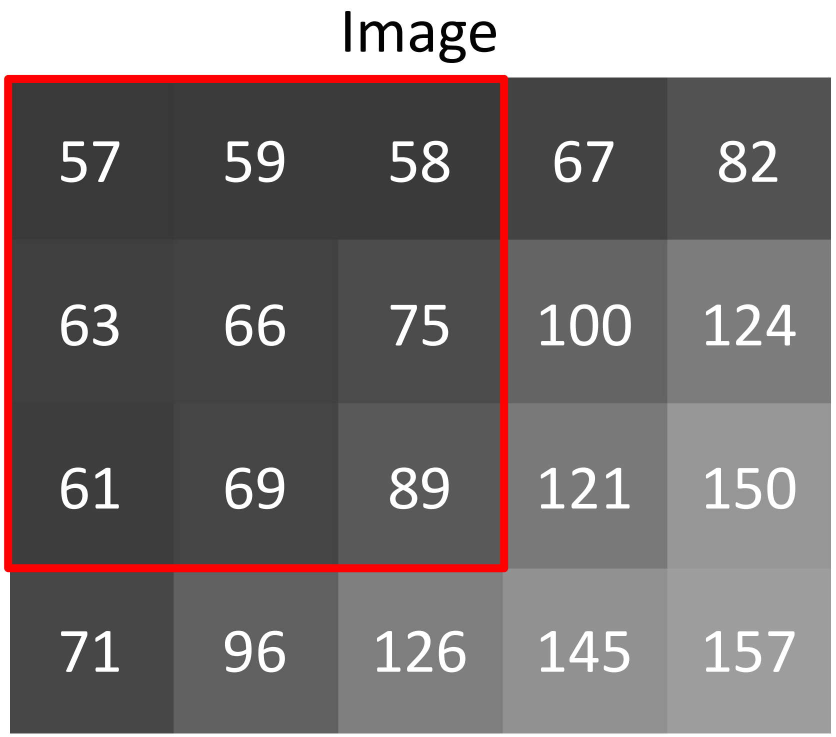
Computer Vision with Embedded Machine Learning

Image Convolution and Filtering

Image



57	59	58	67	82
63	66	75	100	124
61	69	89	121	150
71	96	126	145	157

Kernel

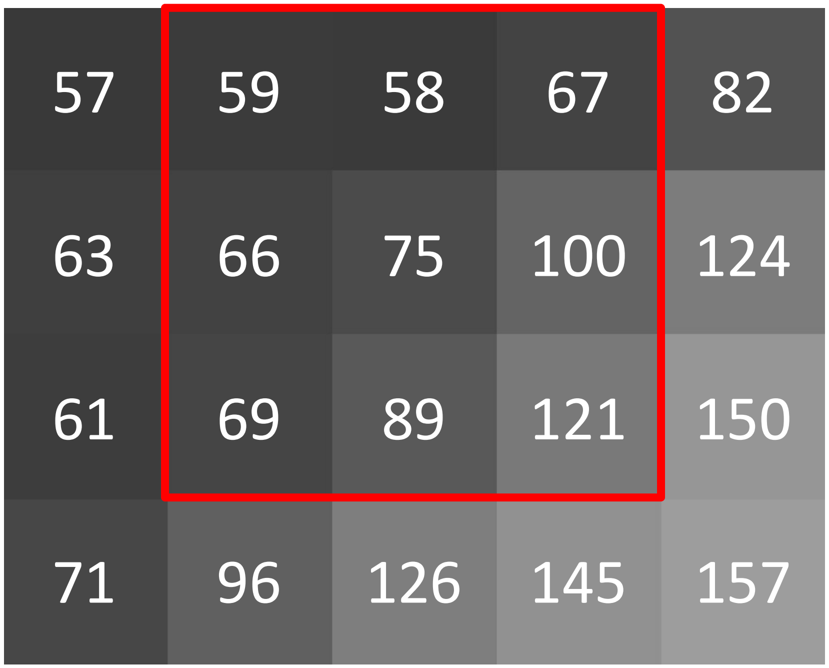
0	-1	0
-1	5	-1
0	-1	0

$$(57 \cdot 0) + (59 \cdot -1) + (58 \cdot 0) +$$
$$(63 \cdot -1) + (66 \cdot 5) + (75 \cdot -1) +$$
$$(61 \cdot 0) + (69 \cdot -1) + (89 \cdot 0) = 64$$

Output

64		

Image



57	59	58	67	82
63	66	75	100	124
61	69	89	121	150
71	96	126	145	157

Kernel

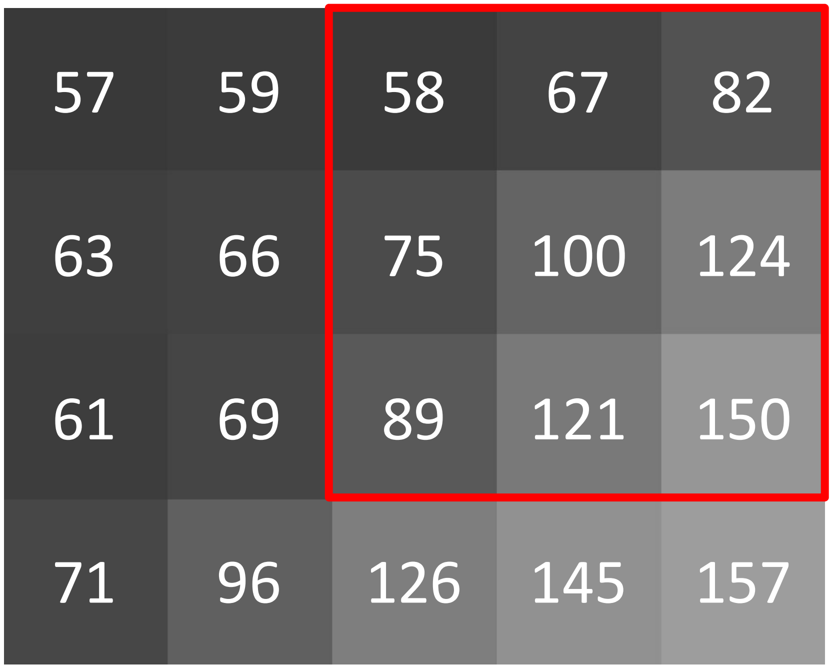
0	-1	0
-1	5	-1
0	-1	0

$$(59 \cdot 0) + (58 \cdot -1) + (67 \cdot 0) + \\ (66 \cdot -1) + (75 \cdot 5) + (100 \cdot -1) + \\ (69 \cdot 0) + (89 \cdot -1) + (121 \cdot 0) = 62$$

