Deep Learning & Engineering Applications

3. Introduction to ConvNets

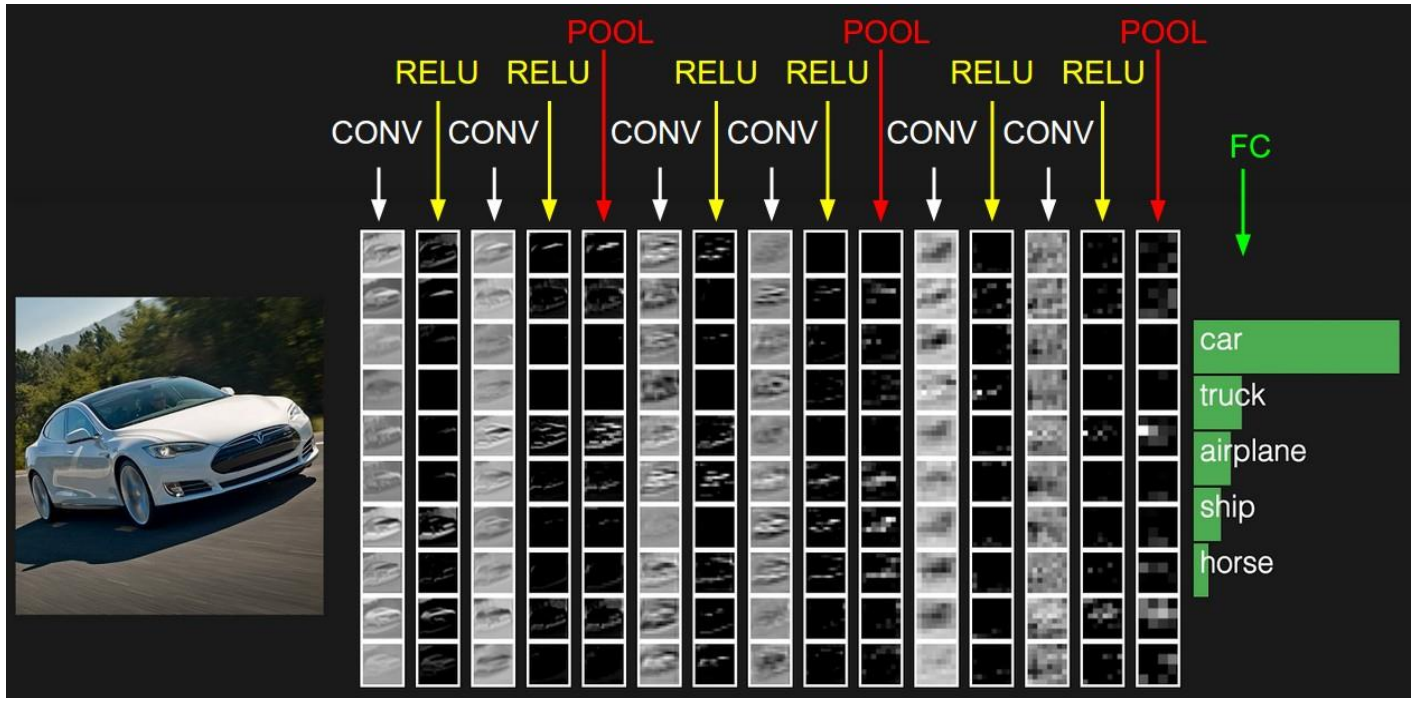
JIDONG J. YANG

COLLEGE OF ENGINEERING

UNIVERSITY OF GEORGIA

Basic Structure of CNN

CONV → RELU → POOL

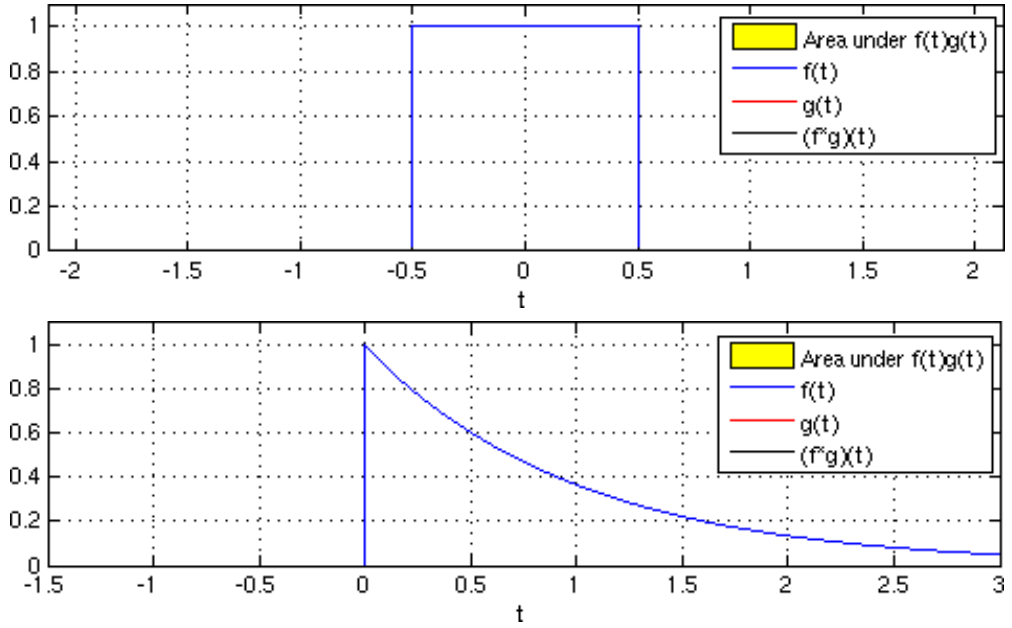


http://cs231n.stanford.edu

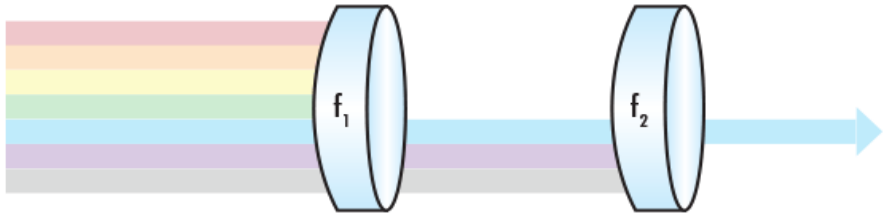
What is convolution?

A mathematical operation on two functions (f and g) that produces a third function expressing how the shape of one is modified by the other.

