FACULTY OF
COMPUTER SCIENCE & ARTIFICIAL INTELLIGENCE

AI & DS DEPARTMENT
Semester 5 /Level 2

AI 303:

Neural Networks
Fall 2023 – 2024

Dr. Sahar Ghanem

(slides adapted from- Dr. M. R. Hasan,

School of Computing, University of Nebraska-Lincoln)

Convolutional Neural Network

Padding

Deep Learning

What We Will Cover

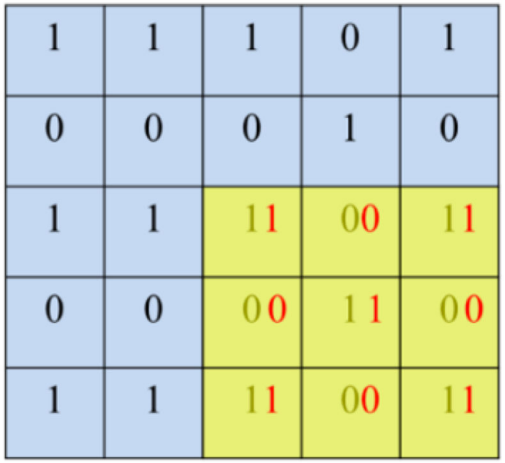
- Padding
- SAME vs VALID
- Activation

Convolution: Size of Feature Map

- Recall how the convolution operation **reduces the size** of an image (feature map).
- Let's perform convolution:

  - A single-channel input X (image or feature map) of height n_h and width n_w
  - A single-channel filter K of height f_h and width f_w
  - Stride s
- Size of the output (feature map) Z:
  - Z (height): $(n_h - f_h)/s+1$
  - Z (width): $(n_w - f_w)/s+1$

$$Z\ (\text{height}) = (5 - 3)/2 + 1 = 2$$
$$Z\ (\text{width}) = (5 - 3)/2 + 1 = 2$$



Convolution: Size of Feature Map

- Since convolutional filters generally have width and height greater than 1, after applying many successive convolutions, the CNN tends to wind up with outputs that are **considerably smaller than the input**.
- For example, say that we have a 240 × 240 pixel image.
- After performing 10 layers of 5 × 5 convolutions, it will shrink to 200 × 200 pixels.
- This **30% reduction** of the image will **lose any interesting information on the boundaries** of the original image.

Convolution: Size of Feature Map

- Can we keep the size of the input ***unchanged***?
- Yes, indeed.
- The most popular technique to keep the input dimension unchanged is **padding**.
- But this is possible only when we stride the filter across the image by one pixel, i.e., when **stride of length 1 is used**.

Convolution: Padding

- When **stride of length 1** is used, we can keep the size of the image (or feature map) unchanged by padding the image with **layers of zeros around its sides**.

0	0	0	0	0	0	0
0	1	1	1	0	1	0
0	0	0	0	1	0	0
0	1	1	1	0	1	0
0	0	0	0	1	0	0
0	1	1	1	0	1	0
0	0	0	0	0	0	0

Convolution: Padding

- We denote the number of layers for padding by "p".
- We add "p" rows from the top and "p" rows from the bottom.
- Similarly, we add "p" columns from the left, and "p" columns from the right.
- Thus, the formula for the **size of the output with padding** becomes.

  - Z (height): $(n_h + 2p - f_h)/s+1$
  - Z (width): $(n_w + 2p - f_w)/s+1$

0	0	0	0	0	0	0
0	1	1	1	0	1	0
0	0	0	0	1	0	0
0	1	1	1	0	1	0
0	0	0	0	1	0	0
0	1	1	1	0	1	0
0	0	0	0	0	0	0

Convolution: Padding

- Given s = 1, the **input dimension is maintained** at the output if:
  - Z (height): $n_h + 2p - f_h + 1 = n_h$
  - Z (width): $n_w + 2p - f_w + 1 = n_w$
- Thus, the formula for p:

$$p = (f_h - 1)/2 \text{ or } (f_w - 1)/2$$

- Z (height): $(n_h + 2p - f_h)/s+1$
- Z (width): $(n_w + 2p - f_w)/s+1$

0	0	0	0	0	0	0
0	1	1	1	0	1	0
0	0	0	0	1	0	0
0	1	1	1	0	1	0
0	0	0	0	1	0	0
0	1	1	1	0	1	0
0	0	0	0	0	0	0

Convolution: Padding

- In the following example, we need to add $p = (3-1)/2 = 1$ layer of zero padding around each side of the 2D input to keep the output dimension unchanged.
- It's called the "**Same**" padding.

