Image processing Fundamentals

Signals and Systems - Spring 2023

Presented by
Mehrshad Saadatinia

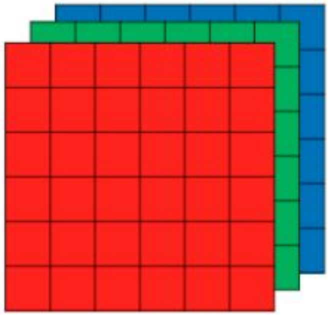
contents

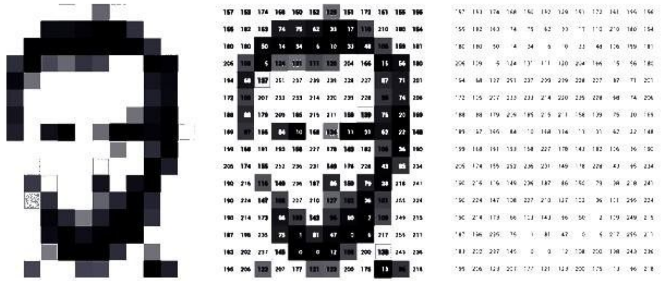
Different Signals (in terms of the dependent variable)

1. **Temporal**: EEG waves, radio signals, ... (1D)
2. **Spatial**: Images (2D or 3D)
3. **Combination of both**: Sequence of Images, Video

Images Are Signals

Images have spatial dependent variable, and are either 2-dimensional (grayscale), or 3-dimensional (RGB)





Images are stored in computer, as arrays (or volumes) of integers between 0 to 255

Image Processing

Image processing is a method to perform some operations on an image, in order to get an enhanced image or to extract some useful information from it.

It is a subcategory of signal processing

Some examples)

- Image Enhancement, Image reconstruction, Super-resolution,...
- Feature extraction, Segmentation, Classification, ...
- Image captioning, ...

Spatial Filtering

The use of filters in order to process images in a certain way. (enhance, extract features, ...)

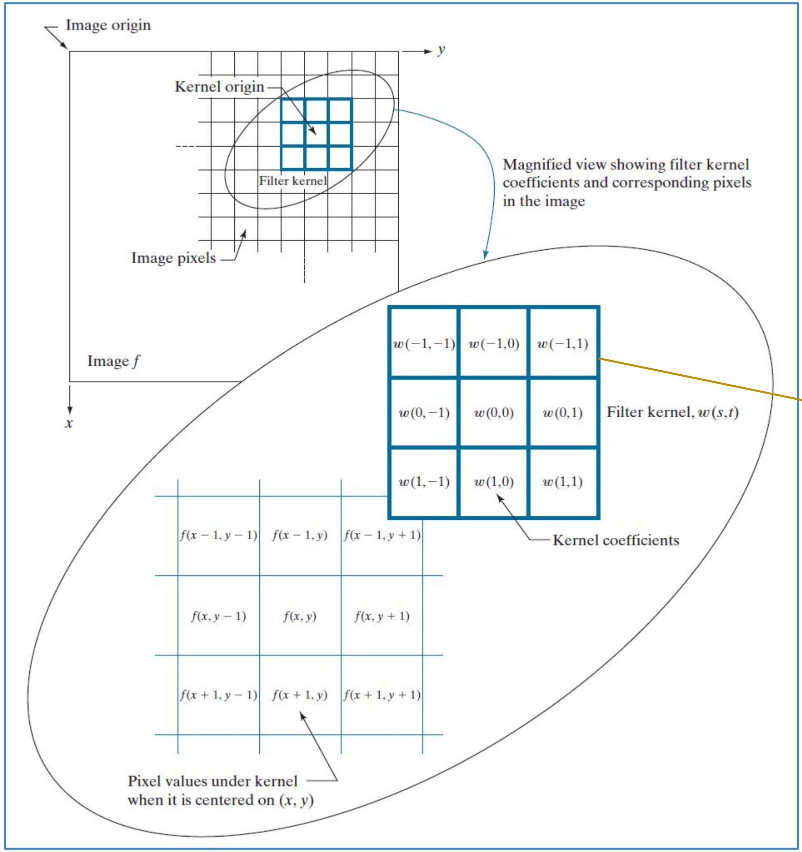
Spatial filtering modifies an image by replacing the value of each pixel by a function

of the values of the pixel and its neighbors.

