Deep Learning

A journey from feature extraction and engineering to end-to-end pipelines

Part 3: Convolutional Neural Networks

Andrei Bursuc

With slides from A. Karpathy, F. Fleuret, J. Johnson, S. Yeung, G. Louppe, Y. Avrithis ...

Outline

1. Computer Vision
   - before things went deep: handcrafted features
2. Neural networks
   - empirical risk minimization, stochastic gradient descent, backpropagation
   - multilayer perceptrons
3. Going deeper
   - convolutional layers, deep regularization, deep architectures
4. Under the hood
   - understanding and visualizing CNNs, adversarial attacks
   - CPU vs GPU, using CNNs in practice
5. Unsupervised and self-supervised learning
   - generative models: autoencoders, variational autoencoders, generative adversarial networks
   - self-supervised learning

Outline

1. Computer Vision
   - before things went deep: handcrafted features
2. Neural networks
   - empirical risk minimization, stochastic gradient descent, backpropagation
   - multilayer perceptrons
3. Going deeper
   - convolutional layers, deep regularization, deep architectures
4. Under the hood
   - understanding and visualizing CNNs, adversarial attacks
   - CPU vs GPU, using CNNs in practice
5. Unsupervised and self-supervised learning
   - generative models: autoencoders, variational autoencoders, generative adversarial networks
   - self-supervised learning

Convolutional layers

Why would we need them?

If they were handled as normal "unstructured" vectors, large-dimension signals such as sound samples or images would require models of intractable size.

For instance a linear layer taking a 256×256 RGB image as input, and producing an image of same size would require:

$$(256 \times 256 \times 3)\hat{}2 \simeq 3.87e + 10$$

parameters, with the corresponding memory footprint ($\simeq$150Gb !), and excess of capacity.

Slide credit: F. Fleuret, EE-559 Deep learning

Why would we need them?

Moreover, this requirement is inconsistent with the intuition that such large signals have some "invariance in translation". A representation meaningful at a certain location can / should be used everywhere.

Slide credit: F. Fleuret, EE-559 Deep learning

Why would we need them?

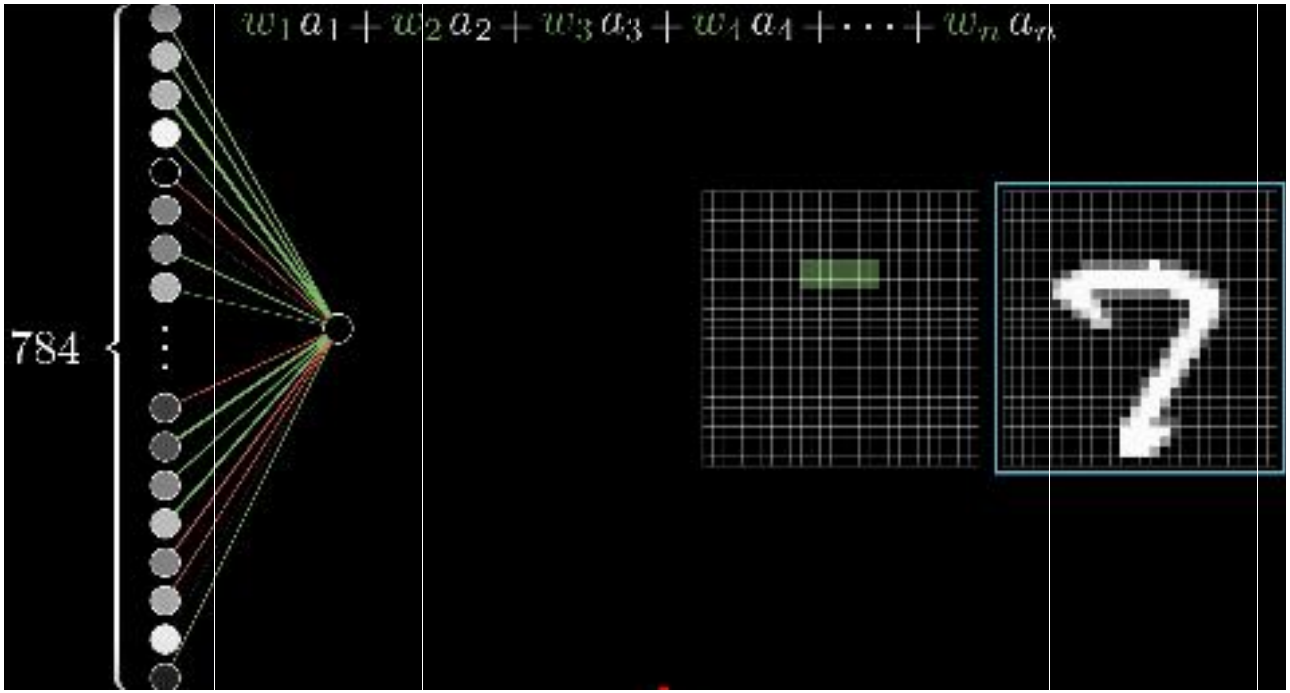
Moreover, this requirement is inconsistent with the intuition that such large signals have some "invariance in translation". A representation meaningful at a certain location can / should be used everywhere.

A convolutional layer embodies this idea. It applies the same linear transformation locally, everywhere, and preserves the signal structure.

Slide credit: F. Fleuret, EE-559 Deep learning

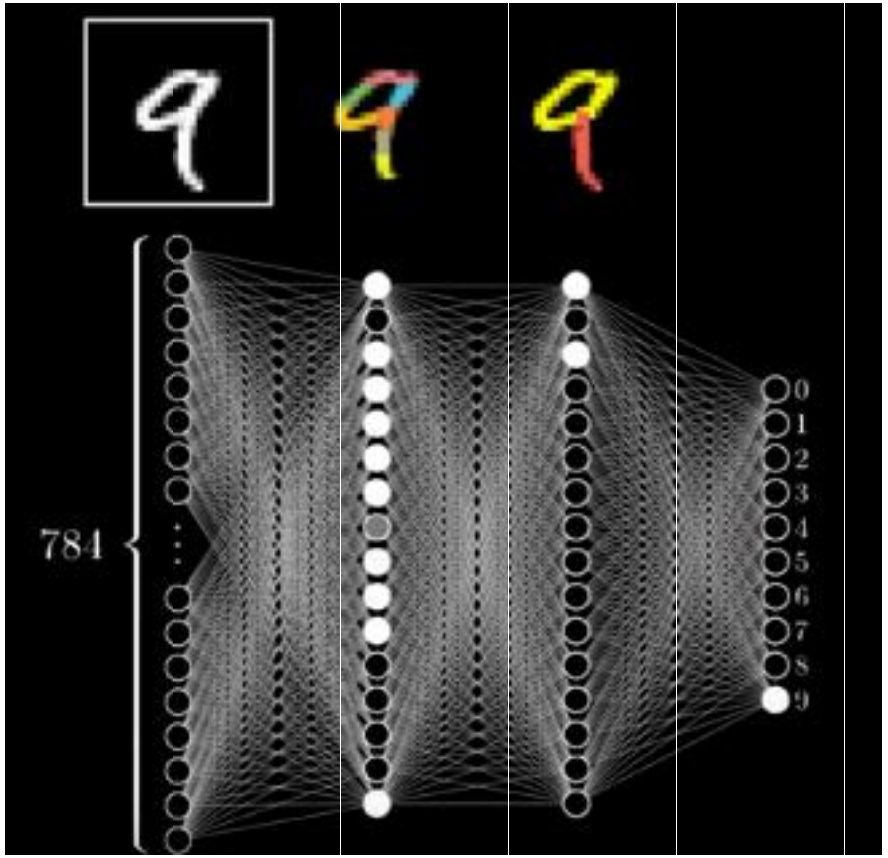
Why would we need them?

- One neuron gets specialized for detecting a full-image pattern, while being sensible to translations

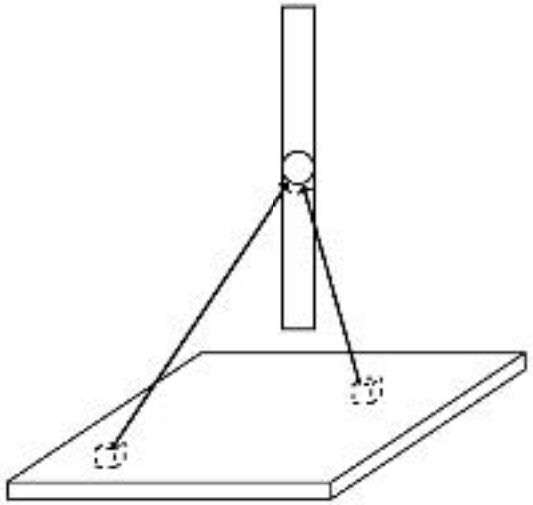


Why would we need them?

- Each neuron gets specialized for detecting a full-image pattern.
- Neurons from later layer work similarly
- This is a big waste of parameters without good performance.



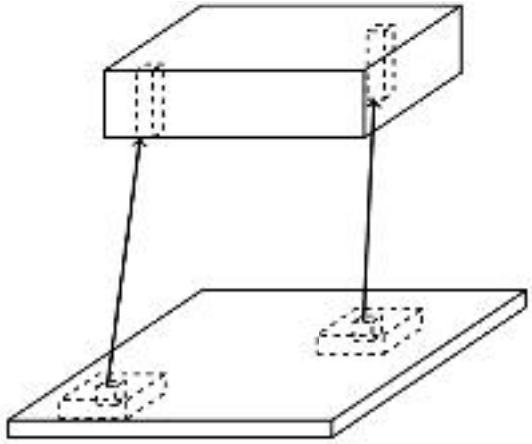
Fully connected layer



In a fully connected layer, each hidden unit $h_j = \sigma(\mathbf{w}_j^T \mathbf{x} + b_j)$ is connected to the entire image.

- Looking for activations that depend on pixels that are spatially far away is supposedly a waste of time and resources.
- Long range correlations can be dealt with in the higher layers.

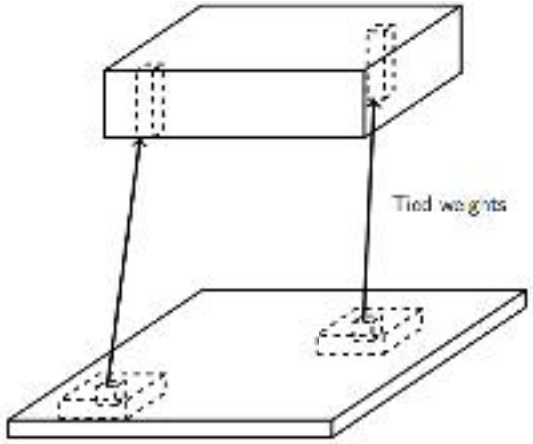
Locally connected layer



In a locally connected layer, each hidden unit h_j is connected to only a patch of the image.

- Weights are specialized locally and functionally.
- Reduce the number of parameters.
- What if the object in the image shifts a little?

Convolutional layer



In a convolutional layer, each hidden unit h_j is connected to only a patch of the image, and share its weights with the other units h_i .

- Weights are specialized functionally, regardless of spatial location.
- Reduce the number of parameters.

Convolution

Discrete convolution (actually cross-correlation) between two functions f and g:

$$(f \star g)(x) = \sum_{a+b=x} f(a).\, g(b) = \sum_{a} f(a).\, g(x+a)$$

Convolution

Discrete convolution (actually cross-correlation) between two functions f and g:

$$(f \star g)(x) = \sum_{a+b=x} f(a).\, g(b) = \sum_{a} f(a).\, g(x+a)$$

In computer vision, we typically use 2D-convolutions (actually 2D cross-correlation):

$$(f \star g)(x, y) = \sum_{n} \sum_{m} f(n, m).\, g(x+n, y+m)$$

Convolution

Discrete convolution (actually cross-correlation) between two functions f and g:

$$(f \star g)(x) = \sum_{a+b=x} f(a).\, g(b) = \sum_{a} f(a).\, g(x+a)$$

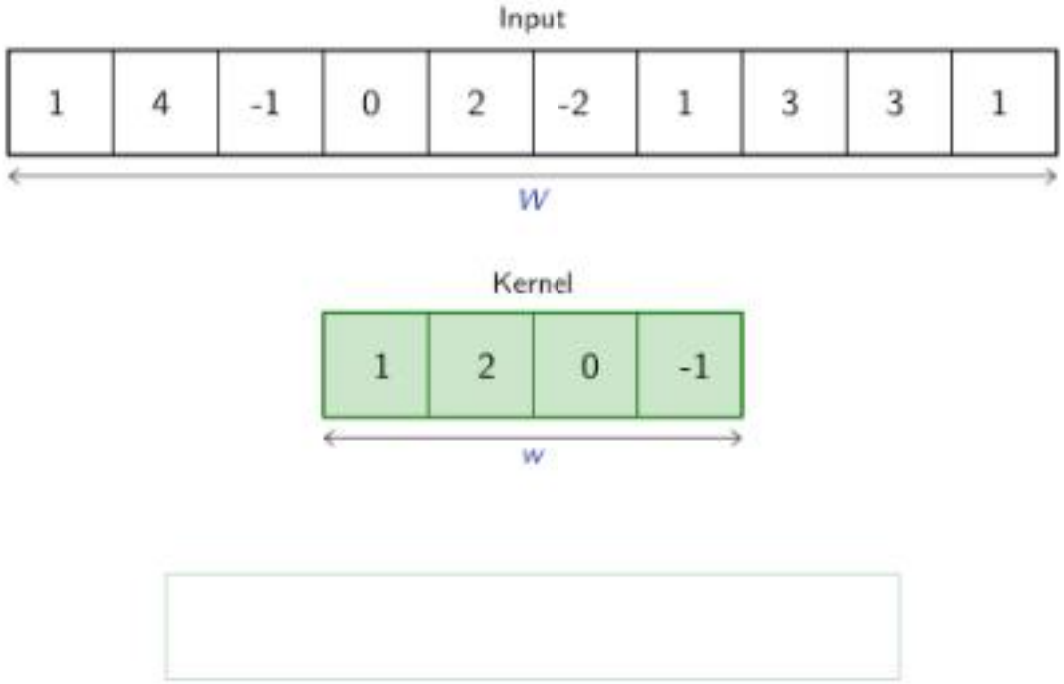
In computer vision, we typically use 2D-convolutions (actually 2D cross-correlation):

$$(f \star g)(x, y) = \sum_{n} \sum_{m} f(n, m).\, g(x+n, y+m)$$

f is a convolution kernel applied to the 2-d map g (think image)

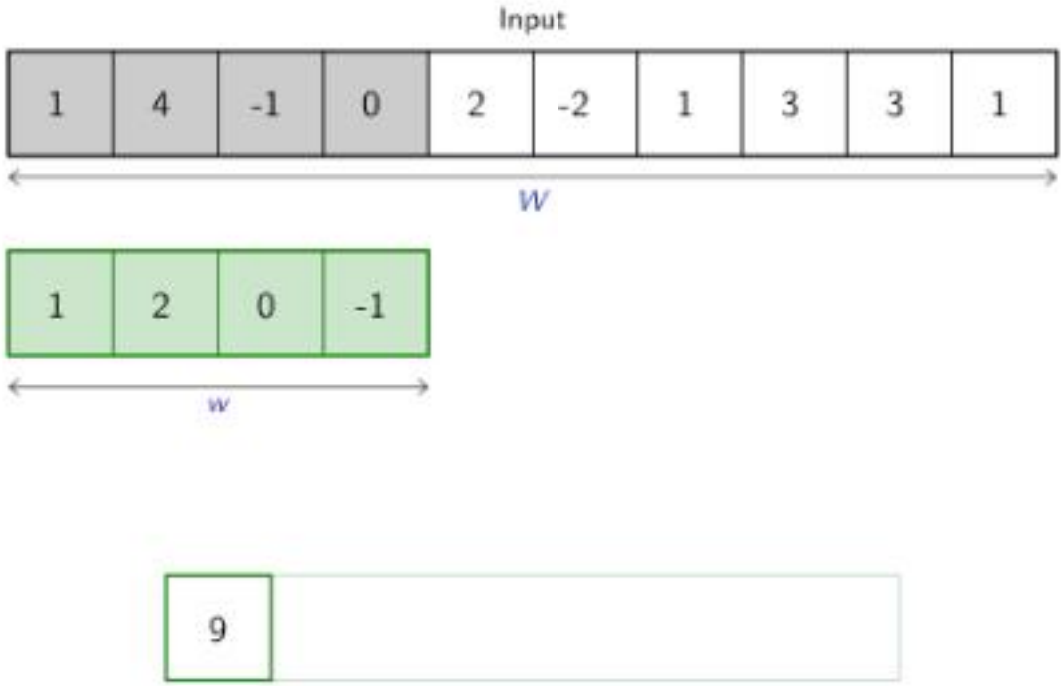
Slide credit: C. Ollion & O. Grisel, M2DS Deep Learning

Convolution 1d



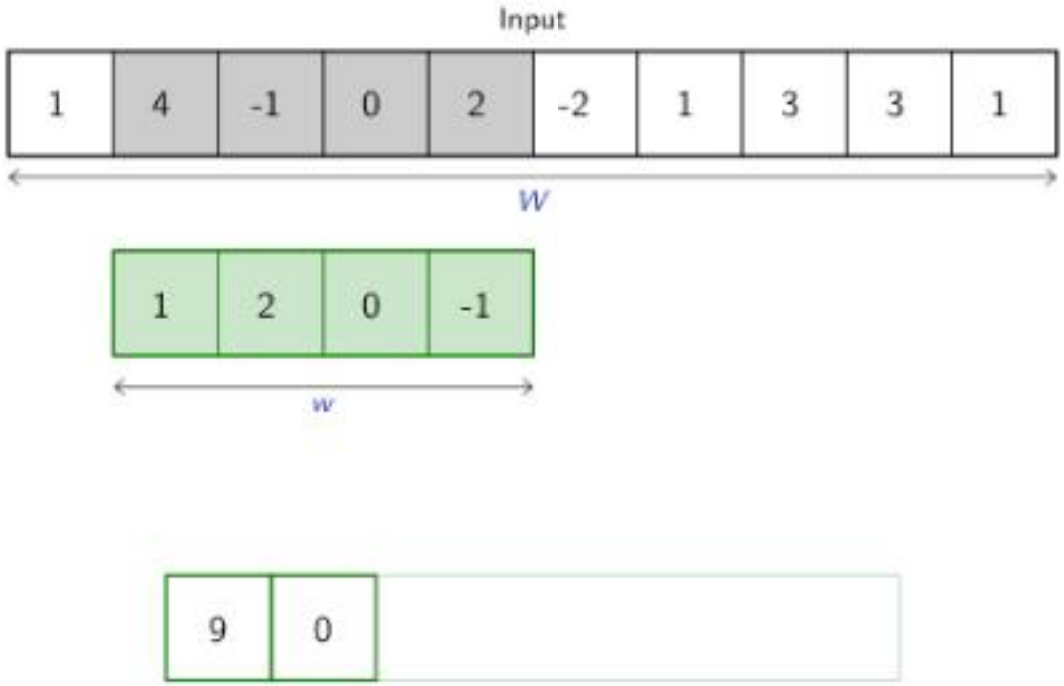
Slide credit: F. Fleuret, EE-559 Deep learning

Convolution 1d



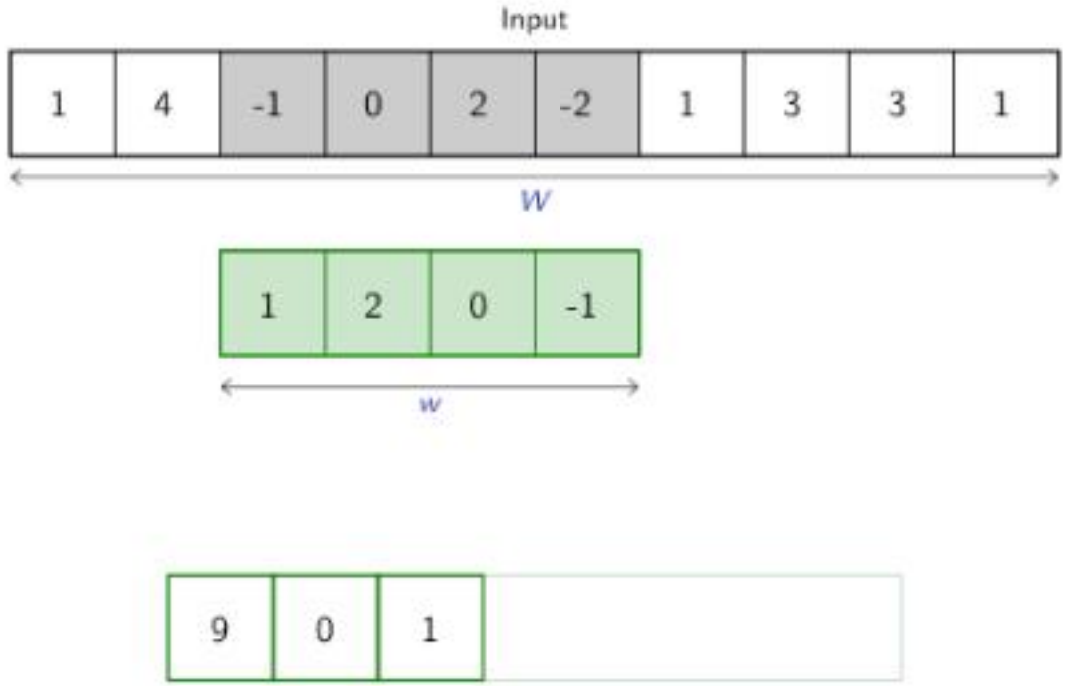
Slide credit: F. Fleuret, EE-559 Deep learning

Convolution 1d



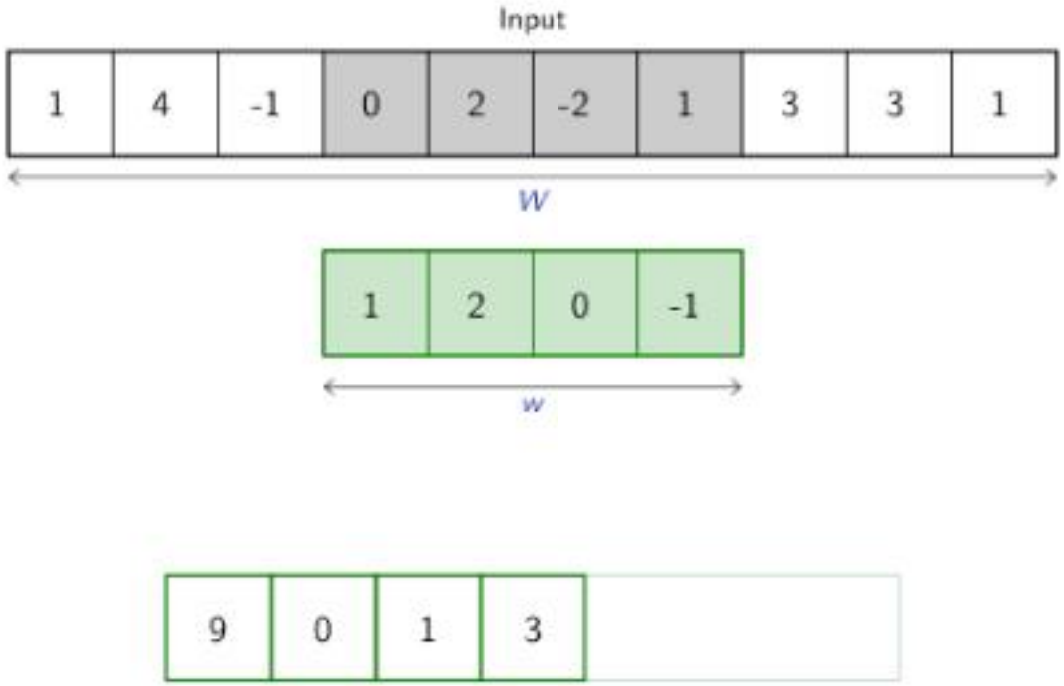
Slide credit: F. Fleuret, EE-559 Deep learning

Convolution 1d



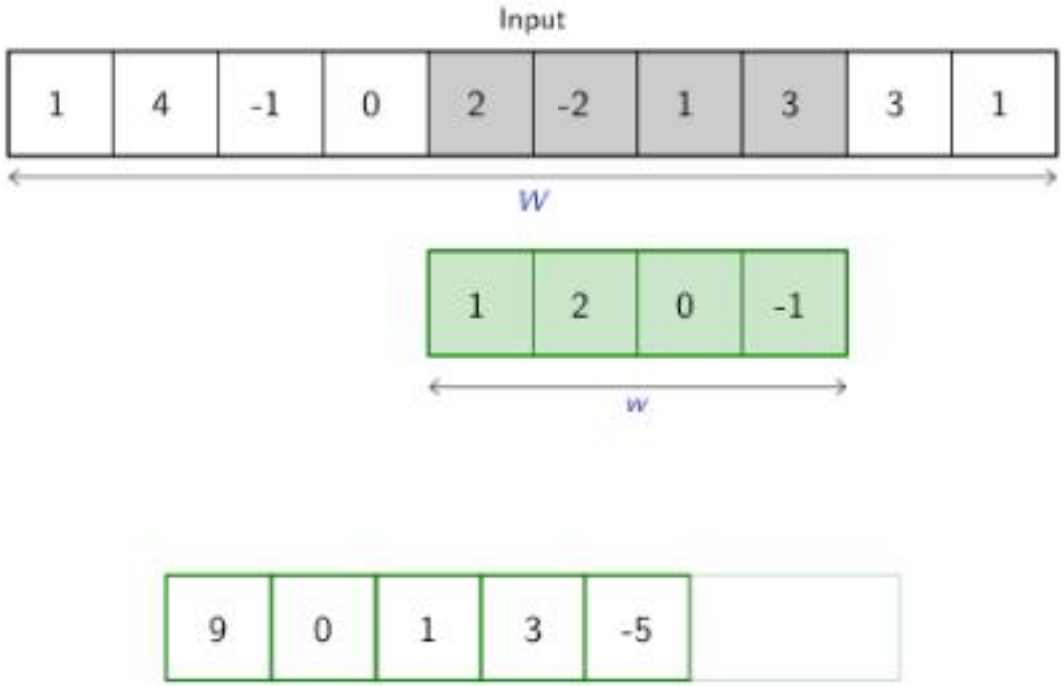
Slide credit: F. Fleuret, EE-559 Deep learning

Convolution 1d



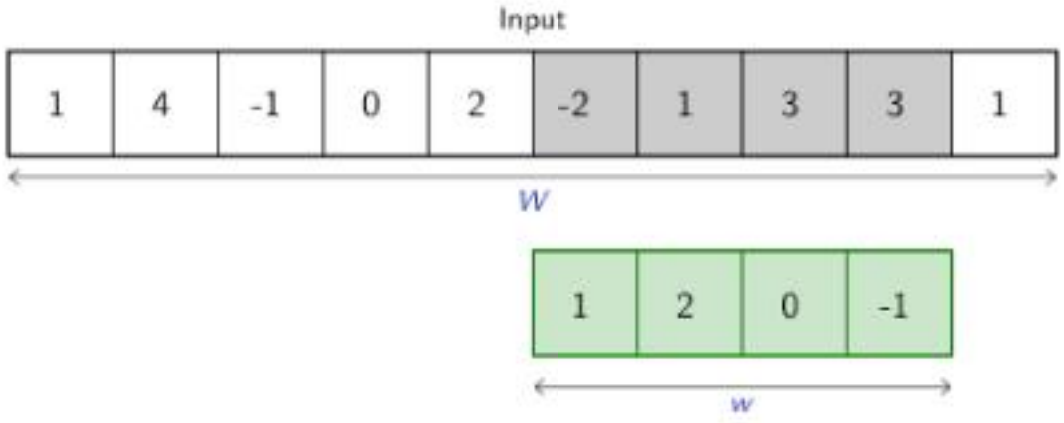
Slide credit: F. Fleuret, EE-559 Deep learning

Convolution 1d



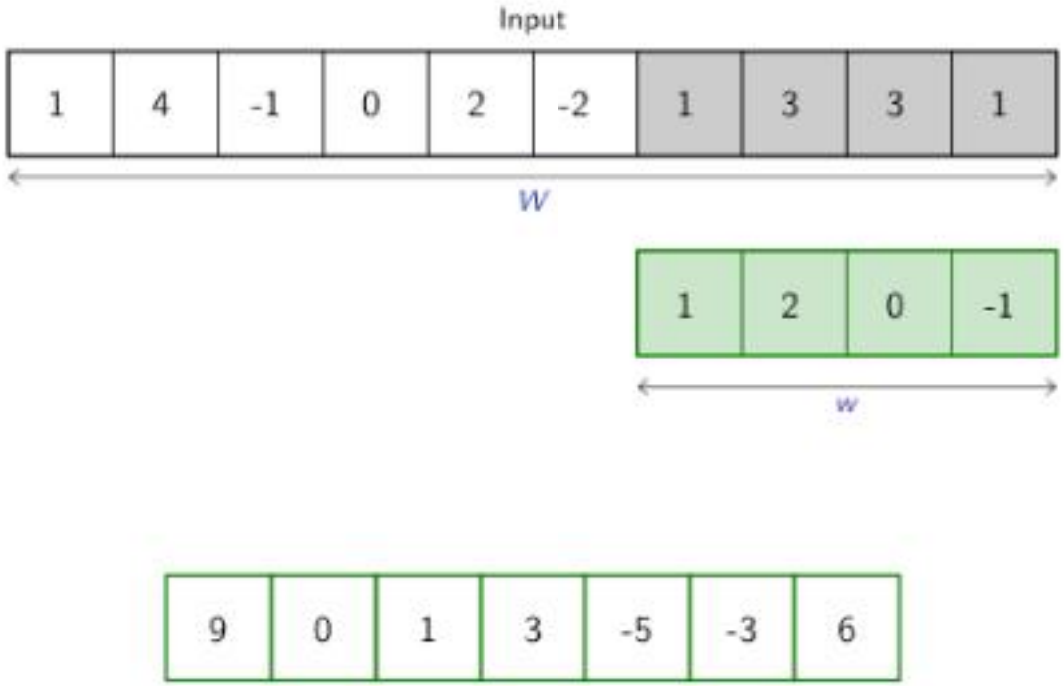
Slide credit: F. Fleuret, EE-559 Deep learning

