

Image Classification

Giacomo Boracchi,
DEIB, Politecnico di Milano
October, 4th, 2022

giacomo.boracchi@polimi.it

<https://boracchi.faculty.polimi.it/>

Materials

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

What is Computer Vision About

Visual Recognition Problems

Computer Vision

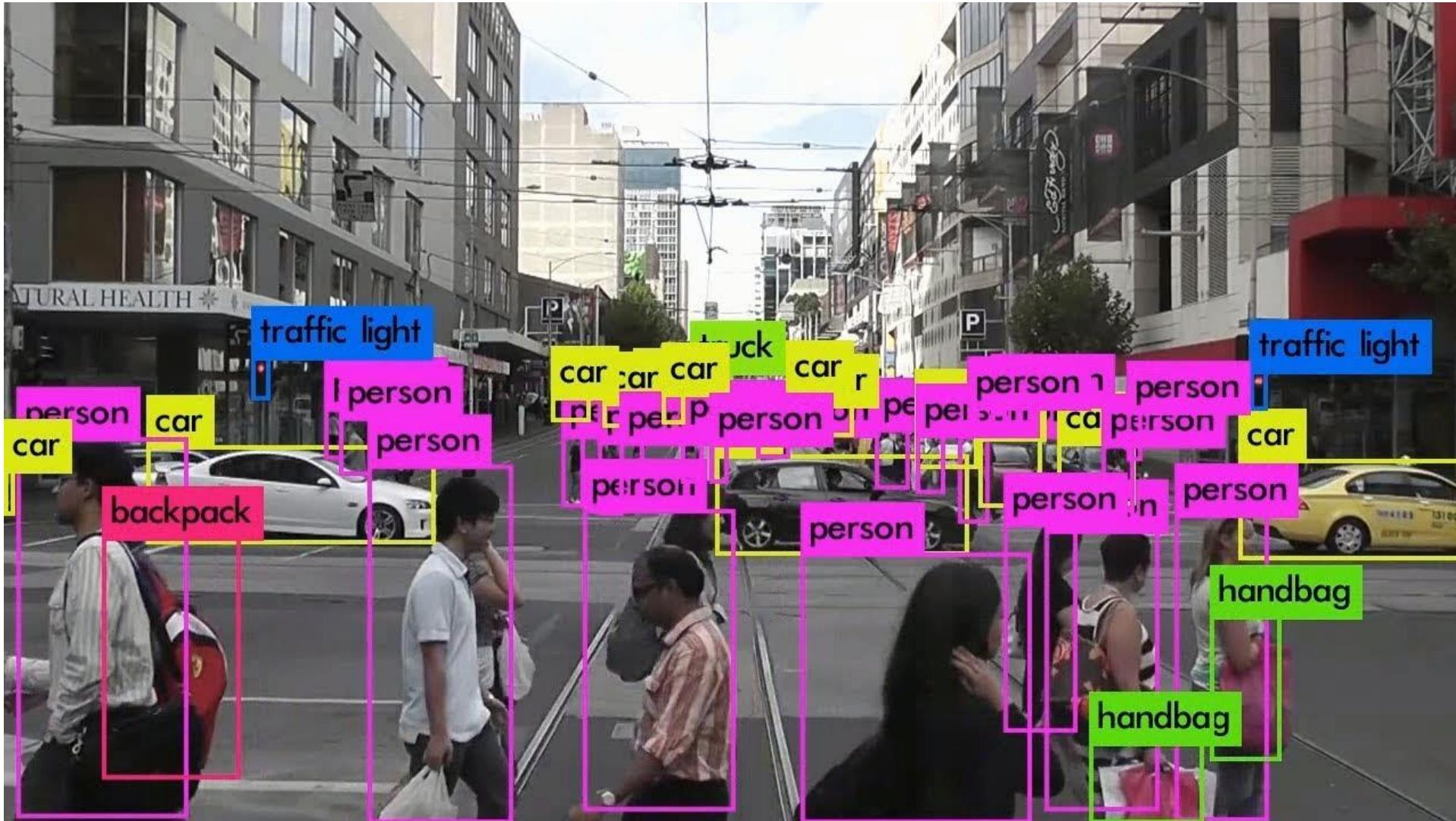
An interdisciplinary scientific field that deals with **how computers** can be made to gain **high-level understanding from digital images or videos**

Computer Vision

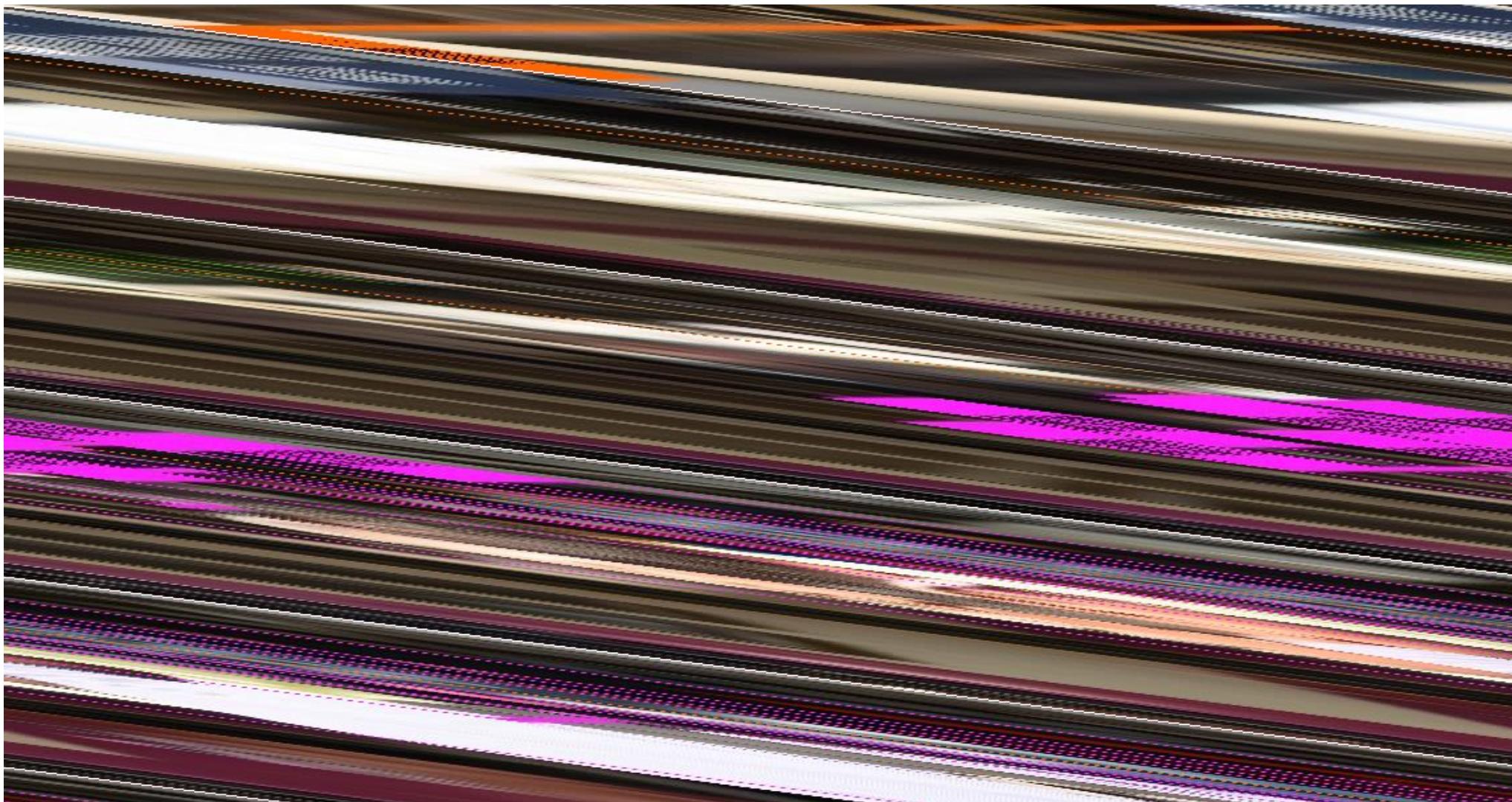
An interdisciplinary scientific field that deals with **how computers** can be made to **gain high-level understanding from digital images or videos**

... which has grown incredibly fast in the last years

Object Detection



Object Detection

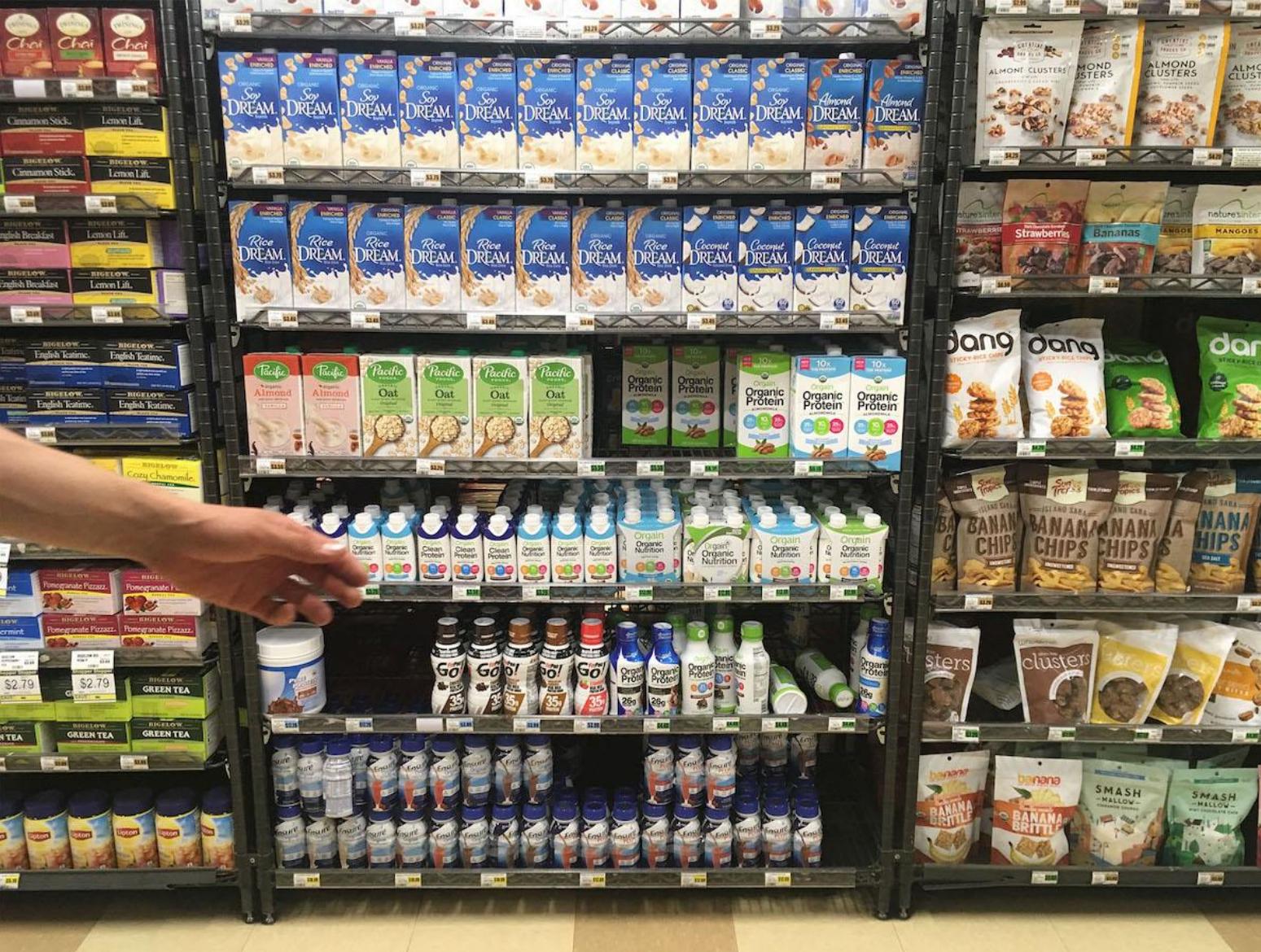


Pose Estimation



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).

Automatic Shelf Analysis



Automatic Shelf Analysis



Our Solution



Our Solution



LASONIL ANTIDOLORE GEL 50 G	4
SPLENDID CLASSICO	6
TACHIPIRINA COMPRESSE	4
ASPIRINA DI 500MG CPR	4
ASPIRINA C CPR EFF	5
CACAO AMARO PENNY	6
CAMOMILLA BONOMELLI	4
SPLENDID RISTRETTO	3
BUDINO RISTORA	4
MOMENT COMPRESSE	1



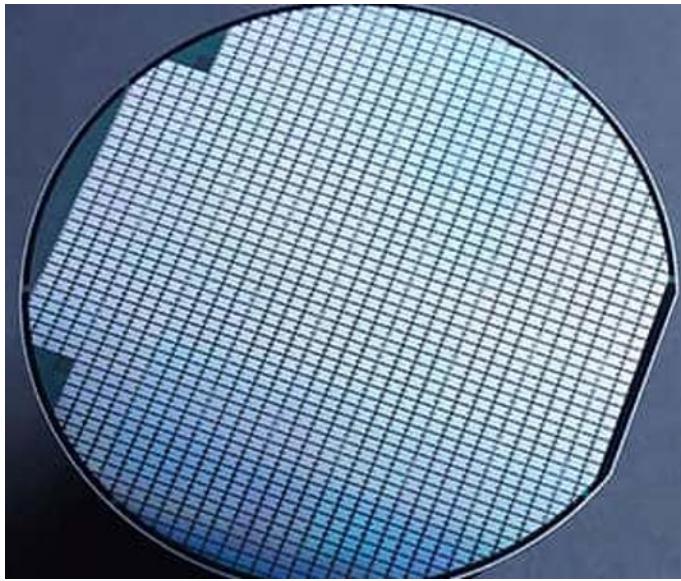
OneVision+
COMPUTER VISION FOR CRM



Our Solution



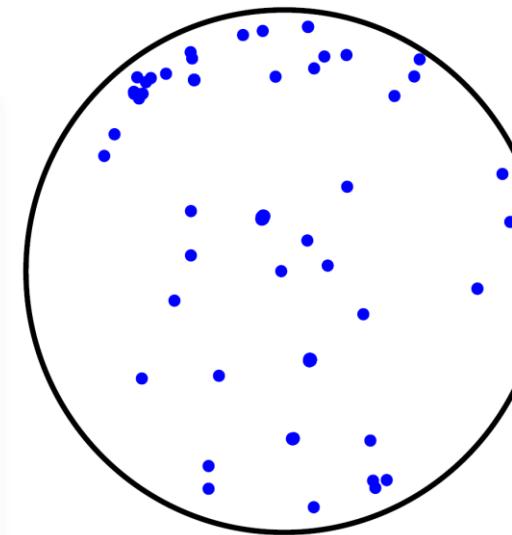
Quality Inspection



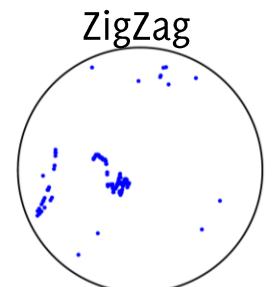
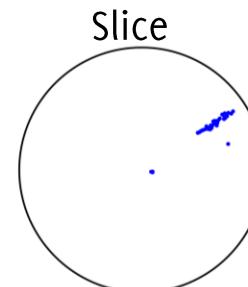
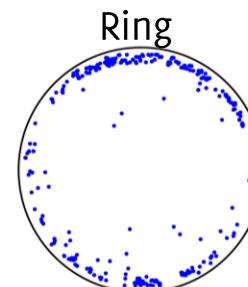
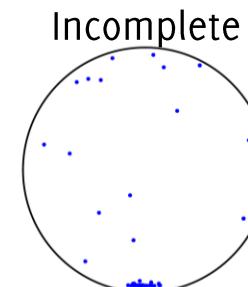
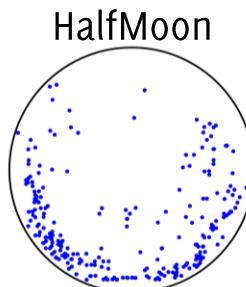
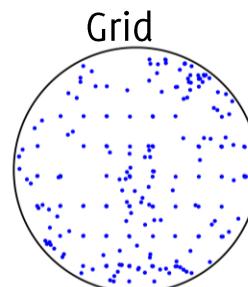
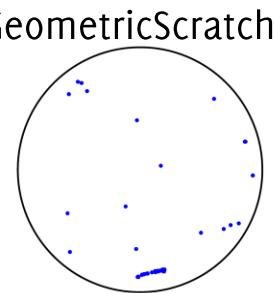
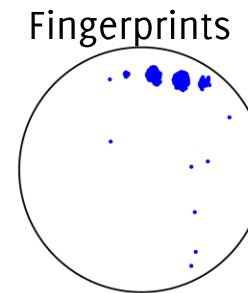
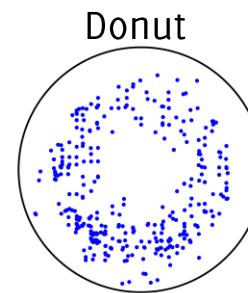
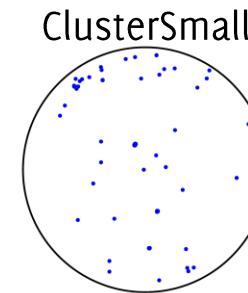
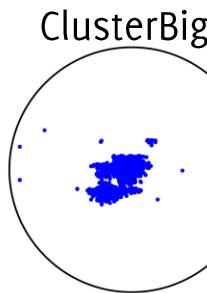
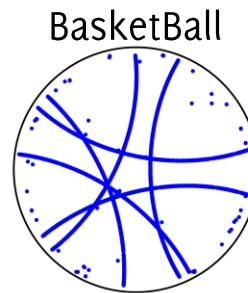
Inspection
Tool

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

Wafer Defect Map (WDM)



Silicon Wafer



Our SSCN architecture

Perfoms both Classification and Novel-class detection

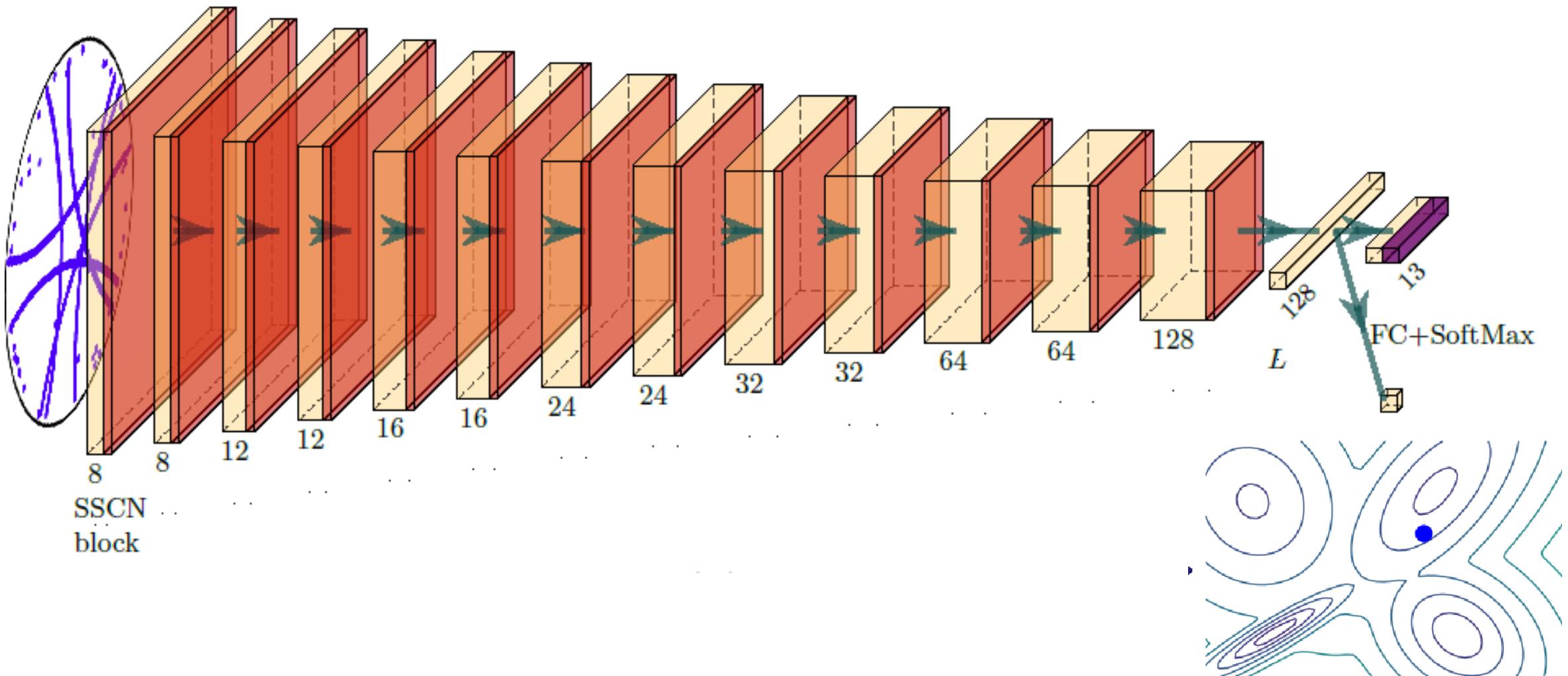


Image Captioning



"little girl is eating piece of cake."



