Image Classification

Giacomo Boracchi,
DEIB, Politecnico di Milano
October, 21th, 2020

giacomo.boracchi@polimi.it
home.deib.polimi.it/boracchi/

Who I am

Giacomo Boracchi (giacomo.boracchi@polimi.it)

Mathematician (Università Statale degli Studi di Milano 2004),

PhD in Information Technology (DEIB, Politecnico di Milano 2008)

Associate Professor since 2019 at DEIB, Polimi (Computer Science)

My research interests are mathematical and statistical methods for:

- Image analysis and processing
- Machine Learning and in particular unsupervised learning, change and anomaly detection

… and the two combined

Materials

https://boracchi.faculty.polimi.it/teaching/AN2DL.htm

What is image classification (and computer vision) about

Computer Vision

An interdisciplinary scientific field that deals with **how computers** can be made to **gain high-level understanding from digital images or videos**

… which has grown incredibly fast in the last years

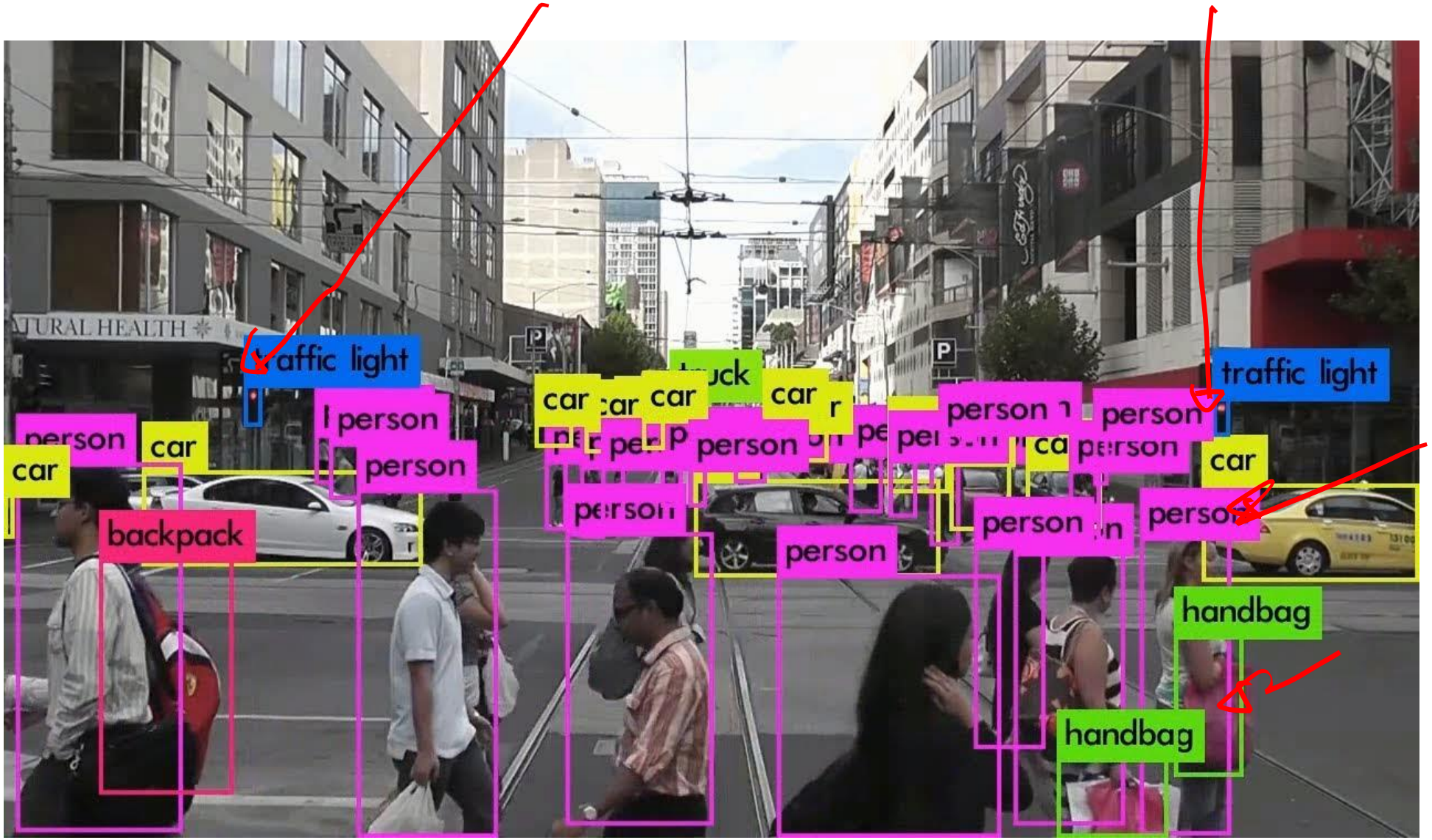
Computer Vision

An interdisciplinary scientific field that deals with **how computers** can be made to **gain high-level**

understanding from digital images or videos

… which has grown incredibly fast in the last years

Object Detection



Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767.

Pose Estimation

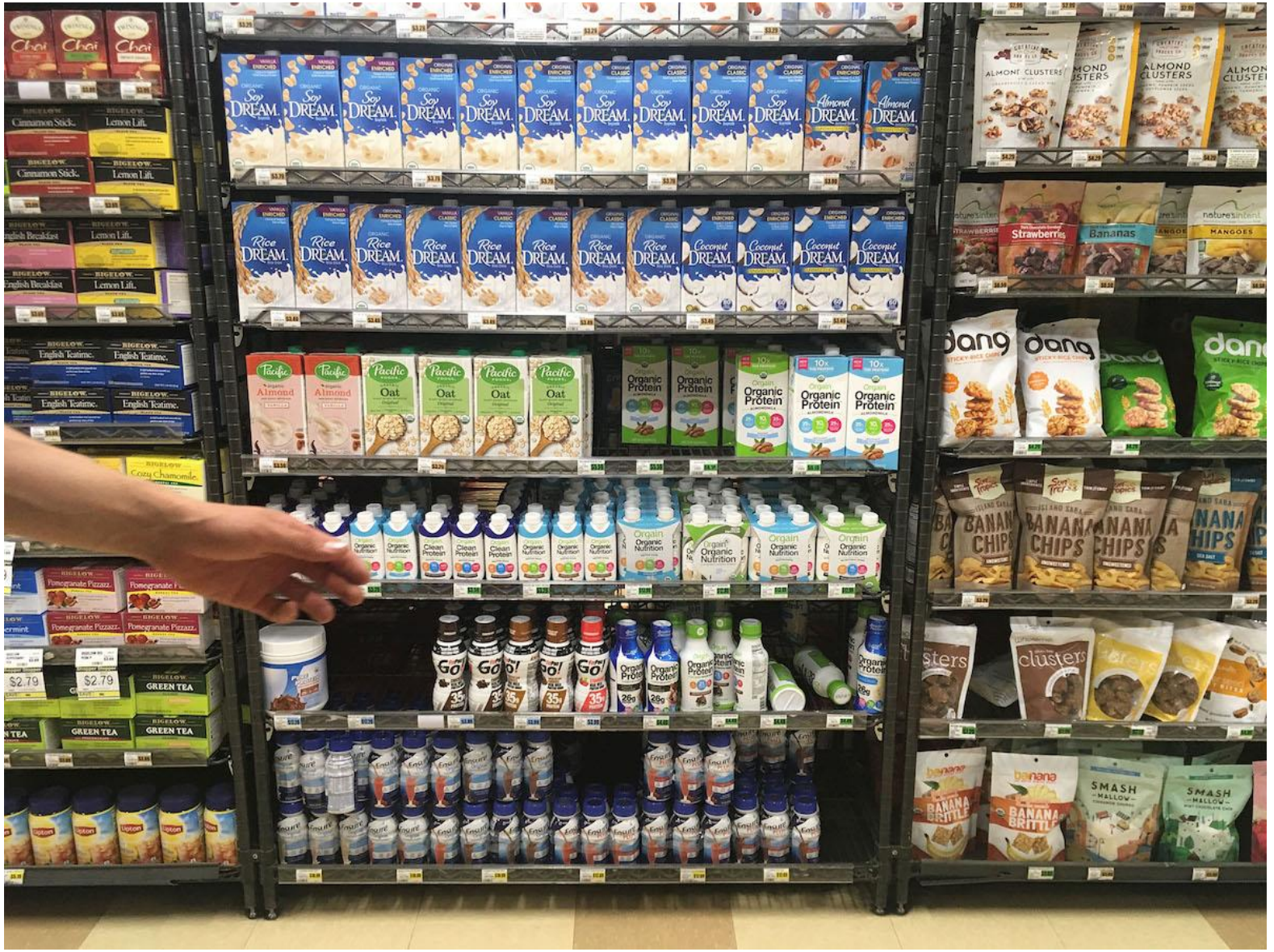
Cao, Z., Simon, T., Wei, S. E., & Sheikh, Y. (2017). Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 7291-7299).

Autonomous Driving

Ian Maddox [CC BY-SA (https://creativecommons.org/licenses/by-sa/4.0)]

By Dllu - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=64517567

Automatic Shelf Analysis



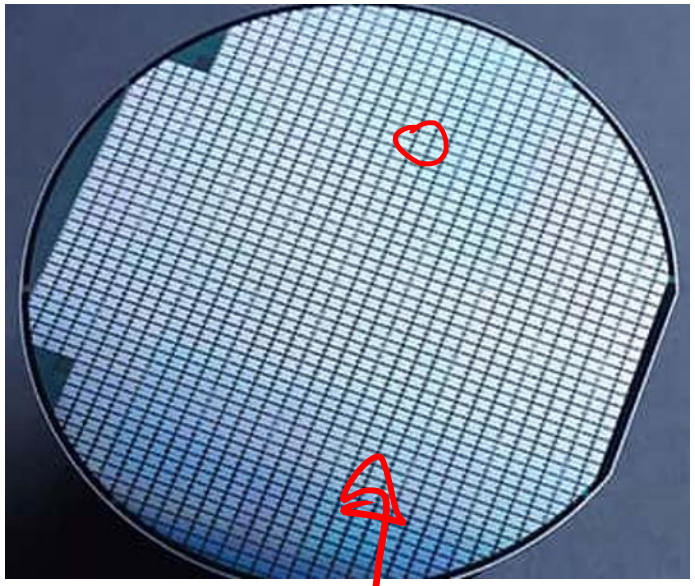
http:focal.systems

Automatic Shelf Analysis

http:focal.systems

Quality Inspection

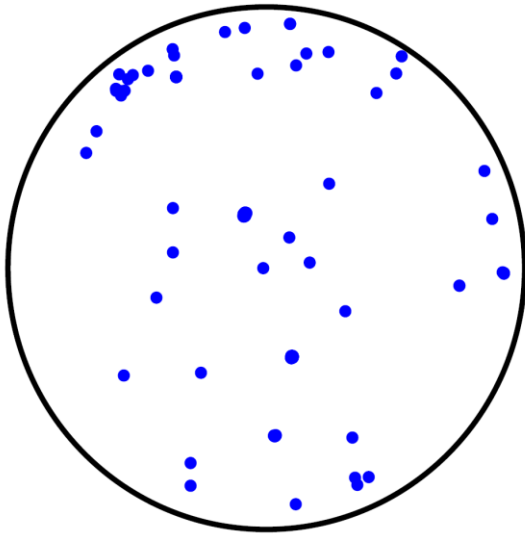
Wafer Defect Map (WDM)

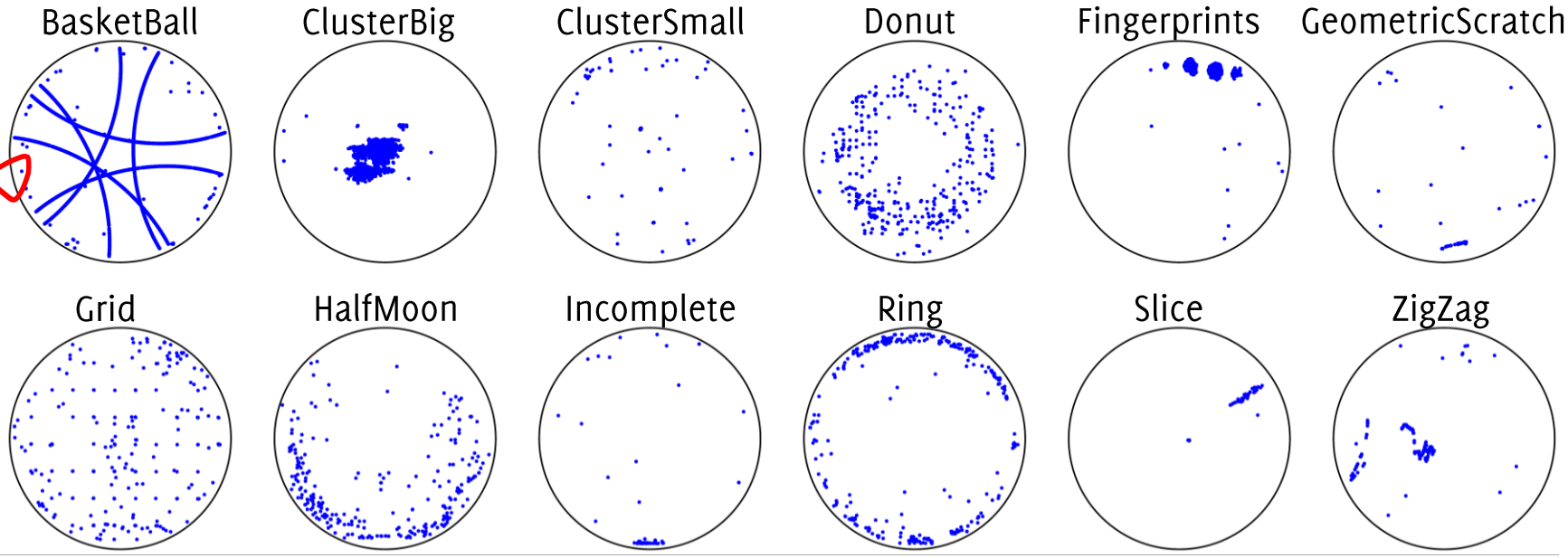


Silicon Wafer

Inspection Tool

X	Y
1863	709
1346	3067
2858	17095
3392	3508
...	...
282	6532
892	18888
4427	9873





Di Bella, Carrera, Rossi, Fragneto, Boracchi *Wafer Defect Map Classification Using Sparse Convolutional Neural Networks* ICIAP09

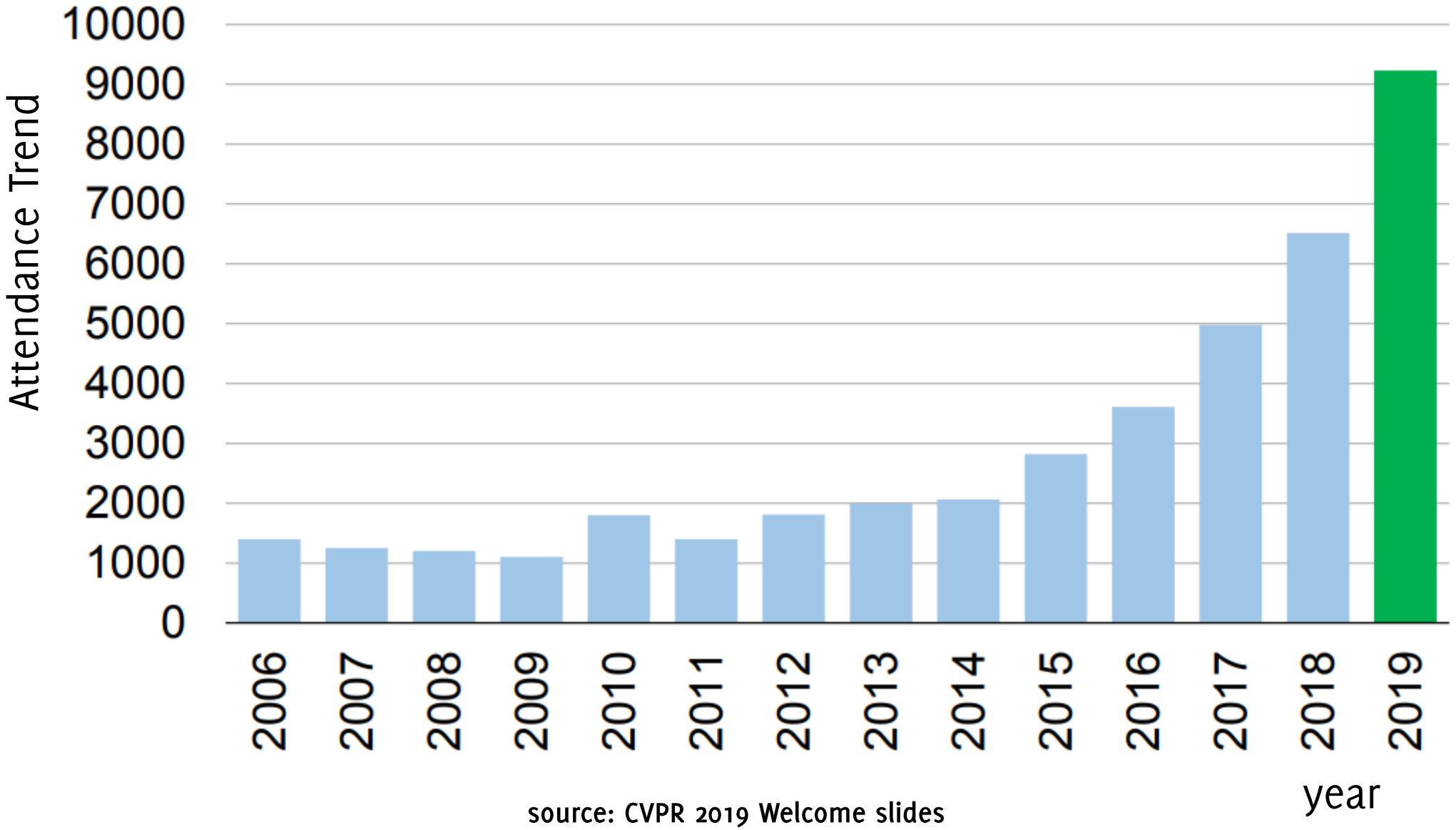
Image Captioning

"little girl is eating piece of cake."

"black cat is sitting on top of suitcase."

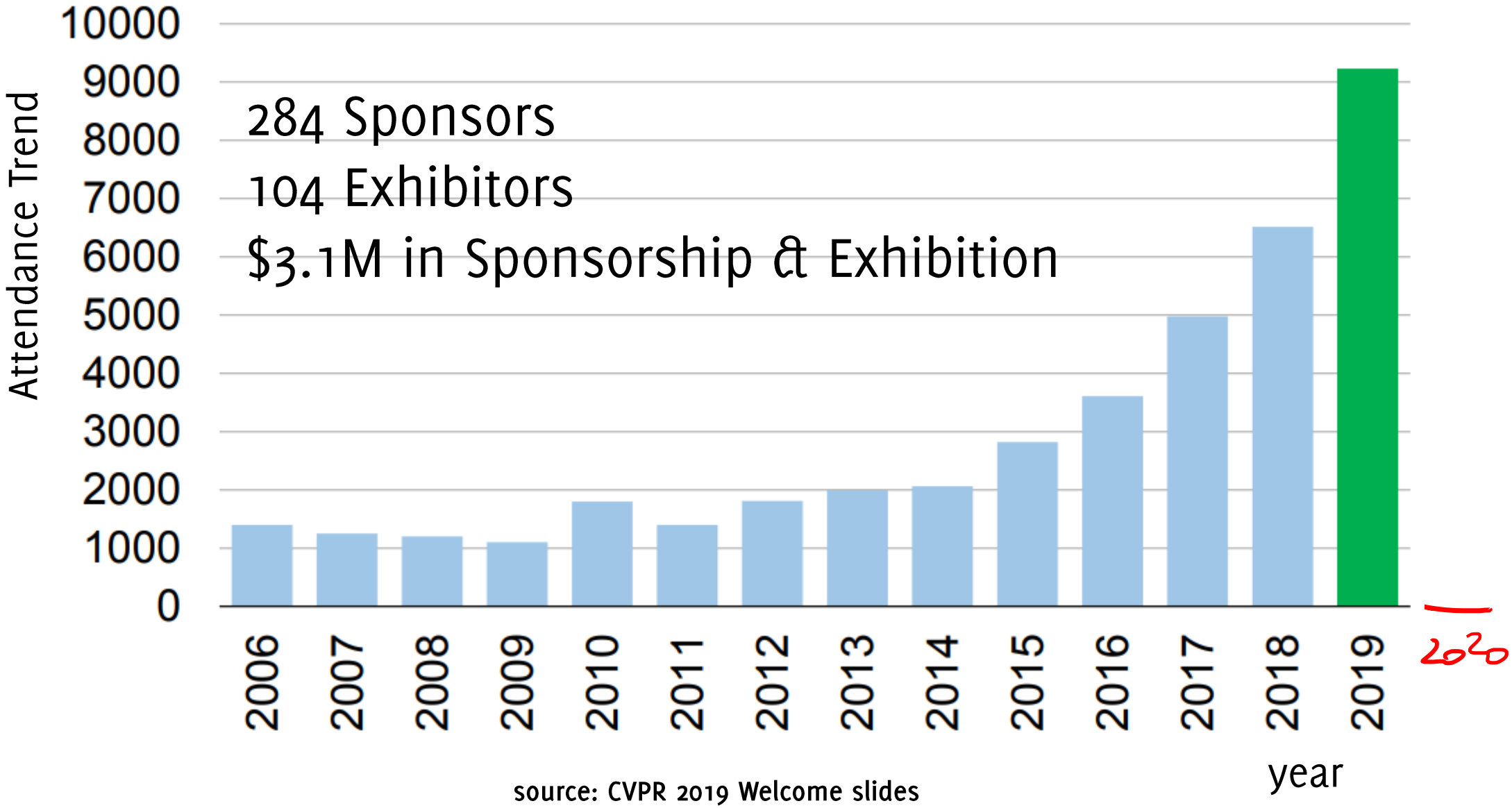
Andrej Karpathy, Li Fei-Fei "Deep Visual-Semantic Alignments for Generating Image Descriptions" CVPR 2015

