Convolutional Neural Networks

Dripta Mj
Department of Mathematics
Ramakrishna Mission Vivekananda Educational and Research Institute
Belur Math, India

Object detection

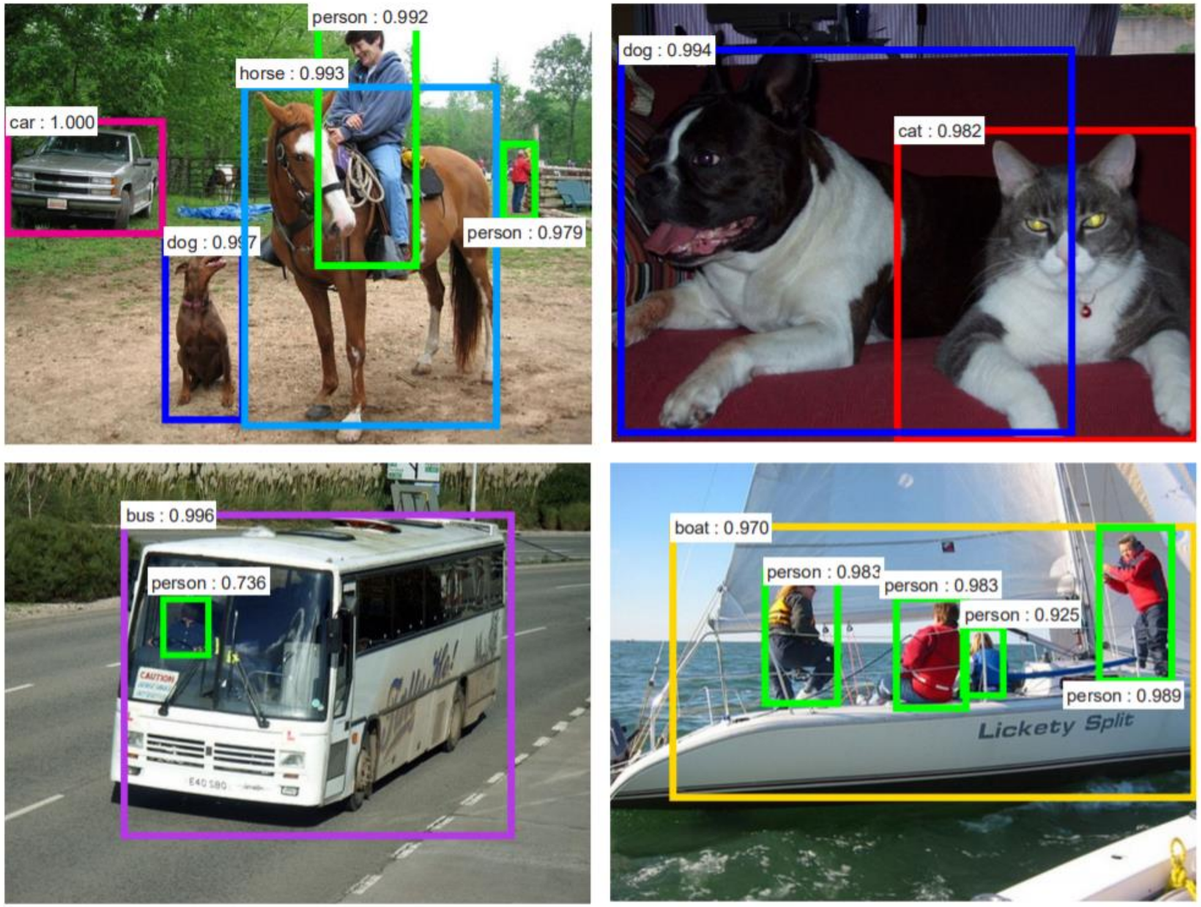


Figure source: Shaoqing, R. *et. al.*, Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, NIPS 2015.

Creativity

Figure source: Gatys, Ecker and Bethge, Image style transfer using convolutional neural networks, CVPR 2016.

Introduction to CNN

- CNN: Neural networks using convolution in place of general matrix multiplication in at least one layers.
- Neural network for grid-like topology.
- The name CNN indicate that the mathematical operation **convolution** is used.

Convolution

- Convolution is the mathematical operation on two functions that produces a third function

$$g(t) = \int_{\tau=-\infty}^{\infty} x(\tau)k(t-\tau)\mathrm{d}\tau$$

- Example: If the input x is noisy, then k may considered as a weighting function so that the output $g(t)$ is a smoothed estimate at a particular location t.

- Usually the convolution operation is denoted as

$$g(t) = (x * k)(t)$$

- Terminology:
  - The first argument is usually referred to as the input.
  - The second argument is known as the kernel.
  - The output is sometimes called as the feature map.

Convolution

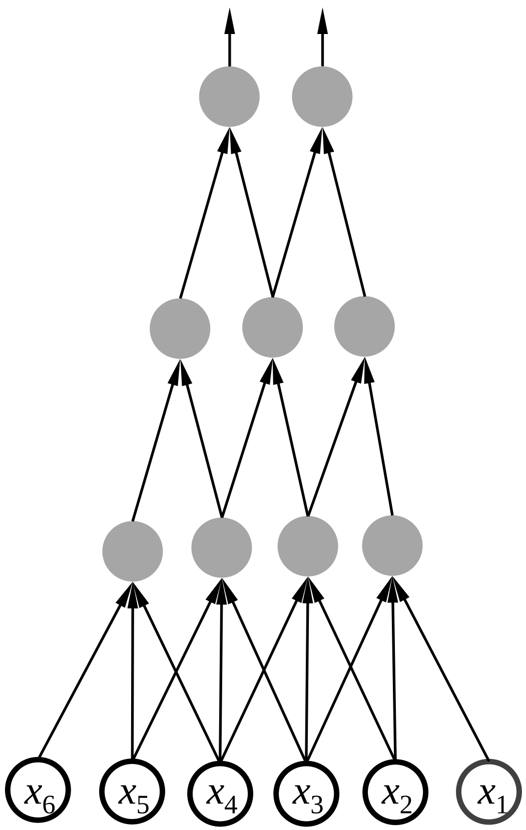
- In ML, we work on a discrete set of data points. For data sampled at regular intervals (say at integer values), the convolution can be defined as

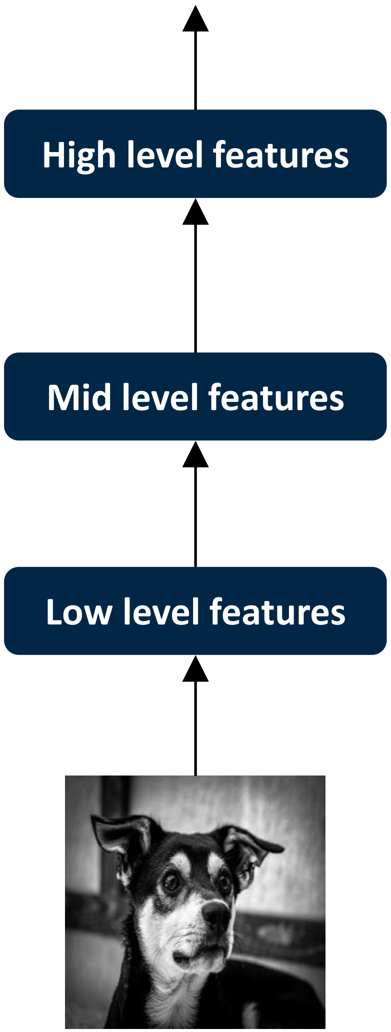
$$g(t) = (x * k)(t) = \sum_{\tau=-\infty}^{\infty} x(\tau)k(t-\tau)$$

- In many problems, the input dataset is usually a multidimensional array (tensor), and the kernel is a tensor of parameters that are learnt by the algorithm.
- In practice, the inputs and the kernel are finite dimensional, and so their values are taken to be 0 outside a finite set of points.
- The convolution can then be implemented as finite summation.
- Convolution can be implemented on more than one dimension simultaneously.
- Consider 2D image as the input $\mathbf{x}$.
- Using a 2D kernel, the convolution can then be written as

$$g(i,j) = (x * k)(i,j) = \sum_{\alpha}\sum_{\beta} \mathbf{x}(\alpha,\beta)k(i-\alpha, j-\beta)$$