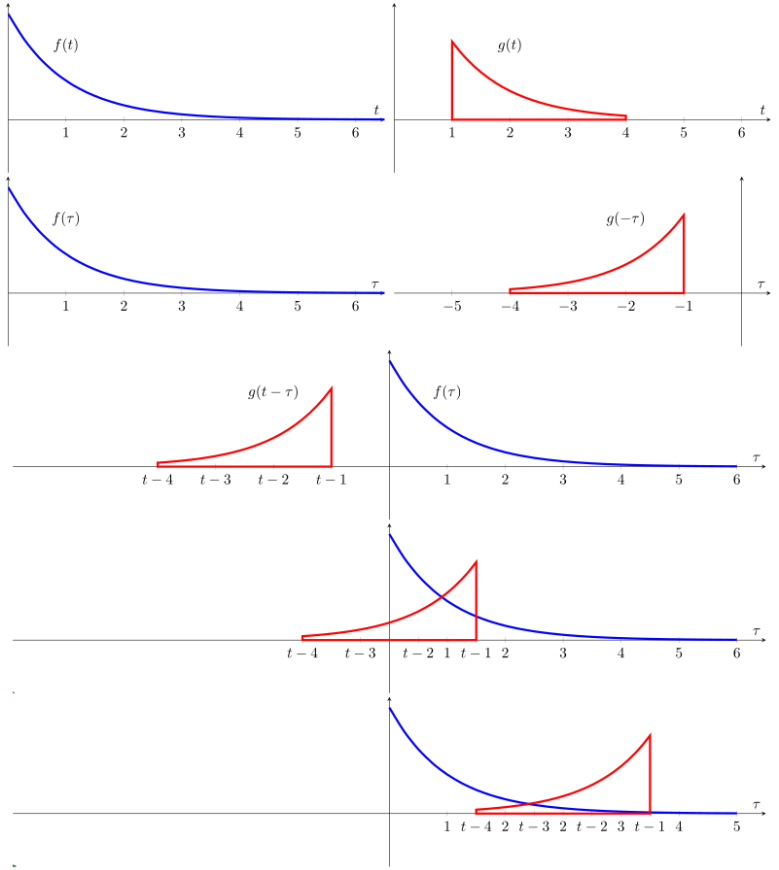
$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau$$



Brian Amberg derivative work: Tinos (talk), licensed by CC BY-SA 3.0
https://en.wikipedia.org/wiki/Convolution#/media/File:Convolution_of_box_signal_with_itself2.gif
https://en.wikipedia.org/wiki/Convolution#/media/File:Convolution_of_spiky_function_with_box2.gif

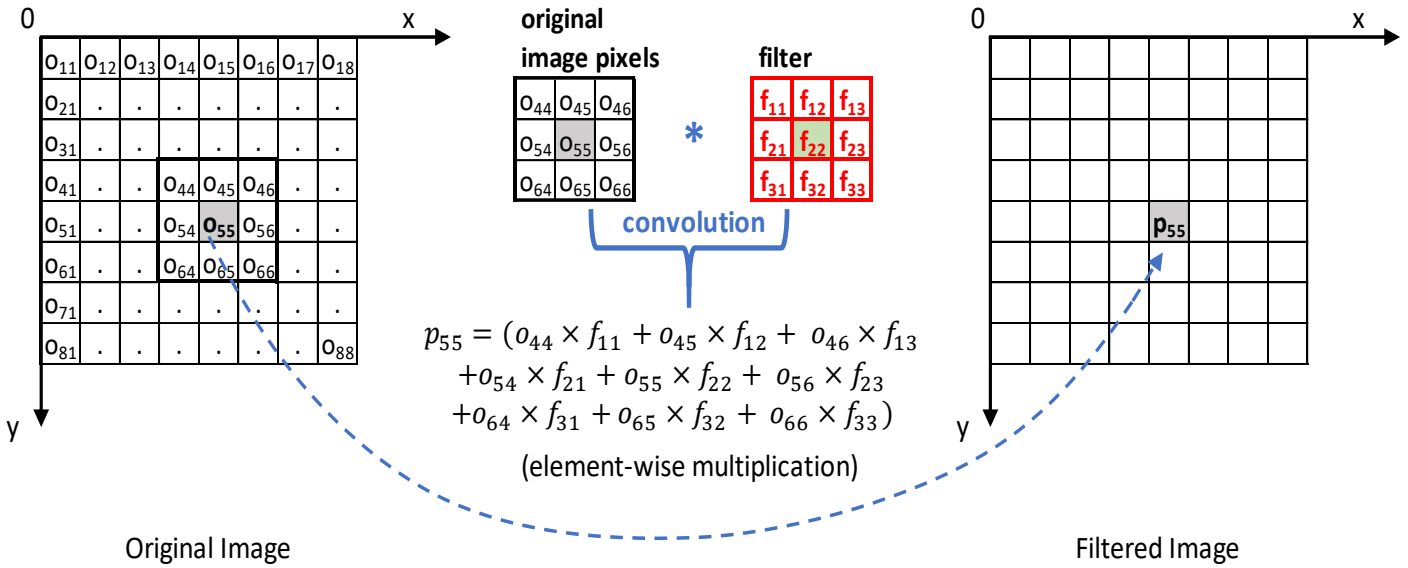


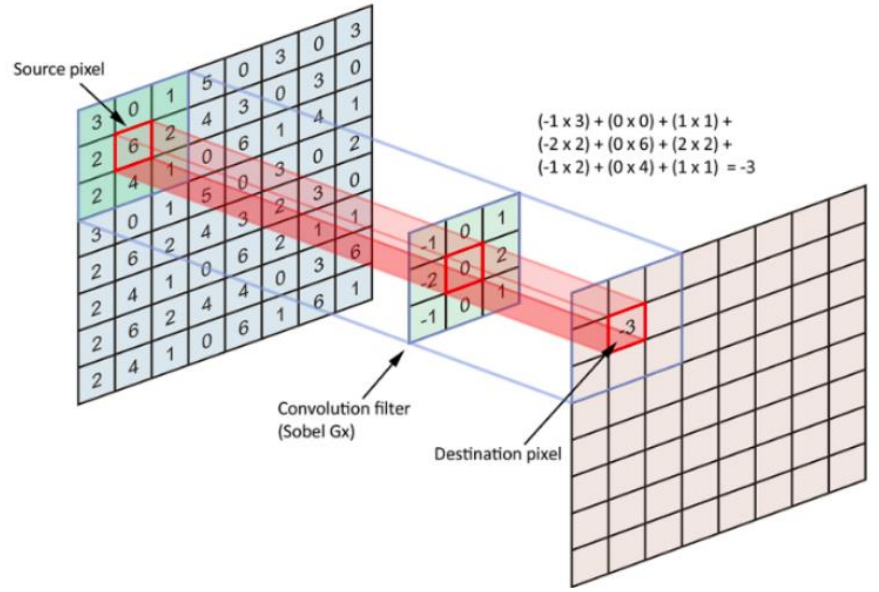
ttps://www.edmundoptics.com/knowledge-center/application-notes/optics/custom-
andpass-filter-using-shortpass-and-longpass-filters



