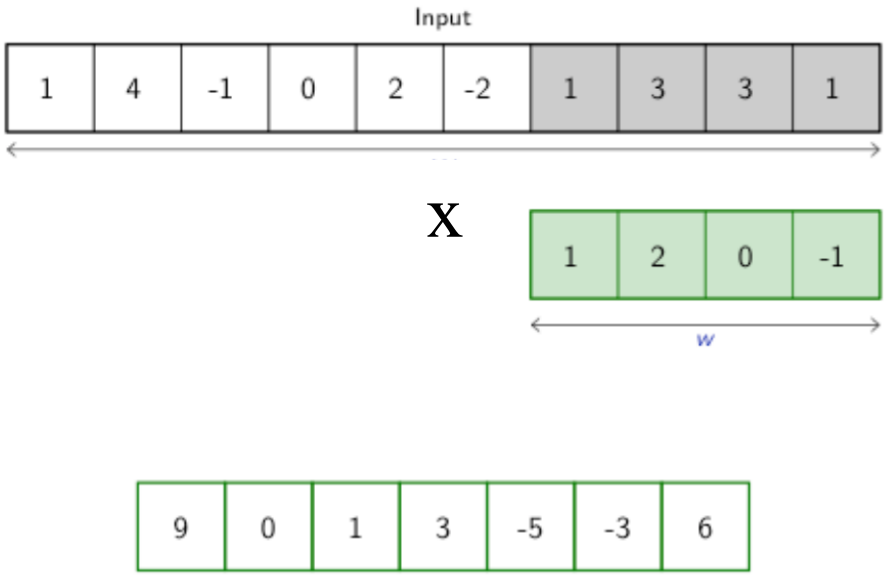
1-D CONVOLUTION

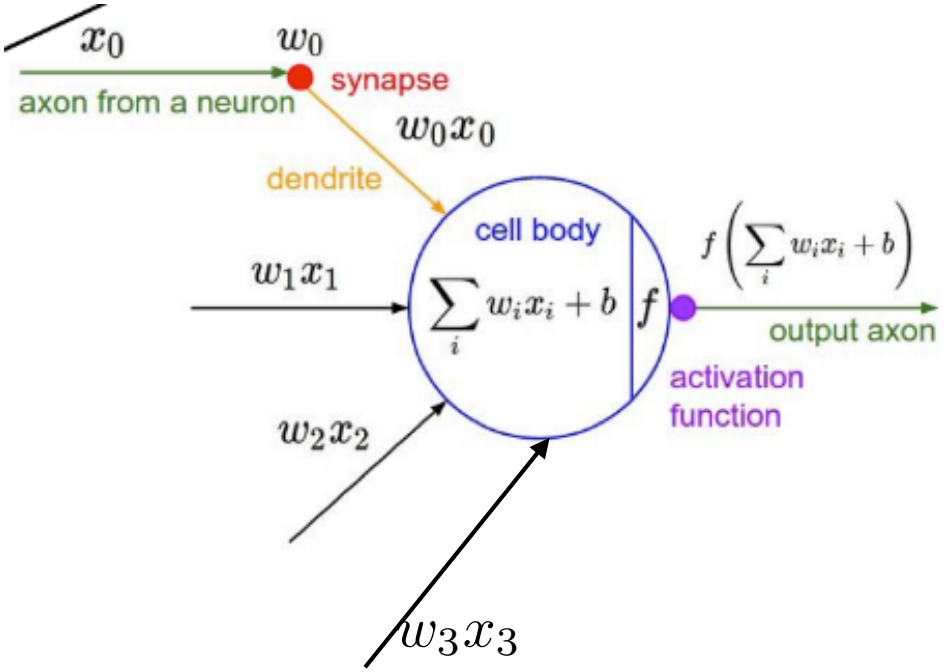


9	0	1	3	-5	-3	6

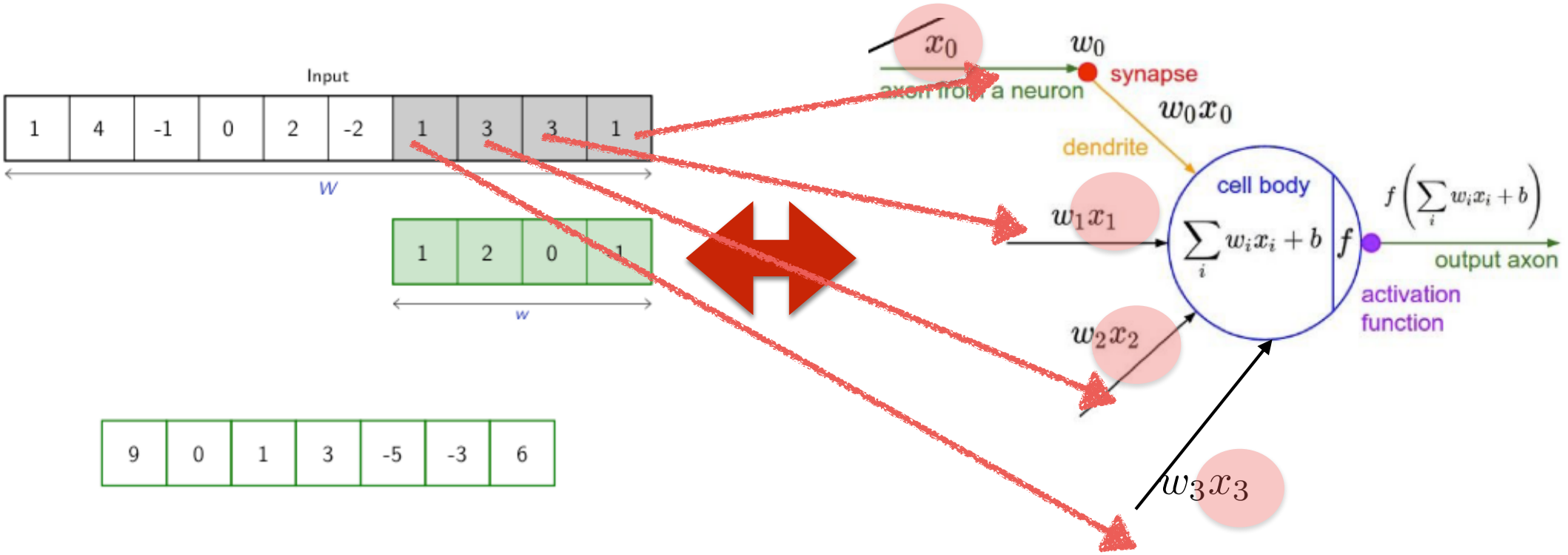
credit

THE CONVOLUTION BUILDING BLOCK OPERATION IS EQUIVALENT TO A NEURON WITH AS MANY INPUTS AS KERNEL ELEMENTS AND WEIGHTS EQUAL TO THE KERNEL

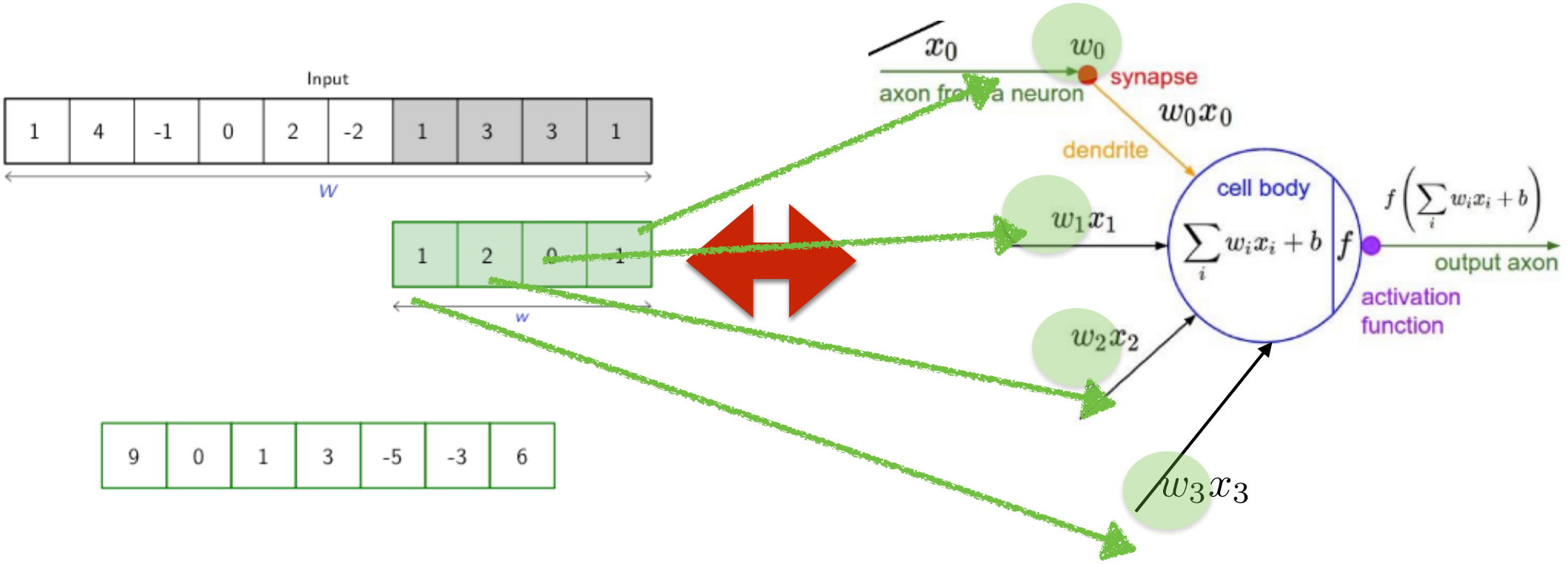




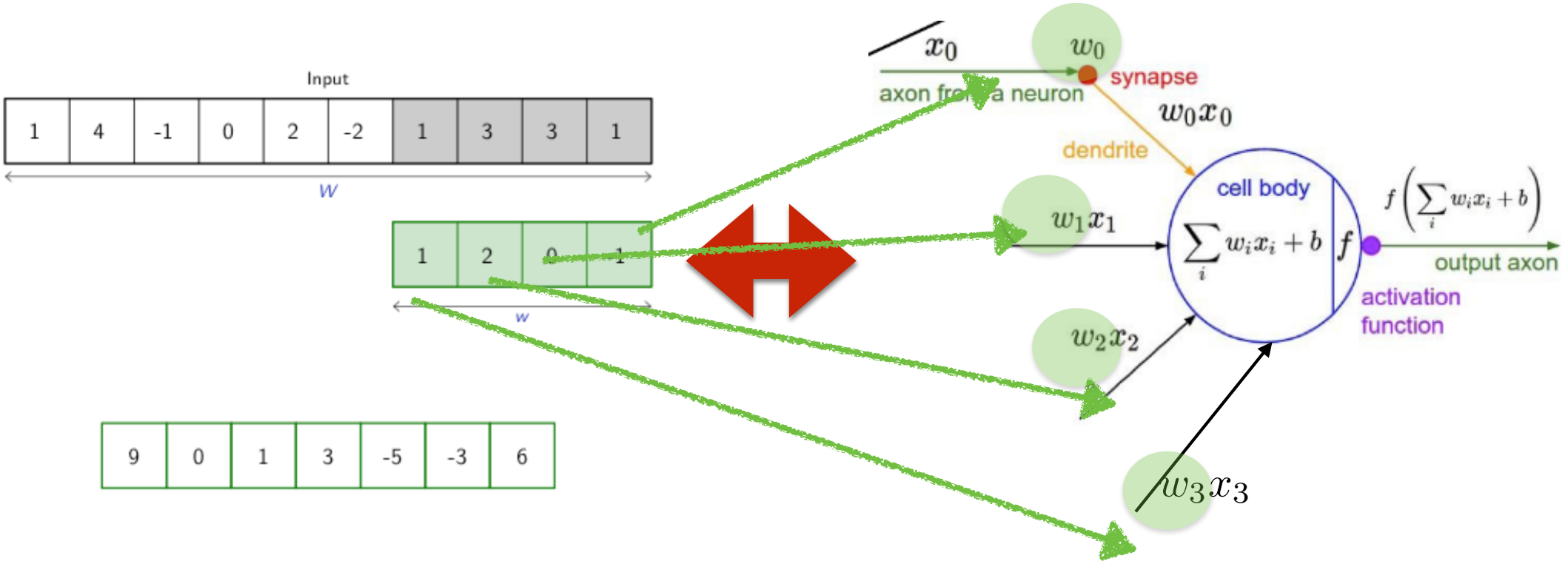
THE CONVOLUTION BUILDING BLOCK OPERATION IS EQUIVALENT TO A NEURON WITH AS MANY INPUTS AS KERNEL ELEMENTS AND WEIGHTS EQUAL TO THE KERNEL



THE CONVOLUTION BUILDING BLOCK OPERATION IS EQUIVALENT TO A NEURON WITH AS MANY INPUTS AS KERNEL ELEMENTS AND WEIGHTS EQUAL TO THE KERNEL



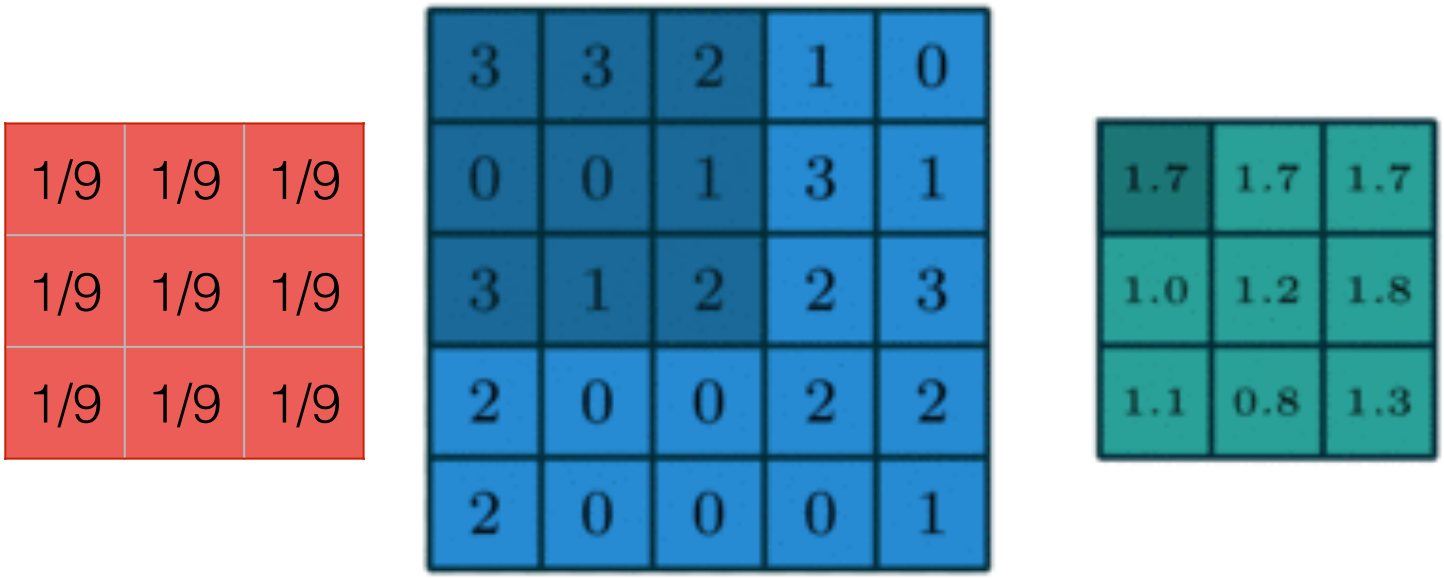
THE CONVOLUTION BUILDING BLOCK OPERATION IS EQUIVALENT TO A NEURON WITH AS MANY INPUTS AS KERNEL ELEMENTS AND WEIGHTS EQUAL TO THE KERNEL