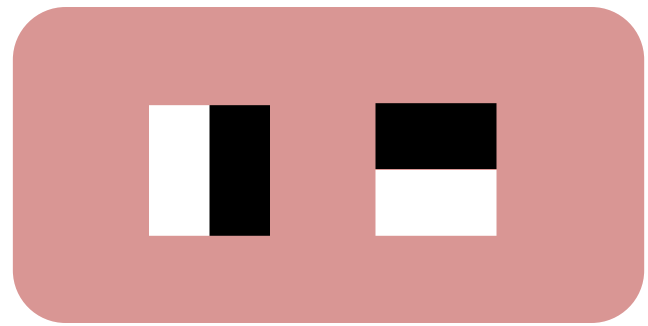
Intuition


x_6
x_5
x_4
x_3
x_2
x_1

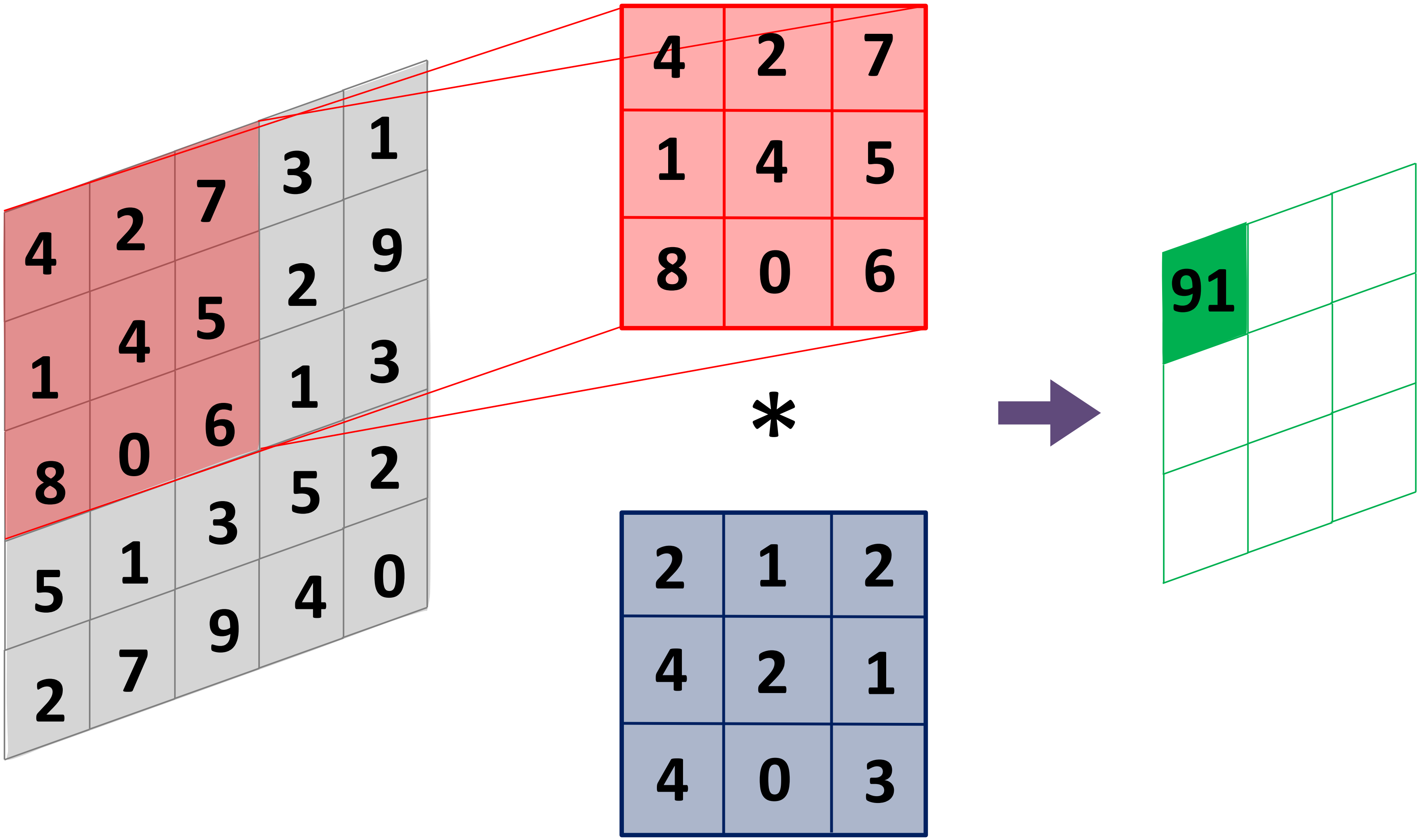

High level features
Mid level features
Low level features

Intuition

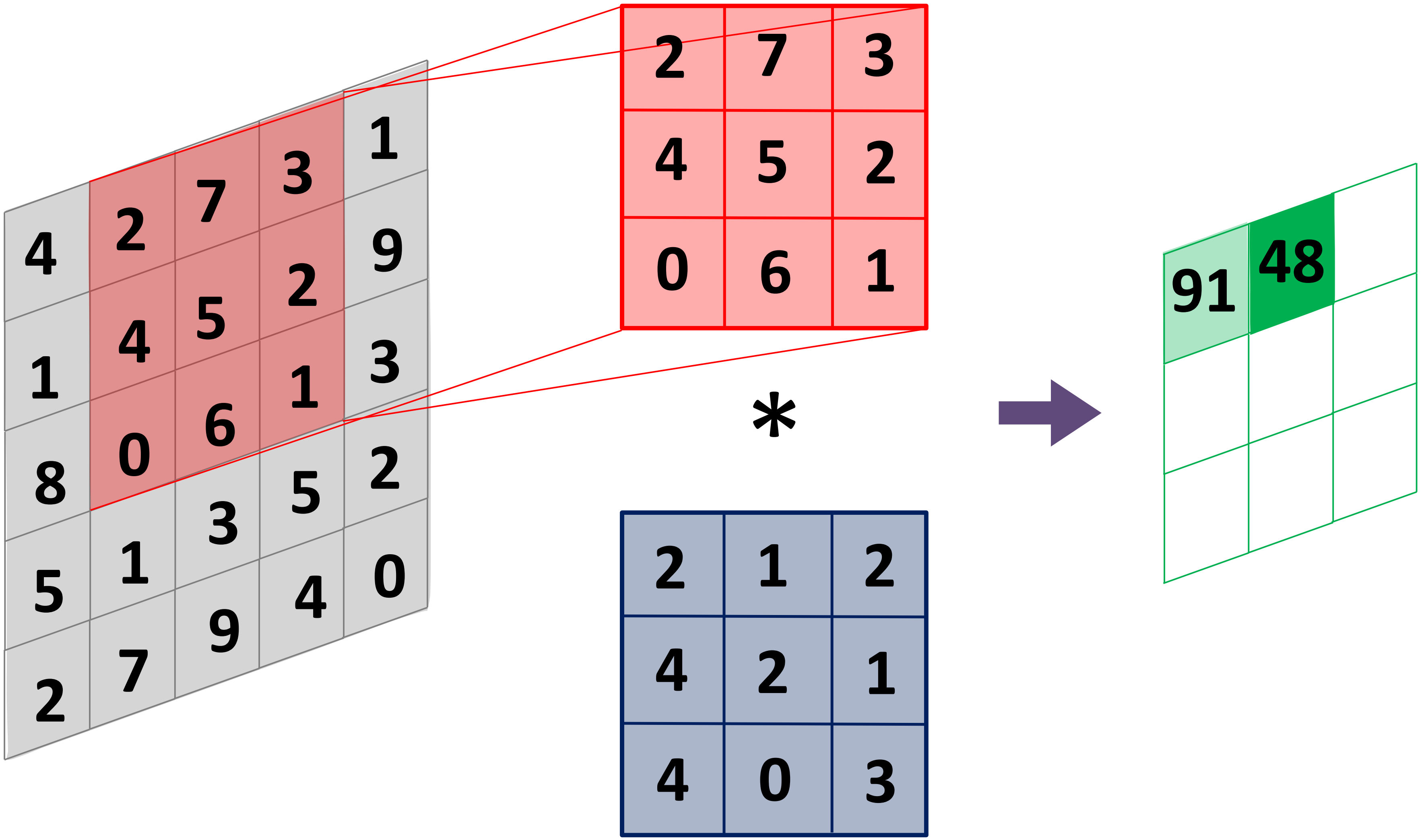
Low-level Filters



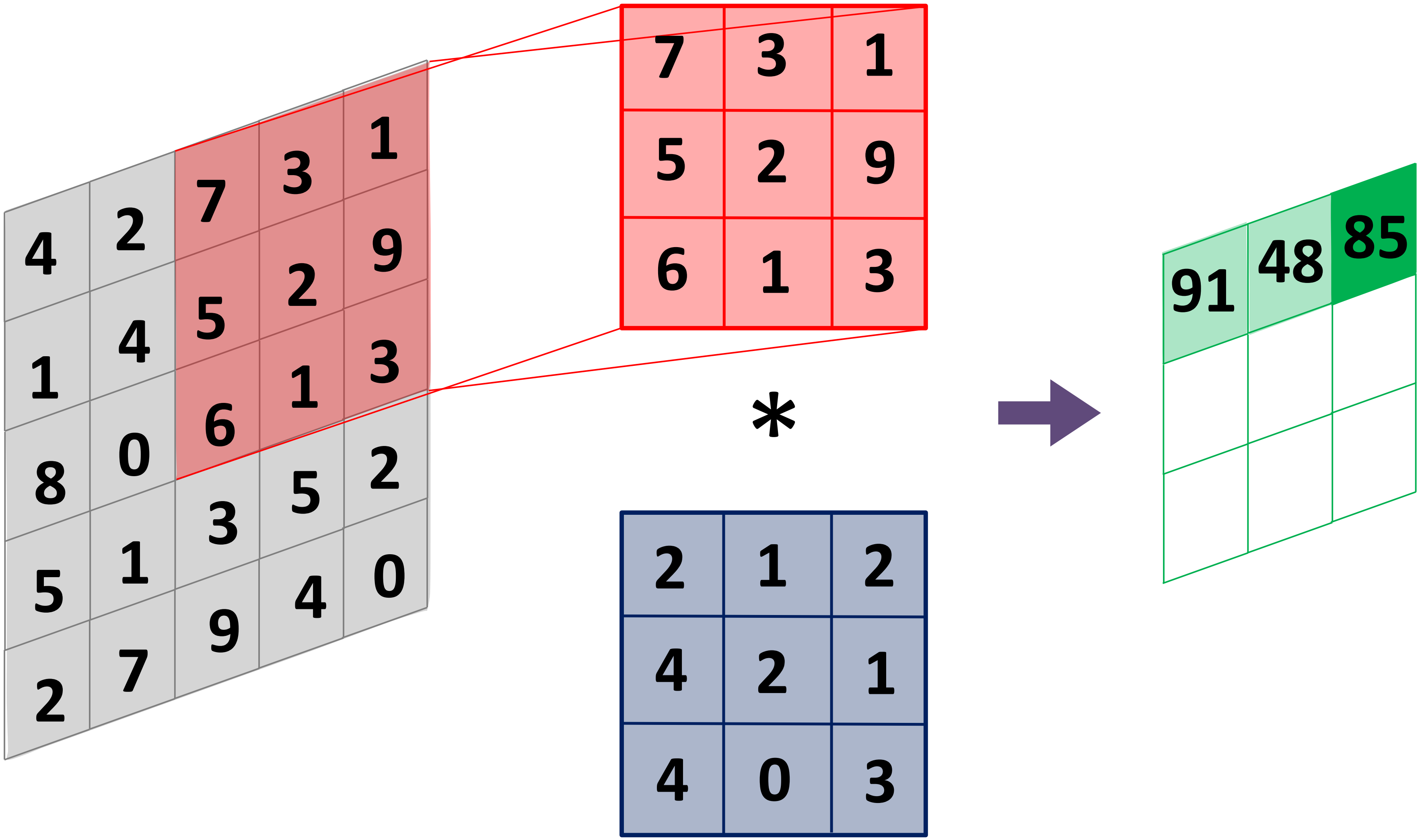
Convolution



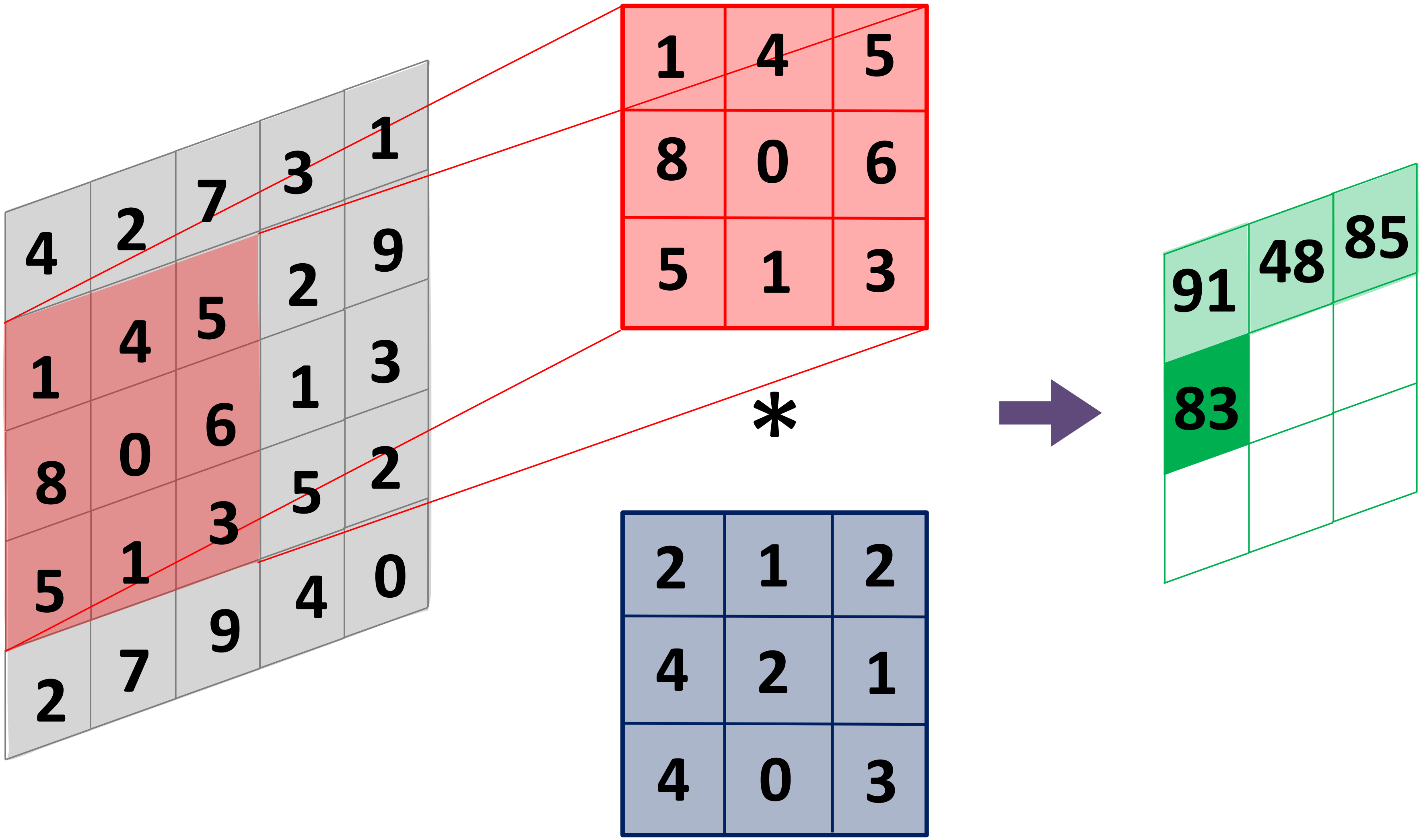
Convolution



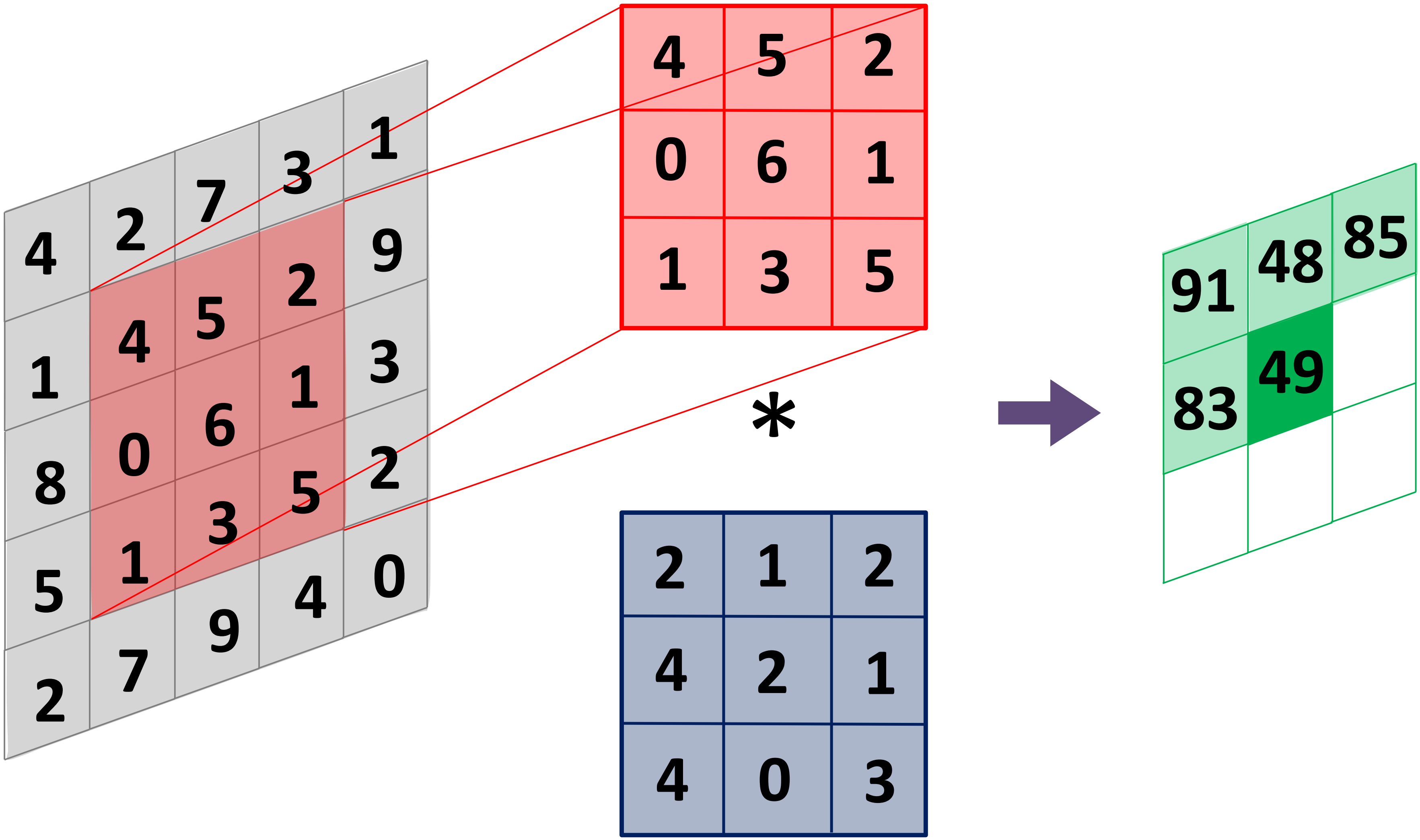
Convolution



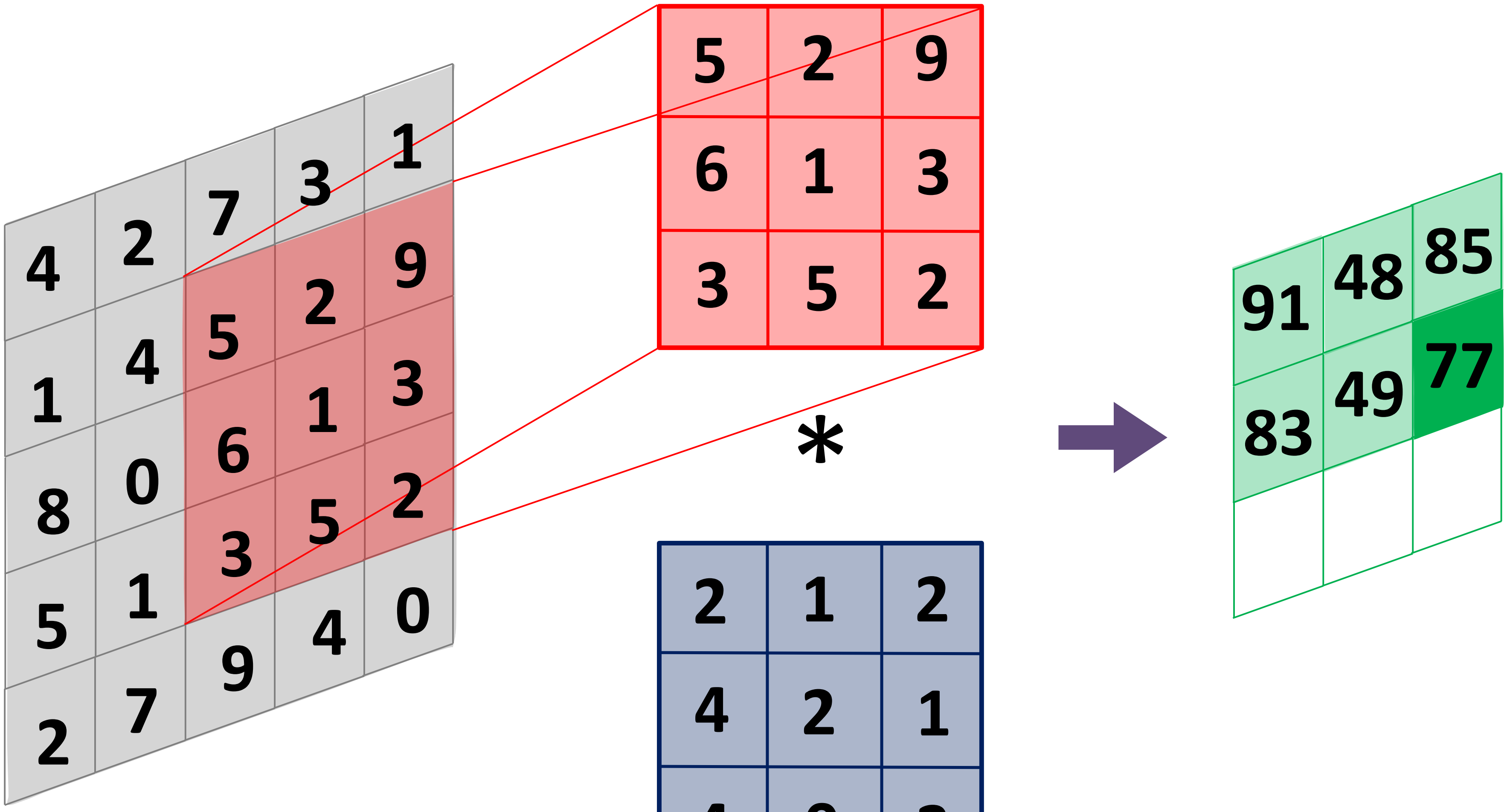
Convolution



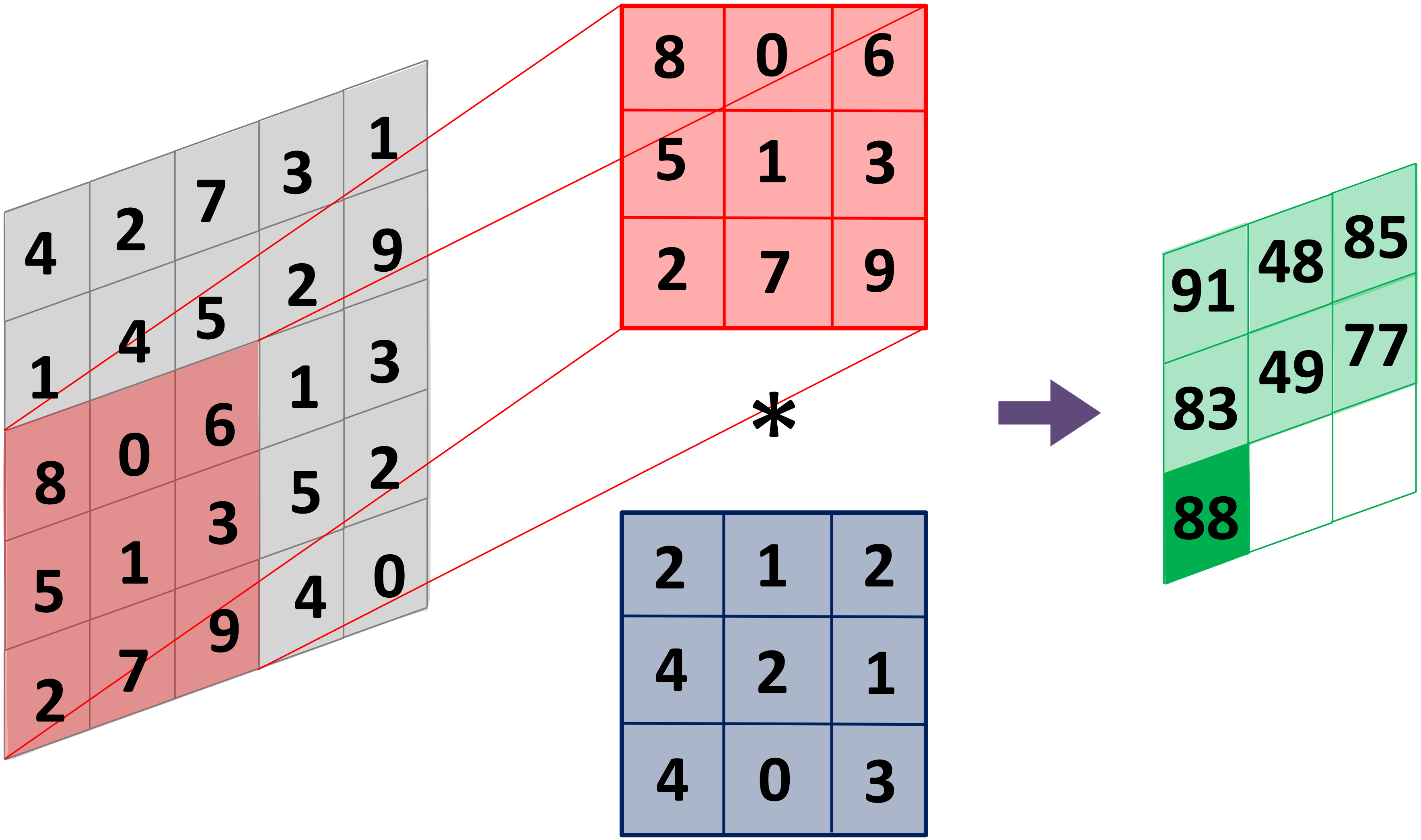
Convolution



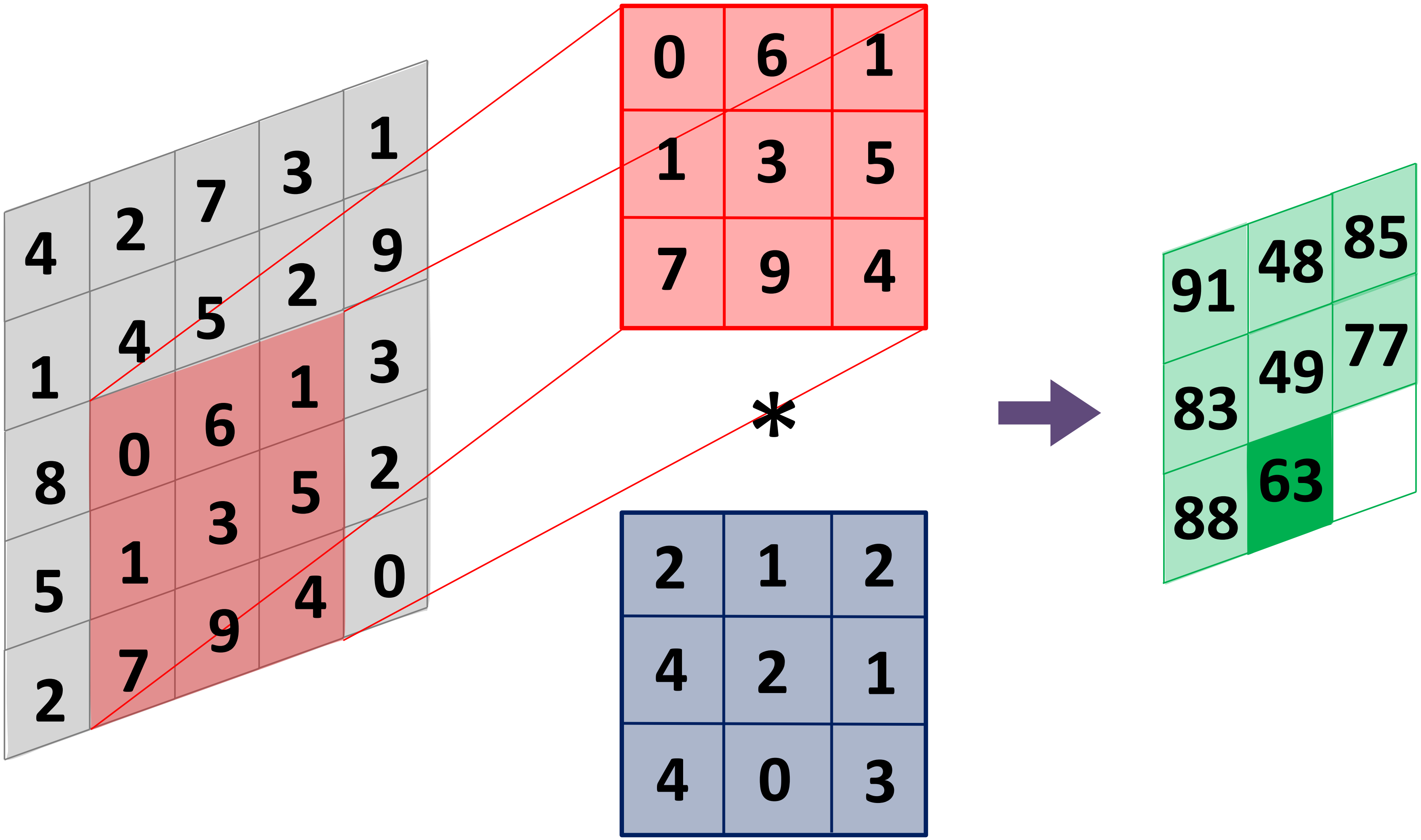
Convolution