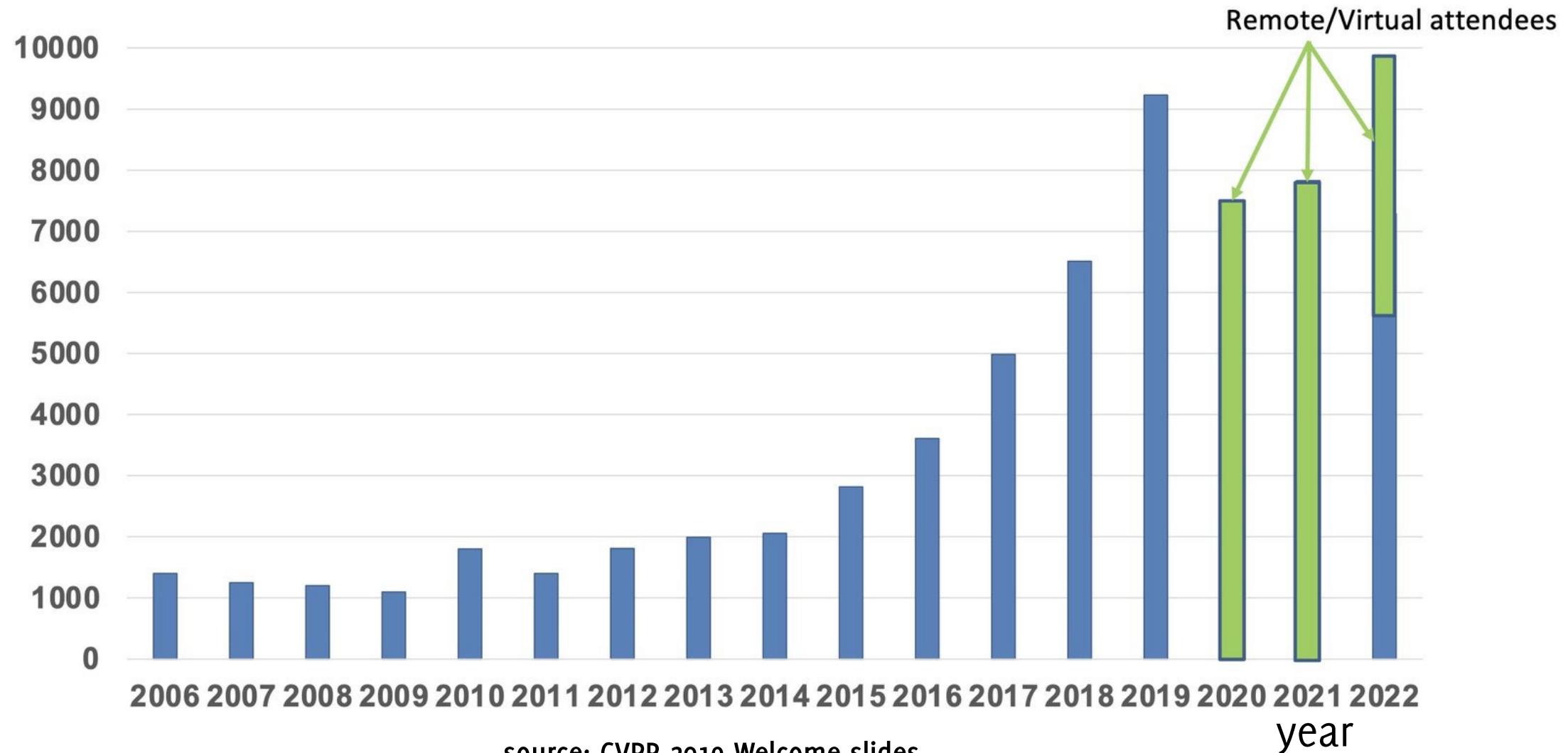
"black cat is sitting on top of suitcase."

Even though sometimes it fails

The screenshot shows a Windows desktop environment. On the left is a pinned Microsoft Edge browser window displaying a Google search results page for the query "smoothing". The search results include links to the Wikipedia pages for "Smoothing - Wikipedia" and "Lisciamento - Wikipedia", along with a "People also ask" section and a snippet from context.reverso.net. To the right of the browser is a pinned Microsoft Edge window showing a search result for "smoothing" on context.reverso.net, which translates the term to Italian. The desktop taskbar at the bottom shows several pinned files and applications, including "relazione (5).pdf", "BANDO SI4.0_202....PDF", "bando-SI40-2021.pdf", "MobaXterm_Install....zip", and "main.c". The system tray on the far left indicates a battery level of 42%, a signal strength icon, and the date/time as Wednesday, 20/10/2021, 11:23.

CVPR

CVPR Attendance Trend (as of June 20, 2022)



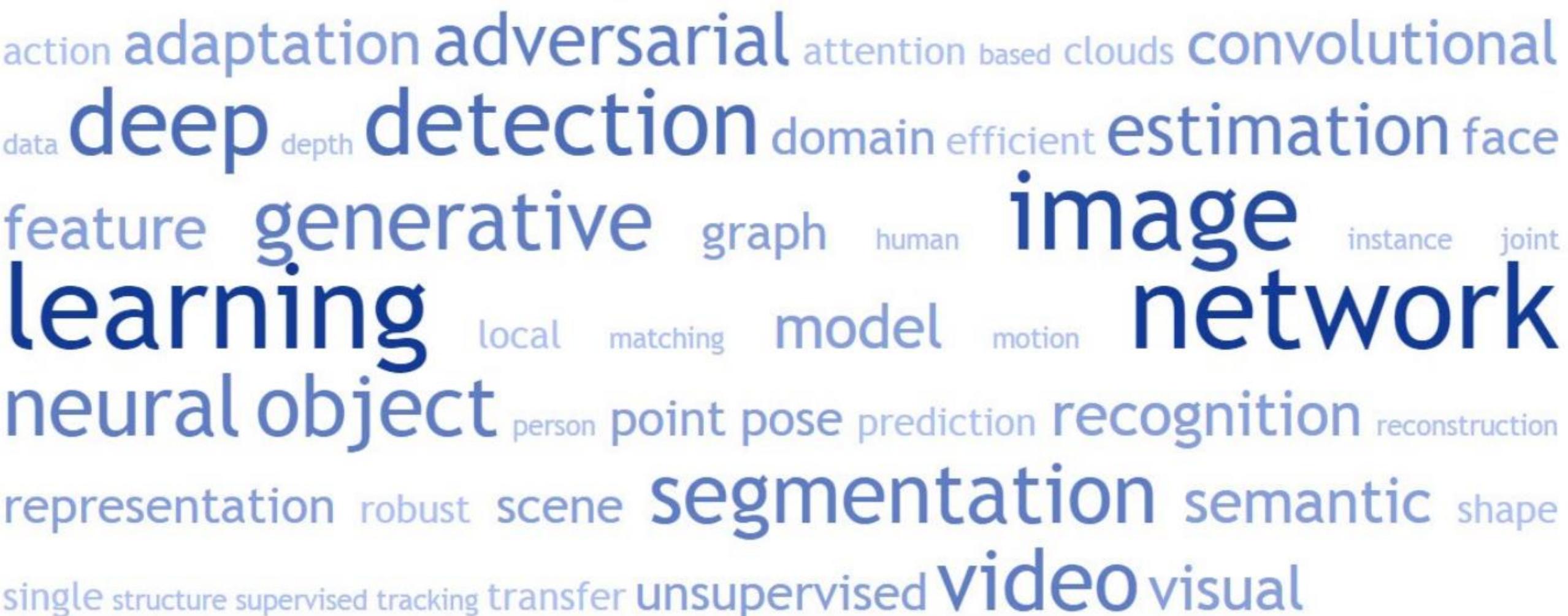
Lately, connection with ML

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

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

Acceptance rate is roughly even across topics



CVPR 2022

MENU ▾ Select Primary Subject Area

papers

Oral

Poster

		# papers	Oral	Poster
1	Recognition: detection, categorization, retrieval	177	25	152
2	Image and video synthesis and generation	157	26	131
3	3D from multi-view and sensors	137	26	111
4	Low-level vision	110	19	91
5	Vision + language	105	20	85
6	Segmentation, grouping and shape analysis	99	16	83
7	Transfer/ low-shot/ long-tail learning	86	15	71
8	Deep learning architectures and techniques	85	20	65
9	Self- & semi- & meta- & unsupervised learning	84	7	77
10	Video analysis and understanding	77	15	62
11	Pose estimation and tracking	62	14	48
12	Representation learning	61	11	50
13	3D from single images	60	10	50
14	Scene analysis and understanding	56	9	47
15	Face and gestures	54	7	47
16	Computational photography	53	10	43
17	Motion and tracking	53	8	45
18	Adversarial attack and defense	52	10	42
19	Datasets and evaluation	52	7	45
20	Machine learning	41	7	34
21	Action and event recognition	40	8	32
22	Efficient learning and inferences	40	3	37
23	Medical, biological and cell microscopy	37	5	32

Today...

We will start fresh from the simplest visual recognition problem:

Image Classification

Then we will move to the most advanced ones...

But first, let's have a look at images

Images



Photo Credits: Andrea Sanfilippo

RGB Images

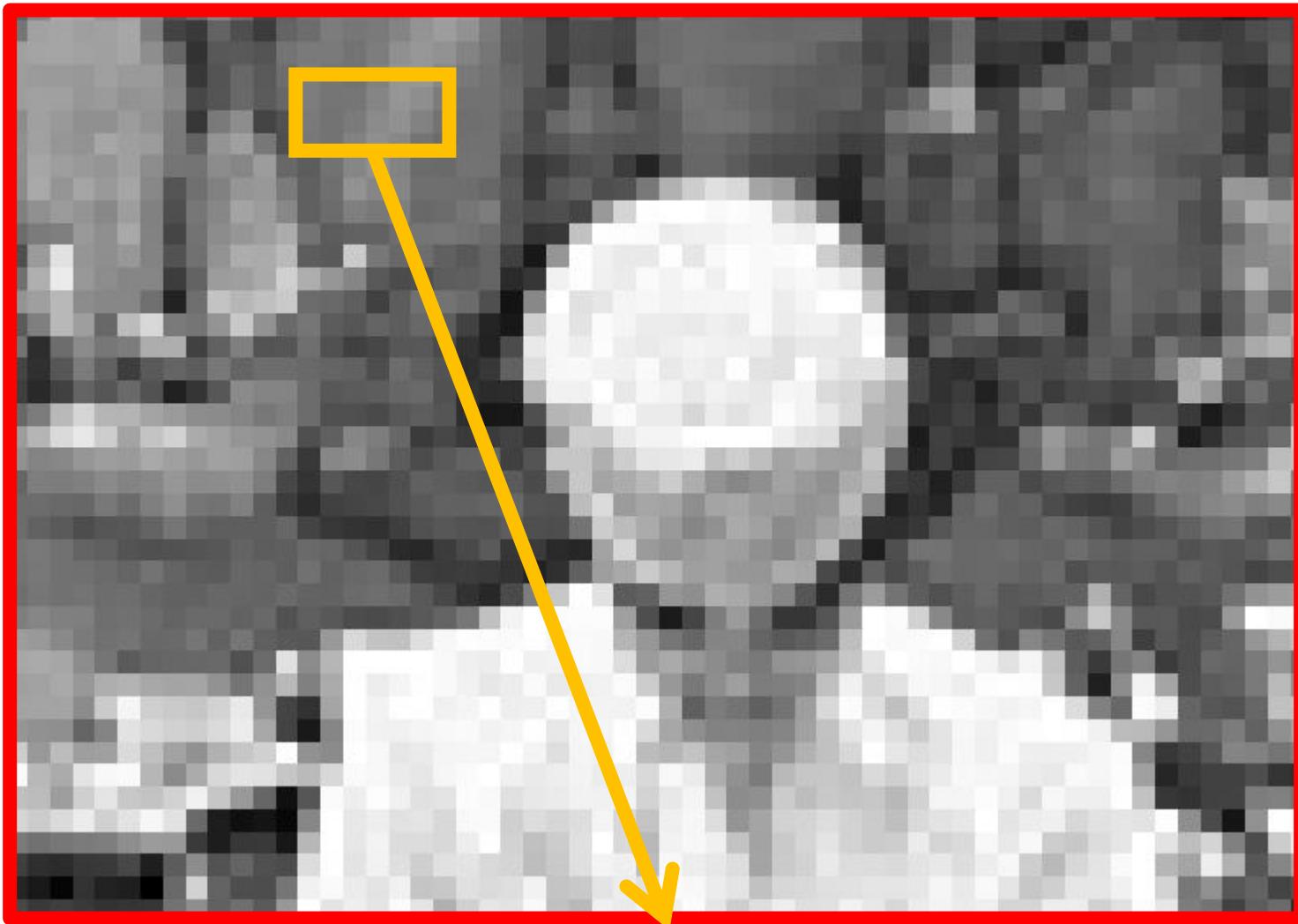


RGB Images

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

RGB Images

[253, 5, 6]

This is an RGB triplet
R = 253; G = 2; B = 6

[0, 205, 155]

[15, 17, 19]

[230, 234, 233]



