**Hugo Proença**

University of Beira Interior

IT: Instituto de Telecomunicações,

Portugal

# *Image Understanding*

# Image Understanding: Summary

1. Image/Scene Understanding
    1. Summary
    2. Understanding *vs.* Interpretability/Explainability?
    3. Hardness
2. Convolutional Neural Networks
    1. Image Classification/Regression
    2. Object Detection
    3. Semantic Segmentation
    4. Generative Models
3. Image Understanding *vs.* Interpretability/Explainability
    1. Understanding and Interpretability/Explainability
    2. Visual Interpretability
4. Conclusions

# Image Understanding

- Image understanding can be understood as the problem of **interpreting** images by **locating** and **recognizing objects** with their **attributes** and other **higher-level features.**
- **Image understanding** comes typically after low-level processing phases (pre-processing, normalization, edge detection, etc…)
  - It regards the **process of interpreting those regions/objects,** to perceive the image contents. This may include figuring out what the objects are and their spatial relationship. It may also include ultimately making some decision for further action.
  - This is typically the **bottom-up** approach: starts from the image data, towards objects representations and classification, in order to provide information to higher-level analysis processes and decisions.

# Understanding *vs.* Interpretability/Explainability?

- *'Understand*' and *'interpret*' have a similar semantic, but often there is a subtle distinction. *'To understand*' is what you think it means; *'to interpret*' *is to be able to draw conclusions from a concept which you understand*.

- [**WikiDiff**] "*The difference between interpret and understand is that interpret is to explain or tell the meaning of; to expound; to translate orally into intelligible or familiar language or terms; to decipher; to define; -- applied especially to language, but also to dreams, signs, conduct, mysteries, etc; as, to interpret the hebrew language to an englishman; to interpret an indian speech while understand is to be aware of the meaning of.*"

- Most of the SOTA image-based models remain complex **black boxes**, meaning their internal logic and inner workings are hidden to the user and even experts cannot **understand the rationale** behind their predictions.

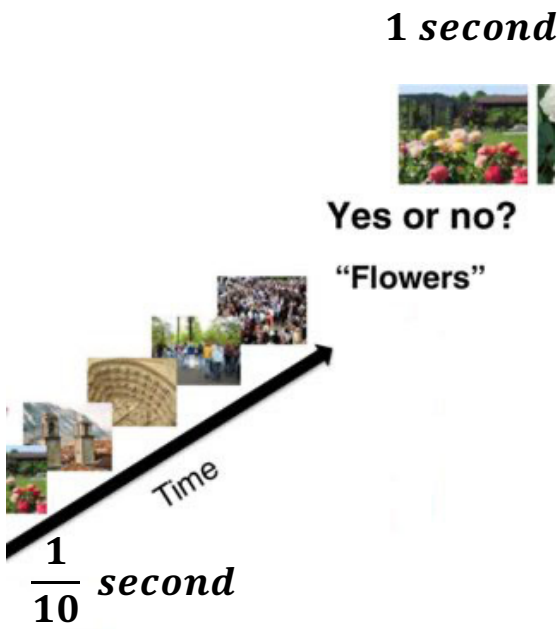
- This way, it is evident **that there is a significant overlap** between the concepts of understanding/interpreting, which also happens between "**Image understanding**" and "**Image interpretation**"

# Image Understanding

- Image understanding can also work in the **top-down** perspective. In such case, we start by making some high-level hypotheses of the image content, and then use the data to validate/reject those hypotheses.
  - In this paradigm, most of the approaches are **model-based**: That is, they start by an approximate model of what they think they're looking at, then fit that model to the data.
  - In a sense, primitive-fitting approaches such as the Hough transform used this idea.
  - This idea can be extended further to so-called deformable models, in which you can deform the model to better fit the data. The goodness of the match is the inverse of how much you must work to deform the model to fit the data.
- **Active Vision:** It is a third alternative, neither quite bottom-up or top-down, often used when the vision system is part of a larger system capable of acting so as to be able to influence the position/view/etc. In such cases, *active vision* approaches accurately model the way people interact with the world.
  - The idea is to make a tentative hypothesis (using either top-down or bottom-up processing) then ask: "*based on what I already know, what should I do to be able to acquire the information that will help me analyze the image*?"

# Image/Scene Understanding: Hardness

- Let's start by playing an easy test: Mary Potter's test.
- This test was created in 1976, and the idea is to perceive whether during a rapid sequential visual presentation (1/10 second per image), one image can be understood by observers, i.e., if they are able to comprehend its visual information.
- This test has two phases:
  - At first, a set of images is rapidly shown (1/10 second each) to observers.
  - Then, another set of images is shown (one second each) and the task demanded to observers is to discriminate between the previously seen/unseen images



# Mary Potter Test - 1

# Mary Potter Test - 2

# Mary Potter Test - 3

# Mary Potter Test - 4

# Mary Potter Test - 5

# Mary Potter Test - 6

# Mary Potter Test - 7

# Mary Potter Test - 8

# Mary Potter Test - 9

# Mary Potter Test – Have we Seen This Image?

**#1**

# Mary Potter Test – Have we Seen This Image?

**#2**

# Mary Potter Test – Have we Seen This Image?

**#3**

# Mary Potter Test – Have we Seen This Image?

**#4**

# Mary Potter Test – Have we Seen This Image?

**#5**

# Mary Potter Test – Have we Seen This Image?

**#6**

# Mary Potter Test – Have we Seen This Image?

These images were OK. We've seen them before:

**#1,5**

But these we have not!

# Mary Potter Test – Have we Seen This Image?

The main conclusions is that it is **easy to remember the global layout** one one image, and in some sense its meaning, but clearly **some object and details are ignored**!

# Mary Potter Test – Have we Seen This Image?

The main conclusion is that it **is easy to remember the global layout** of one image, and in some sense its meaning, but clearly **some object and details are ignored**!

# Scene Understanding and Semantic Segmentation

- As deep learning-based **Semantic Segmentation** models appeared, the initial idea was that they will provide a breakthrough advance in scene understanding (and even in general artificial intelligence).
- The promise was to obtain models that are highly efficient (e.g., 30 FPS) and accurate in delimiting all the objects in general scenes.
- However, “reality” has been putting additional obstacles (e.g., cross-domain performance, and lack of context)

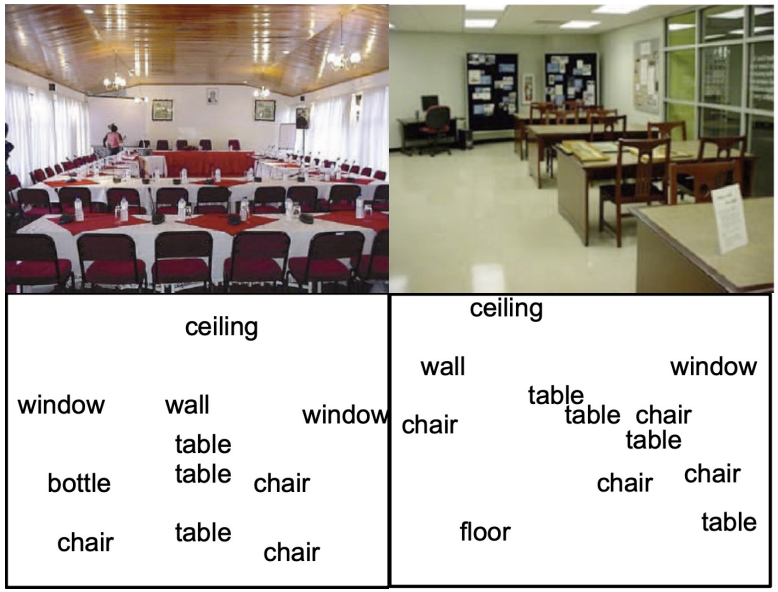
# Scene/Global Image Understanding

- Fully understanding a scene requires more than recognizing all objects in the image.
- It is **heavily dependent of the context**, and that is exactly the hard part of SOTA computer vision systems: they don't simply understand it.
- Two scenes can be different, due to the existence of **different objects**, with **different spatial layout**:


✓ **Feasible**

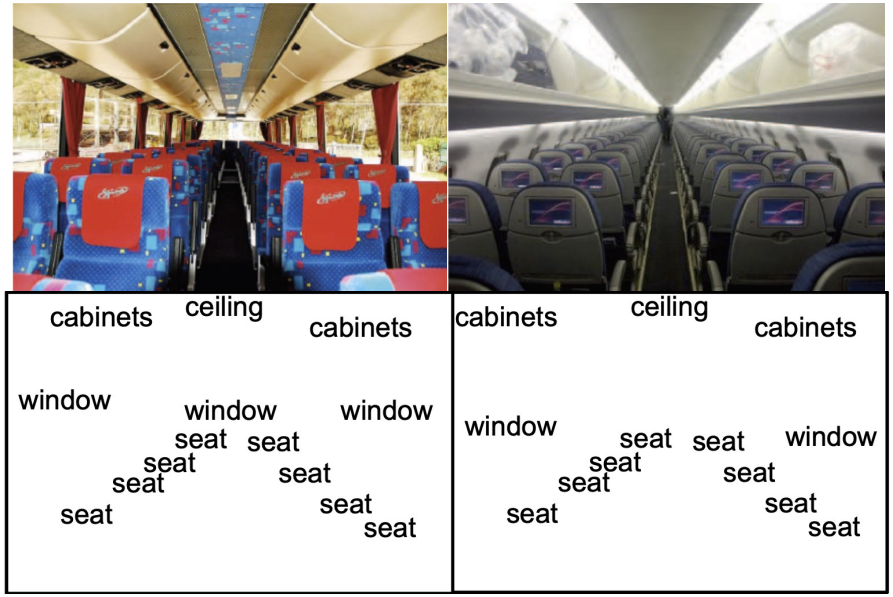
# Scene/Global Image Understanding

- But there harder cases, where either the objects **are equal** (and only the **layout changes**), but even the **objects** and **layout** can be **the same**.




**Same objects, Different layout**

**Feasible** ✓

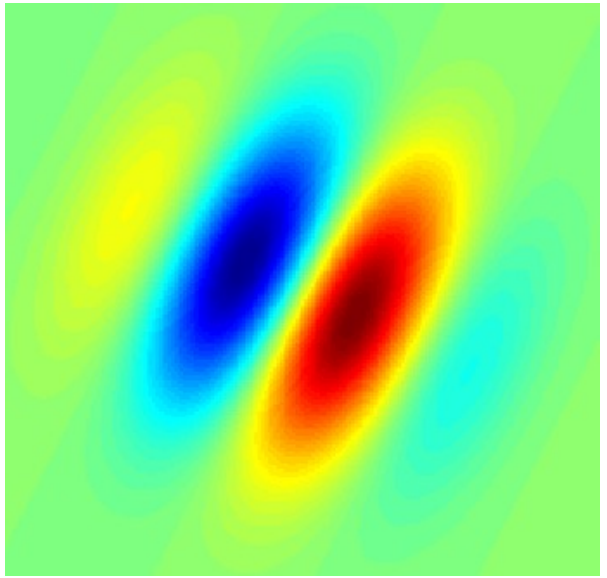



**Same objects, Similar layout**

**Particularly hard!** ✗

# Convolutional Neural Networks (CNNs)

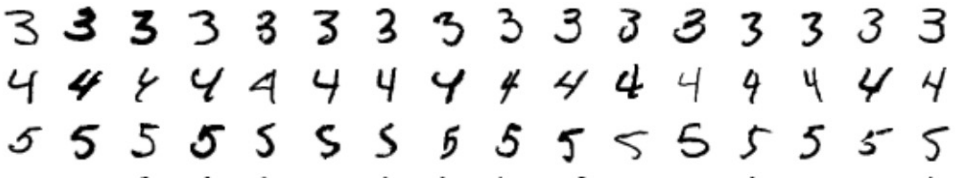
- **Convolutional Neural Networks** (**CNNs**) are a type of models that have been increasing their popularity in most tasks related to Computer Vision (e.g., image classification, object detection and segmentation)
- Their roots date back to 1959, when David Hubel and Torsten Wiesel described ***simple cells*** and ***complex cells*** in the human visual cortex. They proposed that both kinds of cells can be used in Pattern Recognition.
  - A simple cell responds to edges and bars of a particular orientation at a particular position of an image.
  - A complex cell responds to to a specific bar at any position of an image (i.e., at the bottom, middle or top)
  - Hubel and Wiesel found out that the behavior of complex cells can be obtained by **summing up the output of multiple simple cells**.



**Simple cell**

# Convolutional Neural Networks (CNNs)

- Than, in 1998, Y. LeCun *et* al proposed the seminal concept of CNN in their "*Gradient-Based Learning Applied to Document Recognition*" paper.
- This work described a CNN model where simpler features at the first layers are aggregated at subsequent layers, creating more complicated spatial-aware features.
- At the end, features are flattened and feed a traditional feed-forward network that performs classification.
- In their case, the problem regarded the well known **MNIST database** of handwritten digits.





## Gradient-Based Learning Applied to Document Recognition

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner

*Abstract*—
**Multilayer Neural Networks trained with the backpropagation algorithm constitute the best example of a successful Gradient-Based Learning technique. Given an appropriate network architecture, Gradient-Based Learning algorithms can be used to synthesize a complex decision surface that can classify high-dimensional patterns such as handwritten characters, with minimal preprocessing. This paper reviews various methods applied to handwritten character recognition and compares them on a standard handwritten digit recognition task. Convolutional Neural Networks, that are specifically designed to deal with the variability of 2D shapes, are shown to outperform all other techniques.**

**Real-life document recognition systems are composed of multiple modules including field extraction, segmentation, recognition, and language modeling. A new learning paradigm, called Graph Transformer Networks (GTN), allows such multi-module systems to be trained globally using Gradient-Based methods so as to minimize an overall performance measure.**

**Two systems for on-line handwriting recognition are described. Experiments demonstrate the advantage of global training, and the flexibility of Graph Transformer Networks.**

**A Graph Transformer Network for reading bank check is also described. It uses Convolutional Neural Network character recognizers combined with global training techniques to provides record accuracy on business and personal checks. It is deployed commercially and reads several million checks per day.**

*Keywords*— **Neural Networks, OCR, Document Recognition, Machine Learning, Gradient-Based Learning, Convolutional Neural Networks, Graph Transformer Networks, Finite State Transducers.**

### Nomenclature

- GT Graph transformer.
- GTN Graph transformer network.
- HMM Hidden Markov model.
- HOS Heuristic oversegmentation.
- K-NN K-nearest neighbor.
- NN Neural network.
- OCR Optical character recognition.
- PCA Principal component analysis.
- RBF Radial basis function.
- RS-SVM Reduced-set support vector method.
- SDNN Space displacement neural network.
- SVM Support vector method.
- TDNN Time delay neural network.
- V-SVM Virtual support vector method.

