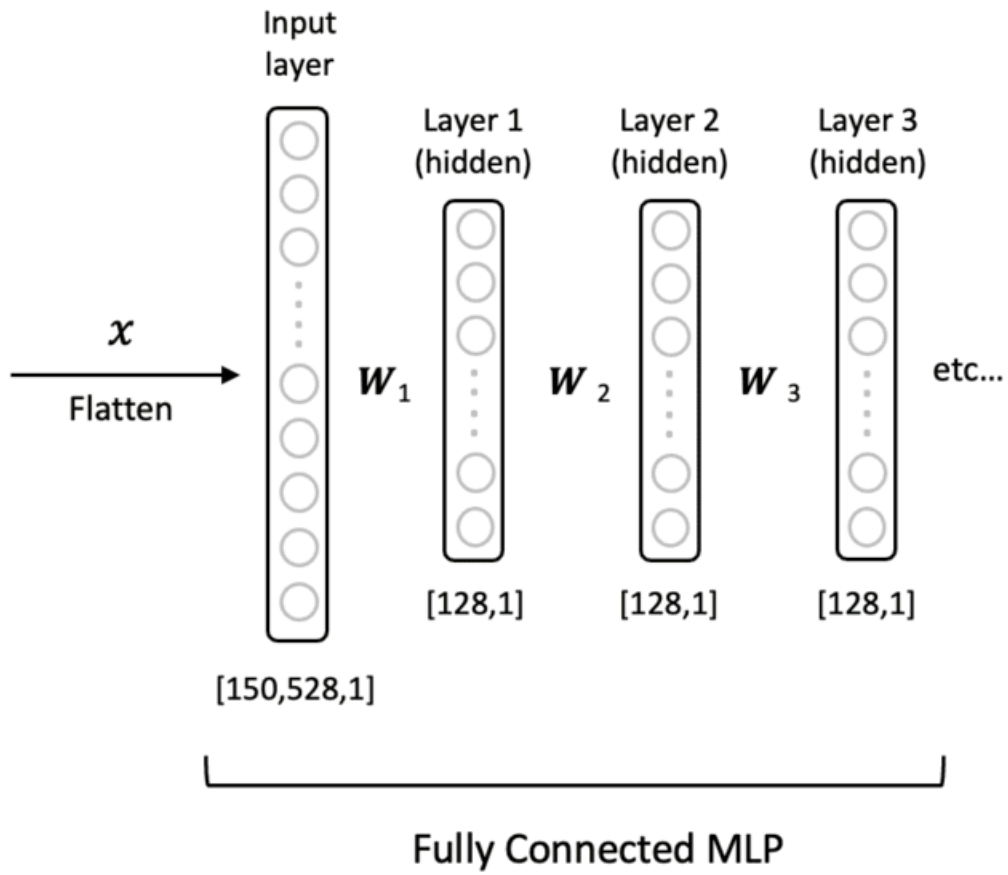
Redes Neurais Convolucionais

Input Image

224 x 224 x 3



Number of Trainable Weights

W_1 = 150,528 x 128 = 19,267,584 Weights

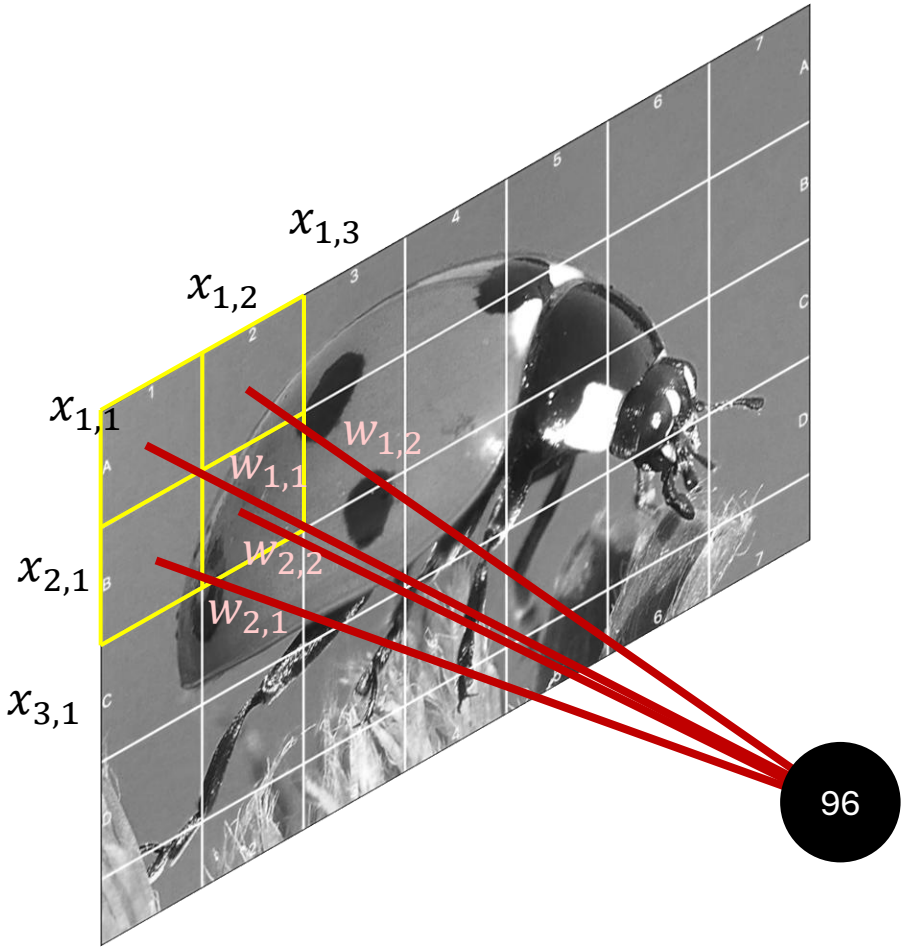
W_2 = 19,267,584 x 128 ~ 2.4 Billion

W_3 = 2.4B x 128 > 300 Billion

Muitos parâmetros =
Maior chance de *overfitting*
e alto custo computacional
(principalmente memória)

