Module 1: A Beginner's Guide to Computer Vision

Video 5: Understanding a CNN - Striding & Padding

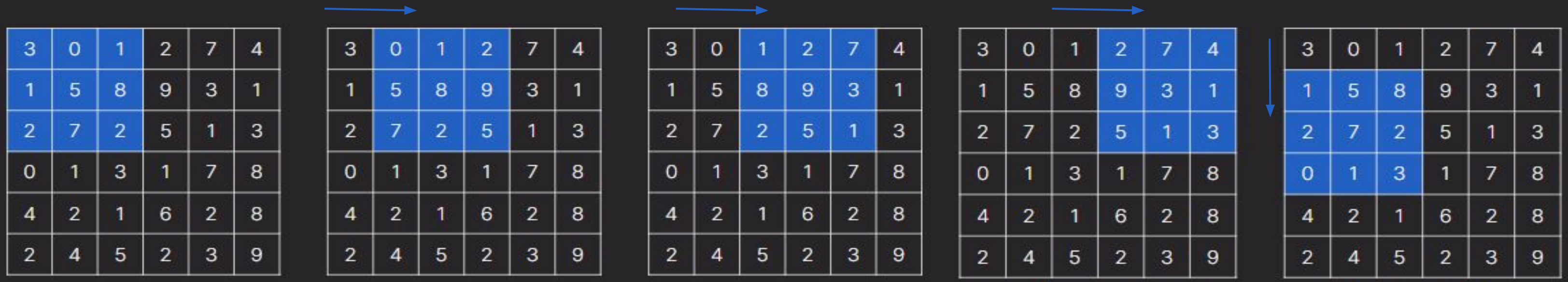
Analytics Vidhya

Analytics Vidhya

Striding

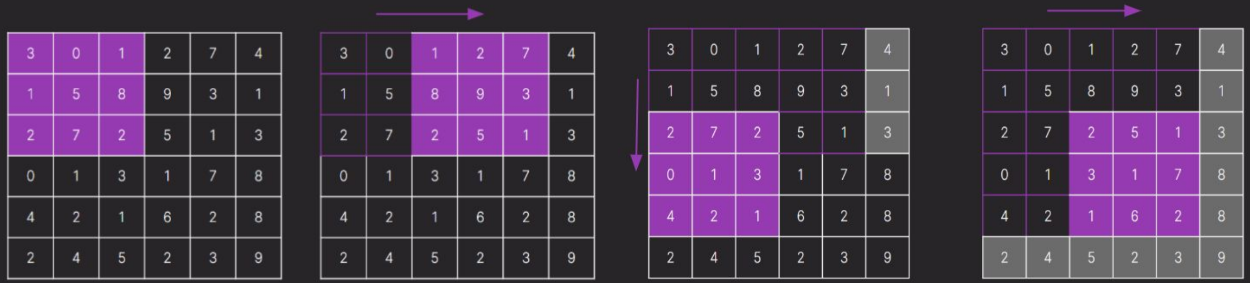
Step size or the number of pixels shifts over the input image.

Stride = 1



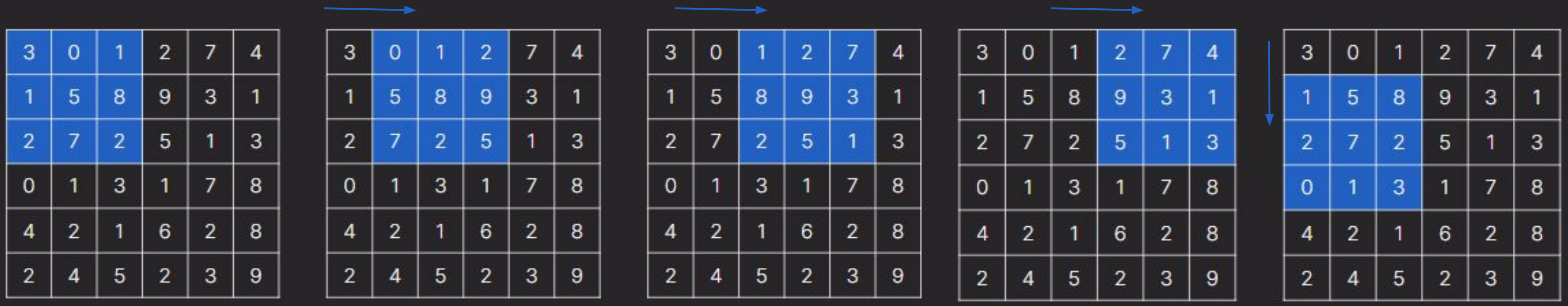
3×3 First Unit Stride | 3×3 Second Unit Stride | 3×3 Third Unit Stride | 3×3 Fourth Unit Stride | 3×3 Fifth Unit Stride