Output

64	62	

Image



57	59	58	67	82
63	66	75	100	124
61	69	89	121	150
71	96	126	145	157

Kernel

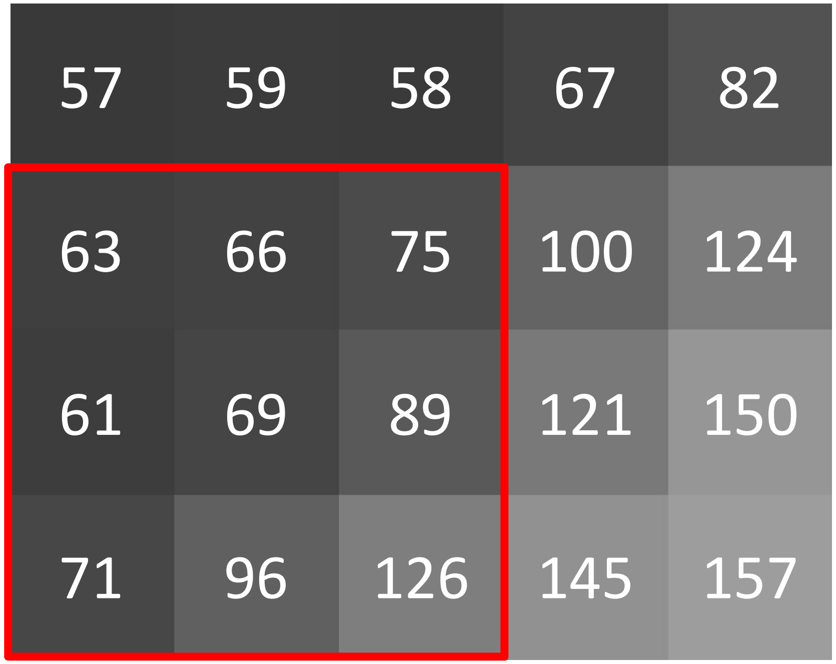
0	-1	0
-1	5	-1
0	-1	0

$$\begin{aligned}&(58 \cdot 0) + (67 \cdot -1) + (82 \cdot 0) + \\ &(75 \cdot -1) + (100 \cdot 5) + (124 \cdot -1) + \\ &(89 \cdot 0) + (121 \cdot -1) + (150 \cdot 0) = 113\end{aligned}$$

Output

64	62	113

Image



57	59	58	67	82
63	66	75	100	124
61	69	89	121	150
71	96	126	145	157

Kernel

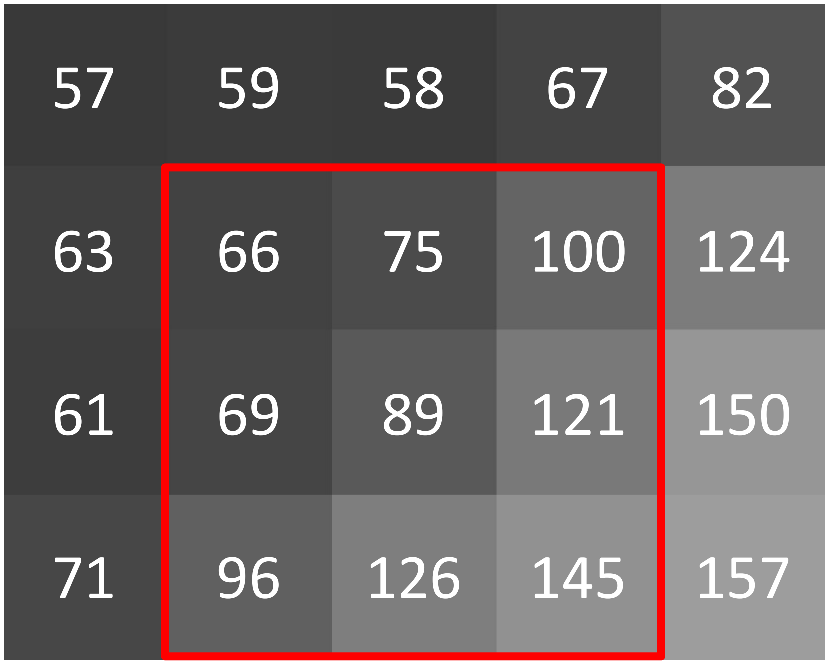
0	-1	0
-1	5	-1
0	-1	0

$$(63 \cdot 0) + (66 \cdot -1) + (75 \cdot 0) + \\ (61 \cdot -1) + (69 \cdot 5) + (89 \cdot -1) + \\ (71 \cdot 0) + (96 \cdot -1) + (126 \cdot 0) = 33$$

Output

64	62	113
33		

Image



57	59	58	67	82
63	66	75	100	124
61	69	89	121	150
71	96	126	145	157

Kernel

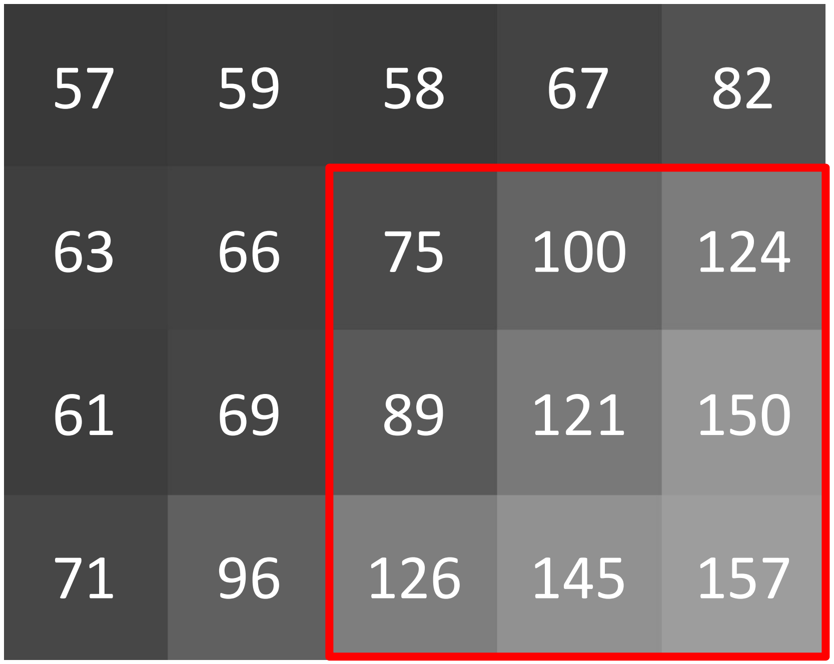
0	-1	0
-1	5	-1
0	-1	0

$$(66 \cdot 0) + (75 \cdot -1) + (100 \cdot 0) + (69 \cdot -1) + (89 \cdot 5) + (121 \cdot -1) + (96 \cdot 0) + (126 \cdot -1) + (145 \cdot 0) = 54$$