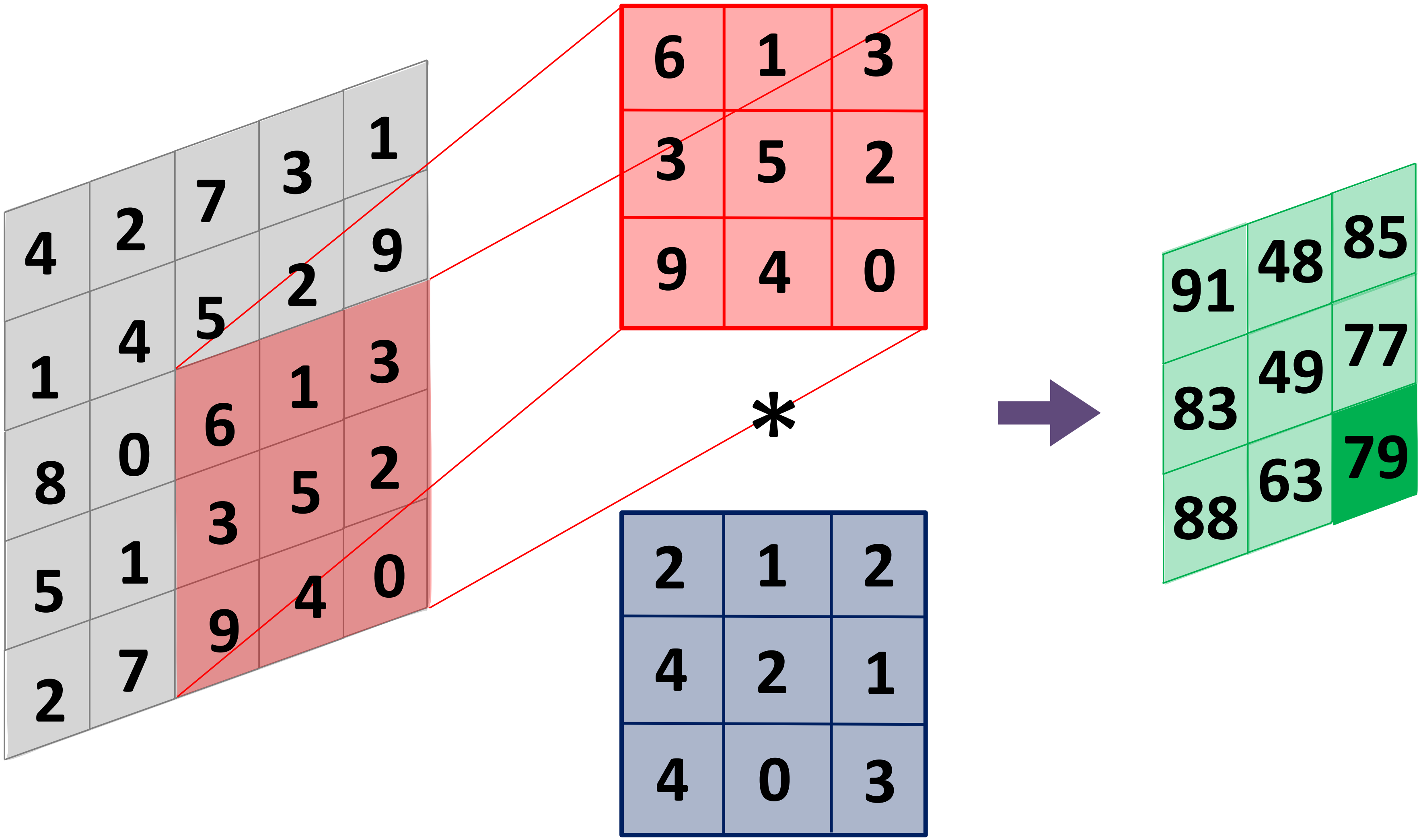
Convolution



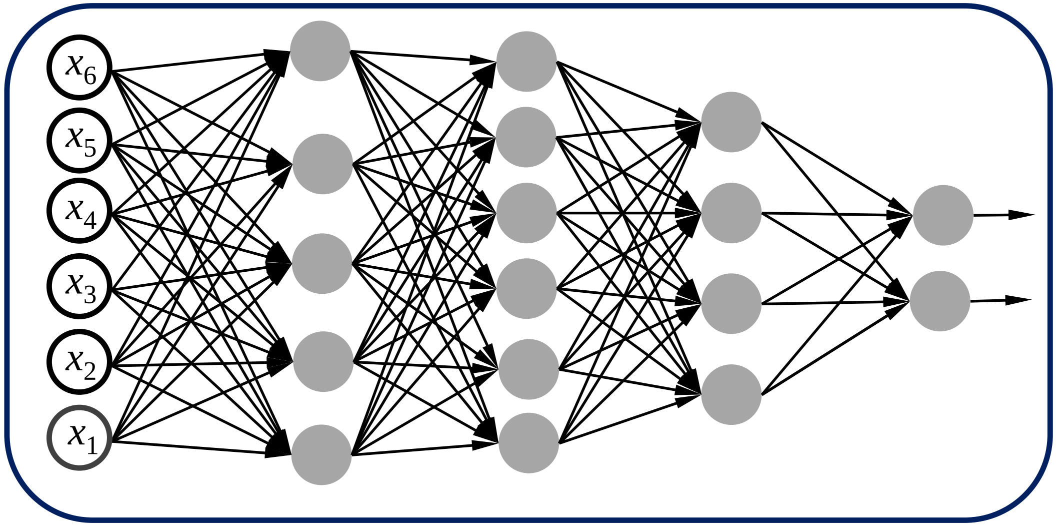
Convolution

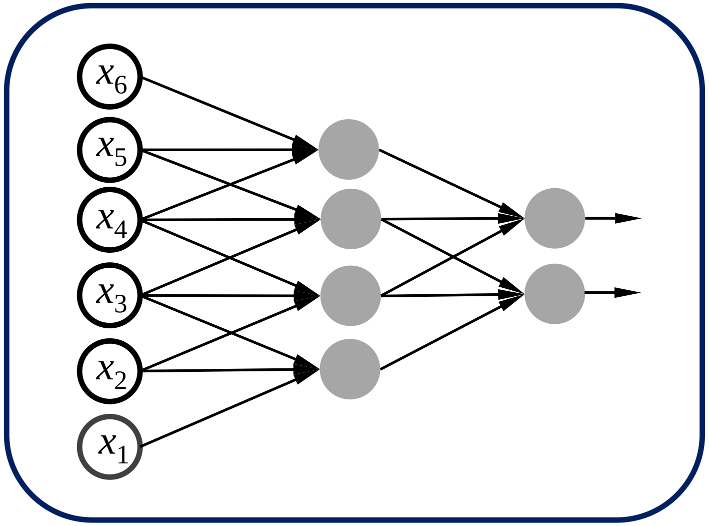


Convolution



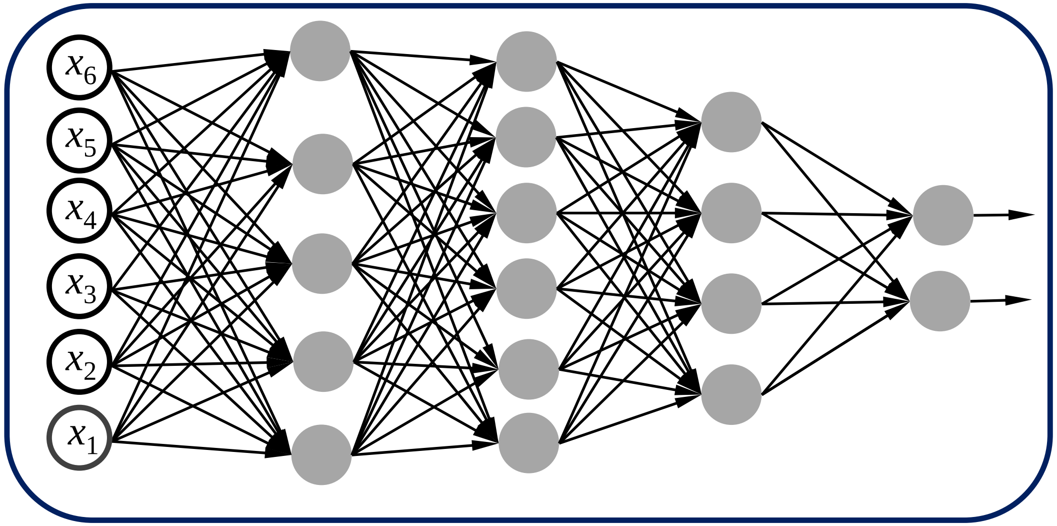
Key aspects

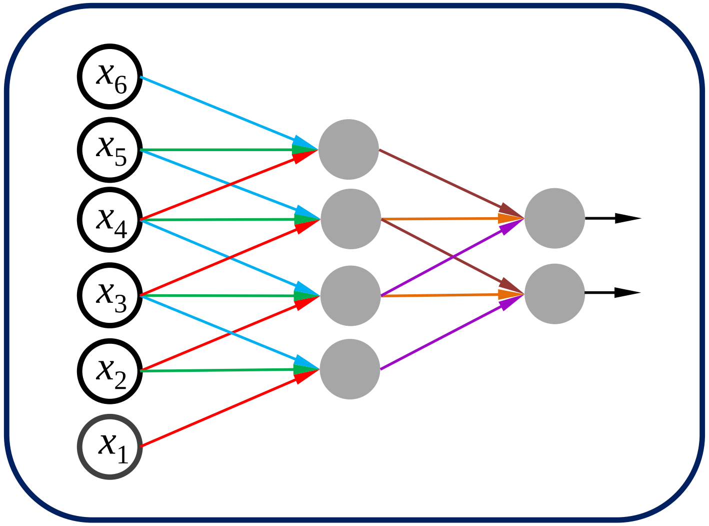




- Sparse connectivity:
  - Traditional neural network layers have a separate parameter for interaction between each input and output unit.
  - In CNN, kernels are used which have size smaller than that of the inputs.
  - Therefore the number of parameters are much less.
  - Fewer operations are required to compute the outputs.

