Deep Scene Understanding from Images for Monitoring Applications

Matteo Poggi, Fabio Tosi, Pierluigi Zama Ramirez
Computer Vision Lab (CVLab), University of Bologna

Computer Vision Laboratory
ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

About us:

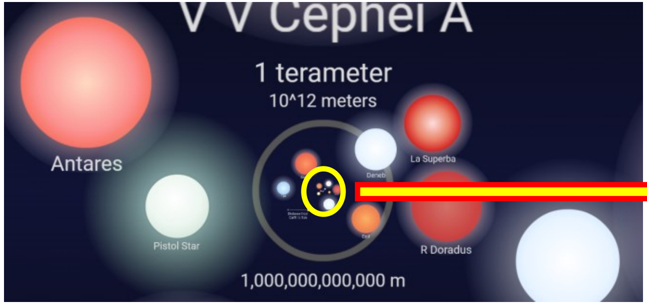
Matteo Poggi (RTDB @ CVLab)

Fabio Tosi (postDoc @ CVLab)

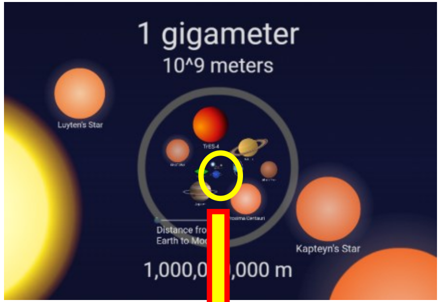
Pierluigi Zama Ramirez (postDoc @ CVLab)

Our goal

We wish to show you cool applications and results in the field of computer vision, without limiting too much the scope of our course. We also hope to include attendees with **any degree of expertise**



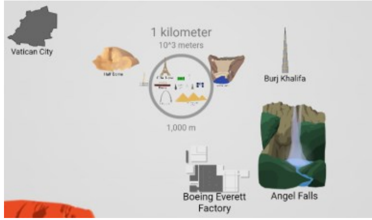
Whole human knowledge



Whole Computer Vision knowledge

You are going to land here!

Deep Scene Understanding and related tasks knowledge



Depth estimation knowledge

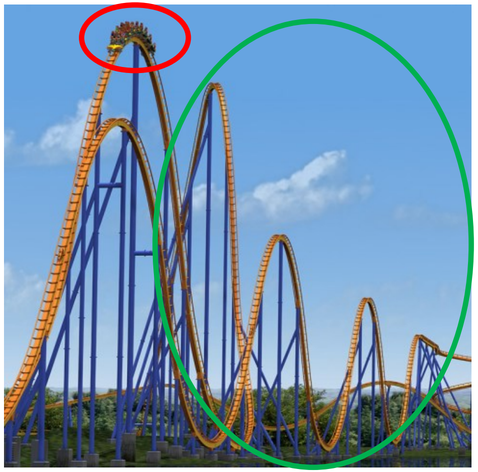
Credits: https://htwins.net/scale2/

Broad Topics

Problems and methods to extract knowledge about the surrounding environment **from images**

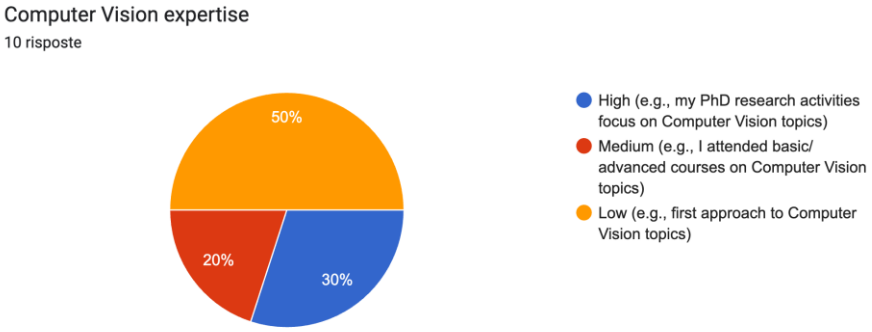
Prerequisites

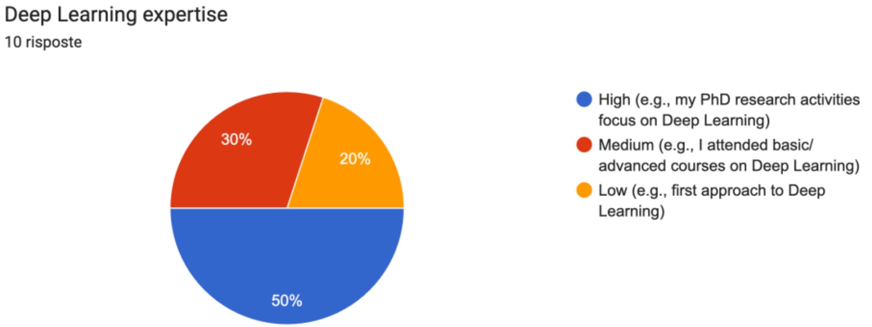
"Basic" knowledge about **computer vision*** and **deep learning** (**CNNs** in particular)



* the very basic, necessary concepts will be introduced, so do not worry :)

Who is the course thought for?





Schedule:

6 lessons (**3 hours** each), final exam (**2 hours**)

- Lesson 1 (July 17, 10.00-13.00)
- Lesson 2 (July 18, 10.00-13.00)
- Lesson 3 (July 19, 10.00-13.00)
- Lesson 4 (July 24, 10.00-13.00)
- Lesson 5 (July 25, 10.00-13.00)
- Lesson 6 (July 26, 10.00-13.00)

- Final exam (July 27, 10.00-12.00)

The final exam will consist of a report about one paper/topic discussed in lessons 1-6
Either during the dedicated lecture (July 27), or later at your convenience

1- Introduction

Summary of contents:

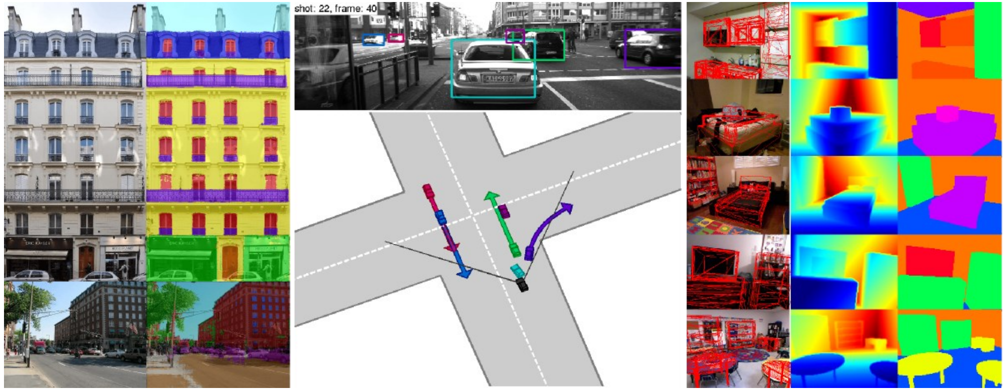
- **Scene understanding, what and where**:
  an introduction to scene understanding, why we do need it, digital images, some basic information we can extract from them and related problems

- **Analytical vs Data-Driven approaches:**
  how to approach to computer vision problems, algorithms vs neural networks, Convolutional Neural Networks (CNNs)

- **Supervision paradigms:**
  basic principles to train a deep learning-based approach to deal with computer vision problems, advantages, limitations and costs

Scene understanding, what and where

Interpreting the environment around us and the agents interacting with it is an important prerequisite necessary to design several **applications**



Credits: https://ps.is.mpg.de/projects/scene-understanding

For instance, let's suppose we want to implement an **assistive/autonomous driving system.**
To properly navigate through traffic, our vehicle needs to be **aware** of what is happening around it.

Scene understanding, what and where

In the last years, we witnessed a race towards implementing **autonomous driving systems**.
A common taxonomy of systems has been defined, according to the level of automation deployed:

Level 0: **No Driving automation** – The driver entirely control the vehicle

Level 1: **Driver Assistance** – Basic assitance in breaking or steering (adaptive cruise control/breaking)

Level 2: **Partial Automation** – ADAS, assistance in both breaking and steering (Tesla AutoPilot)

Level 3: **Conditional Automation** – ADAS programmed with environmental detection features, allowing for self-drive in certain conditions (Audi Traffic Jam Pilot)

Level 4: **High Automation** – higher-level assistant capable of making decisions when ADAS fails, with a human passenger still present (thought for driverless taxi/vehicles on a fixed route)

Level 5: **Full Automation** – the vehicle is entirely autonomous and can drive everywhere (no pilot)

Scene understanding, what and where

Our system should both know **what** is facing and **where**.
In the case of a **driving agent**, some examples of meaningful information are:

What (semantical content)

- Understand traffic signals/lights
- Recognize sidewalks/pedestrian crossing from the road
- Detect other vehicles and classify them (cars/trucks/bicycles...)
- ...

Where (geometric content)

- Estimating the free-space in front of our vehicle
- Finding out the distance to the closest obstacle
- Understanding the trajectories of the nearby vehicles
- ...

Scene understanding, what and where

How can we collect these cues? With **sensors**!