Convolution 1d



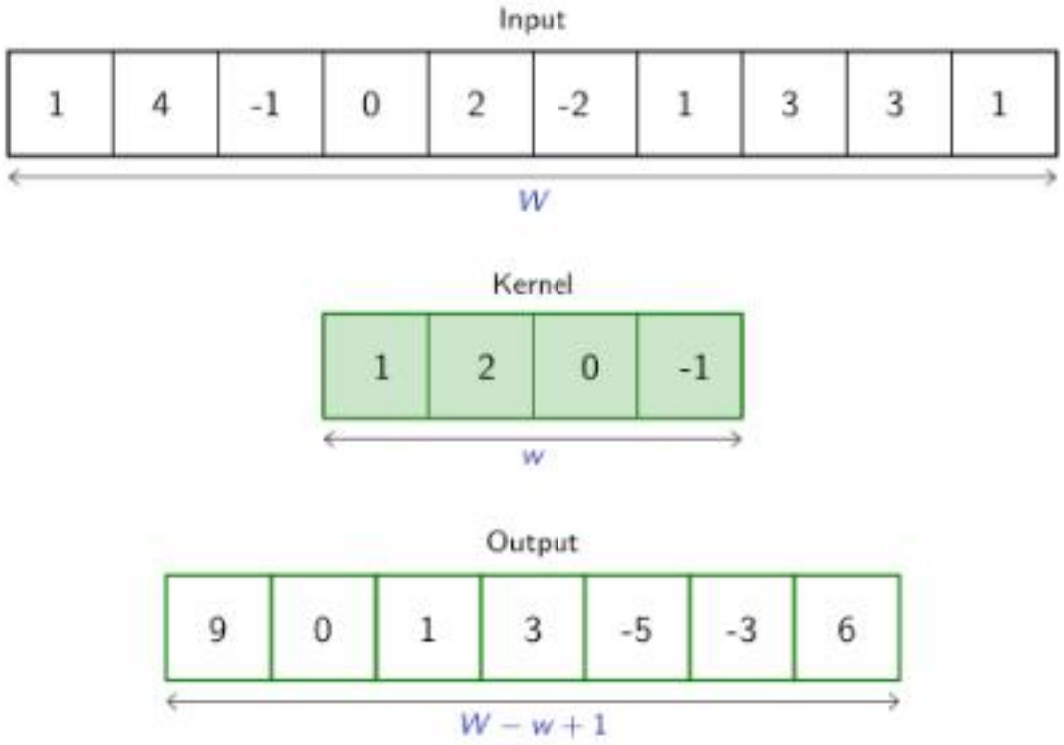
Slide credit: F. Fleuret, EE-559 Deep learning

Convolution 1d



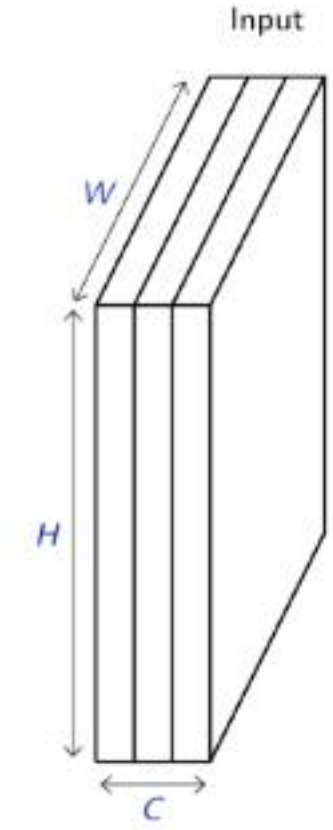
Slide credit: F. Fleuret, EE-559 Deep learning

Convolution 1d



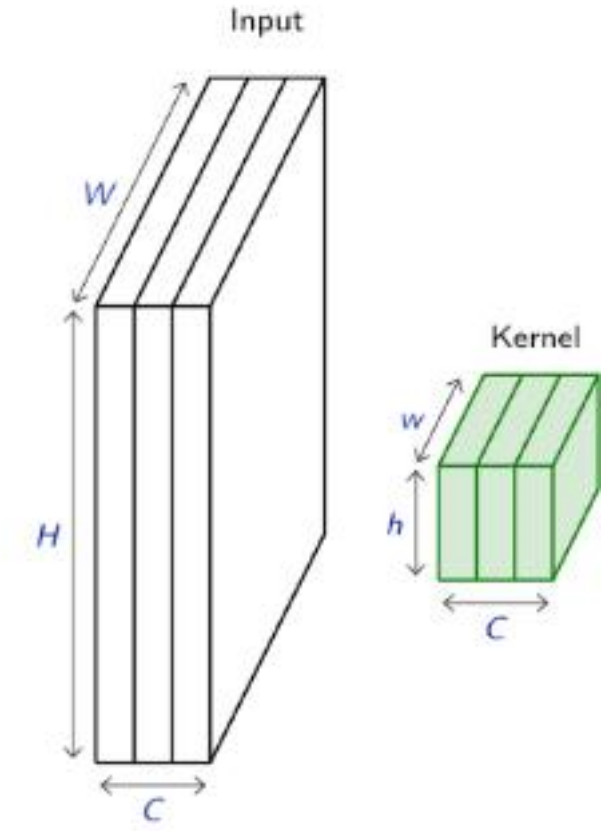
Slide credit: F. Fleuret, EE-559 Deep learning

Convolution 2d



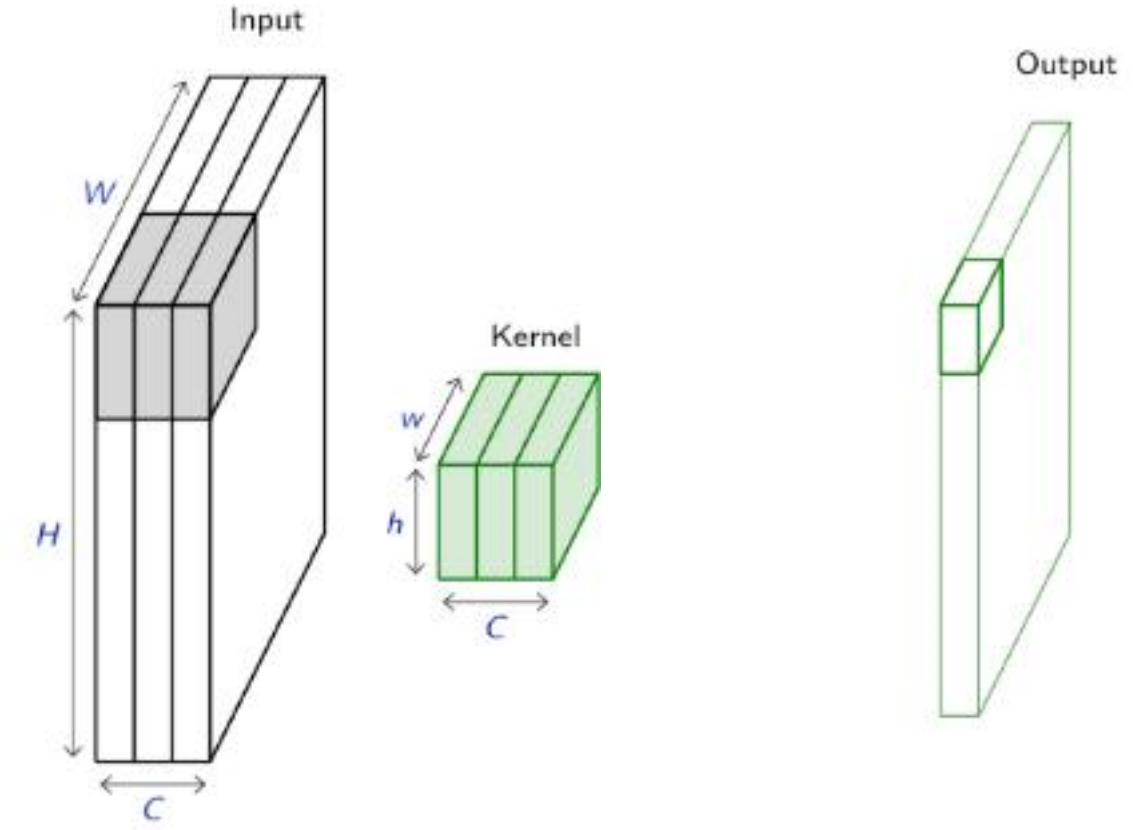
Slide credit: F. Fleuret, EE-559 Deep learning

Convolution 2d



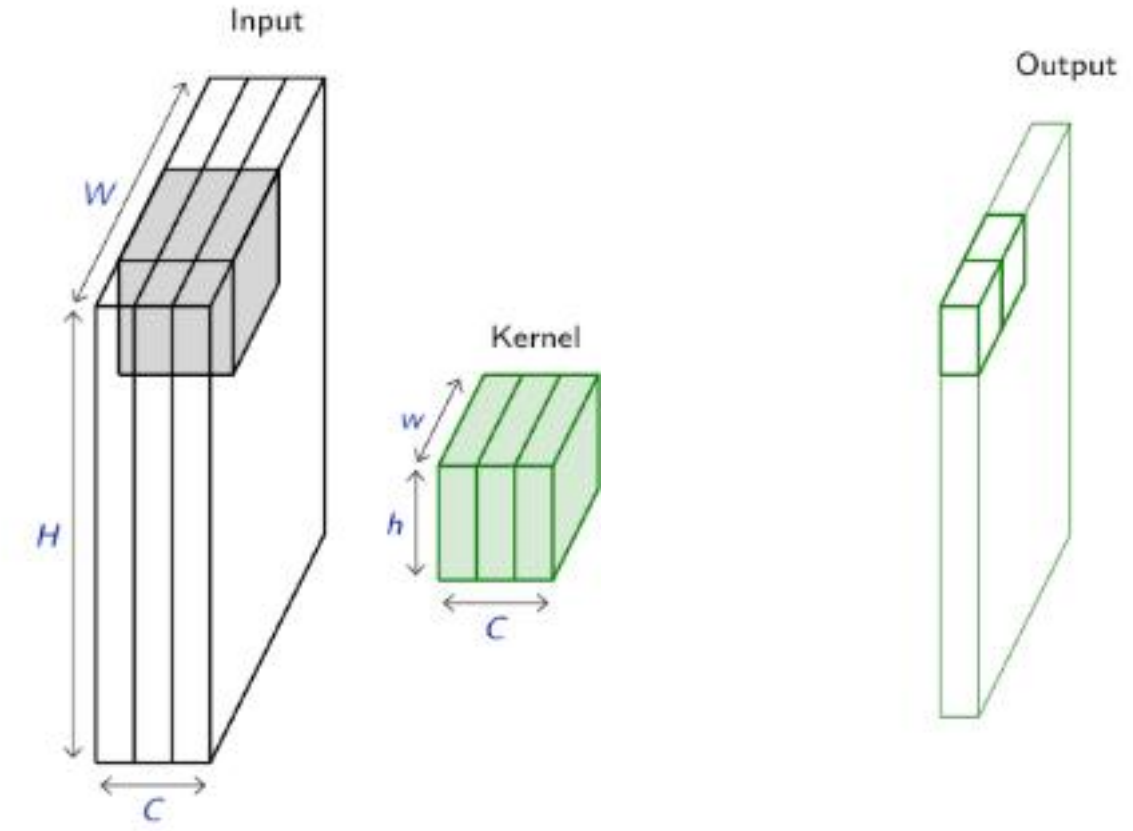
Slide credit: F. Fleuret, EE-559 Deep learning

Convolution 2d



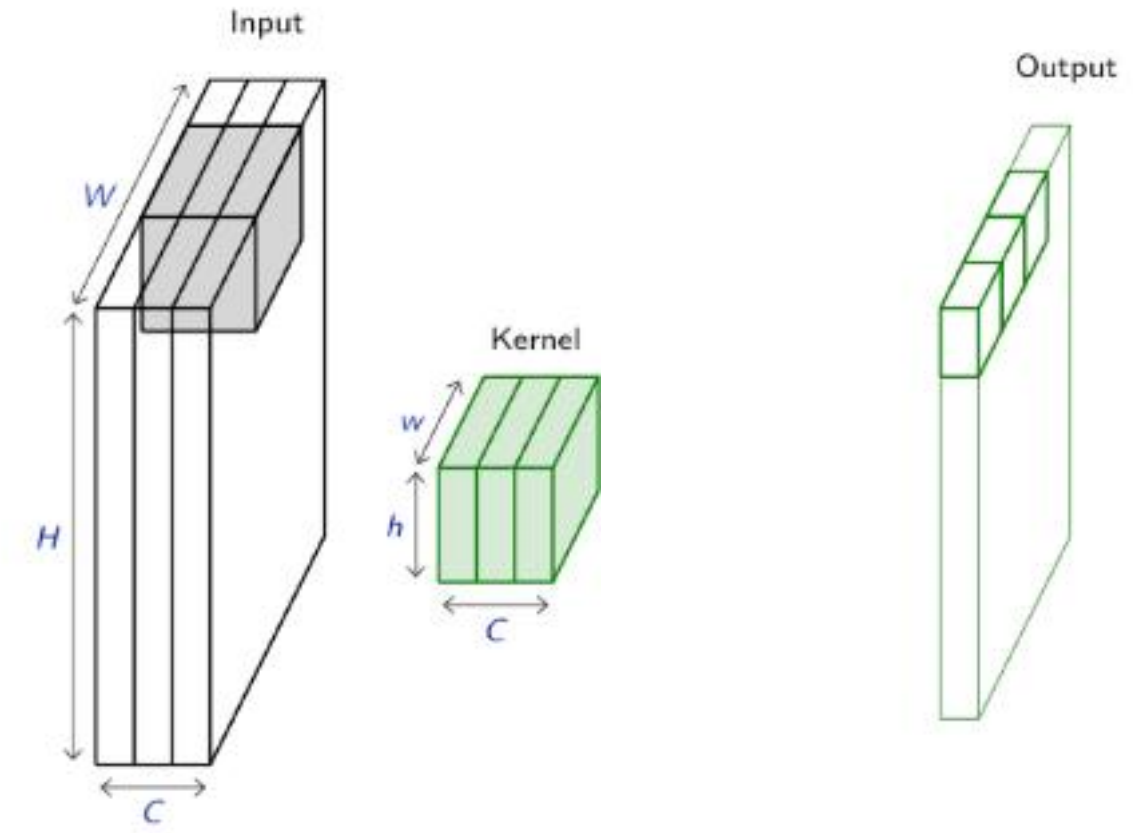
Slide credit: F. Fleuret, EE-559 Deep learning

Convolution 2d



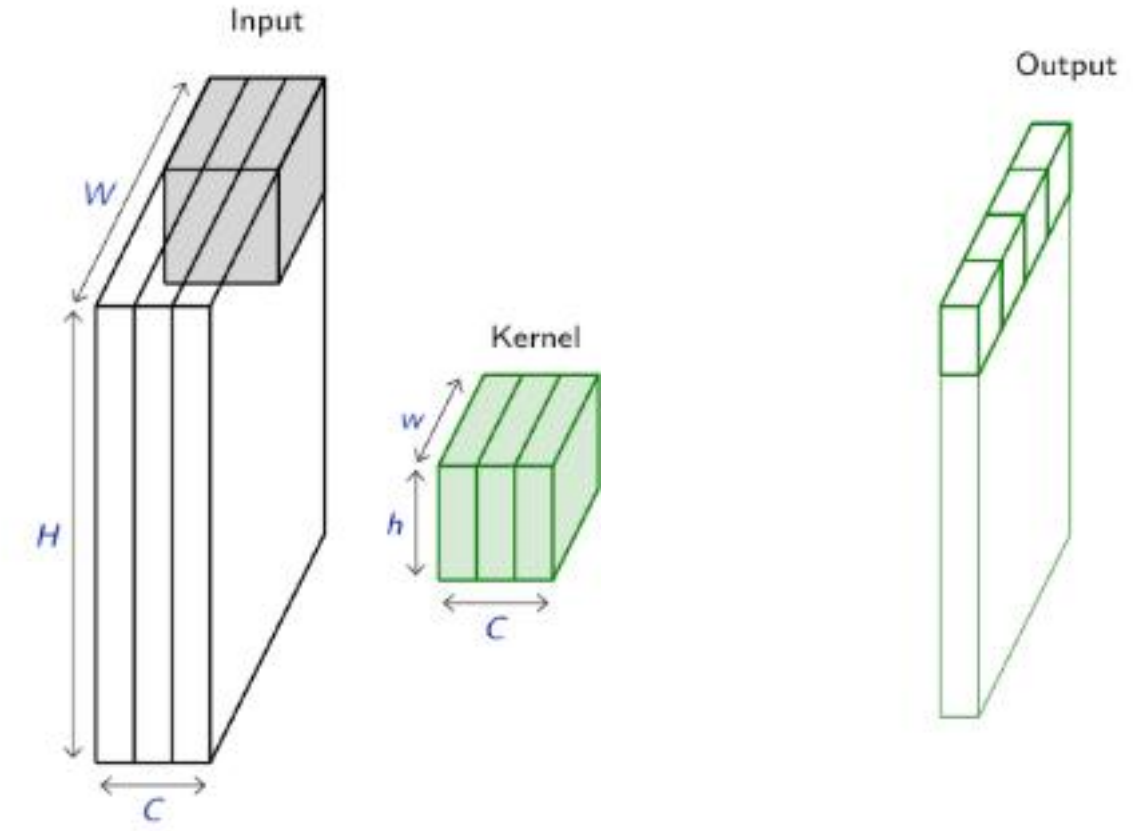
Slide credit: F. Fleuret, EE-559 Deep learning

Convolution 2d



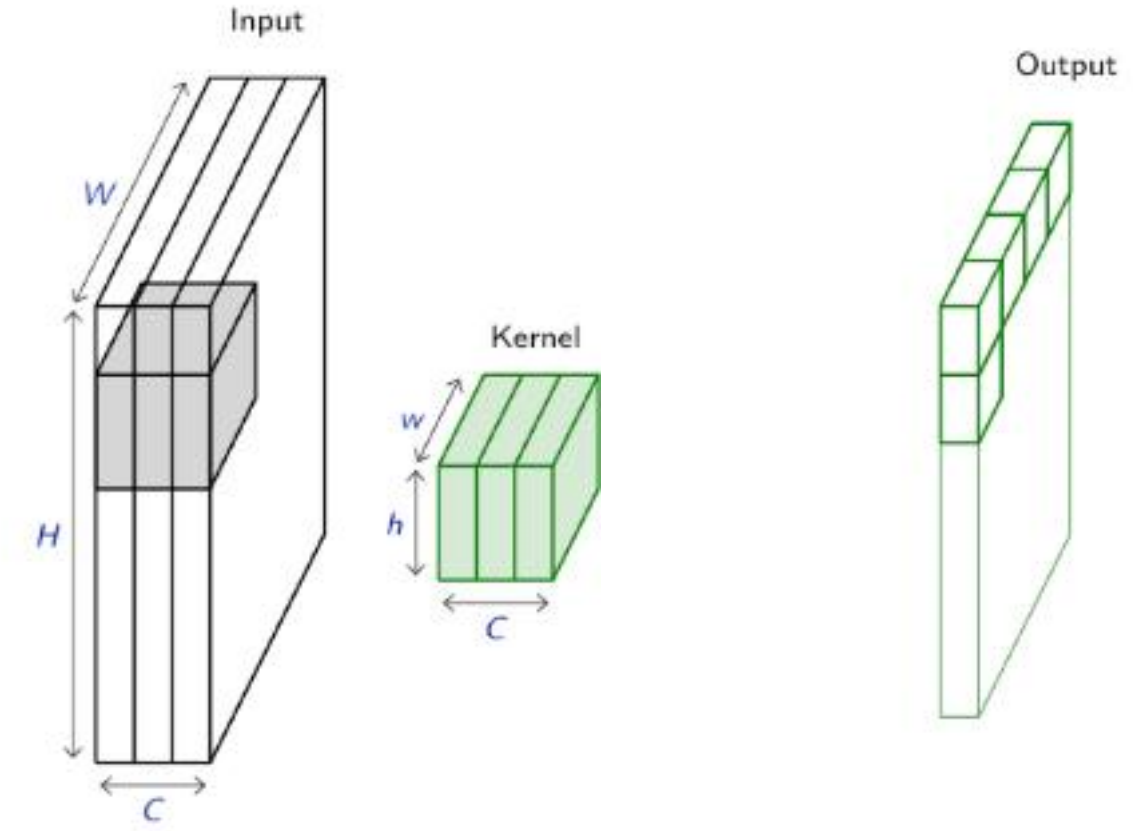
Slide credit: F. Fleuret, EE-559 Deep learning

Convolution 2d



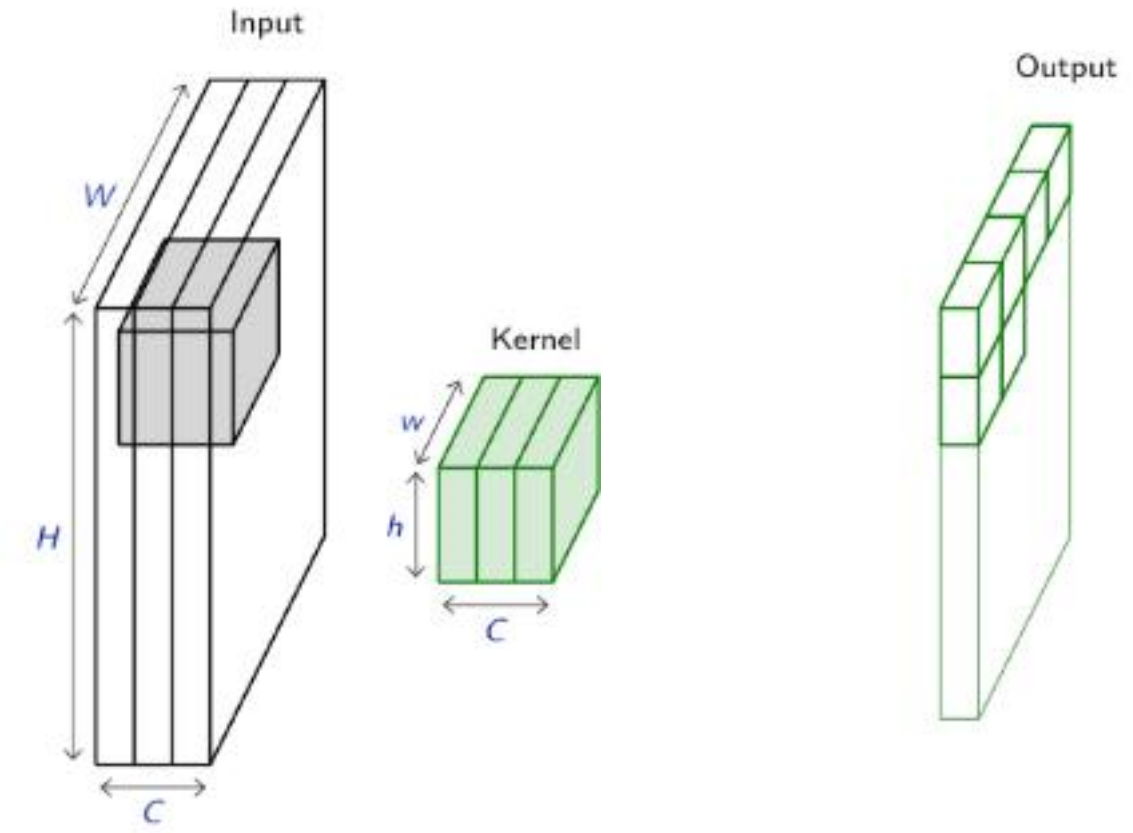
Slide credit: F. Fleuret, EE-559 Deep learning

Convolution 2d



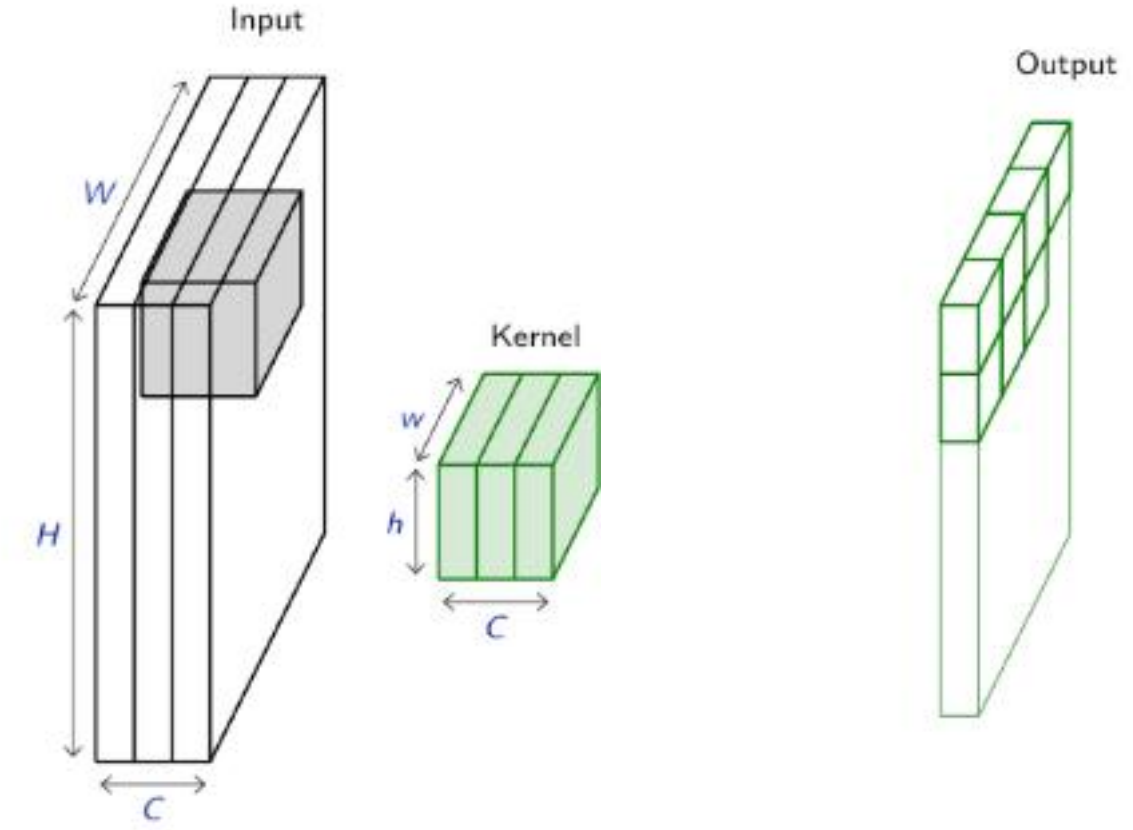
Slide credit: F. Fleuret, EE-559 Deep learning

Convolution 2d



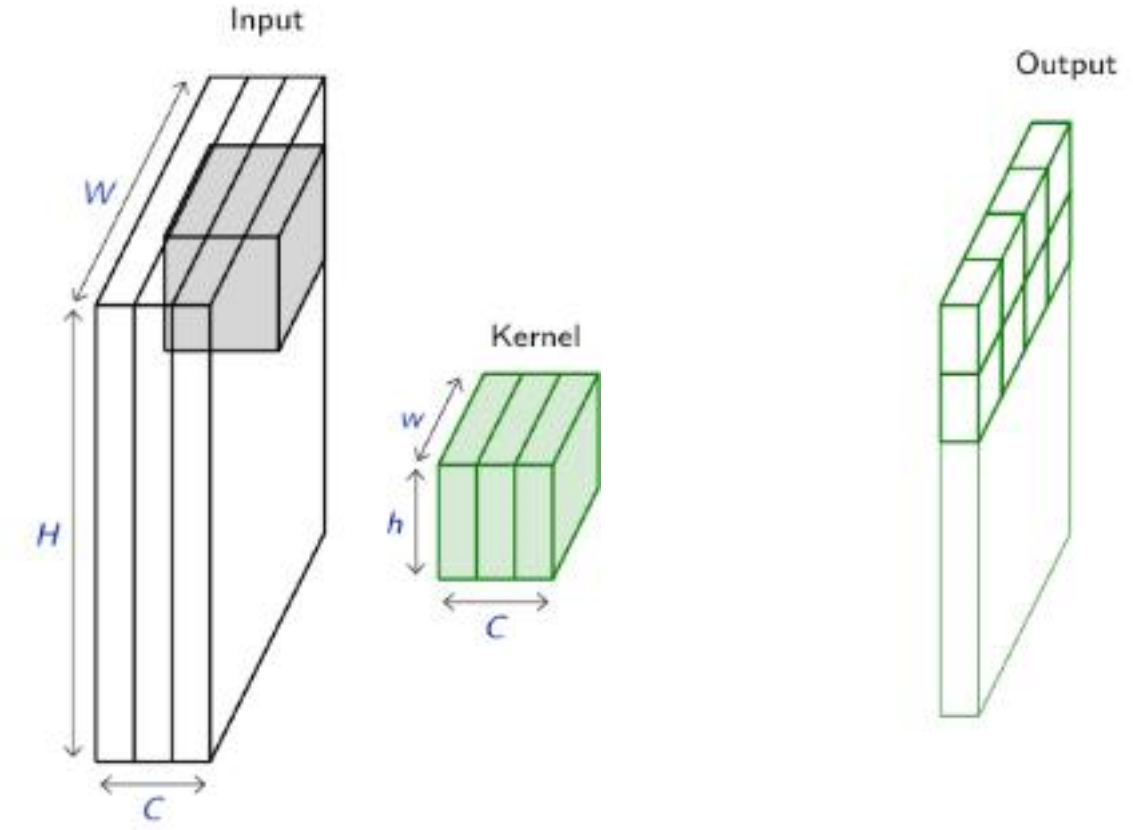
Slide credit: F. Fleuret, EE-559 Deep learning

