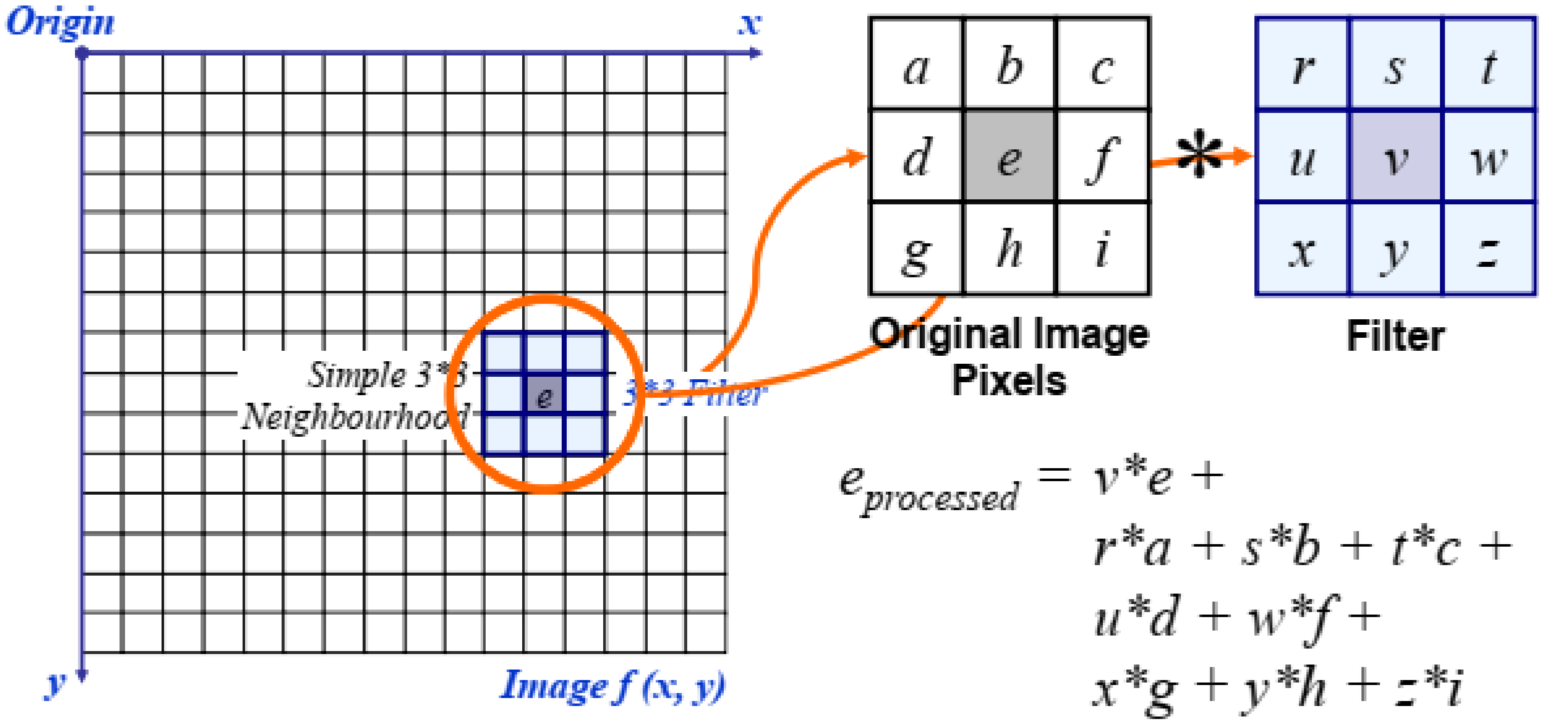
Introduction to CNNs

Concept of filtering



$$
\begin{aligned}
e_{processed} = {} & v*e + \\
& r*a + s*b + t*c + \\
& u*d + w*f + \\
& x*g + y*h + z*i
\end{aligned}
$$

The above is repeated for every pixel in the original image to generate the filtered image

– Average all of the pixels in a neighbourhood around a central value

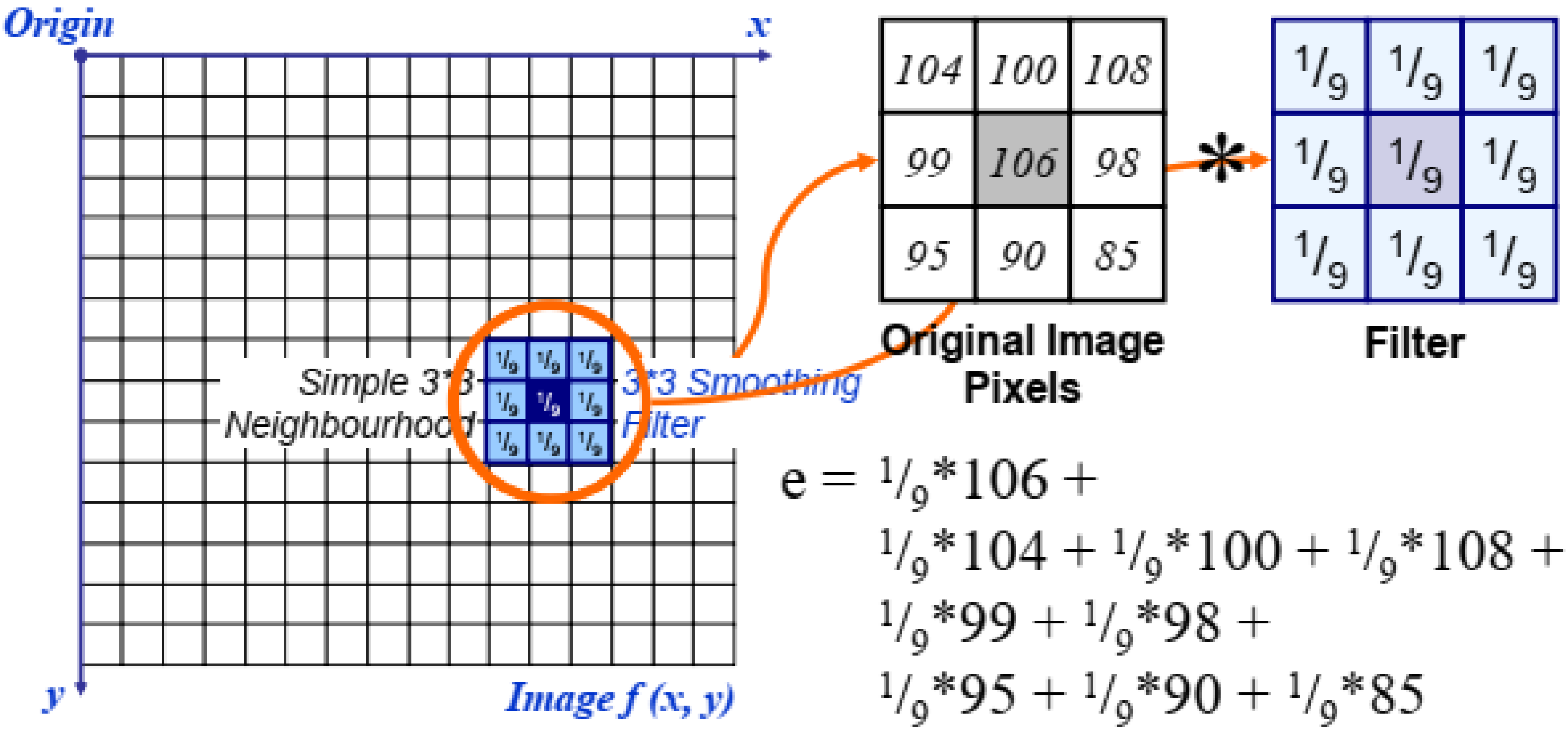
– Useful in removing noise from images

– Also useful for highlighting gross detail

– Edge blurring

$^1/_9$	$^1/_9$	$^1/_9$
$^1/_9$	$^1/_9$	$^1/_9$
$^1/_9$	$^1/_9$	$^1/_9$

Simple averaging filter



$$
\begin{aligned}
e = {} & {}^1/_9*106 + \\
& {}^1/_9*104 + {}^1/_9*100 + {}^1/_9*108 + \\
& {}^1/_9*99 + {}^1/_9*98 + \\
& {}^1/_9*95 + {}^1/_9*90 + {}^1/_9*85 \\
= {} & 98.3333
\end{aligned}
$$

The above is repeated for every pixel in the original image to generate the smoothed image.

More effective smoothing filters can be generated by allowing different pixels in the neighbourhood different weights in the averaging function

- Pixels closer to the central pixel are more important
- Often referred to as a *weighted averaging*