https://en.wikipedia.org/wiki/Convolution

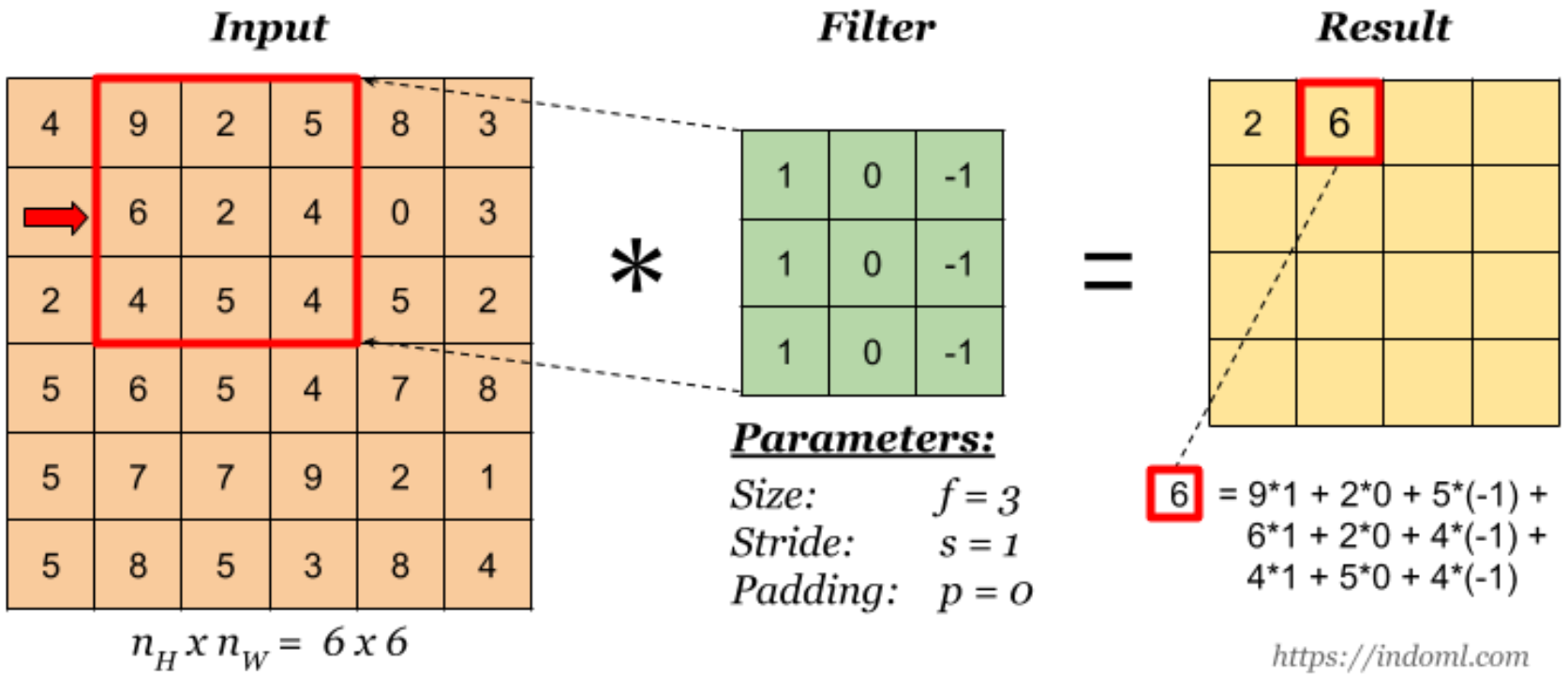
Discrete Inputs (e.g., pixels of images)



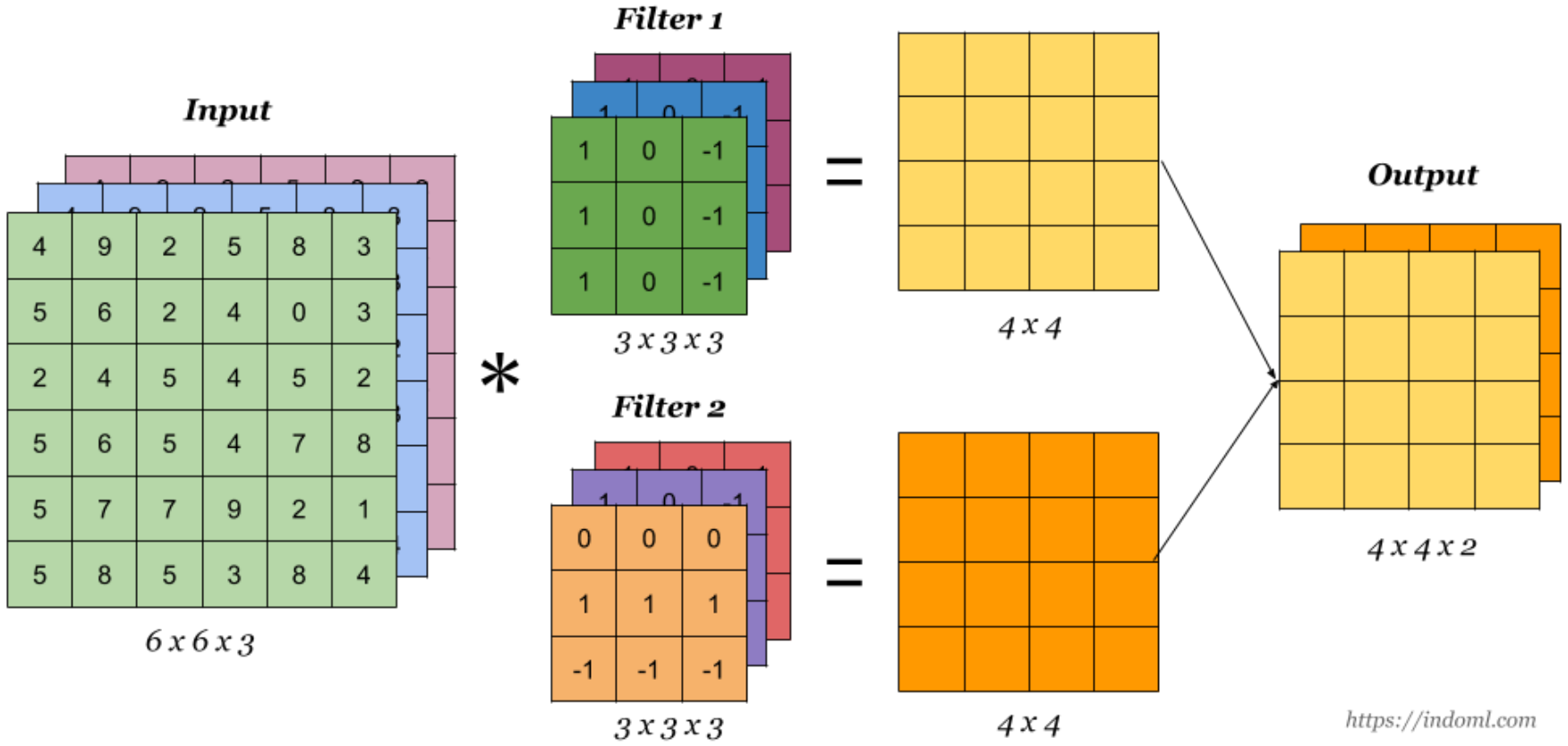


The filter slides over the input and performs its output on the new layer.—Source: https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

Example: 2D Convolution



Example: 3D Convolution



A nice animation on convolution operation: http://cs231n.github.io/convolutional-networks/

