

07 -Penapisan Citra dan Konvolusi

IF4073 Pemrosesan Citra Digital

Oleh: Rinaldi Munir



Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2024

- Di dalam pengolahan citra, sebuah citra perlu dilakukan proses penapisan (*image filtering*) untuk memperoleh citra sesuai dengan tujuan yang diinginkan.

$f(x,y)$



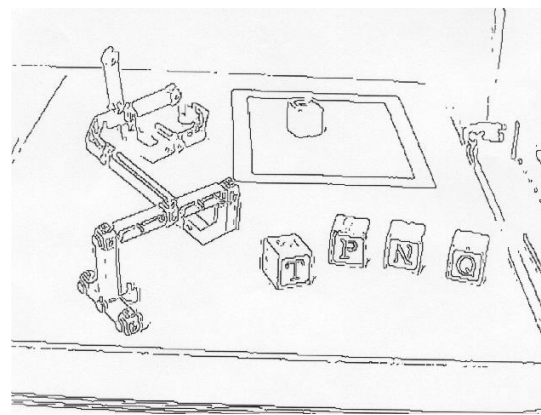
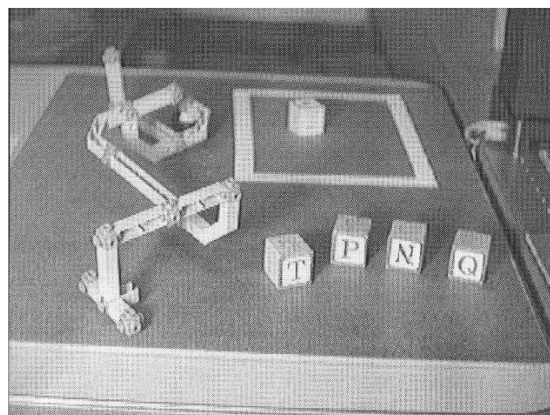
filtering



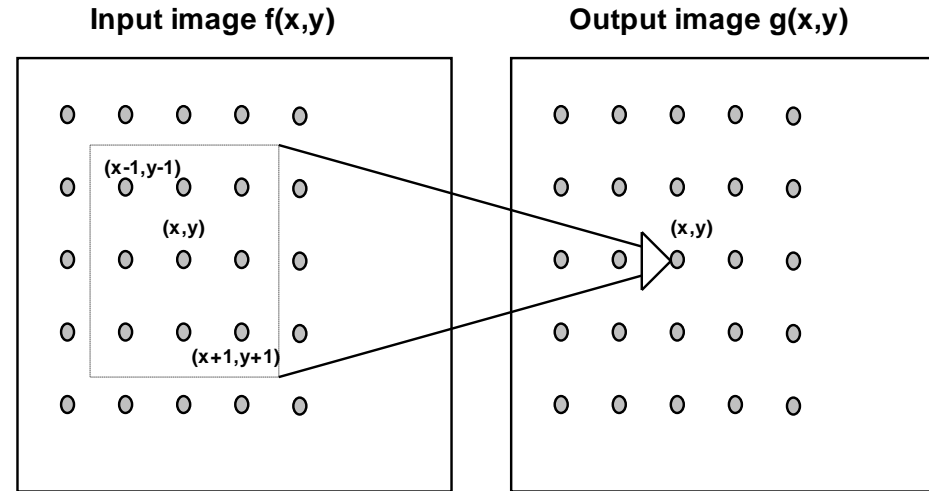
$g(x,y)$



filtering



- Penapisan citra berarti memodifikasi *pixel-pixel* di dalam citra berdasarkan transformasi terhadap nilai-nilai *pixel* tetangganya.



10	5	3
4	5	1
1	1	7

Local image data

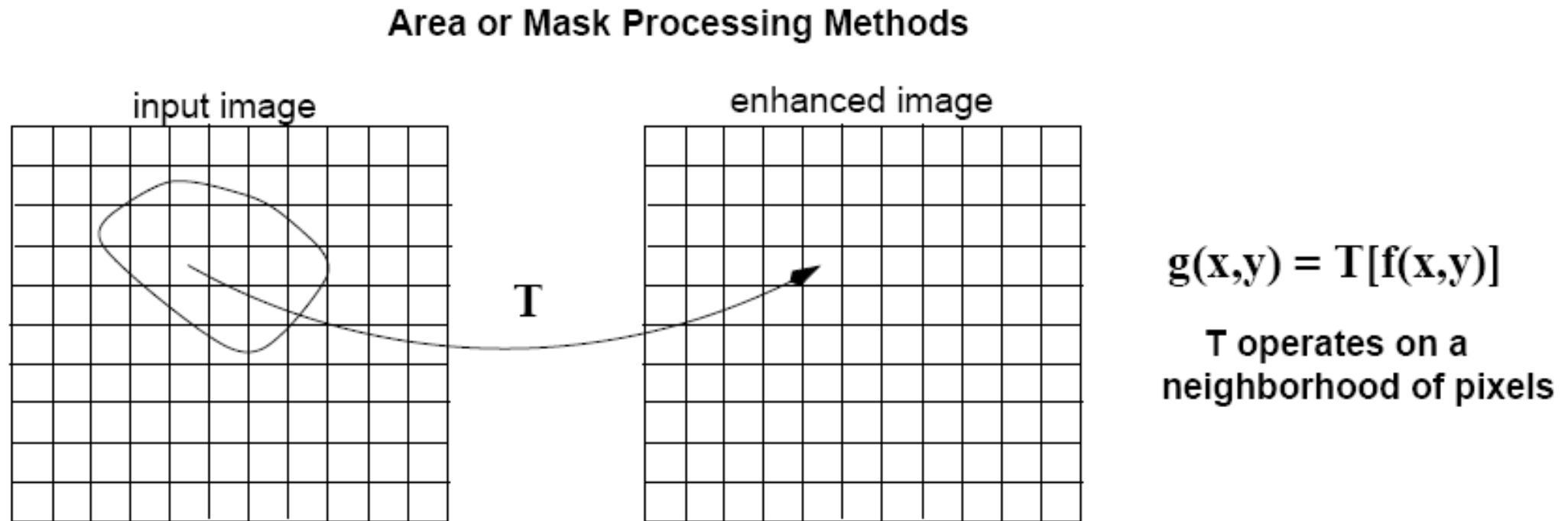
transformasi



	7	

Modified image data

- Penapisan citra termasuk ke dalam tipe operasi aras lokal



- Sebuah operator khusus untuk penapisan citra adalah **konvolusi** (*convolution*) atau *linear filtering*.

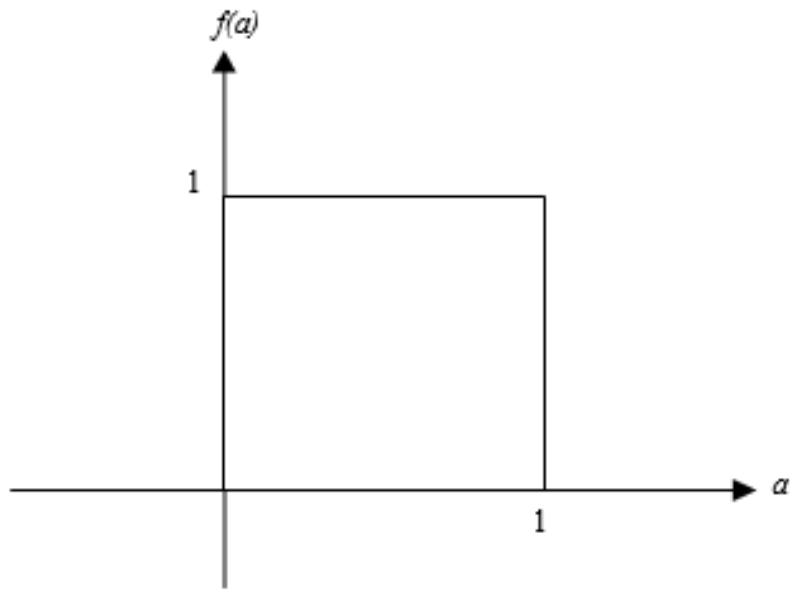
Teori Konvolusi

- Konvolusi 2 buah fungsi $f(x)$ dan $g(x)$ didefinisikan sebagai berikut:

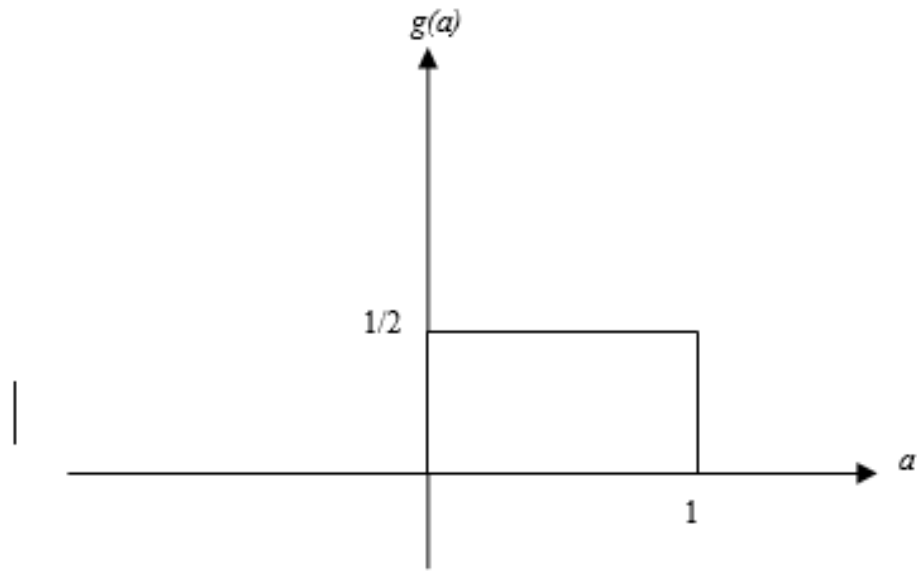
$$h(x) = f(x) * g(x) = \int_{-\infty}^{\infty} f(a)g(x - a)da$$

- Tanda * menyatakan operator konvolusi, dan peubah a adalah peubah bantu (*dummy variable*).
- $g(x)$ disebut **kernel** atau **mask** konvolusi.
- Kernel $g(x)$ dapat dibayangkan sebagai sebuah jendela yang dioperasikan secara bergeser pada sinyal masukan $f(x)$
- Jumlah perkalian kedua fungsi pada setiap titik merupakan hasil konvolusi yang dinyatakan dengan sinyal luaran $h(x)$.

Contoh 1: Misalkan fungsi $f(x)$ dan $g(x)$ diperlihatkan pada gambar berikut.