### I. Introduction

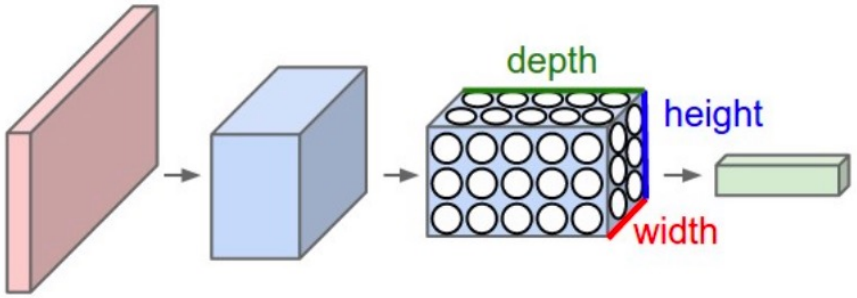
Over the last several years, machine learning techniques, particularly when applied to neural networks, have played an increasingly important role in the design of pattern recognition systems. In fact, it could be argued that the availability of learning techniques has been a crucial factor in the recent success of pattern recognition applications such as continuous speech recognition and handwriting recognition.

The main message of this paper is that better pattern recognition systems can be built by relying more on automatic learning, and less on hand-designed heuristics. This is made possible by recent progress in machine learning and computer technology. Using character recognition as a case study, we show that hand-crafted feature extraction can be advantageously replaced by carefully designed learning machines that operate directly on pixel images. Using document understanding as a case study, we show that the traditional way of building recognition systems by manually integrating individually designed modules can be replaced by a unified and well-principled design paradigm, called *Graph Transformer Networks*, that allows training all the modules to optimize a global performance criterion.

Since the early days of pattern recognition it has been known that the variability and richness of natural data, be it speech, glyphs, or other types of patterns, make it almost impossible to build an accurate recognition system entirely by hand. Consequently, most pattern recognition systems are built using a combination of automatic learning techniques and hand-crafted algorithms. The usual method of recognizing individual patterns consists in dividing the system into two main modules shown in figure 1. The first module, called the feature extractor, transforms the input patterns so that they can be represented by low-dimensional vectors or short strings of symbols that (a) can be easily matched or compared, and (b) are relatively invariant with respect to transformations and distortions of the input patterns that do not change their nature. The feature extractor contains most of the prior knowledge and is rather specific to the task. It is also the focus of most of the design effort, because it is often entirely hand-crafted. The classifier, on the other hand, is often general-purpose and trainable. One of the main problems with this approach is that the recognition accuracy is largely deter-

# Convolutional Neural Networks (CNNs)

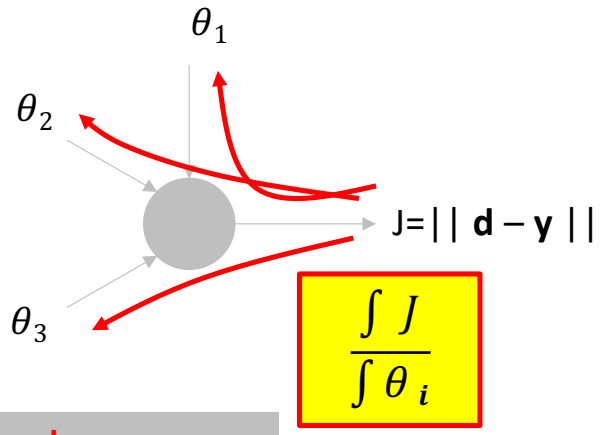
- The property of **shift invariance** gives to CNNs the **biological inspiration** of the **human visual system, while** keeping the **number of weights relatively small**, making learning a feasible task.
- In opposition to traditional nets, neurons in CNNs are arranged in **three dimensions**.
- Essentially, each layer of a CNN transforms a 3D input into a 3D output.
- The efficacy of CNNs in visual tasks is the main reason behind the popularity of deep learning. They are powering major advances in computer vision, with applications for robotics, security and medical diagnosis.



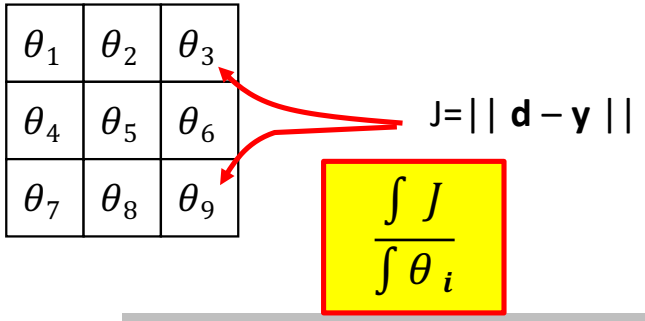

- However: They are seen as an evidently "temporary solution", even by their own inventors (e.g., capsule networks might be the next step?)

# Convolutional Neural Networks (CNNs)

- The breakthrough conceptual advance of **Deep Learning** solutions is to perform in a "single-shot" the feature encoding and classification/regression (decision) phases.
- But, how exactly are such kind of frameworks extracting the ideal feature sets, for a specific problem?



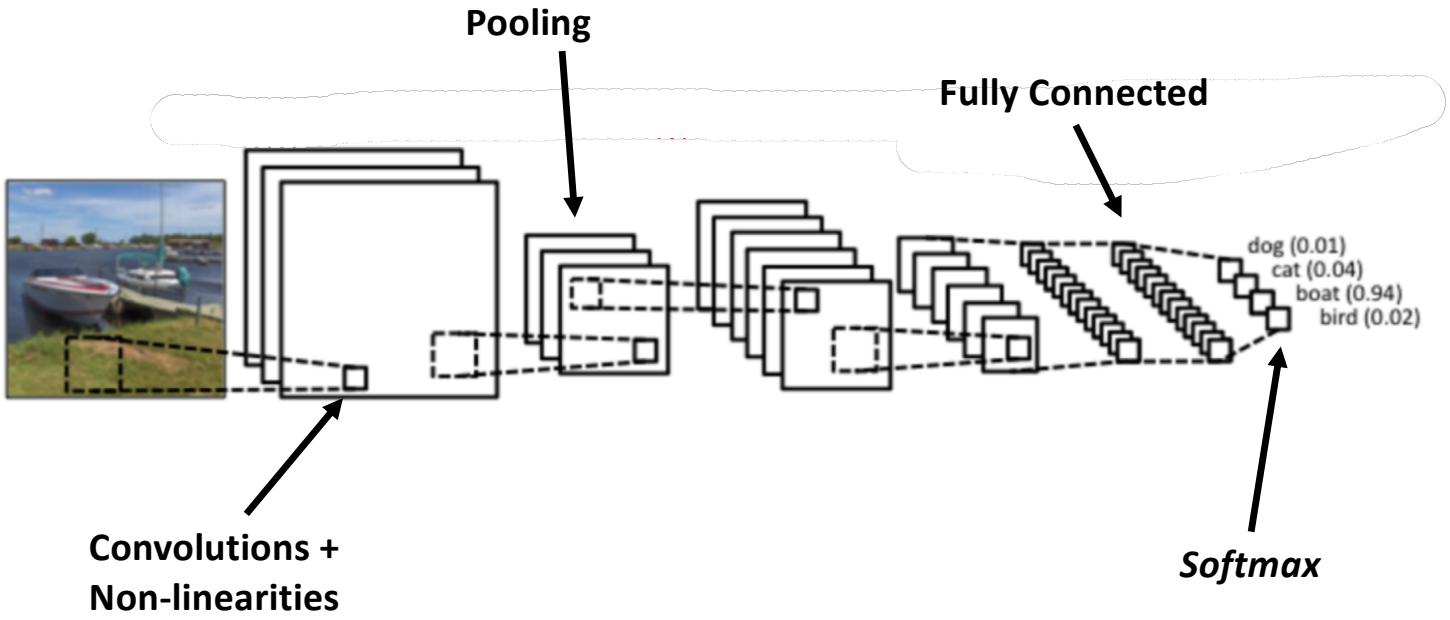

**Traditional Networks:** Each **red arrow** adjusts the values of one network parameter, depending of its importance in the observed error (during backpropagation)




**CNNs:** Each **red arrow** adjusts the values of one component of a convolution kernel, depending of its importance in the observed error (during backpropagation)

# Convolutional Neural Networks (CNNs)

- The typical architecture of CNNs is as follows:




These operations are the basic building blocks of *most CNNs*, so understanding how these work is an important step to actually understand the functioning of these powerful models.

# Convolutional Neural Networks (CNNs)

- **Convolution Layers**
  - These blocks are the most important and perform the convolution between an input map **x** with a bank of D'' multi-dimensional filters **f,** to obtain the result **y.**

$$\mathbf{x} \in \mathbb{R}^{H \times W \times D}, \quad \mathbf{f} \in \mathbb{R}^{H' \times W' \times D \times D''}, \quad \mathbf{y} \in \mathbb{R}^{H'' \times W'' \times D''}.$$

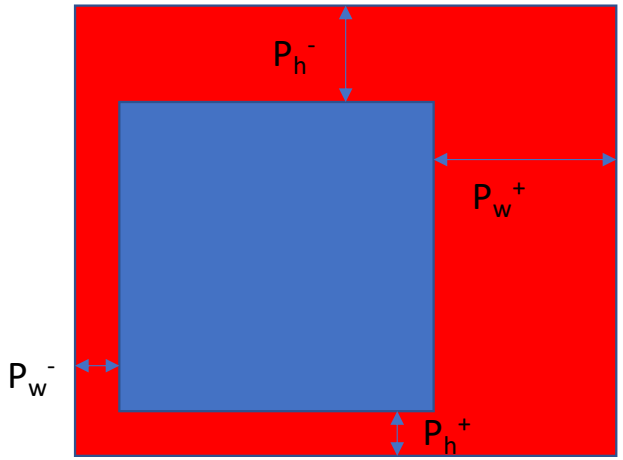
- Formally, the outputs **y** are given by:

$$y_{i''j''d''} = b_{d''} + \sum_{i'=1}^{H'} \sum_{j'=1}^{W'} \sum_{d'=1}^{D} f_{i'j'd} \times x_{i''+i'-1, j''+j'-1, d', d''}.$$

# Convolutional Neural Networks (CNNs)

- Convolution (padding and stride)
  - Usually, it is possible to specify top, bottom, left, right paddings ($P_h^-$, $P_h^+$, $P_w^-$, $P_w^+$) of the input array and subsampling strides ($S_h$,$S_w$) of the output array.

$$y_{i''j''d''} = b_{d''} + \sum_{i'=1}^{H'} \sum_{j'=1}^{W'} \sum_{d'=1}^{D} f_{i'j'd} \times x_{S_h(i''-1)+i'-P_h^-, S_w(j''-1)+j'-P_w^-, d', d''}.$$




The output size is given by:

$$H'' = 1 + \left\lfloor \frac{H - H' + P_h^- + P_h^+}{S_h} \right\rfloor.$$

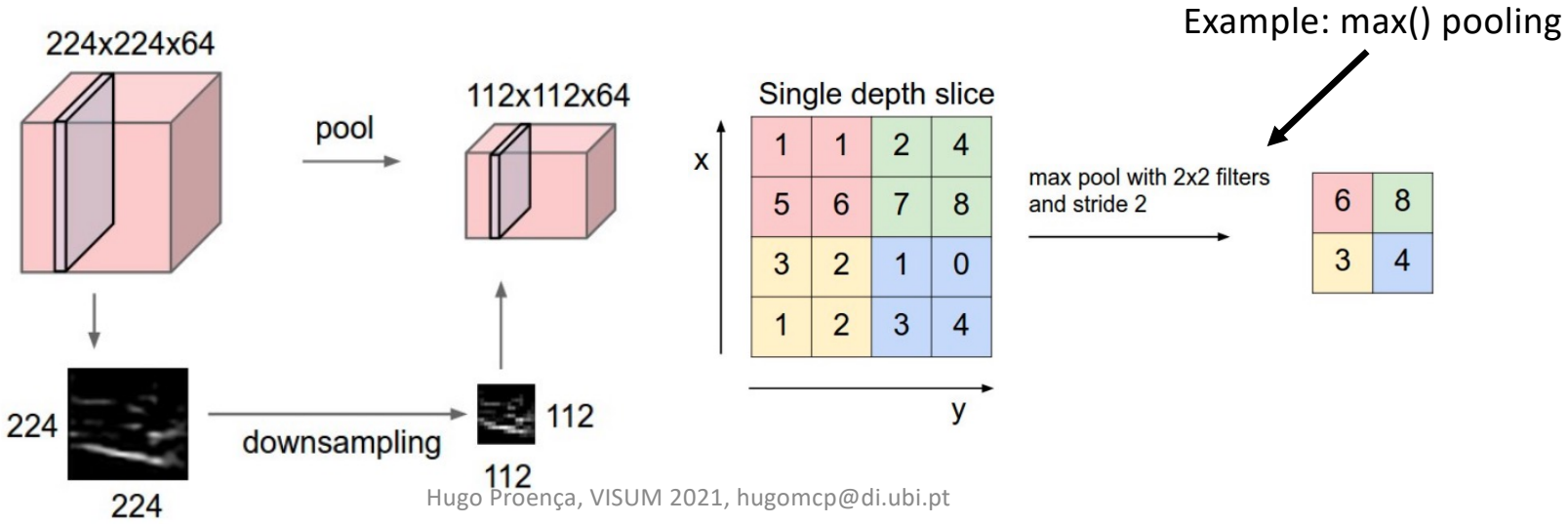
# Convolutional Neural Networks (CNNs)