Convolution 2d



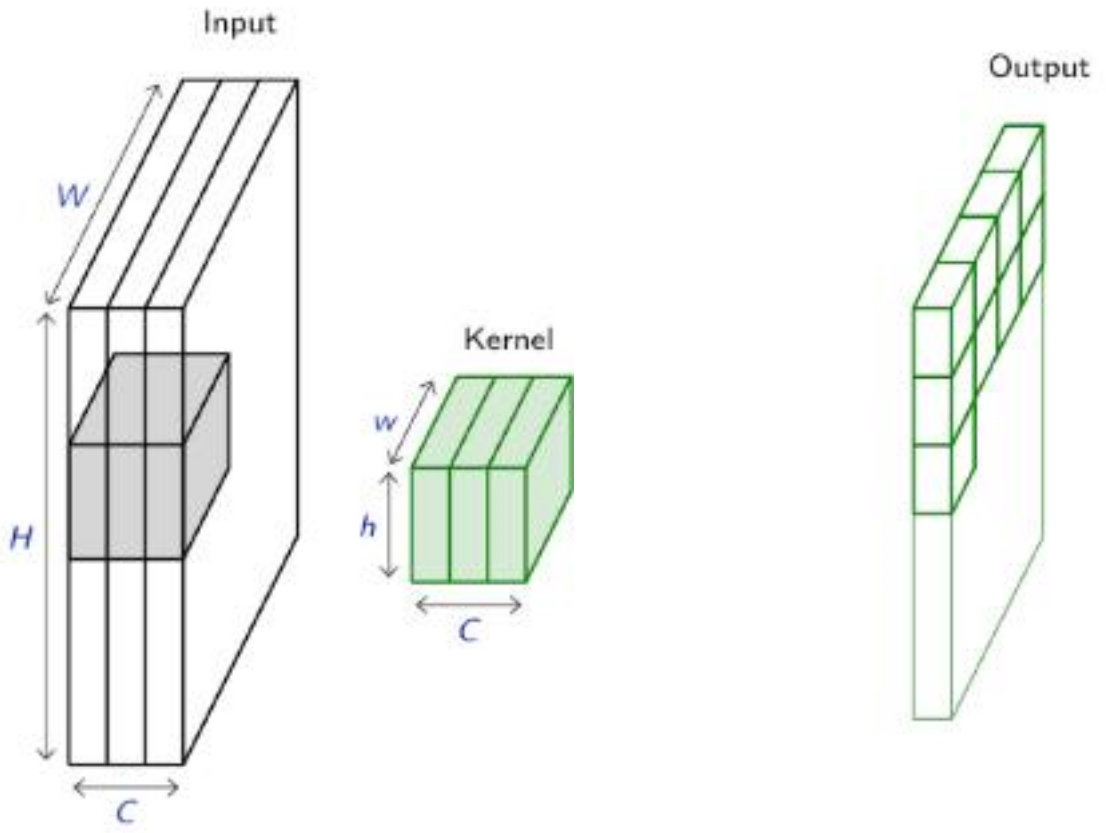
Slide credit: F. Fleuret, EE-559 Deep learning

Convolution 2d



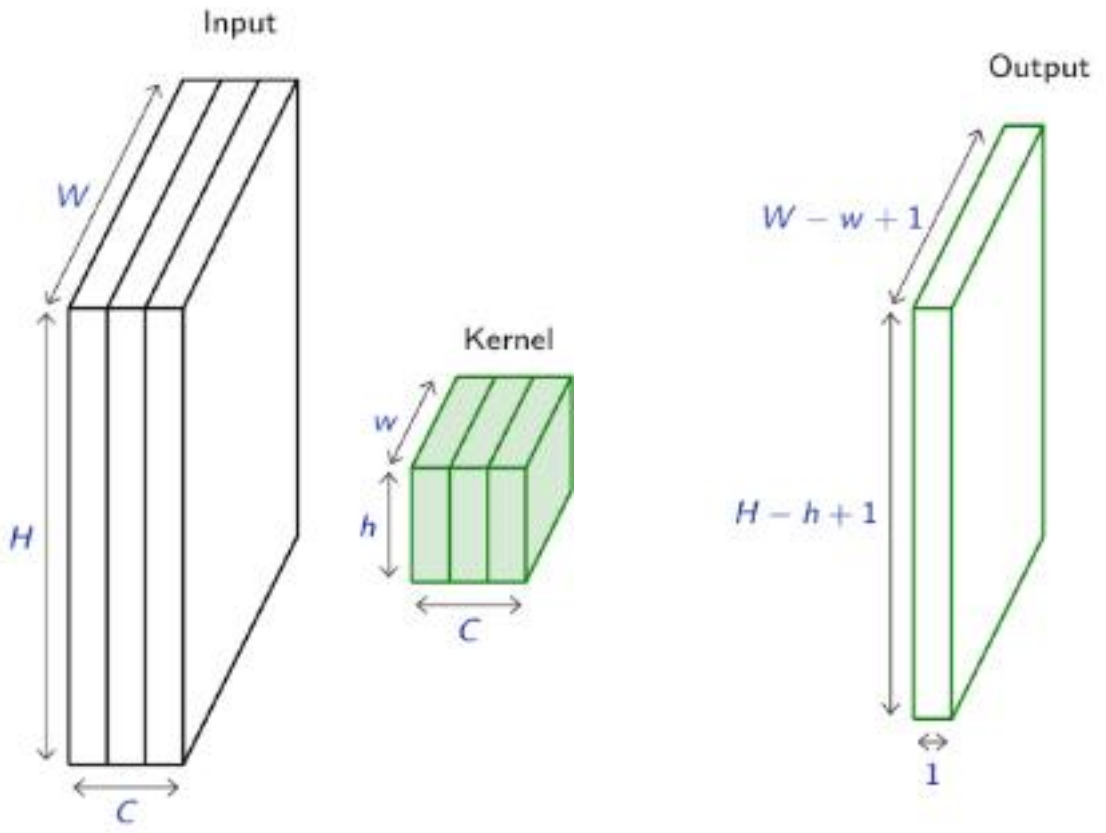
Slide credit: F. Fleuret, EE-559 Deep learning

Convolution 2d



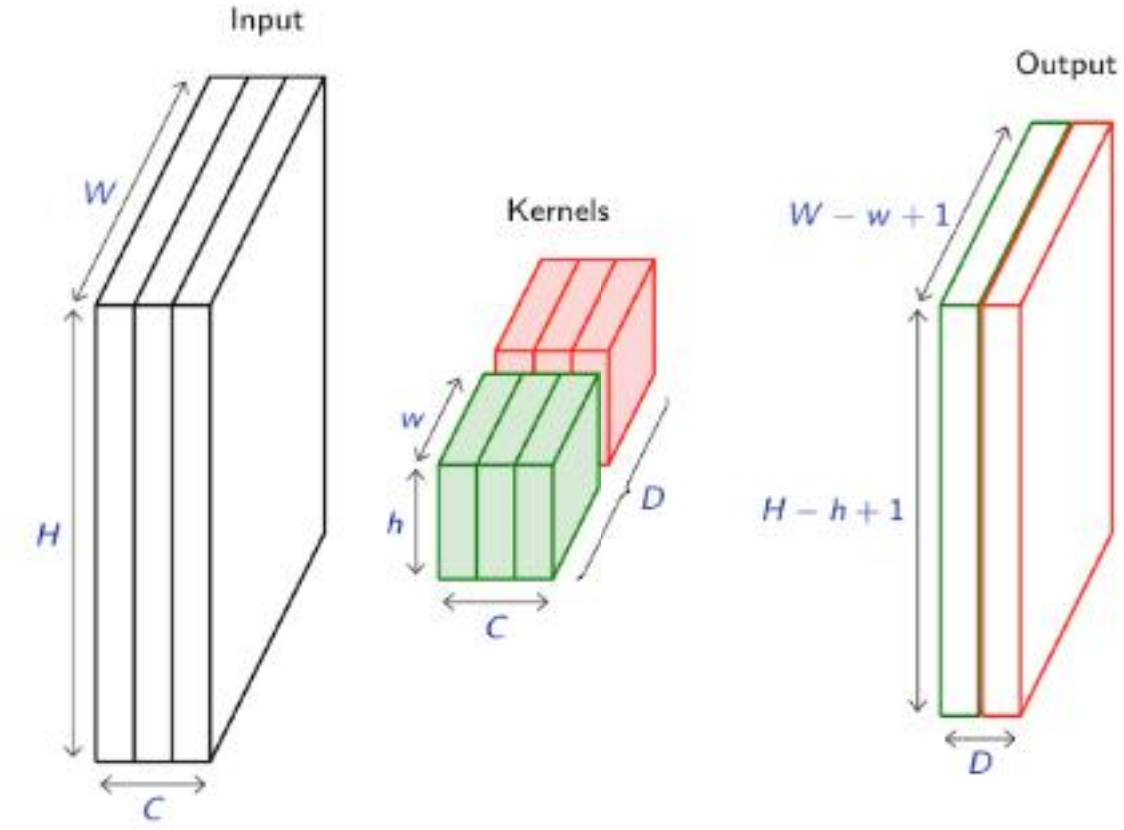
Slide credit: F. Fleuret, EE-559 Deep learning

Convolution 2d



Slide credit: F. Fleuret, EE-559 Deep learning

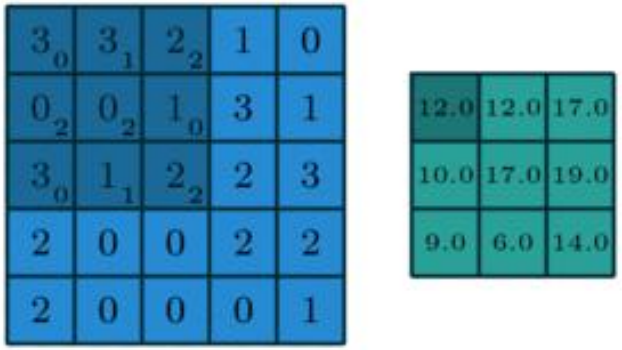
Convolution 2d



Slide credit: F. Fleuret, EE-559 Deep learning

A convolution on an image

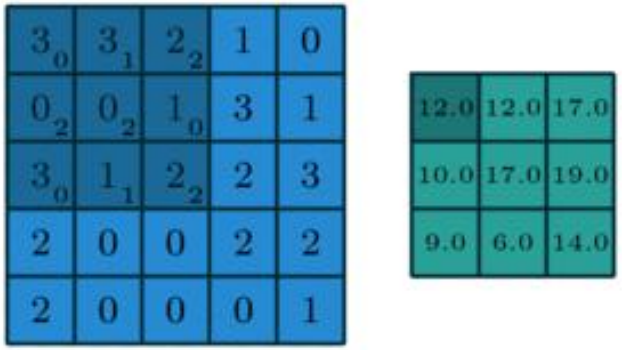
- Image: *im* of dimensions 5×5
- Kernel: *k* of dimensions 3×3



These slides extensively use convolution visualisation by V. Dumoulin available at https://github.com/vdumoulin/conv_arithmetic

A convolution on an image

- Image: im of dimensions 5×5
- Kernel: k of dimensions 3×3

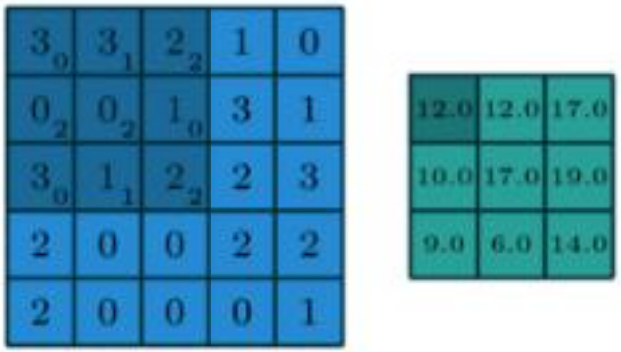


$$(k \star im)(x, y) = \sum_{n=0}^{2} \sum_{m=0}^{2} k(n, m).\, im(x + n - 1, y + m - 1)$$

These slides extensively use convolution visualisation by V. Dumoulin available at https://github.com/vdumoulin/conv_arithmetic

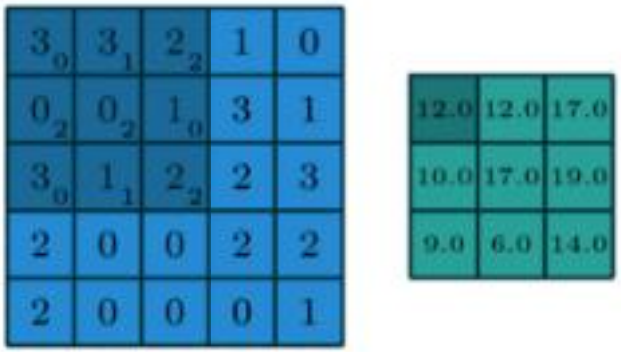
Slide credit: C. Ollion & O. Grisel, M2DS Deep Learning

Kernels as neural networks



- x is a 3×3 chunk of the image
- Each output neuron is parametrized with the kernel weights $\mathbf{w}$

Kernels as neural networks



- x is a 3×3 chunk of the image
- Each output neuron is parametrized with the kernel weights $\mathbf{w}$

The activation obtained by sliding the 3×3 window and computing:

$$z(x) = relu(\mathbf{w}^T x + b)$$

Slide credit: C. Ollion & O. Grisel, M2DS Deep Learning

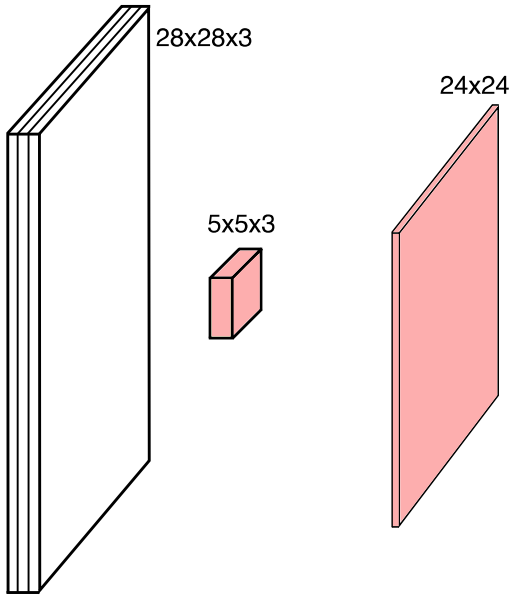
Channels

Colored image = tensor of shape (height, width, channels)

Channels

Colored image = tensor of shape (height, width, channels)

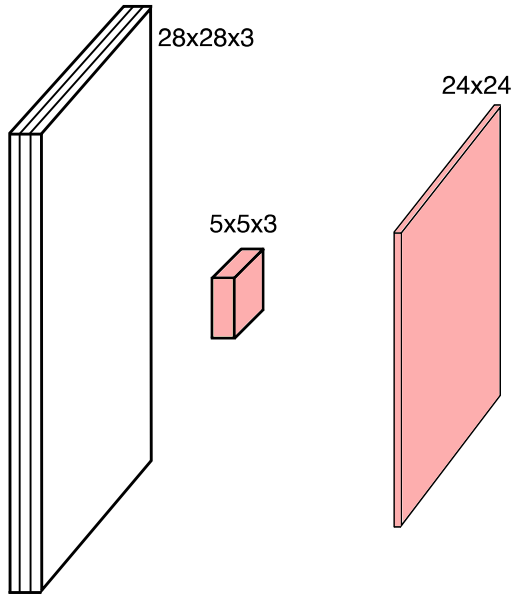
Convolutions can be computed across channels:



Channels

Colored image = tensor of shape (height, width, channels)

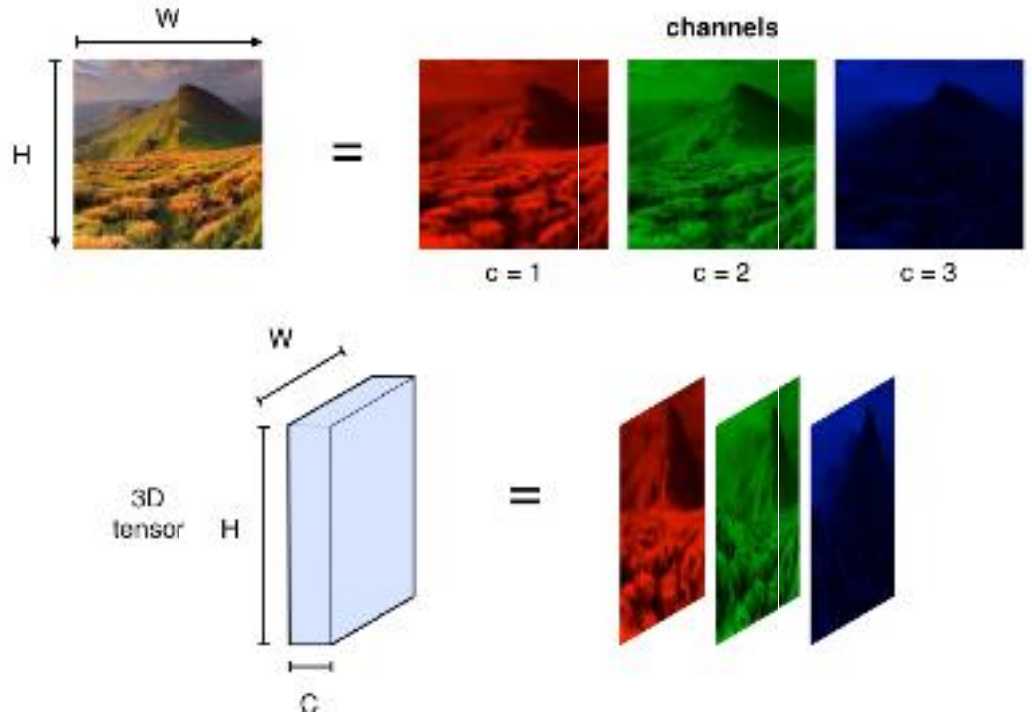
Convolutions can be computed across channels:



$$(k \star im)(x, y) = \sum_{c=0}^{2} \sum_{n=0}^{4} \sum_{m=0}^{4} k(n, m, c).\, im(x + n - 2, y + m - 2, c)$$

Channels

- For first layer, RGB channels of input image can be easily visualized
- Number of channels is typically increased at deeper levels of the network



Multiple convolutions

Each filter generates a one-channel feature map of responses.

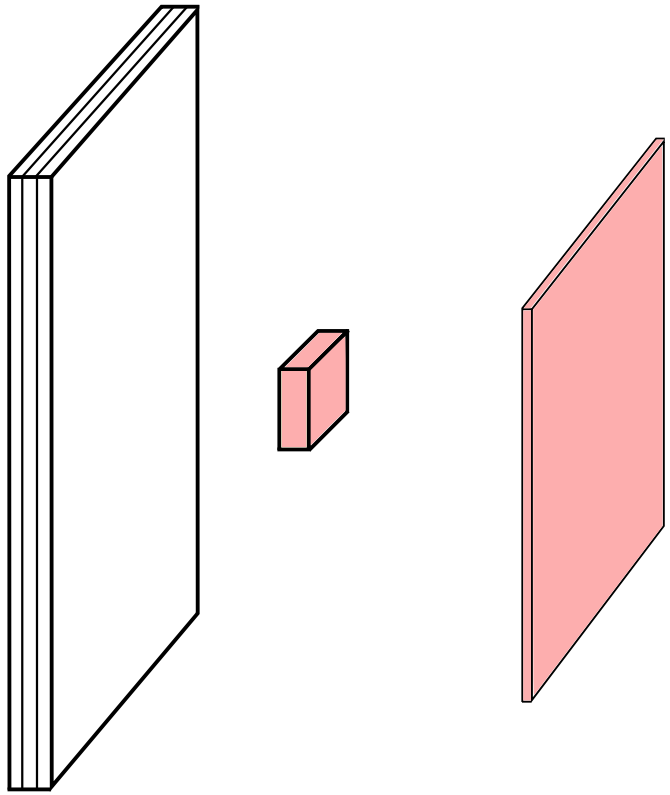


Figure credit: C. Ollion & O. Grisel, M2DS Deep Learning

Multiple convolutions

Each filter generates a one-channel feature map of responses.

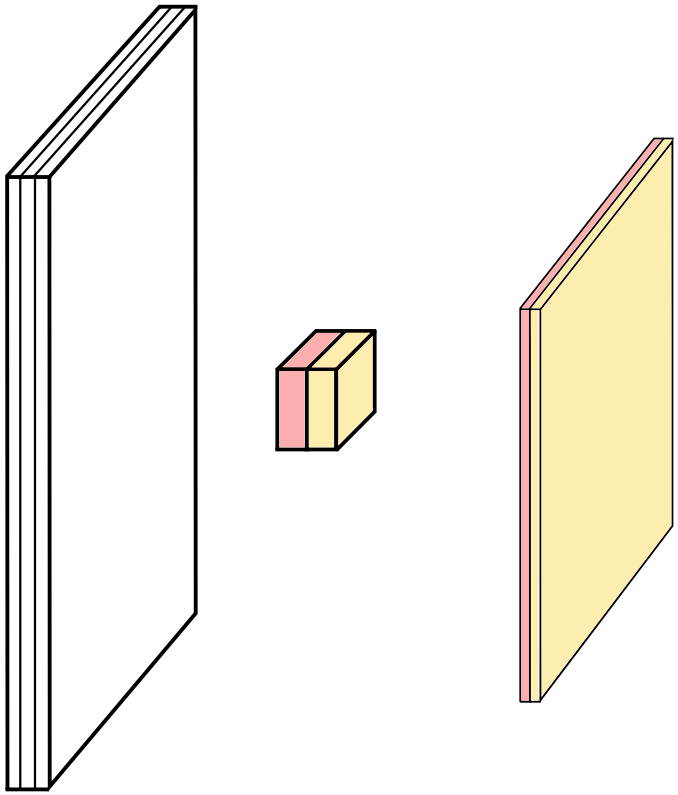


Figure credit: C. Ollion & O. Grisel, M2DS Deep Learning

Multiple convolutions

Each filter generates a one-channel feature map of responses.

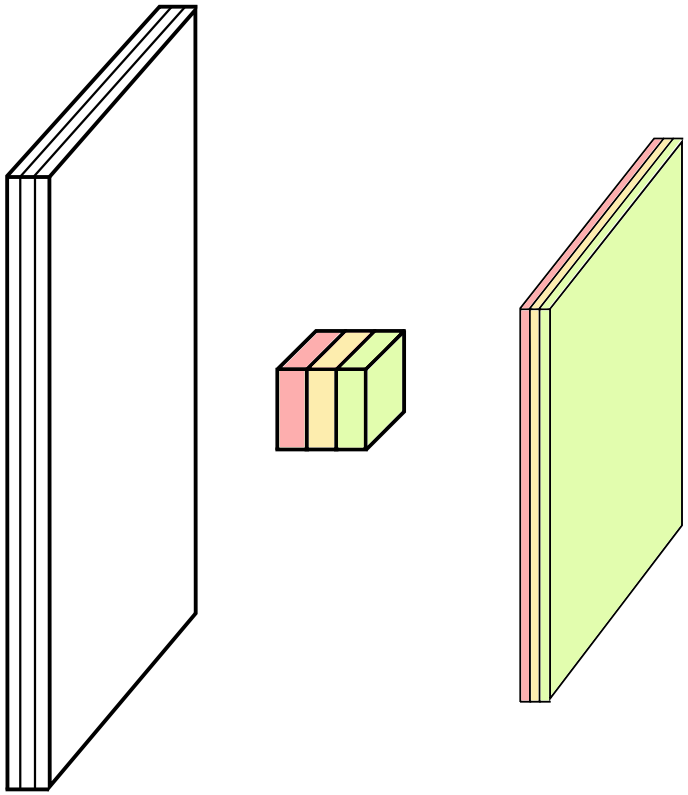


Figure credit: C. Ollion & O. Grisel, M2DS Deep Learning

Multiple convolutions

Each filter generates a one-channel feature map of responses.

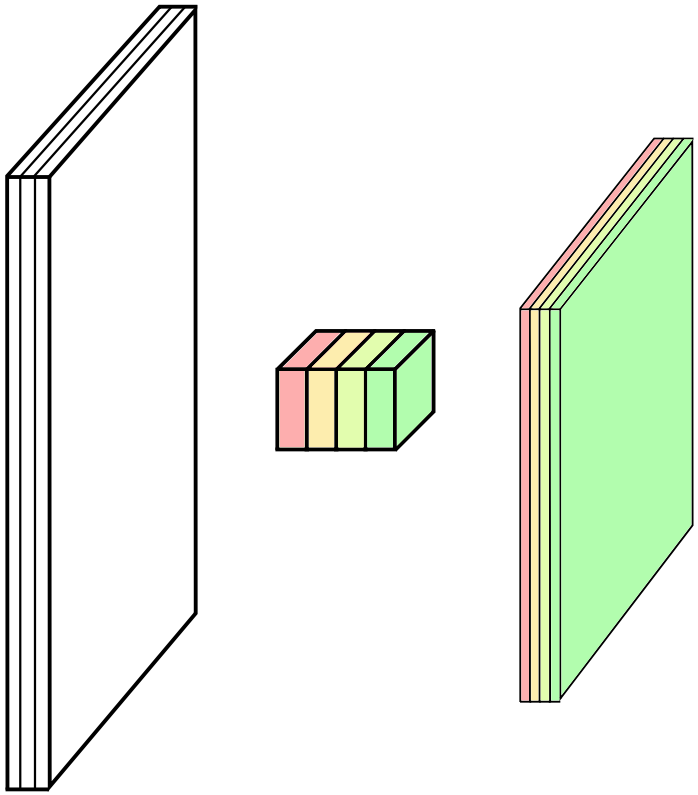
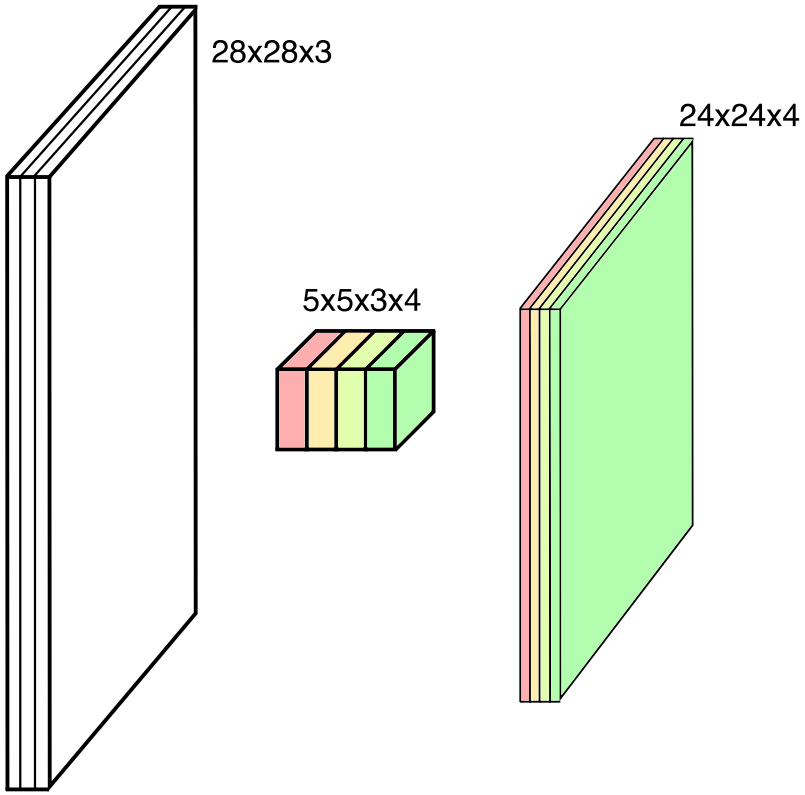


Figure credit: C. Ollion & O. Grisel, M2DS Deep Learning

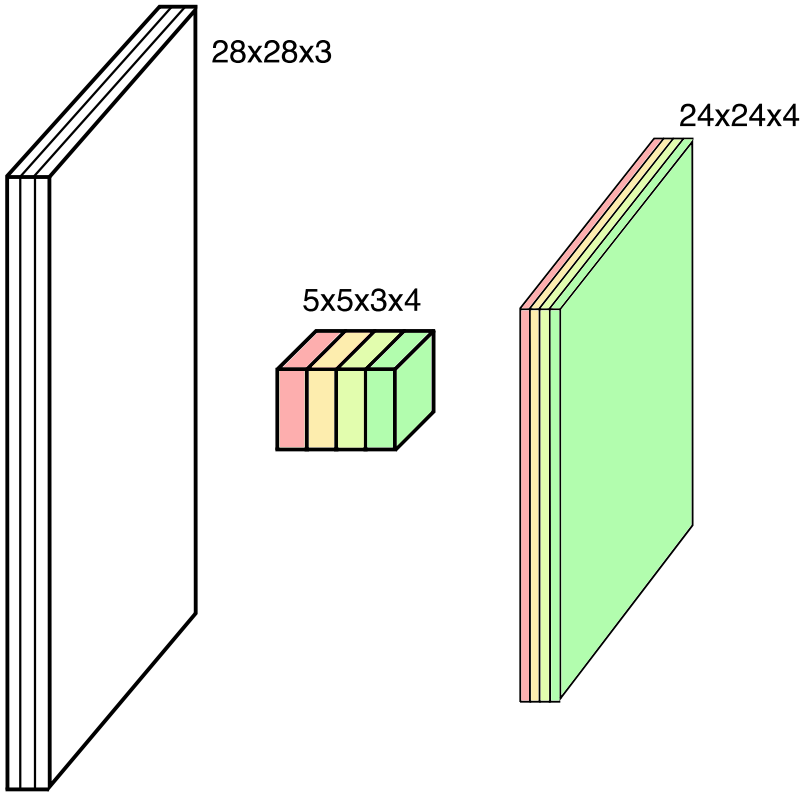
Multiple convolutions

Each filter generates a one-channel feature map of responses.