Output

64	62	113
33	54	

Image



57	59	58	67	82
63	66	75	100	124
61	69	89	121	150
71	96	126	145	157

stride = 1

Kernel

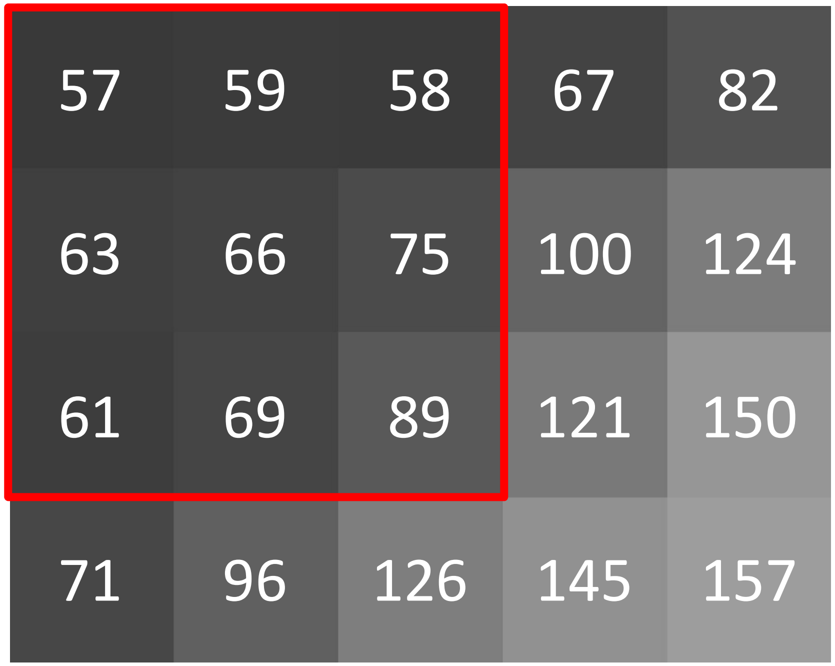
0	-1	0
-1	5	-1
0	-1	0

$$(75 \cdot 0) + (100 \cdot -1) + (124 \cdot 0) + \\ (89 \cdot -1) + (121 \cdot 5) + (150 \cdot -1) + \\ (126 \cdot 0) + (145 \cdot -1) + (157 \cdot 0) = 121$$

Output

64	62	113
33	54	121

Image



57	59	58	67	82
63	66	75	100	124
61	69	89	121	150
71	96	126	145	157

stride = 2

Kernel

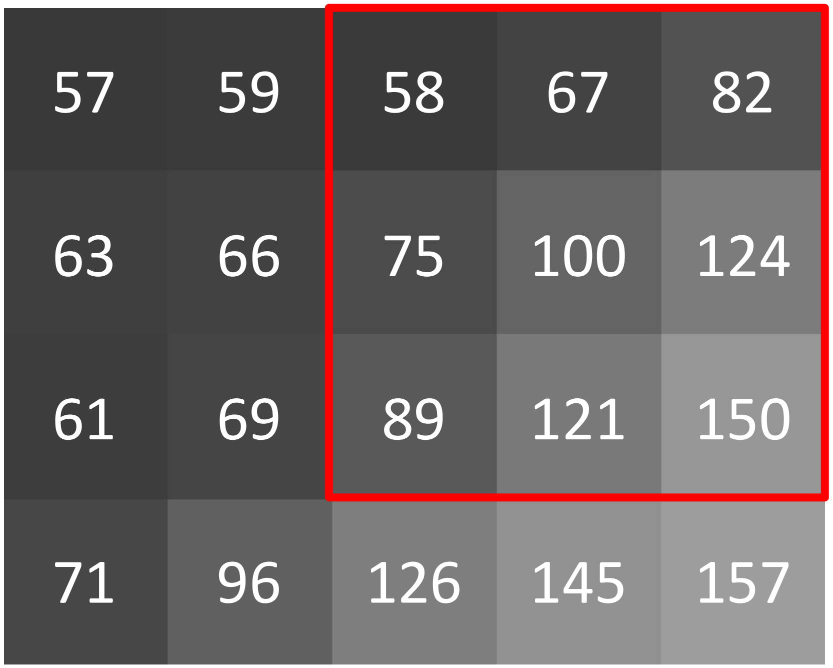
0	-1	0
-1	5	-1
0	-1	0

$$(57 \cdot 0) + (59 \cdot -1) + (58 \cdot 0) + (63 \cdot -1) + (66 \cdot 5) + (75 \cdot -1) + (61 \cdot 0) + (69 \cdot -1) + (89 \cdot 0) = 64$$

Output

64	

Image



57	59	58	67	82
63	66	75	100	124
61	69	89	121	150
71	96	126	145	157

stride = 2

Kernel

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

$$(58 \cdot 0) + (67 \cdot -1) + (82 \cdot 0) +$$
$$(75 \cdot -1) + (100 \cdot 5) + (124 \cdot -1) +$$
$$(89 \cdot 0) + (121 \cdot -1) + (150 \cdot 0) = 113$$

Output

64	113

$$O\left(i,j\right)=\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}I((s\cdot i)+m,(s\cdot j)+n)\cdot K(m,n)$$

s: Stride

H: Number of rows in image (height)
W: Number of columns in image (width)

