Hands-on Introduction to Deep Learning

Convolutional Neural Networks

INSTITUT DU
DÉVELOPPEMENT ET DES
RESSOURCES EN
INFORMATIQUE
SCIENTIFIQUE

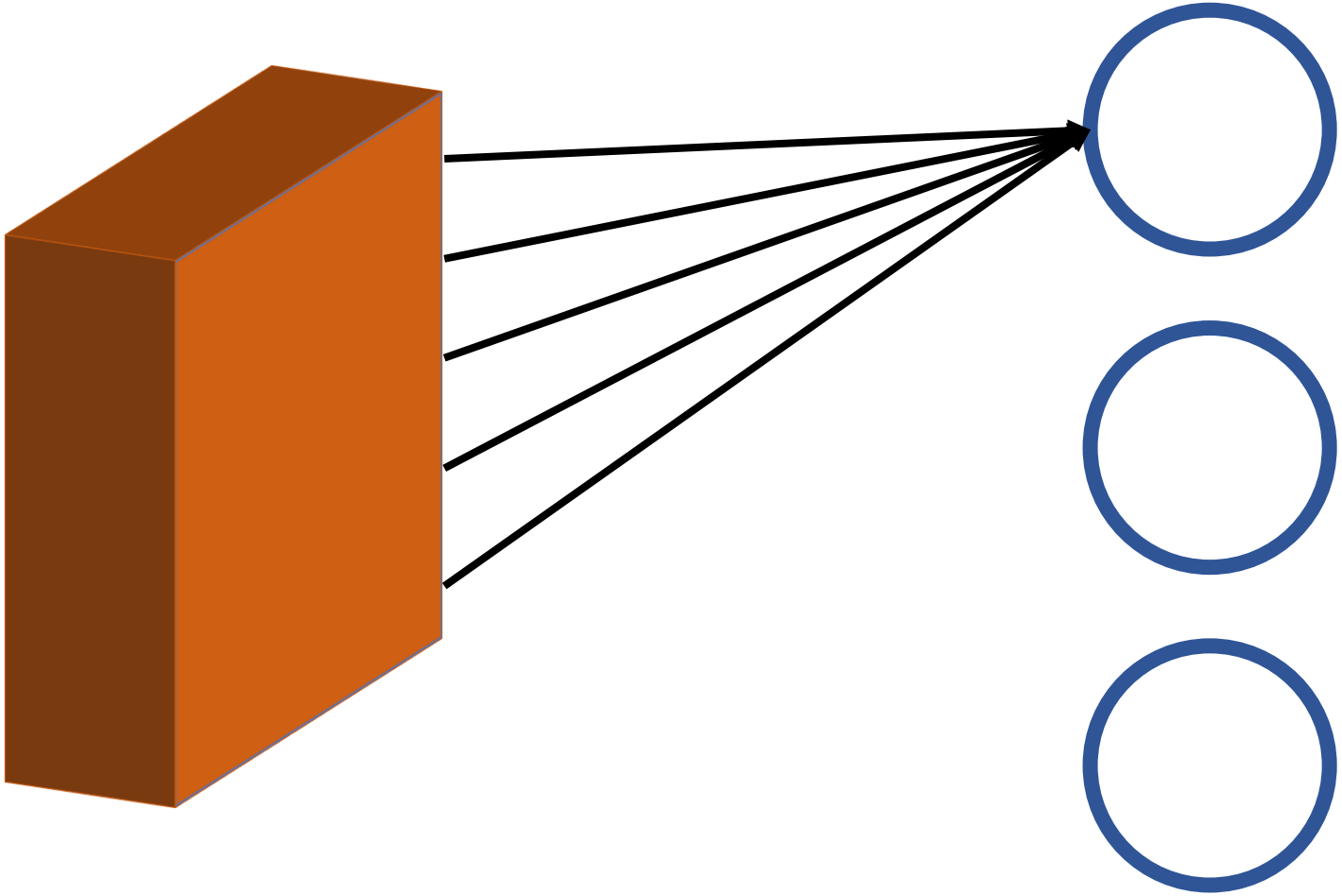


Image : 3 x 5 x 5

1 layer, 3 neurons :
75 weights per neuron

Motivations

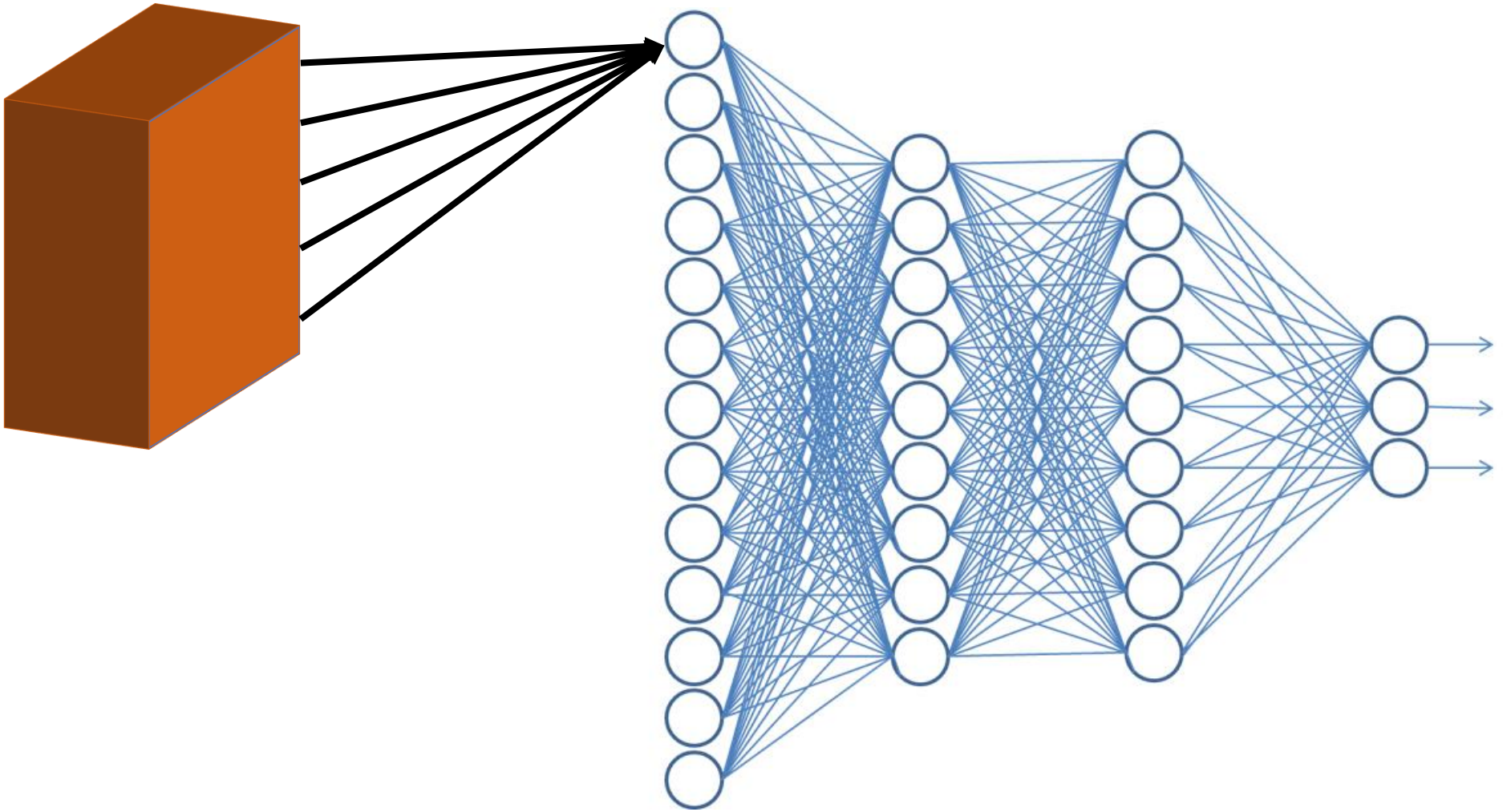
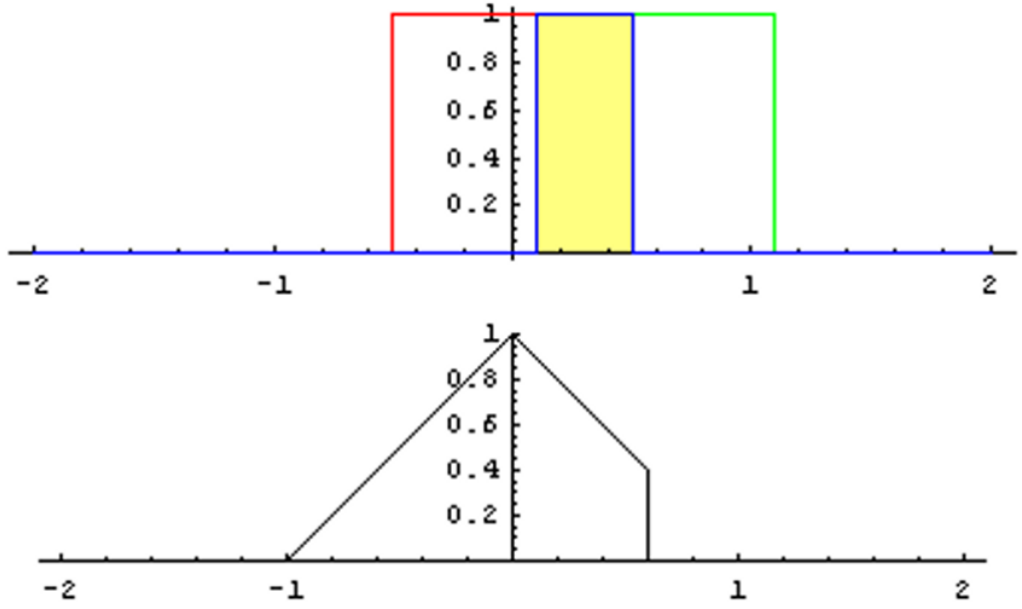