- **Spatial Pooling Layers**
  - The typical blocks are the *max()* and *sum*() functions, respectively obtaining the maximum and the summed responses of each feature channel in a H' x W' patch.
- Pooling progressively reduces the spatial size of the input representation.
  - This reduces the number of parameters and, therefore, limits the probability of over-fitting;
  - Also, it makes the network invariant to small transforms, distortions and translations in the input image (a small distortion in input will not change the output of pooling).

$$y_{i''j''d} = \max_{1 \le i' \le H', 1 \le j' \le W'} x_{i''+i'-1, j''+j'-1, d}. \quad y_{i''j''d} = \frac{1}{W'H'} \sum_{1 \le i' \le H', 1 \le j' \le W'} x_{i''+i'-1, j''+j'-1, d}.$$

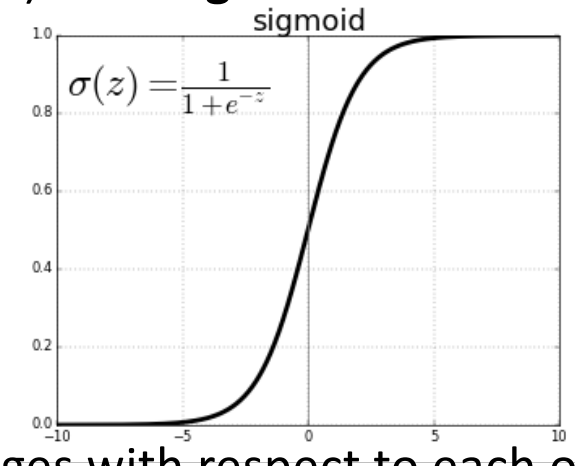
# Convolutional Neural Networks (CNNs)

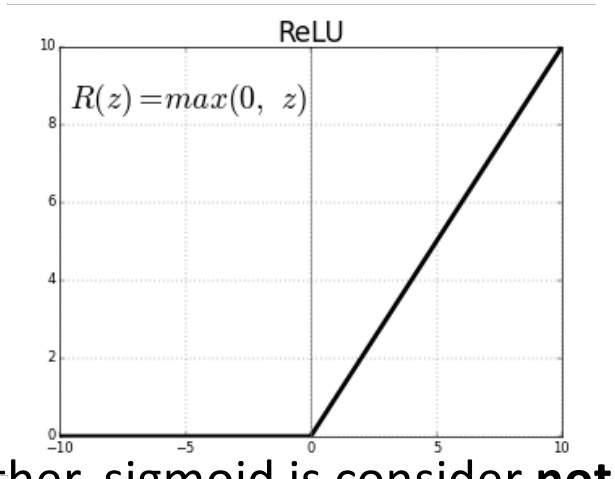
- **Pooling Layers**
  - Note that Pooling down samples the input volume only spatially;
    - The input depth is equal to the output depth;
  - The pooling operation is often considered **deprecated**. To reduce the size of the representation, larger strides in the convolution layers can be used.



# Convolutional Neural Networks (CNNs)

- **Non-Linearity Functions**
  - There are two main non-linear activation functions used in CNNS: "**ReLU**" (Rectified Linear Units) and "**sigmoid**":




  - As advantages with respect to each other, sigmoid is consider **not to blow up activation**, while ReLU **does not vanishes the gradient**
    - In the case of Sigmoid, when the input grows to infinitely large, the derivative tends to 0.
  - However, in the case of ReLU, there is no mechanism to constrain the output of the neuron, as the input is often the output.

# Convolutional Neural Networks (CNNs)

- **Fully Connected Layers**
  - Neurons in a fully connected layer have full connections to all activations in the previous layer, as in a regular feed-forward network.
  - In practical terms, these neurons resemble pretty much the neurons in ”Convolution” layers.
    - The main difference between fully connected and Convolution layers is that the neurons in the former layer are connected only to a local region in the input, and that many of the neurons in a “Convolution” volume share parameters.
    - However, the neurons in both layers still compute dot products, so their functional form is identical.
  - For example, a Fully Connected layer with K=4096 that is looking at some input volume of size 7×7×512 can be expressed as a Convolution layer with F=7 x 7 x 4096 (padding 0, stride 1).
    - In other words, we are setting the filter size to be exactly the size of the input volume;
    - Hence the output will simply be 1×1×4096.
    - Fully Connected layers have typically the majority of weights in the network.

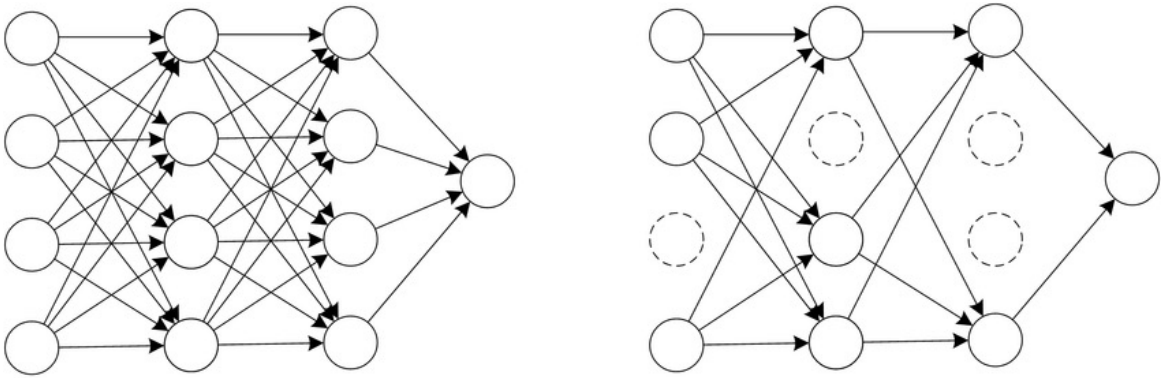
# Convolutional Neural Networks (CNNs)

- **Softmax Layers**
  - This block can be seen as the combination of an activation function (exponential) and a normalization operator.
  - It is usually applied as the transfer function of the last layer of the CNN, where the idea is to push up the maximum value of the responses to “1”, and all the other values to “0”.
  - In practice, it simulates the probability of the input corresponding to each category, represented by a neuron in the output layer.

$$y_{ijk} = \frac{e^{x_{ijk}}}{\sum_{t=1}^{D} e^{x_{ijt}}}.$$

# CNNs: Other Layers

- **Dropout** Layers. This kinds of layers drops out units of a neural network during the learning phase.
  - Typically, a proportion (0, 1) of neurons is randomly chosen and not considered for a particular "**forward**/**backward**" pass.
  - Dropout is an approach to regularization in neural networks which helps to avoid interdependent learning amongst the neurons.
  - Recall that regularization is way to **prevent over-fitting**, by adding a penalty to the loss function.
  - It is applied exclusively to the fully connected layers of a CNN model.

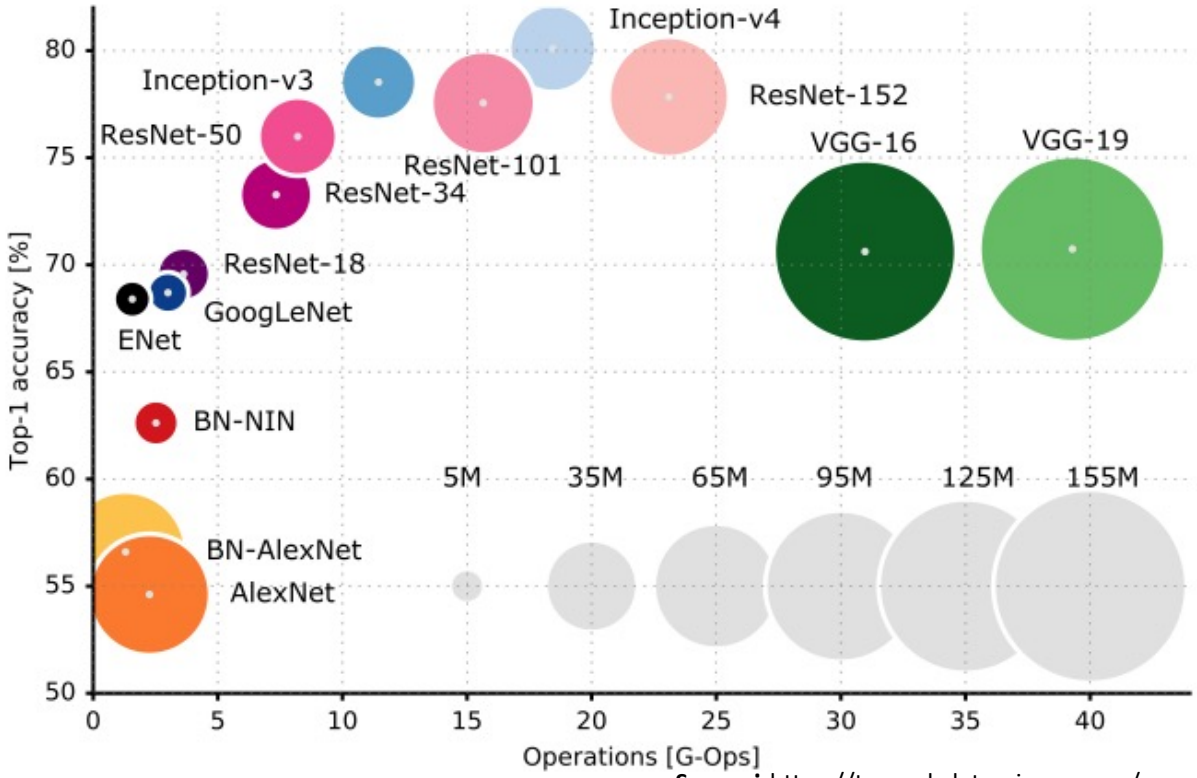


# CNNs: Other Layers

- **Batch Normalization** Layers. To increase the stability of a neural network, this kind of layers normalizes the output of a previous layer by **subtracting** the batch mean and **dividing** by the batch standard deviation.
- This kind of layer can be added both after fully connected layers, but also after convolutional layers.
- Typically, using batch normalisation: 1) allows **higher learning rates;** 2) makes weights **easier to initialise**, helping to reduce the sensitivity to the initial starting weights.
- As the activations of one layer are the inputs of the next one, each layer in the neural network receives – at each iteration – diferente input distributions. This is problematic because it forces each layer to continuously adapt to its changing inputs.
- Using Batch Normalization allows the layer to learn on a more stable distribution of inputs (close to a standardized Gaussian distribution) and is accepted that it speeds up the training of the network.

# Convolutional Neural Networks (CNNs)

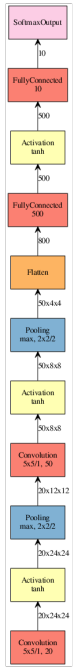
- Accuracy vs. Number of operations for a single forward step. Circumference radii corresponds to the number of parameters




**Source**: https://towardsdatascience.com/neural-network-architectures-156e5bad51ba

# Convolutional Neural Networks (CNNs)

- An illustration of the most popular deep learning architectures is provided in
  http://josephpcohen.com/w/visualizing-cnn-architectures-side-by-side-with-mxnet/

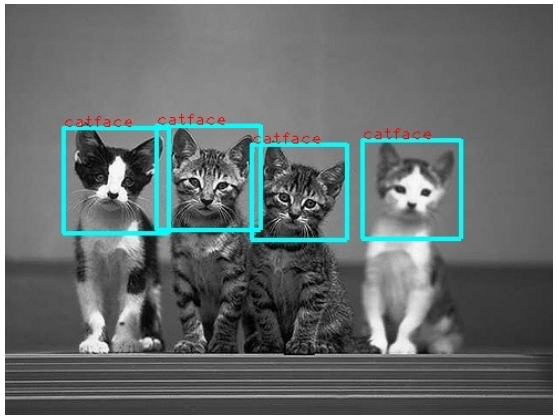


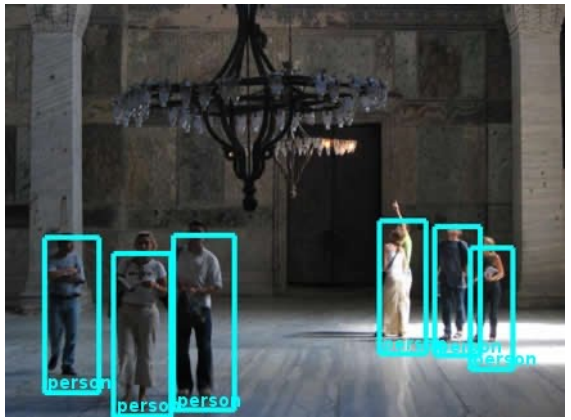



LeNet AlexNet VGG GoogLeNet Inception Resnet

# Object Detection

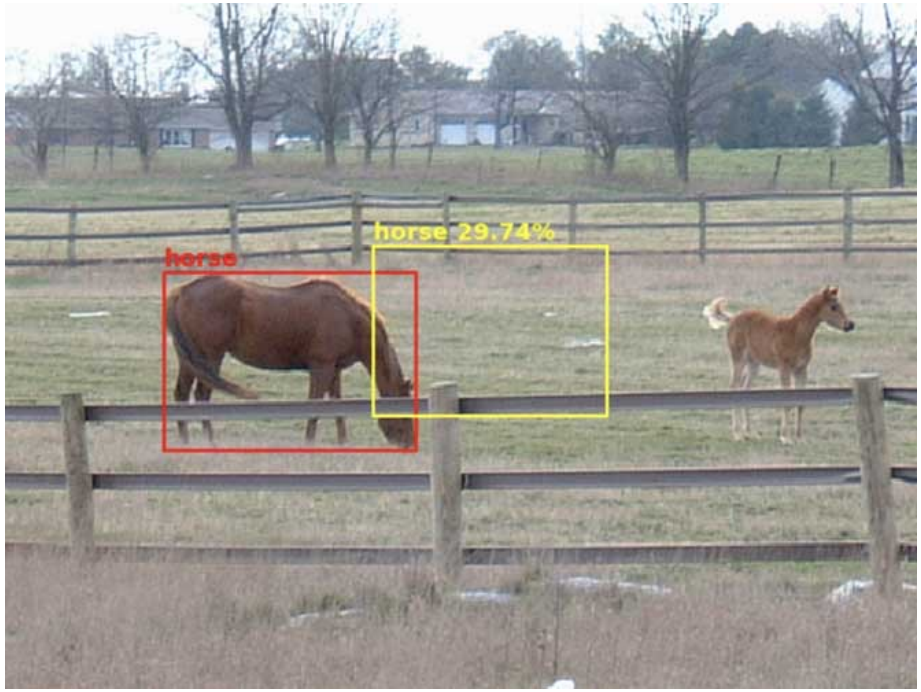
- The goal of Object Detection is to **roughly detect and parameterize** the regions-of-interest (ROI) that contain instances of the object to be handled by the system.
- By rough parameterizations, we mean a relatively small set of numbers that can define a region in the image.
  - E.g., (x1, y1, x2, y2) for a rectangular patch
- It is an <u>image (patch) classification</u> task that is often regarded as a *relatively easy* task