M: Number of rows in kernel (height)
N: Number of columns in kernel (width)

i goes from 0 to $\left\lfloor\frac{H-M}{s}\right\rfloor+1$

j goes from 0 to $\left\lfloor\frac{W-N}{s}\right\rfloor+1$

$$O\left(i,j\right) = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} I((s\cdot i)+m,(s\cdot j)+n)\cdot K(m,n)$$

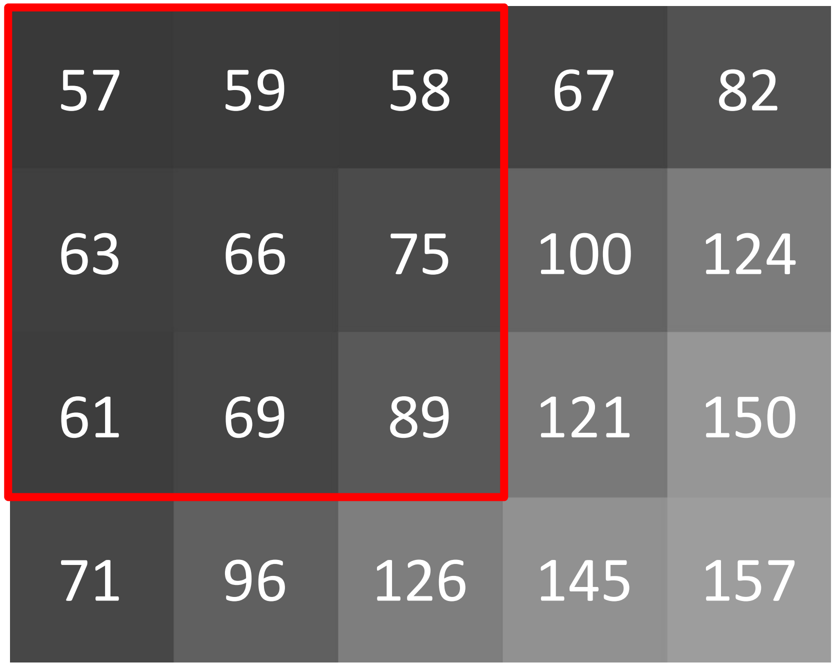
I_{00}	I_{01}	I_{02}	I_{03}	I_{04}
I_{10}	I_{11}	I_{12}	I_{13}	I_{14}
I_{20}	I_{21}	I_{22}	I_{23}	I_{24}
I_{30}	I_{31}	I_{32}	I_{33}	I_{34}

K_{00}	K_{01}	K_{02}
K_{10}	K_{11}	K_{12}
K_{20}	K_{21}	K_{22}

O_{00}	O_{01}	O_{02}
O_{10}	O_{11}	O_{12}

with stride = 1

Image



57	59	58	67	82
63	66	75	100	124
61	69	89	121	150
71	96	126	145	157

8-bit pixel values must be between 0 and 255!

Kernel

1	5	1
5	5	5
1	5	1

$$(57 \cdot 1) + (59 \cdot 5) + (58 \cdot 1) + \\ (63 \cdot 5) + (66 \cdot 5) + (75 \cdot 5) + \\ (61 \cdot 1) + (69 \cdot 5) + (89 \cdot 1) = 1925$$

Output

255		

Image

57	59	58	67	82
63	66	75	100	124
61	69	89	121	150
71	96	126	145	157

8-bit pixel values must be between 0 and 255!

Kernel

1	5	1
5	5	5
1	5	1

Output

255	255	255
255	255	255

Image

0.22	0.23	0.23	0.26	0.32
0.25	0.26	0.29	0.39	0.49
0.24	0.27	0.35	0.47	0.59
0.28	0.38	0.49	0.57	0.62