Image : 3 x Height x Width

Complex model

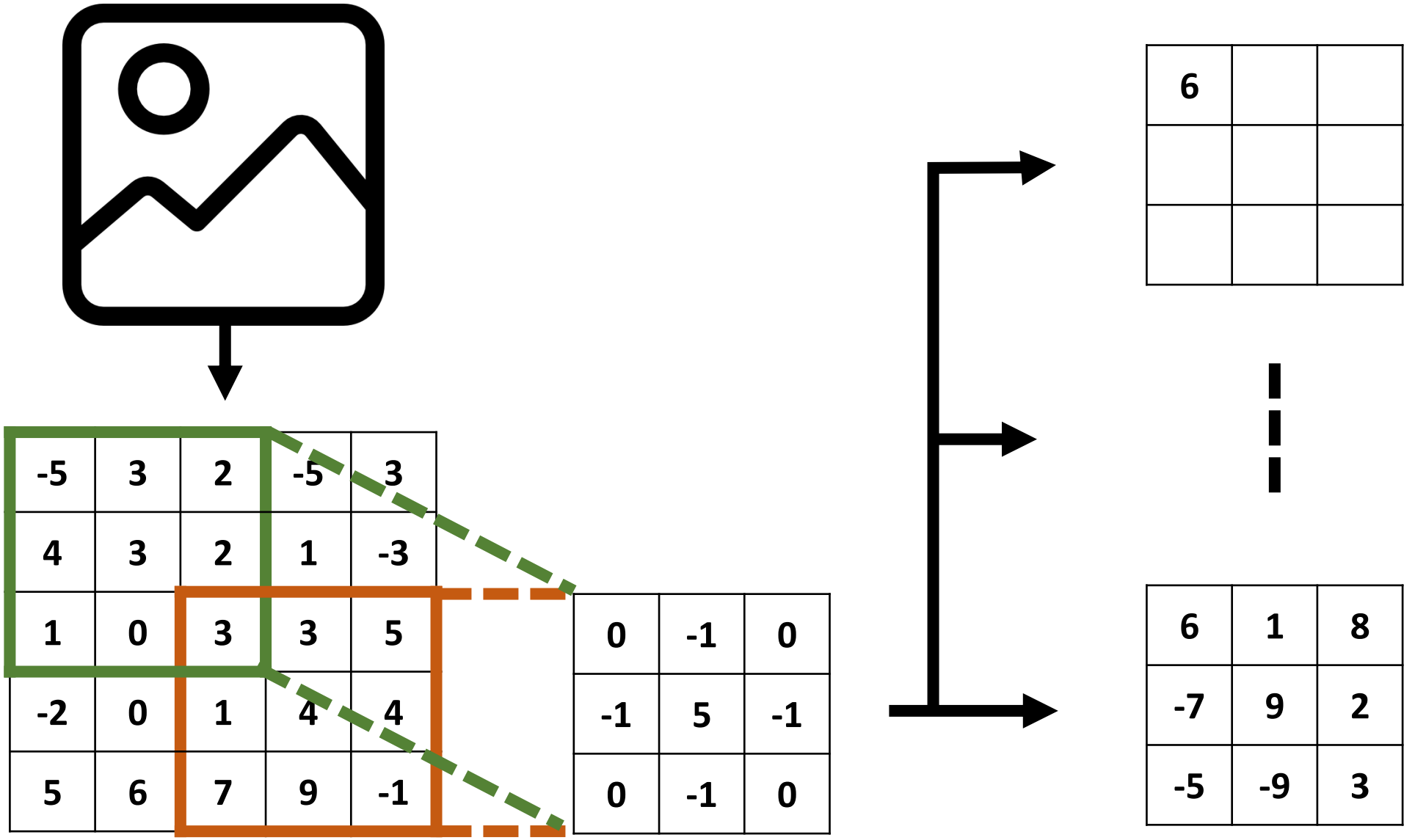
Motivations

$$f * g = \int_{-\infty}^{\infty} f(\tau) g(t - \tau) d\tau$$



$$f * g = \sum_{\substack{0 \le i \le n \\ 0 \le j \le m}} f(x_{i,j}) g(x_{k-i,l-j})$$