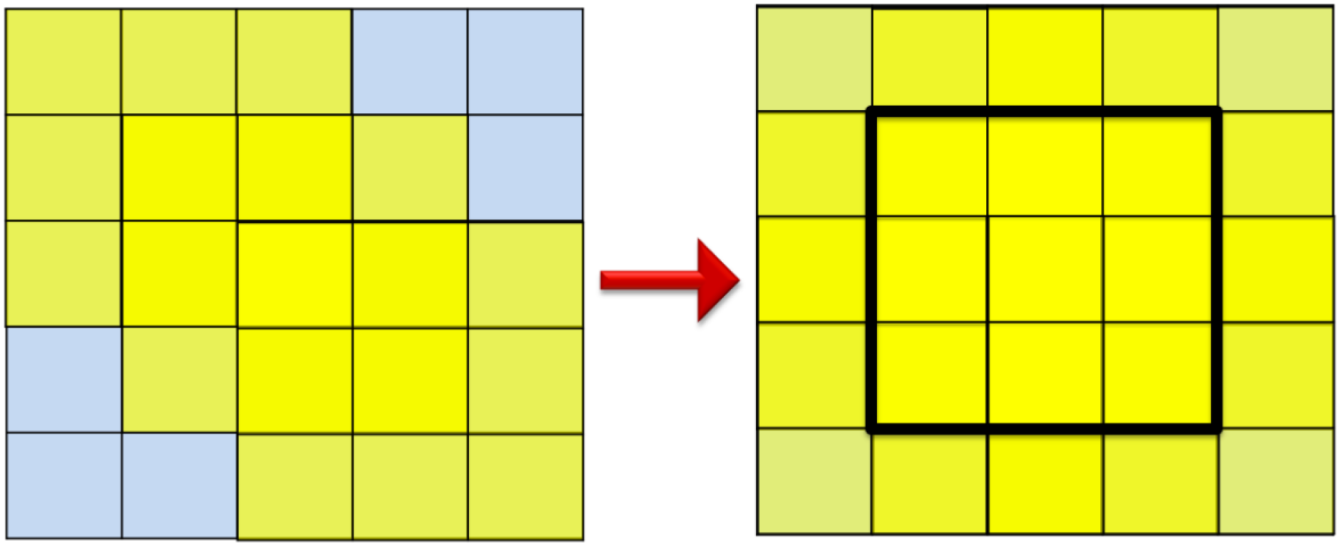
When no padding is used, we call it "**Valid**" padding.

0	0	0	0	0	0	0
0	1	1	1	0	1	0
0	0	0	0	1	0	0
0	1	1	1	0	1	0
0	0	0	0	1	0	0
0	1	1	1	0	1	0
0	0	0	0	0	0	0

$p = (f_h - 1)/2$ or $(f_w - 1)/2$

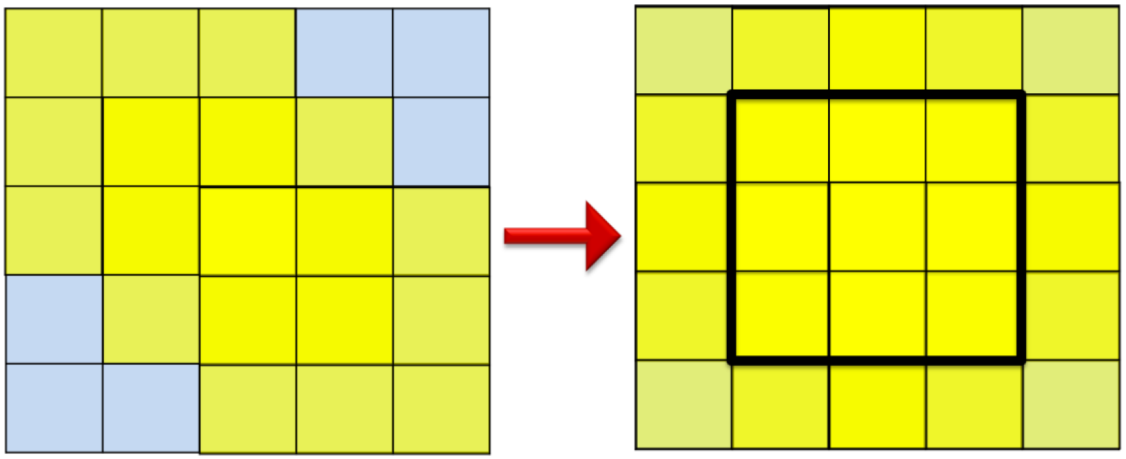
Convolution: Padding

- What is the **benefit** of the Same padding?
- With Valid padding, we tend to **lose pixels on the perimeter** of the image.
- Observe that the pixels at the **boundaries** (outside the bold squared region on the right figure) are **convolved less frequently** than the pixels at the middle.