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]
```

When loaded in memory, image sizes are much larger than on the disk where images are typically compressed (e.g. in jpeg format)

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

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

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



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 for images or videos

- e.g. during training a neural network these information are not compressed!

Higher dimensional images

Multispectral imaging and remote sensing

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 values in the
different spectral bands

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

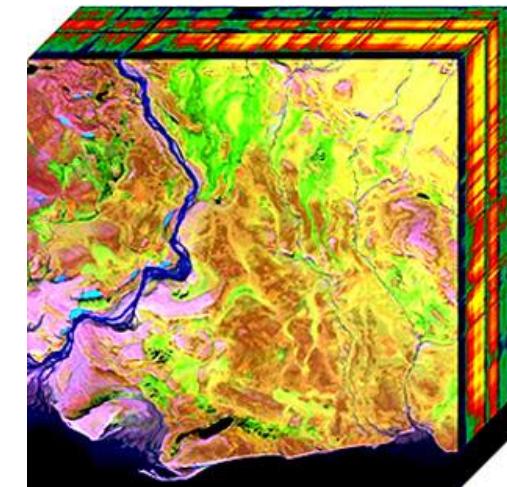
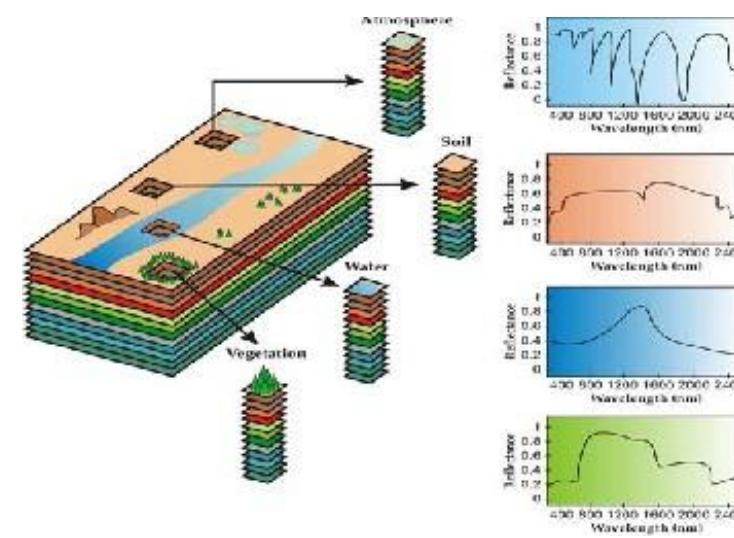
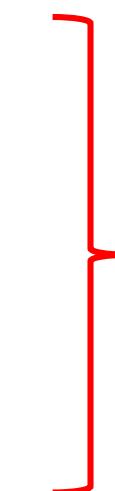
Higher dimensional Images

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



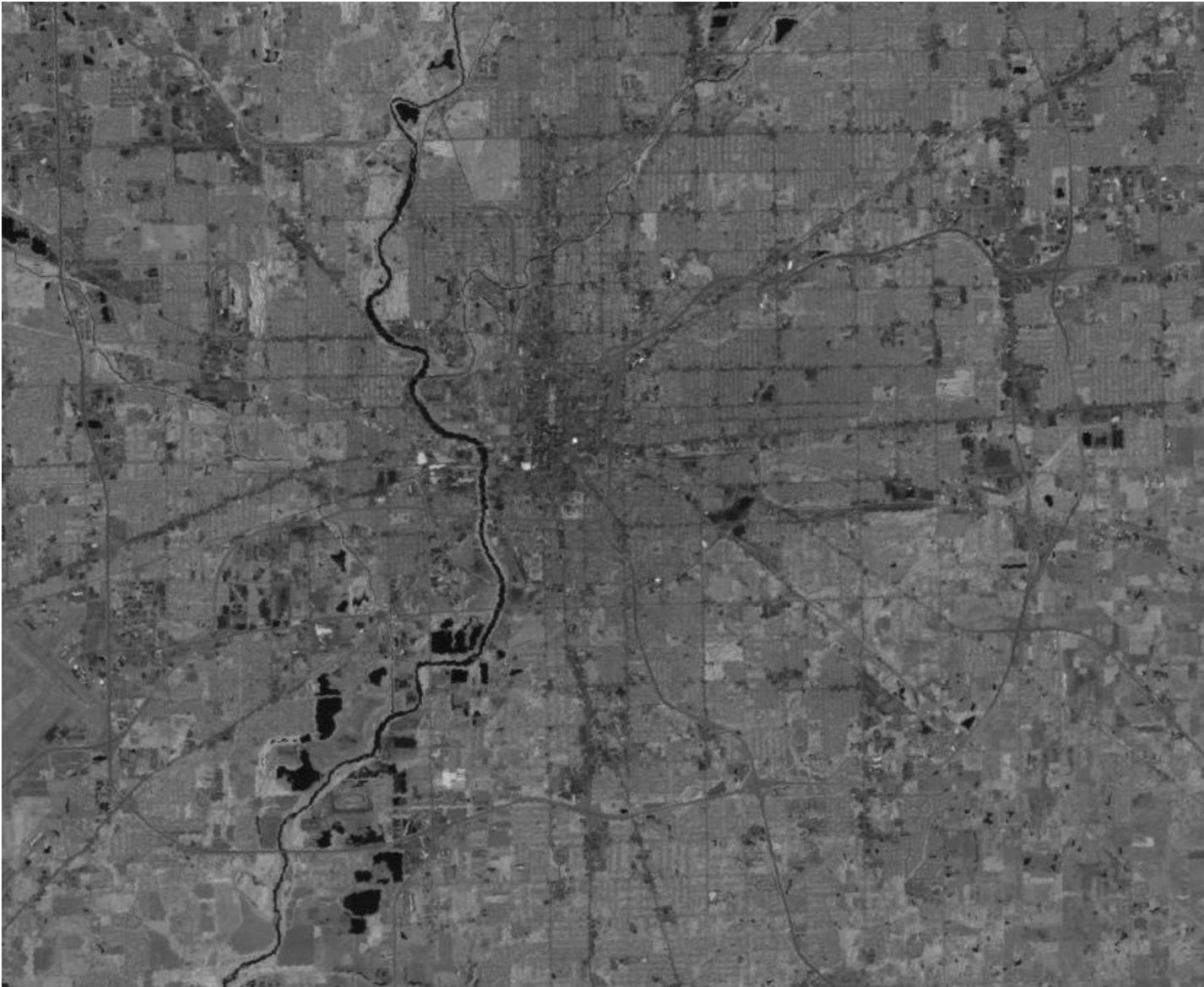
Band 2



Band 3



Band 4



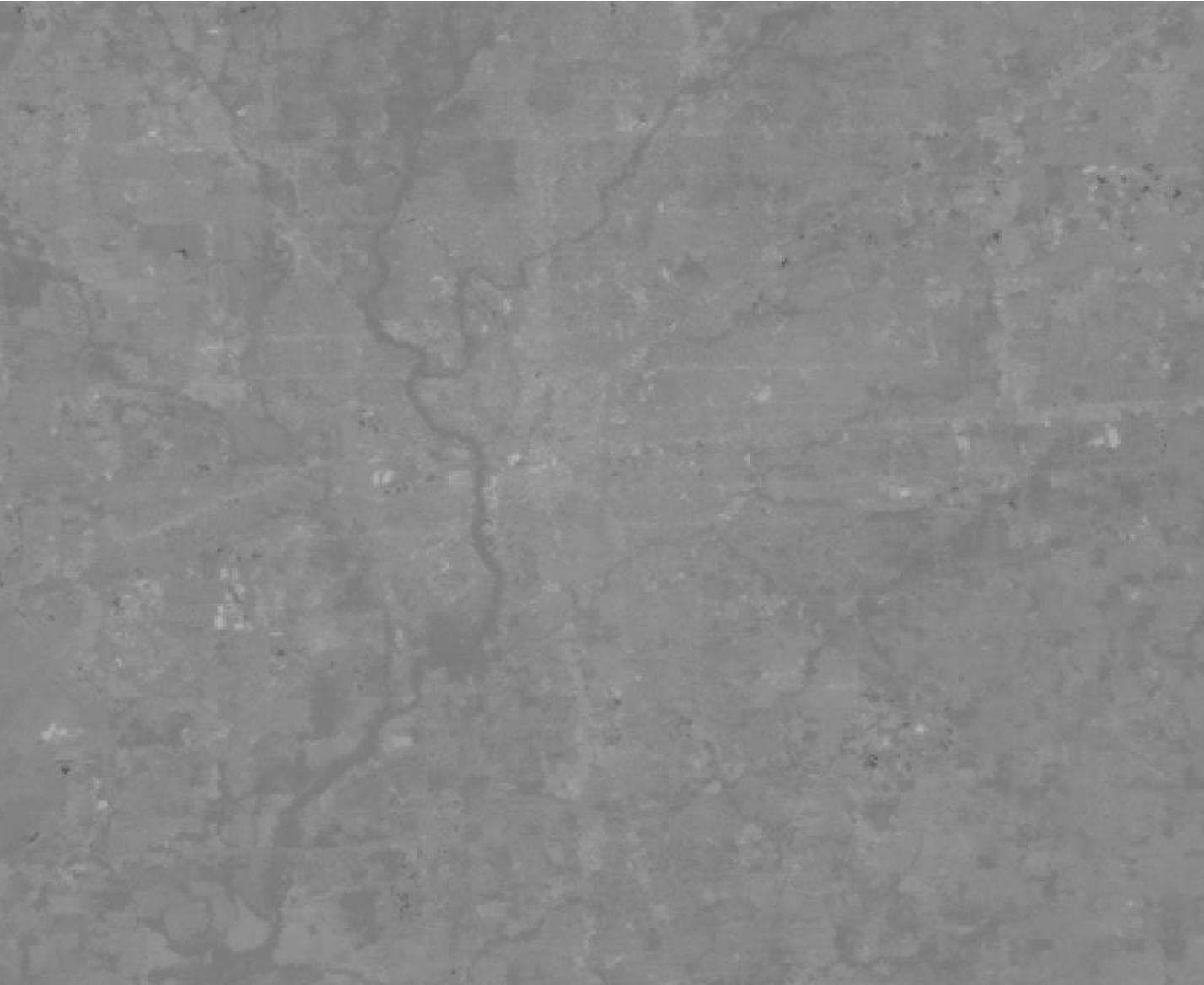
Band 5



Band 6



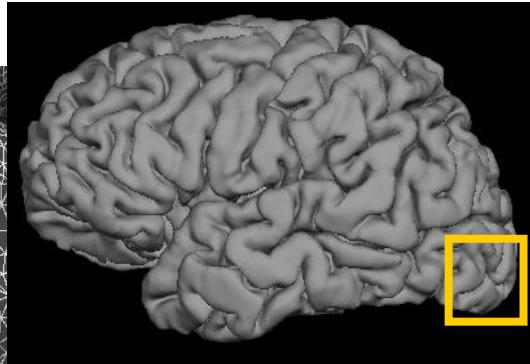
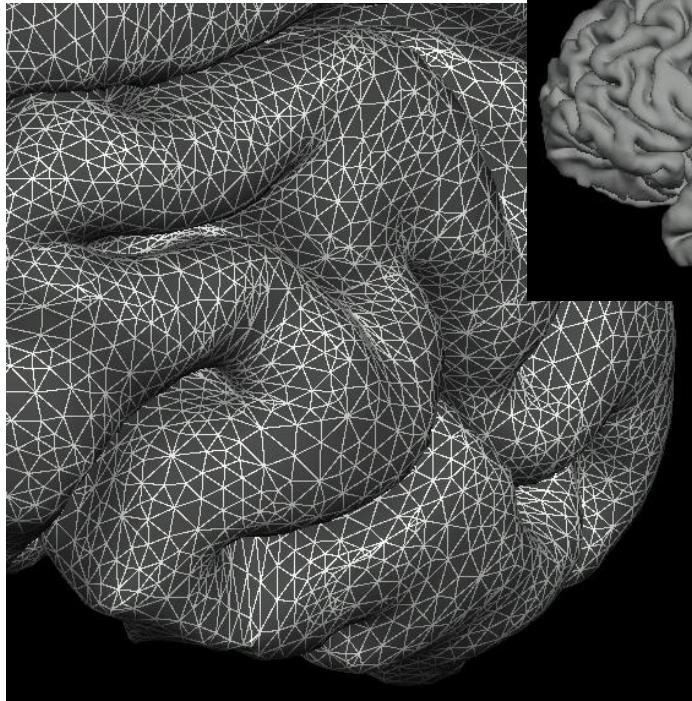
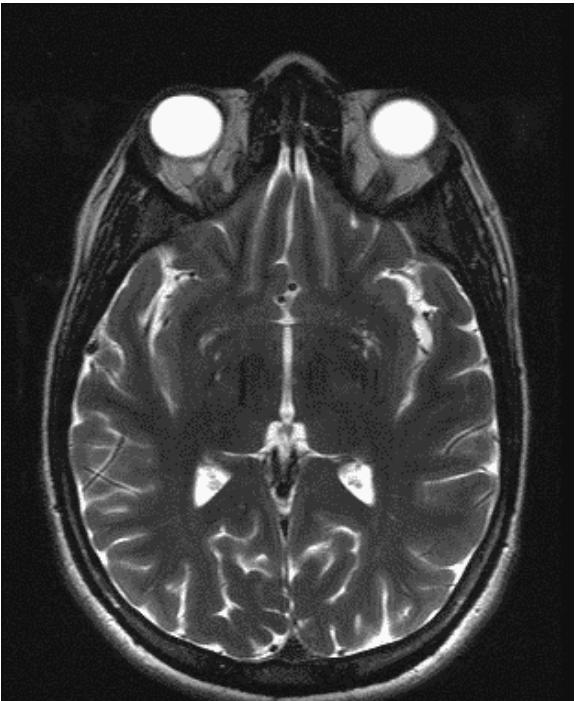
Band 7



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 surfaces



Here data points are scattered, namely they do not belong to a fixed grid. This makes learning more complicated

Local (Spatial) Transformations: Correlation

Most important image processing operations for
classification problems

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 is a function transforming the image
- U is a neighbourhood, identifies a region of the image that will concur in the output definition

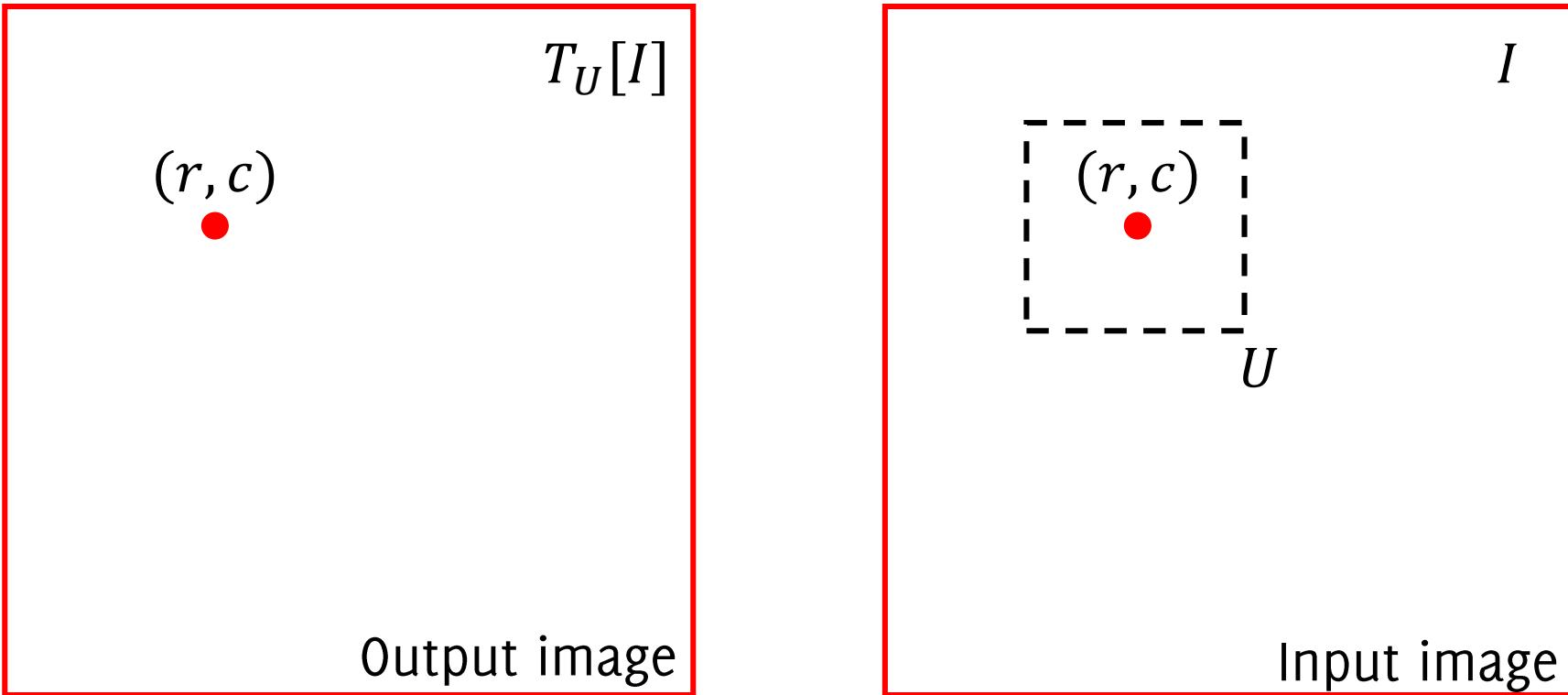