The term filter comes from the fact that it accepts or rejects some frequencies, effectively filtering the image

A very essential tool for image processing

Mechanism of Linear Spatial Filtering



$$
\boxed{\begin{aligned} g(x,y) = w(-1,-1)f(x-1,y-1) + w(-1,0)f(x-1,y) + \ldots \\ + w(0,0)f(x,y) + \ldots + w(1,1)f(x+1,y+1) \end{aligned}}
$$

How to apply spatial filtering

Given a mxn kernel where: m=2a+1 and n=2b+1

This is how spatial filter is applied on the image

$$g(x, y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s, t) f(x + s, y + t)$$

Spatial (cross) correlation

2D convolution

2D convolution

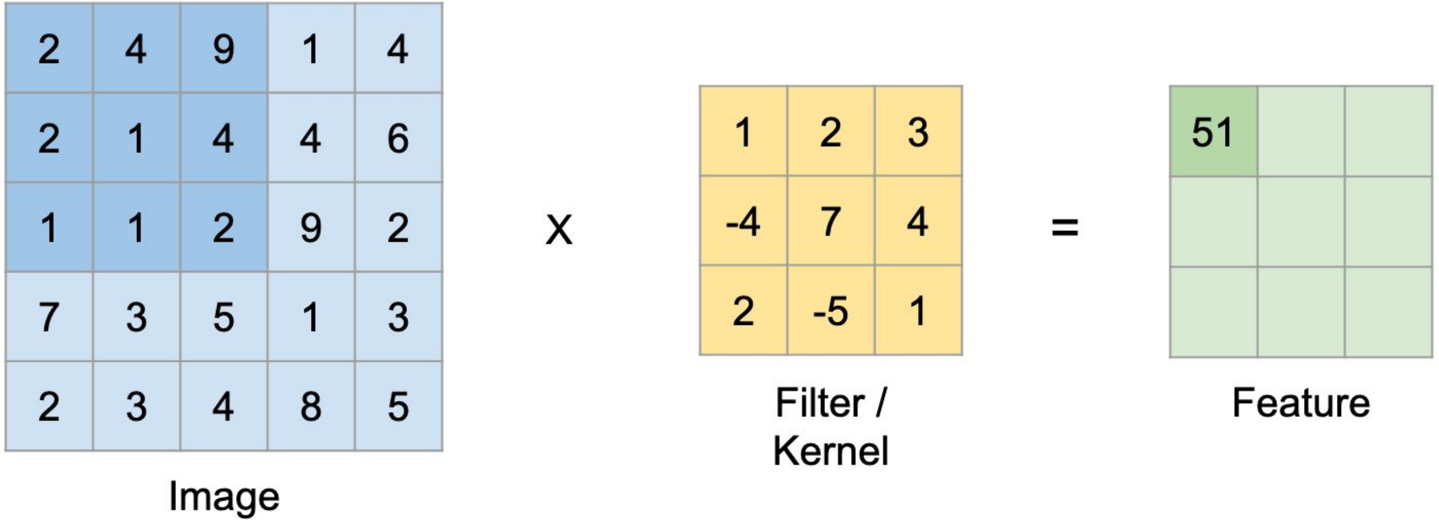
$$(w \star f)(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x-s, y-t)$$

1D convolution

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$

For symmetric kernels, correlation and convolutions are the same

2D convolution



What are the dimensions of output image

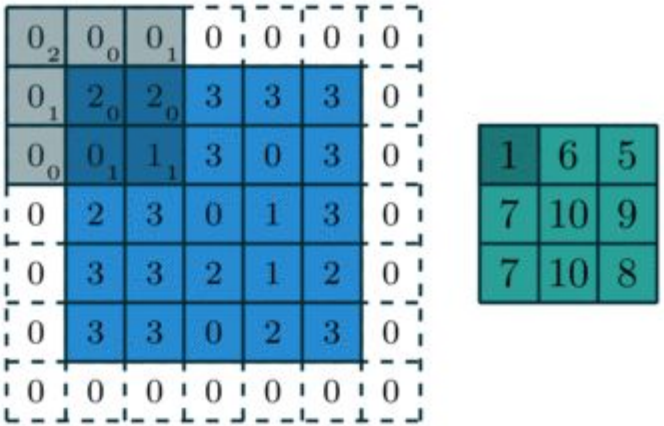
Input Image	*	filter	=	Output Image
$(m \times n)$		$(f \times f)$		$(m - f + 1) \times (n - f + 1)$

2D convolution

How to avoid size shrinkage?

2D convolution

How to avoid size shrinkage?



zero-padding

How much should we pad?

Input Image	*	filter	=	Output Image
$(m + 2p) \times (n + 2p)$		$(f \times f)$		$(m \times n)$

$m + 2p - f + 1 = m$ therefore: $2p = f - 1$

$$p = \frac{f - 1}{2}$$

How much should we pad?

