LESSON 9

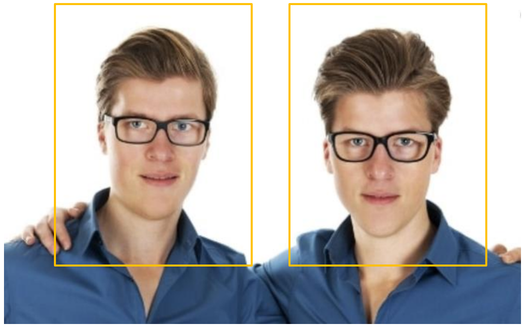
Labelling Errors,
Similarity in Datasets and Images




Outline

- **Labelling errors**
  - **Supervisor errors**
  - **Changes in time**
  - **Checks**
- **Similarity**
  - **in datasets**
  - **in images**
- **Main points**

y = «cat»

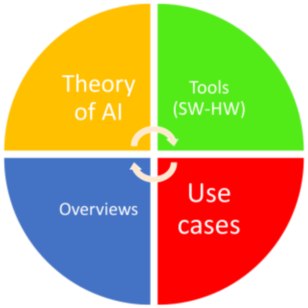




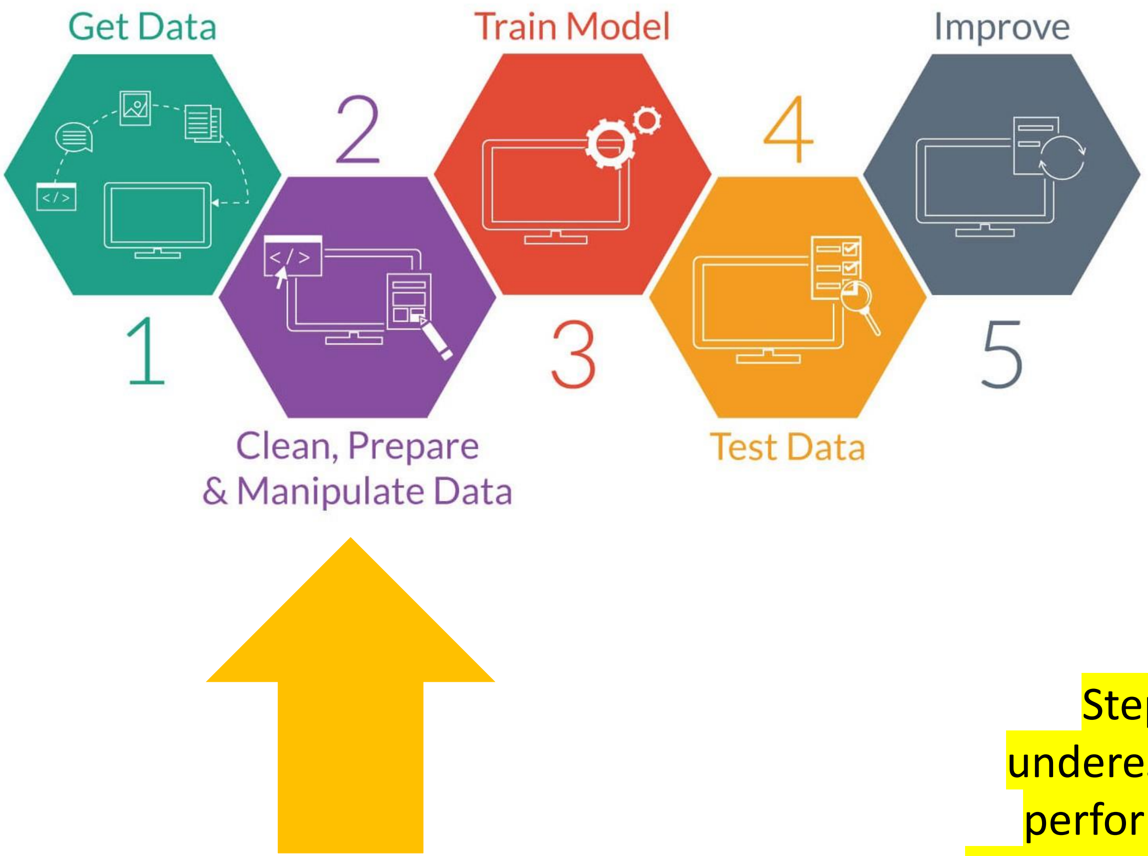

THEORY

Supervisor errors and Labelling errors

Typical errors and their consequences



Step 2 of the ML workflow



Step2 is too often underestimated. If poorly performed, it generates relevant loss of performances

Label Errors in Datasets

Human supervisors are the typical sources of labels for our datasets.

- Direct supervisor errors

y = «cat»

- Example: Medical Diagnosis
  - Inputs = XRAY, blood exams
  - Output = diagnosis

y = «healthy»

Label Errors in Datasets (2)

- A Diagnosis can change in time → labels are not updated

This will limit the capability of the AI model to perform **early detection** of the problem

x

y = «b. penumonia»

Second Xray image with a better machine

y = «covid-19»

Label Errors in Datasets

- Data interchange/automatic conversion errors

```
Patient XML data structure
<doctor authorized code = %code>
<data = %data>
<robot number = %robot>
<person = %name>
        <General>
                <hemoglobin = %>
                <oxygen = %>
                <temperature = %>
                <pressure = %>
                <hear rate = %>
        </General>
        <Bio-sensor>
                <sensor 1> data </sensor1>
                <sensor2> data </sensor2>
                <sensor3> data </sensor3>
        </Bio-sensor>
</person>
<real-time data>
<sensor no = %no>
        Data_stream….
</sensor>
</realtime data>
```

Automatic format/label conversion

y = «fat»

Jury Chechi @2019
(Olimpic gold medal in Rings 1996)

Basic checks for labelling errors

- Same input vectors (duplications)
  - With same labels
    - no training problems, but waste of storage, memory, processing time
  - With opposite labels (training problems!)

x1

y1 = «cat»

x2

y2 = «dog»

Basic checks for labelling errors (2)

- What to check in the training and validation database even it is not easy with large DB (Dbsize>10e6 images)
- A simple one to N comparison → iter.=N*(N-1)

x_1

y_1 = «dog»

…

x_i

y_i = «cat»

x_j

y_j = «dog»

…

x_N

y_N = «dog»

Basic checks for labelling errors (3)

- Example: the *Serengeti Dataset* (Public)
  - Unlabeled: 3.2 million images corresponding to 1.2 million capture events (seq. of images; tot 7.1M images)
    - Unsupervised is not bad→ Good for tuning, feature extraction, further study
  - **Labelled DB Test set:** volunteer-labeled test set of 17400 capture events.
- A basic check→ iter= N*(N-1)=**285M comparisons**



A very simple procedure

(to start the analysis)

STEP 1) Load x_i and x_j

STEP 2) Check for duplication

STEP 3) Labels are different?

YES // bad case, choose your option

→ OPTION1: Remove the sample

→ OPTION2: Ask for assistance with a supervisor (second reading)

→ Change the label and merge the sample or reject the sample

NO // good

→ reject one sample

→ manage indexes i,j

→ return to STEP 1)

Which option?
- Extralarge dataset?
- Expensive data?
- ....

Basic checks for duplications: **hash functions**

If you are dealing with large data/vectors a simple comparison

if(x_i == x_j)

requires element2element or pixel2pixel comparison → time consuming
(even if perfectly parallelizable → good for CUDAs)

→Using a **standard** file hash (if already available)
- →Classical MD5 and SHA-1 algorithms

→Create **image-hash** information offline
- →Specific hash functions for images are available

Basic checks for duplications/similarity (3)

Even if you are dealing with images or vectors (or unstructured data) is **NOT** just about

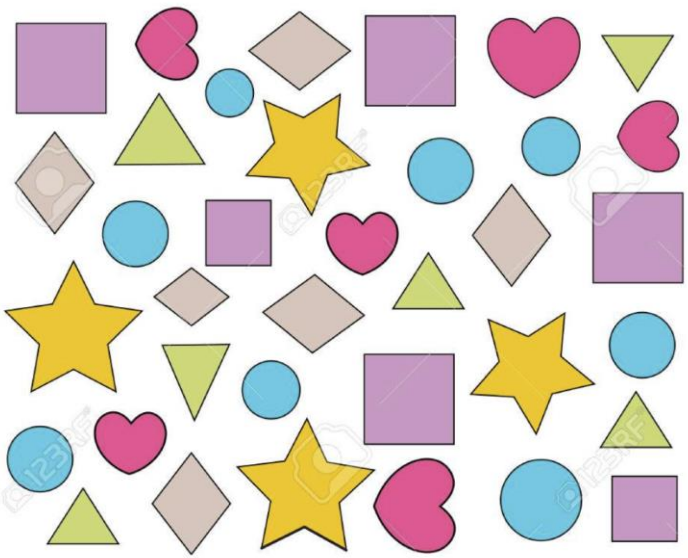
if(x_i == x_j)

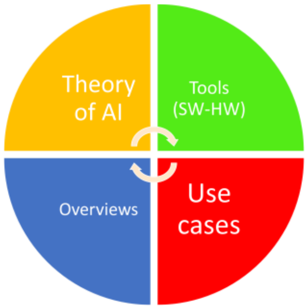
but something like

if (**similarity**(x_i , x_j) > fixed_threshold)

THEORY
Similarity in datasets

What is relevant what is useless?



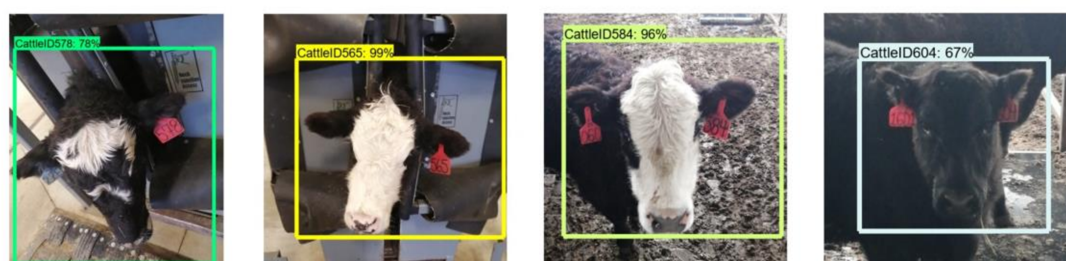


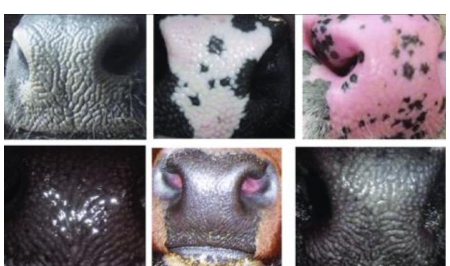
Tuning the similarity metrics: An example

Bottom line:
the similarity metric to be chosen depends on the type of application

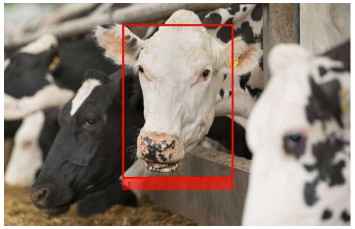
- Caw face recognition
  - Startups are using facial recognition software to increase the productivity of dairy cows.
  - Tracking activity
  - Automatic Food delivery
  - Drug delivery
- Main modules
  - 1 NN to segment the face
  - 1 NN to identify the caw