WITH THE ADVANTAGE THAT THE SAME WEIGHTS ARE APPLIED TO ALL THE SIGNAL: TRANSLATION INVARIANCE

2-D CONVOLUTION

SAME IDEA, BUT THE KERNEL IS NOW 2D

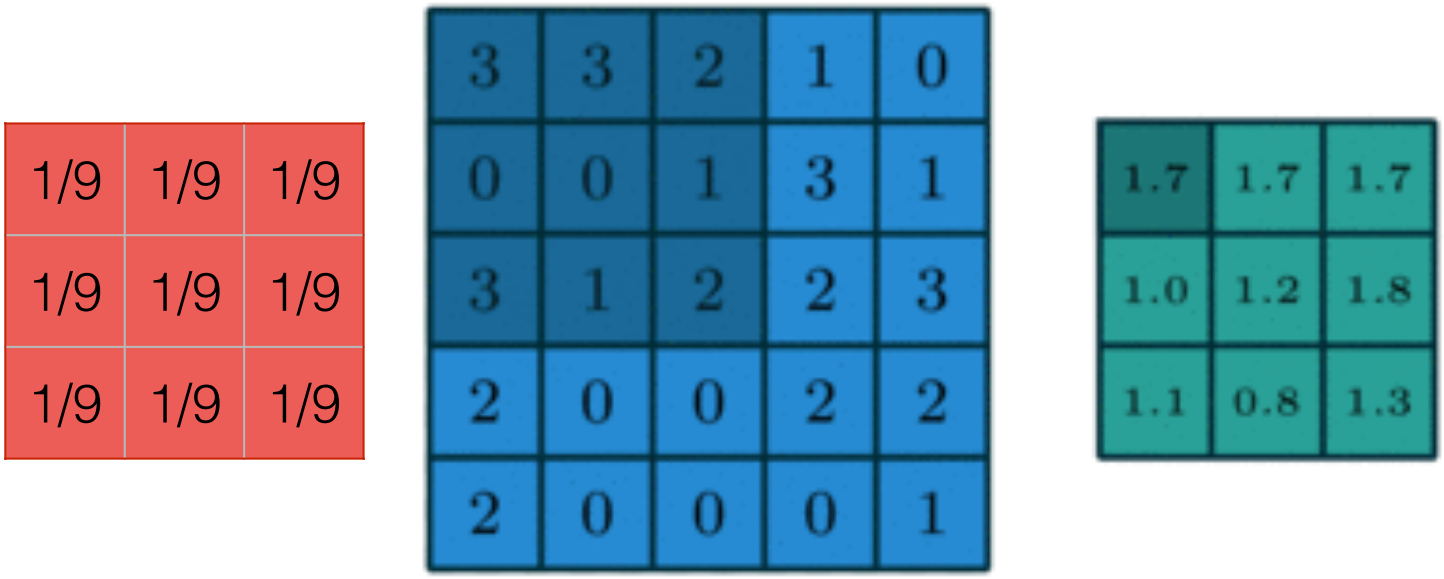


KERNEL **INPUT (IMAGE)** **OUTPUT**

Credit: animations from https://github.com/vdumoulin/conv_arithmetic

2-D CONVOLUTION

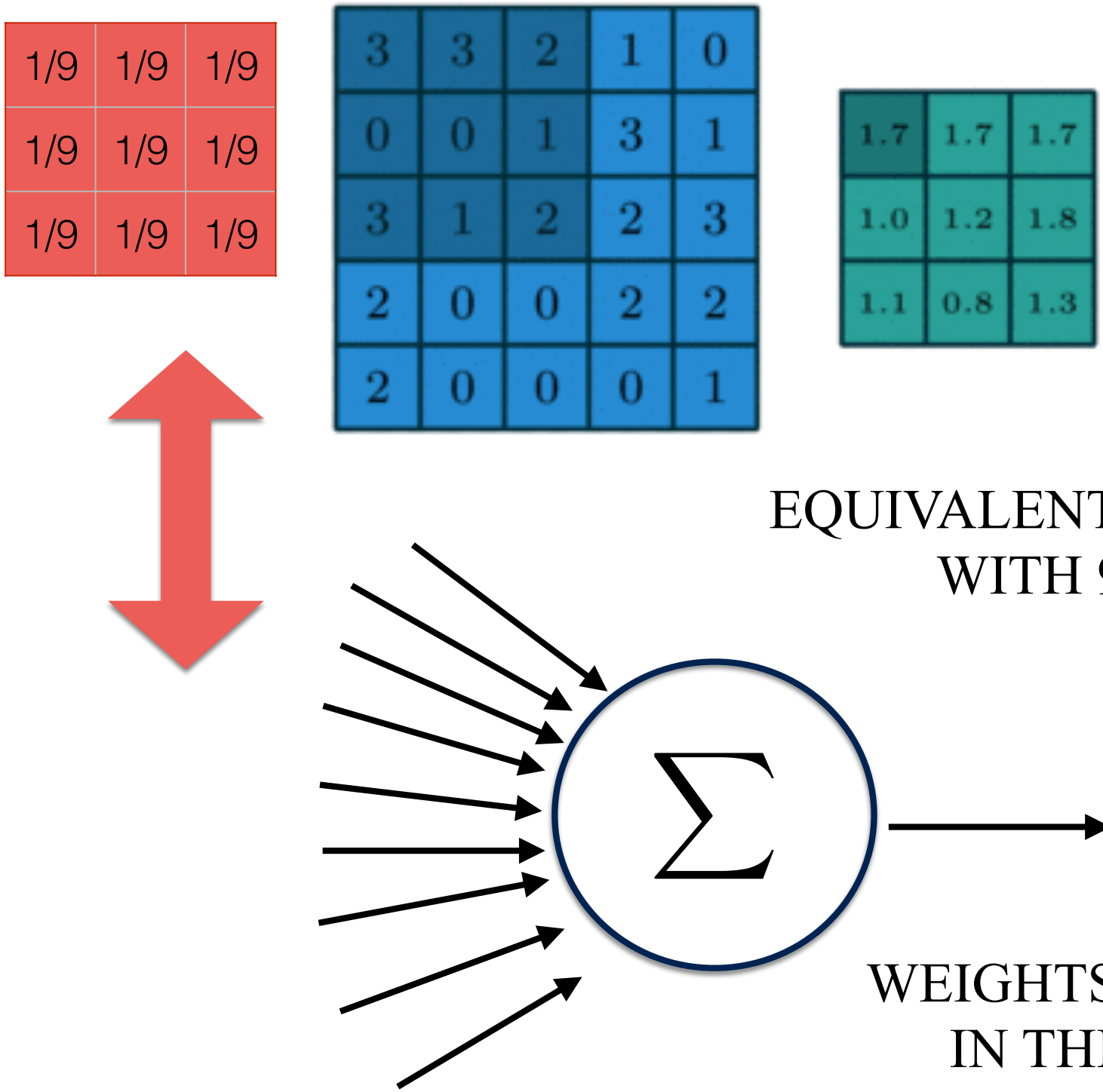
SAME IDEA, BUT THE KERNEL IS NOW 2D

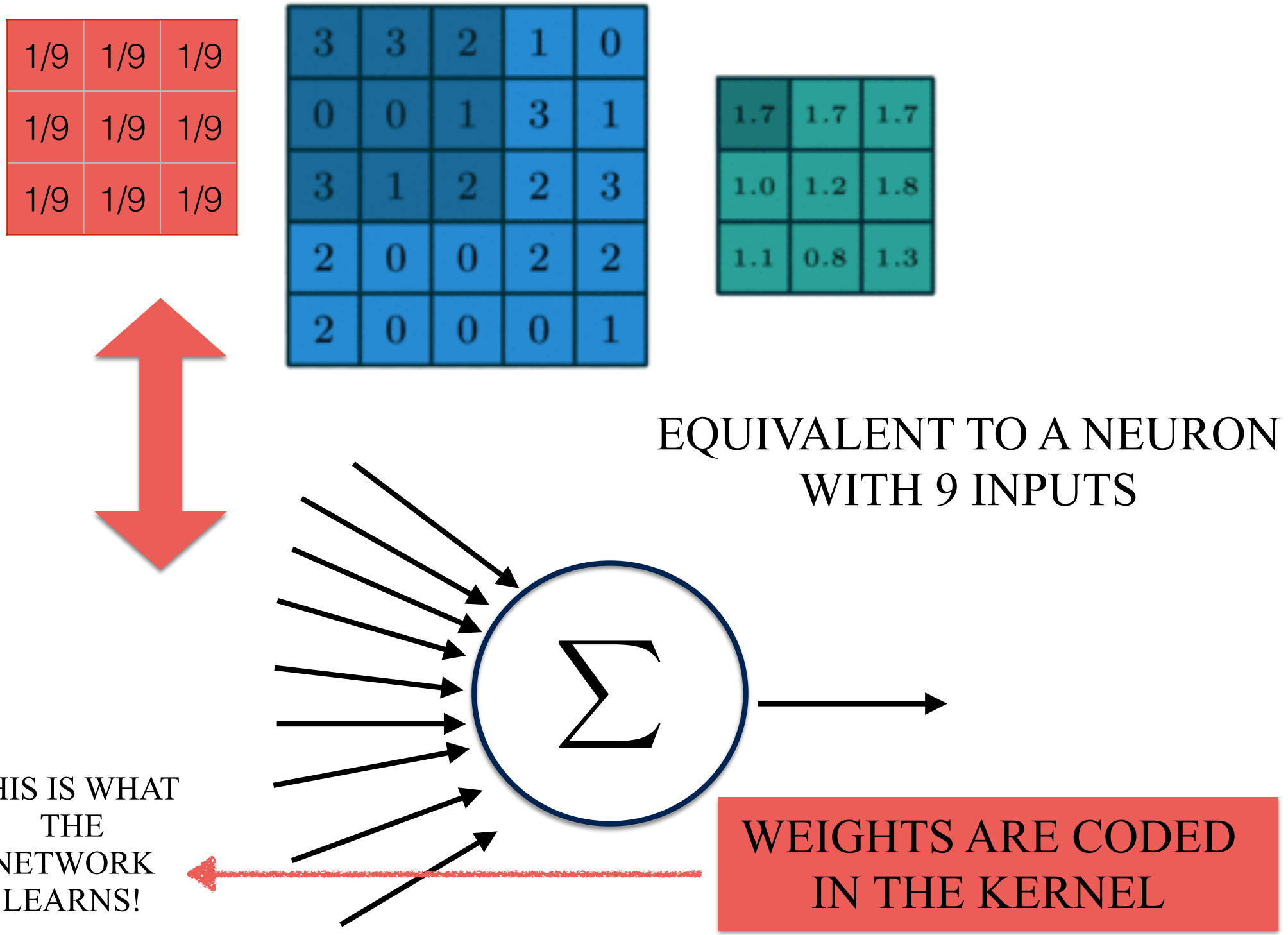


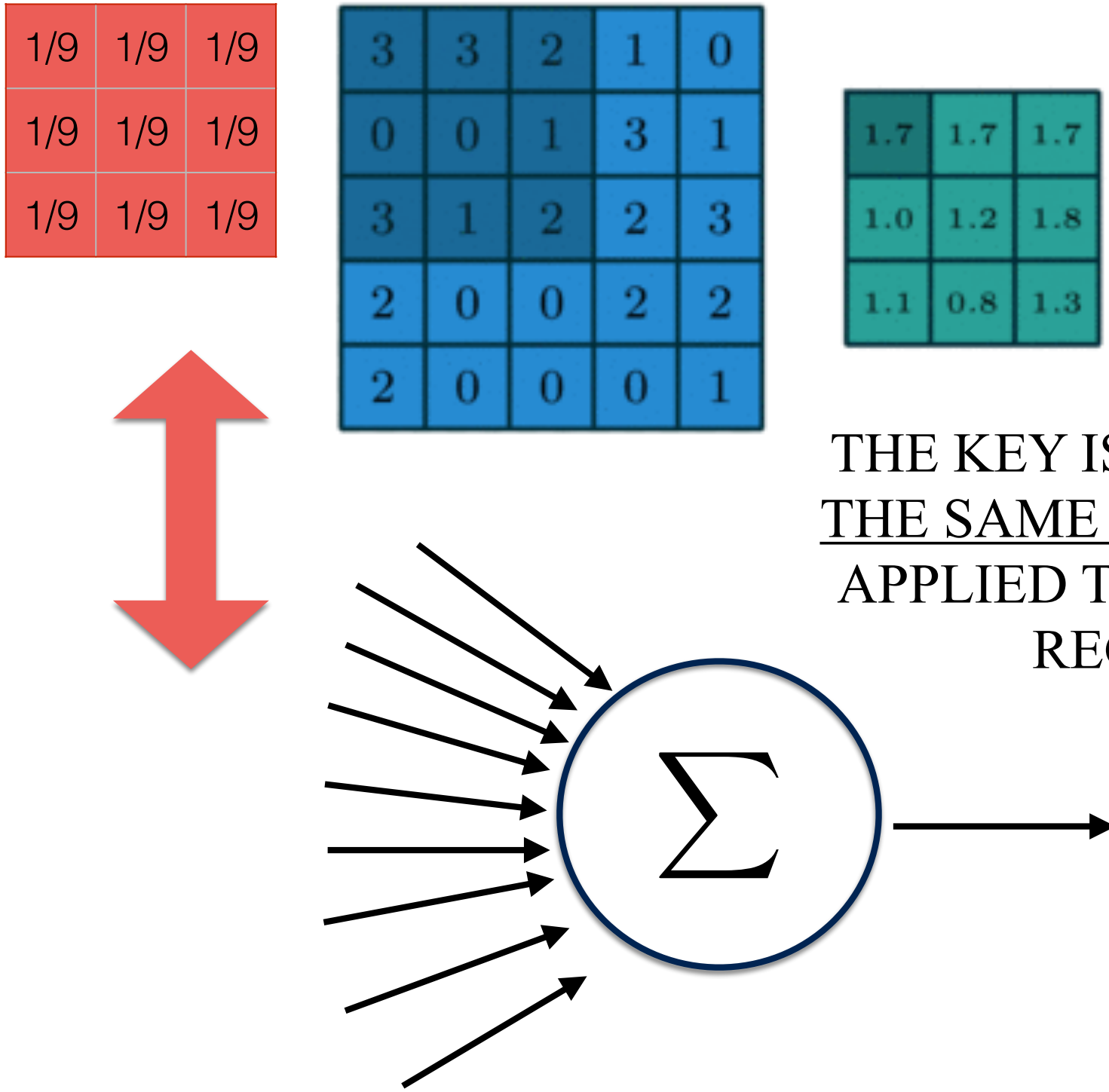
IN THE EXAMPLE: EACH 3x3 REGION GENERATES AN OUTPUT

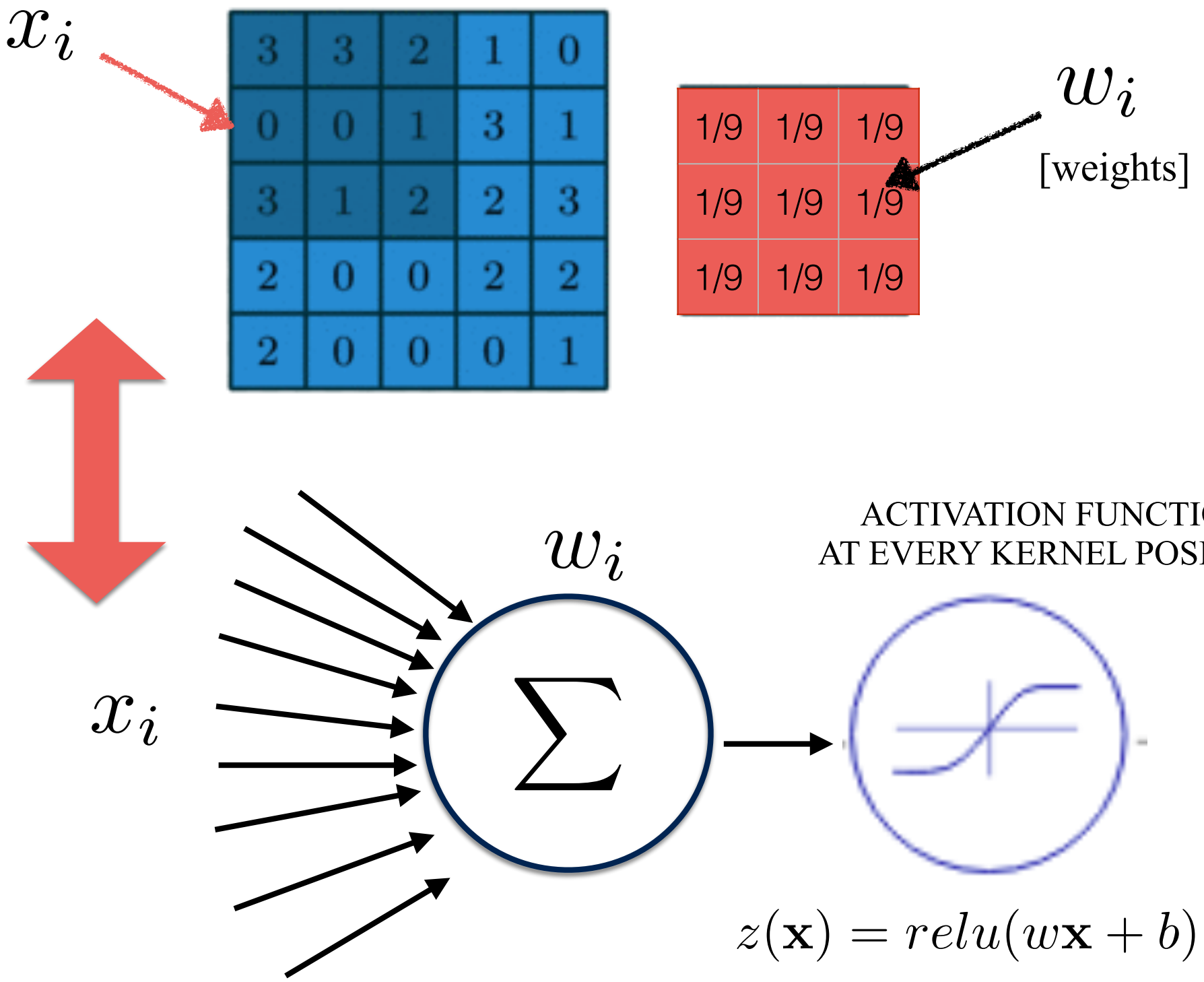
$$Size_{output} = Size_{input} - Size_{kernel} + 1$$

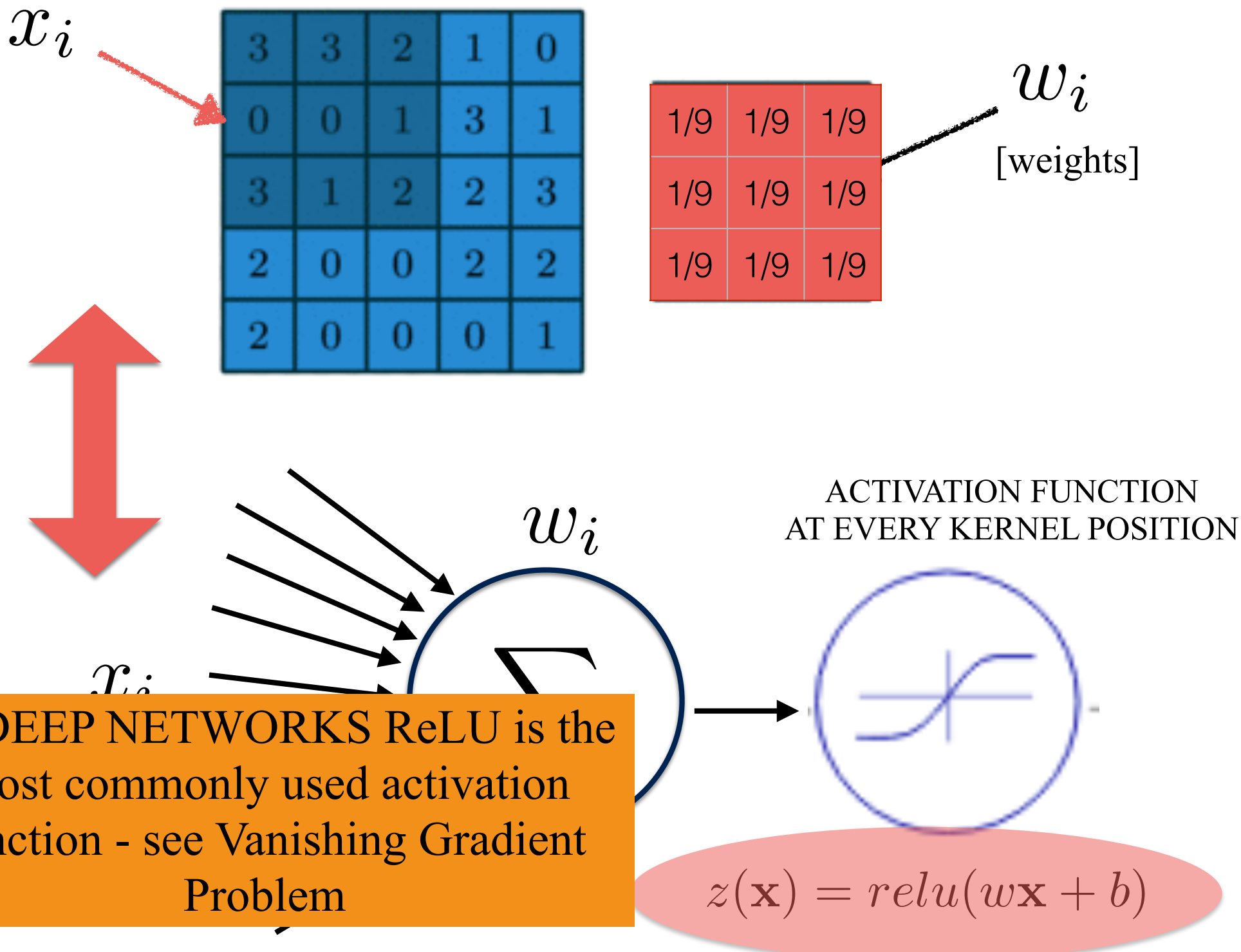
Credit: animations from https://github.com/vdumoulin/conv_arithmetic