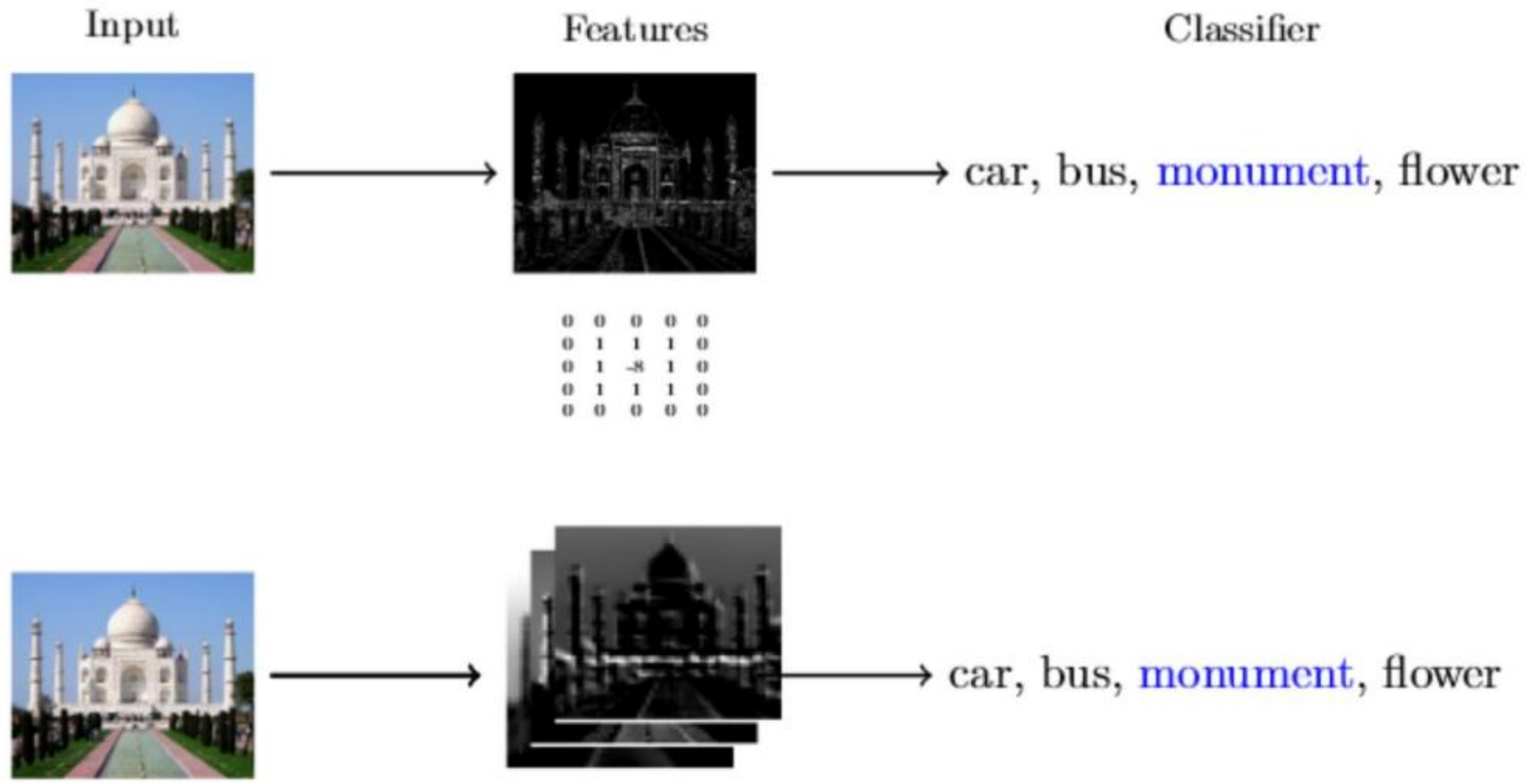
$^{1}/_{16}$	$^{2}/_{16}$	$^{1}/_{16}$
$^{2}/_{16}$	$^{4}/_{16}$	$^{2}/_{16}$
$^{1}/_{16}$	$^{2}/_{16}$	$^{1}/_{16}$

Weighted averaging filter

Edge Detection filters

0	1	2
−1	0	1
−2	−1	0

−2	−1	0
−1	0	1
0	1	2

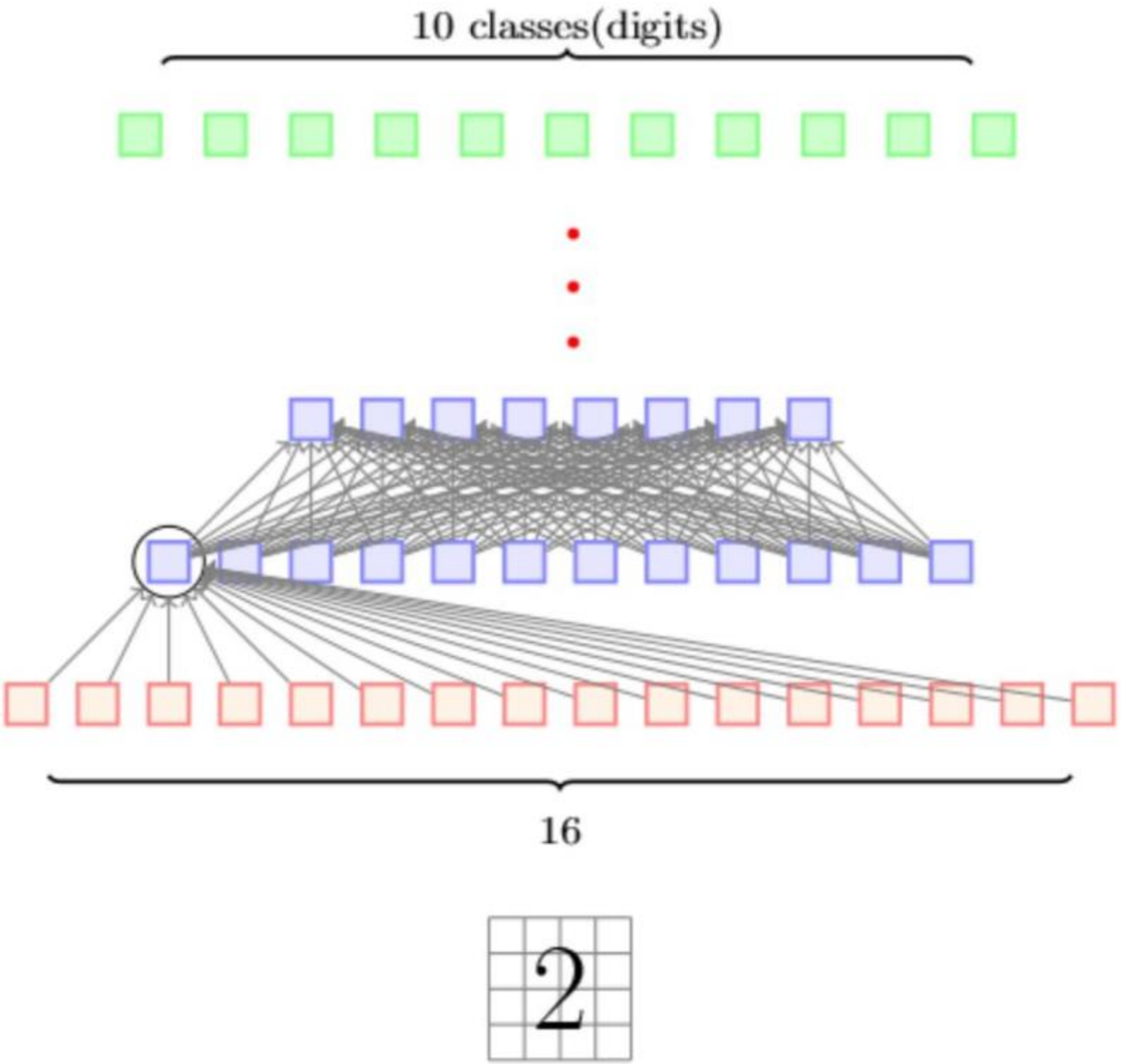


Instead of using handcrafted kernels such as edge detectors **can we learn meaningful kernels/filters in addition to learning the weights of the classifier?**

Convolution Neural Networks

Class of ANNs that are Shift/Space invariant

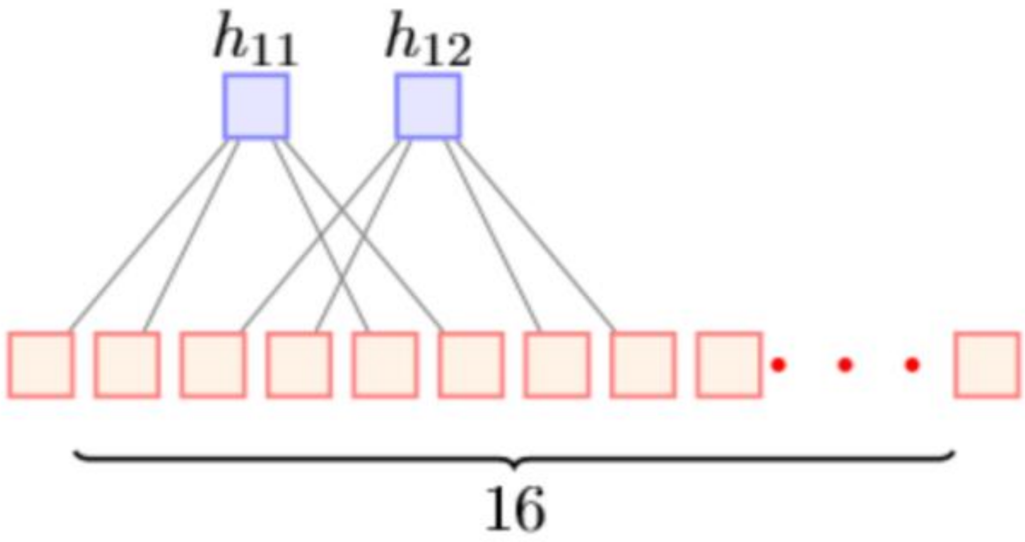
An MLP for processing an image

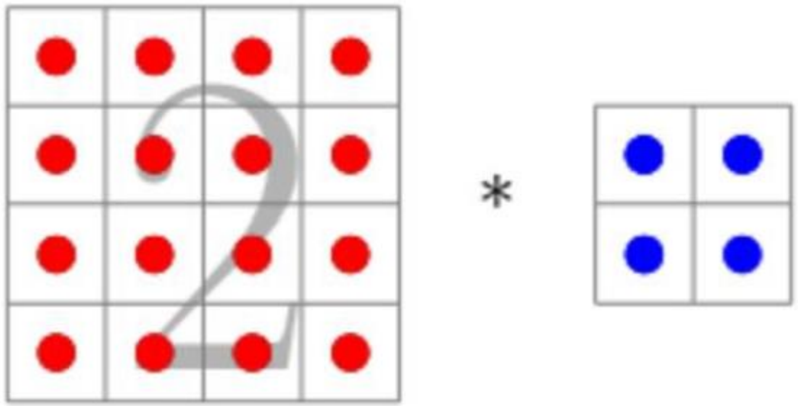


Why CNNs?