Terminology

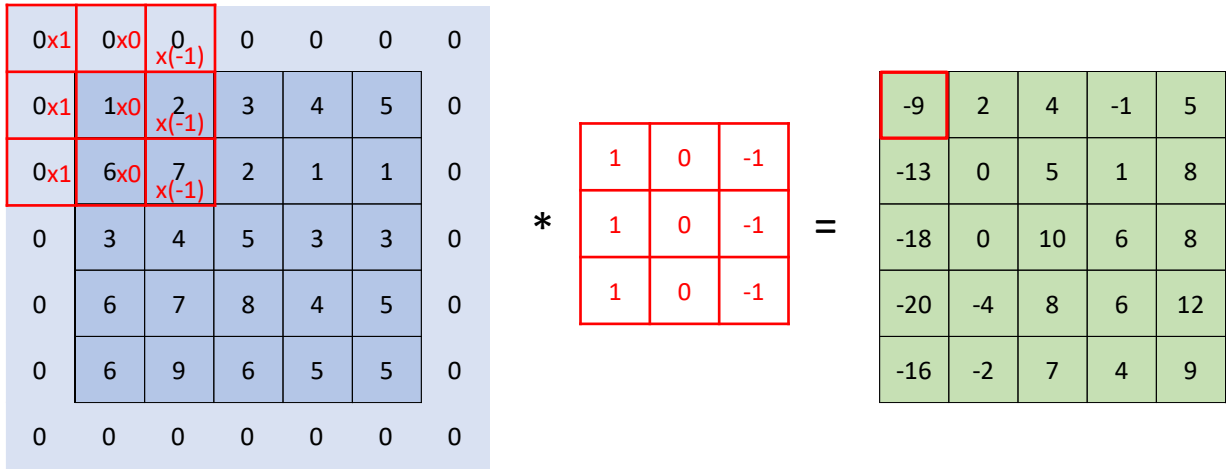
Padding

- Typically add extra zero-value pixels to margins

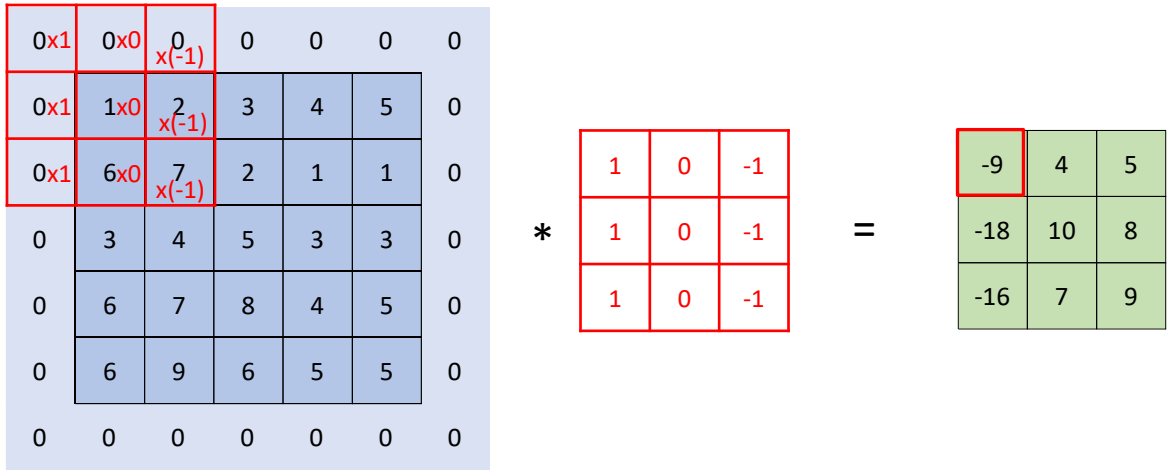
Stride

- The number of columns or rows traversed per slide.

Stride = 1



Stride = 2



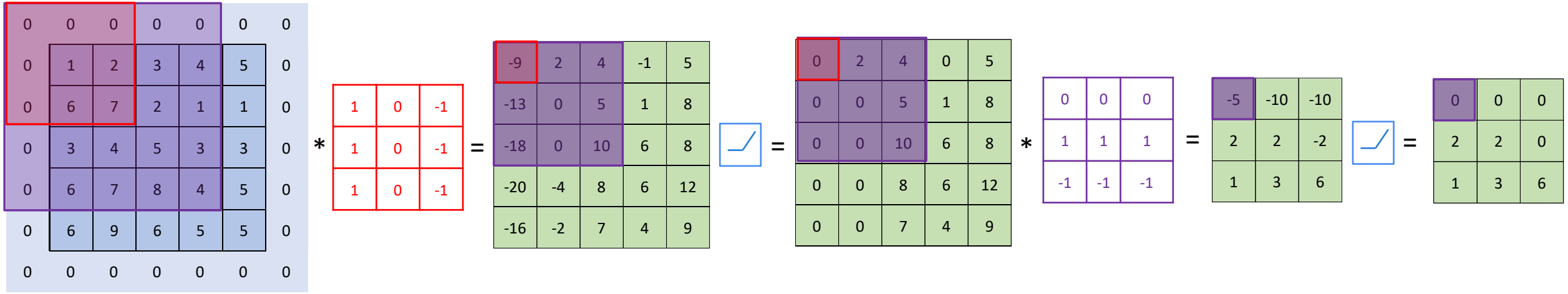
Terminology

Feature Map

- The convolutional layer output (after activation) is often referred to as a feature/activation map (i.e., learned representations or features) in the spatial dimensions (w x h).

Receptive Field

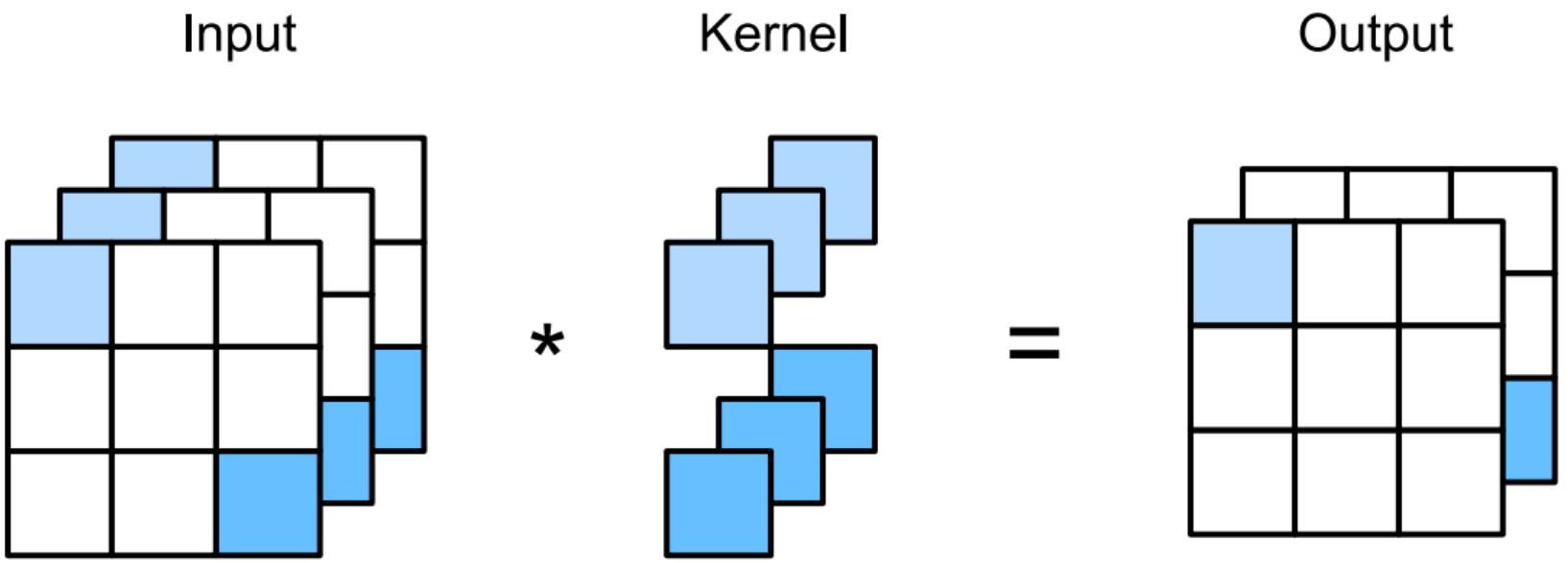
- For any element z of a layer, its receptive field refers to all the elements (from all the previous layers) that may affect the calculation of z during the forward propagation.