(a)



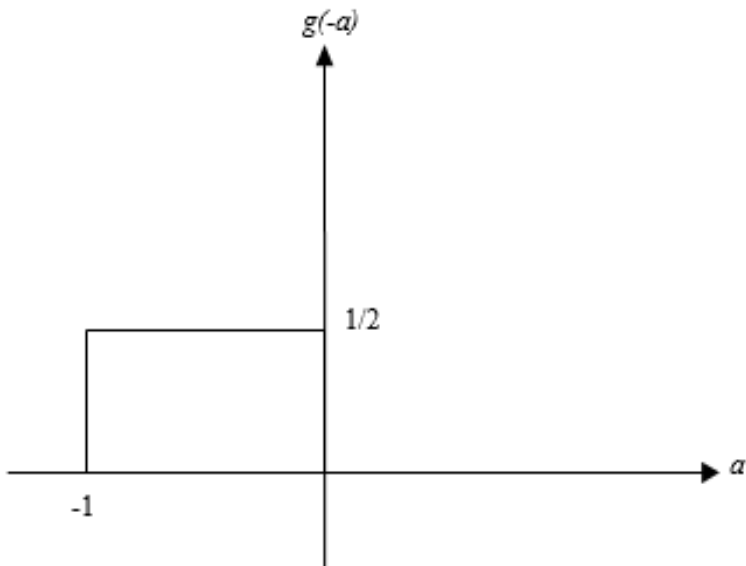
(b)

Tentukan hasil konvolusi $f(x)$ dengan $g(x)$

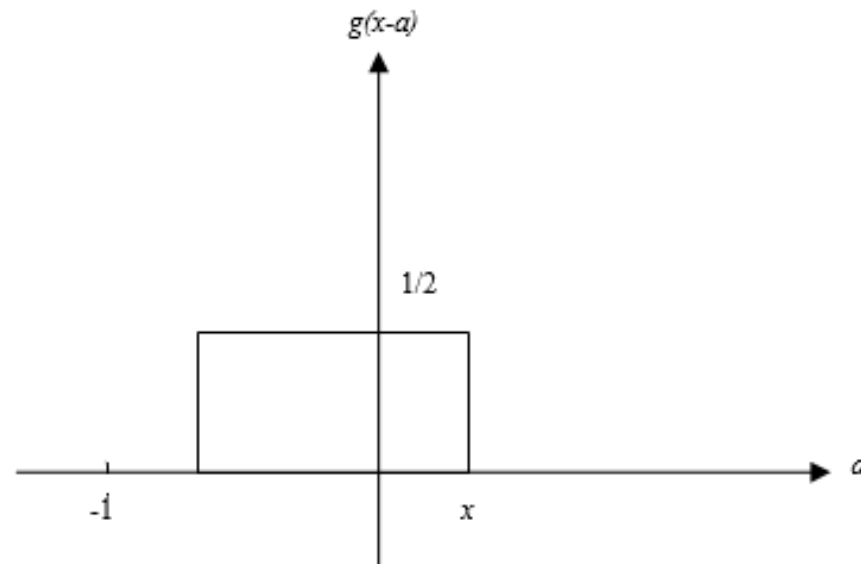
Penyelesaian:

Rumus konvolusi: $h(x) = f(x) * g(x) = \int_{-\infty}^{\infty} f(a)g(x-a)da$

Step 1: Tentukan $g(-a)$

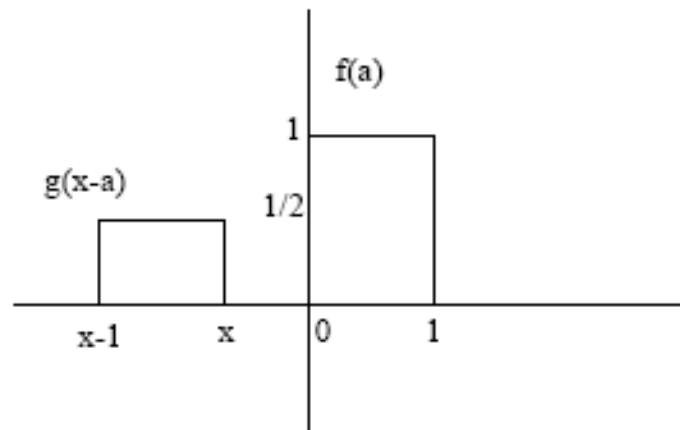


Step 2: Tentukan $g(x-a)$



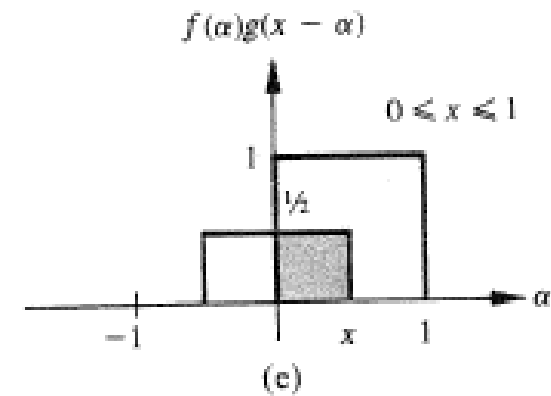
Step 3: consider all possible cases for x:

Case 1: $x < 0$



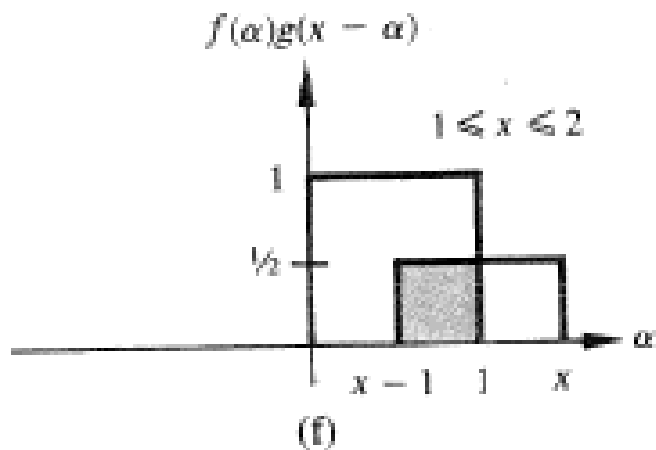
no overlap:
$$\int_{-\infty}^{\infty} f(a)g(x-a)da = 0$$

Case 2: $0 \leq x \leq 1$



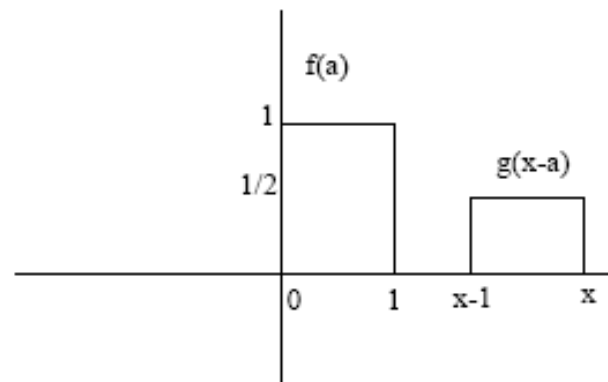
$$\int_{-\infty}^{\infty} f(a)g(x-a)da = \int_0^x 1 \cdot \frac{1}{2} da = \frac{x}{2}$$

Case 3: $1 \leq x \leq 2$



$$\int_{-\infty}^{\infty} f(a)g(x-a)da = \int_{x-1}^1 1 \cdot \frac{1}{2} da = 1 - \frac{x}{2}$$

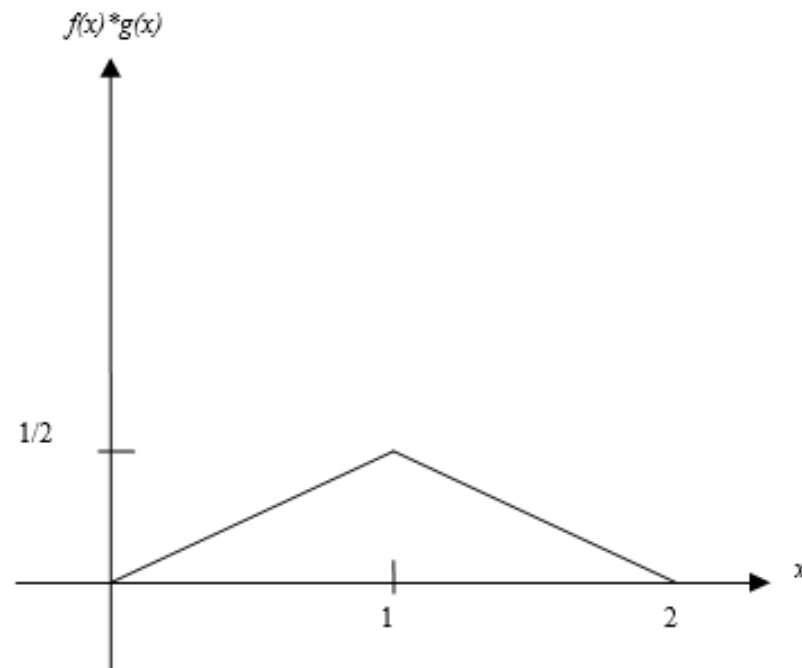
Case 4: $x > 2$



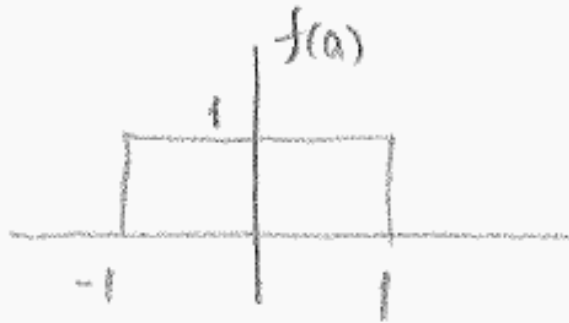
no overlap:
$$\int_{-\infty}^{\infty} f(a)g(x-a)da = 0$$

Hasil akhir:

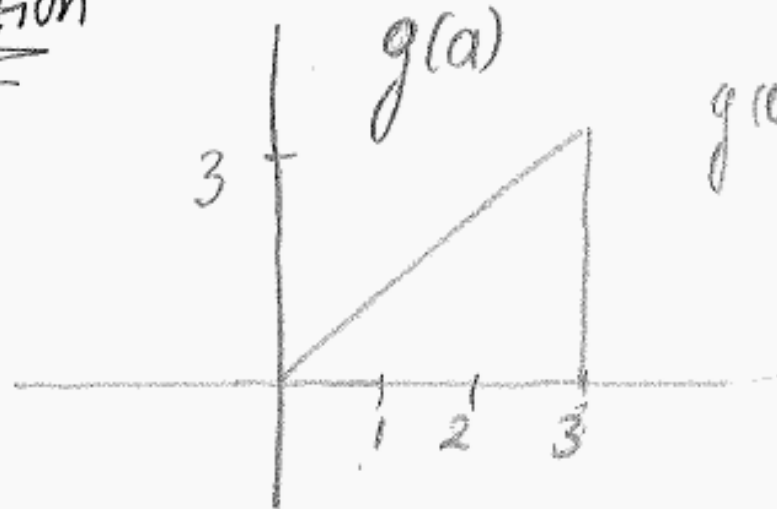
$$f(x) * g(x) = \begin{cases} x/2 & 0 \leq x \leq 1 \\ 1 - x/2 & 1 \leq x \leq 2 \\ 0 & \textit{elsewhere} \end{cases}$$



one more example: convolution

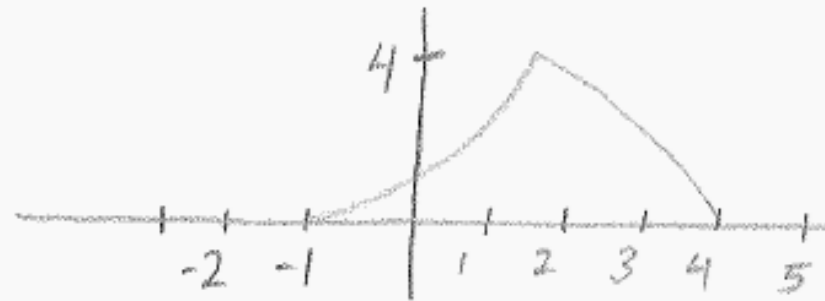


*



$$g(a) = \begin{cases} x & 0 \leq x \leq 3 \\ 0 & \text{elsewhere} \end{cases}$$

$$f(x) * g(x) = \begin{cases} \frac{1}{2}(x+1)^2 & -1 \leq x \leq 1 \\ 2x & 1 \leq x \leq 2 \\ 4+x - \frac{1}{2}x^2 & 2 \leq x \leq 4 \\ 0 & \text{otherwise} \end{cases}$$



Konvolusi pada fungsi diskrit

- Konvolusi pada fungsi kontinu $f(x)$ dan $g(x)$:

$$h(x) = f(x) * g(x) = \int_{-\infty}^{\infty} f(a)g(x-a)da$$

- Konvolusi pada fungsi diskrit:

$$h(x) = f(x) * g(x) = \sum_{a=-\infty}^{\infty} f(a)g(x-a)$$

- Konvolusi bersifat komutatif:

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

Sifat-sifat konvolusi

- Commutative:

$$f * g = g * f$$

- Associative:

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

- Homogeneous:

$$f * (\lambda g) = \lambda f * g$$

- Additive (Distributive):

$$f * (g + h) = f * g + f * h$$

- Shift-Invariant

$$f * g(x - x_0, y - y_0) = (f * g)(x - x_0, y - y_0)$$

Konvolusi pada fungsi dwimatra

- Citra adalah sinyal dwimatra (fungsi dwimatra).
- Untuk fungsi dwimatra (fungsi dengan dua peubah), operasi konvolusi didefinisikan sebagai berikut:

a) untuk fungsi kontinu

$$h(x, y) = f(x, y) * g(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(a, b) g(x - a, y - b) da db$$

b) untuk fungsi diskrit

$$h(x, y) = f(x, y) * g(x, y) = \sum_{a=-\infty}^{\infty} \sum_{b=-\infty}^{\infty} f(a, b) g(x - a, y - b)$$

- $g(x, y)$ dinamakan *convolution filter*, *convolution mask*, *kernel*, atau *template*

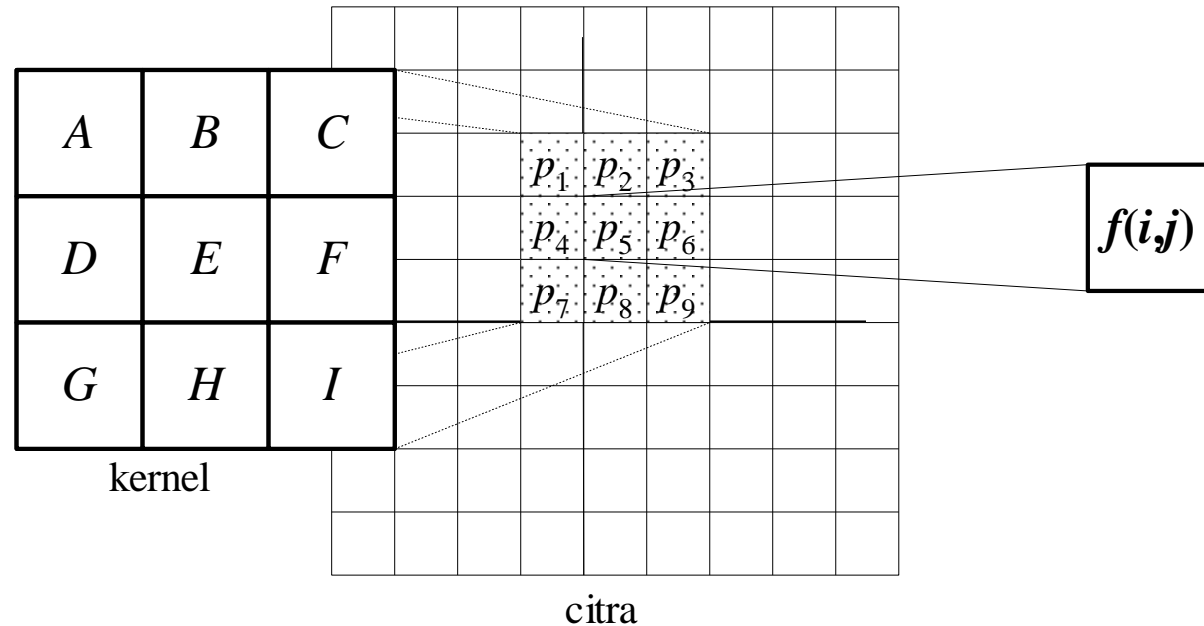
- Dalam ranah diskrit kernel konvolusi dinyatakan dalam bentuk matriks (umumnya 3×3 , ada juga 2×2 atau 2×1 atau 1×2).

- **Kernel:** A kernel is a (usually) small matrix of numbers that is used in image convolutions.
 - Differently sized kernels containing different patterns of numbers produce different results under convolution.
 - The size of a kernel is arbitrary but 3×3 is often used

Example kernel:

0	1	0
1	1	1
0	1	0

- Ilustrasi konvolusi:



- Jadi, konvolusi dapat dipandang sebagai kombinasi linier dari vektor *pixel* dengan vektor kernel.

$$f(i,j) = (A p_1 + B p_2 + C p_3 + D p_4 + E p_5 + F p_6 + G p_7 + H p_8 + I p_9)$$

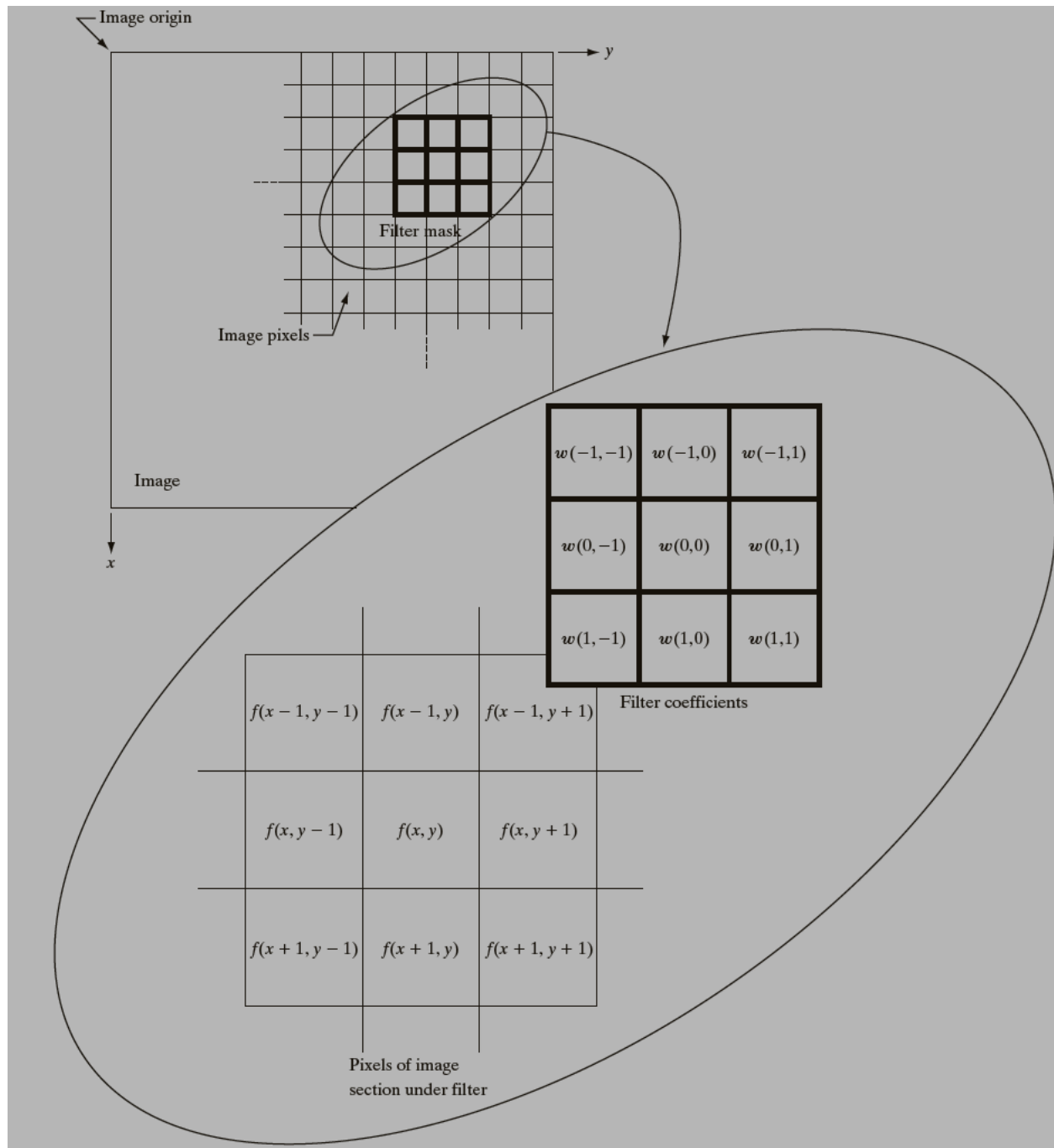
- Jika jumlah nilai di dalam kernel > 1 , maka $f(i,j)$ dibagi dengan jumlah tersebut. Jika jumlahnya nol, maka $f(i,j)$ tidak perlu dibagi (atau bagi dengan 1).

$$V = \left\lfloor \frac{\sum_{i=1}^q \left(\sum_{j=1}^q f_{ij} d_{ij} \right)}{F} \right\rfloor$$

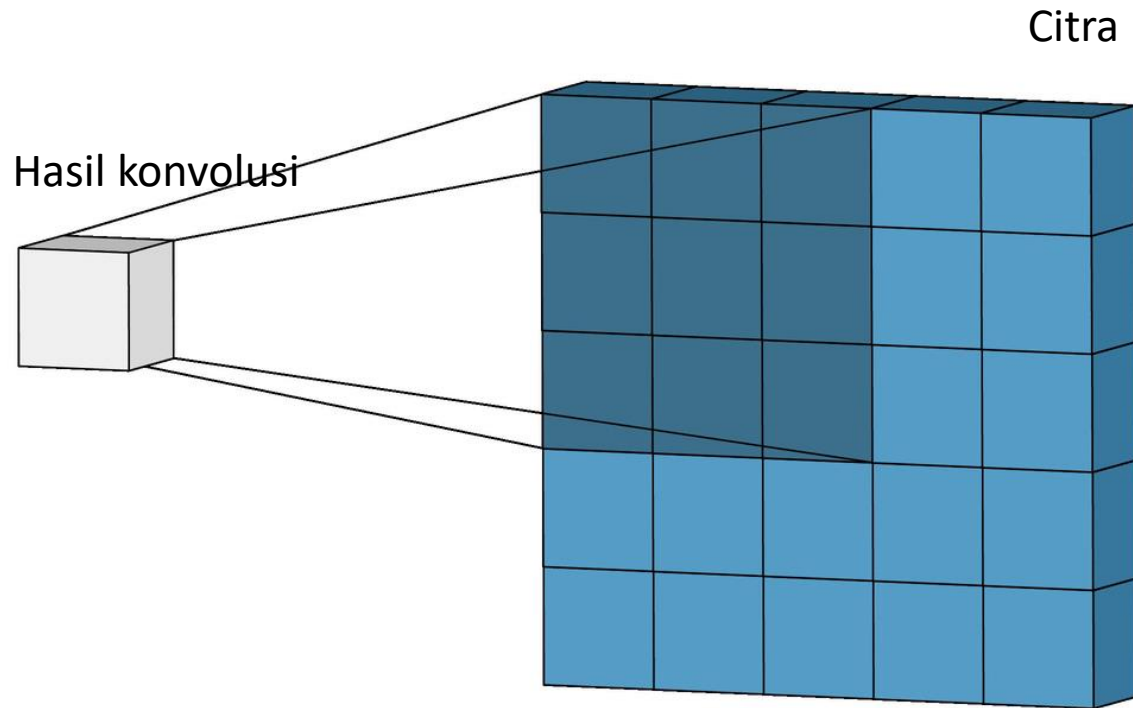
Where:

- f_{ij} = the coefficient of a convolution kernel at position i,j (in the kernel)
- d_{ij} = the data value of the pixel that corresponds to f_{ij}
- q = the dimension of the kernel, assuming a square kernel (if $q = 3$, the kernel is 3×3)
- F = either the sum of the coefficients of the kernel, or 1 if the sum of coefficients is 0
- V = the output pixel value

In cases where V is less than 0, V is clipped to 0.



- Operasi konvolusi dilakukan dengan menggeser *kernel* di dalam citra *pixel per pixel*. Hasil konvolusi disimpan di dalam matriks yang baru.