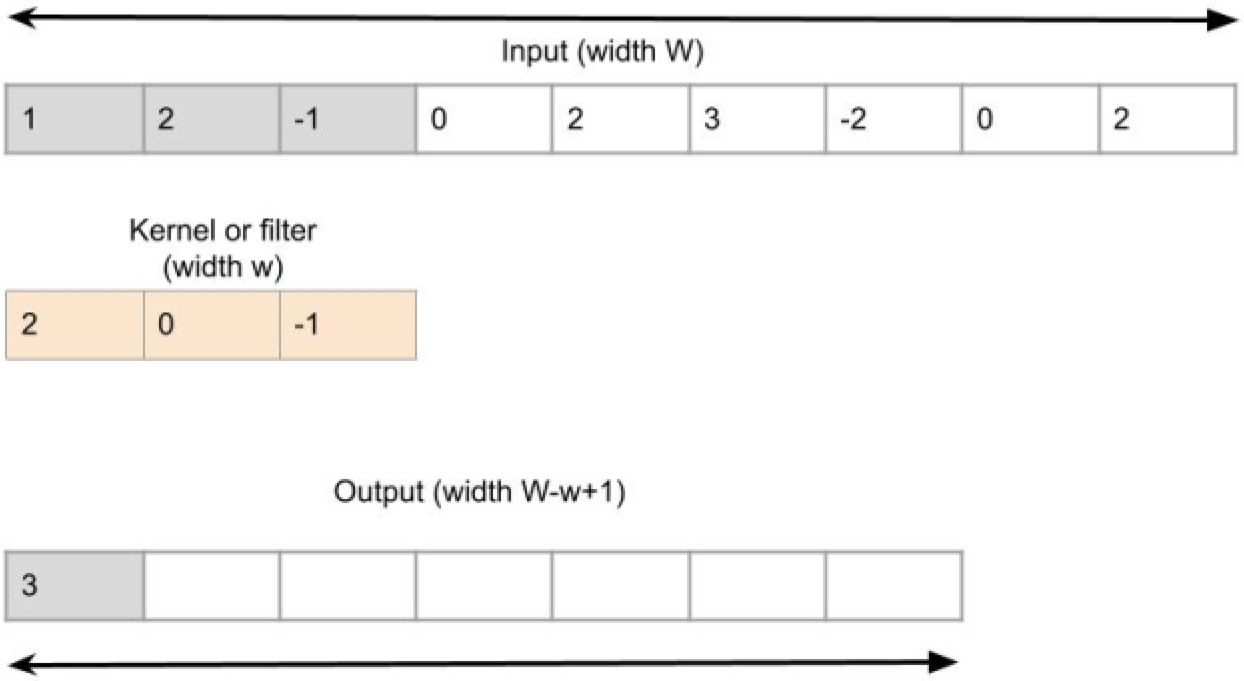
- Have invariance in translation
- Features may occur at different locations in the signal
- Convolution incorporates this idea: Applies same linear operation at all the locations and preserves the structure

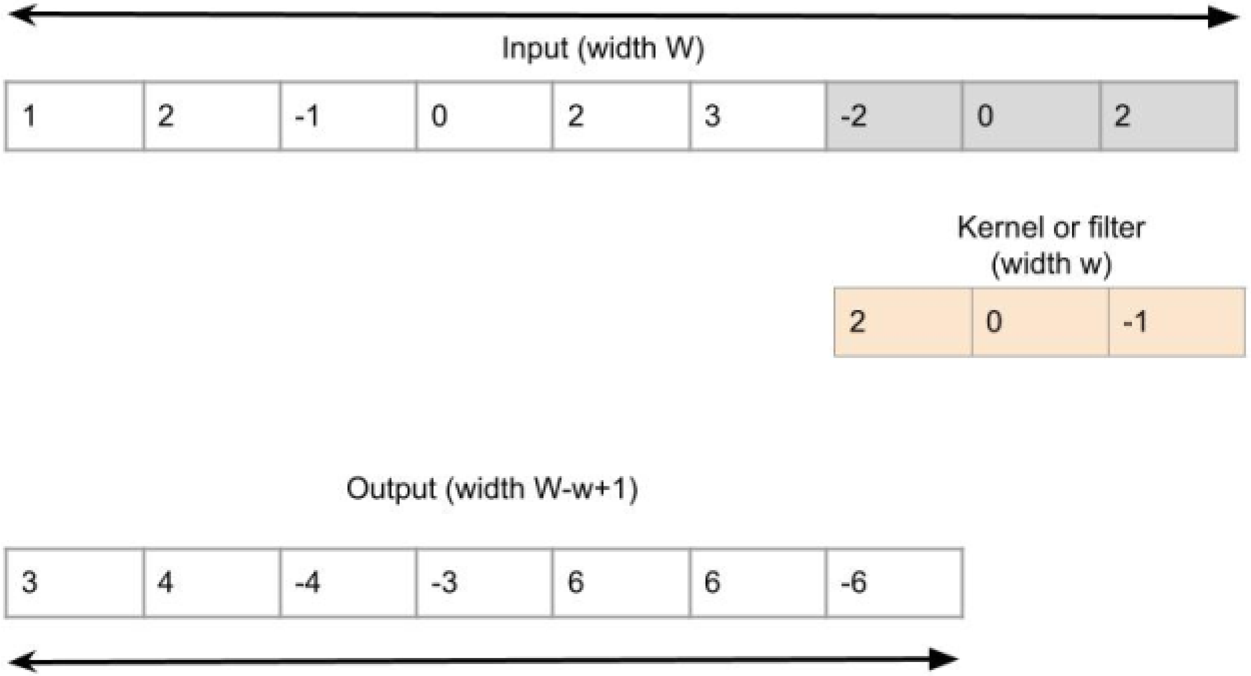
- We are taking advantage of the structure of the image(interactions between neighboring pixels are more interesting)
- This **sparse connectivity** reduces the number of parameters in the model
- Another characteristic of CNNs is **weight sharing**





Review of convolution

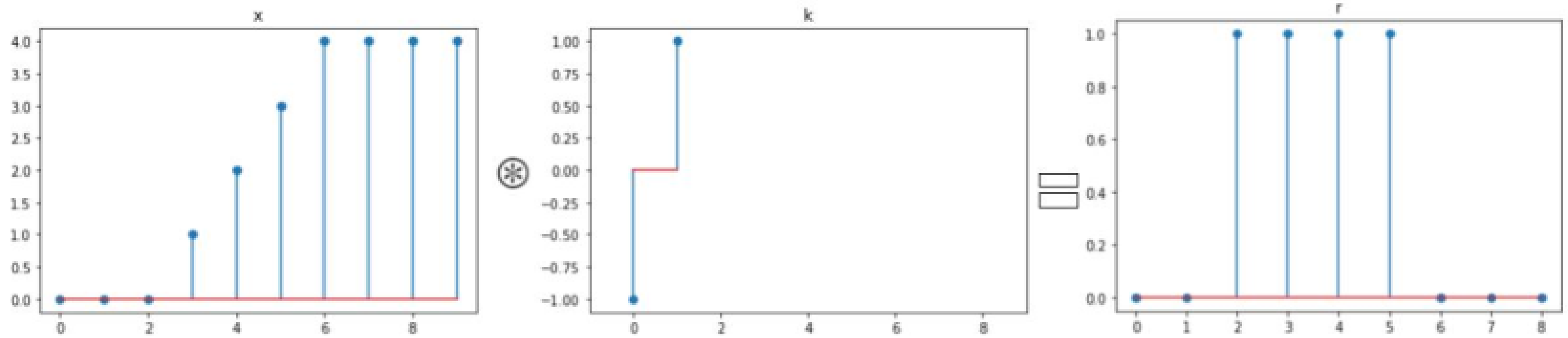


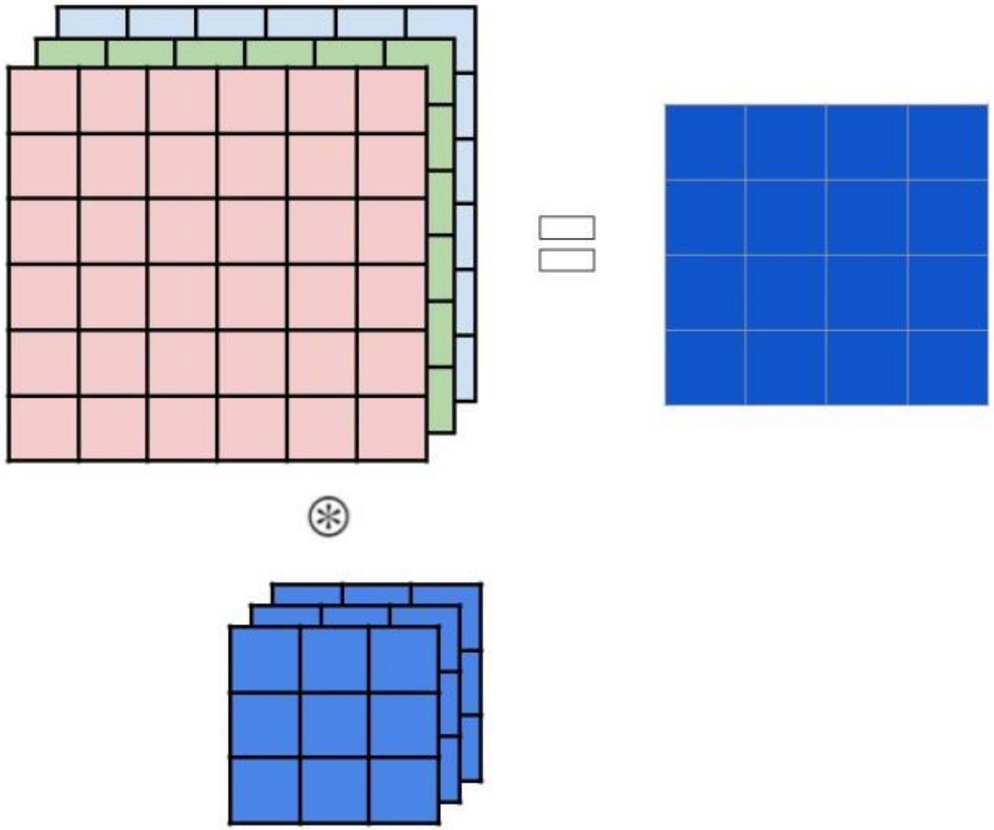

Input (width W)
1
2
-1
0
2
3
-2
0
2
Kernel or filter
(width w)
2
0
-1
Output (width W-w+1)
3
4
-4
-3
6
6
-6