1/9 1/9 1/9
1/9 1/9 1/9
1/9 1/9 1/9
3 3 2 1 0
0 0 1 3 1
3 1 2 2 3
2 0 0 2 2
2 0 0 0 1
1.7 1.7 1.7
1.0 1.2 1.8
1.1 0.8 1.3
EQUIVALENT TO A NEURON
WITH 9 INPUTS
Σ
WEIGHTS ARE CODED
IN THE KERNEL


1/9 1/9 1/9
1/9 1/9 1/9
1/9 1/9 1/9
3 3 2 1 0
0 0 1 3 1
3 1 2 2 3
2 0 0 2 2
2 0 0 0 1
1.7 1.7 1.7
1.0 1.2 1.8
1.1 0.8 1.3
EQUIVALENT TO A NEURON
WITH 9 INPUTS
Σ
THIS IS WHAT
THE
NETWORK
LEARNS!
WEIGHTS ARE CODED
IN THE KERNEL

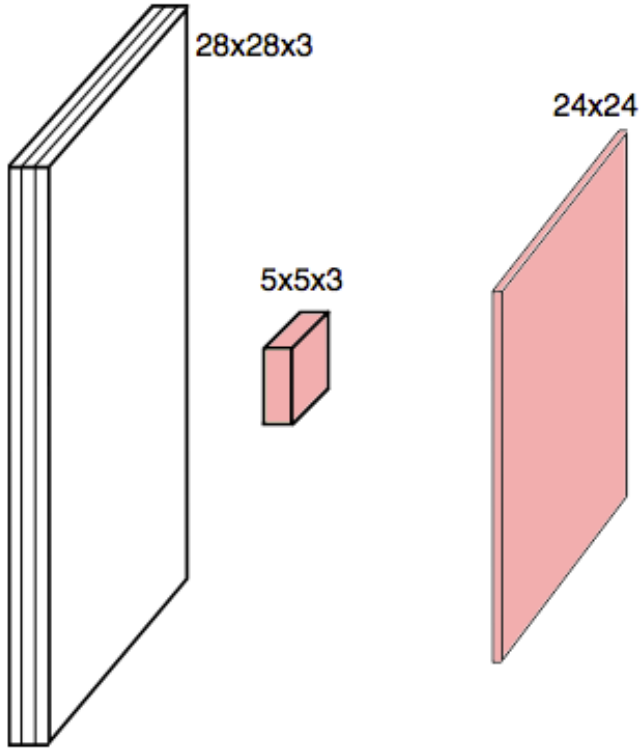

1/9 1/9 1/9
1/9 1/9 1/9
1/9 1/9 1/9
3 3 2 1 0
0 0 1 3 1
3 1 2 2 3
2 0 0 2 2
2 0 0 0 1
1.7 1.7 1.7
1.0 1.2 1.8
1.1 0.8 1.3
THE KEY IS AGAIN THAT
THE SAME WEIGHTS ARE
APPLIED TO ALL IMAGE
REGIONS
Σ

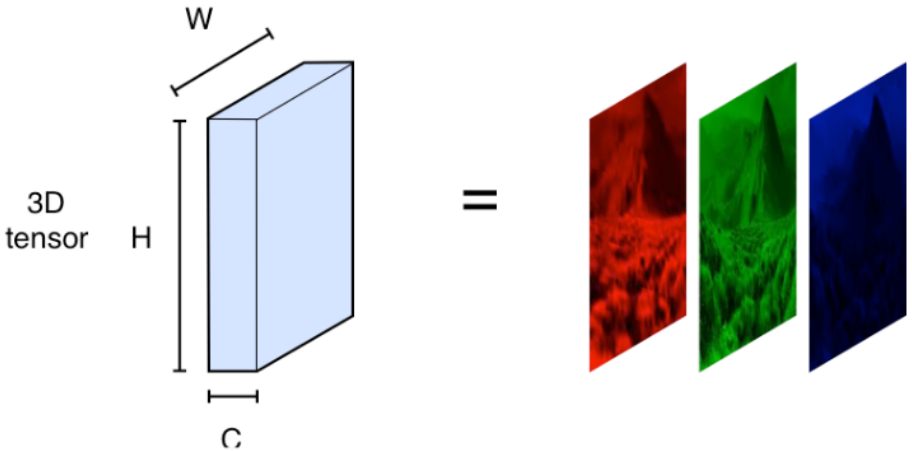

x_i
3 3 2 1 0
0 0 1 3 1
3 1 2 2 3
2 0 0 2 2
2 0 0 0 1
1/9 1/9 1/9
1/9 1/9 1/9
1/9 1/9 1/9
w_i
[weights]
w_i
ACTIVATION FUNCTION
AT EVERY KERNEL POSITION
x_i
$\sum$
$z(\mathbf{x}) = relu(w\mathbf{x} + b)$


x_i
3 3 2 1 0
0 0 1 3 1
3 1 2 2 3
2 0 0 2 2
2 0 0 0 1
1/9 1/9 1/9
1/9 1/9 1/9
1/9 1/9 1/9
w_i
[weights]
w_i
x_i
ACTIVATION FUNCTION
AT EVERY KERNEL POSITION
IN DEEP NETWORKS ReLU is the most commonly used activation function - see Vanishing Gradient Problem
$z(\mathbf{x}) = relu(w\mathbf{x} + b)$

CONVOLUTIONS CAN ALSO BE COMPUTED ACROSS CHANNELS (OR COLORS)

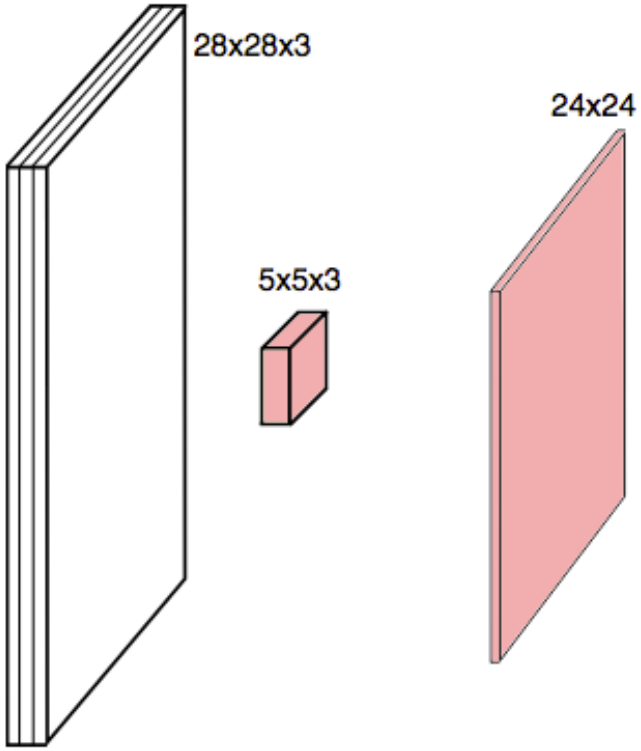
A COLOR IMAGE IS A TENSOR
OF SIZE height x width x channels

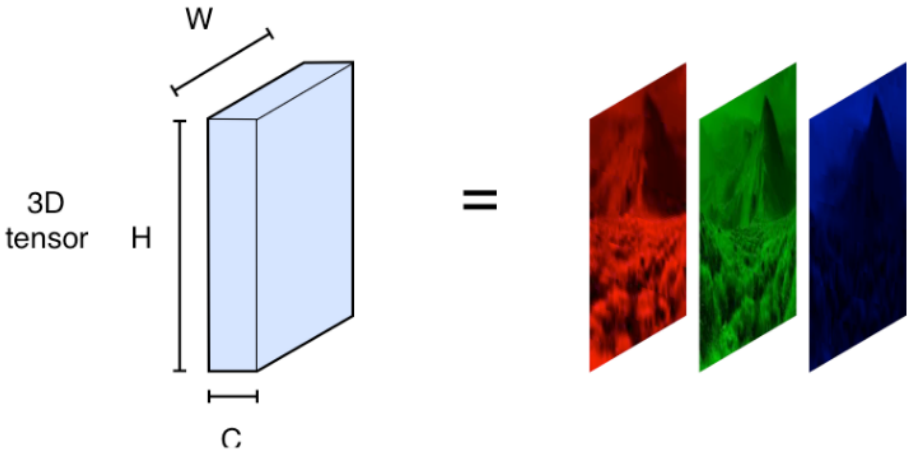




CONVOLUTIONS CAN ALSO BE COMPUTED ACROSS CHANNELS (OR COLORS)