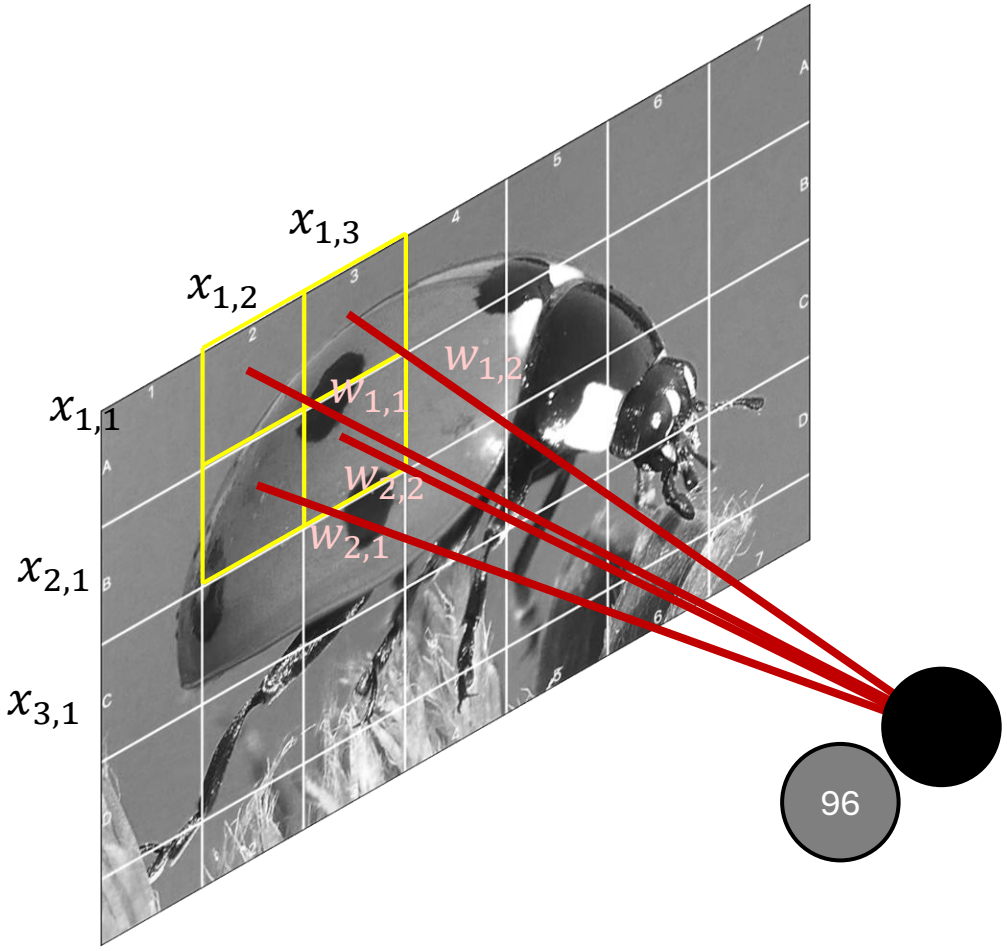
Translation Invariance

Localidade



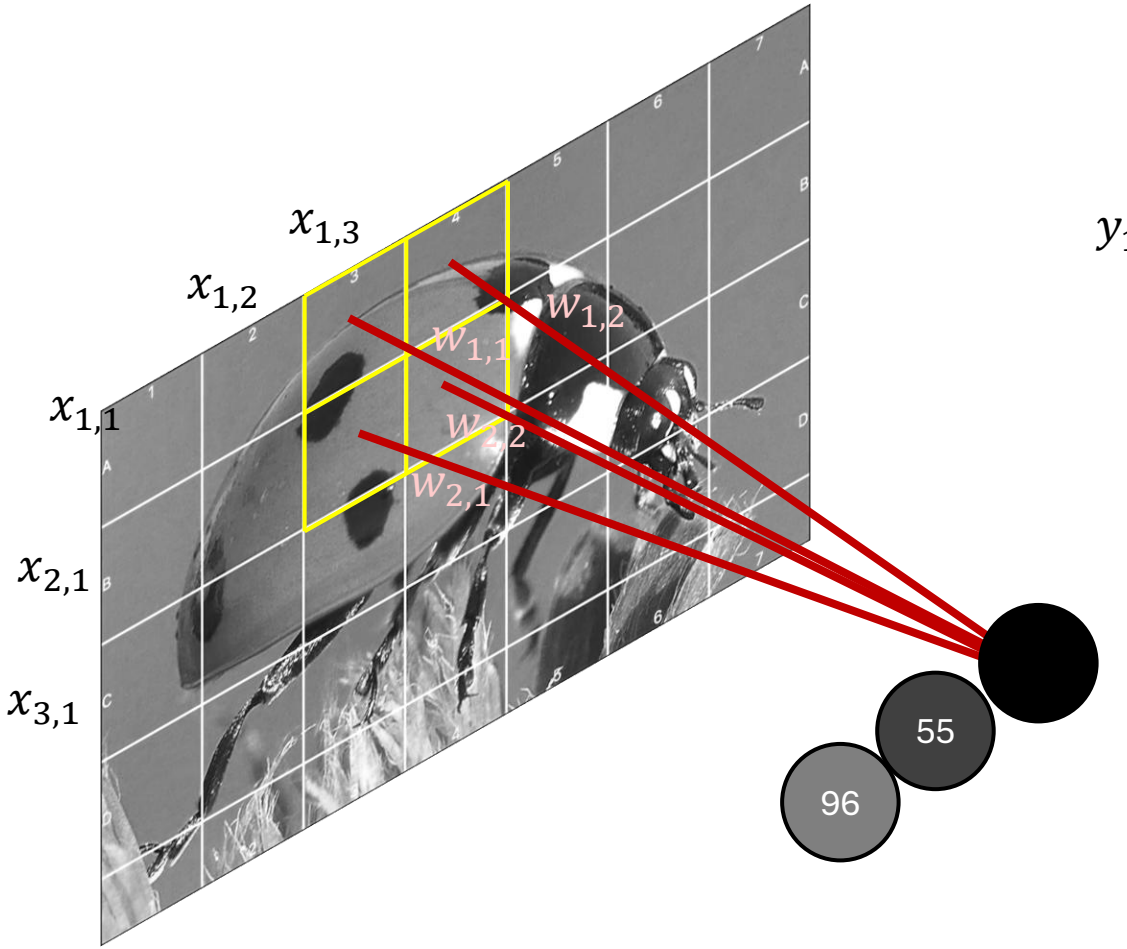
$$y_{1,1} = \text{ReLU}\left(b + w_{1,1}x_{1,1} + w_{1,2}x_{1,2} + w_{2,1}x_{2,1} + w_{2,2}x_{2,2}\right)$$

$$y_{1,1} = \text{ReLU}\left(b + \sum\sum x_{i,j}w_{i,j}\right)$$



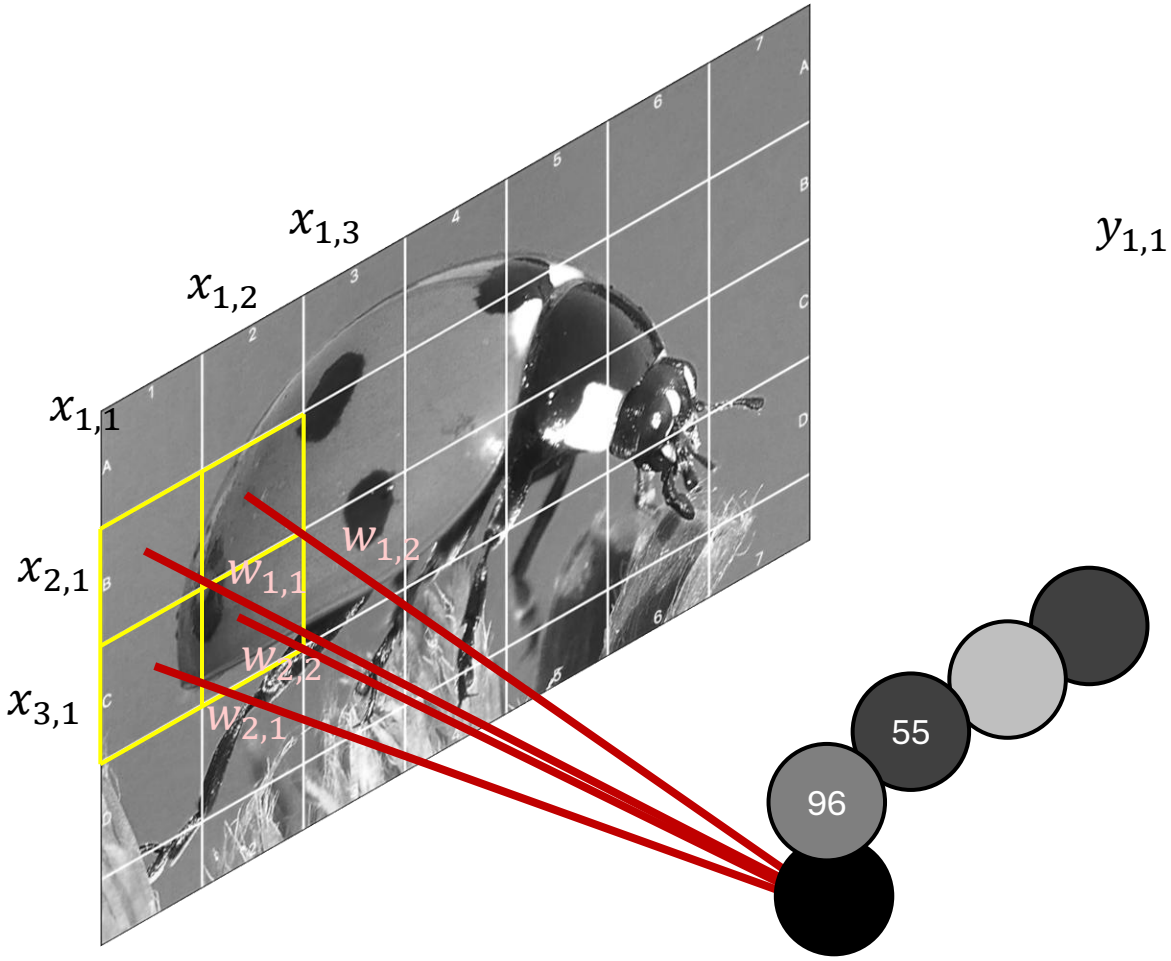
$$y_{1,1} = \text{ReLU}\left(b + w_{1,1}x_{1,1} + w_{1,2}x_{1,2} + w_{2,1}x_{2,1} + w_{2,2}x_{2,2}\right)$$

$$y_{1,1} = \text{ReLU}\left(b + \sum\sum x_{i,j}w_{i,j}\right)$$



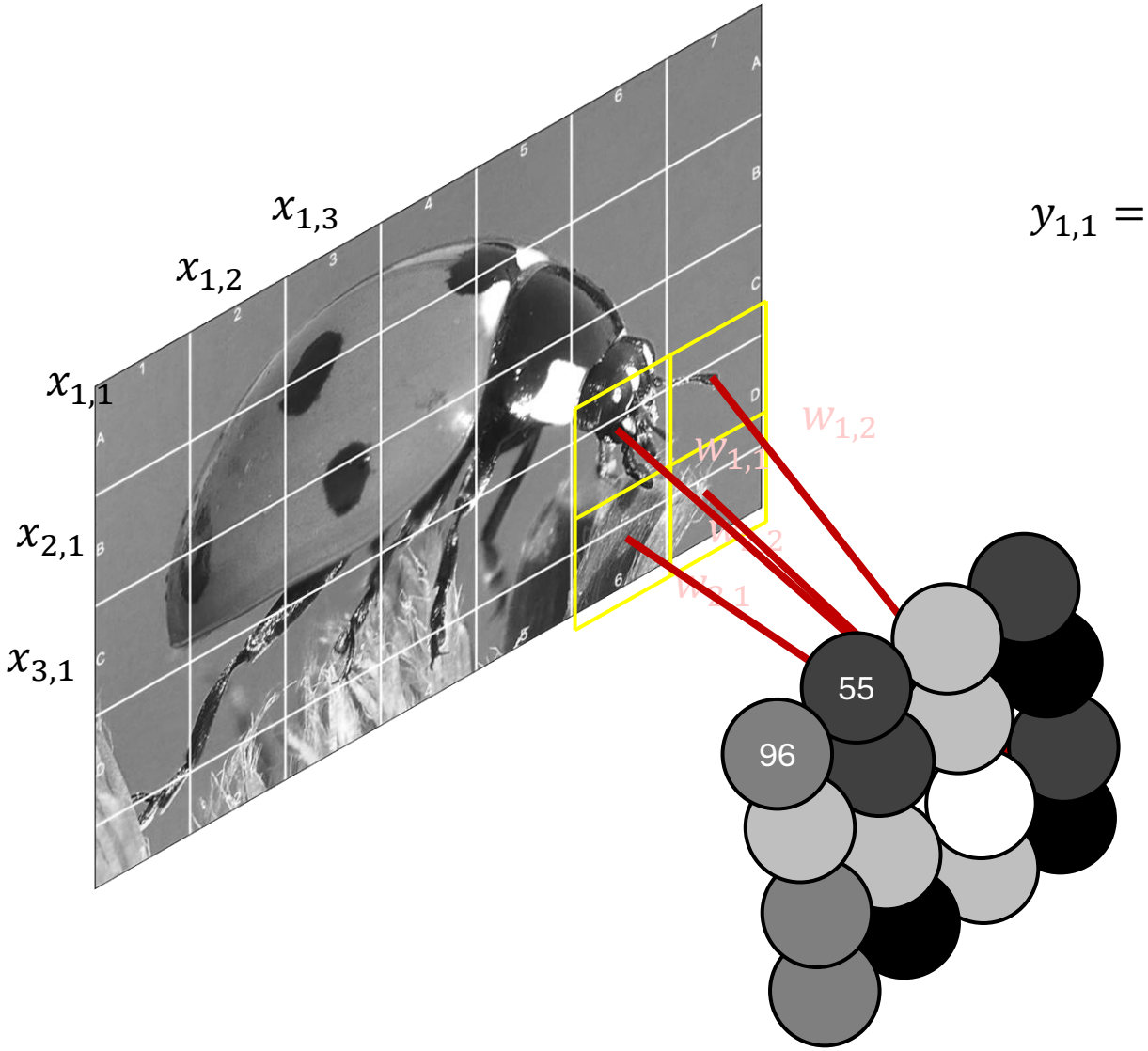
$$y_{1,1} = \text{ReLU}\left(b + w_{1,1}x_{1,1} + w_{1,2}x_{1,2} + w_{2,1}x_{2,1} + w_{2,2}x_{2,2}\right)$$