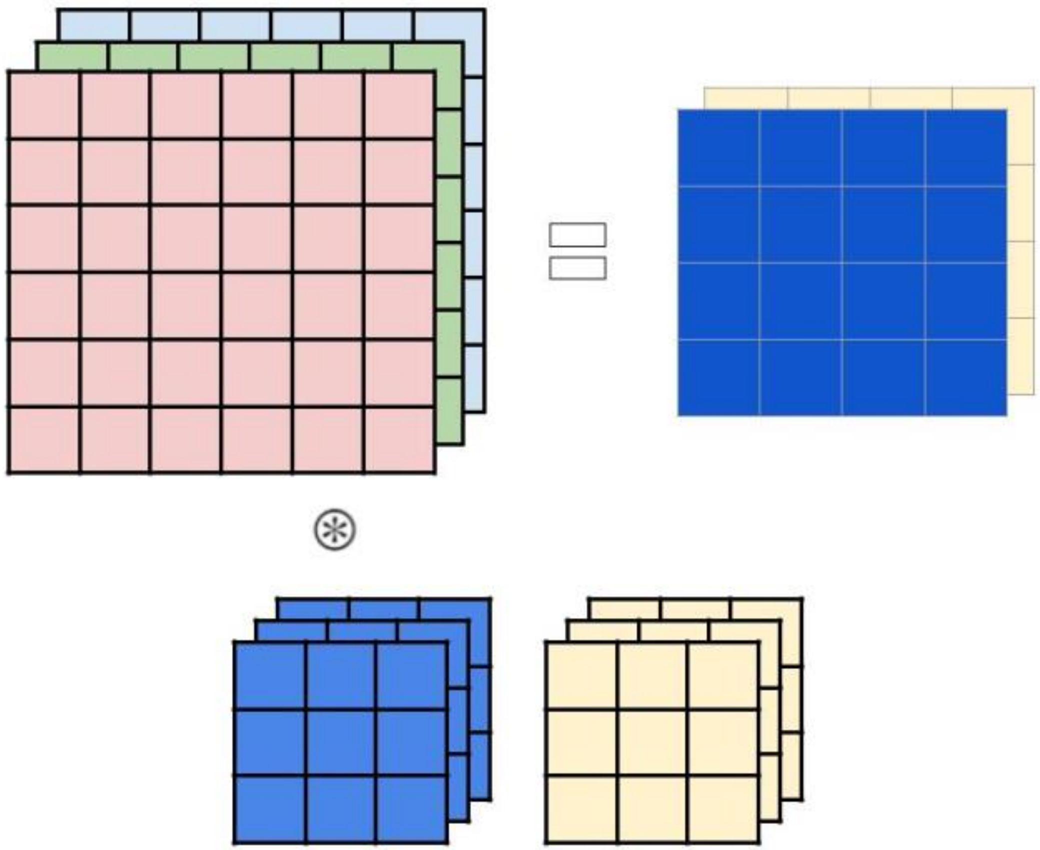
- Preserves the structure
  - if the i/p is a 2D tensor $\rightarrow$ o/p is also a 2D tensor
  - There exist a relation between the locations of i/p and o/p values

- Powerful feature extractor
- For instance, it can perform differential operation and look for interesting patterns in the input

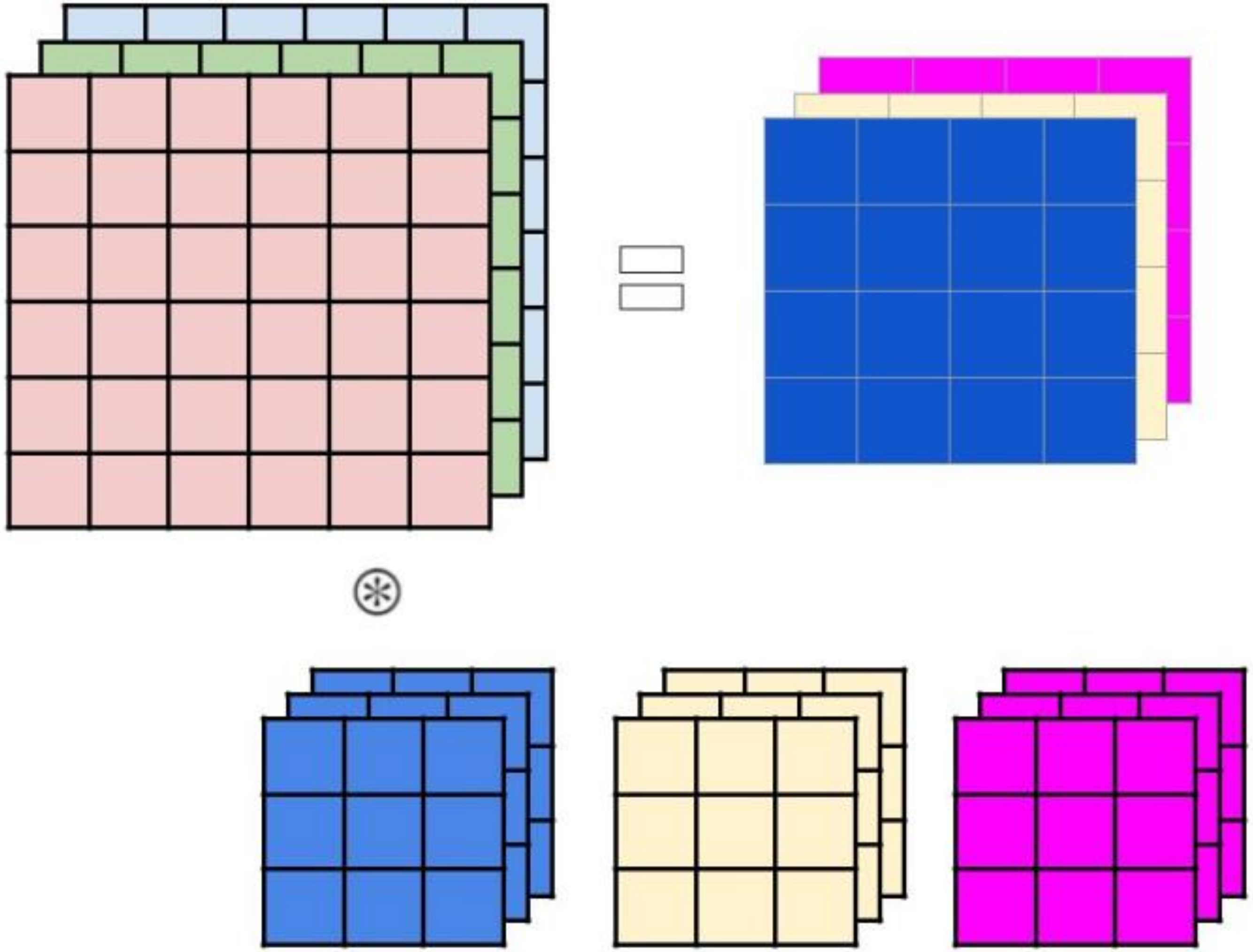
$$(0,0,0,1,2,3,4,4,4,4) \circledast (-1,1) = (0,0,1,1,1,1,0,0,0)$$