# Object Detection

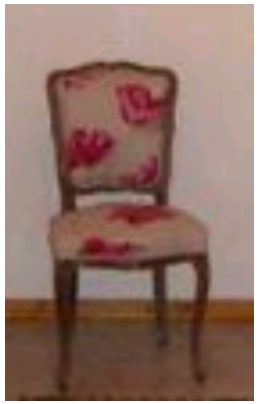
- However, there are multiple varying factors in the acquired data, that should be appropriately handled by the detector:
  - Lighting (shadows);
  - Shape (e.g., scale, translation, rotation);
  - Pose (perspective);
  - Background clutter;
  - Object deformations;
  - Occlusions;
  - Resolution (blur);



# Object Detection - Hardness

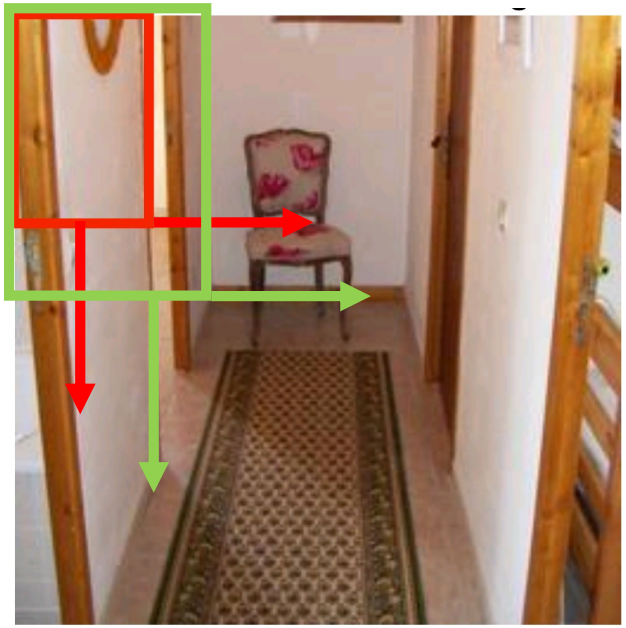
- Intuitively, if we have a good model of the object to be detected, and a scene where the object appears with similar features, the problem is not too hard:

**Chair**

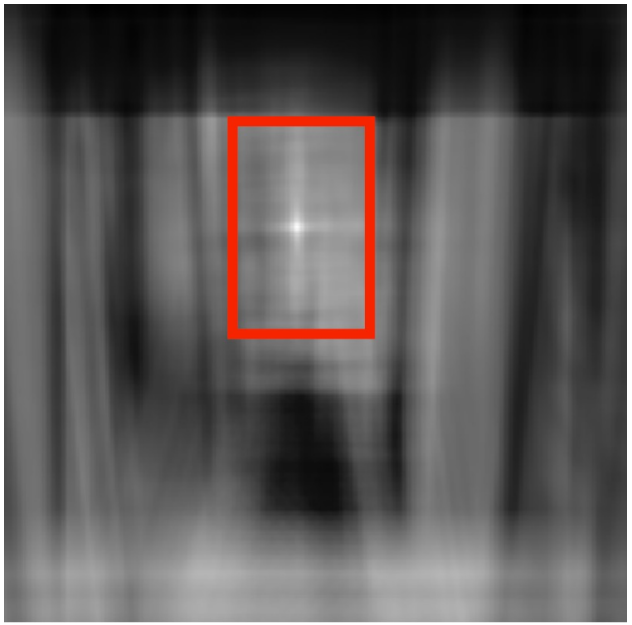



**Multi-scale Analysis**

***"Find the Chair in the Image"***



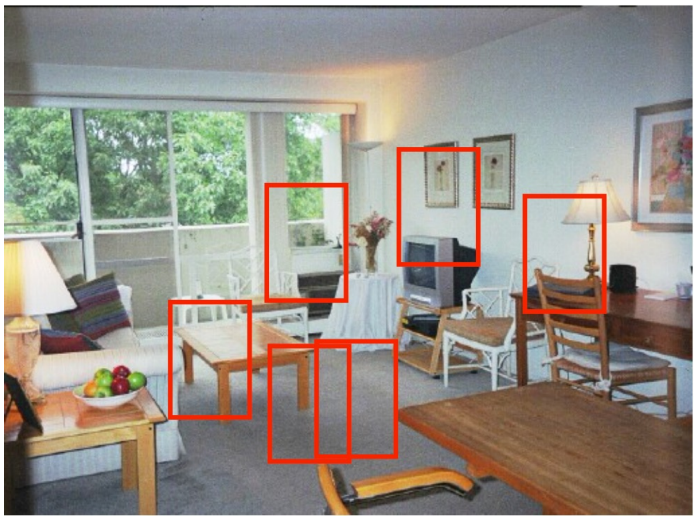
**Resulting *Heat Map***



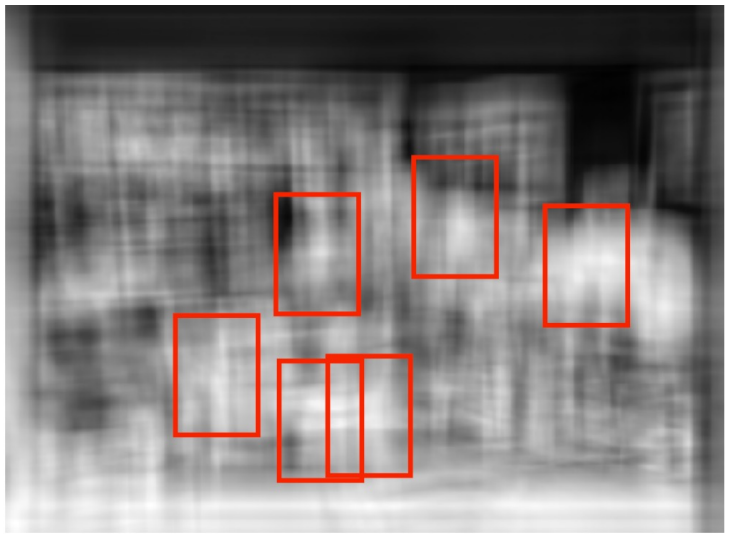
# Object Detection - Hardness

- The problem becomes much harder in case of “changes in domain” , i.e., when the learning data (model) and the test set have very different features (i.e., “real-world” conditions)

***“Find the Chair in the Image”***



**Heat Maps**



# Object Detection – AdaBoost Detector

- Originally proposed by Viola and Jones, it was up to a few years ago (pre deep learning era) the most popular object detection algorithm
  - Paul A. Viola, Michael J. Jones: Robust Real-Time Face Detection. International Journal of Computer Vision 57(2): 137-154 (2004).
- It requires a set of “binary” training data, labelled as positive (1) and negative (0) samples.

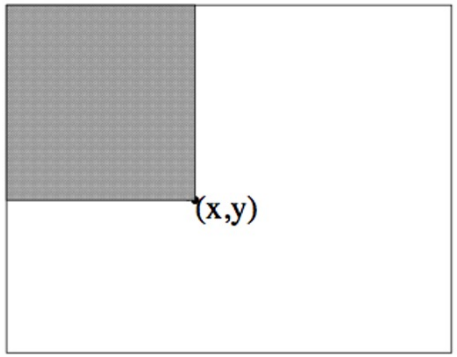
**Faces**

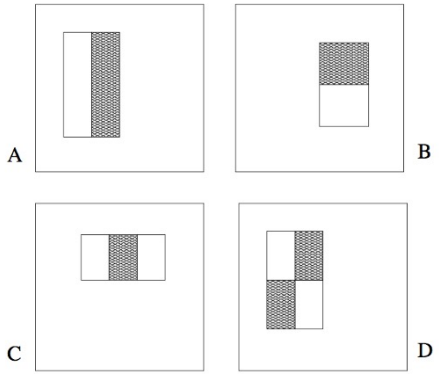
**Non-Faces**

- Builds a strong detector from a set of very simple (weak) detectors.
- It exploits the correlation between weak detectors.
- This yields a detector able to work in real-time.

# Object Detection – AdaBoost Detector

- In this detector, the Features are reminiscent of Haar-basis functions:
- The value of each feature (rectangular region with vertices $v_1$, $v_2$, $v_3$, and $v_4$) is given by the difference between sum of intensities in rectangular regions.
- In order to obtain each value in an efficient way, the concept of integral image *"ii(x,y)"* was proposed: $ii(x,y) = \sum_{y'=1}^{y} \sum_{x'=1}^{x} i(x,y)$ , where *i(x,y)* is the image intensity
- The concept of rectangular image enables to obtain an image integral (summation), just by accessing to one image position. Based on this, there are four different types of features (A, B, C, D) were proposed.



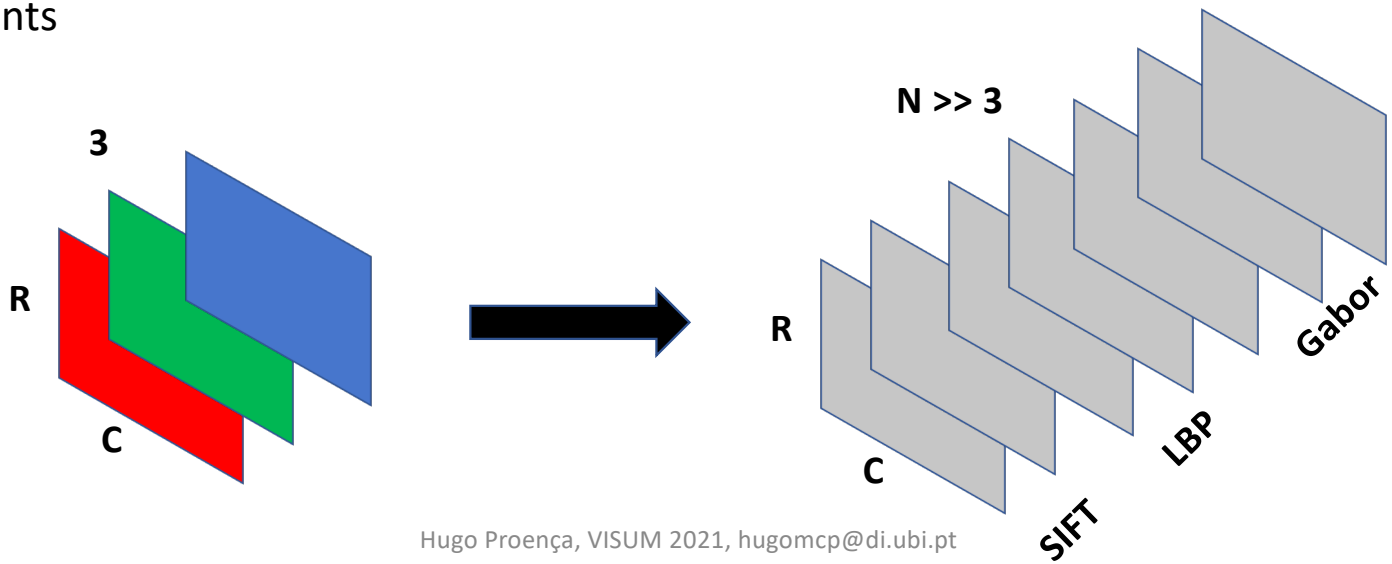



# Object Detection – AdaBoost Detector

- Finally, a set of **Weak Classifiers** is created (concept of Boosting).
- Each weak classifier h(x,y,p,t) only analyses one of the features, centered at a given position (x,y), where the most discriminating power between the negative and positive samples occurs at threshold “t” and comparison sign “p”
  - $h(x,y,p,t) = p.f(x,y) < p.t$
- For each possible feature, a weak classifier is built.
- A set of weights are initialized, according to the total of negative and positive samples
- At each iteration the algorithm chooses the best weak classifier (i.e., the one with lowest error)
- Update the weights of the remaining classifiers, so that those that do not fail in the cases where selected weak classifiers fail are privileged
- The final strong classifier is a function of all the weak classifiers and of their corresponding weights.