CVPR



source: CVPR 2019 Welcome slides

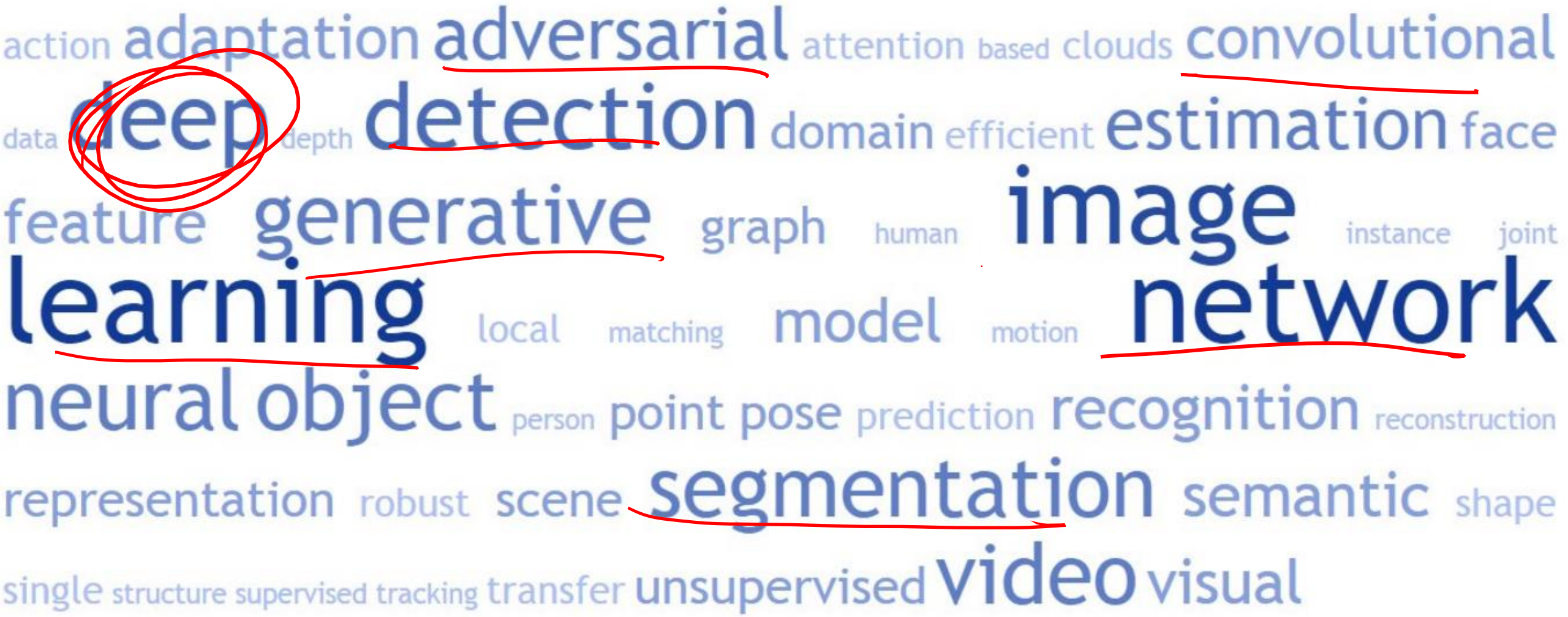
CVPR



source: CVPR 2019 Welcome slides

1,294 papers in CVPR'19 (25.2% acceptance rate)

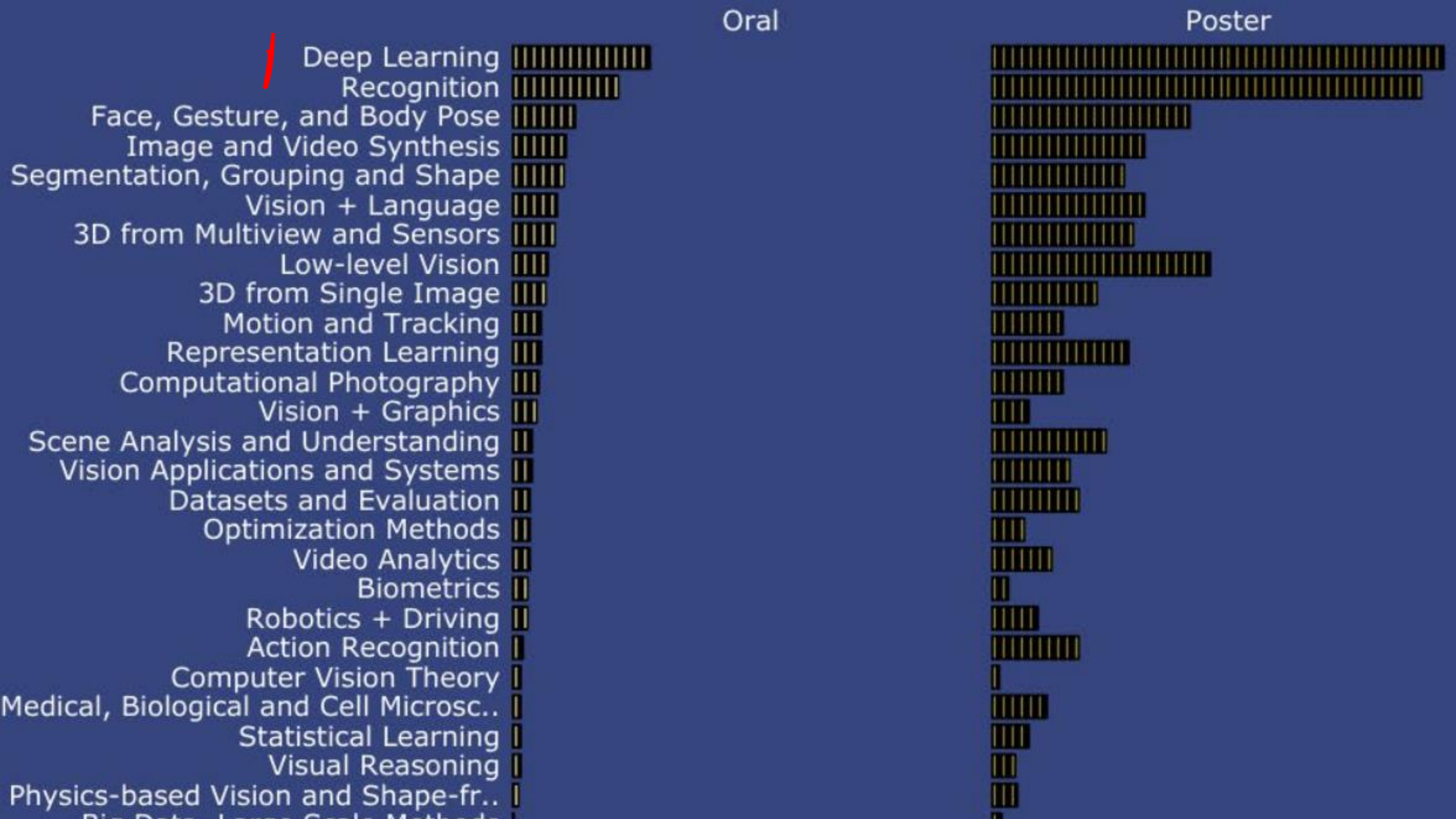
Acceptance rate is roughly even across topics



Lately, connection with ML

There has seen a dramatic change in CV:

- Once, most of techniques and algorithms **build upon a mathematical/statistical** description of **images**
- Nowadays**, machine-learning** methods are much more popular

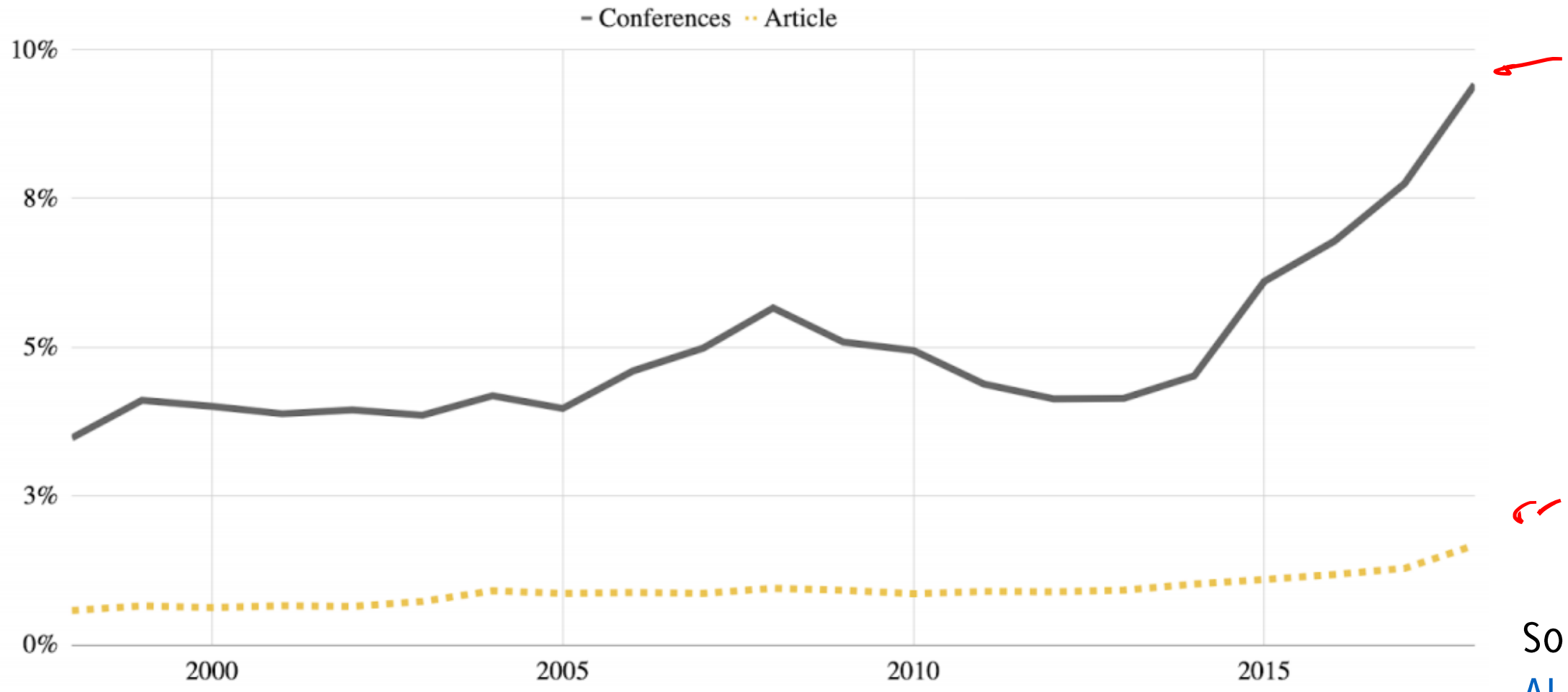

Oral
Poster
Deep Learning
Recognition
Face, Gesture, and Body Pose
Image and Video Synthesis
Segmentation, Grouping and Shape
Vision + Language
3D from Multiview and Sensors
Low-level Vision
3D from Single Image
Motion and Tracking
Representation Learning
Computational Photography
Vision + Graphics
Scene Analysis and Understanding
Vision Applications and Systems
Datasets and Evaluation
Optimization Methods
Video Analytics
Biometrics
Robotics + Driving
Action Recognition
Computer Vision Theory
Medical, Biological and Cell Microsc..
Statistical Learning
Visual Reasoning
Physics-based Vision and Shape-fr..

All in all…

If you plan pursuing a research-oriented career:

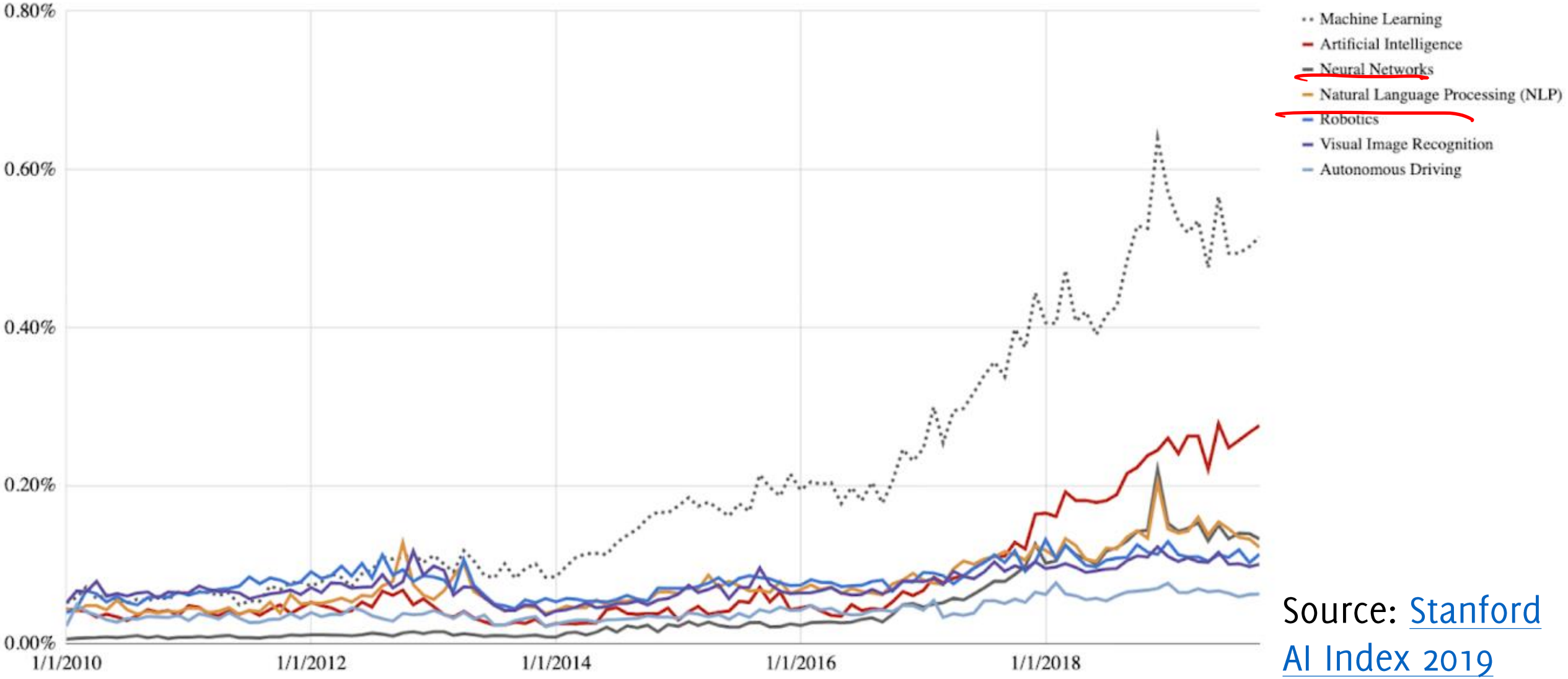
- If you go towards ML and/or CV, consider these fields are increasingly contaminated
- In any case a strong background in Computer Vision and Pattern Recognition is definitively a plus considering the widespread use of imaging data and the need of automation
- This course is a good way to practice more fundamental theory aspects related to neural networks
- Companies are increasinly interested in Computer Vision and Pattern Recogntion appliction

AI publications over all the publications



Source: Stanford AI Index 2019.

Percentages of AI posted job positions



Source: Stanford AI Index 2019

Images

Giacomo Boracchi

Photo Credits: Andrea Sanfilippo

RGB Images

$I \in \mathbb{R}^{R \times C \times 3}$

#rows #cols #channels

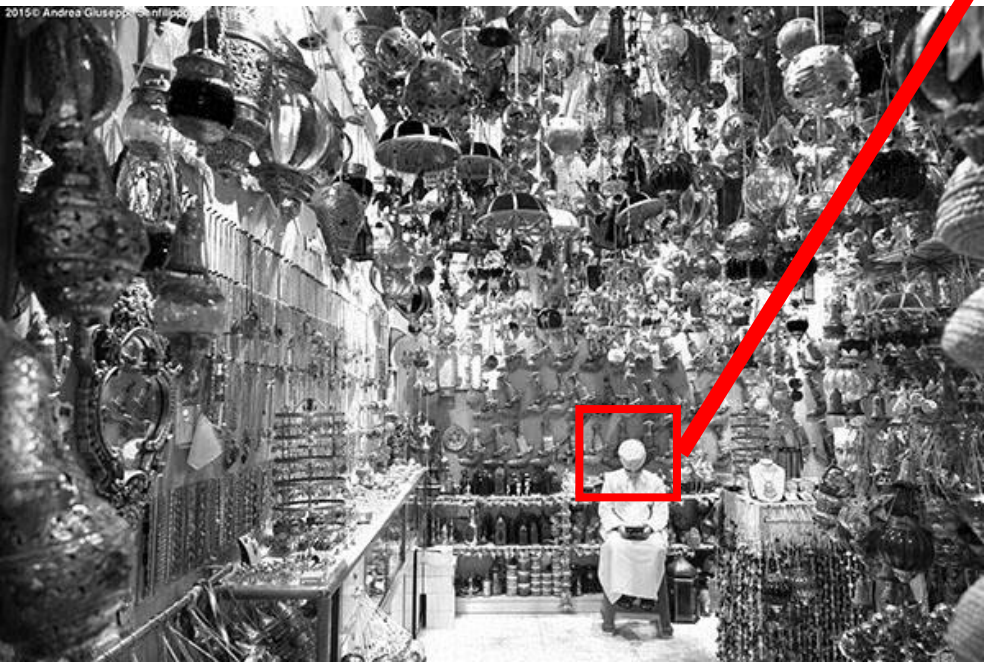
$R \in \mathbb{R}^{R \times C}$

$G \in \mathbb{R}^{R \times C}$

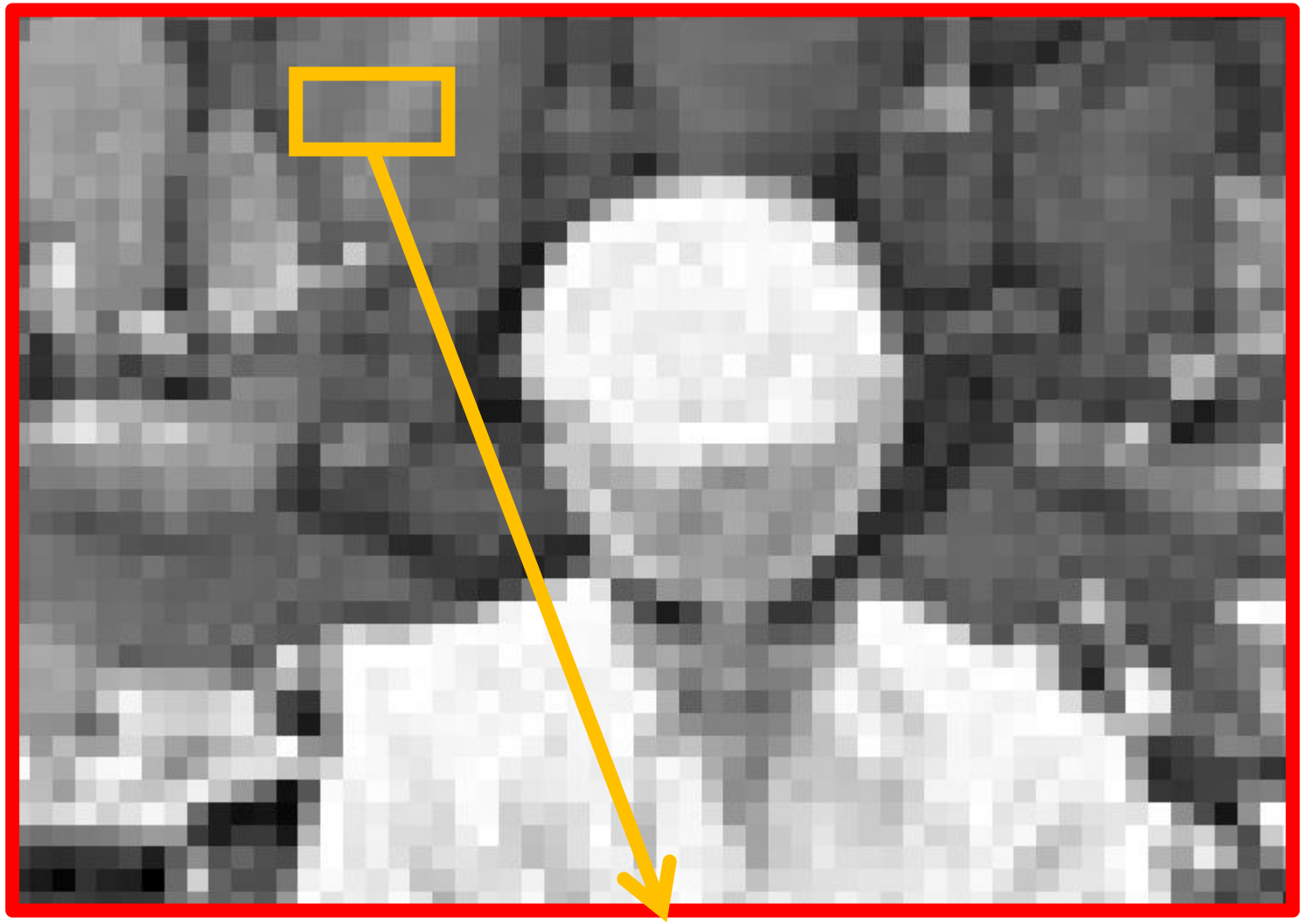
$B \in \mathbb{R}^{R \times C}$

RGB Images

Images are saved by encoding each color information in 8 bits. So images are rescaled and casted in [0,255]



$$R \in \mathbb{R}^{R \times C}$$



123	122	134	121	132
122	121	125	132	124
119	127	137	119	139

1 Byte

RGB Images

[0, 205, 155]

[15, 17, 19]

[230,234, 233]

[253, 5, 6]

[106,124,138]

Giacomo Boracchi

The input of our classifier!