$$y_{1,1} = \text{ReLU}\left(b + \sum\sum x_{i,j}w_{i,j}\right)$$



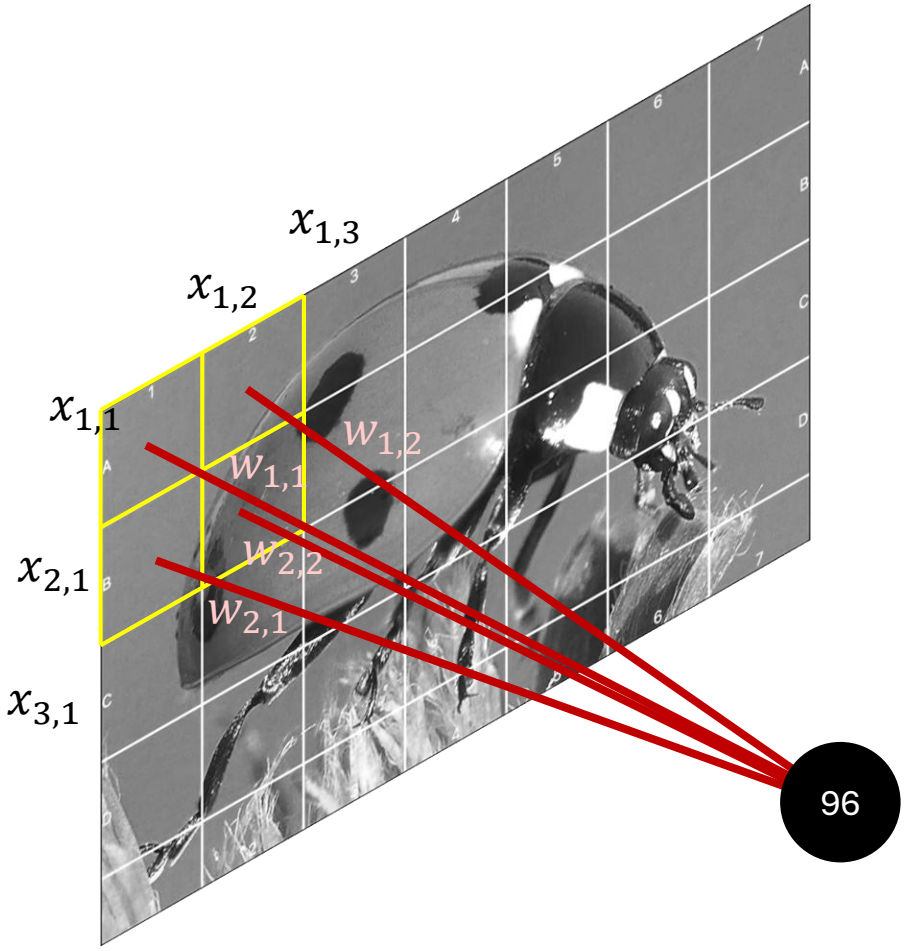
$$y_{1,1} = \text{ReLU}\left(b + w_{1,1}x_{1,1} + w_{1,2}x_{1,2} + w_{2,1}x_{2,1} + w_{2,2}x_{2,2}\right)$$

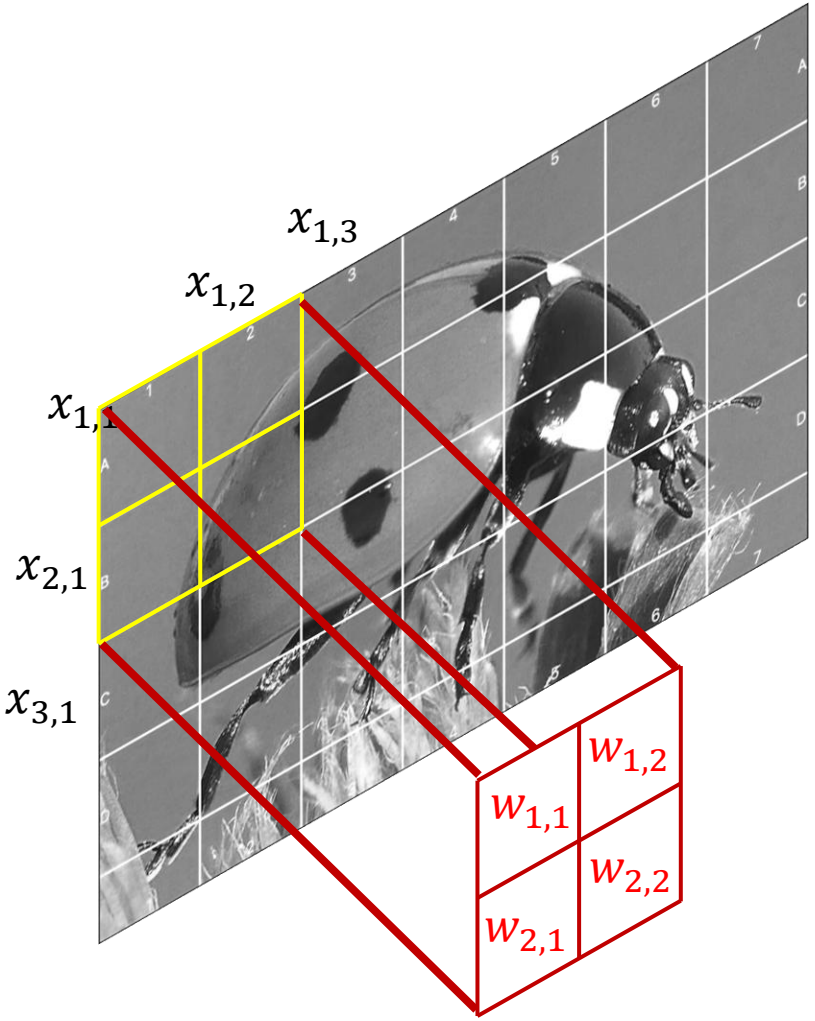
$$y_{1,1} = \text{ReLU}\left(b + \sum\sum x_{i,j}w_{i,j}\right)$$



$$y_{1,1} = \text{ReLU}\left(b + w_{1,1}x_{1,1} + w_{1,2}x_{1,2} + w_{2,1}x_{2,1} + w_{2,2}x_{2,2}\right)$$