Floating point pixel values will likely be between 0.0 and 1.0

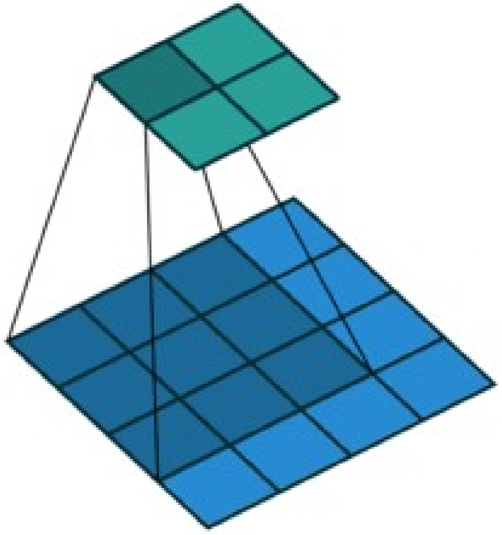
Kernel

1	5	1
5	5	5
1	5	1

Output

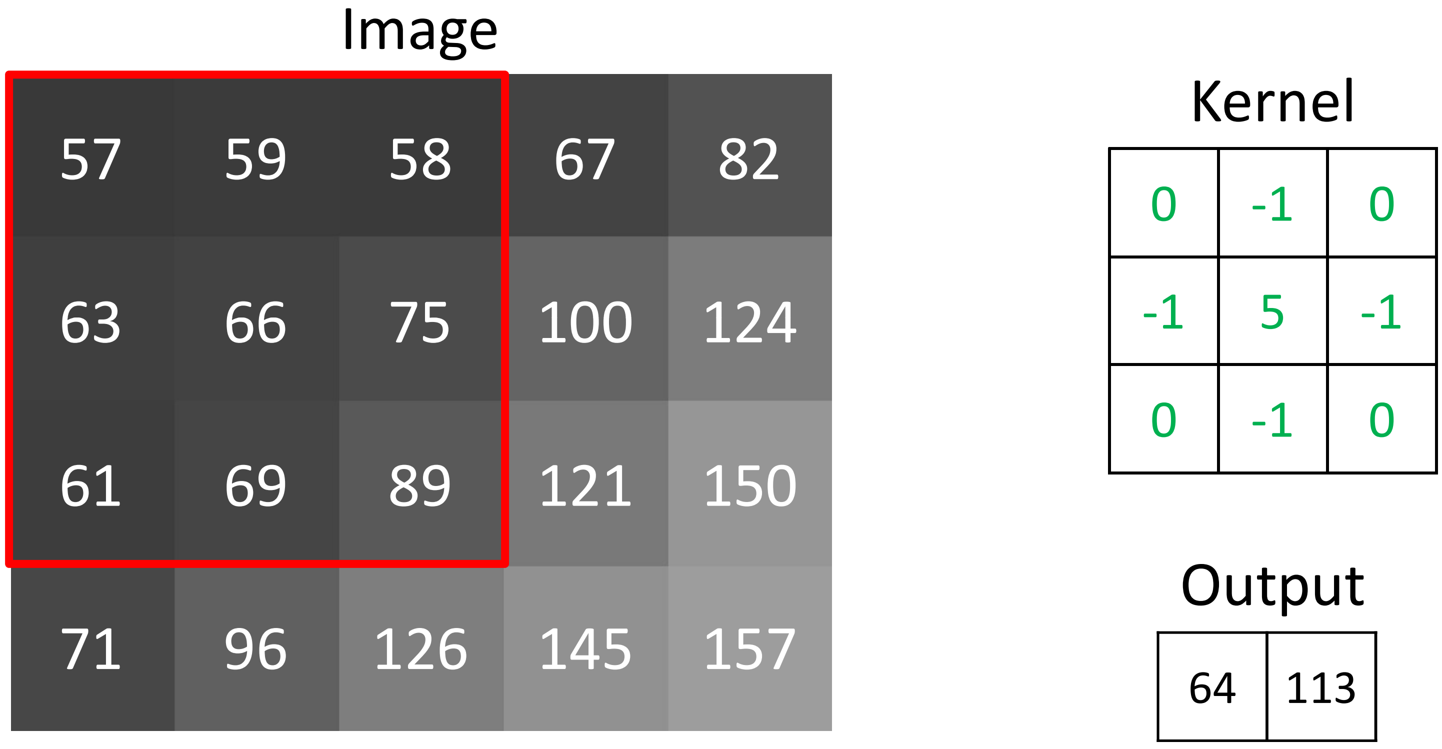
1.0	1.0	1.0
1.0	1.0	1.0

Valid Padding



Animations from github.com/vdumoulin/conv_arithmetic (MIT license)