Three dimensional arrays $I \in \mathbb{R}^{R \times C \times 3}$

```
from skimage.io import imread

# Read the image
I = imread('bazar.jpg')

# Extract the color channels of the image
R = I[:, :, 0]
G = I[:, :, 1]
B = I[:, :, 2]
```

Videos

Higher dimensional images

Videos are sequences of images (frames)

If a frame is

$$I \in \mathbb{R}^{R \times C \times 3}$$

a video of T frames is

$$V \in \mathbb{R}^{R \times C \times 3 \times T}$$

```
print(V.shape)
  (144, 180, 3, 30)
```

In this example: $R = 144, C = 180$, thus these 5 color frames contains: 388.800 values in [0,255], thus in principle, 388 KB

Dimension Increases very quickly

Without compression: 1Byte per color per pixel

1 frame in full HD: $R = 1080, C = 1920 \approx 6MB$

1 sec in full HD (24fps) $\approx 150MB$

Fortunately, visual data are very redundant, thus compressible

This has to be taken into account when you design a Machine learning algorithm to be used on images

Higher dimensional images

Multispectral imaging and remote sensing

Giacomo Boracchi

Higher dimensional Images

These images are stacking multiple layers as color-planes

Multi-spectral or Hyper-spectral images:

each band covers a certain wavelength that is meaningful for interpretation soil in areal images

- 0.45-0.52 µm Blue-Green
- 0.52-0.60 µm Green
- 0.63-0.69 µm Red
- 0.76-0.90 µm Near IR
- 1.55-1.75 µm Mid-IR
- 10.40-12.50 µm Thermal IR
- 2.08-2.35 µm Mid-IR

Classification of multispectral images: infer the soil coverage from the vaues in the different spectral bands

In hyperspectral images the number of bands becomes larger and you get a whole energy spectrum in each pixel

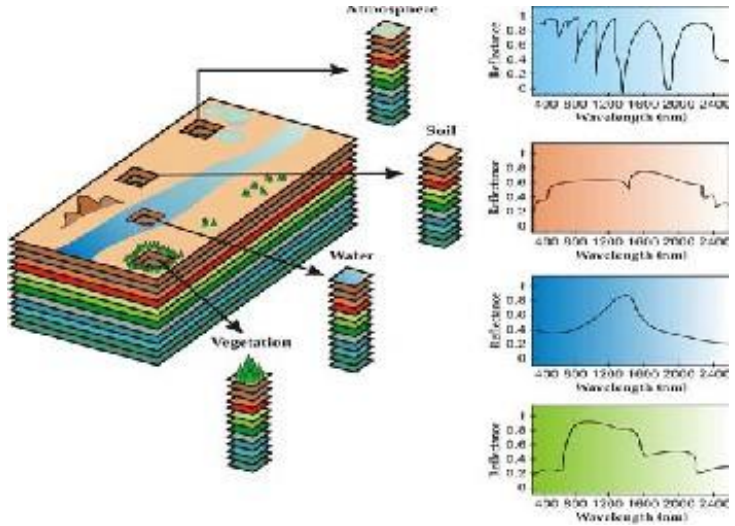
Higher dimensional Images

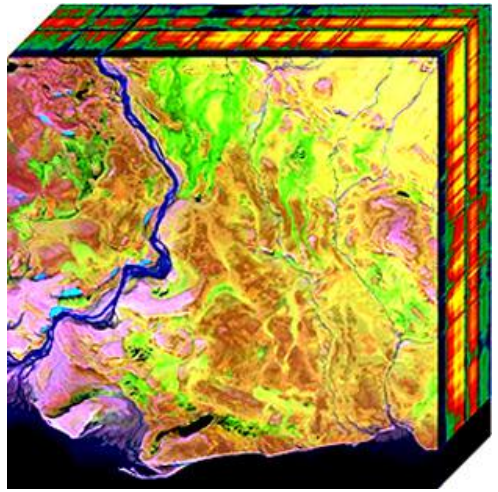
These images are stacking multiple layers as color-planes

Multi-spectral or Hyper-spectral images:

each band covers a certain wavelength that is meaningful for interpretation soil in areal images

- 0.45-0.52 µm Blue-Green
- 0.52-0.60 µm Green
- 0.63-0.69 µm Red
- 0.76-0.90 µm Near IR
- 1.55-1.75 µm Mid-IR
- 10.40-12.50 µm Thermal IR
- 2.08-2.35 µm Mid-IR





In hyperspectral images the number of bands becomes larger and you get a whole energy spectrum in each pixel

Band 1

Giacomo Boracchi

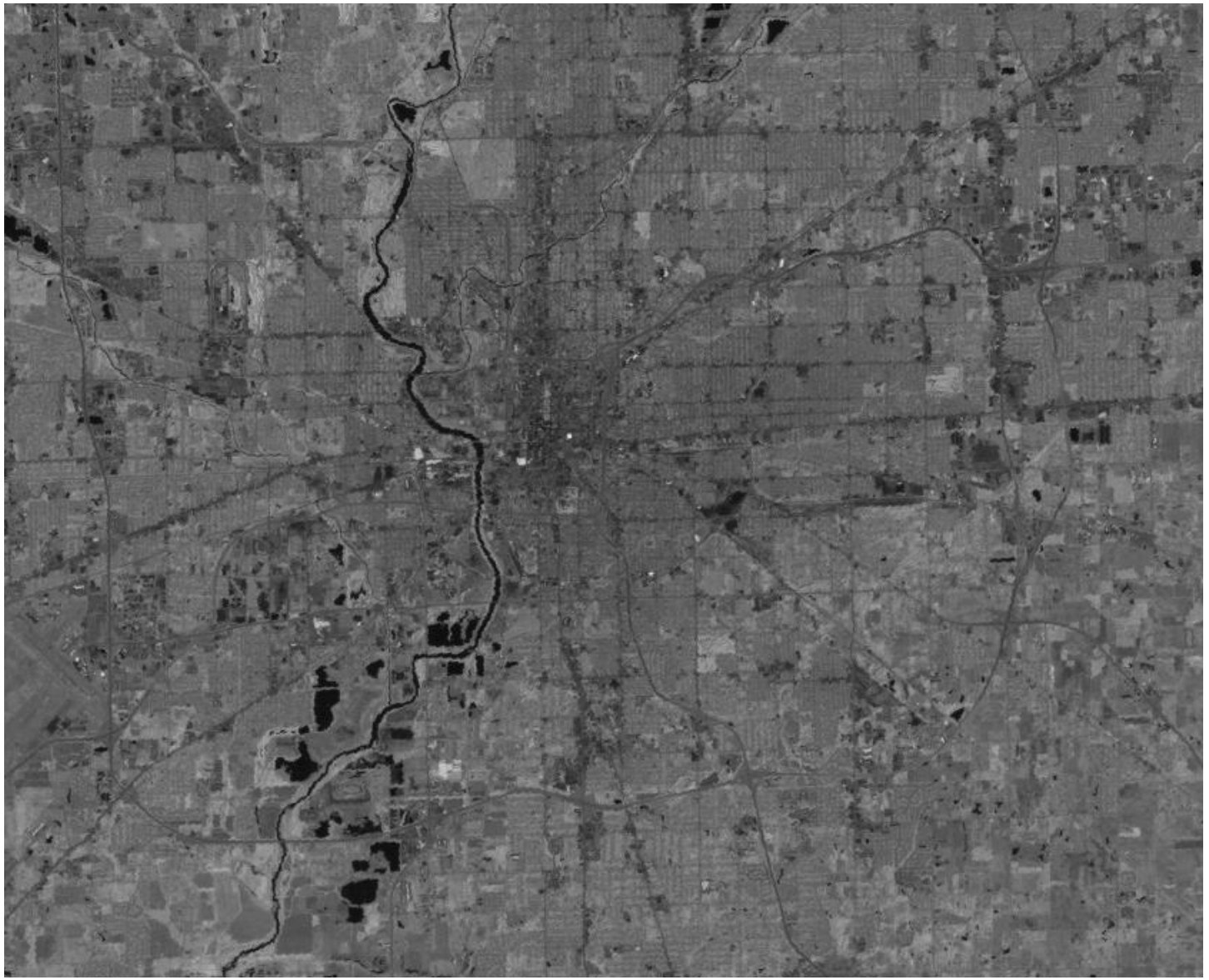
Band 2

Giacomo Boracchi

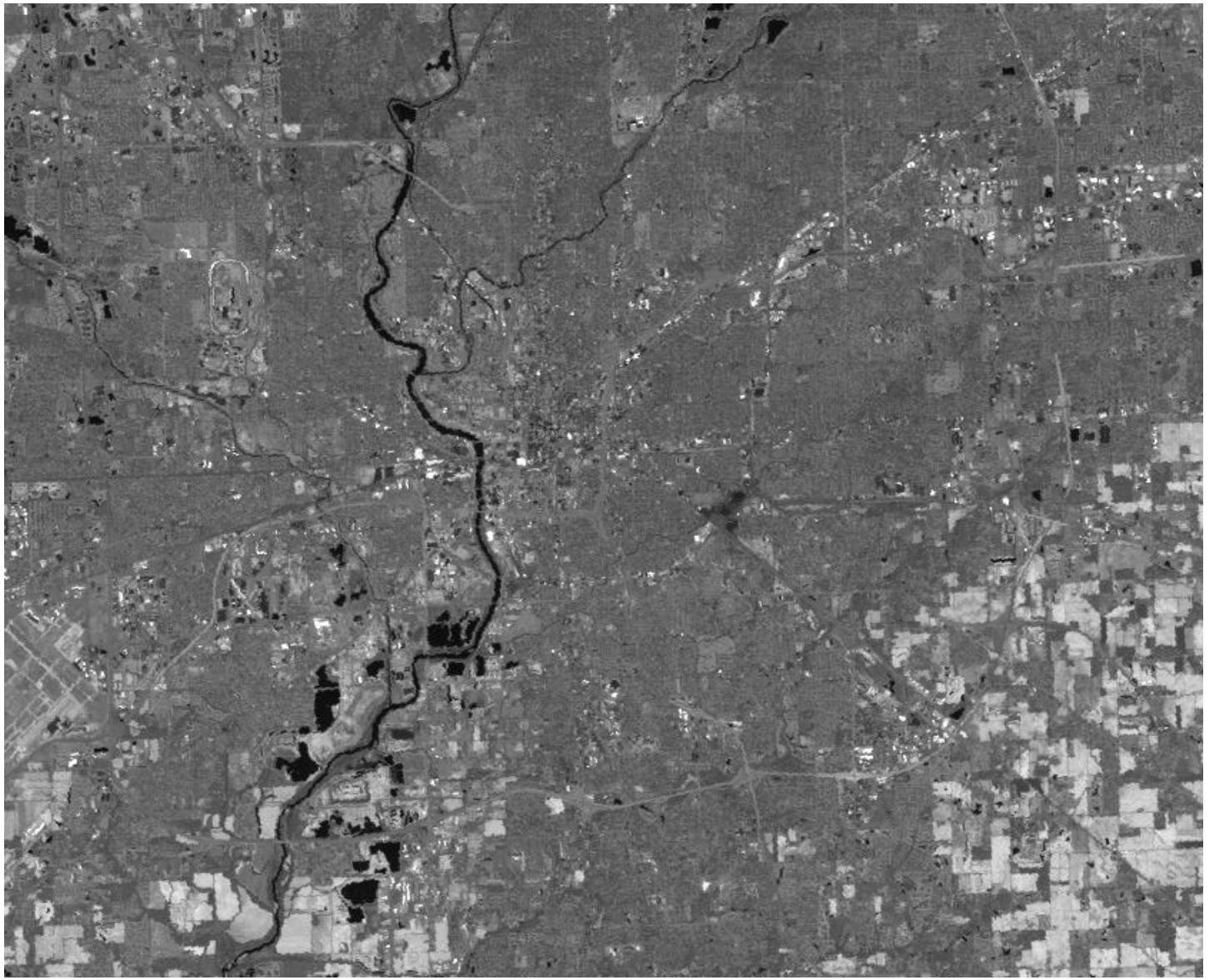
Band 3

Giacomo Boracchi

Band 4



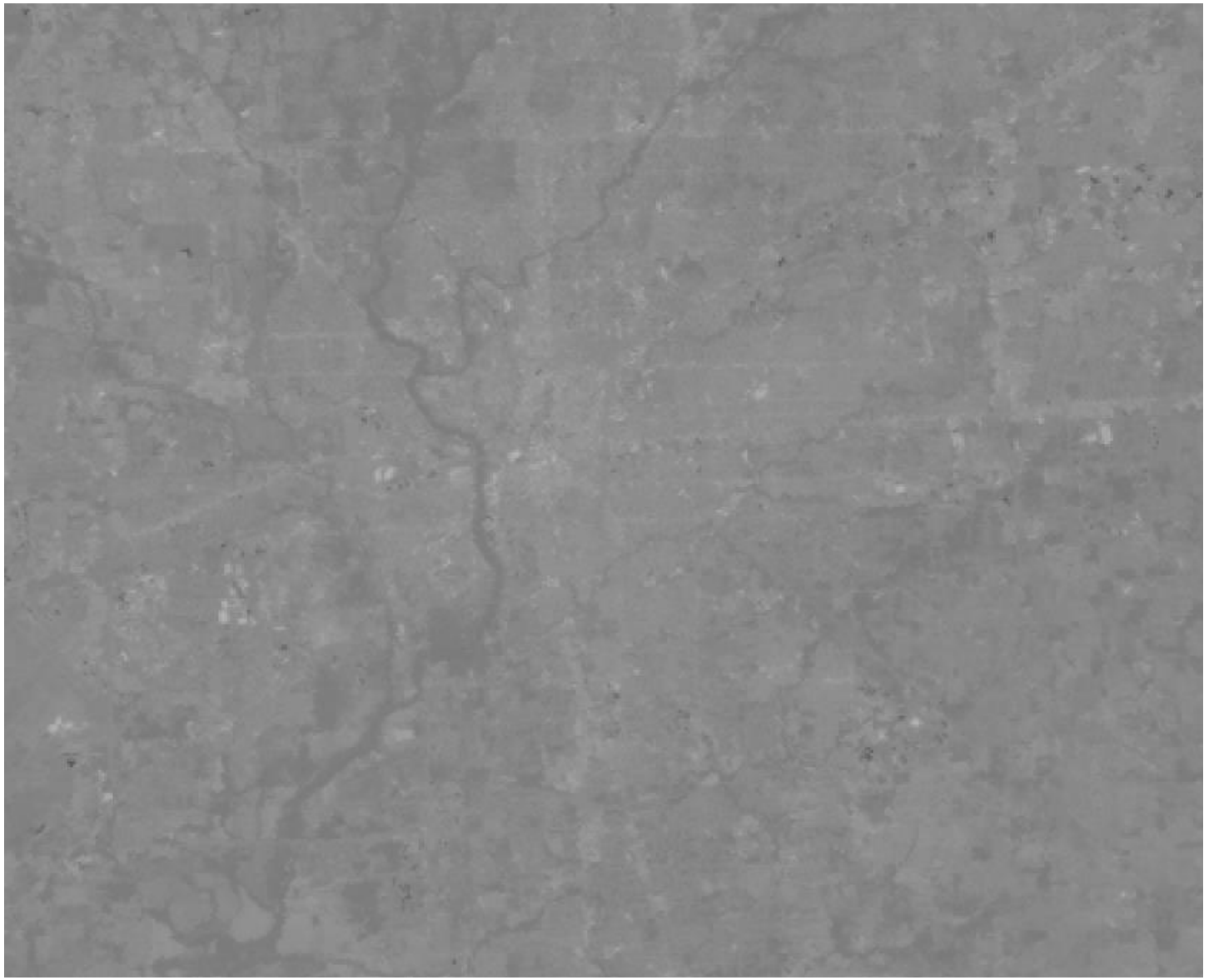
Band 5



Band 6

Giacomo Boracchi

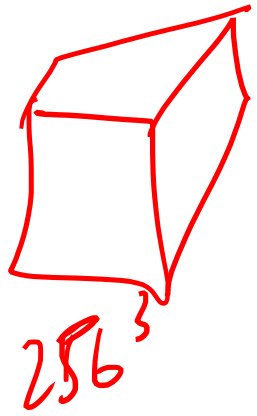
Band 7

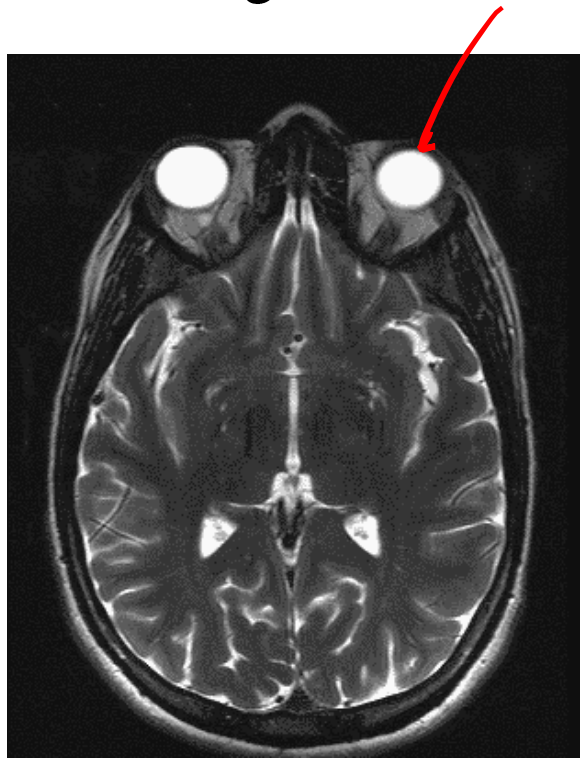


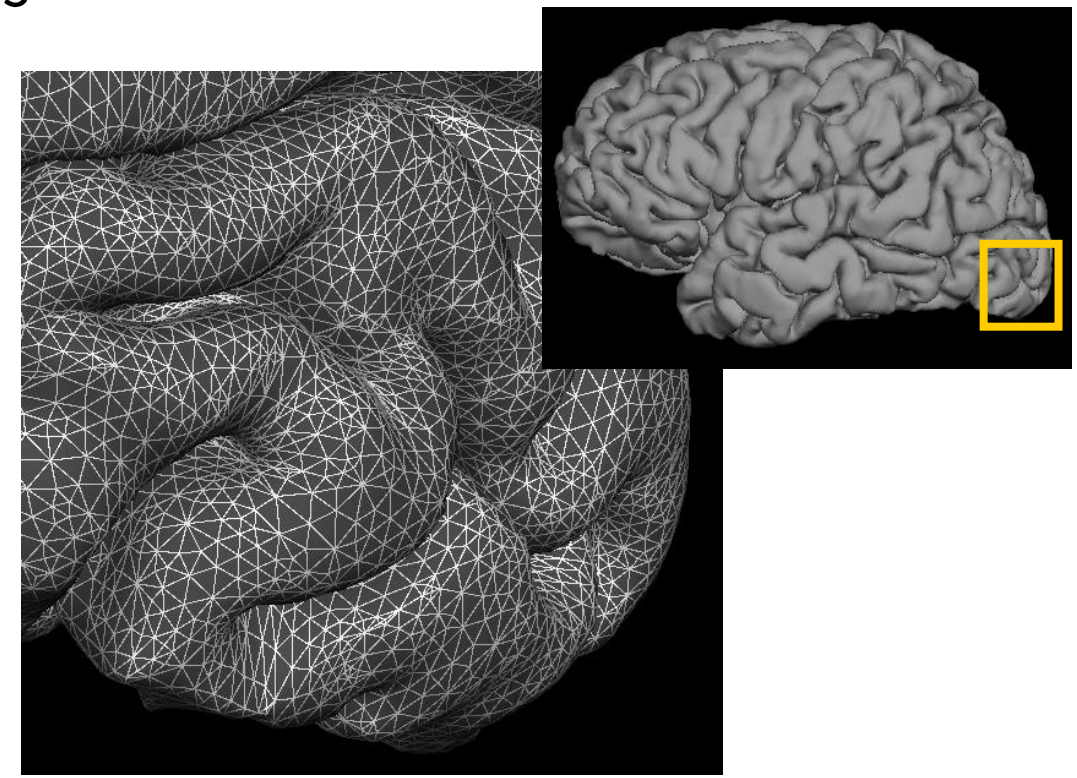
MRI and TAC

(Structural) MRI provide a 3D stack of grayscale images that are analyzing the body at different depth levels

These can be used to perform 3D reconstruction of bones, organs and membranes, e.g. cortical sufaces







Local (Spatial) Transformations: Correlation (and Convolution)

Most important image processing operations for classification problems

Giacomo Boracchi

Local (Spatial) Transformation

In general, these can be written as

$$G(r,c) = T_U[I(r,c)]$$

Where

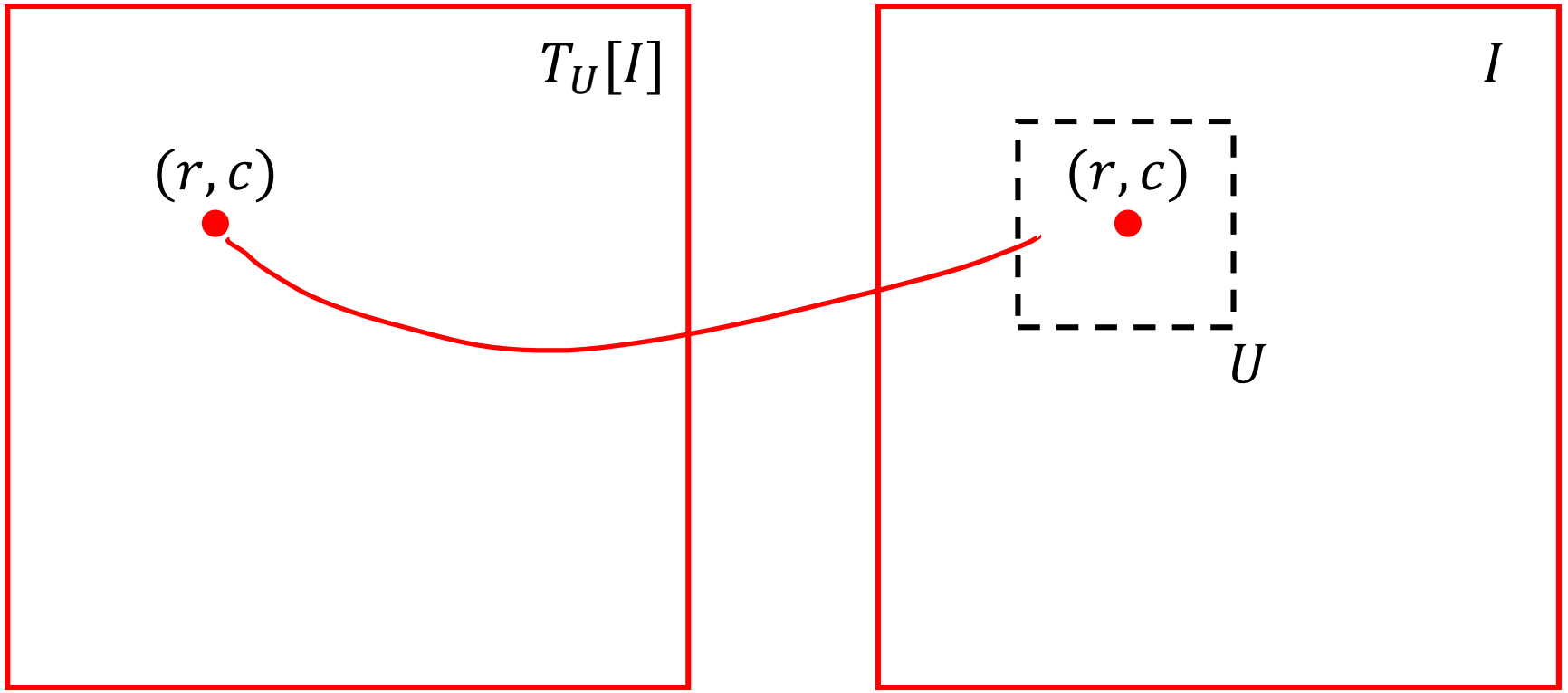
- I is the input image to be transformed
- G is the output
- T_U: $\mathbb{R}^3 \to \mathbb{R}^3$ or T_U: $\mathbb{R}^3 \to \mathbb{R}$ is a function
- U is a neighbourhood, identifies a region of the image that will concur in the output definition