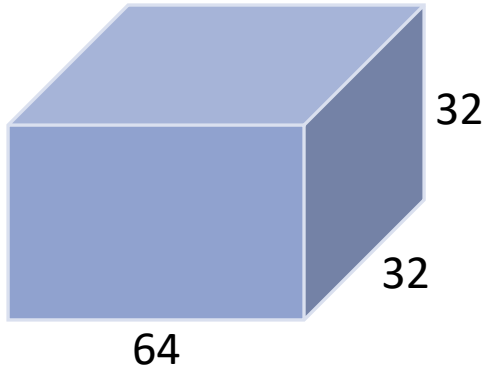
Special Case: 1x1 Convolution

No padding needed.

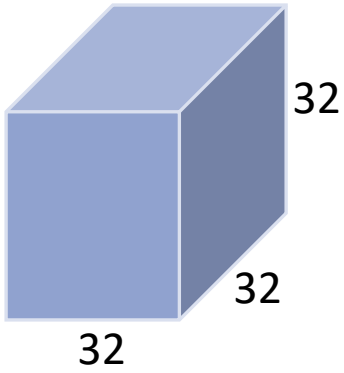
Typically with stride 1, resulting in same h x w.



1x1 Convolution

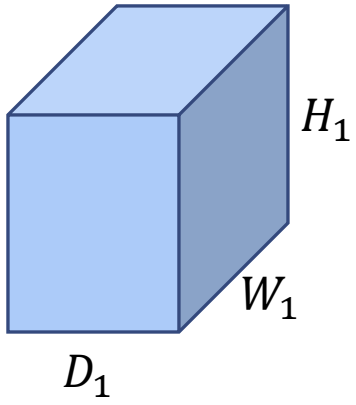


- Apply the same FC layer (shared weights) on each input pixel.
- Preserve spatial dimensions but reduces depth, i.e., projecting depth to lower dimension (combination of feature maps).



More generally

A volume of size
$W_1 \times H_1 \times D_1$



Four Hyperparameters

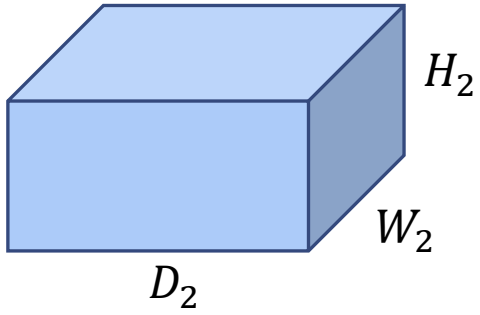
- Number of filters K
- Their spatial extent: F
- Stride: S
- Amount of zero Padding: P

$$W_2 = \frac{(W_1 + 2P - F)}{S} + 1$$

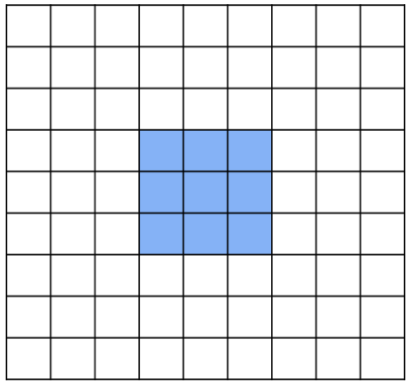
$$H_2 = \frac{(H_1 + 2P - F)}{S} + 1$$

$$D_2 = K$$

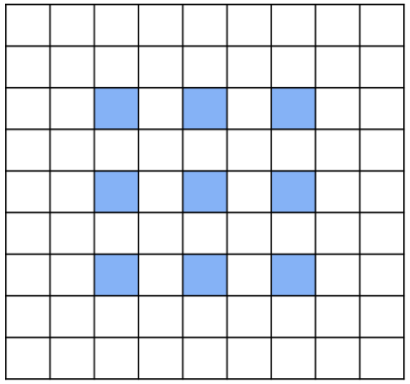
A volume of size
$W_2 \times H_2 \times D_2$



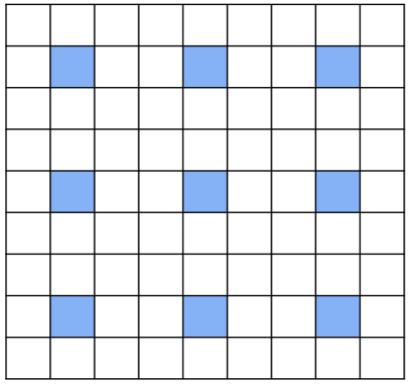
Dilation, non-square kernels, unequal stride and padding



dilation =1



dilation =2



dilation =3

$$W_2 = \frac{(W_1 + 2P_w - L_w(F_w - 1) - 1)}{S_w} + 1$$

$$H_2 = \frac{(H_1 + 2P_H - L_H(F_H - 1) - 1)}{S_H} + 1$$

PyTorch Example

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)

Non-squSquare kernels and equal stride

- nn.Conv2d(16, 32, 3, stride=2)

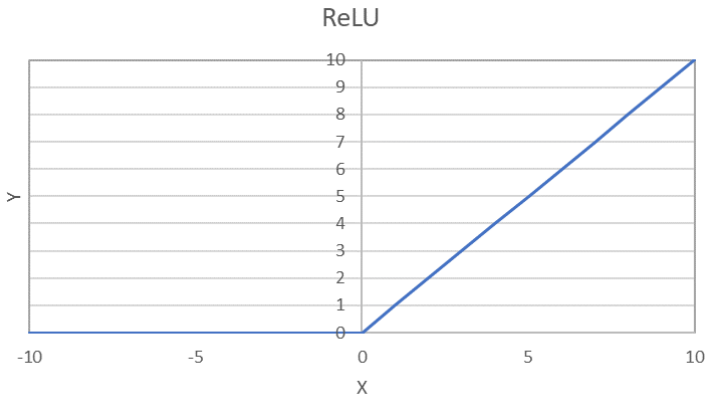
Non-square kernels and unequal stride and with padding

- nn.Conv2d(16, 32, (3, 5), stride=(2, 1), padding=(4, 2))

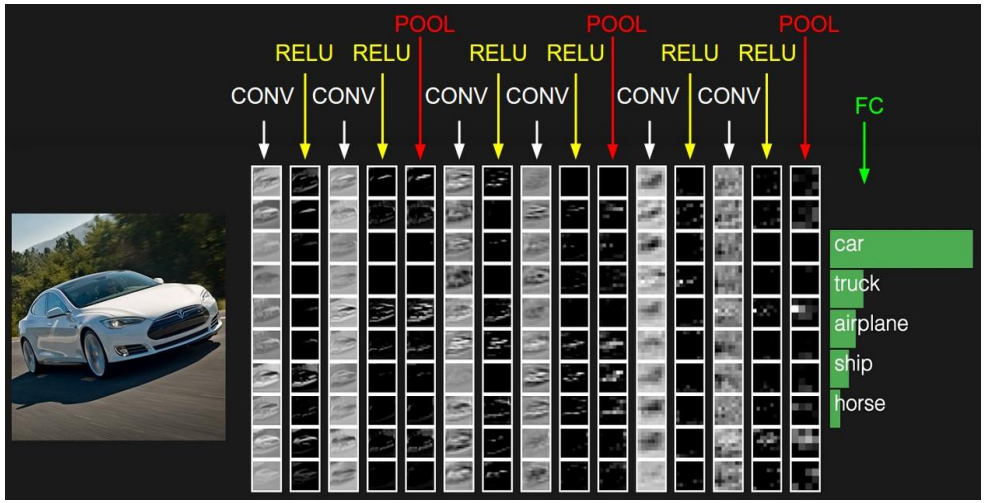
Non-square kernels and unequal stride and with padding and dilation