In computer-vision terms:

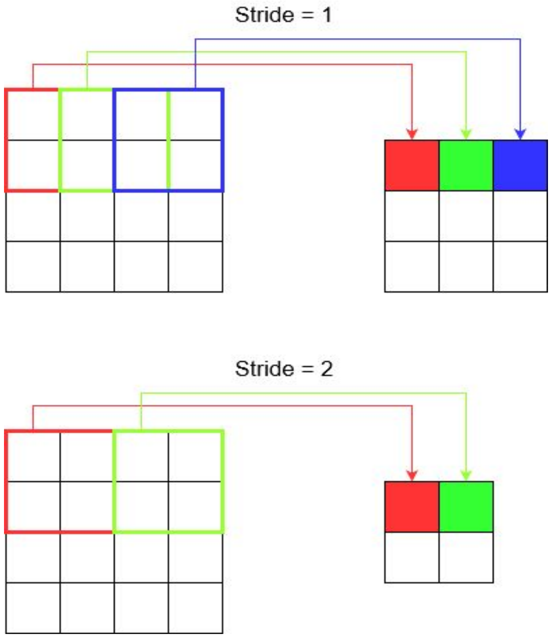
Valid padding means **no padding**

Same padding means **(f-1)/2 padding**

Full padding, increases output size

Taking larger steps
(stride)

Occasionally we may want to take larger steps than 1
We introduce stride for this purpose



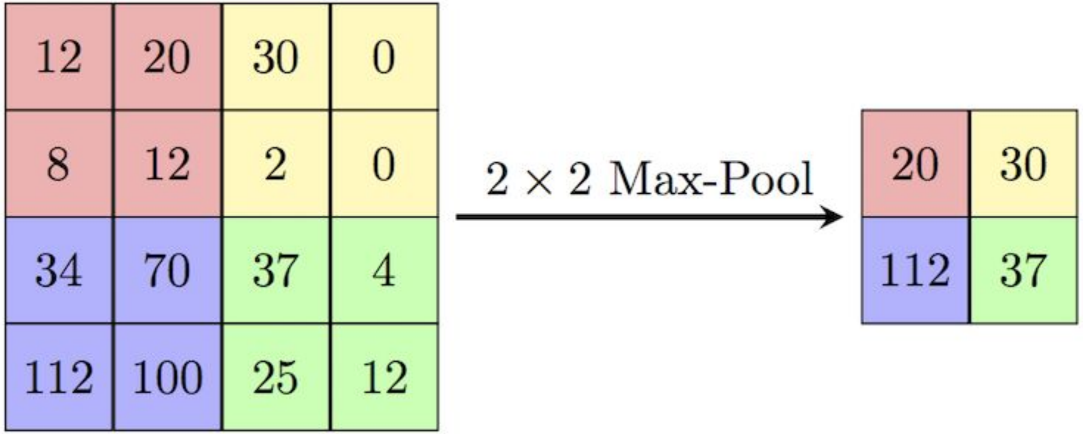
Taking larger steps
(stride)

Output size formula with padding and stride

Input Image	*	filter	=	Output Image
$(m+2p)\times(n+2p)$		$(f\times f)$		$(\frac{m+2p-f}{s}+1)\times(\frac{n+2p-f}{s}+1)$