Convolution



Convolution 2D

Input

Edge detection

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Box mean

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

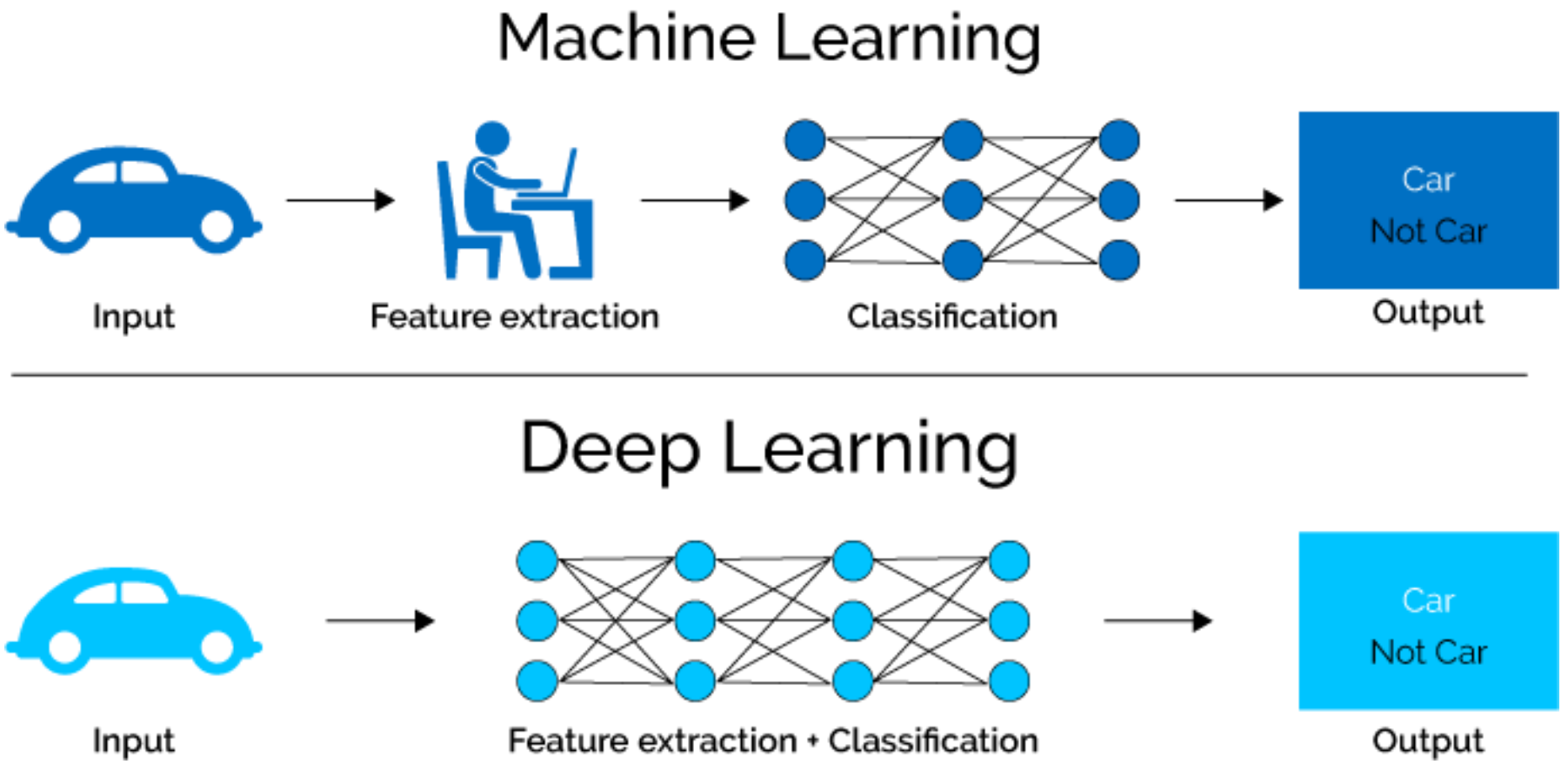
Sharpen

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

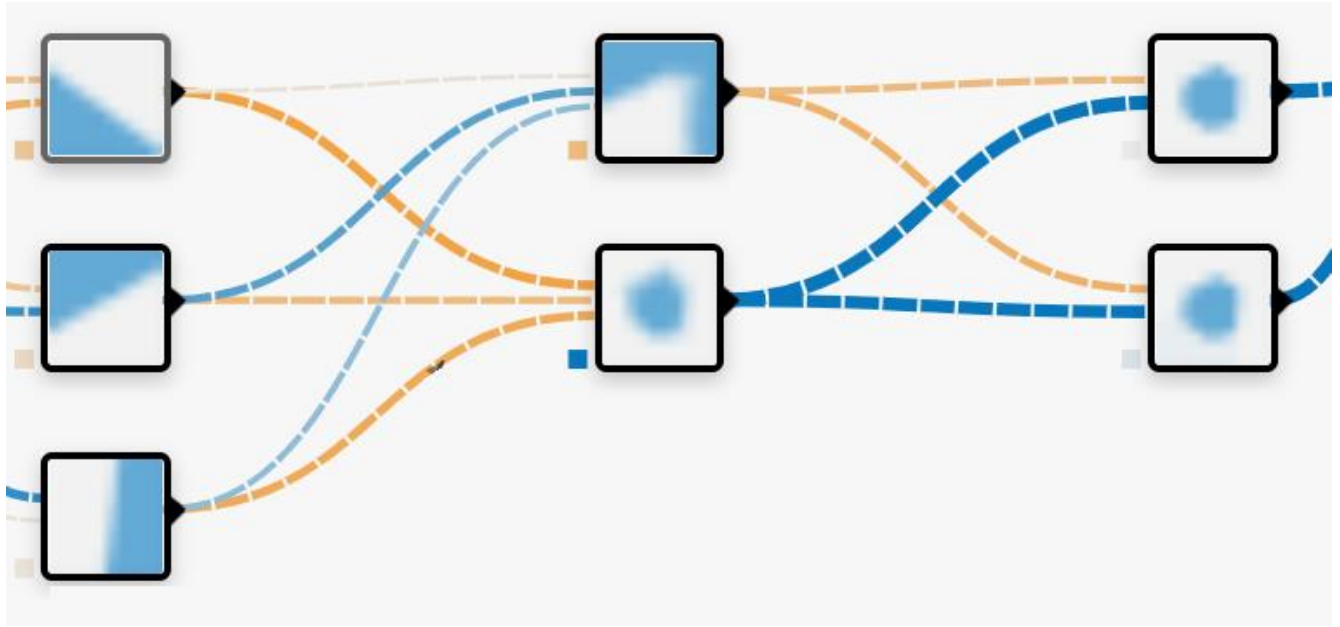
Gaussian blur

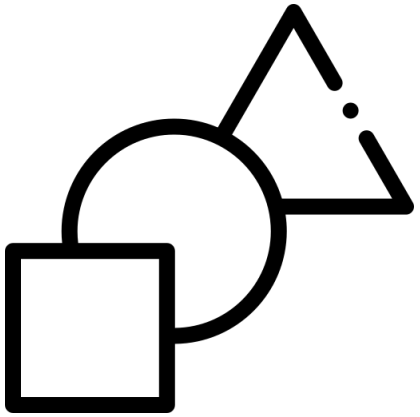
$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