Laserscanner: it emits signals to measure the distance of objects over which they impact. Allows to reconstruct the 3D structure of an environment.
Alternatives: sonars, radars, …
Weaknesses: sparse measurements.

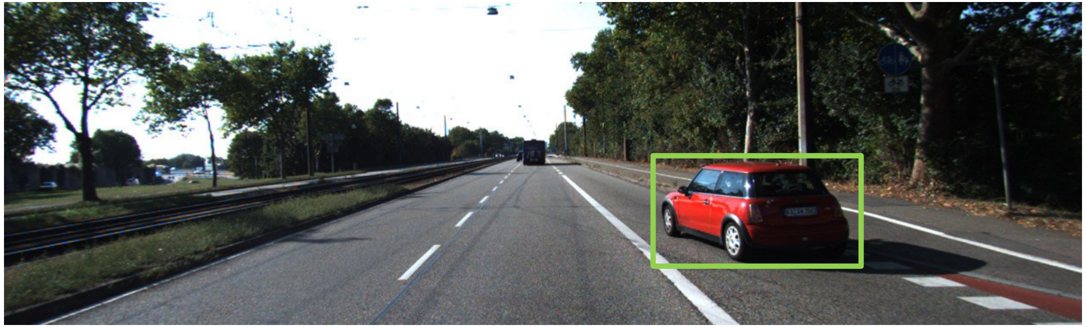
Credits: http://www.cvlibs.net/datasets/kitti/

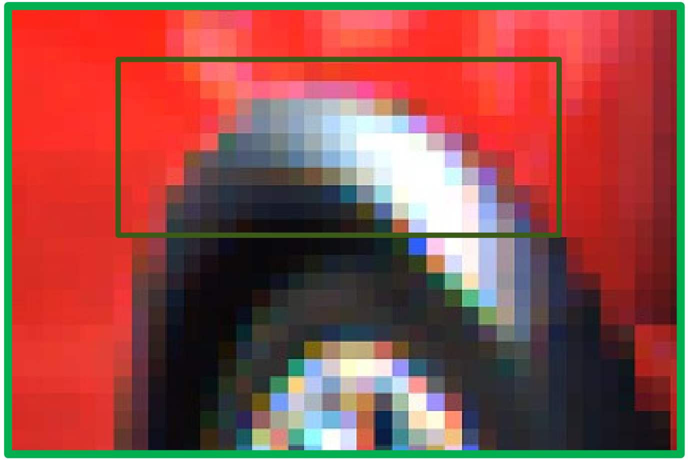
GPS: records the position of the vehicle on a global system. Allows to know movements, trajectories etc.
Alternatives: IMU, …
Weaknesses: can't tell anything about other vehicles.

Camera: collects images of the environment, from which we can extrapolate several information.
Literally, "A picture is worth a thousand words".
Weaknesses: we need **algorithms** to extract information!

Digital images and pixels

Let’s zoom into a digital image…





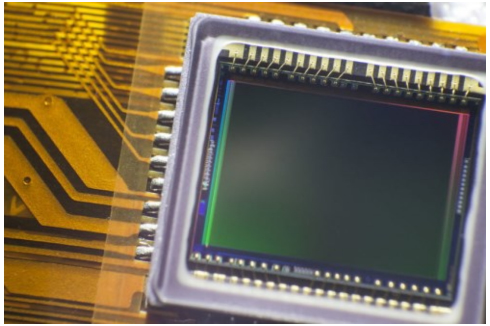
Digital images and pixels

Digital images are collected by means of **physical sensors**

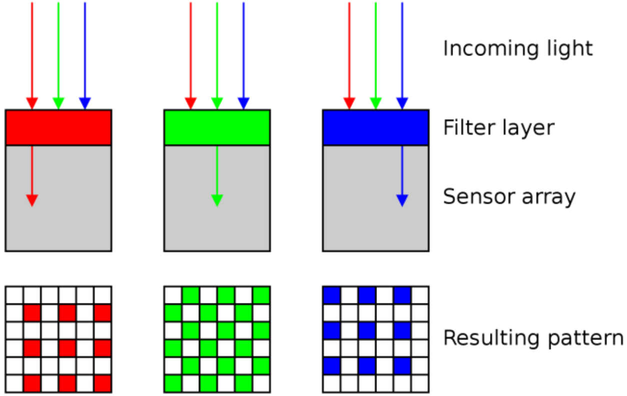
An imaging sensor consists of a grid of **photosensitive elements**

The resolution of a camera depends on the **size** of such a matrix

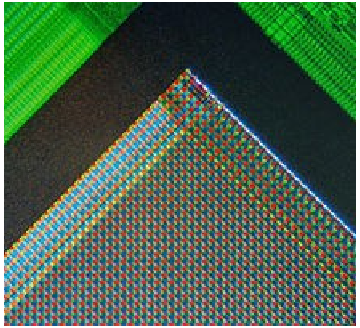
Color images are usually obtained through **different filters** and **interpolation**



Credits: https://www.digitaltrends.com/photography/quanta-image-sensor-low-light-camera/



Credits: https://en.wikipedia.org/wiki/Image_sensor



Credits: https://en.wikipedia.org/wiki/Image_sensor

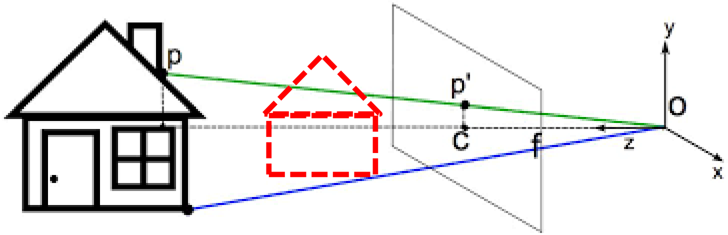
Digital images and the real world

When a scene is captured through a camera, we project 3D points into a 2D space (**image plane**)

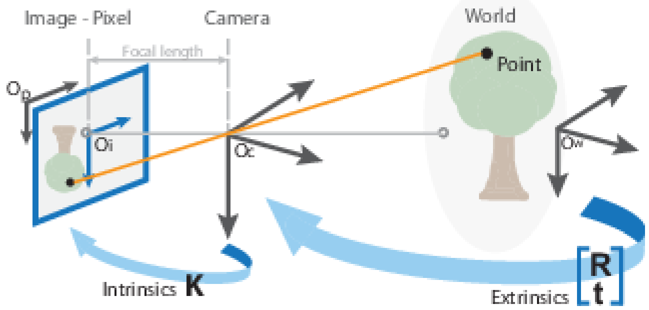
This mapping is **not** one-to-one: a theoretically **infinite** set of scenes can lead to the same image (for instance, black vs red houses on the right)

The appearence of the image we capture is consequence of sensor properties such as resolution, camera parameters (or **intrinsics**), lens, etc.

Moreover, it also depends on the camera position (**t**) and orientation (**R**) in the world (**extrinsics**)



Credits: https://www.mathworks.com/help/vision/ug/camera-calibration.html



Credits: https://www.mathworks.com/help/vision/ug/camera-calibration.html

Digital images and pixels

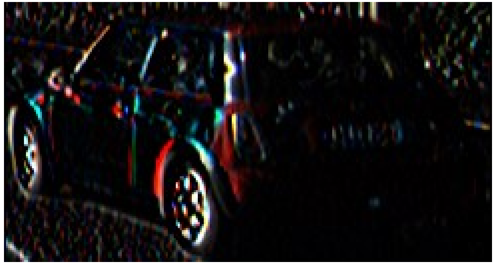
An image is a **2D matrix** (usually encoding 8bit values, from 0 to 255)

We can modify/replace/process the image content by means of operations over the matrix itself.
Some examples:

- **Color inversion**
  image[i,j] = 255 - image[i,j]

- **Convolution (correlation)**
  image[i,j] = sum(image[y,x]*kernel[y,x]
  for y in i-2,i+2 and x in j-2,j+2)