$$y_{1,1} = \text{ReLU}\left(b + \sum\sum x_{i,j}w_{i,j}\right)$$

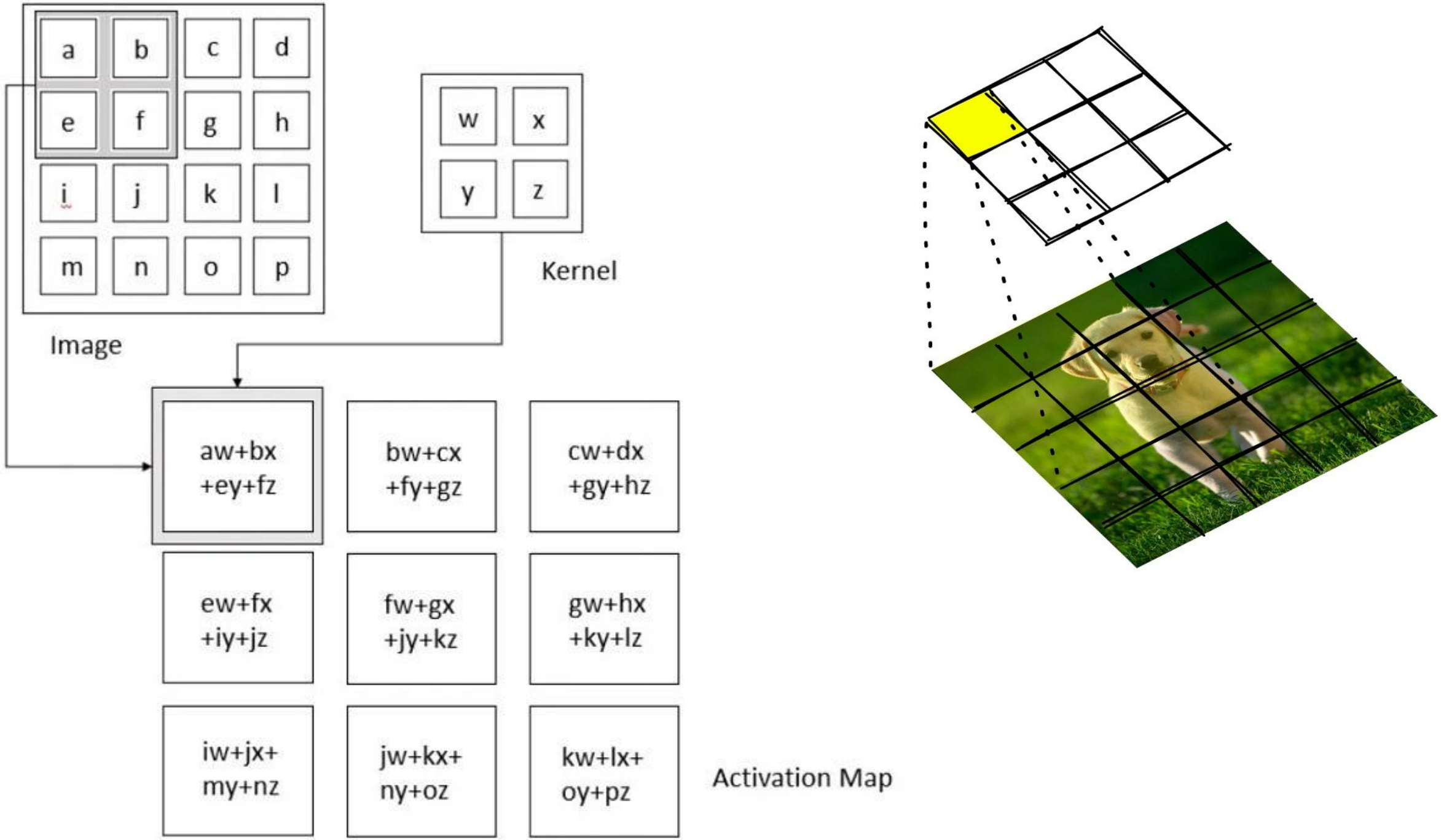


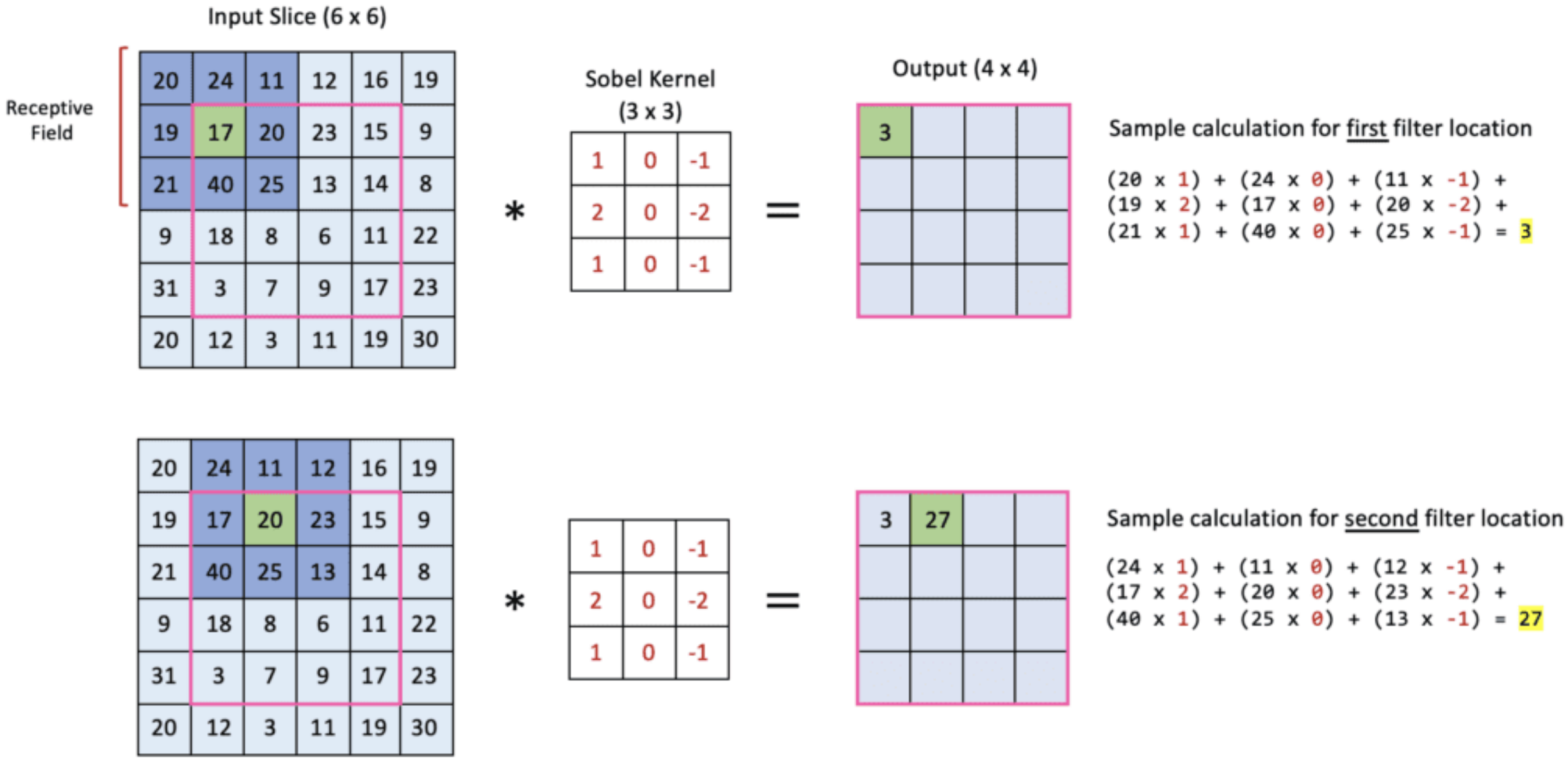


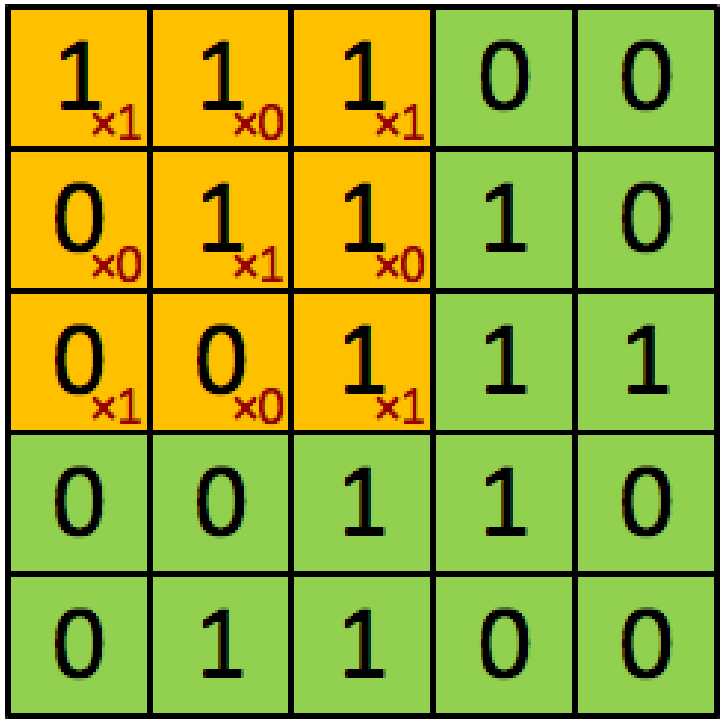
Filtro ou kernel

$$y_{1,1} = \mathrm{ReLU}(b + w_{1,1}x_{1,1} + w_{1,2}x_{1,2} + w_{2,1}x_{2,1} + w_{2,2}x_{2,2})$$

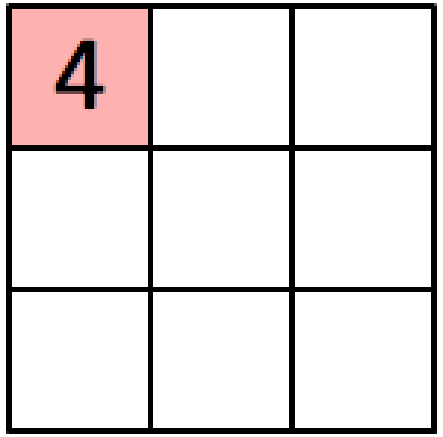
$$y_{1,1} = \mathrm{ReLU}\left(b + \sum\sum x_{i,j}w_{i,j}\right)$$


a b c d
e f g h
i j k l
m n o p
Image
w x
y z
Kernel
aw+bx
+ey+fz
bw+cx
+fy+gz
cw+dx
+gy+hz
ew+fx
+iy+jz
fw+gx
+jy+kz
gw+hx
+ky+lz
iw+jx+
my+nz
jw+kx+
ny+oz
kw+lx+
oy+pz
Activation Map


Input Slice (6 x 6)
Receptive Field
20 24 11 12 16 19
19 17 20 23 15 9
21 40 25 13 14 8
9 18 8 6 11 22
31 3 7 9 17 23
20 12 3 11 19 30
*
Sobel Kernel (3 x 3)
1 0 -1
2 0 -2
1 0 -1
=
Output (4 x 4)
3
Sample calculation for first filter location
(20 x 1) + (24 x 0) + (11 x -1) +
(19 x 2) + (17 x 0) + (20 x -2) +
(21 x 1) + (40 x 0) + (25 x -1) = 3
20 24 11 12 16 19
19 17 20 23 15 9
21 40 25 13 14 8
9 18 8 6 11 22
31 3 7 9 17 23
20 12 3 11 19 30
*
1 0 -1
2 0 -2
1 0 -1
=
3 27
Sample calculation for second filter location
(24 x 1) + (11 x 0) + (12 x -1) +
(17 x 2) + (20 x 0) + (23 x -2) +
(40 x 1) + (25 x 0) + (13 x -1) = 27