T operates on I “around” U . This means that the output at pixel (r, c) i.e., $T_U[I](r, c)$ is defined by all pixel in U in the input image

$$\{I(r + u, c + v), \quad (u, v) \in U\}$$

Such that U is described by a set of «displacements»

Spatial Filters

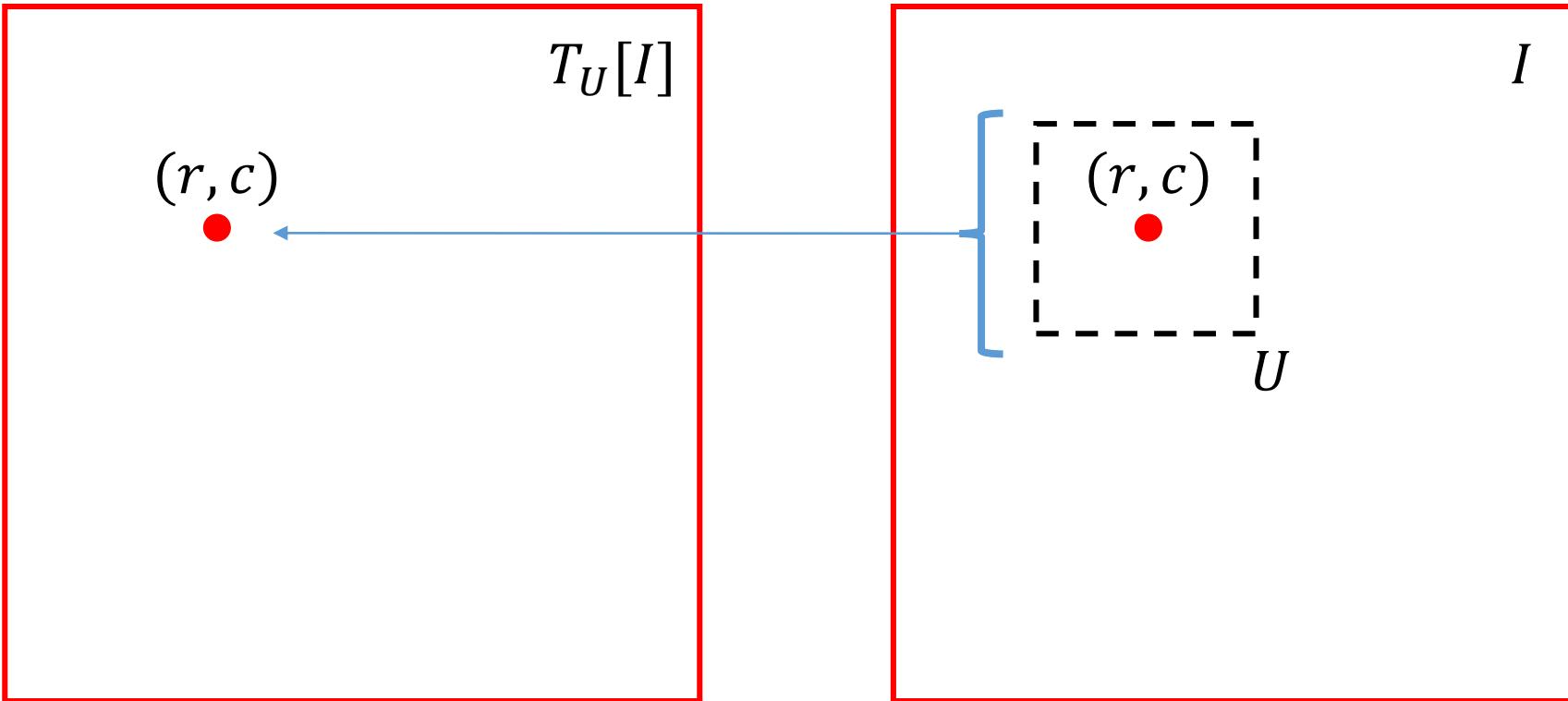
The dashed square represents $\{I(r + u, c + v), (u, v) \in U\}$



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

Spatial Filters

The dashed square represents $\{I(r + u, c + v), (u, v) \in U\}$



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

Linear Spatial Filters

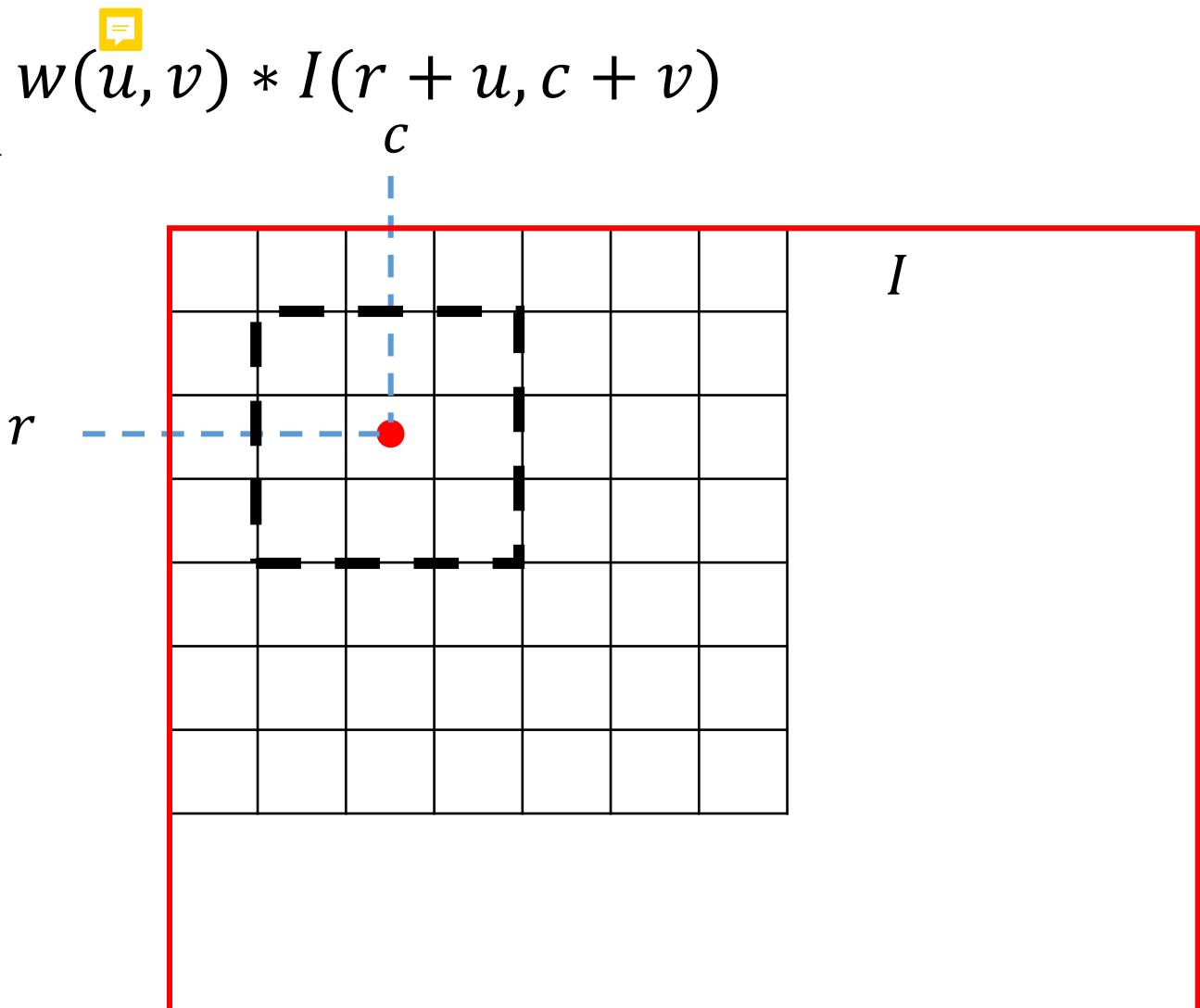
Linear Transformation: Linearity implies that

$$T[I](r, c) = \sum_{(u,v) \in U} w(u, v) * I(r + u, c + v)$$

U

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

We can consider weights
as an image, or a filter w
The filter w entirely
defines this operation



Linear Spatial Filters

Linear Transformation: Linearity implies that

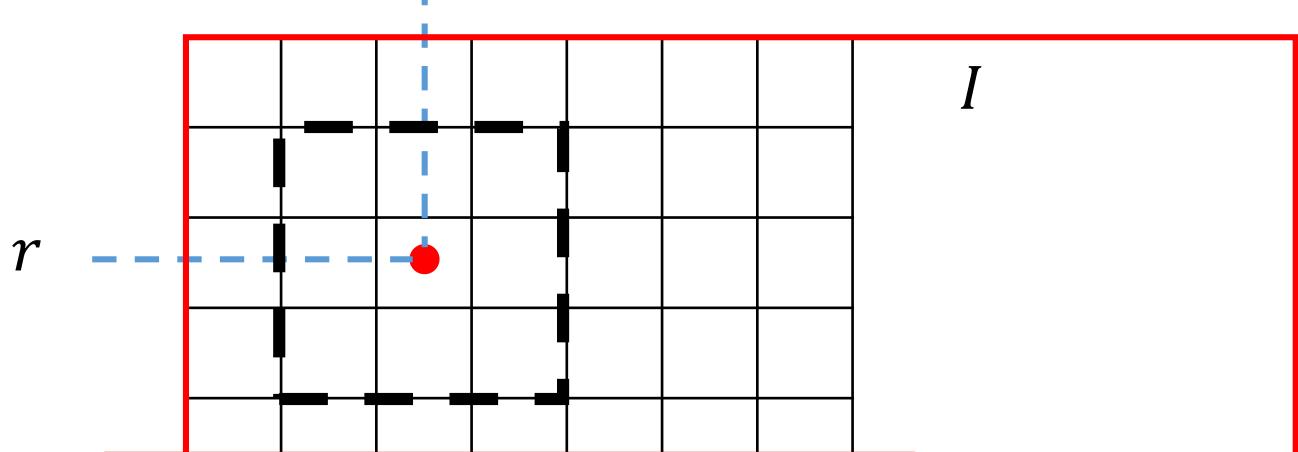
$$T[I](r, c) = \sum_{(u,v) \in U} w(u, v) * I(r + u, c + v)$$

Diagram illustrating the filter U (3x3 kernel) applied to the input image I at position (r, c) :

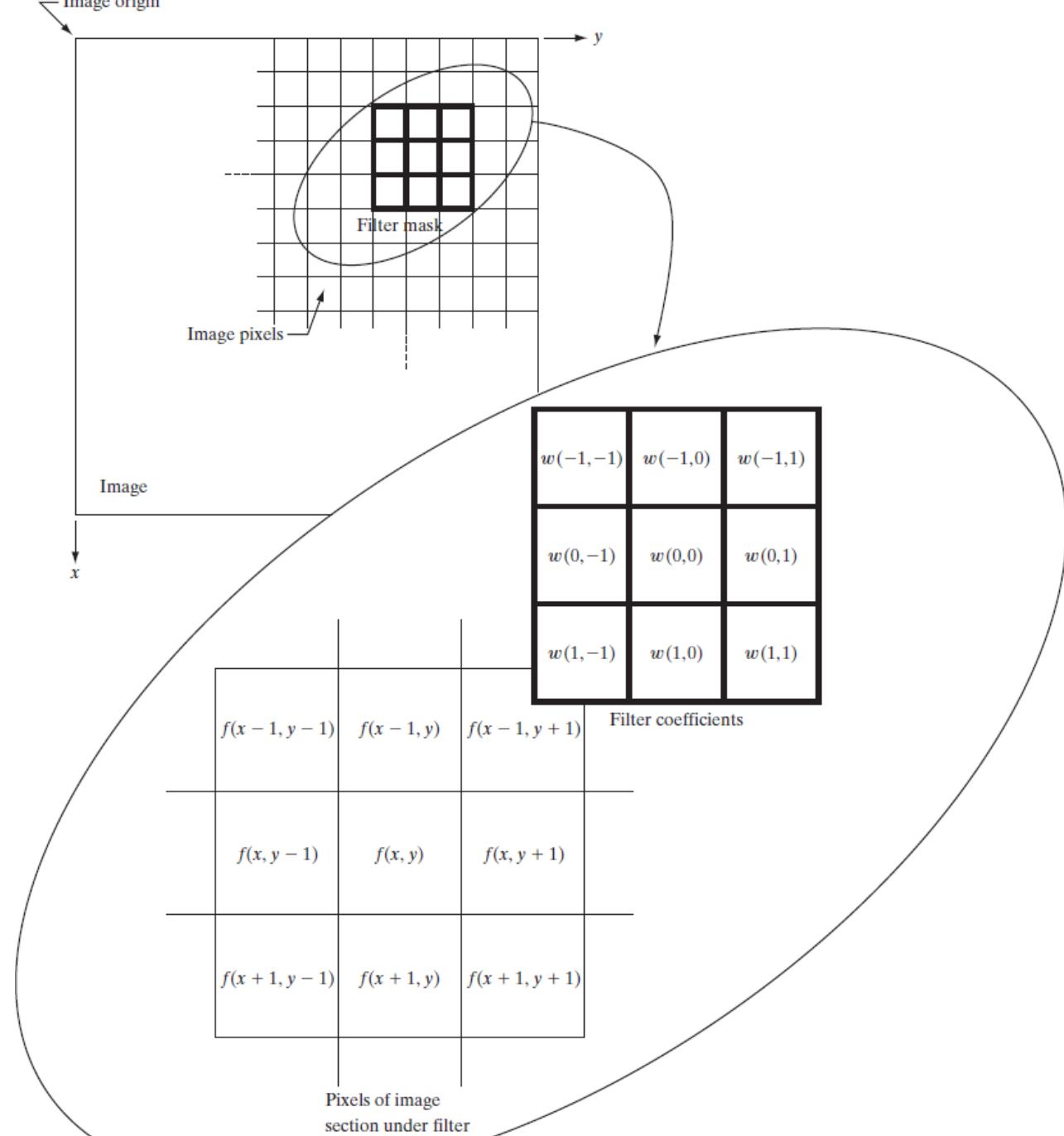
$w(-1, -1)$	$w(-1, 0)$	$w(-1, 1)$
$w(0, -1)$	$w(0, 0)$	$w(0, 1)$
$w(1, -1)$	$w(1, 0)$	$w(1, 1)$

The red dot at (r, c) indicates the center of the kernel as it slides across the input image I .

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



Correlation

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

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

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

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

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