Features: body spots



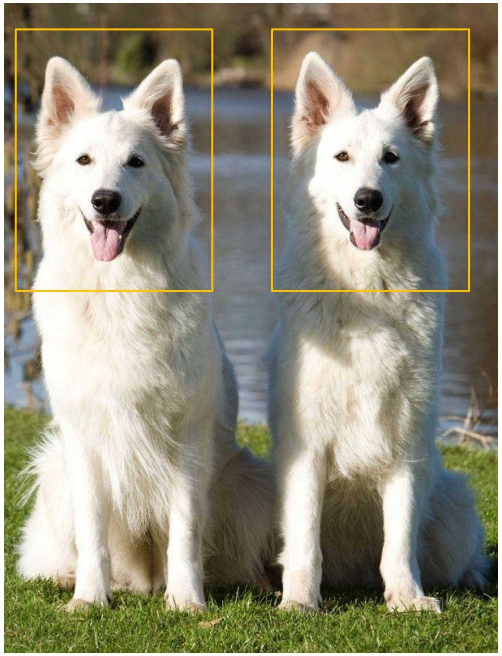


Features:
muzzle spots



Why performing checks for similarity

- Why that?
  - Too similar data/images are providing little more information
  - Make the dataset more complex to be handled
    - Storing data
    - Loading data
    - Training models,
    - Etc.
- The similarity metrics must be tuned according to your application



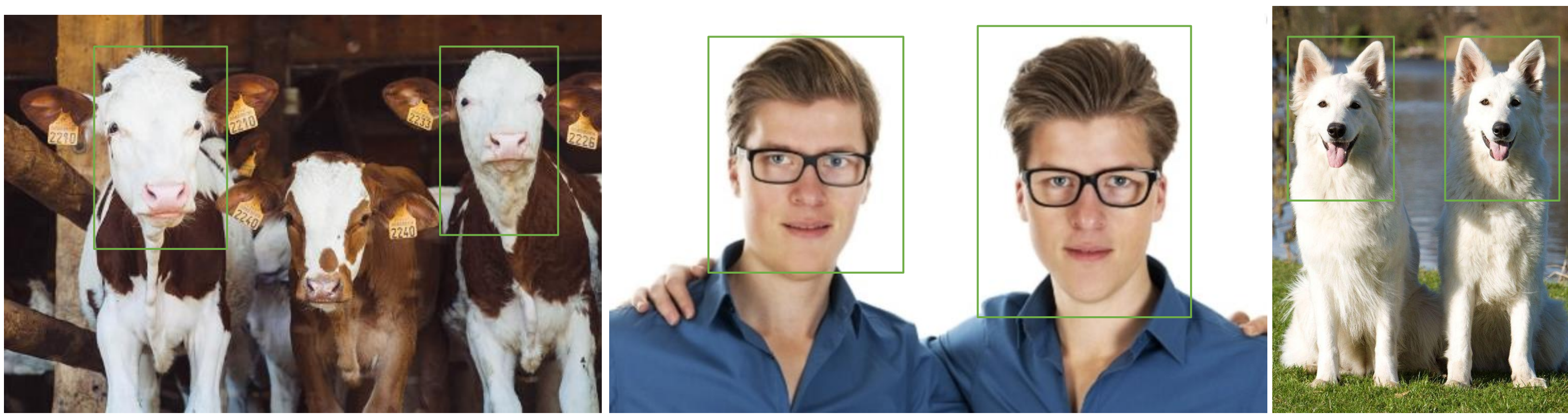
Two very similar dogs. Too similar to teach the models to identify the breed of the dog, but necessary to identify the single dog

Tuning the Similarity Level Examples

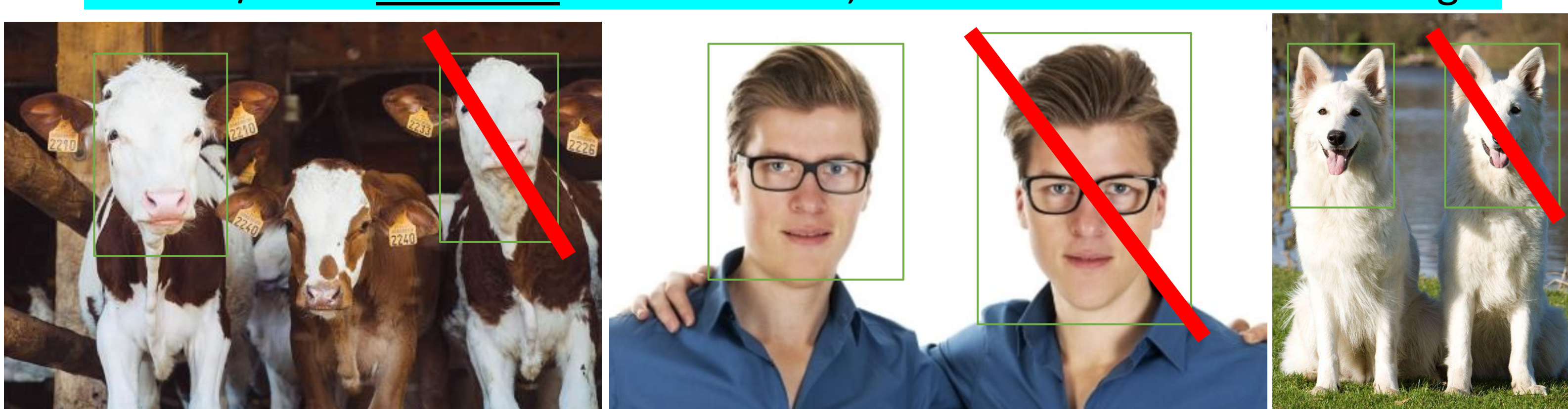
The similarity metrics must be tuned according to your application

… if (**similarity**(x_i , x_j) > fixed_threshold)

For identification task these are very good samples!



For face/snout detection are redundant, better to add more other images



Similarity and data augmentation

Too much **Similarity** → waste of space and time

Data Augmentation → improve generalization

D.A. will be discussed in the deeplearning section of the course

Model generalization in ML is how good the model is at learning from the given data and applying the learnt information elsewhere

y=«hotdog»

05nbZ1xxVNwUTcGwLbp7CN	01/24/18	178	NAV	Myself	205,082
07W9fL0ee9LsqD9c3kNrdo	01/28/18	23	Migos	Notice Me	606,513
07W9fL0ee9LsqD9c3kNrdo	01/27/18	16	Migos	Notice Me	763,370
07W9fL0ee9LsqD9c3kNrdo	01/26/18	17	Migos	Notice Me	940,976
09IStsImFySgyp0pIQdqAc	01/31/18	5	Zedd	The Middle	1,028,2
09IStsImFySgyp0pIQdqAc	01/30/18	6	Zedd	The Middle	1,009,0
09IStsImFySgyp0pIQdqAc	01/29/18	7	Zedd	The Middle	900,547
09IStsImFySgyp0pIQdqAc	01/28/18	26	Zedd	The Middle	592,893
09IStsImFySgyp0pIQdqAc	01/27/18	32	Zedd	The Middle	628,880
09IStsImFySgyp0pIQdqAc	01/26/18	36	Zedd	The Middle	678,529
09IStsImFySgyp0pIQdqAc	01/25/18	104	Zedd	The Middle	280,723
09ts3GnICqYEU5PkQCpJK3	01/31/18	32	Justin Timberlake	Say Something	606,396
09ts3GnICqYEU5PkQCpJK3	01/30/18	35	Justin Timberlake	Say Something	610,166
09ts3GnICqYEU5PkQCpJK3	01/29/18	33	Justin Timberlake	Say Something	587,693
09ts3GnICqYEU5PkQCpJK3	01/28/18	39	Justin Timberlake	Say Something	528,251
09ts3GnICqYEU5PkQCpJK3	01/27/18	30	Justin Timberlake	Say Something	639,293
09ts3GnICqYEU5PkQCpJK3	01/26/18	23	Justin Timberlake	Say Something	855,127

Similarity in datasets

- In the following we focus on images, but a similar approach can be used in general data

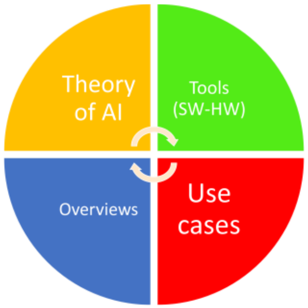
- Unstructured data
  - It’s better to extract features to use structured data techniques
- Structured data → features vectors → metrics
  - Euclidean norm or Manhattan Distance
  - Mahalanobis Distance
  - *Pearson Correlation Coefficient*
  - Complexity, Coherence, Structure, Entropy
  - …



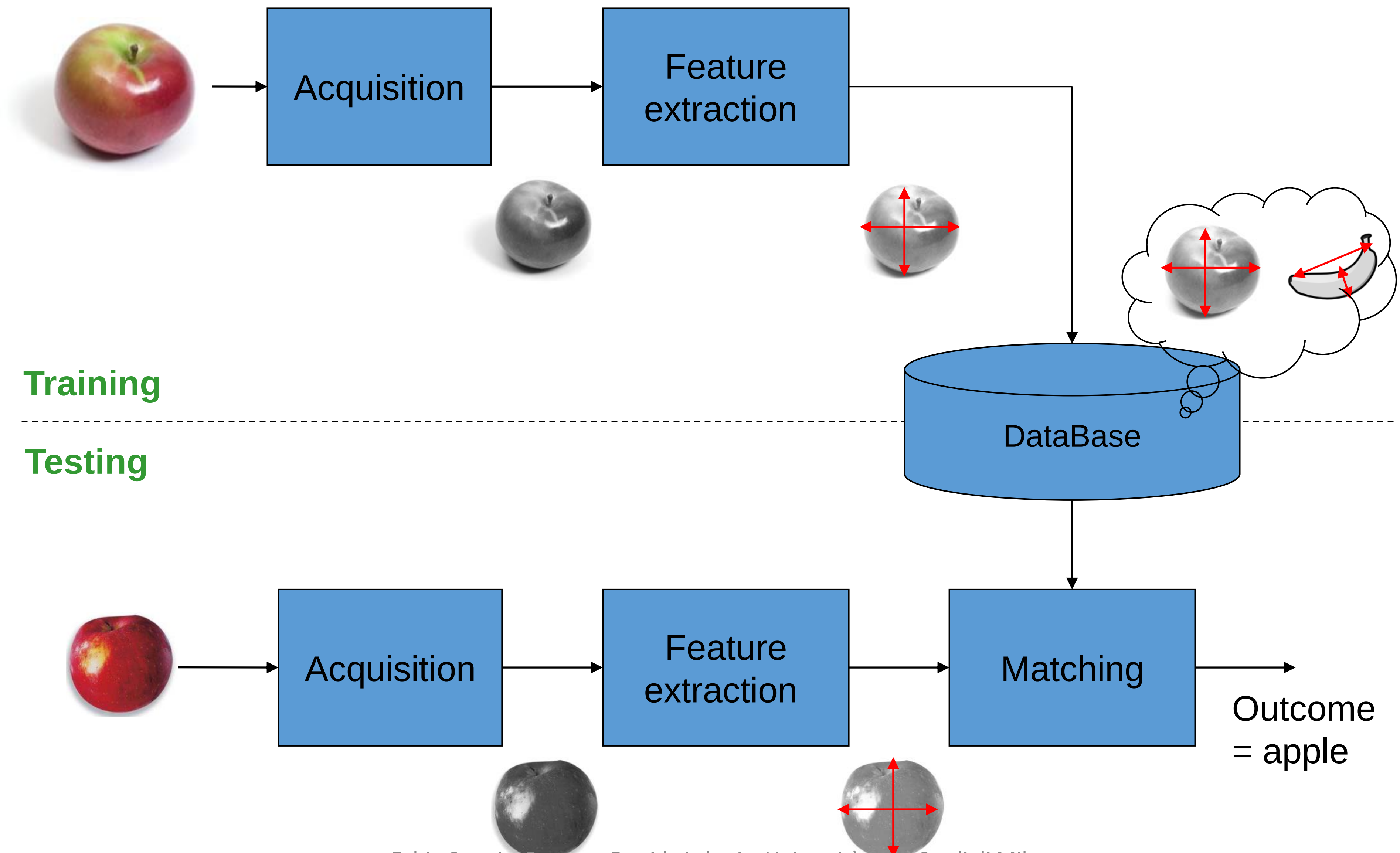

THEORY

Similarity & Pattern Rec.

