# LESSON 23

Image classification
Intelligent vision System,
Object detection,
Segmentation and Recognition,
Machine Learning Hardware




# Outline

- Image classification and Intelligent Vision Systems
  - Object detection, Segmentation, Recognition
  - Classical, Deep and Hybrid solutions
  - Use cases
- Hardware for Machine learning



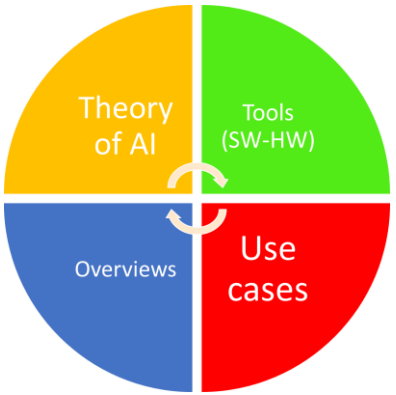


Dog

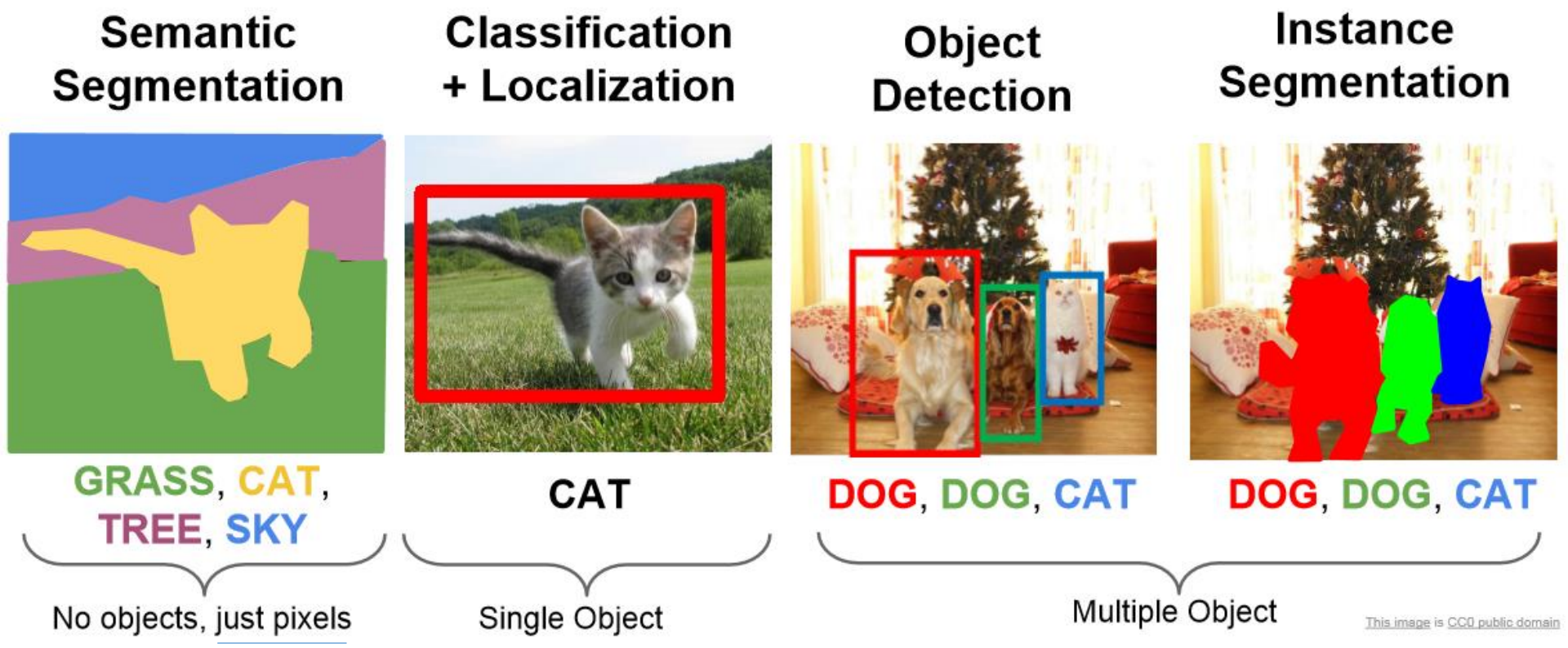

Cat

# THEORY

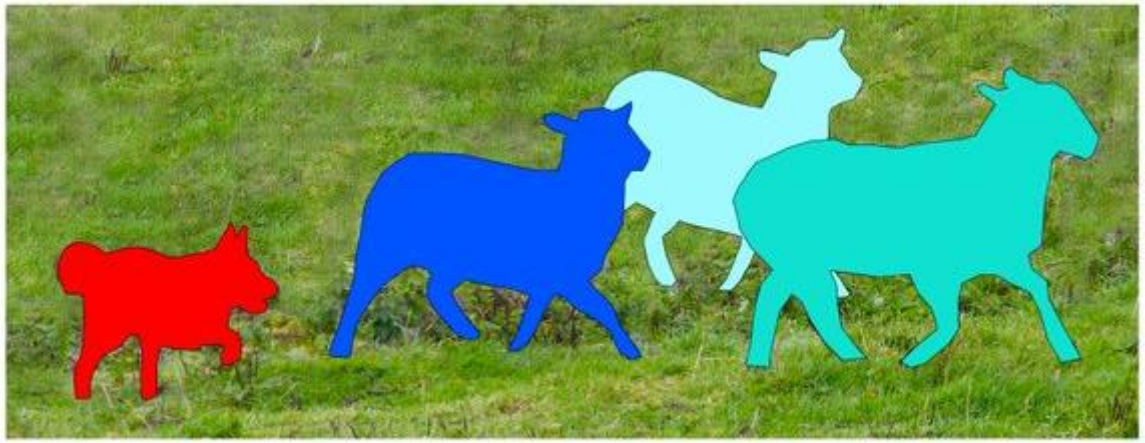
# Image classification



# Intelligent Vision tasks

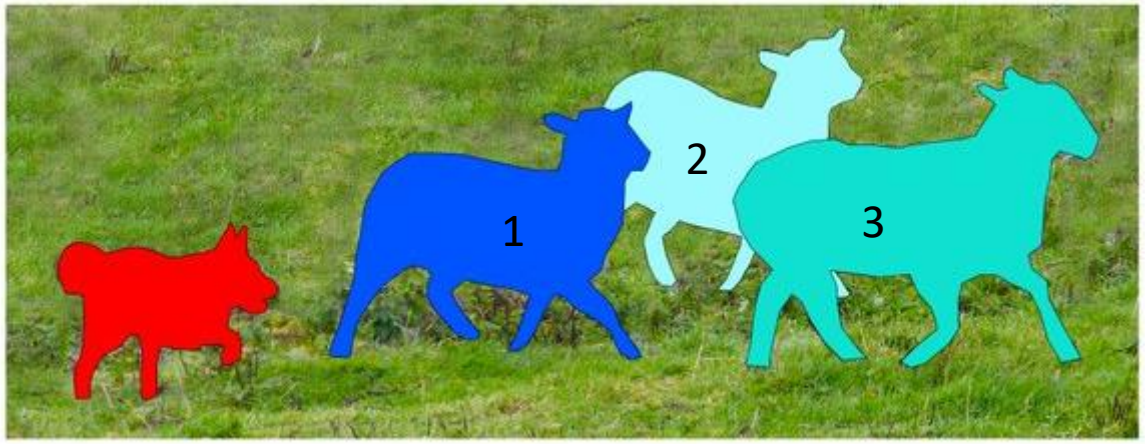


# A quick test 1 …



What is this Intelligent vision task?

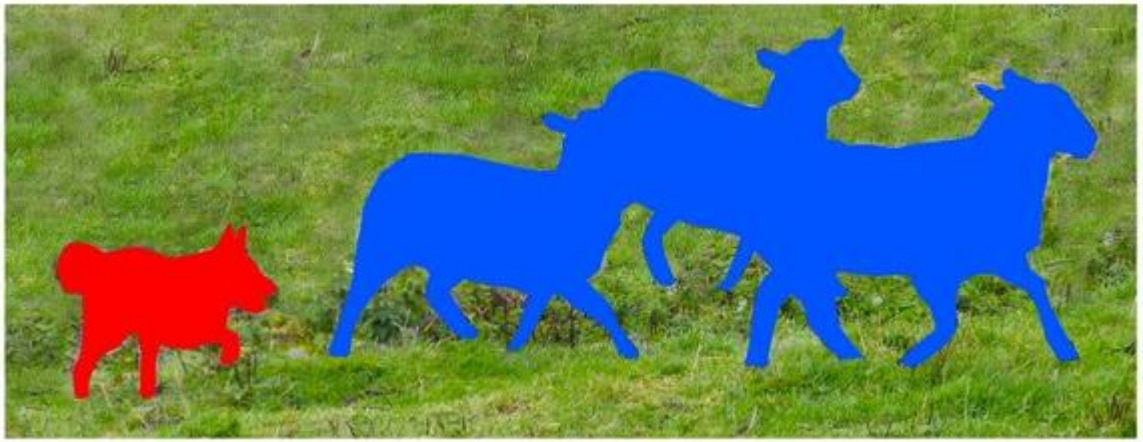
# A quick test 1




What is this Intelligent vision task?

ISTANCE SEGMENTATION!

# A quick test 2 …



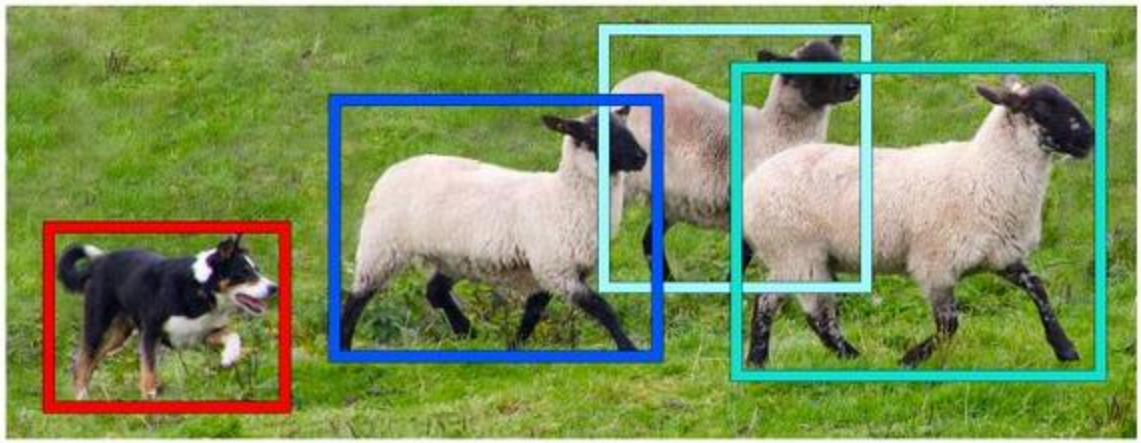
What is this Intelligent vision task?

# A quick test 2 …


What is this Intelligent vision task?

SEMANTIC SEGMENTATION!

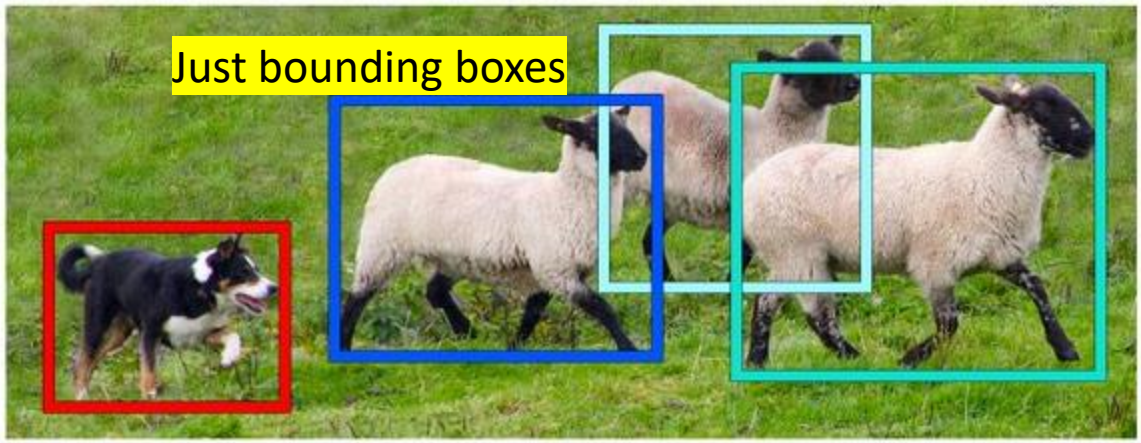
# A quick test 3 …



What is this Intelligent vision task?

# A quick test 3 …

We have just the so-called bounding box




What is this Intelligent vision task?

OBJECT DETECTION

# A quick test 4 …


What Intelligent vision task does it correspond?

# A quick test 4 …

We have just one class in output = Sheep


What is this Intelligent vision task?

IMAGE CLASSIFICATION

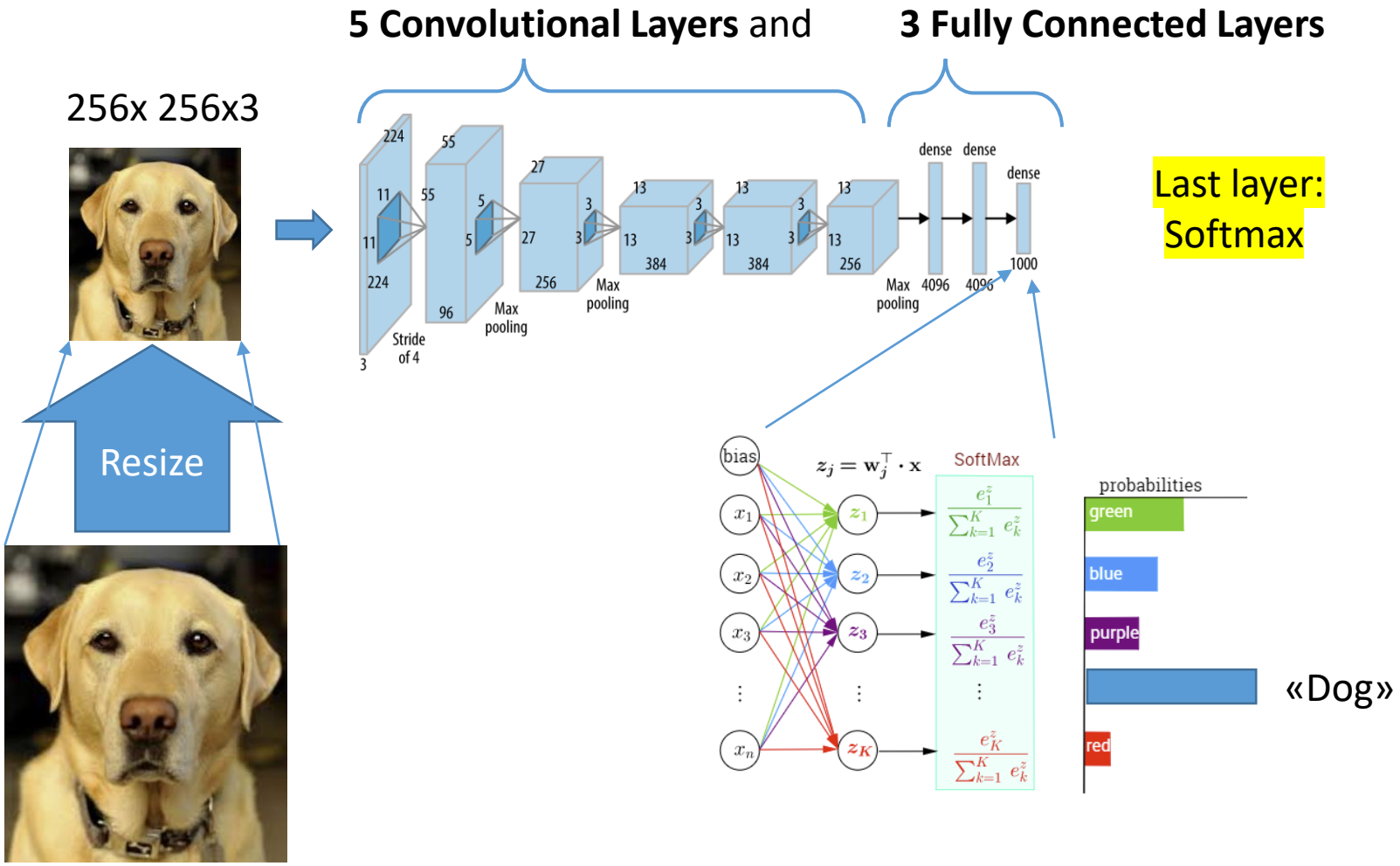
# Direct image classification

Image classification

«Dog»

No more information about the object

# Direct image classification: Example with AlexNet



**ImageNet** Large-Scale Visual Recognition Challenge
(>14Mimages, 1000 different classes)

# Image classification

Good for training
and fine tuning
(after selection)

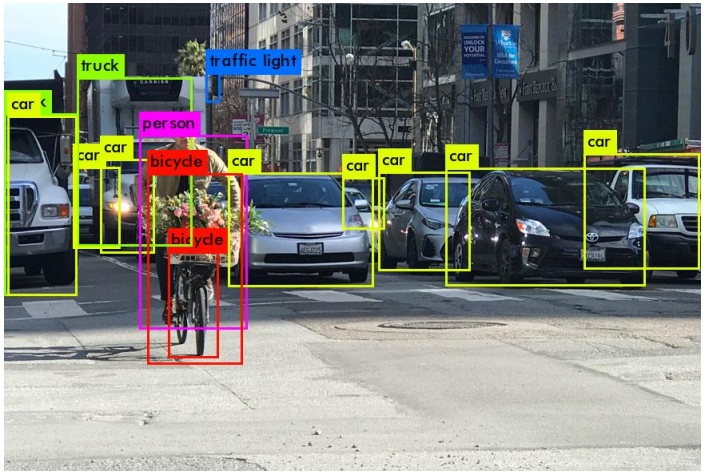
EASY SET

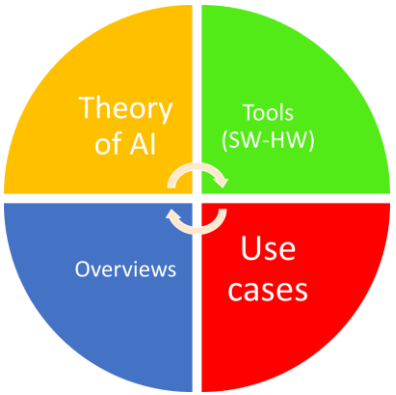

HARD SET


Good for Google…
not for an industrial
application…
But good to train to
extract feature!