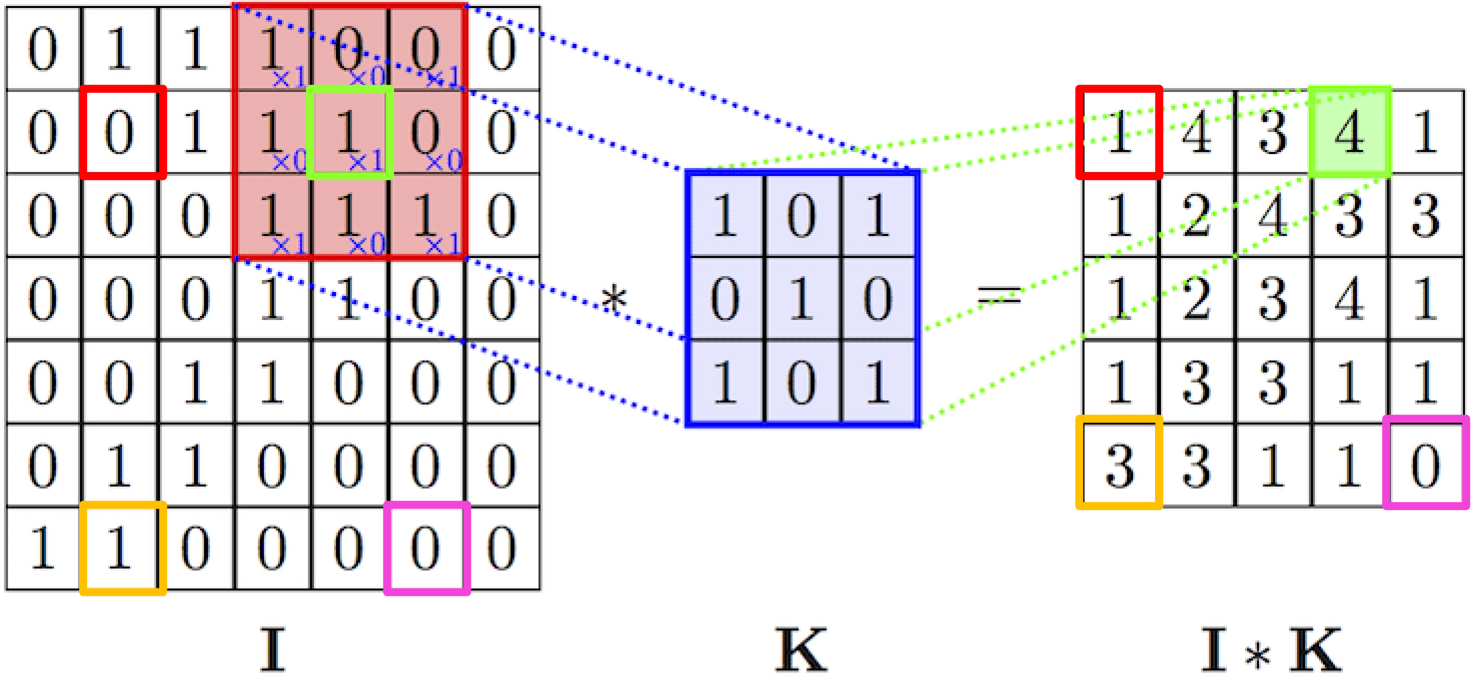
-1	0	+1
-1	0	+1
-1	0	+1



- **… more!**

Digital images and pixels

Convolution



Scene understanding, what and where

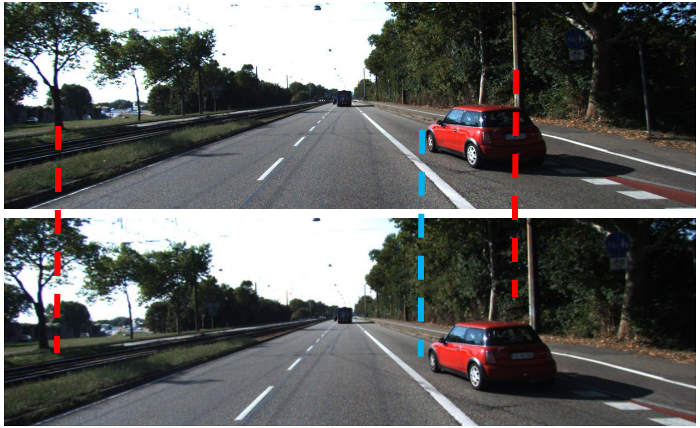
As humans, we can easily answer to many of these questions with **our eyes.**

What can we tell about this scene from **a single image**?

- We are on a road, **probably** driving forward
- We have a red car nearby (yet not directly in front of us), **probably** driving forward too
- The nearest vehicle (the red car) is **probably** 10-15 meters away
- ...

Scene understanding, what and where

As humans, we can easily answer to many of these questions with **our eyes.**

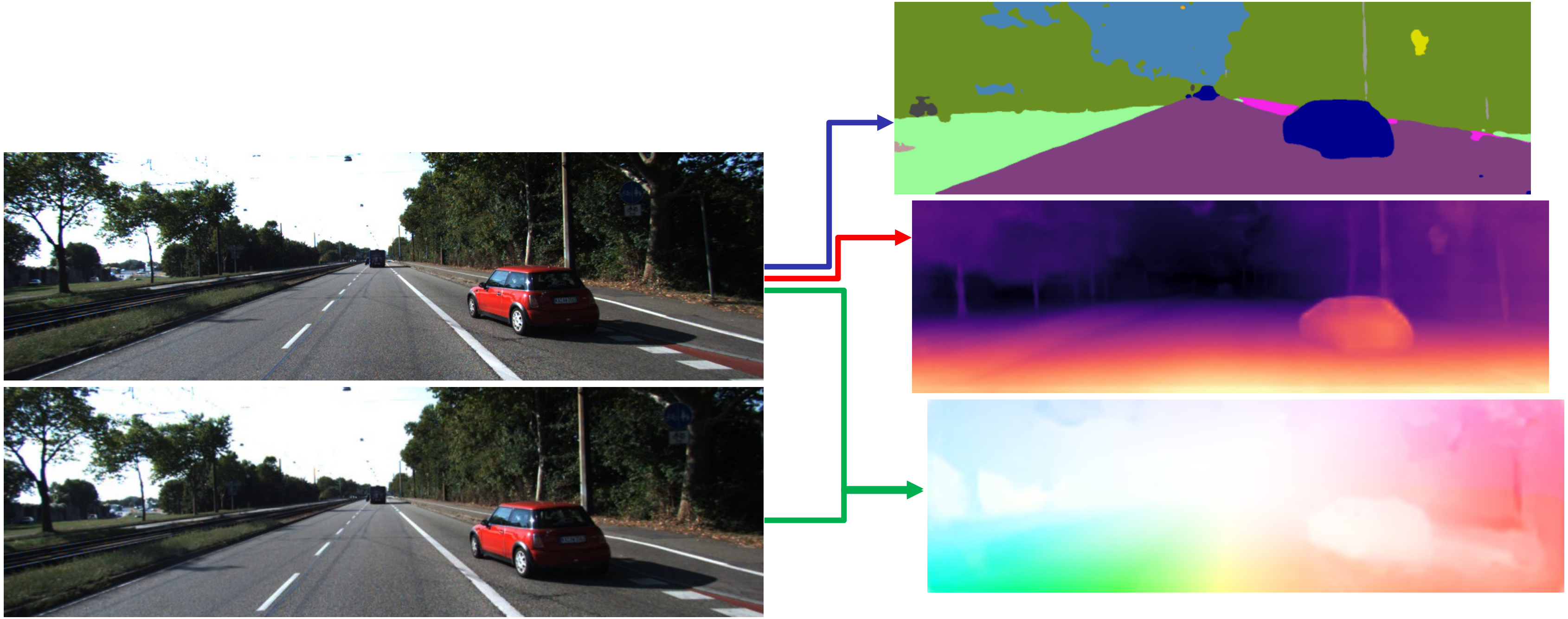


What can we tell about this scene from **two consecutive image**?

- We are definitely driving forward
- The red car is moving slower than us
- ...

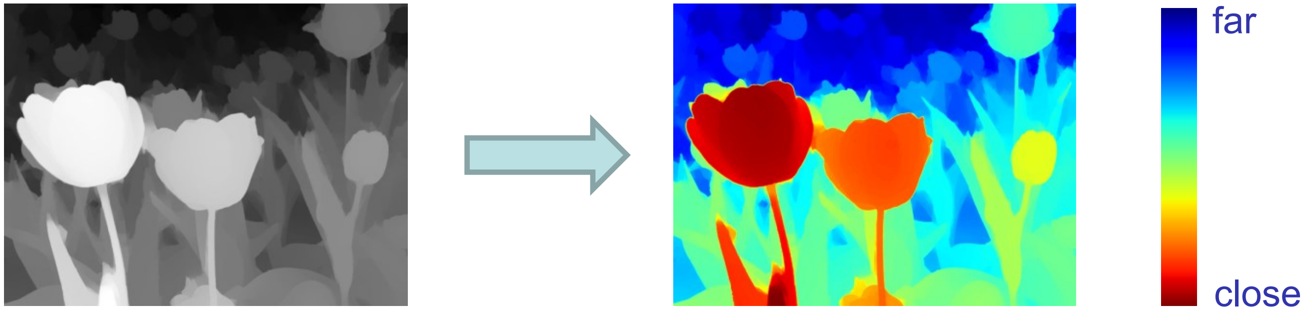
Scene understanding, what and where

As we do extract this knowledge from images, we can do the same by means of **computer vision** and **deep learning** methodologies



Colormaps

A colormap implements a mapping from a grayscale color space into an RGB(A) one. It is often used to enhance visual perception of some patterns.
Example: a **depth map** encoding in each pixel its distance from the camera



Colormaps are meaningful from a **qualitative** point of view (red pixels are closer than yellow pixels), while they are not meaningful of real depth values, unless the mapping is **reversible** (often, it is not).