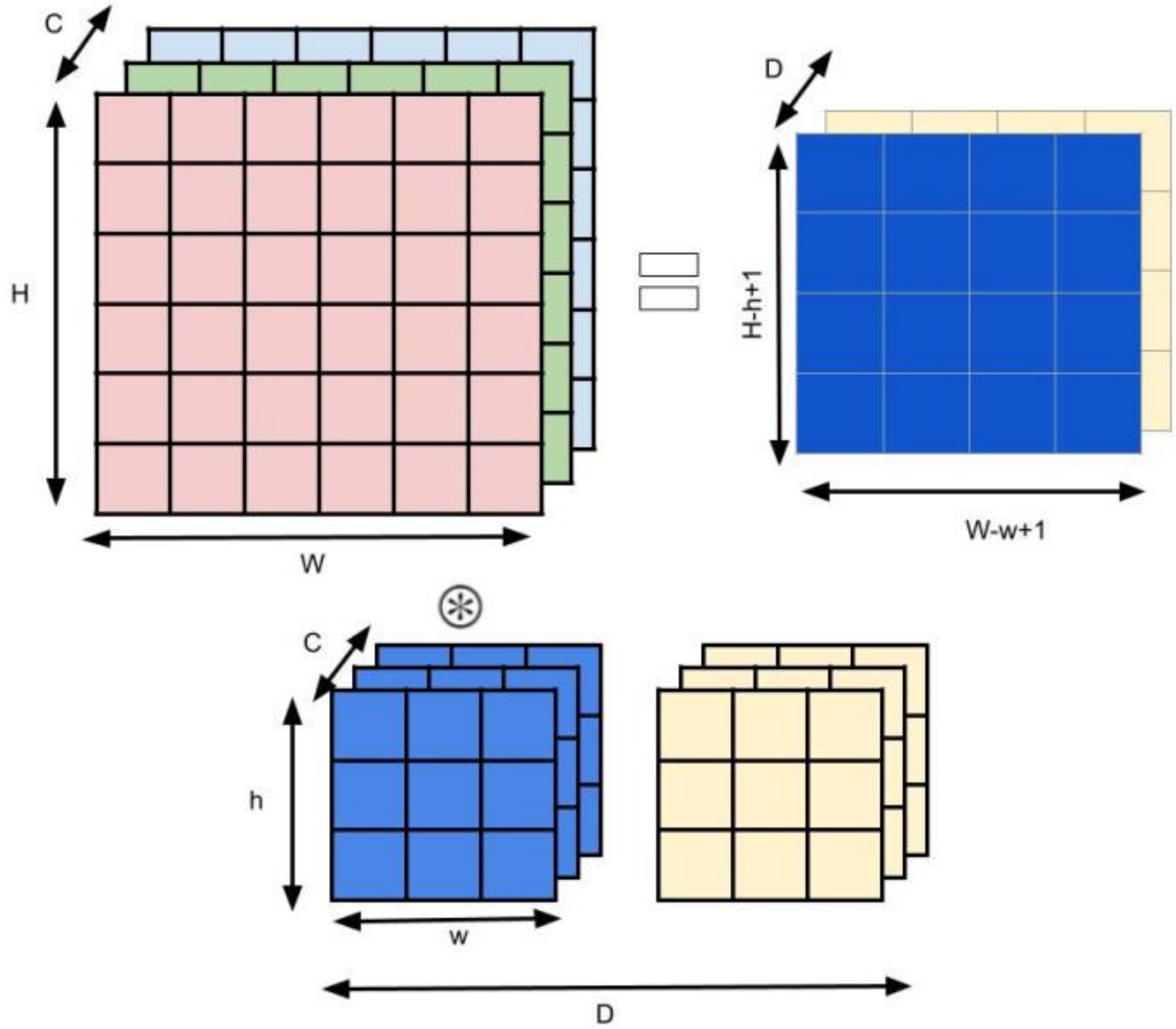




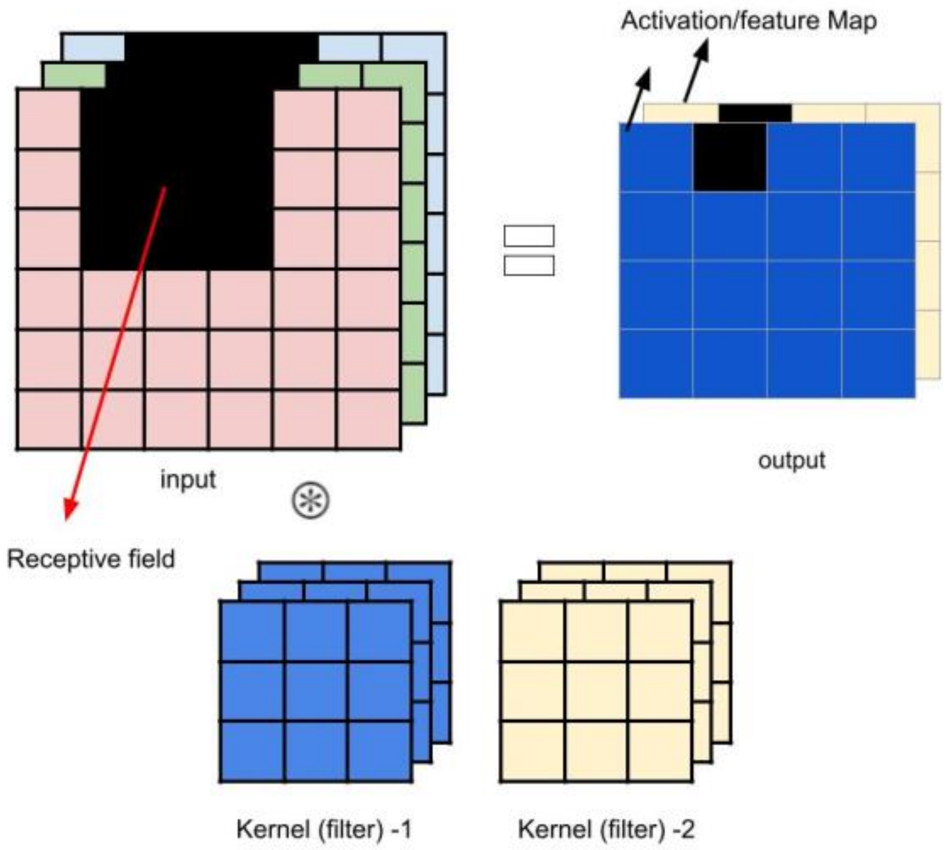


- CNNs process 3D tensors of size $C \times H \times W$ with kernels of size $C \times h \times w$ and result in 2D tensors of size $H - h + 1 \times W - w + 1$

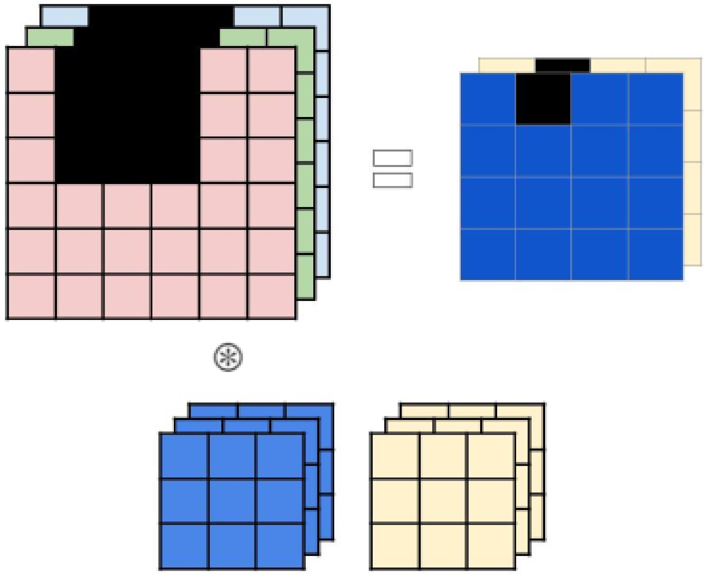




C
H
W
D
H-h+1
W-w+1
C
h
w
D


Activation/feature Map
input
output
Receptive field
Kernel (filter) -1
Kernel (filter) -2

Another way to interpret convolution is that an affine function is applied on an input block of size $C \times h \times w$



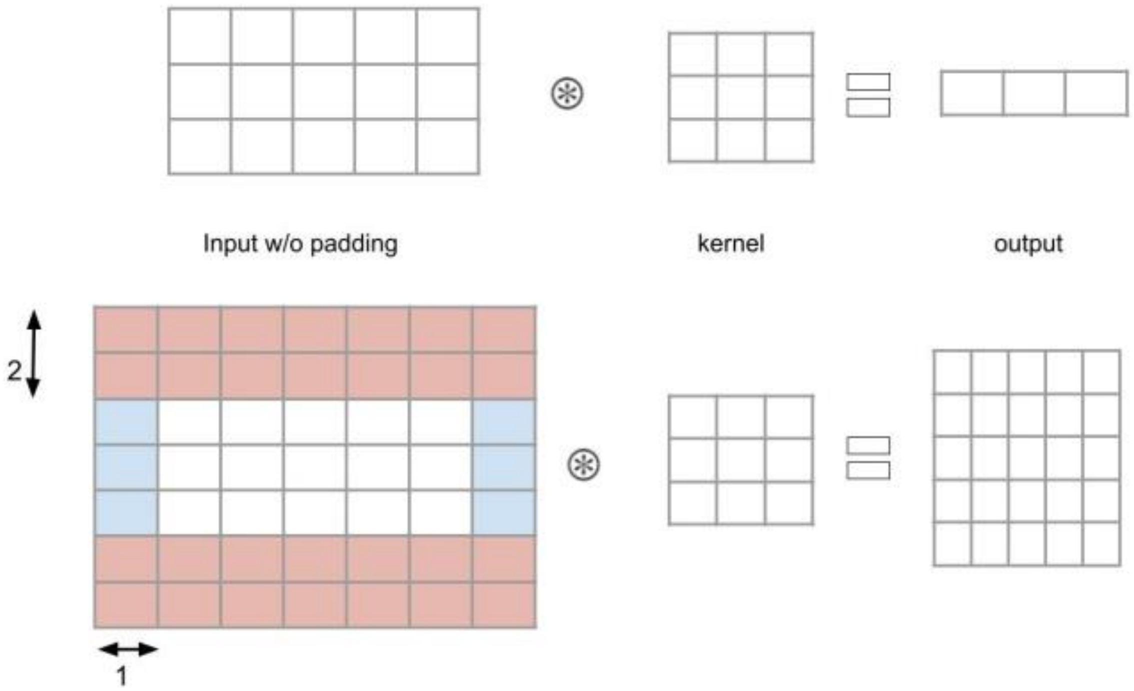
Same affine function is applied on all such blocks in the input

Convolution

- Preserves the input structure
  - 1D signal outputs 1D signal, 2D signal outputs 2D signal
  - Adjacent components in o/p are influenced by adjacent parts in the i/p