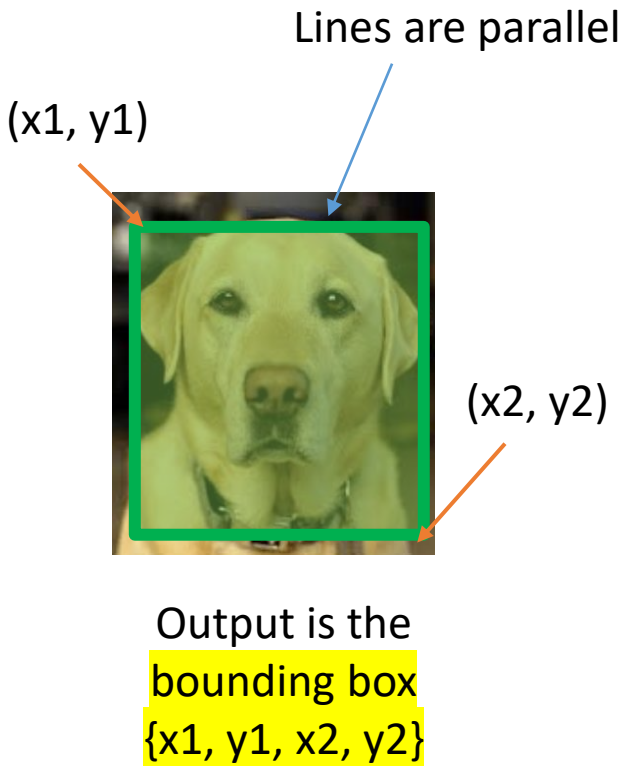

# THEORY
# Object detection



# Object detection

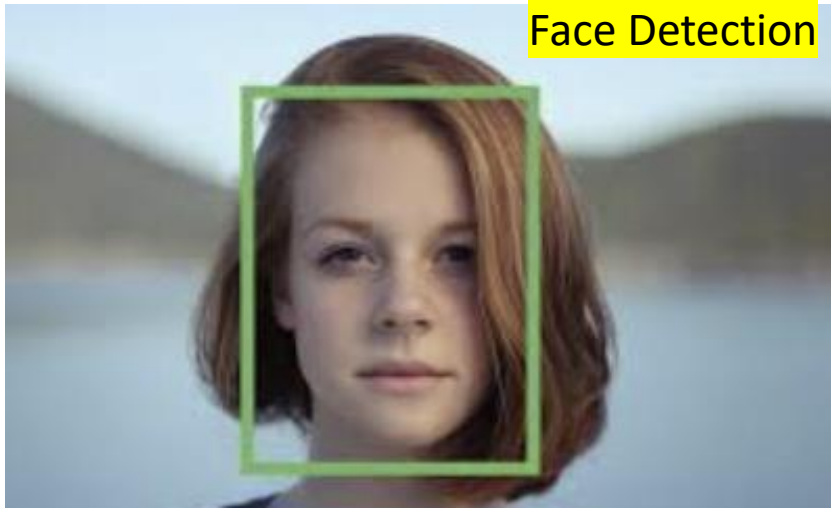
Object detection

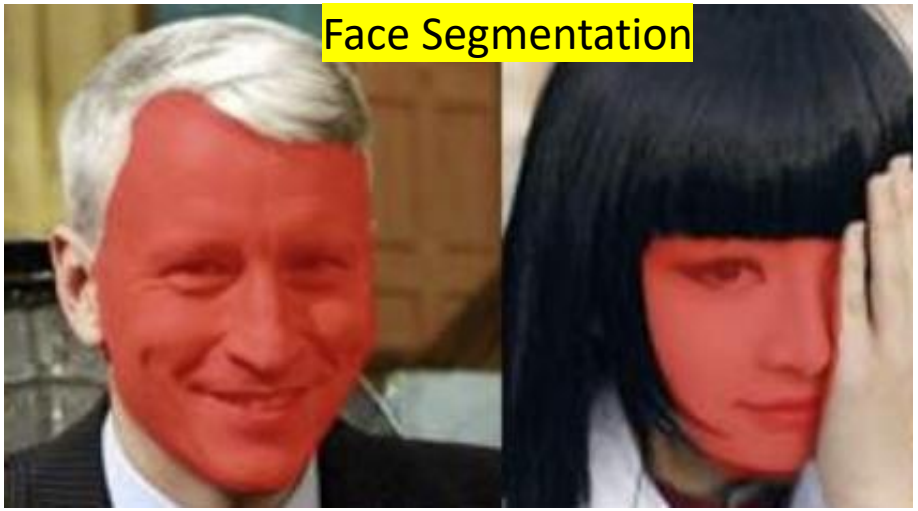


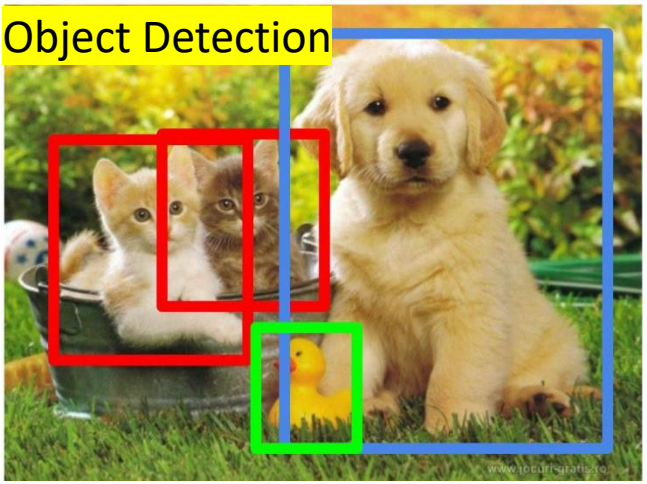

Output is the bounding box {x1, y1, x2, y2}

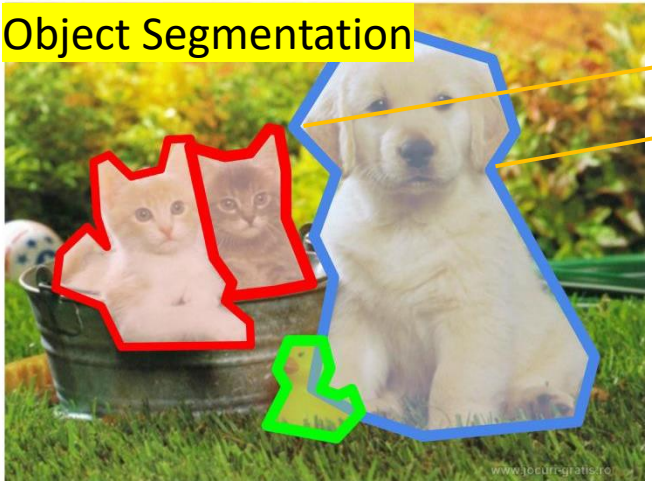
It's almost impossible to get measurements

# Detection in not segmentation!



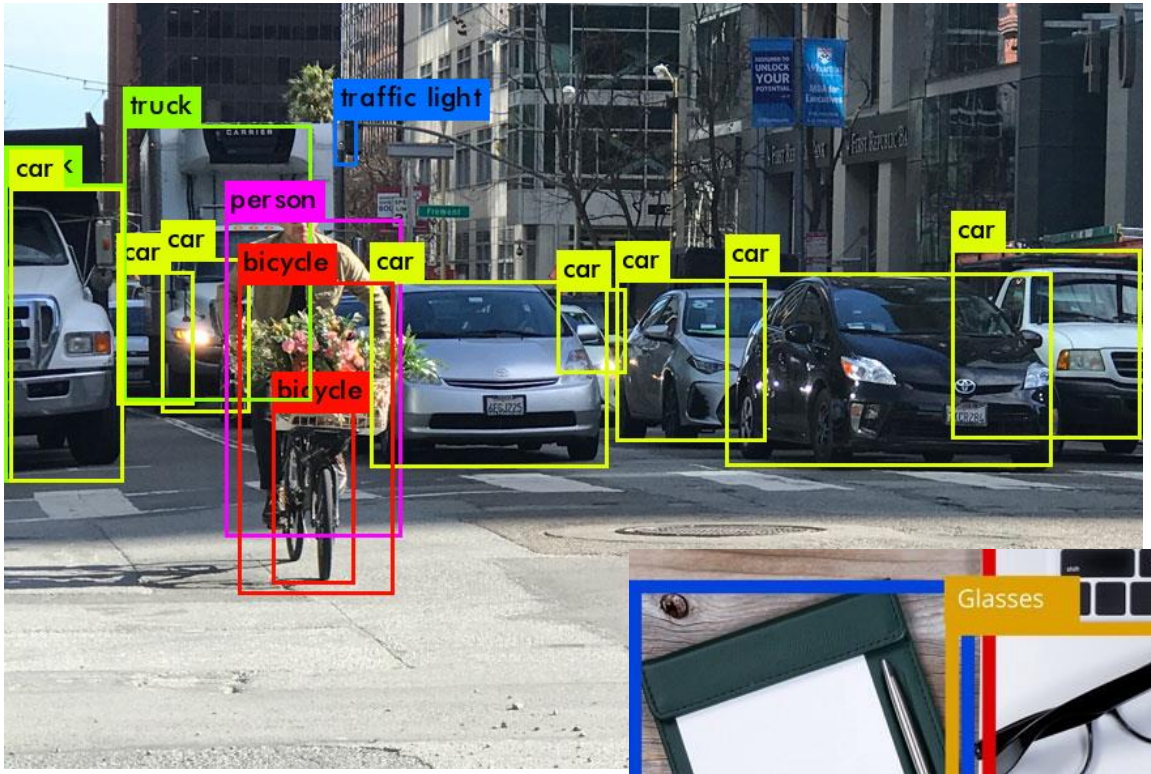




Real measures of the objects!

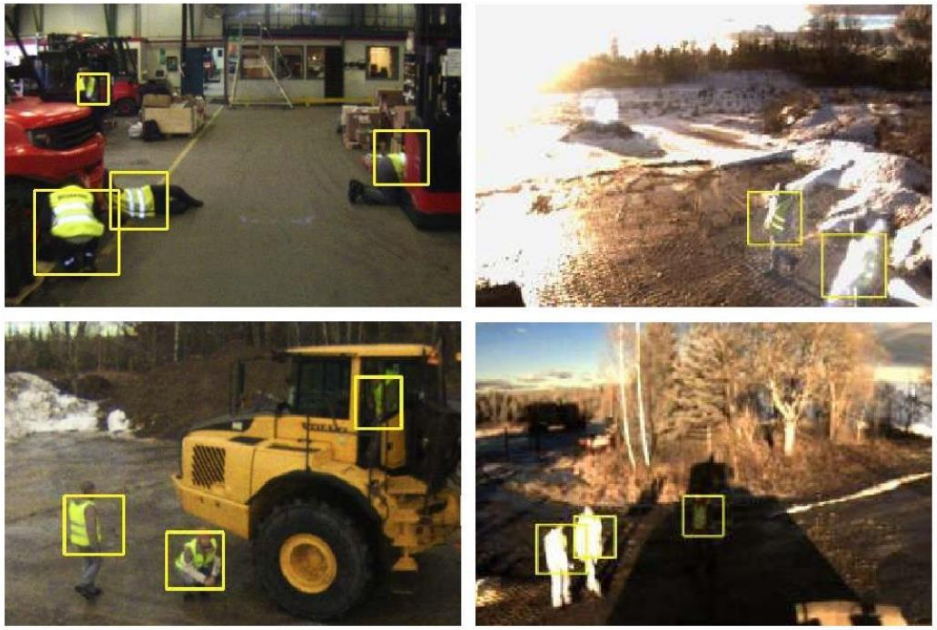
# Multiple object detection




Not an easy task
in complex scenes

Overlapping objects

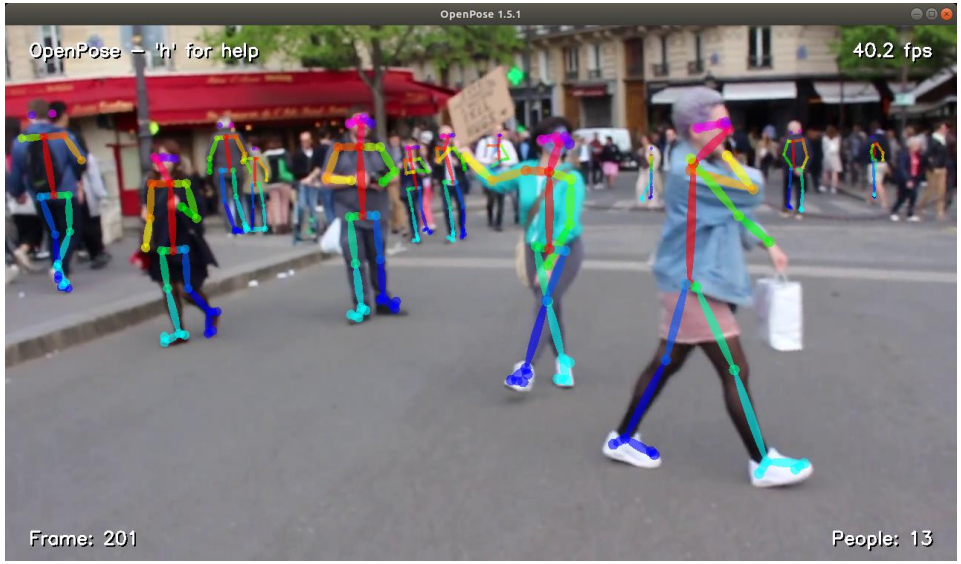
# Detection
# Example: person detection in images

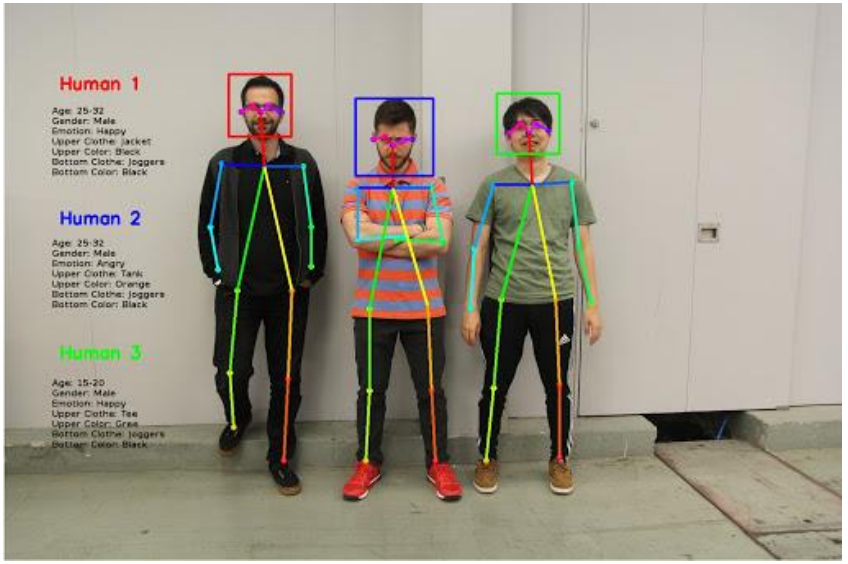


Industrial sites

Survelliance

# Articulated body pose estimation: OpenPose



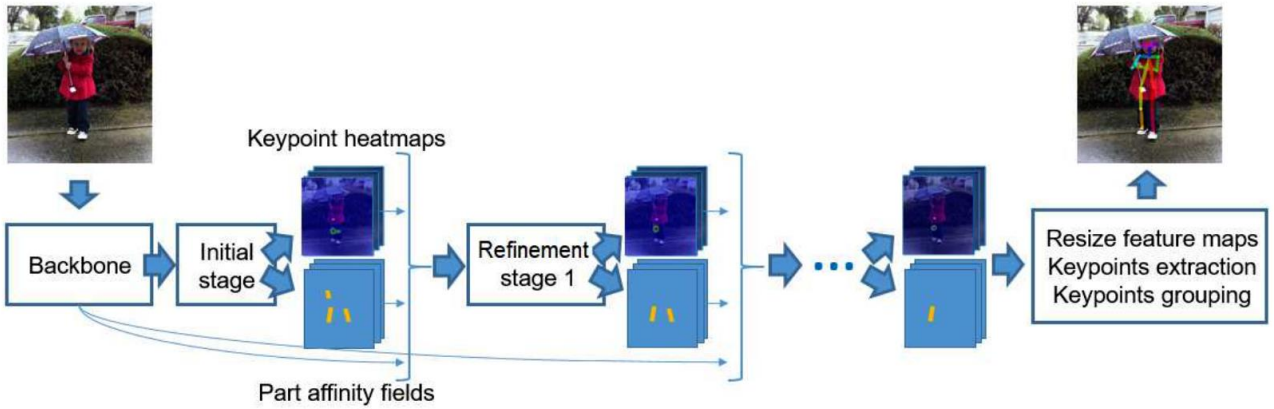
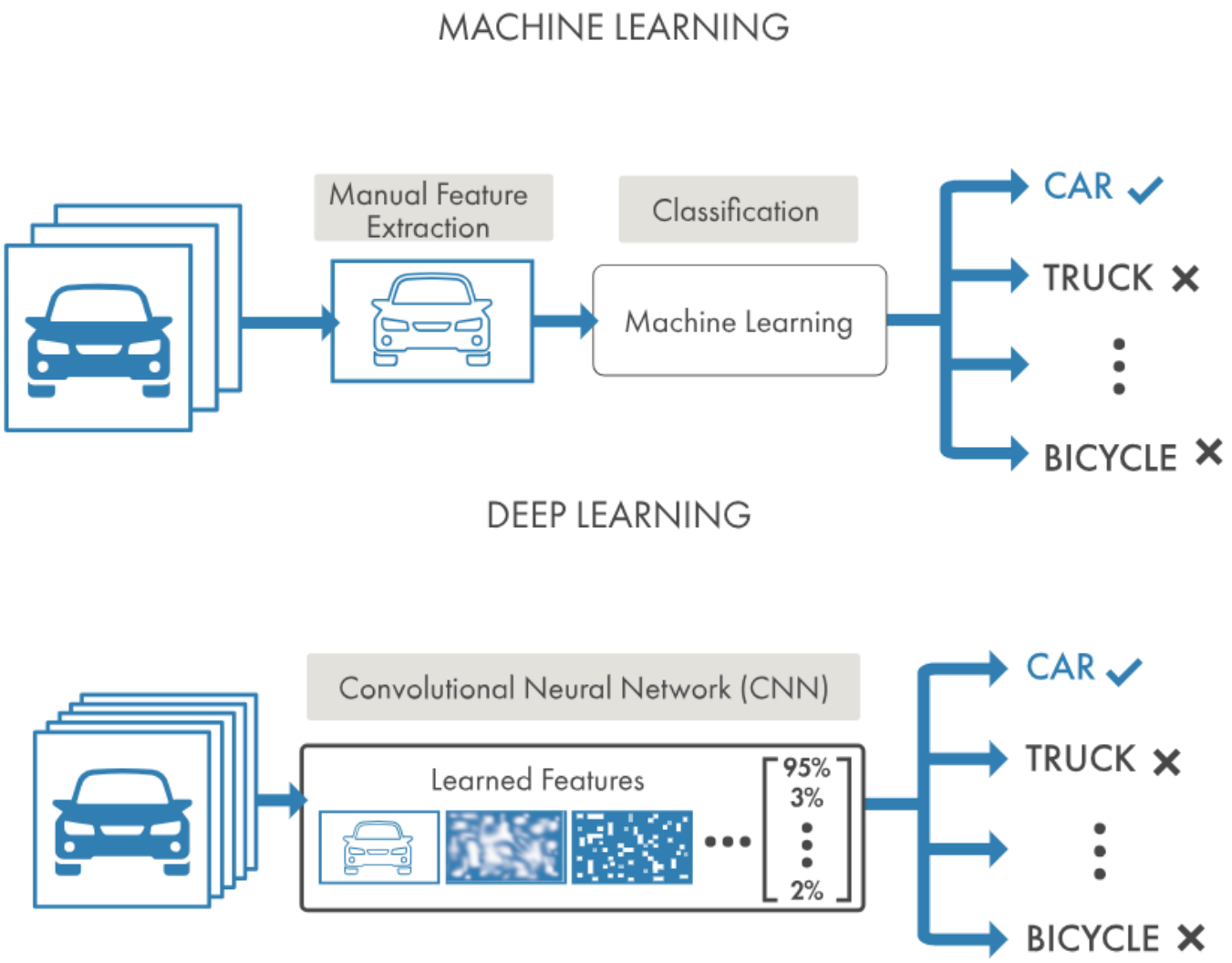




Figure 1: OpenPose pipeline.