Contoh 1. Misalkan citra $f(x, y)$ berukuran 5×5 dikonvolusi dengan sebuah *kernel* atau *mask* berukuran 3×3 , masing-masing adalah sbb:

$$f(x, y) = \begin{bmatrix} 4 & 4 & 3 & 5 & 4 \\ 6 & 6 & 5 & 5 & 2 \\ 5 & 6 & 6 & 6 & 2 \\ 6 & 7 & 5 & 5 & 3 \\ 3 & 5 & 2 & 4 & 4 \end{bmatrix}$$

$$g(x, y) = \begin{bmatrix} 0 & -1 & 0 \\ -1 & \bullet 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Catatan:

Jumlah nilai di dalam kernel =
 $-1 + 4 - 1 - 1 - 1 = 0$.

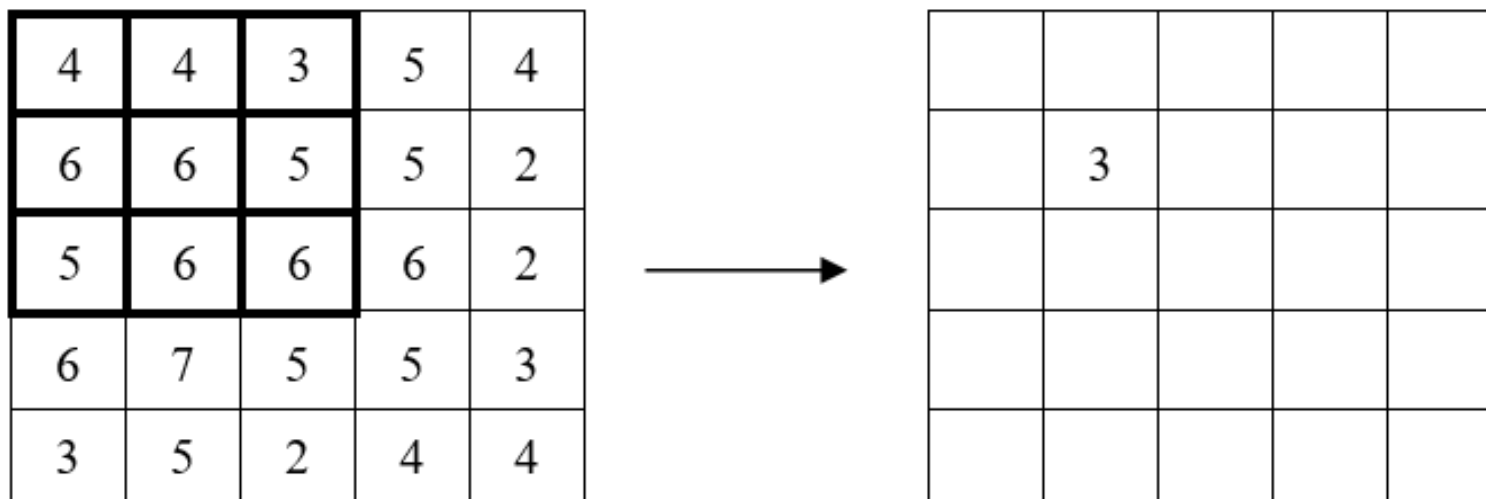
Hasil konvolusi tidak perlu dibagi

(Keterangan: Tanda \bullet menyatakan posisi $(0, 0)$ dari *kernel*)

$$g(x, y) = \begin{bmatrix} 0 & -1 & 0 \\ -1 & \bullet 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

- Operasi konvolusi $f(x, y) * g(x, y)$ adalah sbb:

(1) Tempatkan *kernel* pada sudut kiri atas, kemudian hitung nilai *pixel* pada posisi (0, 0) dari *kernel*:

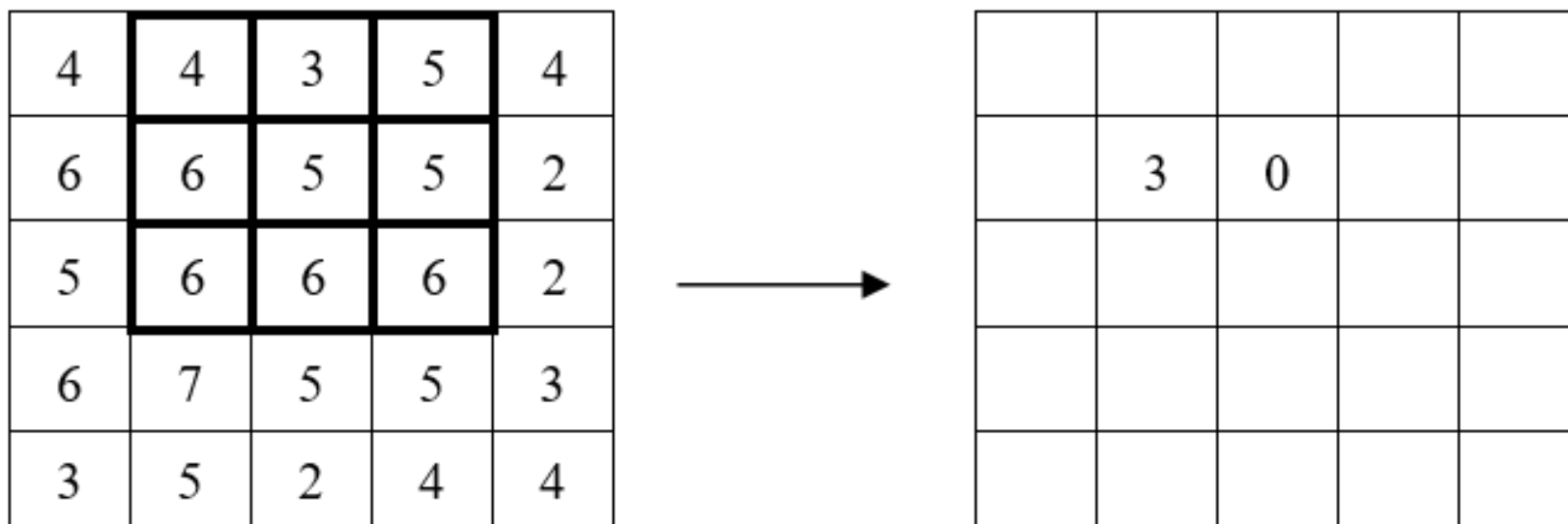


Hasil konvolusi = 3. Nilai ini dihitung dengan cara berikut:

$$(0 \times 4) + (-1 \times 4) + (0 \times 3) + (-1 \times 6) + (4 \times 6) + (-1 \times 5) + (0 \times 5) + (-1 \times 6) + (0 \times 6) = 3$$

$$g(x, y) = \begin{bmatrix} 0 & -1 & 0 \\ -1 & \bullet 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

(2) Geser *kernel* satu *pixel* ke kanan, kemudian hitung nilai *pixel* pada posisi 0) dari *kernel*:

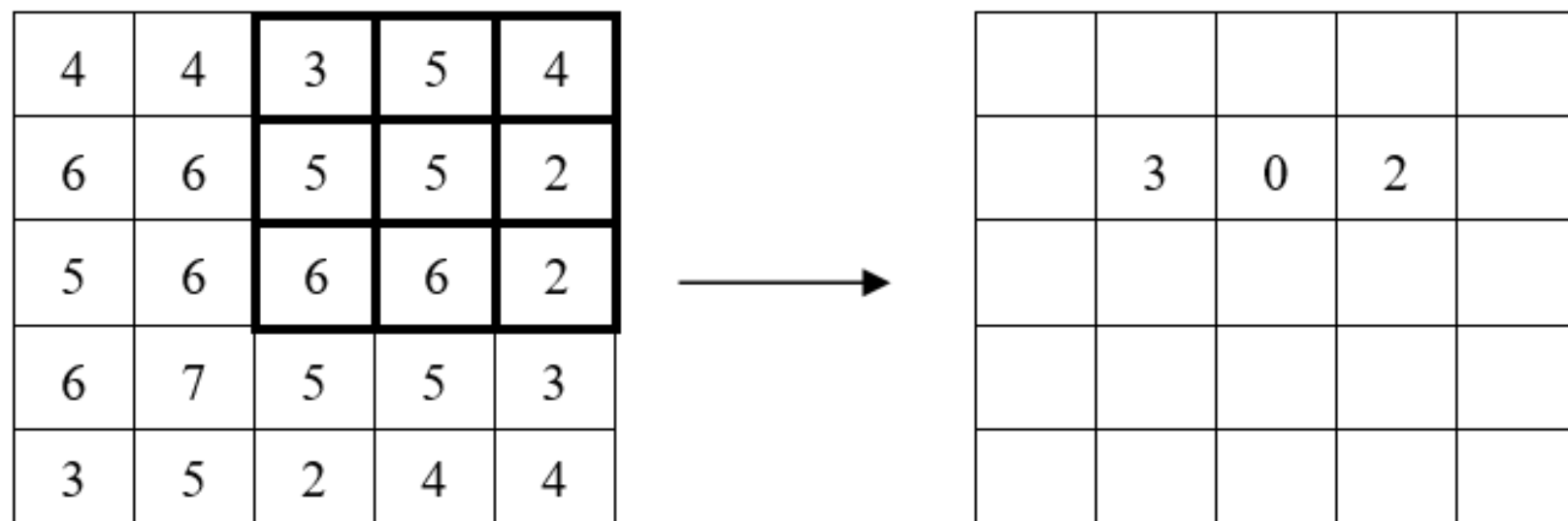


Hasil konvolusi = 0. Nilai ini dihitung dengan dengan cara berikut:

$$(0 \times 4) + (-1 \times 3) + (0 \times 5) + (-1 \times 6) + (4 \times 5) + (-1 \times 5) + (0 \times 6) + (-1 \times 6) + (0 \times 6) = 0$$

$$g(x, y) = \begin{bmatrix} 0 & -1 & 0 \\ -1 & \bullet 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

(3) Geser *kernel* satu *pixel* ke kanan, kemudian hitung nilai *pixel* pada posisi (0, 0) dari *kernel*:

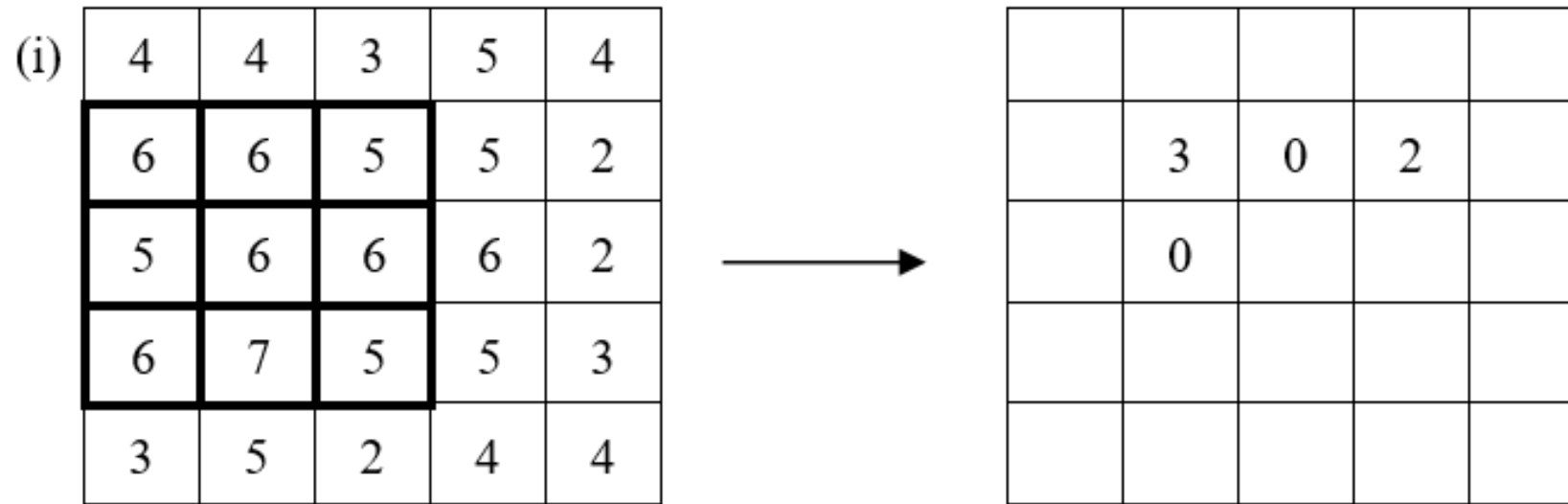


Hasil konvolusi = 2. Nilai ini dihitung dengan cara berikut:

$$(0 \times 3) + (-1 \times 5) + (0 \times 4) + (-1 \times 5) + (4 \times 5) + (-1 \times 2) + (0 \times 6) + (-1 \times 6) + (0 \times 2) = 2$$

$$g(x, y) = \begin{bmatrix} 0 & -1 & 0 \\ -1 & \bullet 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