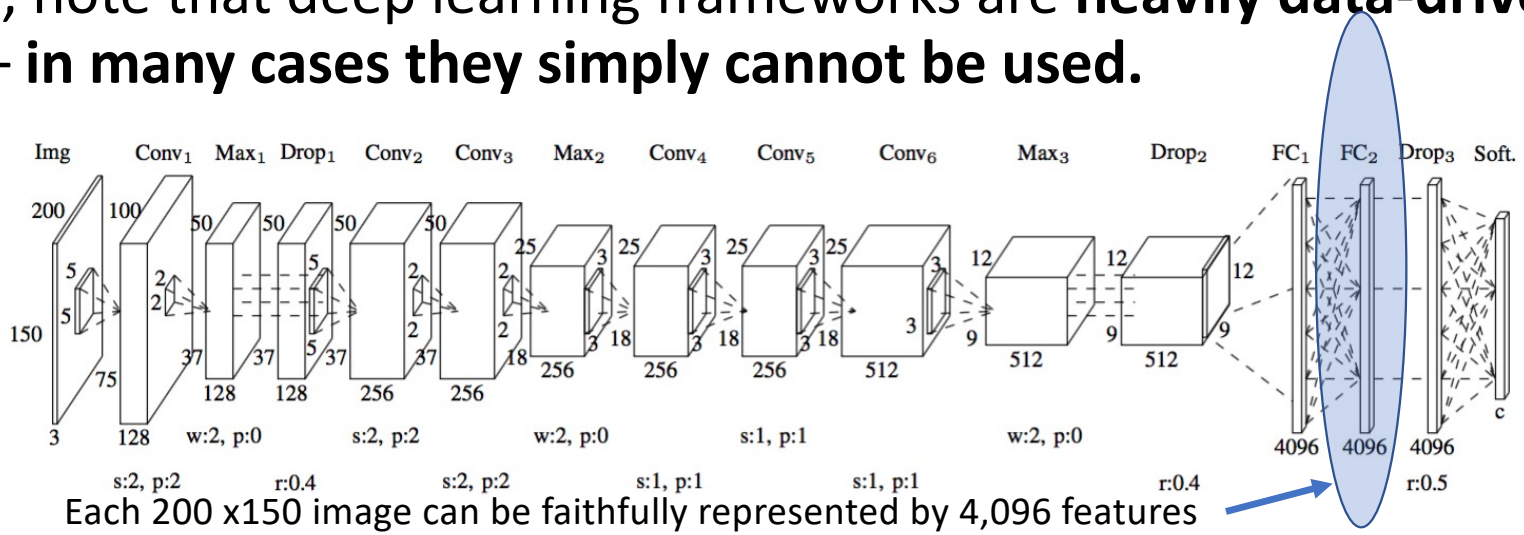
# Feature Representation in Object Detection

- Traditionally, there was a set of handcrafted **Feature Descriptors** that translated the data from the original image space into a different domain:
  - Using techniques such as Local Binary Patterns (LBPs), SIFT, SURF descriptors, Gaussian derivatives, banks of Gabor filters, Haar wavelets,...
    - All these feature extractors return much more information than in the original space, i.e., the dimensionality of the problem substantially increases.
    - In practice, each pixel intensity/color (in $R/R^3$) is replaced by a set of features with n (n>>3) elements



# Image Features: (Preliminary Remark)

- Feature Extraction is clearly the phase of the classical image processing chain able to be more evidently replaced (with advantages) by deep learning frameworks:
  - By using the outputs of some deep layer as feature descriptors
  - By using auto-encoders.
- However, note that deep learning frameworks are **heavily data-driven**, and – hence – **in many cases they simply cannot be used.**




Each 200 x150 image can be faithfully represented by 4,096 features

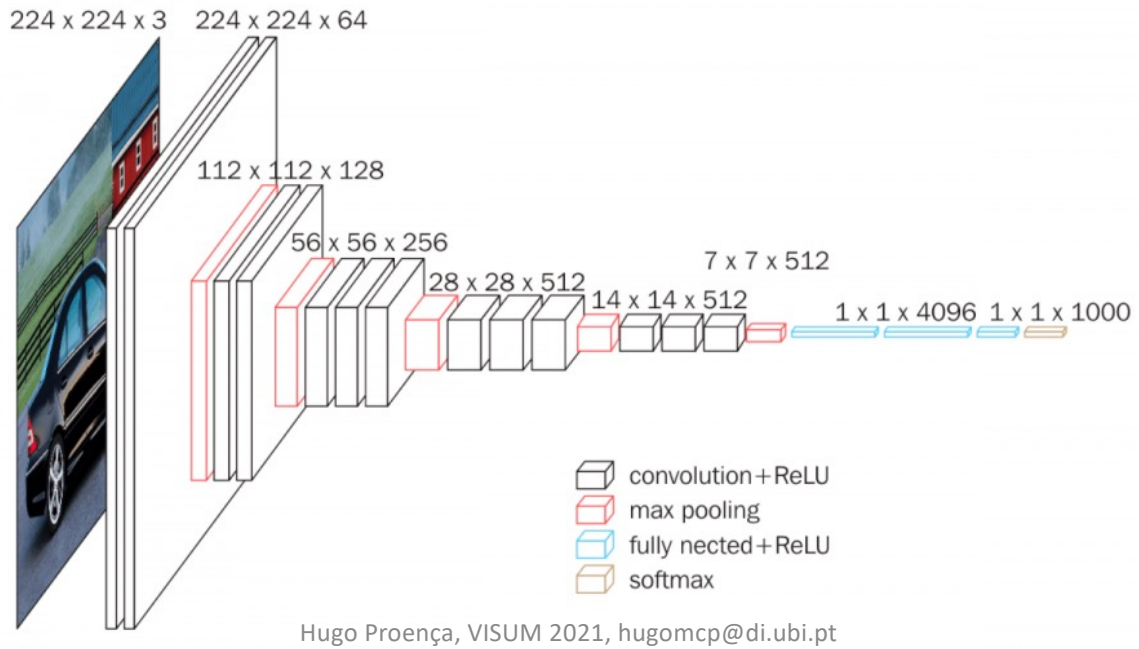
# SSD: Single Shot Multibox Detector

- **SSD. The Single Shot MultiBox Detector** (by W. Liu *et* al.) was released at the end of November 2016 and reached new records in terms of performance and precision for object detection tasks
  - Scored over 74% mAP (mean Average Precision) at 59 frames per second on standard datasets such as PascalVOC and COCO.
- There are three key ideas in this methods:
  - **Single Shot**: this means that the tasks of object localization and classification are done in a single forward pass of the network
  - **MultiBox**: this is the name of a technique for bounding box regression developed (we will briefly cover it shortly)
  - **Detector/Classifier**: In summary, the network is an object detector that also classifies those detected objects

W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C-Y. Fu, A. Berg, SSD: Single Shot MultiBox Detector, ECCV 2016, doi: 10.1007/978-3-319-46448-0_2

# SSD: Single Shot Multibox Detector

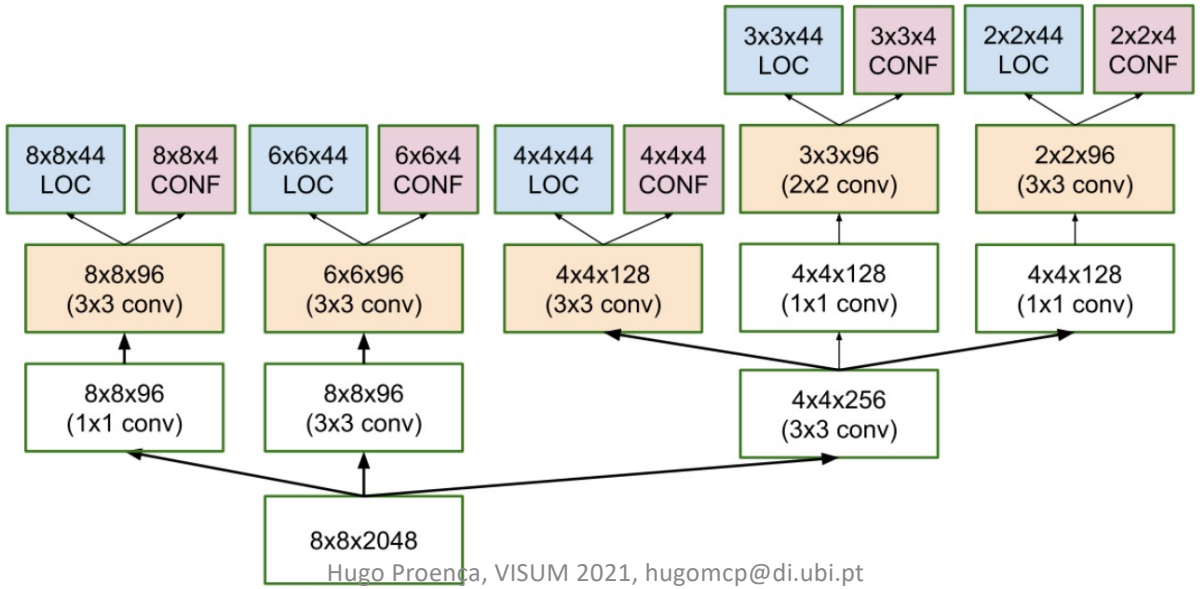
- The initial phase of this detector works under the *VGG-16* paradigm.
  - The VGG-16 is (was) an extremely popular base network in image classification tasks.
  - It has a simple feed-forward architecture.



# SSD: Single Shot Multibox Detector

- In terms of the SSD architecture, the main (single) difference of the SSD detector is to discard the fully connected layers, replaced by a set of auxiliary convolutional layers, that enable to extract features at multiple scales and progressively decrease the size of the input to each subsequent layer.

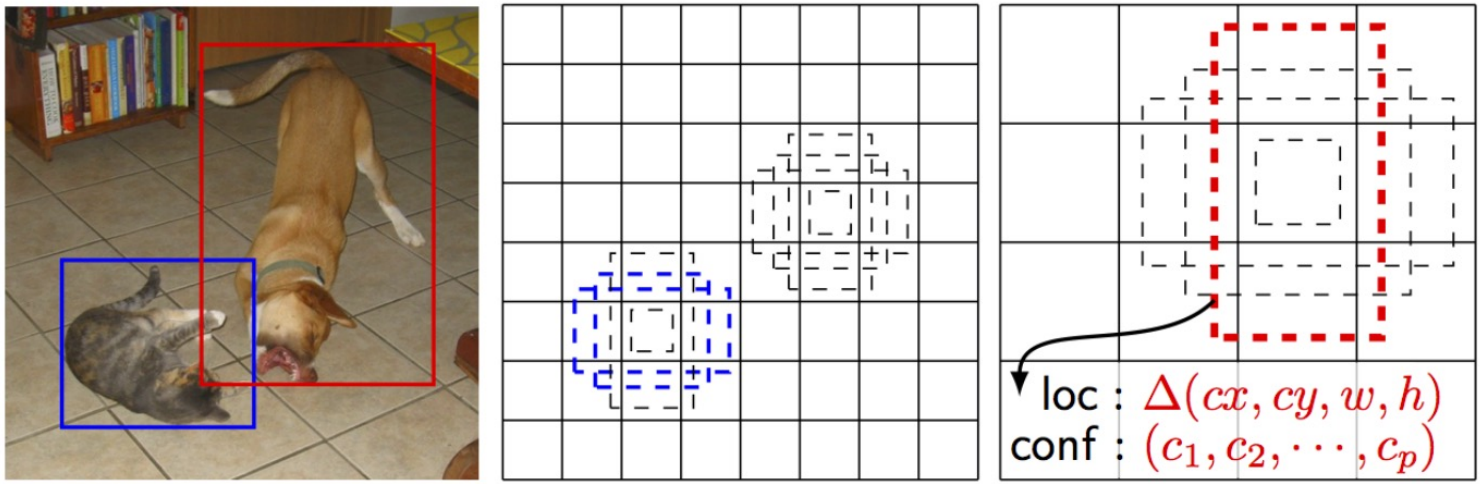
The **Confidence** scores. Set of values (i.e., a vector with as many components as the objects to be detected);




The **Localization** scores, which are 4D vectors that provide the adjustments of the "default boxes" with respect to the actual position of objects.

# SSD: Single Shot Multibox Dectector

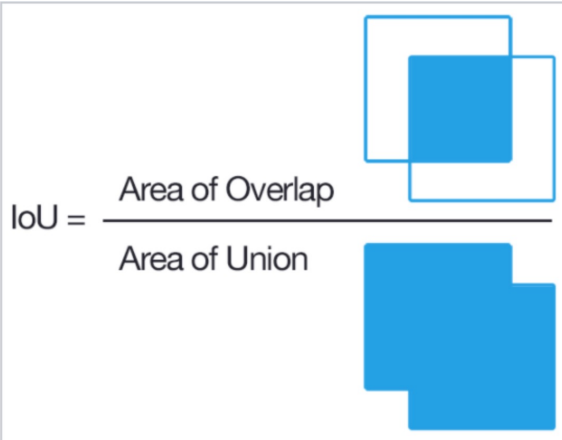
- For each position in a (multiscale) Feature Map, there is a set “default boxes” that provide the initial positions for the objects to be detected.
  - The confidence scores (with one extra term for “Nothing”) give the probability of one object centered at that position
  - The “location” terms provide the adjustements of the actual object position with respect to the default box




(a) Image with GT boxes (b) $8 \times 8$ feature map (c) $4 \times 4$ feature map

# SSD: Single Shot Multibox Dectector

- Essentially, the SSD network can be seen as a multi-object detector, that provides for each cell of the feature map:
  - "N+1" confidence values, being the "+1" considered to denote the "None/Nothing" Object
  - Four regression values that adjust the position/size of the default box to the detected object in terms of translation (cx, cy) and scale (w, h)
- Each default "bounding box" of fixed size, and at a given position is considered to contain one object if the Intersection-of-Union (**IoU**) coefficient is higher than a threshold (tipically 0.5).
  - In that case, the **regression coefficients** are adjusted to maximize the **IoU** value




# Object Detection – Hardness

- The first major hard factor of object detection is its “dual goals”: not only we need to classify parts of the images (ROIs), but also, we should determine the objects positions (i.e., obtaining a rough parameterization of the object).
- Typically, this is done by means of loss functions that *average* two terms: the classification and localization losses:

$$\mathcal{L}(p, u, t^u, v) = \overbrace{\mathcal{L}_c(p, u)}^{classification} + \lambda \overbrace{[u \geq 1]\, \mathcal{L}_l(t^u, v)}^{localization}$$

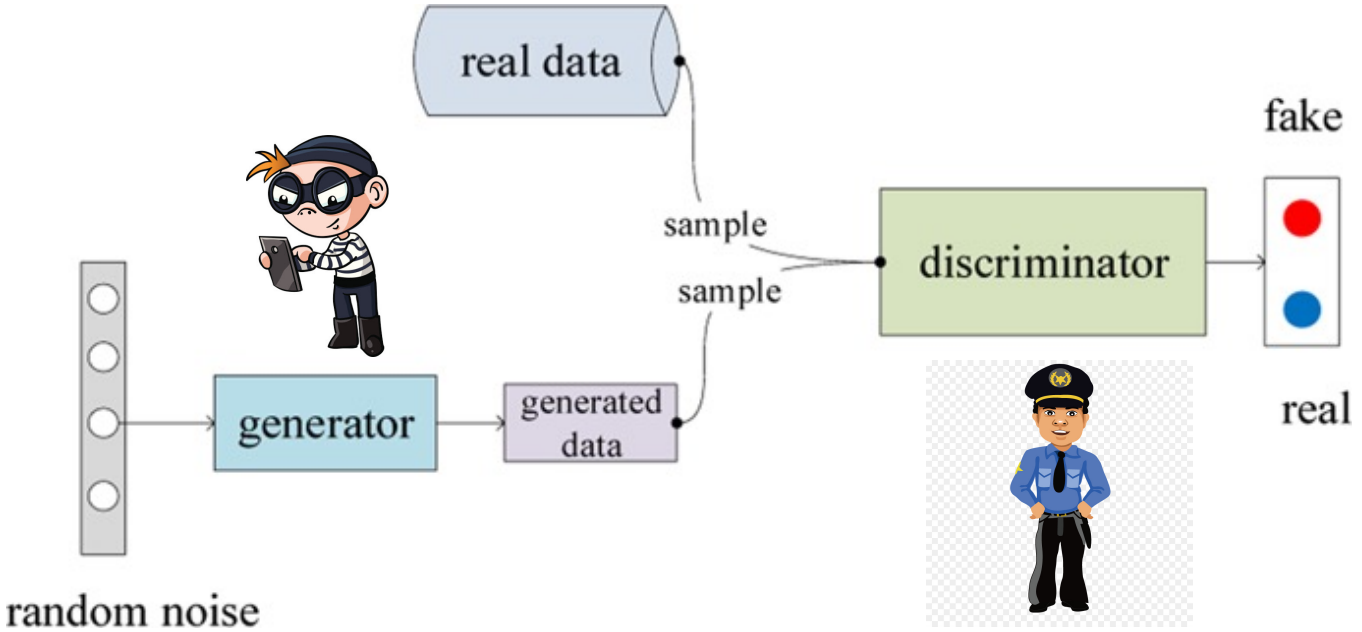
- In some sense, the network should find a balance between how well it can classify and localize objects.
- It is not guaranteed that the overall mean of the loss $\mathcal{L}$ corresponds to simultaneously to the minimum of $\mathcal{L}_c$ and of the $\mathcal{L}_l$ terms.