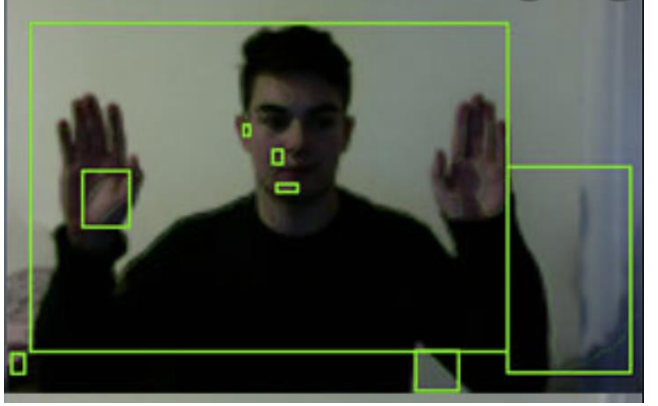
https://github.com/CMU-Perceptual-Computing-Lab/openpose

# Object Classification/Detection

## Classical Machine Learning V.S. Deep Learning

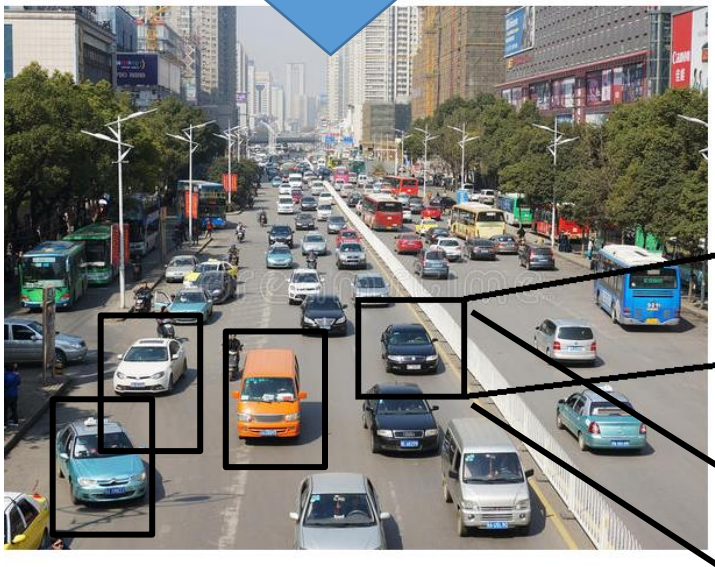


# Hybrid solutions

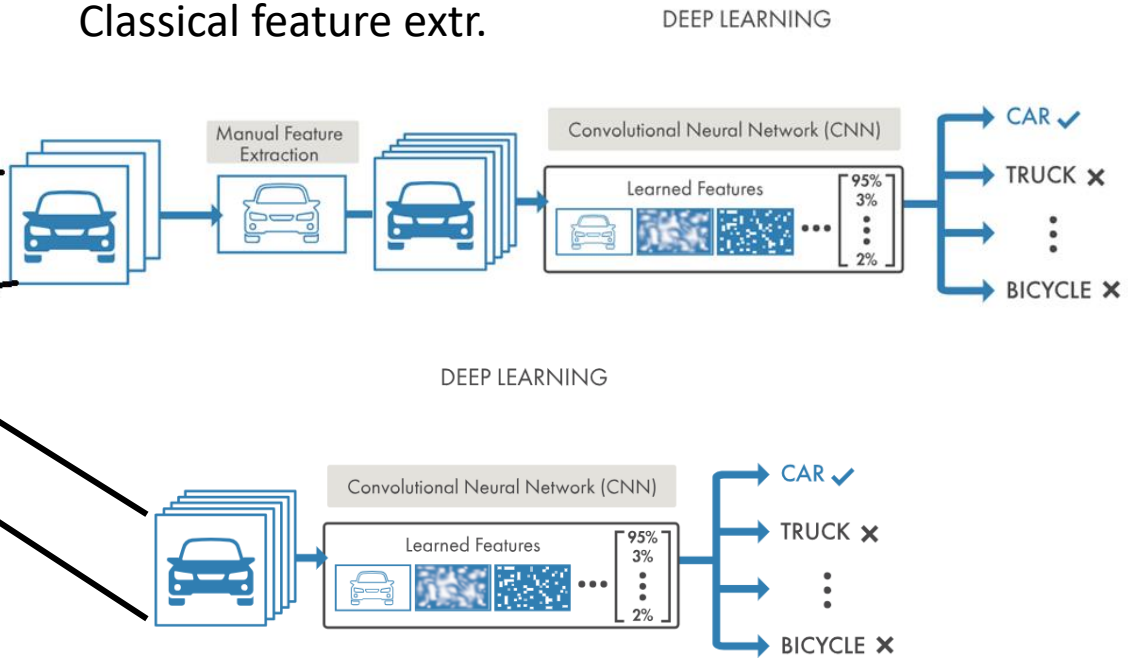


Blob detectors:
where the image responds
to specific kernel convolution
(see “similarity” lessons)

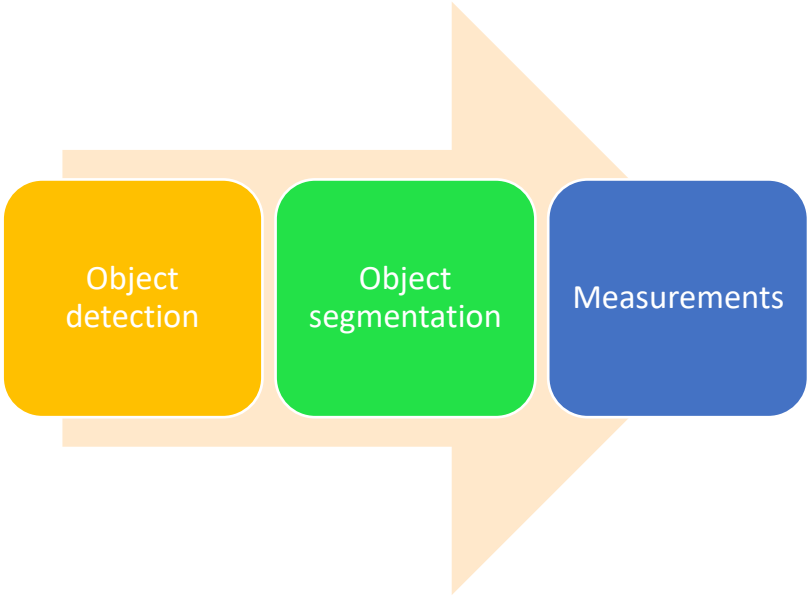
Starting from classical
blob detectors/segmentation
/block processing

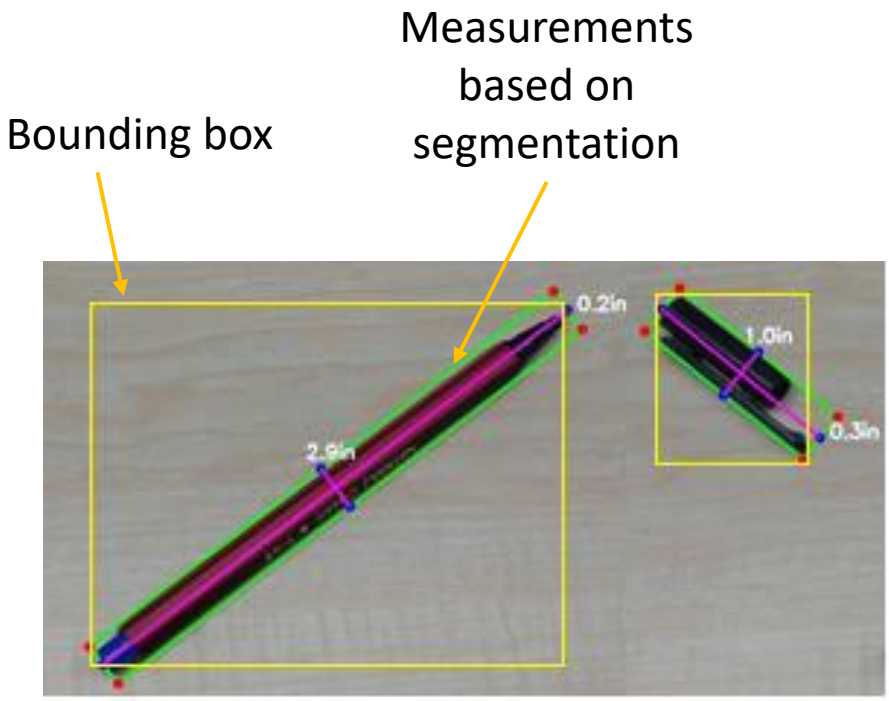


Classical feature extr.