The basis of most Machine learning methods

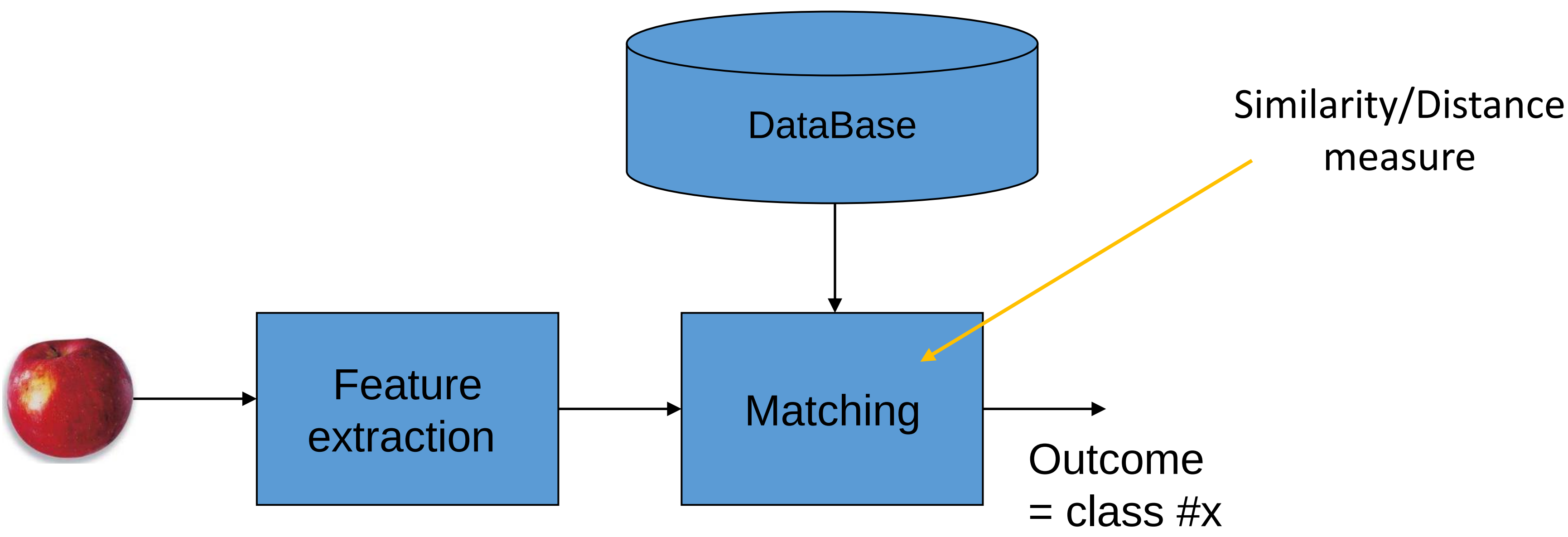


Pattern Recognition Systems



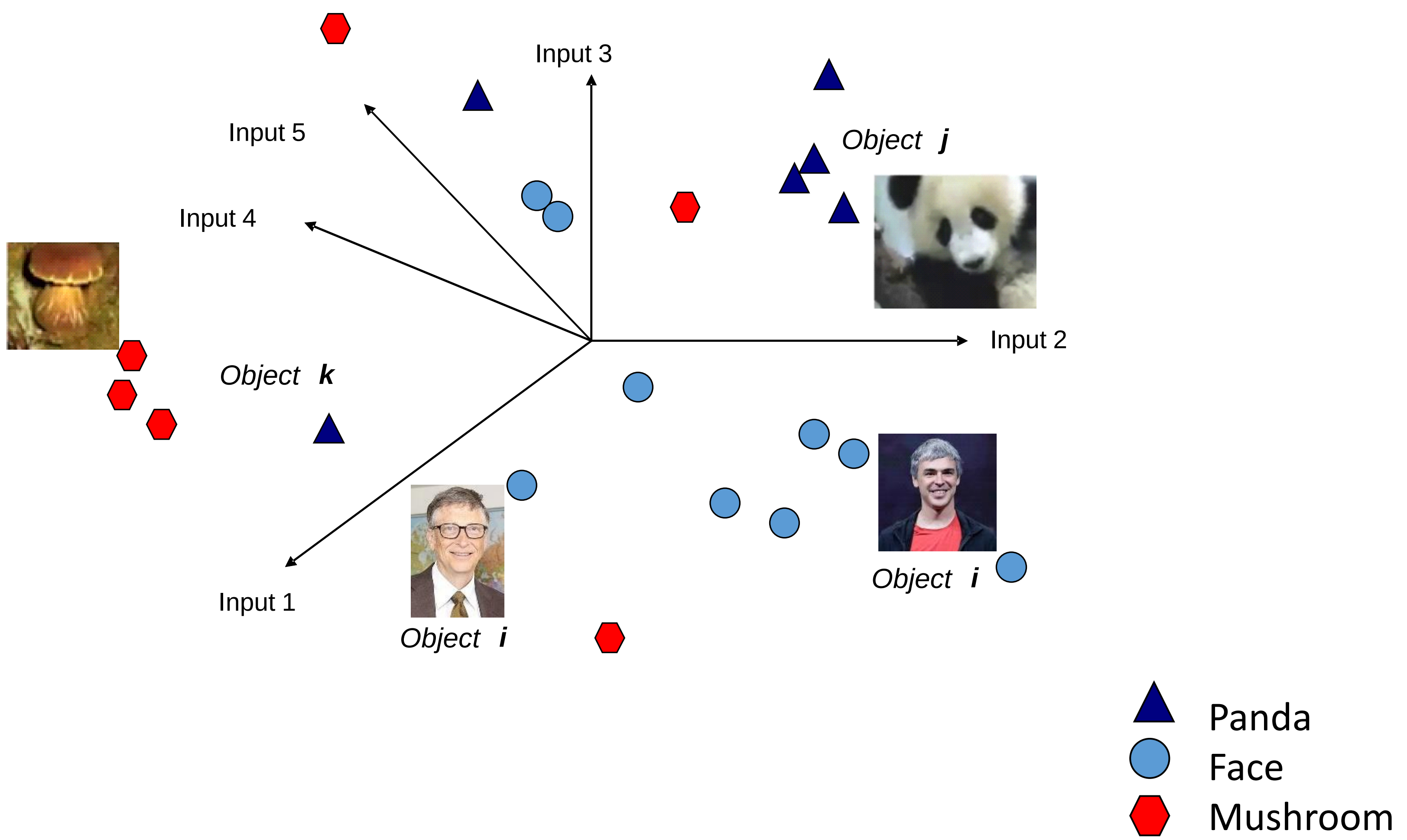
Similarity is the basis of pattern recognition

- Not just to clean our dataset…

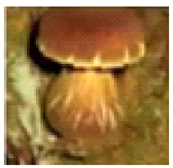


Remember the Input space!

Extracting features (a vector) from structured/unstructured data will allow you to think under a **common framework in ML**!



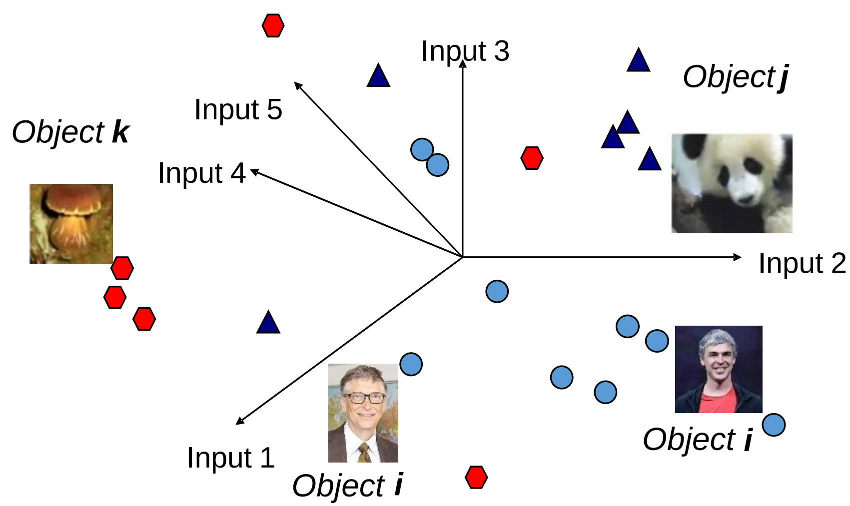
Input space → Feature space

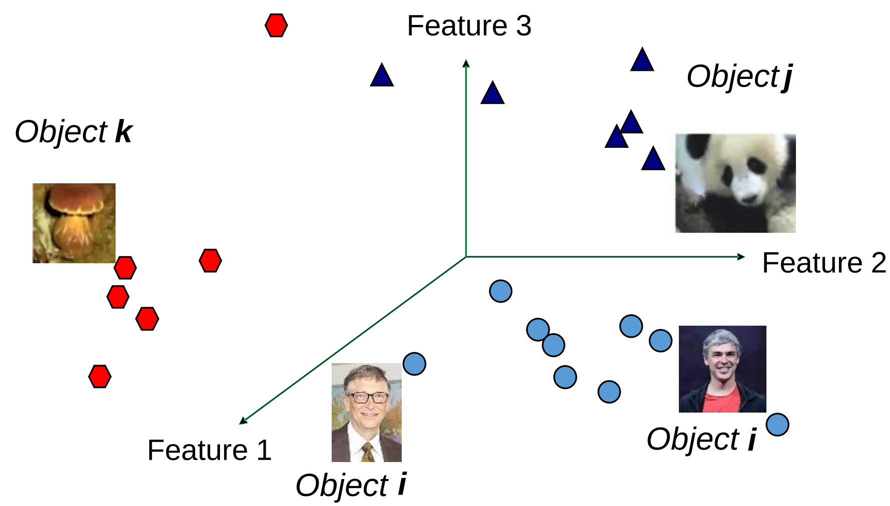


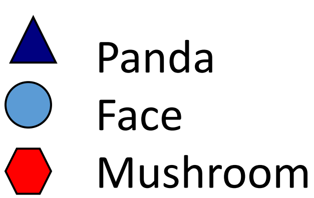
[input1
input2
…
inputN]

→ **Feature extraction** →

[feature1
feature2
…
featureM]