A COLOR IMAGE IS A TENSOR
OF SIZE height x width x channels



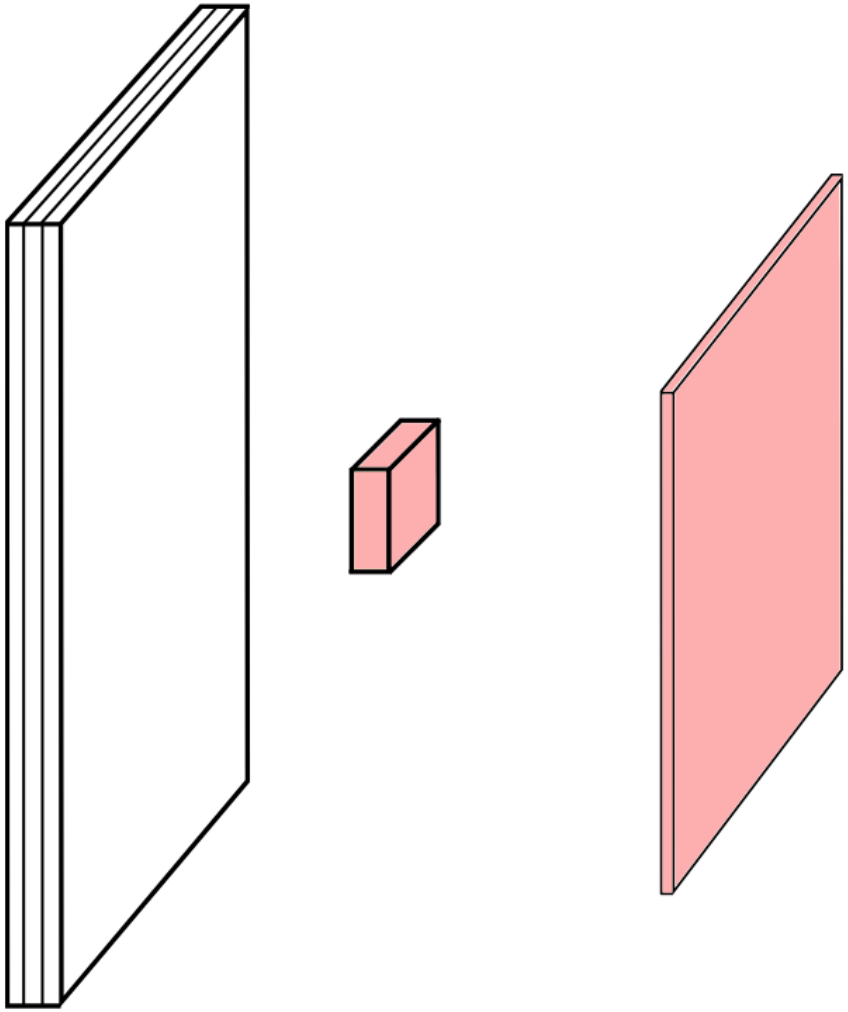


THEN THE KERNEL HAS ALSO 3 CHANNELS

IN ASTRONOMY …

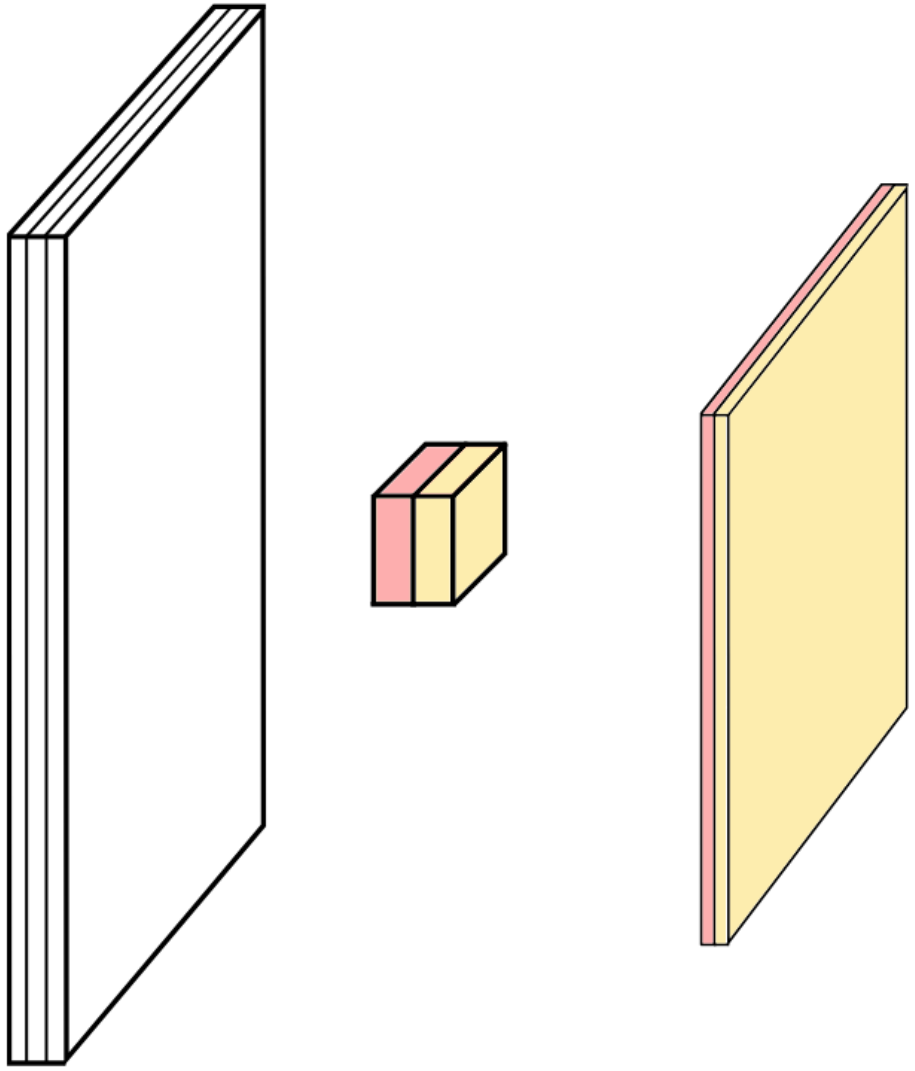
IT OPENS THE DOOR TO ANALYZE MULTIPLE FILTERS SIMULTANEOUSLY

MULTIPLE CONVOLUTIONS WITH DIFFERENT KERNELS CAN BE PERFORMED



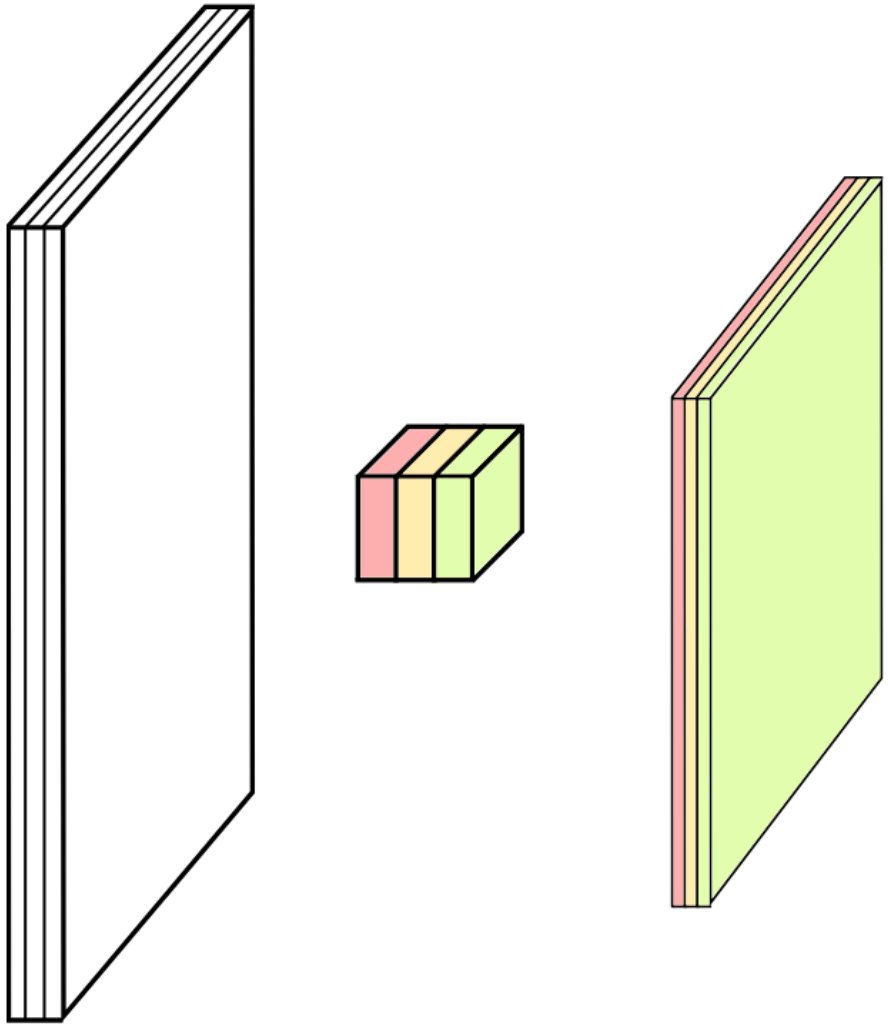
credit

MULTIPLE CONVOLUTIONS WITH DIFFERENT KERNELS CAN BE PERFORMED



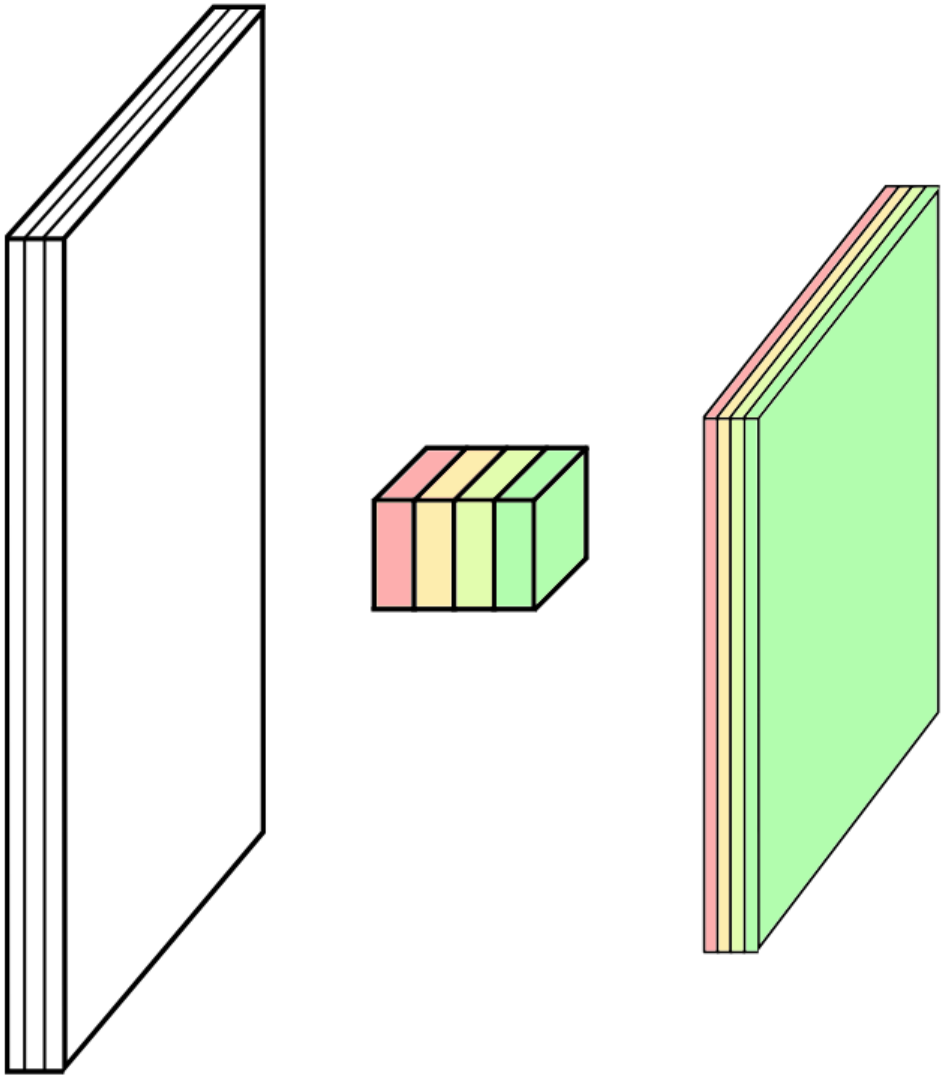
credit

MULTIPLE CONVOLUTIONS WITH DIFFERENT KERNELS CAN BE PERFORMED



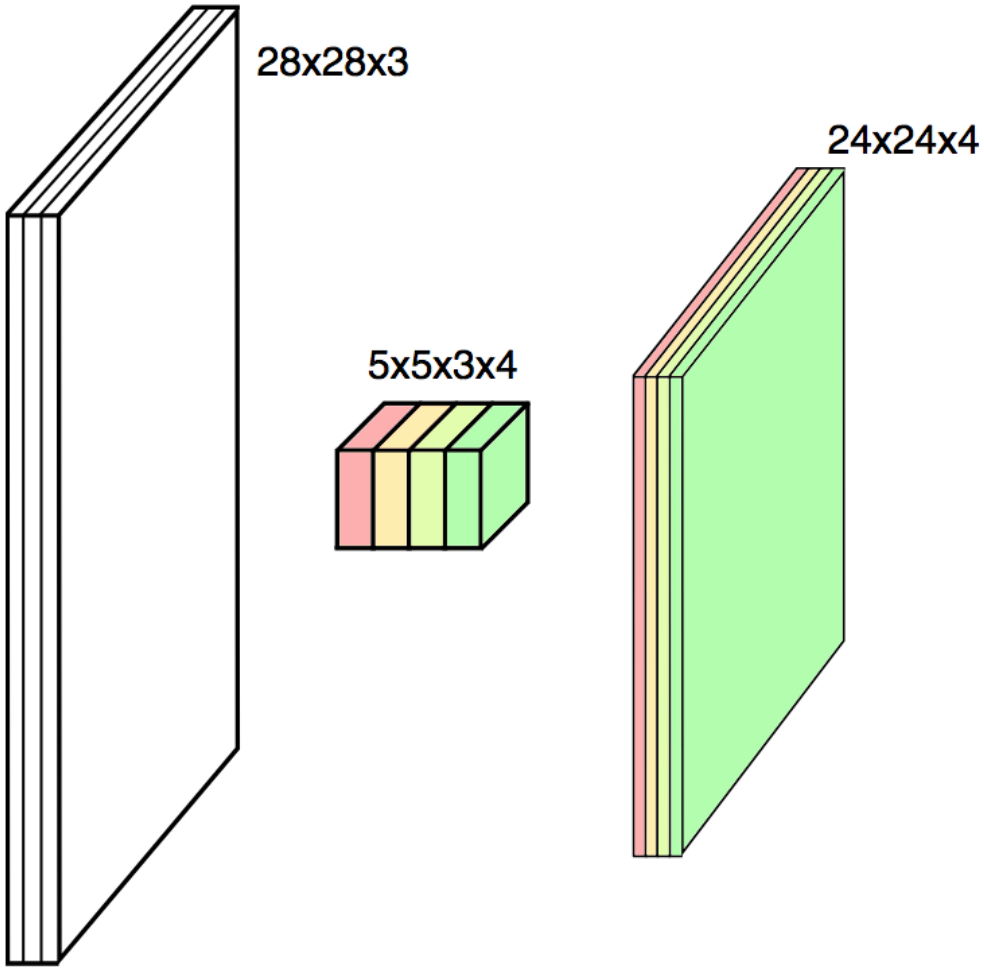
credit

MULTIPLE CONVOLUTIONS WITH DIFFERENT KERNELS CAN BE PERFORMED



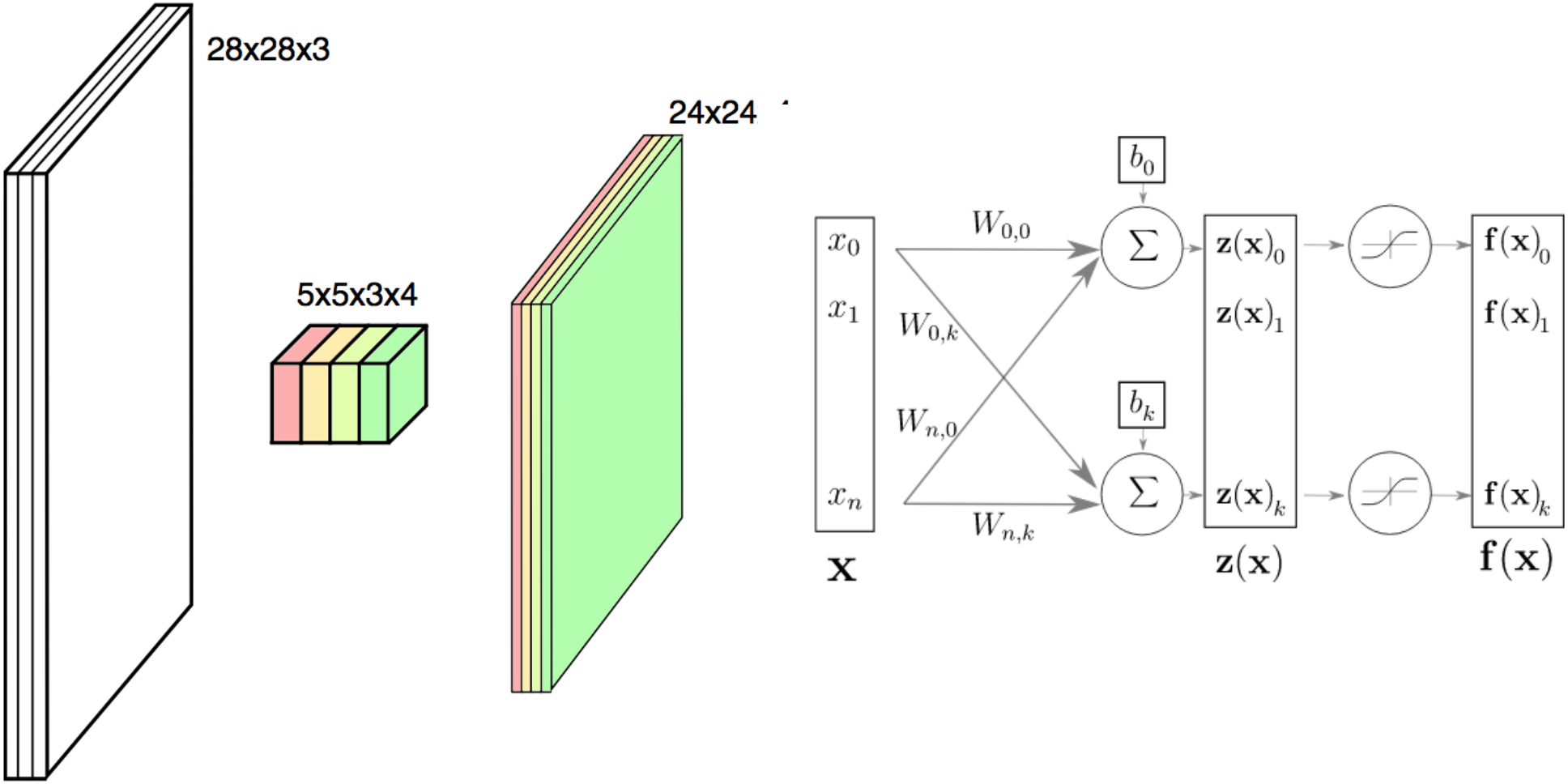
credit

MULTIPLE CONVOLUTIONS WITH DIFFERENT KERNELS CAN BE PERFORMED



credit

MULTIPLE CONVOLUTIONS WITH DIFFERENT KERNELS CAN BE PERFORMED

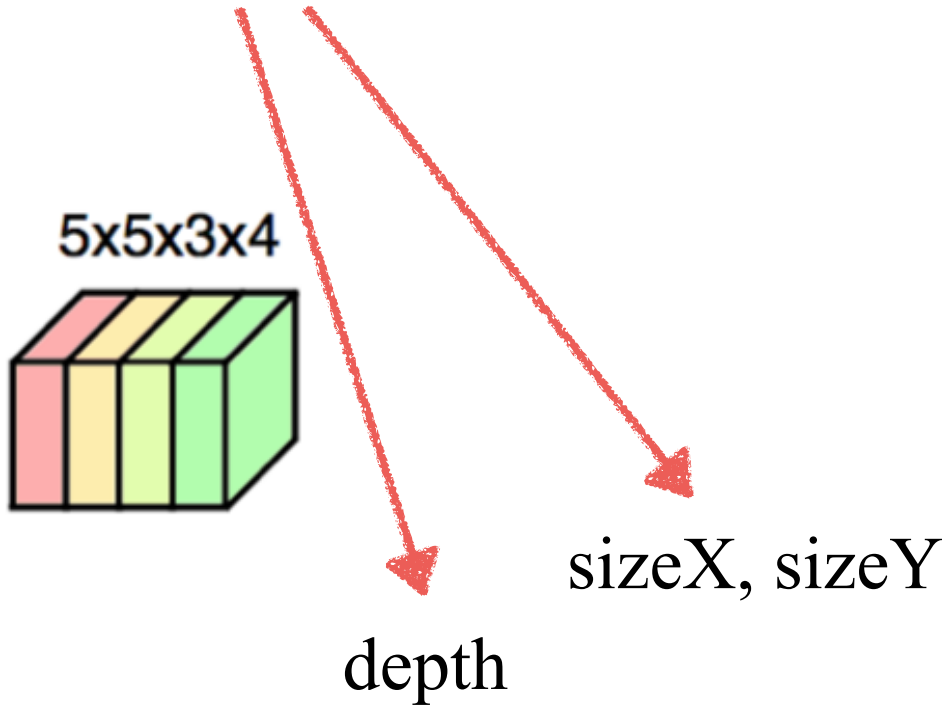


credit

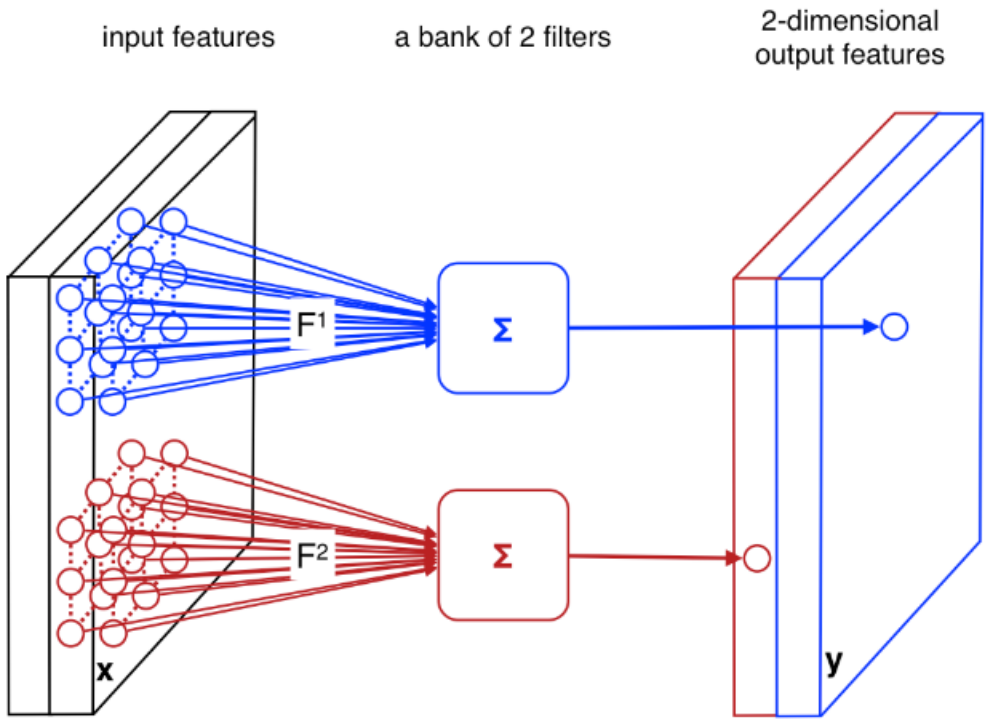
IN KERAS…

model = Sequential()

model.add(Convolution2D(4,5,5, activation="relu"))



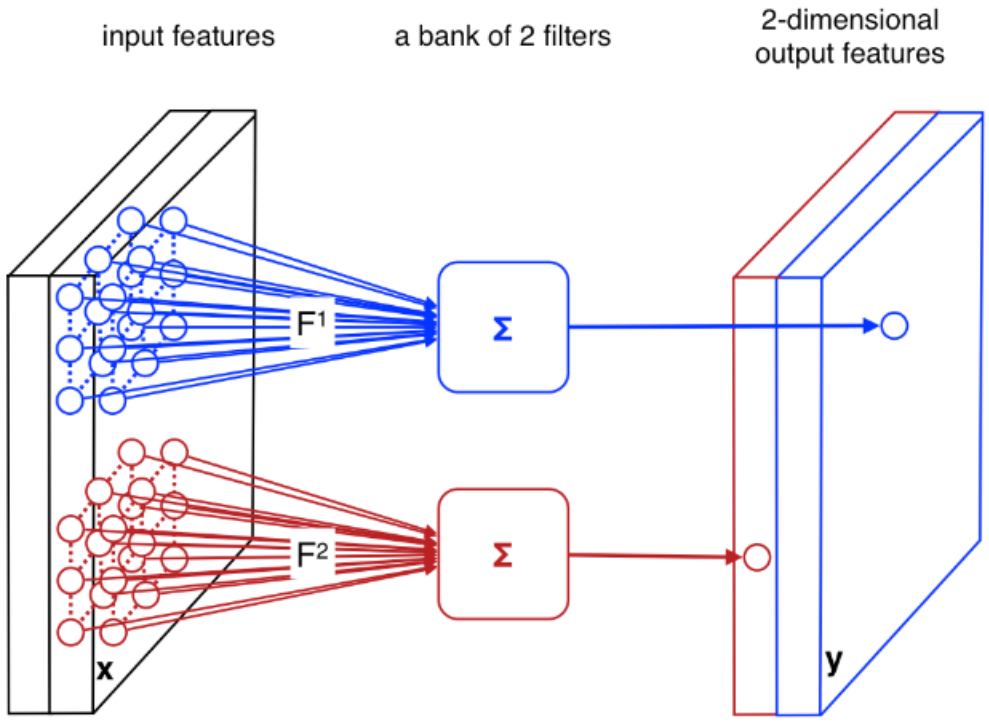
SINCE CONVOLUTIONS OUTPUT ONE SCALAR, THEY CAN BE SEEN AS AN INDIVIDUAL NEURON WITH A RECEPTIVE FIELD LIMITED TO THE KERNEL DIMENSIONS



Credit

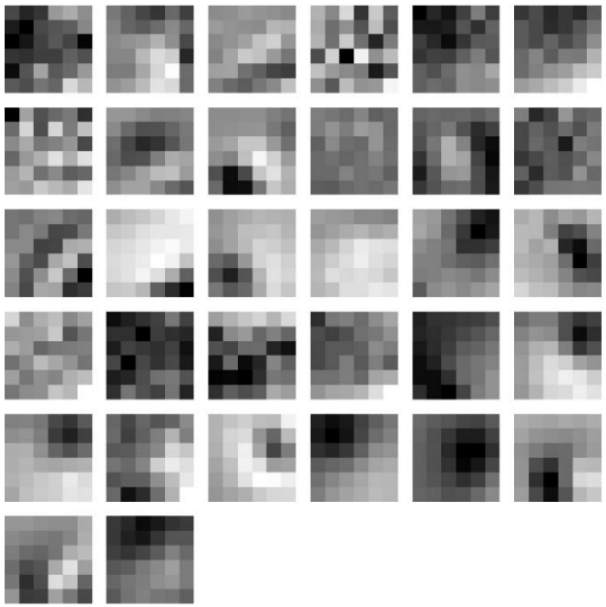
SINCE CONVOLUTIONS OUTPUT ONE SCALAR< THEY CAN BE SEEN AS AN INDIVIDUAL NEURON WITH A RECEPTIVE FIELD LIMITED TO THE KERNEL DIMENSIONS