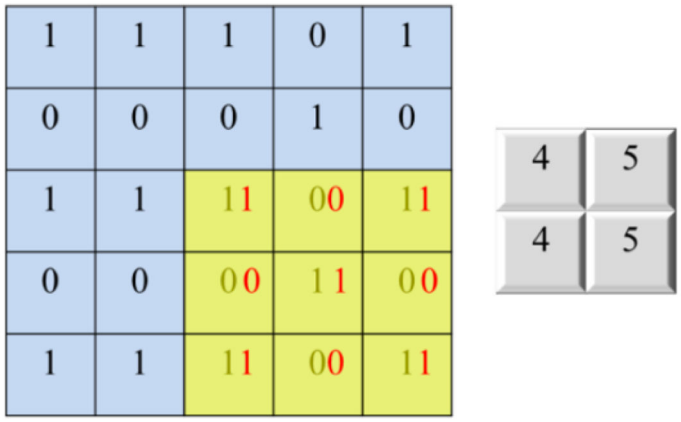
Convolution: Padding

- Since we typically use small kernels, for any given convolution, we might only lose a few pixels.
- But this **can add up** as we apply many successive convolutional layers.
- The Same padding resolves this problem by adding extra pixels of filler (zero values) around the boundary of the input image.
- Thus, it **increases the effective size of the image**.



Convolution: Padding

- What is the role of padding when **stride > 1** is used?
- When performing convolution, by default, we slide one input pixel at a time.
- However, sometimes, we need to we move the filter **more than one pixel at a time** (i.e., stride > 1), skipping the intermediate locations:
  - To achieve computational efficiency or
  - To downsample.