- nn.Conv2d(16, 32, (3, 5), stride=(2, 1), padding=(4, 2), dilation=(3, 1))

Rectified Linear Unit (ReLU)



$$f(x) = \max(0, x)$$



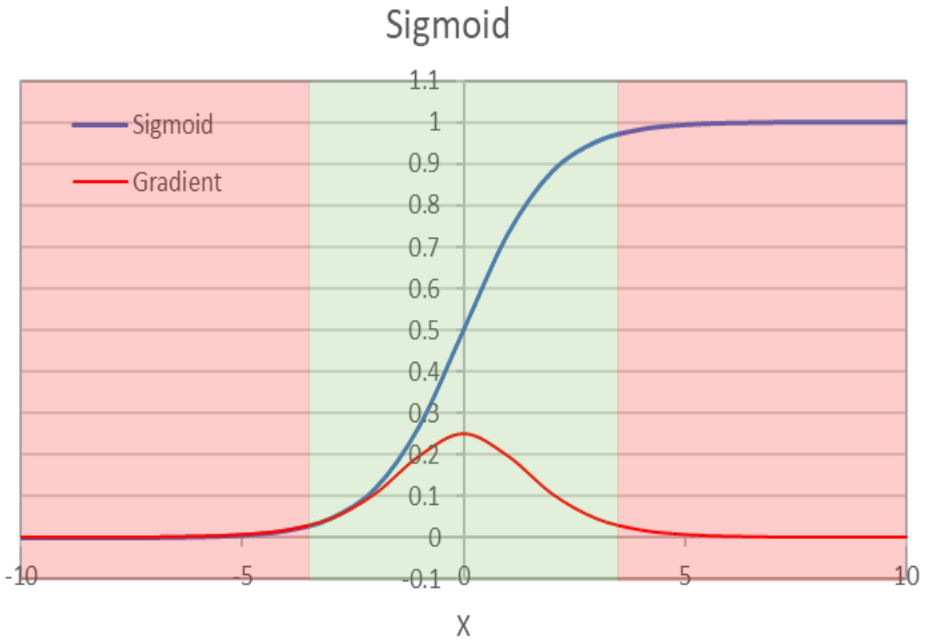
- Recall: convolution is just “linear” operation.
- Need nonlinearity to learn interesting features.

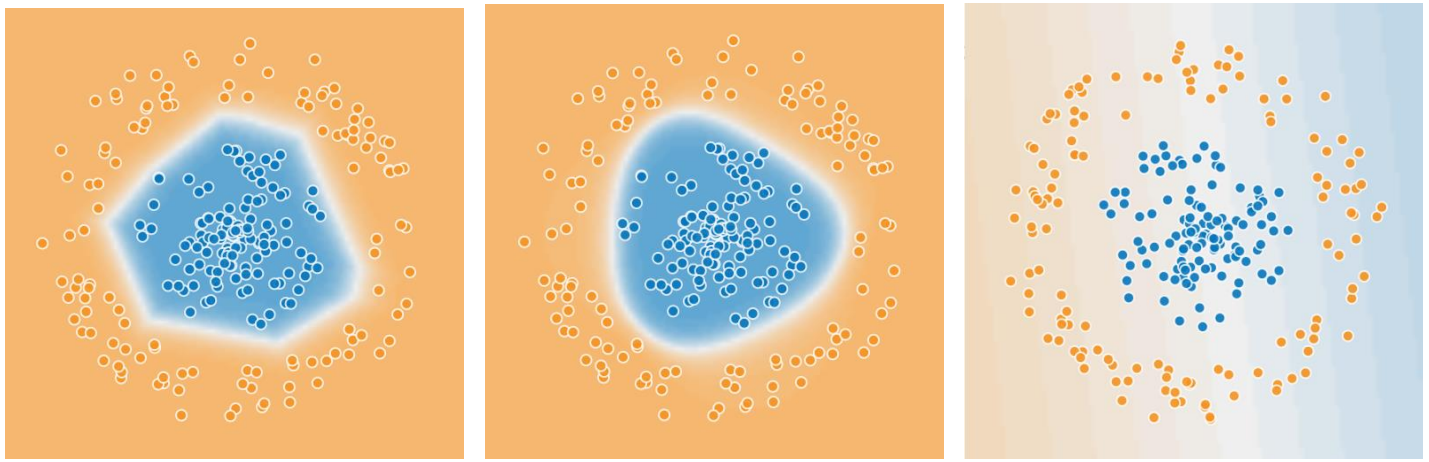
Why does ReLU work?

Add nonlinearity (piecewise linear).

Deal better with the vanishing gradient issue.

Easier to train.



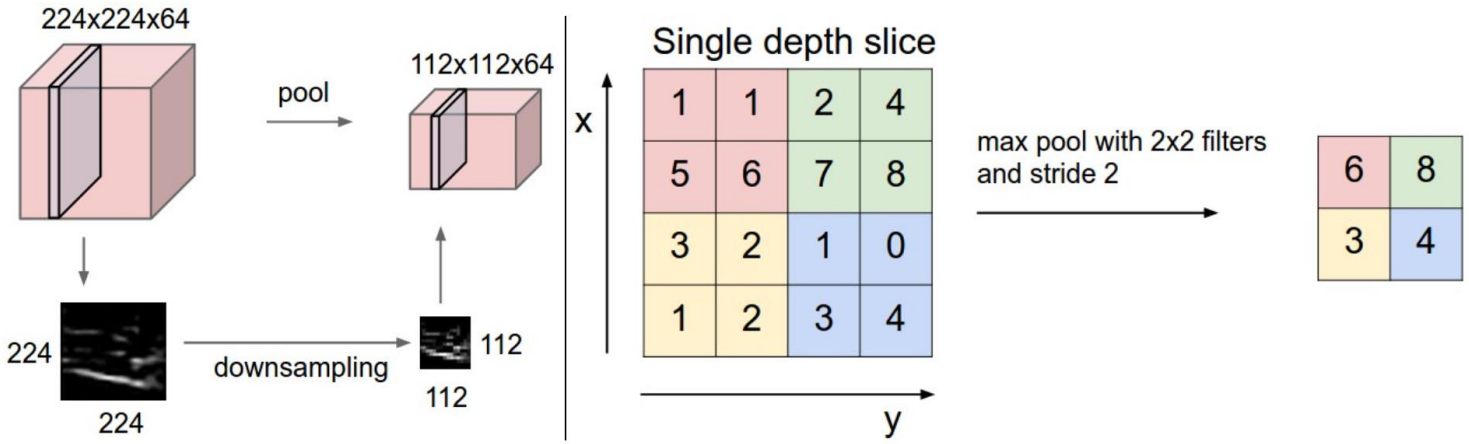


https://playground.tensorflow.org

The DNN structure requires a long path to back-propagate gradients. Each step backward, the gradient of the current layer is multiplied to the gradient of the previous layer according to the chain rule. If the gradients are small (e.g., < 1), the multiplication will lead to vanishing gradients, a common problem for training DNN with traditional sigmoid or tanh activation functions.

What is pooling?