T operates on I "around" U

The output at pixel (r,c) i.e., $T_U[I(r,c)]$ is defined by all pixels $\{I(x,y),\ \ (x-r, y-c) \in U\}$

Local (Spatial) Filters

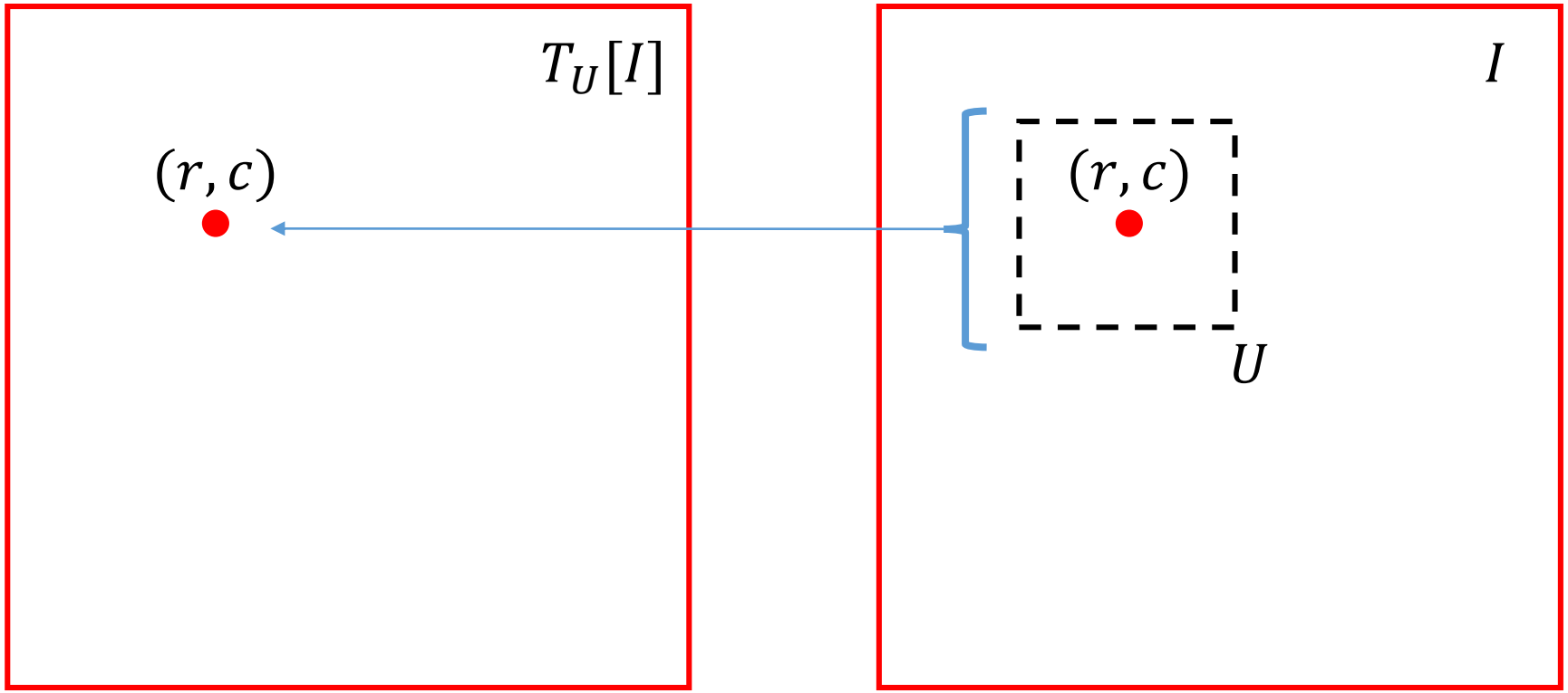
The dashed square represents $\{I(x,y),\ \ (x-r, y-c) \in U\}$



- The location of the output does not change
- The operation is repeated for each pixel
- T can be either linear or nonlinear

Local (Spatial) Filters

The dashed square represents $\{I(x,y),\ \ (x-r,y-c) \in U\}$



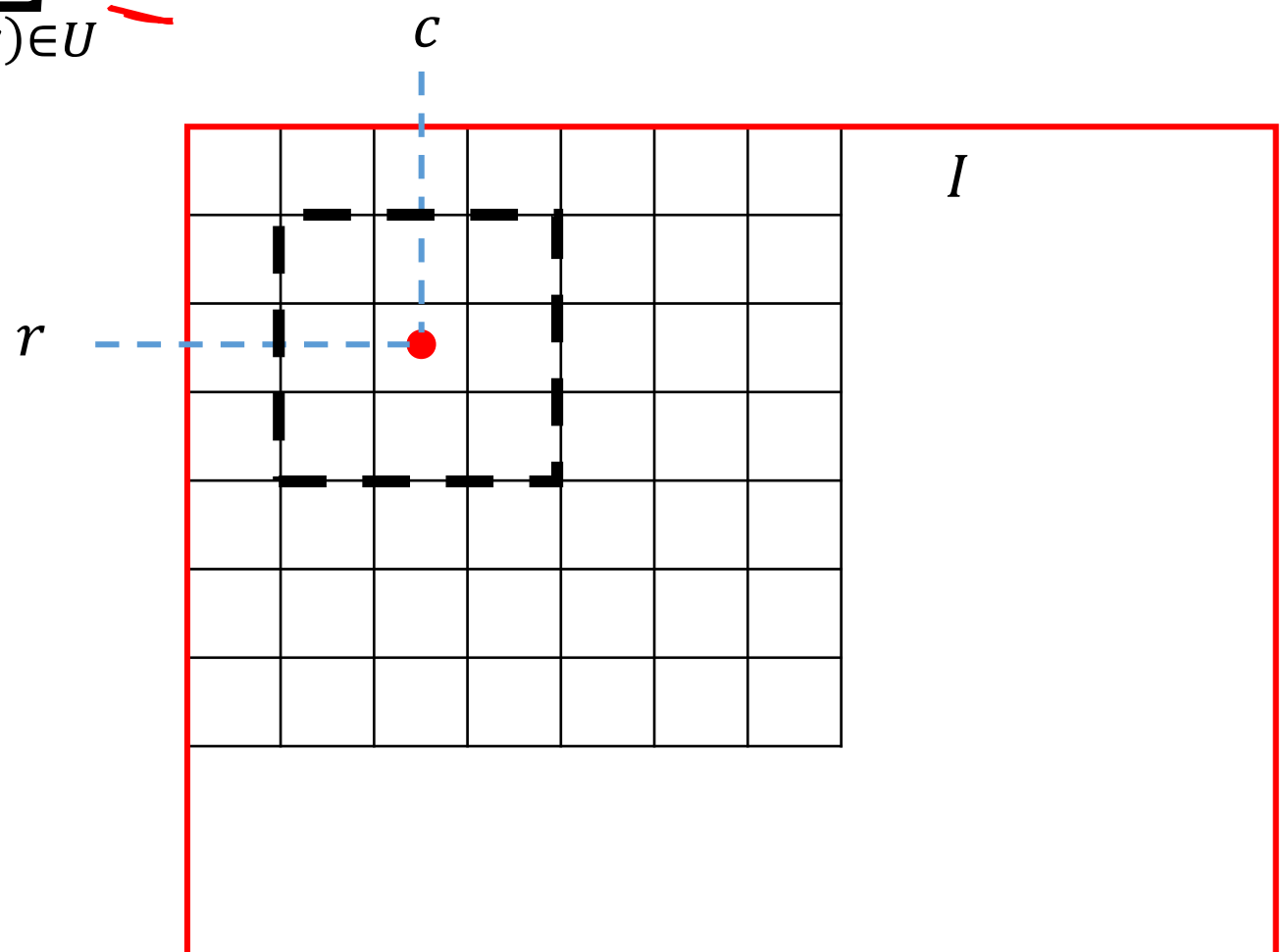
- The location of the output does not change
- The operation is repeated for each pixel
- T can be either linear or nonlinear

Local Linear Filters

Linear Transformation: Linearity implies that

$$T\big(I(r,c)\big) = \sum_{(x,y)\in U} w_i * I(r+x, c+y)$$

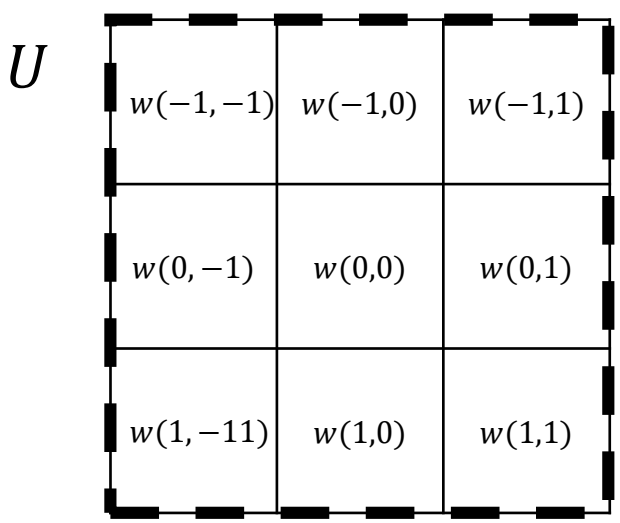
Considering some weights $\{w_i\}$



Local Linear Filters

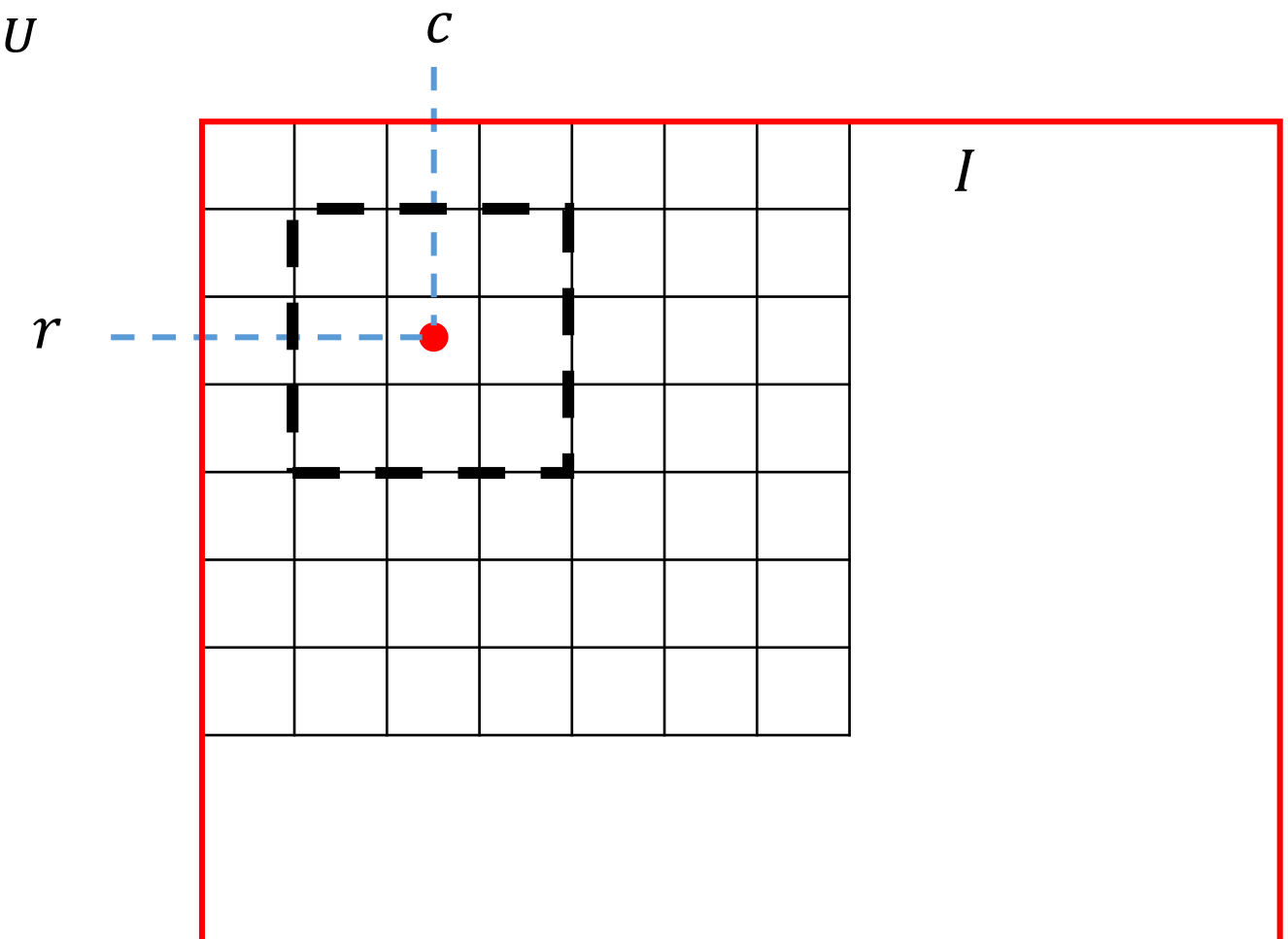
Linear Transformation: Linearity implies that

$$T\big(I(r,c)\big) = \sum_{(x,y)\in U} w(x,y) * I(r+x, c+y)$$



We can consider weights as an image, or **a filter** h

The filter h entirely defines this operation



Local Linear Filters

Linear Transformation: Linearity implies that

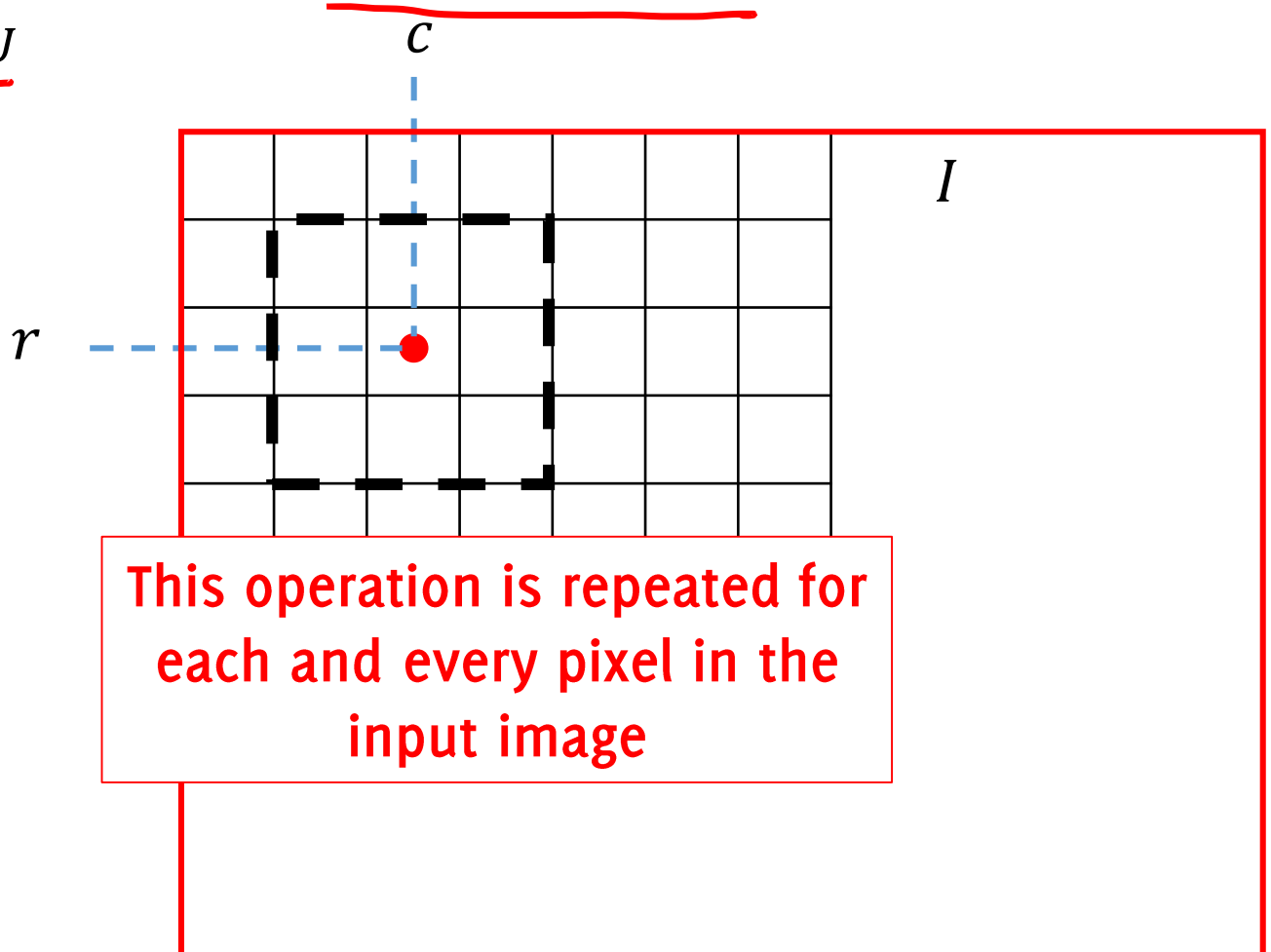
$$T(I(r,c)) = \sum_{(x,y)\in U} w(x,y) * I(r+x, c+y)$$

U

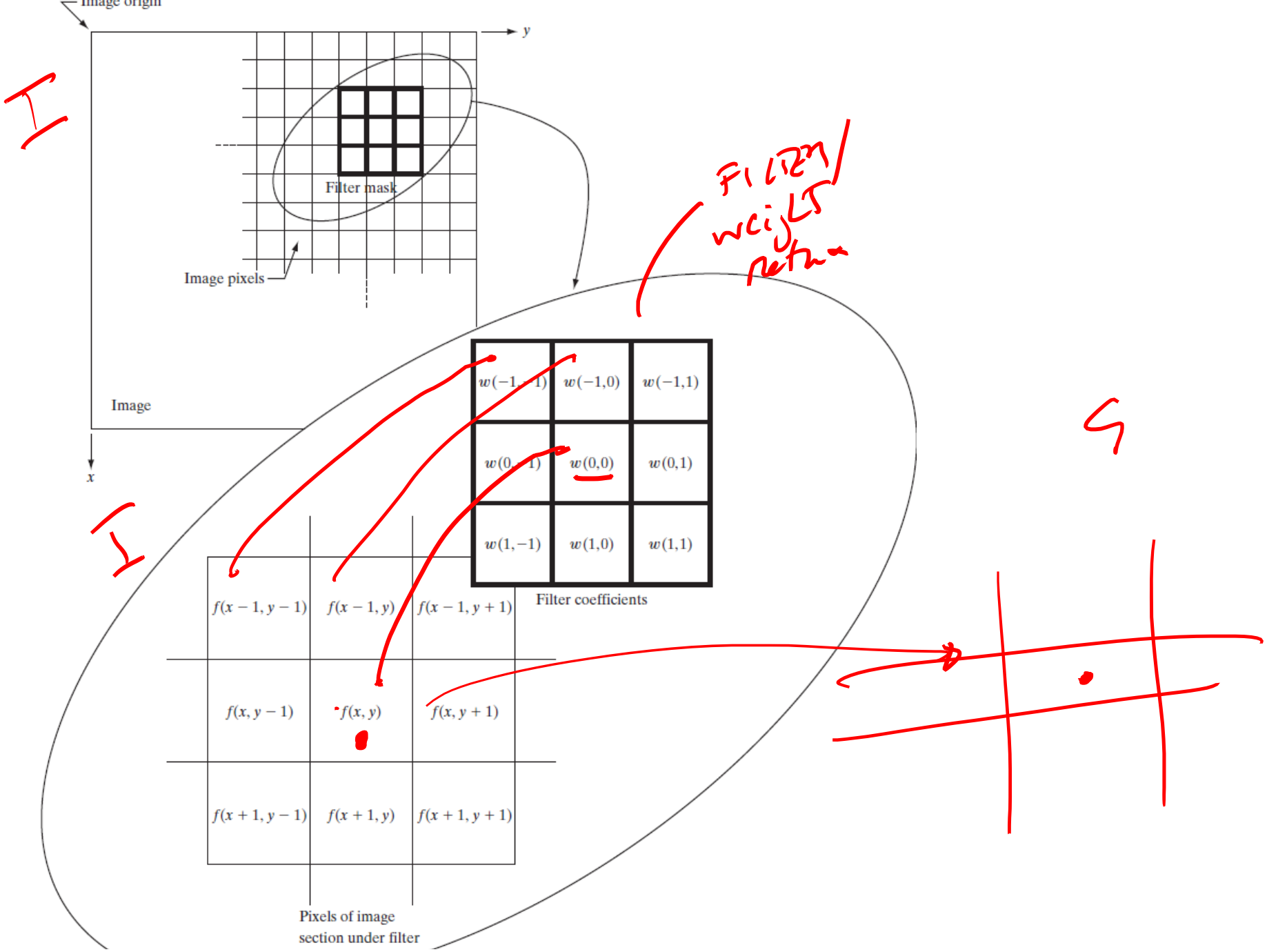
$w(-1,-1)$	$w(-1,0)$	$w(-1,1)$
$w(0,-1)$	$w(0,0)$	$w(0,1)$
$w(1,-11)$	$w(1,0)$	$w(1,1)$