Stride = 2



3×3 1st stride | 3×3 2nd stride | 3×3 3rd stride | 3×3 4th stride

Stride = 1



3×3 First Unit Stride

3×3 Second Unit Stride

3×3 Third Unit Stride

3×3 Fourth Unit Stride

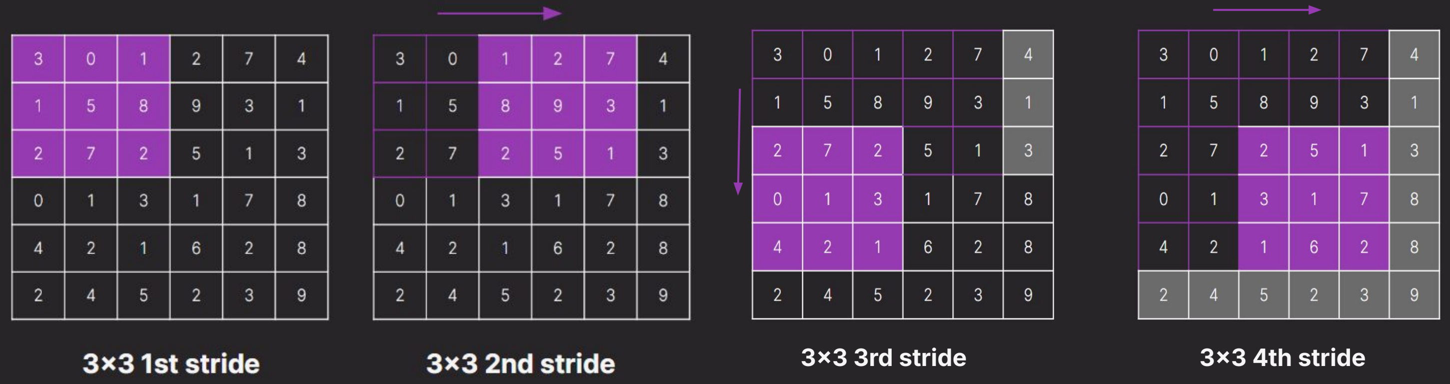
3×3 Fifth Unit Stride

- High resolution output.
- Retains spatial information.

- High computational cost.
- Large output size not always ideal.

Analytics Vidhya

Stride = 2



- ✓ Reduced computational load.
- ✓ Smaller output ideal for deeper networks.

- Some features are missed.
- Some loss of spatial resolution.

The Math Behind Convolution

$$H_{out} = \left\lfloor \frac{(H_{in} - F + 2p)}{s} \right\rfloor + 1$$
$$W_{out} = \left\lfloor \frac{(W_{in} - F + 2p)}{s} \right\rfloor + 1$$

- W_{in} = Input image width. Example: 6 for a 6×6 input matrix.
- H_{in} = Input image height
- F = Filter size (width and height are assumed to be the same)
- p = Padding
- s = Stride value

The Math Behind Convolution

$$H_{out} = \left\lfloor \frac{(H_{in} - F + 2p)}{s} \right\rfloor + 1 = \frac{6 - 3 + 2 \times 0}{1} + 1 = 4$$

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

6 × 6 Matrix of the image

*

-5	0	5
-0.5	0	0.5
-5	0	5

3×3 Filter

=

-6.5	2	22.5	-4
50	19	-5.5	-6
-13.5	10	2	3.5
28.5	-8	10.5	71

Convoluted Matrix for the image with stride = 1
4×4

The Math Behind Convolution

$$\text{Output\_size} = \frac{6 - 3 + 2 \times 0}{2} + 1 = 2.5$$

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

Stride = 2

Round down to 2

- Resulting in a 2×2 matrix

The Math Behind Convolution

$$
\begin{bmatrix} 3 & 0 & 1 & 2 & 7 & 4 \\ 1 & 5 & 8 & 9 & 3 & 1 \\ 2 & 7 & 2 & 5 & 1 & 3 \\ 0 & 1 & 3 & 1 & 7 & 8 \\ 4 & 2 & 1 & 6 & 2 & 8 \\ 2 & 4 & 5 & 2 & 3 & 9 \end{bmatrix}
*
\begin{bmatrix} -5 & 0 & 5 \\ -0.5 & 0 & 0.5 \\ -5 & 0 & 5 \end{bmatrix}
=
\begin{bmatrix} 8.5 & -4 \\ 28.5 & 71 \end{bmatrix}
$$

6×6 Matrix of the image

3×3 Filter

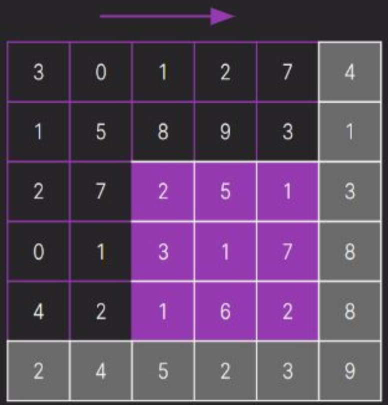
Convoluted Matrix for the Image

(Stride = 2) 2×2

Analytics Vidhya

Padding

- Padding preserves edge information in the feature map.
- Padding helps in increasing the spatial size of the output.