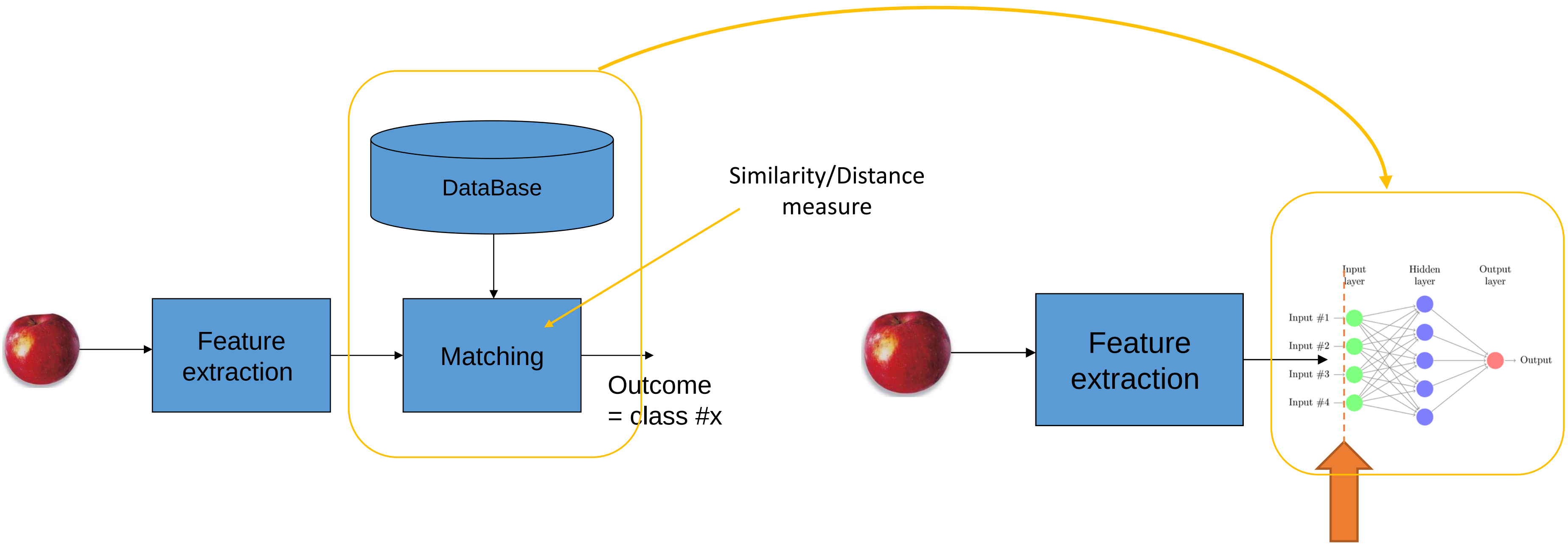






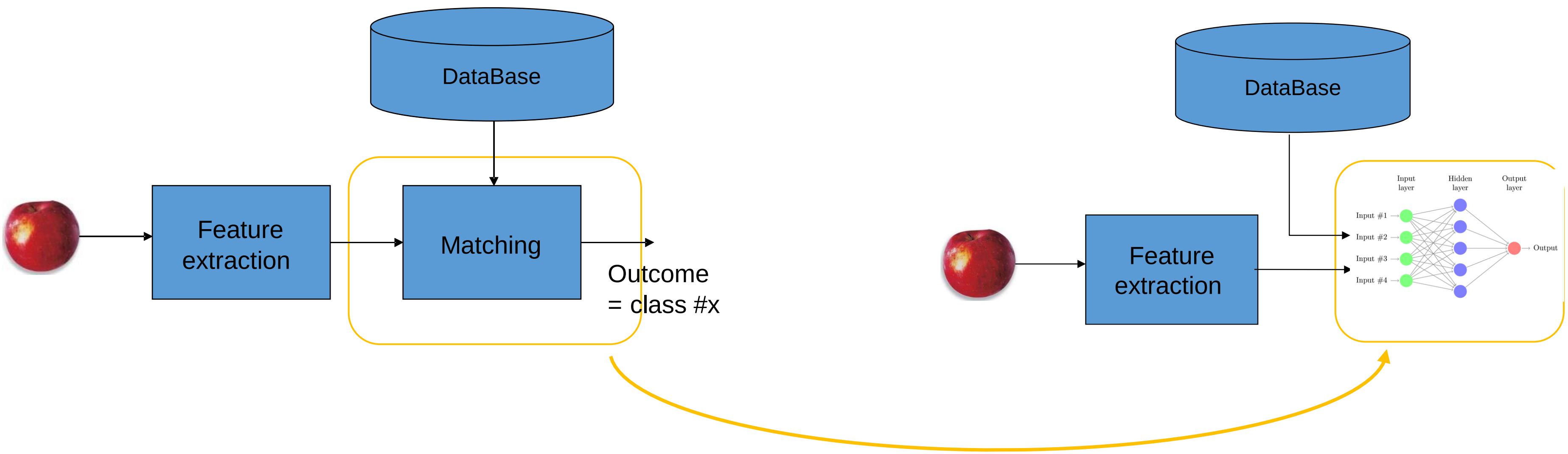
Feature Engineering:
A good design → less dimensions and better clusters

What is doing a NN during classification…



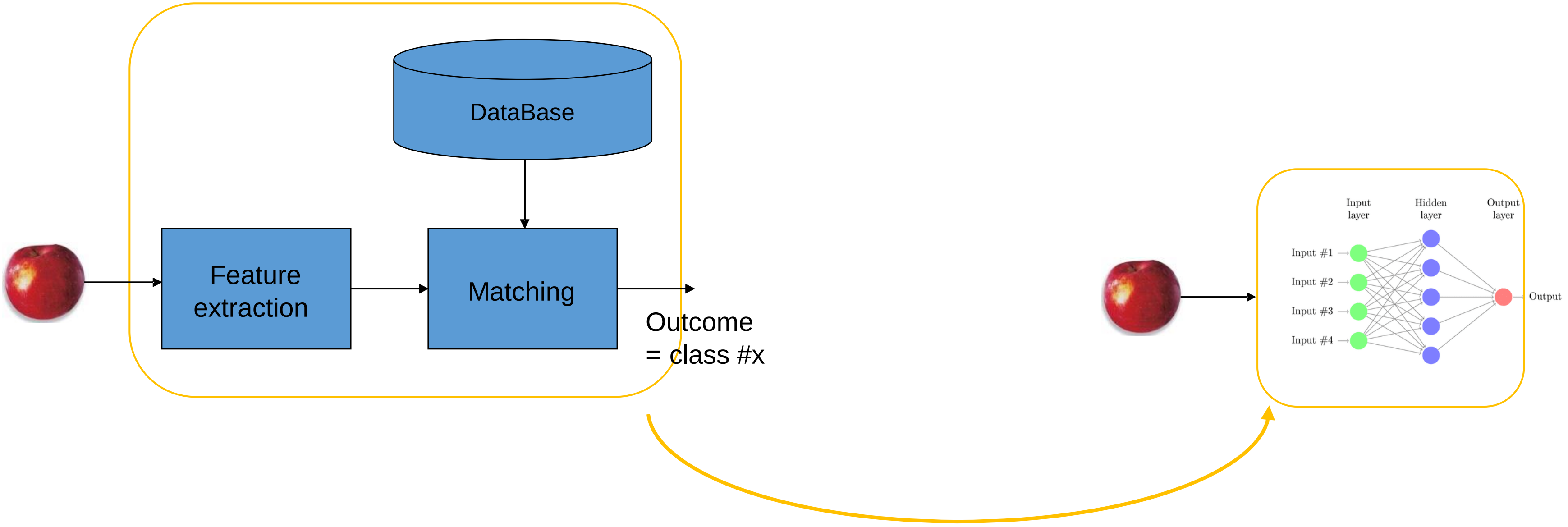
The «storing» and matching function will be mapped by the NN (complex!)

NN as … similarity function



Only the matching function will be mapped by the NN
(an easier task to solve)

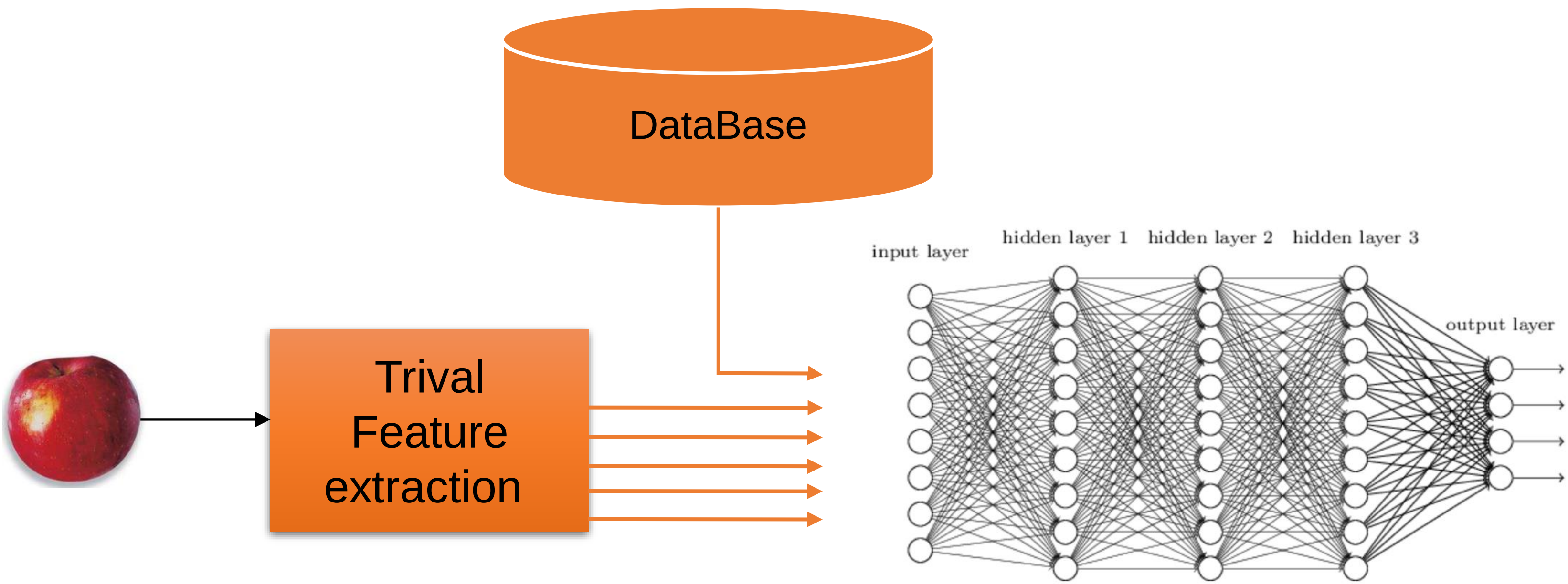
Deep NN ....



All steps are inside the deep learning model

Trivial feature extraction VS Feature Engineering

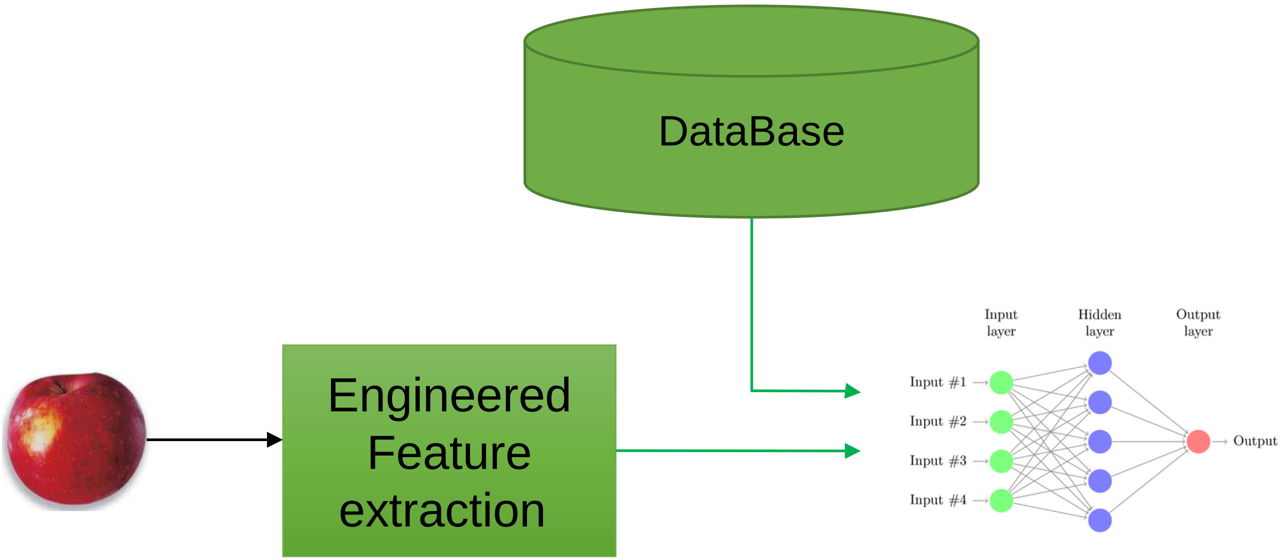
Note:
the DB must contain the same type of features of the feature extractor



Extracting a lot of noisy features….