Key aspects



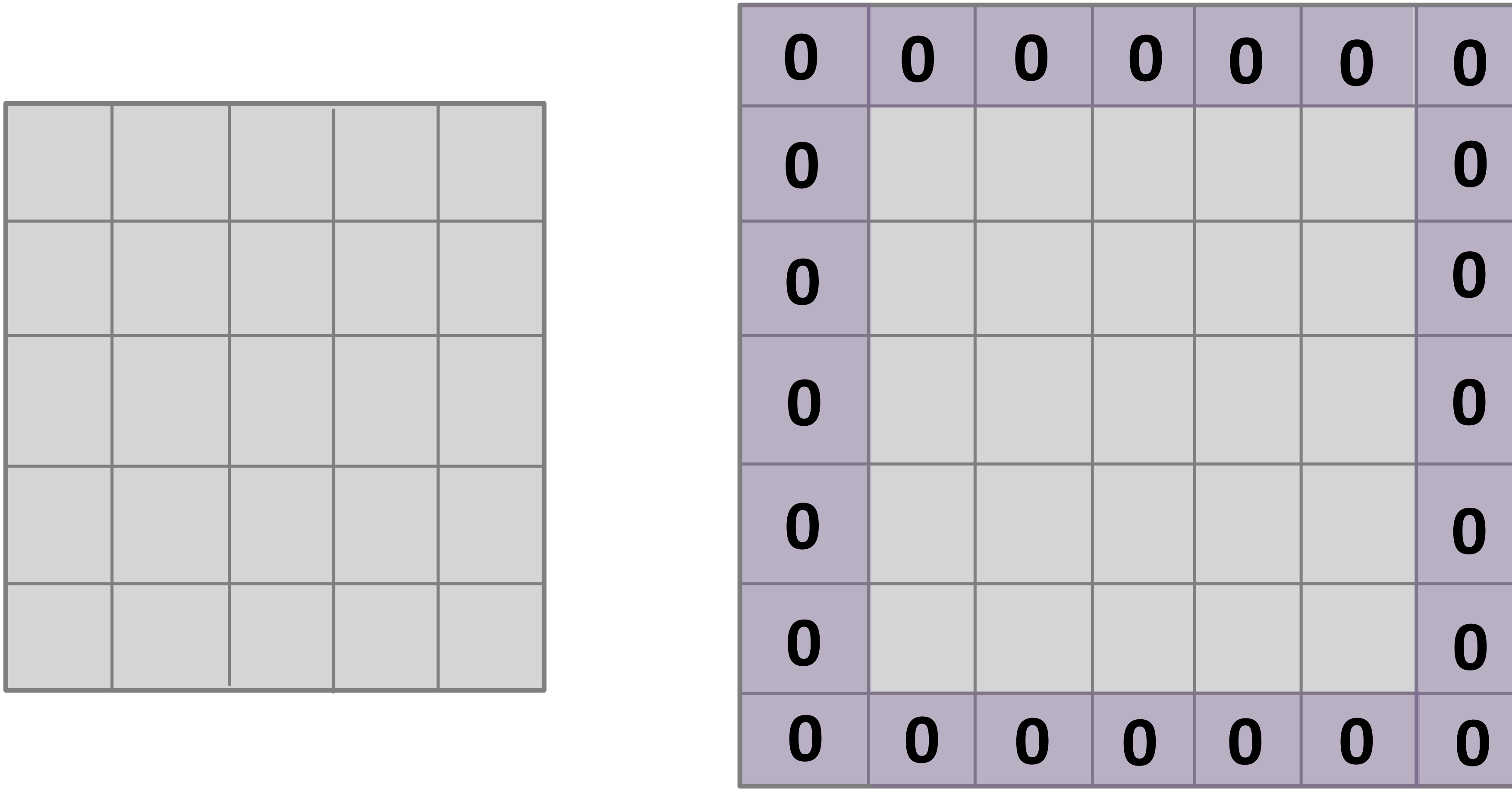


- Sharing of parameters:
  - In traditional neural networks, a parameter is used only once while evaluating the output of a unit.
  - In CNN, the same set of parameters are used for all the inputs.
  - Do not need to learn separate parameters for every input. Only need to learn the same set of parameters used for all the inputs

Key aspects

- Equivariance to translation:
  - If the location of a certain feature is changed, then the output of the convolution also changes accordingly.
  - Same features occur at multiple locations in the input space.
  - The output of the convolution indicate where different features occur in the input space.
  - For example, an edge detecting filter will generate a 2D map of the occurrence of such an edge in the input.
  - Convolution is not equivariant to transformations such as scaling, rotation.

Padding

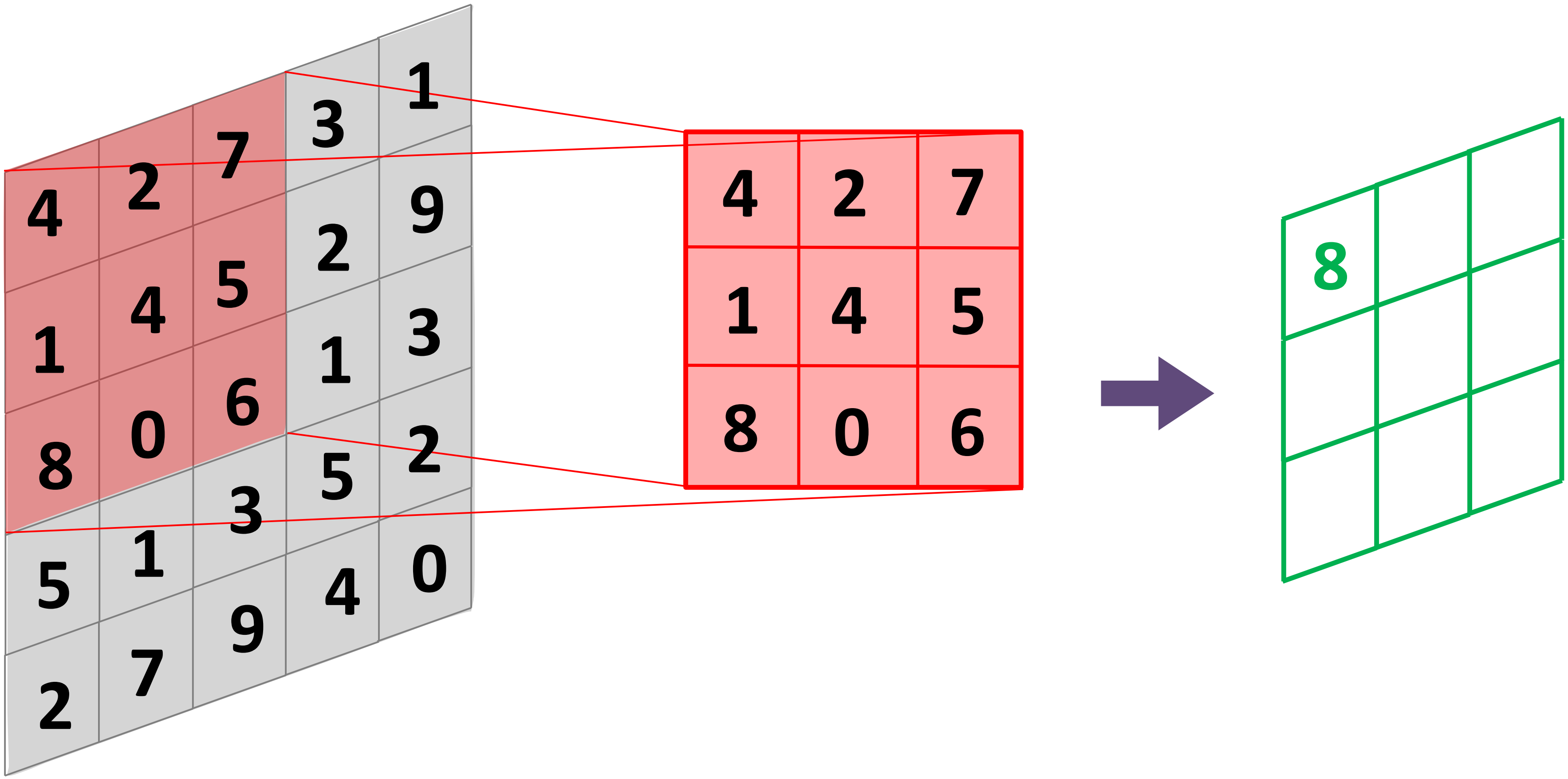


- Why padding?
- Risk losing information from the edges of inputs with no padding.
- Without padding in deep networks, the inputs to later layers will be significantly reduced in size.

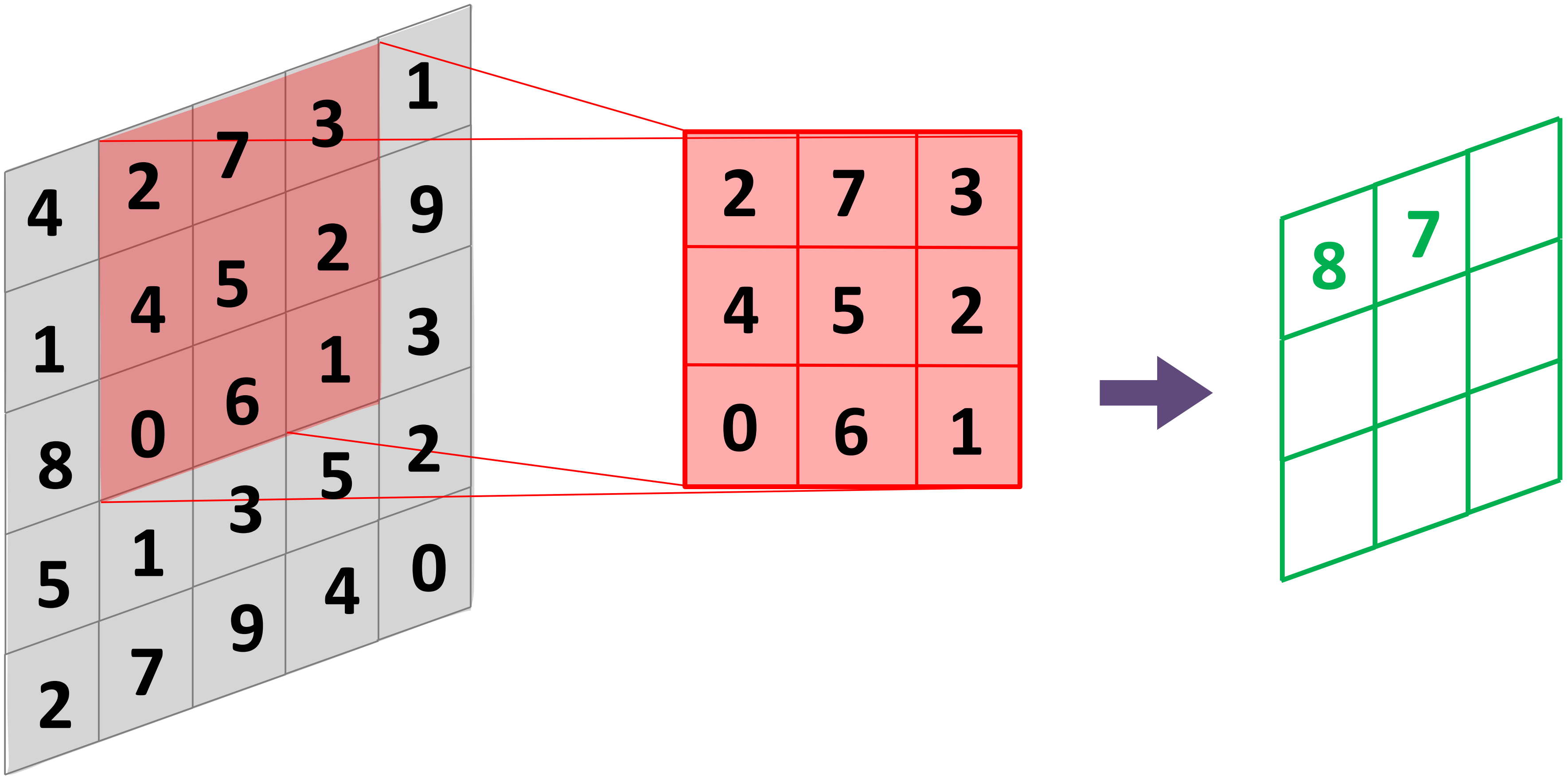
Pooling

- Replaces the output with a summary statistic of the nearby outputs.
- Motivation: Pooling assists in making a representation approximately invariant to small translations of the input.
- Pooling is useful as in many cases we are concerned about the presence of some feature rather than their exact location.
- Example:
  - Max pooling: Computes the maximum output within a rectangular neighborhood.
  - Average pooling: Average of a rectangular neighborhood.
  - L^2 norm of a rectangular neighborhood.