Image

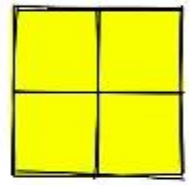


Convolved Feature

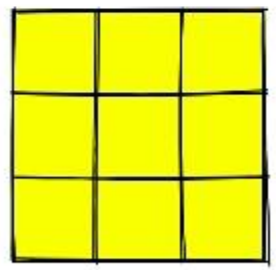
Feature map

Input image: 5x5px

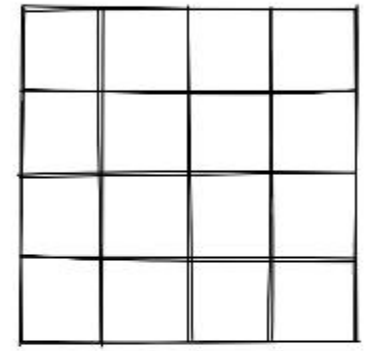
Filter



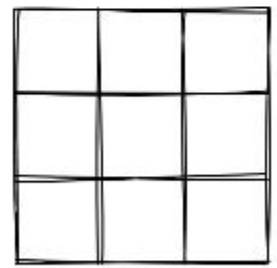
2x2



3x3

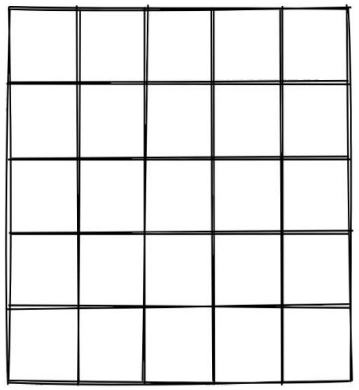


4x4



3x3

Padding



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

$6 \times 6 \rightarrow 8 \times 8$

*

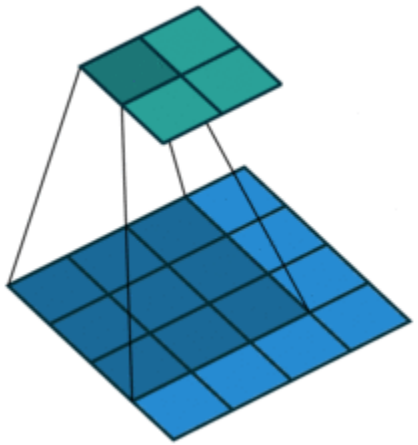
1	0	-1
1	0	-1
1	0	-1

3×3