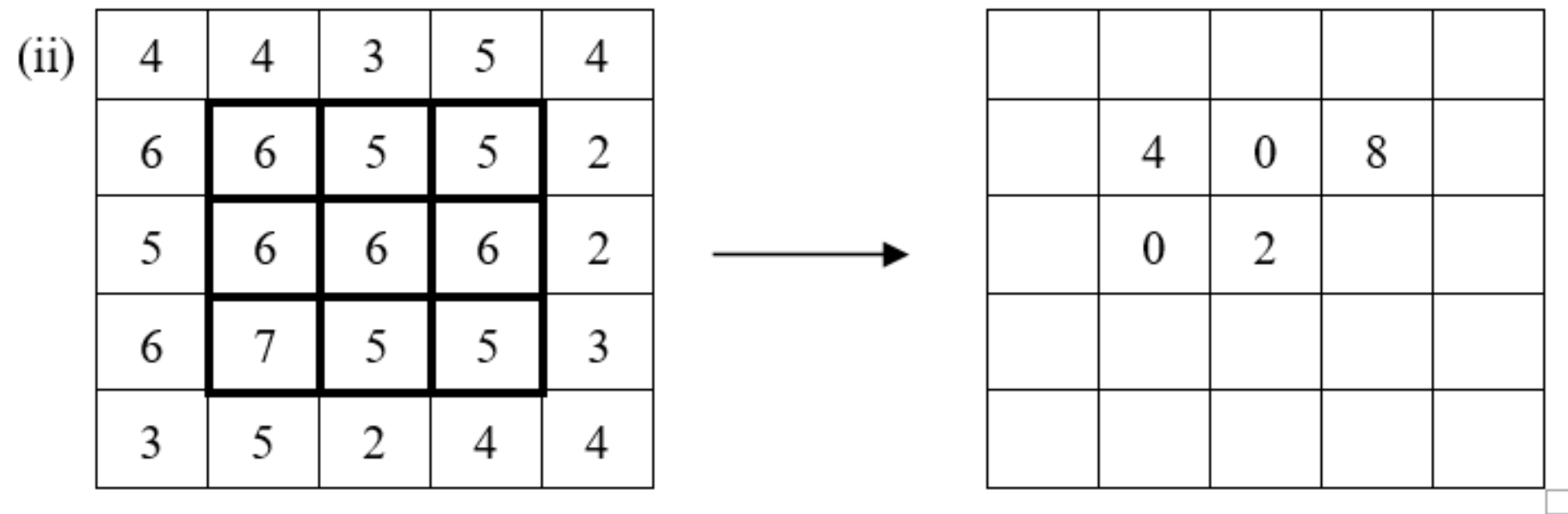
(4) Selanjutnya, geser *kernel* satu *pixel* ke bawah, lalu mulai lagi melakukan konvolusi dari sisi kiri citra. Setiap kali konvolusi, geser *kernel* satu *pixel* ke kanan:



Hasil konvolusi = 0. Nilai ini dihitung dengan cara berikut:

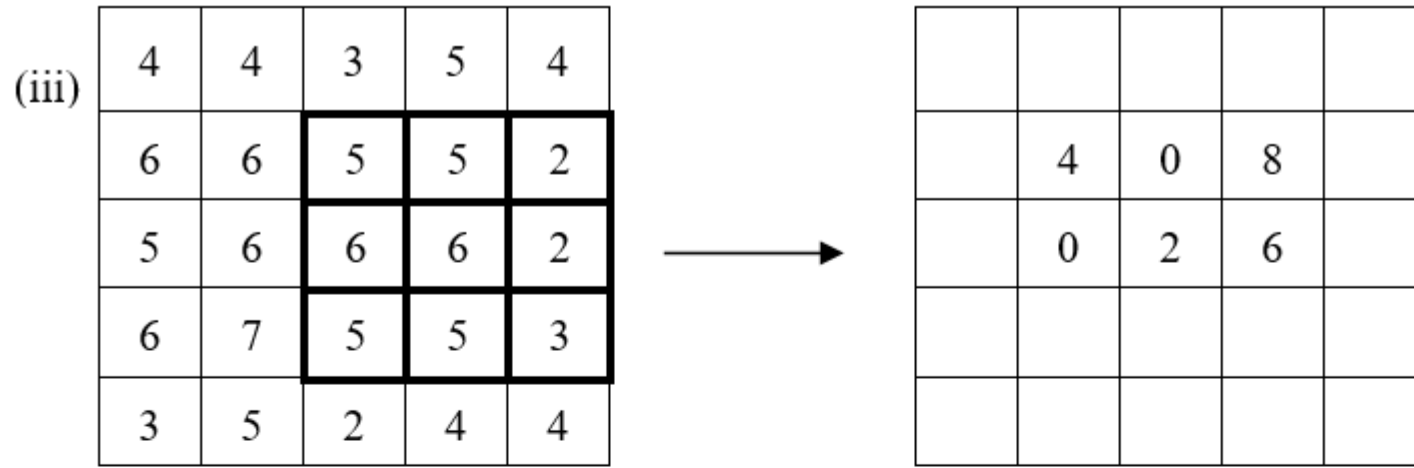
$$(0 \times 6) + (-1 \times 6) + (0 \times 5) + (-1 \times 5) + (4 \times 6) + (-1 \times 6) + (0 \times 6) + (-1 \times 7) + (0 \times 5) = 0$$

$$g(x, y) = \begin{bmatrix} 0 & -1 & 0 \\ -1 & \bullet 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$



Hasil konvolusi = 2. Nilai ini dihitung dengan cara berikut:

$$(0 \times 6) + (-1 \times 5) + (0 \times 5) + (-1 \times 6) + (4 \times 6) + (-1 \times 6) + (0 \times 7) + (-1 \times 5) + (0 \times 5) = 2$$



$$g(x, y) = \begin{bmatrix} 0 & -1 & 0 \\ -1 & \bullet 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

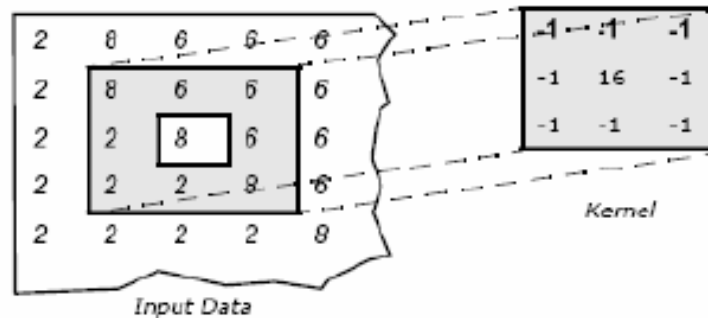
Hasil konvolusi = 6. Nilai ini dihitung dengan cara berikut:

$$(0 \times 5) + (-1 \times 5) + (0 \times 2) + (-1 \times 6) + (4 \times 6) + (-1 \times 2) + (0 \times 5) + (-1 \times 5) + (0 \times 3) = 6$$

Dengan cara yang sama seperti di atas, maka *pixel-pixel* pada baris ketiga dikonvolusi sehingga menghasilkan:

	4	0	8	
	0	2	6	
	6	0	2	

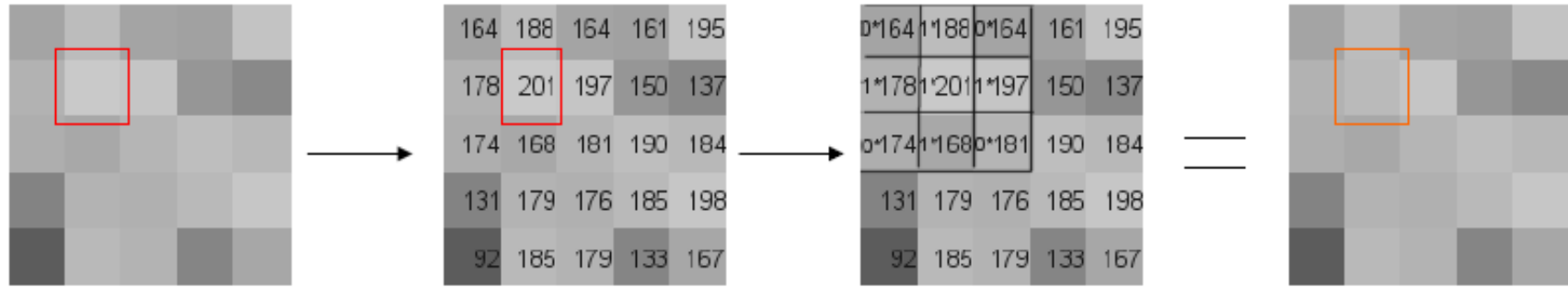
More examples



$$\begin{aligned} & \text{integer } [(-1 \times 8) + (-1 \times 6) + (-1 \times 6) + (-1 \times 2) + (16 \times 8) + \\ & (-1 \times 6) + \\ & (-1 \times 2) + (-1 \times 2) + (-1 \times 8) \div (-1 + -1 + -1 + -1 + 16 + -1 + \\ & -1 + -1 + -1)] \\ & = \text{int} [(128-40) / (16-8)] \\ & = \text{int} (88 / 8) = \text{int} (11) = 11 \end{aligned}$$

Sumber: Image Convolution, Jamie Ludwig, Satellite Digital Image Analysis, 581
Portland State University

Example



Original image

Image with color values placed over it

Image with 3x3 kernel placed over it

Output image

164	188	164
178	201	197
174	168	181

Color values

×

0	1	0
1	1	1
0	1	0

Kernel

Divided by the sum of the kernel

$932 \div 5 = \text{new pixel color}$

- Masalah timbul bila *pixel* yang dikonvolusi adalah *pixel–pixel* pinggir (*border*), karena beberapa koefisien konvolusi tidak dapat diposisikan pada *pixel-pixel* citra (efek “menggantung”),

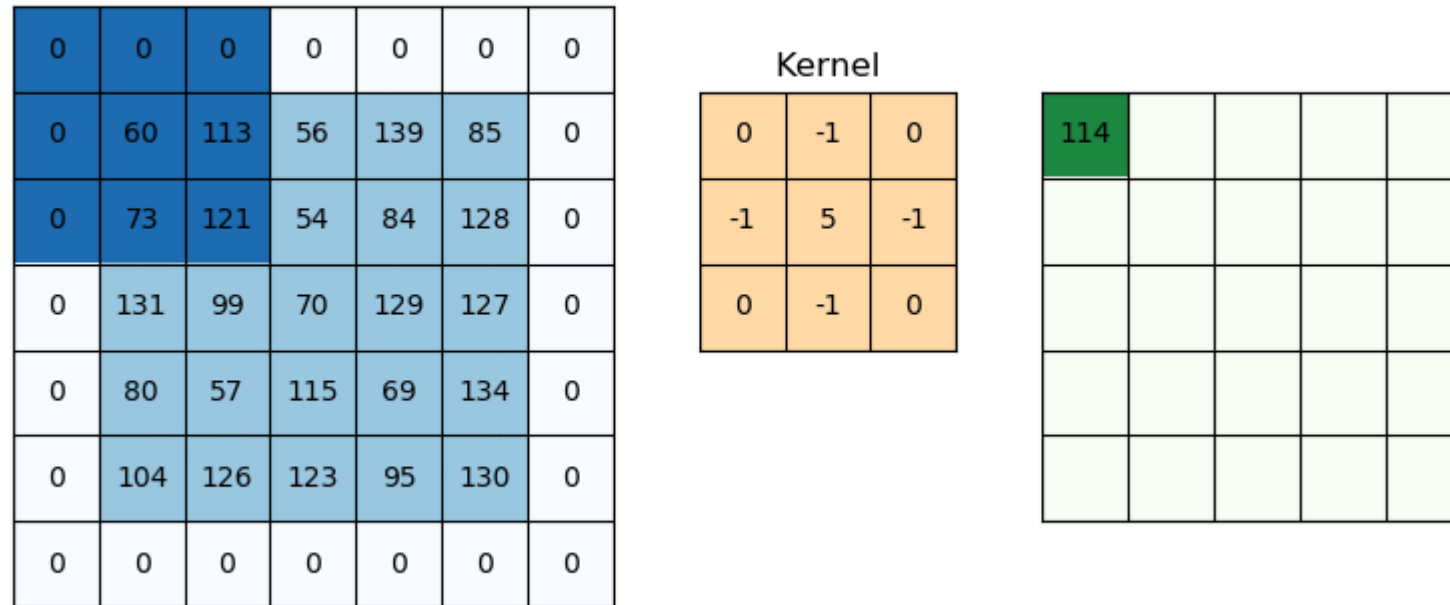
4	4	3	5	4	?
6	6	5	5	2	?
5	6	6	6	2	?
6	7	5	5	3	
3	5	2	4	4	