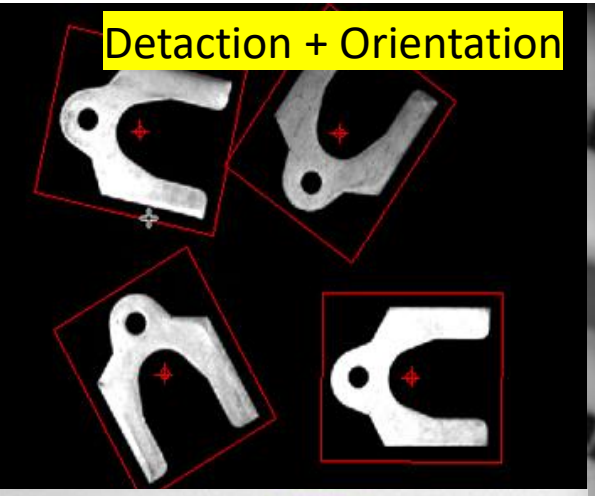


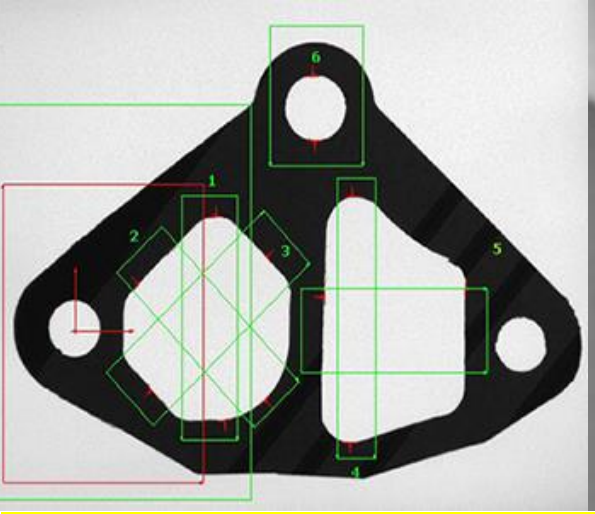
# Object Measurement

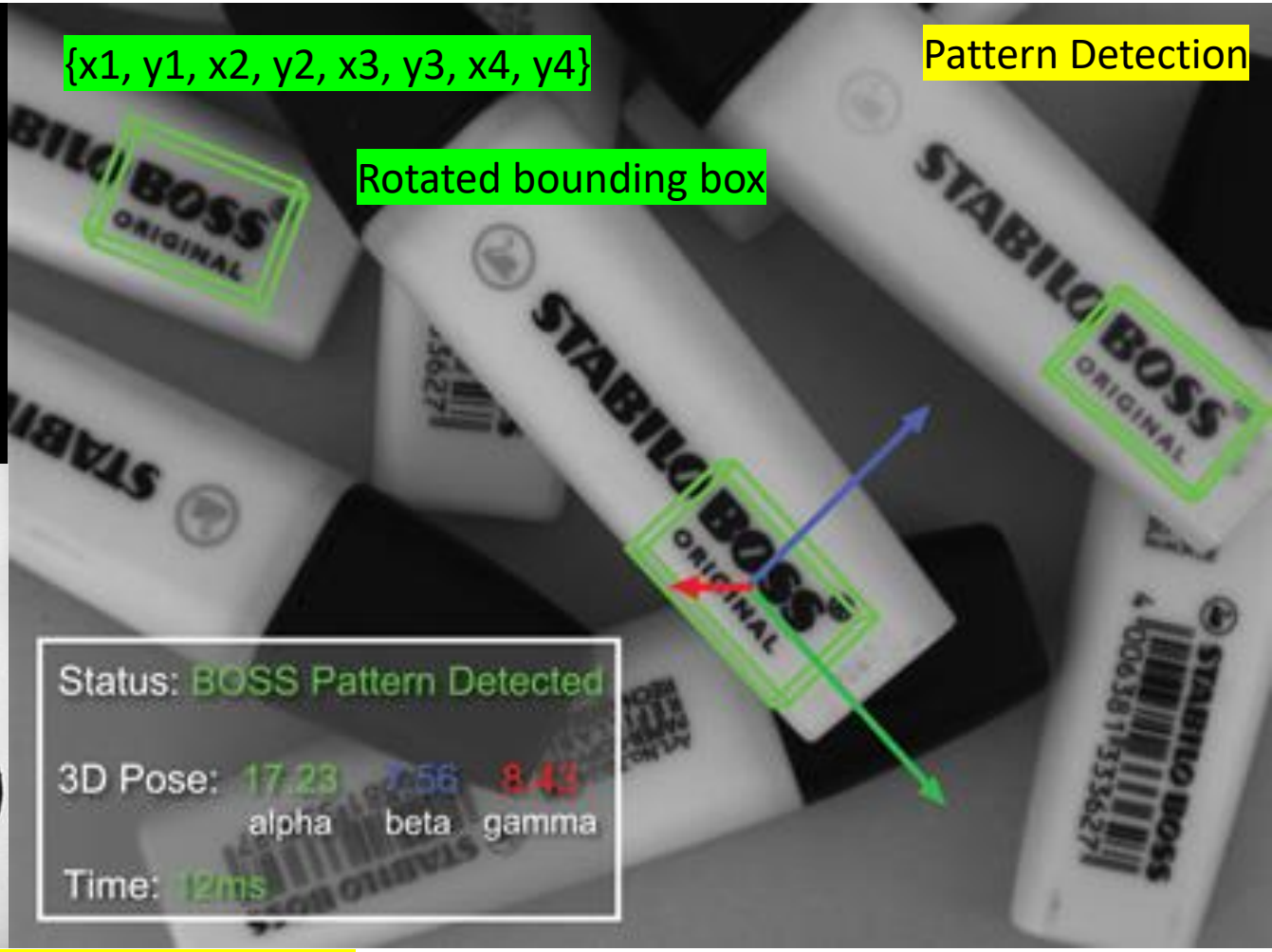


# Object measurements in Industrial applications



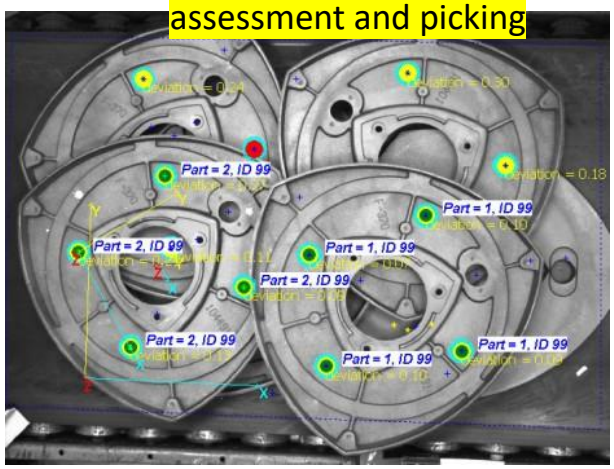





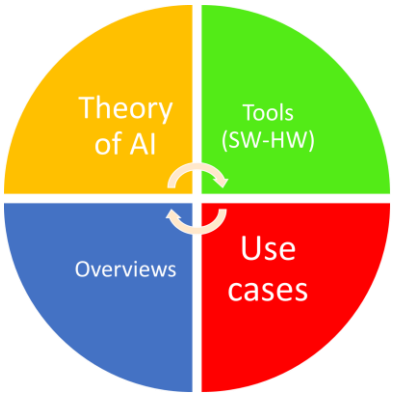

Measurements with traditional pattern matching

# Object segmentation and measurement: Bin picking

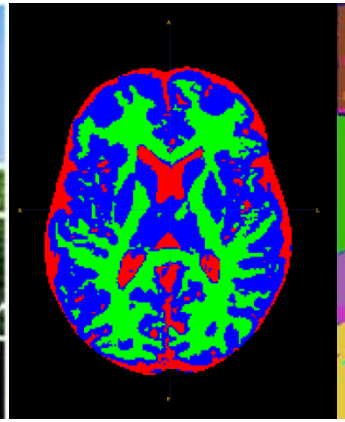

Processing the exact grabbing position with intelligent vision system

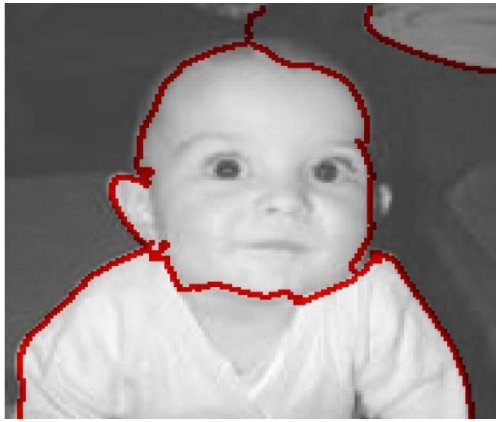



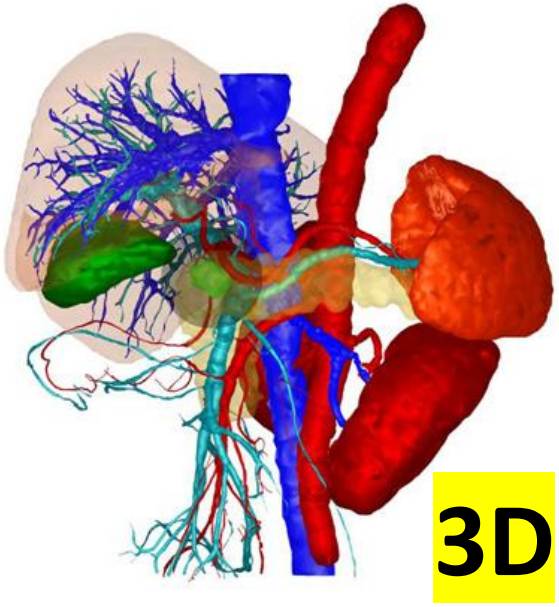
# THEORY
# Object segmentation

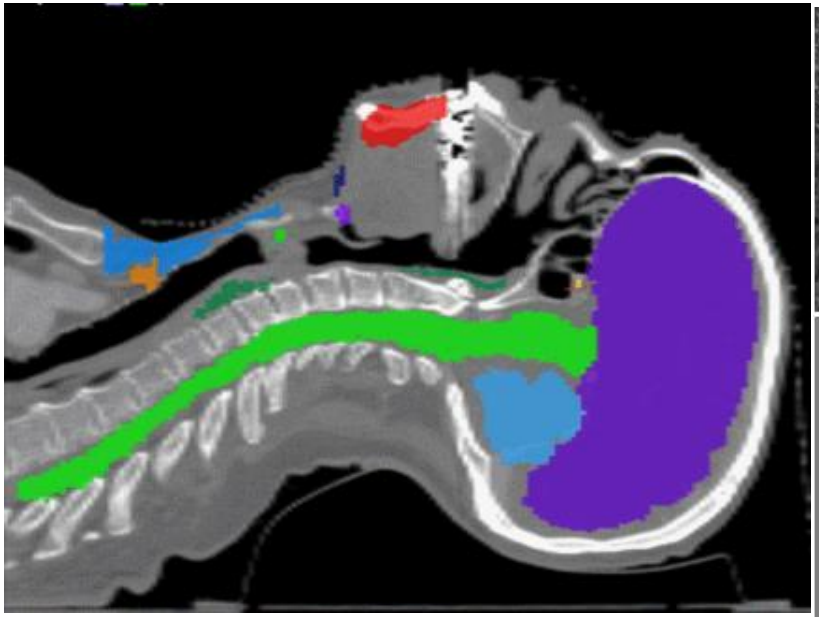


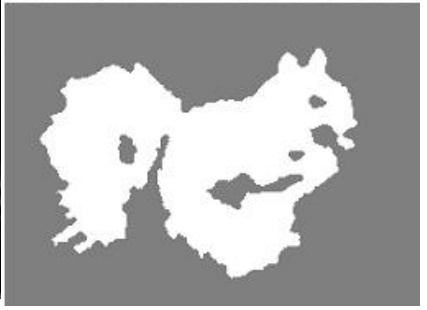
# General segmentations







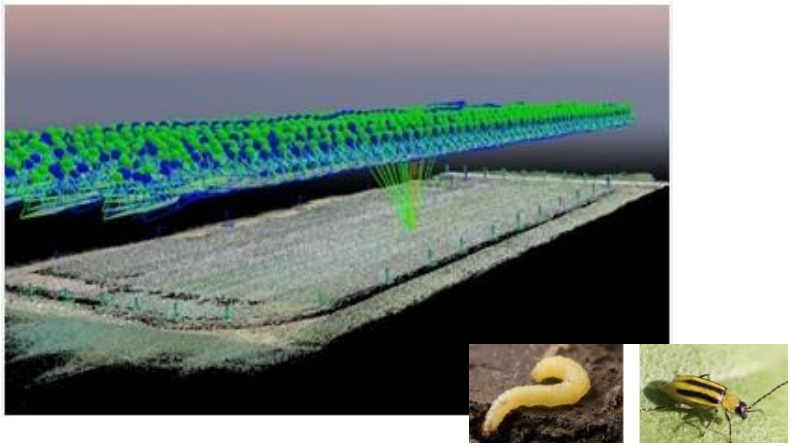


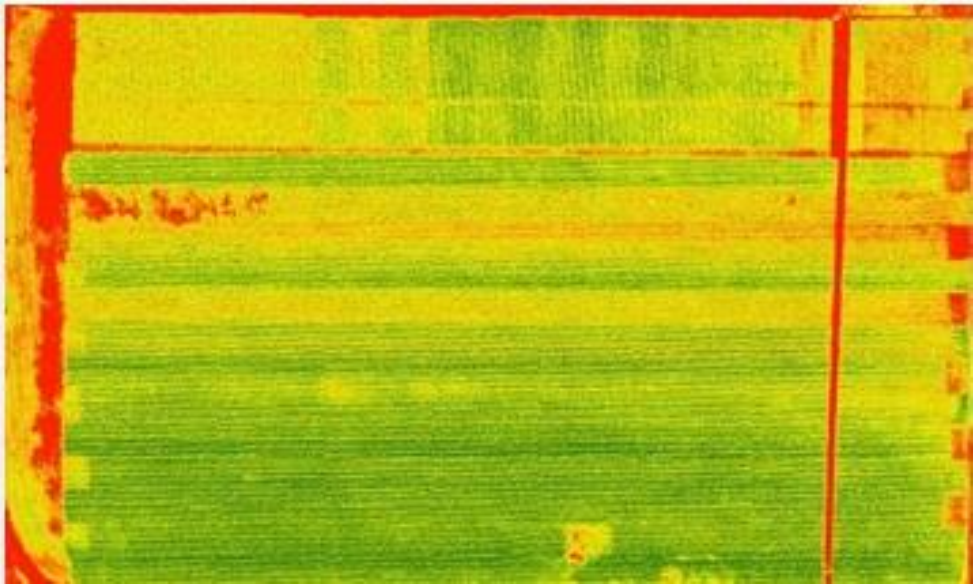
From 3D data it’s possible to segment volumes of the objects

# Enviromental applications: Drones

frames

Segmentation and measurements





- Example of an outbreak of Corn rootworm.
- The static images and video would help the farmer or crop service vendor identify generally where the damage is occurring.
- Integration with Geographic Information Systems (GIS).

# Satellites and airborn Enviromental applications

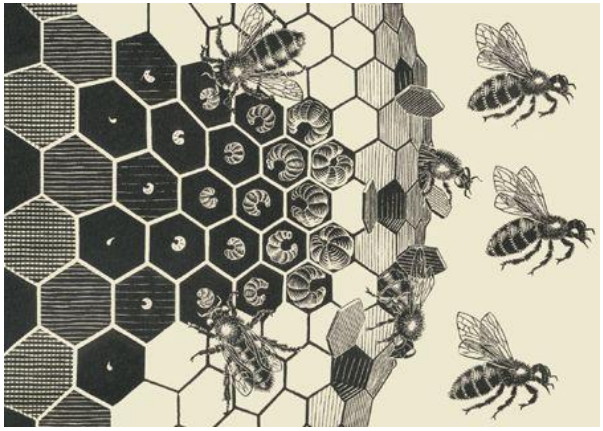

Multiband

Intelligent Vision Systems

Segmentation and measurements

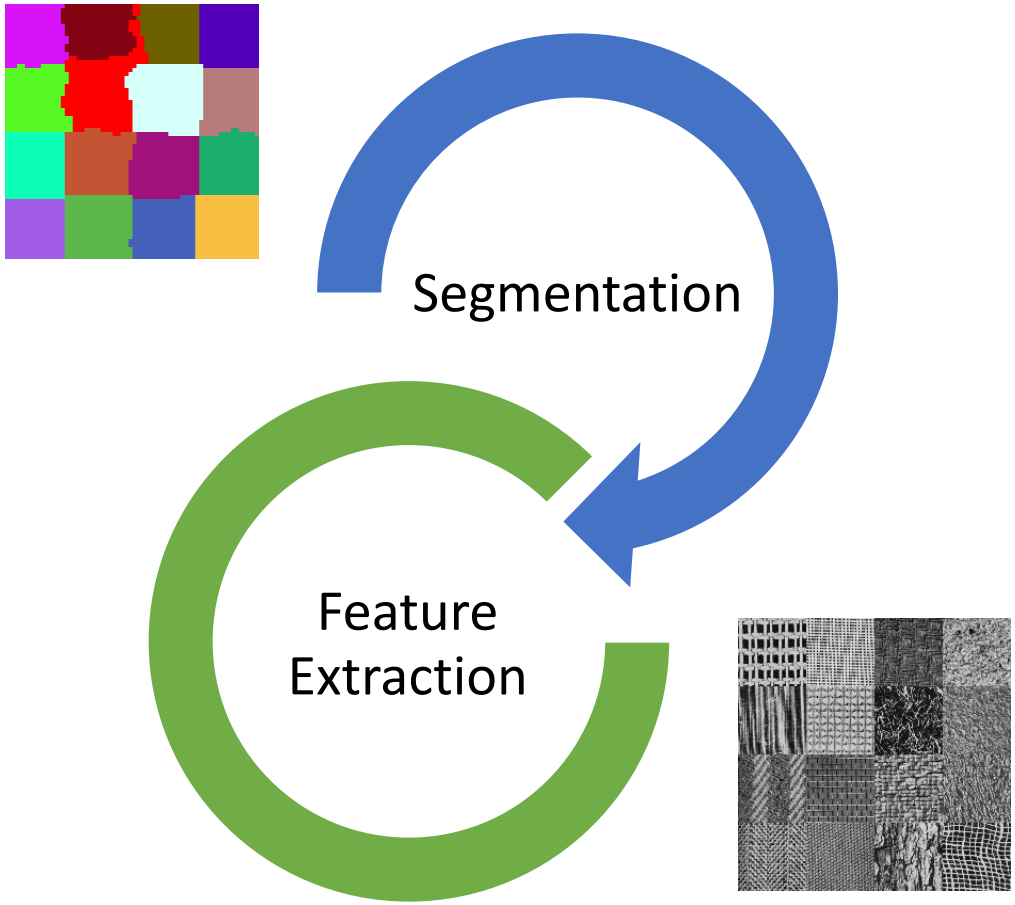
# Segmentation “definition”

- Segmentation
  - Split or separate an image into regions
  - To facilitate recognition, understanding, and Region Of Interests (ROI) processing



- Ill-defined problem
  - The definition of a region is context-dependent

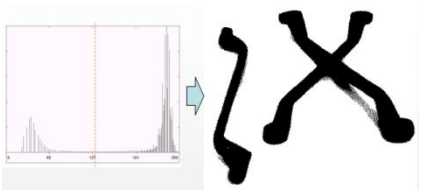
# Two close tasks



# Basic taxonomy

Traditional Segmentation methods

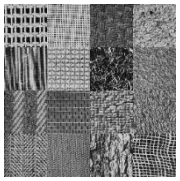
**Global knowledge**, (using histogram image features)

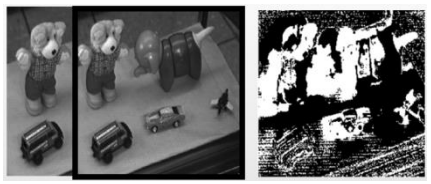


**Edge-based** segmentations

**Region-based** segmentations

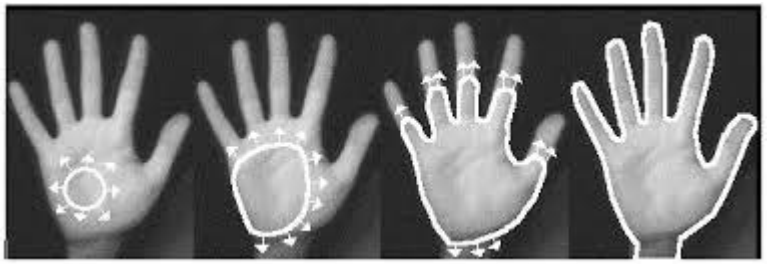
Brightness, texture, velocity field, color,..





# Some classical methods
## (for your reference in case of need)

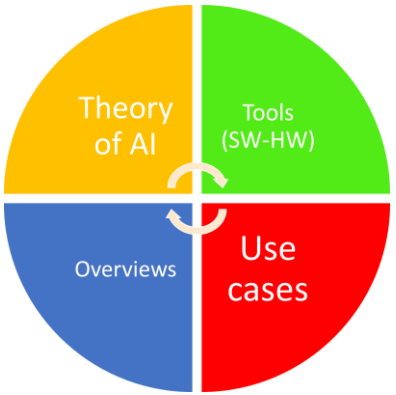
- Thresholding
- Clustering methods
- Compression-based methods
- Histogram-based methods
- Edge detection
- Region-growing methods
- Split-and-merge methods
- Partial differential equation-based methods
  - Parametric
  - Level set
  - Fast Marching
- Graph partitioning methods
- Active Contour Models - “Snakes”
- Watershed transformation
- Model based segmentation
  - Deformable models (Snakes)
- Multi-scale segmentation
  - One-dimensional hierarchical signal segmentation
  - primal sketch
- Semi-automatic segmentation
- Neural networks segmentation



An example of a region-growing method (Snake)



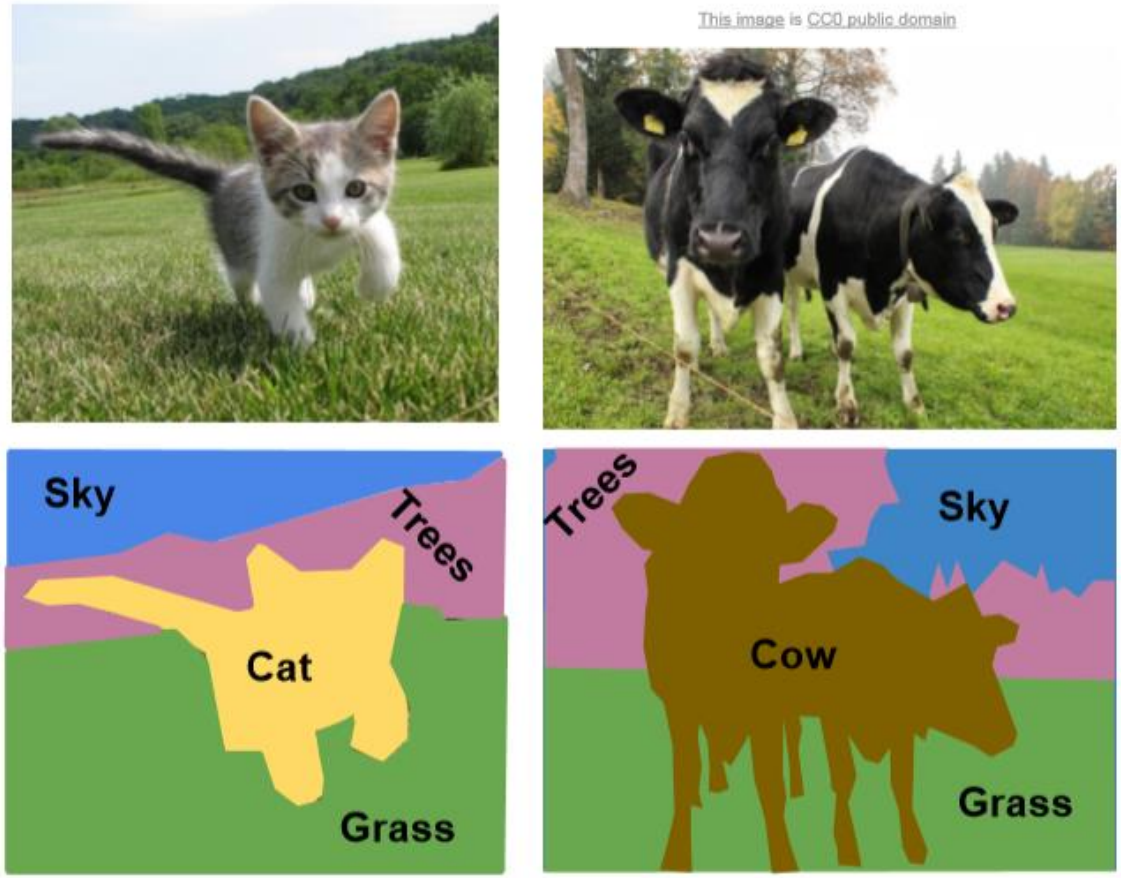

# THEORY
# Semantic segmentation



# Semantic segmentation

Label each pixel in the image with a category label

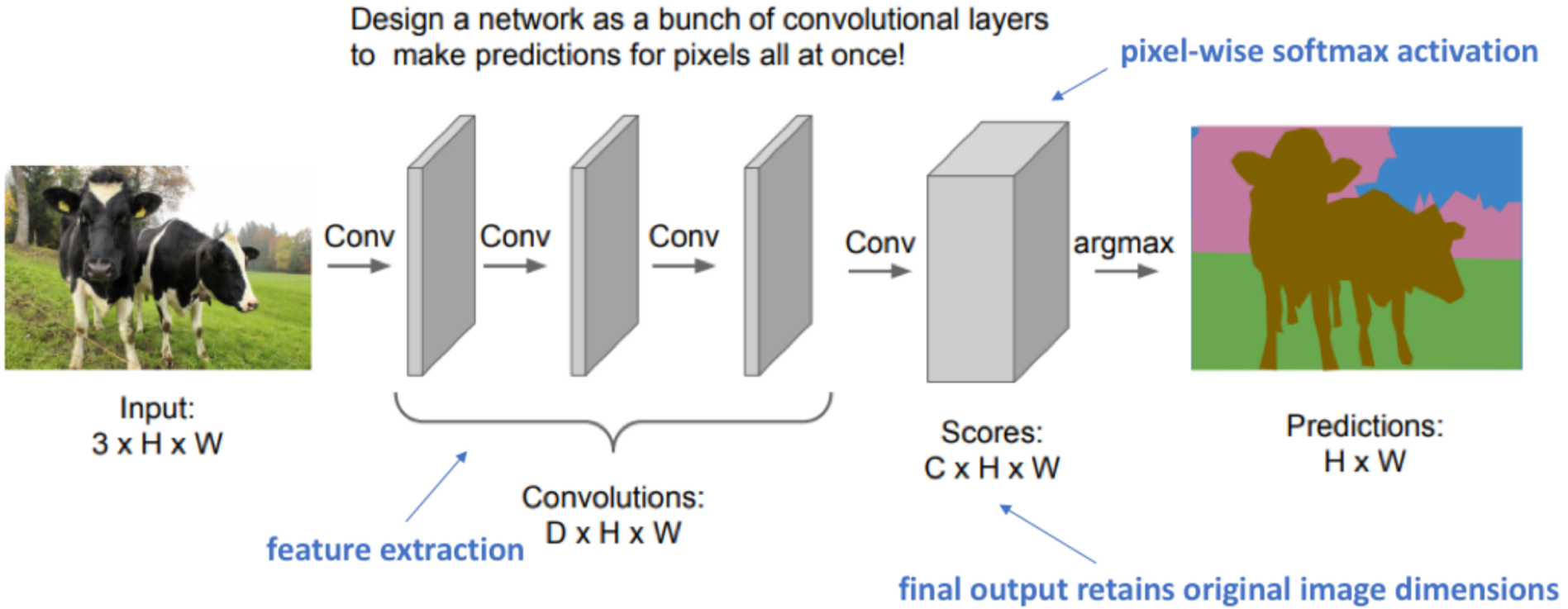
Don't differentiate instances, only care about pixels