Valid Padding



Image

57	59	58	67	82
63	66	75	100	124
61	69	89	121	150
71	96	126	145	157

stride = 2

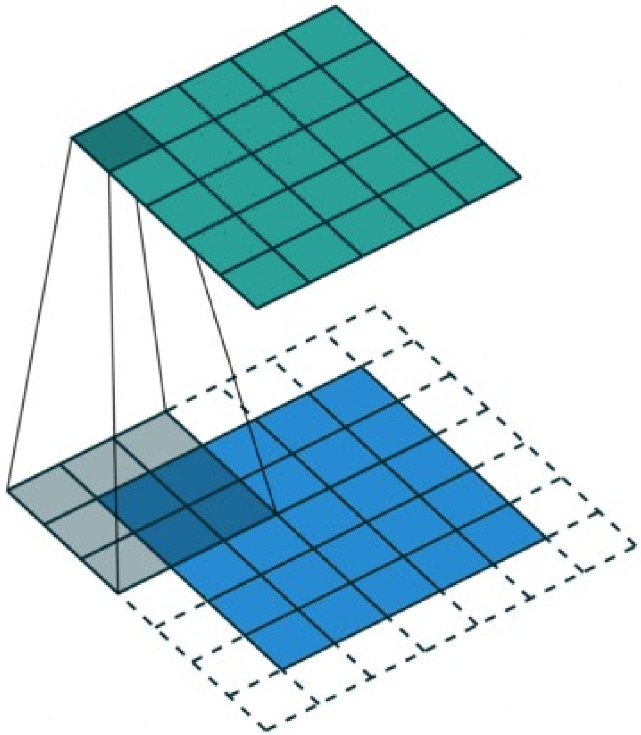
Kernel

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

Output

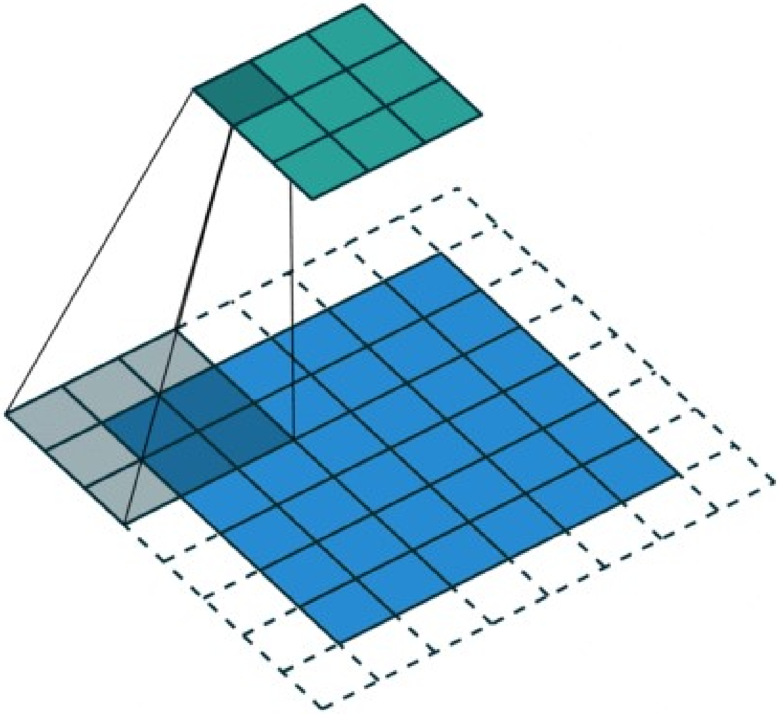
64	113

Same Padding

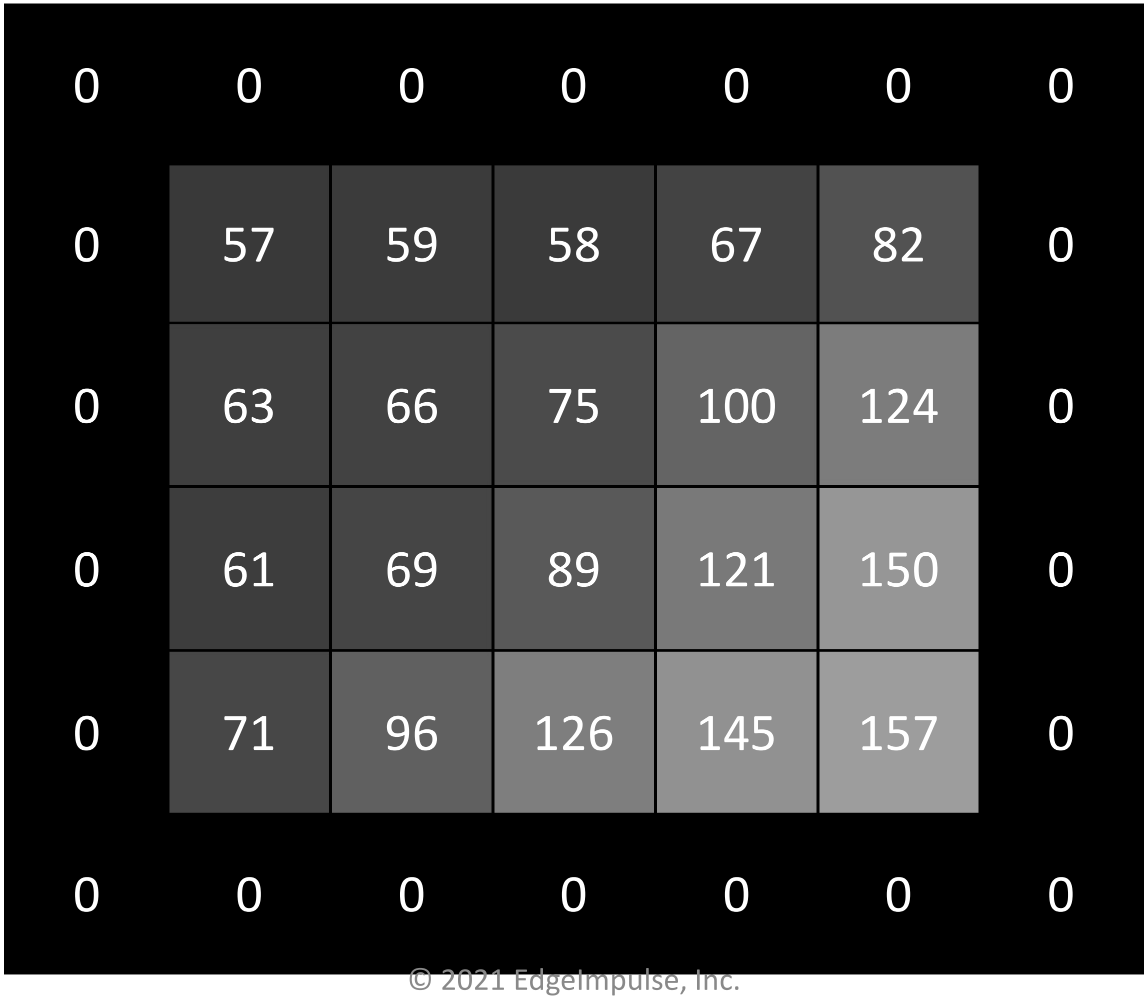


Animations from github.com/vdumoulin/conv_arithmetic (MIT license)

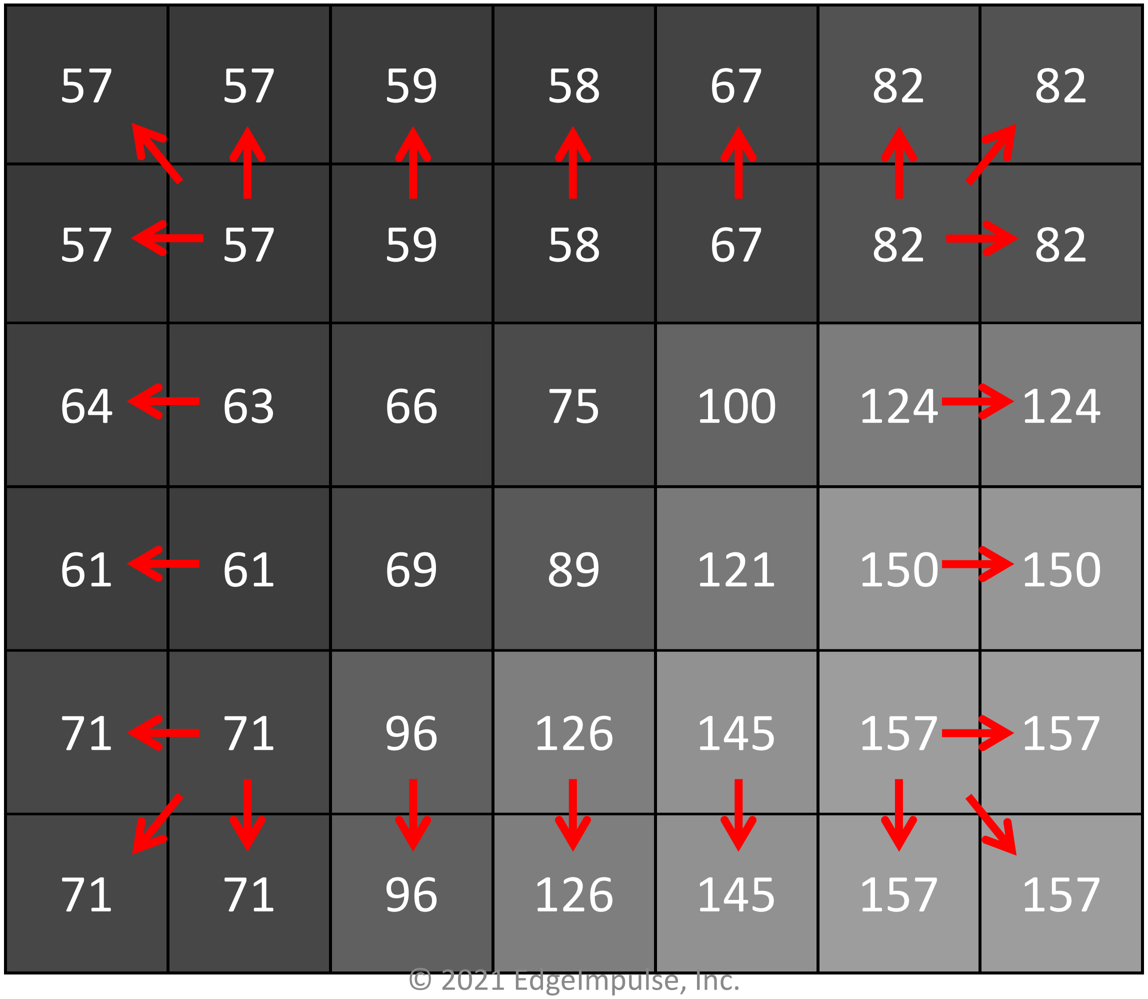
Padding with stride=2



Animations from github.com/vdumoulin/conv_arithmetic (MIT license)