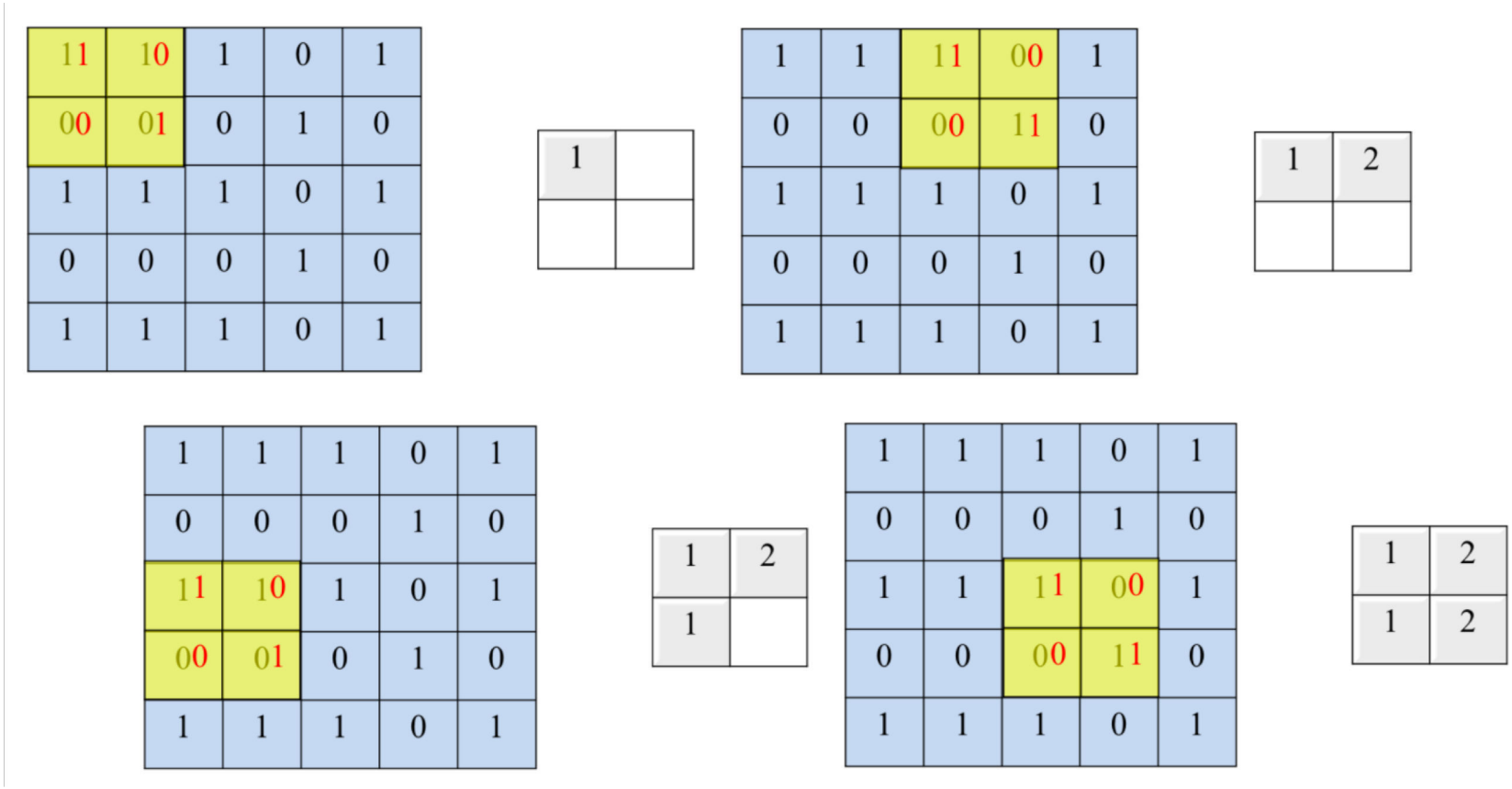
Convolution: Padding

- When stride > 1 is used, the "Same" padding **cannot keep the size** of the input unchanged.
- However, it can prevent the reduction being **too aggressive**.

Convolution: Padding

- Example: The dimension of the input is 5 x 5 (n x n).
- We apply a 2 x 2 (f x f) filter with stride = 2.
- Thus, the **side of the output** is:
  - $(n - f)/stride + 1 = (5 - 2)/2 + 1 = 2.5$
- Depending of the type of the padding, the fraction number (for the output sides) will be either **rounded down or up**.
  - VALID: the output size will be 2 x 2 (rounded down).
  - SAME: the output size will be 3 x 3 (rounded up).