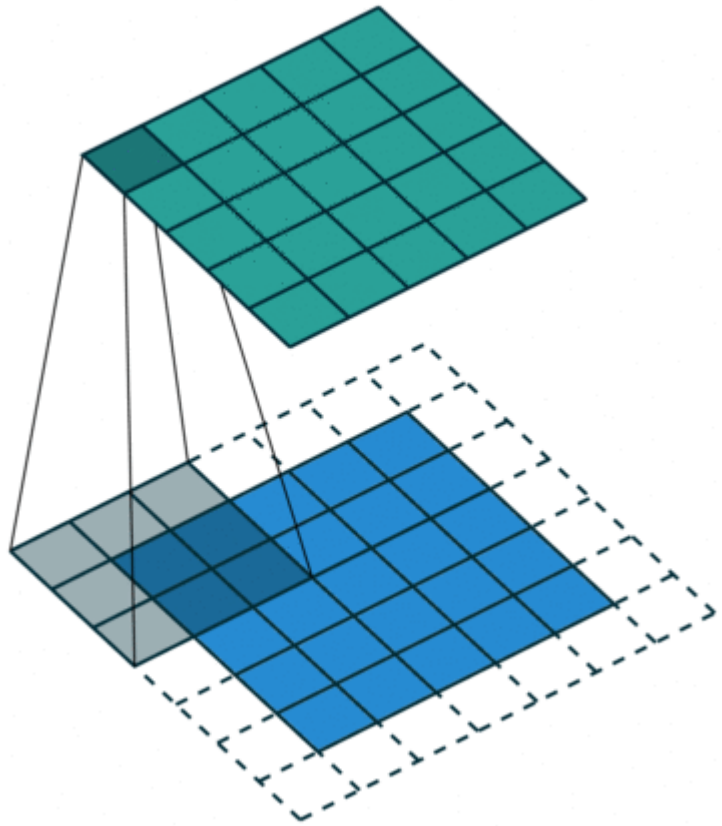
=

-10	-13	1			
-9	3	0			

6×6

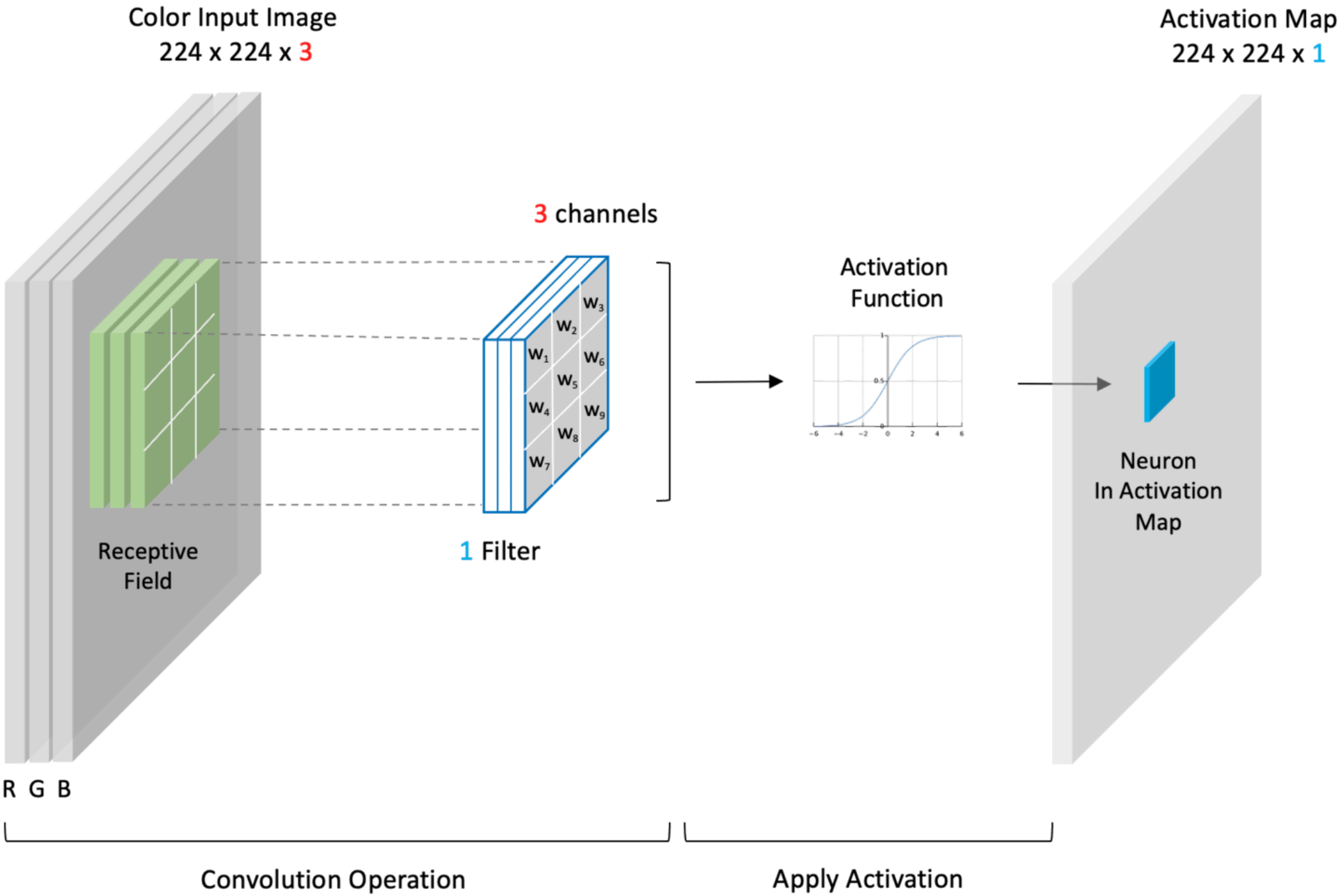


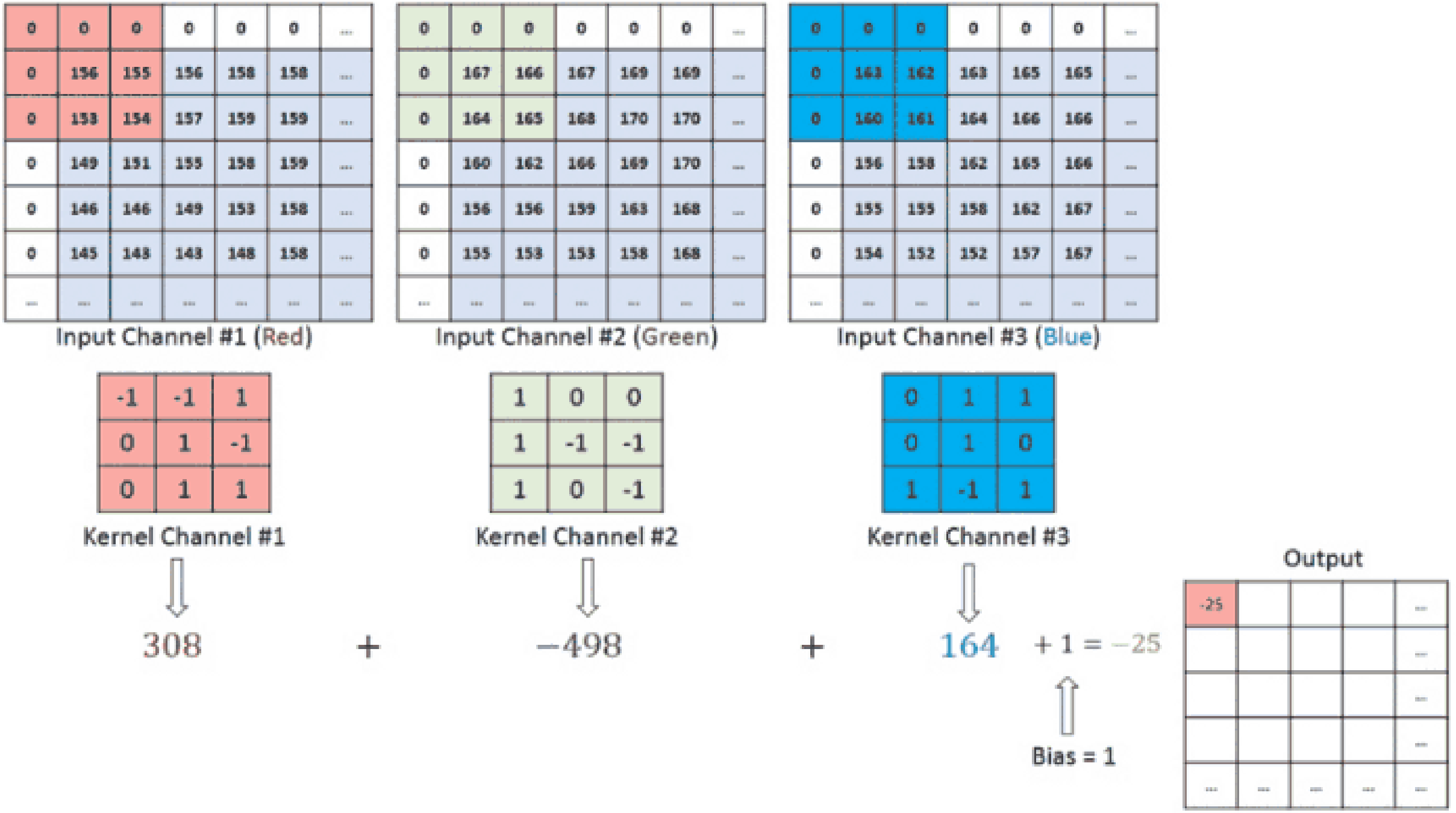
No Padding, Stride = 1

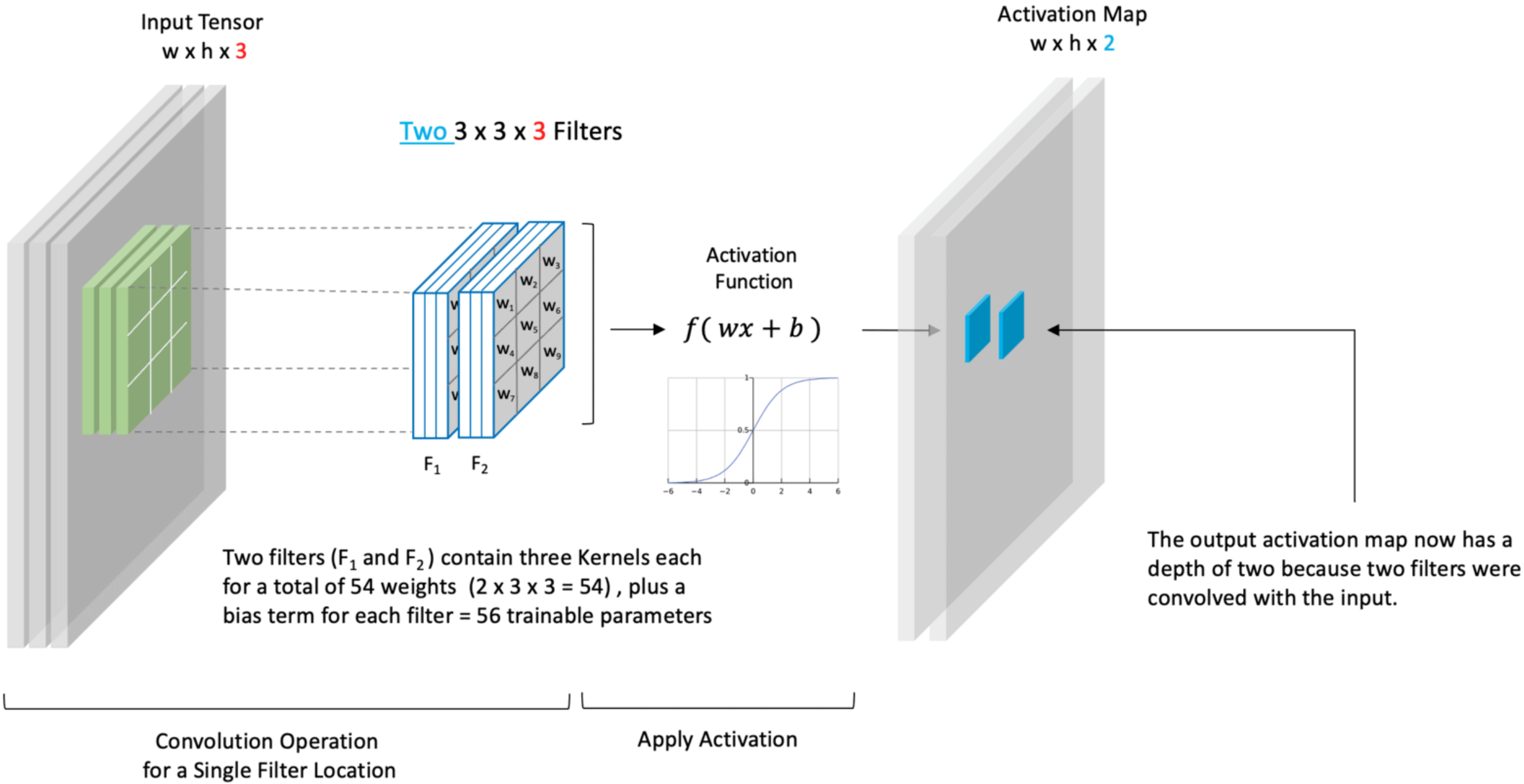


Same Padding, Stride =1

Ver código de convolução no notebook


Color Input Image
224 x 224 x 3
Activation Map
224 x 224 x 1
3 channels
Activation
Function
Receptive
Field
1 Filter
Neuron
In Activation
Map
R G B
Convolution Operation
Apply Activation