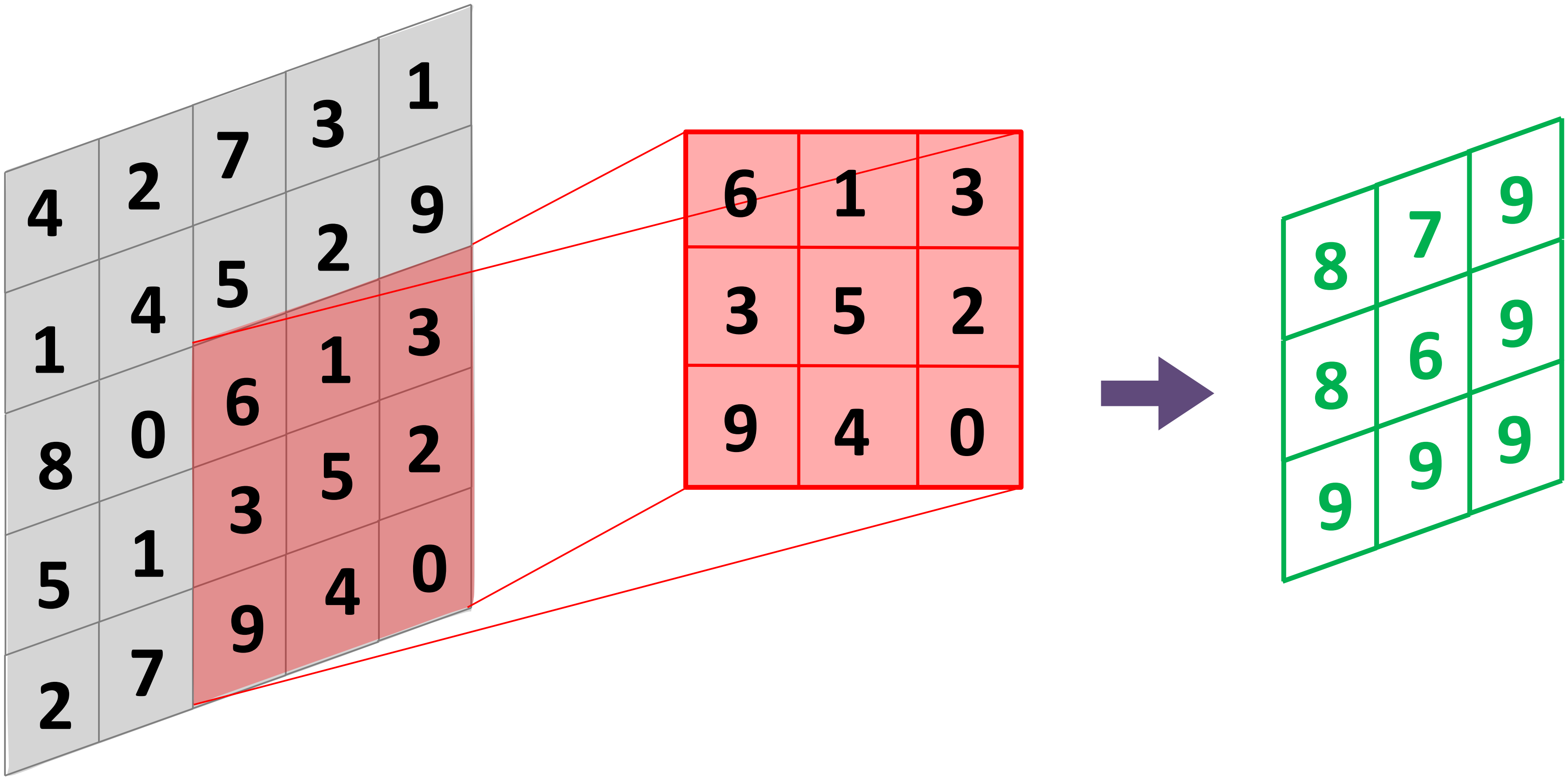
Max pooling



Max pooling

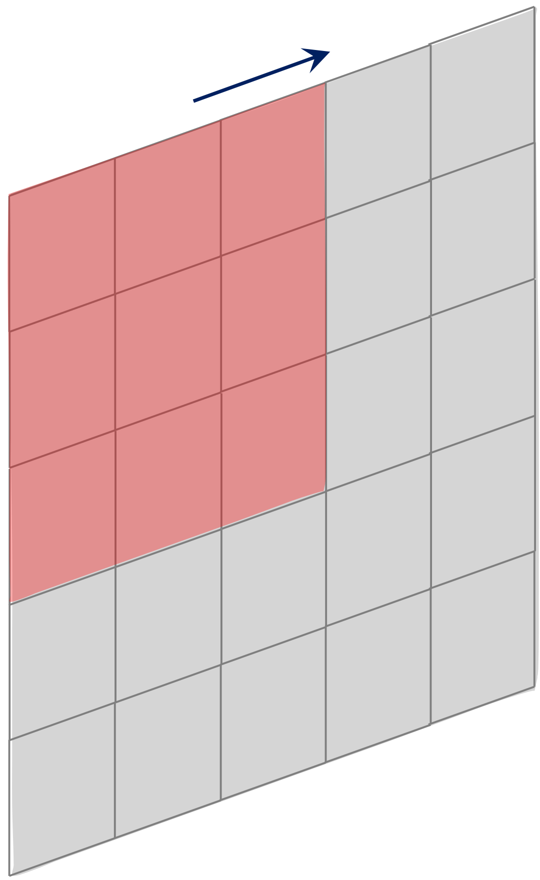


Max pooling



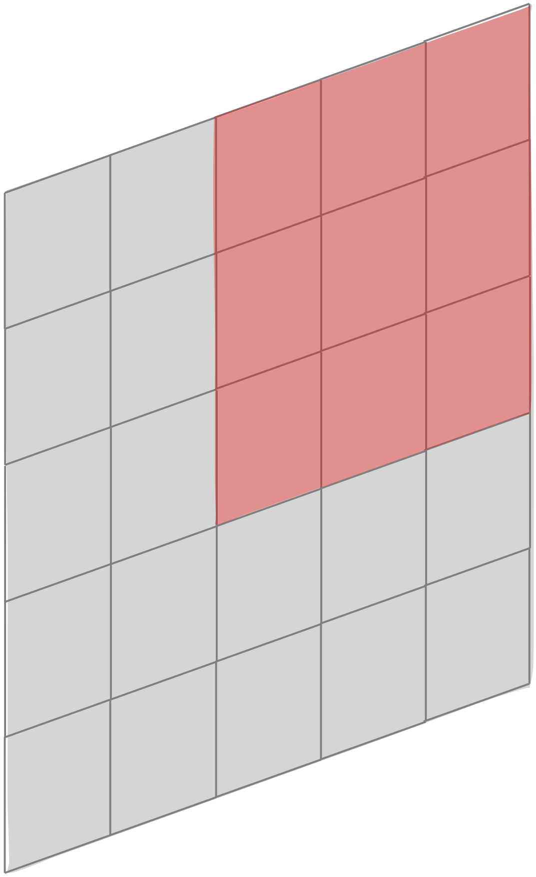
Stride

- Some positions of the kernel can be skipped to reduce the computational cost.
- Samples are taken every (say) s grid points in a particular direction.



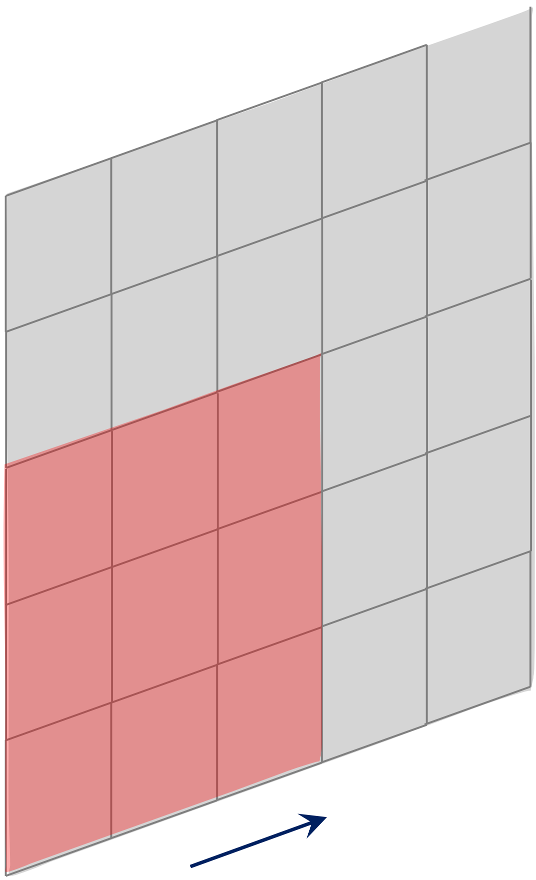
Stride

- Some positions of the kernel can be skipped to reduce the computational cost.
- Samples are taken every (say) s grid points in a particular direction.



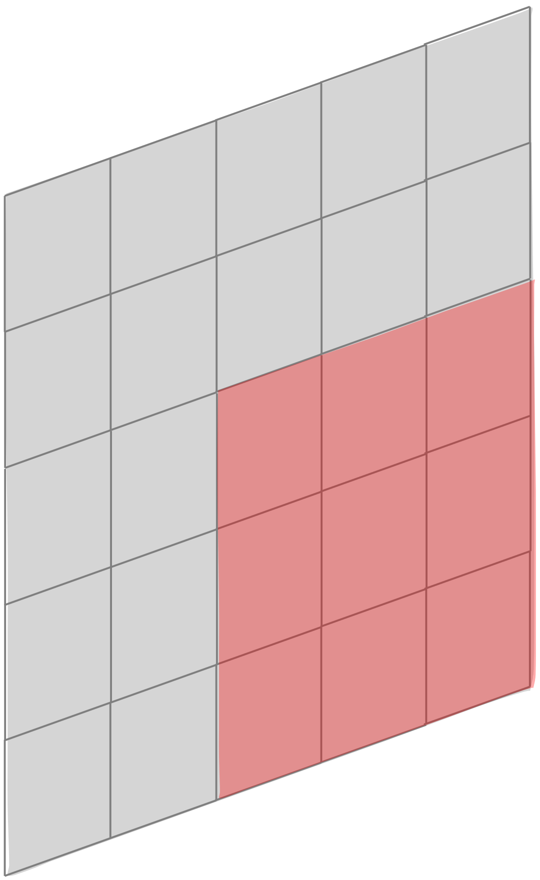
Stride

- Some positions of the kernel can be skipped to reduce the computational cost.
- Samples are taken every (say) s grid points in a particular direction.



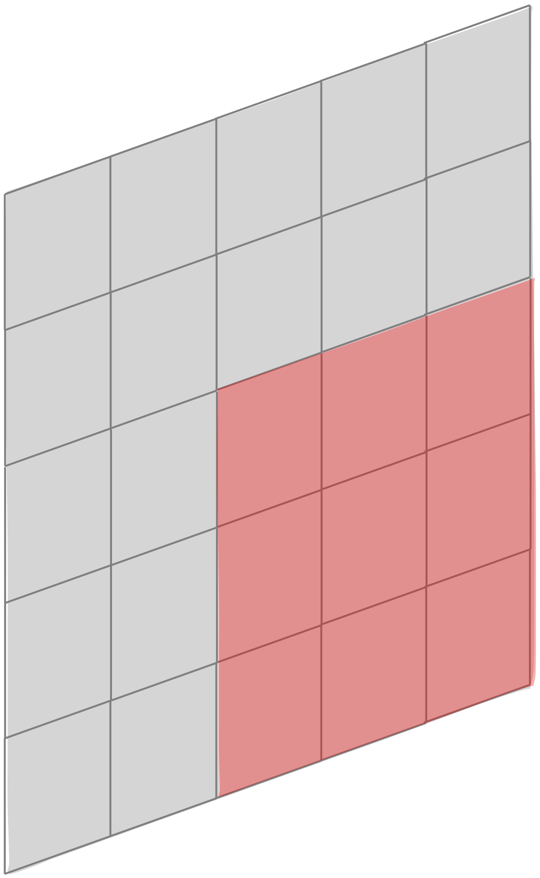
Stride

- Some positions of the kernel can be skipped to reduce the computational cost.
- Samples are taken every (say) s grid points in a particular direction.

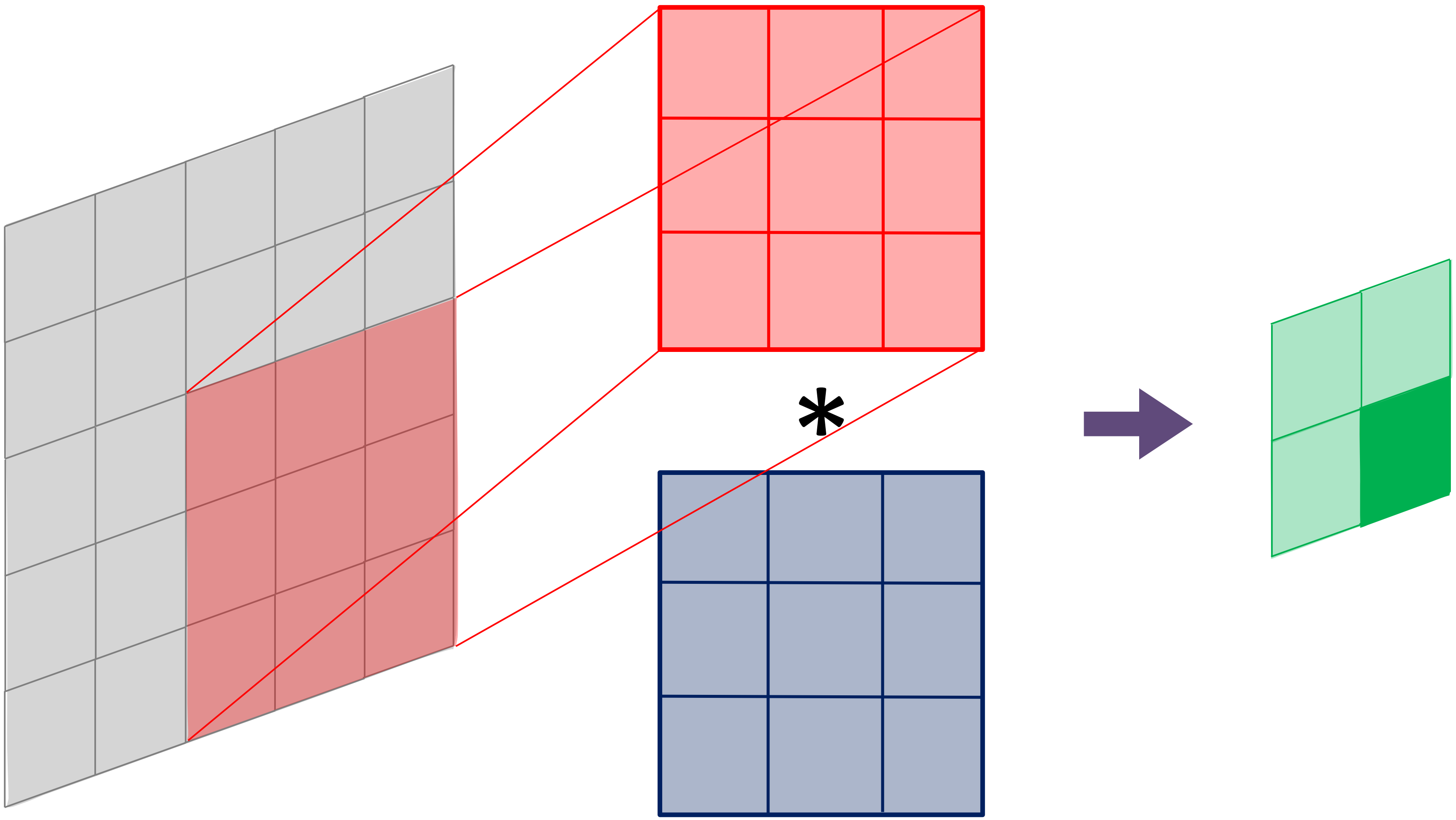


Stride

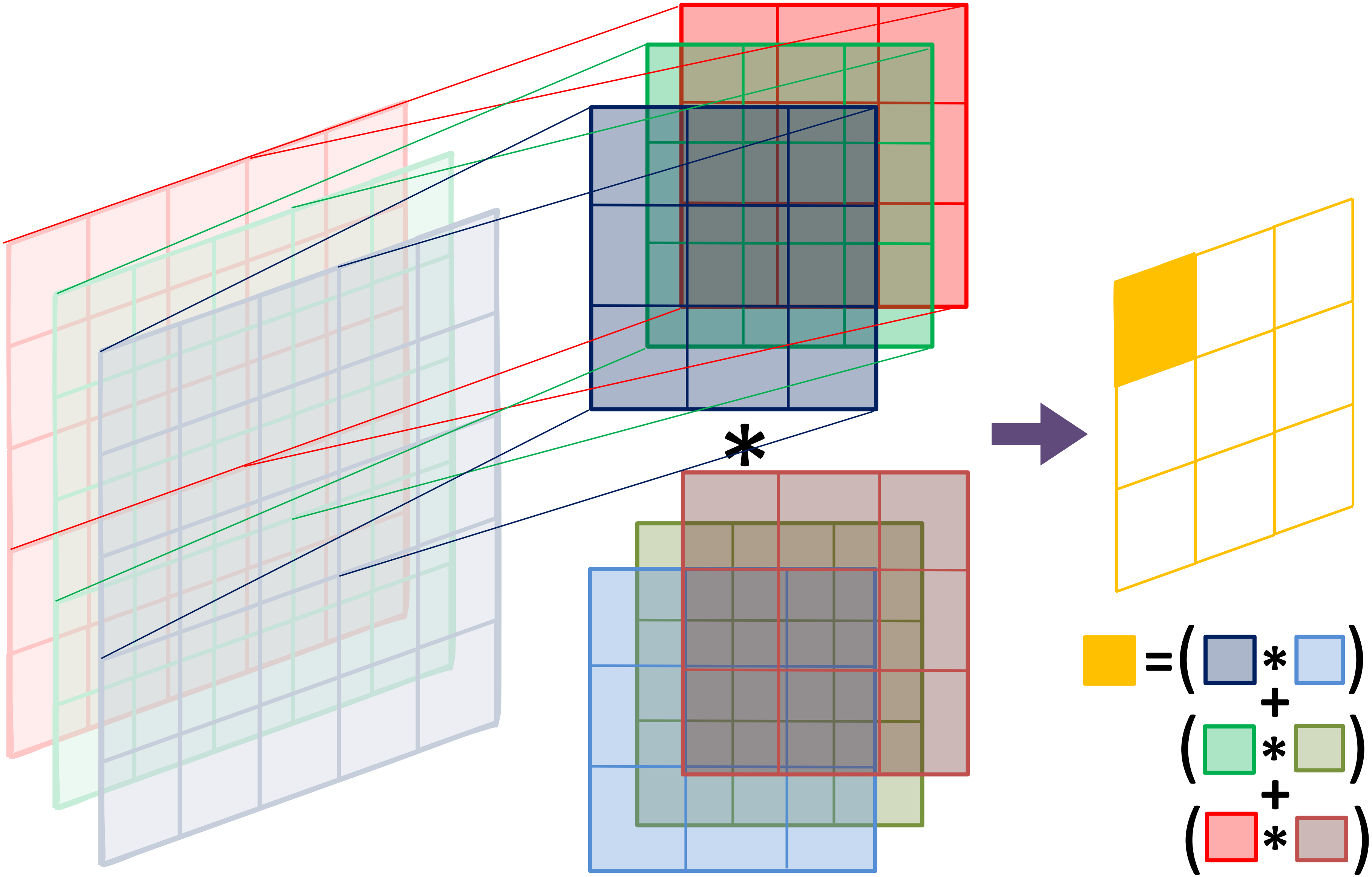
- s is referred to as the stride of the downsampled convolution.
- It is possible to define a separate stride s for each direction of motion.