Pooling layers

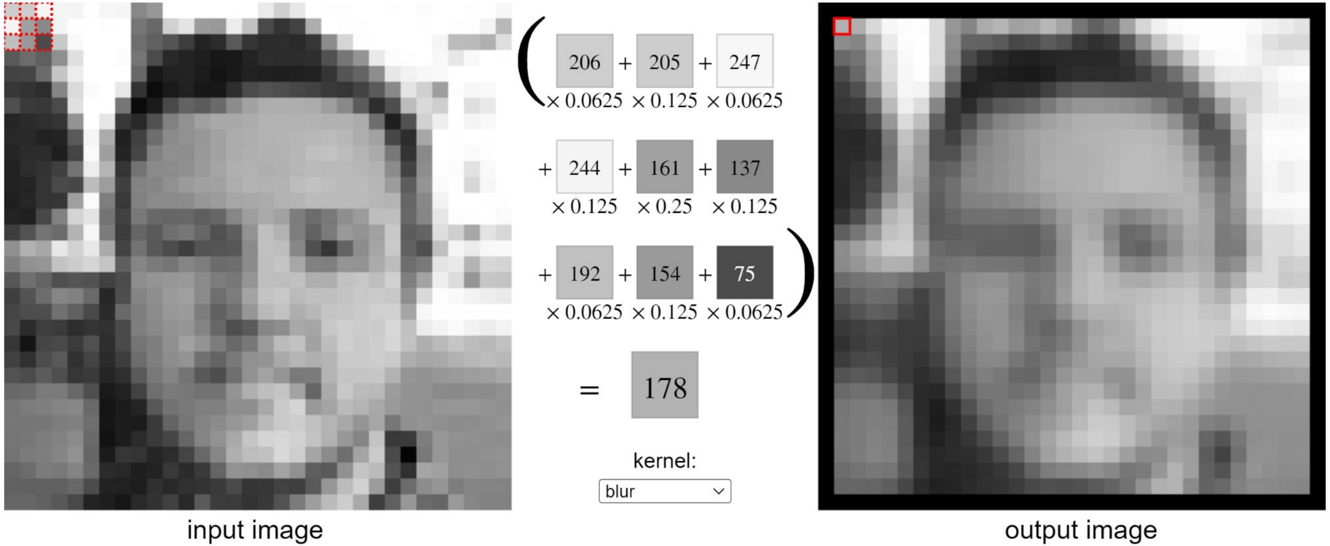
Is used to shrink the feature maps while keeping the most important features



Filters and their applications

Filters in practice

From setosa.io/ev/image-kernels/

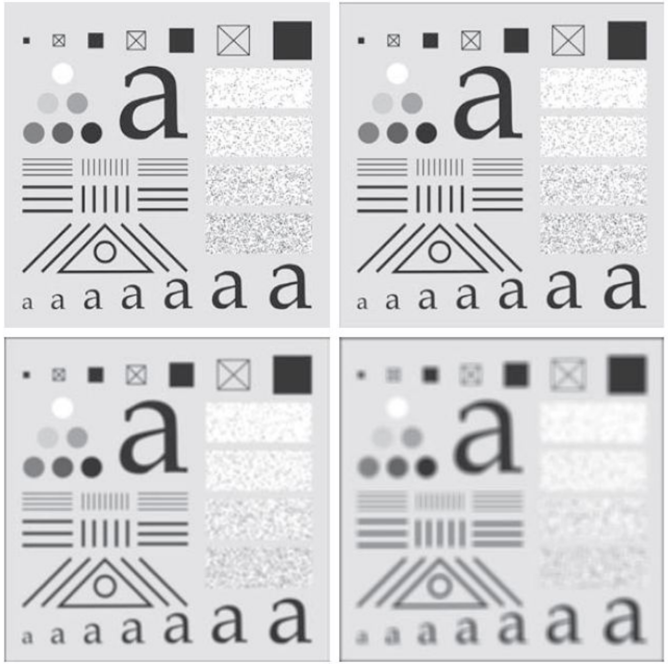


Smoothing (lowpass) filters

Average box filters (regular, weighted)

$$\frac{1}{9}\times\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix} \qquad \frac{1}{16}\times\begin{bmatrix}1&2&1\\2&4&2\\1&2&1\end{bmatrix}$$

Used mostly in image enhancement for noise (high freq. components) removal



Sharpening (highpass) filters

Used mostly in image enhancement for highlighting details, or emphasising obscure elements, rejects low frequency components

Laplacian filter

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

$$\begin{cases} \dfrac{\partial^2 f}{\partial x^2} = f(x+1, y) + f(x-1, y) - 2f(x, y) \\ \\ \dfrac{\partial^2 f}{\partial y^2} = f(x, y+1) + f(x, y-1) - 2f(x, y) \end{cases}$$

Sharpening (highpass) filters

$$\nabla^2 f(x, y) = f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) - 4f(x, y)$$