- Masalah “menggantung” seperti ini selalu terjadi pada *pixel-pixel* pinggir kiri, kanan, atas, dan bawah.
- Solusi untuk masalah ini adalah:
 1. *Pixel-pixel* pinggir diabaikan, tidak di-konvolusi. Solusi ini banyak dipakai di dalam pustaka fungsi-fungsi pengolahan citra. Dengan cara seperti ini, maka *pixel-pixel* pinggir nilainya tetap sama seperti citra asal

4	4	3	5	4
6	4	0	8	2
5	0	2	6	2
6	6	0	2	3
3	5	2	4	4

Gambar *Pixel-pixel pinggir (yang tidak diarsir) tidak dikonvolusi (dari Contoh 1)*

2. Elemen yang ditandai dengan “?” diasumsikan bernilai 0, sehingga konvolusi *pixel-pixel* pinggir dapat dilakukan. Penambahan *pixel* bernilai 0 dinamakan *padding*.

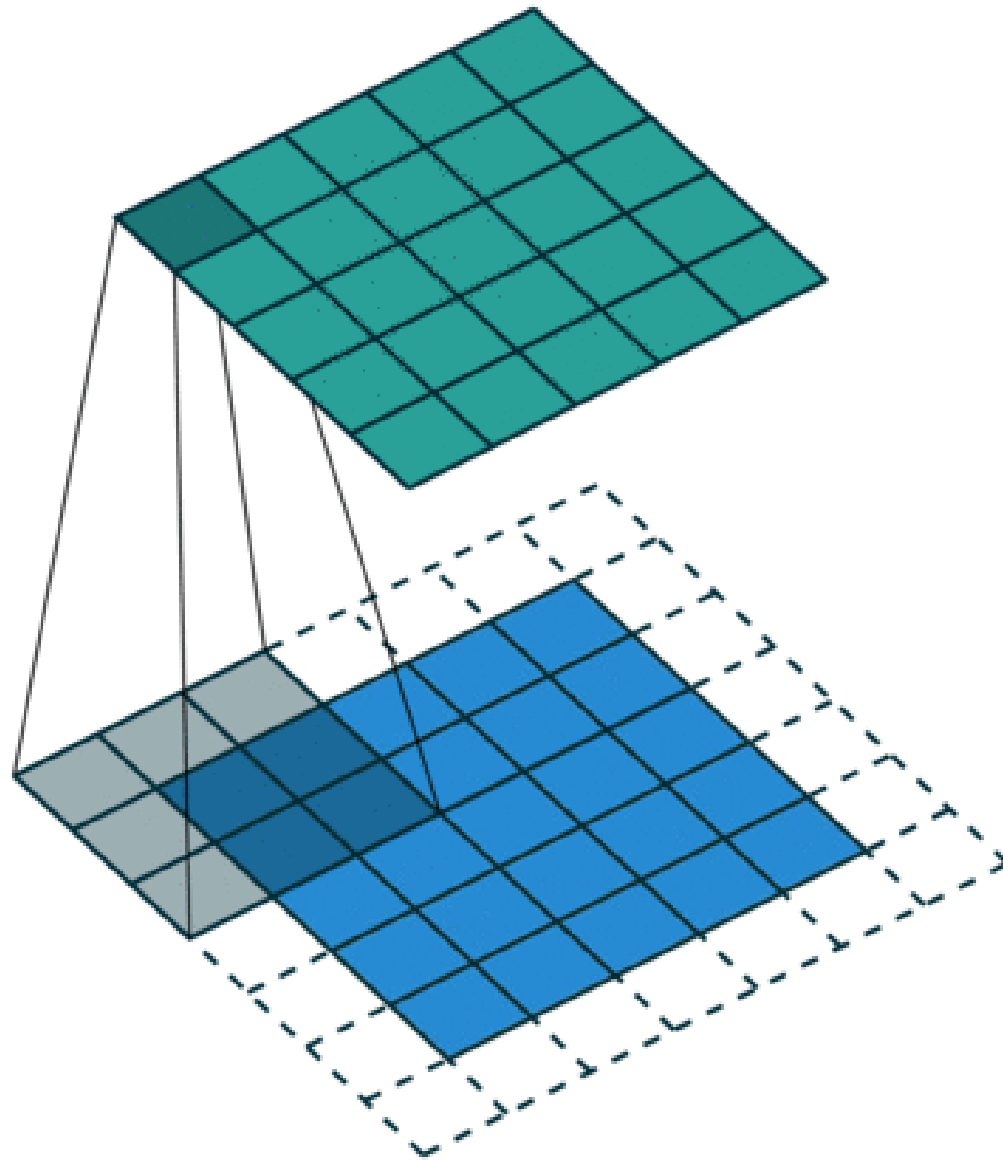


Sumber gambar: <https://medium.com/@draj0718/zero-padding-in-convolutional-neural-networks-bf1410438e99>

3. Duplikasi elemen citra, misalnya elemen kolom pertama disalin ke kolom $N - 1$, kolom ke- N disalin ke kolom $N+1$, baris pertama disalin ke baris $M-1$, baris ke M disalin ke baris $M+1$, lalu konvolusi dapat dilakukan terhadap *pixel-pixel* pinggir tersebut.

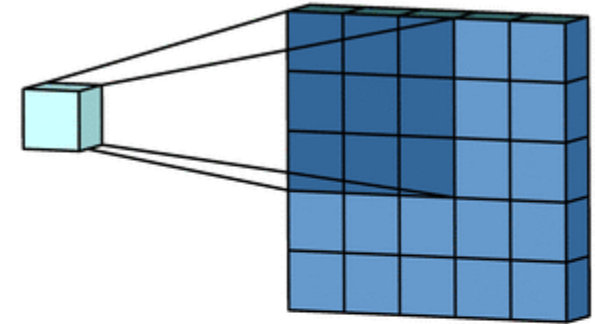
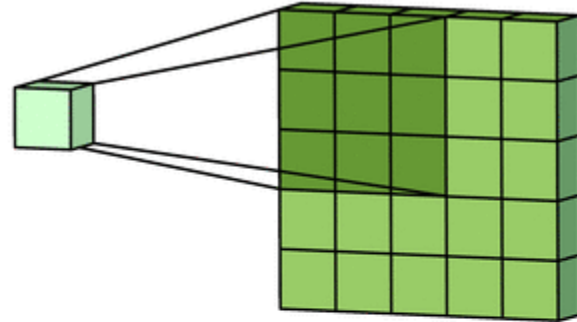
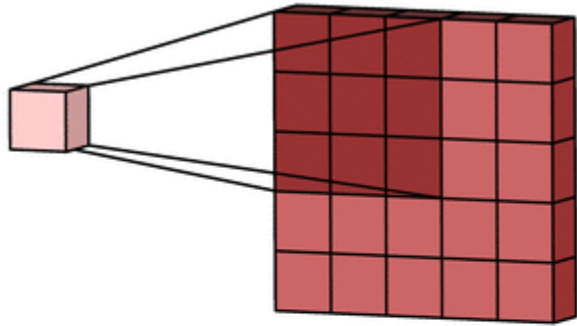
$$\begin{bmatrix} 20 & 30 & 30 & 40 & 45 \\ 25 & 35 & 40 & 42 & 40 \\ 30 & 32 & 42 & 40 & 44 \\ 35 & 38 & 41 & 45 & 48 \\ 40 & 39 & 40 & 40 & 40 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 20 & 30 & 30 & 40 & 45 & 0 \\ 20 & 20 & 30 & 30 & 40 & 45 & 45 \\ 25 & 25 & 35 & 40 & 42 & 40 & 40 \\ 30 & 30 & 32 & 42 & 40 & 44 & 44 \\ 35 & 35 & 38 & 41 & 45 & 48 & 48 \\ 40 & 40 & 39 & 40 & 40 & 40 & 40 \\ 0 & 40 & 39 & 40 & 40 & 40 & 0 \end{bmatrix}$$

- Solusi dengan ketiga pendekatan di atas mengasumsikan bagian pinggir citra lebarnya sangat kecil (hanya satu *pixel*) relatif dibandingkan dengan ukuran citra, sehingga *pixel-pixel* pinggir tidak memperlihatkan efek yang kasat mata



Sumber: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>

- Untuk citra berwarna, konvolusi citra dengan kernel dilakukan pada setiap kanal warna R, G, dan B.



Algoritma *Konvolusi citra dengan sebuah mask yang berukuran 3×3 .*

```
void konvolusi(citra Image, citra ImageResult, imatriks Mask, int M, int N)
/* Mengkonvolusi citra Image yang berukuran M x N dengan mask 3 x 3. Hasil
konvolusi disimpan di dalam matriks ImageResult.
*/
{ int i, j;

  for (i=1; i<=M-3; i++)
    for(j=1; j<=N-3; j++)
      ImageResult[i][j]=
        Image[i-1][j-1]*Mask[0][0] +
        Image[i-1][j+1]*Mask[0][1] +
        Image[i-1][j]*Mask[0][2] +
        Image[i][j-1]*Mask[1][0] +
        Image[i][j]*Mask[1][1] +
        Image[i][j+1]*Mask[1][2] +
        Image[i+1][j-1]*Mask[2][0] +
        Image[i+1][j]*Mask[2][1] +
        Image[i+1][j+1]*Mask[2][2];
}
```

Pixel yang dikonvolusi adalah elemen (i, j) . Delapan buah *pixel* yang bertetangga dengan *pixel* (i, j) adalah sbb:

$i-1, j-1$	$i-1, j$	$i-1, j+1$
$i, j-1$	i, j	$i, j+1$
$i+1, j-1$	$i+1, j$	$i+1, j+1$

Konvolusi berguna pada proses pengolahan citra seperti:

- perbaikan kualitas citra (*image enhancement*)
- penghilangan derau
- mengurangi erotan
- penghalusan/pelembutan citra
- deteksi tepi, penajaman tepi
- dll

Contoh: Untuk mendeteksi tepi-tepi di dalam citra, penapis yang digunakan adalah sebuah kernel g berukuran 3×3 :

$$g(x, y) = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

```
f = imread('cameraman.bmp');  
figure, imshow(I);  
g = [-1 -1 -1; -1 8 -1; -1 -1 -1];  
h = uint8(conv2(double(f), double(g)));  
figure; imshow(ha)
```