Multiple convolutions

Each filter generates a one-channel feature map of responses.

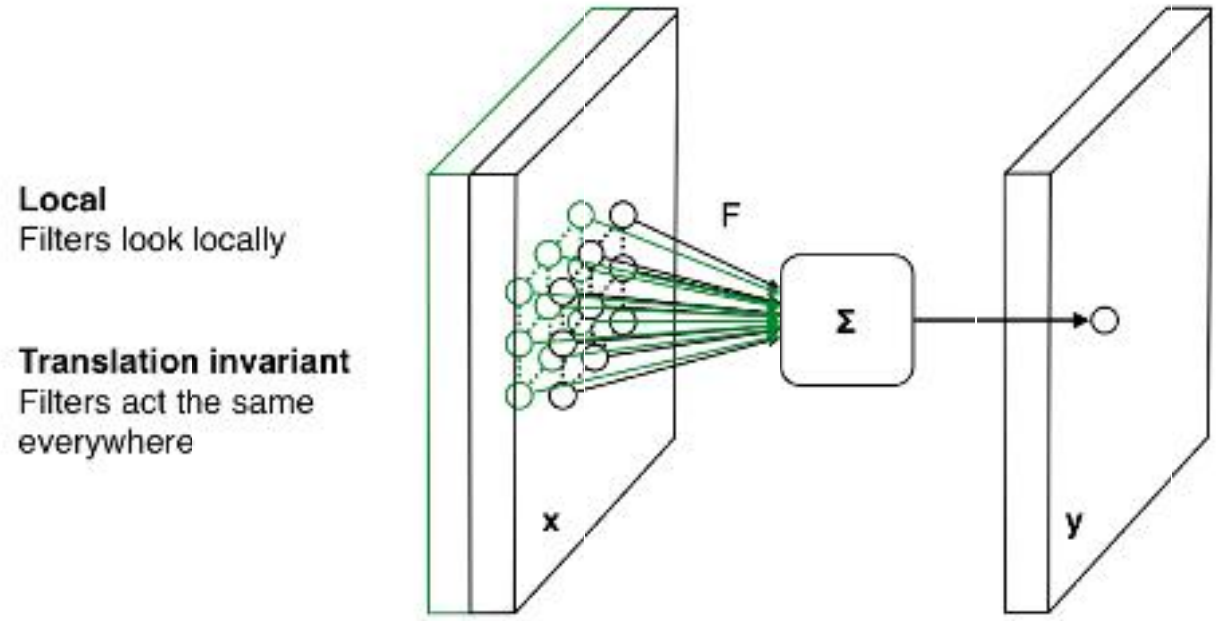


- Kernel size aka receptive field (usually $1, 3, 5, 7, 11$)
- Ouput dimension: length - kernel_size + 1

Figure credit: C. Ollion & O. Grisel, M2DS Deep Learning

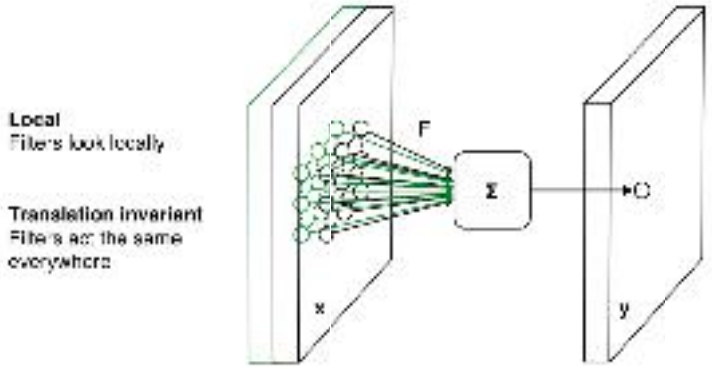
Multiple convolutions

- Since convolutions output one scalar at a time, they can be seen as an individual neuron from a MLP with a receptive field limited to the dimensions of the kernel
- The same neuron is "fired" over multiple areas from the input.

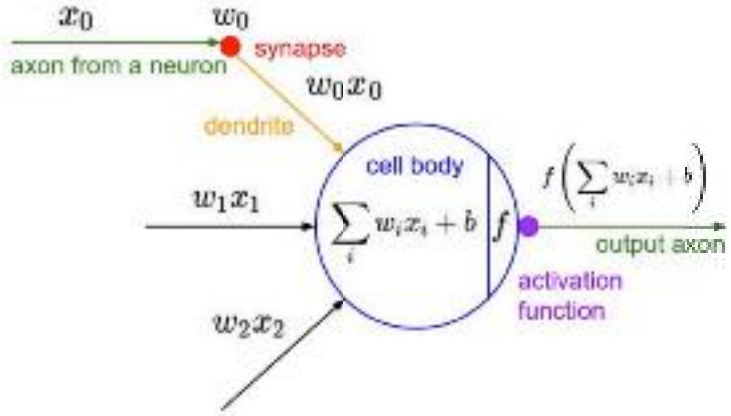


Multiple convolutions

- Since convolutions output one scalar at a time, they can be seen as an individual neuron from a MLP with a receptive field limited to the dimensions of the kernel
- The same neuron is "fired" over multiple areas from the input.

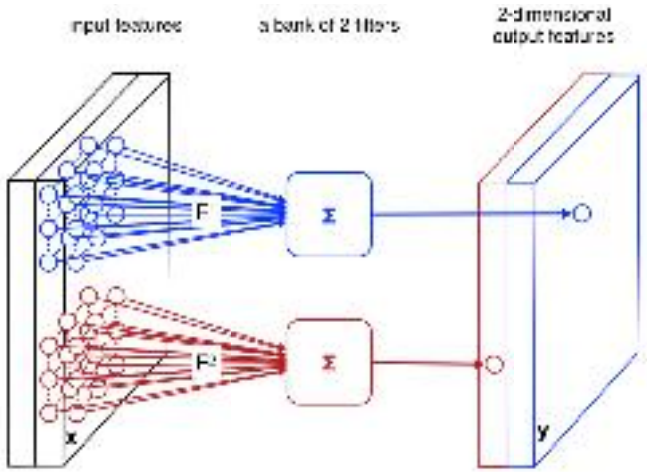


Remember this?

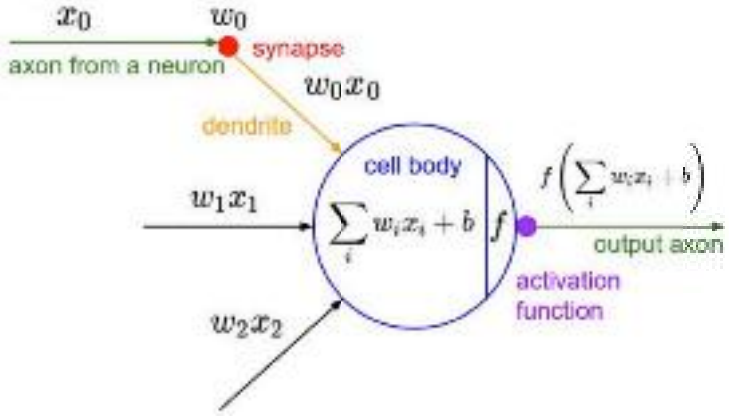


Multiple convolutions

- Since convolutions output one scalar at a time, they can be seen as an individual neuron from a MLP with a receptive field limited to the dimensions of the kernel
- The same neuron is "fired" over multiple areas from the input.

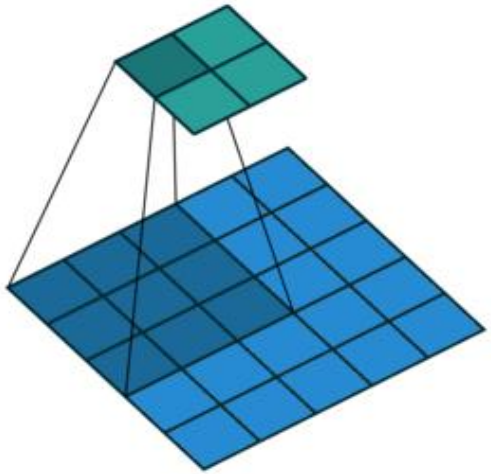


Remember this?



Strides

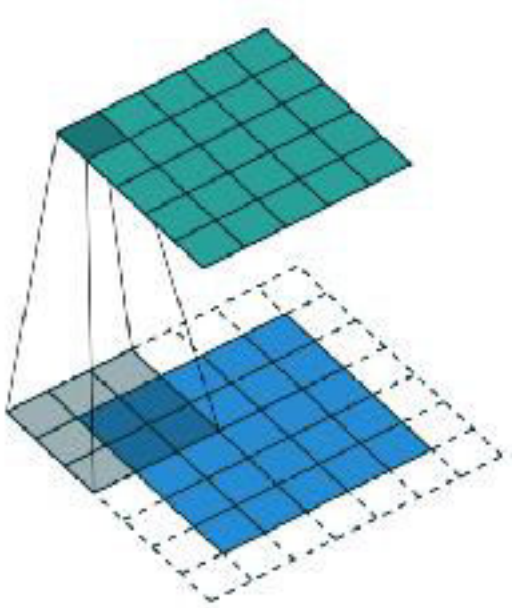
- Strides: increment step size for the convolution operator
- Reduces the size of the ouput map



Example with kernel size 3×3 and a stride of 2 (image in blue)

Padding

- Padding: artifically fill borders of image
- Useful to keep spatial dimension constant across filters
- Useful with strides and large receptive fields
- Usually: fill with 0s



Padding

- Example: input $C \times 3 \times 5$

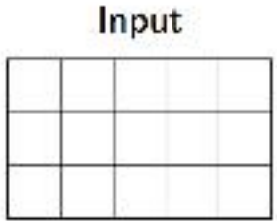


Figure credit: F. Fleuret, EE-559 Deep learning

Padding

- Example: input $C \times 3 \times 5$, padding of (2, 1)

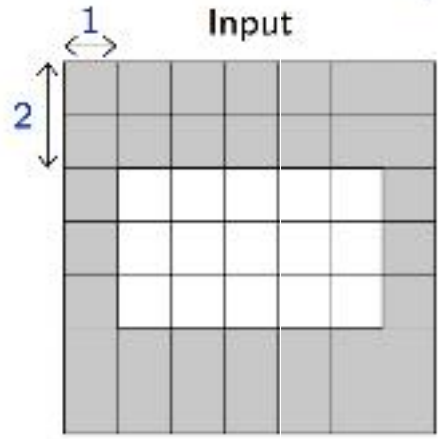


Figure credit: F. Fleuret, EE-559 Deep learning

Padding

- Example: input $C \times 3 \times 5$, padding of $(2, 1)$, a stride of $(2, 2)$

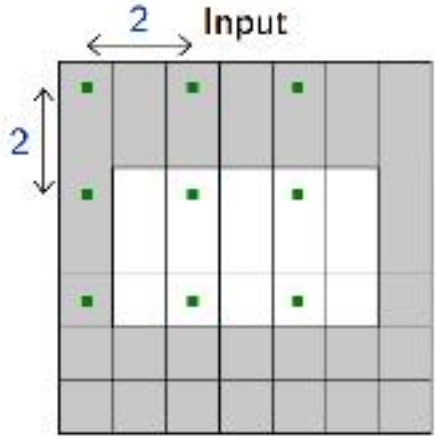


Figure credit: F. Fleuret, EE-559 Deep learning

Padding

- Example: input $C \times 3 \times 5$, padding of $(2, 1)$, a stride of $(2, 2)$, kernel of size $C \times 3 \times 5$

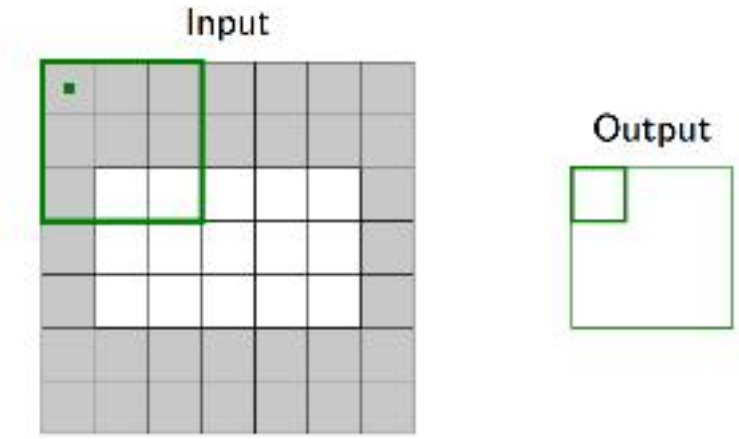


Figure credit: F. Fleuret, EE-559 Deep learning

Padding

- Example: input $C \times 3 \times 5$, padding of $(2, 1)$, a stride of $(2, 2)$, kernel of size $C \times 3 \times 5$

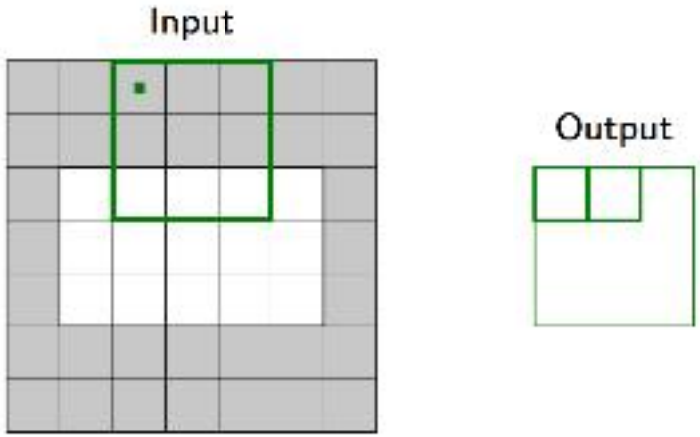


Figure credit: F. Fleuret, EE-559 Deep learning

Padding

- Example: input $C \times 3 \times 5$, padding of $(2, 1)$, a stride of $(2, 2)$, kernel of size $C \times 3 \times 5$

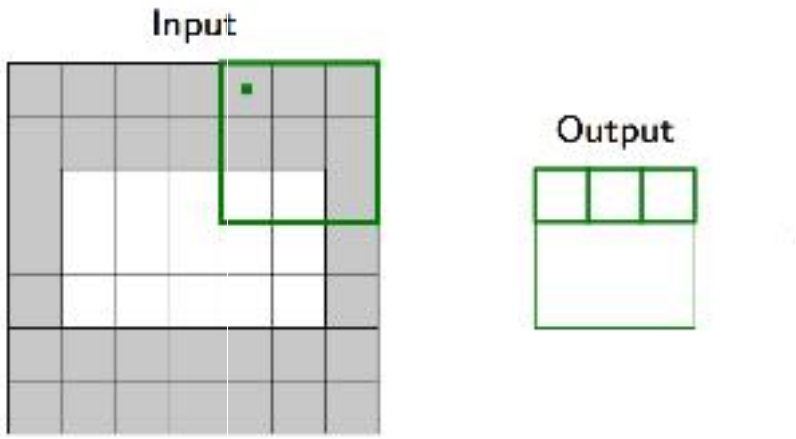


Figure credit: F. Fleuret, EE-559 Deep learning

Padding

- Example: input $C \times 3 \times 5$, padding of $(2, 1)$, a stride of $(2, 2)$, kernel of size $C \times 3 \times 5$

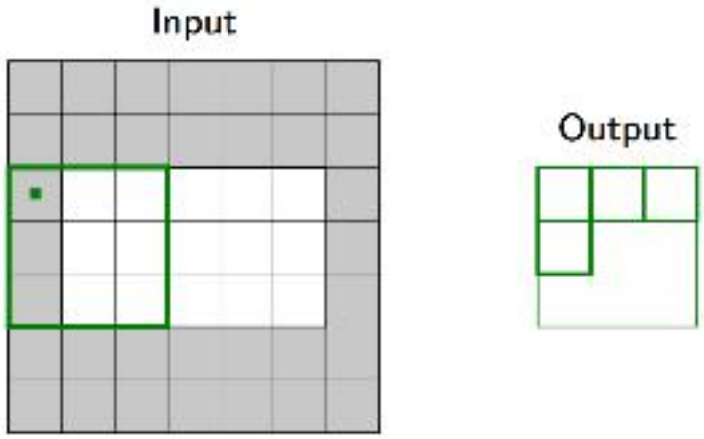


Figure credit: F. Fleuret, EE-559 Deep learning

Padding

- Example: input $C \times 3 \times 5$, padding of $(2, 1)$, a stride of $(2, 2)$, kernel of size $C \times 3 \times 5$

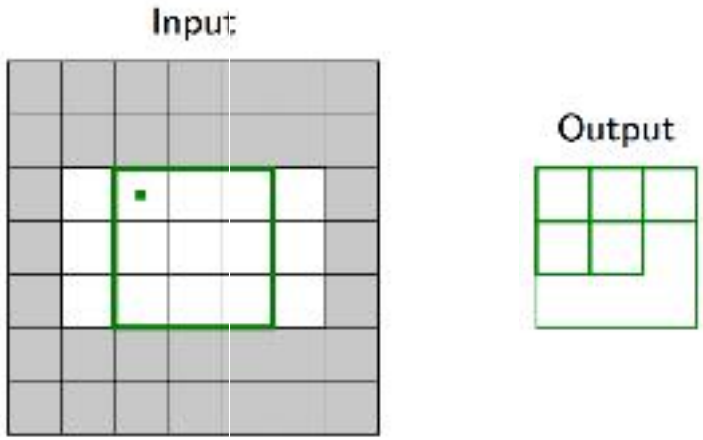


Figure credit: F. Fleuret, EE-559 Deep learning

Padding

- Example: input $C \times 3 \times 5$, padding of $(2, 1)$, a stride of $(2, 2)$, kernel of size $C \times 3 \times 5$

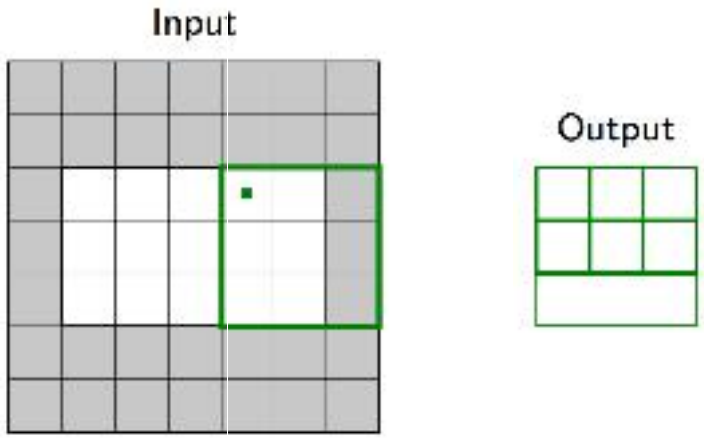


Figure credit: F. Fleuret, EE-559 Deep learning

Padding

- Example: input $C \times 3 \times 5$, padding of $(2, 1)$, a stride of $(2, 2)$, kernel of size $C \times 3 \times 5$

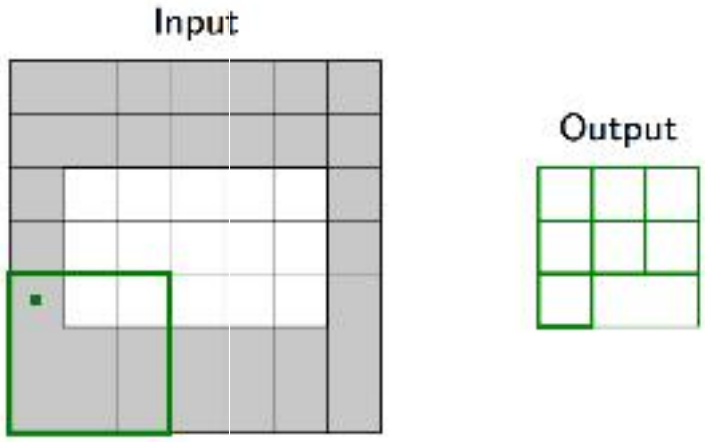


Figure credit: F. Fleuret, EE-559 Deep learning

Padding

- Example: input $C \times 3 \times 5$, padding of $(2, 1)$, a stride of $(2, 2)$, kernel of size $C \times 3 \times 5$

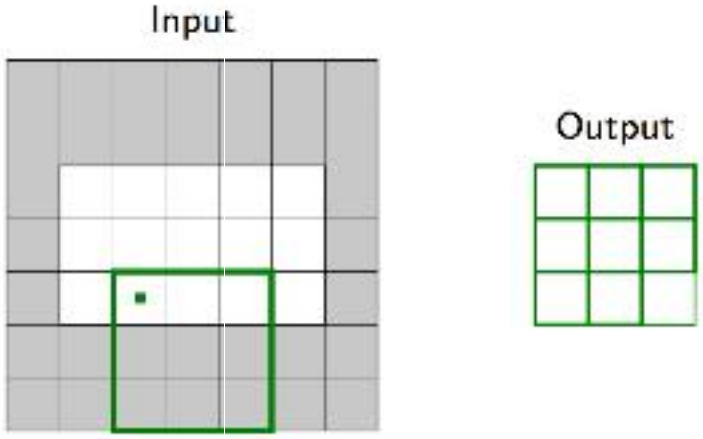


Figure credit: F. Fleuret, EE-559 Deep learning

Padding

- Example: input $C \times 3 \times 5$, padding of $(2, 1)$, a stride of $(2, 2)$, kernel of size $C \times 3 \times 5$

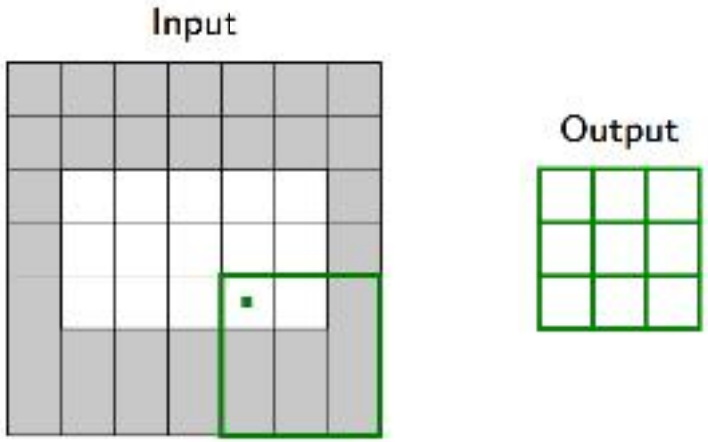


Figure credit: F. Fleuret, EE-559 Deep learning

Padding

- Example: input $C \times 3 \times 5$, padding of $(2, 1)$, a stride of $(2, 2)$, kernel of size $C \times 3 \times 5$

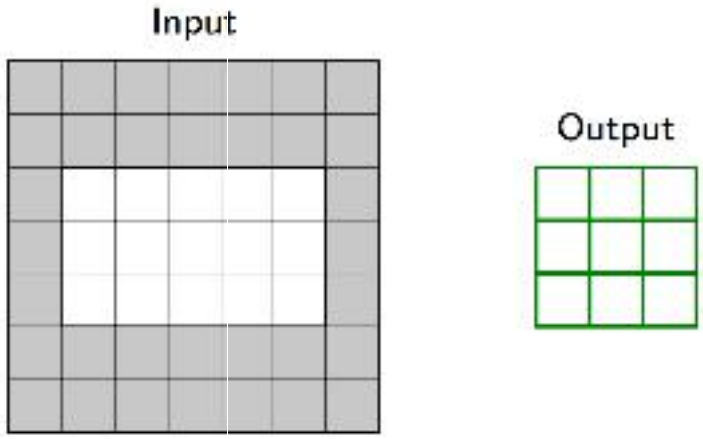
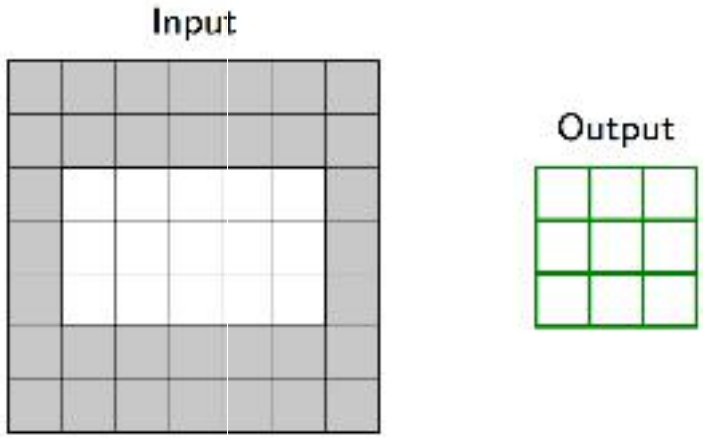


Figure credit: F. Fleuret, EE-559 Deep learning

Padding

- Example: input $C \times 3 \times 5$, padding of $(2, 1)$, a stride of $(2, 2)$, kernel of size $C \times 3 \times 5$



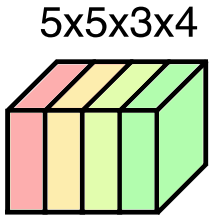
- Pooling operations have a default stride equal to their kernel size, and convolutions have a default stride of 1.
- Padding can be useful to generate an output of same size as the input.