# Adversarial Learning

- Facebook's AI research director **Yann LeCun** called adversarial training "*the most interesting idea in the last 10 years in Machine Learning*".
- **Generative Adversarial Networks** (GANs) are architectures that use two neural networks, competing one against the other (thus the "adversarial" term) in order to generate new, synthetic instances of data that can pass for real data.
  - GANs were introduced in a paper by Ian Goodfellow and other researchers at the University of Montreal, including Yoshua Bengio, in 2014.
- GANs' potential for both good and evil is huge, because they learn to mimic **any distribution of data**.
- GANs can be taught to create worlds eerily like our own in almost any domain: images, music, speech, prose…

# Generative Adversarial Networks

- The basic idea in GANs is to have one network (**Generator**) trying to fool the other one, while the later (**Discriminator**) tries not to be fooled.
- This can be seen as a **Police Officer**←→ **Thief** game that, according to **Nash Game Theory**, typically converges into an equilibrium state.



# Generative Adversarial Networks

- The **Discriminator** network is a typical binary classification CNN, that learns to distinguish between fake and real data.
- The **Generator** network receives one latent code (randomly generated, i.e., white noise) and produces one instance.
- The overall cost function is given by a two-player min-max game:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

- That can be decomposed into:

Discriminator

$$\max_D V(D) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

*"recognize genuine"*

*"recognize fakes"*

Generator

$$\min_G V(G) = \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

*"fool" D*

# Generative Adversarial Networks

GANs are trained in an iterative way:

1. Generate a set of Fake Data **F**
2. Set Discriminator.trainable = TRUE
3. Train the Discriminator (with Real data **R** **(labelled 0)** and Fake Data **F** **(labelled 1)**)
   **//Goal: learn to distinguish Real from Fake data**
4. Set Discriminator.trainable =FALSE
5. Train the GAN (with Fake Data F (**labelled 0**)) **//Goal: learn to fool D**

**Key Step!!**

5. Move to Step 1.

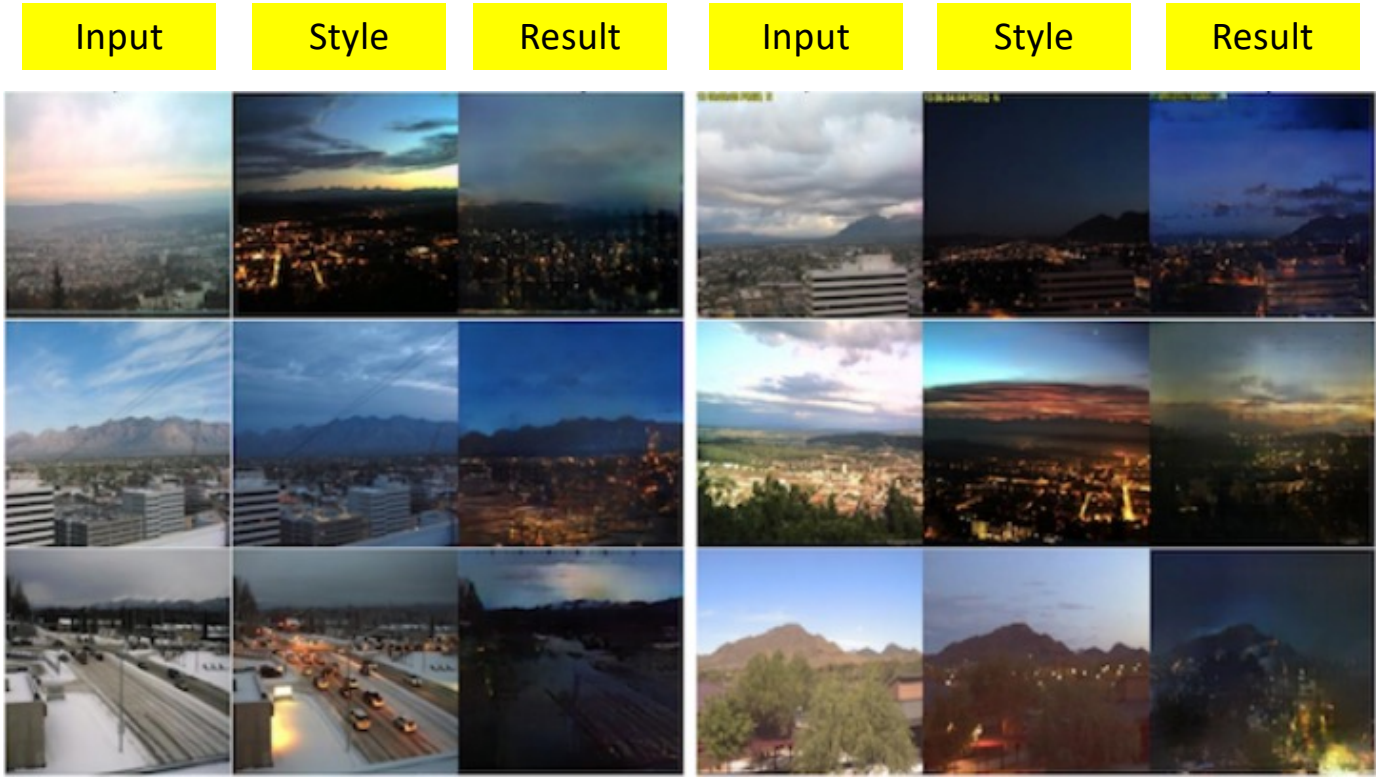
# GANs Applications

- Plausible realistic photographs of human faces (obtained by Style-Gan-2)

**These persons don't exist!!**

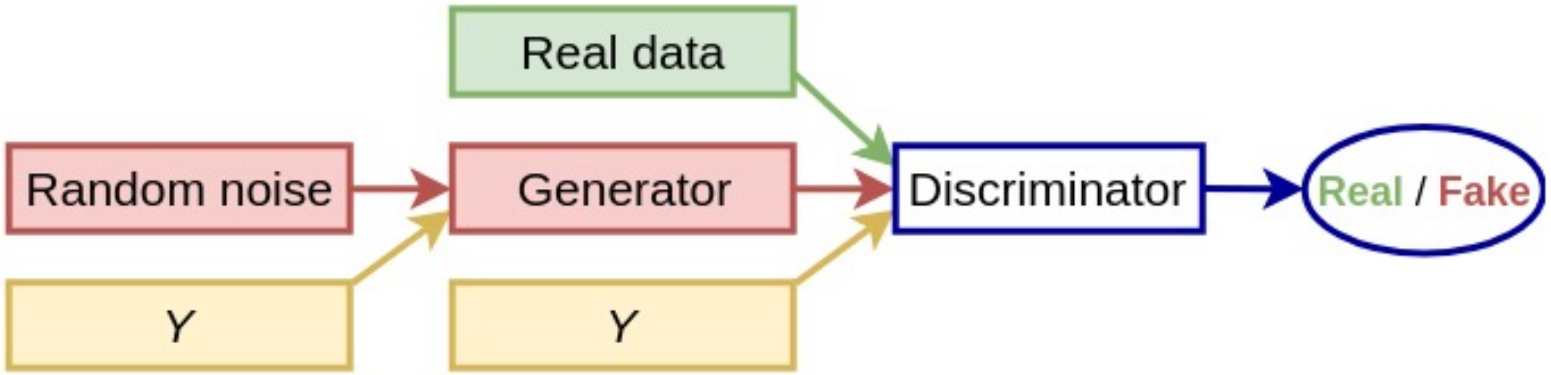
# GANs Applications

- Image-to-Image Translation:



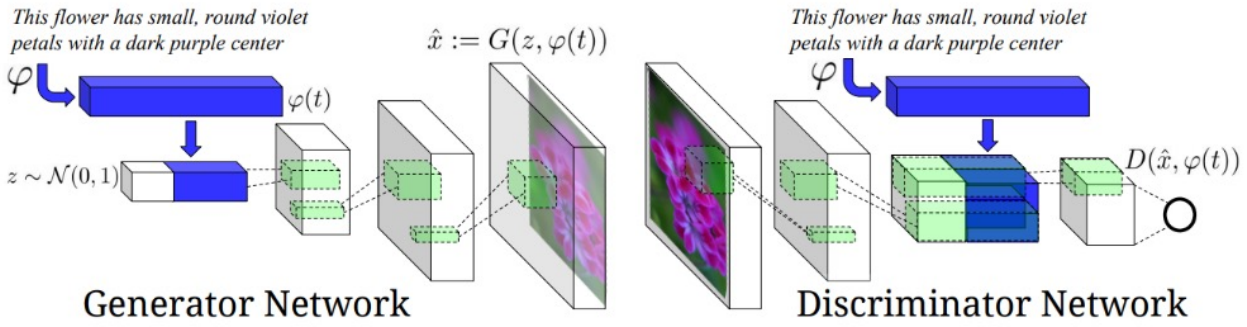
# Conditional GANs

- Despite the remarkable effectiveness of GANs in generating synthetic (artificial) instances of one specific phenomenon, they provide a limited control over the specific features of the output.
  - Recall that the input is a random noise vector.
- Class-Conditional GANs (**cGANs**) introduce the label information to the learning architecture, enabling to produce instances of a specific class.
  - The Discriminator reports “1” only for genuine images with correct labels, and “0” for all other cases (genuine images with bad labels, and fake images with any label).



# Conditional GANs (Applications)

- "*Text-To-Image Synthesis*": This is the problem of asking to a network, to generate images with specific features:




- "Style Transfer": Transferring style between different kinds of objects:

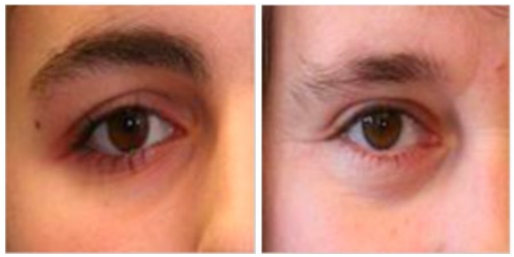
# Image Understanding vs. Interpretability/Explainability

- In the computer vision context, the ability to provide the reasoning behind a decision has been at the core of substantial research efforts.
- Explanations serve not only to increase the trust and understanding amongst the users of a system, but also to augment the system's overall accountability and transparency.
- Typically, a recognition problem involves a set of unique and non-transferable features that can unmistakably be used to recognize an object.

**Problem: Does this "object" regard a *genuine* comparison?**

Pair of periocular regions

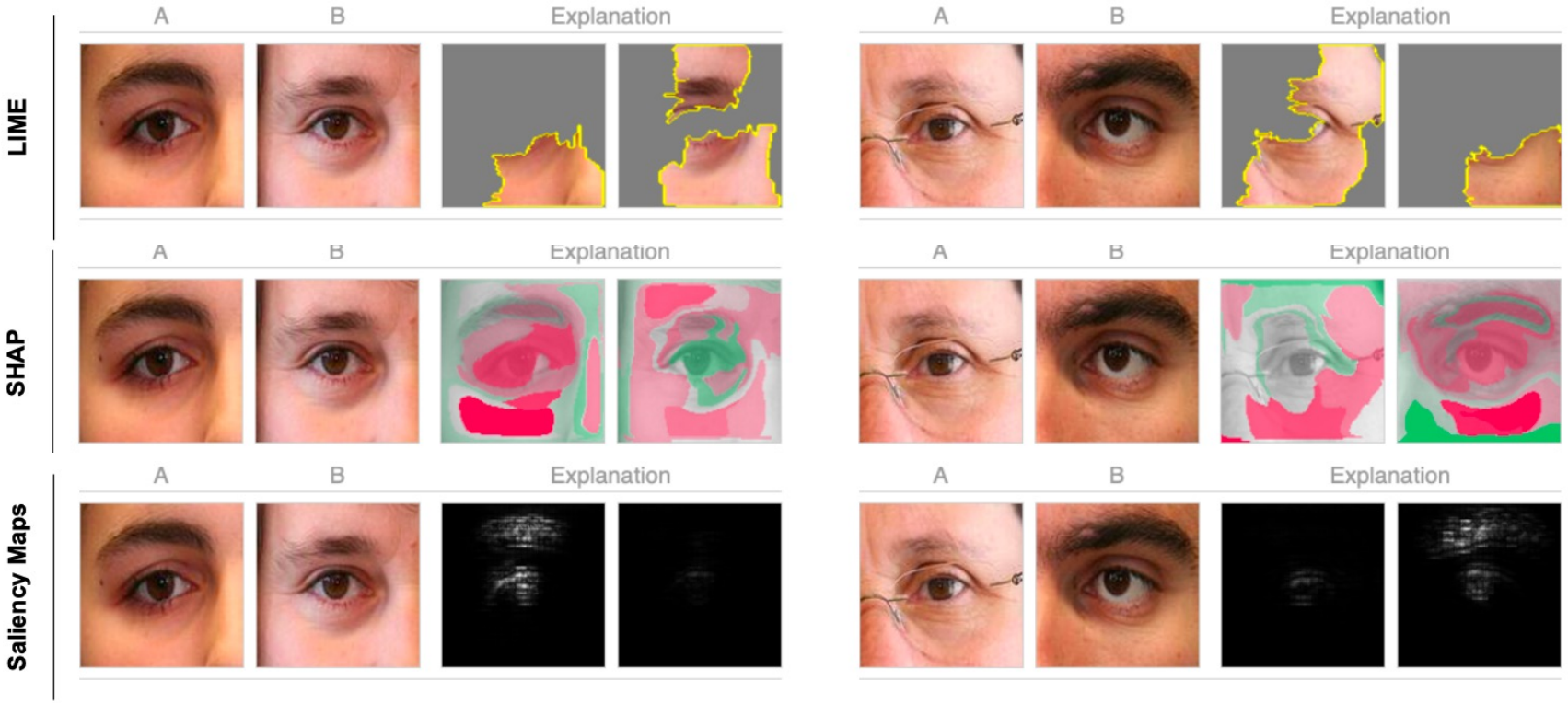
Are from the same subject



Deep learning-based black-box solutions will be highly effective in this task, upon the existence of enough learning data. **However, they do not give any cue (hint) to understand the problem/solution.**