THE SAME NEURON IS FIRED WITH DIFFERENT AREAS FROM THE INPUT

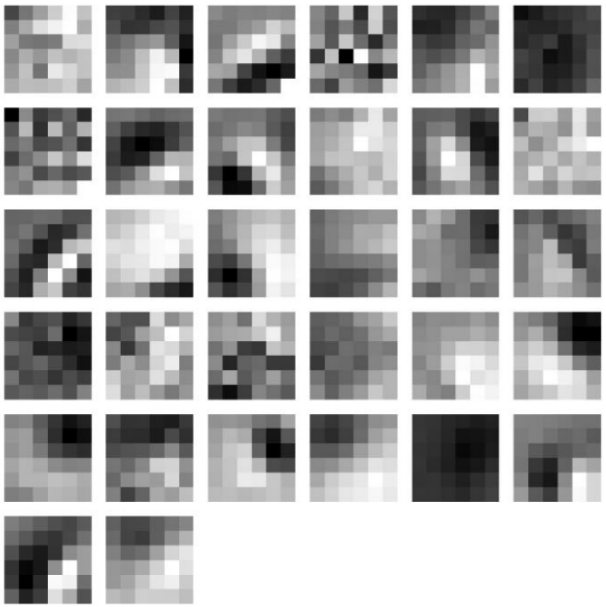


Credit

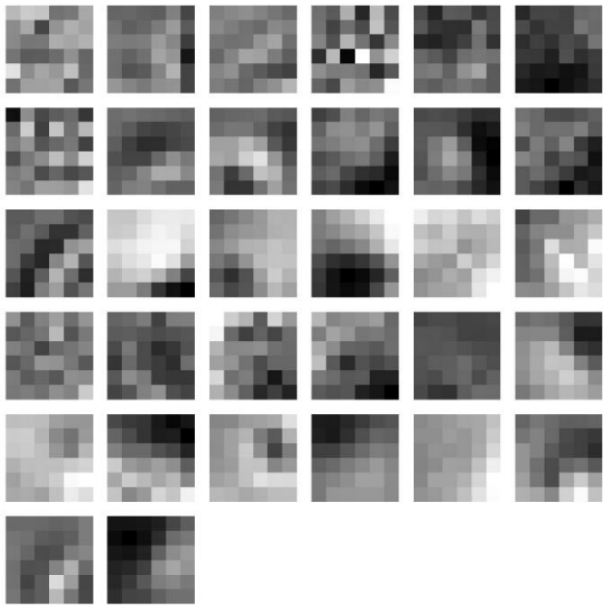
EXAMPLE OF 32 FILTERS LEARNED IN A CONVOLUTIONAL LAYER



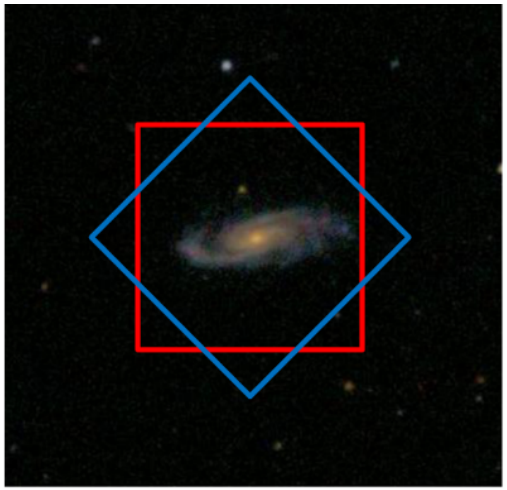
(a) red channel



(b) green channel

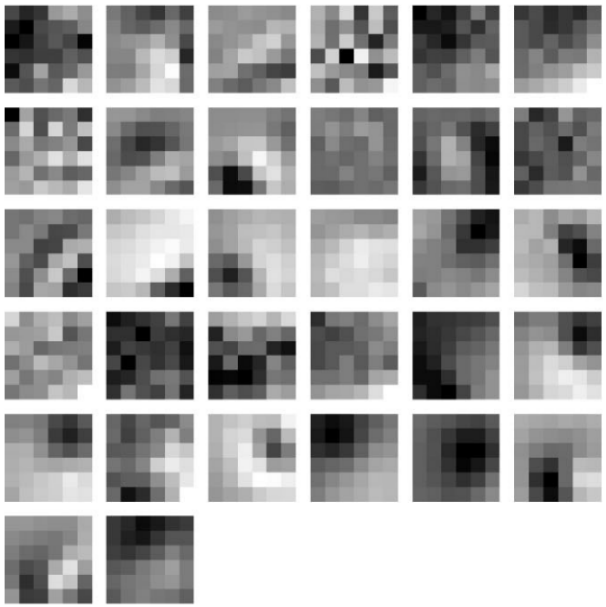


(c) blue channel

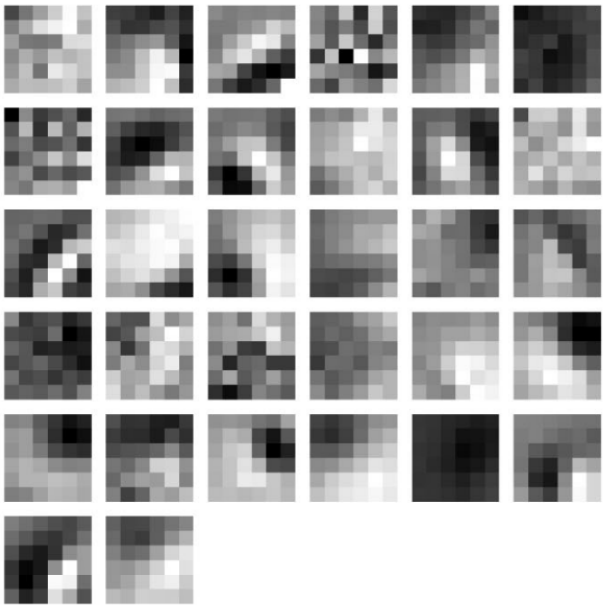


Dieleman+16

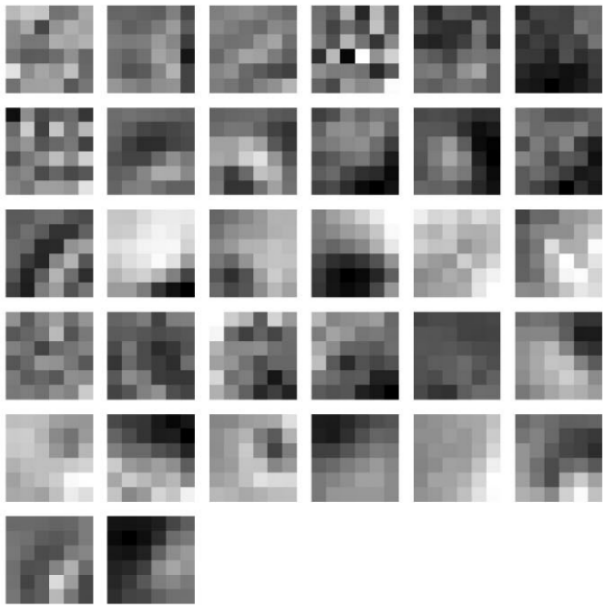
EXAMPLE OF 32 FILTERS LEARNED IN A CONVOLUTIONAL LAYER



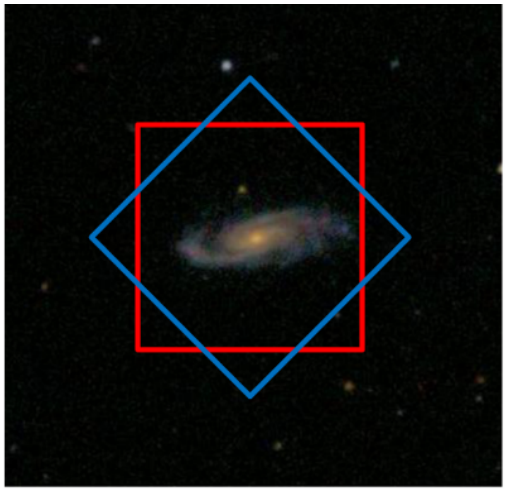
(a) red channel



(b) green channel



(c) blue channel

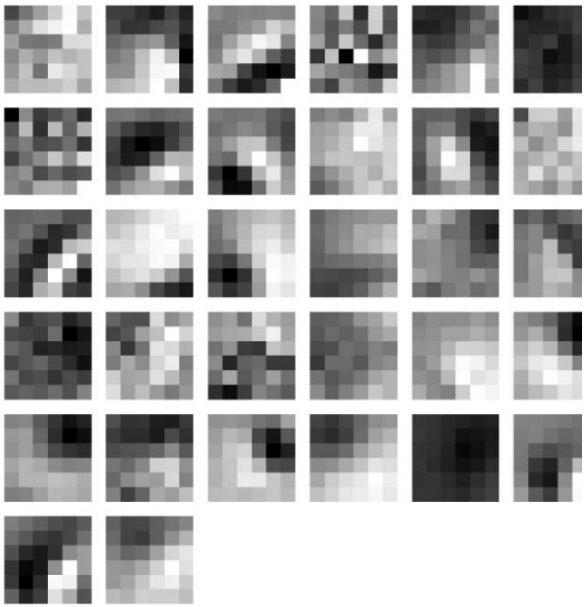


Dieleman+16

THESE ARE CALLED **FEATURE MAPS**

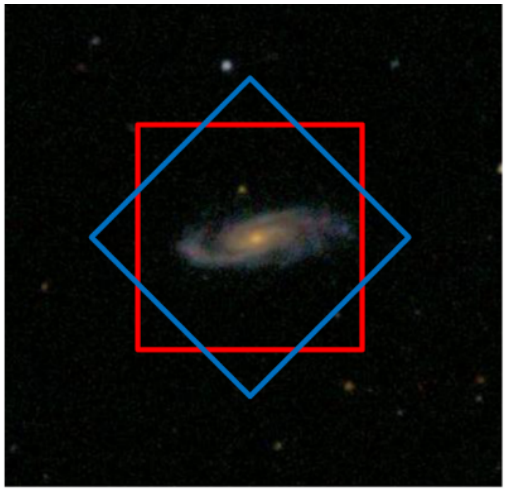
EXAMPLE OF 32 FILTERS LEARNED IN A CONVOLUTIONAL

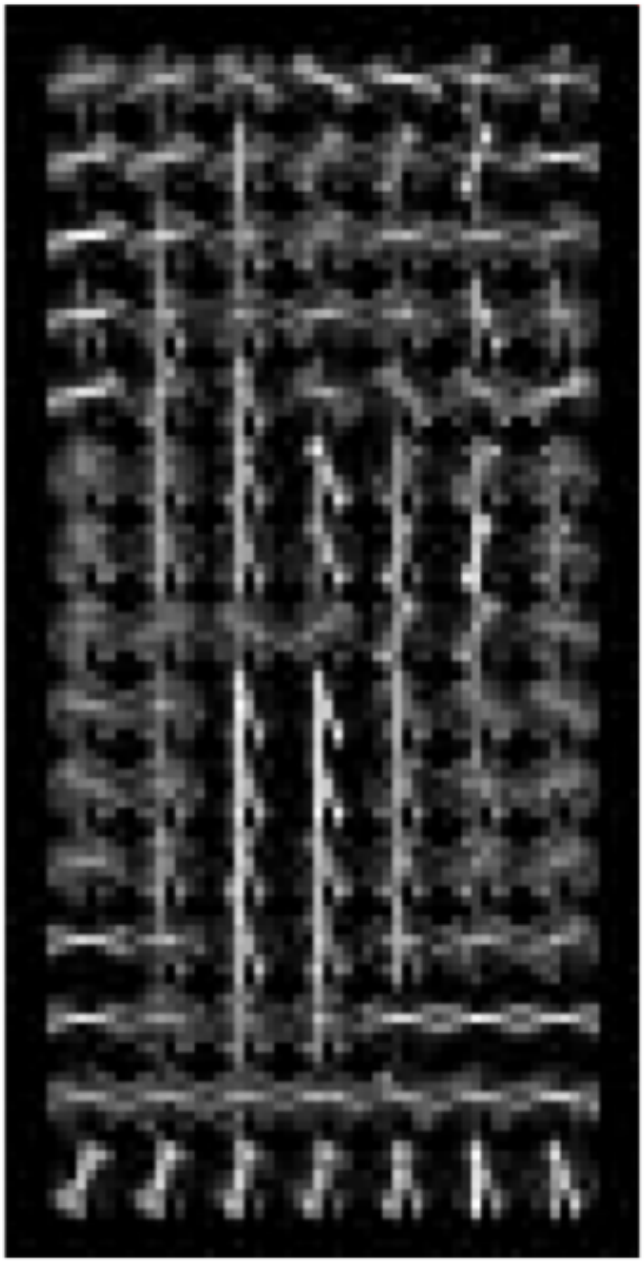
(a) red channel



(b) green channel

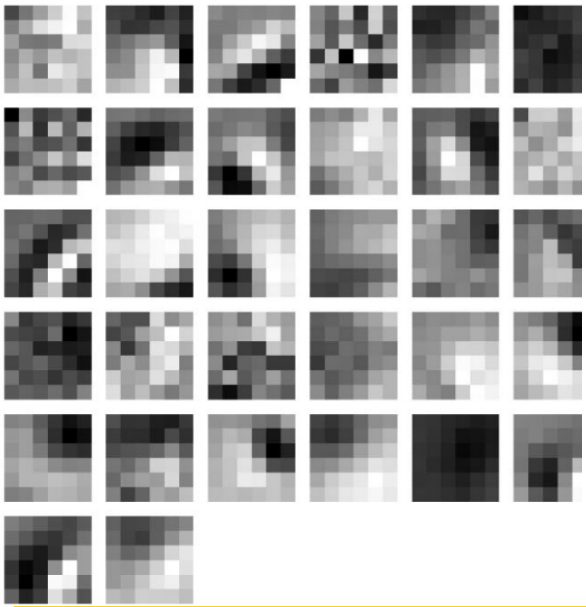
REMEMBER THE HoG ?

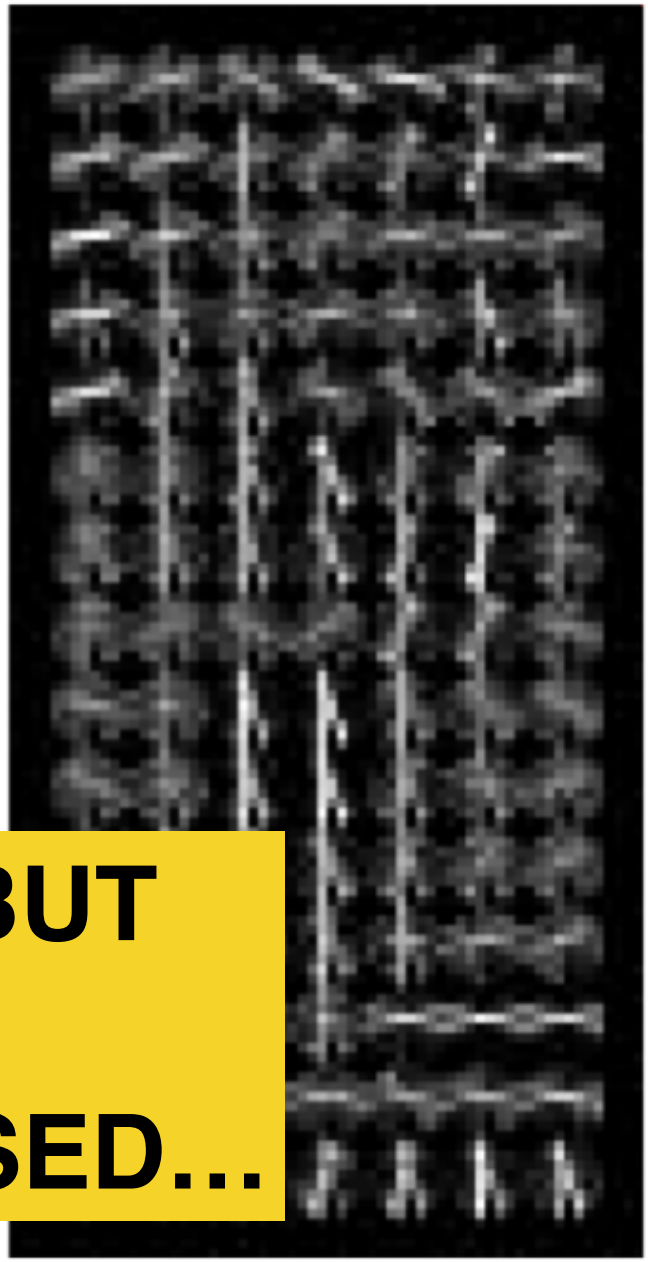




THESE ARE CALLED **FEATURE MAPS**

EXAMPLE OF 32 FILTERS LEARNED BY A CONVOLUTIONAL

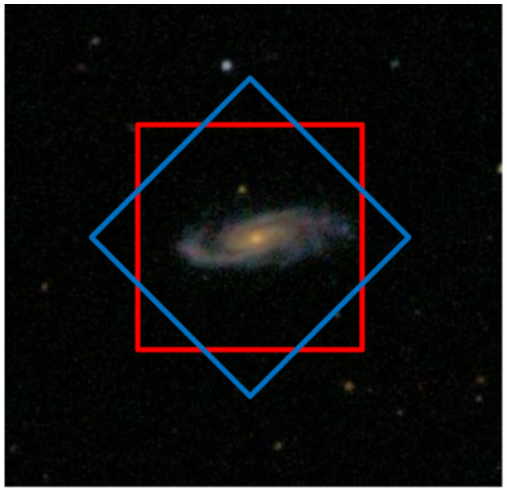




REMEMBER THE HoG ?