```
from scipy import signal
```

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

Correlation for BINARY target matching

I

w

y, original image

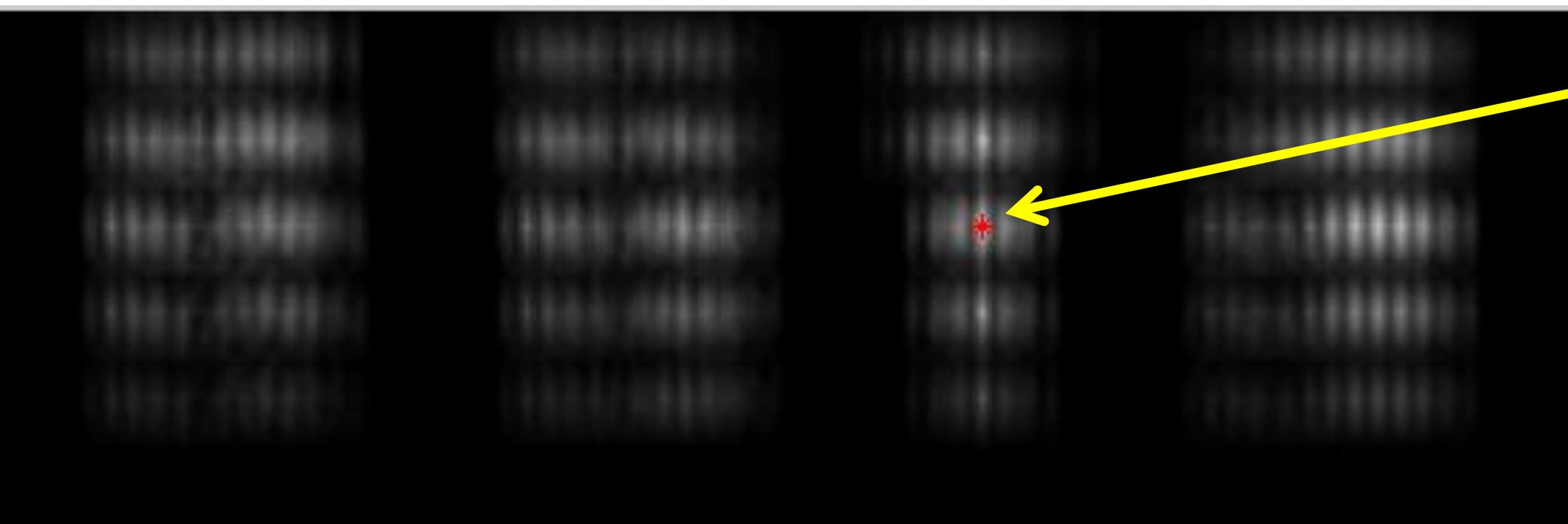
IQRM1	DIF1	Det1	#FA1
0.201	0.145	NO	2.000
0.794	0.142	NO	2.000
0.765	0.409	NO	6.000

$$\otimes \begin{matrix} \text{template} \\ \text{NO} \\ \text{NO} \\ \text{NO} \end{matrix} =$$

Easy to understand with binary images

Target used as a filter

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



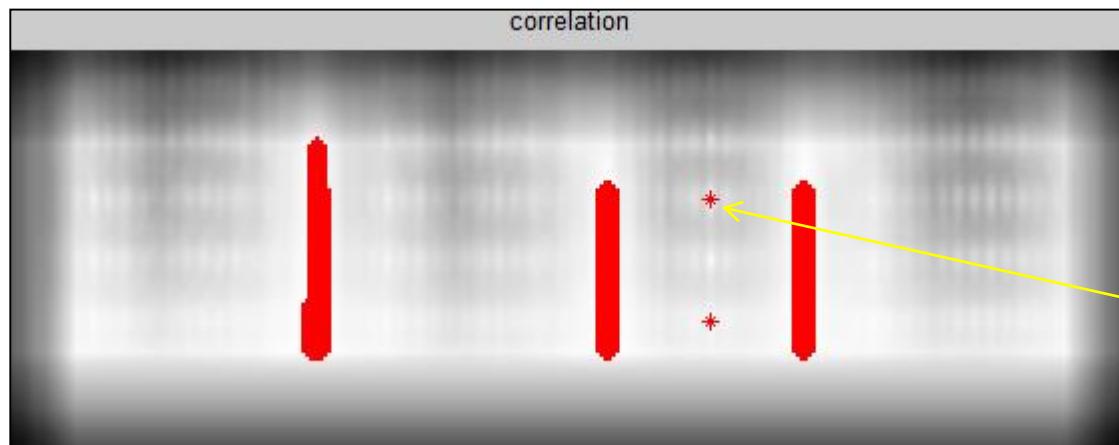
The maximum
is here



However...

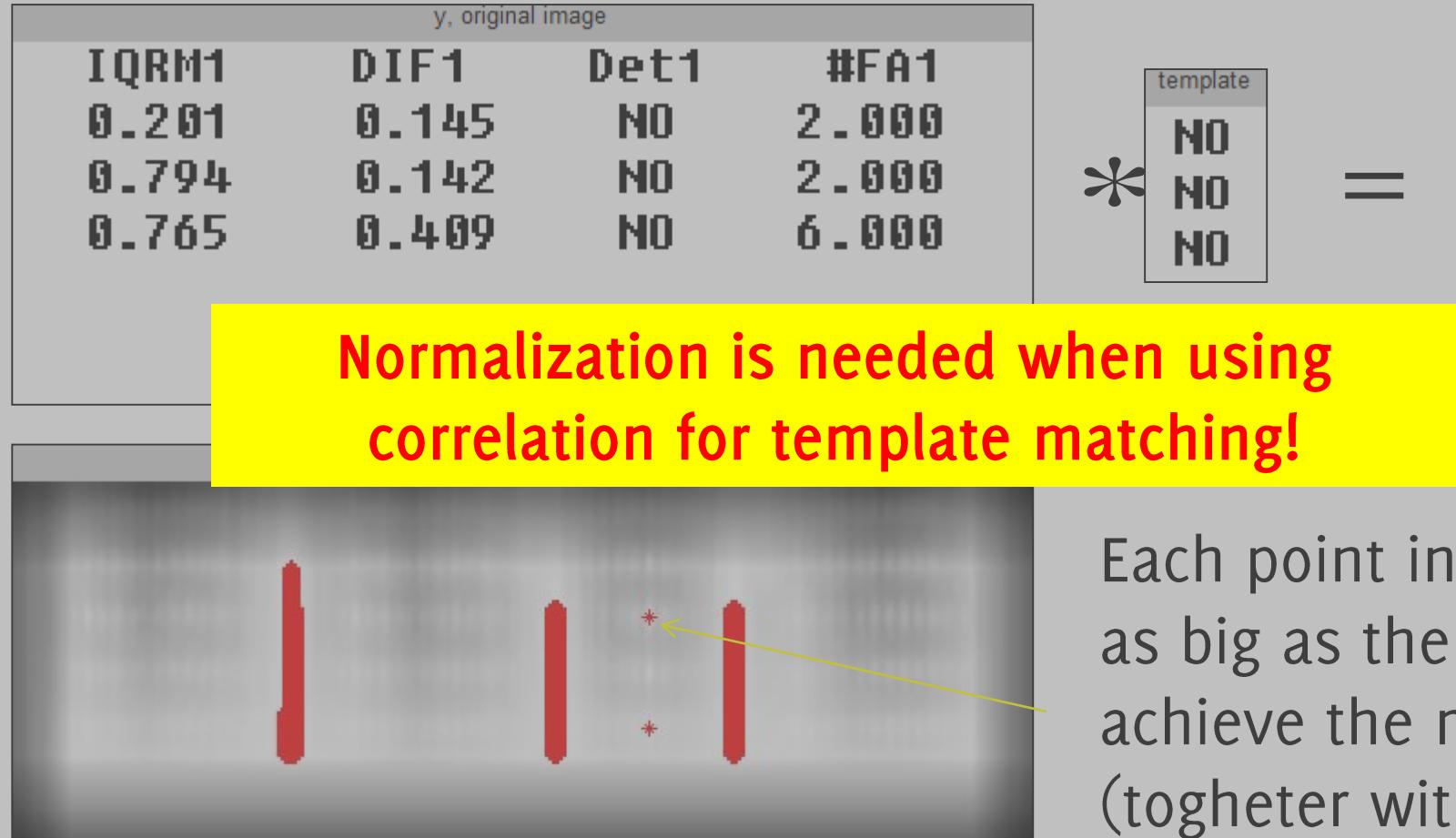
y, original image			
IQRM1	DIF1	Det1	#FA1
0.201	0.145	NO	2.000
0.794	0.142	NO	2.000
0.765	0.409	NO	6.000

$$\begin{matrix} * \\ \text{template} \\ \text{NO} \\ \text{NO} \\ \text{NO} \end{matrix} =$$



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

However...



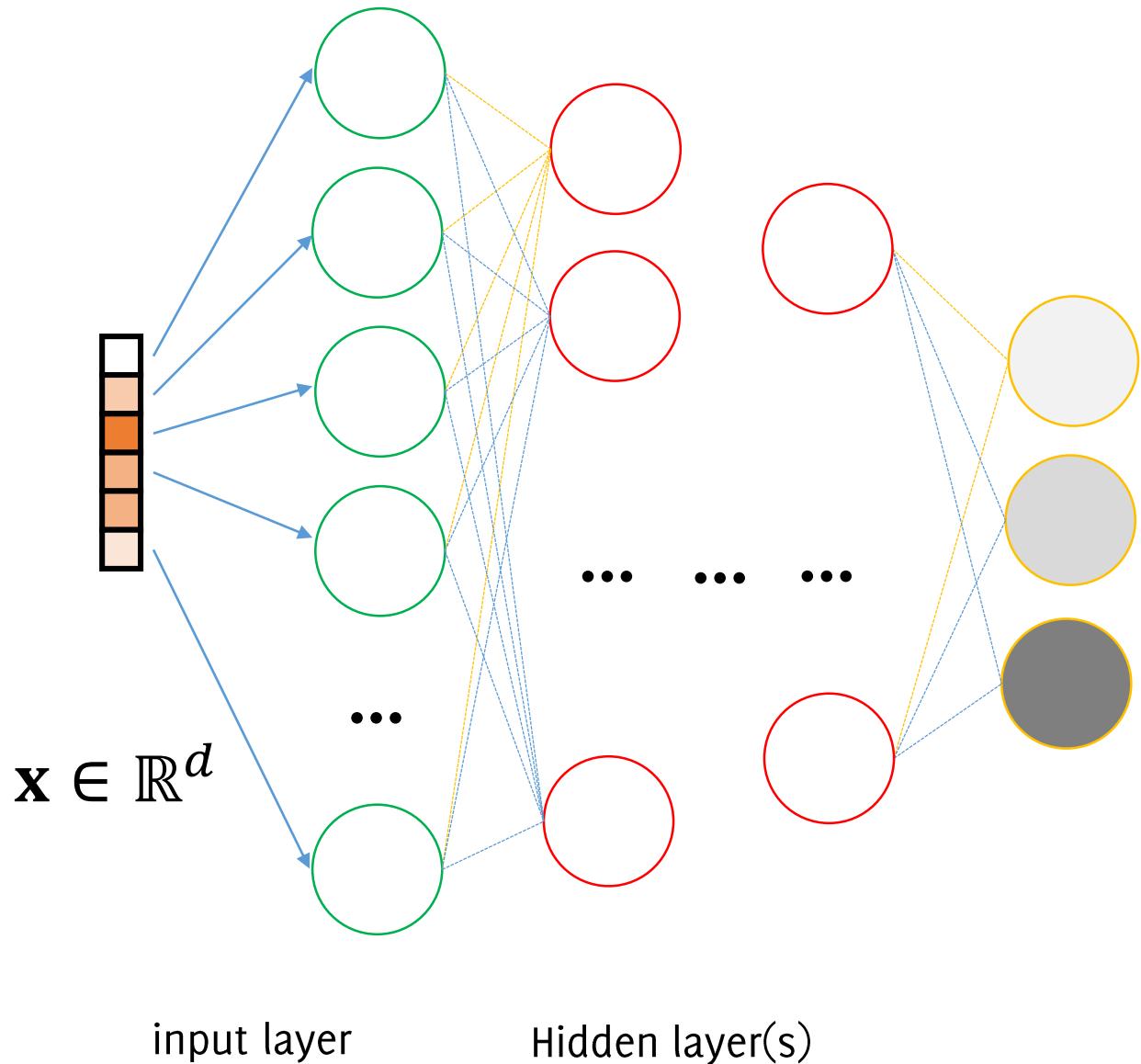
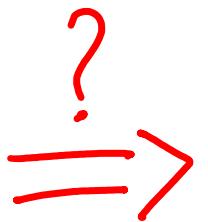
Linear Classifier

the basic building block for deep architectures

How to feed images to NN?

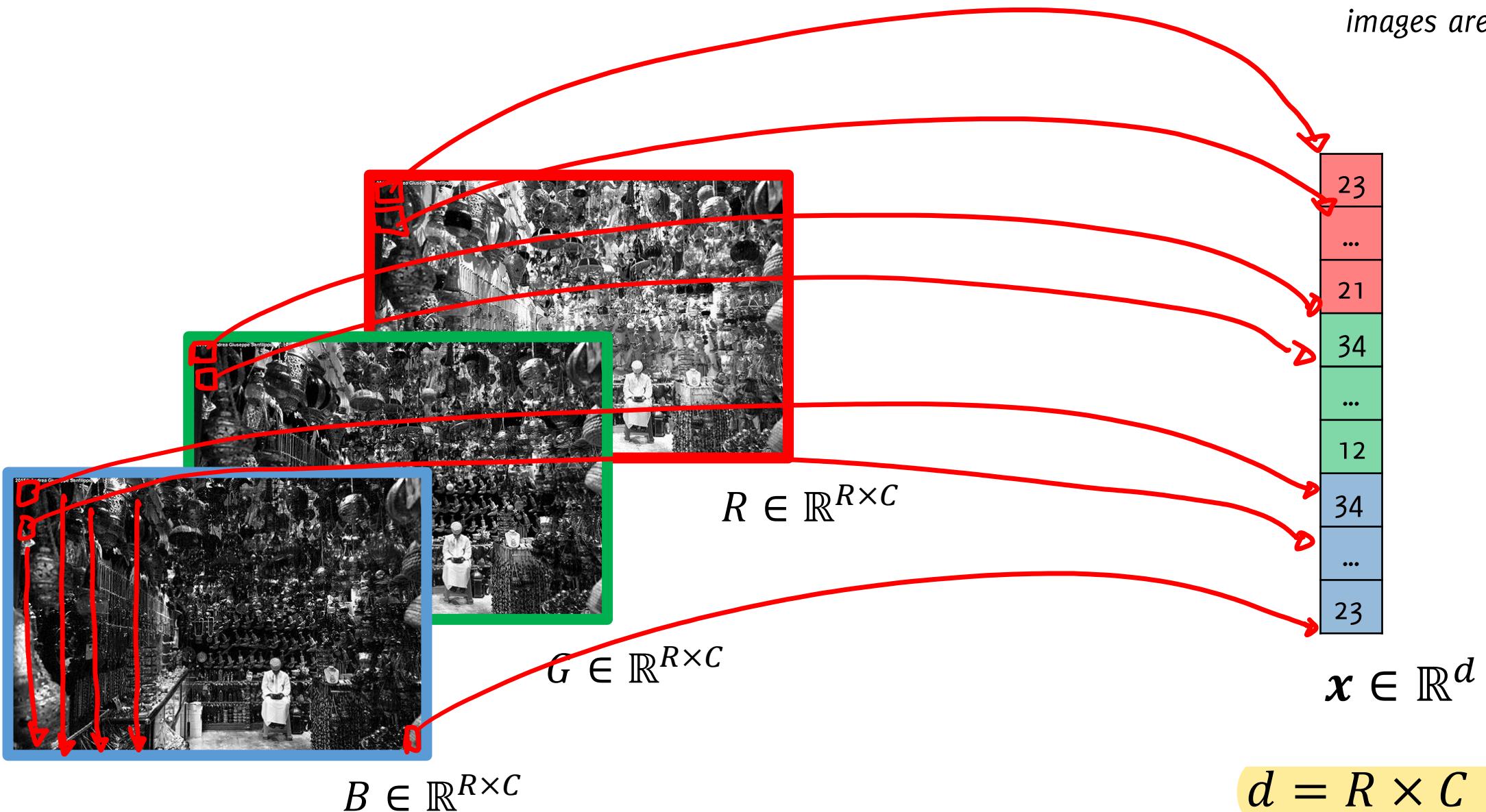


$$I \in \mathbb{R}^{R \times C \times 3}$$



Column-wise unfolding

Colors recall the color plane where images are from



$$d = R \times C \times 3$$

Classification over the CIFAR-10 dataset

The CIFAR-10 dataset contains 60000 images:

- Each image is 32x32 RGB
- Images are in 10 classes
- 6000 images per class

$$x \in \mathbb{R}^d, d = 3072$$

airplane



automobile



bird



cat



deer



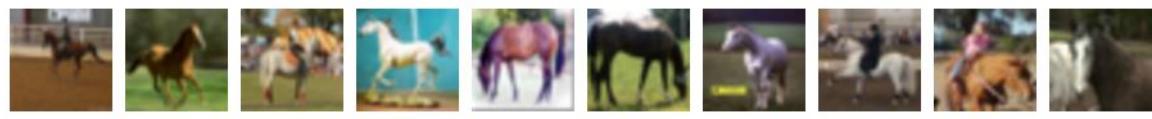
dog



frog



horse



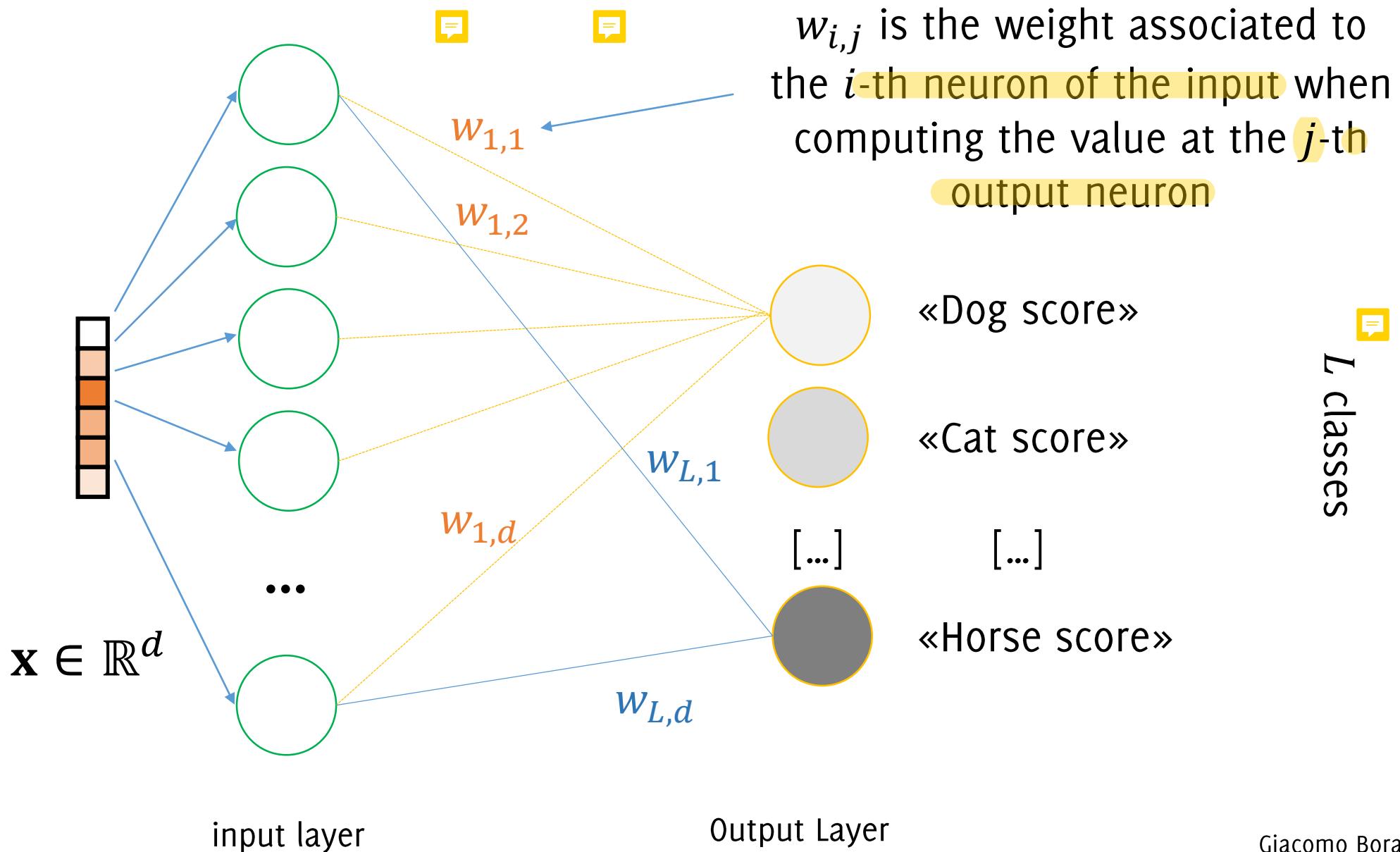
ship



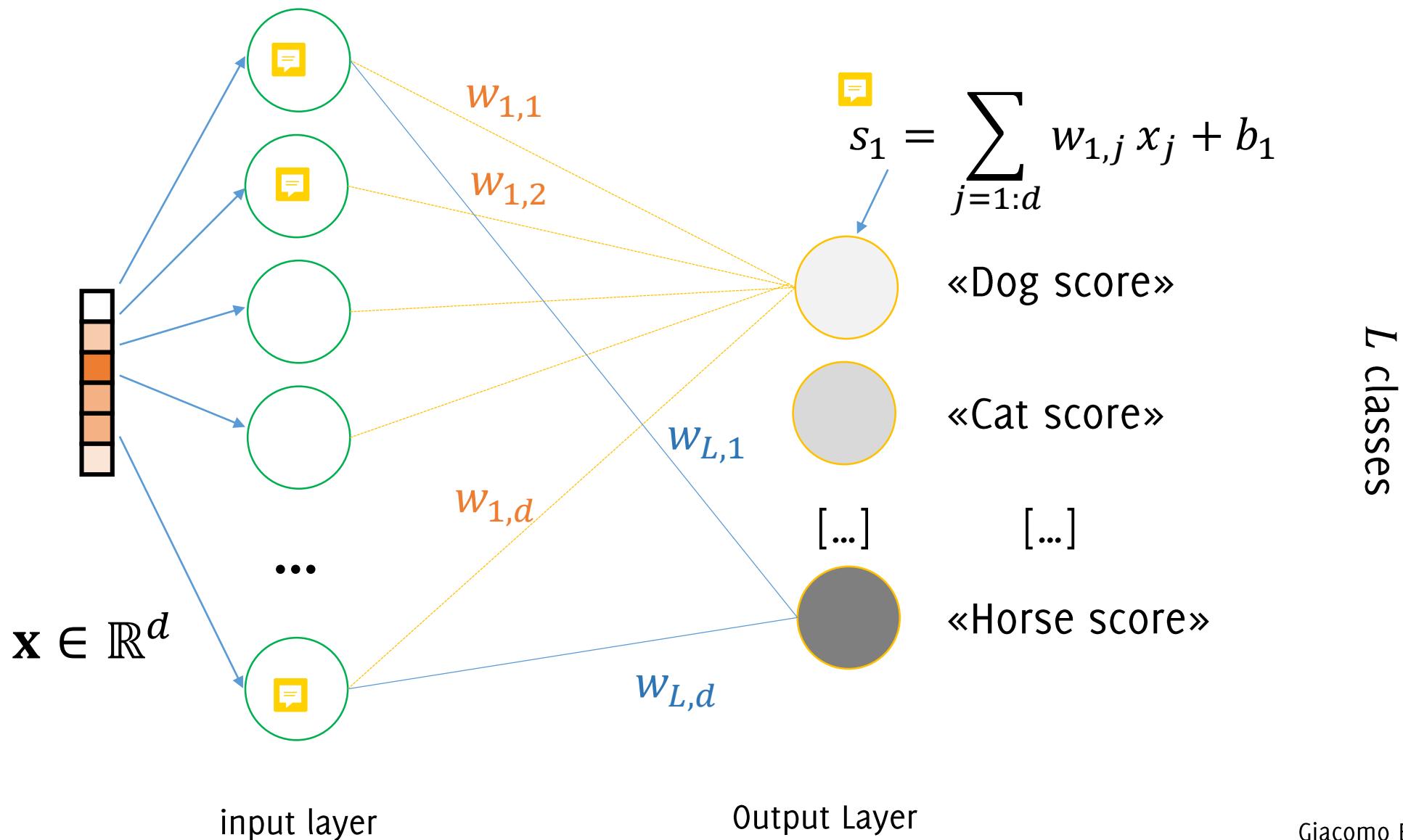
truck



A 1-layer NN to Classify Images



A 1-layer NN to Classify Images