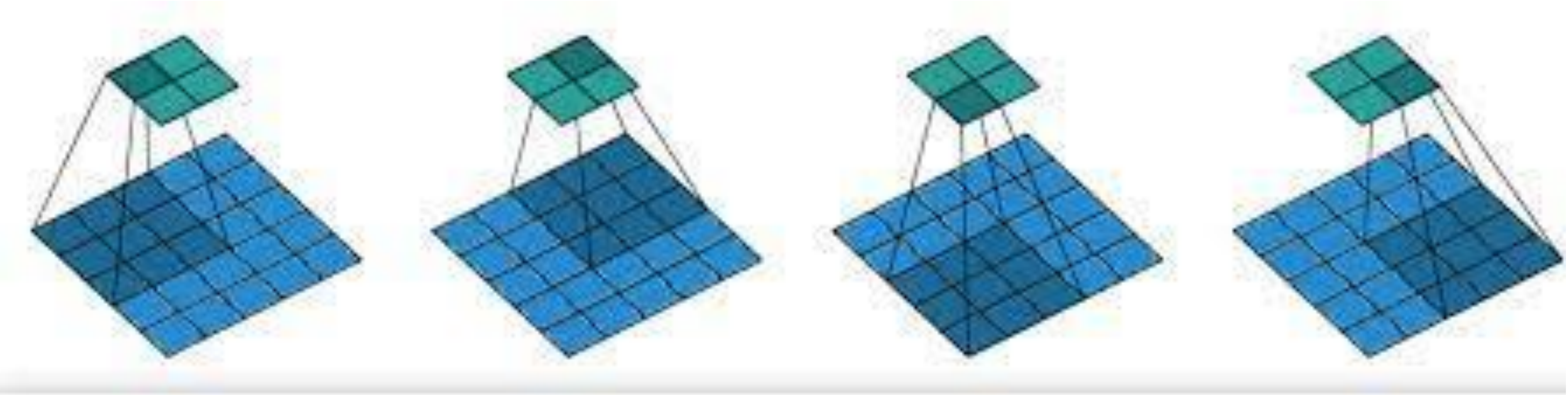
Padding in Convolution

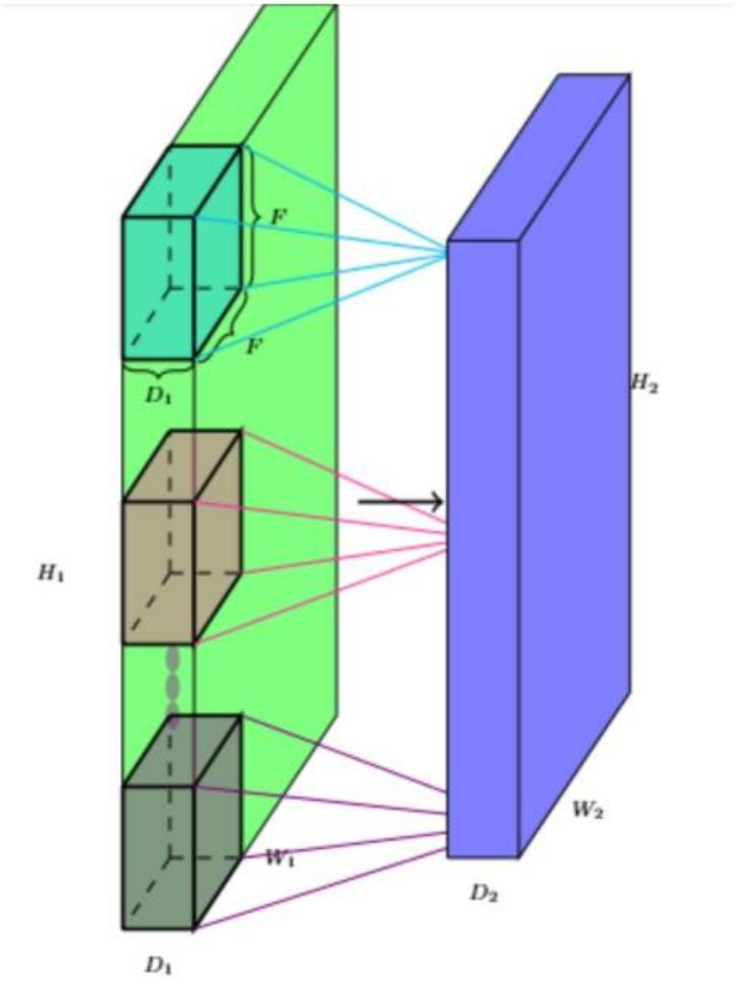
- Adds zeros around the input
- Takes cares of size reduction after convolution
- Instead of zeros, one may pad with signal values at the edges



Stride in Convolution

- Specifies the step size taken while performing convolution
- Default value is 1, i.e., move the kernel across the signal densely (without skipping)





Note the output size will be slightly less than the input.

$$\text{In general, } W_2 = W_1 - F + 1$$
$$H_2 = H_1 - F + 1$$

Incorporate padding of zeros to make input and output sizes same.

We now have,

$$W_2 = W_1 - F + 2P + 1$$
$$H_2 = H_1 - F + 2P + 1$$

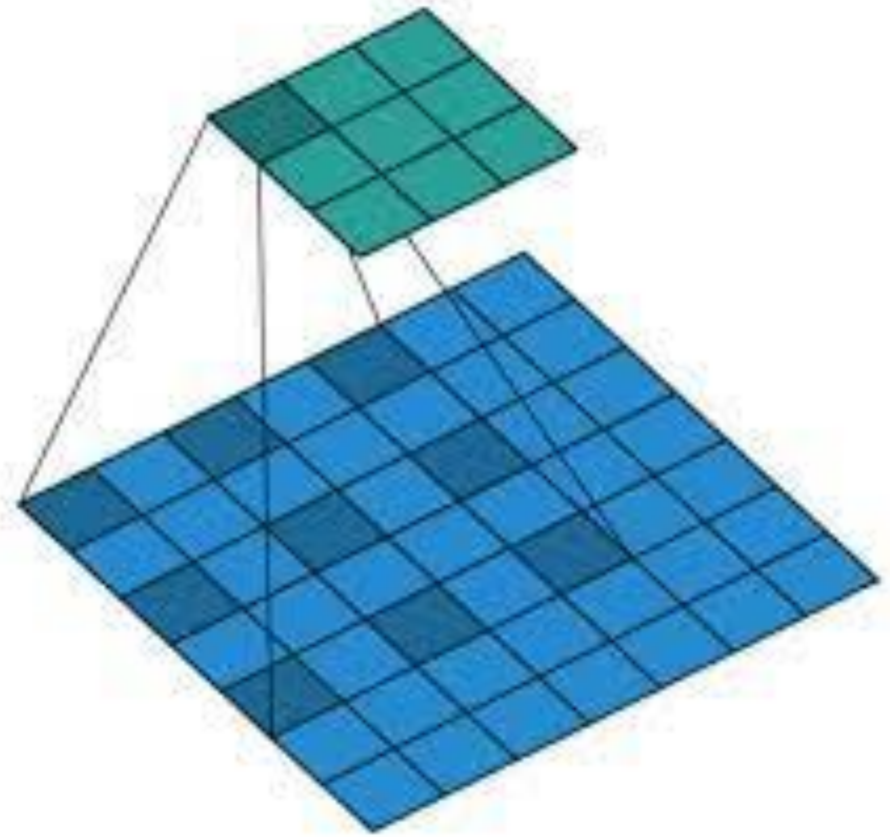
stride S

- It defines the intervals at which the filter is applied (here $S = 2$)
- Here, we are essentially skipping every 2nd pixel which will again result in an output which is of smaller dimensions

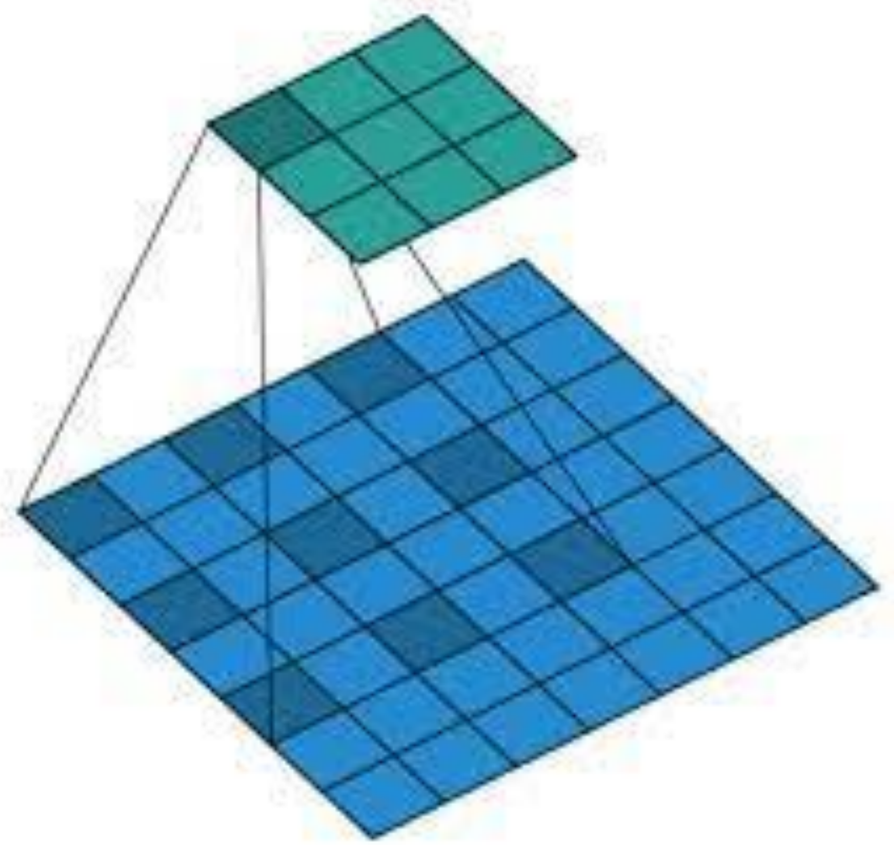
$$W_2 = \frac{W_1 - F + 2P}{S} + 1$$
$$H_2 = \frac{H_1 - F + 2P}{S} + 1$$

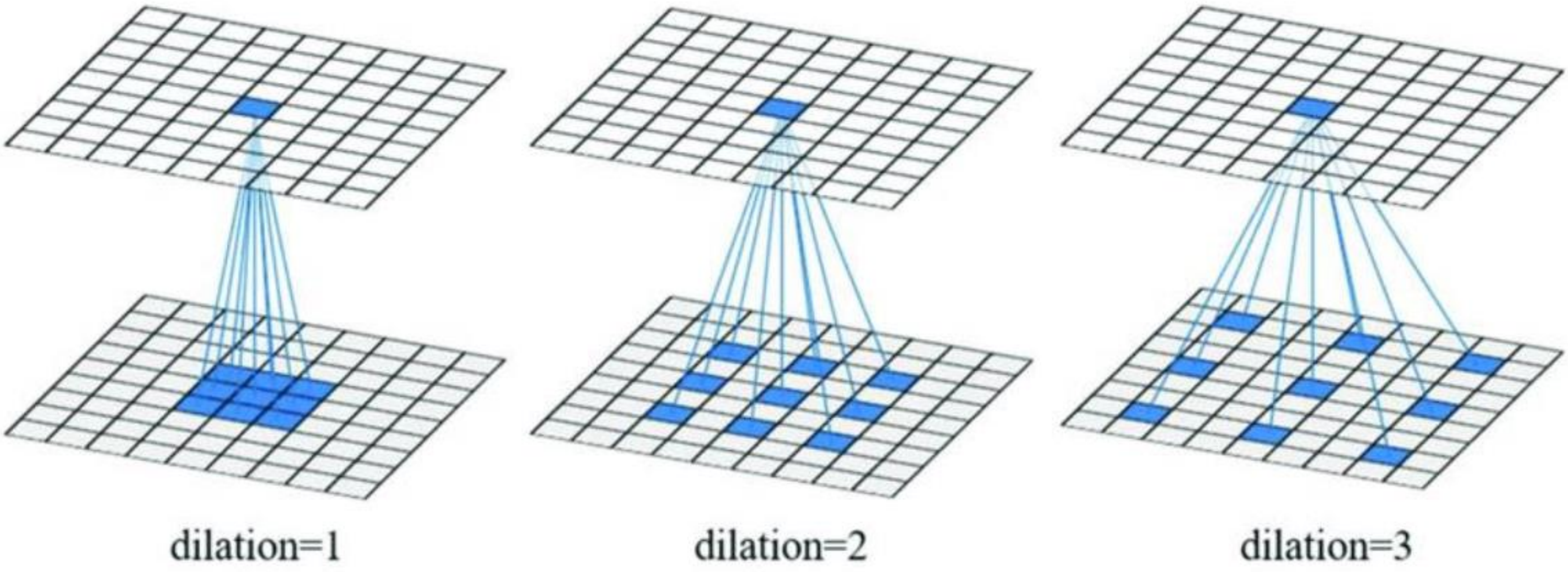
Dilation in Convolution

- Manipulates the size of the kernel via expanding its size without adding weights.
- In other words, it inserts 0s in between the kernel values