While in this case we can easily perceive the relative distance between pixels **without a colormap**, in other cases (such as semantic segmentation or optical flow, see later) it becomes **much harder**

Semantic segmentation

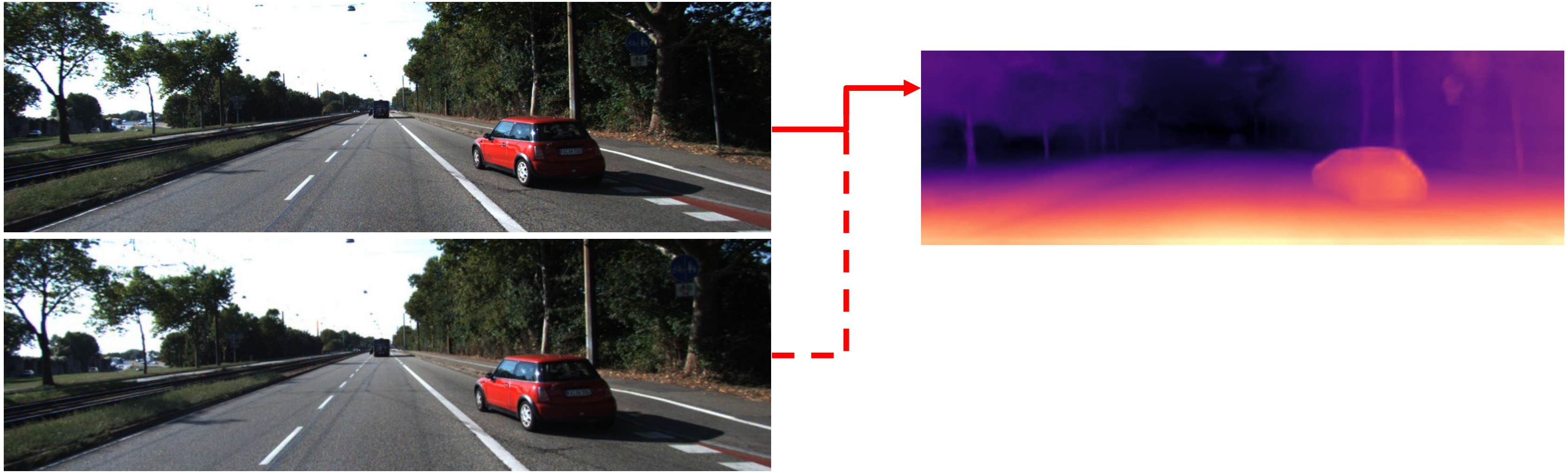
The category into which each pixel is classified is also known as **semantic class**. It distinguishes portions of the image belonging to different elements in the scene (road, cars, vegetation, etc.), representing a **pixel-level classification** of the image

Semantic segmentation

The category into which each pixel is classified is also known as **semantic class**. It distinguishes portions of the image belonging to different elements in the scene (road, cars, vegetation, etc.), representing a **pixel-level classification** of the image

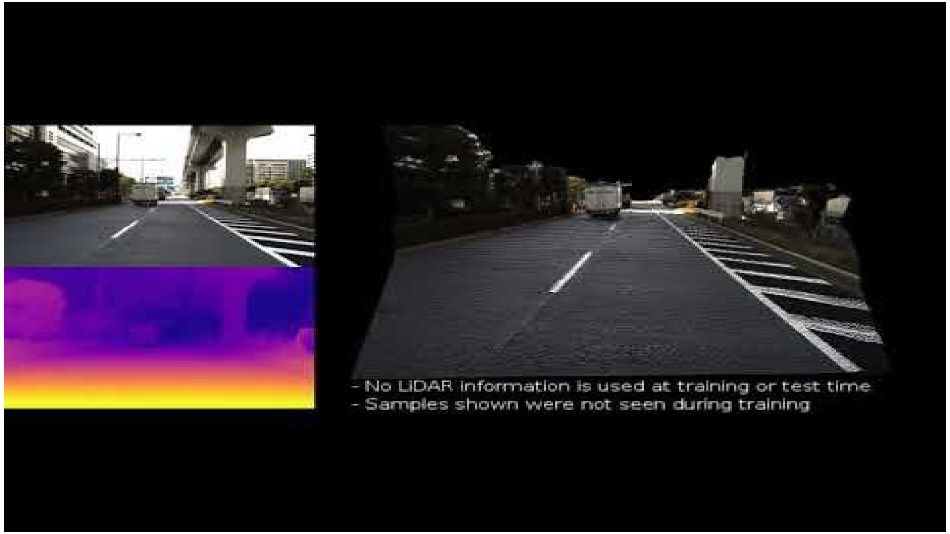
Depth

The distance between each point in the scene and the camera itself is also known as **depth**. It can be estimated either in **absolute** or **relative** scale, from **one** or **multiple** images.



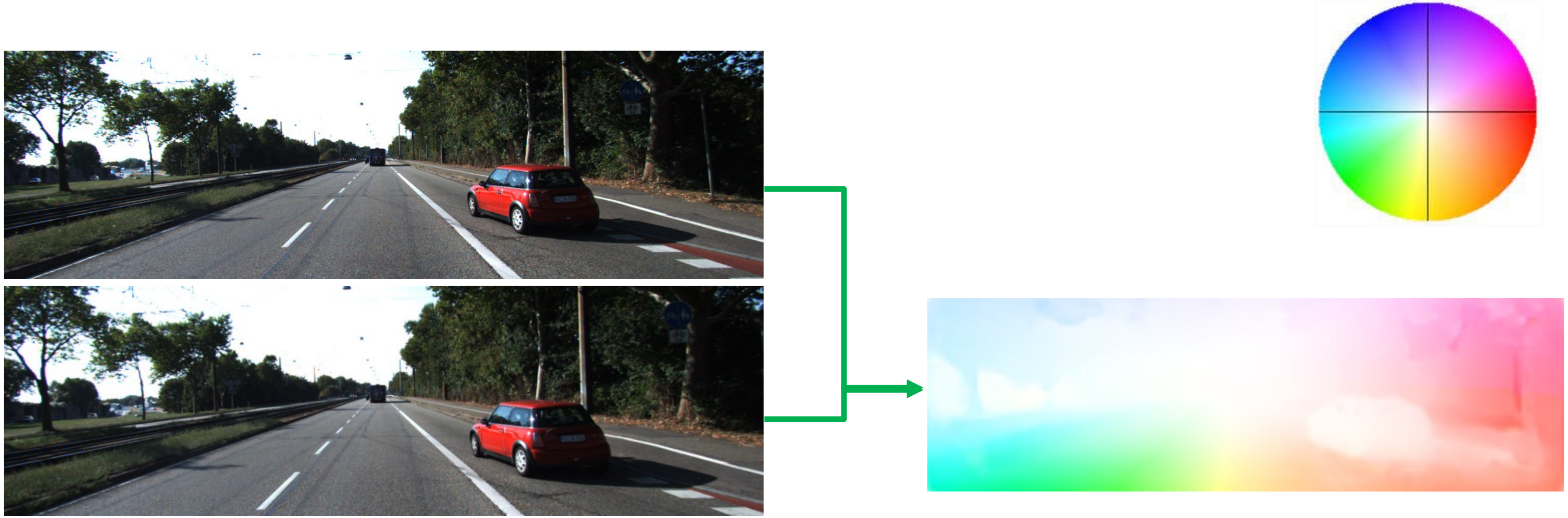
Depth

The distance between each point in the scene and the camera itself is also known as **depth**. It can be estimated either in **absolute** or **relative** scale, from **one** or **multiple** images.



Optical Flow

The motion between corresponding pixels in two, consecutive images is known as **optical flow**.
For each pixel in a frame, a 2D vector encodes the (x,y) translation which brings it to the new coordinates in the subsequent frame.

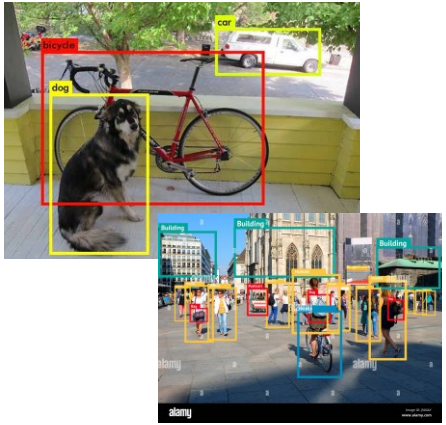


Optical Flow

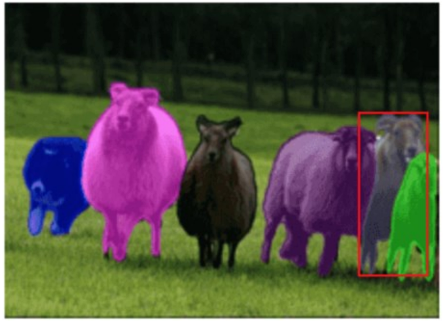
The motion between corresponding pixels in two, consecutive images is known as **optical flow**.
For each pixel in a frame, a 2D vector encodes the (x,y) translation which brings it to the new coordinates in the subsequent frame.

Scene understanding, more tasks!