We can consider weights as an image, or **a filter** h

The filter h entirely defines this operation



This operation is repeated for each and every pixel in the input image



Gonzalez and Woods «Digital image Processing», Prentice Hall;, 3° edition

Correlation

The **correlation** among a filter h and an image is defined as

$$(I \otimes h) = \sum_{u=-L}^{L} \sum_{v=-L}^{L} w(u,v) * I(r+u, c+v)$$

where the filter w is of size $(2L+1) \times (2L+1)$

Correlation

The **correlation** among a filter h and an image is defined as

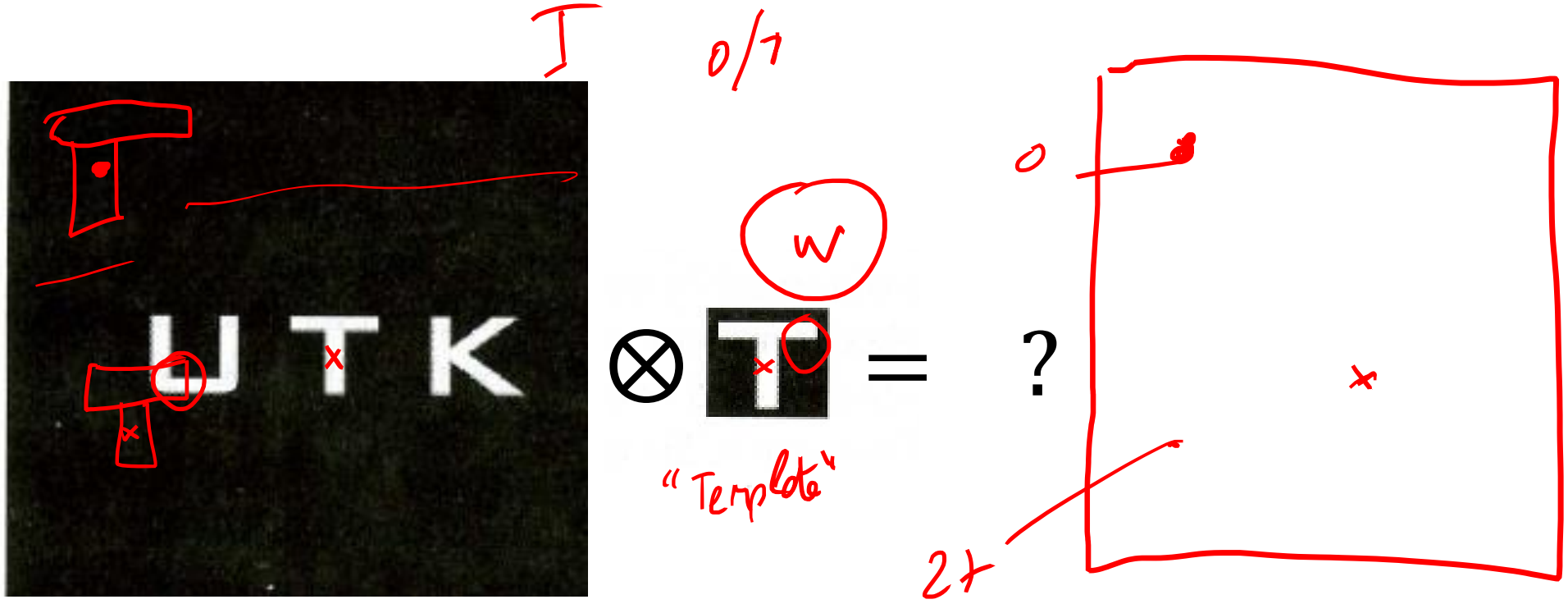
$$(I \otimes h) = \sum_{u=-L}^{L} \sum_{v=-L}^{L} w(u,v) * I(r+u, c+v)$$

where the filter w is of size $(2L+1) \times (2L+1)$

```
from scipy import signal


# correlation
T_corr = signal.correlate(I, h)
```

Correlation for BINARY target matching

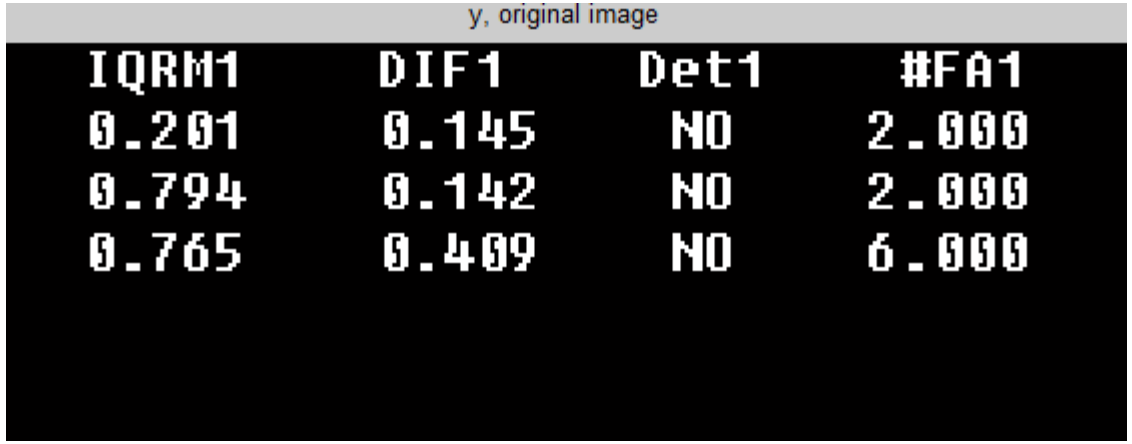


Easy to understand with binary images

Target used as a filter

Gonzalez and Woods «Digital image Processing», Prentice Hall;, 3° edition

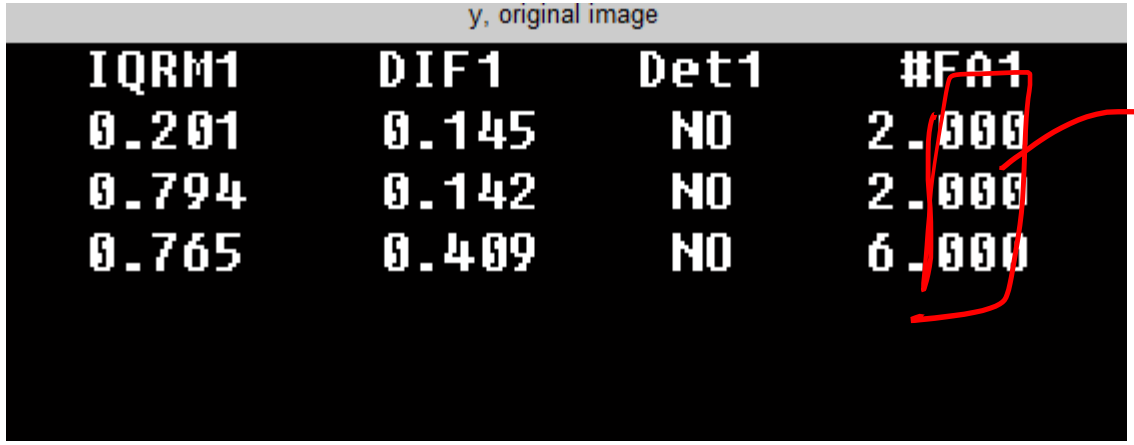
Another example



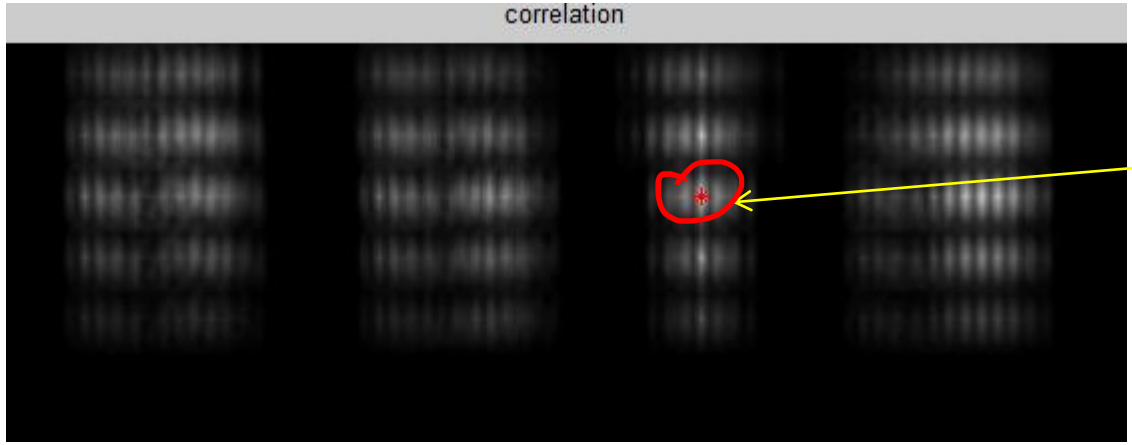
*

=

Another example

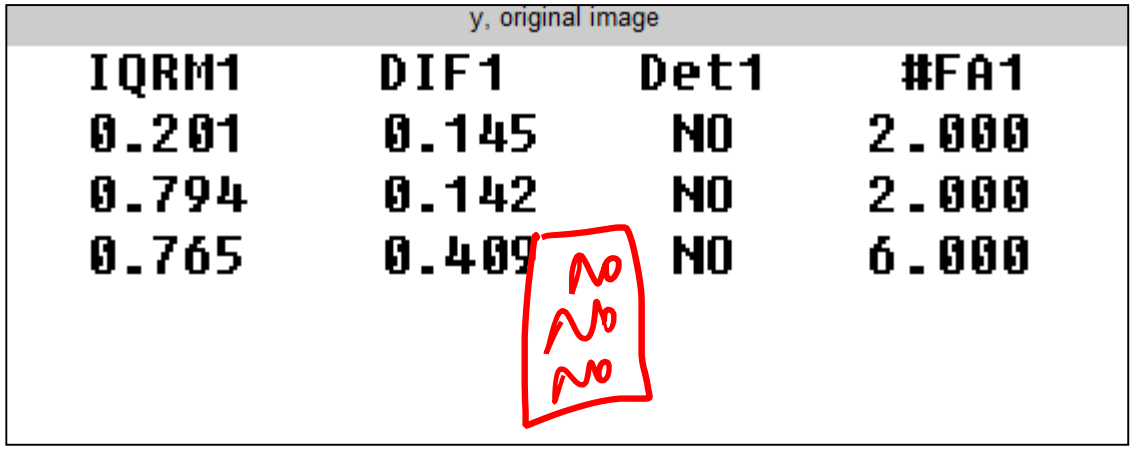


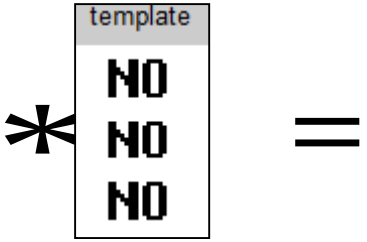
* =



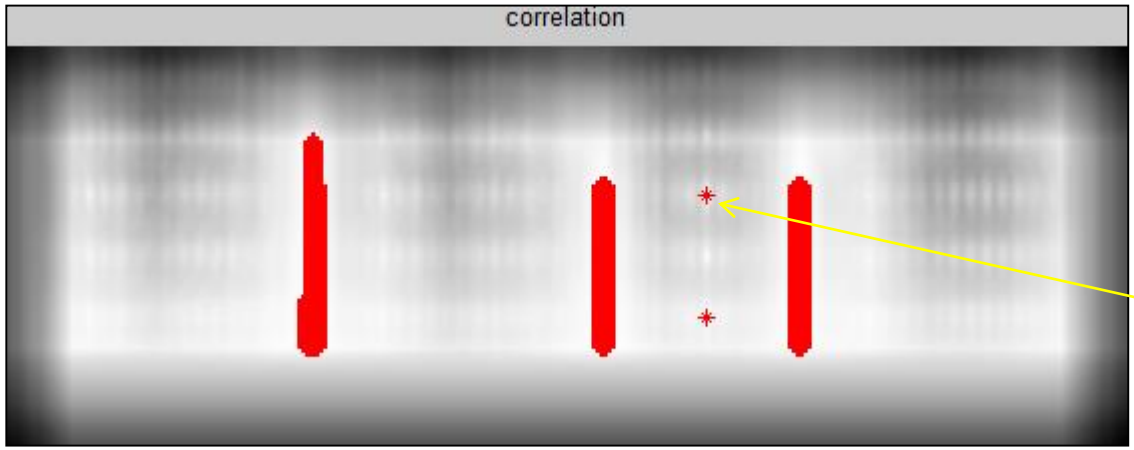
The maximum is here

However…





=



Each point in a white area is as big as the template achieve the maxium value (togheter with the perfect match)

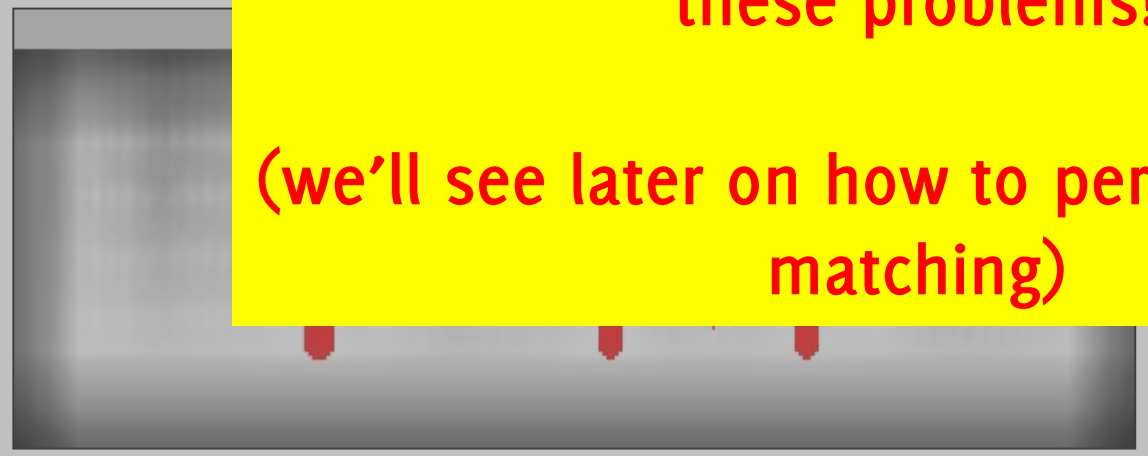
However…

y, original image

IQRM1	DIF1	Det1	#FA1
0.201	0.145	NO	2.000
0.794	0.142	NO	2.000
0.765	0.409	NO	6.000

template

NO
NO
NO



Normalization is needed to prevent these problems!

(we'll see later on how to perform template matching)

a white area is
template
achieve the maxium value
(togheter with the perfect match)

Setting up the stage..

Image Classification

$\Lambda = \{\text{"wheel"}, \text{"cars"}, ..., \text{"castle"}, \text{"baboon"}\}$

I

⇨ “wheel”

I

⇨ “castle”

Image Classification

$\Lambda = \{\text{"wheel"}, \text{"cars"}, ..., \text{"castle"}, \text{"baboon"}\}$

I

⇨ “wheel” 65%, “tyre” 30%..

I

⇨ “castle” 55%, “tower” 43%..

Image Classification, the problem

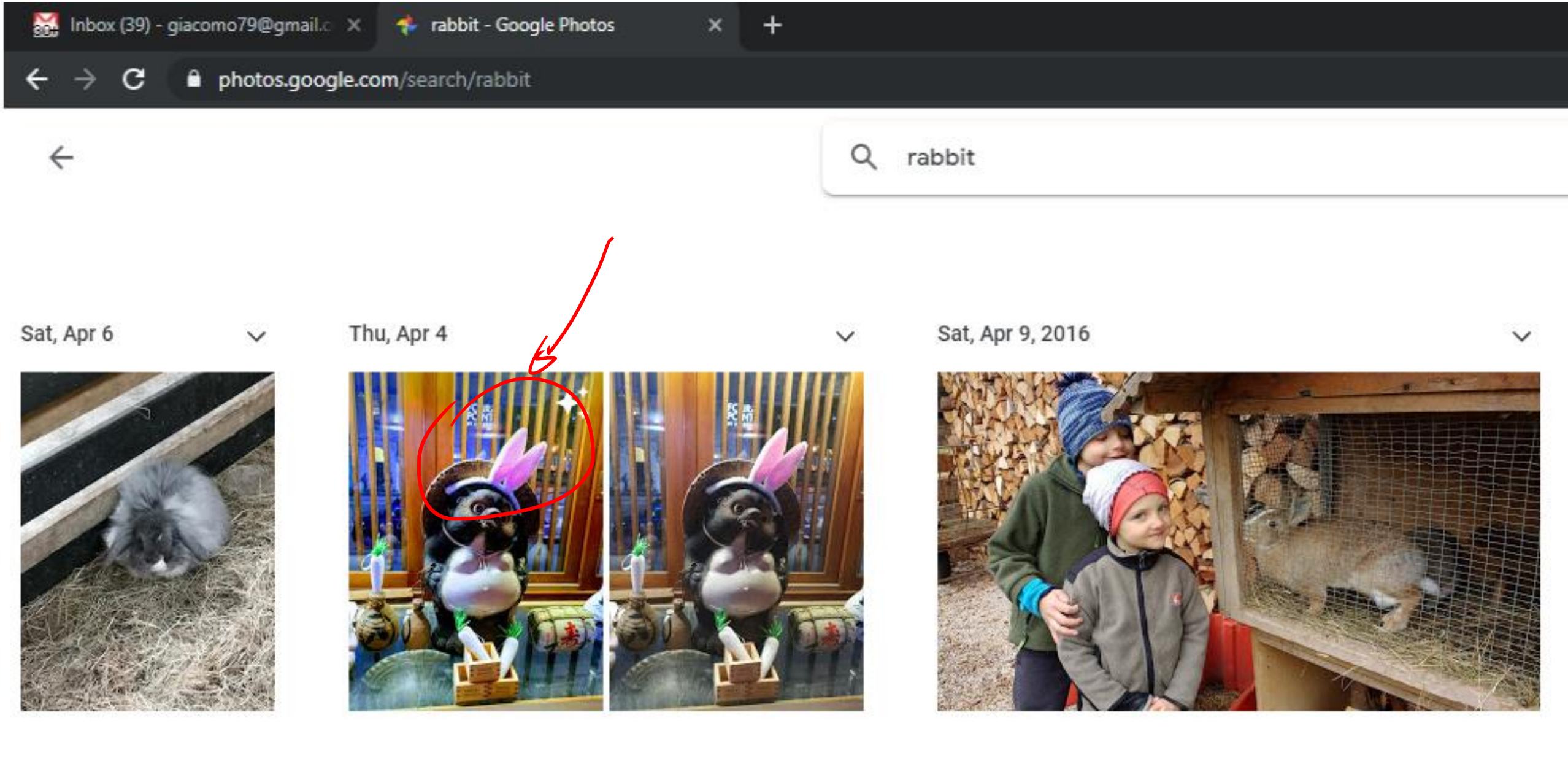
Assign to an input image $I \in \mathbb{R}^{R \times C \times 3}$:

- a label l from a fixed set of categories

$$\Lambda = \{\text{"wheel"}, \text{"cars"}, \ldots, \text{"castle"}, \text{"baboon"}\}$$

$$I \rightarrow l \in \Lambda$$

Image Classification Example



Localization

Localization, the problem

Assign to an input image $I \in \mathbb{R}^{R \times C \times 3}$:

- a label l from a fixed set of categories
$$\Lambda = \{\text{"wheel"}, \text{"cars"}, ..., \text{"castle"}, \text{"baboon"}\}$$
- the coordinates (x, y, h, w) of the bounding box enclosing that object

$$I \rightarrow (x, y, h, w, l)$$

Object Detection

Object Detection, the problem

Assign to an input image $I \in \mathbb{R}^{R \times C \times 3}$:

- **multiple** labels $\{l_i\}$ from a fixed set of categories $\Lambda = \{$"wheel", "cars", ..., "castle", "baboon"$\}$, each corresponding **to an instance of that object**
- the coordinates $\{(x, y, h, w)_i\}$ of the bounding box enclosing **each** object

$$I \rightarrow \{(x, y, h, w, l)_1, \dots, (x, y, h, w, l)_N\}$$

Segmentation

Zheng et al. “Conditional Random Fields as Recurrent Neural Networks”, ICCV 2015

Image Segmentation, the problem

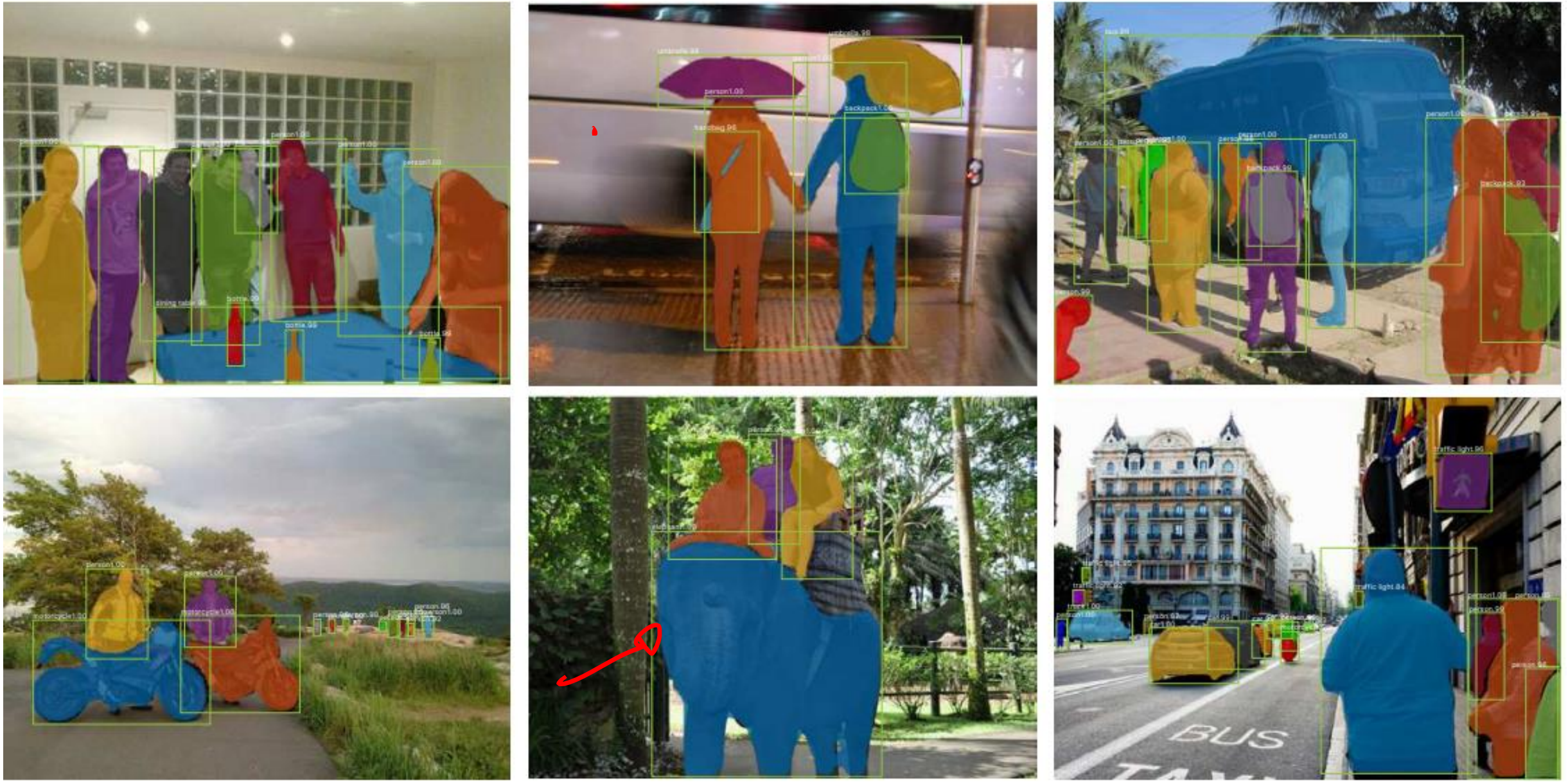
Assign **to each pixel** of an image $I \in \mathbb{R}^{R \times C \times 3}$:

- a label $\{l_i\}$ from a fixed set of categories
  $\Lambda = \{\text{"wheel"}, \text{"cars"}, \ldots, \text{"castle"}, \text{"baboon"}\}$,

$$I \rightarrow S \in \Lambda^{R \times C}$$

where $S(x, y) \in \Lambda$ denotes the class associated to the pixel (x, y)

Instance Segmentation: Mask R-CNN



He, Kaiming, et al. "Mask r-cnn." *CVPR 2017*

Instance Segmentation, the problem

Assign to an input image I:

- **multiple** labels $\{l_i\}$ from a fixed set of categories $\Lambda = \{$"wheel", "cars", ..., "castle", "baboon"$\}$, each corresponding to **an instance of that object**
- the coordinates $\{(x, y, h, w)_i\}$ of the **bounding box enclosing each object**
- the **set of pixels** S in each bounding box corresponding to that label

$$I \rightarrow \{(x, y, h, w, l, S)_1, \dots, (x, y, h, w, l, S)_N\}$$

Instance Segmentation

Giacomo Boracchi

Is Image Classification a Challenging Problem?