```
model.summary();
```

Layer (type)	Output Shape	Param #
Input (InputLayer)	[(None, 32, 32, 3)]	0
Flatten (Flatten)	(None, 3072)	0
Output (Dense)	(None, 10)	30730 

Total params: 30,730

Trainable params: 30,730

Non-trainable params: 0

Why don't we take a larger network?

Dimensionality prevents us from using in a straightforward manner deep NN as those seen so far.

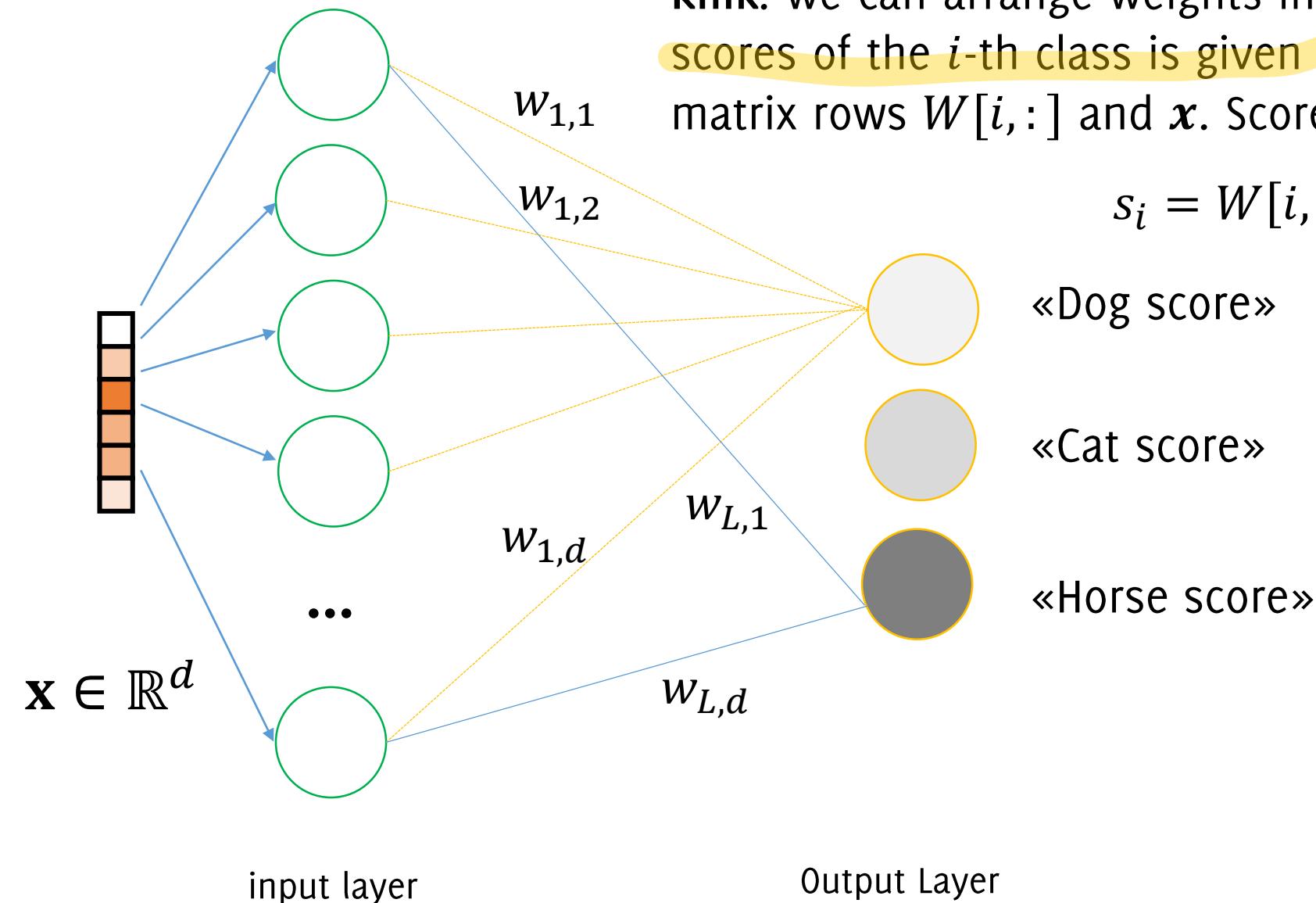
Let's take a network with **an hidden layer having half of the neurons of the input layer**.

On CIFAR10 images, the number of neurons would be:

- 3072 first layer
- 1536 second layer
- 10 output layer

$$\left. \begin{array}{l} 1,536 * 3,072 + 1,536 \\ 10 * 1,536 + 10 \end{array} \right\} = 4,720,128 \text{ parameters (!)}$$
$$= 15,370 \text{ parameters}$$

A 1-layer NN to Classify Images



Rmk: we can arrange weights in a matrix $W \in \mathbb{R}^{L \times d}$, then the scores of the i -th class is given by inner product between the matrix rows $W[i, :]$ and \mathbf{x} . Scores then becomes:

$$s_i = W[i, :] * \mathbf{x} + b_i$$

L classes

Rmk: nonlinearity is not needed here since there are no layers following

Rmk: we can also ignore the softmax in the output since this would not change the order of the scores (would just normalize them)

A 1-layer NN to Classify Images

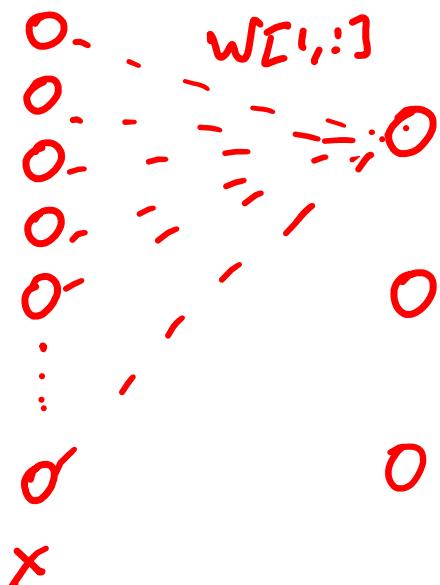
$$W \in \mathbb{R}^{L \times d}$$

$w[1,:]$	-8.1	...	2.7	9.5	...	-9.0	-5.4	...	4.8
$w[2,:]$	9.0	...	5.4	4.8	...	1.2	9.5	...	-8.0
$w[3,:]$	1.2	...	9.5	-8.0	...	8.1	-2.7	...	9.5

$$\begin{array}{c} * \\ + \\ = \end{array} \begin{array}{c} 23 \\ \dots \\ 21 \\ 34 \\ \dots \\ 12 \\ 34 \\ \dots \\ 23 \end{array} \begin{array}{c} -2 \\ 32 \\ -1 \end{array} \begin{array}{c} -4 \\ 22 \\ 33 \end{array} \begin{array}{l} s_1 \text{ dog score} \\ s_2 \text{ cat score} \\ s_3 \text{ horse score} \end{array}$$

$\mathbf{b} \quad \mathcal{K}(x; W, \mathbf{b})$

Rmk: colors indicate to which color plane in the image these weights refer to



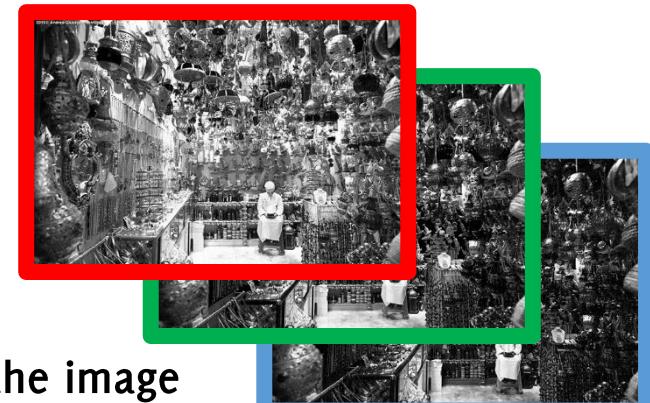
$$s_1 = w[1,:].x + b_1$$

$$s_2 = w[2,:].x + b_2$$

$$s_3 = w[3,:].x + b_3$$

x

Unroll the image column-wise



This simple layer is a linear classifier

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

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

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 vector.

-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

$$\begin{array}{c} * \\ + \\ = \end{array} \begin{array}{c} \begin{matrix} 23 \\ \dots \\ 21 \\ 34 \\ \dots \\ 12 \\ 34 \\ \dots \\ 23 \end{matrix} \\ b \end{array} \begin{array}{c} \begin{matrix} -2 \\ 32 \\ -1 \end{matrix} \\ \mathcal{K}(x; W, b) \end{array} \begin{array}{l} s_1 \text{ dog score} \\ s_2 \text{ cat score} \\ s_3 \text{ rabbit score} \end{array}$$

Unroll the image
column-wise



This simple layer is a linear classifier

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

$$\hat{y}_j = \operatorname{argmax}_{i=1,\dots,L} [s_j]_i$$

being $[s_j]_i$ the i -th component of the vector

$$\mathcal{K}(x) = s = Wx + b$$

-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
...

+

-2
32
-1
33

=

-4
22
33

s_1 dog score
 s_2 cat score
 s_3 horse score

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

x

Rmk: softmax is not needed as long as we take as output the largest score: this would be the one yielding the largest posterior

The Parameters of a Linear Classifier

The score of a class is the weighted sum of all the image pixels.

Weights are actually the classifier parameters.

The weights are:

-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

b

and indicate which are the most important pixels / colours

Why nonlinear layers?

Each layer in a NN can be seen as matrix multiplication (+ bias).

$$\mathbf{s} = W\mathbf{x} + \mathbf{b}$$

If we stack 3 layers without activations:

$$\mathbf{s} = ((W_1\mathbf{x} + \mathbf{b}_1)W_2 + \mathbf{b}_2)W_3 + b_3$$

This becomes equivalent to

$$\mathbf{s} = W\mathbf{x} + \mathbf{b}$$