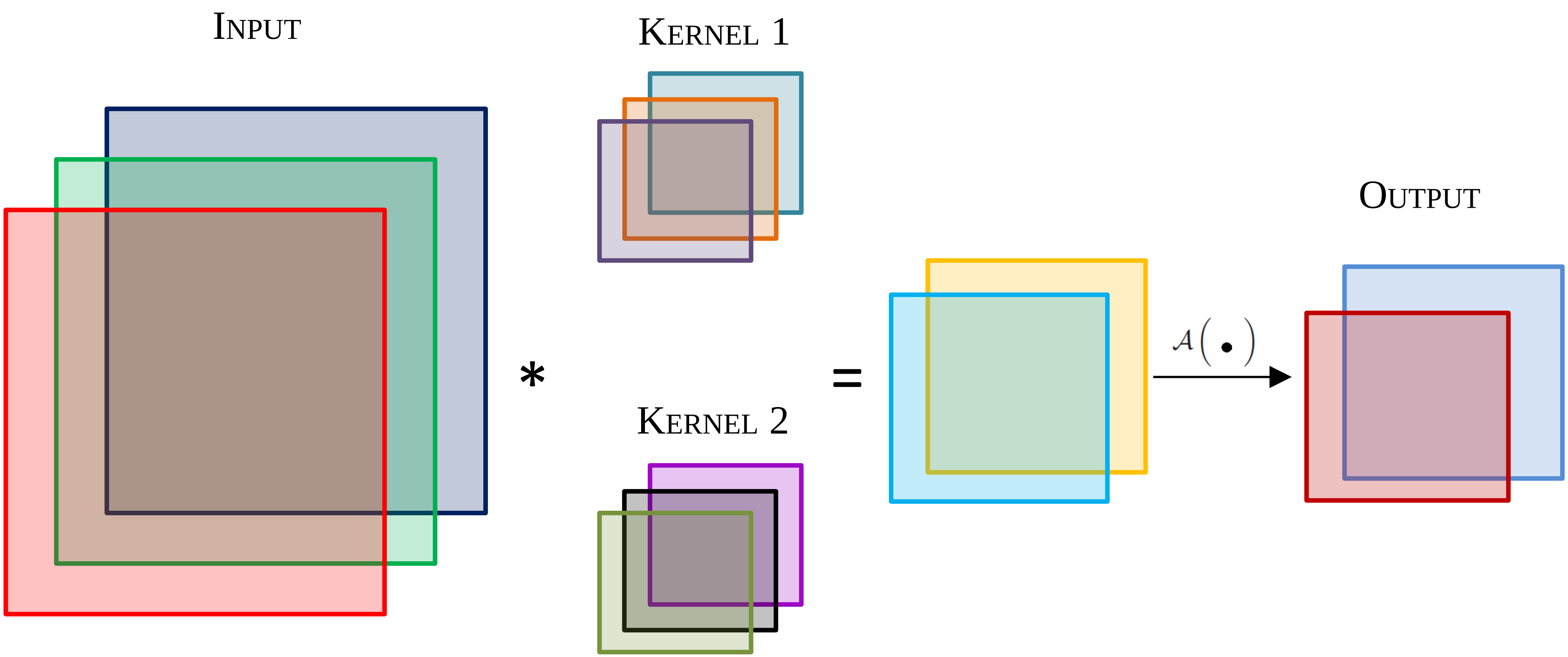
Stride



Multiple channels

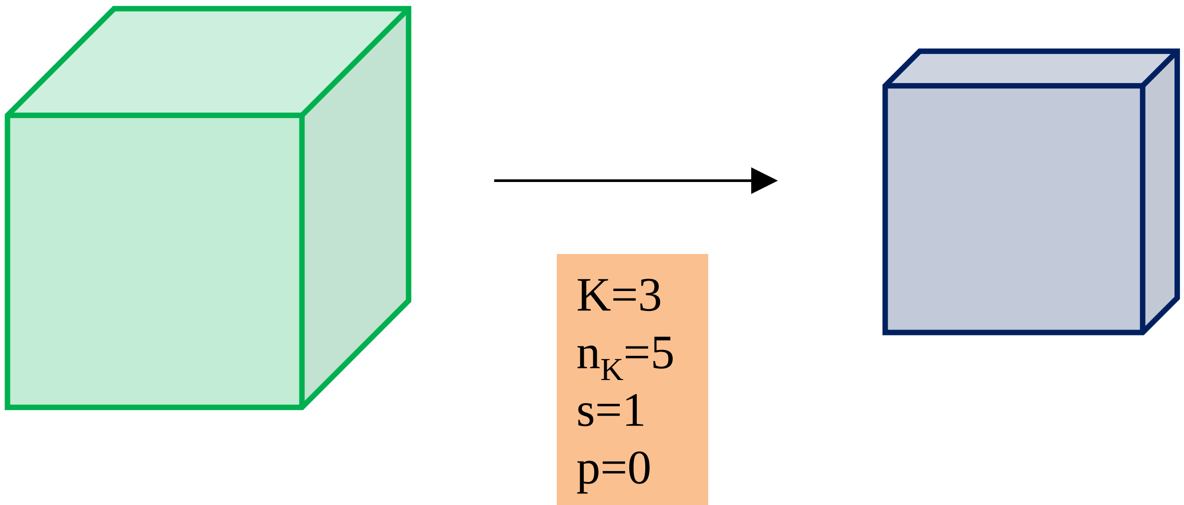


1 Convolution Layer



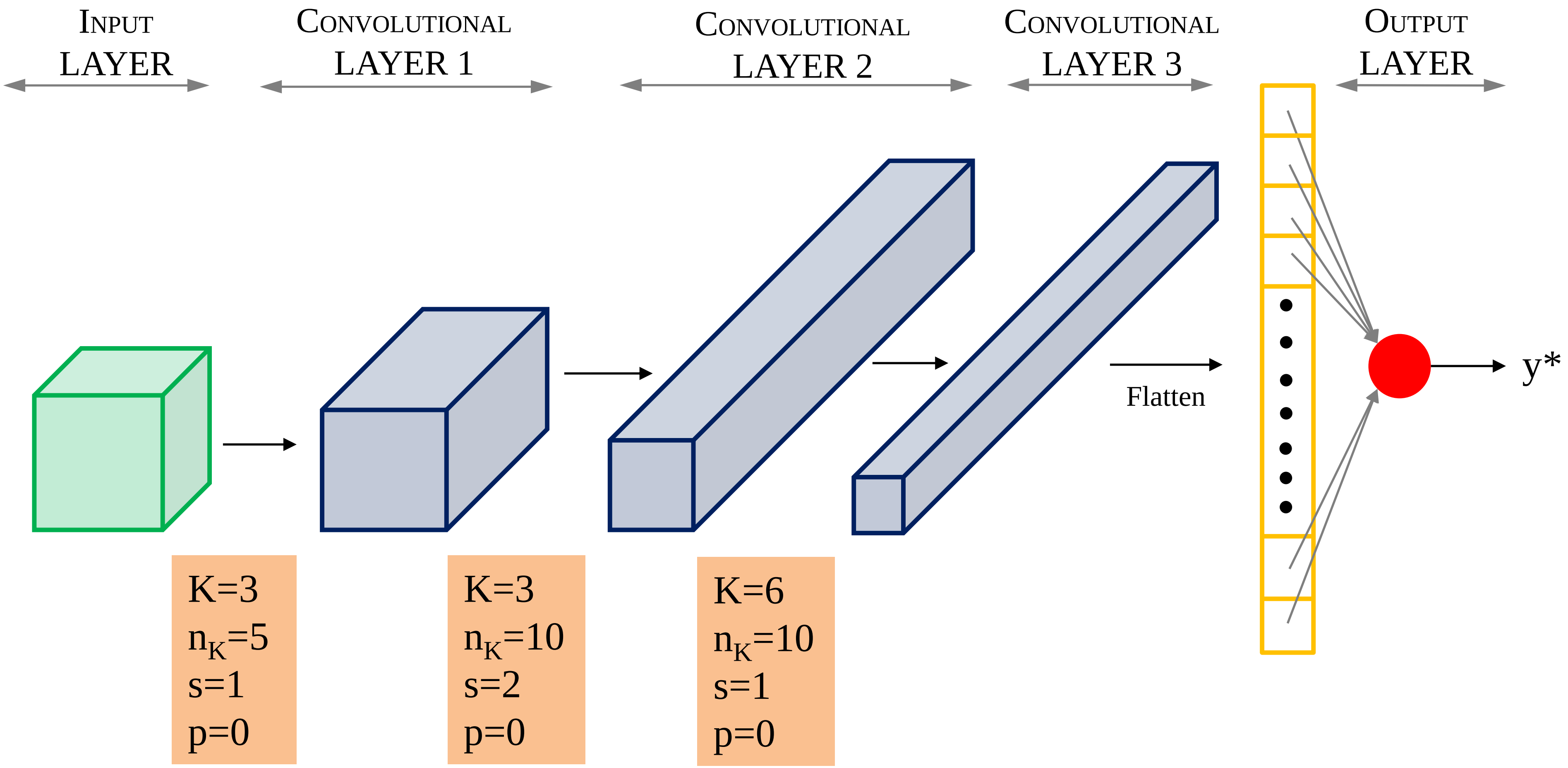
$$\mathcal{A}(\bullet) \longrightarrow \tanh(.)/\mathrm{RELU}(.)$$

Compact representation



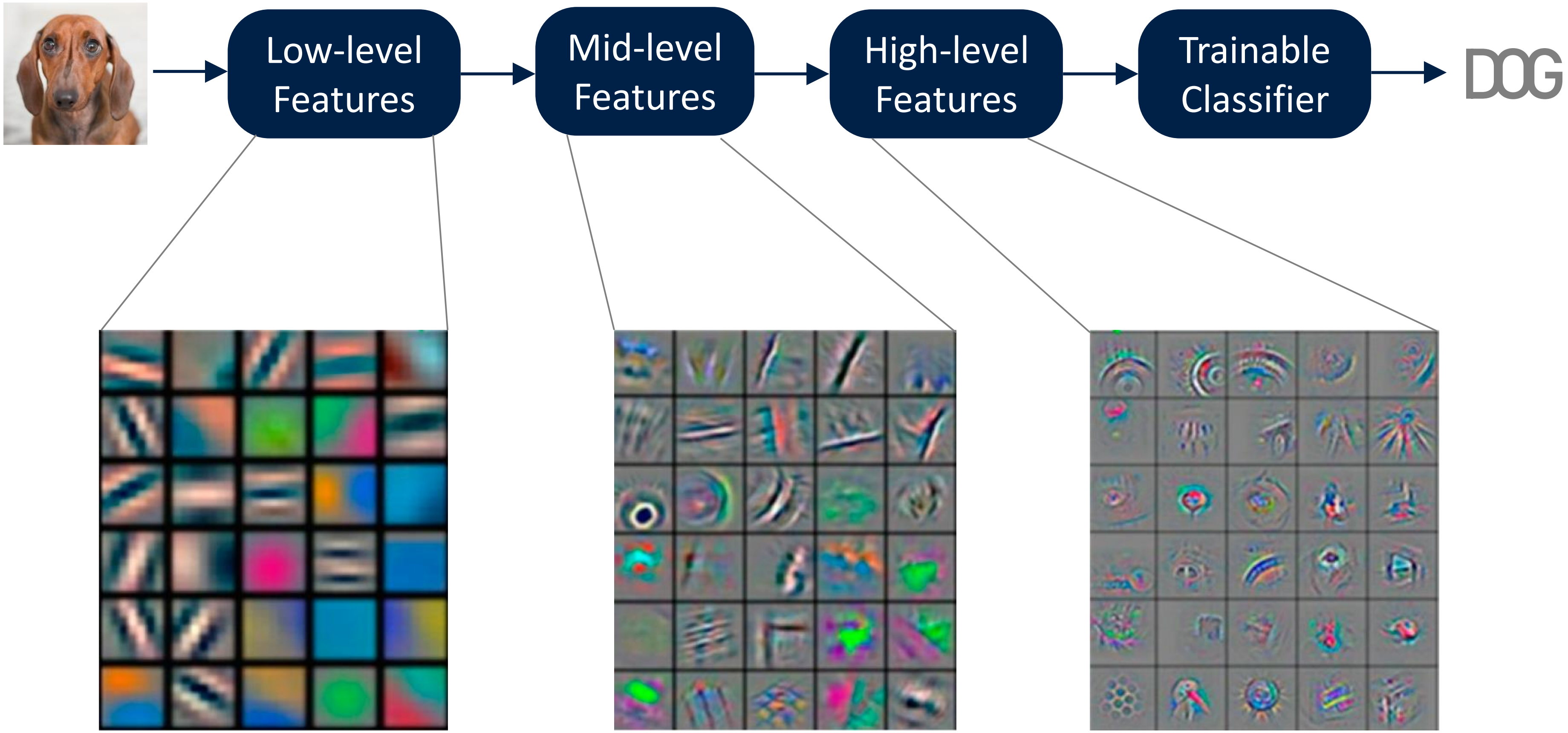
Size of the filter = K x K, n_K = Number of kernels, s = Stride, p = Padding

Sample network



Size of the filter = K x K, n_K = Number of kernels, s = Stride, p = Padding

Feature visualization



Zeiler and Fergus, Visualizing and Understanding Convolutional Networks, 2014

CNN ARCHITECTURES

AlexNet

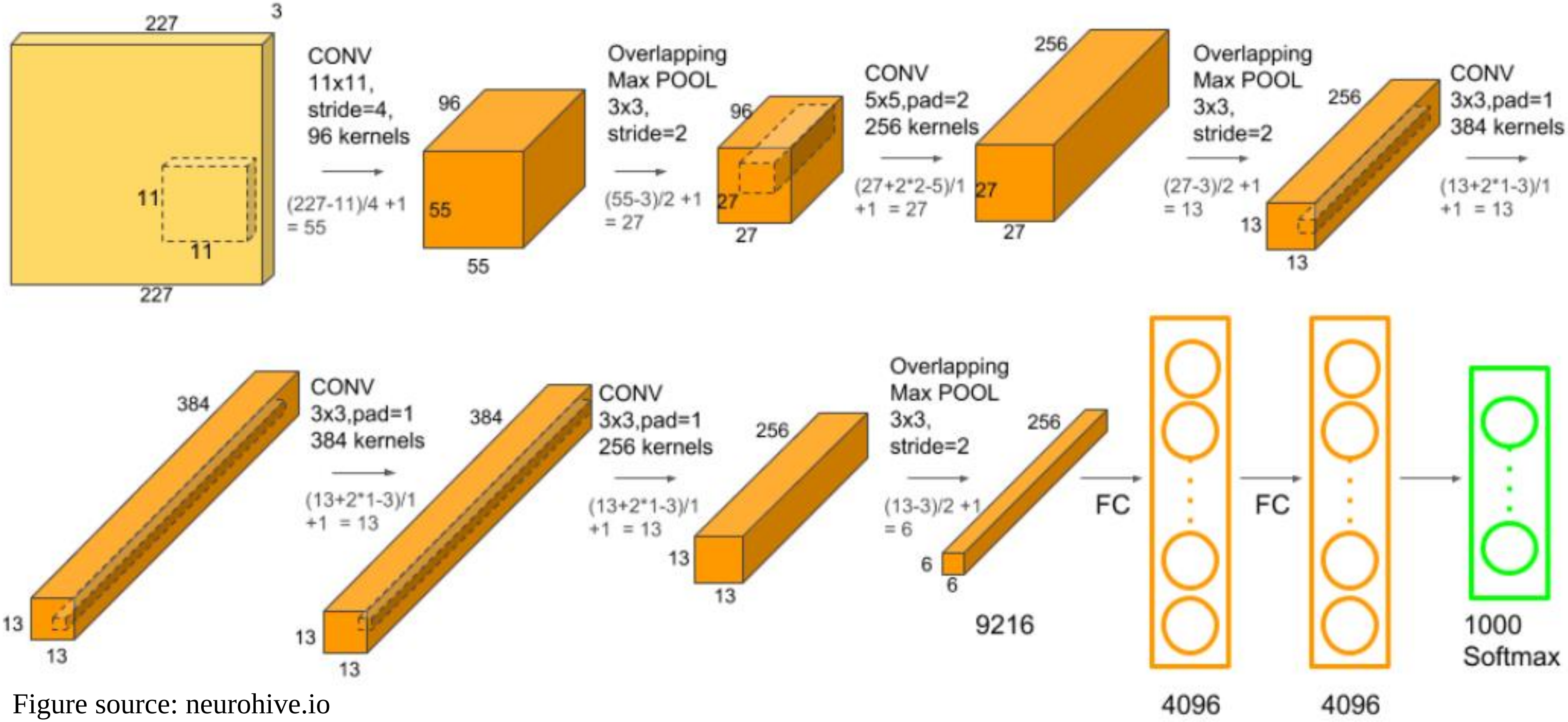
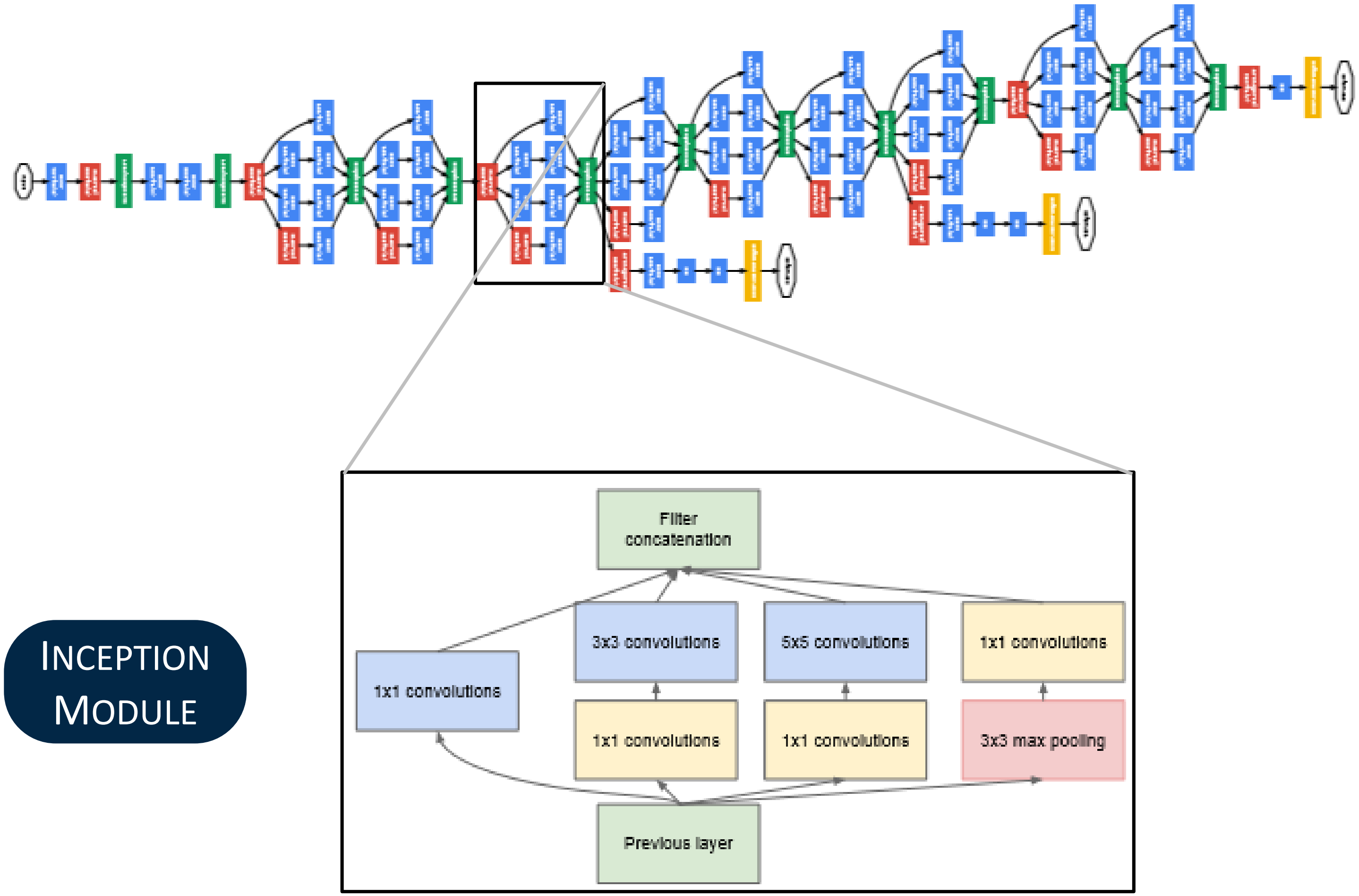