# Pixel classification is a good idea but… not feasible

**Fully Convolutional Network**

Image that every single pixel of the output image is classified with a softmax…




Images from Fei-Fei Li & Justin Johnson & Serena Yeung (Stanford)

# Semantic segmentation
# A fully convolutional solution

**Fully Convolutional Network with Downsampling and Upsampling**

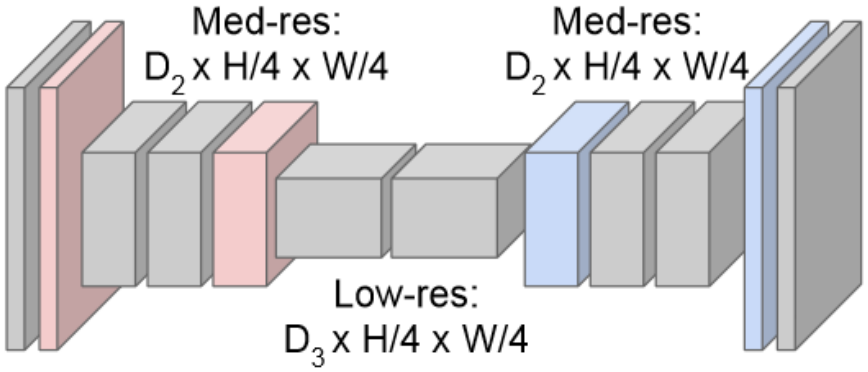
Encoder/decoder structure

**Downsampling**:
Pooling, strided convolution

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!

**Upsampling**:
Unpooling or strided transpose convolution

Input:
3 x H x W




High-res:
$D_1$ x H/2 x W/2

High-res:
$D_1$ x H/2 x W/2

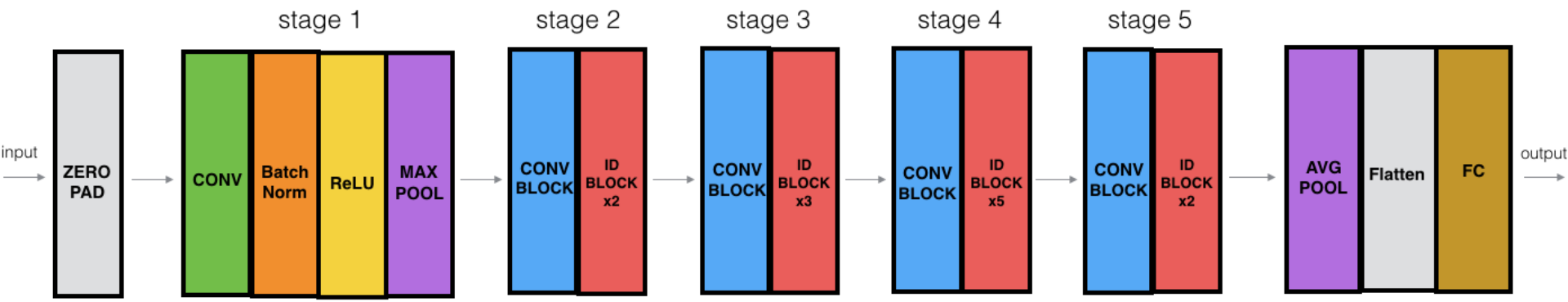
Predictions:
H x W

DECONVOLUTIONS

Images from Fei-Fei Li & Justin Johnson & Serena Yeung (Stanford)

The pretrained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals

# ResNet-50 for classification

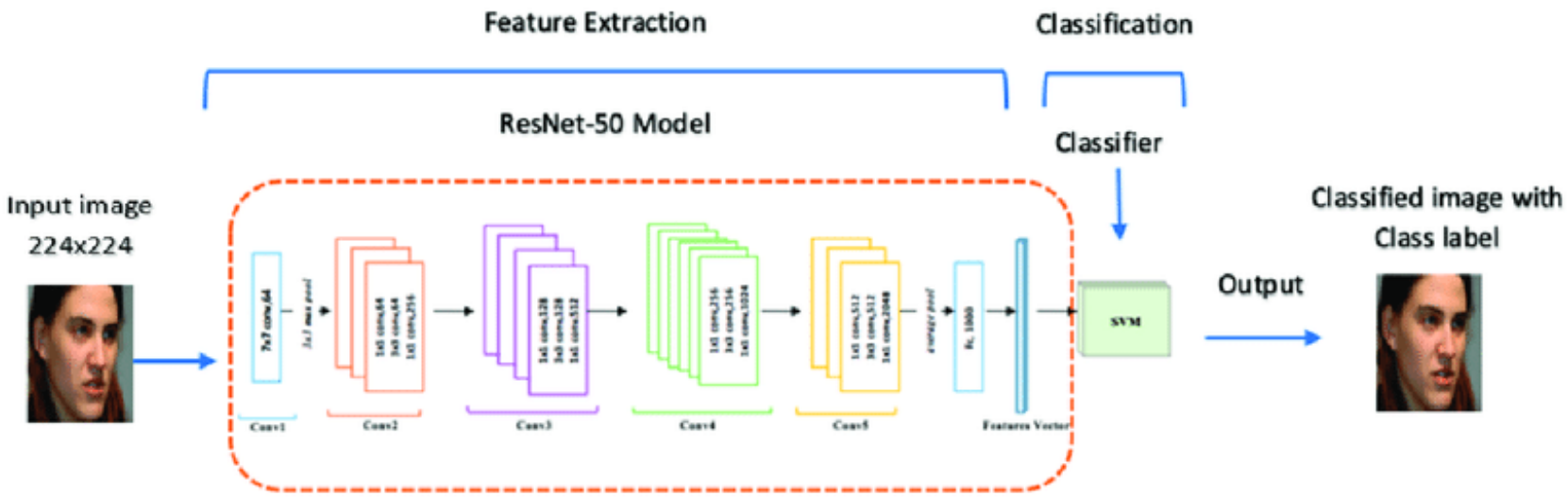



The native network is designed for classification (the output is an integer)!

KERAS: model.param_count()
→ **25M parameters**

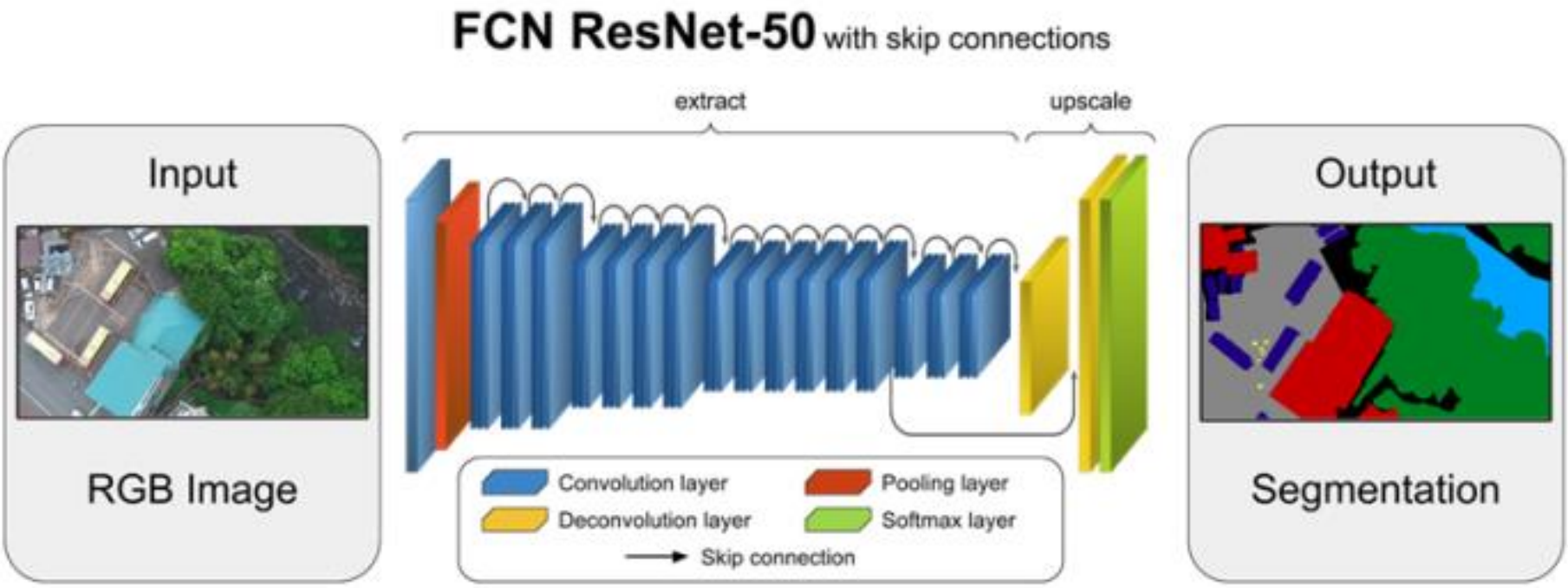
| layer name | output size | 18-layer | 34-layer | 50-layer | 101-layer | 152-layer |
|---|---|---|---|---|---|---|
| conv1 | 112×112 | 7×7, 64, stride 2 | | | | |
| | | 3×3 max pool, stride 2 | | | | |
| conv2_x | 56×56 | $\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$ | $\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$ | $\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$ | $\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$ | $\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$ |
| conv3_x | 28×28 | $\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$ | $\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$ | $\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$ | $\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$ | $\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$ |
| conv4_x | 14×14 | $\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$ | $\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$ | $\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$ | $\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$ | $\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$ |
| conv5_x | 7×7 | $\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$ | $\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$ | $\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$ | $\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$ | $\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$ |
| | 1×1 | average pool, 1000-d fc, softmax | | | | |
| FLOPs | | $1.8\times10^9$ | $3.6\times10^9$ | $3.8\times10^9$ | $7.6\times10^9$ | $11.3\times10^9$ |

# ResNet-50 for feature extraction

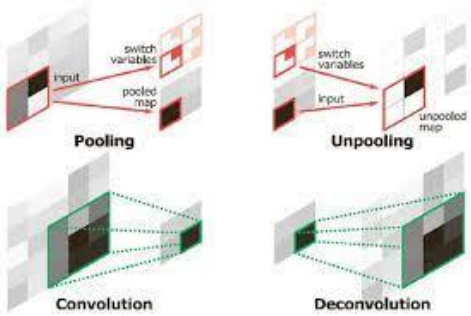



You can use ResNet-50 to process features and then use your classifier to create a new task

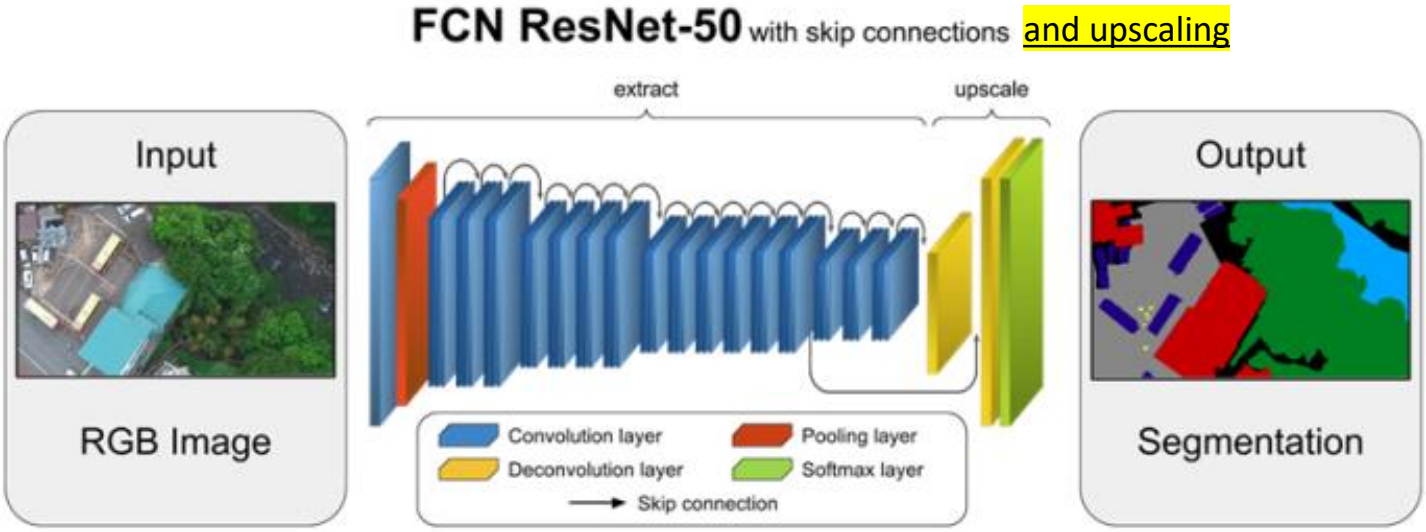
# ResNet-50 → ResNet50 + Upscaling trainable semantic segmentation




Deconvolution is the operation inverse to convolution.



# ResNet-50 with upscaling for classification? NO!




The native network is designed for segmentation (the output is an image) not for classification (the output is an integer)!

# Image classification via DL

Classical Pre-Trained Models for Image Classification

- VGG-16
- VGG-19
- ResNet50
- Inceptionv3
- EfficientNet B0-B7

| Model | Year | Number of Parameters |
|---|---|---|
| VGG-16 | 2014 | 138 Million |
| ResNet-50 | 2015 | 25 Million |
| Inception V3 | 2015 | 24 Million |
| EfficientNetB0 | 2019 | 5.3 Million |
| EfficientNetB7 | 2019 | 66 Million |

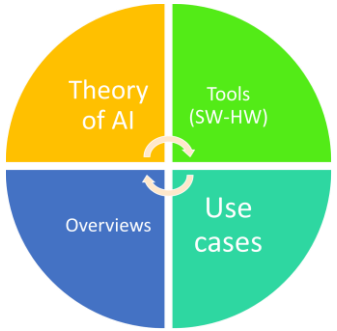




# Use cases

# Objects recognition from UAVs/drones

Unmanned Aerial Vehicle (UAV)