# Image Understanding vs. Interpretability/Explainability

- Traditional ML interpretability and explainability techniques do not provide intuitive (i.e., fully understandable) solutions to the problem:



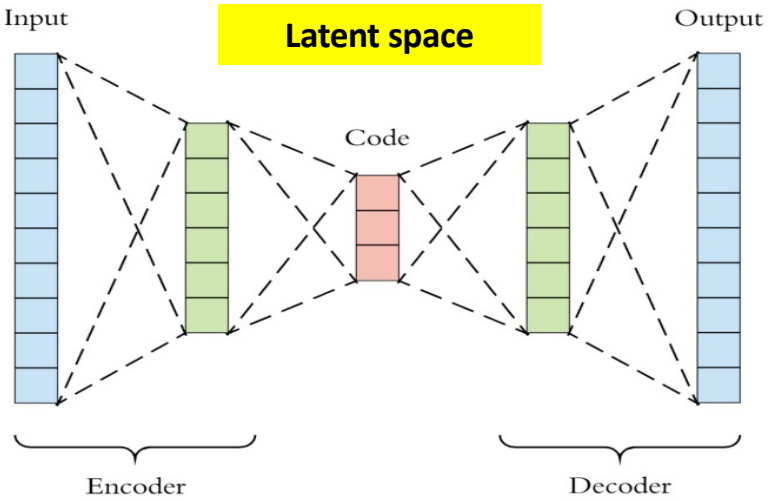
# Image Understanding vs. Interpretability/Explainability

- We developed a deep learning-framework that provides more intuitive “visual explanations” that enable to understand why two objects are/are not of the same class.
  - As proof-of-concept, we use the **biometric recognition** problem, using the **periocular region**.
- We start by using a SOTA semantic segmentation model, that discriminates locally between all the biological components in the periocular area.
  - Unique fully supervised part of the framework

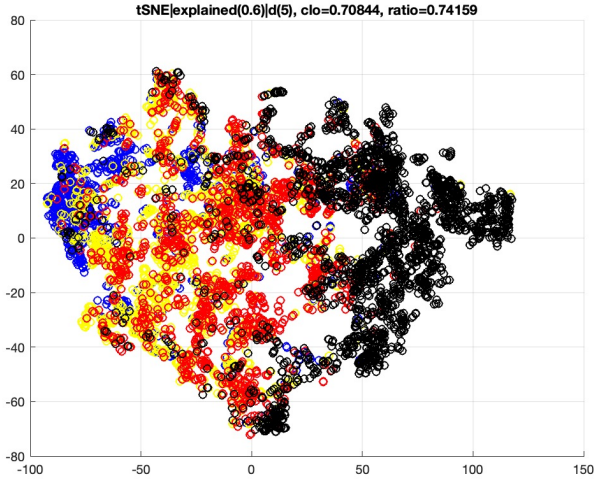
# Image Understanding vs. Interpretability/Explainability

- Next, we crop every biologically component in the periocular region, and normalize its size, which enables to obtain invariance to scale and translation.
- For example, considering the iris region:

- We feed an auto-encoder that is responsible for obtaining a compact representation of this data:



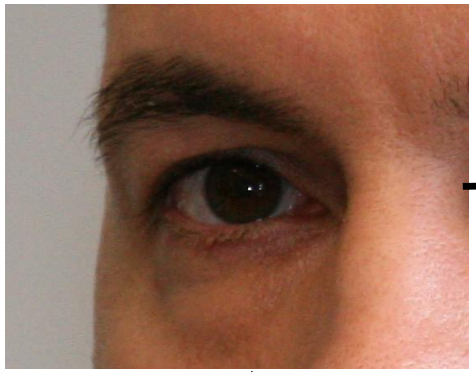

…which are finally mapped into a low-dimensional manifold, which coordinates will be used subsequently




**Iris color "t-SNE" visualization**

# Image Understanding vs. Interpretability/Explainability

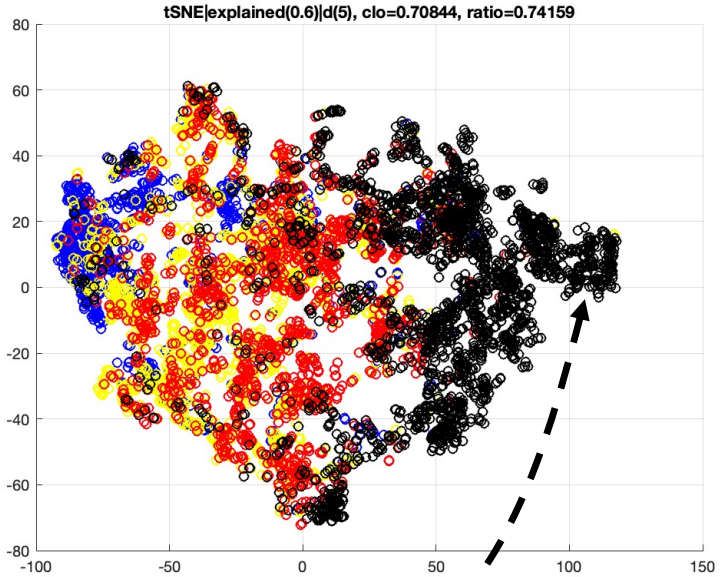
- Next, we create a regression network that receives one periocular sample and returns the corresponding 2D coordinates of the t-SNE low-dimensional mapping
  - **(Un)supervised Learning** , attenuates the known “out-of-sample” problem of manifold mapping



CNN

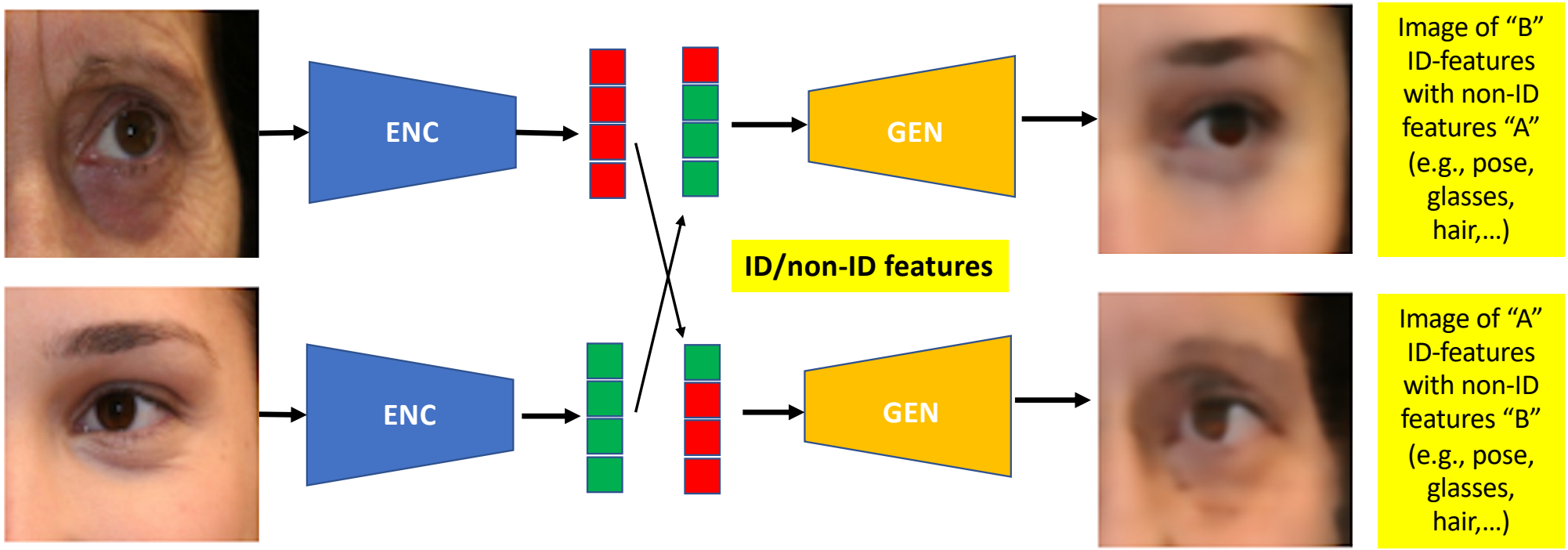
(102.6, 0.21)

…



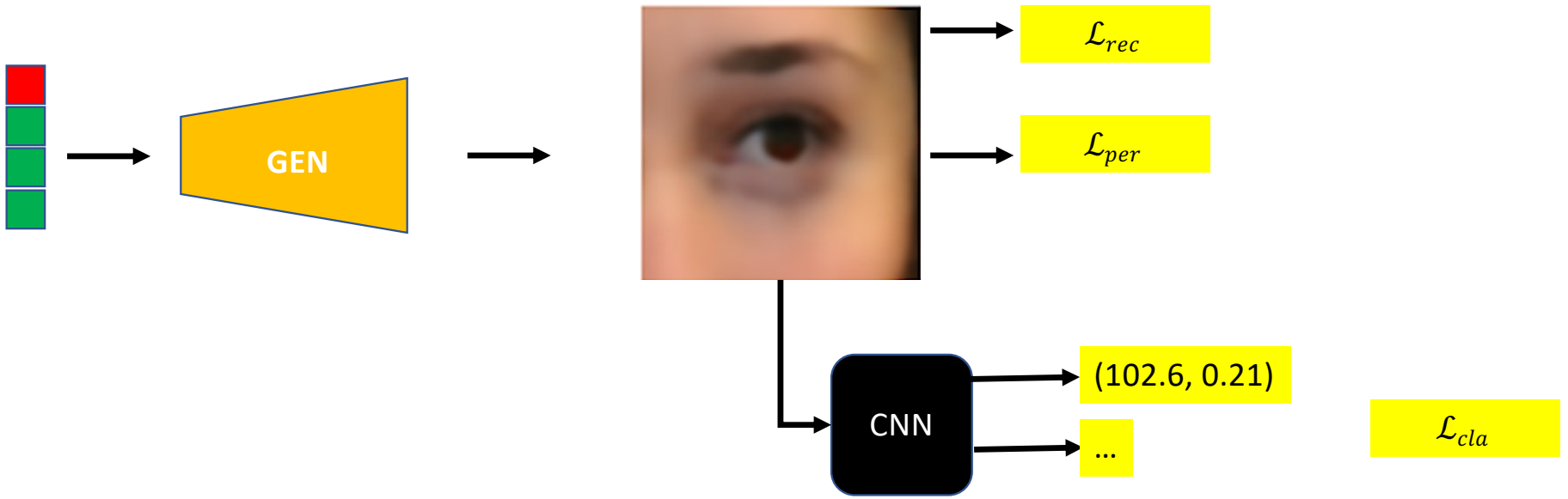
# Image Understanding vs. Interpretability/Explainability

- Finally, the main learning framework is composed of the previously learned regressor (**$CNN_R$**), plus one encoder (**ENC**) and one generator (**GEN**) network that form a **Siamese Network.**
- Receiving a pair of images to be matched, we obtain their latent codes (using the encoder) and swap parts of their latent codes.
- Next, the partially swapped latent codes enter in the Generator network, that is responsible to reconstruct the images, such that their features comply the values from the image they originally taken



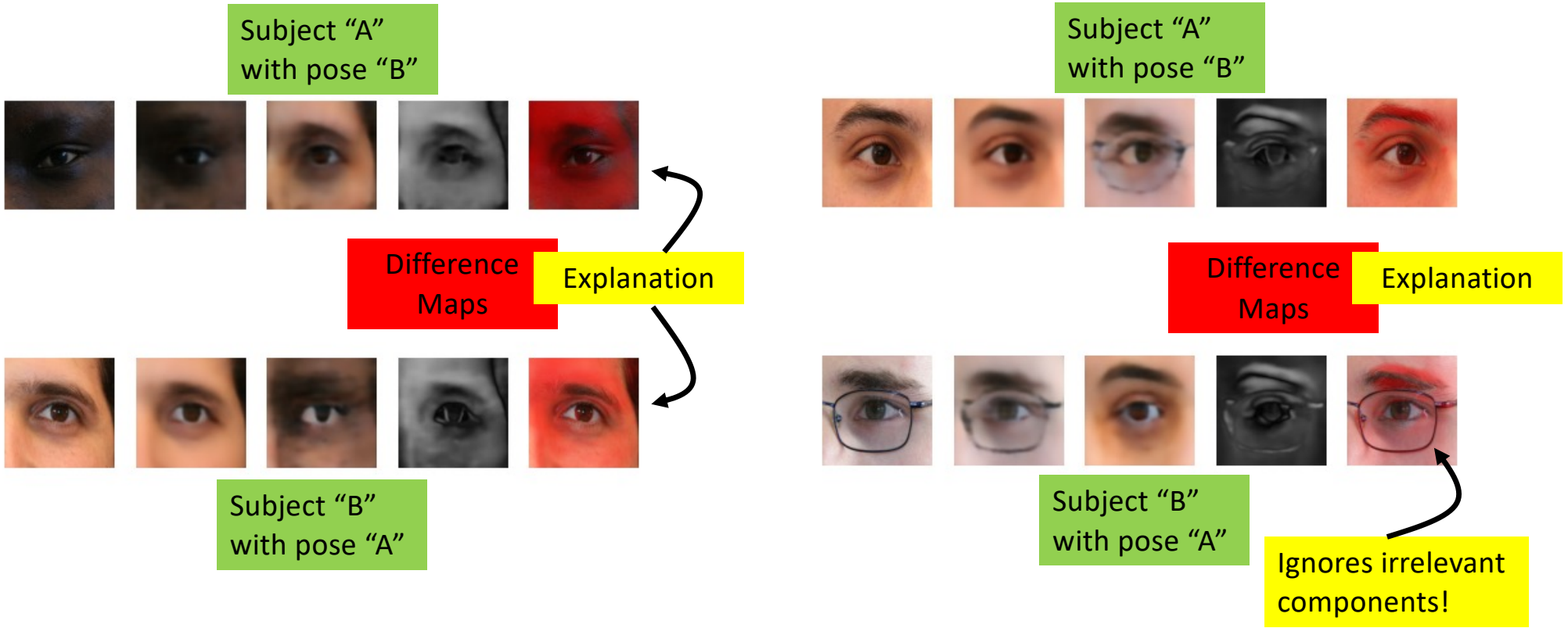
# Image Understanding vs. Interpretability/Explainability