$f(x, y)$

$* g(x, y) =$



$h(x, y)$

Some other kernel examples

1	1	1
1	1	1
1	1	1

Unweighted 3x3 smoothing kernel

0	1	0
1	4	1
0	1	0

Weighted 3x3 smoothing kernel with Gaussian blur

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

Kernel to make image sharper

-1	-1	-1
-1	9	-1
-1	-1	-1

Intensified sharper image



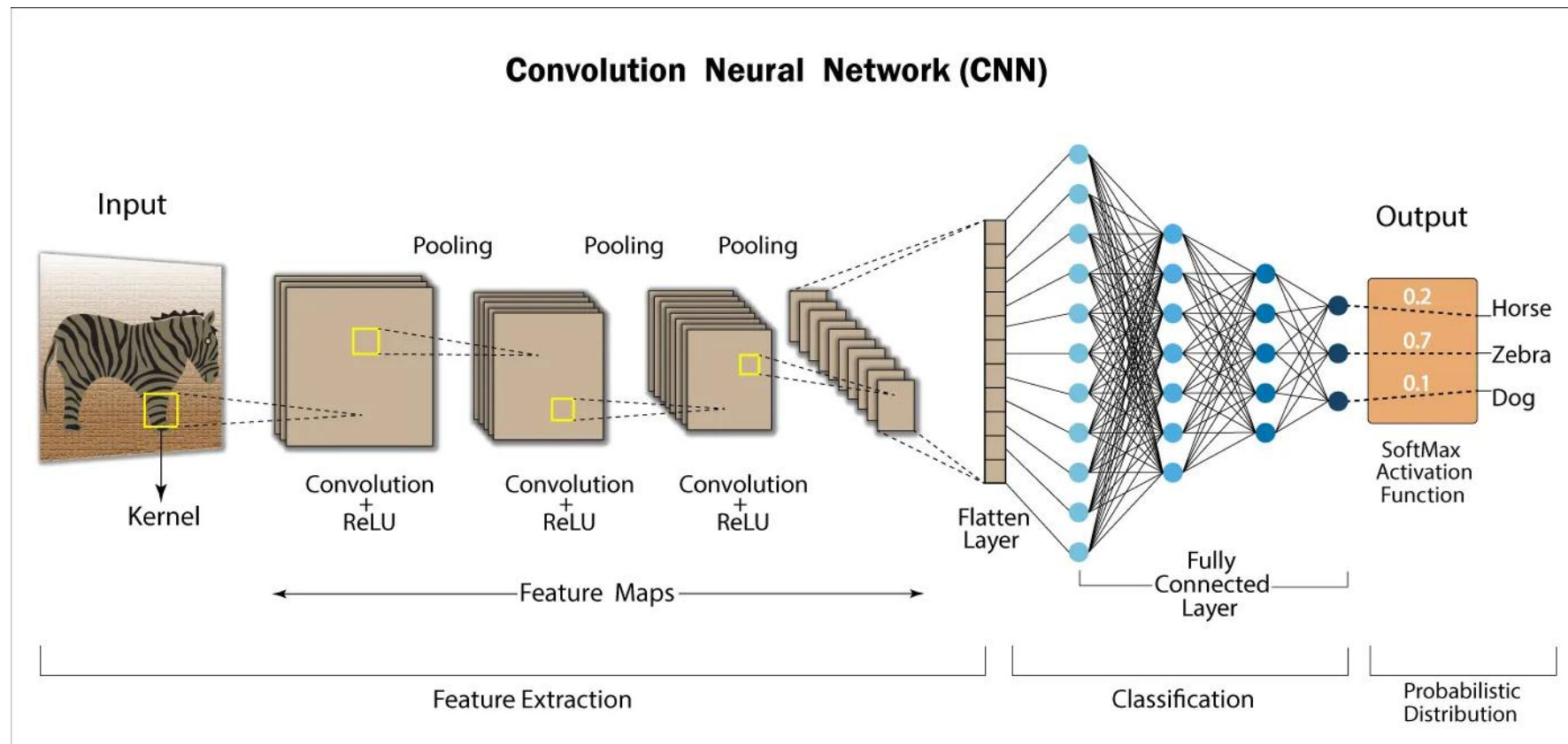
Gaussian Blur

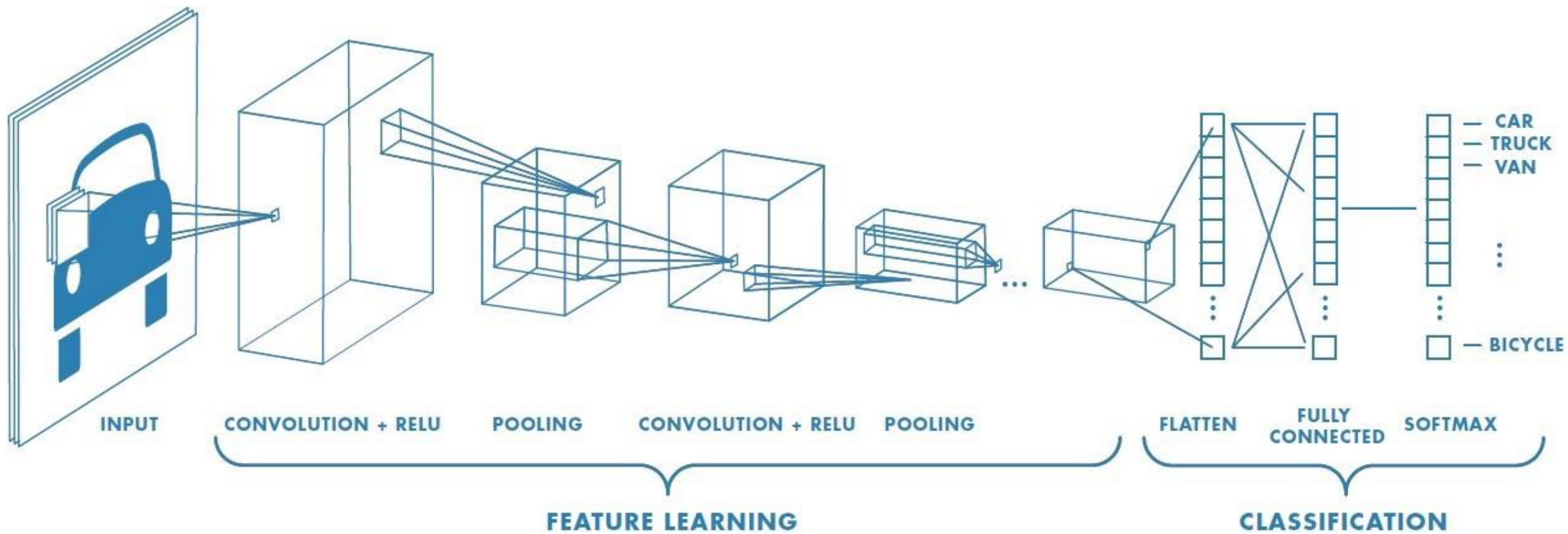


Sharpened image

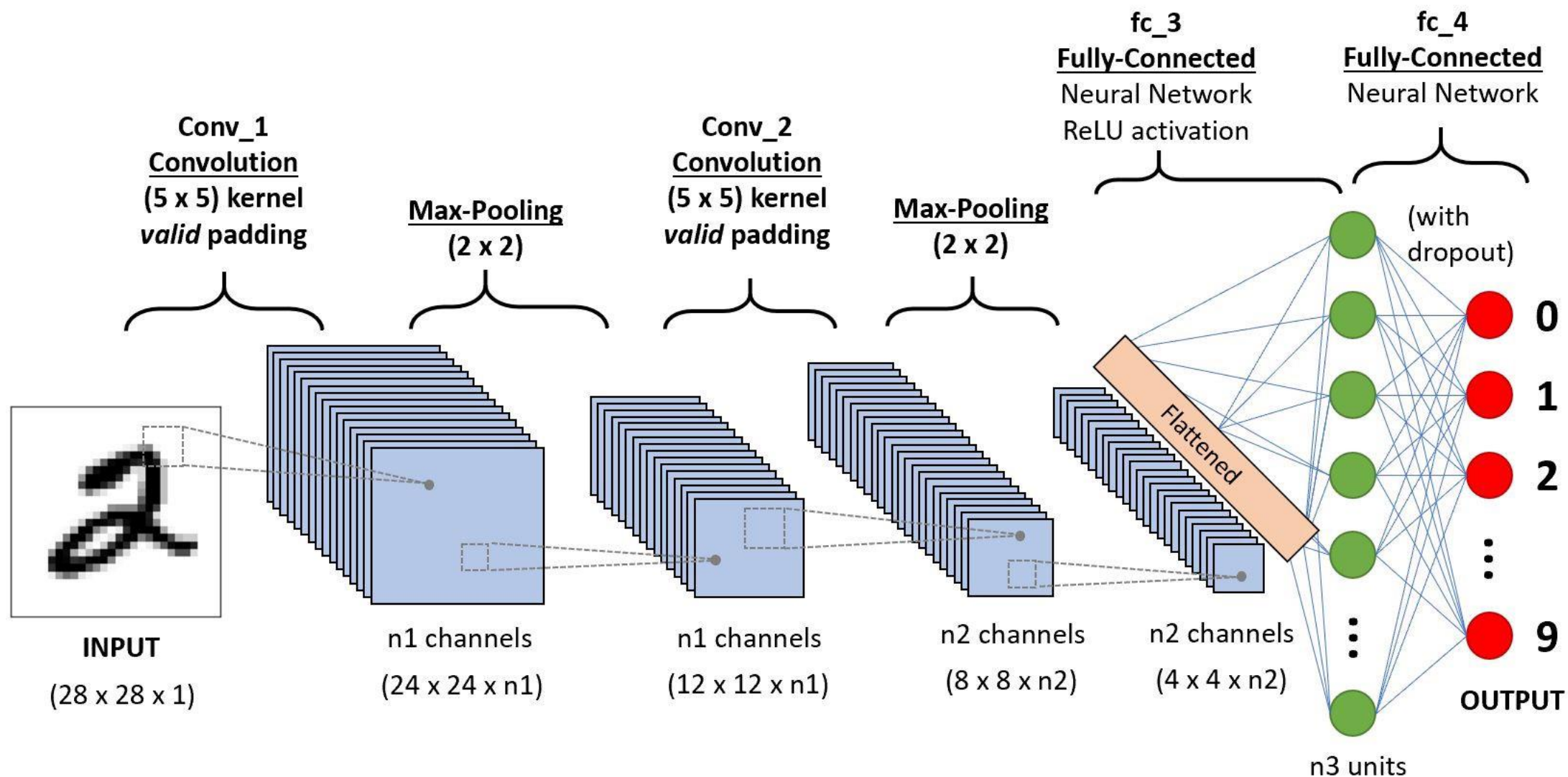
Penggunaan Konvolusi di dalam CNN

- *Convolutional Neural Network* atau CNN merupakan salah satu metode *deep learning* yang umum digunakan untuk pengenalan citra (*image recognition*).