# Drones/UVAs in applications

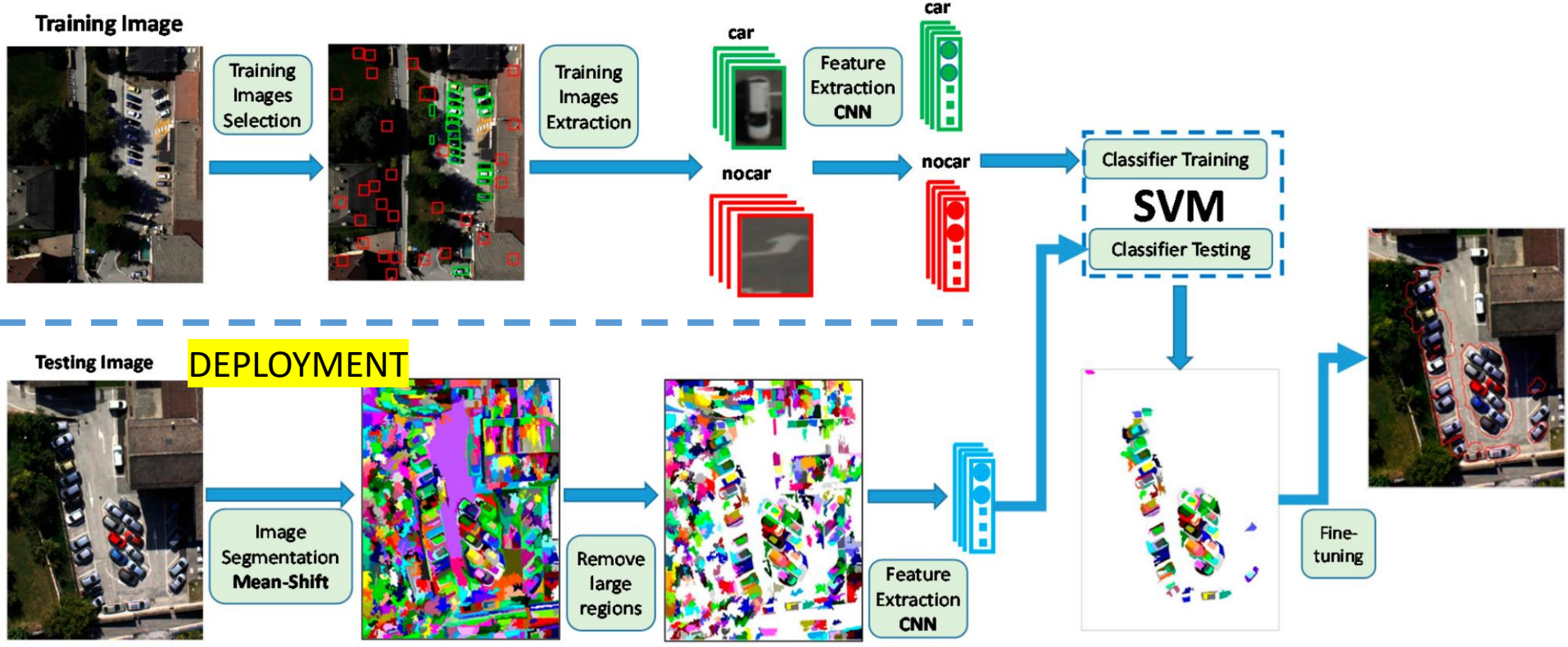



# Objects recognition from UAVs

Landing mark

Traffic control

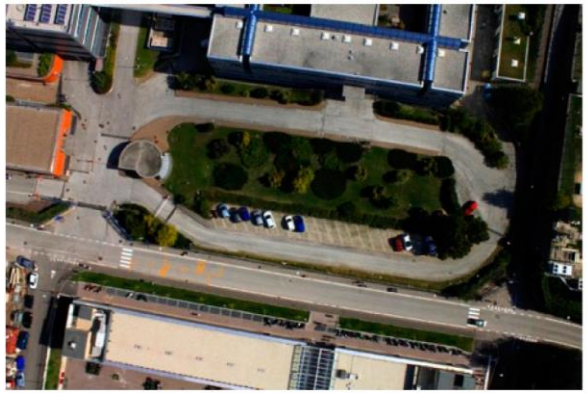
TRAINING

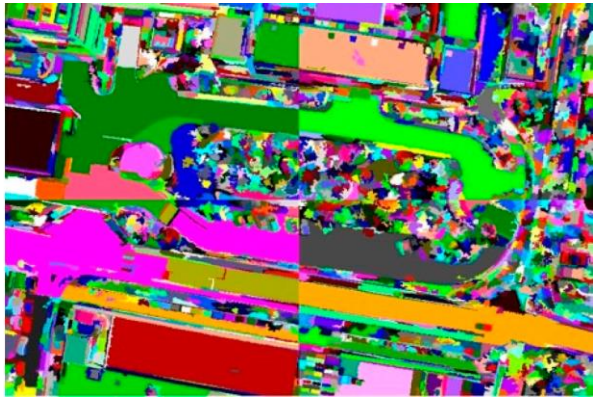


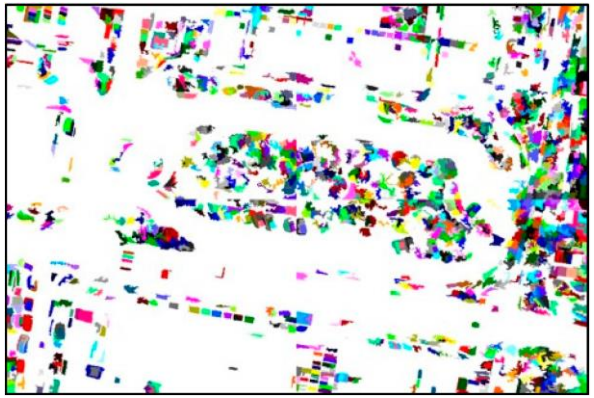

TRADITIONAL SEGMENTATION

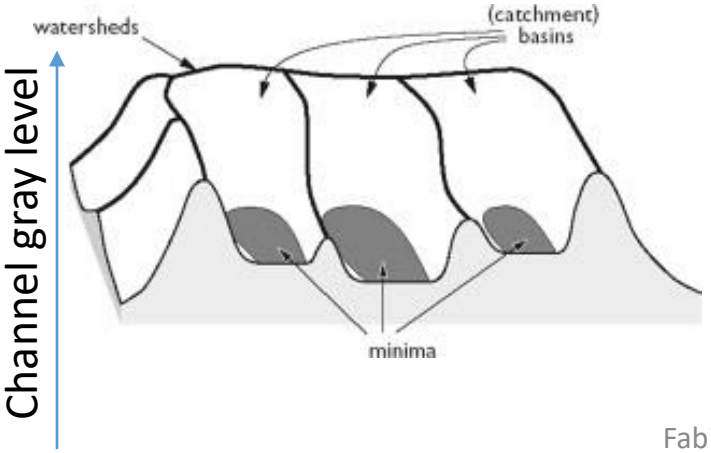
# Obj. Rec. from UAVs: STEP 1 - segmentation

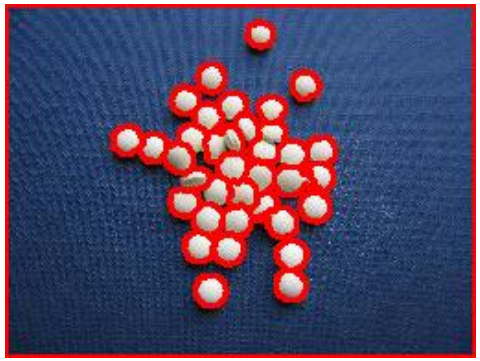
TRADITIONAL SEGMENTATION FOR CANDIDATES









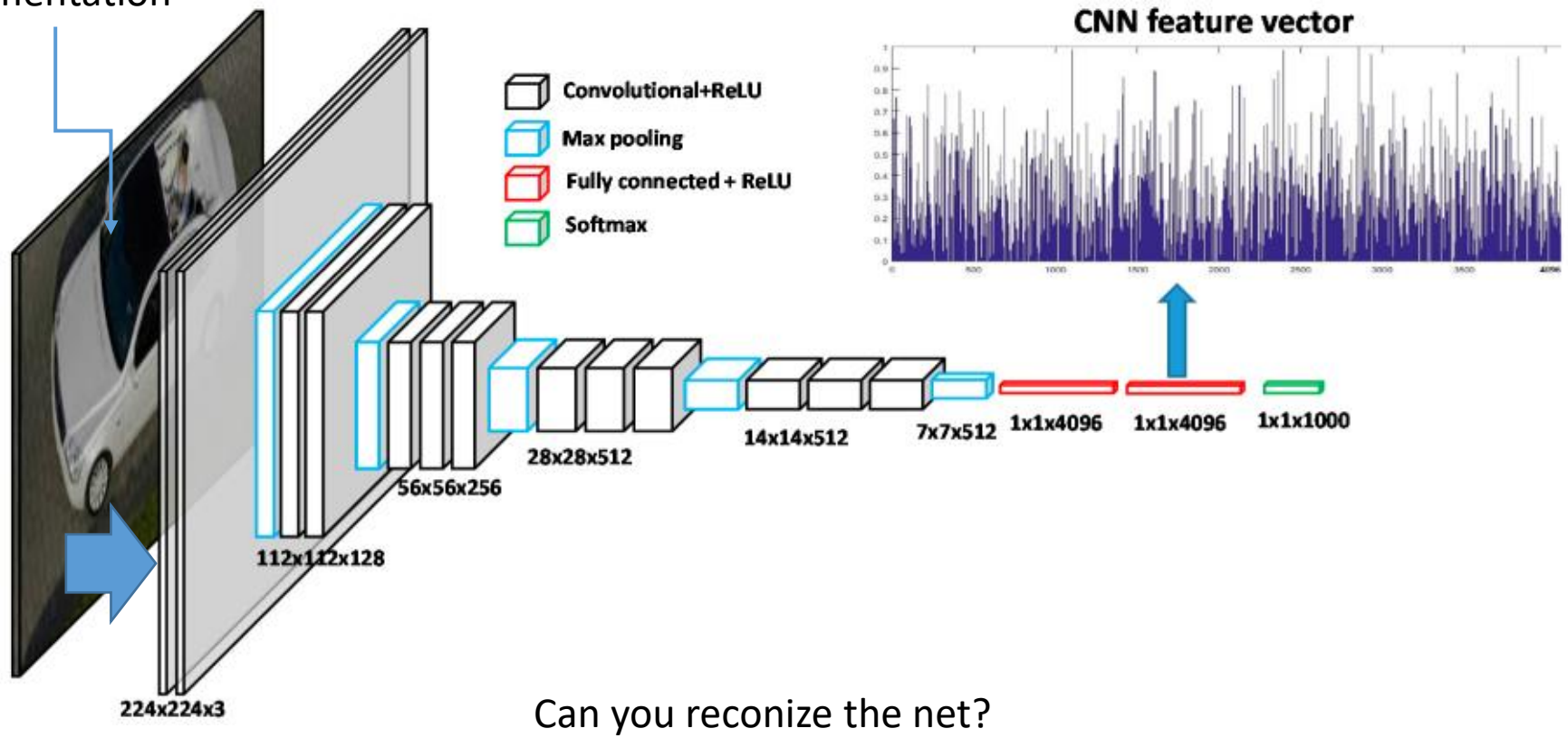



Example:
Watershed Segmentation

# Obj. Rec. from UAVs: STEP 2 - CNN design

You can use tradition classifier like SVM or deep models

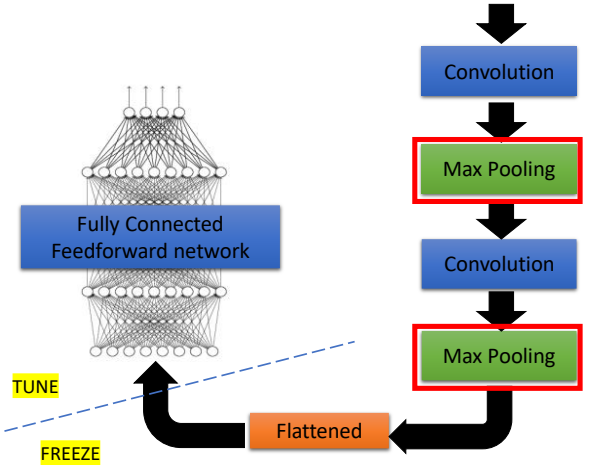
From segmentation



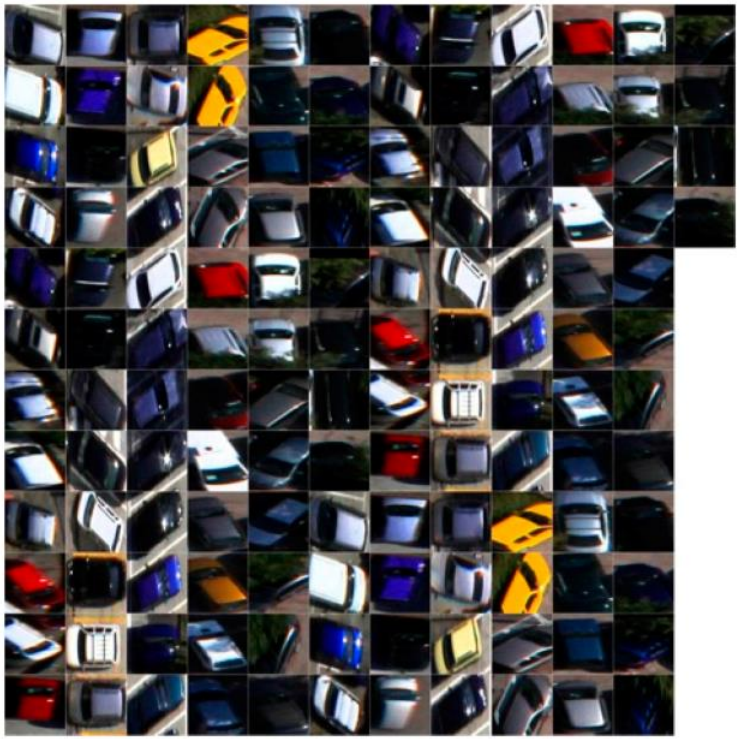

Can you reconize the net?

VGG16

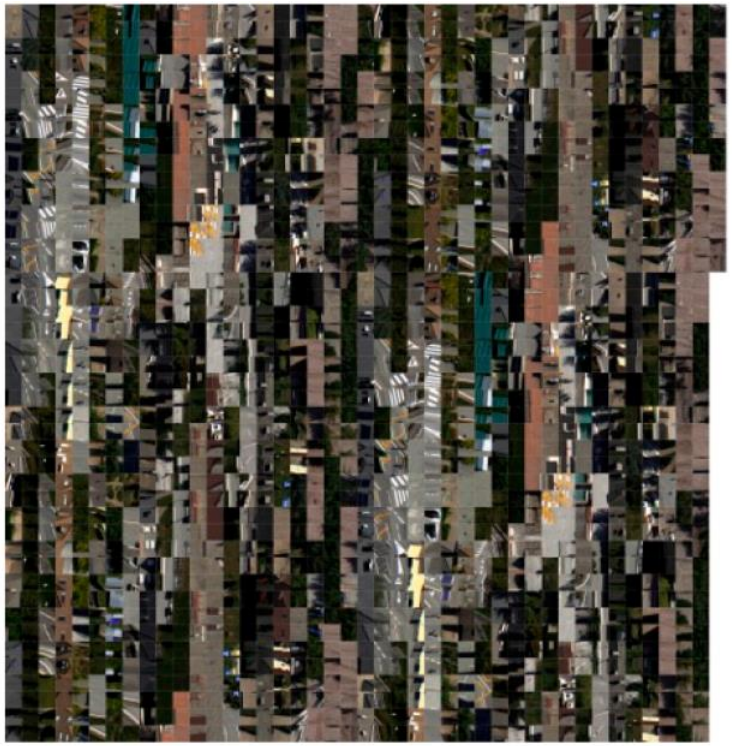
# Obj. Rec. from UAVs: CNN learning dataset creation




FINE TUNING



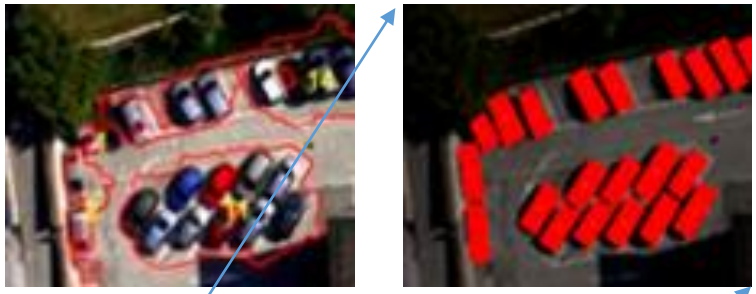
CARS

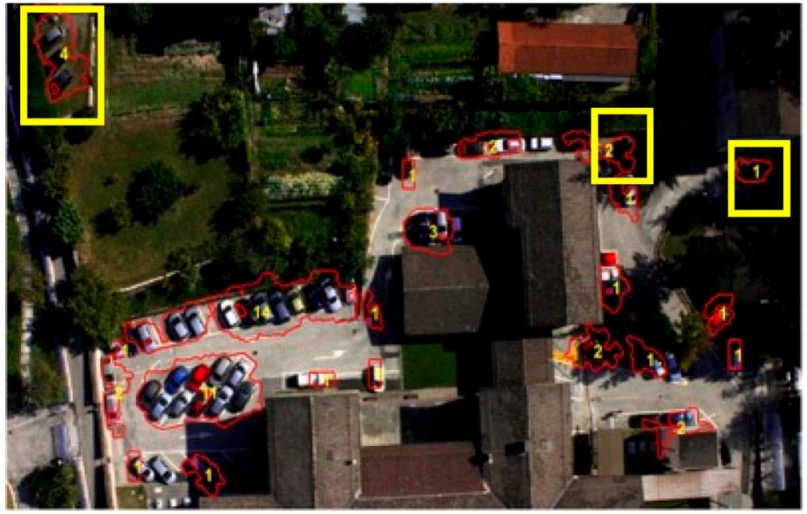


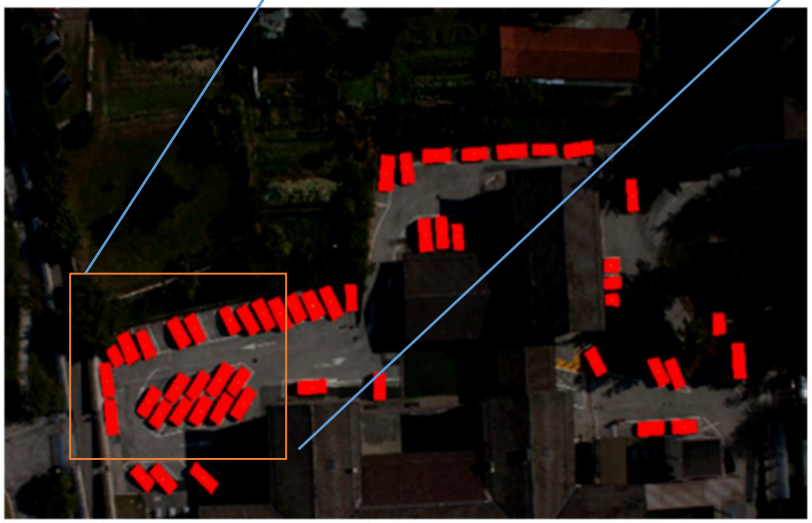
NO CARS

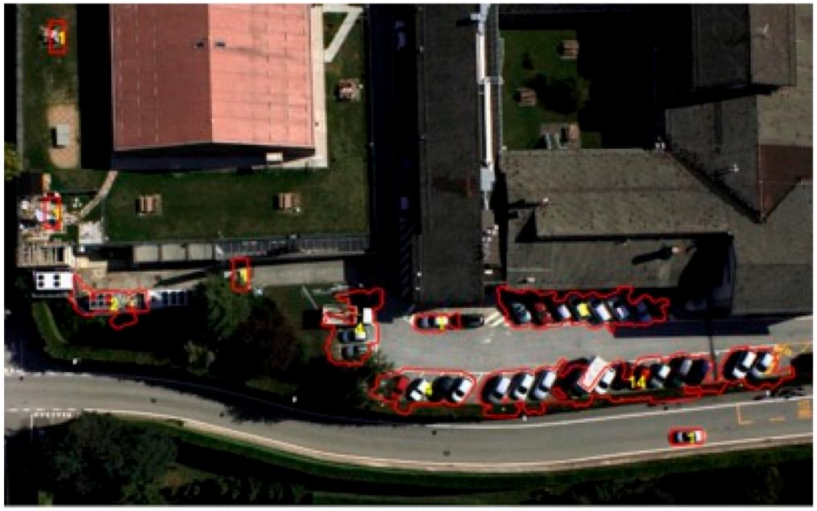
(just a portion)