0	0	0	0	0	0	0
0	57	59	58	67	82	0
0	63	66	75	100	124	0
0	61	69	89	121	150	0
0	71	96	126	145	157	0
0	0	0	0	0	0	0



57	57	59	58	67	82	82
57	57	59	58	67	82	82
64	63	66	75	100	124	124
61	61	69	89	121	150	150
71	71	96	126	145	157	157
71	71	96	126	145	157	157

Image (without padding)

57	59	58	67	82
63	66	75	100	124
61	69	89	121	150
71	96	126	145	157

stride = 1

Kernel

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

Without padding:
- Smaller output
- Information on borders is lost

Output

64	62	113
33	54	121

Input Image (with padding)

0	0	0	0	0	0	0
0	57	59	58	67	82	0
0	63	66	75	100	124	0
0	61	69	89	121	150	0
0	71	96	126	145	157	0
0	0	0	0	0	0	0

stride = 1

Kernel

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

With padding:

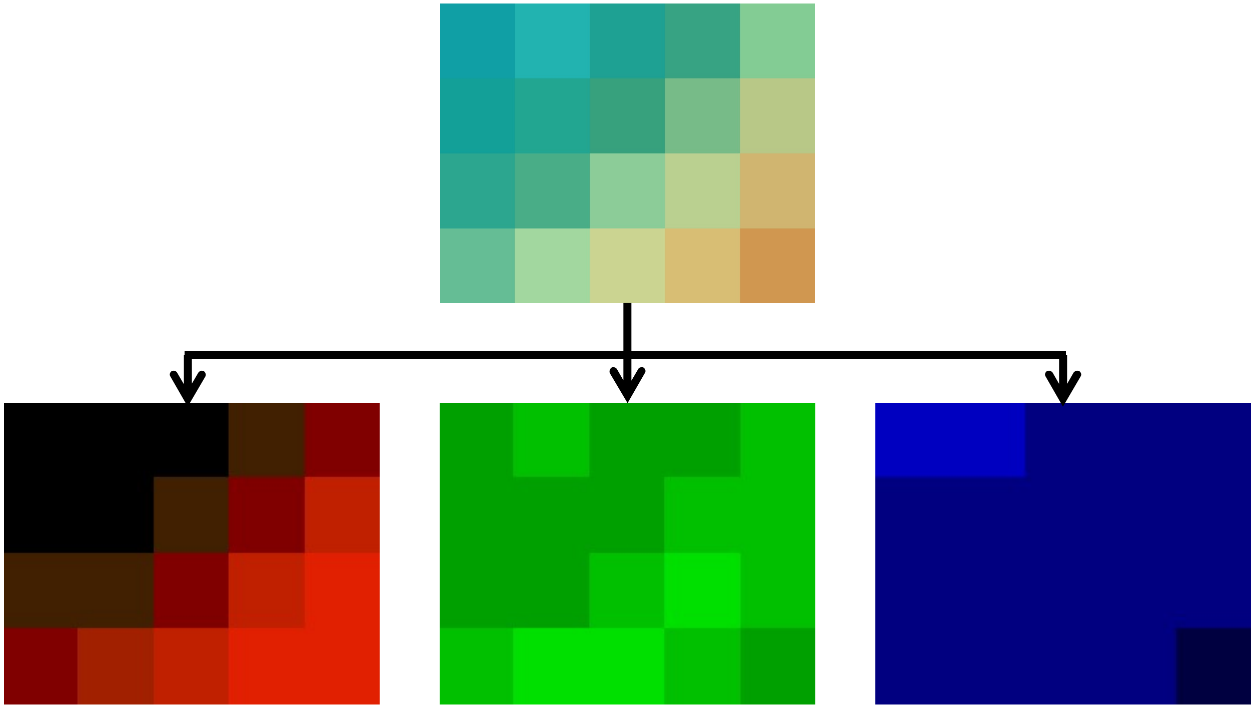
- Larger output matrix
- Information on borders is maintained