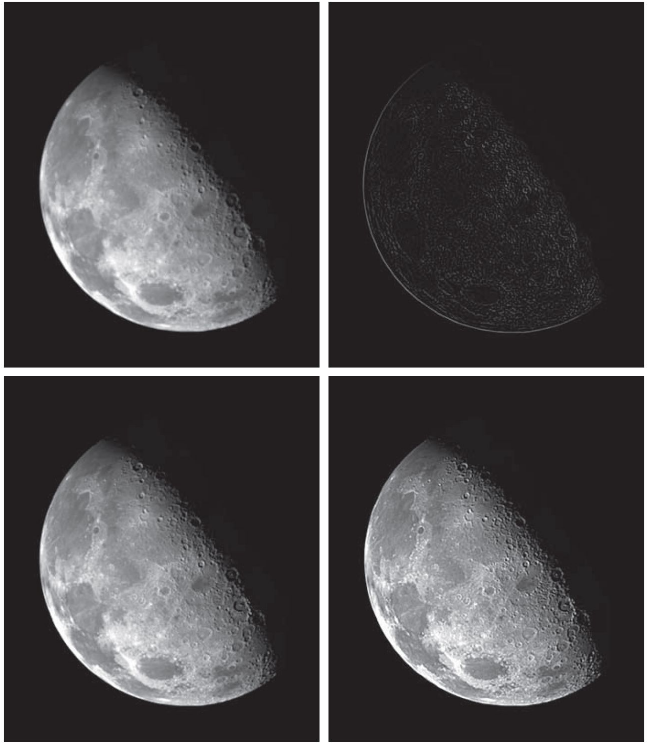
0	1	0
1	−4	1
0	1	0

1	1	1
1	−8	1
1	1	1

0	−1	0
−1	4	−1
0	−1	0

−1	−1	−1
−1	8	−1
−1	−1	−1

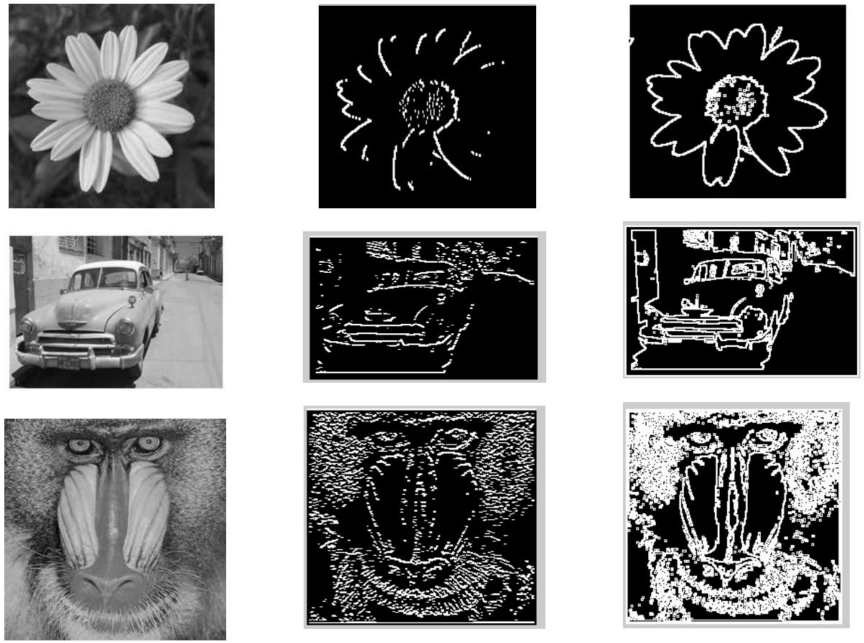
Applications in image enhancement



$$g(x, y) = f(x, y) + c\left[\nabla^2 f(x, y)\right]$$

Edge Detection

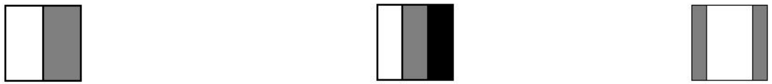
Used mostly when we want to detect the outline of image objects, useful in classification, feature extraction and ...



Edge Detection

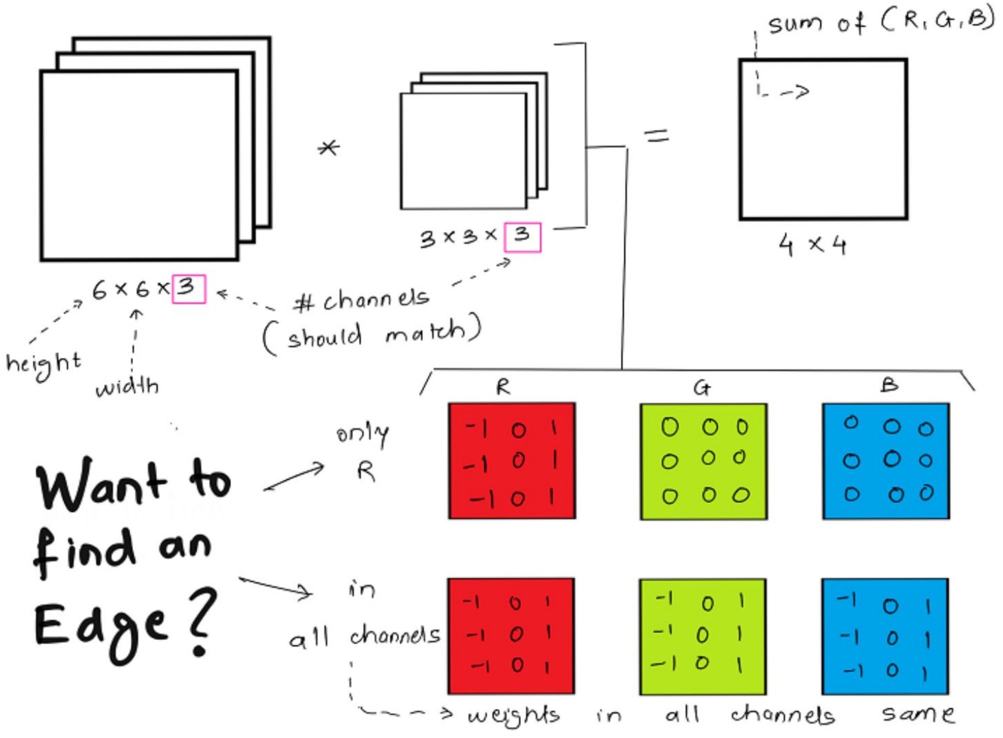
Some edge detection kernels include: sobel, robert, …