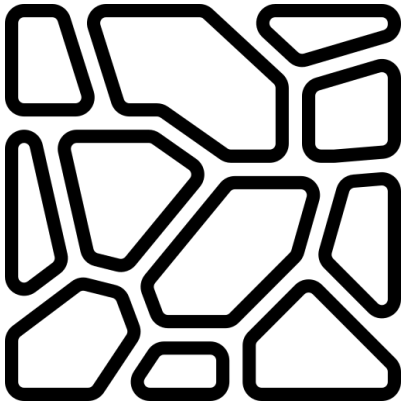
Convolution 2D

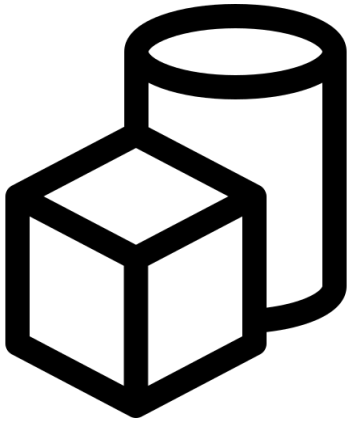


Convolutions role

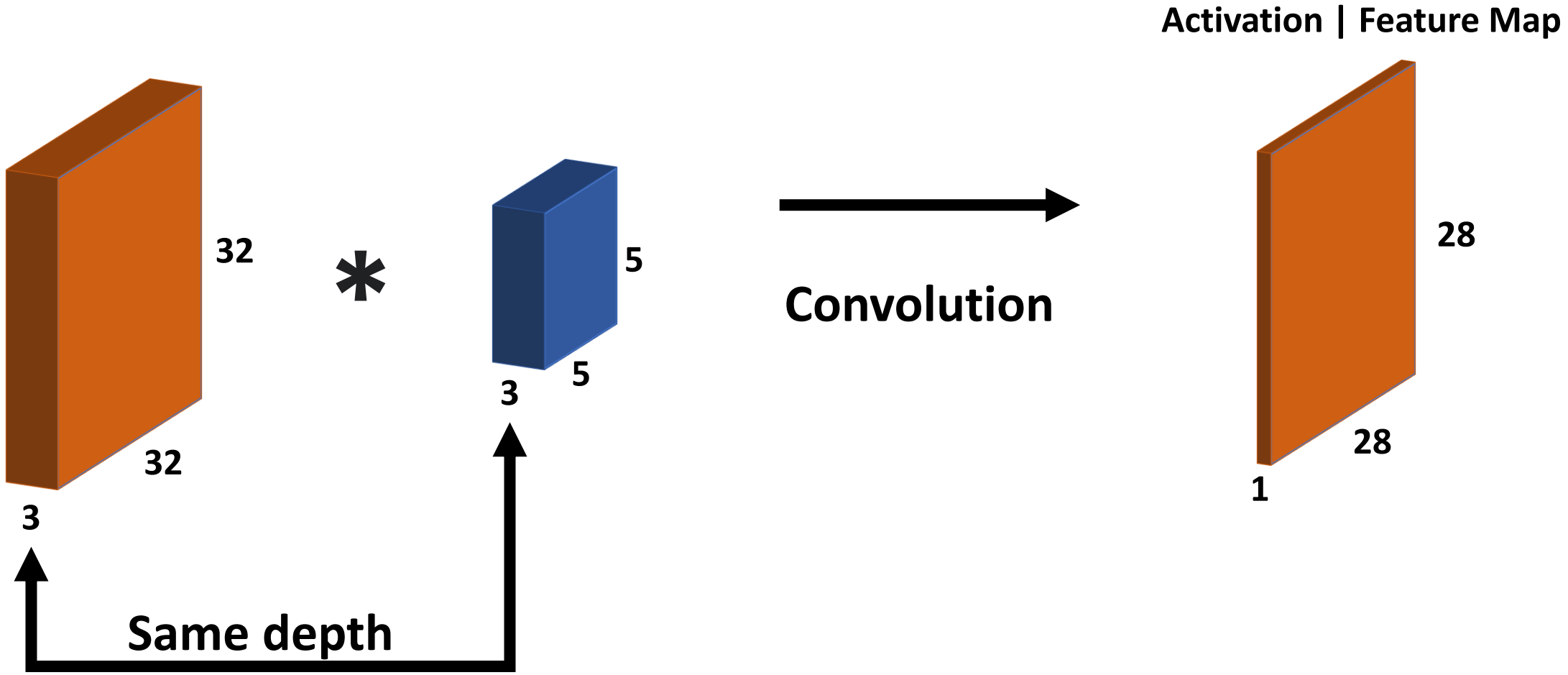




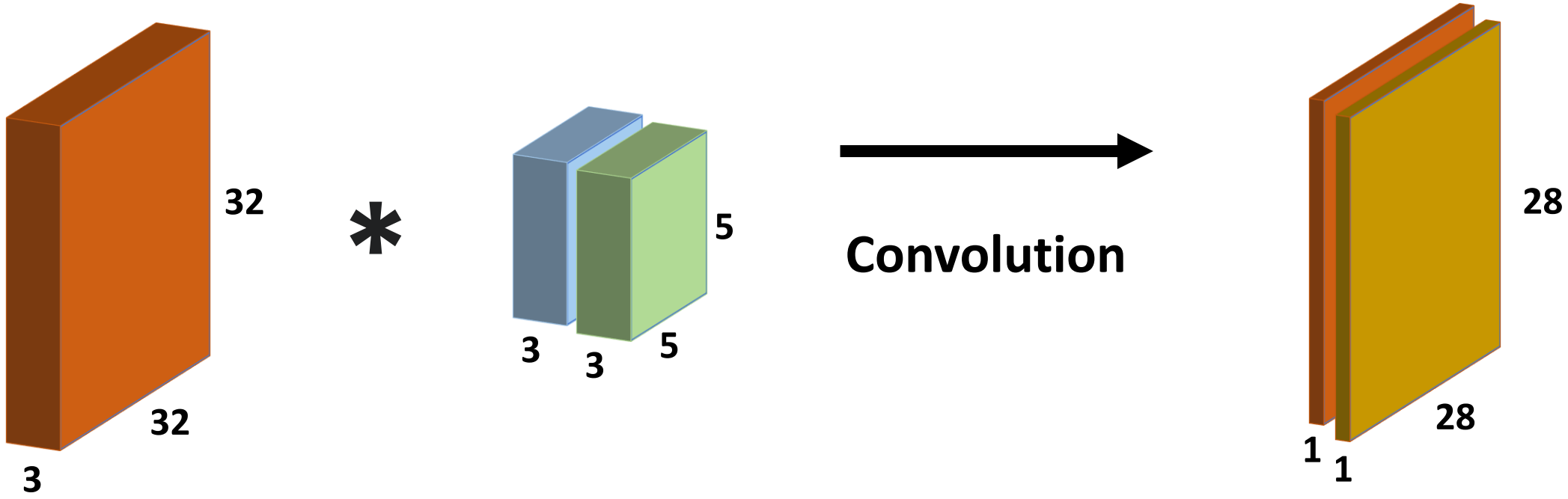




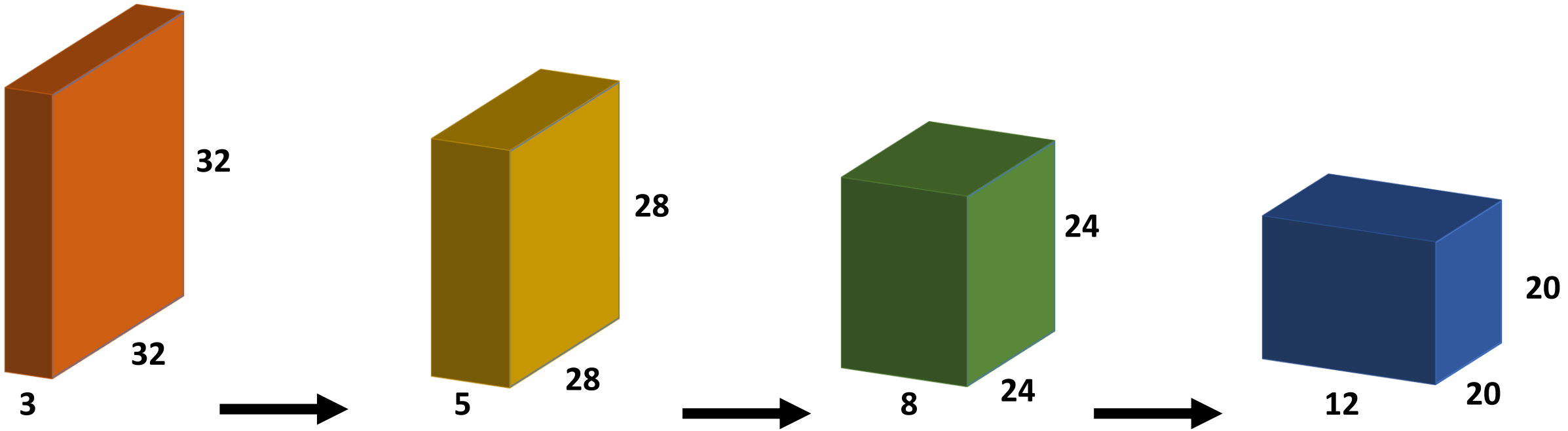
Convolutions role



Convolution 3D



Kernel and activations map



Dimensions evolution

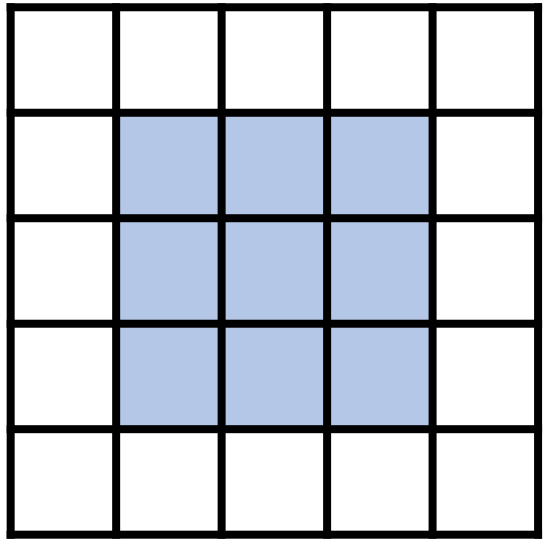
0	0	0	0	0	0	0
0	-5	3	2	-5	3	0
0	4	3	2	1	-3	0
0	1	0	3	3	5	0
0	-2	0	1	4	4	0
0	5	6	7	9	-1	0
0	0	0	0	0	0	0

$$\text{Output} : \left(\frac{N+2 \cdot P-F}{S}+1\right) \times \left(\frac{N+2 \cdot P-F}{S}+1\right)$$

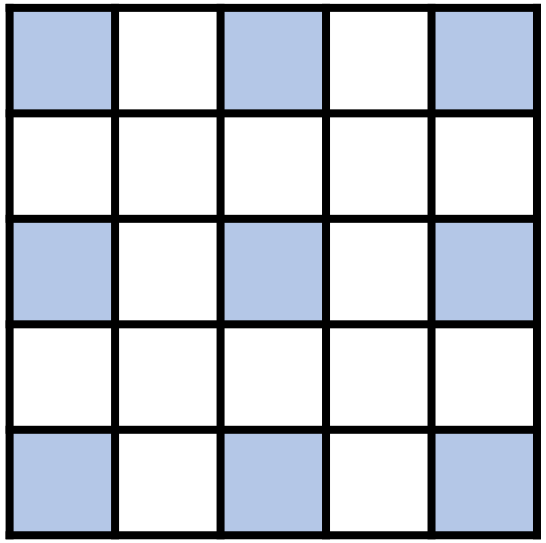
- **Entrée : 5 x 5**
- **Filtre : 3 x 3**
- **Padding : 1**
- **Stride : 1**
- **Sortie : 5 x 5**