Figure credit: F. Fleuret, EE-559 Deep learning

Dealing with shapes

Kernel shape (F, F, C^i, C^o)

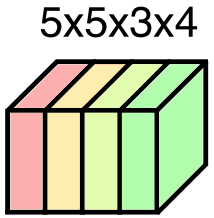
- $F \times F$ kernel size,
- C^i input channels
- C^o output channels



Dealing with shapes

Kernel shape (F, F, C^i, C^o)

- $F \times F$ kernel size,
- C^i input channels
- C^o output channels

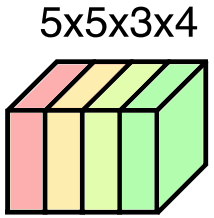


Number of parameters: $(F \times F \times C^i + 1) \times C^o$

Dealing with shapes

Kernel shape (F, F, C^i, C^o)

- $F \times F$ kernel size,
- C^i input channels
- C^o output channels



Number of parameters: $(F \times F \times C^i + 1) \times C^o$

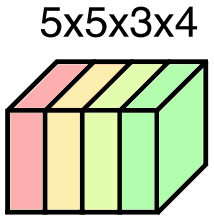
Activation shapes:

- Input (W^i, H^i, C^i)
- Output (W^o, H^o, C^o)

Dealing with shapes

Kernel shape (F, F, C^i, C^o)

- $F \times F$ kernel size,
- C^i input channels
- C^o output channels



Number of parameters: $(F \times F \times C^i + 1) \times C^o$

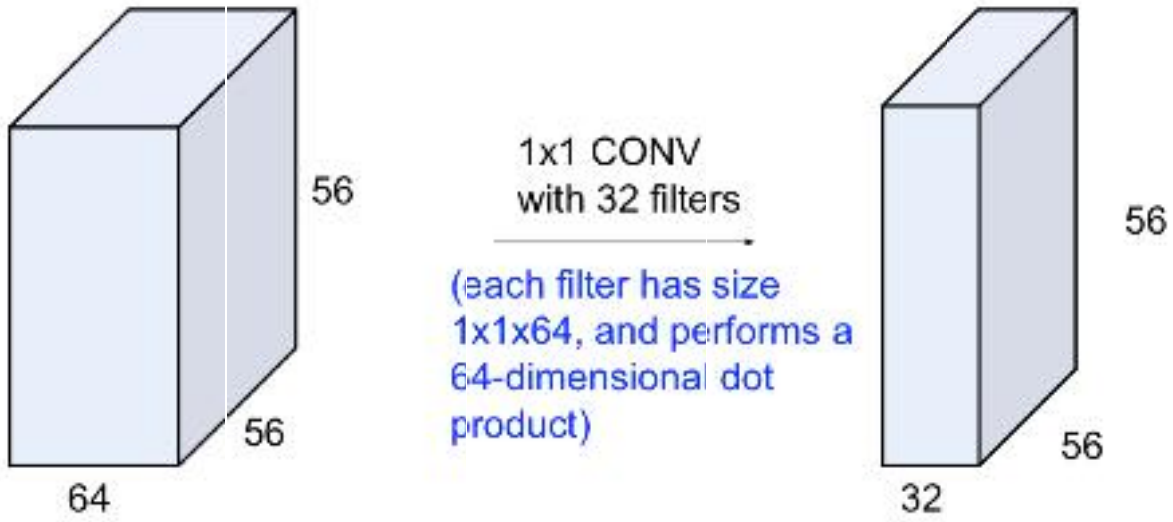
Activation shapes:

- Input (W^i, H^i, C^i)
- Output (W^o, H^o, C^o)

$W^o = (W^i - F + 2P)/S + 1$

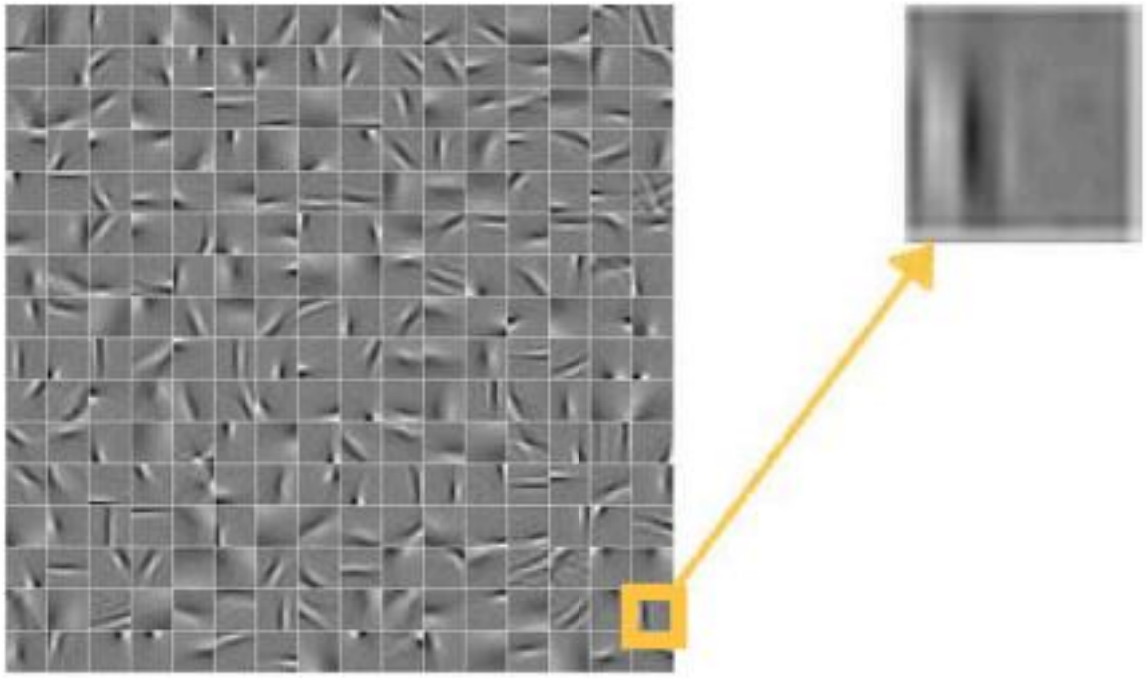
Convolutions

1x1 convolution layers: aggregating pixel information from all feature maps



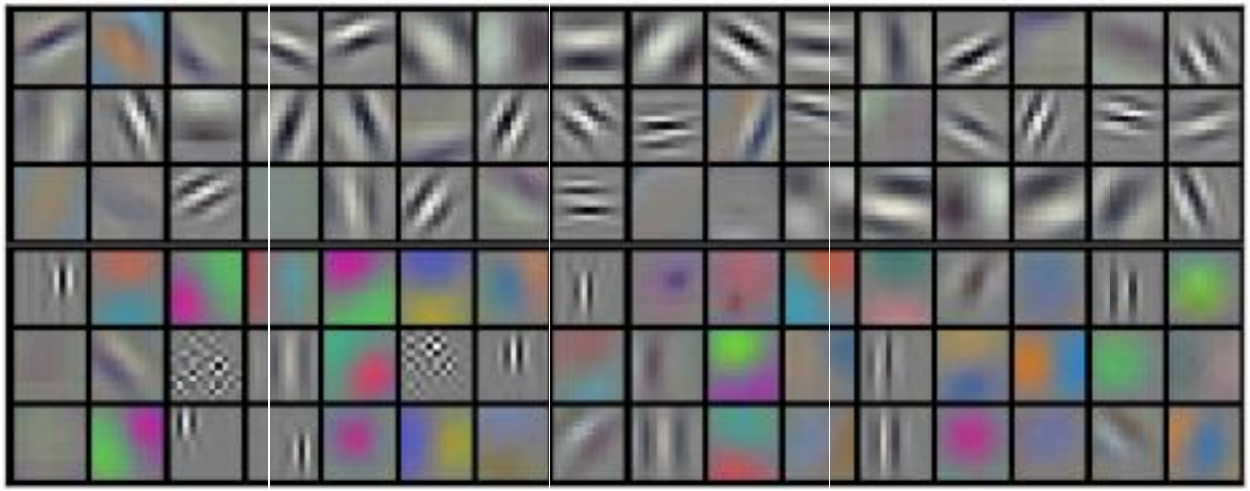
Convolutions

- A bank of 256 filters (learned from data)
- Each filter is 1d (it applies to a grayscale image)
- Each filter is 16 x 16 pixels



Convolutions

- A bank of 256 filters (learned from data)
- 3D filters for RGB inputs



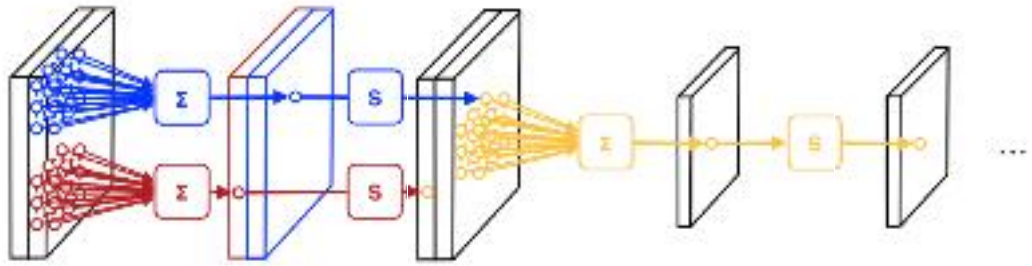
Convolutions

Implementation

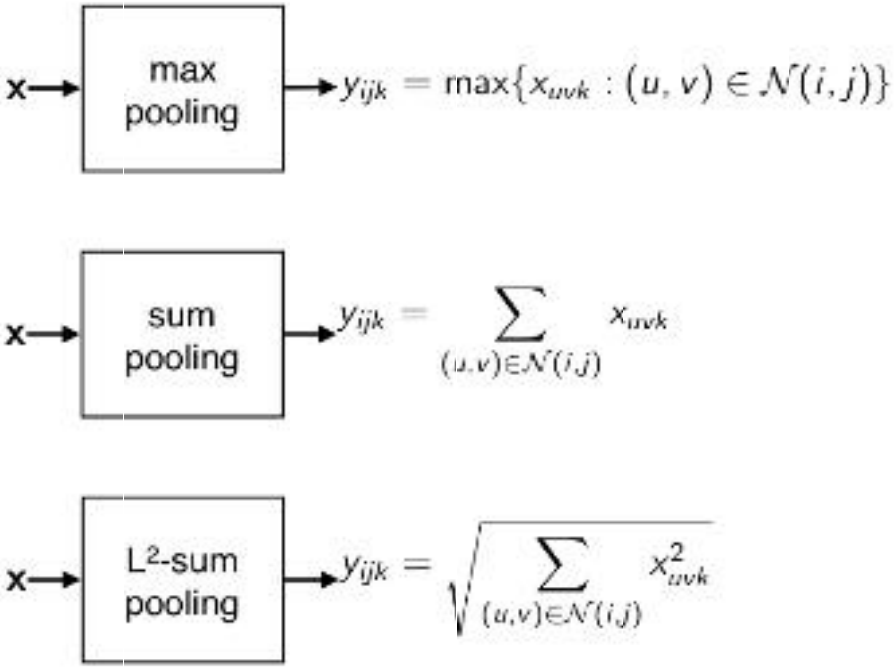
- Arrange data for optimized matrix multiplication (using GEMM)
- Makes life easier for backprop

Downsampling

- Downsampling by a factor S amount to keeping only one every S pixels, discarding others
- Filter banks often incorporate or are followed by 2x output downsampling
- Downsampling is often matched with an increase in the number of feature channels
- Overall the volume of the tensors decreases slowly



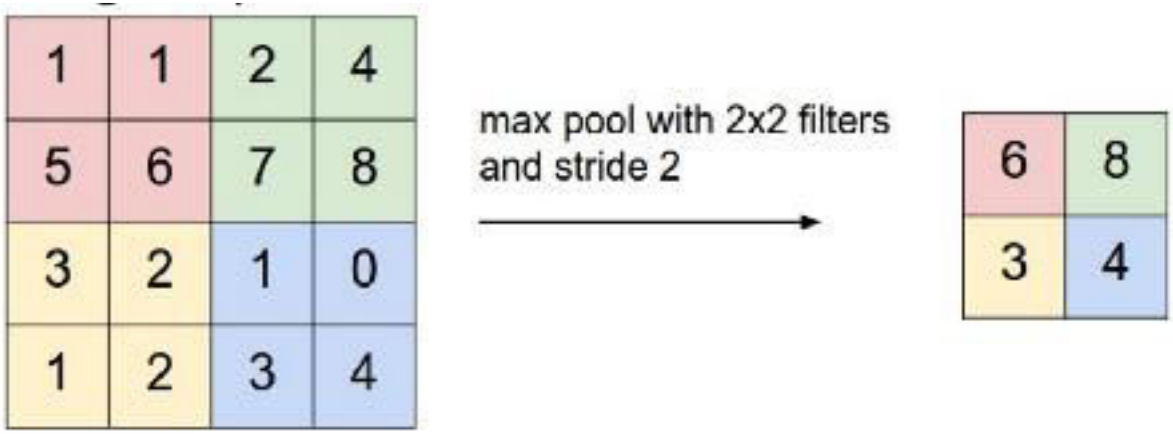
Spatial pooling



By far, the most common variant is **max pooling**.

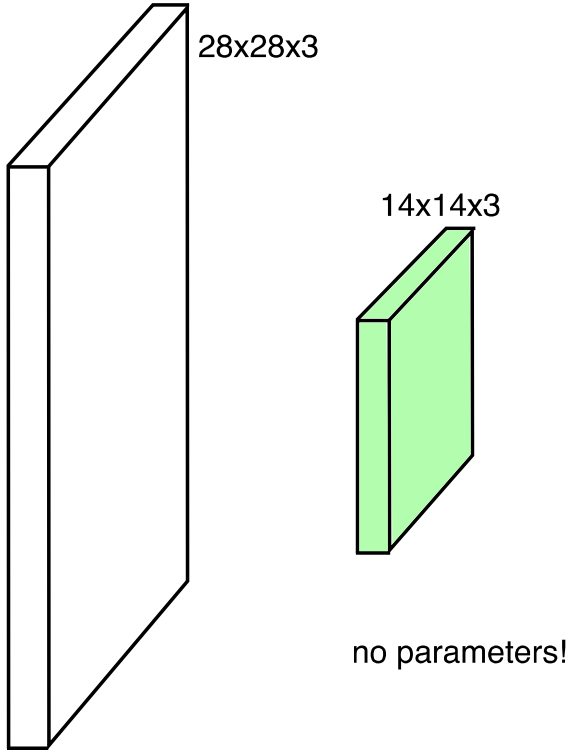
Pooling

- Spatial dimension reduction
- Local invariance
- No parameters: max or average of 2x2 units

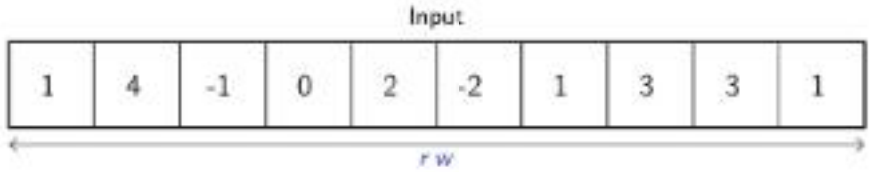


Pooling

- Spatial dimension reduction
- Local invariance
- No parameters: max or average of 2x2 units

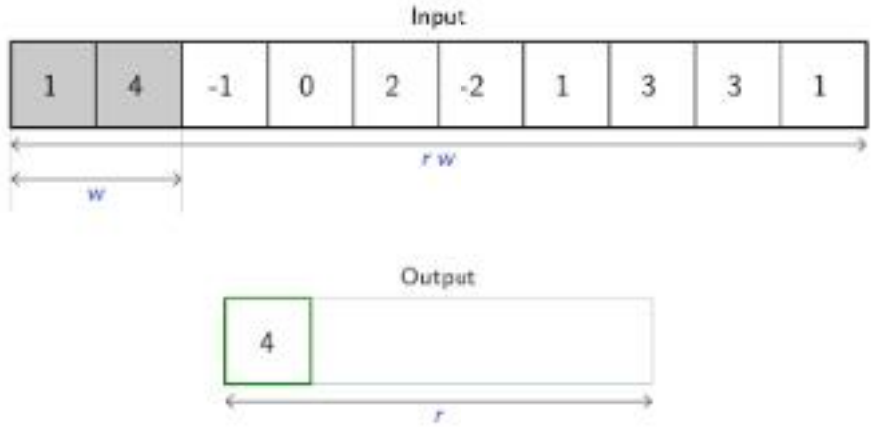


Max-Pooling 1d



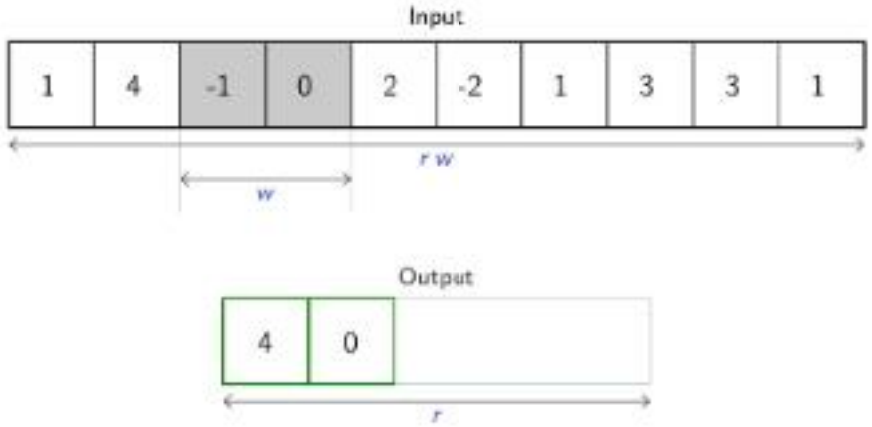
Slide credit: F. Fleuret, EE-559 Deep learning

Max-Pooling 1d



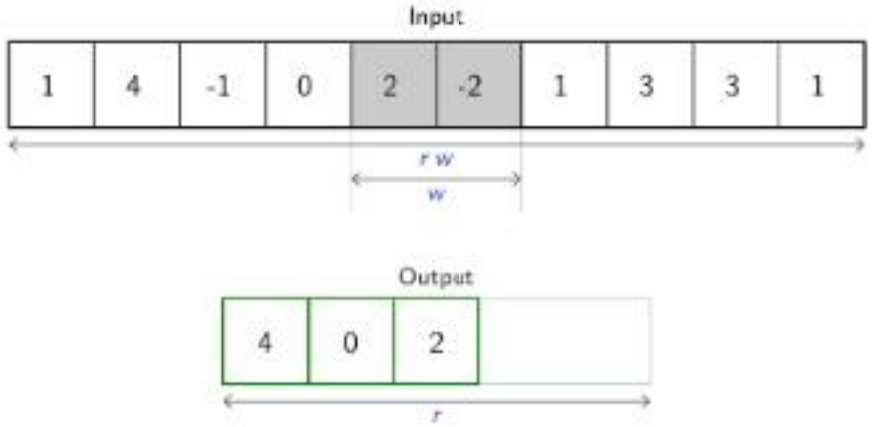
Slide credit: F. Fleuret, EE-559 Deep learning

Max-Pooling 1d



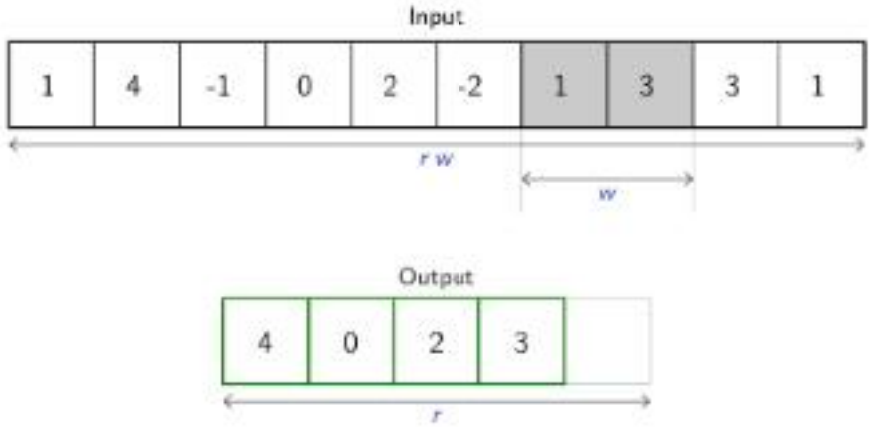
Slide credit: F. Fleuret, EE-559 Deep learning

Max-Pooling 1d



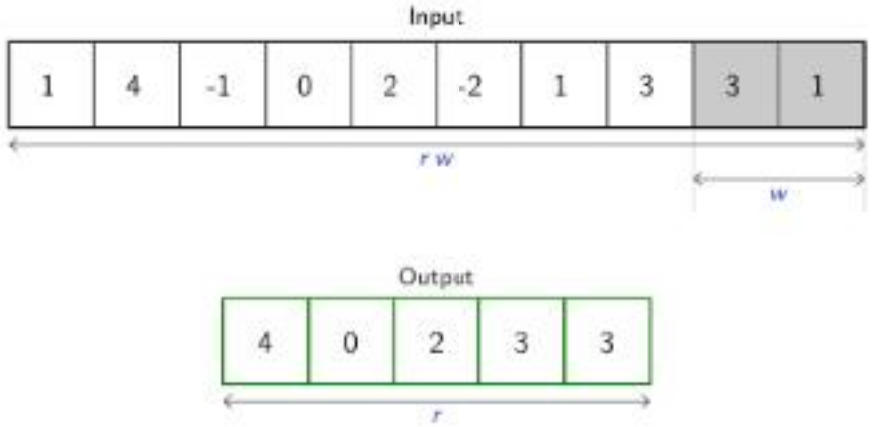
Slide credit: F. Fleuret, EE-559 Deep learning

Max-Pooling 1d



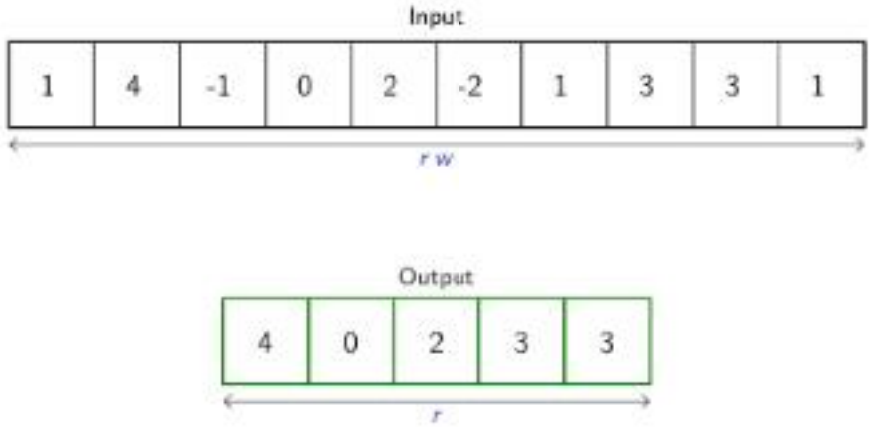
Slide credit: F. Fleuret, EE-559 Deep learning

Max-Pooling 1d



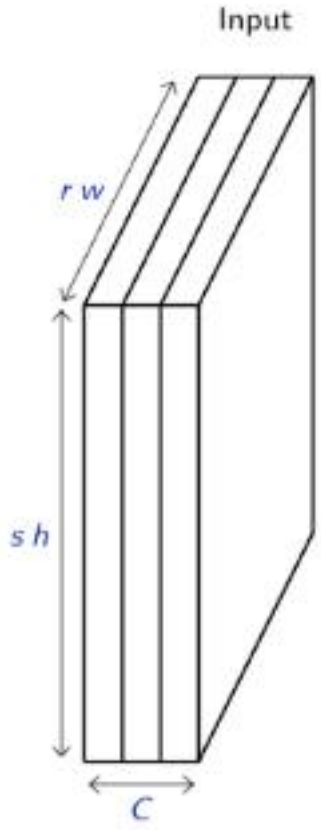
Slide credit: F. Fleuret, EE-559 Deep learning

Max-Pooling 1d



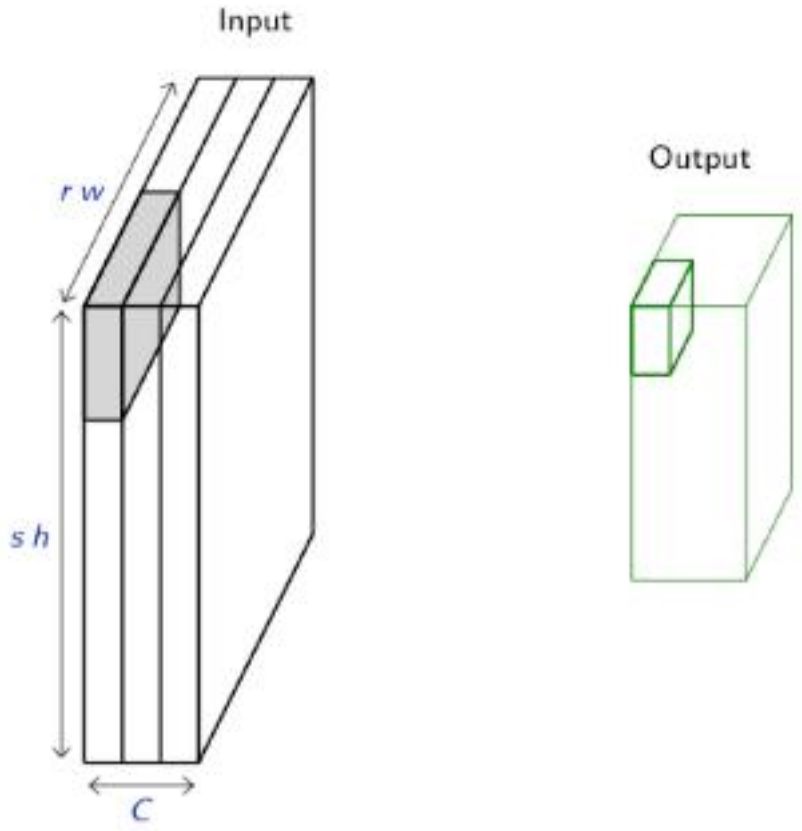
Slide credit: F. Fleuret, EE-559 Deep learning

Max-Pooling 2d



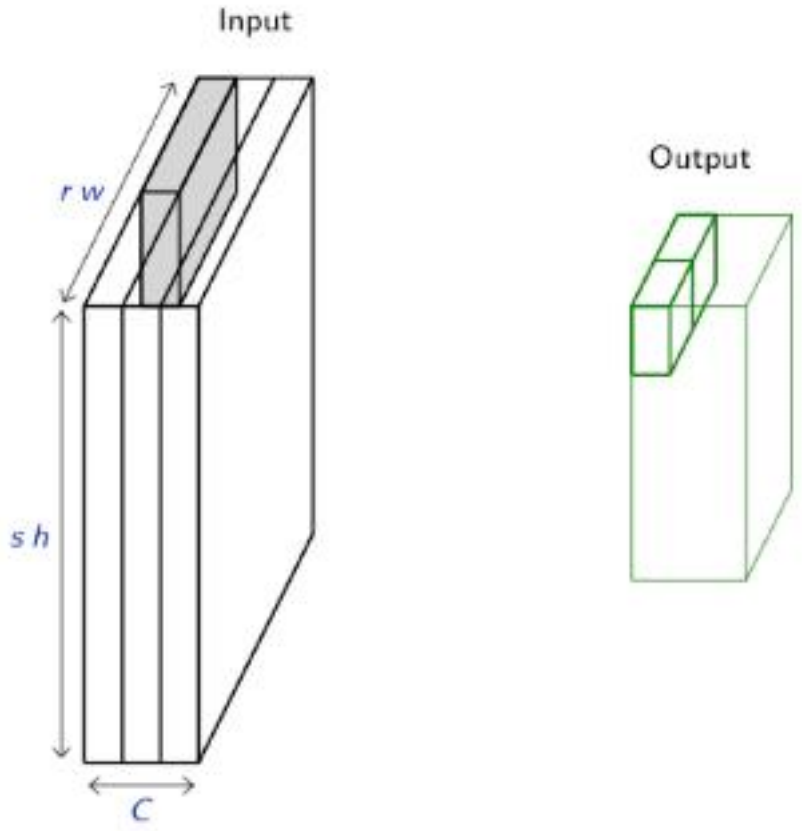
Slide credit: F. Fleuret, EE-559 Deep learning

Max-Pooling 2d



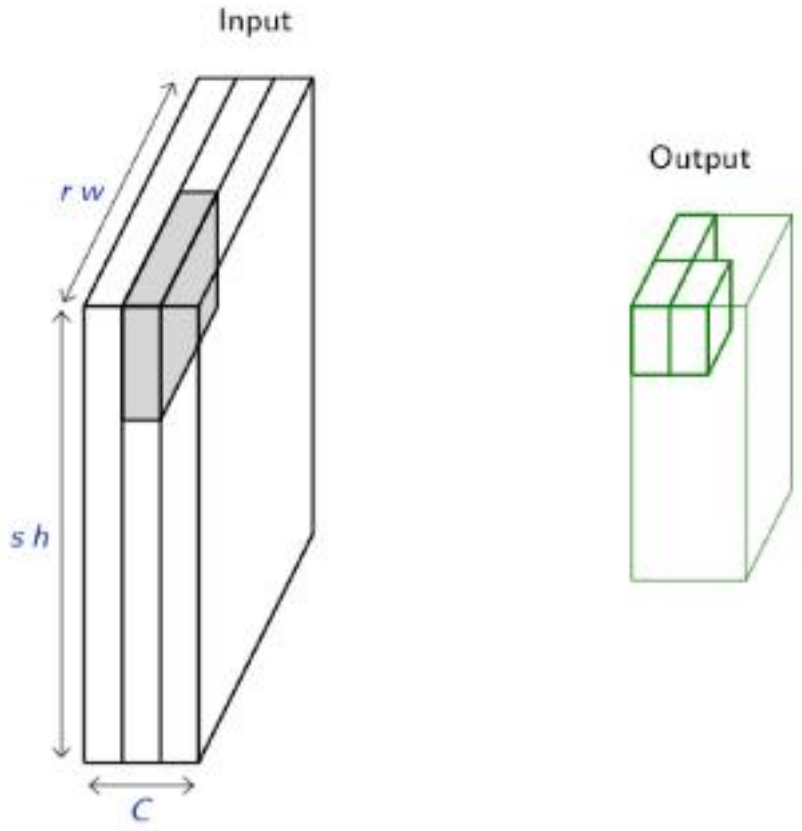
Slide credit: F. Fleuret, EE-559 Deep learning