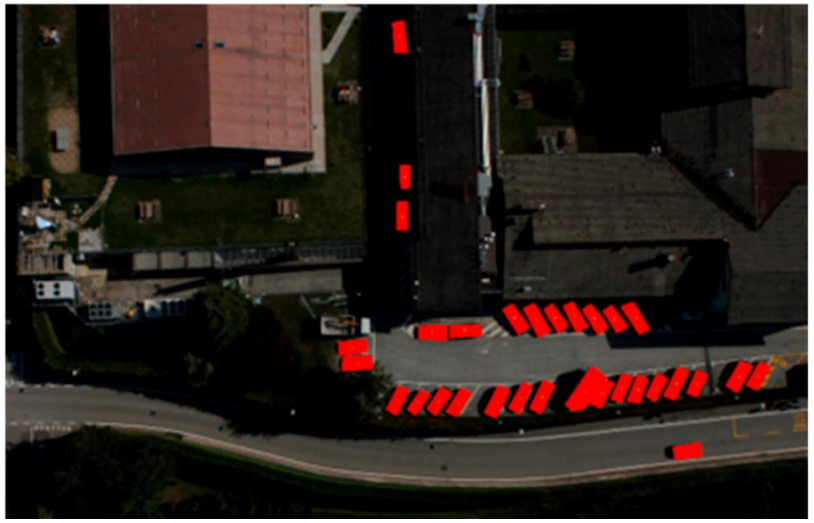
# Obj. Rec. from UAVs: results



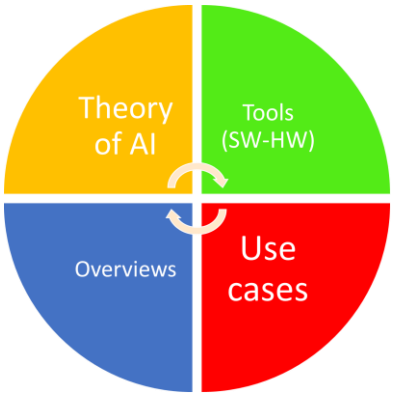








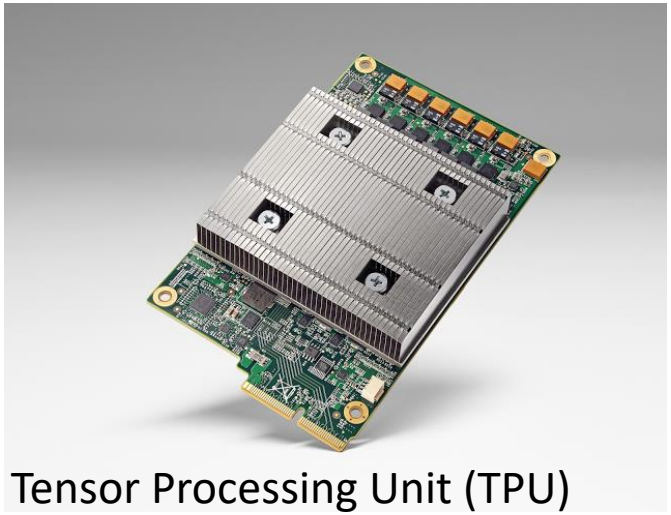
# THEORY
# Machine Learning HW



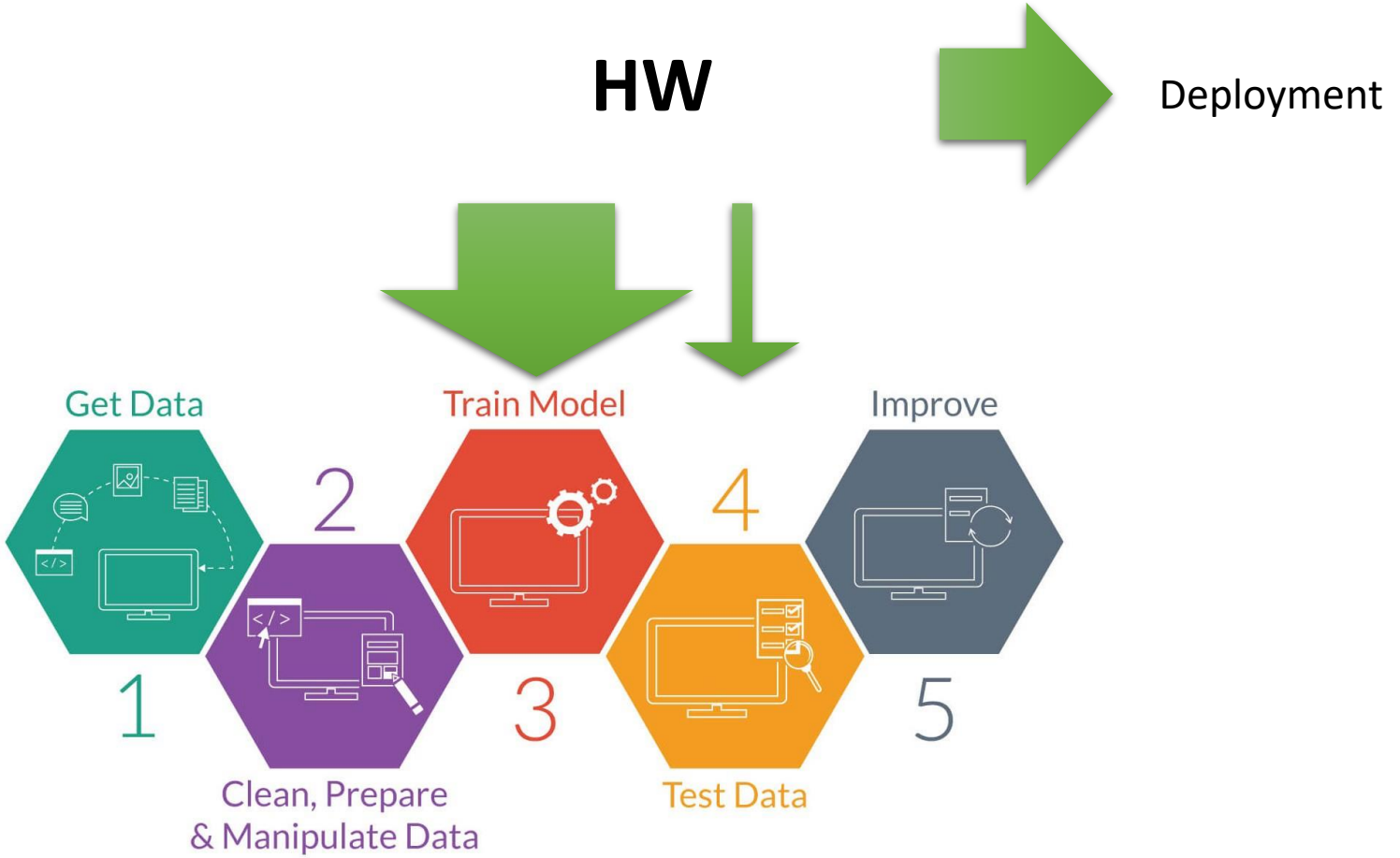
# Why Machine Learning Is Possible?

- Mass Storage
  - More data available (TB HD, data centers,..)
- Higher Performance of Computer
  - Larger memory in handling the data
  - Greater computational power for calculating and even online learning

Nvidia GXT Titan and an InfiniBand

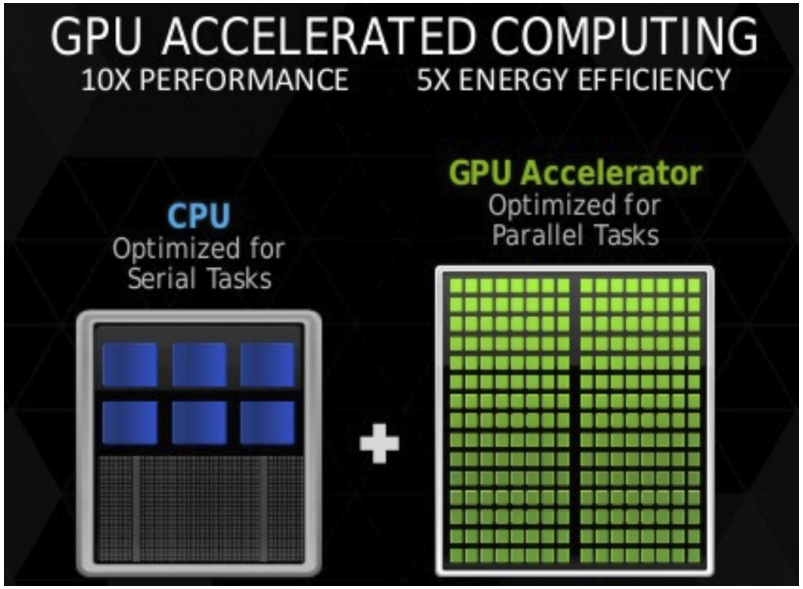

Tensor Processing Unit (TPU)

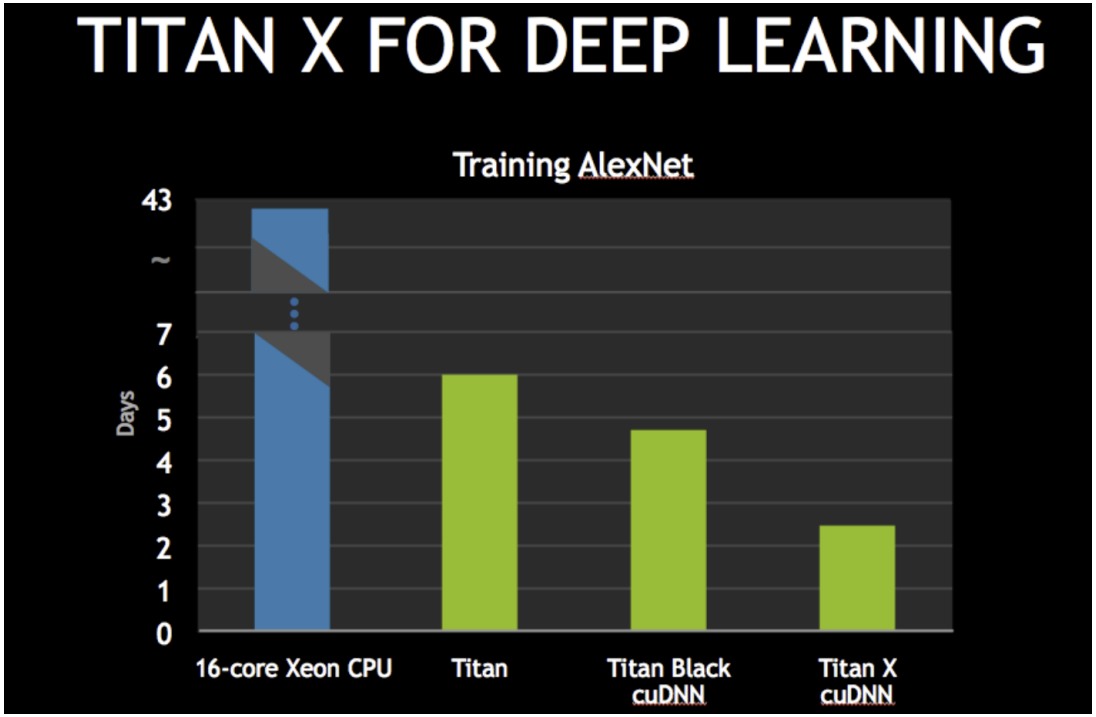
# HW:
# When and where is more relevant?



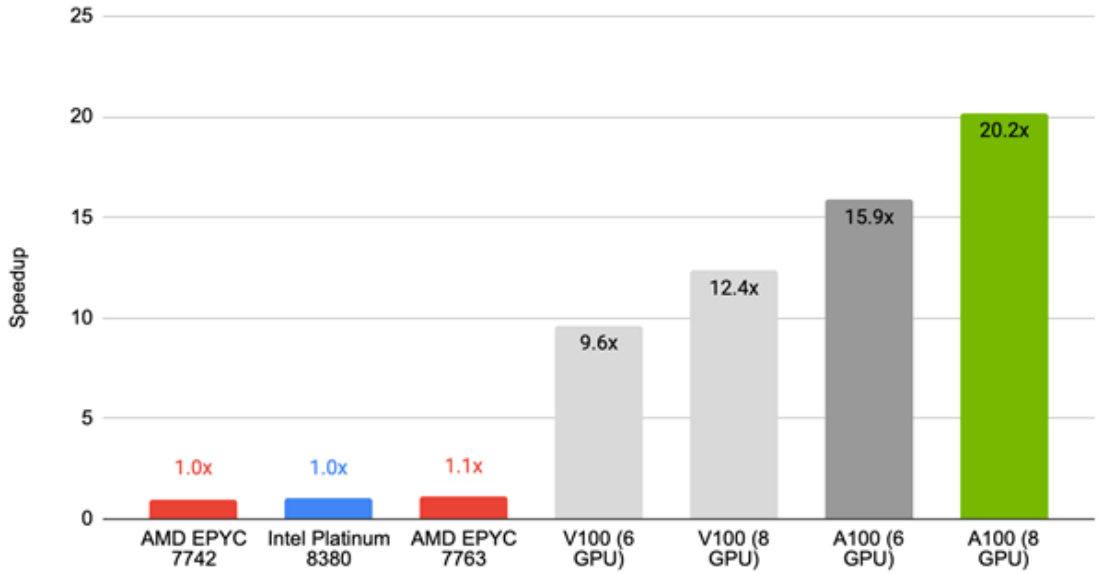
# GPU vs CPU

- In brief….you need a GPU for large projects!

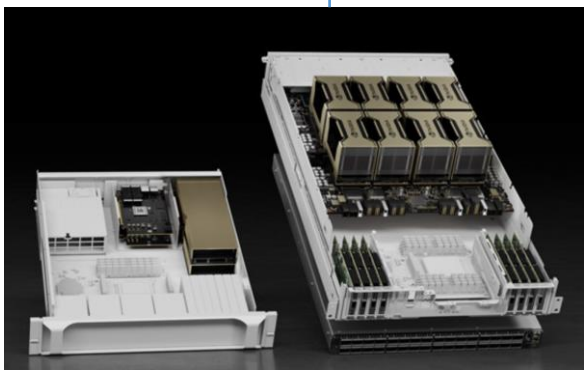


# Some other data




>8Keuro



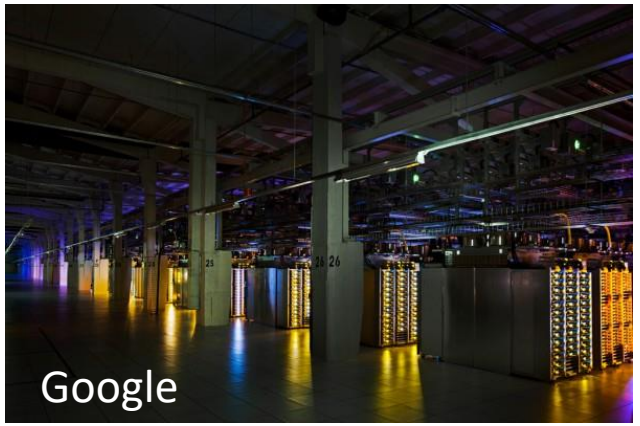
SUPERCOMPUTER
NVIDIA HGX AI
A100 (8GPU)

>100Keuro

# ML Server farm…

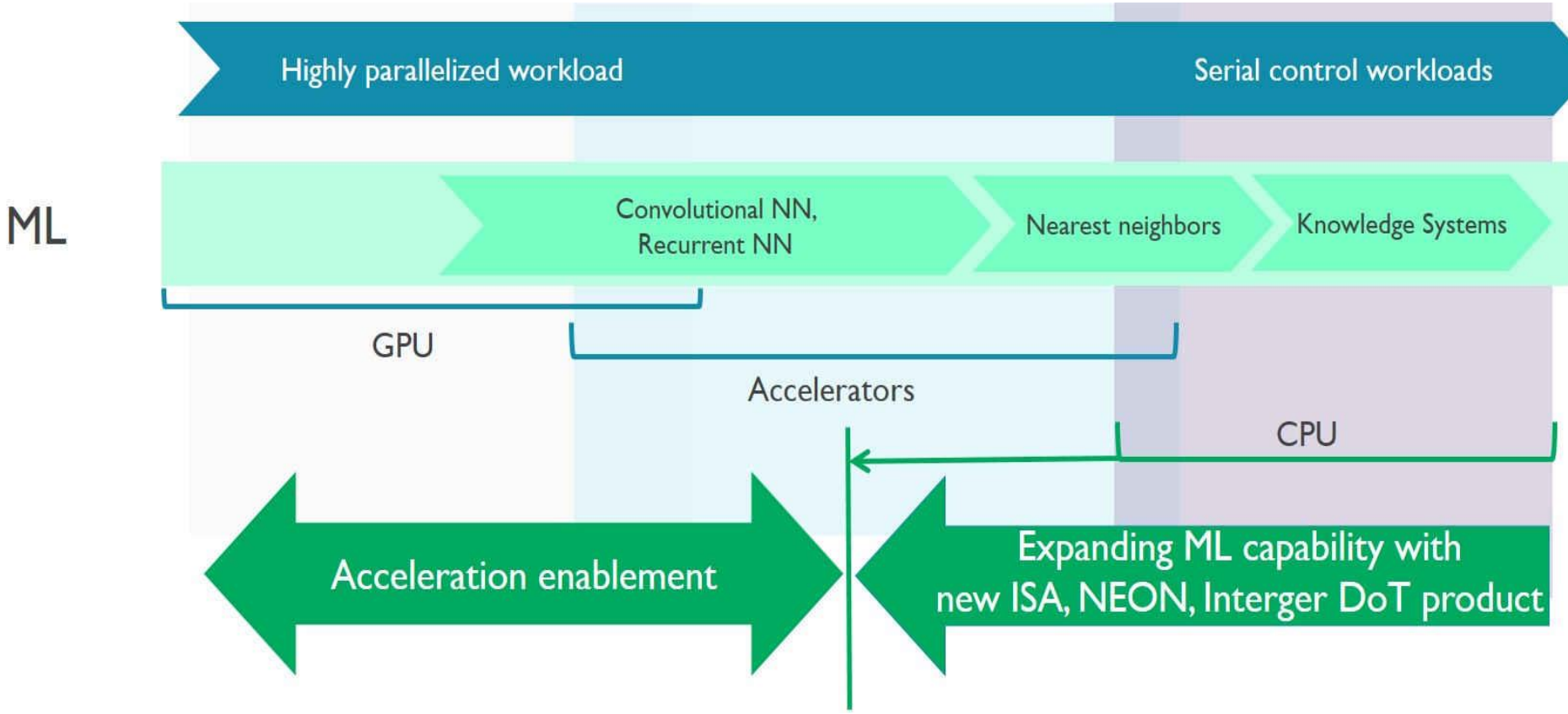
- Amazon (AWS)
- Google
- Facebook
- Microsoft (AzureML)
- …