Stride = 2

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

Input Image

0	0	0	0	0	0	0	0
0	3	0	1	2	7	4	0
0	1	5	8	9	3	1	0
0	2	7	2	5	1	3	0
0	0	1	3	1	7	8	0
0	4	2	1	6	2	8	0
0	2	4	5	2	3	9	0
0	0	0	0	0	0	0	0

Padded Image

Zero Padding

0	0	0	0	0	0	0	0
0	3	0	1	2	7	4	0
0	1	5	8	9	3	1	0
0	2	7	2	5	1	3	0
0	0	1	3	1	7	8	0
0	4	2	1	6	2	8	0
0	2	4	5	2	3	9	0
0	0	0	0	0	0	0	0

- **Same padding** ensures that output feature map has same dimensions as input image

The Math Behind Convolution

$$H_{out} = \left\lfloor \frac{(H_{in} - F + 2p)}{s} \right\rfloor + 1 = \frac{6 - 3 + 2 \times 1}{1} + 1 = 6$$

0	0	0	0	0	0	0	0
0	3	0	1	2	7	4	0
0	1	5	8	9	3	1	0
0	2	7	2	5	1	3	0
0	0	1	3	1	7	8	0
0	4	2	1	6	2	8	0
0	2	4	5	2	3	9	0
0	0	0	0	0	0	0	0

*

-5	0	5
-0.5	0	0.5
-5	0	5

3×3 Filter

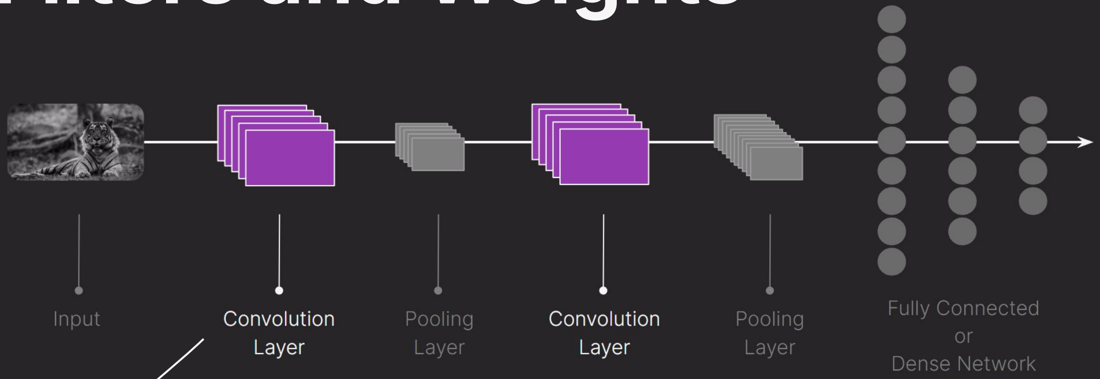
=

25	34	21	-22	-39	-18.5
37.5	-6.5	2	22.5	-4	-41.5
33.5	50	19	-5.5	-6	-50.5
45.5	-13.5	10	2	3.5	-18.5
26	28.5	-8	10.5	71	-51
12	-13.5	19	4	13.5	-11.5

Important Note

- In CNNs, filter values are trainable weights.
- ReLu activation filter is applied to convert negative values to 0.
- Size of the filters is a hyperparameter.

Role of Filters and Weights



-5	0	5
-0.5	0	0.5
-5	0	5

W_1	W_2	W_3
W_4	W_5	W_6
W_7	W_8	W_9

Role of Activation Functions

Input

25	34	21	-22	-39	-18.5
37.5	-6.5	2	22.5	-4	-41.5
33.5	50	19	-5.5	-6	-50.5
45.5	-13.5	10	2	3.5	-18.5
26	28.5	-8	10.5	71	-51
12	-13.5	19	4	13.5	-11.5

Resultant Matrix

Apply the ReLU Activation Function

Analytics Vidhya

Role of Activation Functions

Input

25	34	21	-22	-39	-18.5
37.5	-6.5	2	22.5	-4	-41.5
33.5	50	19	-5.5	-6	-50.5
45.5	-13.5	10	2	3.5	-18.5
26	28.5	-8	10.5	71	-51
12	-13.5	19	4	13.5	-11.5

Resultant Matrix

- On applying the ReLU Activation Function

Input

ReLU Activation Function

25	34	21	-22	-39	-18.5
37.5	-6.5	2	22.5	-4	-41.5
33.5	50	19	-5.5	-6	-50.5
45.5	-13.5	10	2	3.5	-18.5
26	28.5	-8	10.5	71	-51
12	-13.5	19	4	13.5	-11.5

+ **Bias**

0

Role of Activation Functions

ReLu activation filter is applied to convert negative values to 0.