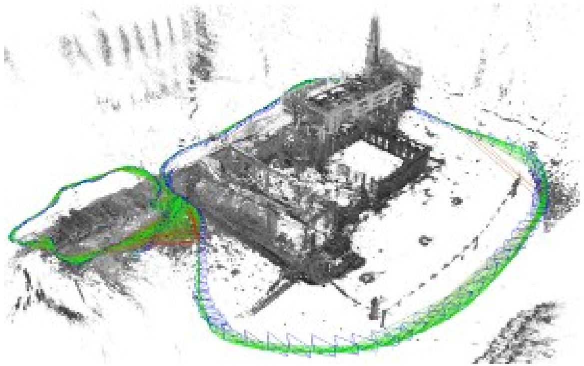
More and more tasks aimed at estimating information about "what" and "where" exists



Object detection



Instance segmentation



SLAM (Simultaneous localization and Mapping)

… and more!!! http://www.cvlibs.net/datasets/kitti/index.php

Scene understanding and applications

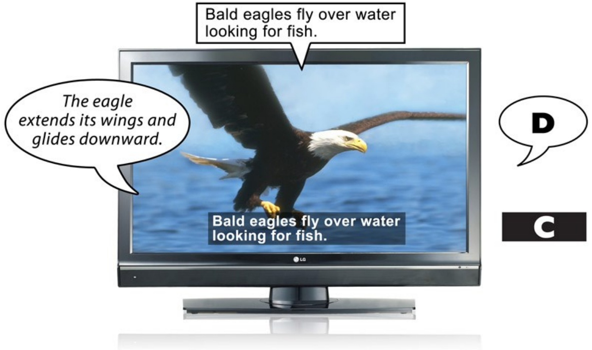
Autonomous driving is one of many applications we can implement starting from a basic understanding of the environment

Augmented reality

Credits: https://theconversation.com/what-is-augmented-reality-anyway-99827

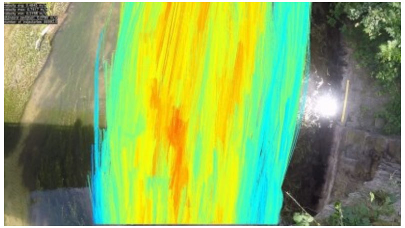
Credits: https://www.forbes.com/sites/forbestechcouncil/2021/12/10/the-state-of-augmented-reality/

Captioning



Credits: https://dcmp.org/learn/5-captioning-guidelines-for-the-dcmp

Monitoring



Analytical vs Data-Driven models

We can design a variety of **algorithms** to process images and extract cues such as those shown so far (and many more!). We can classify these algorithms in two, main families:

Analytical (or **model-based**):
Algorithms belonging to this category are designed from scratch by the developer, who is necessarily driven by **explicit knowledge** about the problem itself. This solution is usually feasible for problems founded on strong priors (e.g., geometry)

Data-driven (or **learning-based**):
Methods belonging to this family implicitly **learn** a solution to the problem from **data**, for instance by means of **neural networks**. This approach is often necessary for problems for which strong priors do not exist. Anyway, given enough **training data**, it can "solve" most computer vision problems

The best results are often obtained combining **the best of the two worlds**

Analytical vs Data-Driven models

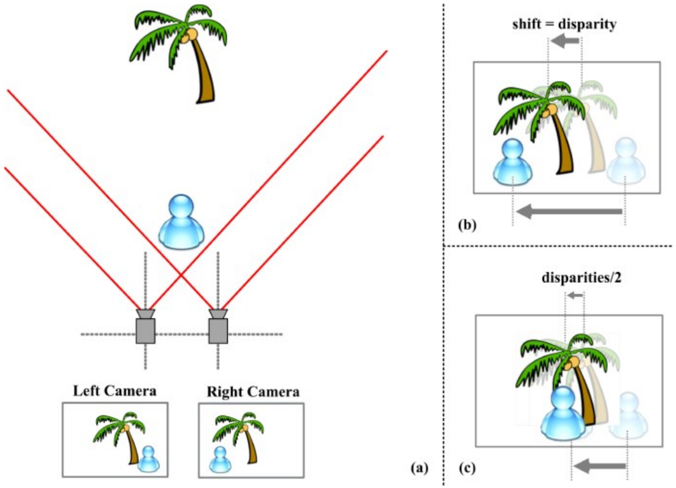
An example of problem which can be tackled by both families of approaches is depth estimation from **multiple images -** for instance two images, also known as **stereo matching** problem.

In this setup, depth is computed through **triangulation**, by finding the position of the very same pixel on the two images and its variation (**disparity**).

This can be carried out with "simple" hand-crafted algorithms, measuring the (dis)similarity between pixels across the images.

$$D_{left}(x,y) = \text{argmin}_d \, || \, c_{left}(x,y) - c_{right}(x-d,y) \, ||$$

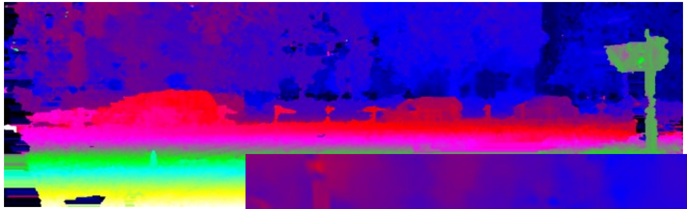
with c_k being the color of a pixel at coordinates (x,y) on image k



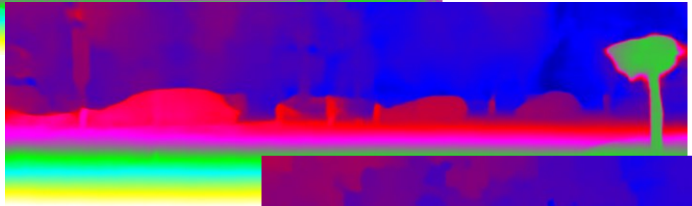
Credits: https://medium.com/mini-distill/pps-efficient-deep-learning-for-stereo-matching-de253fc411d4

Analytical vs Data-Driven models

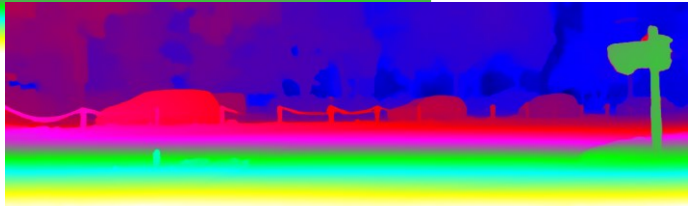
Until 2015, any stereo algorithm was **model-based** (the most famous: SGM).
In 2016, the very first **deep network** for stereo was proposed (DispNet).
Nowadays, deep stereo networks are standard solutions for this task, often **combining** model-based design strategies with deep learning.



2015



2016



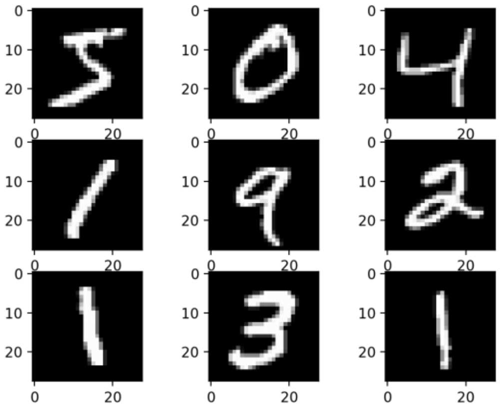
Now

Computer Vision and Machine Learning

These two worlds got in touch **several decades ago**

On of the pivotal tasks carried out on images by means of machine learning is **image classification**: given a single picture, we want to assign it a single **label** among N known labels, distinctive of the content shown in the picture itself

One of the most popular example: **MNIST dataset**

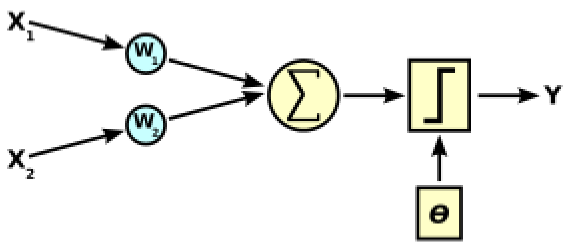


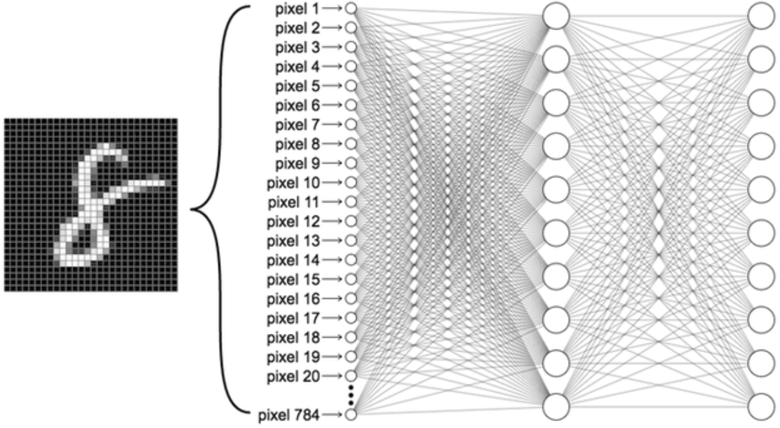
Neural Networks (NNs)

NNs and variants are among the most spread models in machine learning

Multi-Layer Perceptrons (MLPs) represented for a long time one of the most popular choices in several research areas. They consist of a set of multiple layers made of several **nodes**, each of them connected to any node from the previous layer (or **fully-connected**). Each connection defines a **weight**, learned by means of **back-propagation**

How can we process an image by means of a NN?