This is a further confirmation why it becomes pointless to stack many layers without including a nonlinear activations...

Training the Linear Classifier

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

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

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 gradient descent and all its variants
(see Prof. Matteucci classes)

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

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

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

... Once Trained

The training data is used to learn the parameters W, b

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

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

-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

b

Geometric Interpretation of a Linear Classifier

$\mathbf{W}[\mathbf{i}, :]$ 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 (and summing the bias)

$$\mathbf{W}[\mathbf{i}, :] * \mathbf{x} + b_i$$

Thus, the NN computes the inner products against L different weights vectors, and selects the one yielding the largest score (up to bias correction)

Rmk: these “inner product classifiers” operate independently, and the output of the j -th row is not influenced by weights at a different row

Rmk: this would not be the case if the network had hidden layer that would mix the outputs of intermediate layers

Geometric Interpretation of a Linear Classifier

In Python notation:

In Python * denotes the element-wise product, here I mean the inner product of vectors:

$$\text{np.inner}(W[i, :], \mathbf{x}) + b_i$$

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

$$f([x_1, x_2]) = w_1 x_1 + w_2 x_2 + b$$

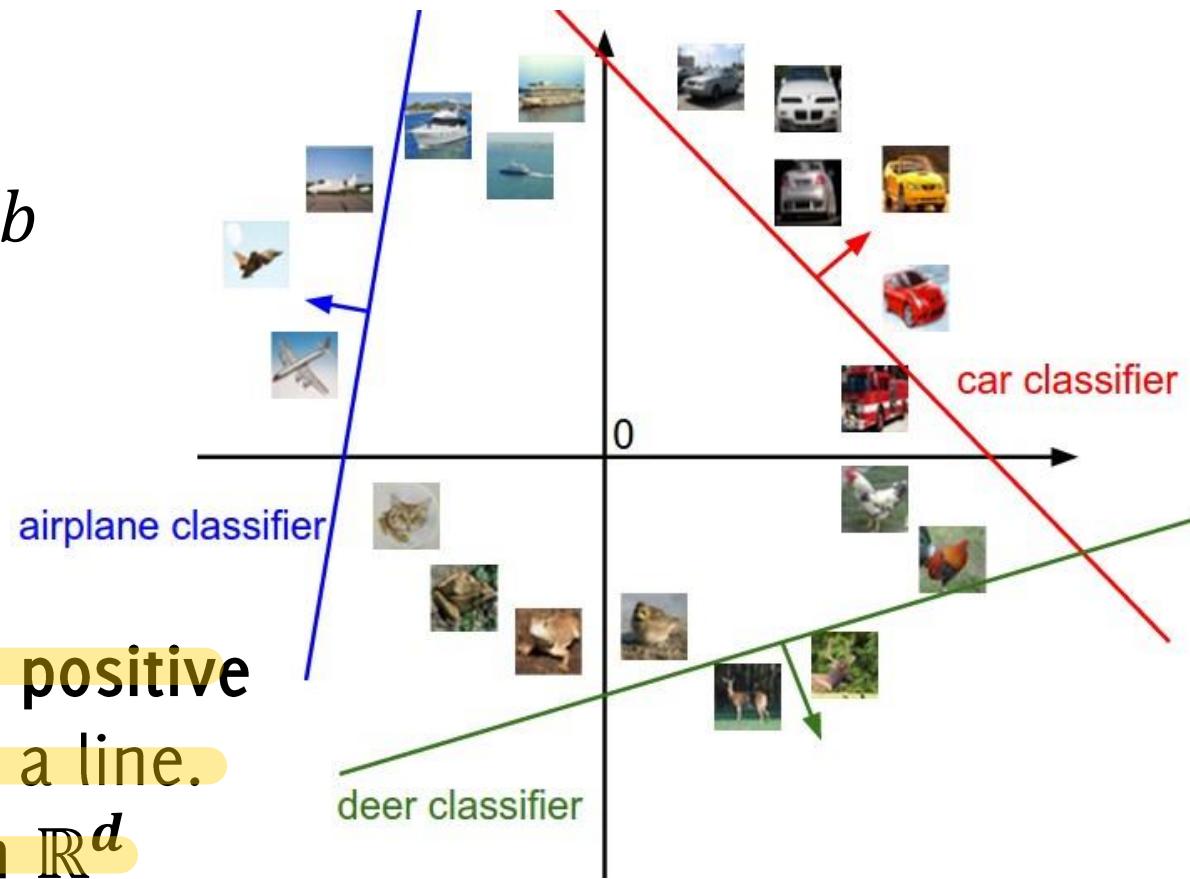
Then, points $[x_1, x_2]$ yielding

$$f([x_1, x_2]) = 0$$

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

In Python 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

$$W[i, :] * \mathbf{x} + b_i$$

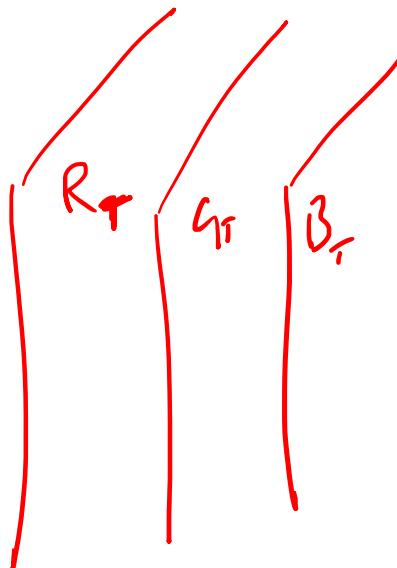
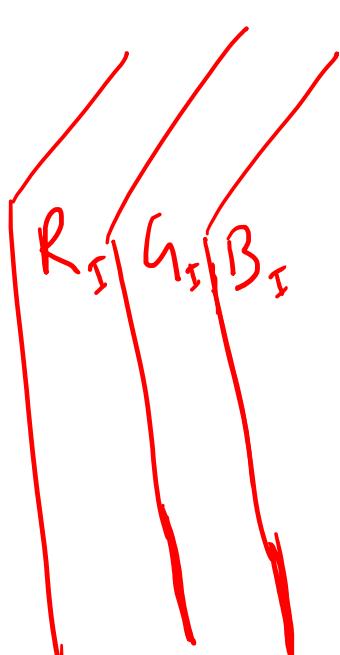
Then, $W[i, :]$ can be seen as a template used in matching (the output of correlation in the central pixel)

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

Correlation between two RGB images

The image and the filter have the same size



$$(R_I * R_T)(0,0) + \\ (G_I * G_T)(0,0) + \\ (B_I * B_T)(0,0)$$

$w_1 \cdot x$

$$R_I \odot R_T + G_I \odot G_T + B_I \odot B_T = \\ \sum_{x=1}^n R_I(x,y) \cdot R_T(x,y) + \dots$$

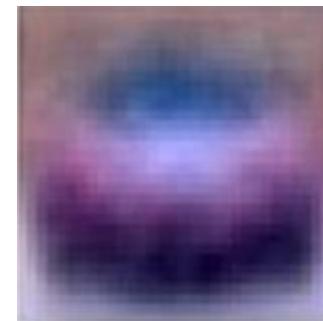
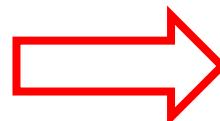
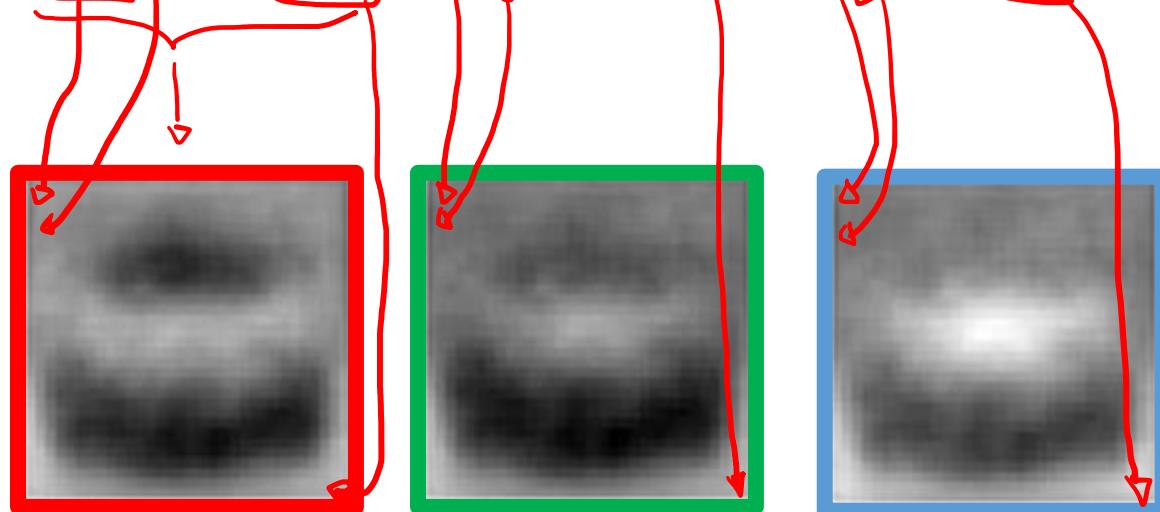
Bring the classifier weights back to images

W

-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

$$d = R \times C \times 3$$

$W[i, :] \in \mathbb{R}^d$, car classifier



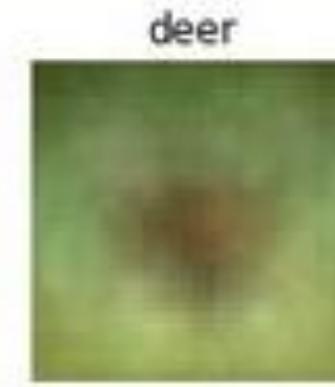
$$R \in \mathbb{R}^{R \times C}$$

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

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

car template in $\mathbb{R}^{R \times C \times 3}$

Templates Learned on the CIFAR-10 dataset



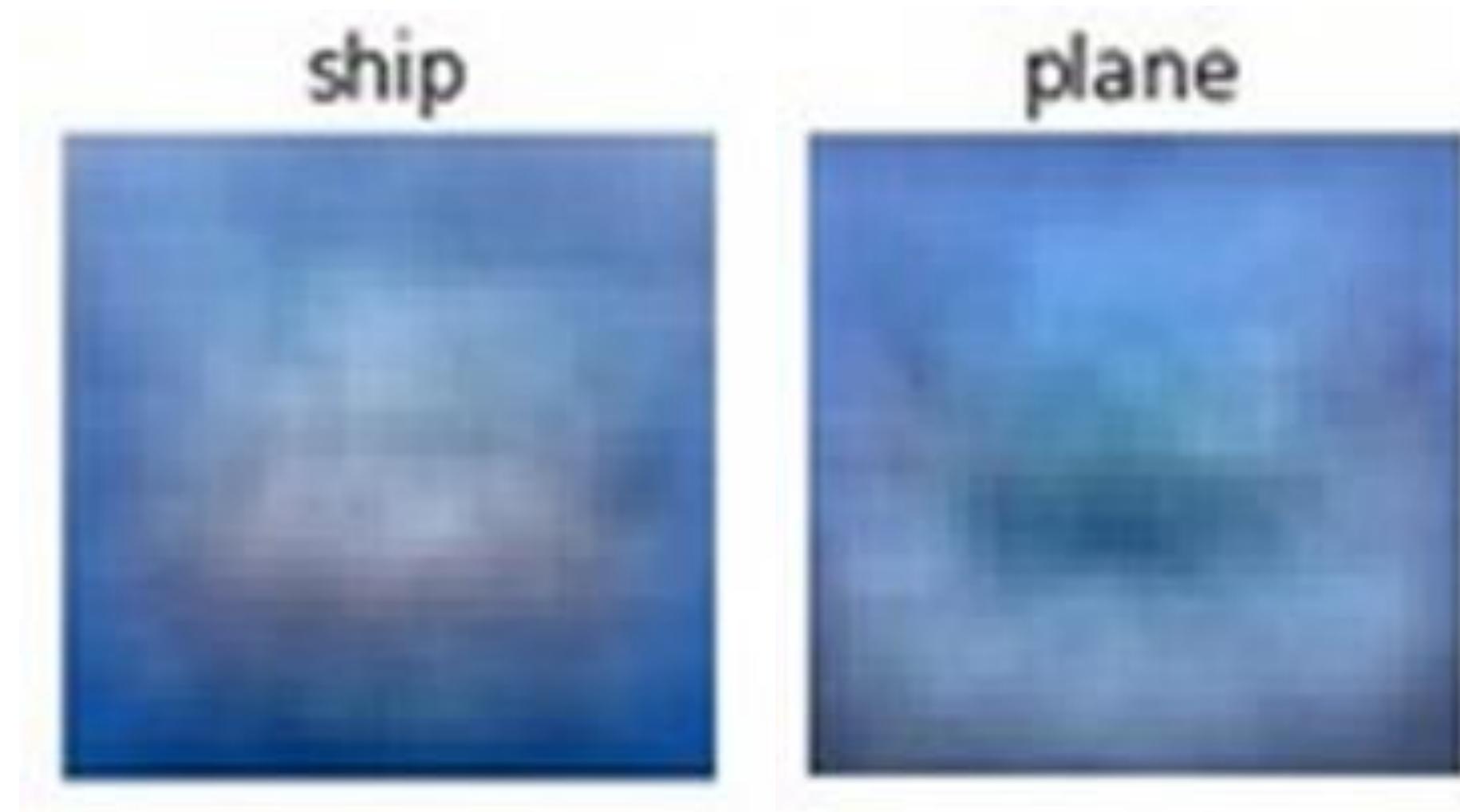
The Class Score

The classification score is then computed as the correlation between each input image and the «template» of the corresponding class



$$(I \otimes T_1)(0,0) = \sum_{(x,y) \in U} T_1(x,y) * I(x,y)$$

Templates Learned on the CIFAR-10 dataset



Templates Learned on the CIFAR-10 dataset

car

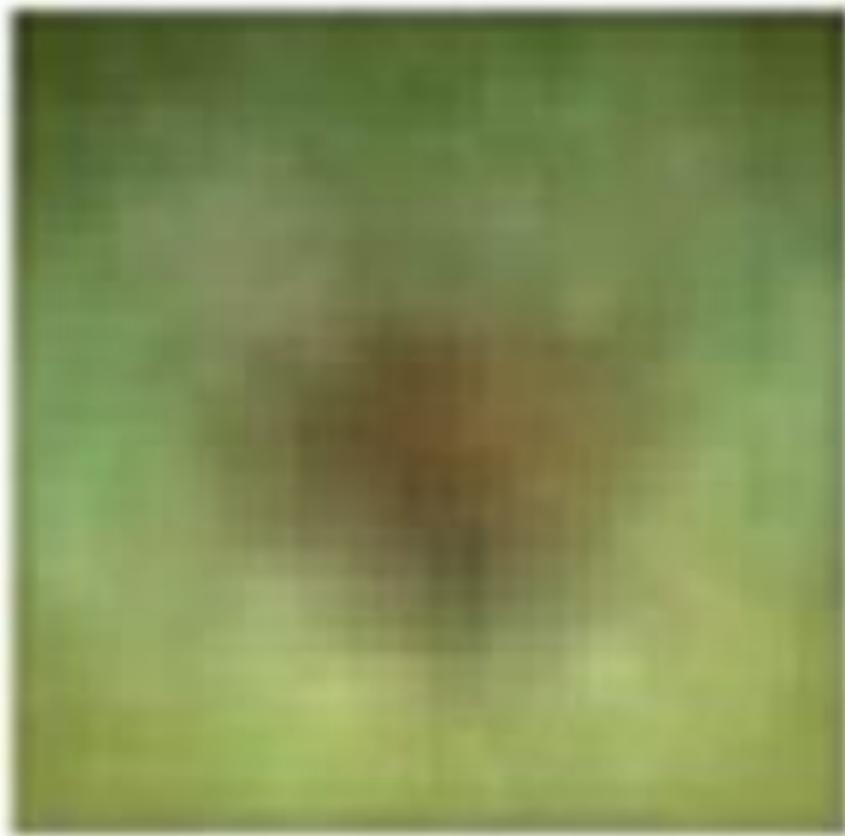


truck



Templates Learned on the CIFAR-10 dataset

deer



bird



Templates Learned on the CIFAR-10 dataset

horse



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 definitely 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 green (and boat is blue)
- Cars are typically red
- Horses have two heads! 😊

The model was definitively trained to achieve higher performance and classification rates

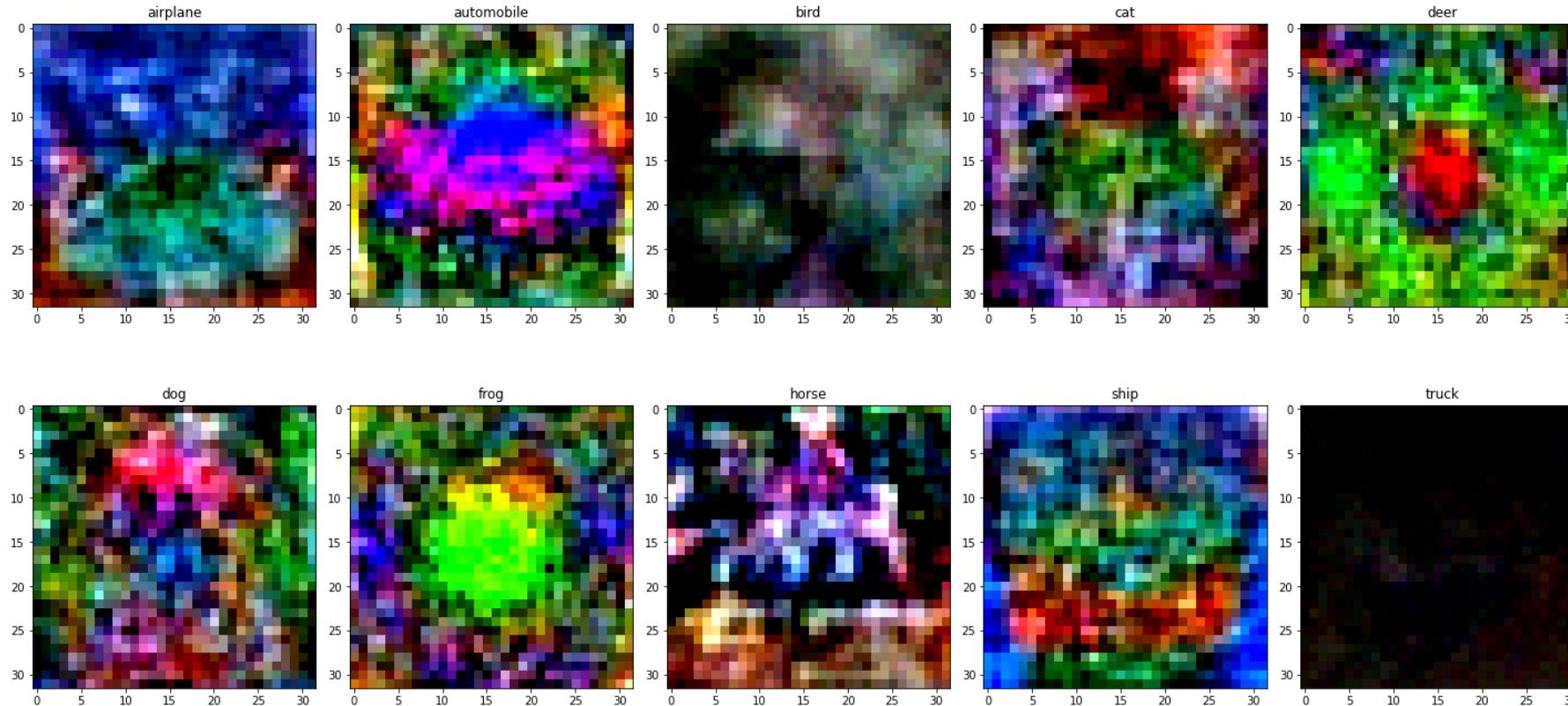
However:

- 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)

There should be a better way
for handling images

Do it yourself!