Max-Pooling 2d



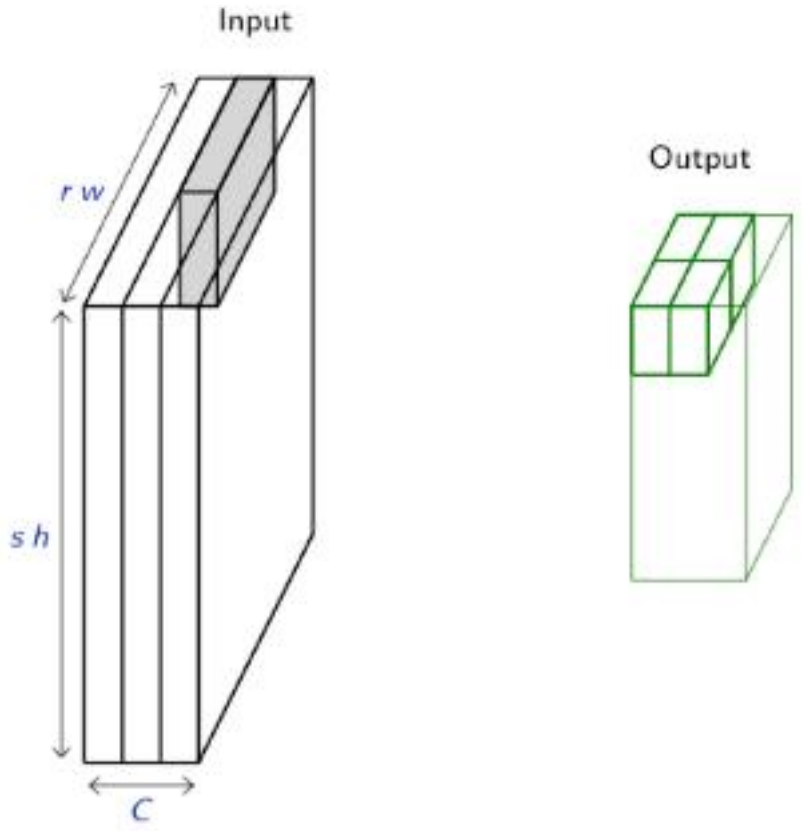
Slide credit: F. Fleuret, EE-559 Deep learning

Max-Pooling 2d



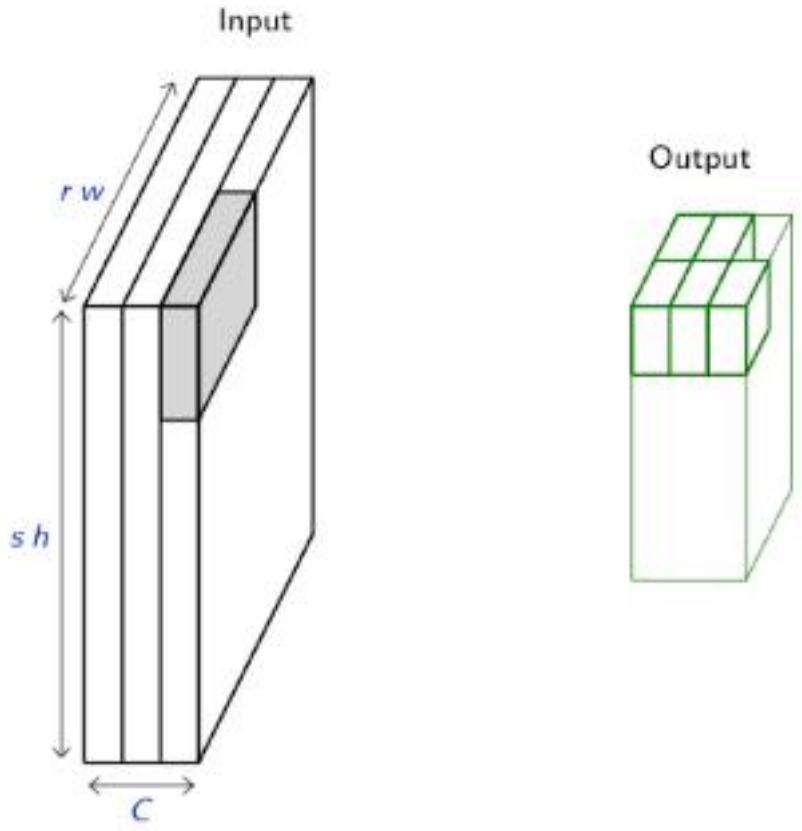
Slide credit: F. Fleuret, EE-559 Deep learning

Max-Pooling 2d



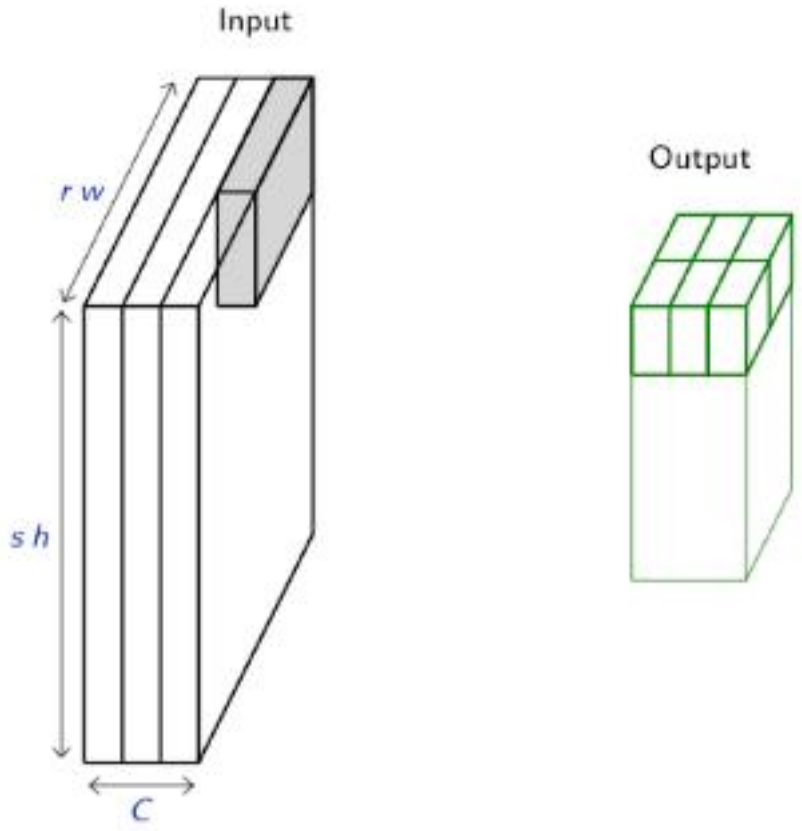
Slide credit: F. Fleuret, EE-559 Deep learning

Max-Pooling 2d



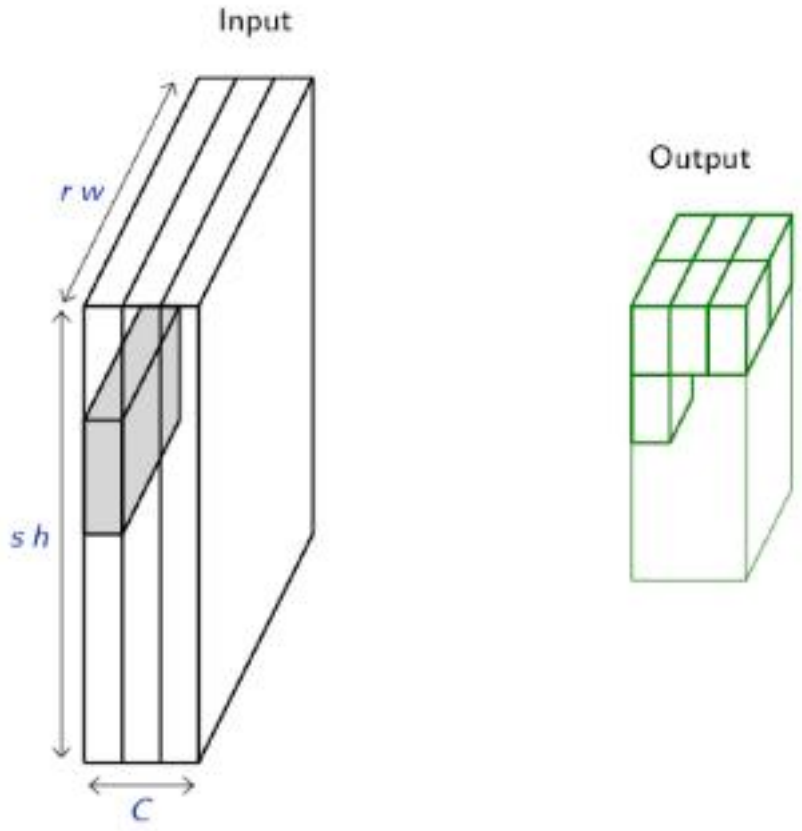
Slide credit: F. Fleuret, EE-559 Deep learning

Max-Pooling 2d



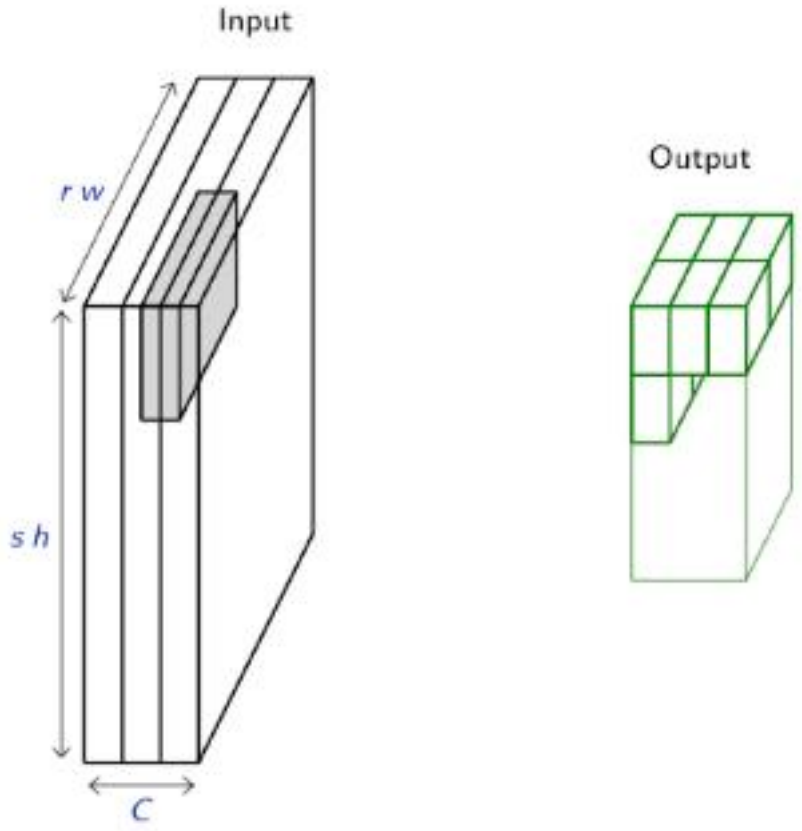
Slide credit: F. Fleuret, EE-559 Deep learning

Max-Pooling 2d



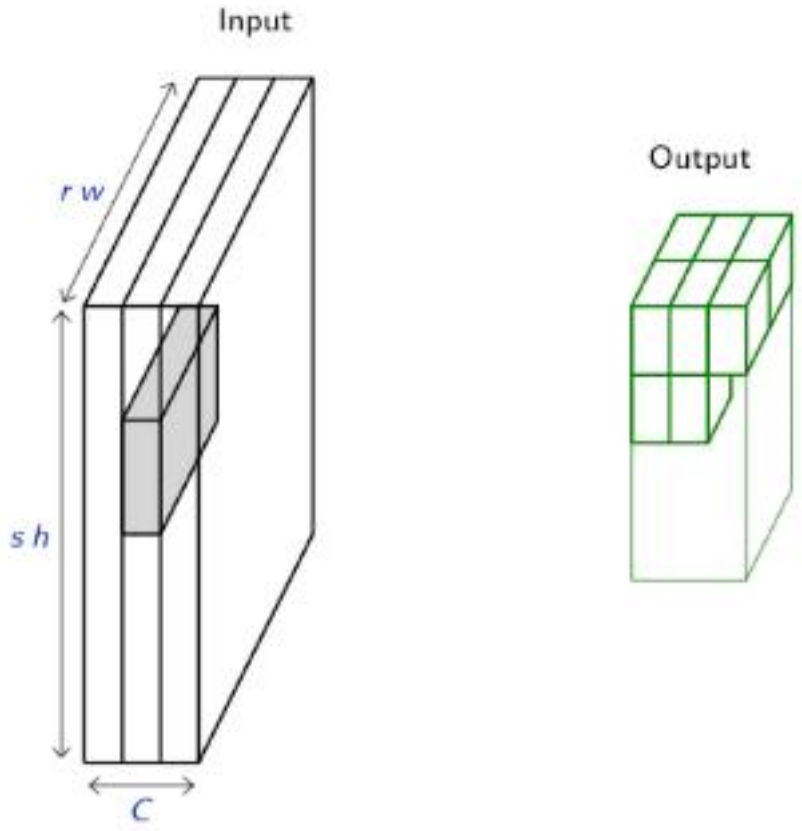
Slide credit: F. Fleuret, EE-559 Deep learning

Max-Pooling 2d



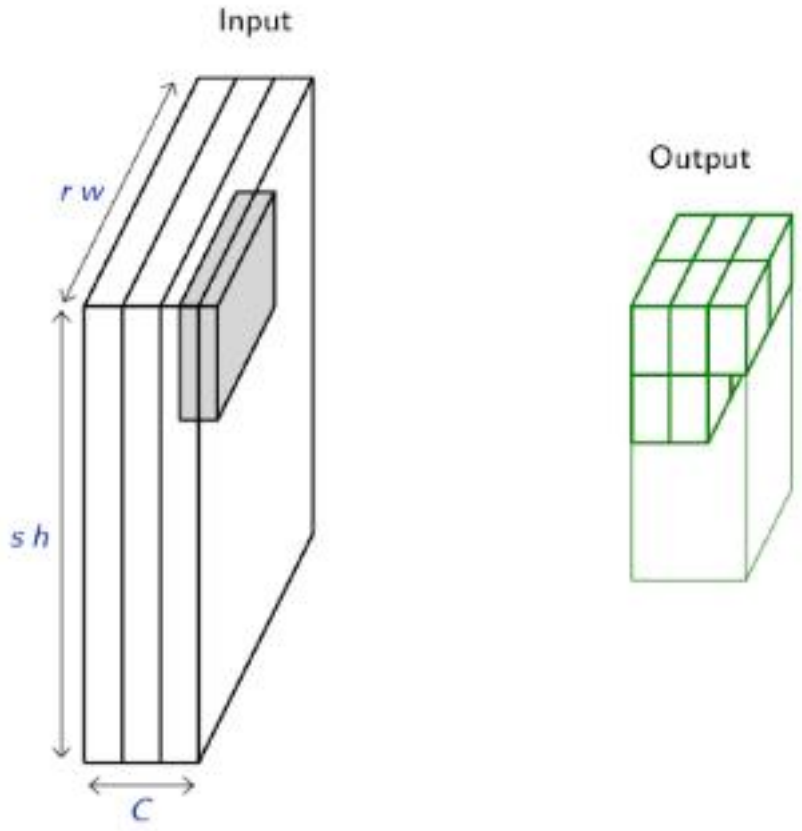
Slide credit: F. Fleuret, EE-559 Deep learning

Max-Pooling 2d



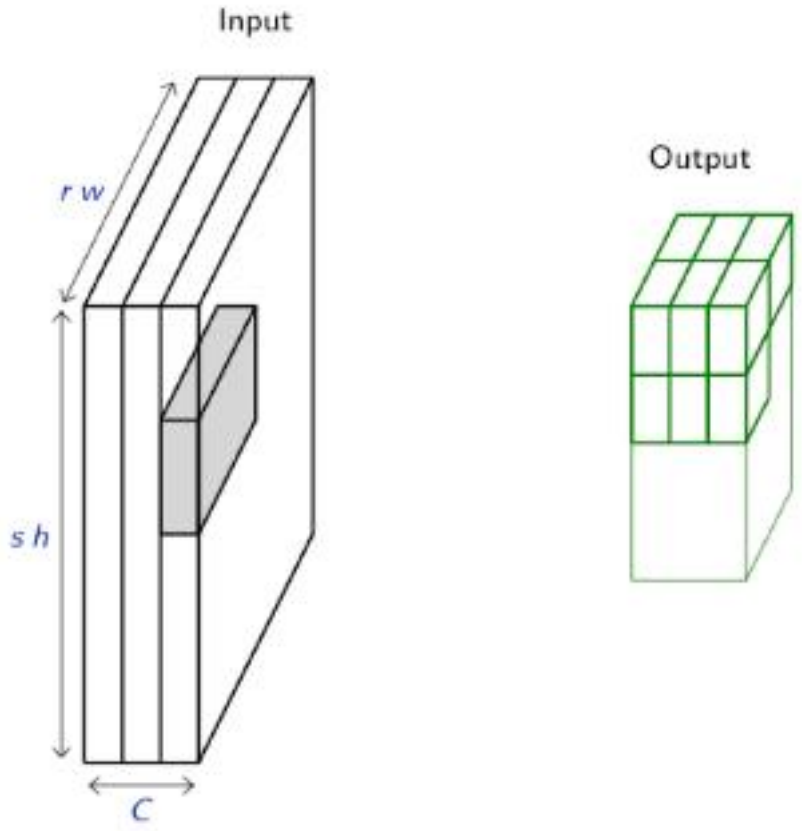
Slide credit: F. Fleuret, EE-559 Deep learning

Max-Pooling 2d



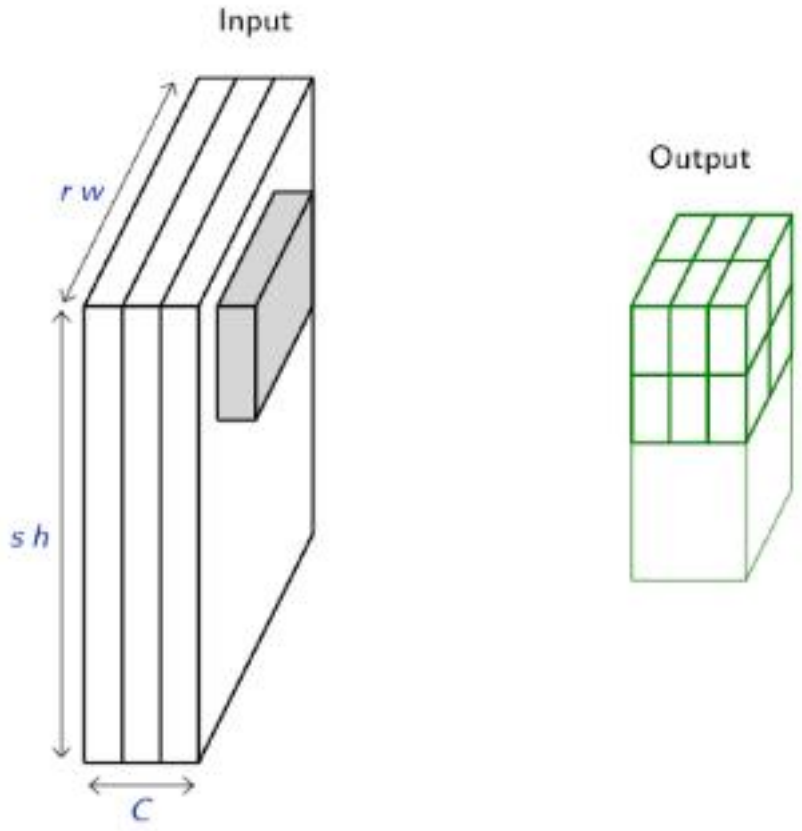
Slide credit: F. Fleuret, EE-559 Deep learning

Max-Pooling 2d



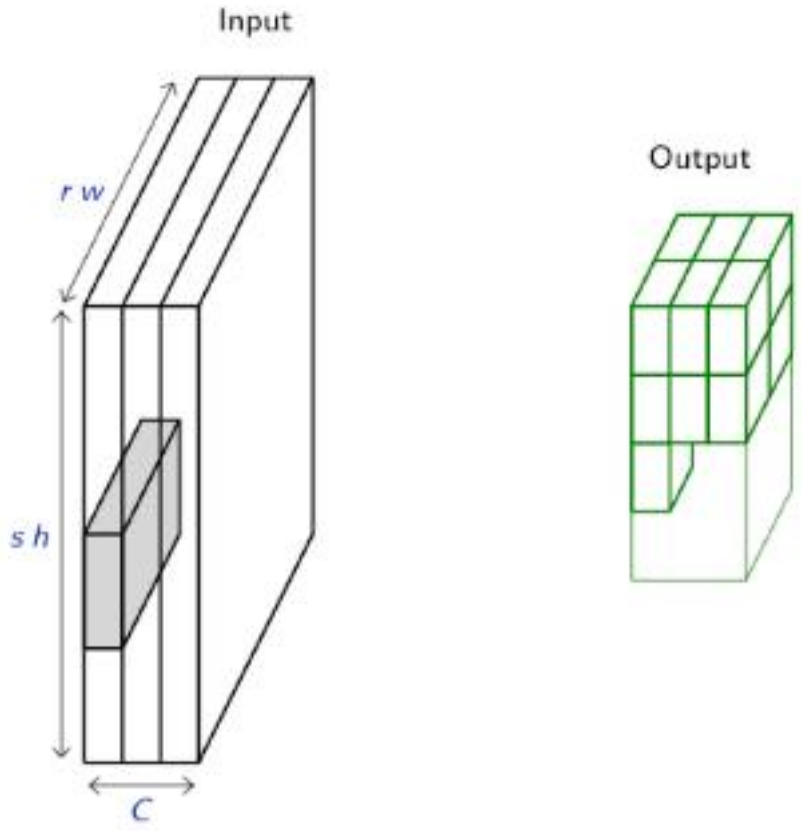
Slide credit: F. Fleuret, EE-559 Deep learning

Max-Pooling 2d



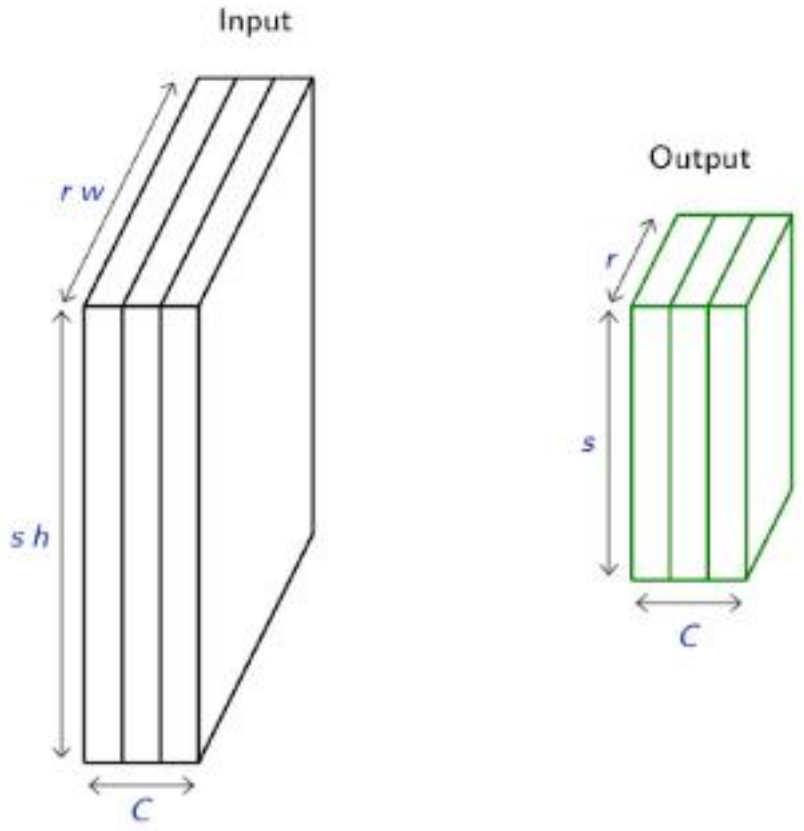
Slide credit: F. Fleuret, EE-559 Deep learning

Max-Pooling 2d



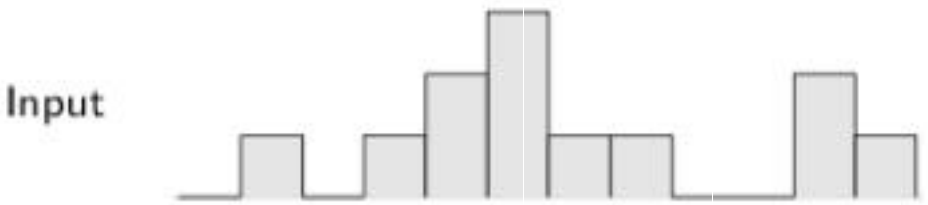
Slide credit: F. Fleuret, EE-559 Deep learning

Max-Pooling 2d



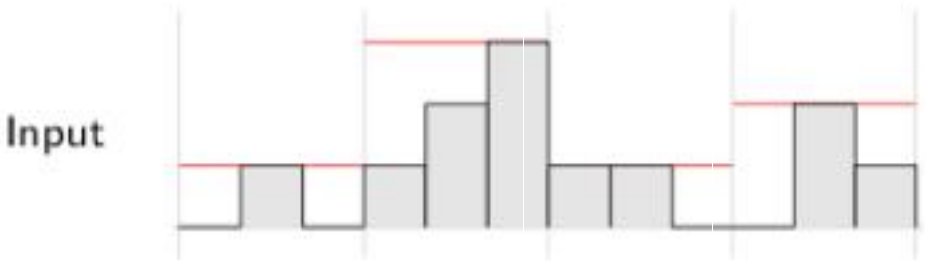
Slide credit: F. Fleuret, EE-559 Deep learning

Translation invariance from pooling



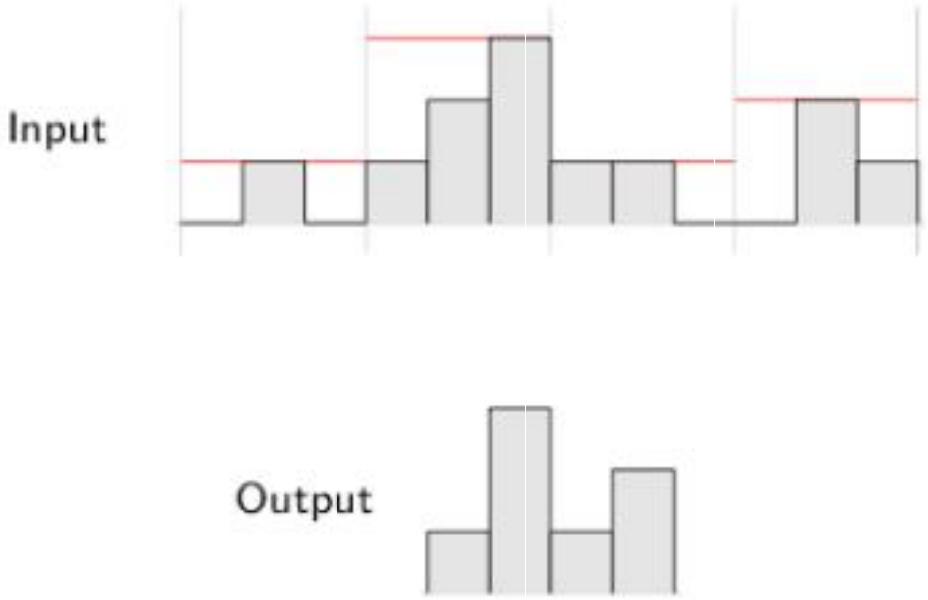
Slide credit: F. Fleuret, EE-559 Deep learning

Translation invariance from pooling



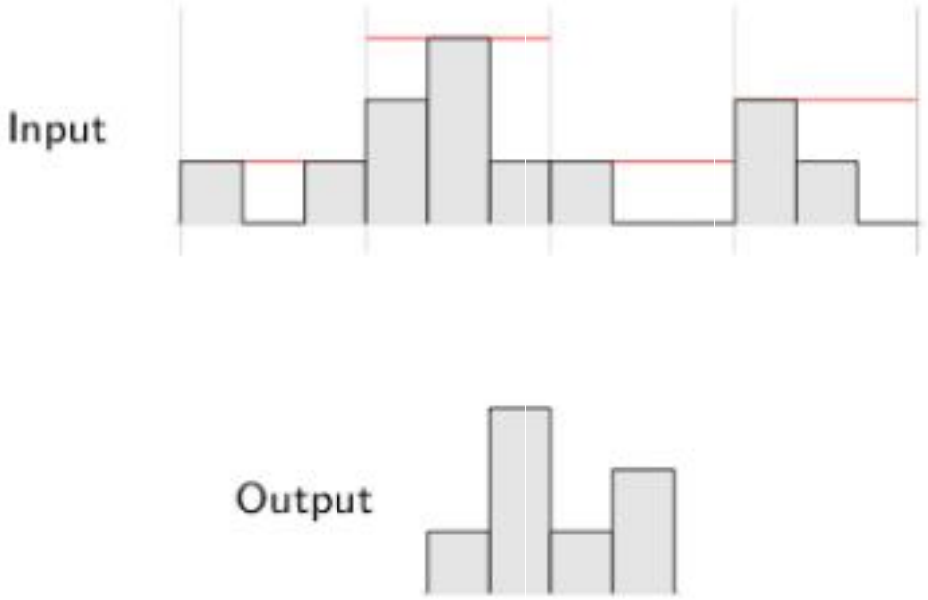
Slide credit: F. Fleuret, EE-559 Deep learning