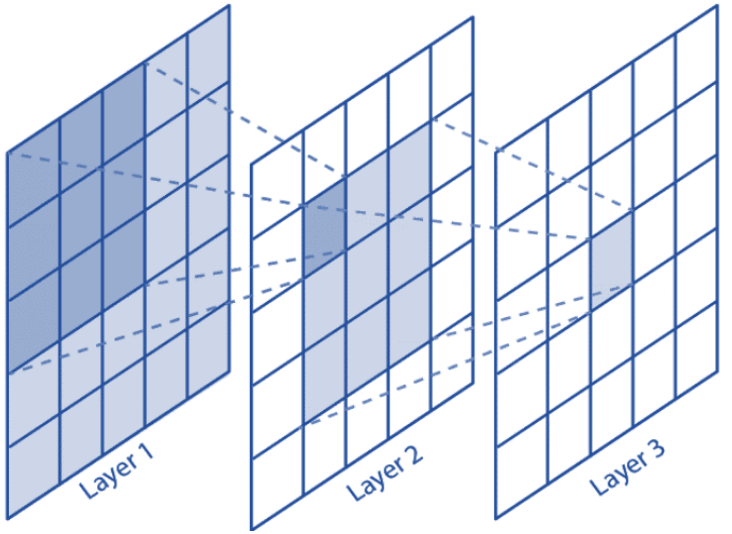
Padding

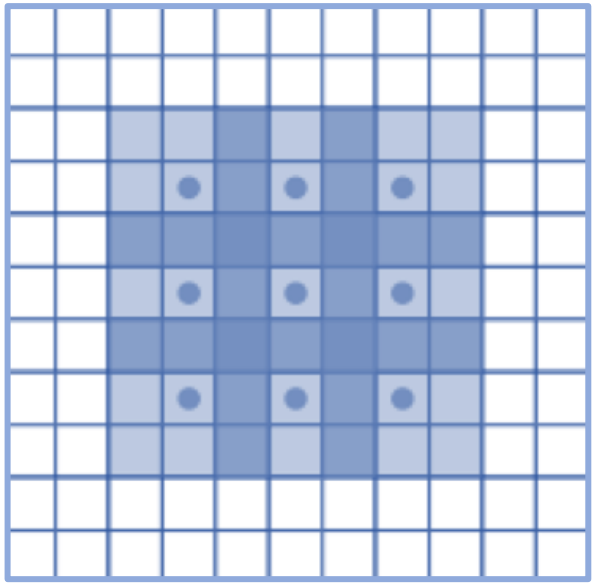
D = 1



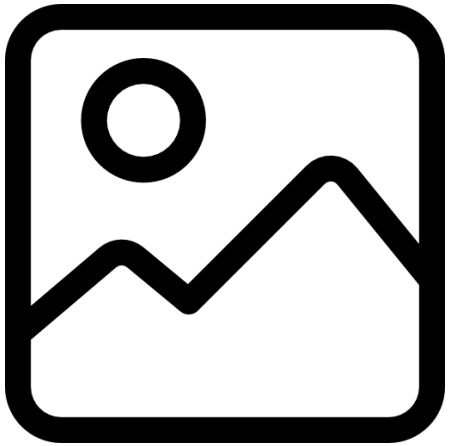
D = 2







Dilated convolution



Compression

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

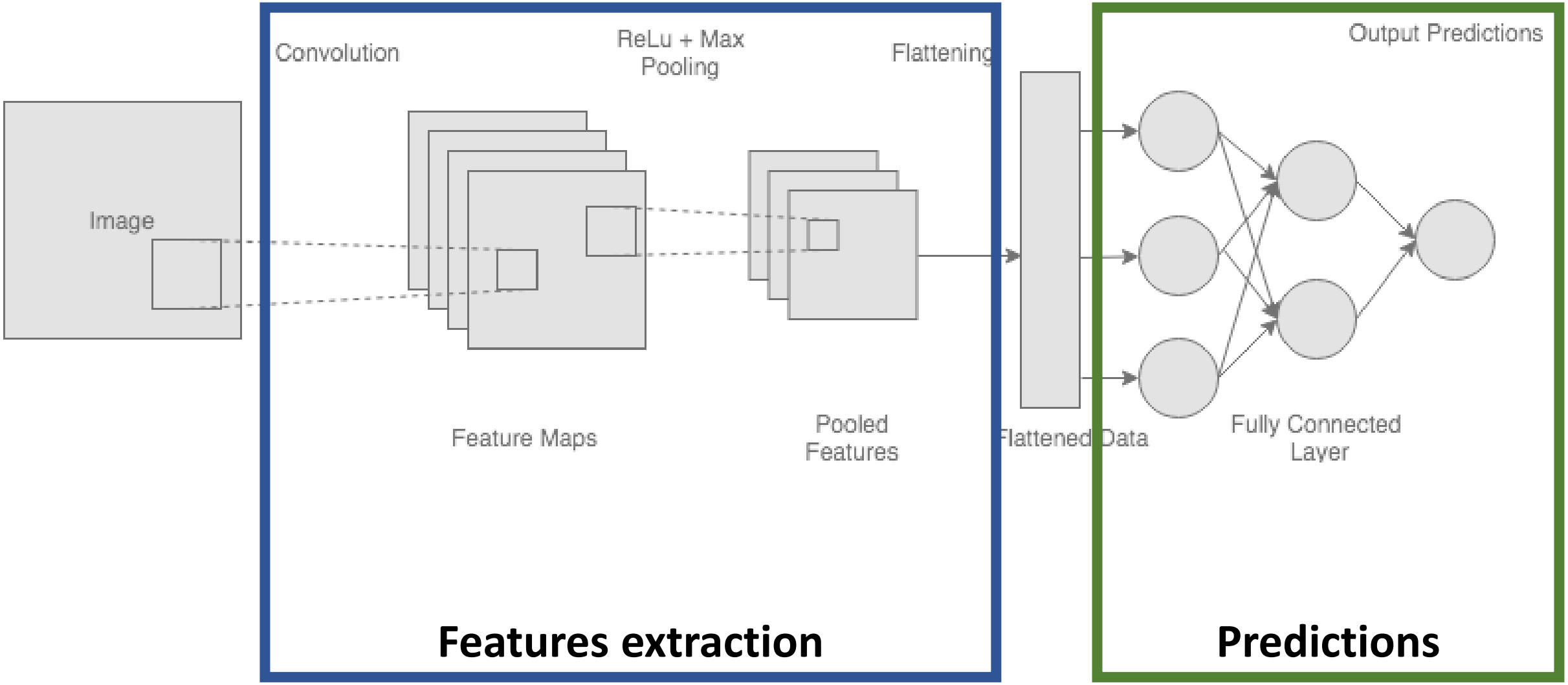
Max Pooling
Filter : 2 x 2
Stride : 2

5	9
8	6

Mean Pooling
Filter : 2 x 2
Stride : 2

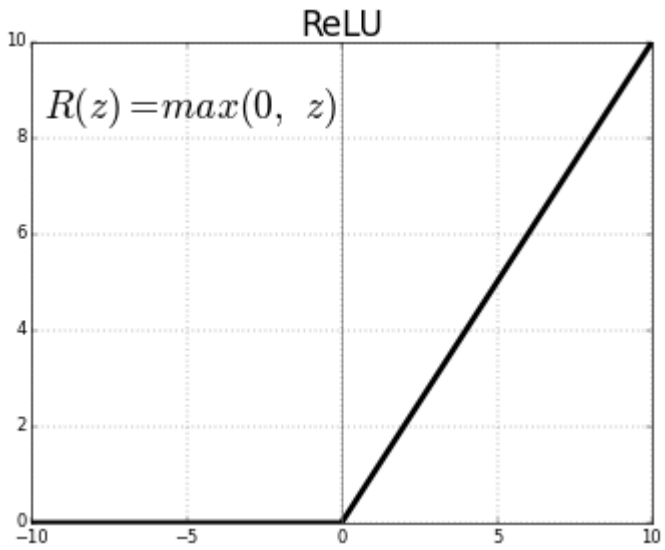
3,5	5,25
4,75	5

Pooling



Final Fully connected layers prototype

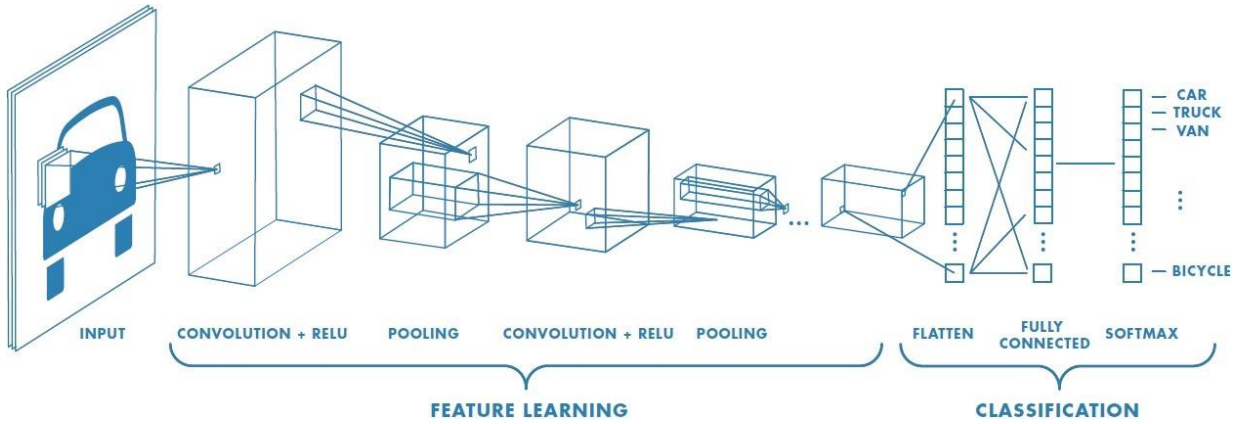
-5	3	2	-5	3
4	3	2	1	-3
1	0	3	3	5
-2	0	1	4	4
5	6	7	9	-1



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

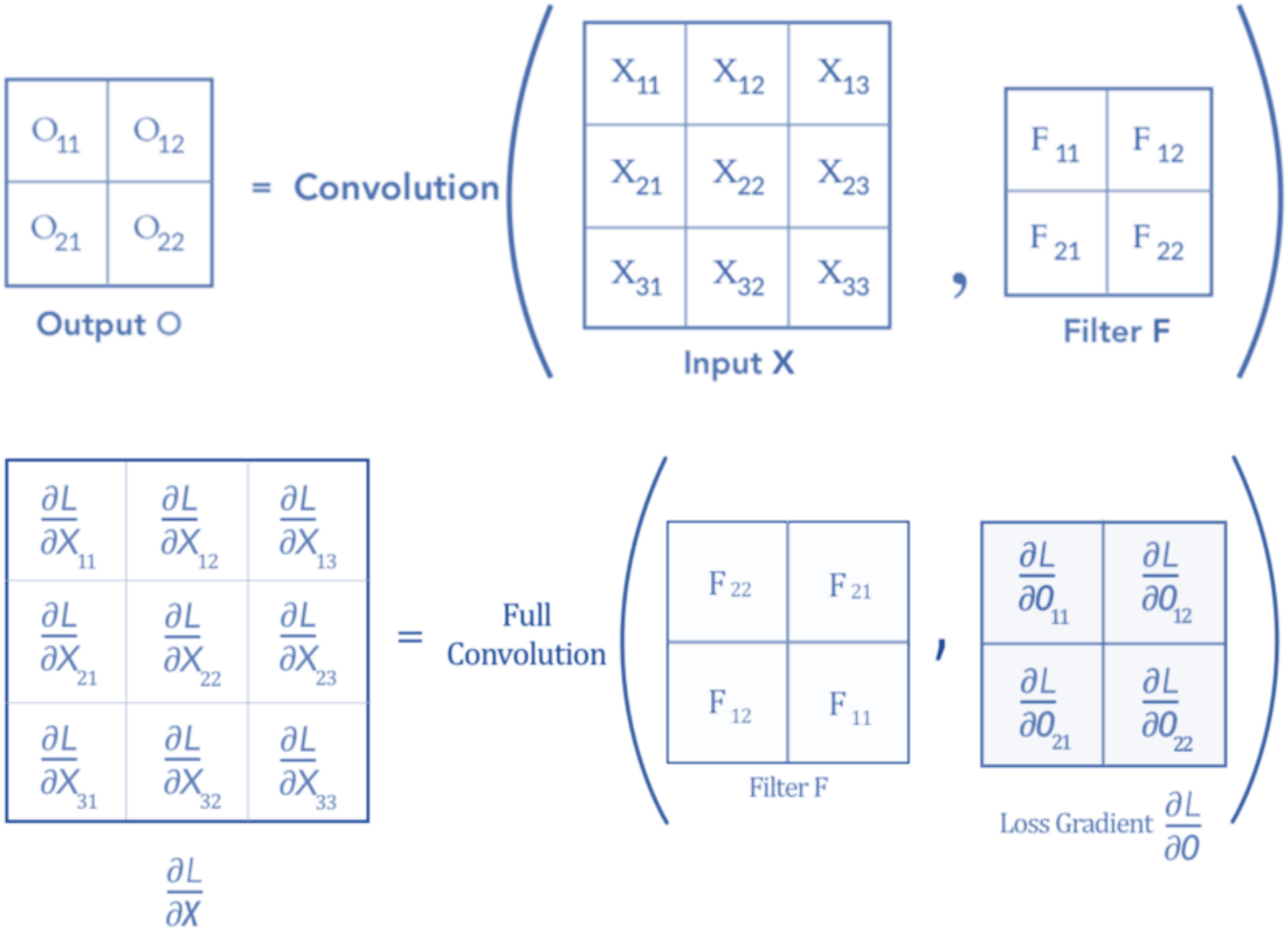
Activation function

```python
import torch.nn as nn
import torch.nn.functional as F


class Net(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.fc1 = nn.Linear(16 * 5 * 5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = torch.flatten(x, 1) # flatten all dimensions except batch
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x


net = Net()
```