Complex NN

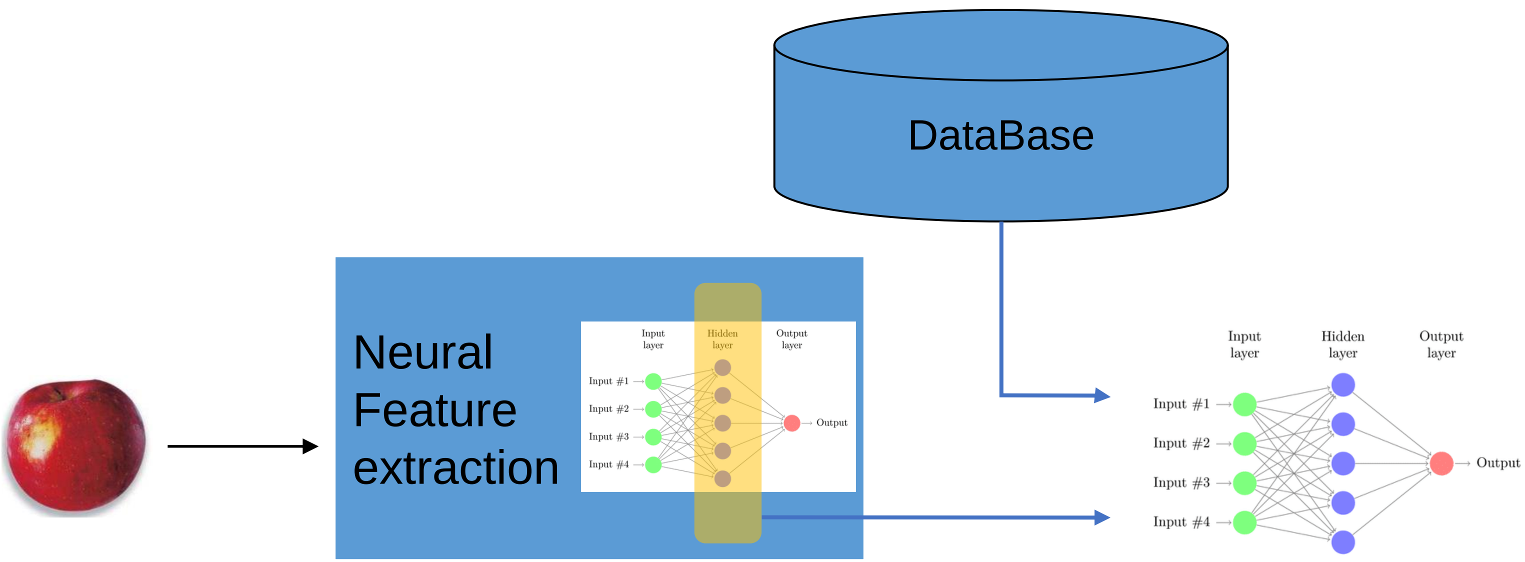
Trivial feature extraction VS Feature engineering



Only few of powerful features....

Simpler NN

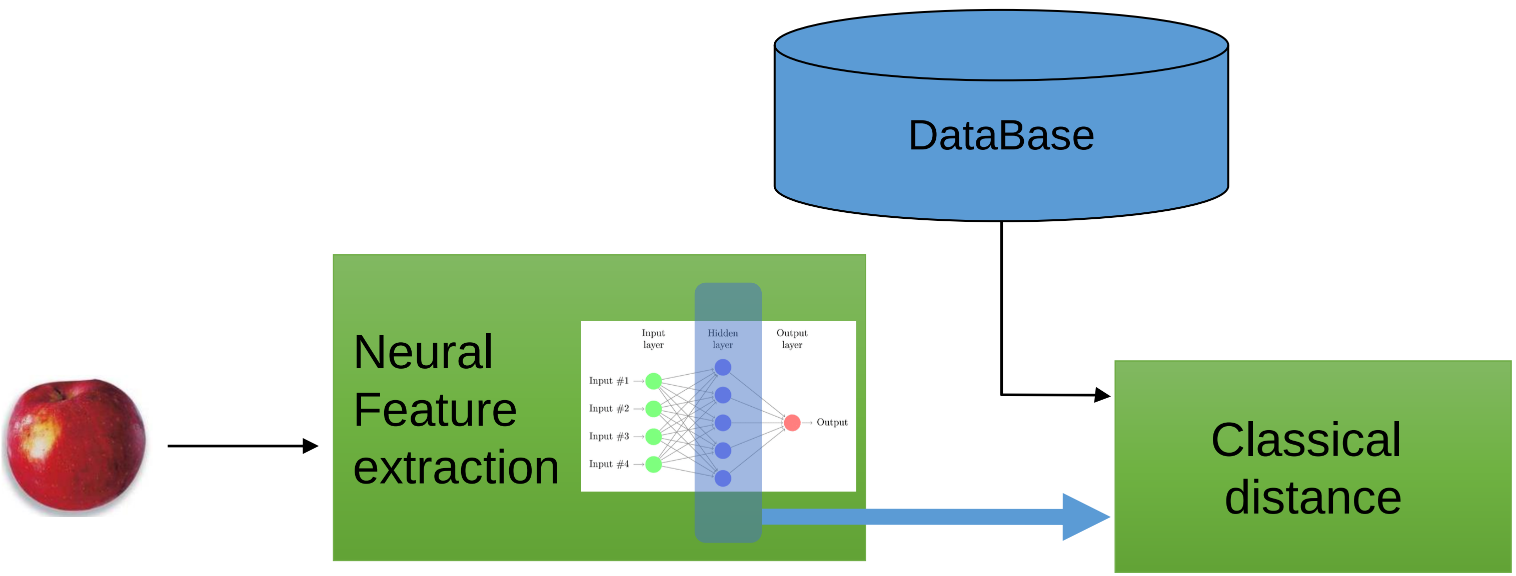
Neural feature extraction



Using NN to extract features
From the hidden layers

Simpler NN

Investing good computation in the first step → simple decisions



… and
improving
explainability

Using NN to extract features
From the hidden layers

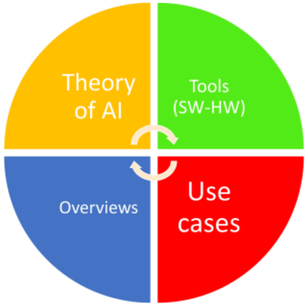
E.g., Euclidean distance

This topic will be discussed later in the course

THEORY

Basic Similarity and Distance in ML

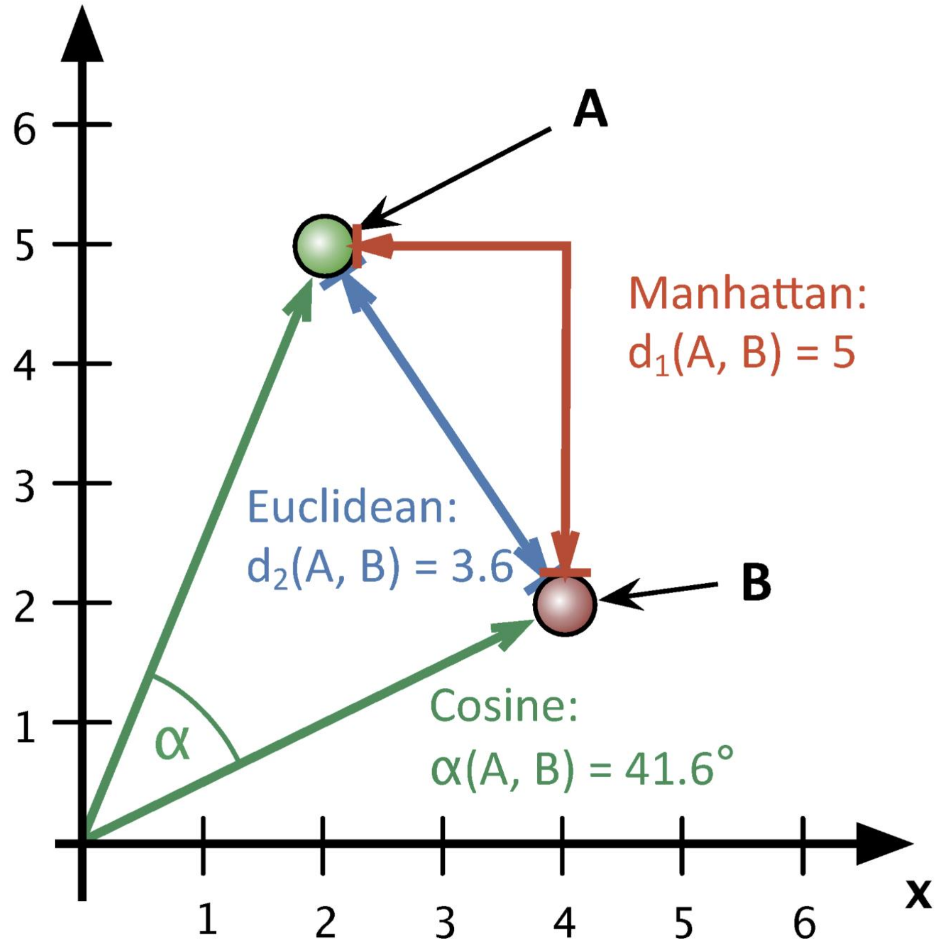
Comparing points in the feature space



Similarity ←→ Distance

- We think about **similarity**,
  but often the metrics is the **distance**
- In some sense is the inverse of distance metrics
- Main points of the transformation
  - Distance → zero so Similarity → inf.
    Distance → inf. so Similarity → 0
- The proper conversion depends on the applications and the mathematical properties you would like to have.
  - Example: Similarity = 1 / (1+distance)
    // 1+ …distance→0
- Often a real conversion is not necessary, just use the distance and find the proper thresholds

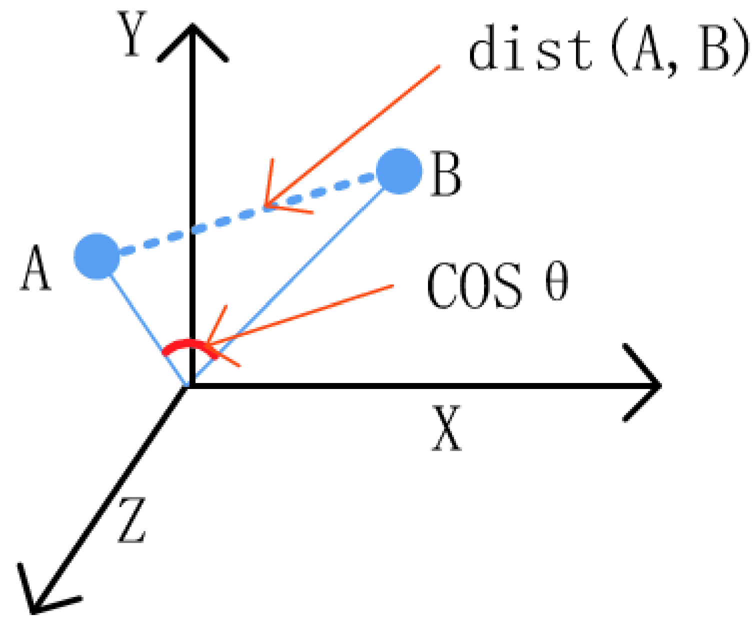
Basic metrics in data similarity→distance