Translation invariance from pooling



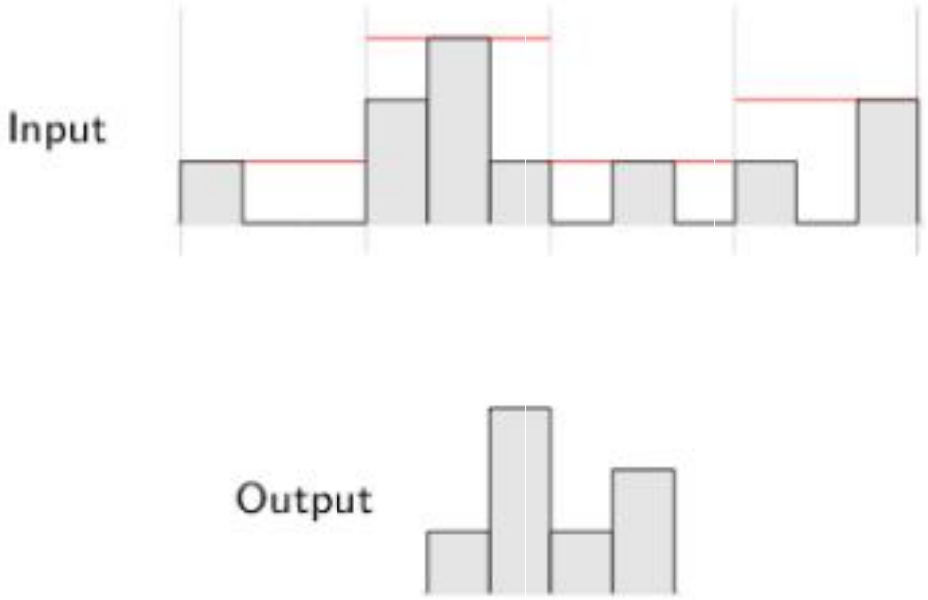
Slide credit: F. Fleuret, EE-559 Deep learning

Translation invariance from pooling



Slide credit: F. Fleuret, EE-559 Deep learning

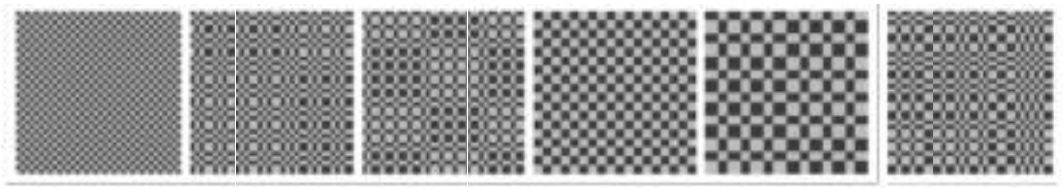
Translation invariance from pooling



Slide credit: F. Fleuret, EE-559 Deep learning

Stochastic pooling

Random pooling mask at each pass



Fractional Max-Pooling, Graham, arXiv 2014

Spectral pooling

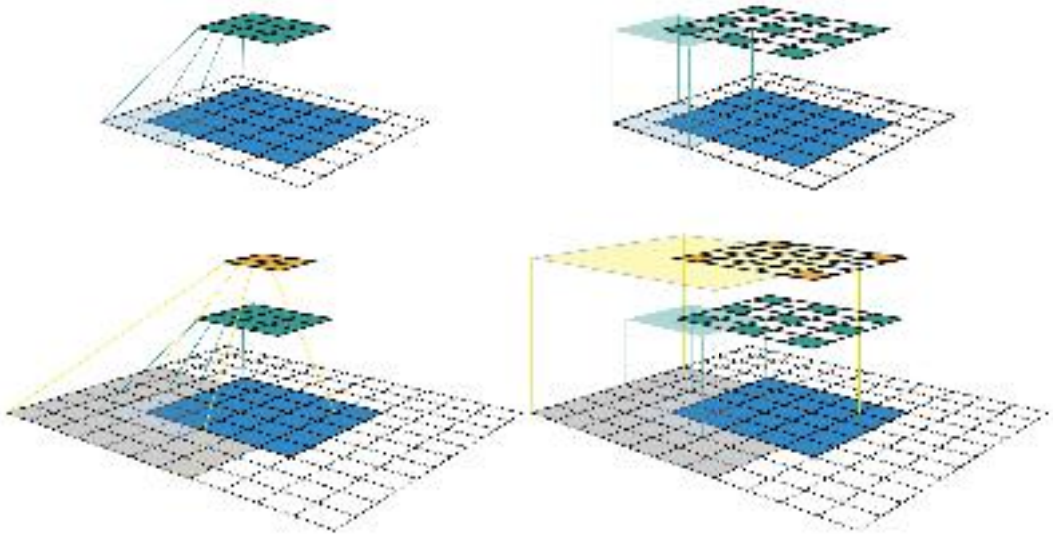
Pooling in the frequency domain

Figure 2: Approximations for different pooling schemes, for different factors of dimensionality reduction. Spectral pooling projects onto the Fourier basis and truncates it as desired. This retains significantly more information and permits the selection of any arbitrary output map dimensionality.

Spectral Representations for Convolutional Neural Networks, Rippel et al., NIPS 2015

Receptive field

- The receptive field is defined as the region in the input space that a particular CNN's feature is looking at (i.e. be affected by).
- A receptive field of a feature can be fully described by its center location and its size
- Example: $k = 3 \times 3; p = 1 \times 1; s = 2 \times 2; input = 3 \times 3$

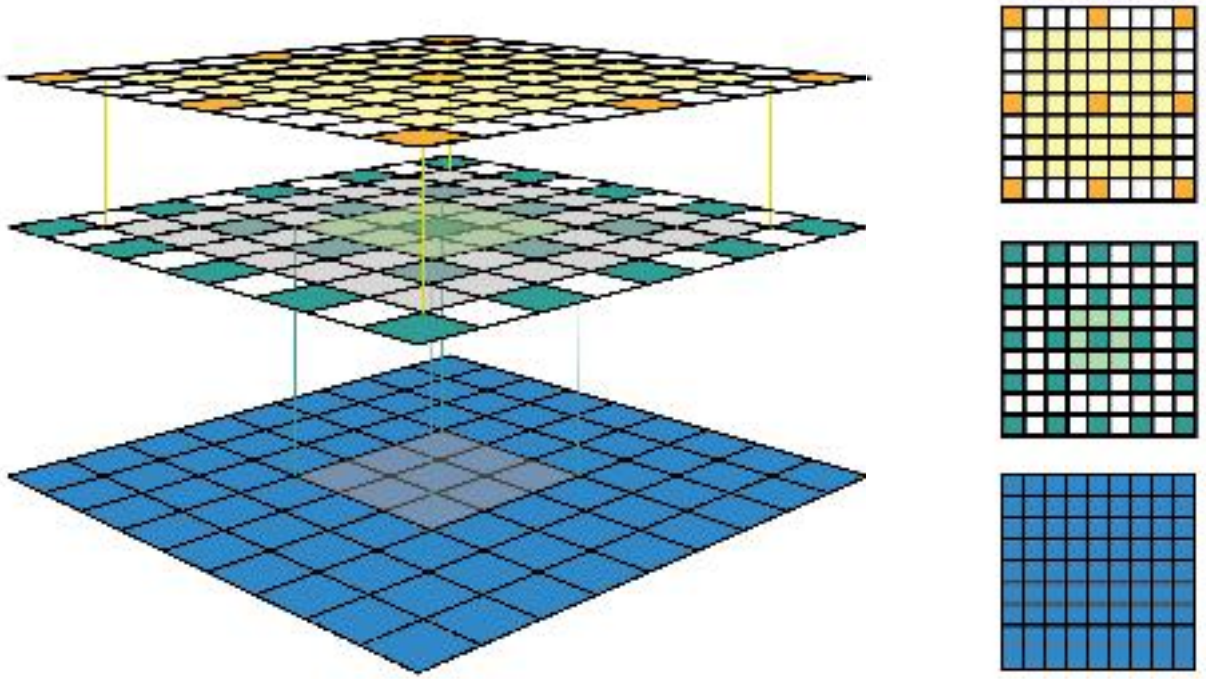


Common way to visualize a CNN feature map.

Fixed-sized CNN feature map visualization, where the size of

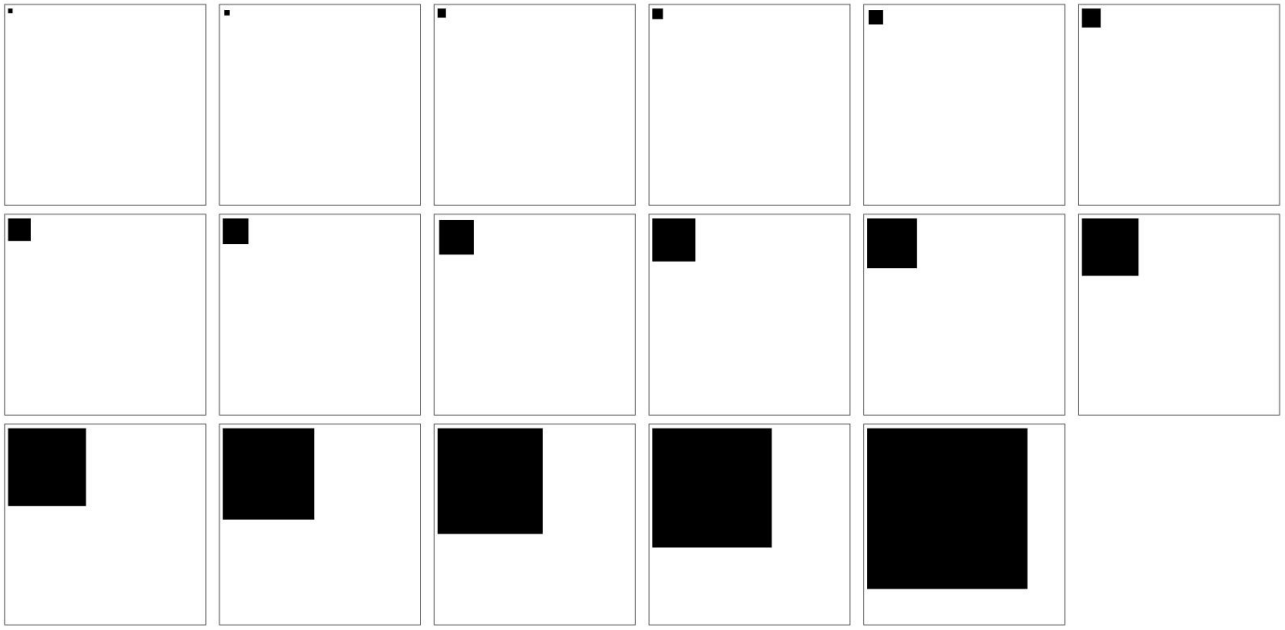
Receptive field

- The receptive field is defined as the region in the input space that a particular CNN's feature is looking at (i.e. be affected by).
- A receptive field of a feature can be fully described by its center location and its size
- Example: $k = 3 \times 3; p = 1 \times 1; s = 2 \times 2; input = 7 \times 7$



Receptive field

- The receptive field is defined as the region in the input space that a particular CNN's feature is looking at (i.e. be affected by).
- A receptive field of a feature can be fully described by its center location and its size



Receptive fields for convolutional and pooling layers of VGG-16

Dilated convolutions

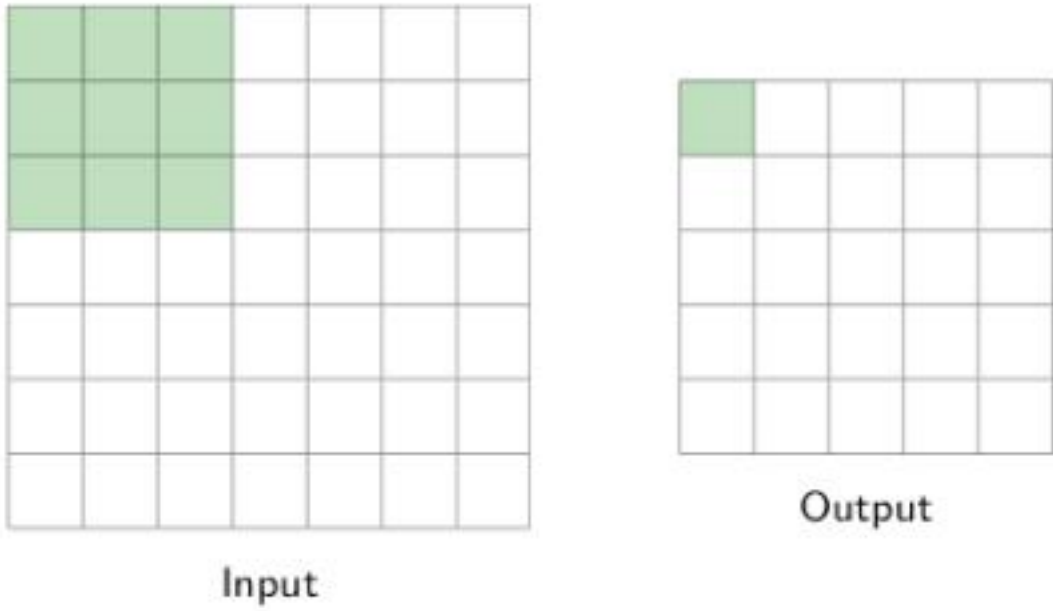


Figure credit: F. Fleuret, EE-559 Deep learning

Dilated convolutions

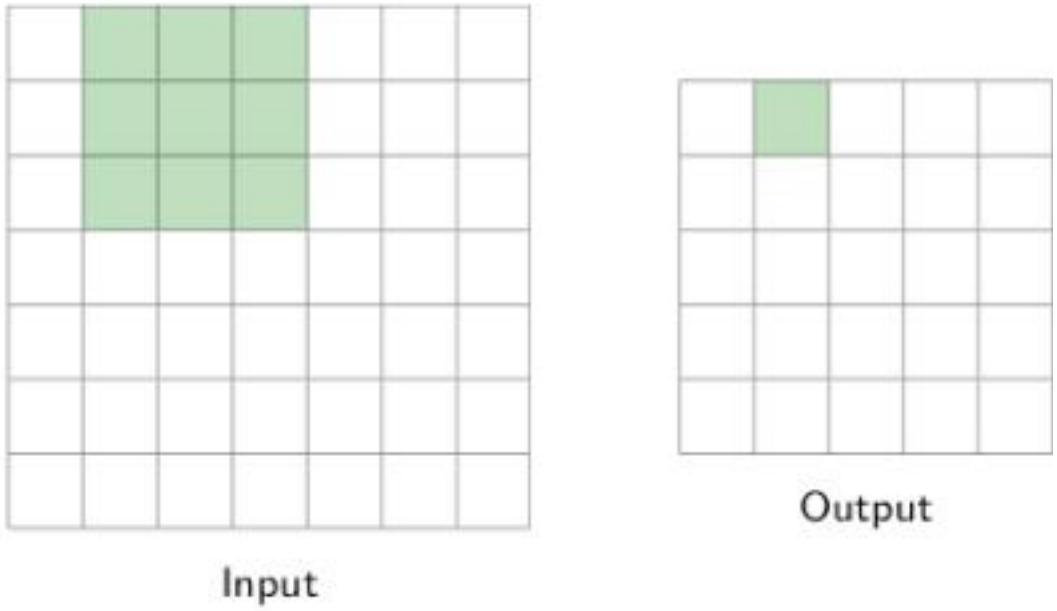


Figure credit: F. Fleuret, EE-559 Deep learning

Dilated convolutions

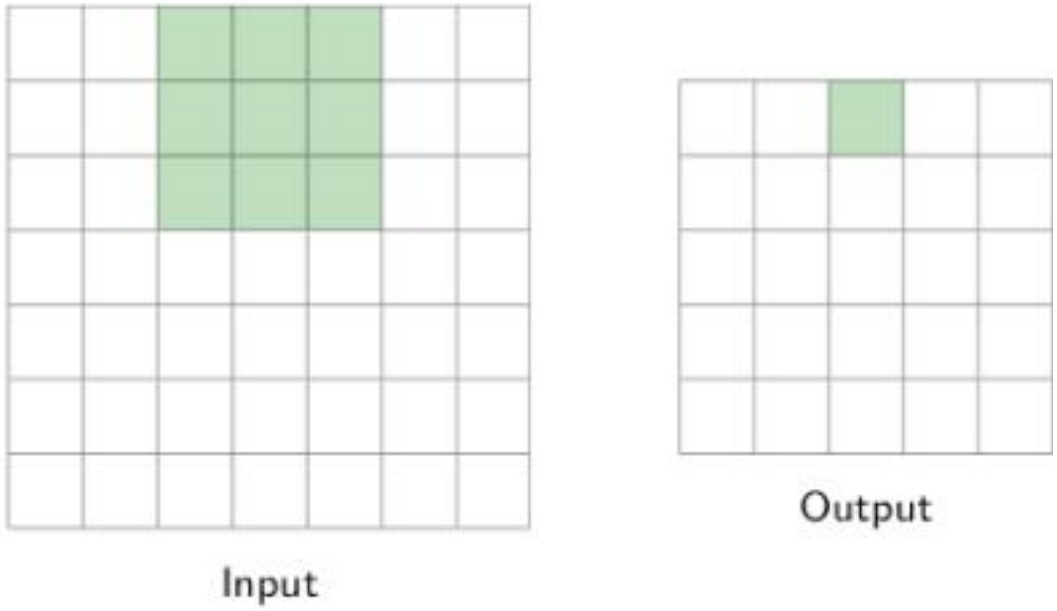


Figure credit: F. Fleuret, EE-559 Deep learning

Dilated convolutions

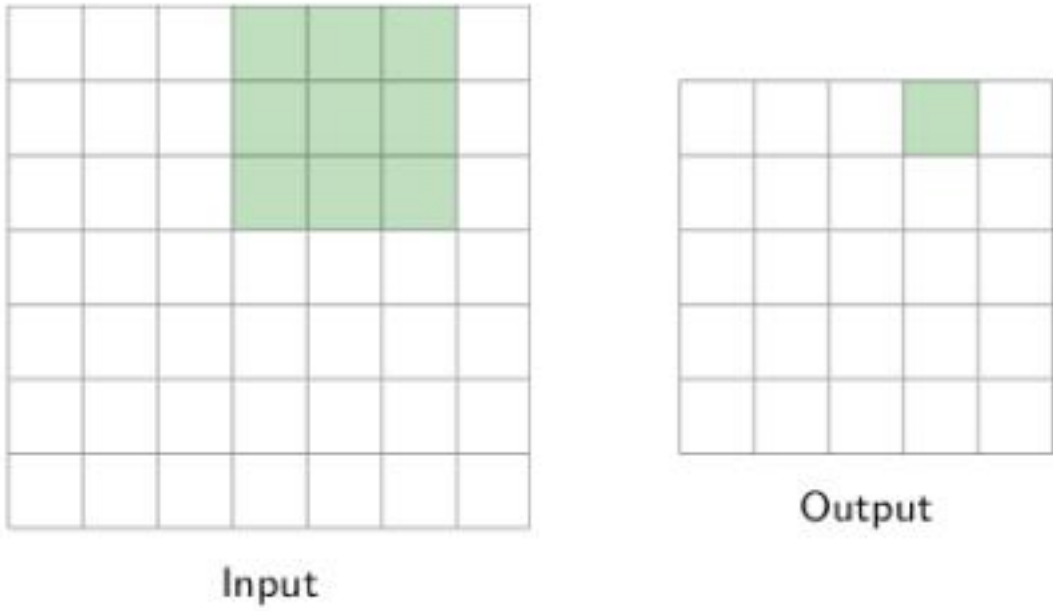


Figure credit: F. Fleuret, EE-559 Deep learning

Dilated convolutions

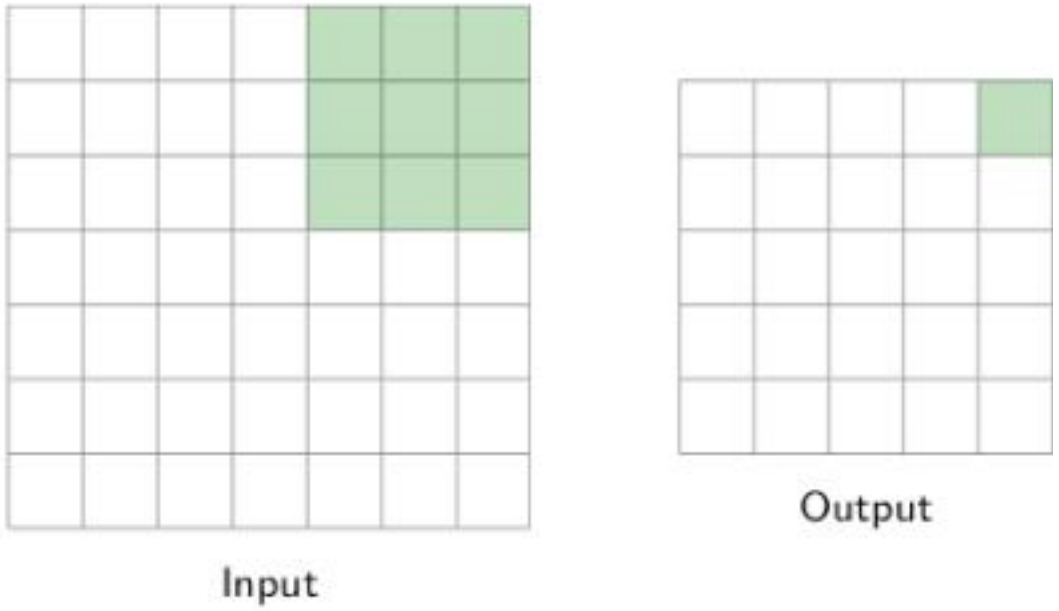


Figure credit: F. Fleuret, EE-559 Deep learning

Dilated convolutions

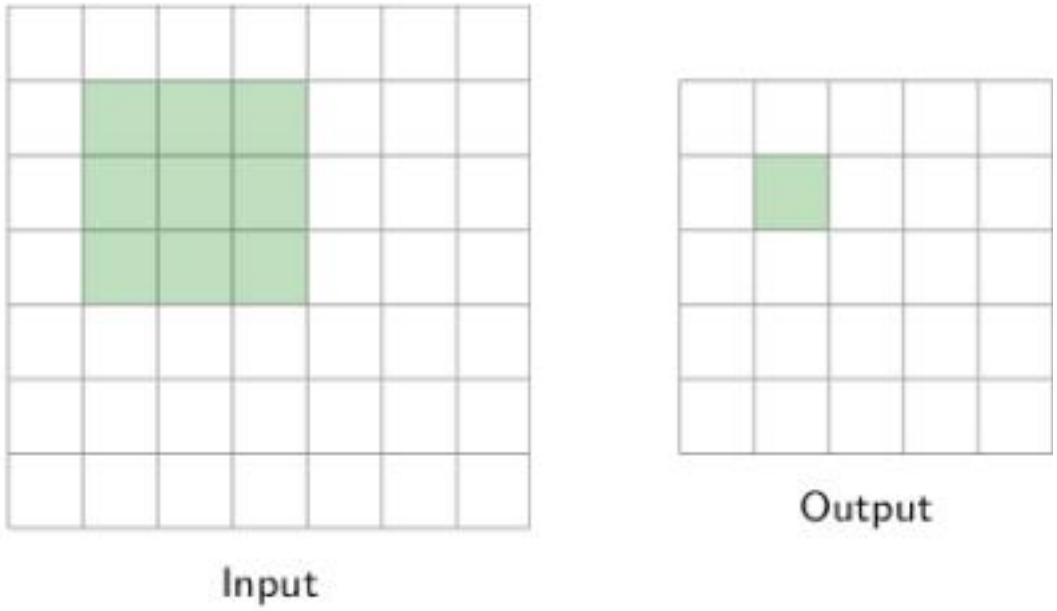


Figure credit: F. Fleuret, EE-559 Deep learning

Dilated convolutions

- Can we do better?
- ... Without adding parameters?