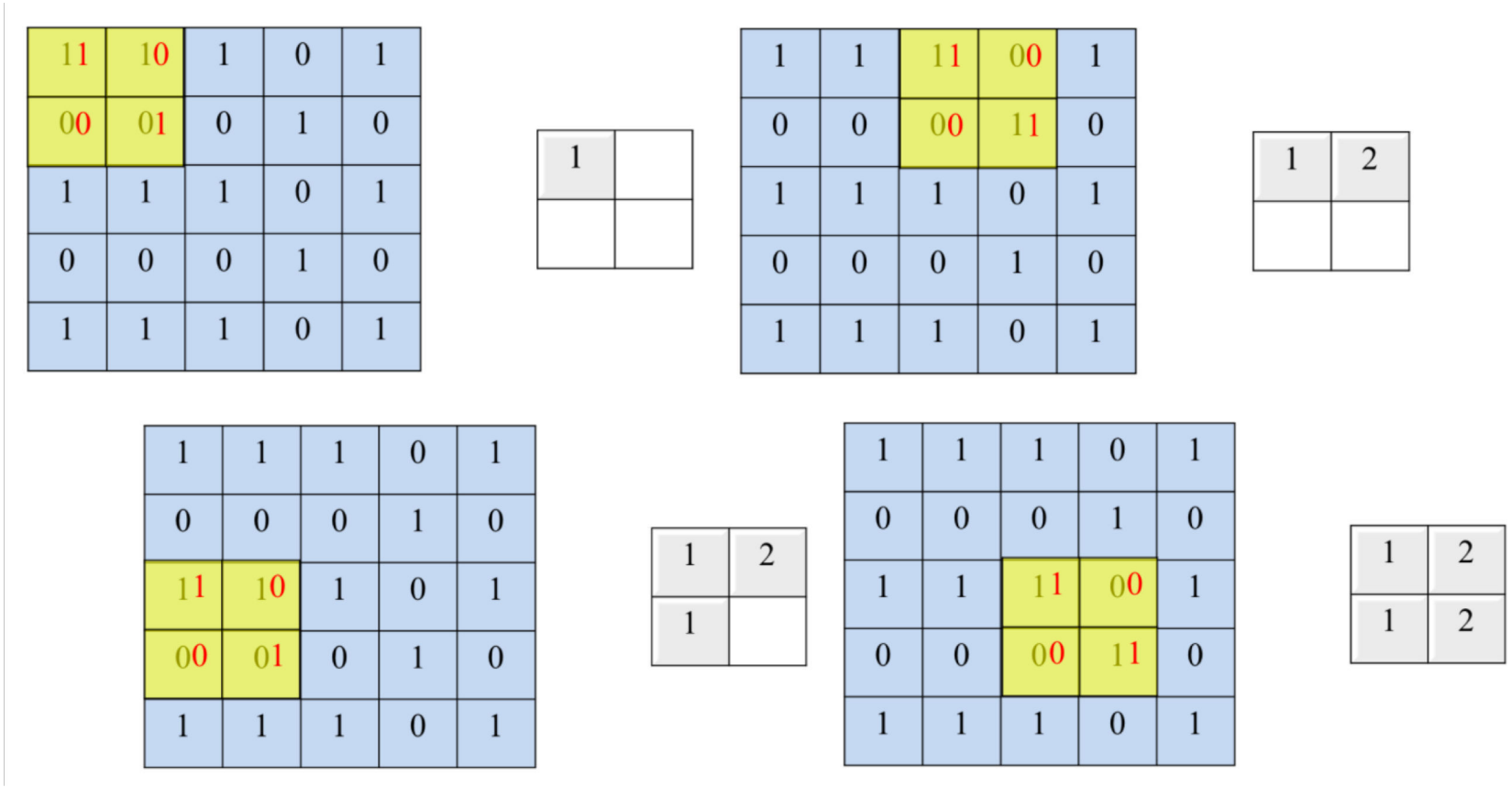
- VALID Padding: Stride > 1
- Some rows and columns at the bottom and right of the input image may be **ignored during convolution**.
- Below the rightmost column and the bottom row are never scanned by the filter.
- As a consequence, the output size becomes 2 x 2.



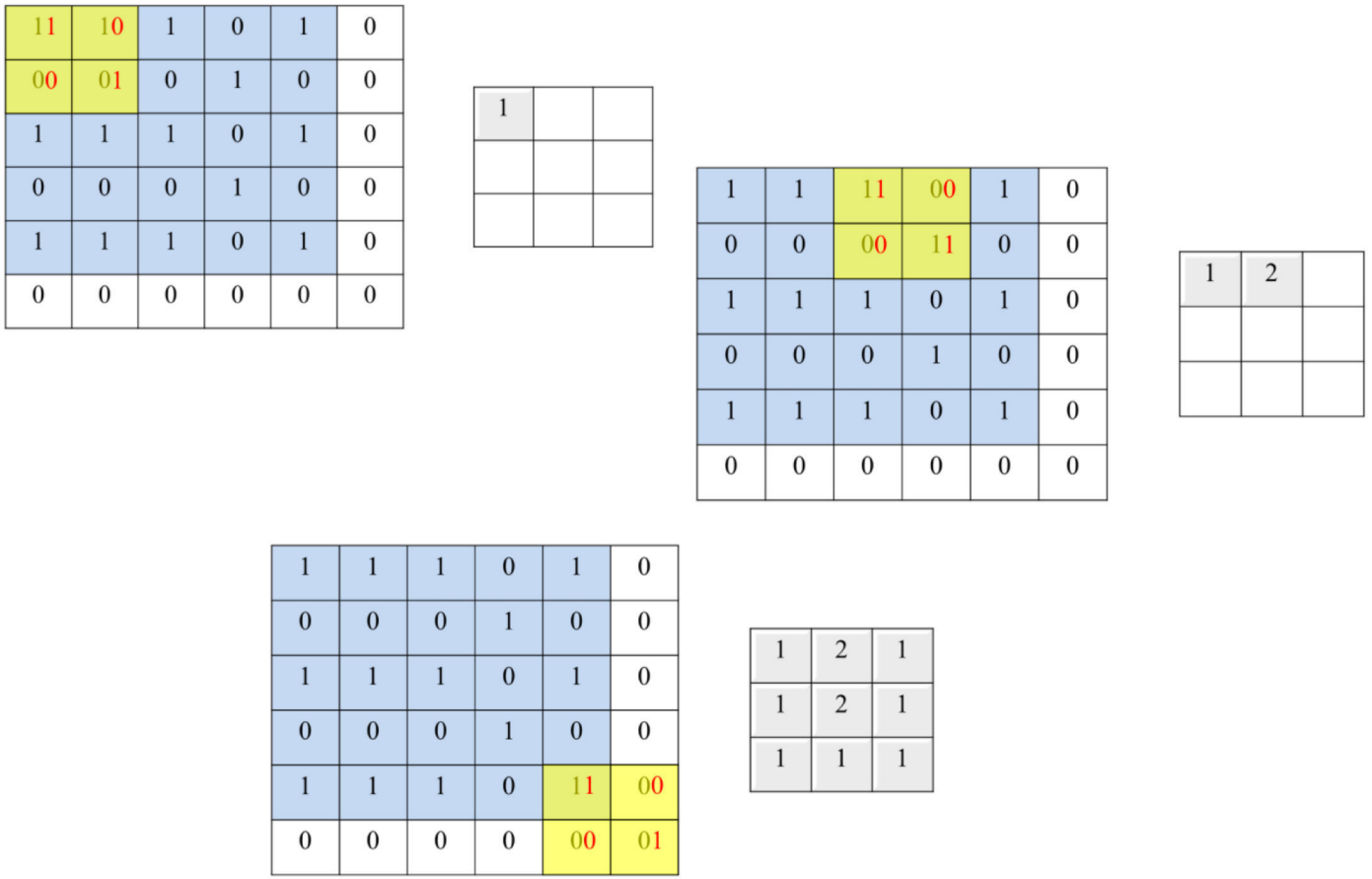
- VALID Padding: Stride > 1
- The VALID padding may result into two issues:

  - The dimension of the image may **reduce significantly** as we perform many successive convolutions in deeper networks.

  - It may throw away **information from the edges**.



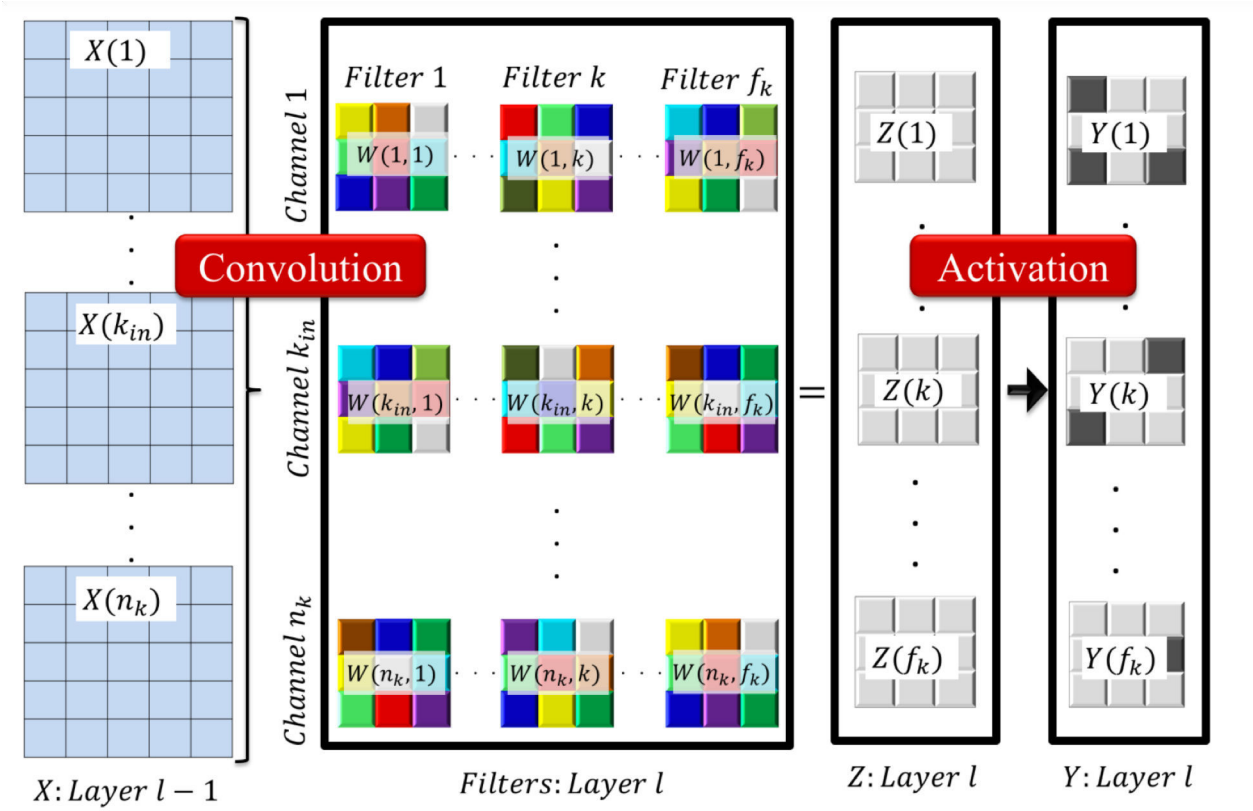
- SAME Padding: Stride > 1
- It rounds up the output size as needed and adds layers of zero padding to the input.
- Below a **column of zero padding layer** is added on the right and a row of zero padding layer is added at the bottom.