Figure source: neurohive.io

- Use of ReLU activation, dropout regularization. Implementation on GPU.
- Total number of parameters $\sim$ 60 million

Krizhevsky *et. al*. ImageNet Classification with Deep Convolutional Neural Network, NIPS 2012.

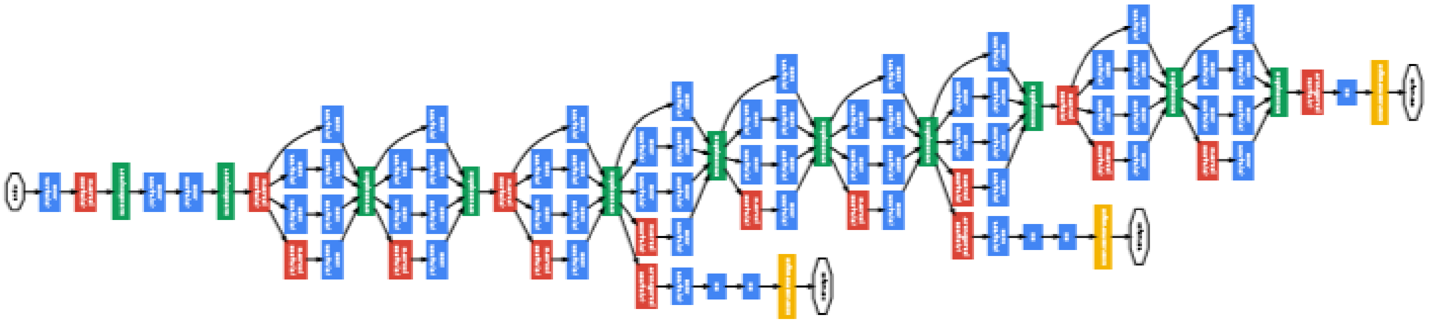
GoogLeNet



INCEPTION MODULE

Figures source: Szegedy *et. al.* Going deeper with convolutions, CVPR 2015

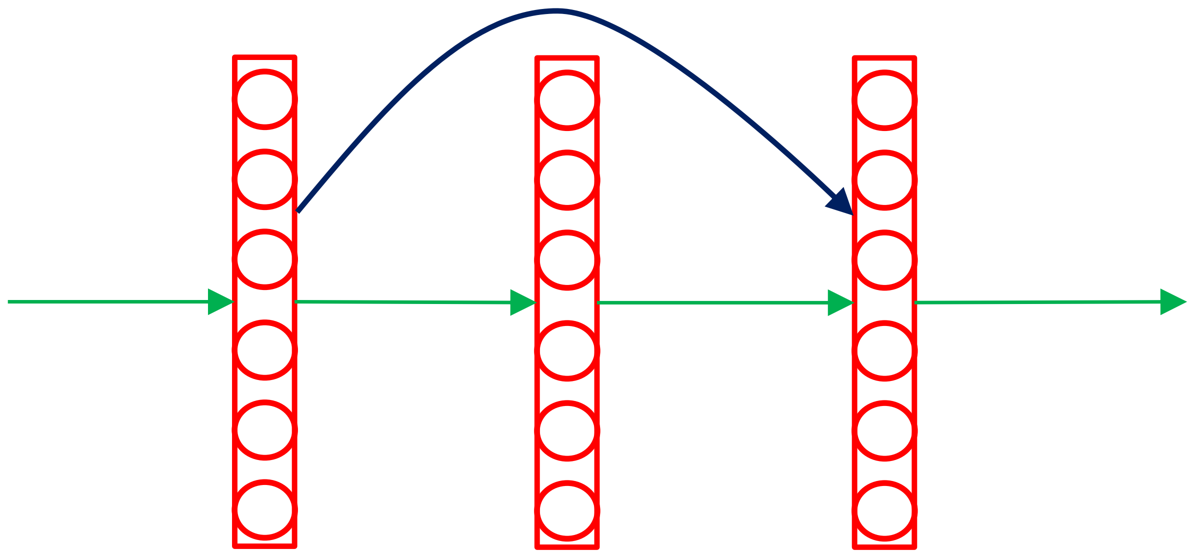
GoogLeNet



- Use series of filters in the same layer to handle multiple scales.
- Use of 1×1 filters for dimensionality reduction.
- Use of auxiliary classifiers to address the issue of vanishing gradient.

Figures source: Szegedy *et. al.* Going deeper with convolutions, CVPR 2015

ResNet



- Deep networks are harder to train due to the problem of exploding/vanishing gradients.
- ResNet uses skip connections where the output from a layer is feed into a deeper layer.
- Skip-connections help in the back-propagation of gradients and thus assist in training deep networks.

He *et. al.* Deep Residual Learning for Image Recognition, CVPR 2016

TRANSFER LEARNING

Transfer Learning

- Training a CNN model from scratch is difficult:
  - Need a large dataset to train the model.
  - Well-known architectures take weeks to train using multiple GPUs.

- Inspiration:
  - Low-level features such as edge, corner, color-blob detectors are generic.
  - High-level features become more specific to the classes in the original trained dataset.

Types of tasks

- Extraction of features
  - Remove the last FC layer, but treat all the other layers as fixed.
  - On passing a new dataset yields a features of fixed dimension for each example.
  - Use the resultant features to train a new classifer.

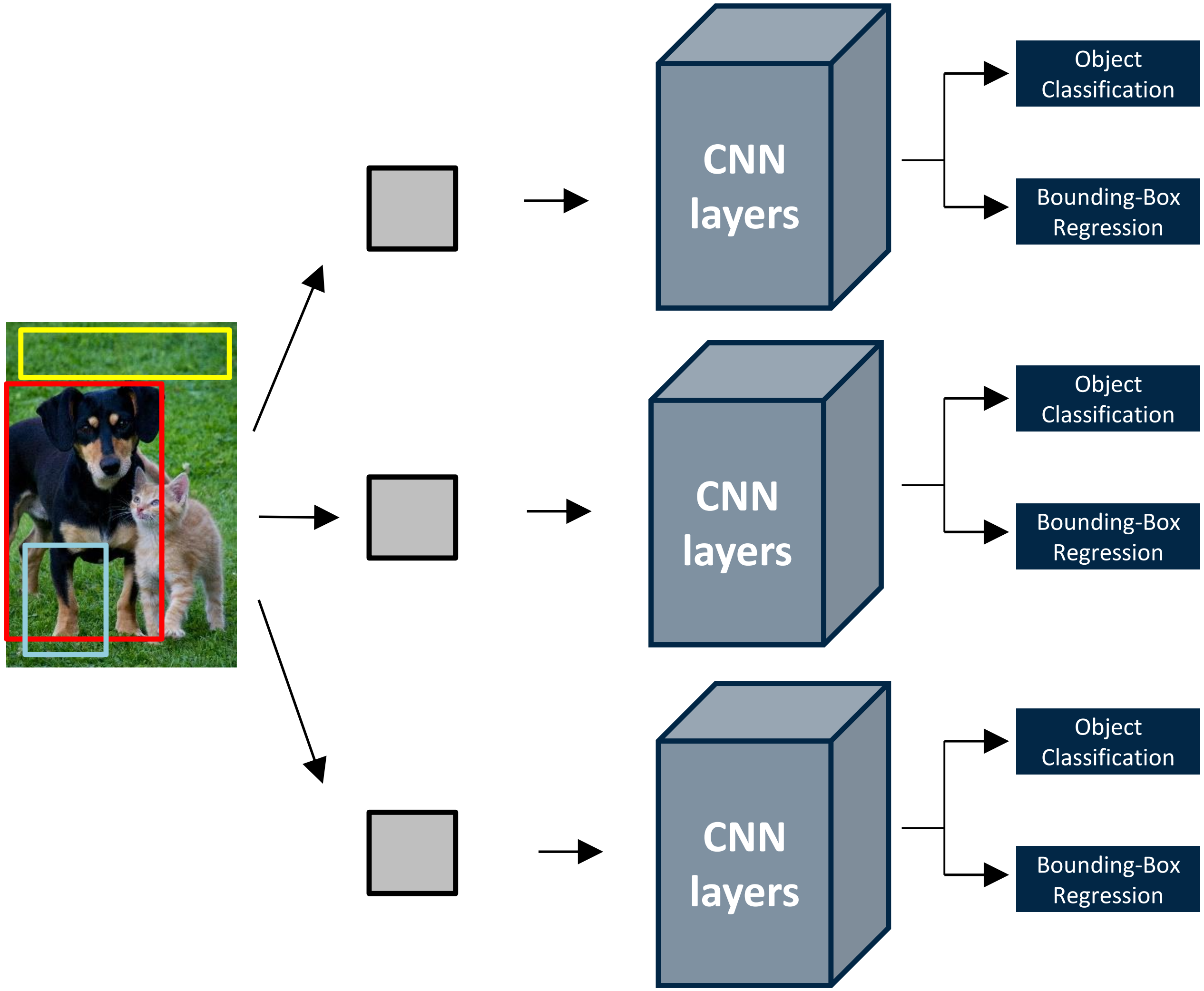
- Fine-tuning CNN model
  - Fine-tune the weights of the pretrained model using backpropagation.
  - The whole network or some higher layers can be fine-tuned.

Scenarios

New dataset characteristics w.r.t. original dataset	Similar	Different
Small	Higher level CNN features are relevant Fine-tuning CNN on the new dataset can lead to overfitting Just train a linear classifier	Final layers contain data-specific features Train a classifier by taking as input the output from an intermediate hidden layer.
Large	Fine-tune the CNN layers	Fine-tune the entire network, and initializing with weights from the pre-trained model