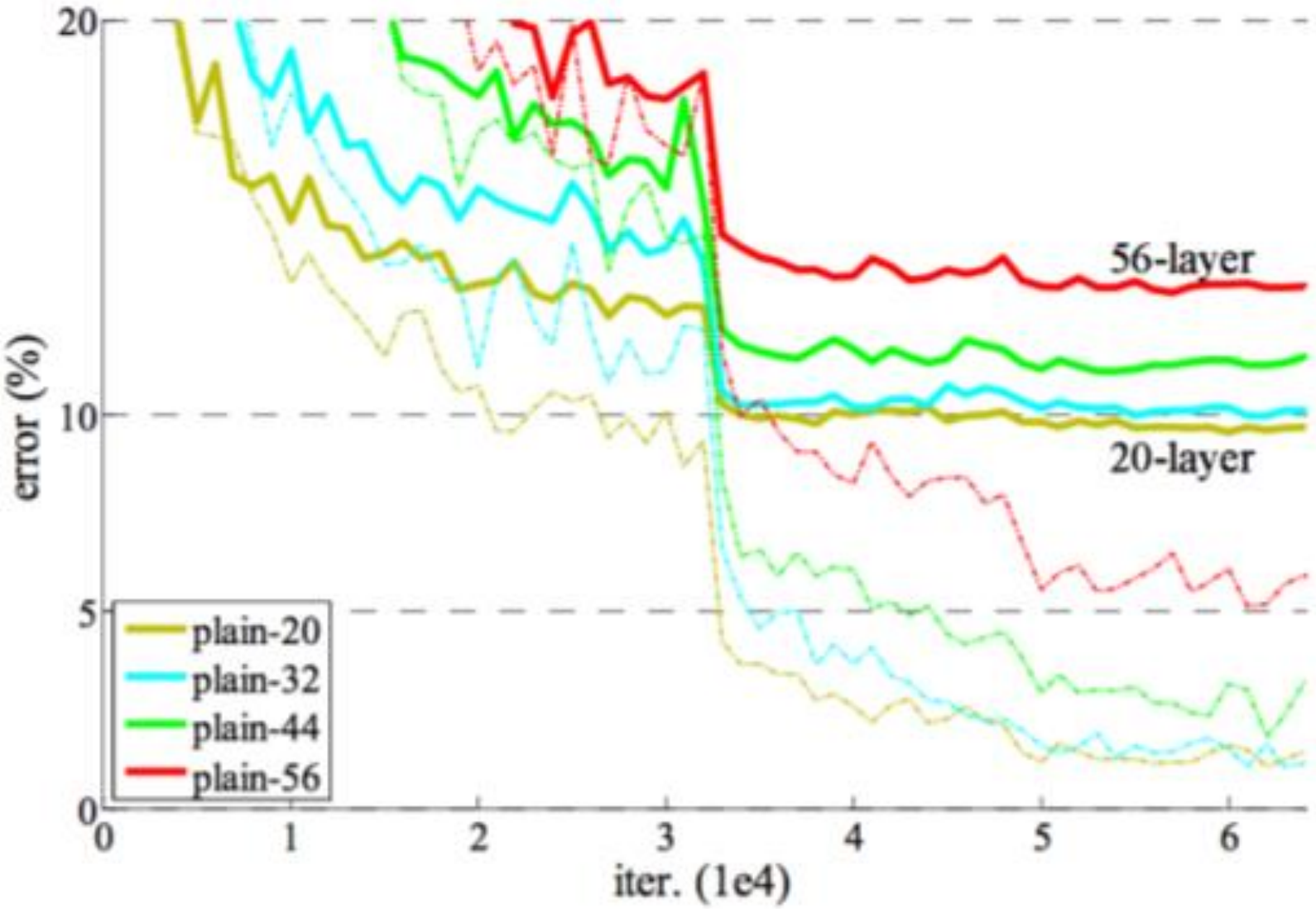

```
torch.nn.Conv2d(
            in_channels,
            out_channels,
            kernel_size,
            stride=1,
            padding=0,
            dilation=1,
            groups=1,
            bias=True,
            padding_mode='zeros',
            device=None,
            dtype=None)
```

Basic CNN implementations (Pytorch)



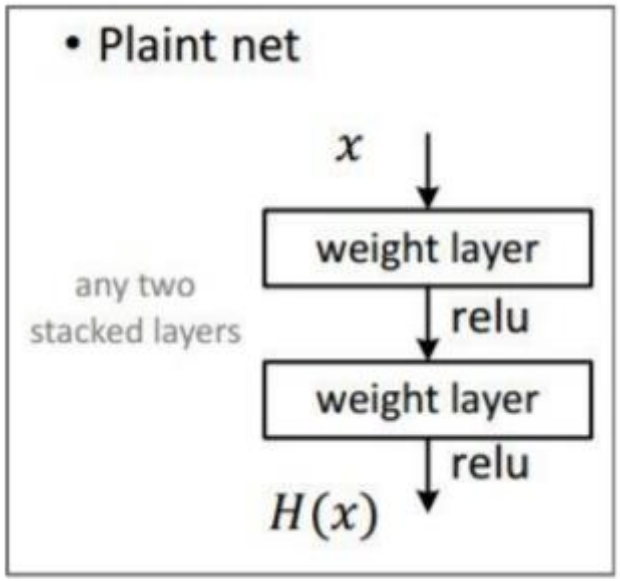
Convolution backpropagation

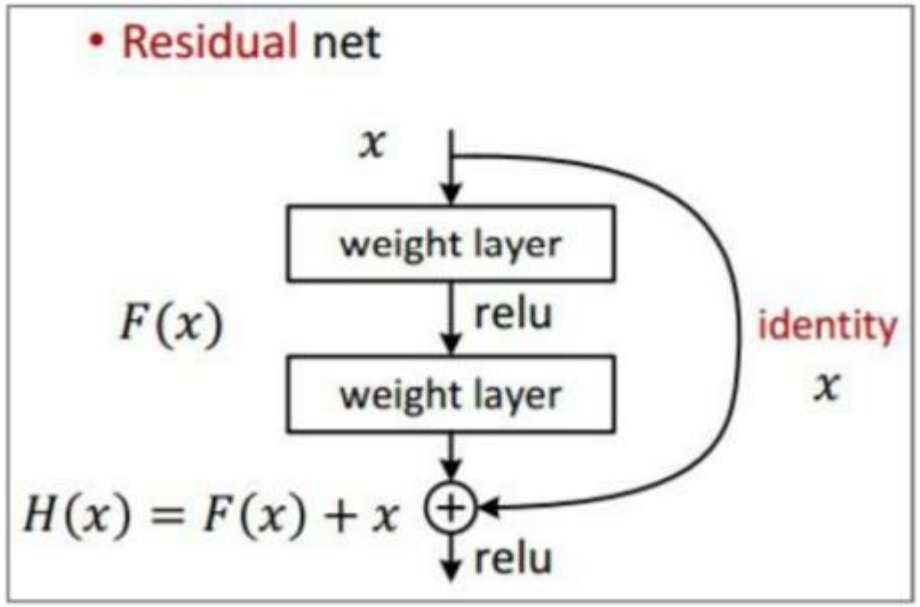
Pavithra Solai. "Convolutions and Backpropagations"