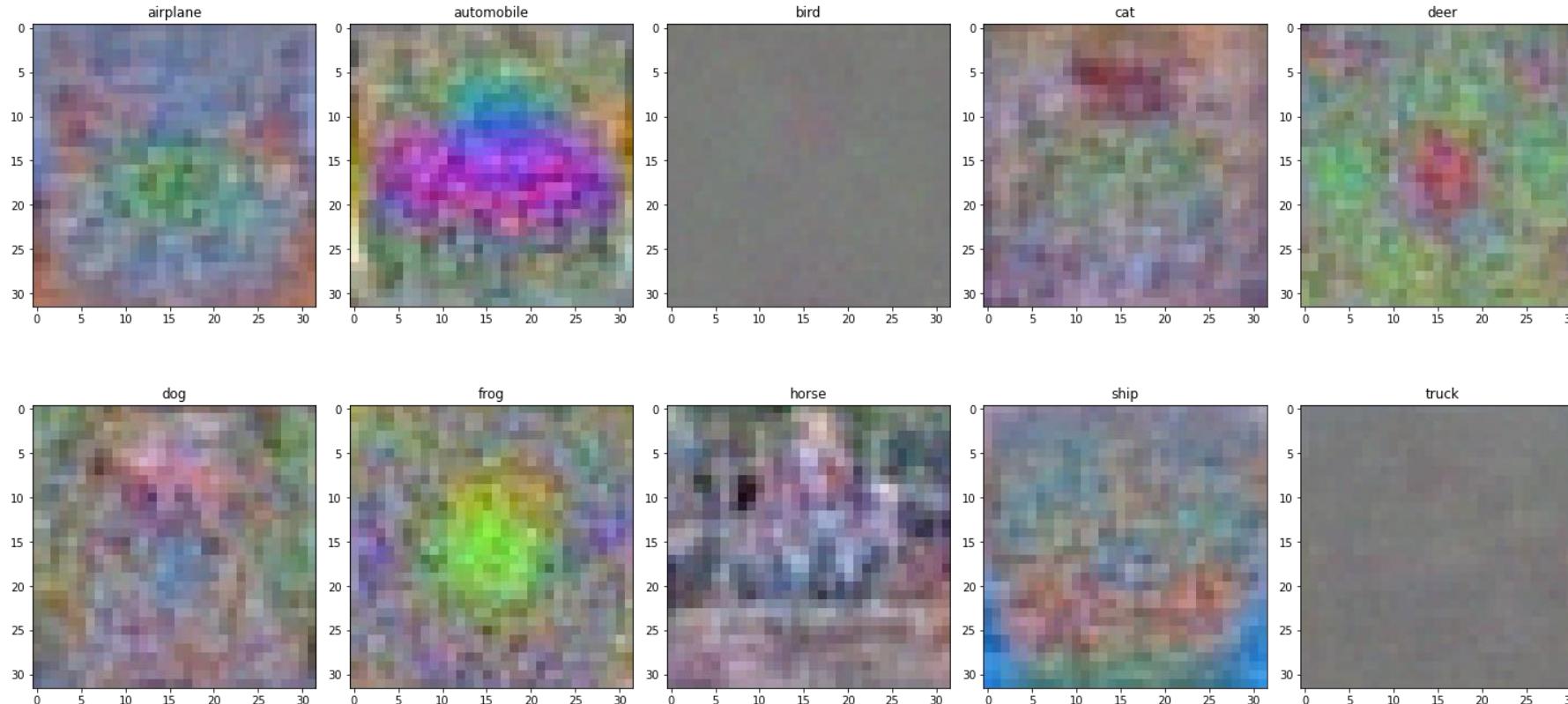
https://colab.research.google.com/drive/1MoXvvDm_eMEeHBussa2SeiLmUBy5g_7T?usp=sharing



Credits Eugenio Lomurno! (visualization with saturated colors)

Do it yourself!

https://colab.research.google.com/drive/1MoXvvDm_eMEeHBussa2SeiLmUBy5g_7T?usp=sharing



Credits Eugenio Lomurno! (visualization with clipped colors)

Is Image Classification a Challenging Problem?

Yes, it is...

Is Image Classification a Challenging Problem?

What does prevent us from

- Stacking multiple hidden layers
- Training the network with a huge amount of images

to successfully classify images?

Would this work?

Is it a challenging problem?

First challenge: dimensionality

Images are very high-dimensional image data

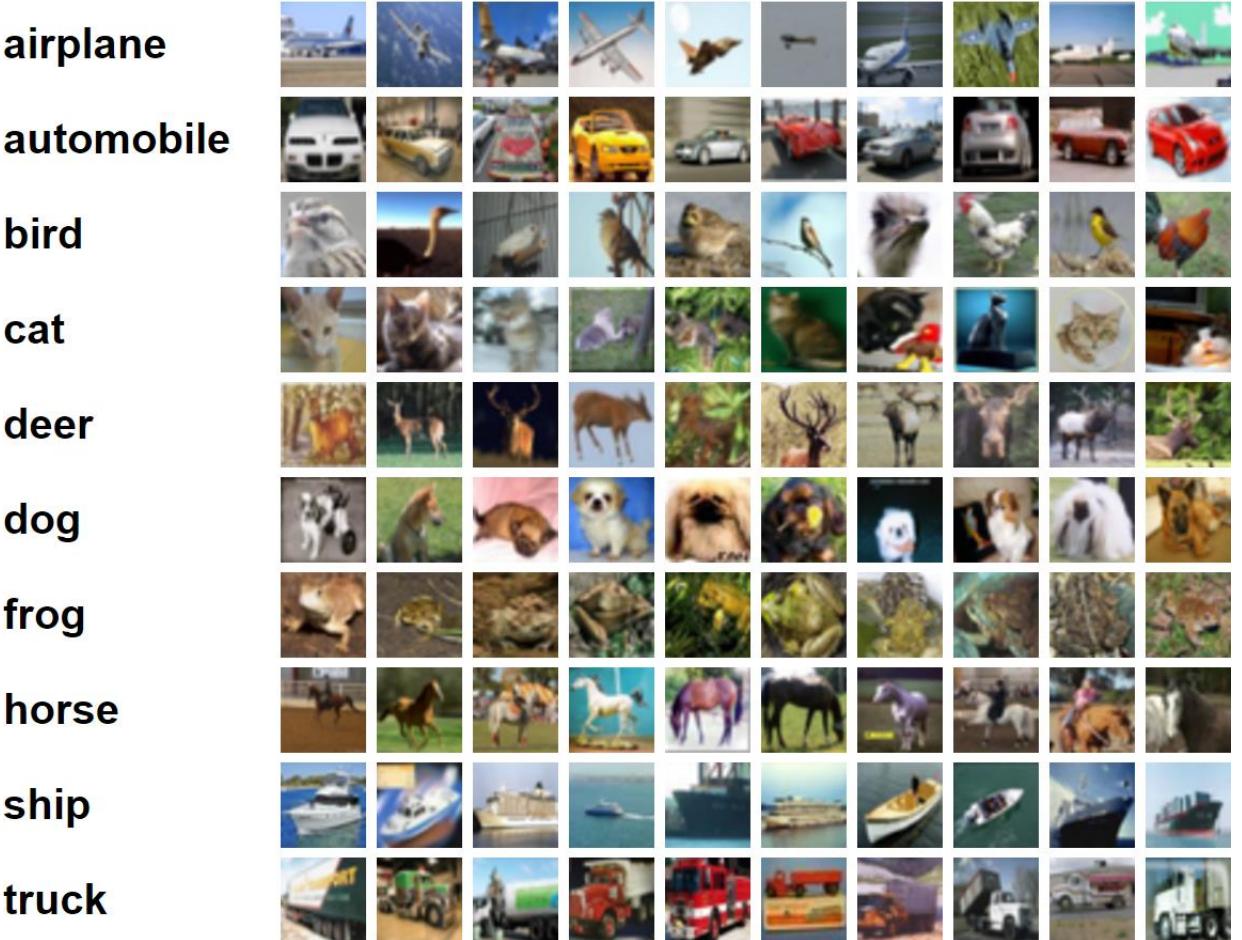
CIFAR-10 dataset

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:

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



This resolution is by far smaller than what we are used to

$$d = 3072$$



Former standard repository for ML research



$d = 3072$

Attributes

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

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

$d = 3072$

Former standard repository for ML research

Bear in mind how large an image is (in terms of Bytes) when you'll be implementing your CNN... the whole batch and the corresponding activations have to be stored in memory!

2K attributes

32K attributes

Second challenge: label ambiguity

A label might not uniquely identify the image

Second challenge: label ambiguity

Man?

Beer?

Dinner?

Restaurant?

Sausages?

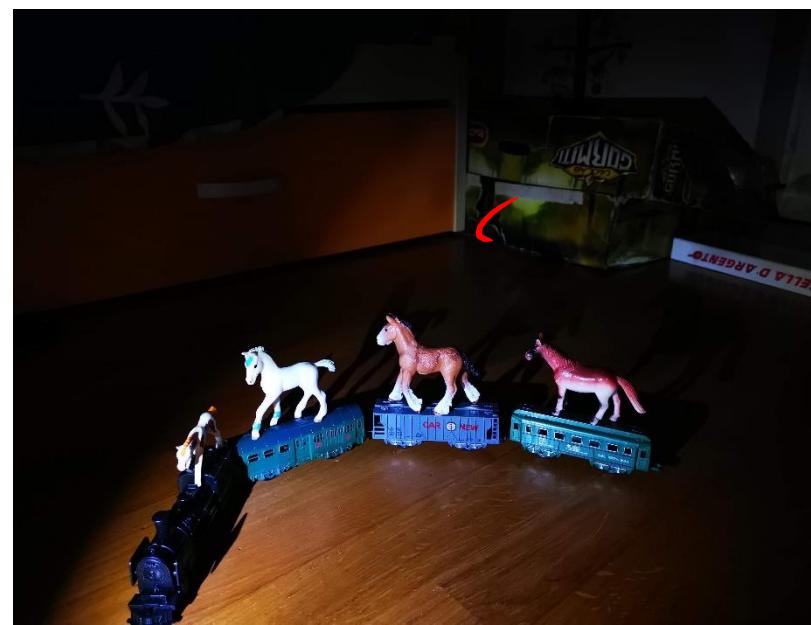
....



Third challenge: transformations

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

Changes in the Illumination Conditions



Deformations



Copyright Christine Matthews



© Copyright Patrick Roper

View Point Change



... and many others

Occlusion



Background clutter



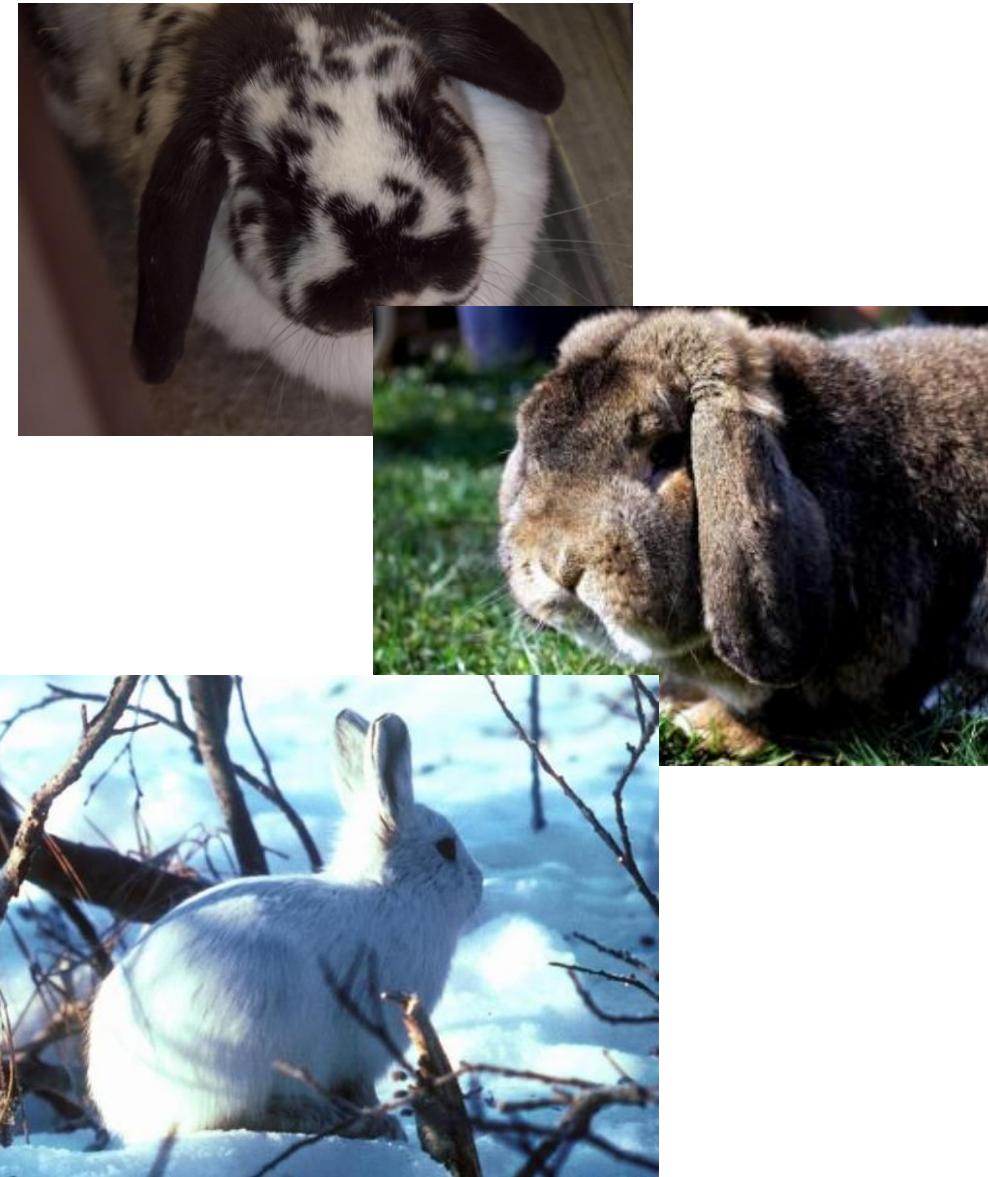
Scale variation



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

$$\hat{y}_j = y_{j^*}, \quad \text{being } j^* = \operatorname{argmin}_{i=1 \dots N} d(\mathbf{x}_j, \mathbf{x}_i)$$

Distances are typically measured as

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

Or

$$d(\mathbf{x}_j, \mathbf{x}_i) = |\mathbf{x}_j - \mathbf{x}_i| = \sum_k |[\mathbf{x}_j]_k - [\mathbf{x}_i]_k|$$

Pixel-wise distance among images

test image				training image				pixel-wise absolute value differences					
56	32	10	18	-	10	20	24	17	=	46	12	14	1
90	23	128	133		8	10	89	100		82	13	39	33
24	26	178	200		12	16	178	170		12	10	0	30
2	0	255	220		4	32	233	112		2	32	22	108

→ 456

K-Nearest Neighborhood Classifiers for Images

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

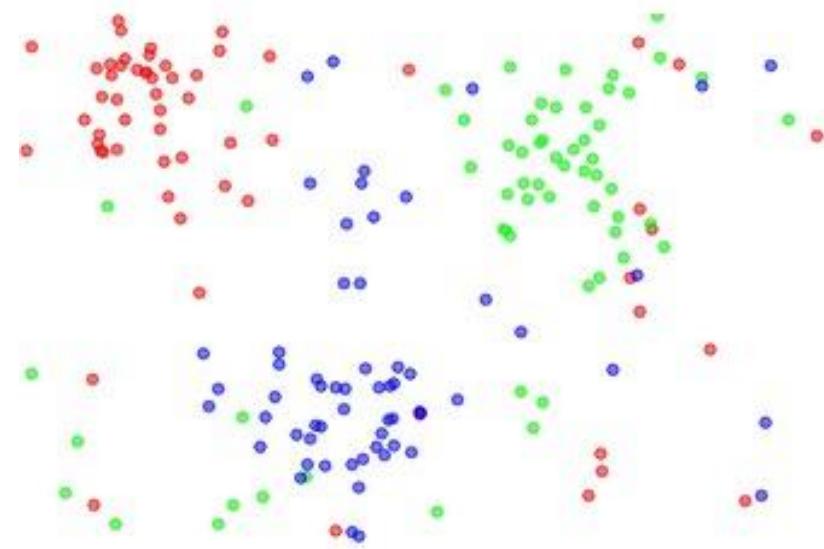
$$\hat{y}_j = y_{j^*}, \quad \text{being } j^* \text{ the mode of } \mathcal{U}_K(x_j)$$

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

Setting the parameter K and the distance measure is an issue

Nearest Neighborhood Classifier (k -NN) for Images

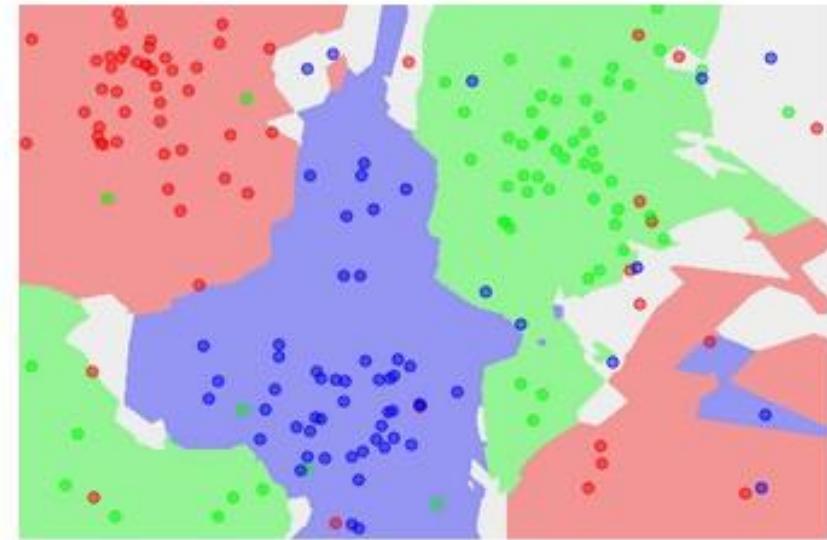
the data



1-NN classifier

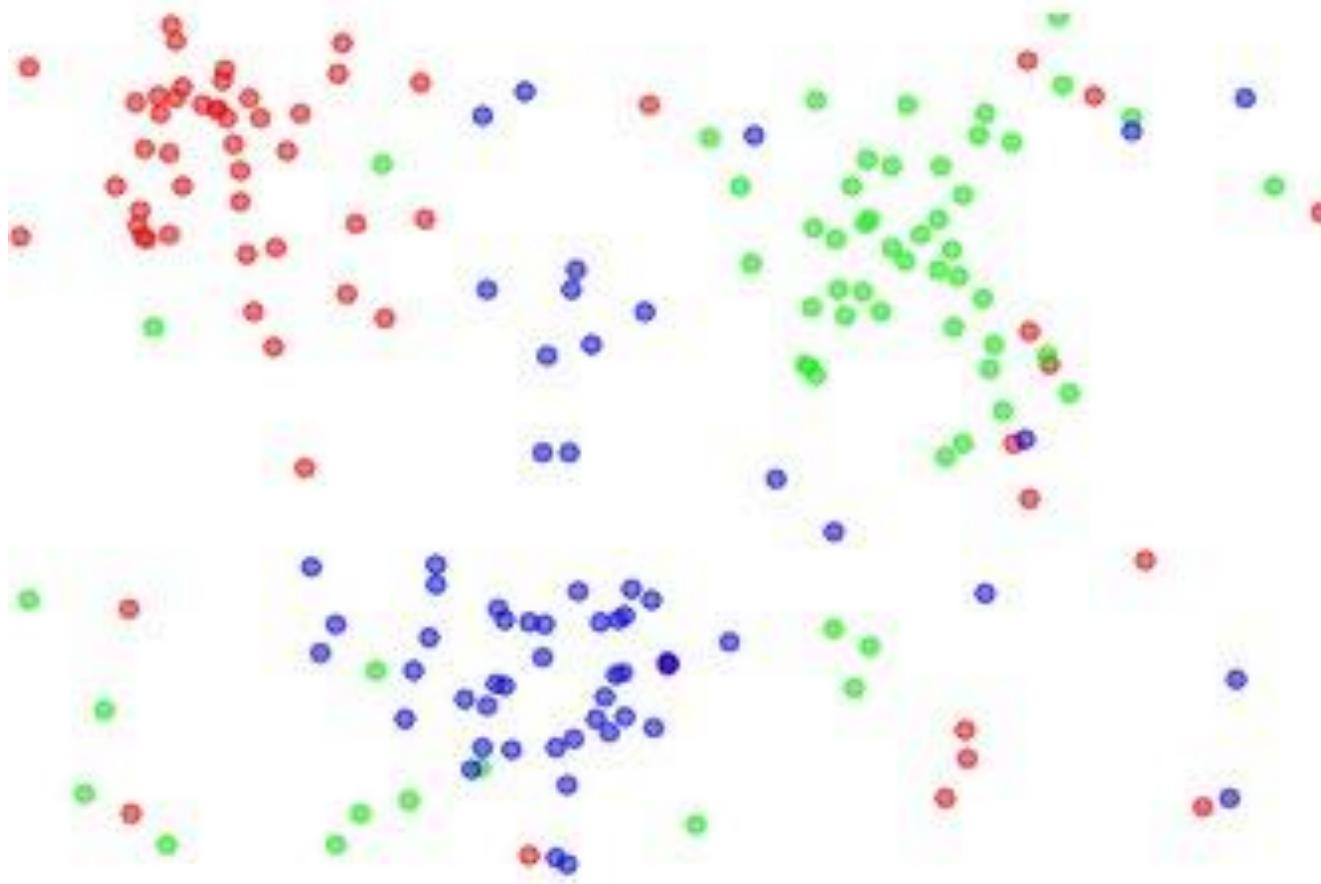


5-NN classifier



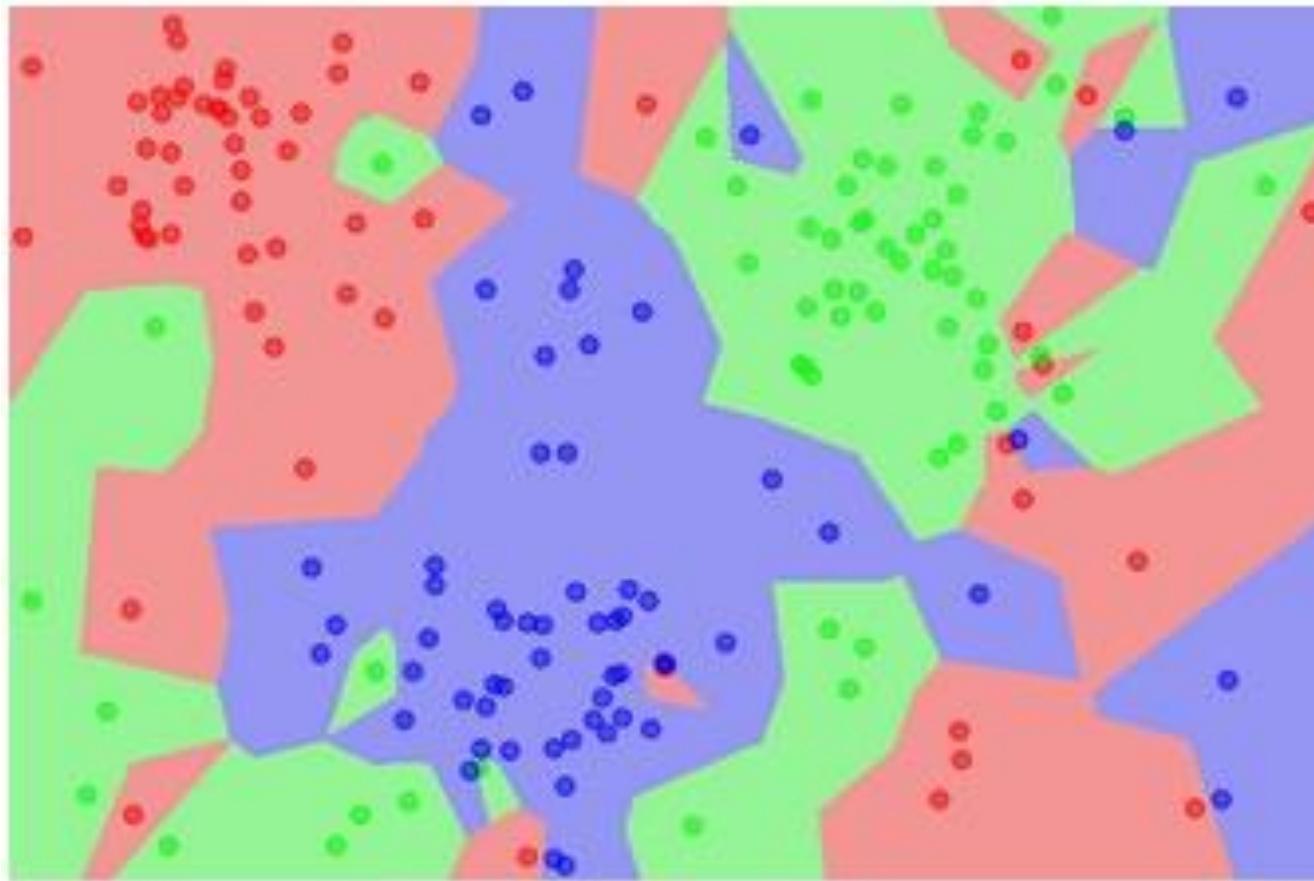
k -NN for Images

the data



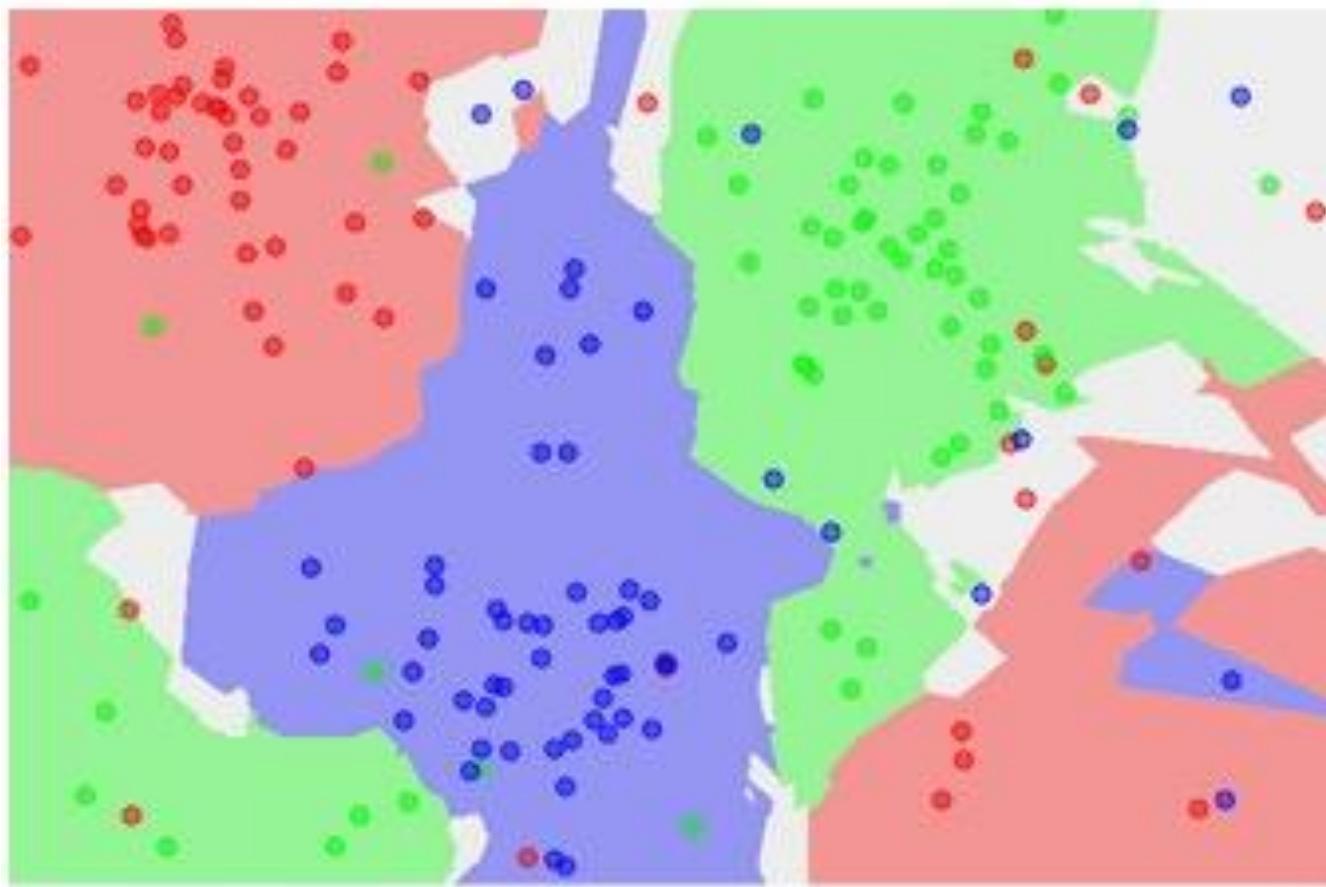
k -NN for Images

NN classifier



k -NN for Images

5-NN classifier



k -NN for Images

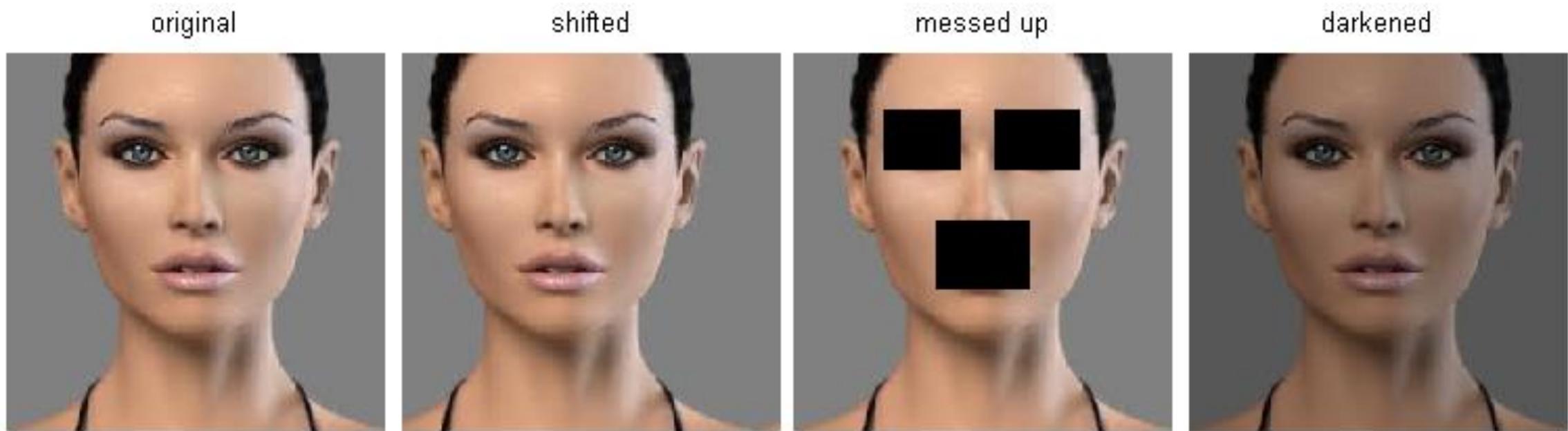
Pros:

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

Cons:

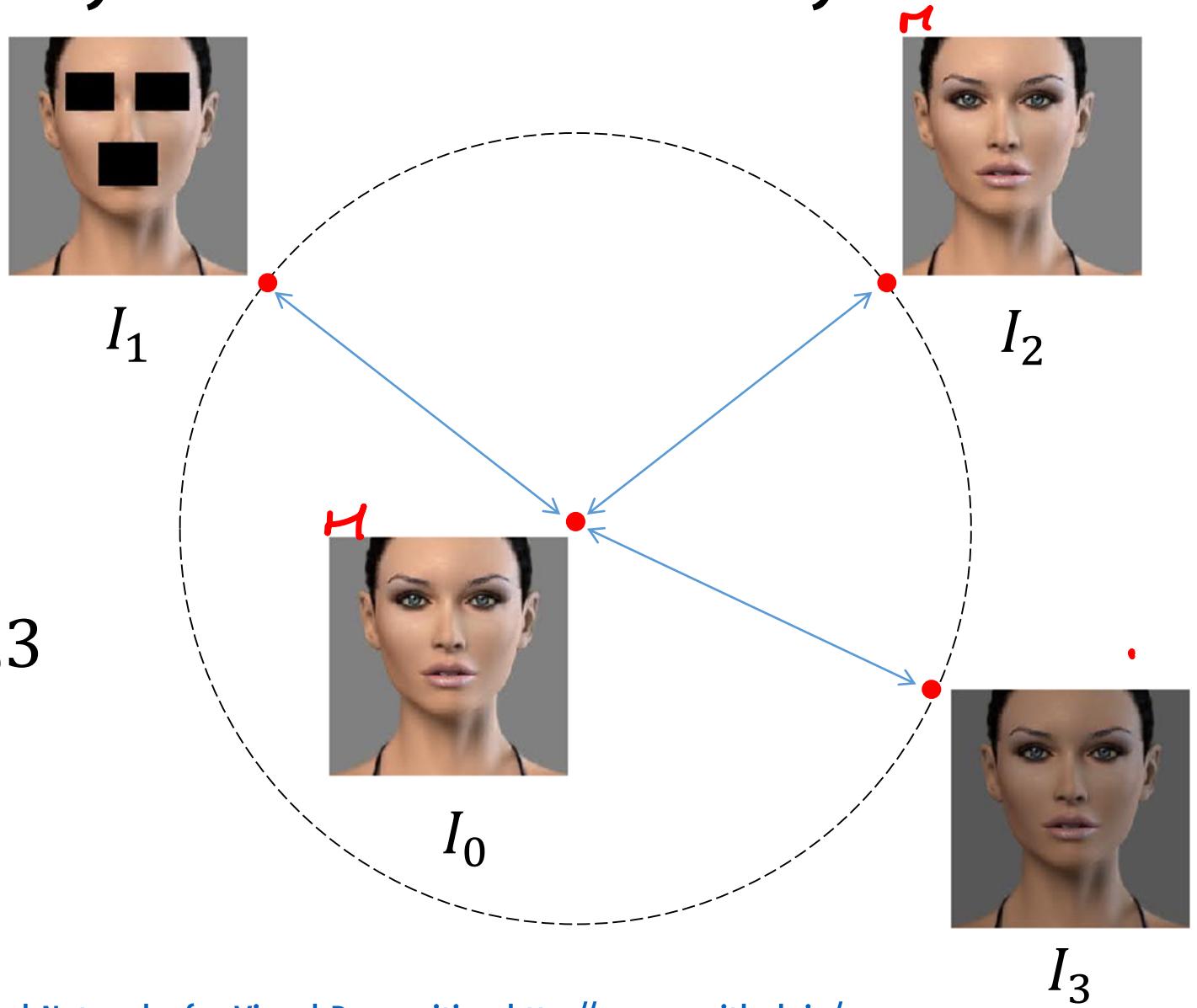
- Computationally demanding at test time (in particular when TR is large and d is also large)
- Large training sets have to be stored in memory
- Rarely practical on images: distances on high-dimensional objects are difficult to interpret

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



Let's see what happens on the whole CIFAR10



On CIFAR10 we see exactly this problem



On CIFAR10 we see exactly this problem



On CIFAR10 we see exactly this problem