$$\underset{6 \times 6}{\begin{bmatrix} 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \end{bmatrix}} * \underset{3 \times 3}{\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}} = \underset{4 \times 4}{\begin{bmatrix} -0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \end{bmatrix}}$$



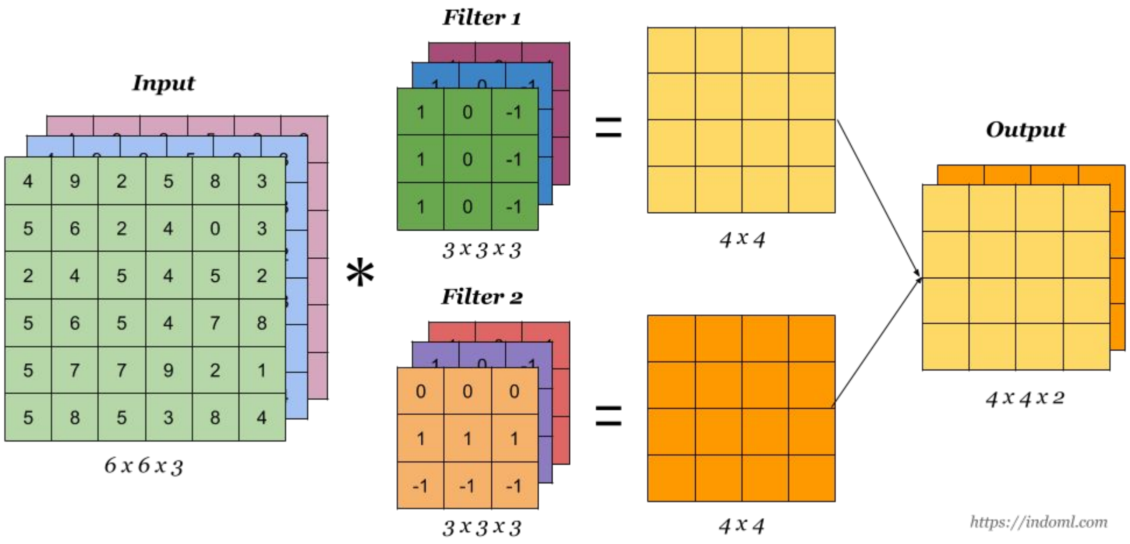
Convolution on RGB Images

How to we convolve an RGB image



Applying multiple kernels on an RGB image

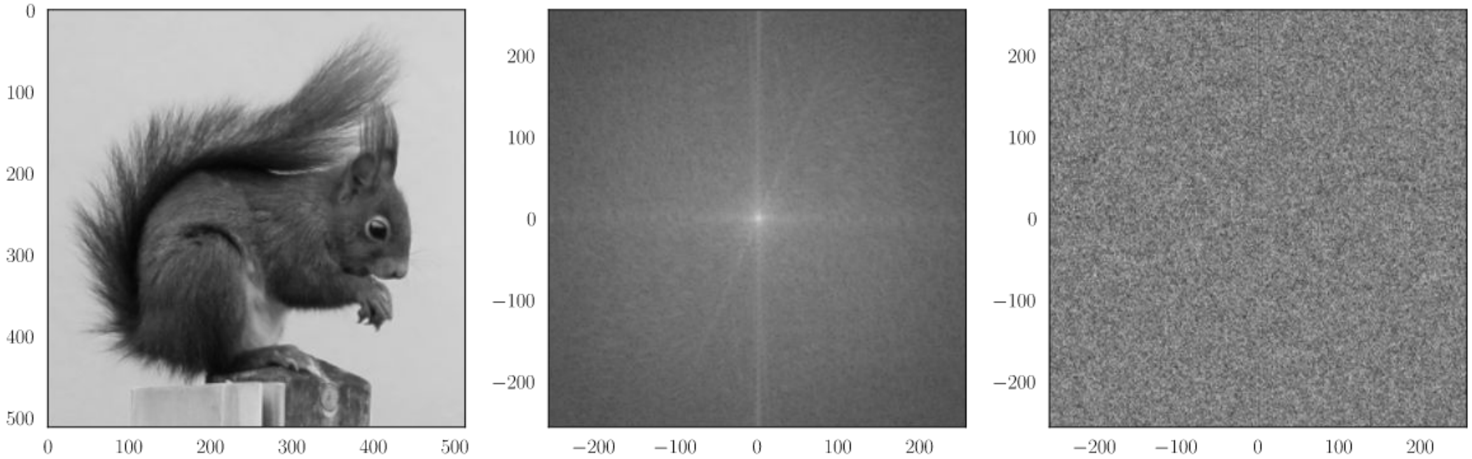
We apply each filter once and then stack all outputs to get a feature map



Filtering in Frequency Domain

If we take an image to Frequency Domain Using **Fourier Transform**, We can use multiplication to apply filters (instead of convolution), then bring the Image back to Spatial Domain using **Inverse Fourier Transform**