SIMILAR, BUT FULLY UNSUPERVISED…

(a) red channel



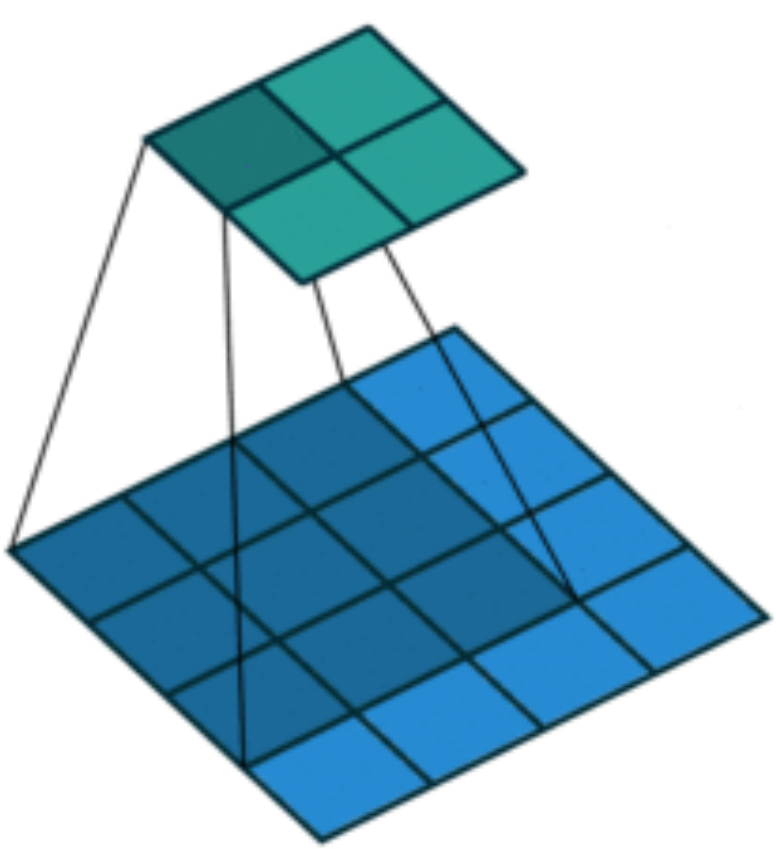
THESE ARE CALLED **FEATURE MAPS**

ESTIMATING SHAPES AND NUMBER OF PARAMETERS

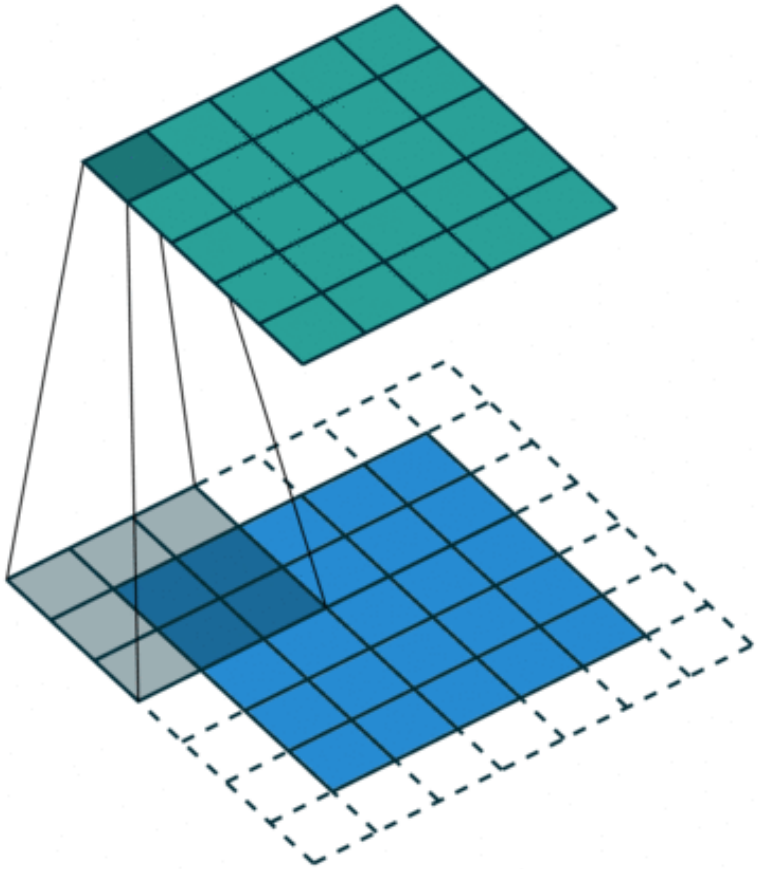
KERNEL SHAPE:	PADDING:	STRIDES:
(F, F, C^i, C^o)	P	S

OUTPUT SIZE: $W_0 = (W^i - F + 2P)/S + 1$

OPTIONS: **PADDING**



NO STRIDES



PADDING

ESTIMATING SHAPES AND NUMBER OF PARAMETERS

KERNEL SHAPE:	PADDING:	STRIDES:
(F, F, C^i, C^o)	P	S

OUTPUT SIZE: $W_0 = (W^i - F + 2P)/S + 1$

NUMBER OF PARAMETERS: $(F \times F \times C^i + 1) \times C^o$

ESTIMATING SHAPES AND NUMBER OF PARAMETERS

KERNEL SHAPE:	PADDING:	STRIDES:
(F, F, C^i, C^o)	P	S

OUTPUT SIZE: $W_0 = (W^i - F + 2P)/S + 1$

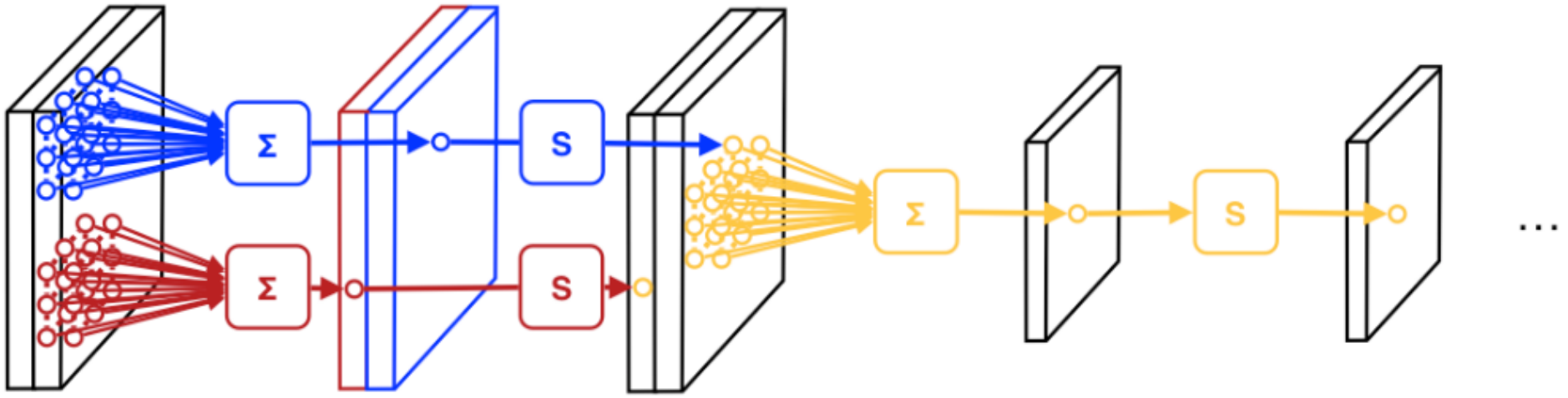
NUMBER OF PARAMETERS: $(F \times F \times C^i + 1) \times C^o$

→ the number of parameters increases fast!

32 filters of 5*5 on a color image —> 2432 parameters to learn

DOWNSAMPLING

DOWNSAMPLING IS APPLIED TO REDUCE THE OVERALL SIZE OF TENSORS



ESTIMATING SHAPES AND NUMBER OF PARAMETERS

KERNEL SHAPE:	PADDING:	STRIDES:
(F, F, C^i, C^o)	P	S

OUTPUT SIZE: $W_0 = (W^i - F + 2P)/S + 1$

NUMBER OF PARAMETERS: $(F \times F \times C^i + 1) \times C^o$

→ the number of parameters increases fast!

POOLING

CONVOLUTIONS ARE OFTEN FOLLOWED BY AN OPERATION OF DOWNSAMPLING [POOLING]