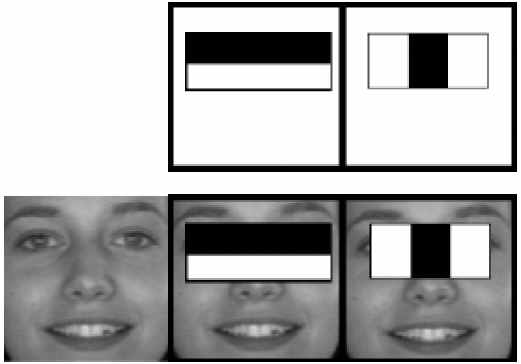
Features extraction

This naive approach has several disadvantages (**scaling**, **invariance**, etc.)

A very popular alternative in the early 2000s consisted of extracting some **features** from images

A feature represent a salient property of the image. Defining a good set of features for a specific task is extremely challenging (requires high expertise on the specific task)

Example: **face detection** (Harr features, used by Viola-Jones algorithm)

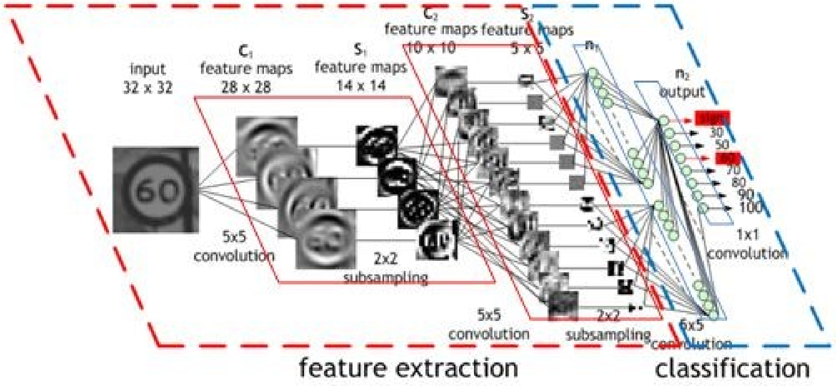


Convolutional Neural Networks (CNNs)

CNNs are the most popular deep learning framework in **computer vision**.
The use of **convolutional layers** make them perfectly suited for image processing.

Early works (late '80) using CNNs in vision aimed at solving **per-image classification** tasks (assigning a category to an entire image according to its content).

For this task, CNNs where usually made of two modules: a **feature extractor**, made of convolution layers, and a classifier, made of fully-connected layers used in standard NNs (MLPs)



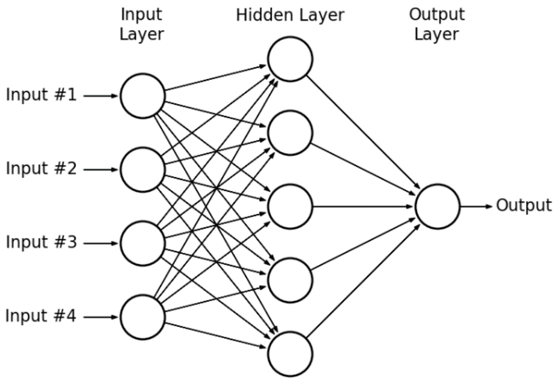
Credits: https://tech.everyeye.it/

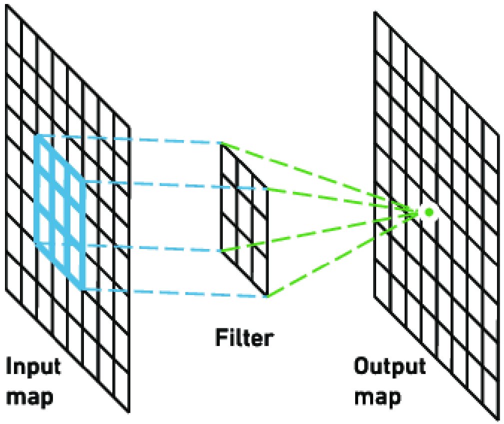
Convolutional Neural Networks (CNNs)

Convolutional layers are defined as a set of learnable weights organized into **kernels**.

The input image is processed through the layer by performing **convolution** (actually, **correlation**) between it and the kernel.

Differently from MLPs, convolutions result much more efficient and introduce some properties (**locality**, **translation invariance**, etc.)

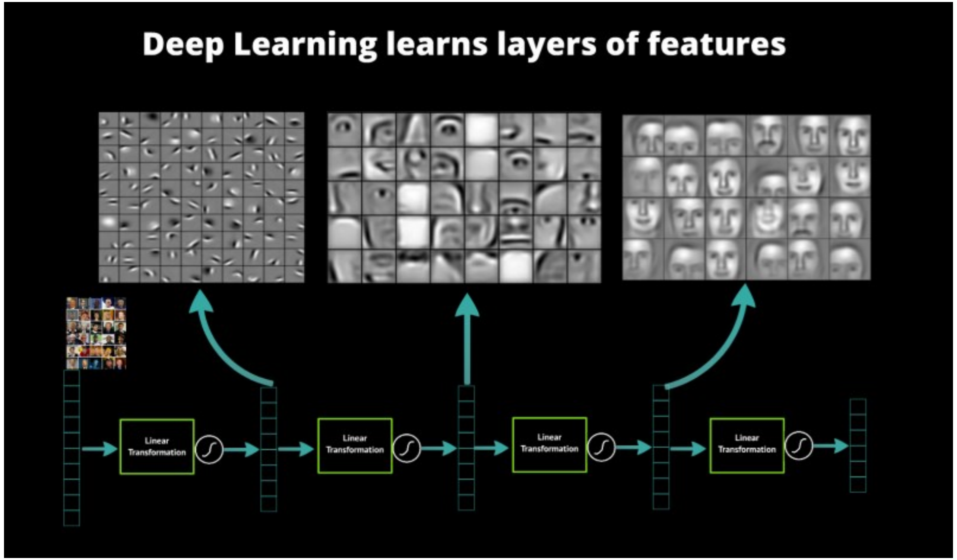




Convolutional Neural Networks (CNNs)

The features extractor learns a **hierarchy of features**, directly from data through back-propagation

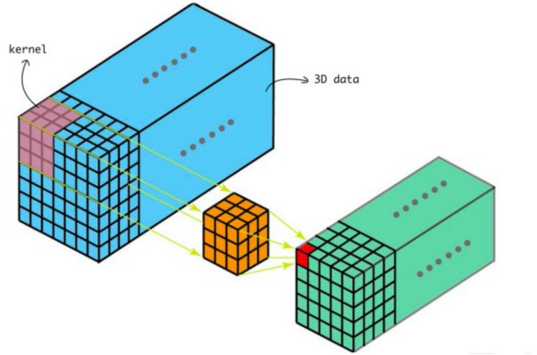
The earliest features extracted by the first convolutional layers are at low-level (edges, corners, etc.), while those extracted through deeper layers will gain **higher and higher** representation power



Credits: https://www.datasciencecentral.com/a-primer-on-deep-learning/

Convolutional Neural Networks (CNNs)

Convolutional layers can be generalized to **abritrary dimensions**, to deal with higher-dimensional structured data. For instance, we can implement a 3D CNN made of **3D convolutional layers**. This is quite common when dealing with stereo depth estimation (see next lectures…)



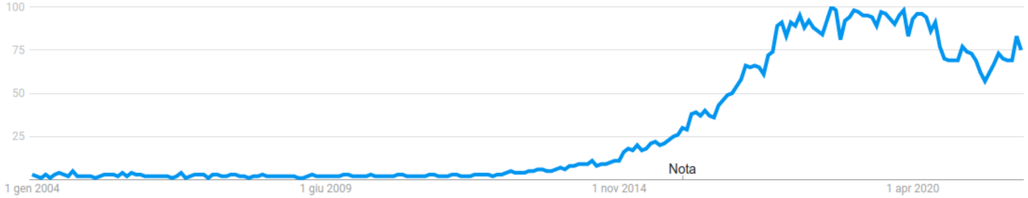
Credits: https://towardsdatascience.com/understanding-1d-and-3d-convolution-neural-network-keras-9d8f76e29610

We can even push it further and implement 4D CNNs, made of **4D convolutional layers**, for instance if we need to model spatio-temporal information (or when dealing with optical flow, see next lectures…).

Main problem: computational costs

Convolutional Neural Networks (CNNs)

The the great results achieved by CNNs has also impacted on the **research trends** in academia and industry, allowing to succesfully tackle tasks which were particularly hard to face before



"Deep Learning"

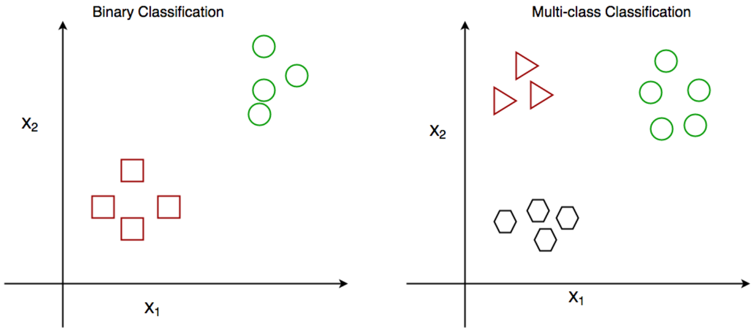
"Semantic Segmentation"

Convolutional Neural Networks (CNNs)

CNNs deals with tasks mentioned so far by either solving **classification** or **regression** problems.