Yes, it is...

First challenge: dimensionality

Images are very high-dimensional image data

CIFAR-10 dataset

The CIFAR-10 dataset contains 60000 images:

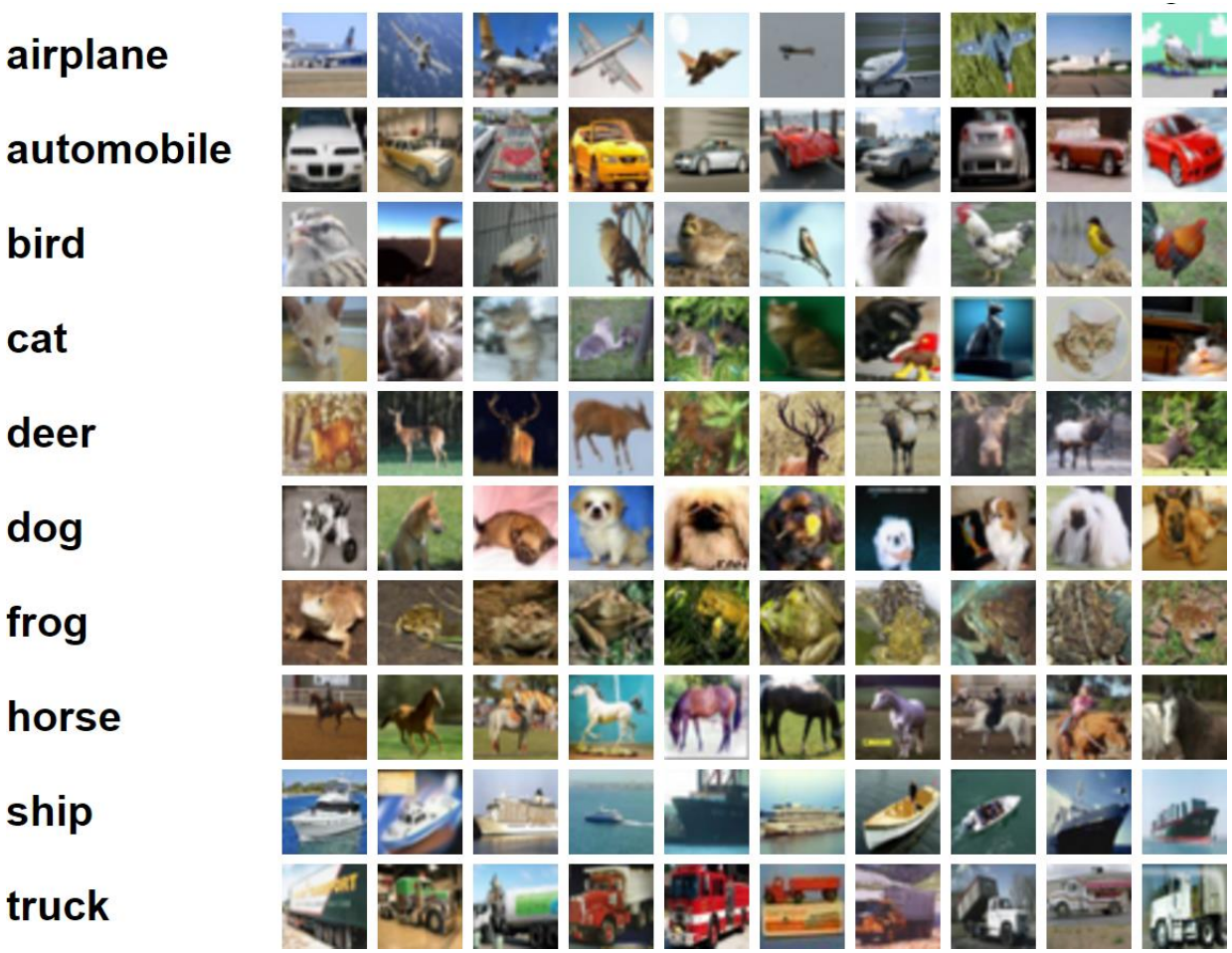
Each image is 32x32 RGB

Images are in 10 classes

6000 images per class

Extremely small images, but high-dimensional:

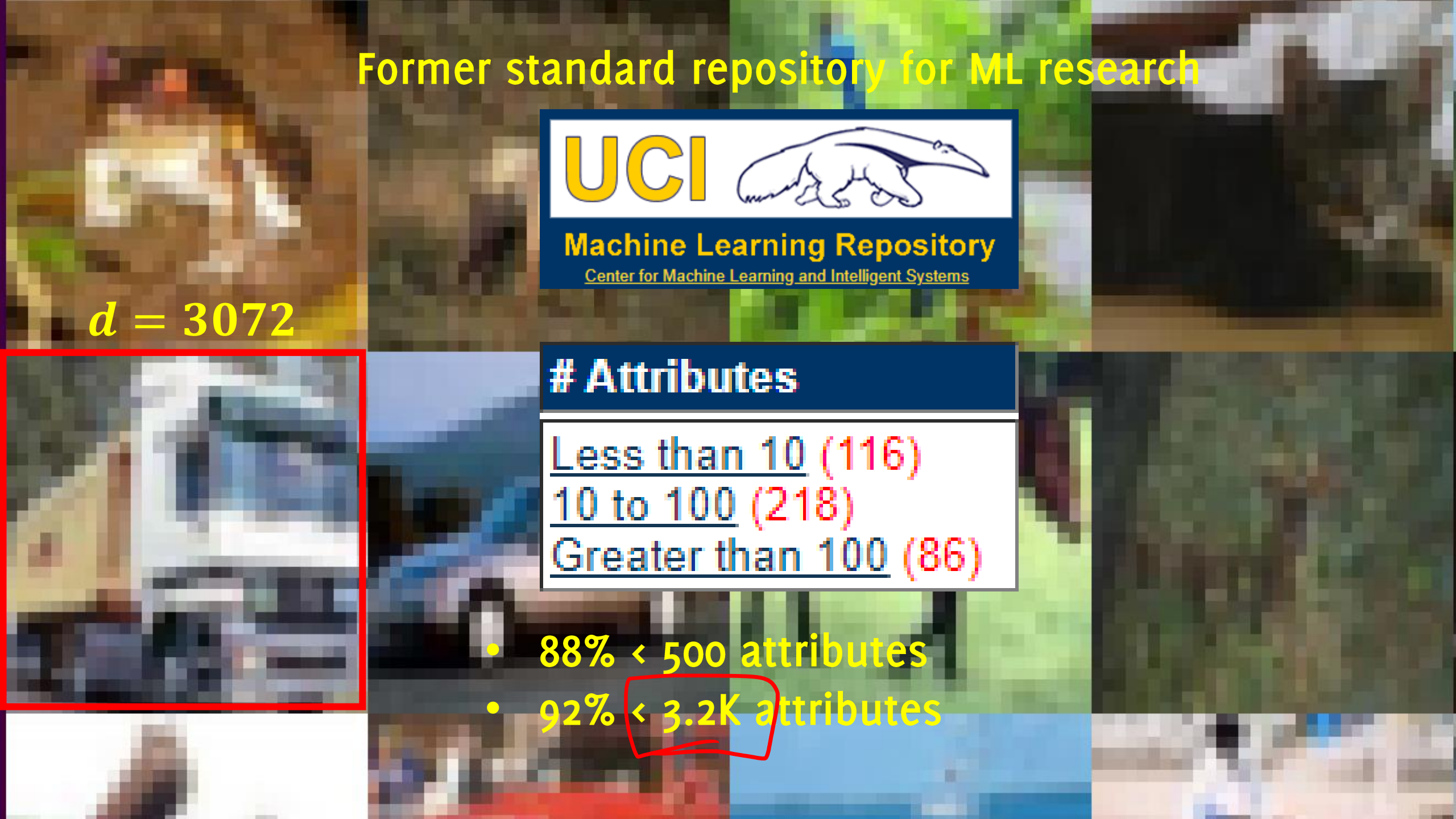
$$d = 32 \times 32 \times 3 = 3072$$



Learning Multiple Layers of Features from Tiny Images, Alex Krizhevsky, 2009.


This resolution is by far smaller than what we are used to
$d = 3072$

Former standard repository for ML research



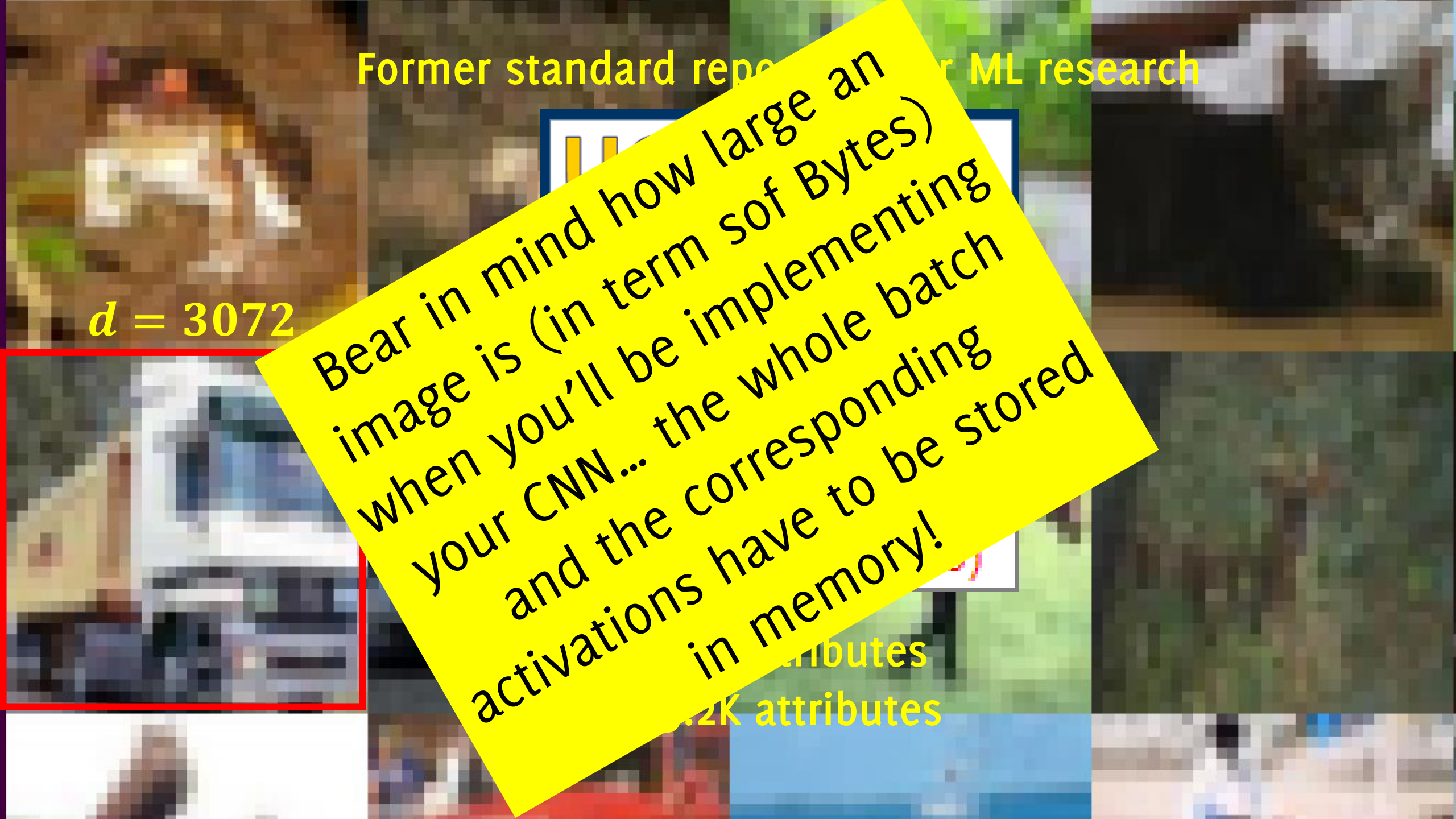
UCI Machine Learning Repository
Center for Machine Learning and Intelligent Systems

$d = 3072$

 # Attributes

Less than 10 (116)
10 to 100 (218)
Greater than 100 (86)

- 88% < 500 attributes
- 92% < 3.2K attributes


Former standard repe r ML research
$d = 3072$
Bear in mind how large an image is (in term sof Bytes) when you'll be implementing your CNN… the whole batch and the corresponding activations have to be stored in memory!
ributes
.2K attributes

Second challenge: label ambiguity

A label might not uniquely identify the image

Second problem: label ambiguity

Man?

Beer?

Dinner?

Restaurant?

Sausages?

….

Third challenge: transformations

There are many transformations that change the image dramatically, while not its label

Illumnation conditions changes






Deformations

Copyright Christine Matthews

© Copyright Patrick Roper

View Point Change

… and many othres

Occlusion

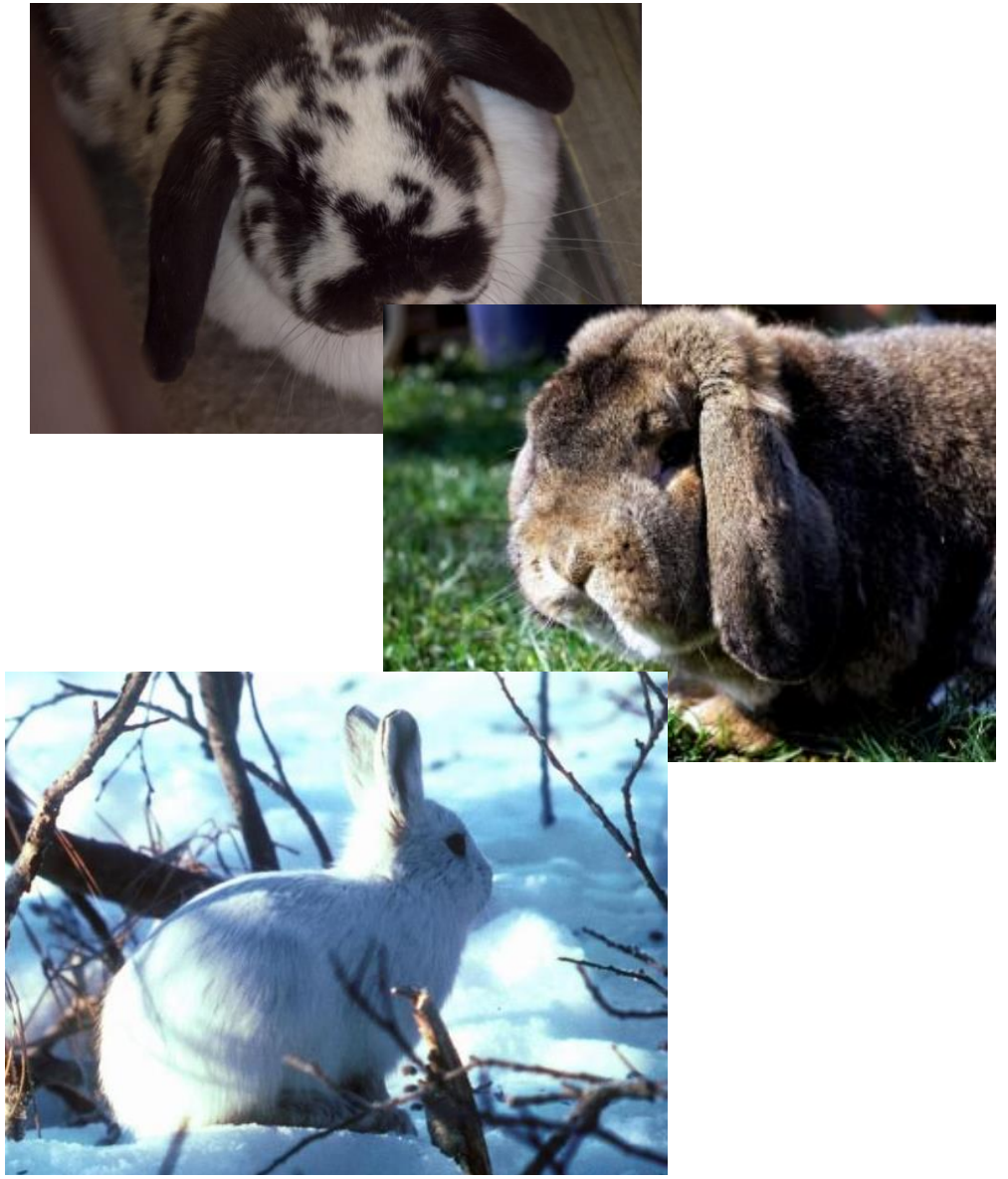
Background clutter

Scale variation

Fourth challenge: inter-class variability

Images in the same class might be dramatically different

Inter-class variability



Fifth problem: perceptual similarity

Perceptual similarity in images is not related to pixel-similarity

Nearest Neighborhood Classifiers for Images

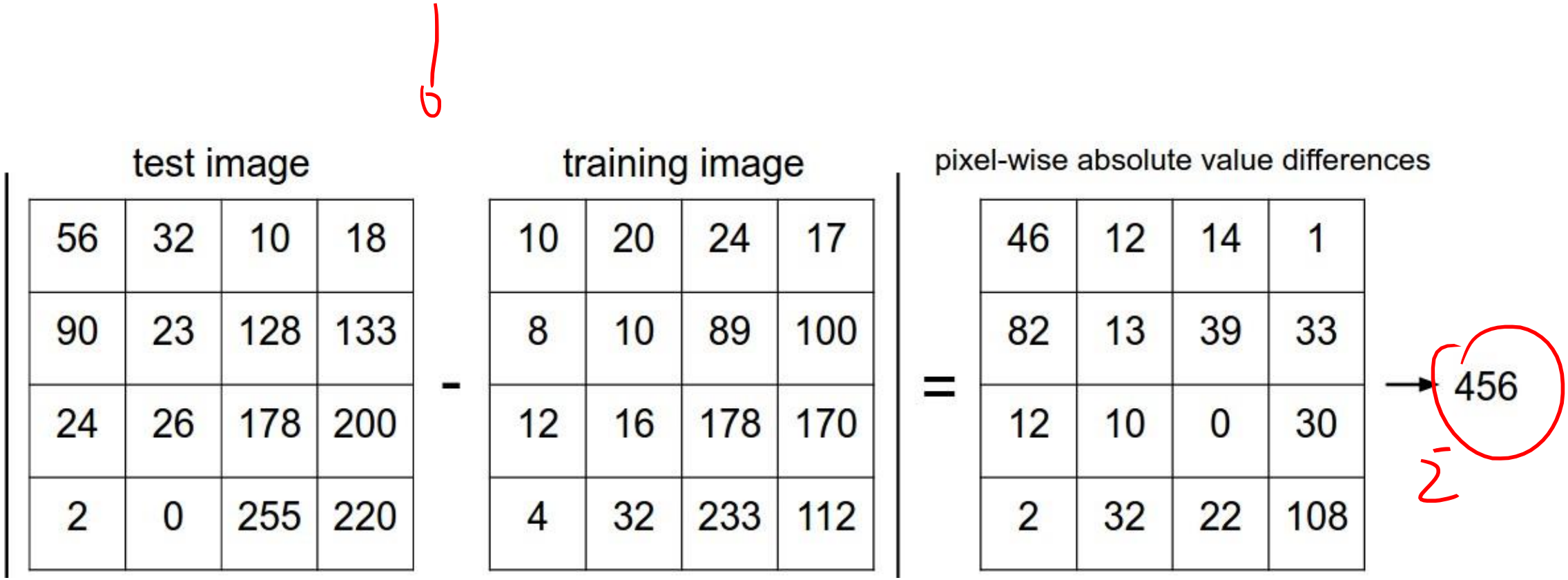
Assign to a each test image, **the label of the closest image** in the training set

$$\hat{y}_j = y_{j^*}, \qquad \text{being} \quad j^* = \underset{i=1\ldots N}{\operatorname{argmin}}\, d(\boldsymbol{x}_j, \boldsymbol{x}_i)$$

Distances are typically measured as

$$d(\boldsymbol{x}_j, \boldsymbol{x}_i) = \|\boldsymbol{x}_j - \boldsymbol{x}_i\|_2 = \sqrt{\sum_k \left([\boldsymbol{x}_j]_k - [\boldsymbol{x}_i]_k\right)^2}$$