VERY SIMPLE OPERATION - ONLY ONE OUT OF EVERY N PIXELS ARE KEPT

OFTEN MATCHED WITH AN INCREASE OF THE FEATURE CHANNELS

TYPES OF POOLING

SUM POOLING $y = \sum x_{uv}$

SQUARE SUM POOLING $y = \sqrt{\sum x_{uv}^2}$

MAX POOLING $y = max(x_{uv})$

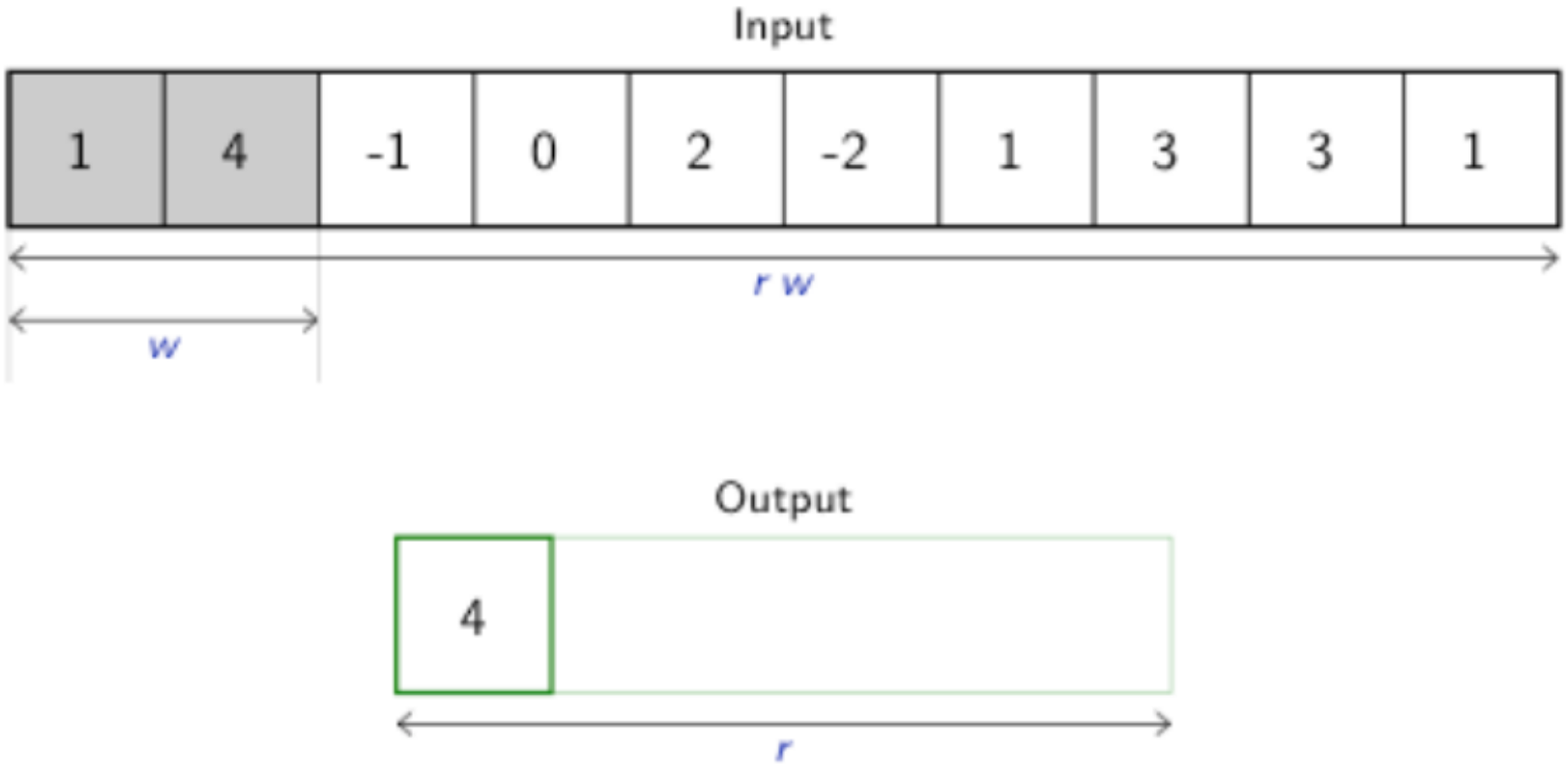
TYPES OF POOLING

SUM POOLING $y = \sum x_{uv}$

SQUARE SUM POOLING $y = \sqrt{\sum x_{uv}^2}$

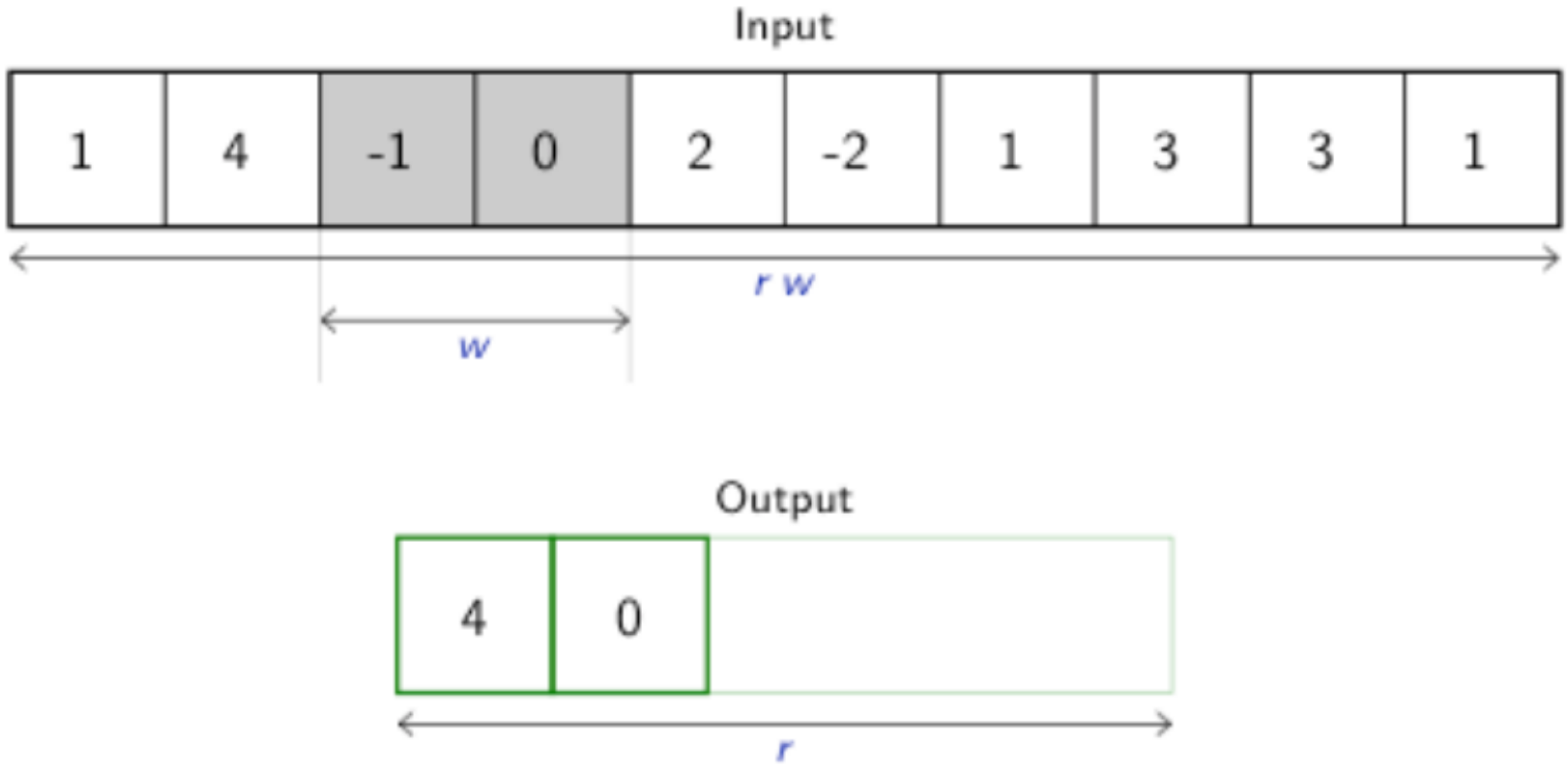
MAX POOLING $y = max(x_{uv})$

MAX POOLING 1D



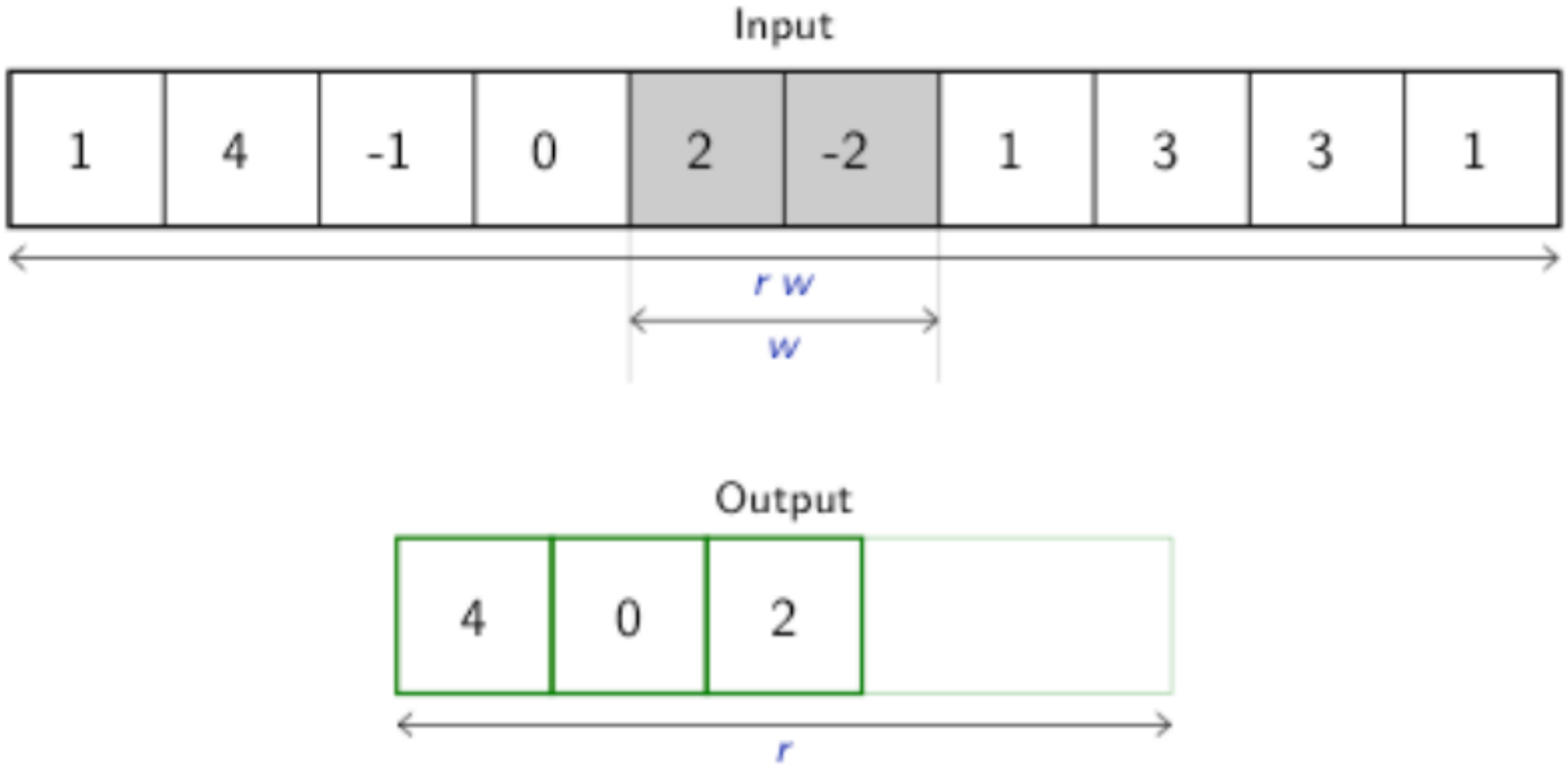
Credit: F. Fleuret

MAX POOLING 1D



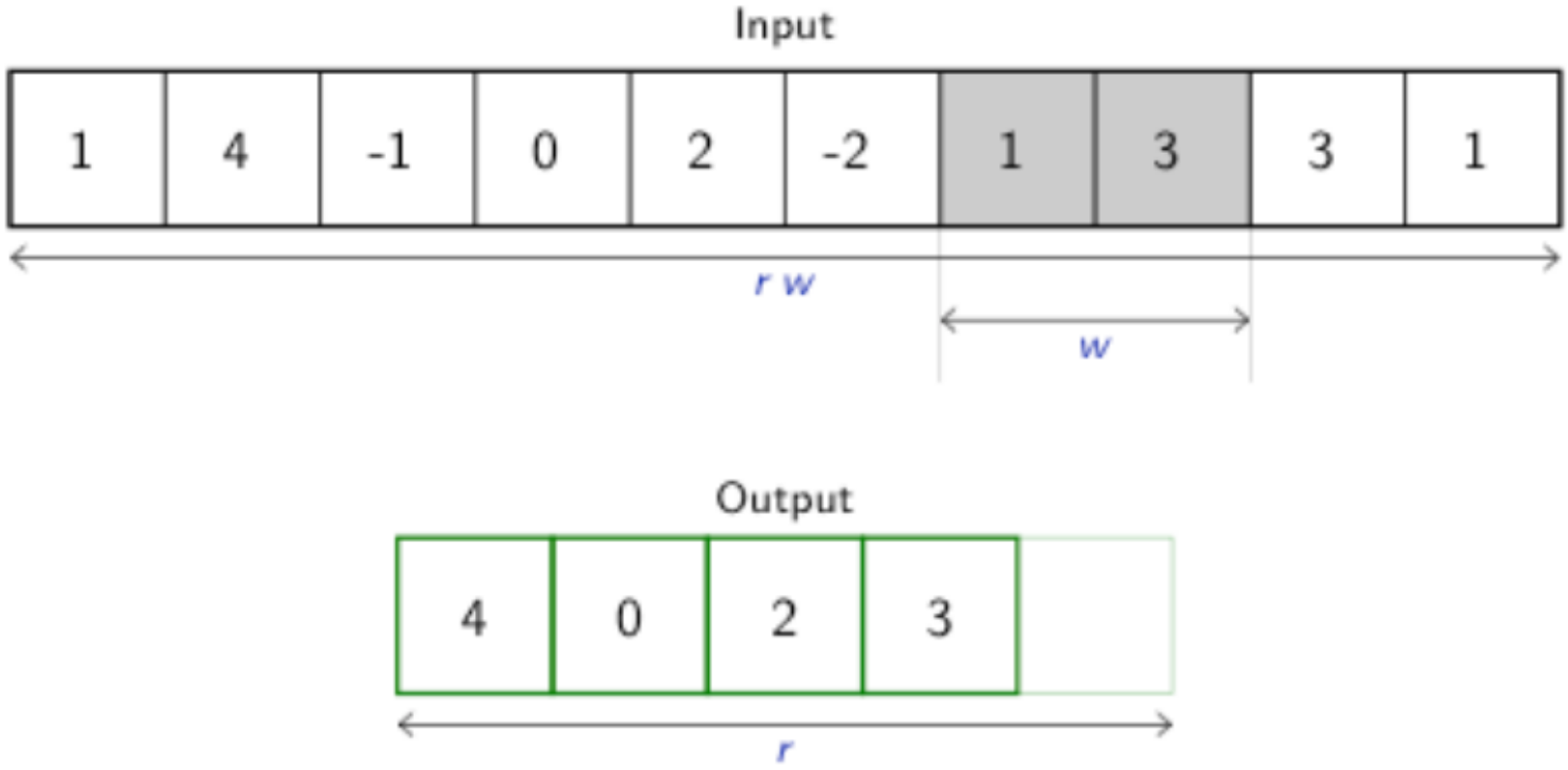
Credit: F. Fleuret

MAX POOLING 1D



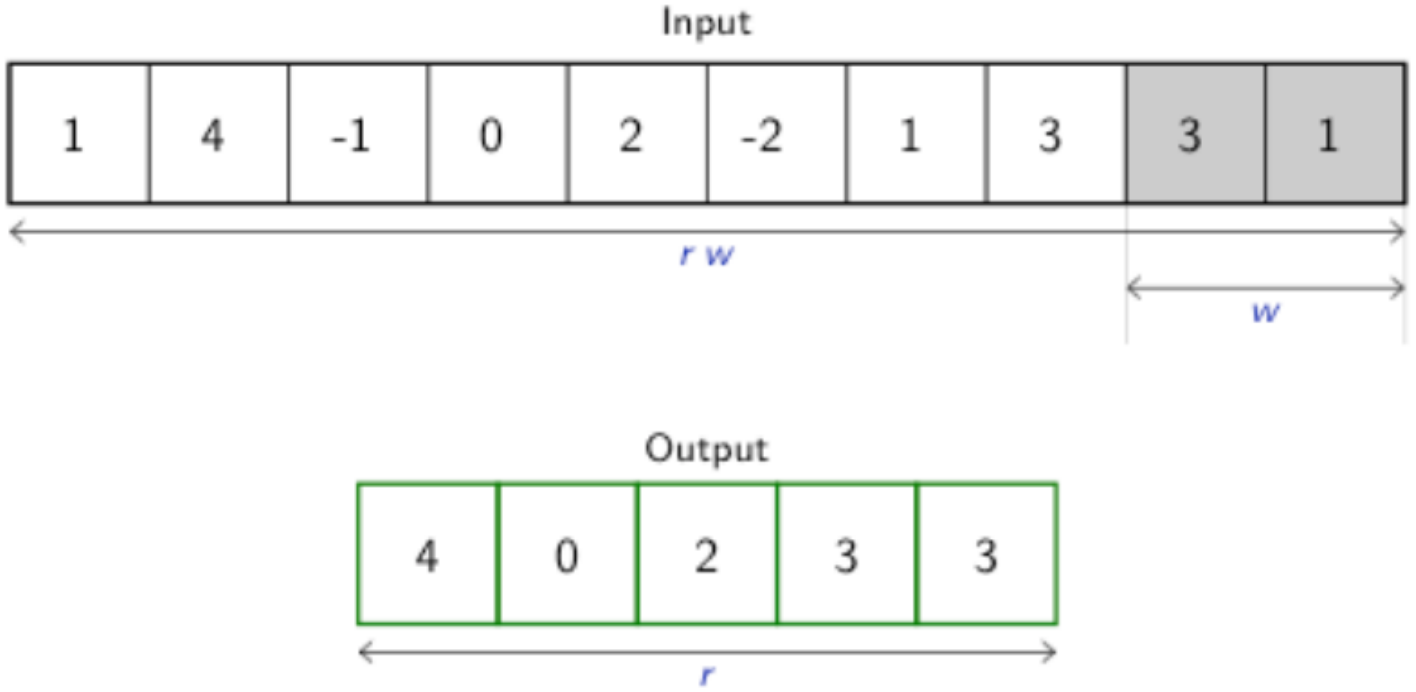
Credit: F. Fleuret

MAX POOLING 1D



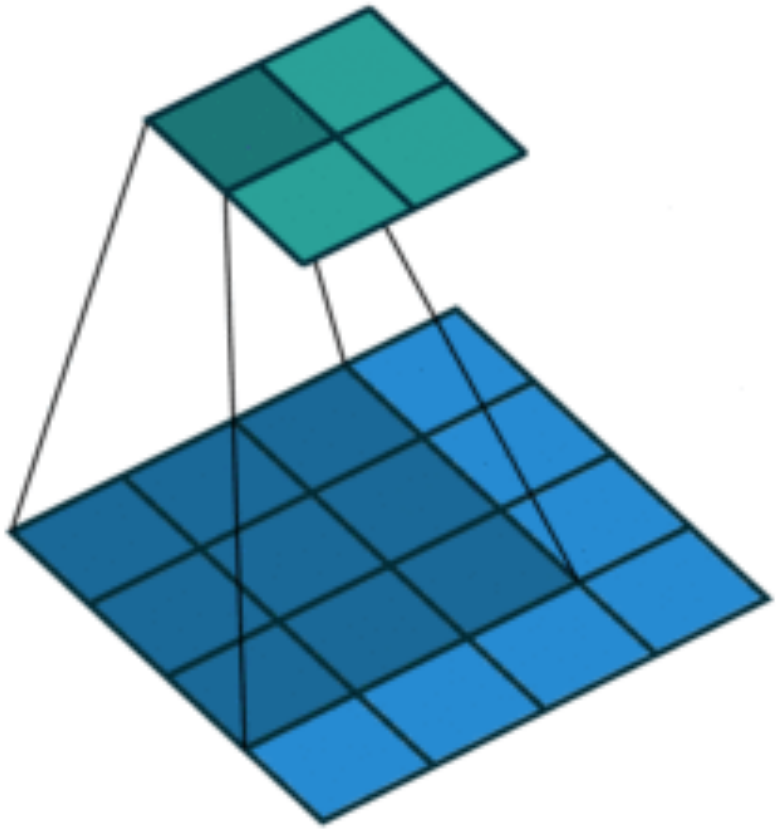
Credit: F. Fleuret

MAX POOLING 1D

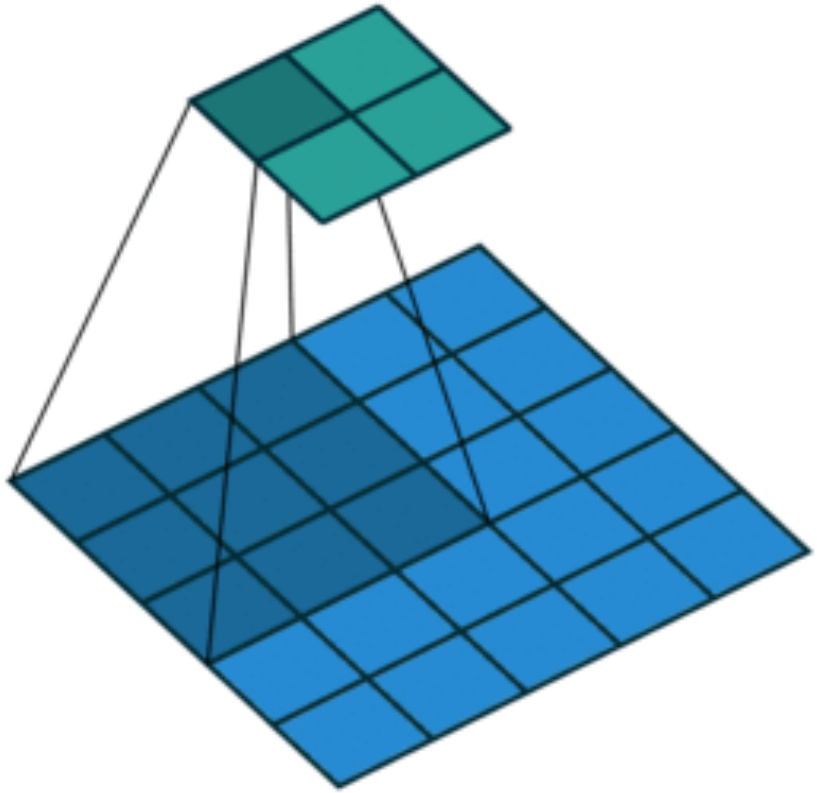


Credit: F. Fleuret