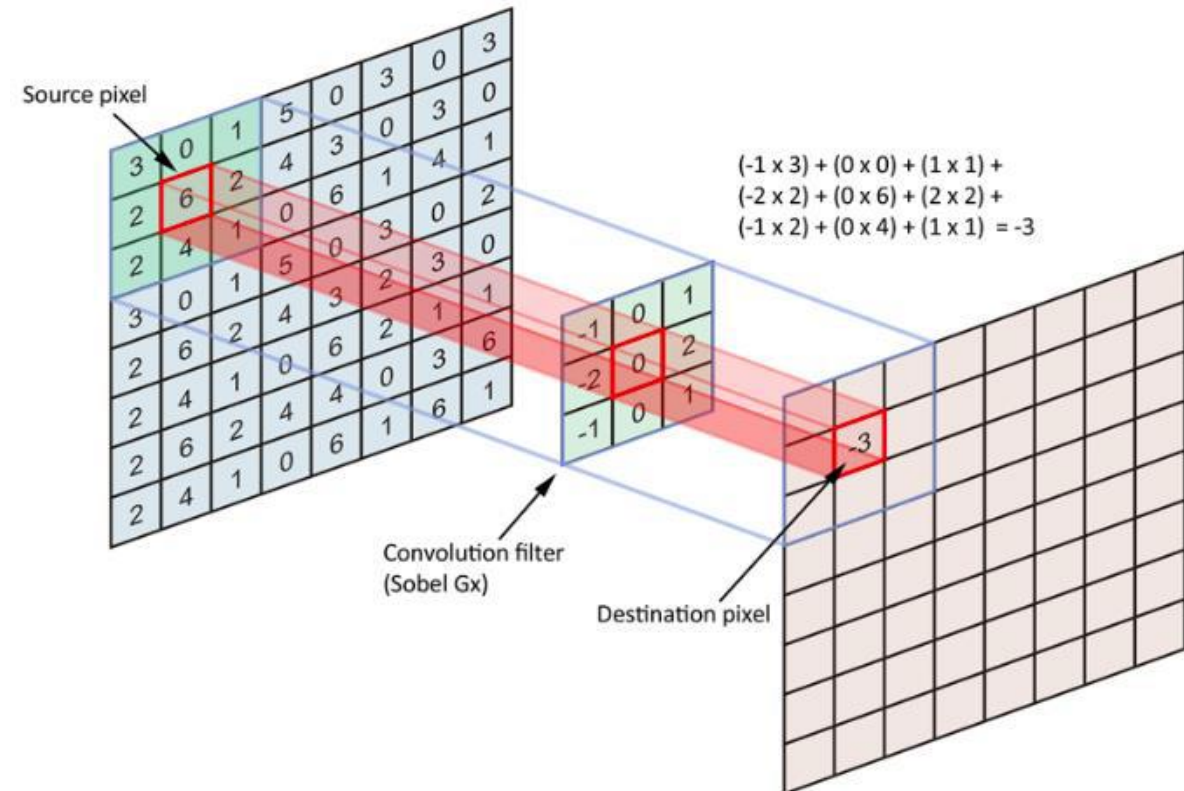


Sumber gambar: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>



Sumber gambar: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>

- *CNN* terdiri dari 3 lapisan utama:
 1. *Convolutional Layer*,
 2. *Pooling Layer*
 3. *Fully-Connected Layer*.
- Konvolusi citra terjadi pada lapisan *Convolutional Layer*
- Pada lapisan ini citra ditapis dengan sebuah kernel atau filter yang berukuran lebih kecil dari citra.
- Konvolusi citra dengan kernel menghasilkan matriks fitur (*feature map*)



Konvolusi pada setiap kanal warna di dalam citra RGB, lalu hasilnya dijumlahkan:

0	0	0	0	0	0	...
0	156	155	156	158	158	...
0	153	154	157	159	159	...
0	149	151	155	158	159	...
0	146	146	149	153	158	...
0	145	143	143	148	158	...
...

Input Channel #1 (Red)

0	0	0	0	0	0	...
0	167	166	167	169	169	...
0	164	165	168	170	170	...
0	160	162	166	169	170	...
0	156	156	159	163	168	...
0	155	153	153	158	168	...
...

Input Channel #2 (Green)

0	0	0	0	0	0	...
0	163	162	163	165	165	...
0	160	161	164	166	166	...
0	156	158	162	165	166	...
0	155	155	158	162	167	...
0	154	152	152	157	167	...
...

Input Channel #3 (Blue)

-1	-1	1
0	1	-1
0	1	1

Kernel Channel #1



308

1	0	0
1	-1	-1
1	0	-1

Kernel Channel #2



-498

0	1	1
0	1	0
1	-1	1

Kernel Channel #3



164

+

+

+ 1 = -25

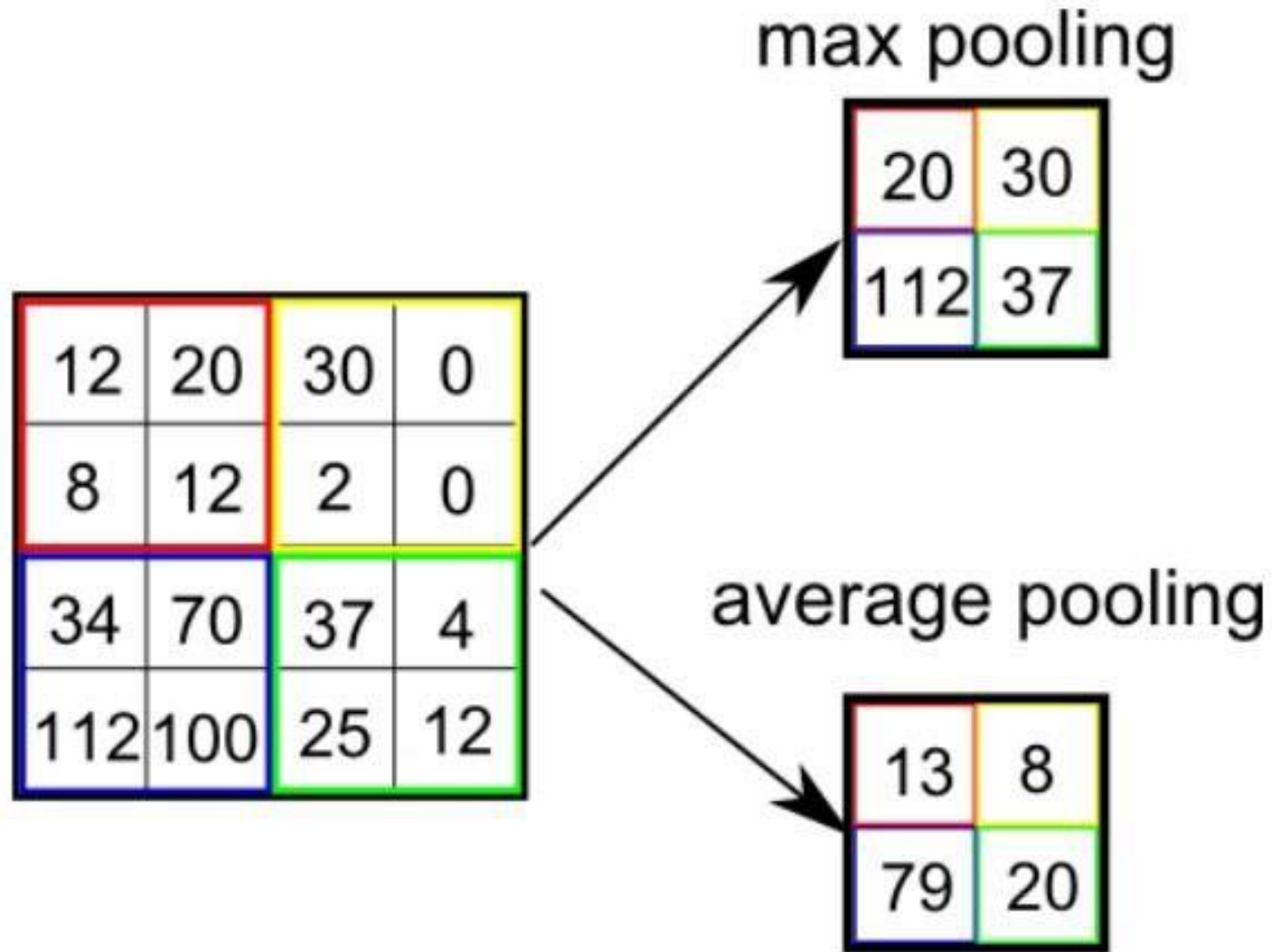
Bias = 1

Output

-25				...
				...
				...
				...
...

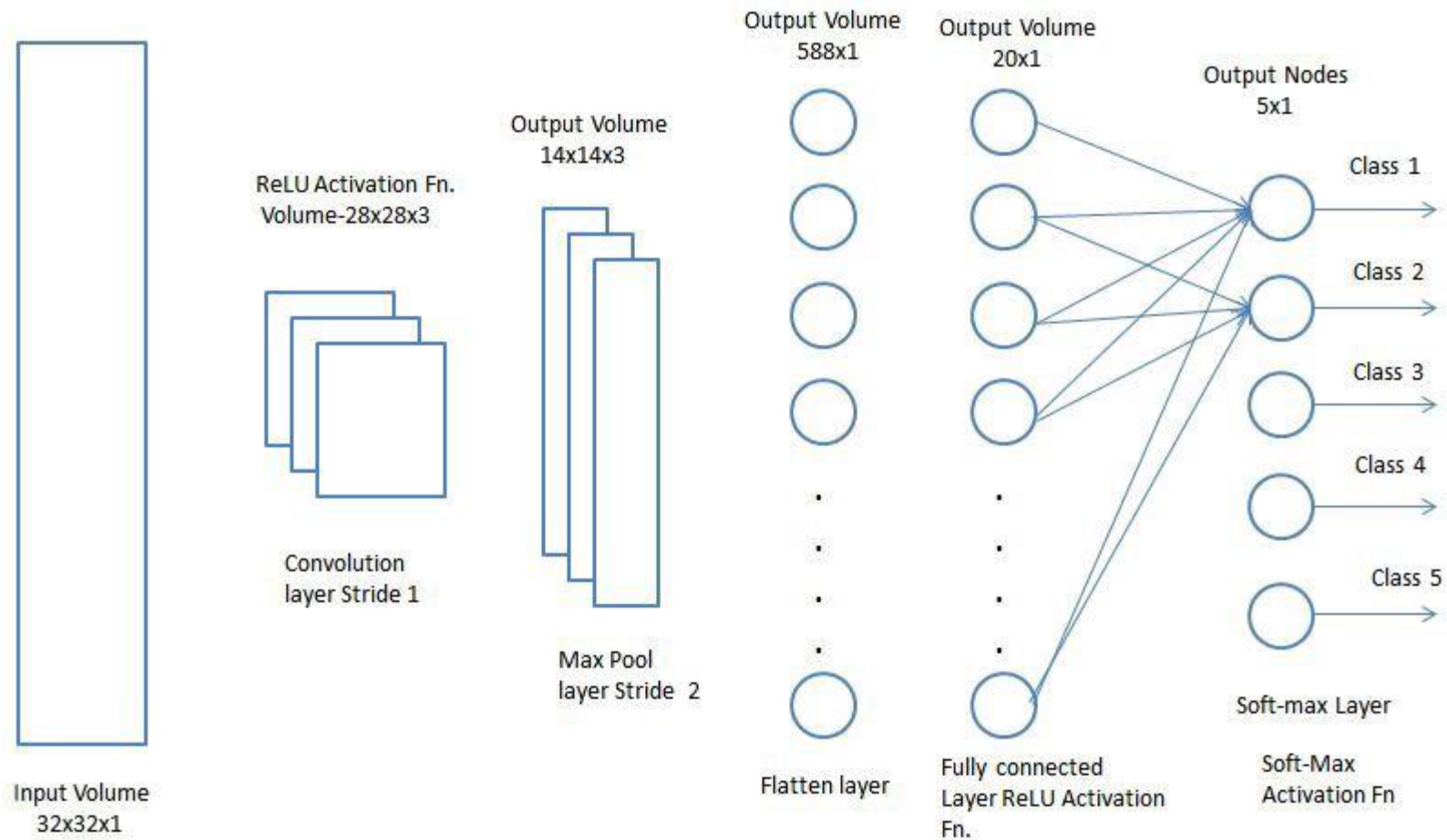
Pooling Layer

- Mirip dengan *Convolutional Layer*, *Pooling layer* bertanggung jawab untuk mengurangi ukuran spasial dari matriks fitur hasil konvolusi.
- Hal ini bertujuan untuk mengurangi daya komputasi yang diperlukan untuk memproses data melalui pengurangan dimensi
- Ada dua jenis *pooling*: *Max Pooling* dan *Average Pooling*.
- *Max Pooling* mengembalikan nilai maksimum dari bagian gambar yang dicakup oleh kernel.
- *Average Pooling* mengembalikan rata-rata semua nilai dari bagian gambar yang dicakup oleh Kernel.



Sumber gambar: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>

Classification - Fully Connected Layer



Beberapa arsitektur CNN:

1. LeNet

2. AlexNet

3. VGGNet

4. GoogLeNet

5. ResNet

6. ZFNet