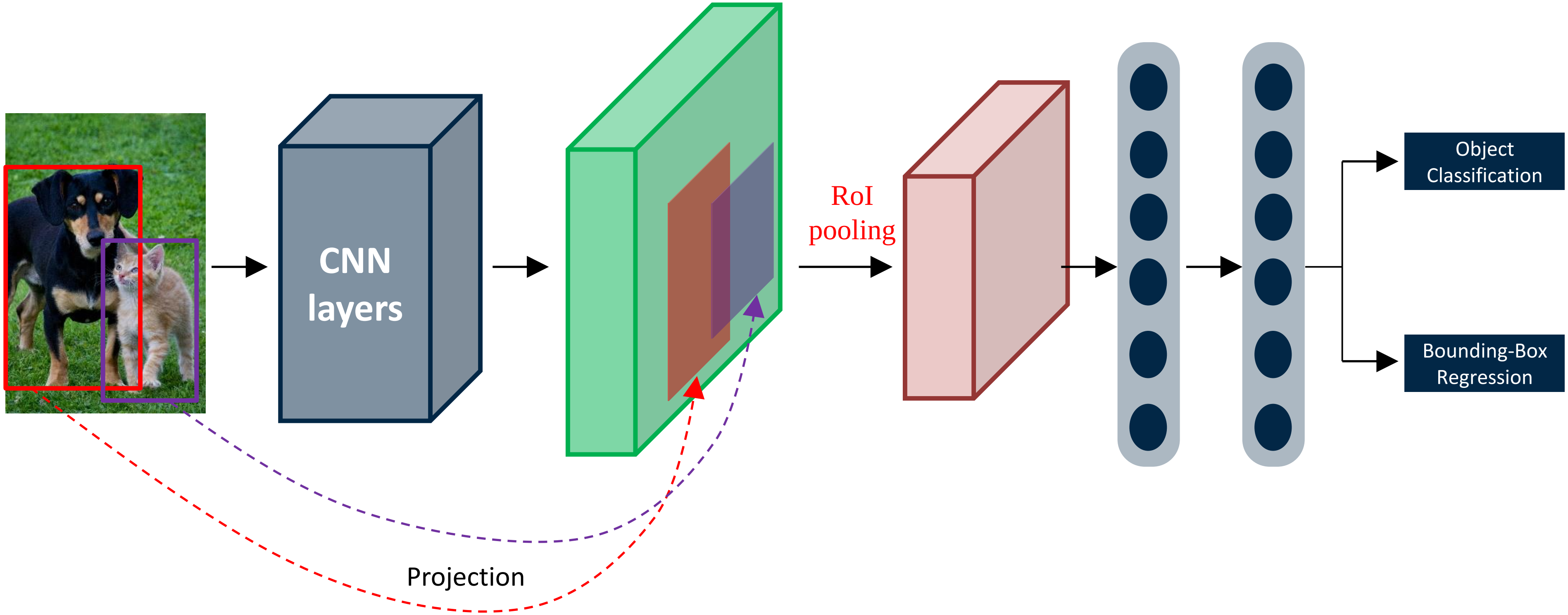
OBJECT DETECTION

R-CNN



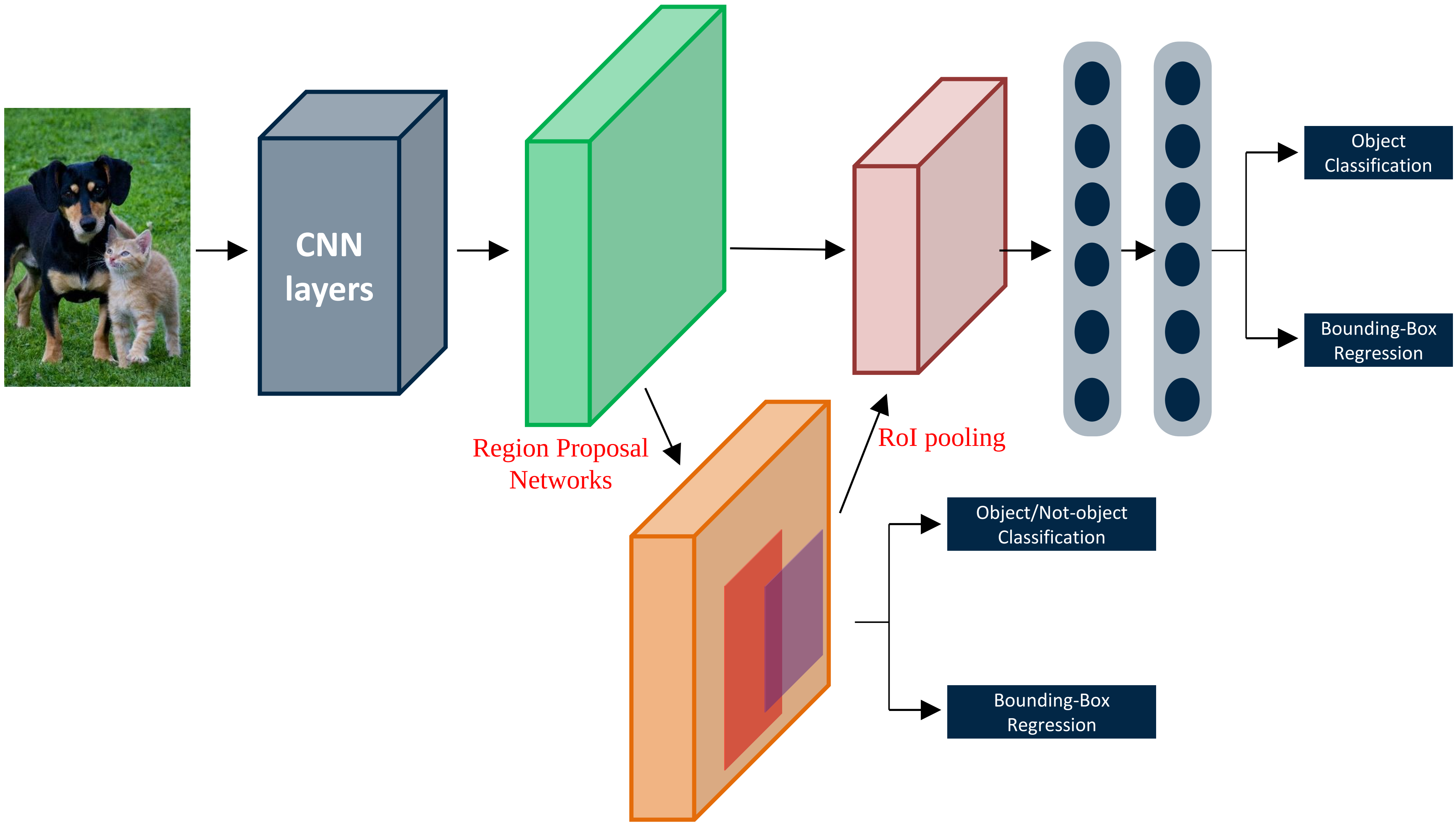
Reference: Girshick *et. al.* Rich feature hierarchies for accurate object detection and semantic segmentation, Proc. IEEE CVPR 2014

Fast R-CNN



Reference: Girshick R. Fast R-CNN, Proc. IEEE ICCV 2015

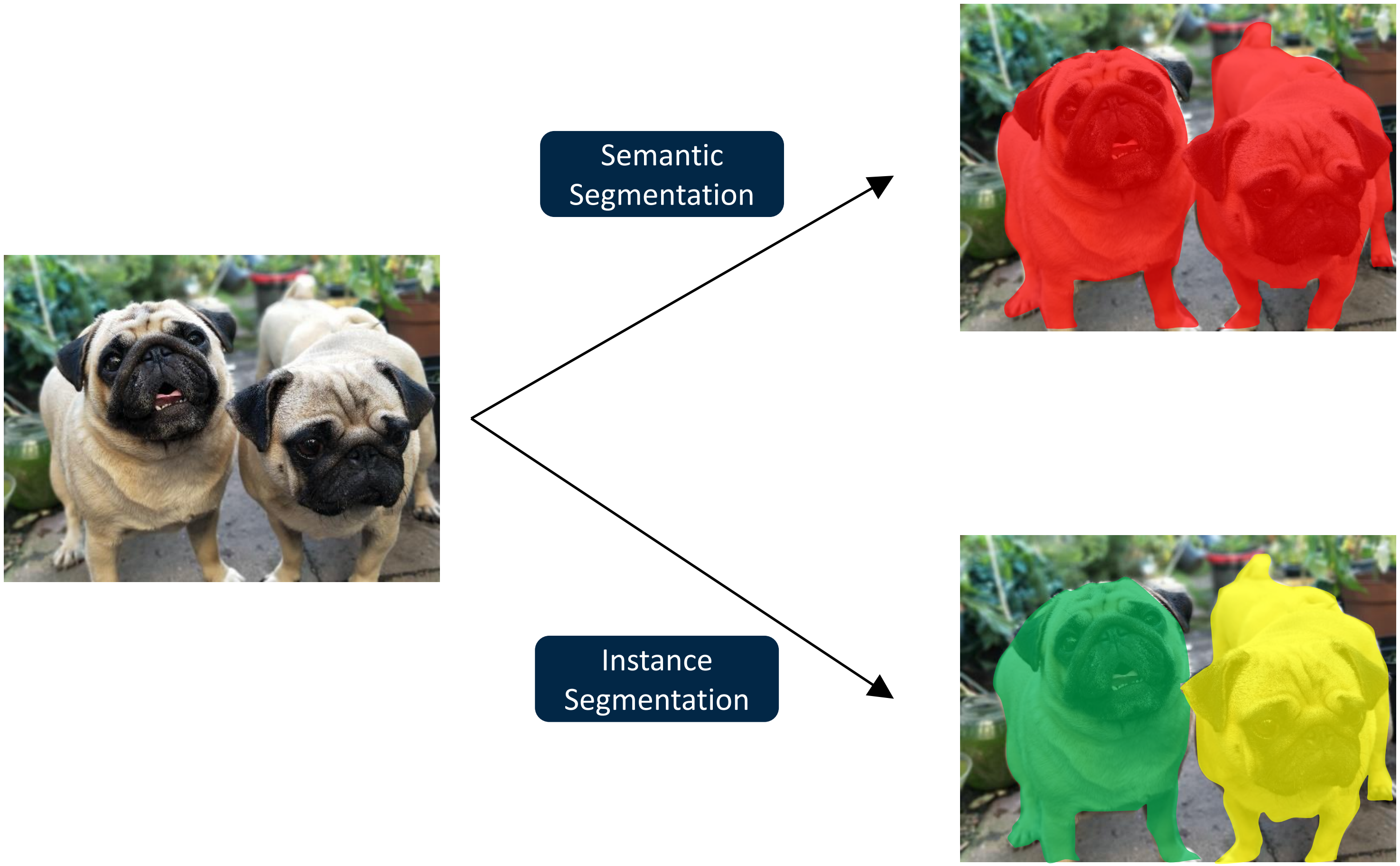
Faster R-CNN



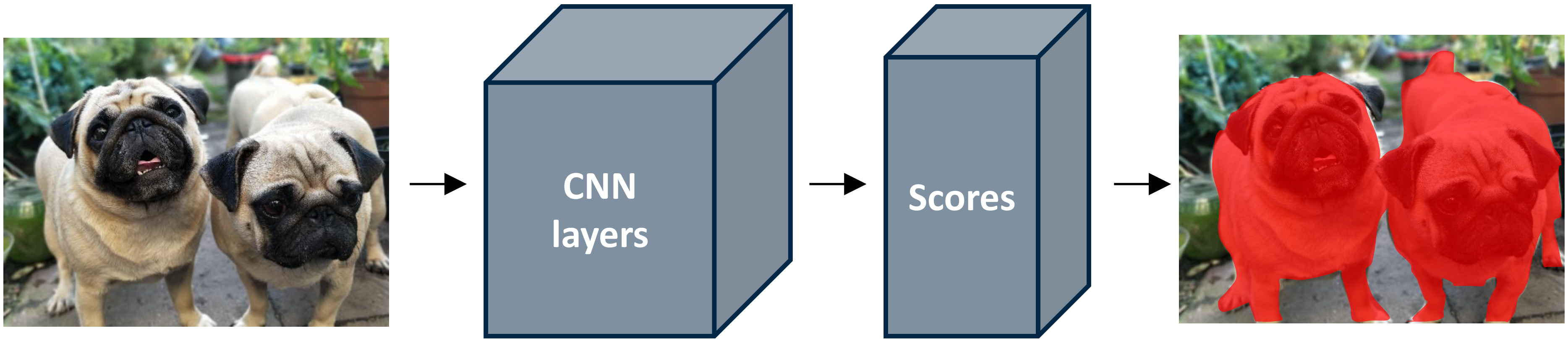
Reference: Ren S. *et. al*. Faster R-CNN: Towards real-time object detection with region proposal networks, Proc. NIPS 2015

SEGMENTATION

Segmentation



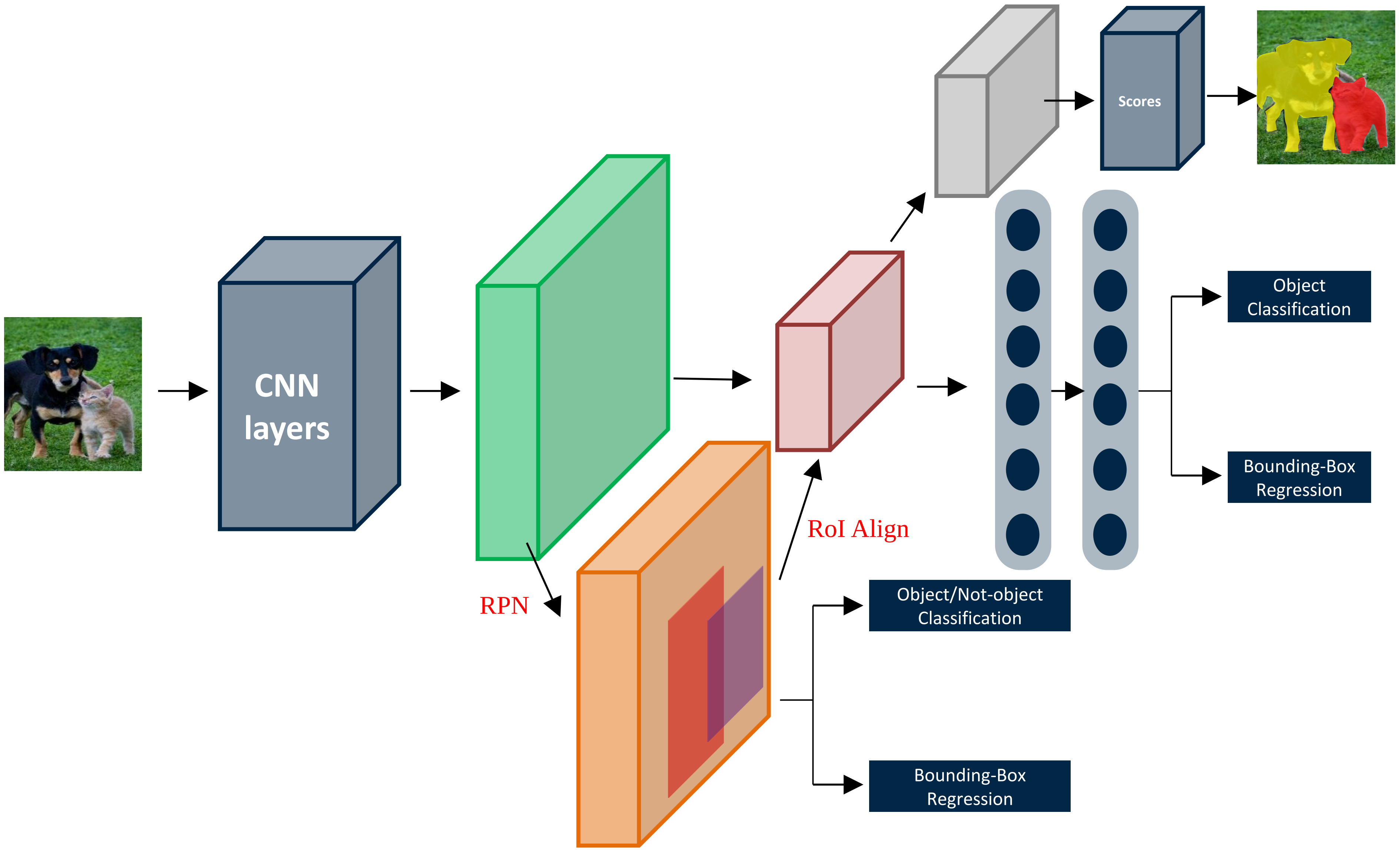
Semantic segmentation: a simple idea



Size: 3 x H x W

Size:
K x H x W

Mask R-CNN



Reference: He *et. al.* Mask R-CNN, Proc. IEEE ICCV 2017

Mask R-CNN

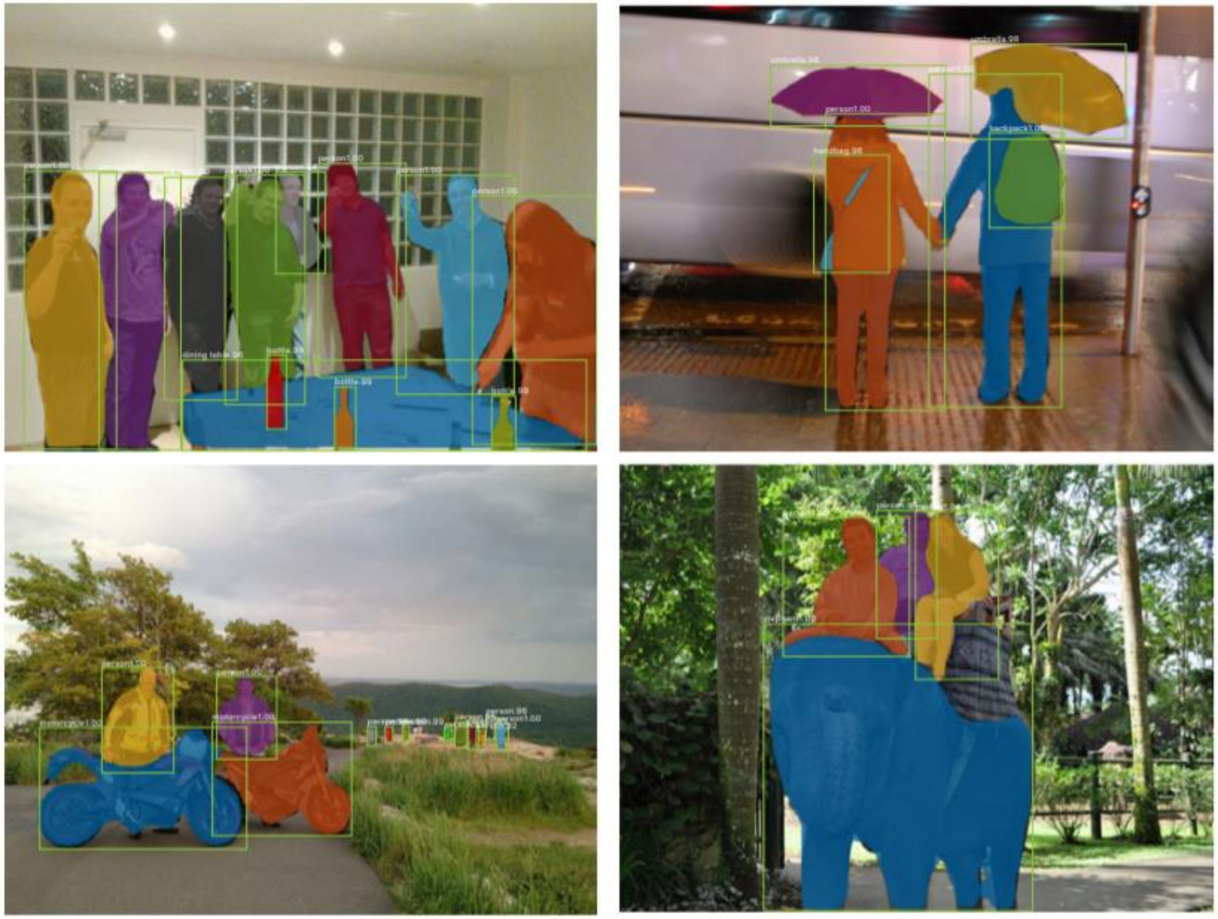


Figure source: He *et. al.* Mask R-CNN, Proc. IEEE ICCV 2017