Data Science - Module 5

Machine Learning
Computer Vision in Python

Andrea Cossu – andrea.cossu@sns.it
Scuola Normale Superiore
University of Pisa

Lecture / Lab Outline

- Working with images
- Transformations on images
- Basic filters / edge detection with Python
- Torchvision package
- Convolutional Neural Networks

https://www.pinterest.co.uk/pin/113998052813119777/

What is an image

- An image is a grid (matrix) of pixels
- Consider gray-scale images
- Each pixel has an intensity value (0-255, 0-1)
  - Large values → "brightness" (white)
  - Small values → darkness (black)
- Size of matrix is Width x Height
- Each pixel can be considered as a feature
- Very large dimensional space for most of the images

Multi channels images

- A channel is a single matrix of pixel intensities
- You can combine multiple channels
  to get a colored image
- Standard RGB images are obtained by combining three matrices
  - Red (W x H) (0-1, 0-255)
  - Green (W x H) (0-1, 0-255)
  - Blue (W x H) (0-1, 0-255)
- Even more features! Size: W x H x C or C x W x H
- An additional channel (alpha channel) is often added to manage transparency of each pixel.

Color spaces

- Gray, RGB, BGR, HSV… (many, many others)
- Each color space determines the final color based on the combination of its components
  - RGB → specify brightness for each base color
  - HSV → specify hue, saturation and brightness (value)
  - http://colorizer.org/
- Different color spaces suits different needs. You can use standard RGB if you are unsure.

Color histogram

- Describe content of image based on color(s) distribution
- For each channel, plot histogram of intensities
- Useful to roughly compare images

1D Convolutions

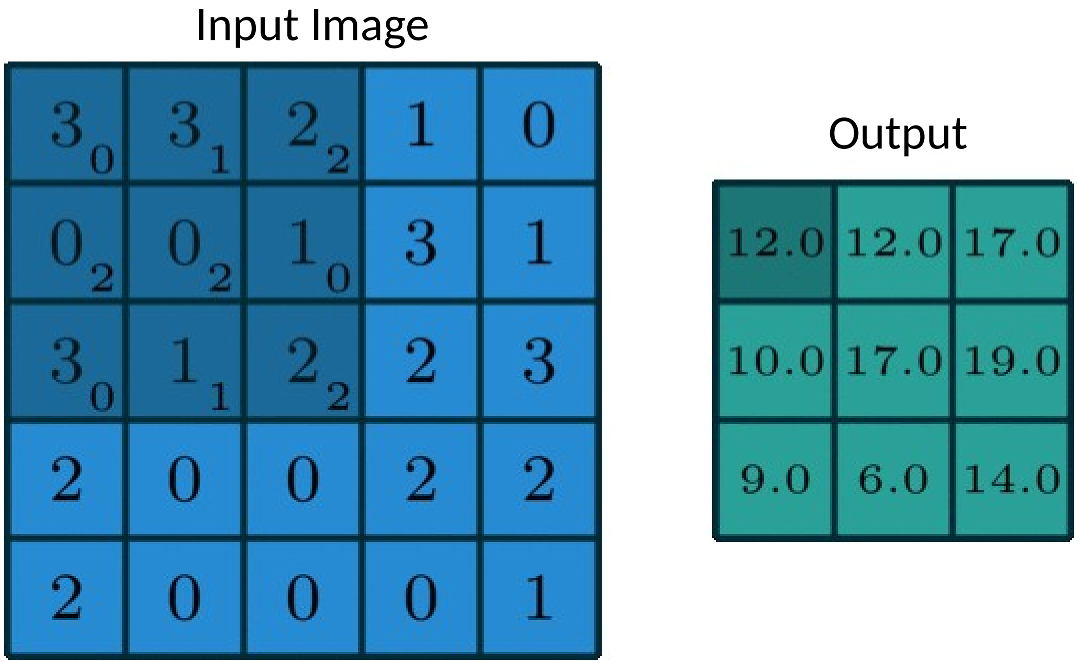
- Time series I, filter/kernel F
- I longer than F
- 1D convolution (flip kernel)

$$(I * F)(t) = \int_{-\infty}^{\infty} I(\tau)F(t-\tau)d\tau$$

- Cross-correlation is very similar, but filter is not flipped

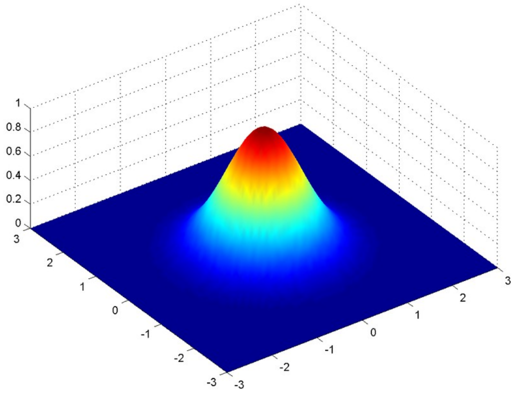
2D convolutions

- Image I, filter/kernel F. Both are 2D matrices.
- I larger than F
- Filter = $\begin{matrix} 0 & 1 & 2 \\ 2 & 2 & 0 \\ 0 & 1 & 2 \end{matrix}$
- To flip or not to flip?
  - Assume already flipped
  - Otherwise, call it cross-correlation