- Euclidean
- Manhattan
- Cosine

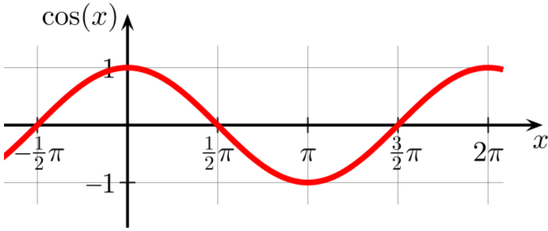
Cosine metrics

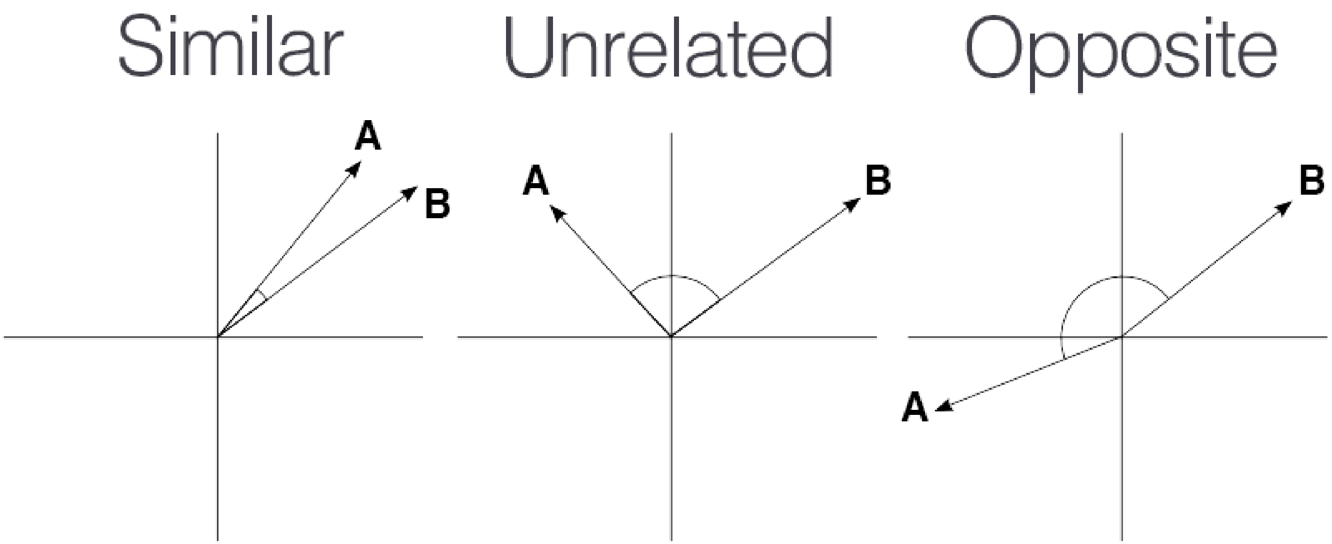
$$\mathbf{A} \cdot \mathbf{B} = \|\mathbf{A}\| \, \|\mathbf{B}\| \cos\theta$$



Cosine distance $= \cos(\theta) = \dfrac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \dfrac{\sum\limits_{i=1}^{n} A_i B_i}{\sqrt{\sum\limits_{i=1}^{n} A_i^2} \sqrt{\sum\limits_{i=1}^{n} B_i^2}}$

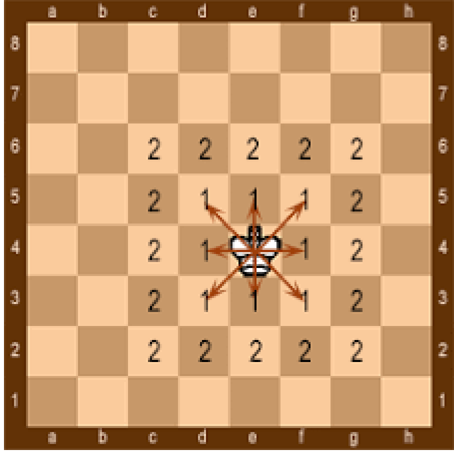
Cosine metrics idea…





Dist. → 1   Dist. → 0   Dist. → -1

Cosine distance $= \cos(\theta) = \dfrac{\mathbf{A}\cdot\mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \dfrac{\sum\limits_{i=1}^{n} A_i B_i}{\sqrt{\sum\limits_{i=1}^{n} A_i^2}\sqrt{\sum\limits_{i=1}^{n} B_i^2}}$



Chebyshev distance (chessboard distance)

- Gives the largest <u>magnitude</u> among each element of a vector

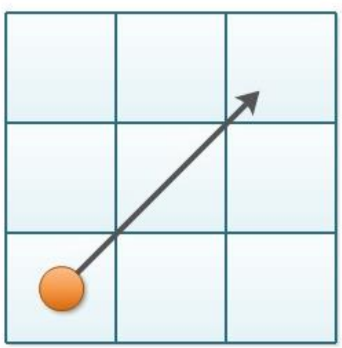
- Ex.: having the vector X= [-6, 4, 2], the L-infinity norm is 6 w.r.t. the origin

- In general given two vectors p and q

$$d(p,q) = \max_i \{|p_i - q_i|\}$$

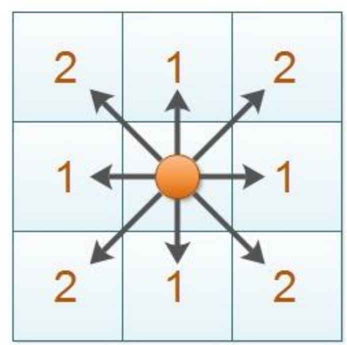
Comparison

Euclidean Distance



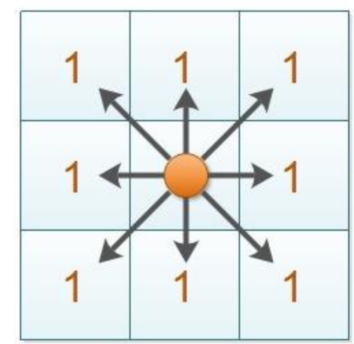
$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Manhattan Distance



$$|x_1 - x_2| + |y_1 - y_2|$$

Chebyshev Distance



$$\max(|x_1 - x_2|, |y_1 - y_2|)$$

Minkowski distance

$X = (x_1, x_2, \ldots, x_n)$ and $Y = (y_1, y_2, \ldots, y_n) \in \mathbb{R}^n$

$$D(X,Y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}}$$

This metric
is adimensional

- p = 1 is the Manhattan distance.
  Synonyms are L1-Norm, Taxicab or City-Block distance.

- p = 2 is the Euclidean distance.
  Synonyms are L2-Norm or Ruler distance.

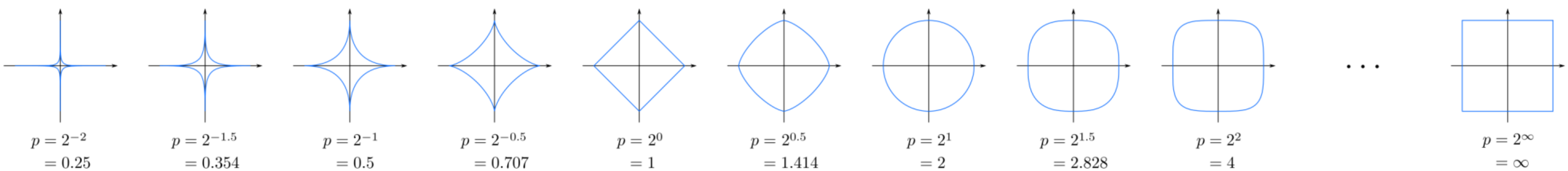
- p →∞ is the Chebyshev distance.
  Synonyms are L-infinity or Lmax-Norm or Chessboard distance

Minkowski distance

$$D(X,Y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}}$$

Example with two dimensions → N=2

The set of all points that are at the unit distance from the centre (unit circle) with various values of p

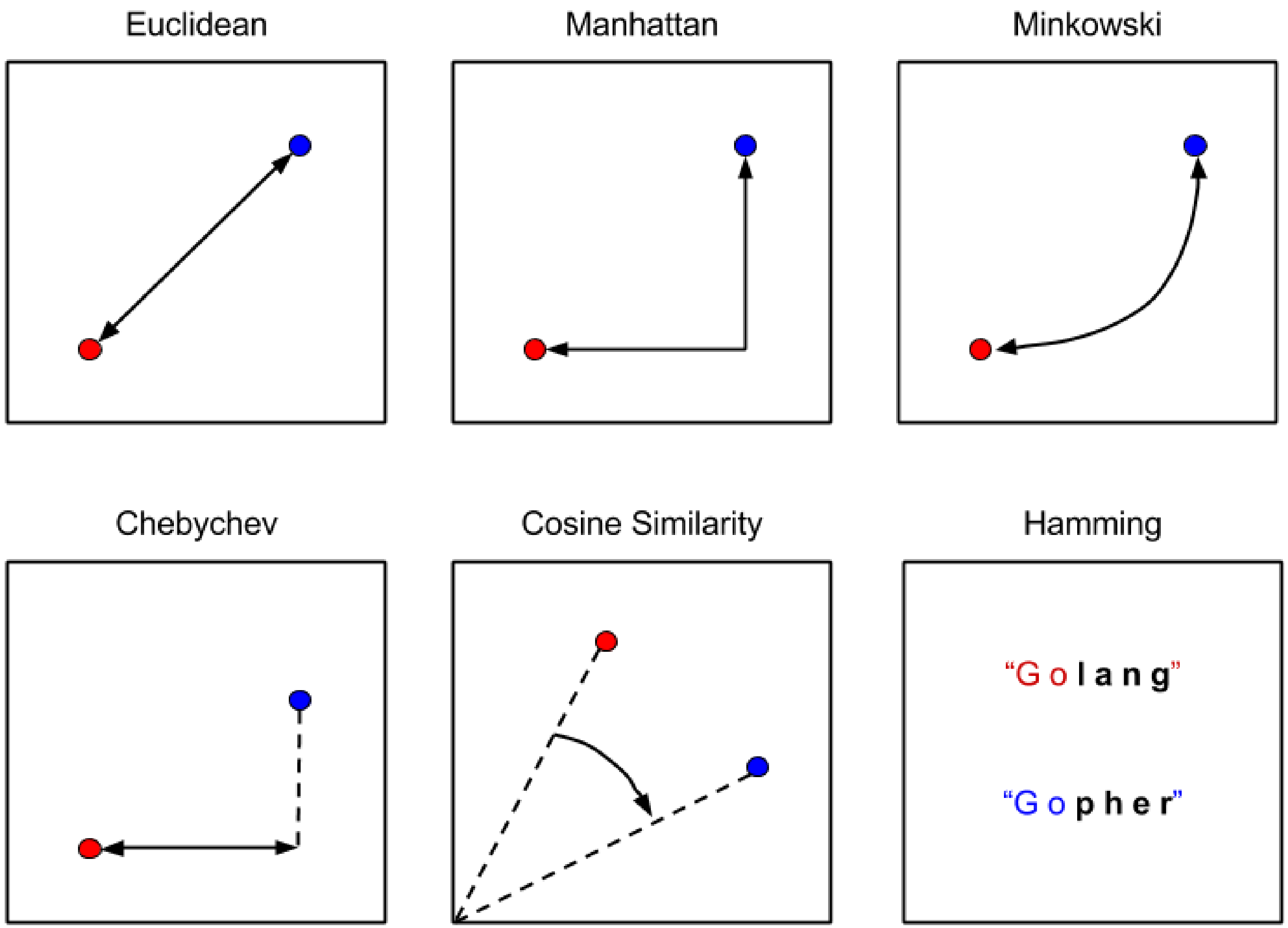


p = 1 is the Manhattan distance. Synonyms are L1-Norm, Taxicab or City-Block distance.

p = 2 is the Euclidean distance. Synonyms are L2-Norm or Ruler distance.