Dilated convolutions

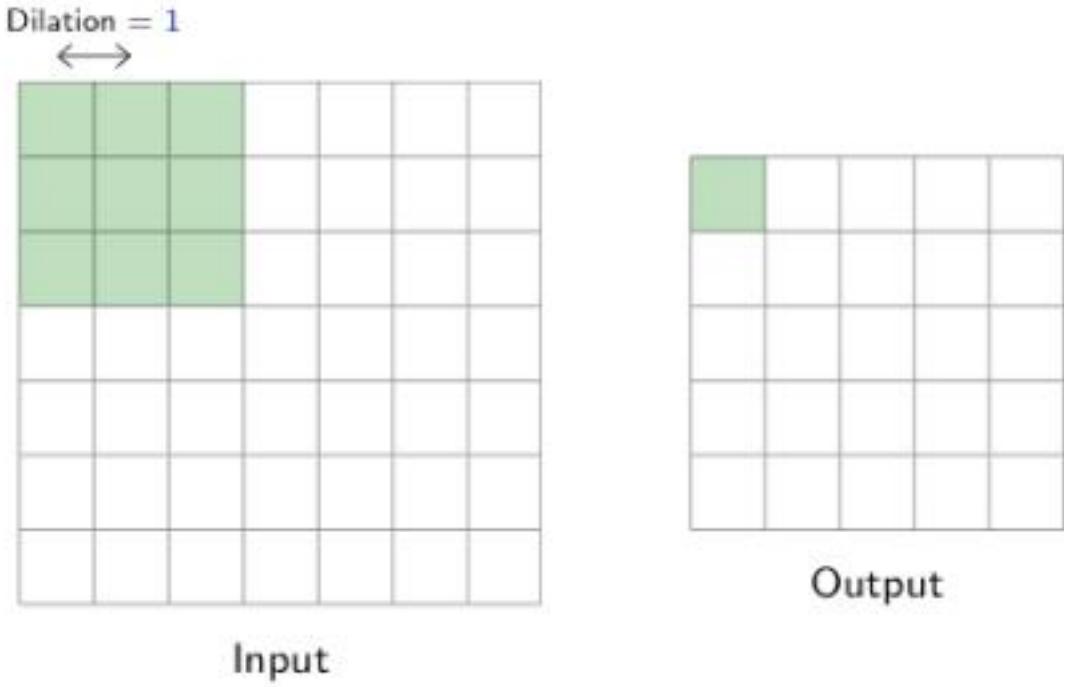


Figure credit: F. Fleuret, EE-559 Deep learning

Dilated convolutions

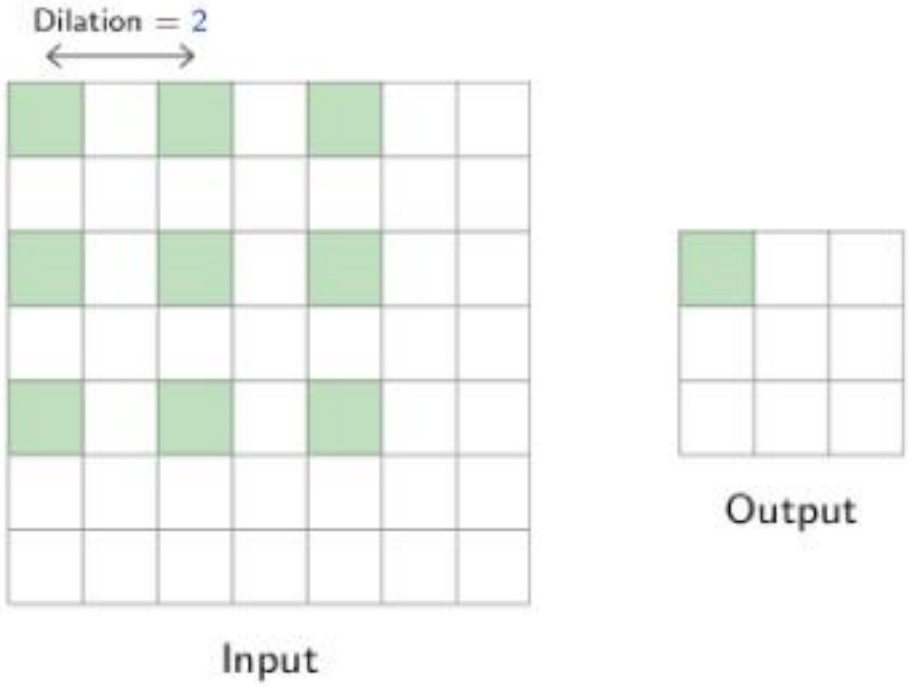


Figure credit: F. Fleuret, EE-559 Deep learning

Dilated convolutions

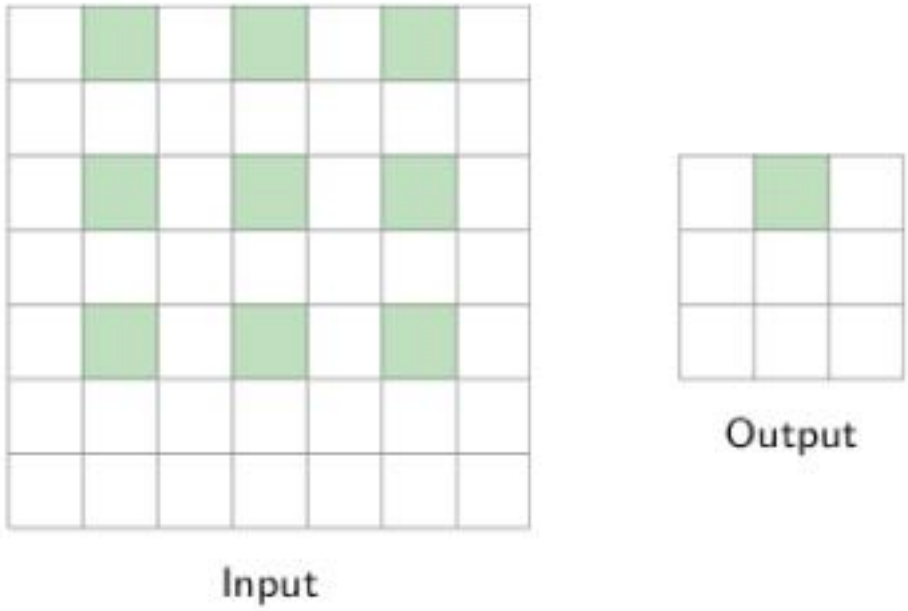


Figure credit: F. Fleuret, EE-559 Deep learning

Dilated convolutions

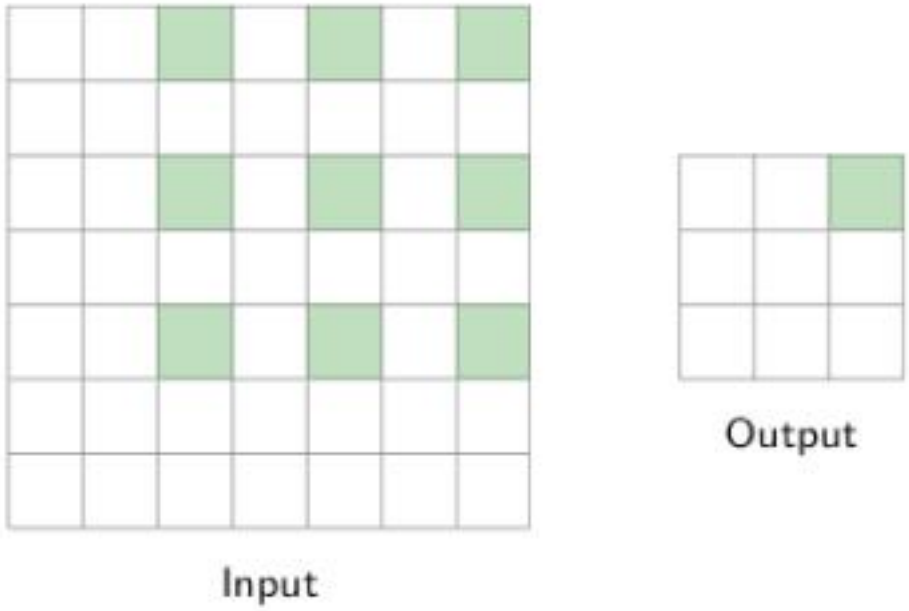


Figure credit: F. Fleuret, EE-559 Deep learning

Dilated convolutions

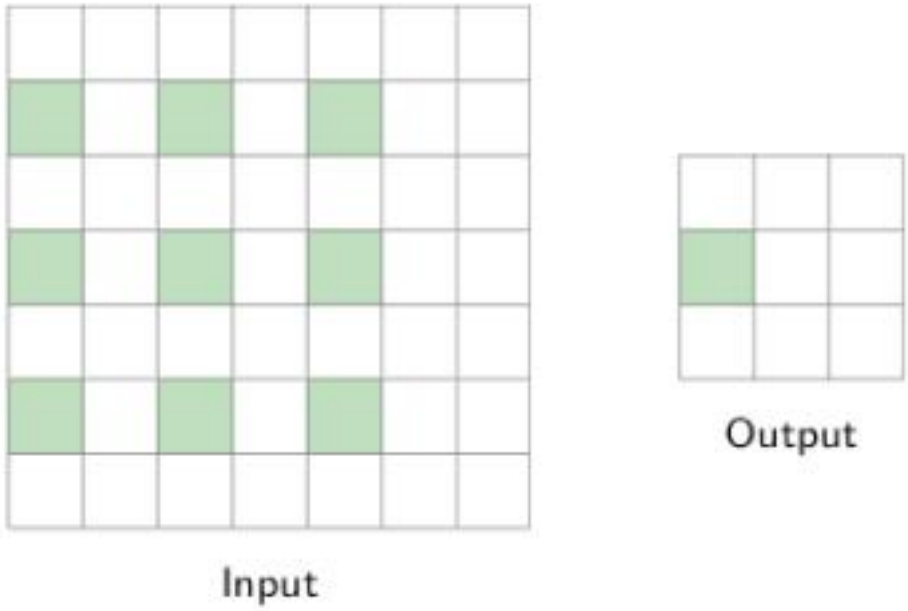


Figure credit: F. Fleuret, EE-559 Deep learning

Dilated convolutions

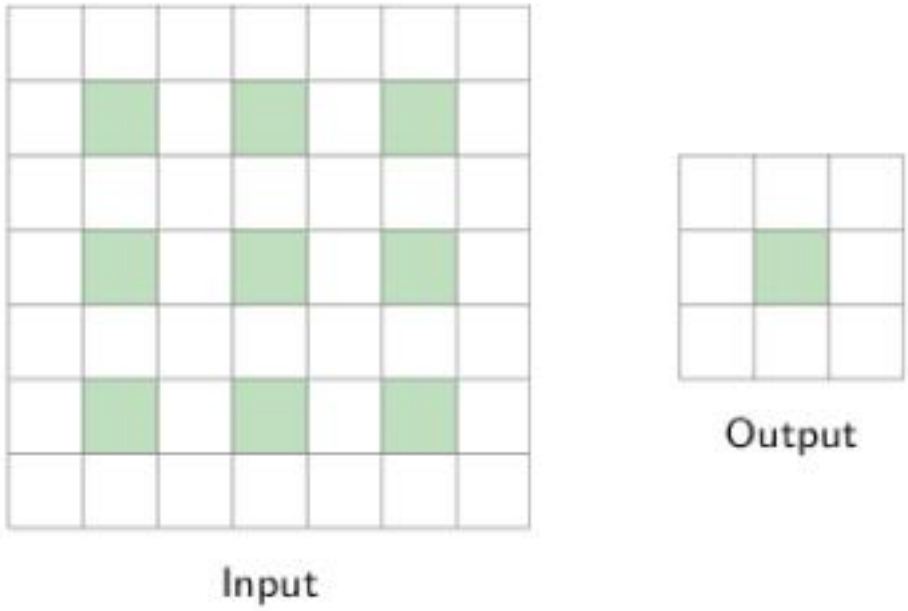


Figure credit: F. Fleuret, EE-559 Deep learning

Dilated convolutions

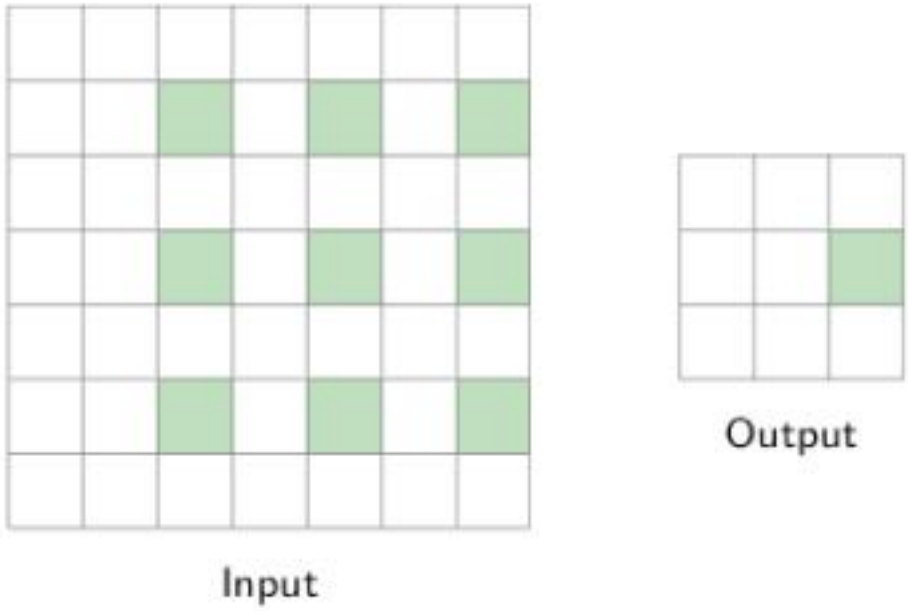


Figure credit: F. Fleuret, EE-559 Deep learning

Dilated convolutions

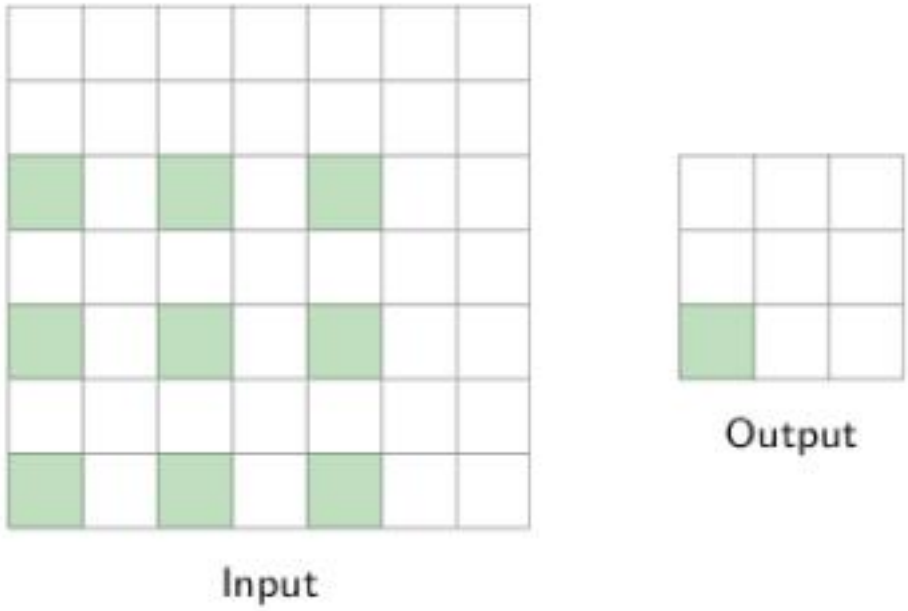


Figure credit: F. Fleuret, EE-559 Deep learning

Dilated convolutions

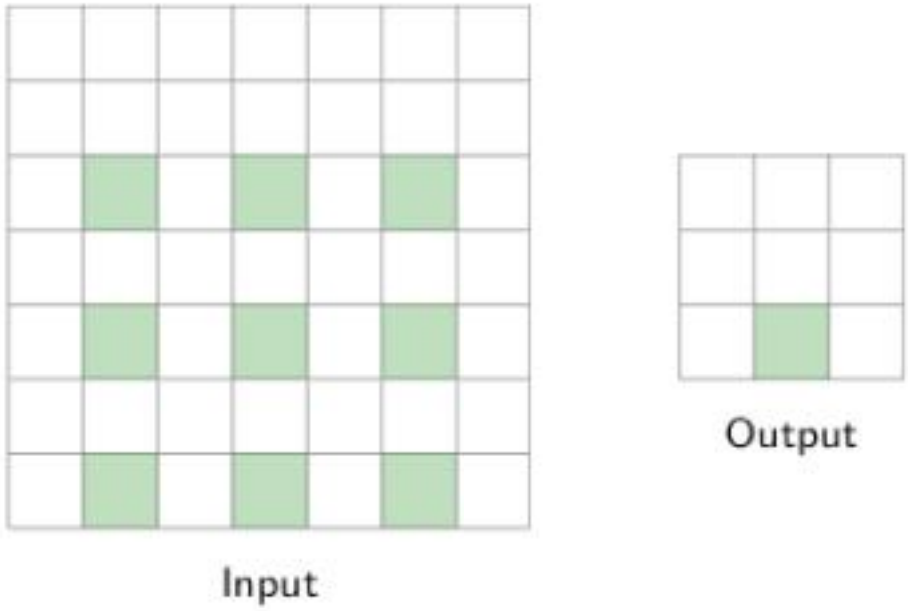


Figure credit: F. Fleuret, EE-559 Deep learning

Dilated convolutions

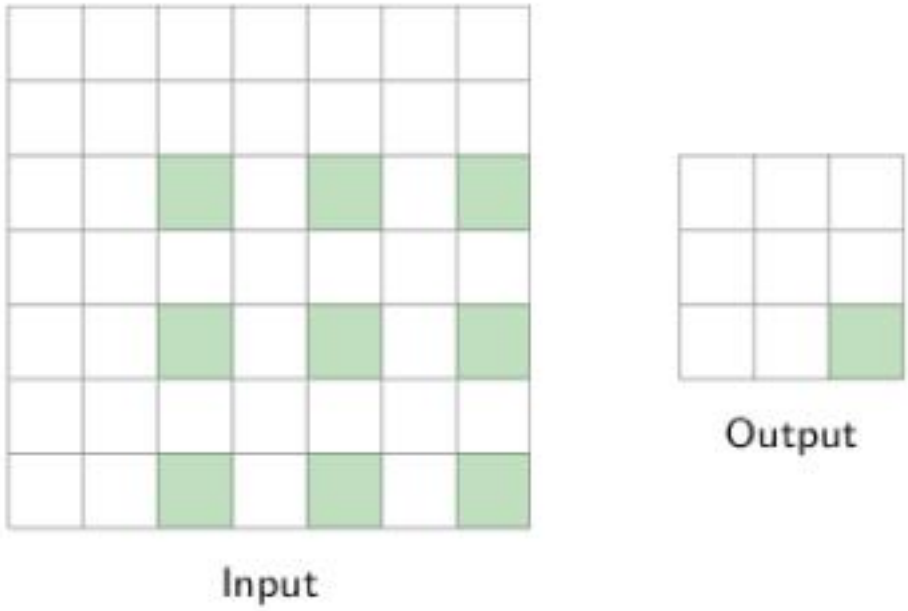
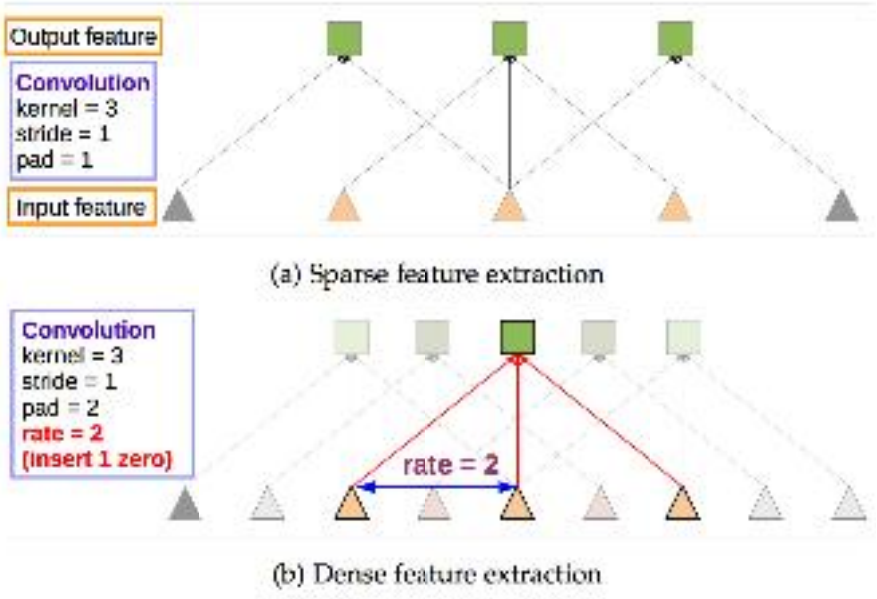
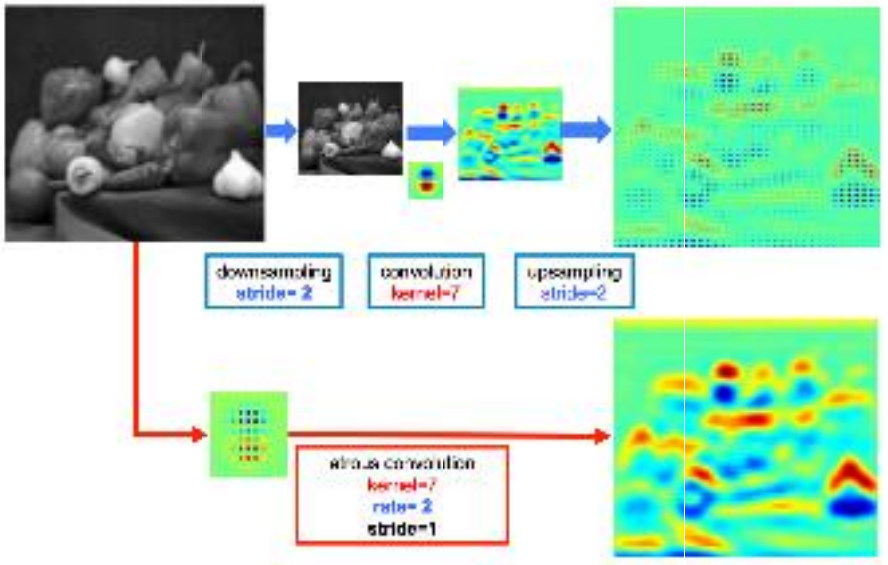


Figure credit: F. Fleuret, EE-559 Deep learning

Dilated convolutions

- also goes by the name convolutions à trous





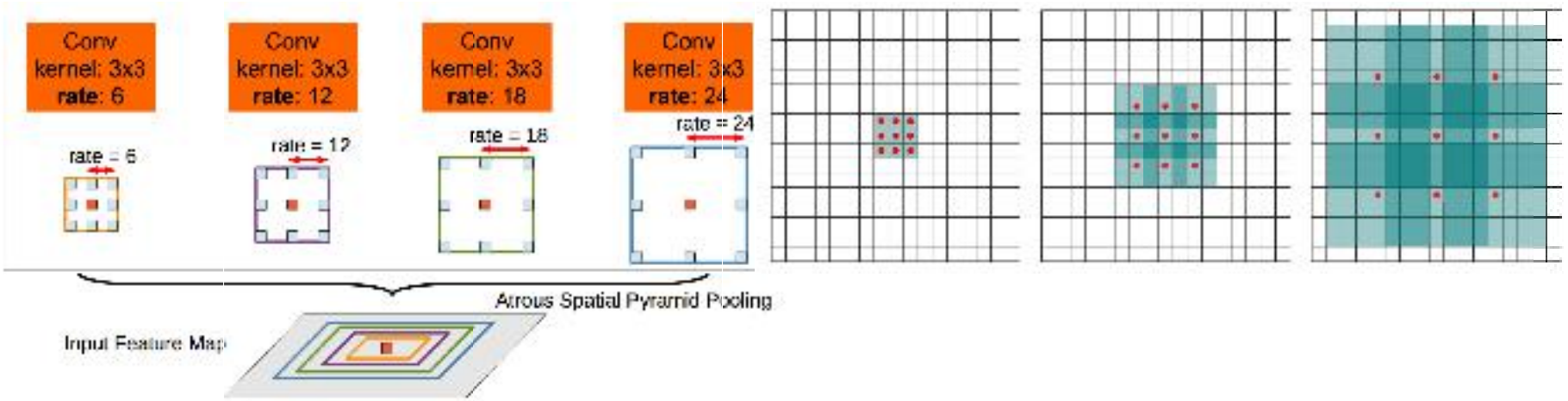
DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs; Chen et al., PAMI 2016

Dilated convolutions

Usage

In parallel

Stacked



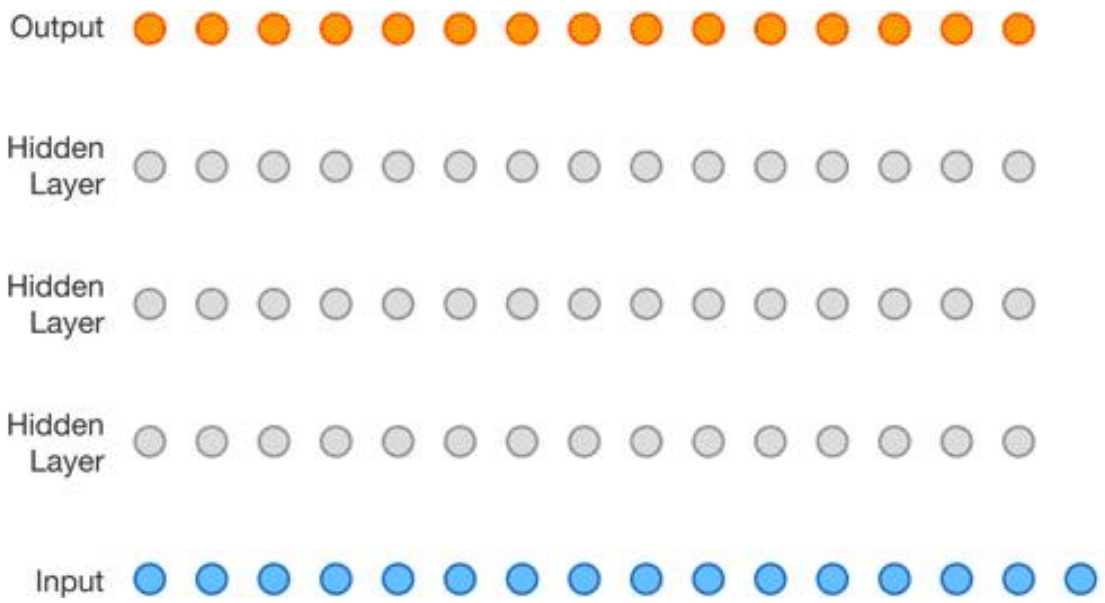
More frequently used

DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs; Chen et al., PAMI 2016

Multi-scale context aggregation by dilated convolutions; Yu and Koltun, ICLR 2016

Dilated convolutions

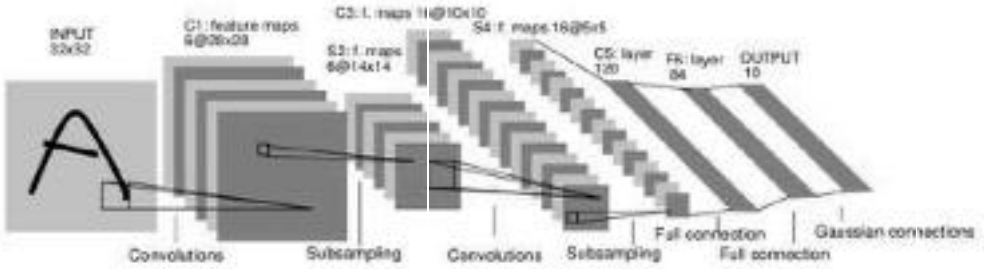
- works for 1d as well
- appealing alternative to recurrent neural networks



WaveNet: A Generative Model for Raw Audio, A. van den Oord et al., 2016

ConvNet

- Neural network with specialized connectivity structure
- Stack multiple stage of feature extractors
- Higher stages compute more global, more invariant features
- Classification layer at the end



LeNet-5, LeCun, 1998

ConvNet

Remember this?

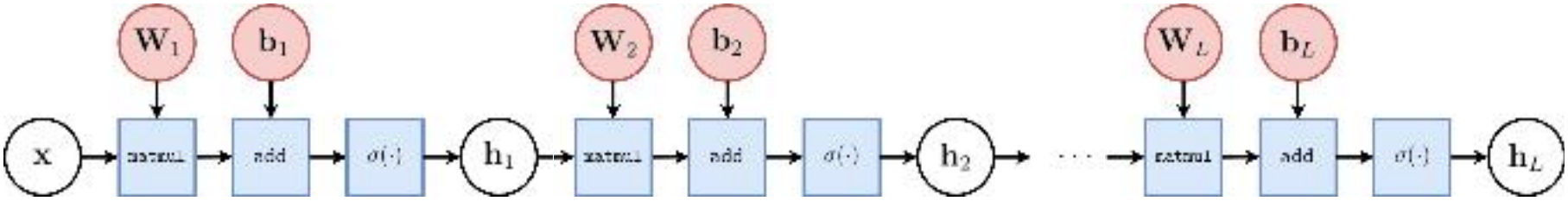


Figure credit: G. Louppe, INFO8004-1 Advanced Machine Learning

ConvNet

Just like multi-layer perceptorns, convolutional layers can be also composed in series, such that:

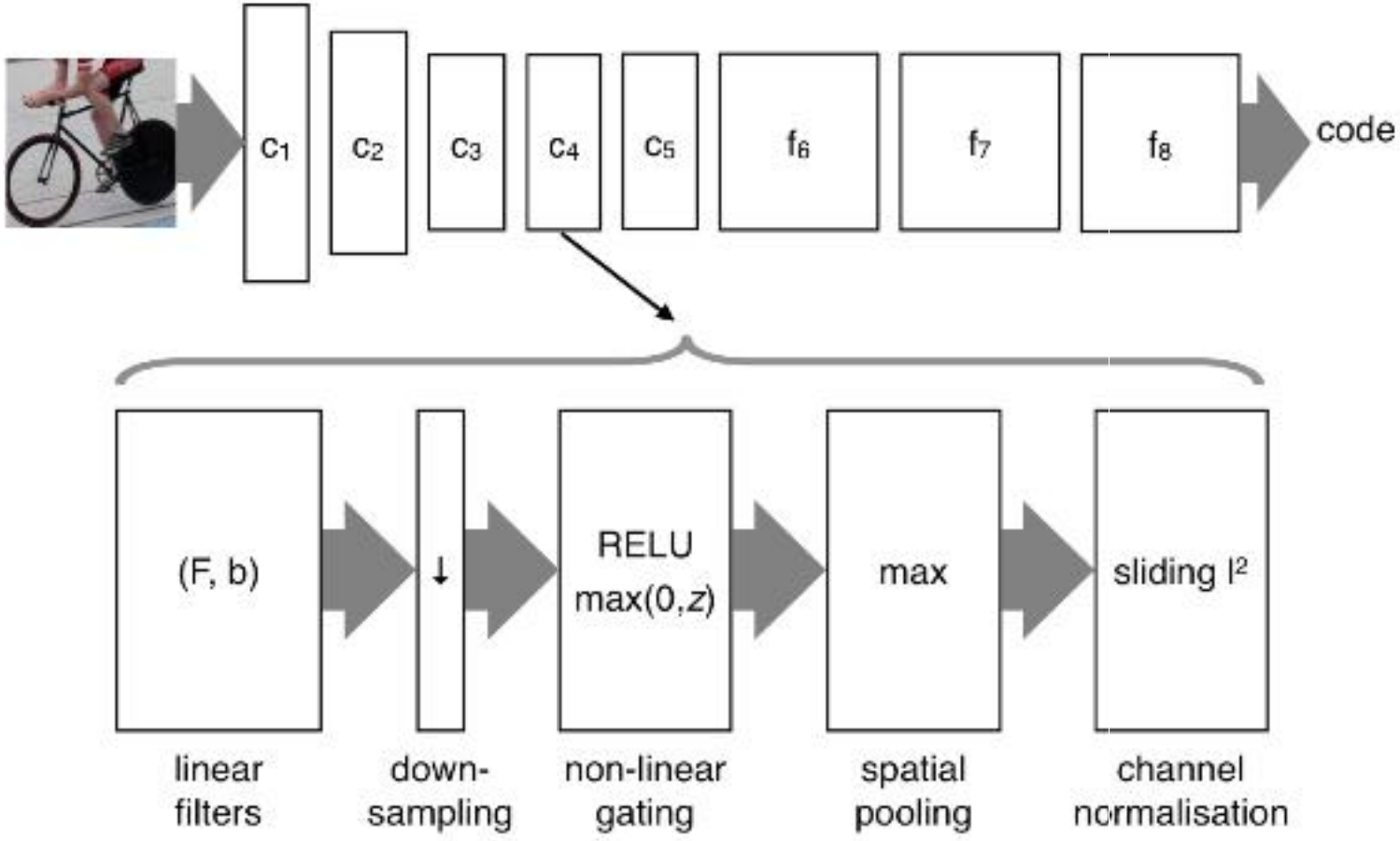
$$
\begin{aligned}
\mathbf{h}_0 &= \mathbf{x} \\
\mathbf{h}_1 &= \sigma(\mathbf{W}_1^T \mathbf{h}_0 + \mathbf{b}_1) \\
&\dots \\
\mathbf{h}_L &= \sigma(\mathbf{W}_L^T \mathbf{h}_{L-1} + \mathbf{b}_L) \\
f(\mathbf{x}; \theta) &= \mathbf{h}_L
\end{aligned}
$$

where θ denotes the model parameters $\{\mathbf{W}_k, \mathbf{b}_k, \dots | k = 1, \dots, L\}$.

- Hidden states $\mathbf{h}_k$ have $2D$ layout and are called feature maps (for MLPs they are $1D$)
- Each filter has its own trainable parameters $\mathbf{W}_k$.
- Weights from all filters and layers are trained jointly with backpropagation and gradient descent

ConvNet

A convolutional layer is composed of convolution, activation and downsampling layers.



ConvNet

Input

ConvNet

Input

Conv blocks

- Convolution + activation (relu)
- Convolution + activation (relu)
- ...
- Maxpooling 2x2

ConvNet

Input

Conv blocks

- Convolution + activation (relu)
- Convolution + activation (relu)
- ...
- Maxpooling 2x2

Output

- Fully connected layers
- Softmax

Motivations

Local connectivity

- A neuron depends only on a few local neurons
- Translation invariance

Motivations

Local connectivity

- A neuron depends only on a few local neurons
- Translation invariance

Comparison to Fully connected

- Parameter sharing
- Make use of spatial structure

Motivations

Local connectivity

- A neuron depends only on a few local neurons
- Translation invariance

Comparison to Fully connected

- Parameter sharing
- Make use of spatial structure

Some analogy to animal vision

Hubel & Wiesel, RECEPTIVE FIELDS OF SINGLE NEURONES IN THE CAT'S STRIATE CORTEX (1959)