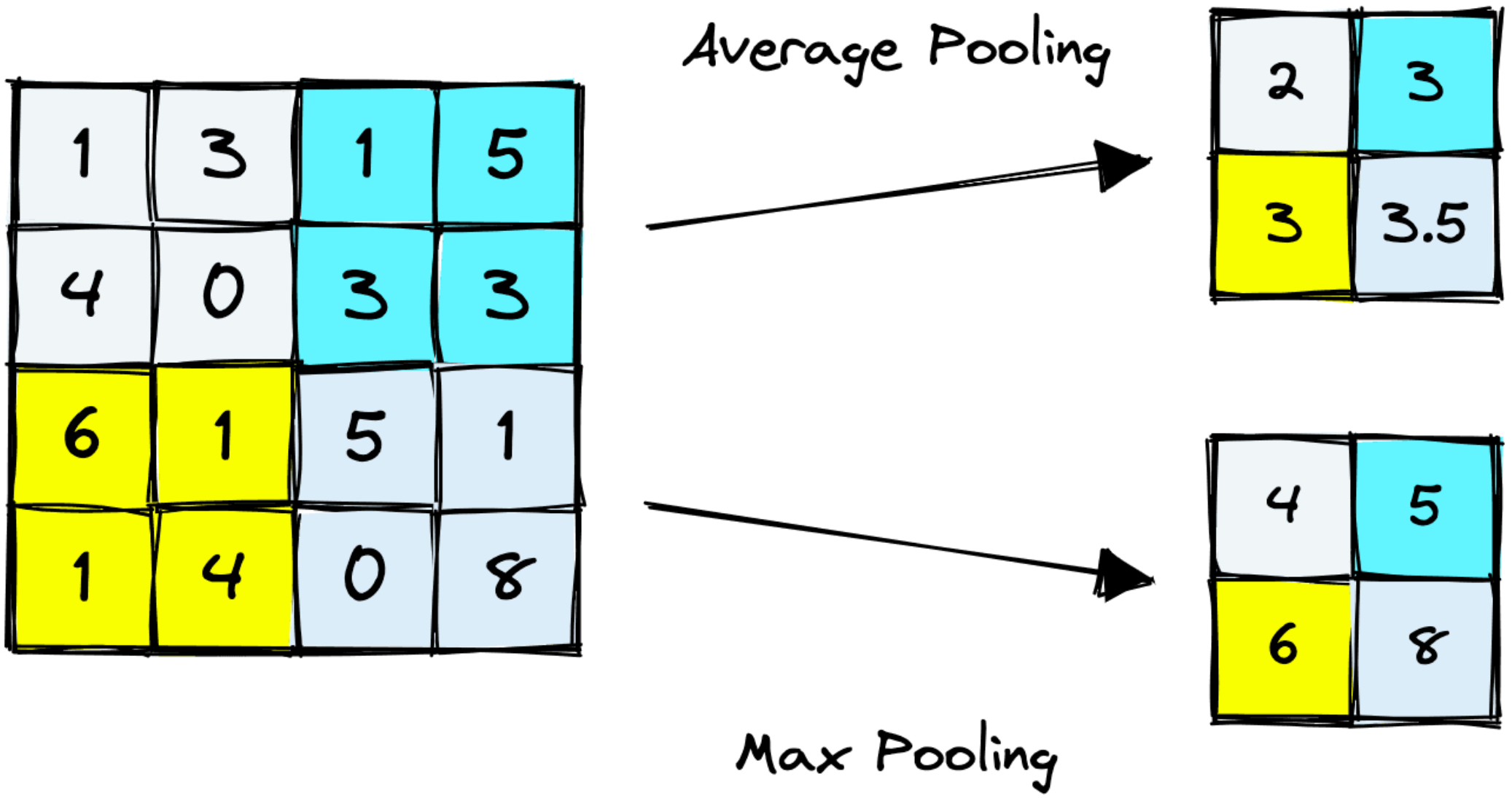

0 0 0 0 0 0 ...
0 156 155 156 158 158 ...
0 153 154 157 159 159 ...
0 149 151 155 158 159 ...
0 146 146 149 153 158 ...
0 145 148 148 148 158 ...
... ...
Input Channel #1 (Red)
0 0 0 0 0 0 ...
0 167 166 167 169 169 ...
0 164 165 168 170 170 ...
0 160 162 166 169 170 ...
0 156 156 159 163 168 ...
0 155 153 153 158 168 ...
... ...
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
1 0 0
1 -1 -1
1 0 -1
Kernel Channel #2
0 1 1
0 1 0
1 -1 1
Kernel Channel #3
308
+
−498
+
164
+ 1 = −25
Bias = 1
Output
-25 ... ...
... ...
... ...
... ...
... ...

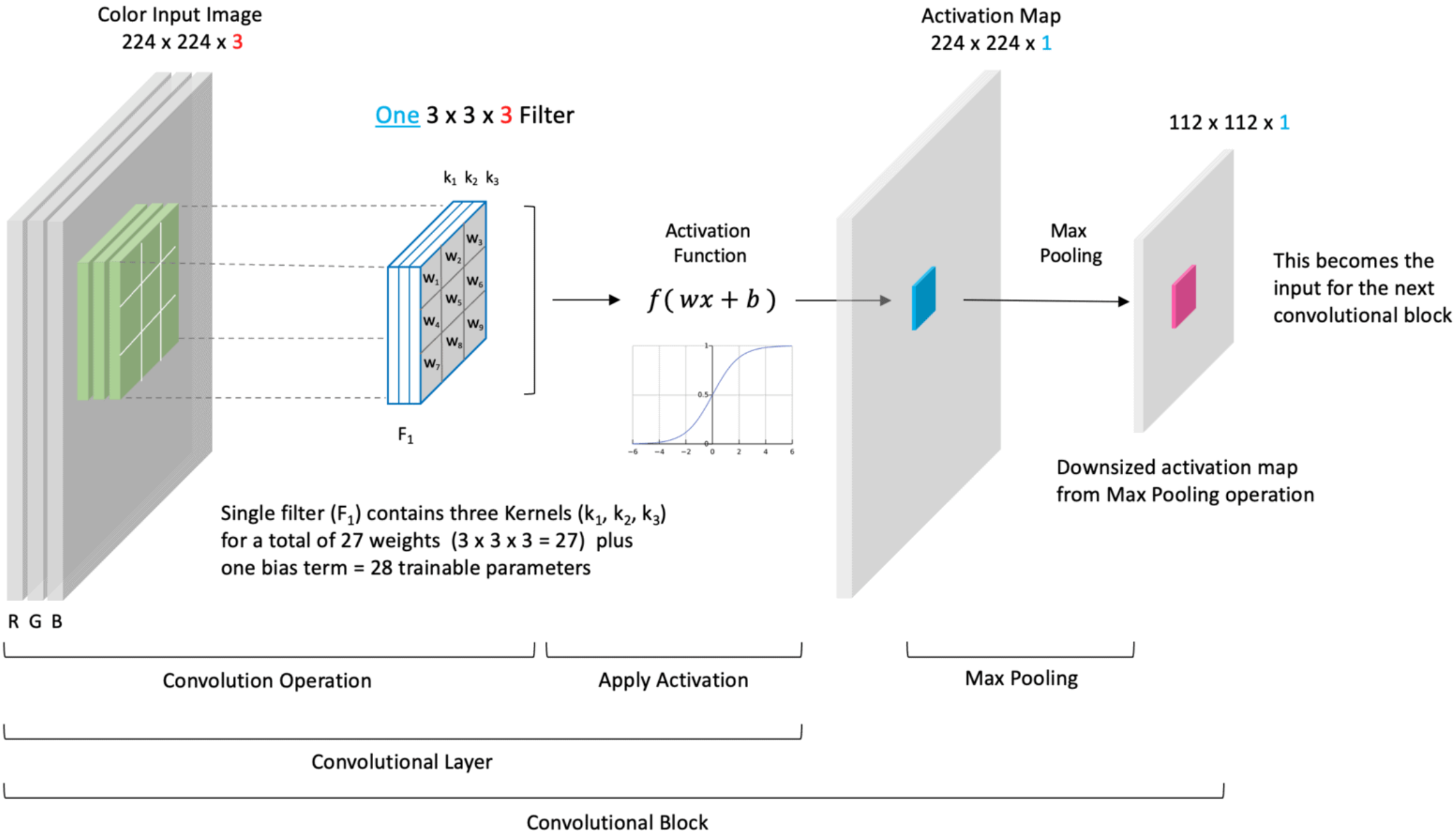

Input Tensor
w x h x 3
Two 3 x 3 x 3 Filters
W_1 W_2 W_3
W_4 W_5 W_6
W_7 W_8 W_9
F_1 F_2
Activation Function
$f(wx + b)$
1
0.5
0
−6 −4 −2 0 2 4 6
Activation Map
w x h x 2
Two filters (F_1 and F_2) contain three Kernels each for a total of 54 weights (2 x 3 x 3 = 54), plus a bias term for each filter = 56 trainable parameters
The output activation map now has a depth of two because two filters were convolved with the input.
Convolution Operation for a Single Filter Location
Apply Activation

Kernel size, Padding e Stride

Veja animação.

Pooling

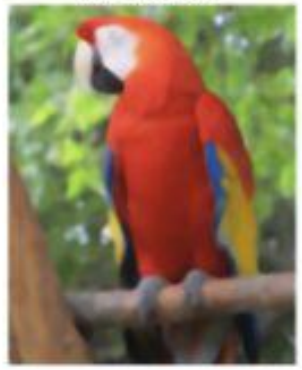


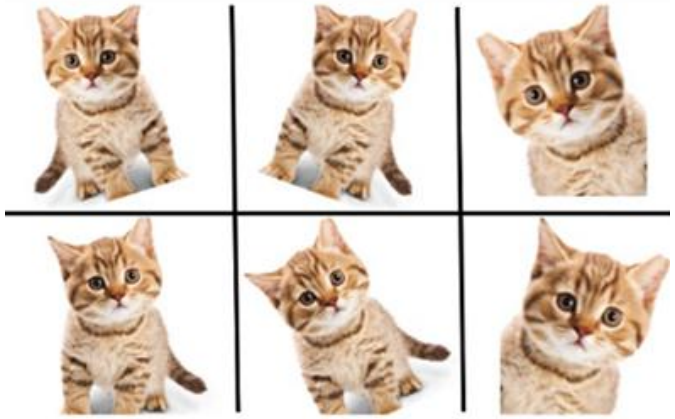

Color Input Image
224 x 224 x 3
One 3 x 3 x 3 Filter
k1 k2 k3
W1 W2 W3 W4 W5 W6 W7 W8 W9
F1
Activation Function
f (wx + b)
Activation Map
224 x 224 x 1
Max Pooling
112 x 112 x 1
This becomes the input for the next convolutional block
Downsized activation map from Max Pooling operation
Single filter (F1) contains three Kernels (k1, k2, k3)
for a total of 27 weights (3 x 3 x 3 = 27) plus
one bias term = 28 trainable parameters
R G B
Convolution Operation
Apply Activation
Max Pooling
Convolutional Layer
Convolutional Block