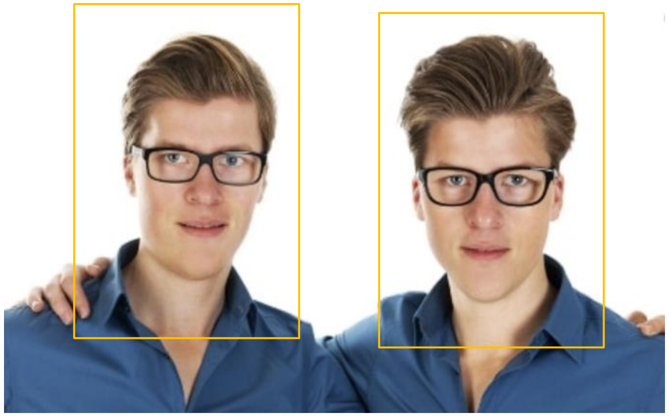
p →∞ is the Chebyshev distance. Synonyms are L-infinity or Lmax-Norm or Chessboard distance

Overview of basic metrics



THEORY
Similarity in images

What is relevant what is useless?



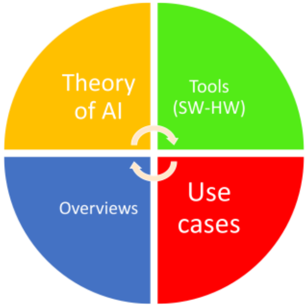


Image Similarity

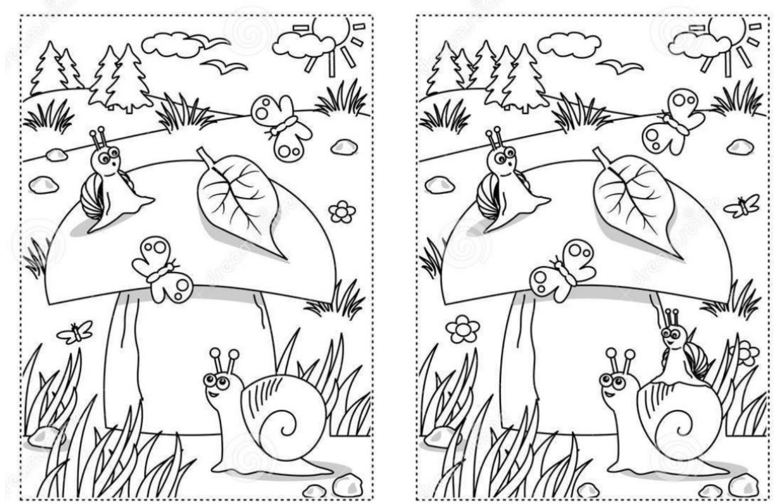


Image Similarity compares two images and returns a value that tells you how visually similar they are

```
score = similarity(image1, image2)
```

- If Score → 0 so images → Contextually similar
- If (score == 0) so images are identical

Image Similarity:
3 main usages scenarios

- **Clean out Datasets**
  - Messy datasets are critical for the ML designer, and cleaning out duplicates is a painful process
- **Image Similarity Search**
  - Have a picture of something and want to see if you have visually similar images? Using Similarity find contextually similar matches in your media library or user data.
- **Track Changes in Imagery**
  - Sometimes it can be hard to see changes in a project you're working on or monitoring.

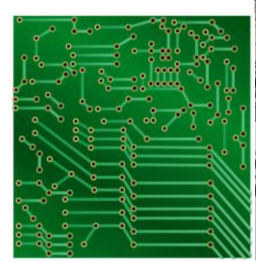
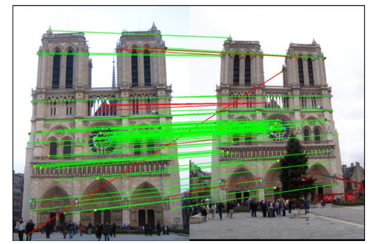
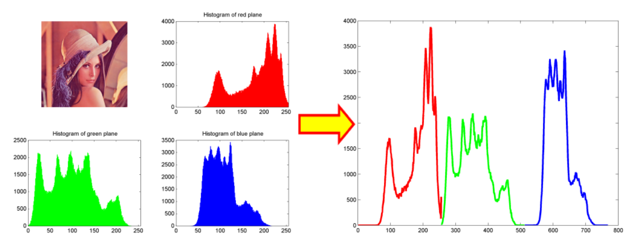


Image Similarity: 3 main approaches

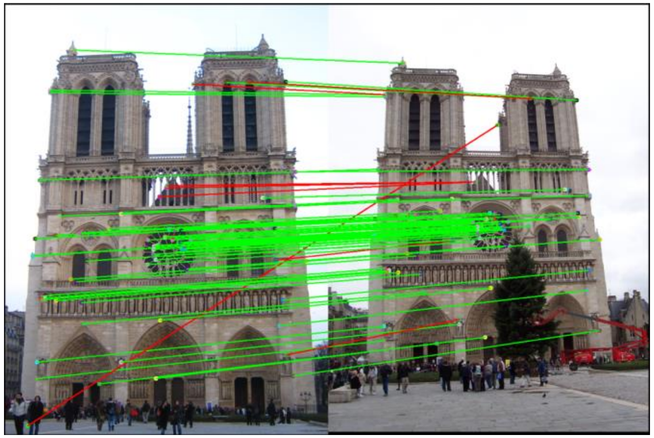
- **Keypoint matching**
  - SIFT
  - SURF
- **Histogram**
- **Image hash**
  - aHash
  - dHash
  - pHash





0001100001010000110 0...1010100110001000000000

Image Similarity: **Keypoint matching**



- Procedure
  1. Compute the abstraction of the image information and make a local decision at every image point to see if there is an image feature of the given type existing in that point.
  2. Compare the detected feature between two images.

PROS

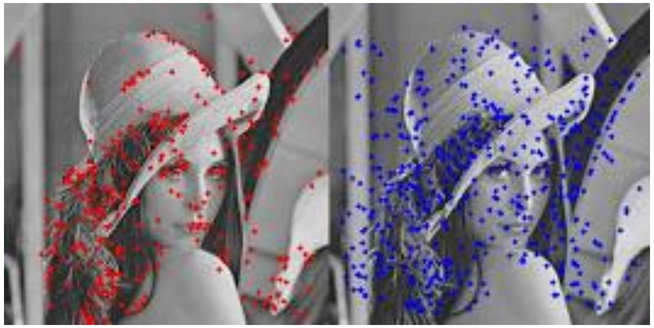
- These methods can match images under different scales, rotations, and lighting.

Very important!

CONS

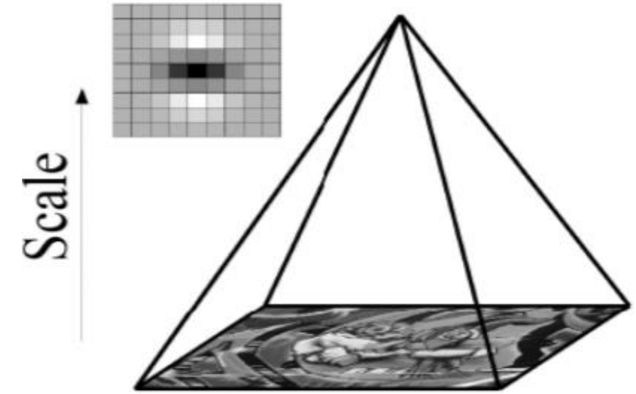
- running time of a naive implementation is $O(n^{2m})$,
  - n is the number of keypoints in each image,
  - m is the number of images in the database.

Keypoint matching: **SIFT and SURF**



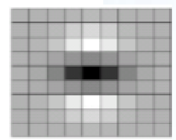
- SCALE INVARIANT FEATURE TRANSFORM (SIFT)
- SPEEDED UP ROBUST FEATURE (SURF)
- SIFT/SURF quite powerful (many pub. libraries…)
  - Image similarity, object recognition, image registration, classification, 3D reconstruction, …
- Discussion (in general not always):
  - **SURF** is better than **SIFT** in rotation invariant, blur and warp transform.
  - **SIFT** is better than **SURF** in different scale images.
  - **SURF** is faster (about 3x) than **SIFT**
  - **SIFT and SURF** are good in illumination changes images

Keypoint matching: **SIFT**



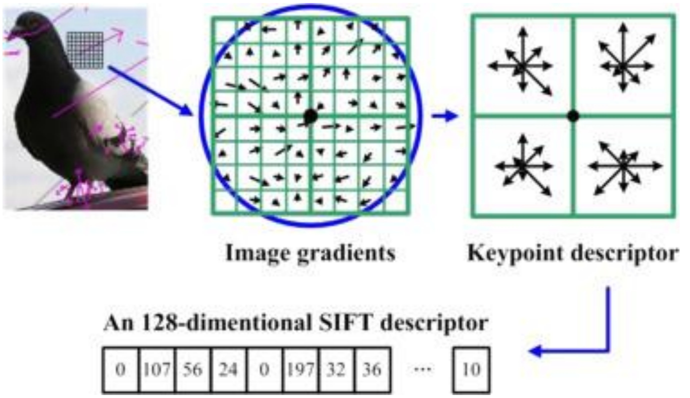
SCALE INVARIANT FEATURE TRANSFORM (SIFT)

1. *Estimate a scale space extrema using the Difference of Gaussian (DoG)*



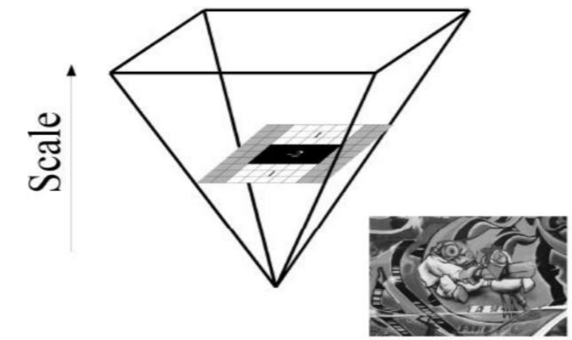
2. *A key point localization where the key point candidates are localized and refined by eliminating the low contrast points*
3. *A key point orientation assignment based on local image gradient and lastly a descriptor generator to compute the local image descriptor for each key point based on image gradient magnitude and orientation*

Keypoint matching: **SIFT usage**



- SIFT keypoints of objects are first extracted from a set of reference images and stored in a database.
- A similar image (or a large object inside) is recognized in a new image by individually comparing each feature from the new image to this database and finding candidate matching features based on Euclidean distance of their feature vectors.
- Plus other refining.. (Outside of the scope of the course)
- **SURF** is used with similar methods

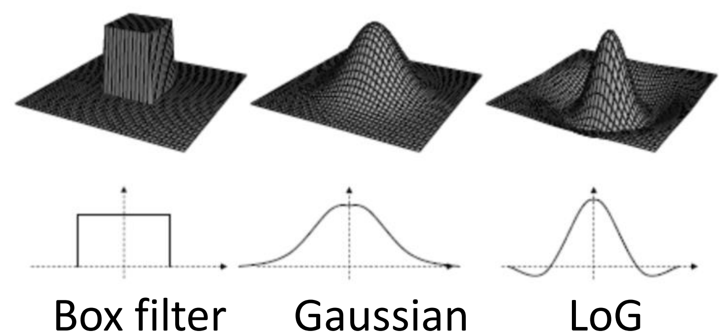
Keypoint matching: **SURF**

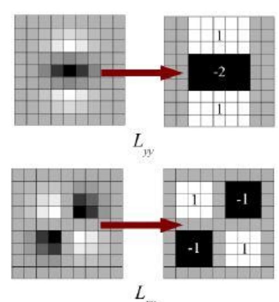


- SPEEDED UP ROBUST FEATURE (SURF)

1. *SURF approximates the Difference of Gaussian DoG with box filters (Faster!)*
2. *BLOB detector which is based on the Hessian matrix to find the points of interest.*
3. *For feature description also SURF uses the wavelet responses.*

Different size of boxfilters
(Laplacian of Gaussian (LoG))
is convoluted with integral image.