Activation Layer

- We have covered the **nuts and bolts** of the convolution operation.
- One last task we need to present that a Conv layer typically does in CNNs.

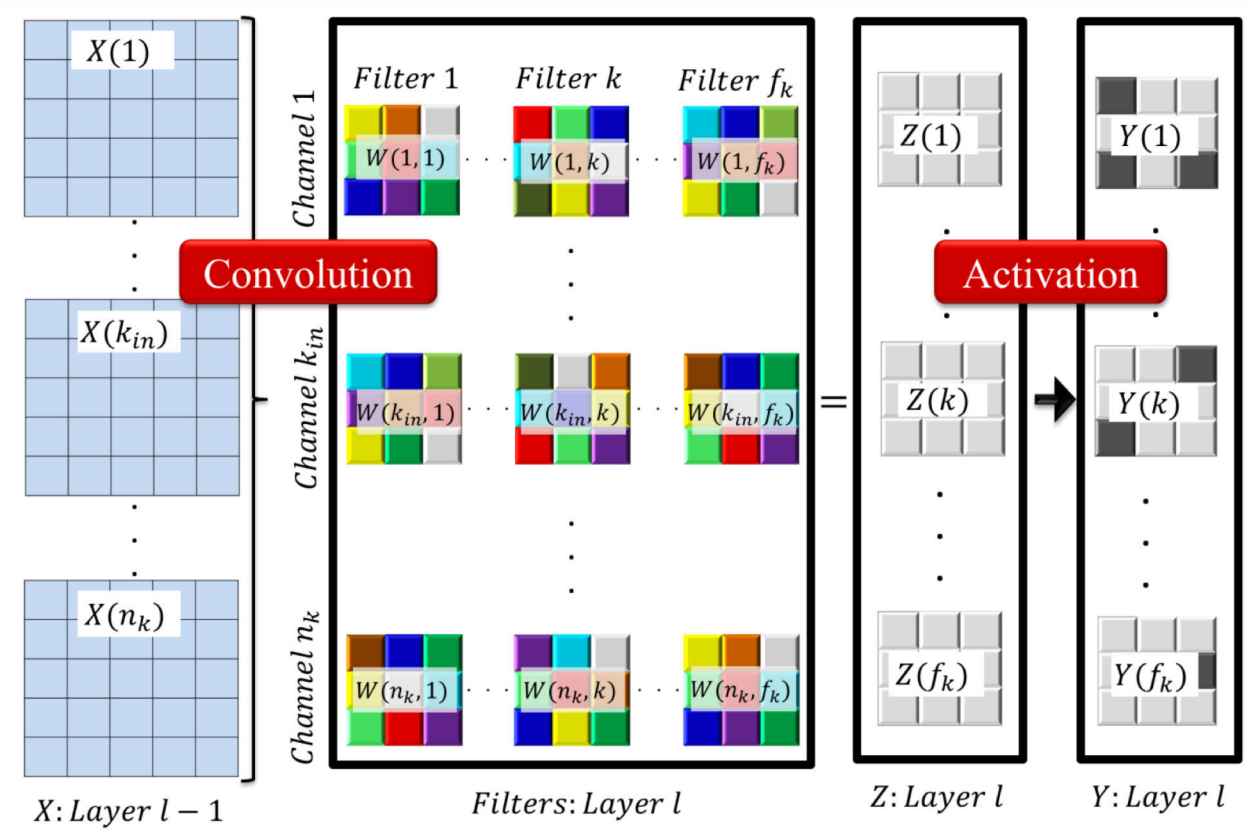
Once the feature maps are computed, those are **passed through a nonlinear activation** function $f()$ to produce an activated feature map (denoted by Y).