https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1

Smoothing



https://commons.wikimedia.org/wiki/File:Gaussian_2d.png

- Let's use convolutions!
- Smoothing / Blurring an image
- Need convolution with a special filter
  - Gaussian filter of fixed size $exp\left(-\frac{H^2+W^2}{2\sigma^2}\right)$
  - There are many other choices depending on what you want to filter out
- Standard deviation defines the "aperture" of the filter

Edge detection

- What is an edge?
  - A region (actually, more a line) in which color intensity drastically changes
- How can we identify that? Gradients!
- Prewitt operator → two matrices, representing finite difference method on horizontal and vertical direction
- Sobel extend Prewitt by adding "focus" on the central difference

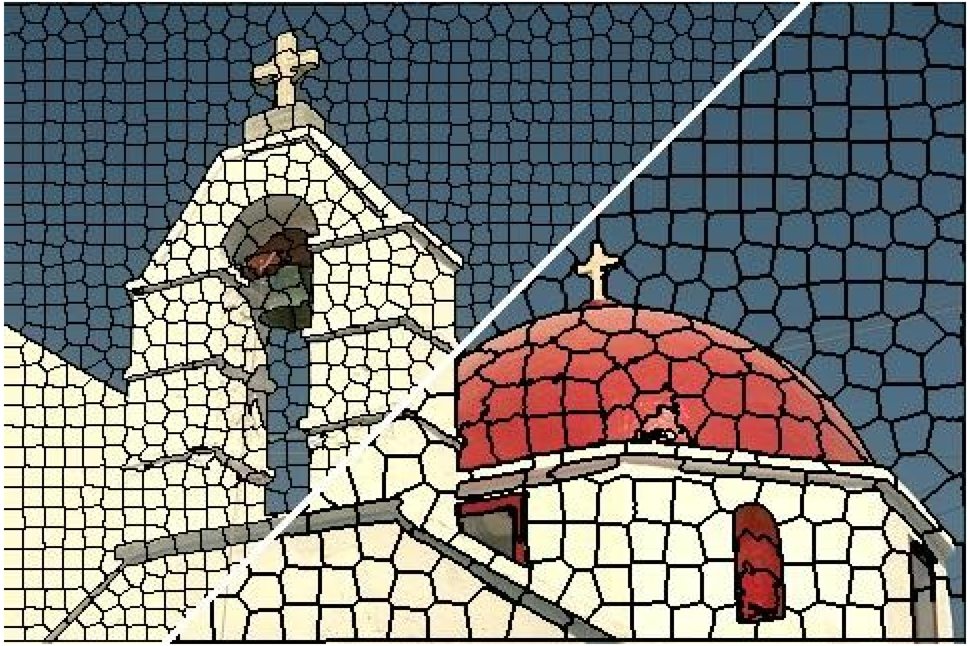
$$G_x = \begin{matrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{matrix} \qquad G_y = \begin{matrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{matrix} \qquad G_x = I(x+1, y) - I(x-1, y)$$

Image segmentation

- Group together pixels, hopefully discovering semantically relevant regions
- Mmmm… sounds a lot like clustering
- Try out K-Means on an image
- Get clusters as result
- Plot the image by coloring pixels belonging to the same cluster with the same color
- It helps to provide additional features (like positional information about pixel) and to choose an appropriate color space
  - Random link: https://en.wikipedia.org/wiki/CIELAB_color_space

Superpixels

- Merge together similar pixels
- Can greatly simplify feature space
- Remove redundant information
- Do not need any task-specific information



https://i.stack.imgur.com/XjYuD.jpg

Applications of Computer Vision

- Image classification → classify the entire image
- Object recognition → classification of (possibly multiple) objects
- Object detection → bounding boxes + classification
- Image segmentation → assign each pixel to a "cluster"

TorchVision: PyTorch for Images

- Tightly associated to Pytorch → ML/DL oriented
- You won’t find a lot of functions which are in OpenCV
- You will find a lot of functions to manage images with neural networks
- You will find datasets → download, data loading, batching, shuffling… automatically managed.