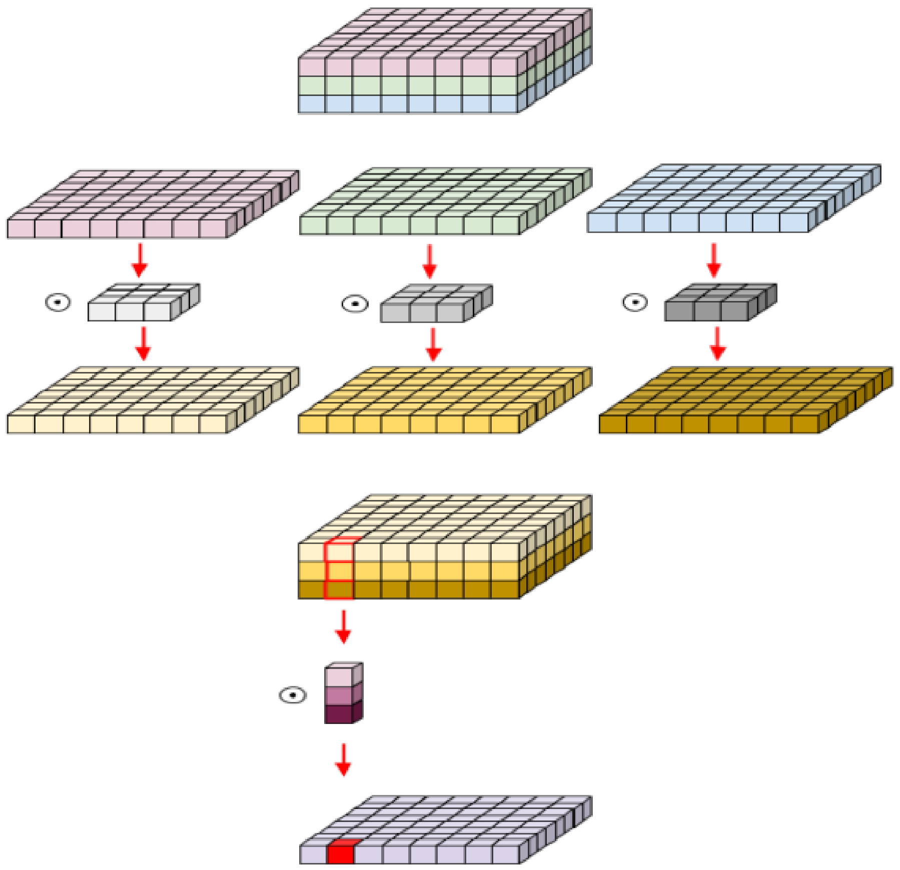


- Expands the kernel by adding rows and columns of zeros
- Default value for dilation is 1, i.e., no zeros placed
- Any higher value of dilation makes the kernel sparse
- Dilation increases the receptive field
- It is referred to as ‘atrous’ convolution