Output

163	114	89	95	219
131	64	62	113	255
102	33	54	121	255
198	214	255	255	255

Original image

RGB channels



Kernel

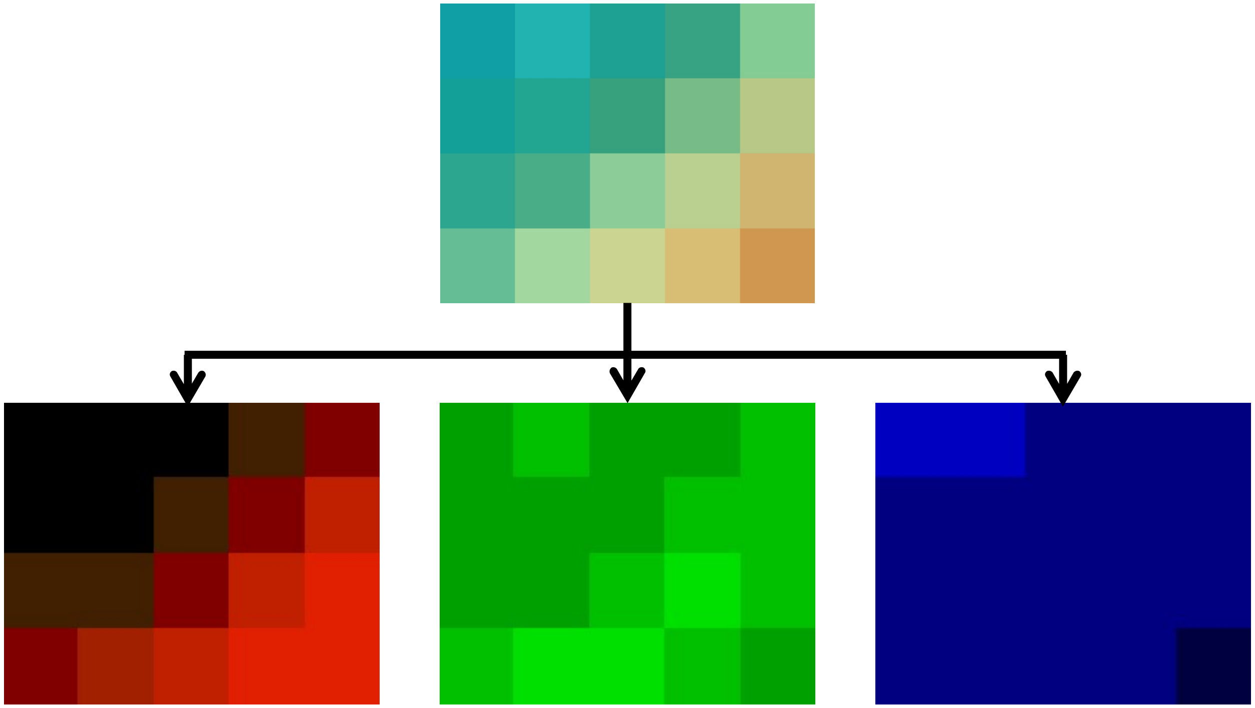
0	-1	0
-1	5	-1
0	-1	0

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

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

Original image

RGB channels



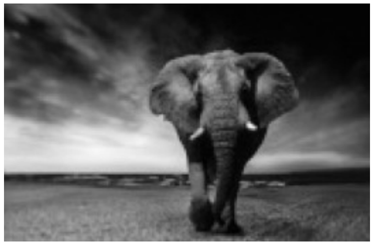
Kernel

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

-1	-2	-1
0	0	0
1	2	1

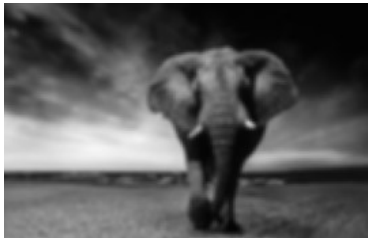
-1	-1	-1
-1	8	-1
-1	-1	-1

Original image (200x130)



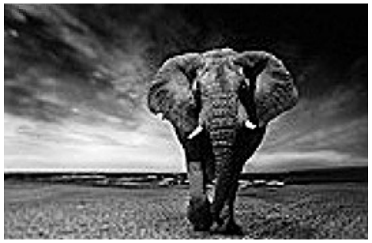
Gaussian blur

1/16	1/8	1/16
1/8	1/4	1/8
1/16	1/8	1/16



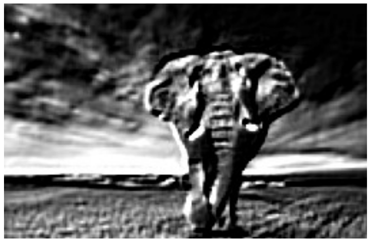
Sharpen

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



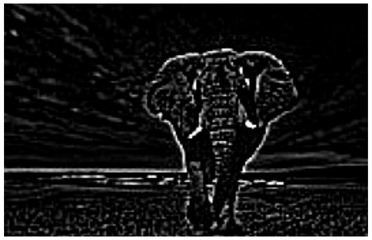
Emboss

-2	-1	0
-1	1	1
0	1	2



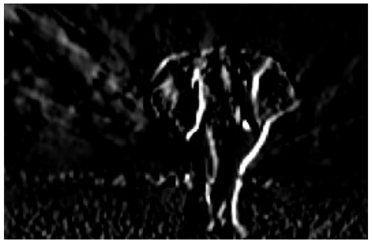
Outline

-1	-1	-1
-1	8	-1
-1	-1	-1



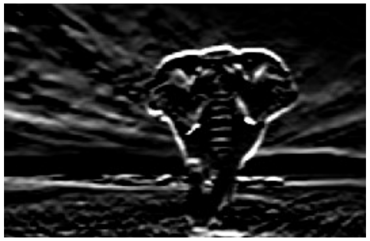
Left Sobel

1	0	-1
2	0	-2
1	0	-1

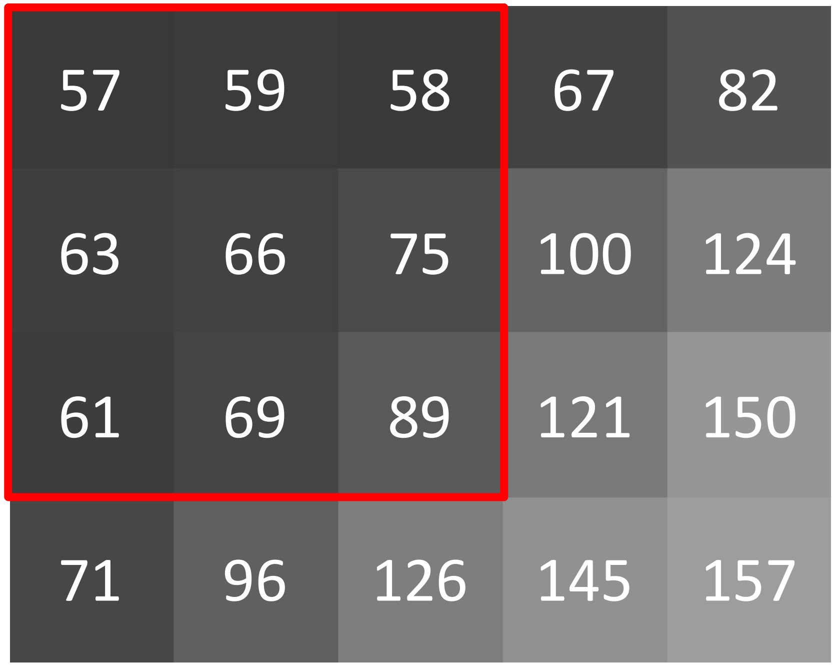


Top Sobel

1	2	1
0	0	0
-1	-2	-1



Image



57	59	58	67	82
63	66	75	100	124
61	69	89	121	150
71	96	126	145	157

8-bit pixel values must be
between 0 and 255!

Kernel

1	5	1
5	5	5
1	5	1

$$(57 \cdot 1) + (59 \cdot 5) + (58 \cdot 1) + \\ (63 \cdot 5) + (66 \cdot 5) + (75 \cdot 5) + \\ (61 \cdot 1) + (69 \cdot 5) + (89 \cdot 1) = 1925$$

Output

255		