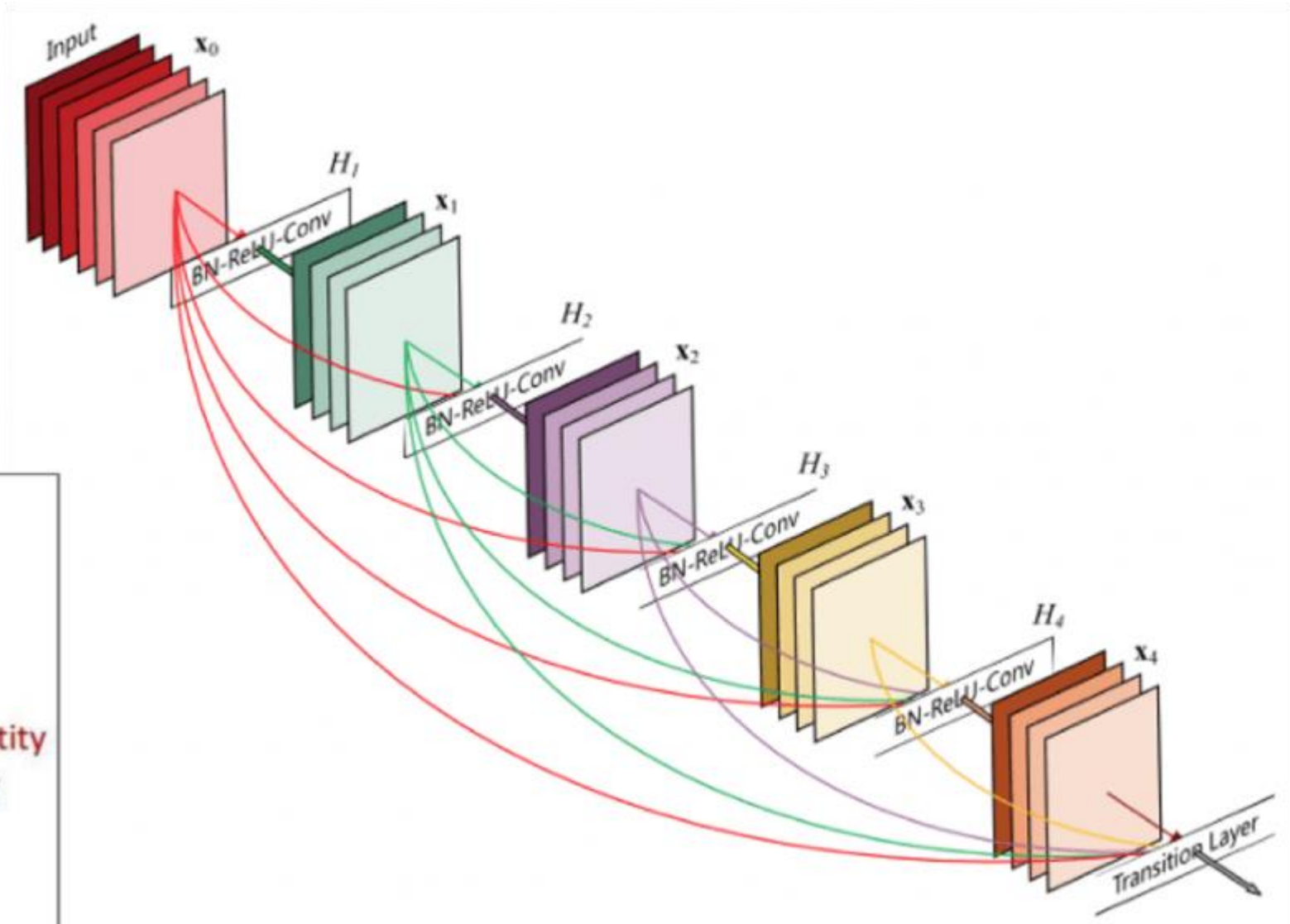


Depth problem

He, Kaiming, et al. "Deep residual learning for image recognition."