Data Augmentation

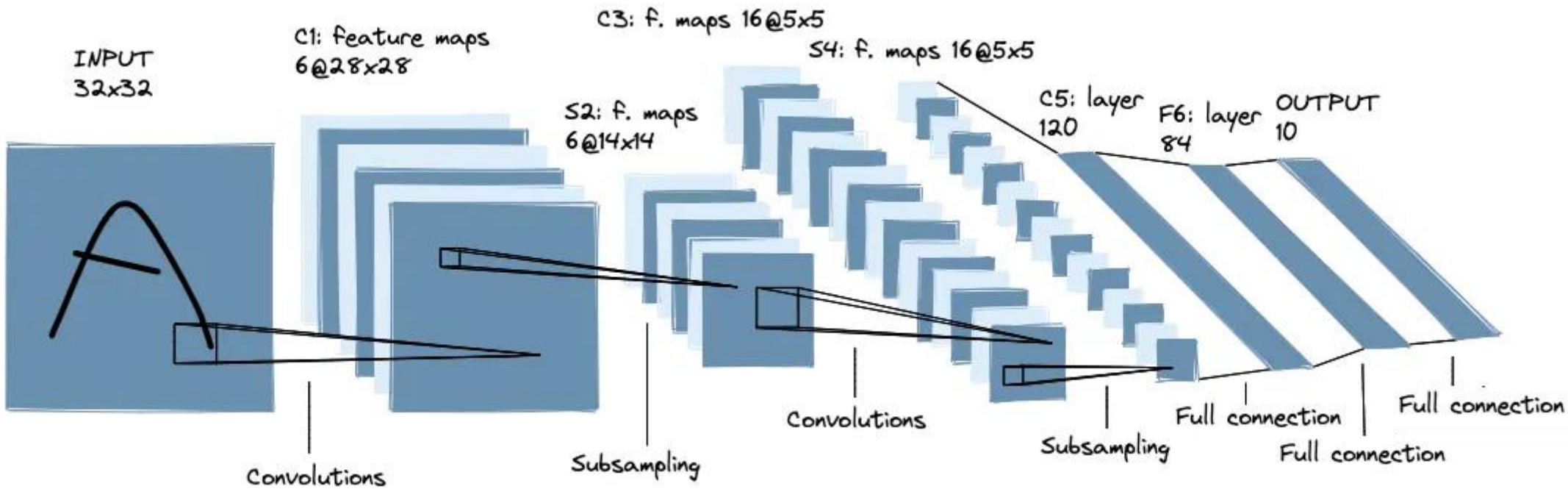
Ajuda a prevenir *overfitting* e torna o modelo robusto à alguns tipos de ruído criando dados artificiais

augmentation





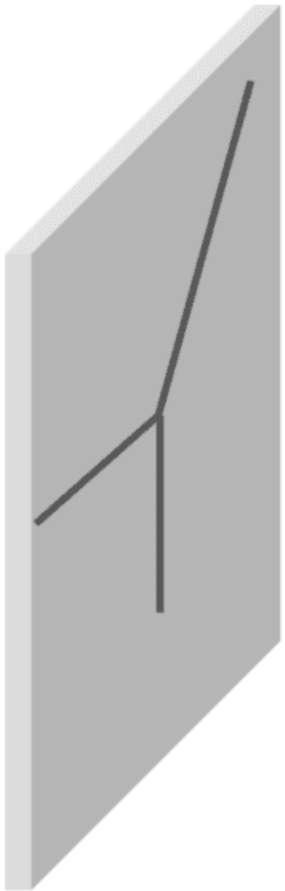
Arquitetura Completa



Ver operações visualmente na CNNExplainer

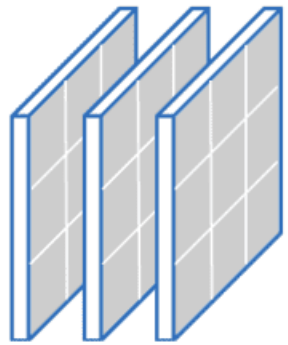
Ver visualização em 3D

Ver como treinar uma rede com pytorch no notebook.



Input Image

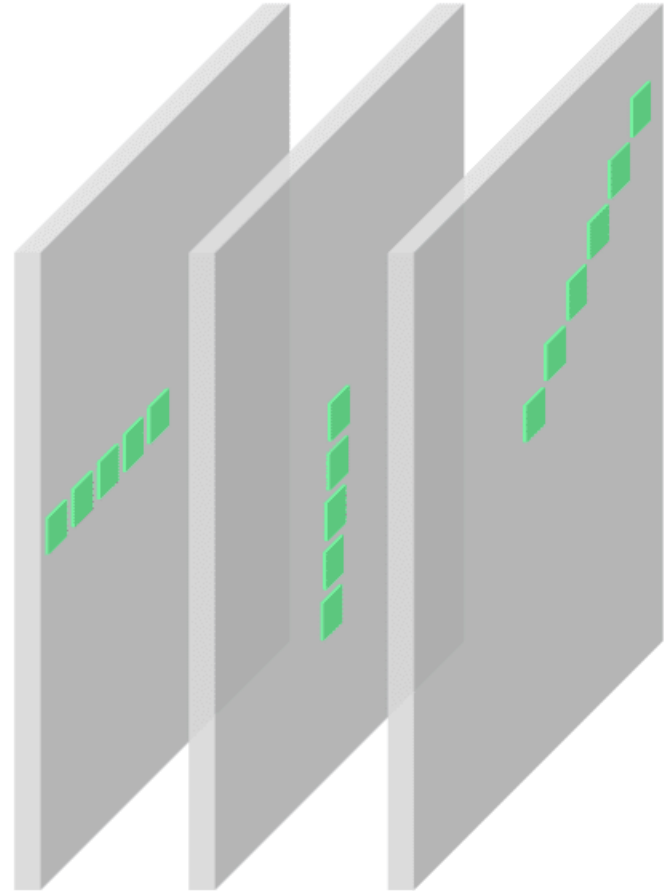
Filters learn to detect structural patterns



F_1 = Horizontal Lines

F_2 = Vertical Lines

F_3 = Diagonal Lines



Activation Maps

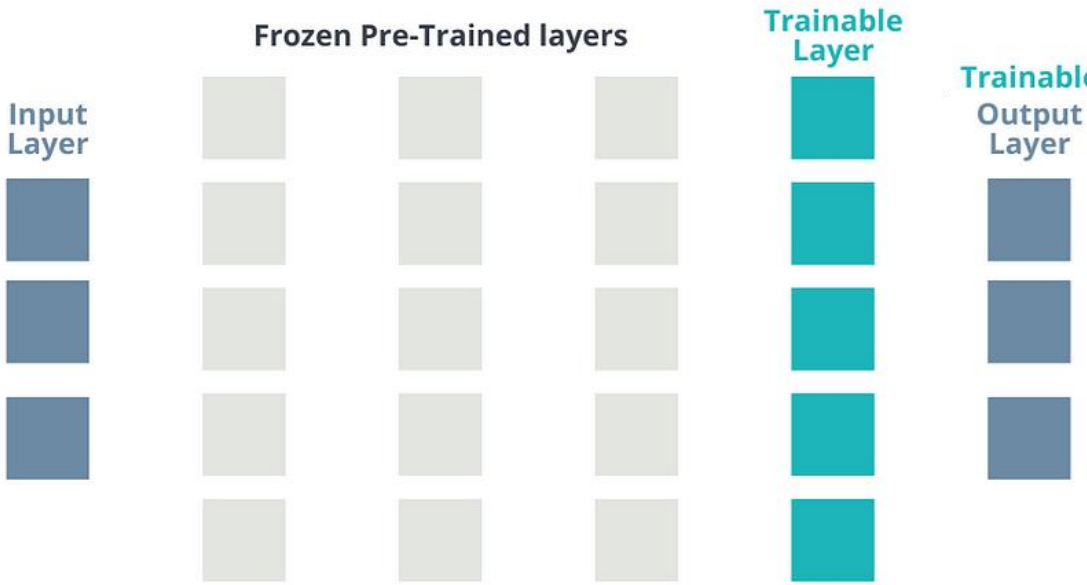
= Highly activated neurons in activation maps

Dê uma olhada no artigo da AlexNet

- Visualização de kernels aprendidos.
- *Embeddings*, distâncias entre *embeddings* e aprendizado de representações.
- Dropout, net.train() e net.eval()

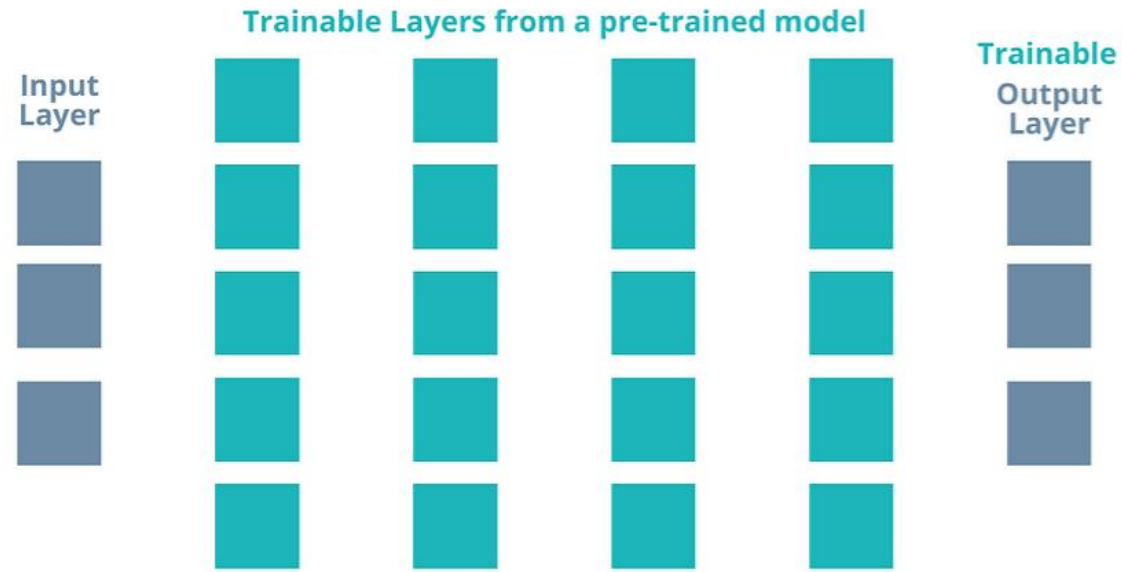
Redes Neurais Pré-Treinadas, Transfer Learning, Feature Extraction, Fine-Tuning

Feature Extraction



In feature extraction, you **freeze the pre-trained model** layers to preserve existing learning and **add new layers** to learn additional information.

Fine Tuning



In fine-tuning, you **unfreeze the entire model** and **train it with a lower learning rate** to adapt to new challenges.

Ver notebook exemplo completo