Activation Layer

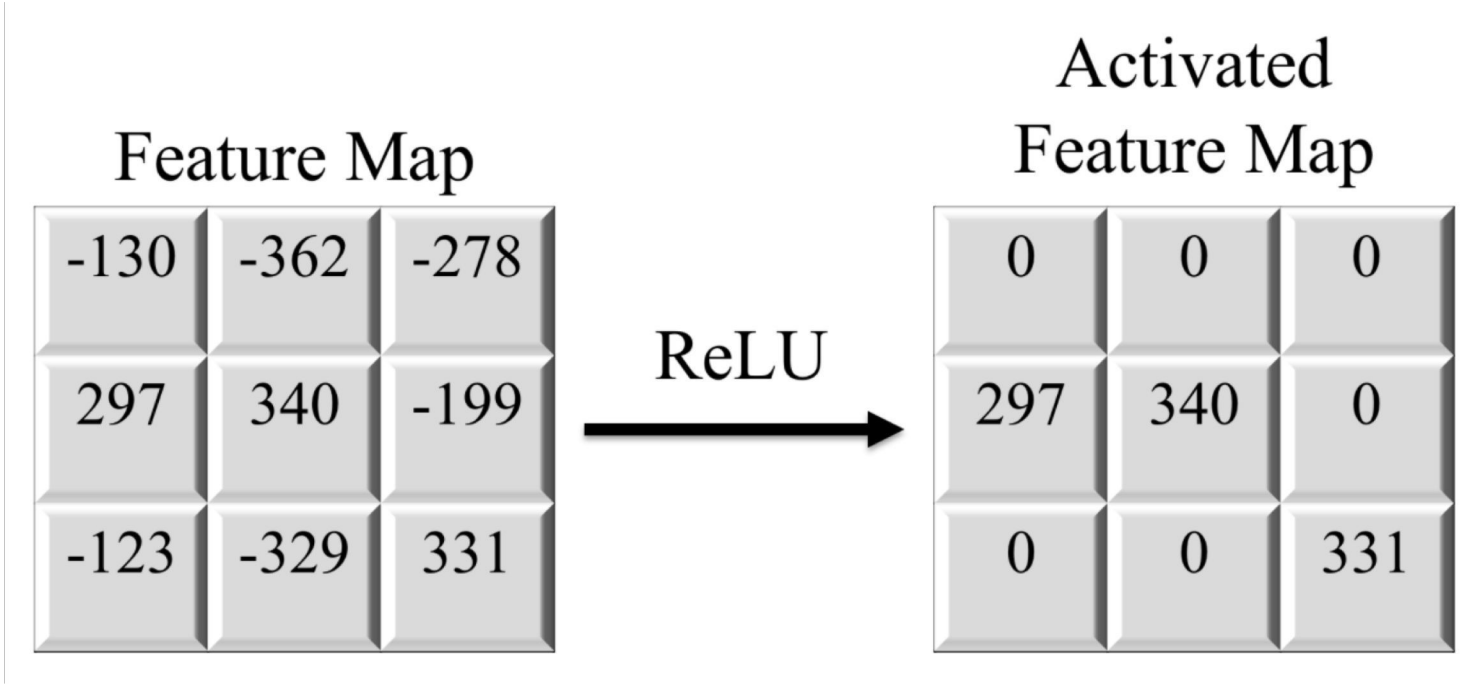
- Each cell of the feature map $Z(i, j, k)$ passes through the activation function to produce an activated output signal.

$$Y(i, j, k) = f(Z(i, j, k)$$



Activation Layer

- The most popular choice for $f()$ is **Rectified Linear Unit (ReLU)** or its variants such as Leaky ReLU, ELU.
- Following figure illustrates how the ReLU activation function computes the **output values from the feature map**.



Activation Layer

- The **activations are done in-place** so there is no need to create a separate output feature map volume.
- Since there is no learning involved, activation layers are usually **removed** from network architecture diagrams.