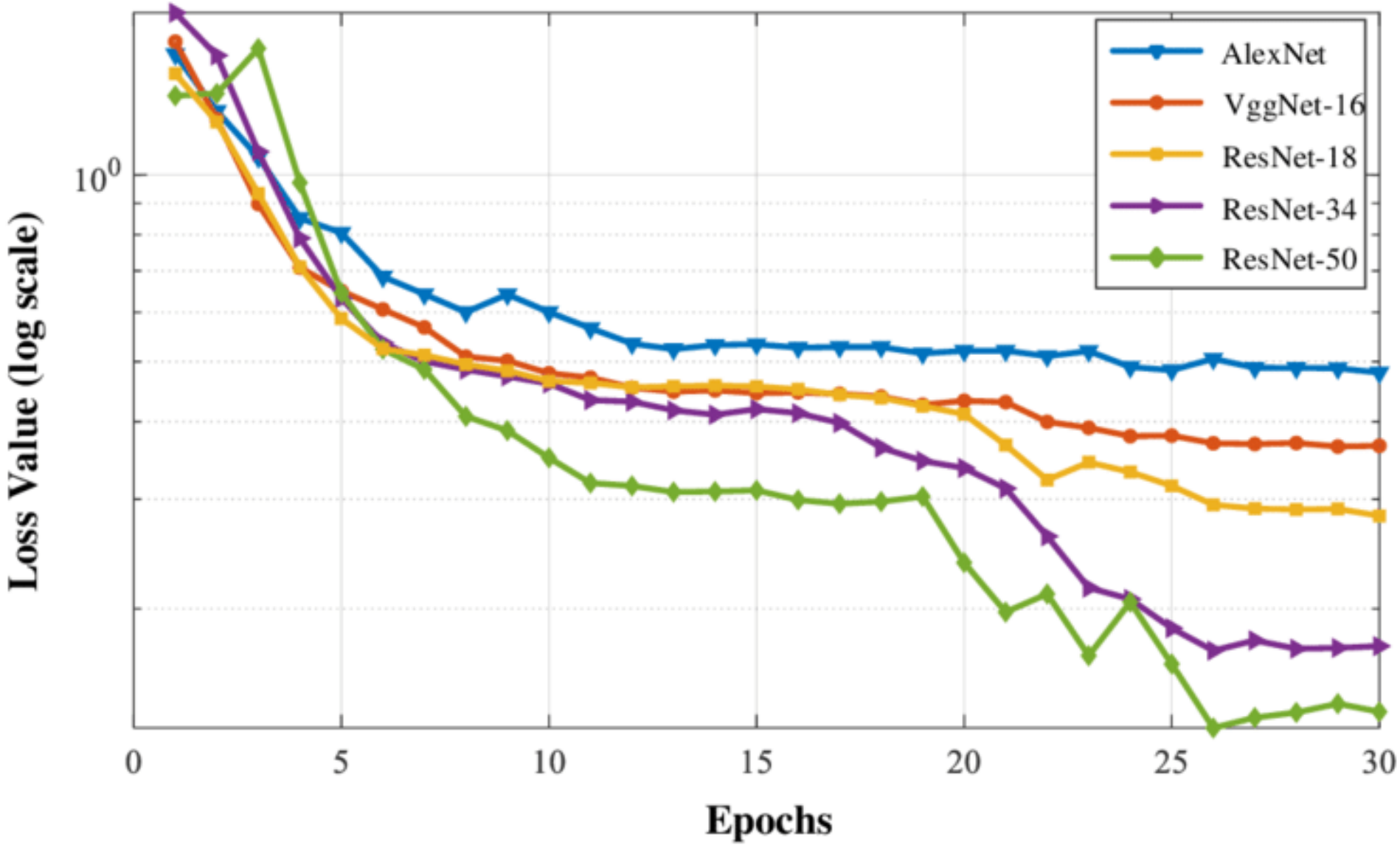






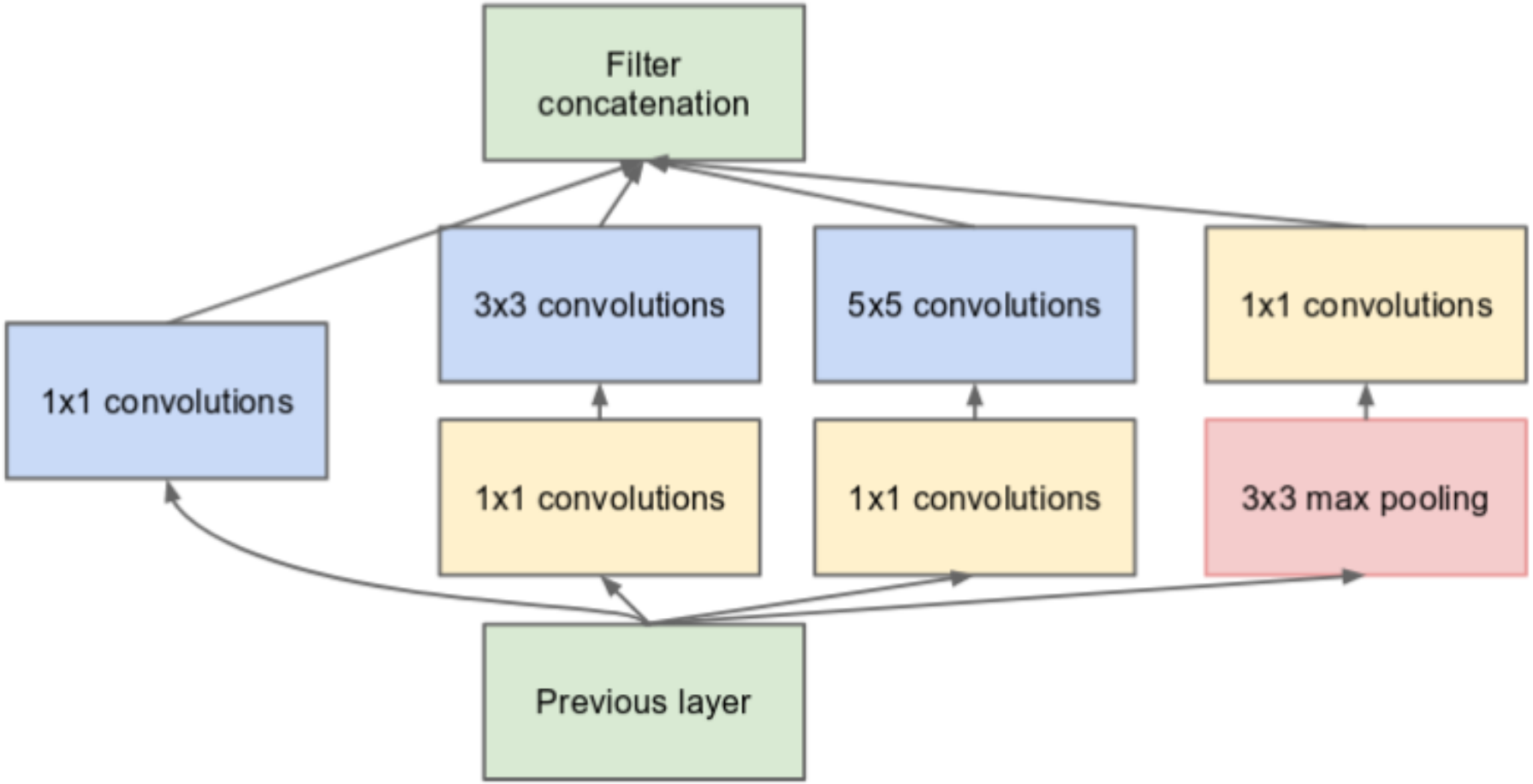
ResNet

He, Kaiming, et al. "Deep residual learning for image recognition."



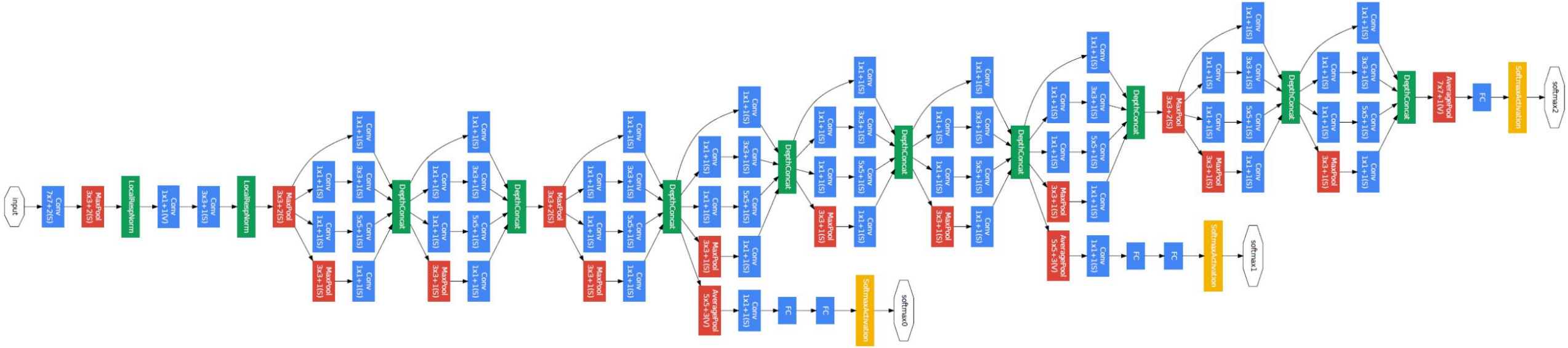
He, Kaiming, et al. "Deep residual learning for image recognition."

ResNet



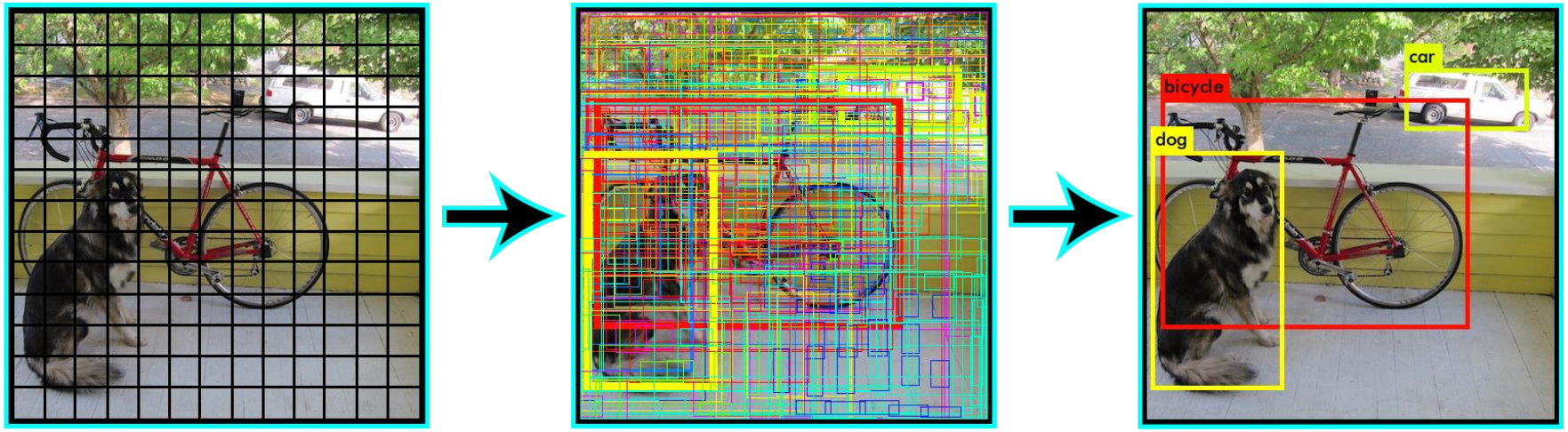
Inception module

Szegedy, Christian, et al. "Going deeper with convolutions."

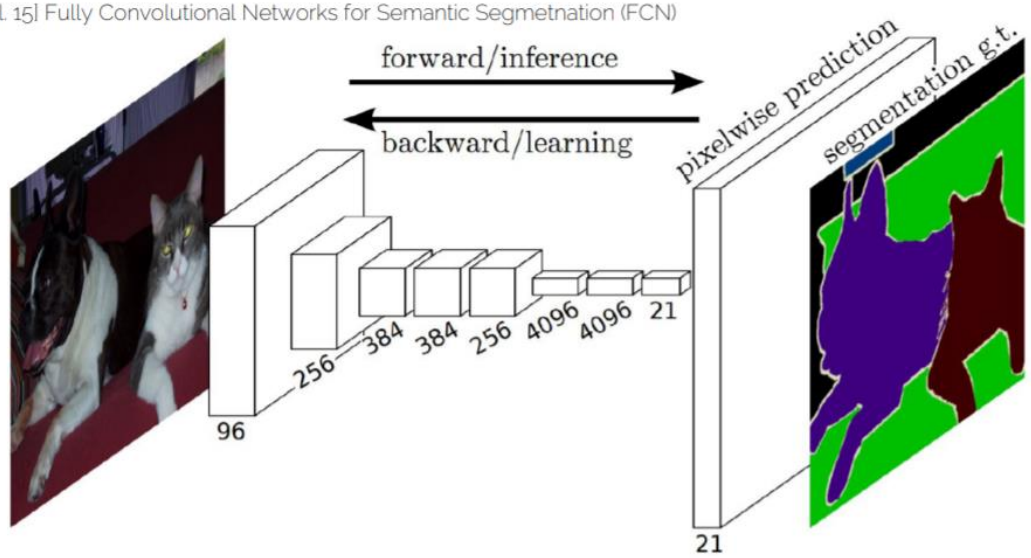


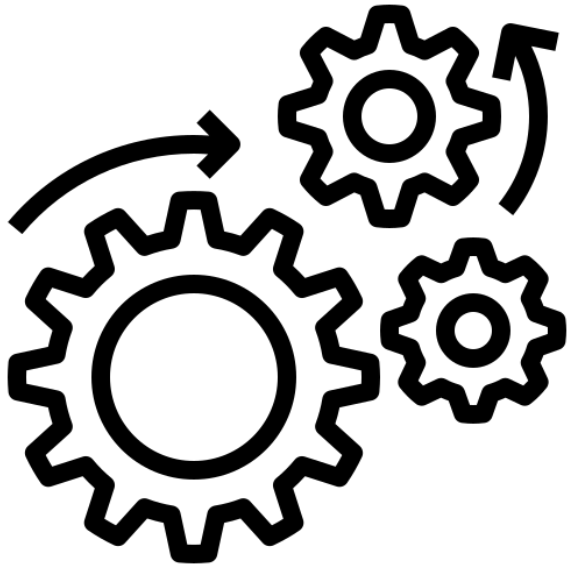
GoogLeNet

Szegedy, Christian, et al. "Going deeper with convolutions."



[Long et al. 15] Fully Convolutional Networks for Semantic Segmetnation (FCN)





Applications

Redmon, Joseph, et al. "You only look once: Unified, real-time object detection."
Long, et al "Fully convolutional networks for semantic segmentation."