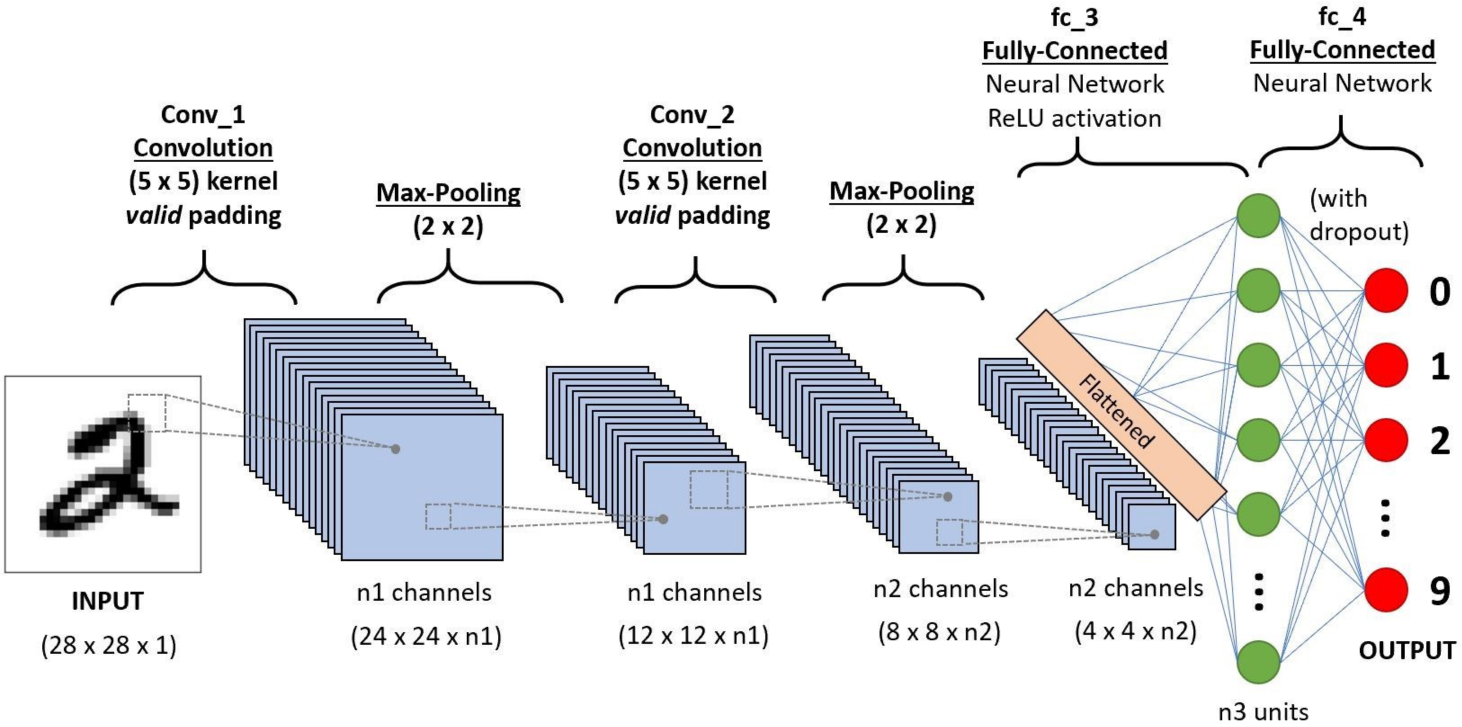
How to manage images with neural networks?

- Feedforward networks: MLP
  - Flatten image, each pixel is a feature
  - Large input layer
  - Very expensive in the #paramaters
- Convolutional Neural Networks: CNN
  - Inductive bias: spatial locality → exploit the fact that we are dealing with images
  - Shared parameters → same adapted filter, applied over image
  - End-to-End: backprop + gradient descent

Convolutional Neural Networks (CNNs)

- Convolutions → we have already seen them
  - In CNN, the parameters are the kernel values (initialized randomly)
  - Padding: add zero to the border of the image s.t. image size stays the same (same padding). Valid padding is sometimes used to indicate no padding
  - Stride: how many pixels to skip when sliding the kernel over the image
- Pooling → ensure invariance to small rotation / translations
  - Use a kernel which takes the maximum of the values in the current region of the image → downsizing image
- CNN = consecutive block of conv + relu + pooling
- At the end, flatten result + MLP to classify

CNN structure



https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Implement a CNN vs. use a CNN

- CNN architecture is complex, there exist many different ones and it is not obvious how to build them
- Images are not simple datasets → needs lots of patterns to train a CNN on reasonably complex tasks
- Implement a CNN from scratch → only if you cannot do otherwise
- Use a *pretrained* CNN and *finetune it*:
  - CNN trained on a large corpus of images (e.g. ImageNet)
  - Remove the MLP component
  - Add your MLP and train it end-to-end (slower learning rate for the convolution kernels) on your custom, smaller dataset. Most of the capabilities you need are already in the CNN!