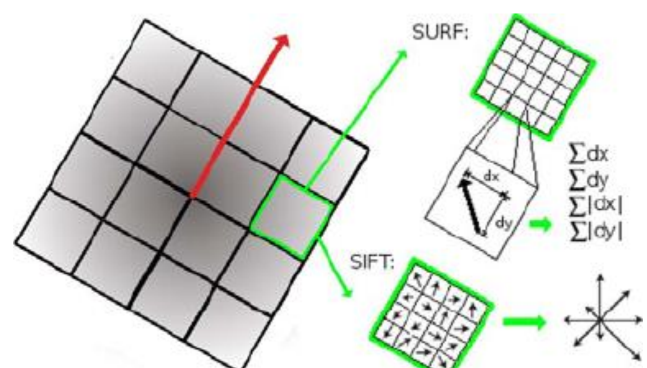
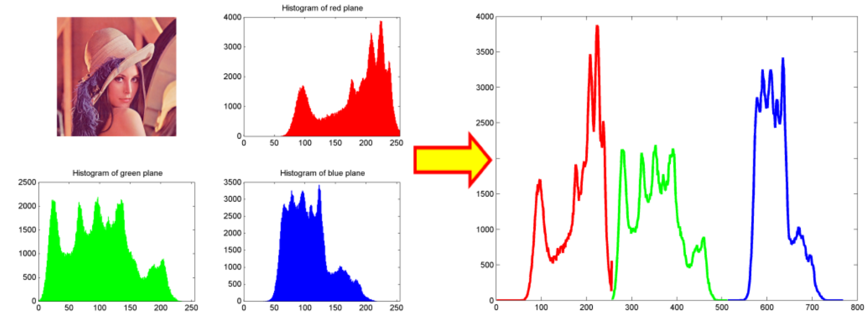


Image Similarity: **Histogram comparison**



- Procedure
  1. Compute the histogram of the image (or quadrants, or blocks)
  2. Create a feature vector
  3. Use a metric to compare images (Wasserstein m.)
- PRO
  - Very fast. Comparison in now on a short 1D vector
  - Scale and rotation compliant
- CON
  - Shapes/Patterns are not relevant!
  - It is too simplistic.
    A banana and a beach
    will be similar

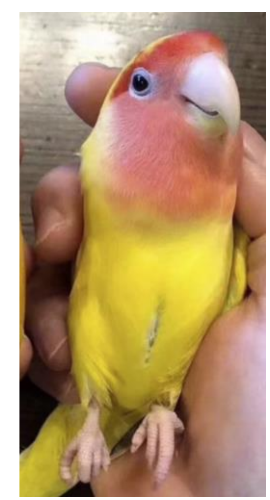
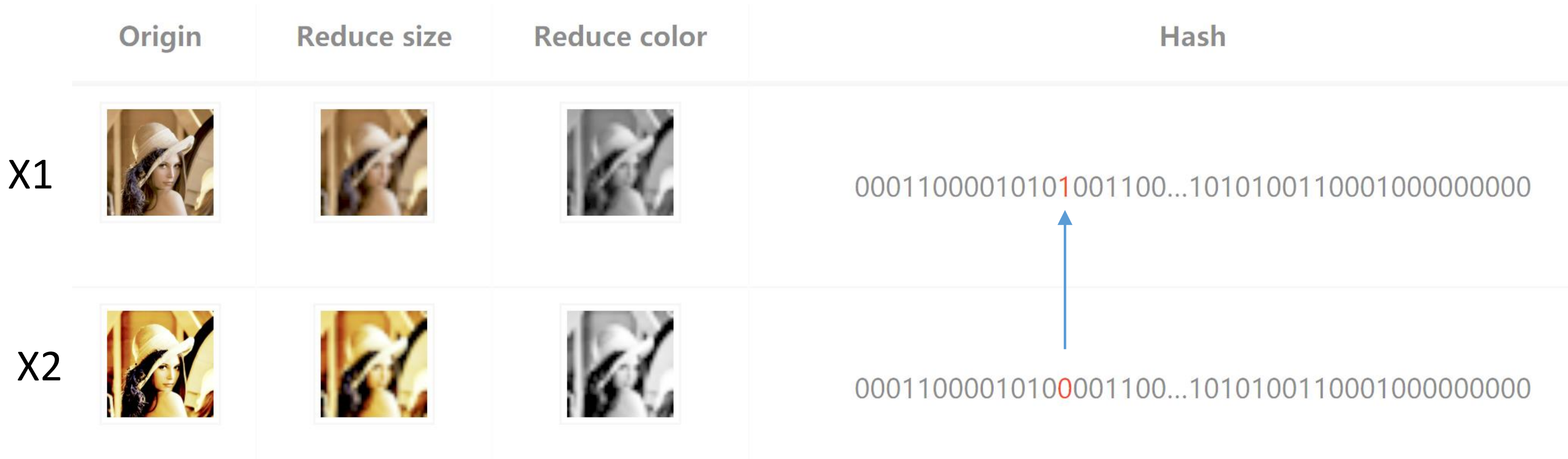


Image Similarity
Image hash

- The image is reduced down to a small hash code (like a “fingerprint”) by identifying salient features in the original image file and hashing a compact representation of those features (rather than hashing the image data directly).
- Finally, count the number of bit positions that are different (Hamming distance).

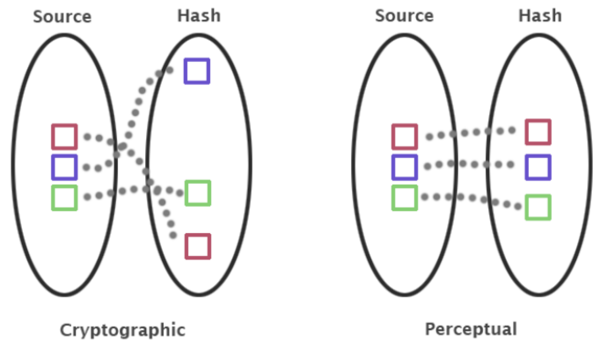


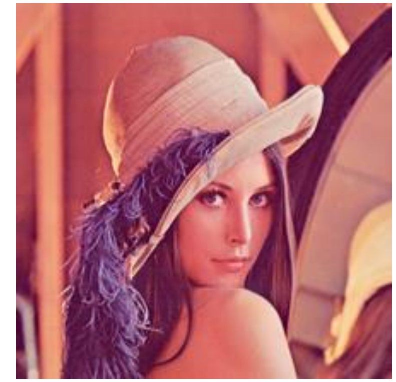
aHash, pHash, dHash

- **3 methods with similar initial steps**
- Looking at **a**verage, **p**erceptive and **g**radient difference and creating a binary feature

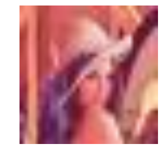
- **aHash** (a = Average)
  1) Reduce size (example: 8x8 R,G,B)
  2) Reduce to gray (8x8)
  3) Reduce to bits (0 if gray > below the mean gray)
  4) Merge the bits into a 64-bit integer. ← your hash

pHash





- **pHash** (p = Perceptive, works in the frequency domain)
  1) Reduce size (32x32 R,G,B)
  2) Reduce to gray (32x32)



32 x 32

$$\begin{bmatrix} 88 & 85 & \cdots & 72 \\ 87 & 87 & \cdots & 114 \\ \vdots & \vdots & \ddots & \vdots \\ 12 & 76 & \cdots & 21 \end{bmatrix}$$

  3) Compute the Discrete Cosine Transform (DCT). It separates the image into a collection of frequencies and scalars (JPEG, and many other format…)

32 x 32

$$\begin{bmatrix} 8365 & -1554 & \cdots & -17 \\ -254 & 338 & \cdots & 13 \\ \vdots & \vdots & \ddots & \vdots \\ -424 & 74 & \cdots & 1.4 \end{bmatrix}$$

  4) Reduce the DCT (just keep the top-left 8x8)

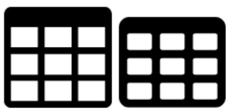
8 x 8

$$\begin{bmatrix} 8365 & -1554 & \cdots & -17 \\ -254 & 338 & \cdots & 13 \\ \vdots & \vdots & \ddots & \vdots \\ -424 & 74 & \cdots & 1.4 \end{bmatrix}$$

  5) Compute the mean DCT value T (not the first value, but using the other 63 values)
  6) Reduce to bits (0 if DCT > below T)
  7) Merge the bits into a 64-bit integer. ← your hash

Images with same pHash

Cluster 802f2a2f2a7f2ad5



Cluster 80542a562f5f7f4a

Cluster 91b16ece313e34c9

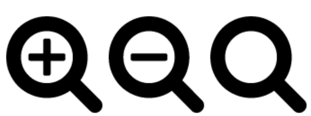
Cluster a0008a0020002000

Cluster aa00800080008000

Cluster aa00800080008200

Cluster aa74f57e942d8225

Cluster ab2df43cd07285c3

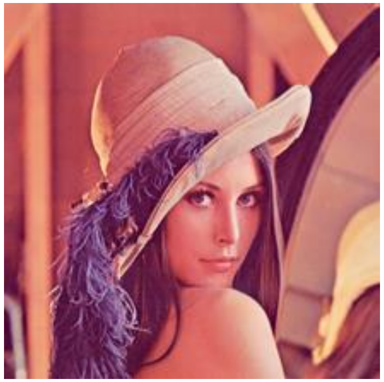


= 8a0303f6df3ec8cd

https://pypi.org/project/ImageHash/

```
>>> hash = imagehash.average_hash(Image.open('test.png'))
>>> print(hash)
d879f8f89b1bbf
```

dHash (very fast)



- **dHash** (d = Difference)
  Calculate the difference for each of the pixel and compares the difference with the average differences.)
  1) Reduce size (9x8 R,G,B)
  2) Reduce to gray (72 gray pixels)
  3) Compute the Differences
     1) Difference between adjacent pixels (relative gradient directions). The 9 pixels per row yields 8 differences between adjacent pixels. Eight rows of eight differences becomes 64 bits.

     Each bit is simply set based on whether the left pixel is brighter than the right pixel
  4) Merge the bits into a 64-bit integer. ← your hash

Main points

- Labelling errors
  - Supervisor errors
  - Changes in time
  - Checks
- Similarity in
  - The general framework of pattern recognition
  - Datasets
    - Eucledean, Manhattan, Cosine, Chebyshev, Minkowski
  - Images
    - Histograms, SIFT, SURF
    - aHash, dHash, pHash