In the case of **classification** problems, the network aims at assigning a **class label** to a specific input data point. The number of classes is usually **finite** and **defined**.

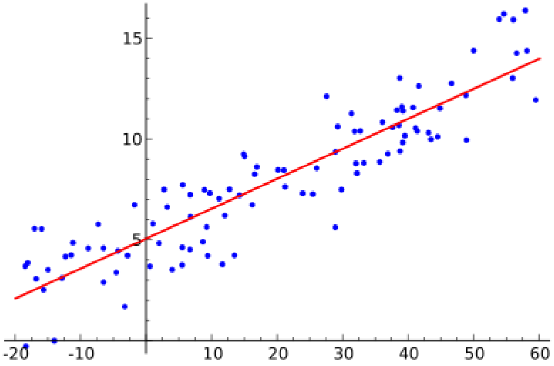
The CNN output consists of a vector of N values (for N classes) and is interpreted as a **probability distribution** of the input point to belong to any of the N classes. The class assignited by the CNN is the one corresponding to the output having highest value



Credits: https://www.geeksforgeeks.org/ml-classification-vs-regression/

Convolutional Neural Networks (CNNs)

In the case of **regression** problems, the network aims at inferring continuous values to a specific input data point.



Credits: https://en.wikipedia.org/wiki/Linear_regression

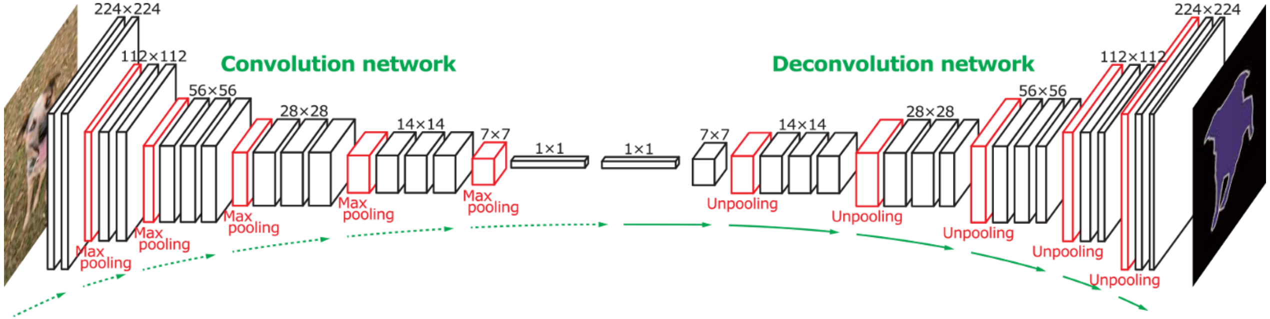
Considering the three tasks so far, we can generally distinguish them into **classification** (semantic segmentation, in which we usually know the exact number of classes we aim at recognizing) and **regression** (depth and optical flow, for which we can estimate continuous, floating point values) problems as well.

We might also tackle depth estimation as a **classification** problem as well (for instance, by defining a set of depth bins and classifying any pixel in a single image to its proper depth bin)

Convolutional Neural Networks (CNNs)

Modern CNNs often deal with **dense prediction tasks**, for which a different output is predicted for **any pixel** in the input image.

This is possible by designing a CNN to be **fully convolutional**, made only of **convolution** (or **devoncolution**) layers. This allows to keep a **spatial structure** of the output prediction.



Credits: https://www.medium.com

The same principle applyies to higher-dimensional CNNs (3D, 4D, etc.)
In the very last years (2020-today), other architectures are becoming popular, such as **Vision Transformers**, **MLP-Mixer**, etc.

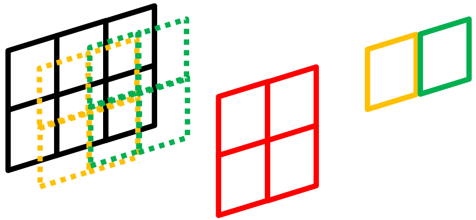
Convolutional Neural Networks (CNNs)

To implement a fully-convolutional network, we either need to:

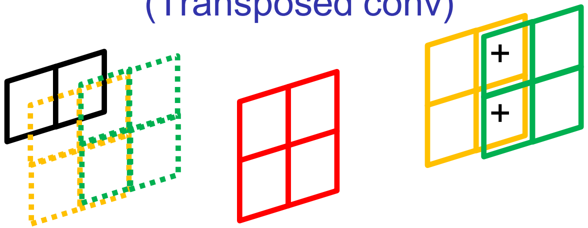
- maintain the original input resolution (**unfeasible**, most of the times)
- implement a layer to restore the initial resolution

Transposed convolution (or deconv layer): reverting the shape of a conv layer

Conv Layer



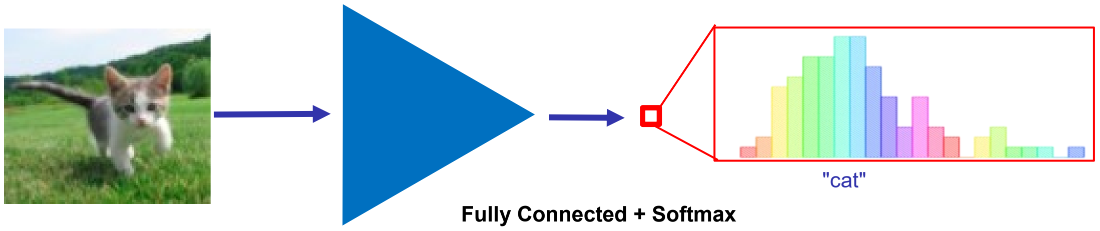
Deconv Layer
(Transposed conv)



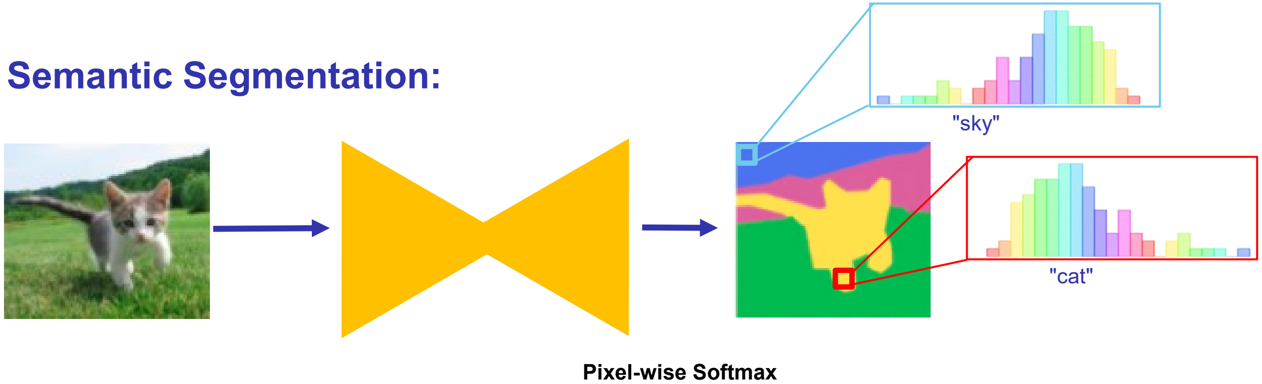
Alternatives: upsampling (nearest/bilinear) + conv layer

Convolutional NN vs Fully-Convolutional NN

Classification:

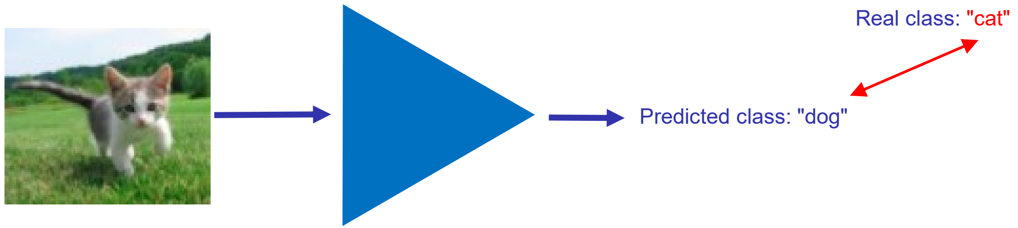


Semantic Segmentation:



Supervision paradigms

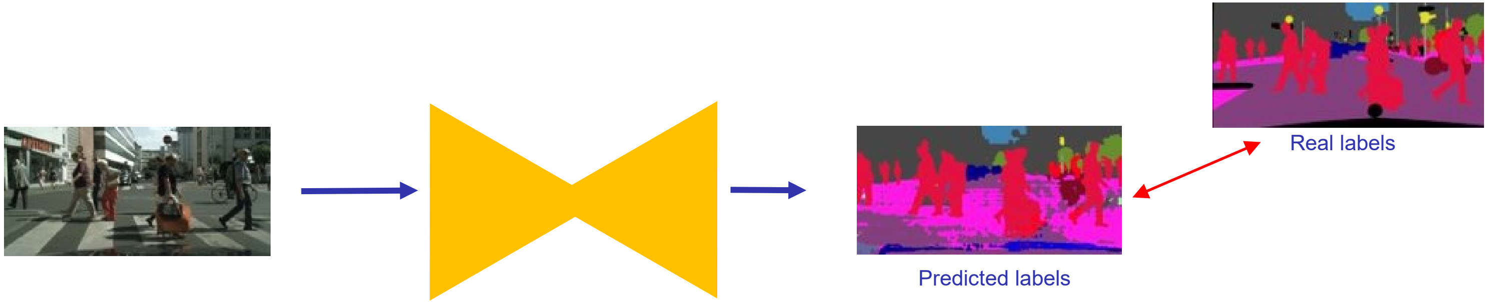
CNNs (and, in general, machine learning frameworks) are **data-driven** approaches. Specifically, the parameters defining the behavior of a CNN are **learned** through optimization over a set of data samples



According to the information provided during this learning phase, also referred to as **training phase**, we identify different **supervision paradigms**, varying in terms of effectiveness and easiness of deployment

Supervision paradigms

Supervised learning: for any training sample, the correct prediction the network should give is provided during training. In case of classification, for each image a single, correct label is required.
In case of dense prediction tasks, a single label for **any pixel** is required!