Pixel-wise distance among images



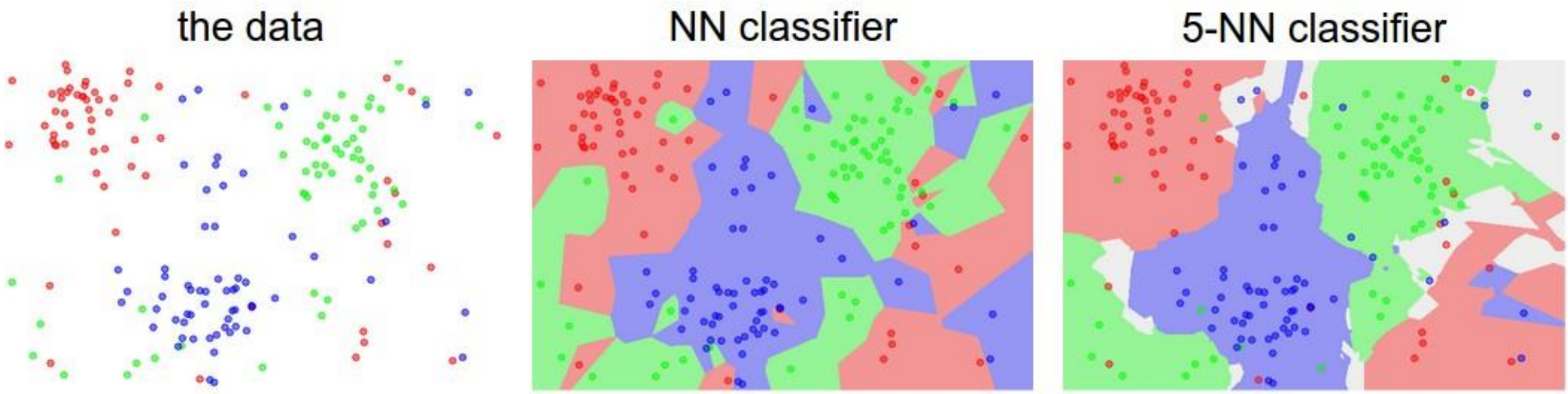
K-Nearest Neighborhood Classifiers for Images

Assign to a each test image, **the most frequent label among the K −closest images in the training set**

$$\hat{y}_j = y_{j^*}, \qquad \text{being } \; j^* \text{ the mode of } \mathcal{U}_K(\boldsymbol{x}_j)$$

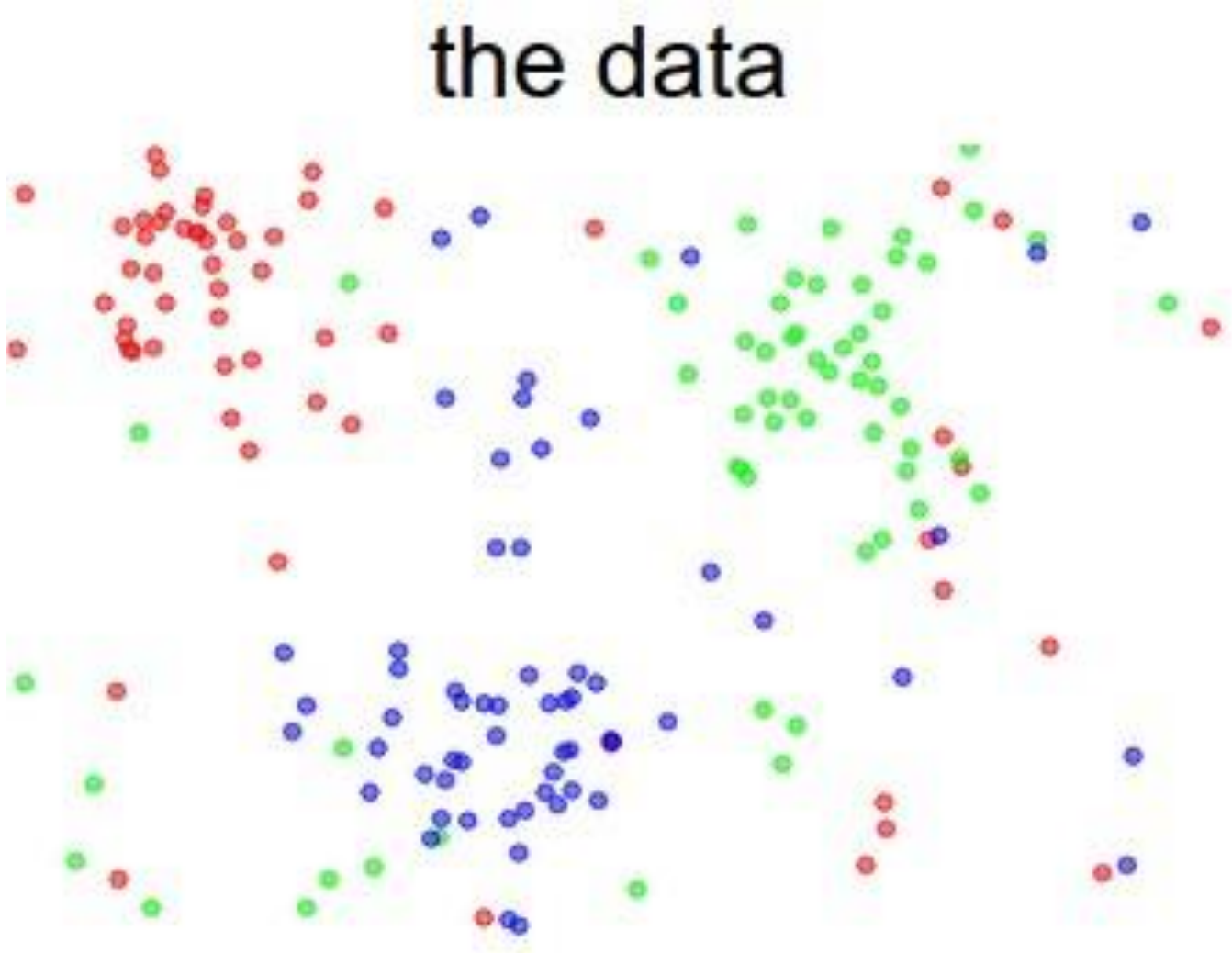
where $\mathcal{U}_K(\boldsymbol{x}_j)$ contains the K closest training images to $\boldsymbol{x}_j$

Choosing the right parameter K and the distance measure is often an issue



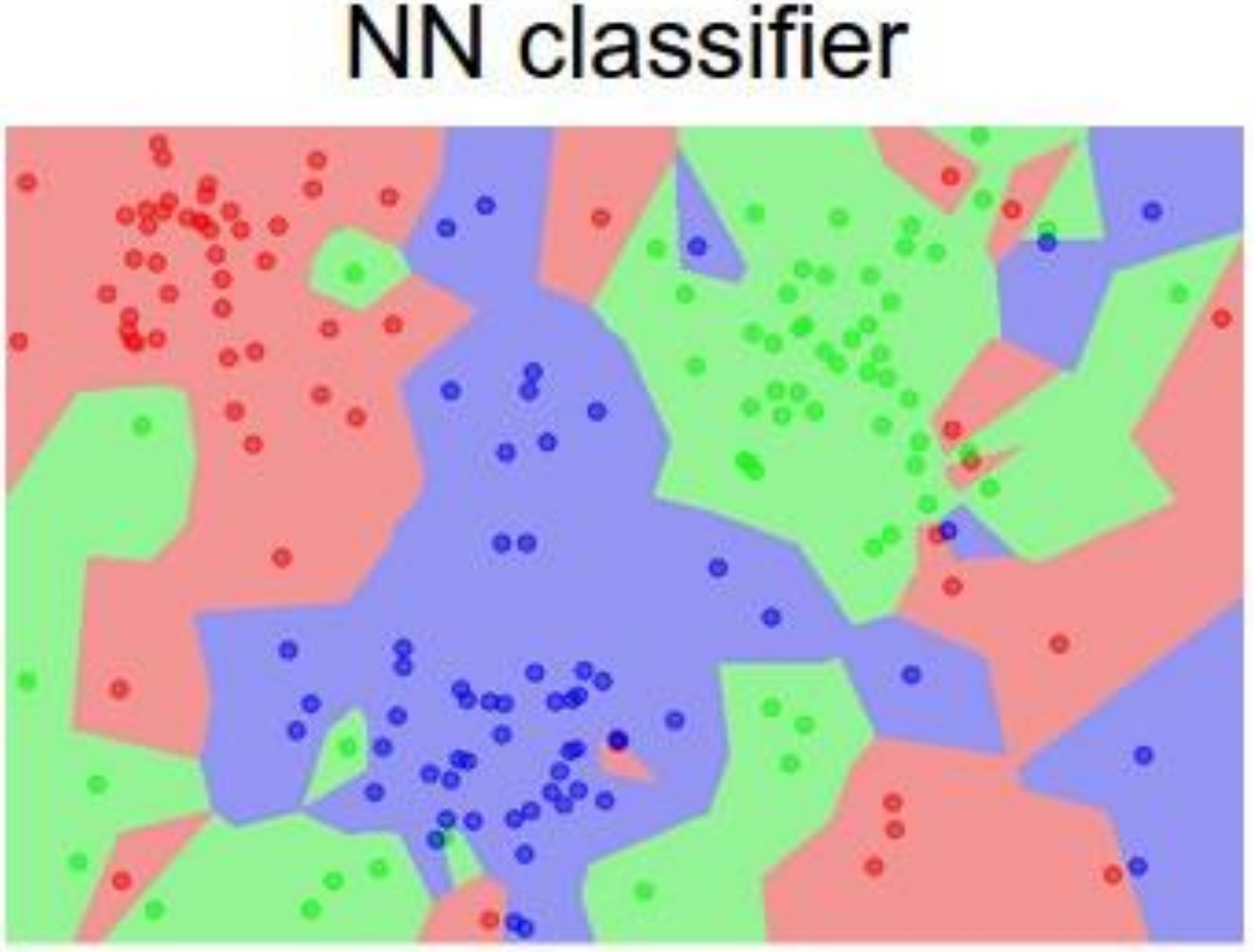
CS231n: Convolutional Neural Networks for Visual Recognition http://cs231n.github.io/

Nearest Neighborhood Classifiers for Images



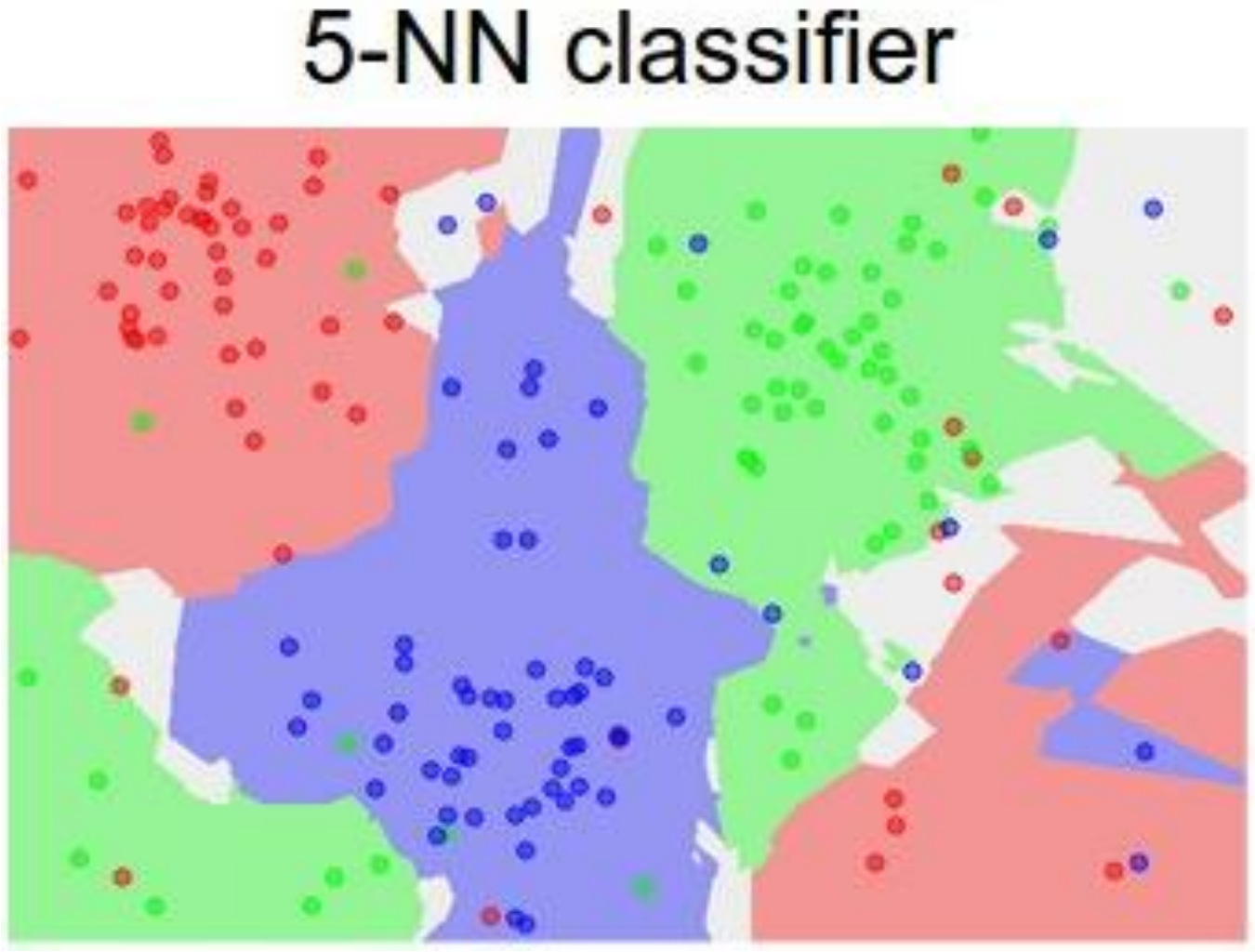
CS231n: Convolutional Neural Networks for Visual Recognition http://cs231n.github.io/

Nearest Neighborhood Classifiers for Images



CS231n: Convolutional Neural Networks for Visual Recognition http://cs231n.github.io/

Nearest Neighborhood Classifiers for Images



CS231n: Convolutional Neural Networks for Visual Recognition http://cs231n.github.io/

Nearest Neighborhood Classifiers for Images

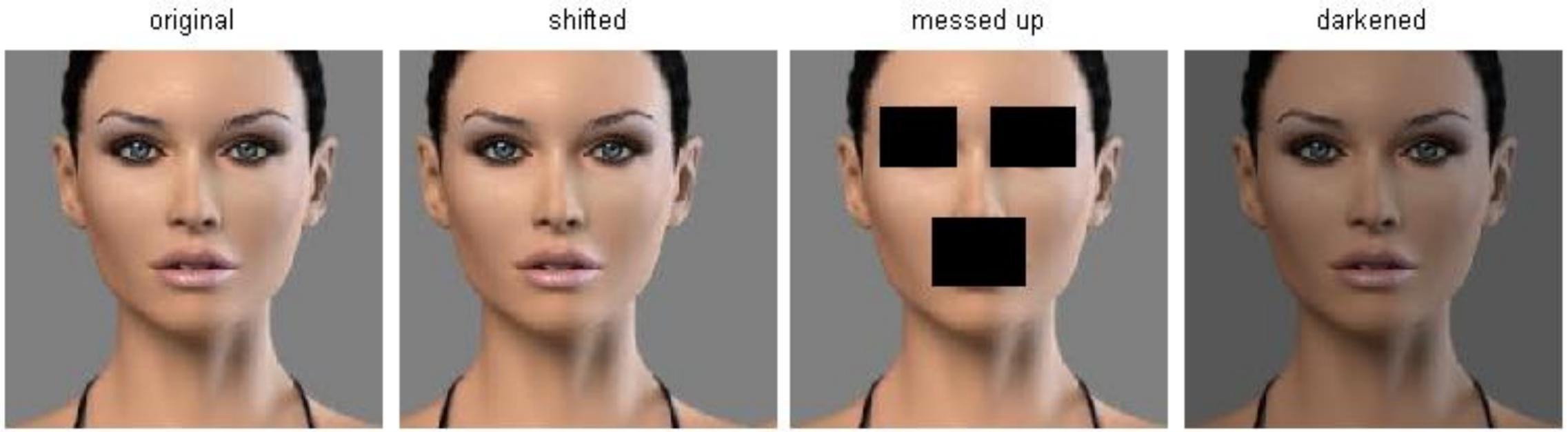
Pros:

- Easy to understand and implement
- It takes no training time

Cons:

- Computationally demanding at test time (in particular when TR is large and d is also large)
- Large training sets have to be stored in memory
- Rarely practical on images: distances on high-dimensional objects are difficult to interpret

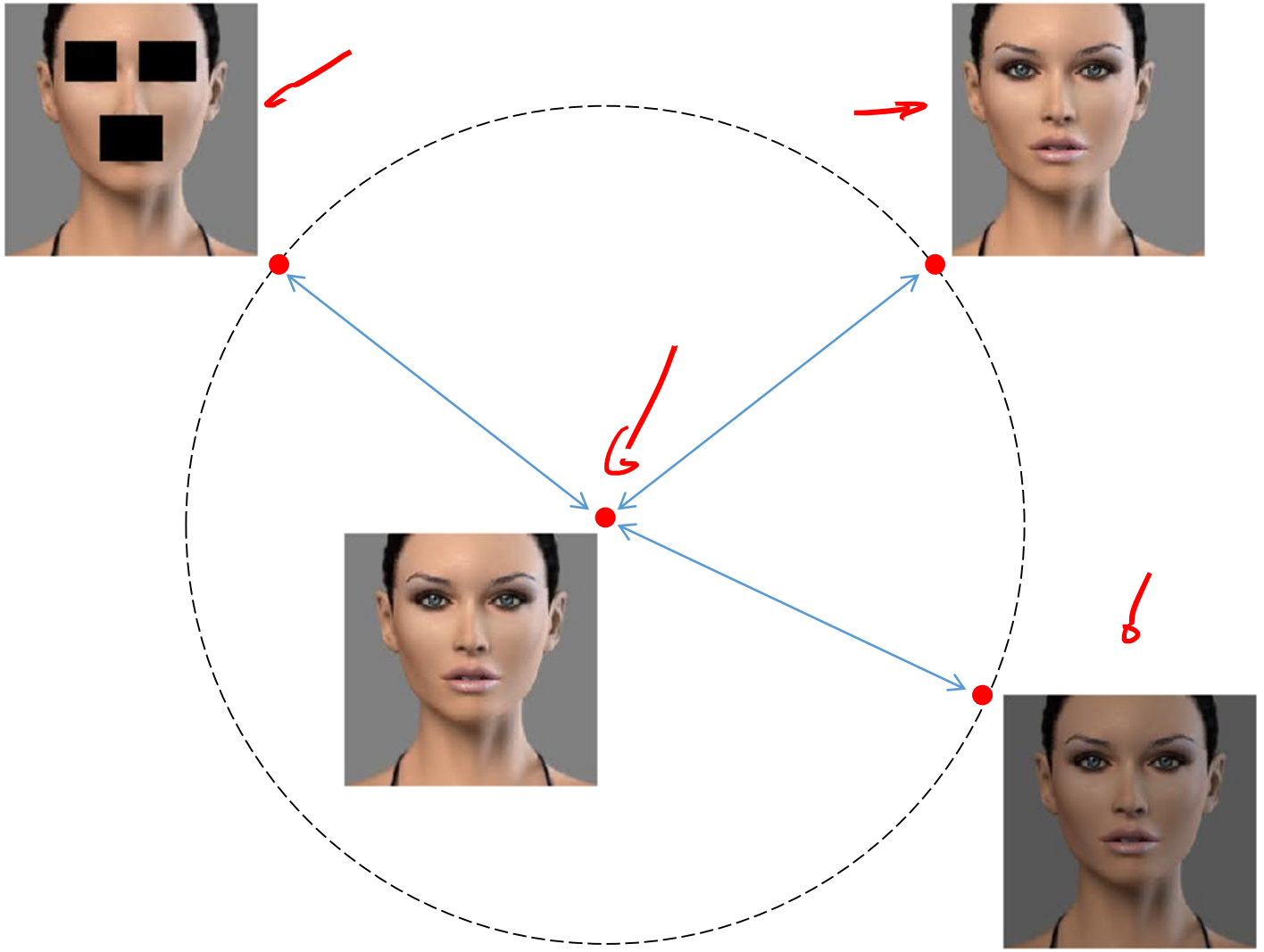
Perceptual Similarity vs Pixel Similarity



The three images have the same pixel-wise distance from the original one...

...but perceptually they are very different

Perceptual Similarity vs Pixel Similarity



CS231n: Convolutional Neural Networks for Visual Recognition http://cs231n.github.io/

Let's see what happens on the whole CIFAR10

On CIFAR10 we see exactly this problem

CS231n: Convolutional Neural Networks for Visual Recognition http://cs231n.github.io/

On CIFAR10 we see exactly this problem

Linear Classifier

the basic building block for deep architectures

Giacomo Boracchi

Linear Classifiers for Images

A classifier can be seen as a function that maps an image $\boldsymbol{x}$ to a confidence scores for each of the L class:

$$\mathcal{K}\colon \mathbb{R}^d \to \mathbb{R}^L$$

where $\mathcal{K}(\boldsymbol{x})$ is a L −dimensional vector and the i −th component

$$s_i = [\mathcal{K}(\boldsymbol{x})]_i$$

contains a score of how likely $\boldsymbol{x}$ belongs to class $\boldsymbol{i}$.

Intuitively, **a good classifier associates to the correct class a score that is larger** than scores associated to incorrect classes.

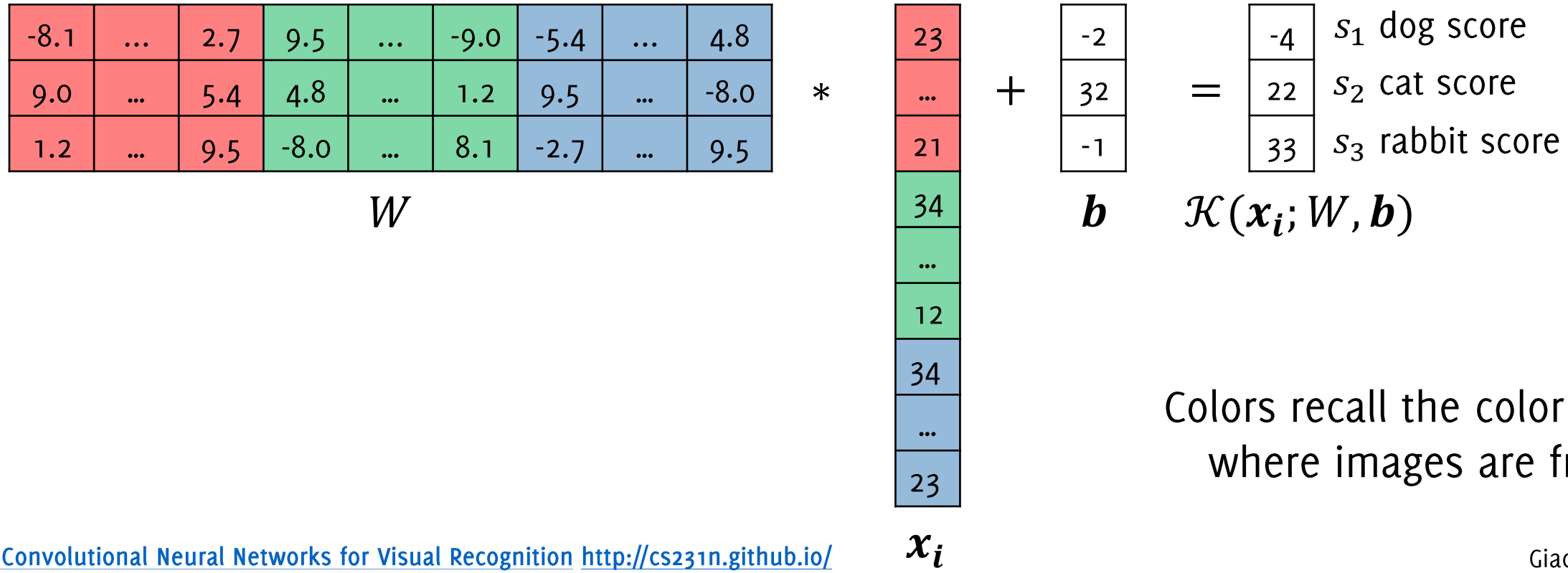
Linear Classifiers for Images

In linear classification $\mathcal{K}$ is a linear function:

$$\mathcal{K}(\boldsymbol{x}) = W\boldsymbol{x} + b$$

where $W \in \mathbb{R}^{L \times d}$, $b \in \mathbb{R}^{L}$ are the parameters of the classifier $\mathcal{K}$.