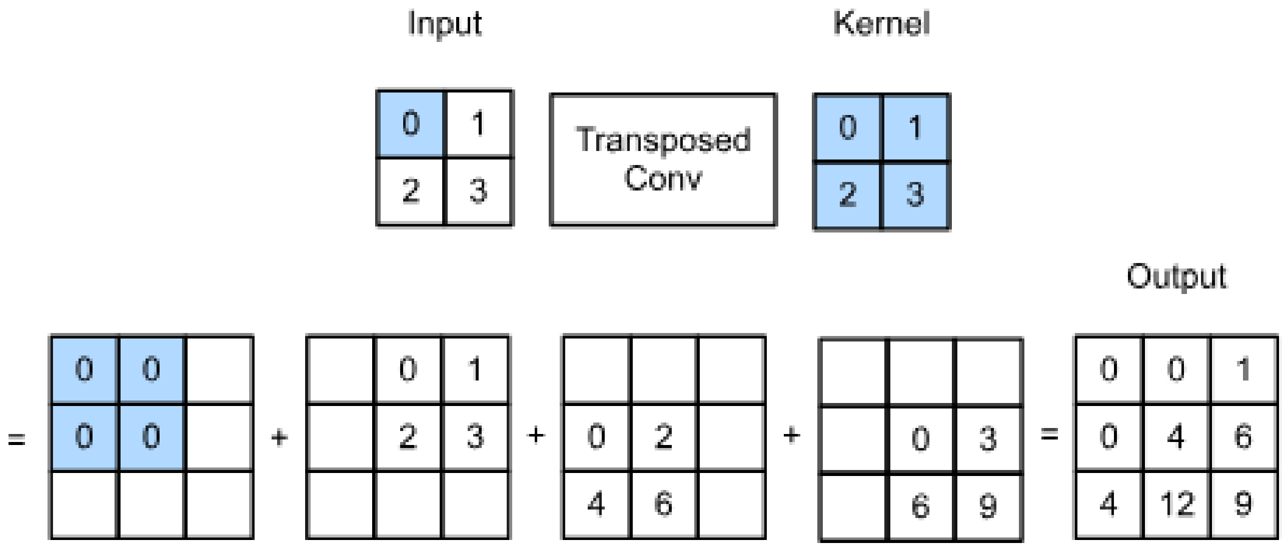



dilation=1
dilation=2
dilation=3

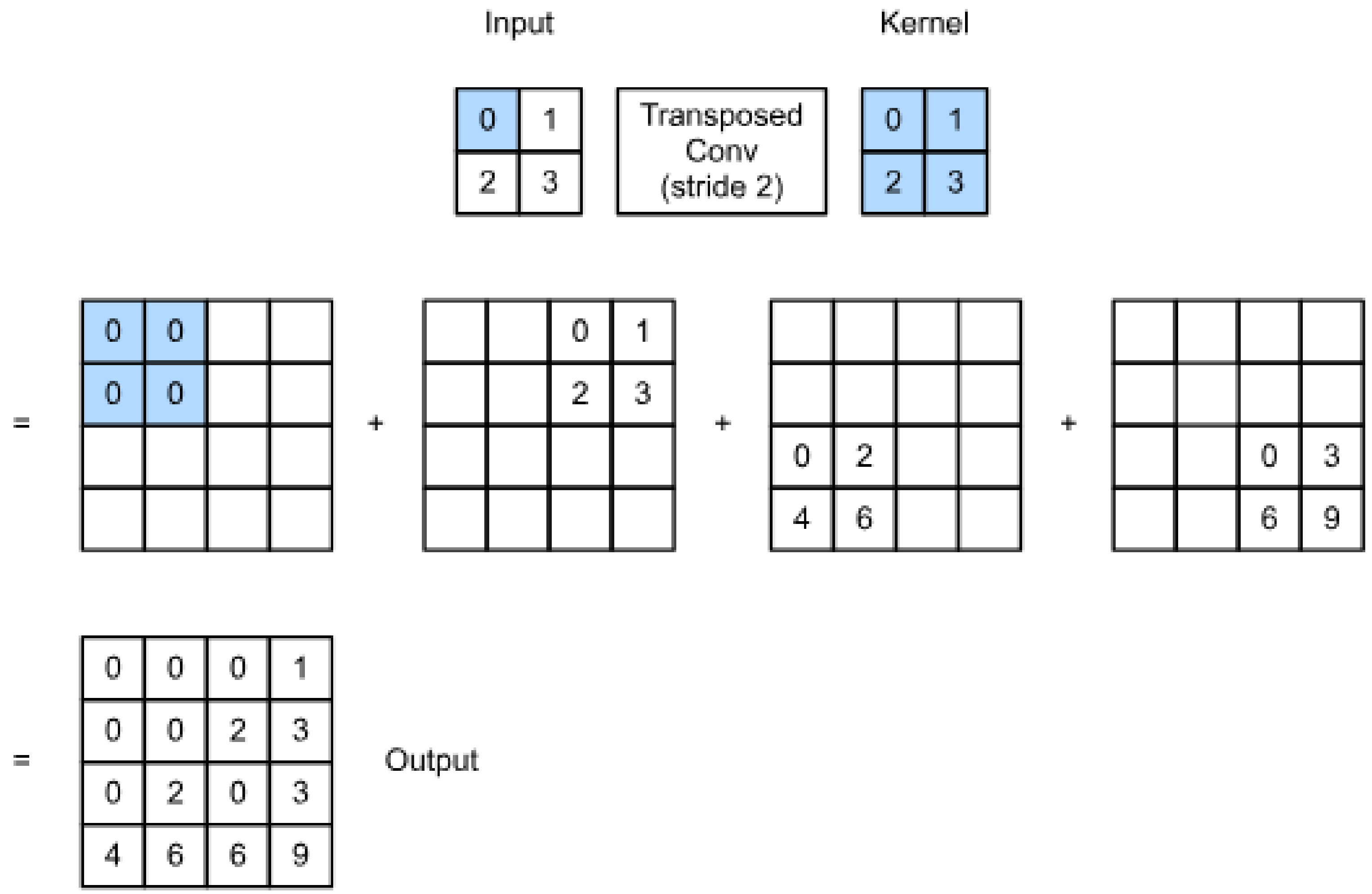
Depthwise separable Convolution



Transposed Convolution

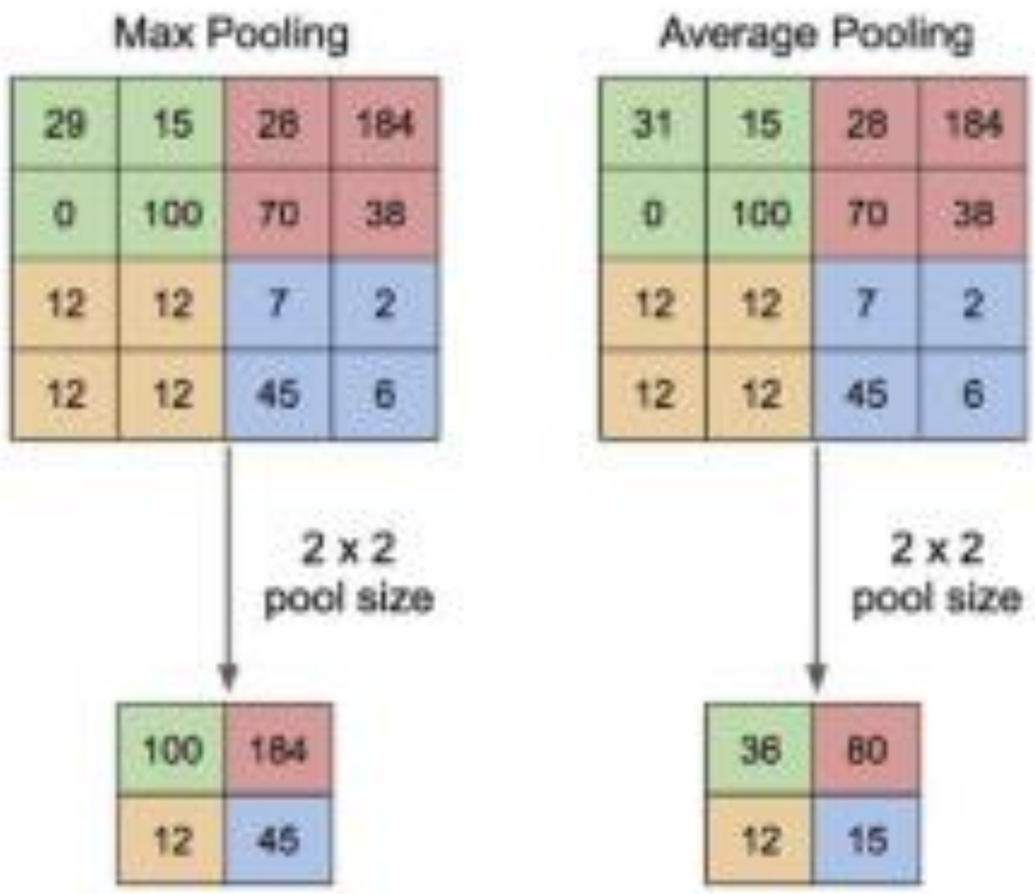


Transposed Convolution

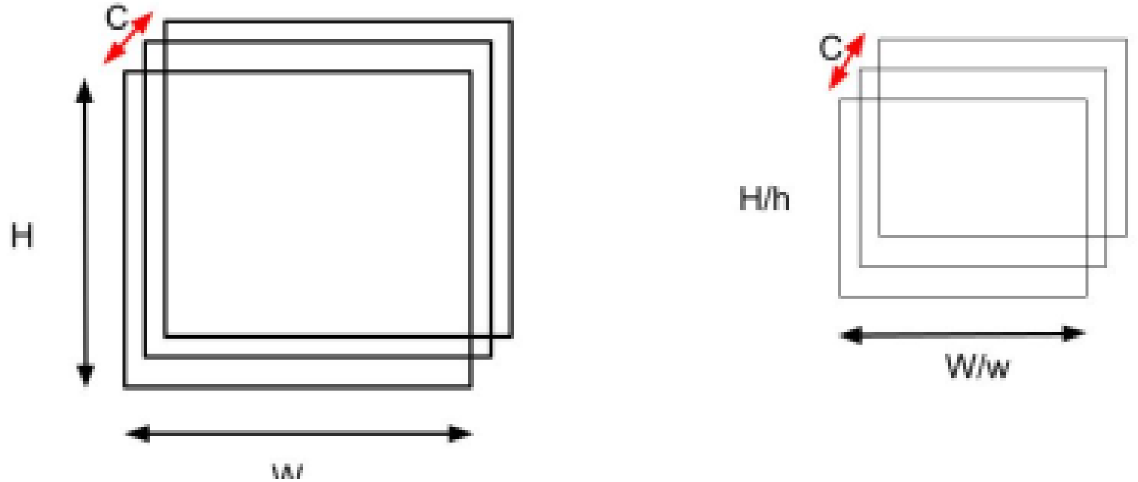


Pooling

- Groups multiple activations and replaces by a representative one
- Reduces the dimensionality of the signal progressively $\rightarrow$ considers non-overlapping stride
- Also called sub-sampling layer
- Generally found between two convolution layers (and parameter free)

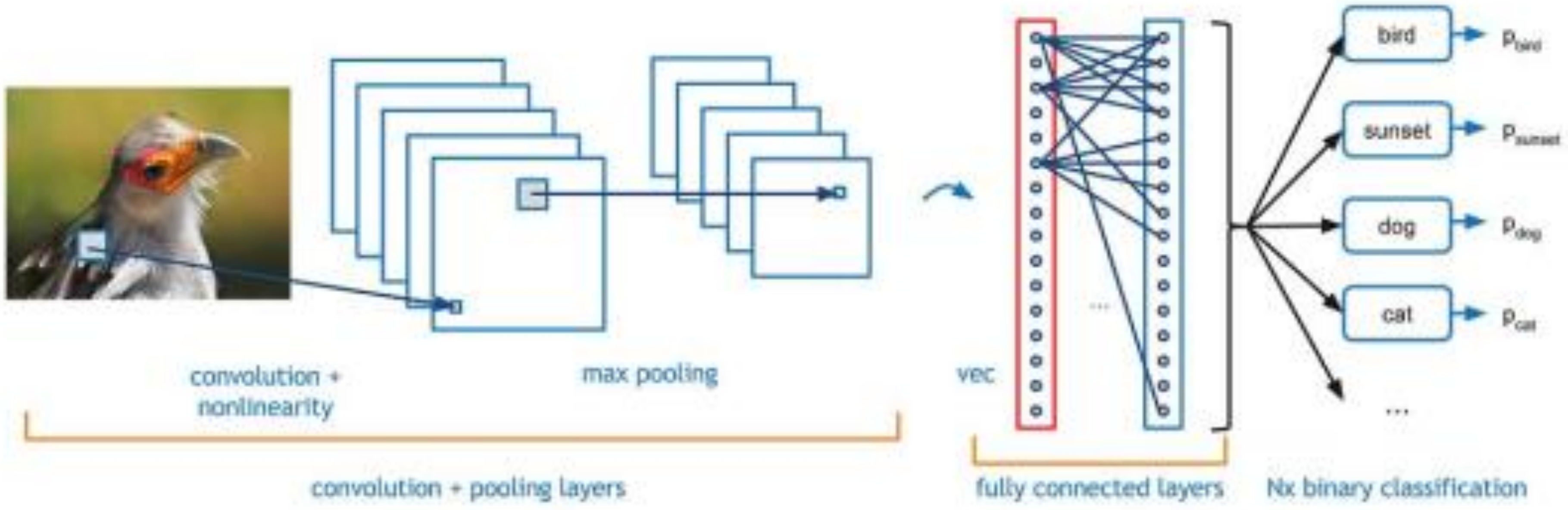

Max Pooling
29
15
28
184
0
100
70
38
12
12
7
2
12
12
45
6
2 x 2
pool size
100
184
12
45
Average Pooling
31
15
28
184
0
100
70
38
12
12
7
2
12
12
45
6
2 x 2
pool size
36
80
12
15

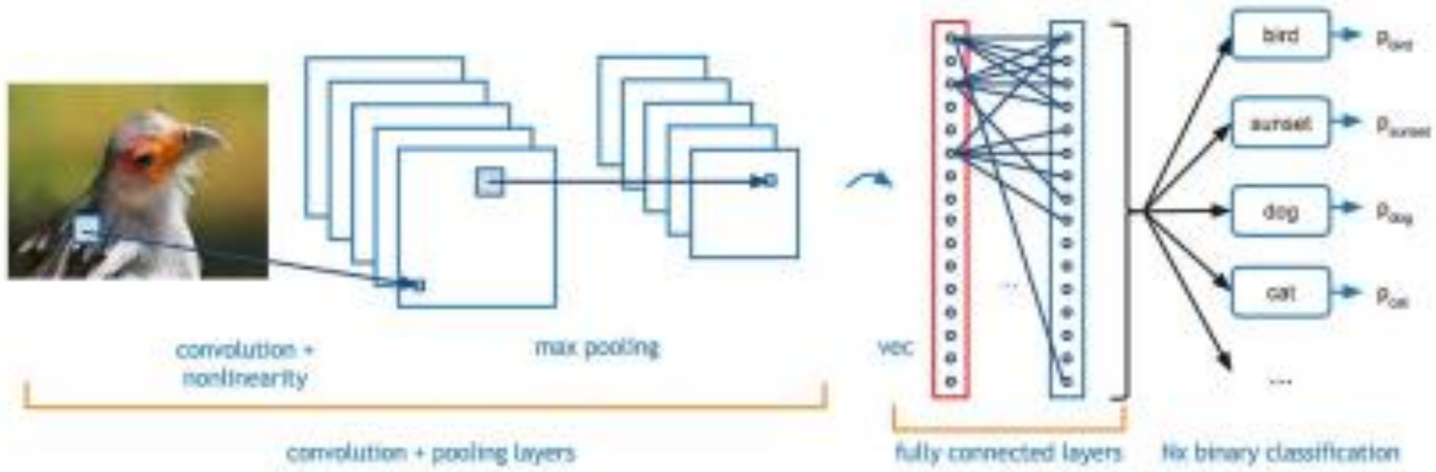
- No reduction in number of channels, only spatial size reduction



- Operation is invariant to any permutation within the block
- Withstands deformations caused by local translations

Simple CNN Architecture





- Initially Conv layer with nonlinearity
- Followed by a few Conv + Nonlinearity layers
- Have Pooling layers in between Conv layers $\rightarrow$ reduce the feature map size sufficiently
- Vectorize and and fully connected layers