Input

25	34	21	-22	-39	-18.5
37.5	-6.5	2	22.5	-4	-41.5
33.5	50	19	-5.5	-6	-50.5
45.5	-13.5	10	2	3.5	-18.5
26	28.5	-8	10.5	71	-51
12	-13.5	19	4	13.5	-11.5

Resultant Matrix

Apply ReLU

25	34	21	0	0	0
37.5	0	2	22.5	0	0
33.5	50	19	0	0	0
45.5	0	10	2	3.5	0
26	28.5	0	10.5	71	0
12	0	19	4	13.5	0

Final Matrix after ReLU and Bias

Role of Filters in CNNs

- Size of these filters is a hyperparameter.
- Odd sizes like 3x3 are preferred.

-5	0	5
-0.5	0	0.5
-5	0	5

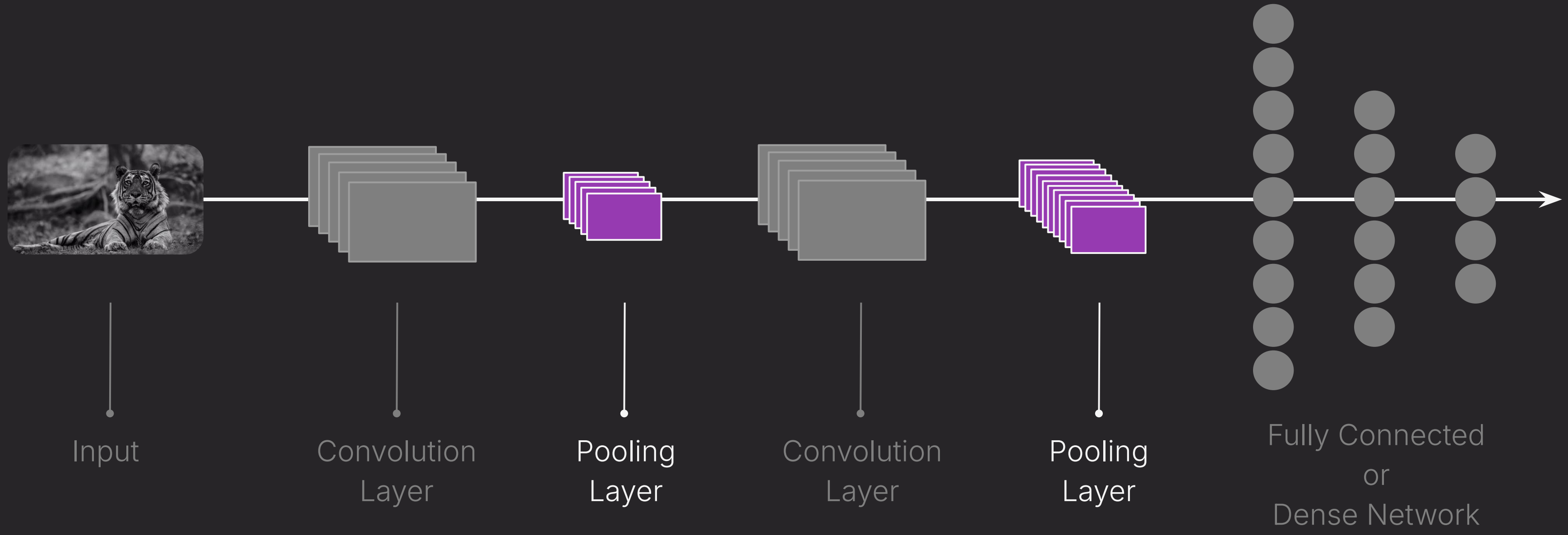
3×3 Matrix

1	0	0	0	-1
1	0	0	0	-1
1	0	5	0	-1
1	0	0	0	-1
1	0	0	0	-1

5×5 Matrix

0	0	0	1	0	0	0
0	0	1	2	1	0	0
0	1	3	5	3	1	0
1	2	5	8	5	2	1
0	1	3	5	3	1	0
0	0	1	2	1	0	0
0	0	0	1	0	0	0

7×7 Matrix


Input
Convolution Layer
Pooling Layer
Convolution Layer
Pooling Layer
Fully Connected or Dense Network