OPTIONS: **STRIDES**

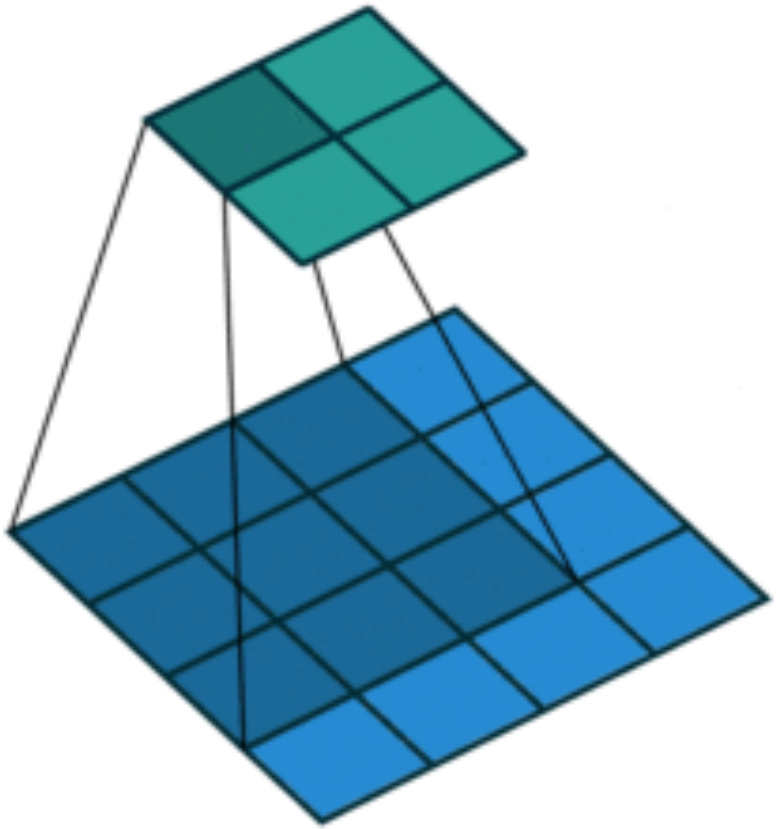


NO STRIDES

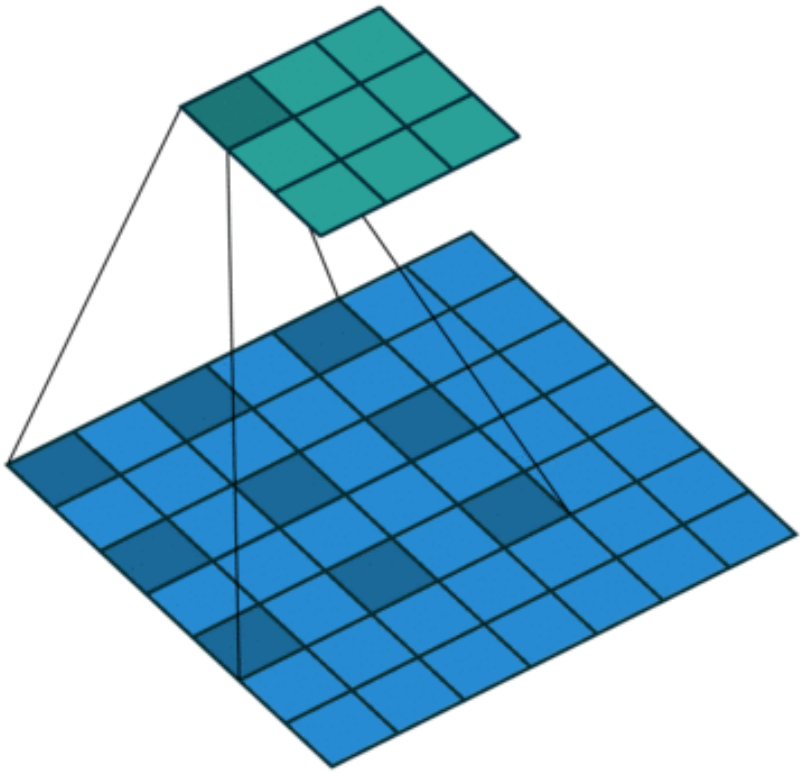


STRIDES

OPTIONS: **DILATION**

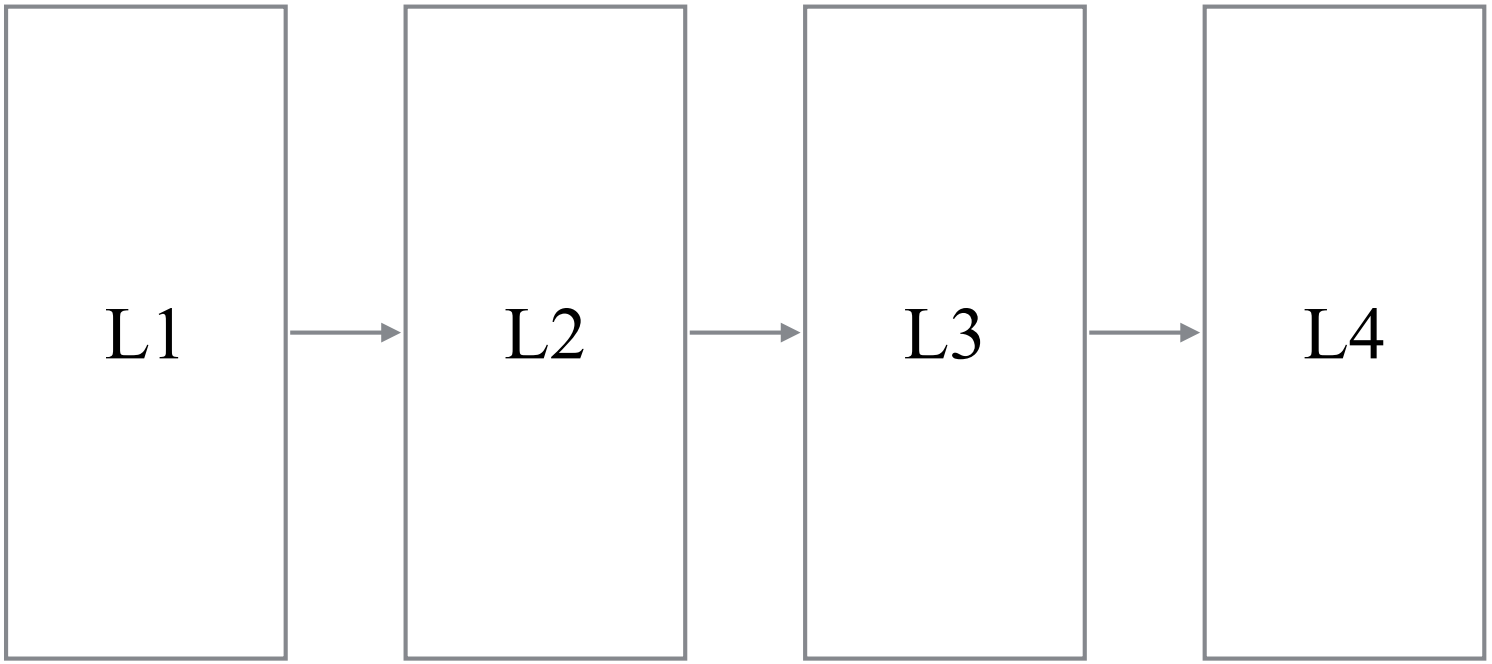


NO STRIDES



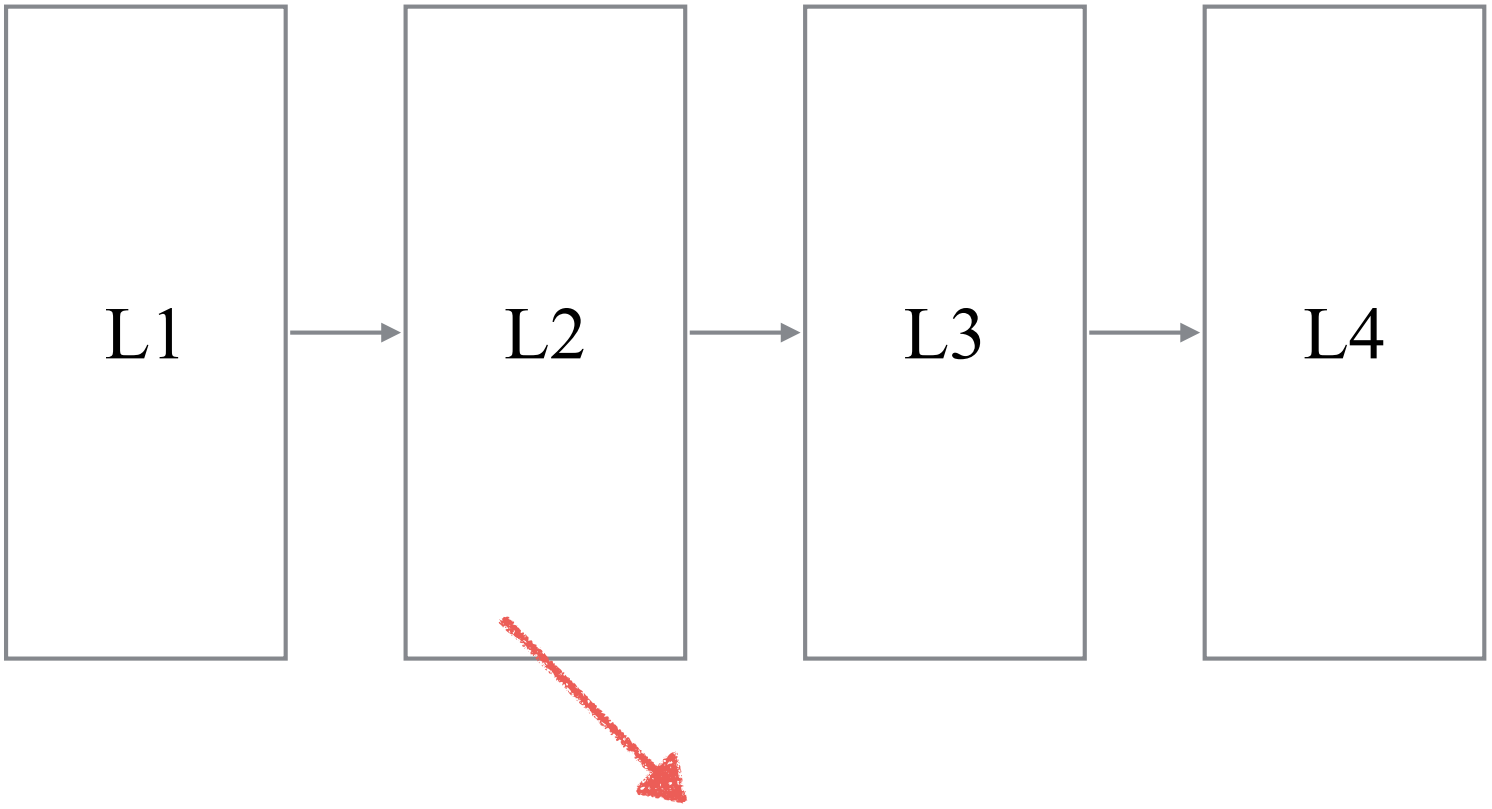
DILATION

CONVNET OR CNN



A CONCATENATION OF MULTIPLE CONVOLUTIONAL BLOCKS

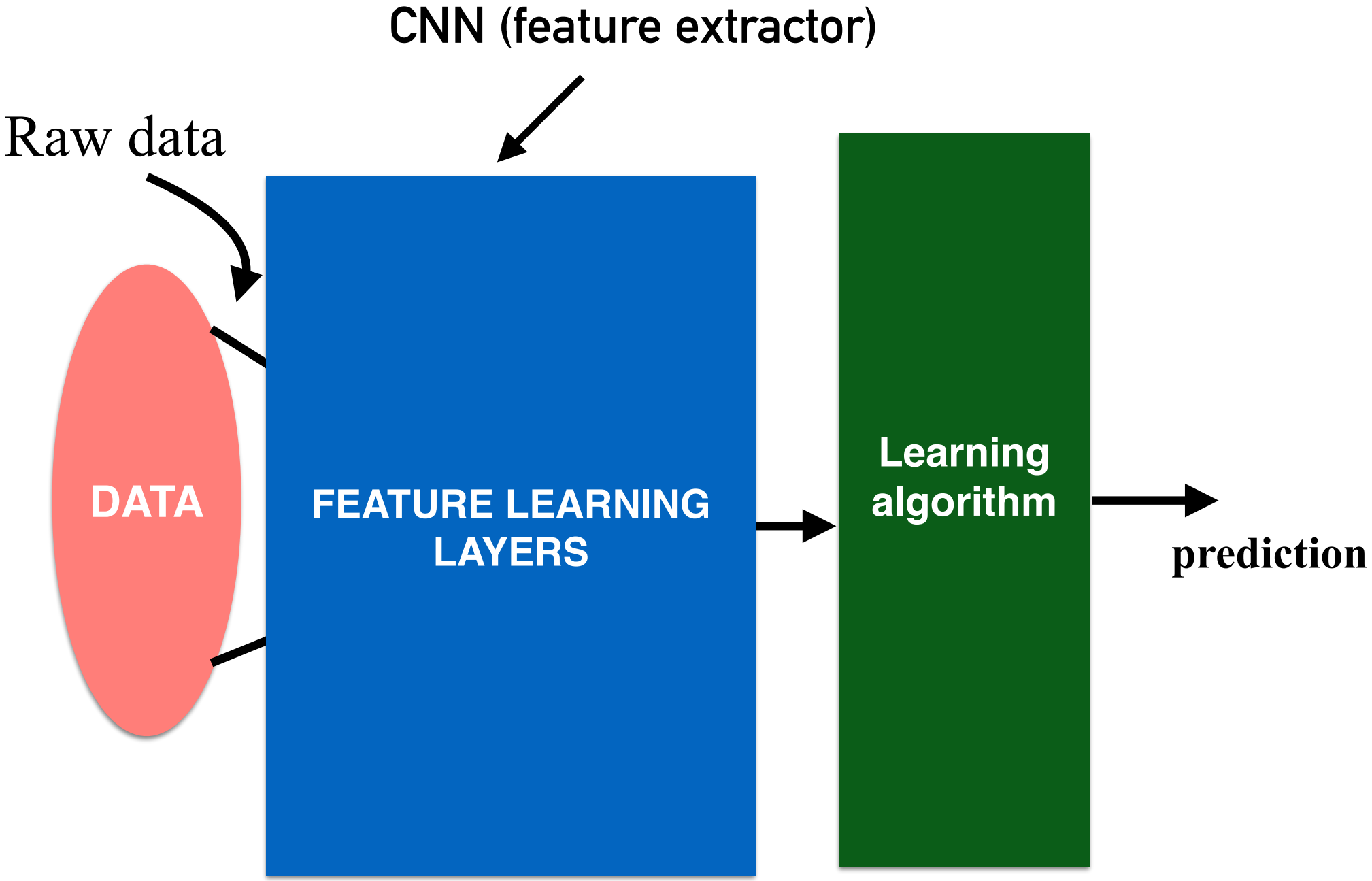
CONVNET OR CNN



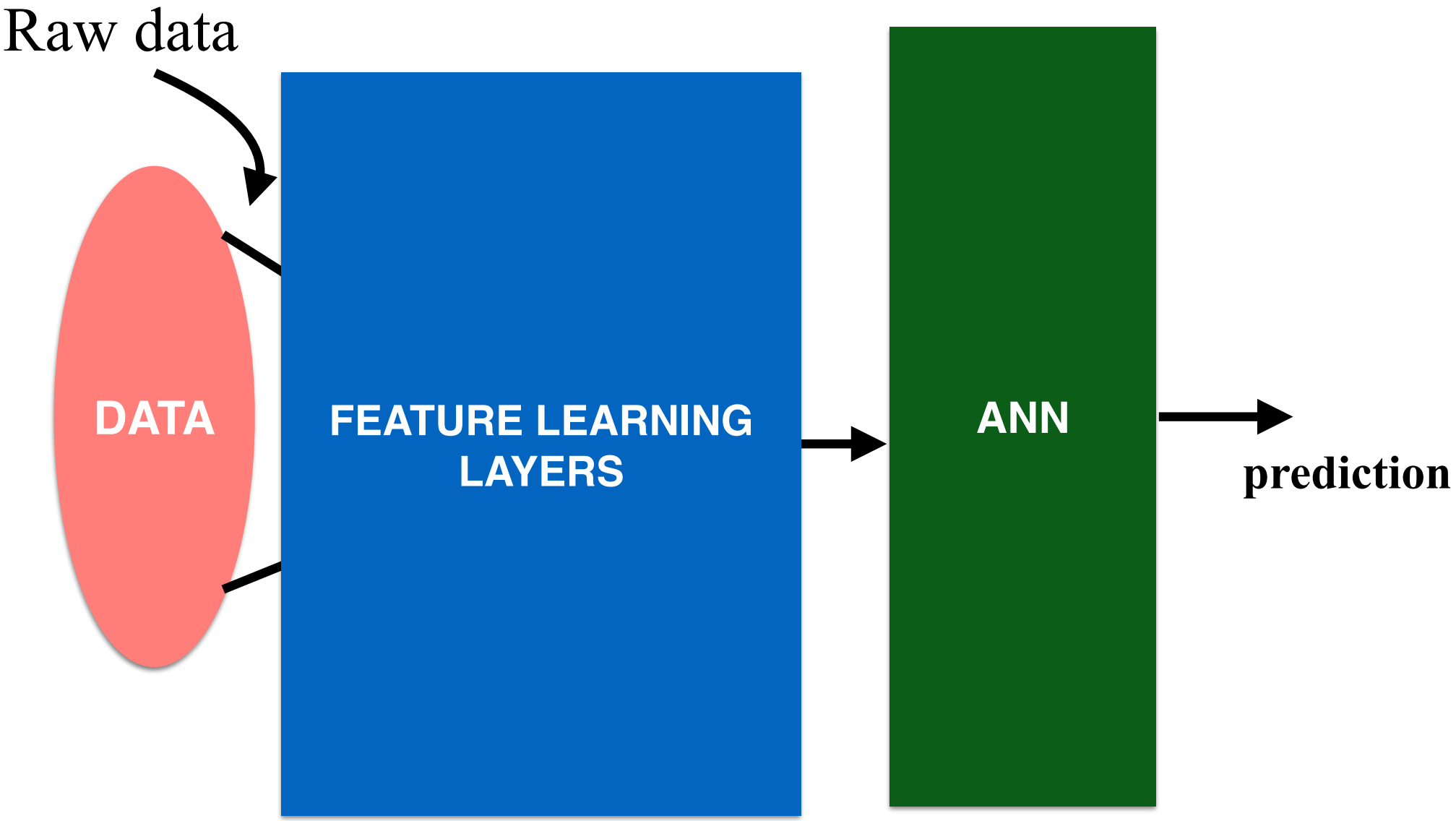
EACH BLOCK TYPICALLY MADE OF:

CONV | ACTIVATION | POOLING

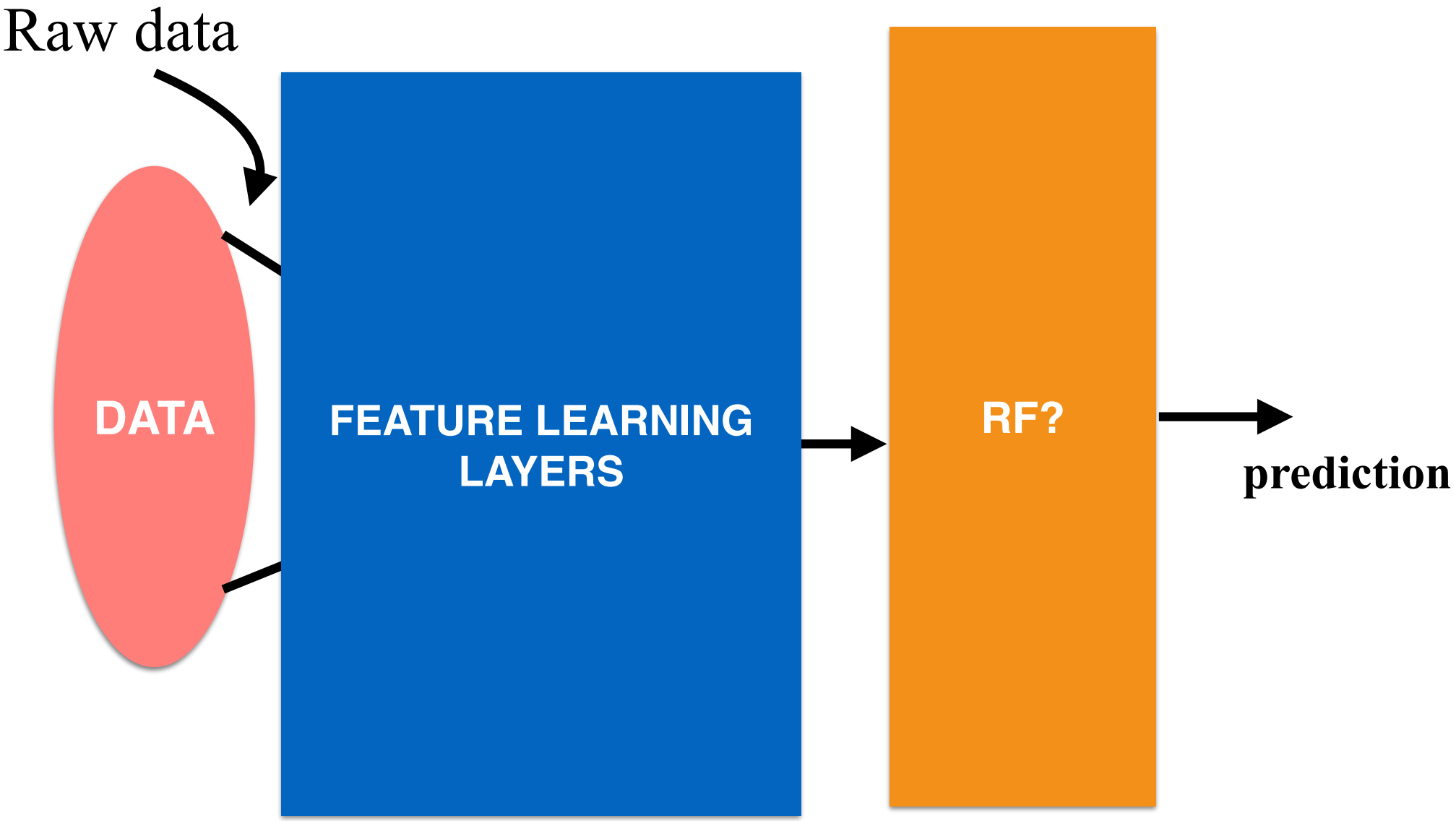
(+dropout for training)


CNN (feature extractor)
Raw data
DATA
FEATURE LEARNING LAYERS
Learning algorithm
prediction

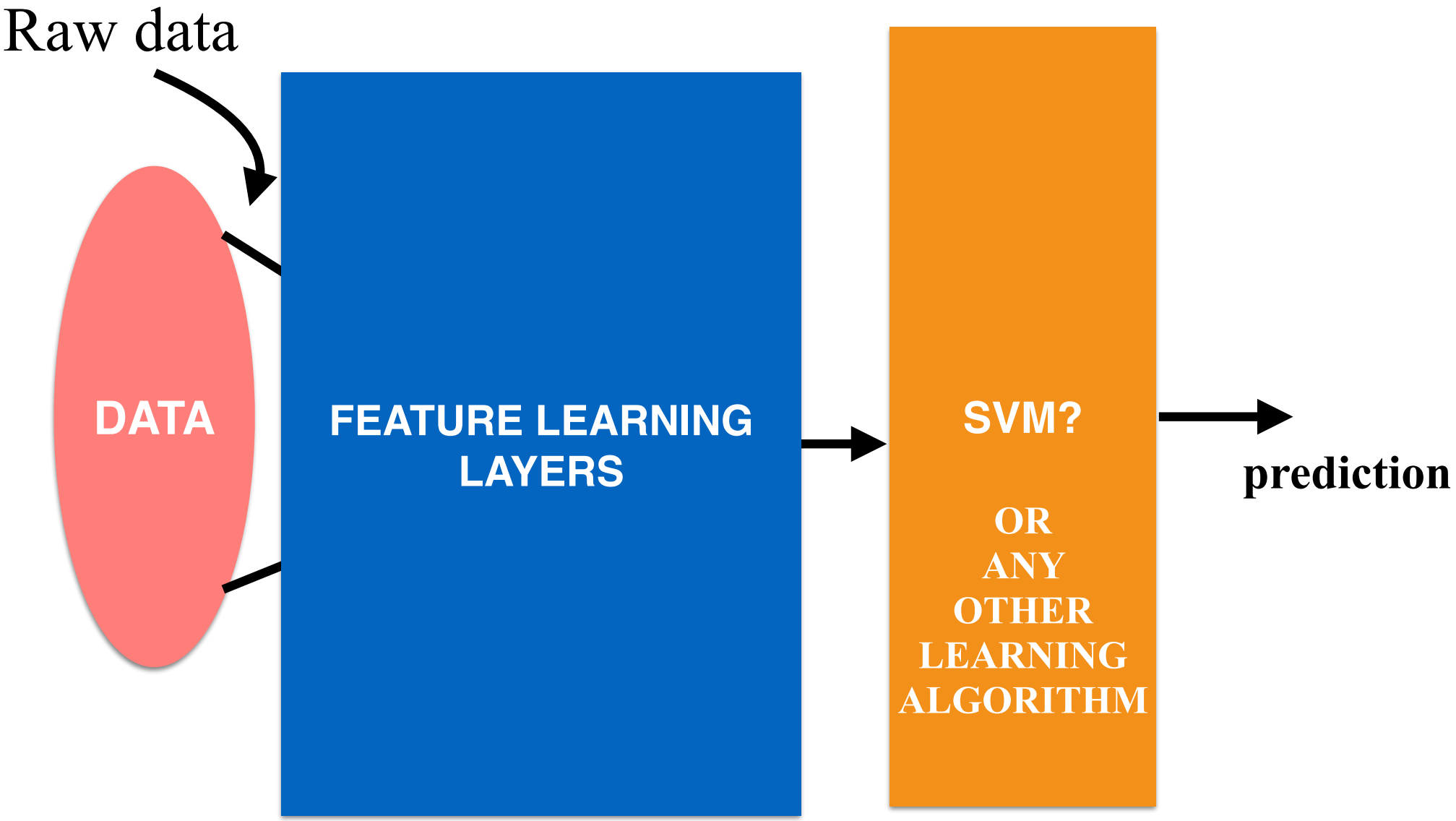
THE LEARNING ALGORITHM CAN BE CHANGED



THE LEARNING ALGORITHM CAN BE CHANGED



THE LEARNING ALGORITHM CAN BE CHANGED



THIS IS A CHANGE OF PARADIGM!