Fourier Transform of Image

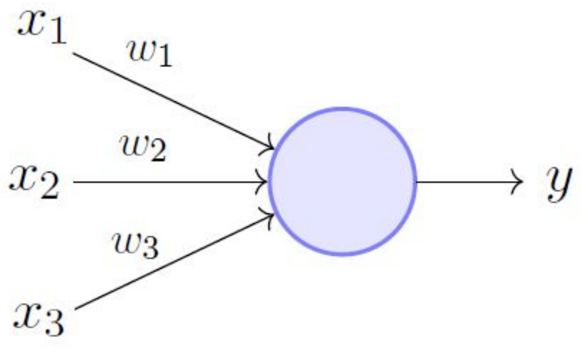


Convolutional Neural Networks

(Bonus)

Artificial Neural Networks

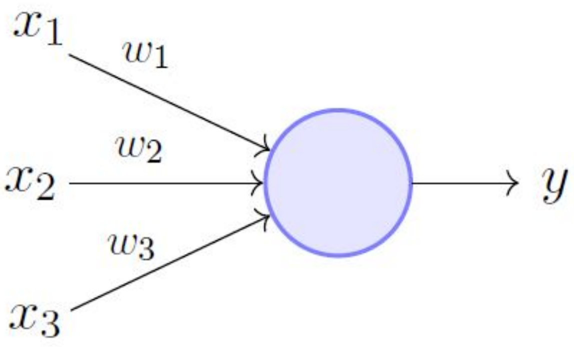
ANNs are generalization of the Perceptron Model



Perceptron Model (Minsky-Papert in 1969)

But what is a Perceptron?

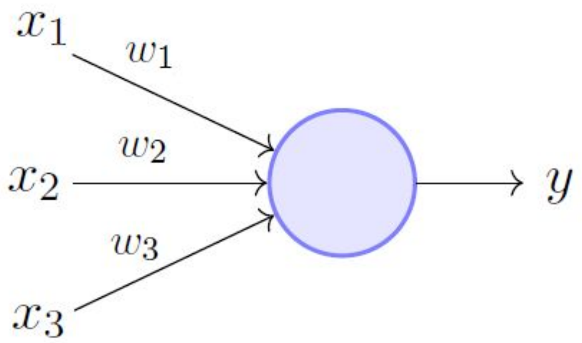
x1, x2, ..., xn are features, we want to find corresponding weights (w1, w2, ... wn) such that : **W.X** is close to our desired value **y**



Normally an activation function is applied to W.X to obtain non-linearity

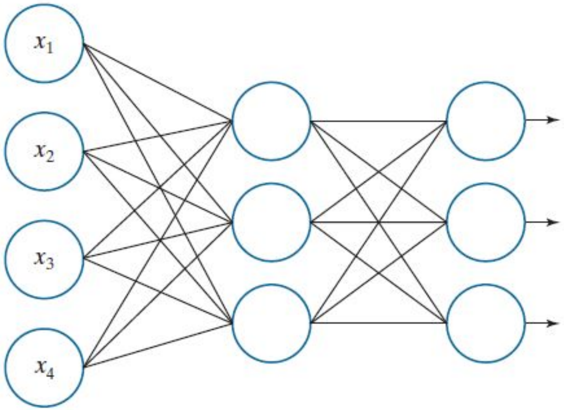
How do we find the optimal weights?