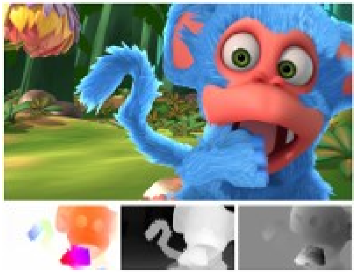
This paradigms allows for training CNNs at their best.
However, a CNN usually require **thousands** of image samples for training, thus manually annotating any pixels in thousands images results extremely costly.

For other tasks such as depth estimation and optical flow manual annotation is **unthinkable** and **additional sensors** are necessary (for depth estimation, LIDAR sensors are often used)

Supervision paradigms

How can we deal with data annotation in a cheap and scalable way?

We can exploit **computer graphics** to render countless images, together with per-pixel labels **for free**

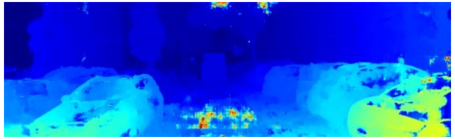


However, although realistic, synthetic images are very different from real ones (in terms of noise, lights, shadows, etc.). Thus, CNNs trained solely on synthetic images often suffer in **real environments**, because of the **domain shift** between the two.

Supervision paradigms

Some examples:

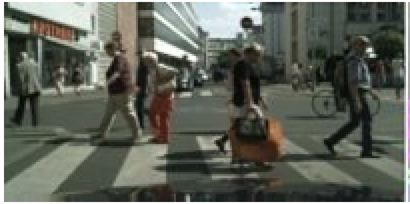
Depth estimation (stereo)

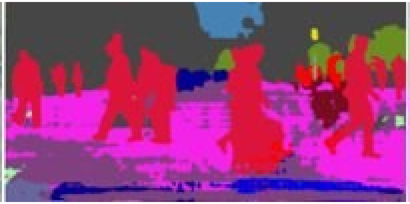


Trained on synthetic images

Trained on real images

Semantic segmentation





Trained on synthetic images

Trained on real images

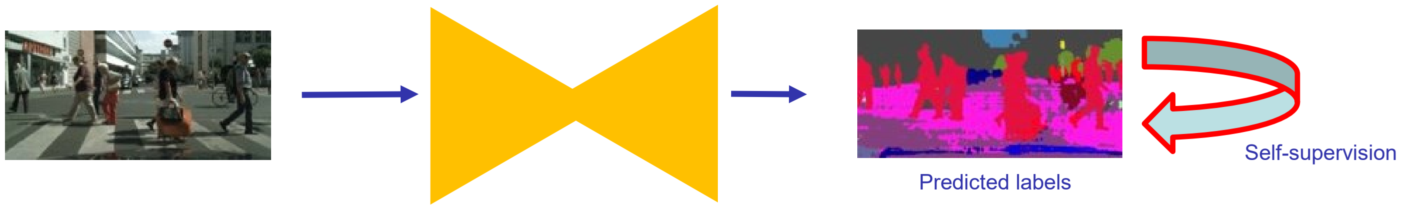
Solutions: strong data augmentation during training, supervised fine-tuning on a few real data, … , **unsupervised learning**

Supervision paradigms

Unsupervised learning: the correct prediction the network should give is **not** provided during training (and, sometimes, is even **unknown** to the developer itself!)

For dense prediction tasks, this allows to get rid off per-pixels annotation, by demanding supervision to mechanisms **specifically designed** for the task itself.

These mechanisms are known as **self-supervision** mechanisms and varies according to the specific task (e.g., can leverage **geometry** when the task itself involve some strong geometric constraints)

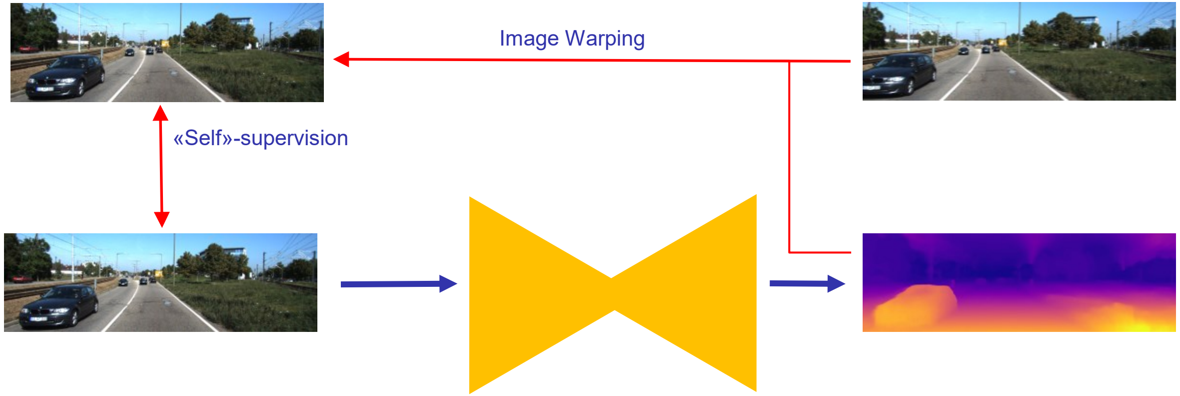


Supervision paradigms

Example: depth estimation from a single image.

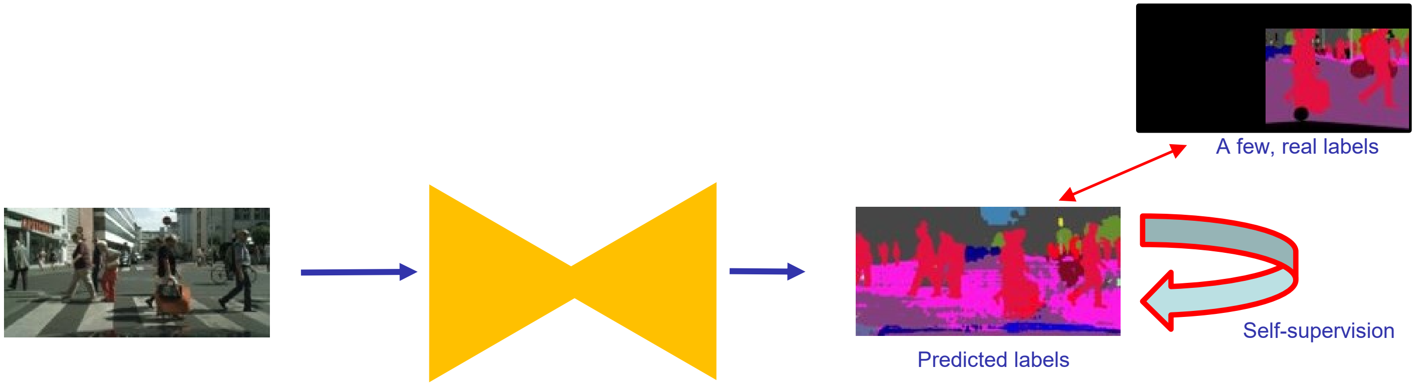
Hypothesis: given two (or more) images and their cameras relative positions, we can **project pixels** across the two view if we know their depth. The same principle is exploited by algorithms estimating depth from multiple images (as we have seen for **stereo matching**).

We can train a CNN to estimate depth from a single image, by i) replacing **depth labels** with a second image framing the same scene and ii) minimizing the projection error enabled by estimated depth



Supervision paradigms

Semi-supervised learning: the correct prediction the network should give is provided for **a subset** of the training data, while for the remaining samples **unsupervised learning** is exploited.



An example of a semi-supervised framework could be a network jointly estimating **depth** and **semantics**, exploiting **geometry** for self-supervising the depth predictions and **manually annotated** labels for semantic predictions.

What we will see next

This course will focus on

- **Depth estimation:**
  different approaches to estimate depth from images. Single image depth estimation (data-driven), depth from two or more images (analytical+data-driven), self-supervised techniques for depth estimation

- **Optical Flow estimation:**
  different approaches to estimate pixels motion, sparse versus dense. Optical flow and relationship with 3D geometry. Analytical methods vs data-driven. Self-supervised techniques for optical flow estimation.

- **Semantic segmentation:**
  evolution of semantic segmentation, network architectures. Domain shift in semantic segmentation, unsupervised adaptation approaches.