IBM


Google


Big Sur servers inside the Facebook data center in Prineville

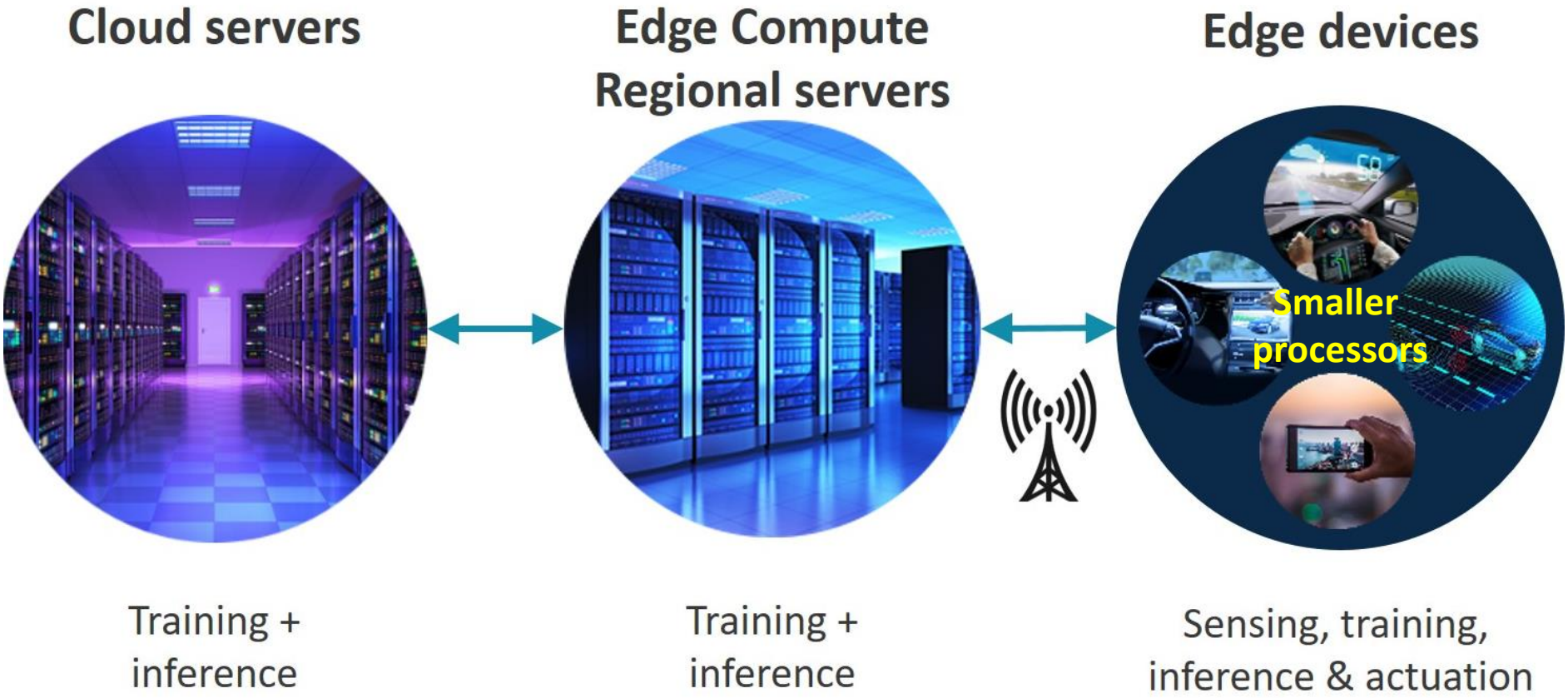
# GPUs: not always needed



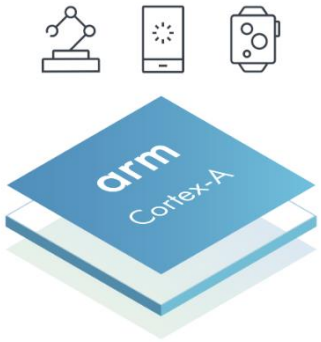
# ML@CPUs

- Machine Learning on CPUs offer advantages
  - Ease of **portability** and use-case flexibility
    - Same code, different applications
  - Market availability at different
    - performances
    - **prices**
  - Deployment across a **wide spectrum of devices** from edge to edge compute and cloud servers

# Moving ML to the «edge»



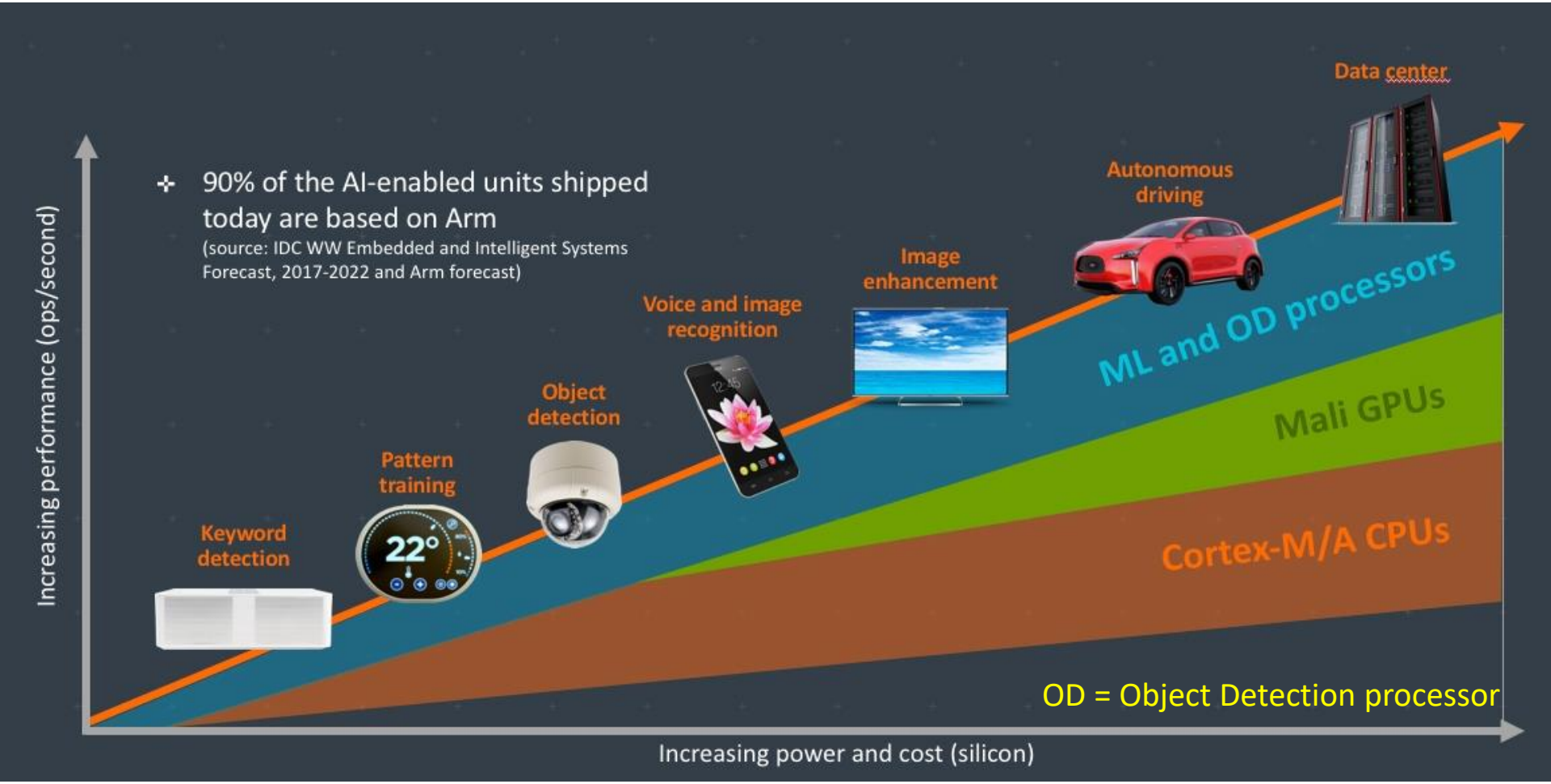
# Example:
# ML on a ARM proc.




- Arm AI Platform
  - A complete, compute platform including Arm Cortex CPUs and Mali GPUs as well as a new line of highly scalable processors that make the platform versatile enough to scale to any device, from IoT to connected cars and servers.
- Set of functions for
  - ML frameworks like Google's TensorFlow
  - Imaging and vision projects
  - Providing **portable code** that can run across various Arm system configurations.
- Example:
  - the neural networks training considering examples often don't require very high accuracy data, meaning that math calculations can usually be **performed on 16-bit or even 8-bit data**, rather than large 32 or 64-bit entries.
- The majority of neural network processing uses **8-bit fixed-point matrix multiplication**, the Armv8.2-A architecture introduced support for half-precision (FP16) and integer dot products (INT8) floating point SIMD (single instruction multiple data) NEON instructions to accelerate ML NN processing.

# HW for ML

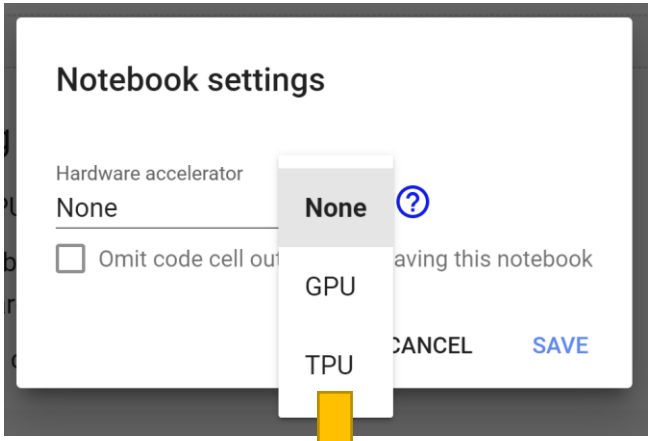
For further reference see
- Proc. Arm Cortex-A77
- GPU Arm Mali-G77
- Chip Arm Machine Learning (ML)

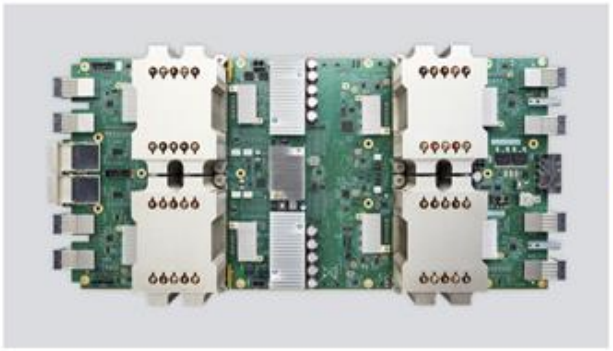


# TPU

TPUs in Colab

- A tensor processing unit (TPU) is an AI accelerator application-specific integrated circuit (ASIC) developed by Google specifically for neural network machine learning

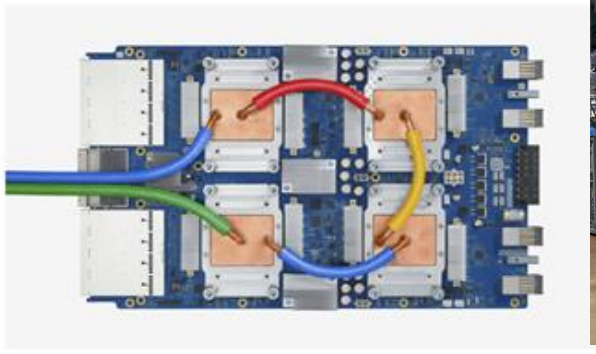


Cloud TPU v2

180 teraFLOPS

64 GB HBM (High Bandwidth Memory)



Cloud TPU v3

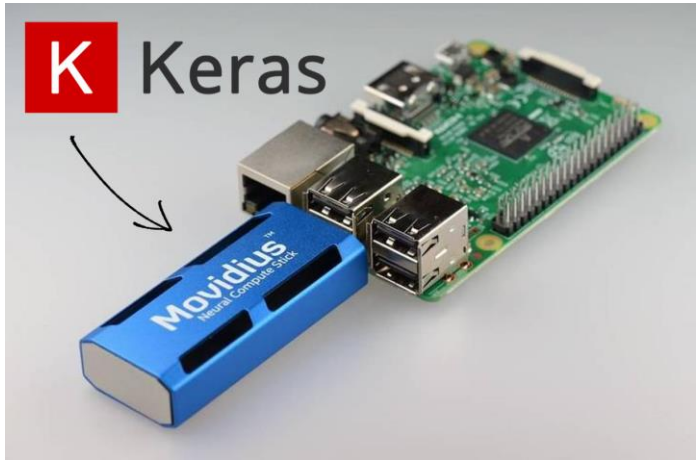
420 teraFLOPS

128 GB HBM

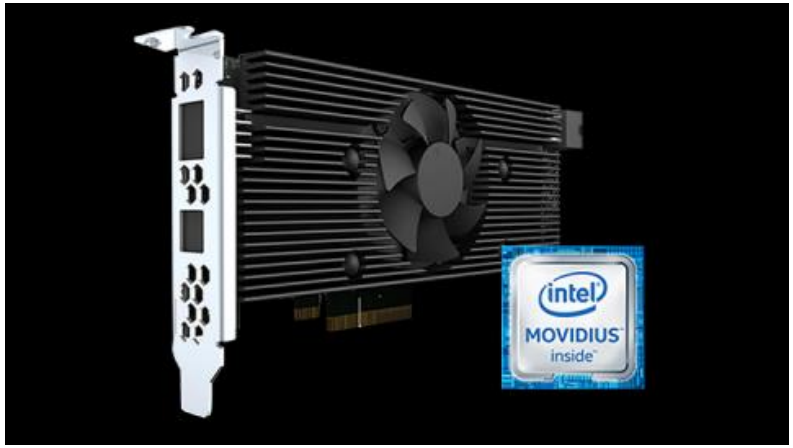
Your CoLab code is running here

# Movidius Neural Compute Stick (Intel)

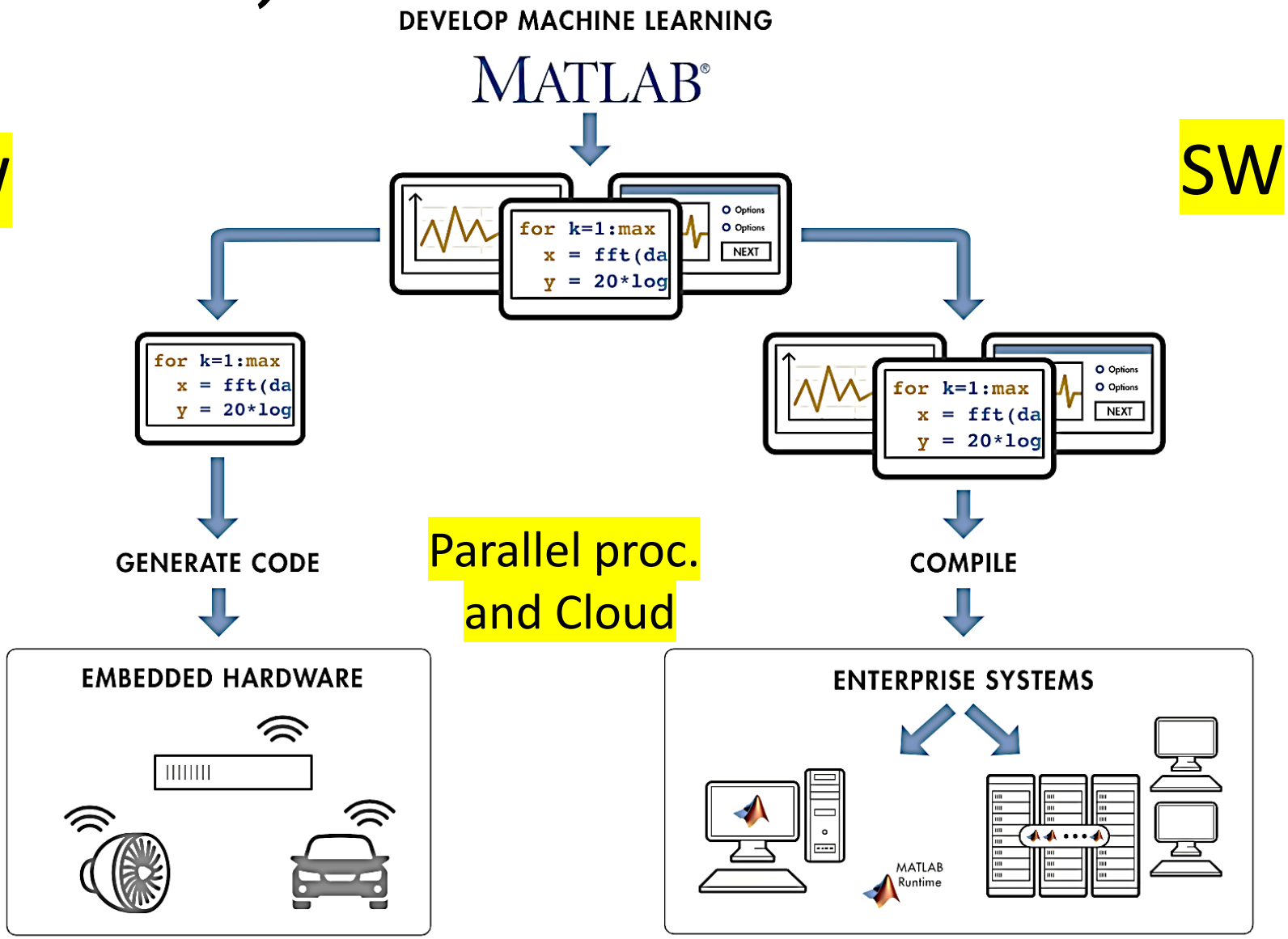
- The Intel Movidius Neural Compute Stick (NCS) is a tiny fanless deep-learning device that can be used to learn AI programming at the edge
- What can you do with a Raspberry PI and a Movidius stick?
  https://developer.movidius.com


Intel® Movidius™
Vision Processing Unit (VPU)



# The Matlab environment (HW+SW)



# Main points

- Intelligent Vision Systems
  - Object detection,
    Segmentation and Recognition
  - Classical, Deep and Hybrid solutions
  - Use cases
- Machine learning Hardware

# Main points

Machine learning: introductions to

- Hardware
- Software and Toolboxes
- Cloud (external resources) – SaaS,
  - Machine Learning as a Service (MLaaS)

Google Cloud

Azure Machine Learning

aws

amazon web services™