W are referred to as the weights, b the bias.



Colors recall the color plane where images are from

Linear Classifiers for Images

Unroll the image column-wise

-8.1	…	2.7	9.5	…	-9.0	-5.4	…	4.8
9.0	…	5.4	4.8	…	1.2	9.5	…	-8.0
1.2	…	9.5	-8.0	…	8.1	-2.7	…	9.5

W

$*$

23
…
21
34
…
12
34
…
23

$\boldsymbol{x_i}$

$+$

-2
32
-1

$\boldsymbol{b}$

$=$

-4	s_1 dog score
22	s_2 cat score
33	s_3 rabbit score

$\mathcal{K}(\boldsymbol{x_i}; W, \boldsymbol{b})$

Linear Classifiers for Images

The classifier assign to an input image the class corresponding to the largest score

$$\hat{y}_j = \underset{i=1,..,L}{\operatorname{argmax}} \left[s_j\right]_i$$

being $\left[\boldsymbol{s}_j\right]_{\boldsymbol{i}}$ the i −th component of the vector

$$\mathcal{K}(\boldsymbol{x}_j) = W\boldsymbol{x}_j + \boldsymbol{b}$$

-8.1	...	2.7	9.5	...	-9.0	-5.4	...	4.8
9.0	...	5.4	4.8	...	1.2	9.5	...	-8.0
1.2	...	9.5	-8.0	...	8.1	-2.7	...	9.5

W

$*$

23
...
21
34
...
12
34

$\boldsymbol{x_i}$

$+$

-2
32
-1

$\boldsymbol{b}$

$=$

-4	s_1 dog score
22	s_2 cat score
33	s_3 rabbit score

$\mathcal{K}(\boldsymbol{x_i}; W, \boldsymbol{b})$

Linear Classifiers for Images

The score of a class is the weighted sum of all the image pixels.
Weights are actually the classifier parameters.

Weights indicate which are the most important pixels / colors

-8.1	...	2.7	9.5	...	-9.0	-5.4	...	4.8
9.0	...	5.4	4.8	...	1.2	9.5	...	-8.0
1.2	...	9.5	-8.0	...	8.1	-2.7	...	9.5

W

-2
32
-1

$\boldsymbol{b}$

Training a Classifier

Given a training set TR and a loss function, define the parameters that minimize the loss function over the whole TR

In case of linear classifier

$$[W, b] = \underset{W \in \mathbb{R}^{L \times d},\ b \in \mathbb{R}^{L}}{\operatorname{argmin}} \sum_{(\boldsymbol{x}_i, y_i) \in TR} \mathcal{L}(\boldsymbol{x}, y_i)$$

Solving this minimization problem provides the weights of our classifier

Loss Function

Loss function: a function $\mathcal{L}$ that measures our unhappiness with the score assigned to training images

The loss $\mathcal{L}$ will be high on a training image that is not correctly classifier, low otherwise.

Loss Function Minimization

Loss function can be minimized by specific algorithms (see Matteo' classes)

The loss function has to be typically regularized to achieve a unique solution satisfying some desired property

$$[W, b] = \underset{W \in \mathbb{R}^{L \times d},\ b \in \mathbb{R}^{L}}{\operatorname{argmin}} \sum_{(\boldsymbol{x}_i, y_i) \in TR} \mathcal{L}(\boldsymbol{x}, y_i) + \lambda\, \mathcal{R}(W, b)$$

being $\lambda > 0$ a parameter balancing the two terms

Once Trained

The classifier is expected to provide to the correct class a score that is larger than that assigned to the incorrect classes.

The training data is used to learn the parameters $W, \boldsymbol{b}$

Once the training is completed, it is possible to discard the whole training set and only keep the learned parameters.

Geometric Interpretation of a Linear Classifier

In Matlab notation:

$W(i,:)$ is a d −dimensional vector containing the weights of the score function for the i −th class

Computing the score function for the i −th class corresponds to computing the inner product

$$W(i,:) * \boldsymbol{x}$$

Thus, linear classifiers identify an hyperplane in a d −dimensional

(in Python $*$ is the element-wise product, here I mean the inner product of vectors):

Geometric Interpretation

Interpret each image as a point in $\mathbb{R}^d$.

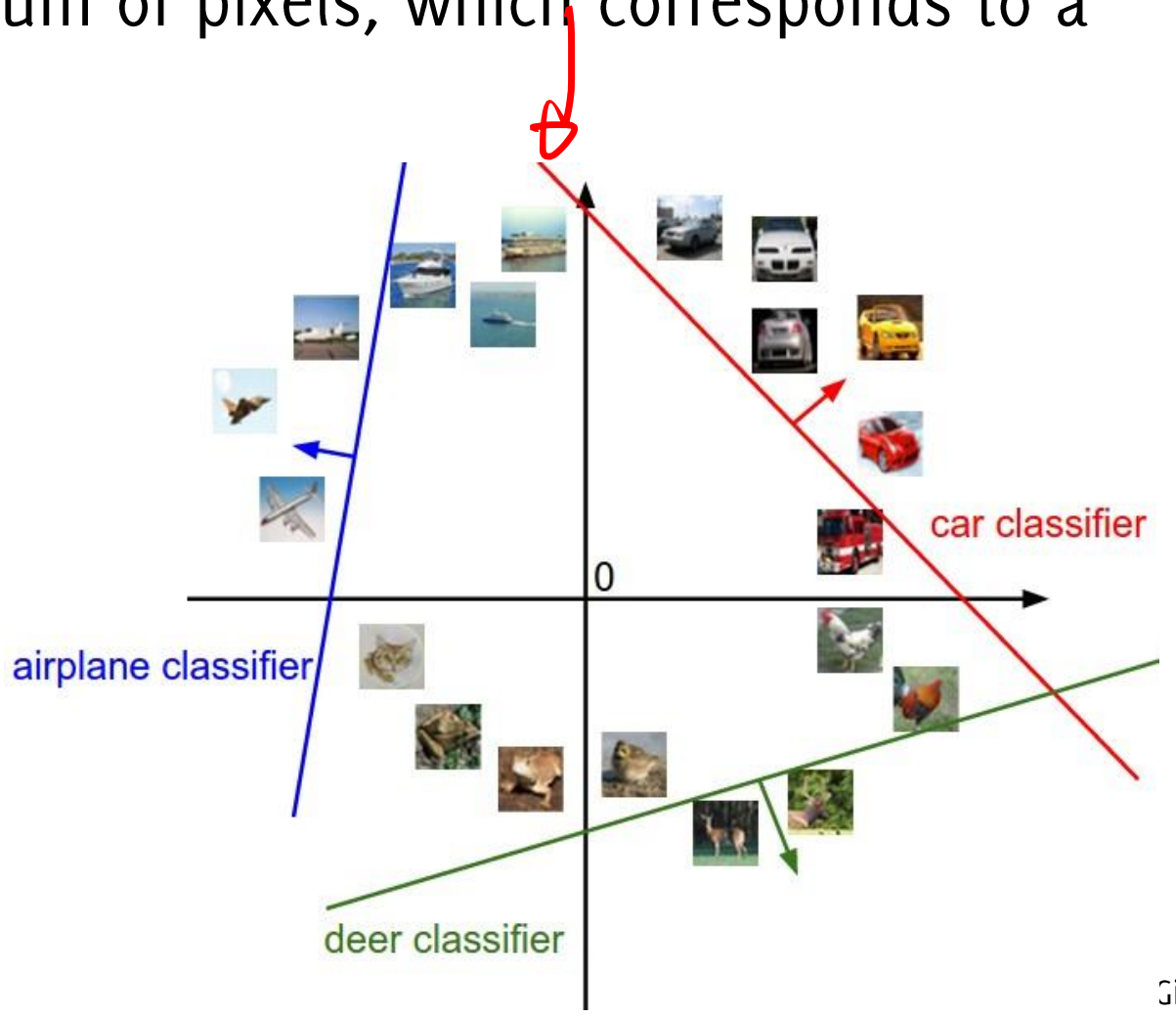
Each classifier is a weighted sum of pixels, which corresponds to a linear function in $\mathbb{R}^d$

In $\mathbb{R}^2$ these would be
$f([x_1, x_2]) = w_1 x_1 + w_2 x_2 + b$

Then, points $[x_1, x_2]$ yielding
$$f([x_1, x_2]) = 0$$

would be lines.

Thus, in $\mathbb{R}^2$ the region that separates positive from negative scores for each class is a line. This region becomes an hyperplane in $\mathbb{R}^d$.



Template Matching Interpretation

Using Matlab notation:

- $W(i,:)$ is a d −dimensional vector containing the weights of the score function for the i −th class
- Computing the score function for the i −th class corresponds to computing the inner product

$$W(i,:) * \boldsymbol{x}$$

Then**, $\boldsymbol{W(i,:)}$ can be seen as a template used in matching (the output of correlation in the central pixel)**

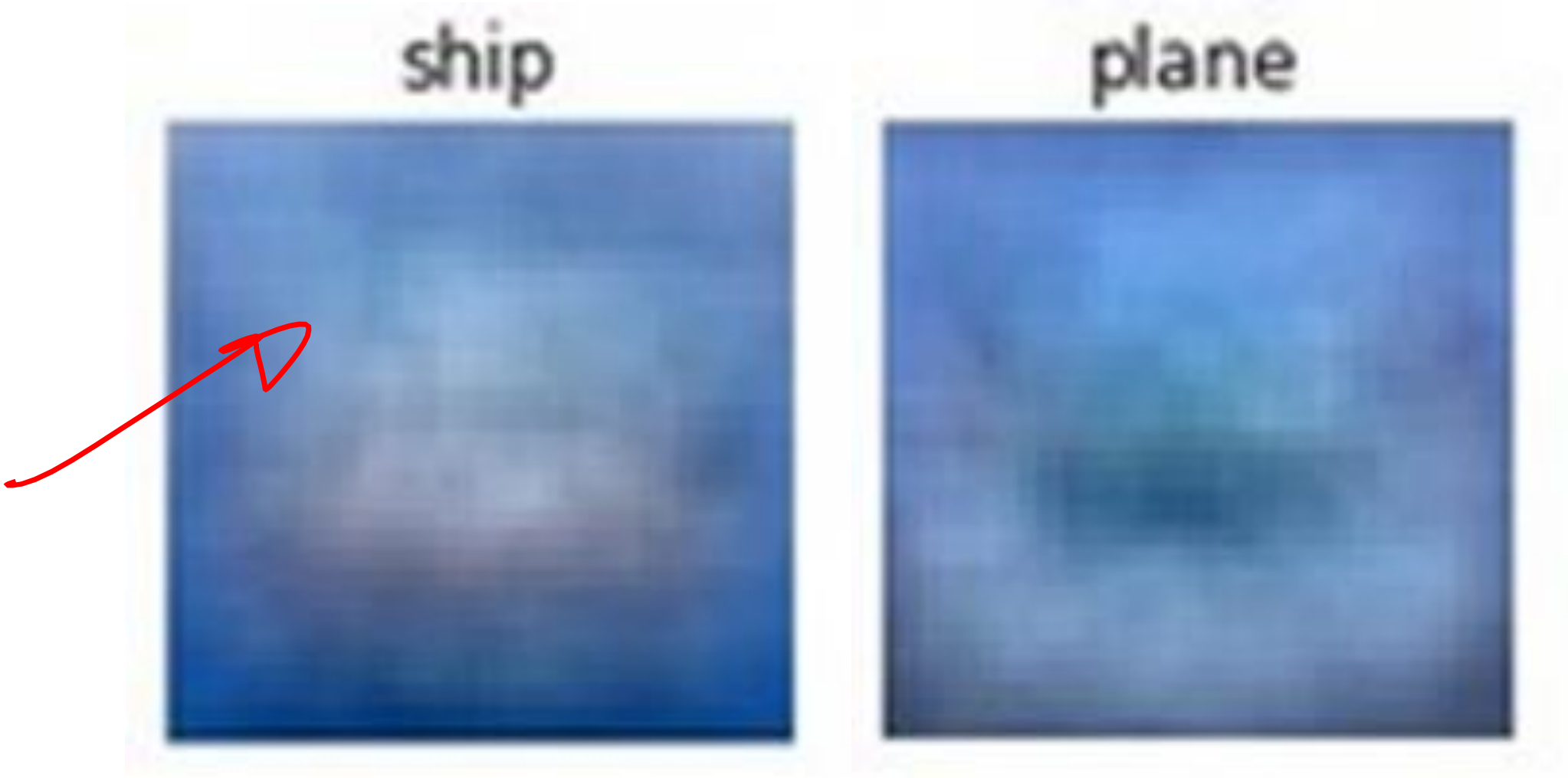
The template $W(i,:)$ is learned to match at best images belonging to the i −th class

Let's have a look at these templates

Templates Learned on the CIFAR-10 dataset

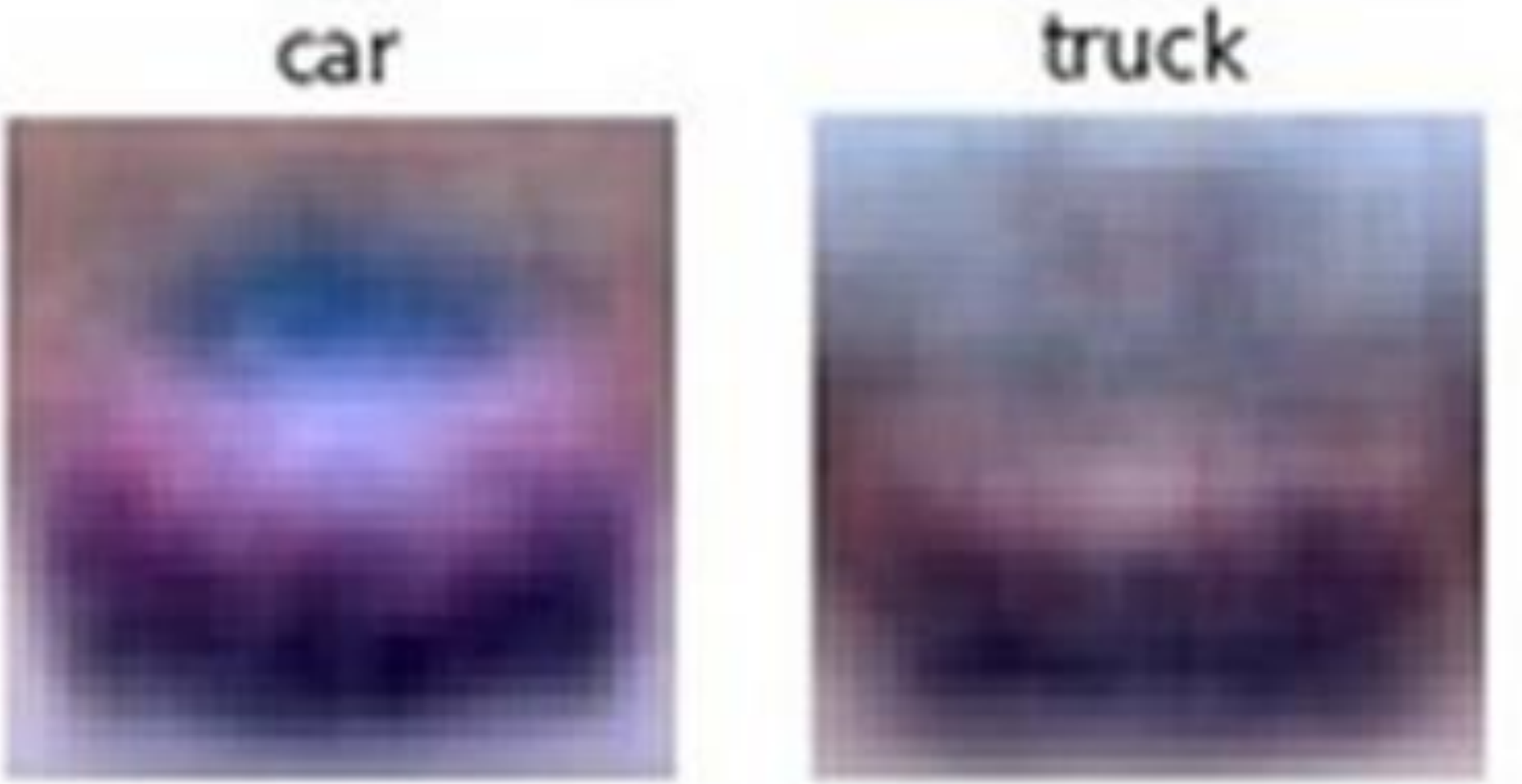
CS231n: Convolutional Neural Networks for Visual Recognition http://cs231n.github.io/

Templates Learned on the CIFAR-10 dataset



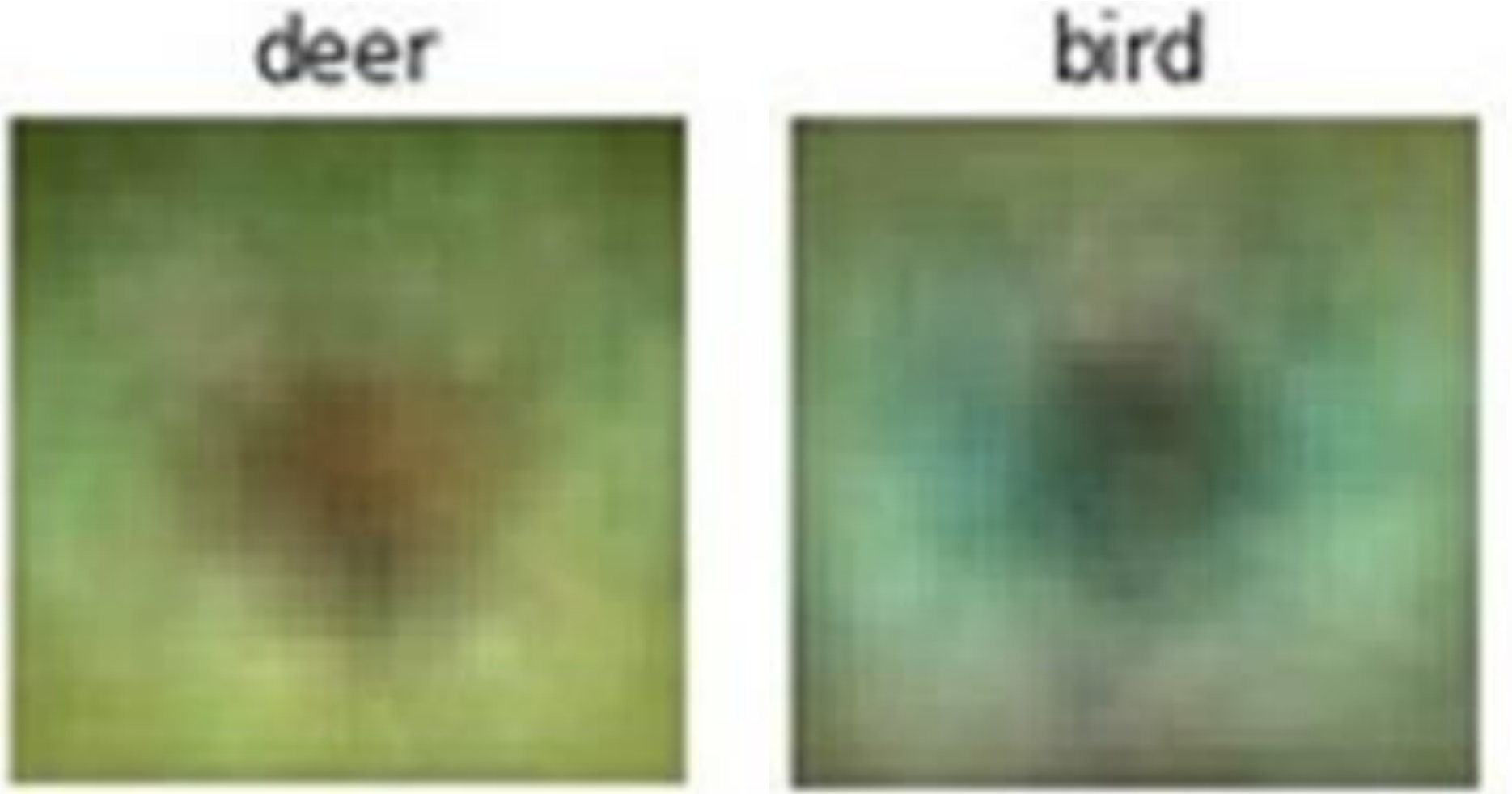
CS231n: Convolutional Neural Networks for Visual Recognition http://cs231n.github.io/

Templates Learned on the CIFAR-10 dataset



CS231n: Convolutional Neural Networks for Visual Recognition http://cs231n.github.io/

Templates Learned on the CIFAR-10 dataset



CS231n: Convolutional Neural Networks for Visual Recognition http://cs231n.github.io/

Templates Learned on the CIFAR-10 dataset

Linear Classifier as a Template Matching

What has the classifier learned?

- That the background of bird and frog is green, (plane and boat is blue)
- Cars are typically red
- Horses have two heads! ☺

The model was definitively too simple / data were not enough for achieving higher performance and better templates

However:

- Linear Classifiers are among the most important layer of NN
- Such a simple model can be interpreted (with more sophisticated models you typically can’t)

Linear Classifier as a Template Matching

What has the classifier learned?

- That the background of bird and frog is ... and boat is blue)
- Cars are typically red
- Horses have two heads! ☺

The model was definitive... / data were not enough for achieving higher perfor... better templates

However:

- Linear Classifiers a... the most important layer of NN
- Such a simple model can be interpreted (with more sophisticated models you typically can't)

There should be a better way for handling images