Weights are usually optimized using a process called Error backpropagation and with Gradient Descent algorithm



Multi-Layer Perceptron

As the name suggests, it is the perceptron in multiple layers
It is also called Fully connected Neural Network



$$\mathbf{W}(2) = \begin{bmatrix} 2.393 & 1.020 & 1.249 & -15.965 \\ 6.599 & -2.705 & -0.912 & 14.928 \\ 8.745 & 0.270 & 3.358 & 1.249 \end{bmatrix} \qquad \mathbf{W}(3) = \begin{bmatrix} 4.093 & -10.563 & -3.245 \\ 7.045 & 9.662 & 6.436 \\ -7.447 & 3.931 & -6.619 \end{bmatrix}$$

$$\mathbf{b}(2) = [4.920 \quad -2.002 \quad -3.485]^T \qquad \mathbf{b}(3) = [3.277 \quad -14.982 \quad 1.582]^T$$

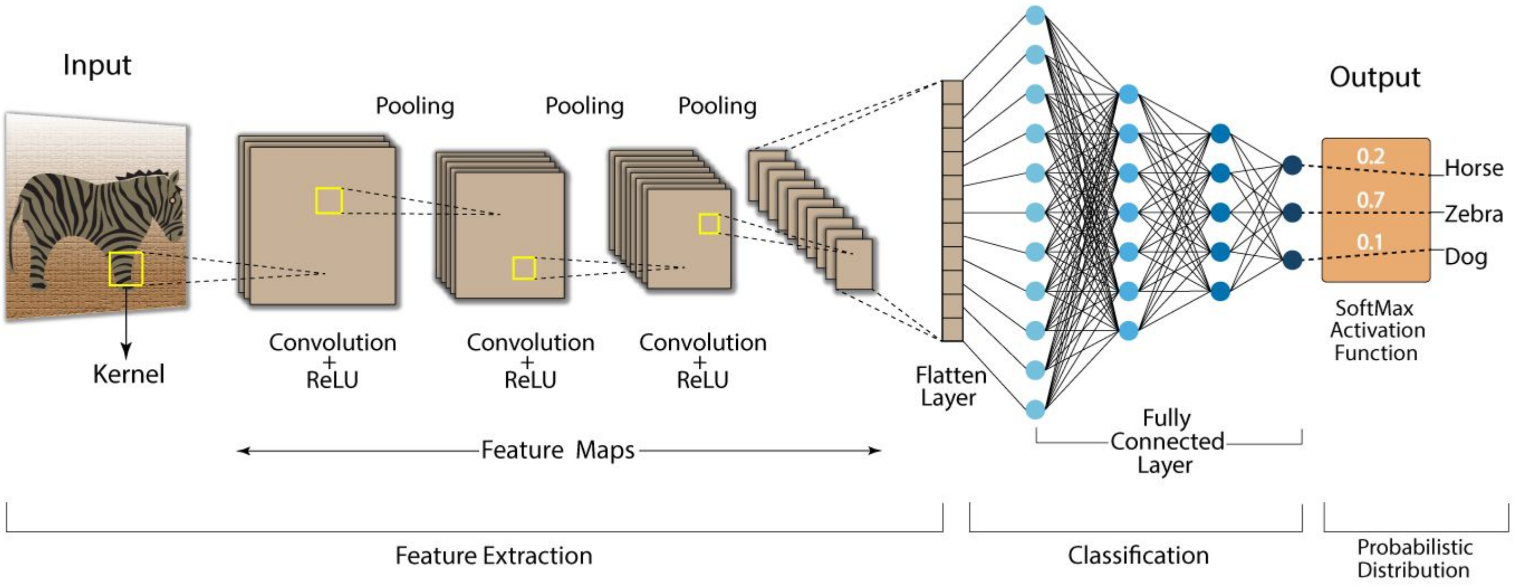
But FC Neural Nets fail to perform well on high quality images, Why?

A 128x128 image has a total number of more than **16K** features
Too large to compute weight for it using a fully connected Neural Net

The alternative?

Convolutional Neural Networks

Convolutional Neural Networks (CNN)



CNN is one of the biggest innovations in Deep Learning and specially Image Processing and has various applications in: Classification, Object Recognition, Segmentation, Feature Extraction, Image Generation and etc.

CNN’s Feature Extraction in practice