- Often referred to as down-sampling.
- Results in a lower resolution version of the feature maps that are more robust to changes in the position of the features in the original image.
- Captures the feature structure at a larger scale.



https://cs231n.github.io/convolutional-networks/

Note: pooling is deterministic (e.g., max, average) and does not have parameters

Conceptual Illustration

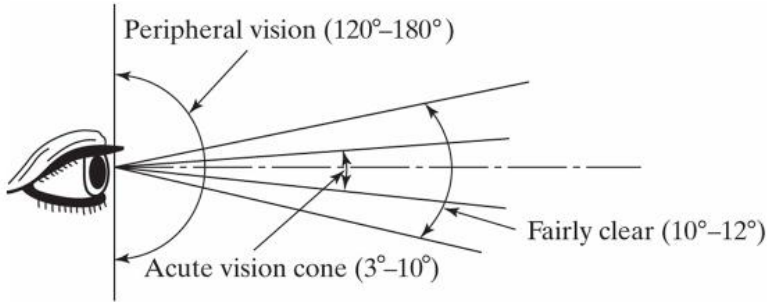
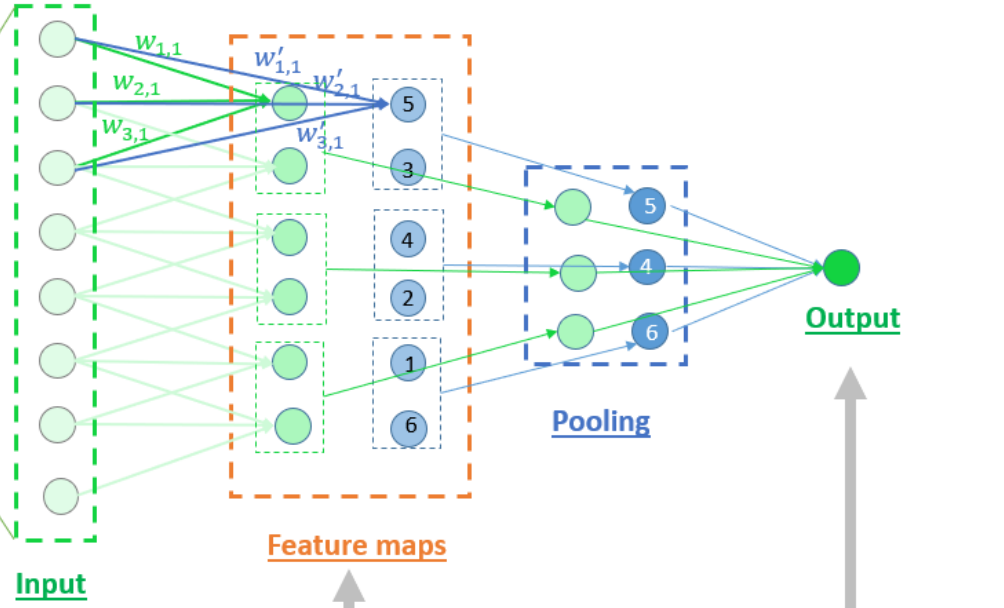


Image Source:
https://en.wikipedia.org/wiki/Leopard



Note: In this example, even though the image is 2D with 3 channels, the layers are shown as 1D for illustration purposes.

Peep leopard in a pipe (only see spots of the leopard).

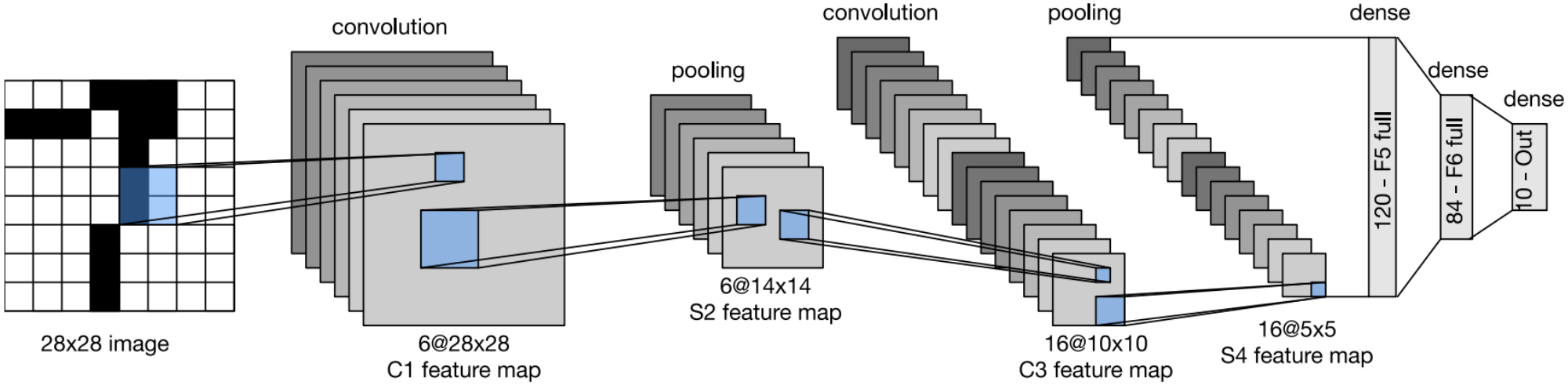
Low-Level Features

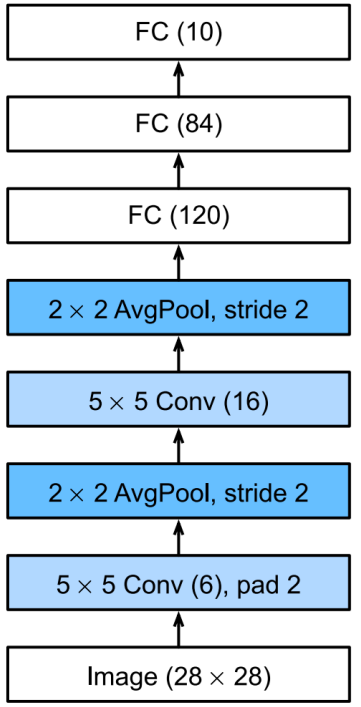
See the whole leopard.

High-Level Features

LeNet-5

Introduced by Yann LeCun for the purpose of recognizing handwritten digits in images. This work represented the culmination of a decade of research.





Note: sigmoid activation function and average pooling were used then.
Later, ReLU and max pooling are shown to work better.

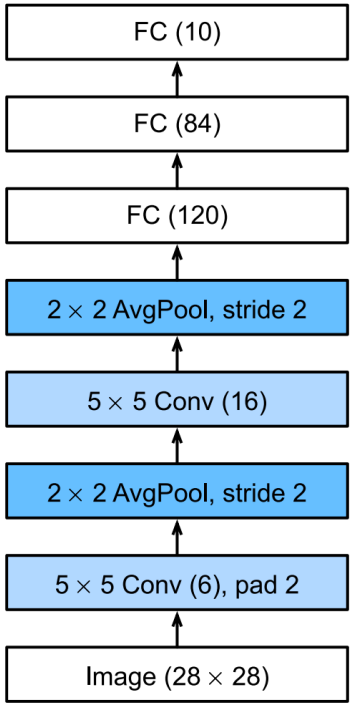
[LeCun et al. 1998] Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278–2324.

LeNet

LeNet seems to be complex. However, implementing it with modern deep learning frameworks (e.g., PyTorch) is remarkably simple.