- The Generator network not only uses the ”Pixelwise Reconstruction Loss”, but also the “Perceptual Loss”, together with the responses provided by the CNN regression network



# Image Understanding vs. Interpretability/Explainability

- Examples:



# Image Understanding: Conclusions

- There is still a long road to be paved, to obtain reliable “Image Understanding”
  - This is particularly evident for “cross-domain” scenarios, where the test data has evidently different features of the learning set
  - The “cross-domain” ability of humans is far from being achieved by automata
- There are substantial research efforts putted in the “Weak Supervised” or even “Unsupervised” learning paradigms
  - At this moment, only a tiny fraction of the available data is tagged (annotated by humans)
  - Hence, only that tiny proportion of data is able to be used by supervised learning frameworks

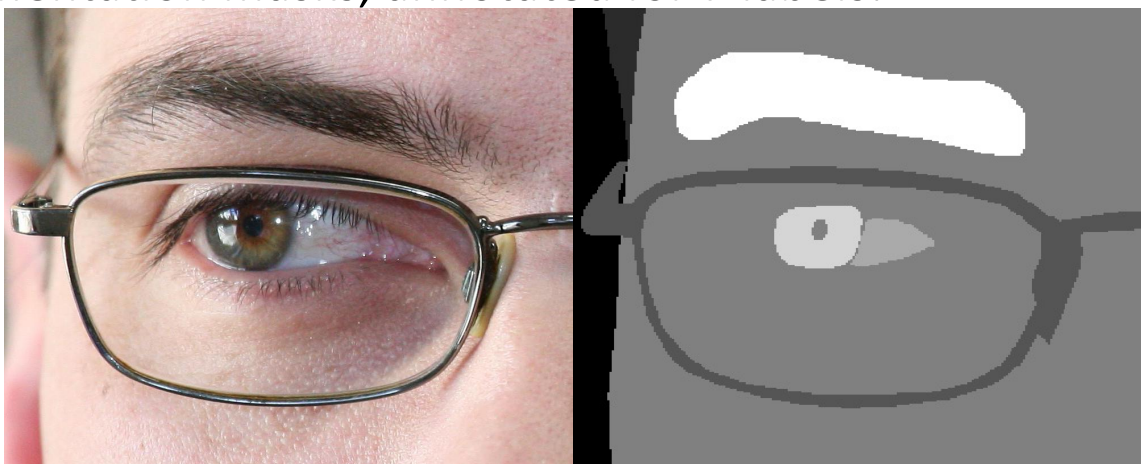
# Image Understanding: Conclusions

Thank you! 😀

Questions?

# Image Understanding: Hands-on

- **How well can we *understand* the ID of a subject, just by considering the interpretable components in his periocular region?**
- Materials:
  - 1 Dataset of periocular images (6,448 images);
  - 1 Semantic annotations file (“.csv” format);
  - 1 set of semantic segmentation masks, annotated for 7 labels:
    - Background;
    - Hair;
    - Glasses;
    - Skin;
    - Iris;
    - Sclera;
    - Eyebrows.



# Image Understanding: Hands-on

- **How well can we *understand* the ID of a subject, just by considering the interpretable components in his periocular region?**
- **Work plan:**

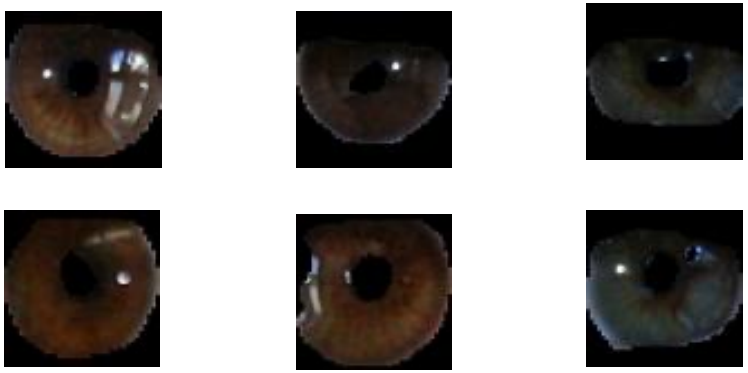
  1. Create piecewise images for each ID-relevant component:
     - Iris
     - Sclera
     - Eyebrows
     - Skin
  2. Turn the piecewise representations of each component invariant (as much as possible) to scale, lighting and translation.
     - For each component, all images should have the same size.
  3. For each component, load all piecewise representations into a big **Matrix.**

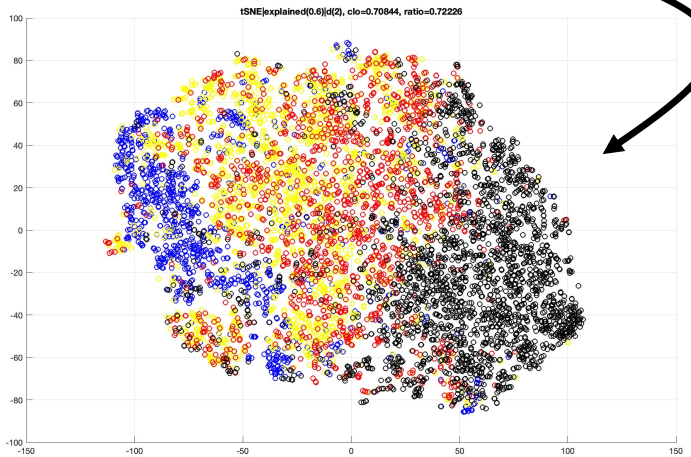
# Image Understanding: Hands-on

- **How well can we *understand* the ID of a subject, just by considering the interpretable components in his periocular region?**
- **Work plan:**

  4. For each component, (e.g., using t-SNE) project the piecewise representations into a low dimensional space (2,…"k" dimensions, your choice).

  Example of *iris piecewise representations* in the original and in the low-dimensional spaces.

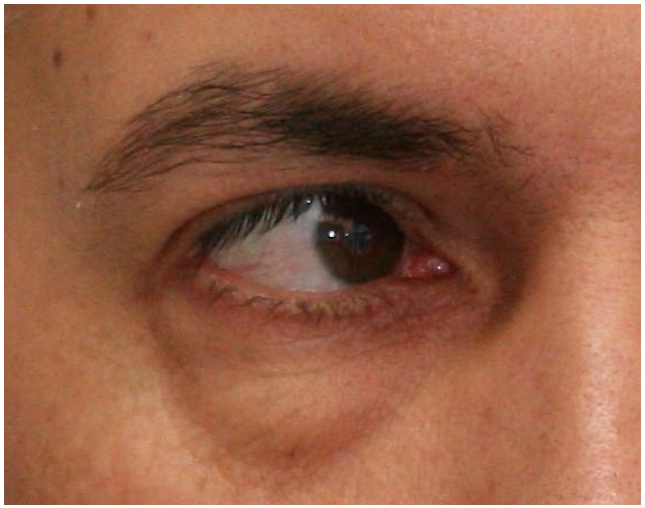


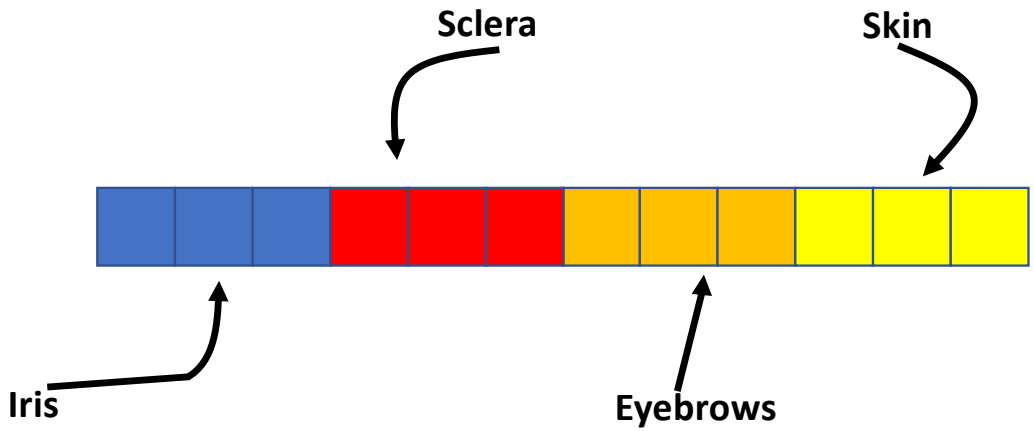


# Image Understanding: Hands-on

- **How well can we understand the ID of a subject, just by considering the interpretable components in his periocular region?**
- **Work plan:**

  5. For each image, create a feature vector that is composed of the concatenation of all low-dimensional piecewise representations:

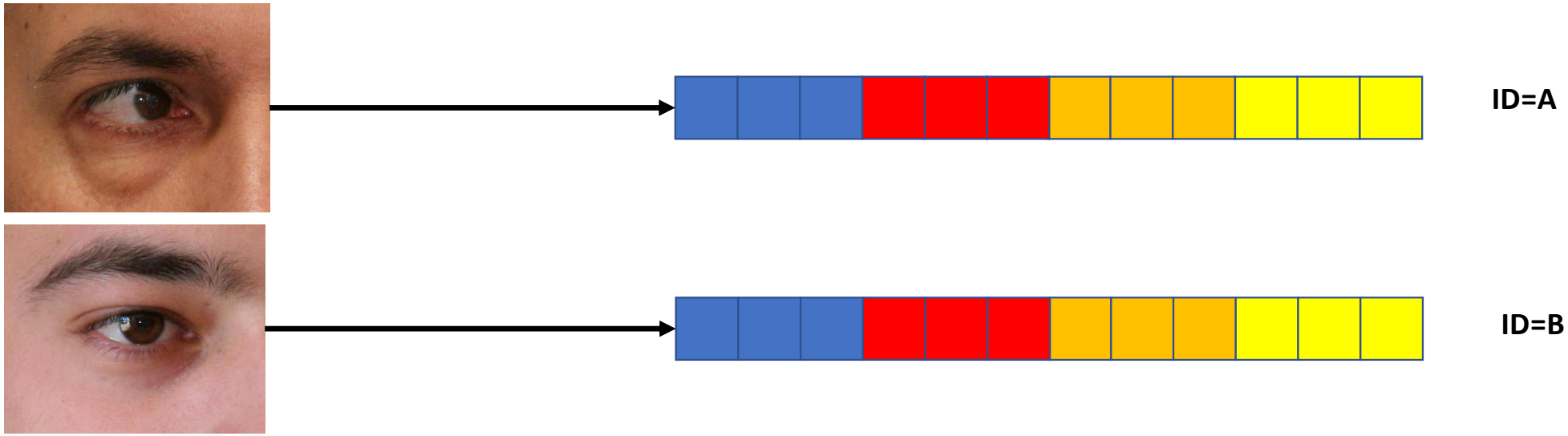




# Image Understanding: Hands-on

- **How well can we understand the ID of a subject, just by considering the interpretable components in his periocular region?**
- **Work plan:**

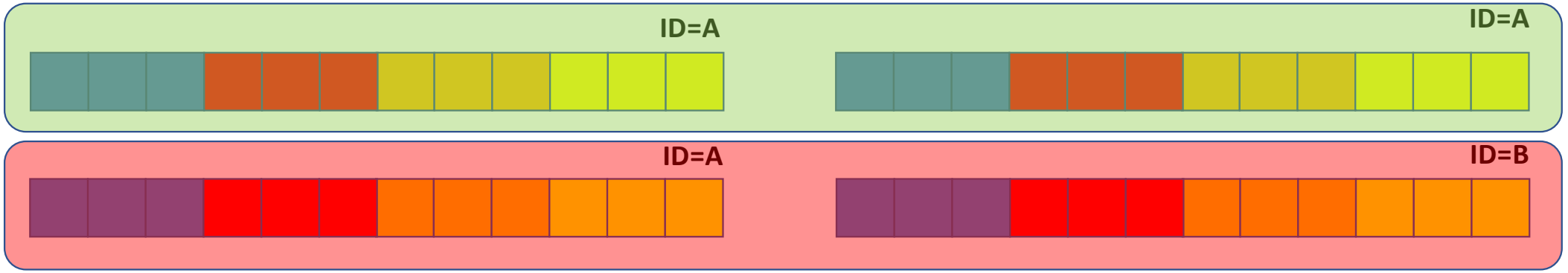
  6. Load all feature vectors into a big **Matrix.** Use **the ID information** (given in the ".csv" file) as label for each image.



# Image Understanding: Hands-on

- **How well can we understand the ID of a subject, just by considering the interpretable components in his periocular region?**
- **Work plan:**

  7. By concatenating feature vectors, create a large set of “Genuine”/”Impostor” comparisons, yielding a binary classification problem (**0**/**1** labels, for “**Impostor**”/”**Genuine**” pairs).




  8. Check the classification performance that can be attained in this set (e.g., NN, SVM,…).