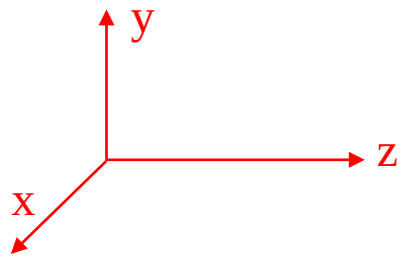
```
import torch
from torch import nn
from d2l import torch as d2l

net = nn.Sequential(nn.Conv2d(1, 6, kernel_size=5, padding=2), nn.Sigmoid(),
                    nn.AvgPool2d(kernel_size=2, stride=2),
                    nn.Conv2d(6, 16, kernel_size=5), nn.Sigmoid(),
                    nn.AvgPool2d(kernel_size=2, stride=2), nn.Flatten(),
                    nn.Linear(16 * 5 * 5, 120), nn.Sigmoid(),
                    nn.Linear(120, 84), nn.Sigmoid(), nn.Linear(84, 10))
```

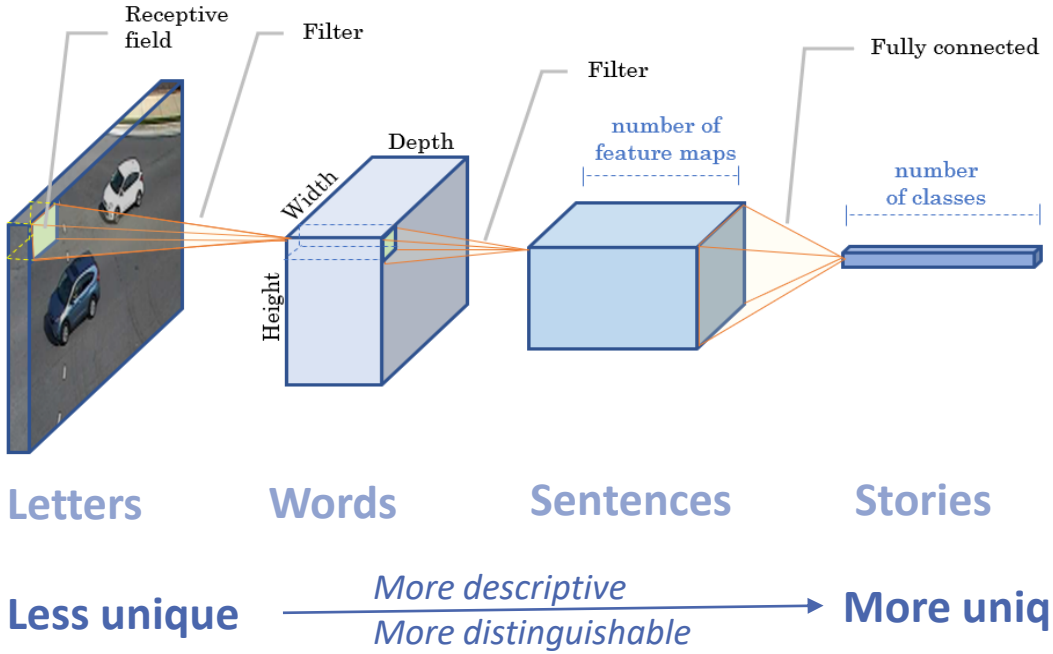


$$W_2 = \frac{(W_1 - F + 2P)}{S} + 1 \qquad H_2 = \frac{(H_1 - F + 2P)}{S} + 1$$

General Concept of CNN



- **Local connectivity**: Mimics visual cortexes containing neurons that individually respond to small regions of the visual field.
- **Shared weights**: Reduce the number of parameters to be learned and make the network more scalable and easier to train.
- **Multiple feature maps**: Allow a rich set of features to be captured at different levels and scales.
- **Pooling**: Results in a lower resolution version of the feature maps that are more robust to changes in the position of the features in the original image. It also captures the feature structure at a larger scale.

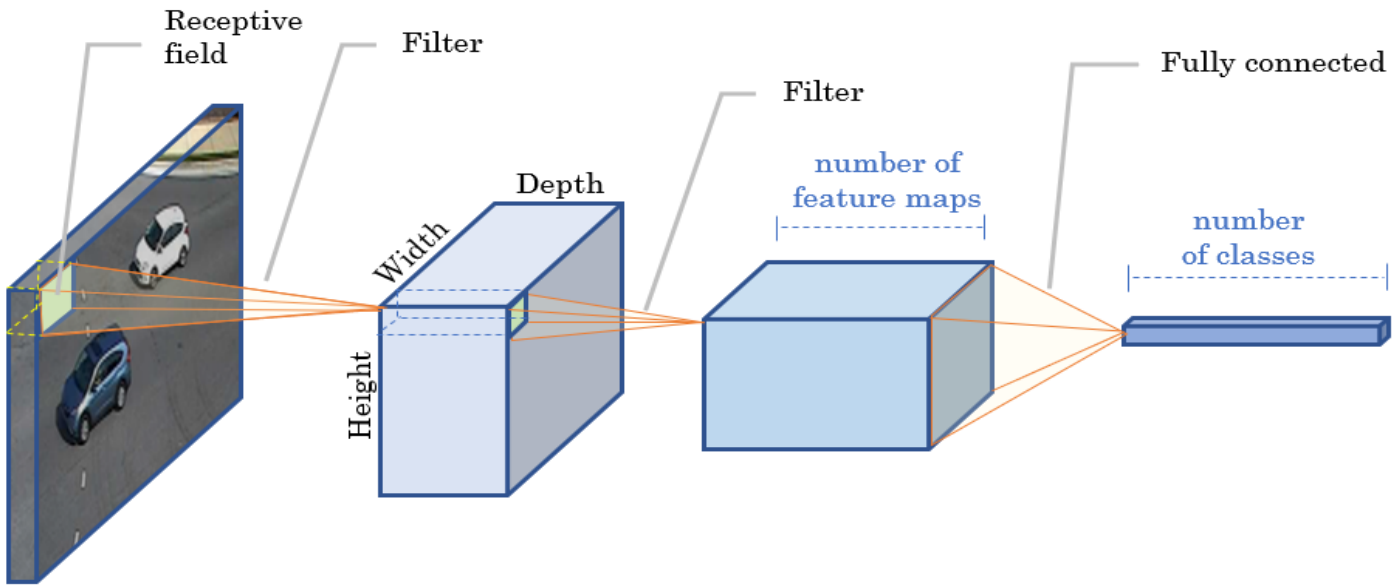


Note the spatial (x, y) dimensions are reduced while the depth/channel (z) dimension is increased as "convolution" progresses from the input layer to the output layer. Why? Does it make sense?

Tensor Representation

Conv layers: Four-dimensional tensors: [batch, **channel, height, and width**].

FC layers: two-dimensional tensors: [batch, **features**]



Fully Convolutional Networks

Can run on images of different sizes.

Multi-Scale Training

- Robust to images of different sizes

YOLOv2

- Use Darknet-19 as backbone
- Since the model downsamples by a factor of 32, image sizes of the multiples of 32: {320, 352, ..., 608} were used.
- Instead of fixing the input image size, every 10 batches the network randomly chooses a new image dimension size.

[Redmon & Farhadi, 2016] https://arxiv.org/pdf/1612.08242.pdf

Darknet - 19

Type	Filters	Size/Stride	Output
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	28×28
Maxpool		$2 \times 2/2$	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Maxpool		$2 \times 2/2$	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	1000	1×1	7×7
Avgpool		Global	1000
Softmax			