COMPUTER VISION
MEI/1

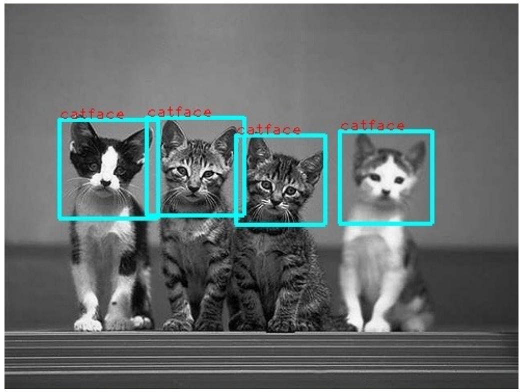
University of Beira Interior, Department of Informatics

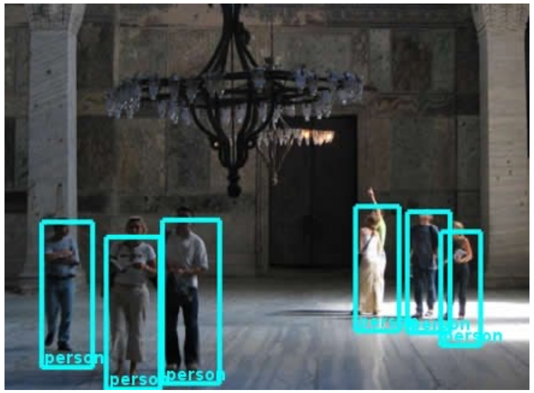
Hugo Pedro Proença

hugomcp@di.ubi.pt, **2023/24**

Object Detection

- With exception to the preprocessing phase, it is usually the earliest phase of computer vision systems.
- Here, the goal is to **roughly detect the regions-of-interest** (**ROI**) that potentially 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., (x_1, y_1, x_2, y_2) for a rectangular patch
  - Can be regarded as an image (patch) classification task





Object Detection Challenges

- There are **multiple varying factors** in the acquired data:
  - **Lighting** (shadows);
  - **Shape** (e.g., scale, translation, rotation);
  - **Pose** (perspective);
  - Background **clutter**;
  - Object **deformations**;
  - **Occlusions**.;
  - **Resolution** (blur);
  - **Perspective.**
- The object detector should handle appropriately these variations.
  - A flexible and robust detector is desired.
    - Ability to handle overlapping instances (usually done in post-processing steps).

Object Detection: Typical Steps

- Image Preprocessing
  - Data Normalization
  - Local rectification
- Overcomplete feature set representation
  - Complete basis do not have linear dependence between basis elements and have the same number of elements as the original space.
  - This is not usually seen in feature representation for image detection purposes, as it is **extremely hard to find a complete basis**.
- Machine-Learning based techniques to build appropriate models
- Postprocessing to fuse multiple detections.
  - Ignore partially overlapping detections.

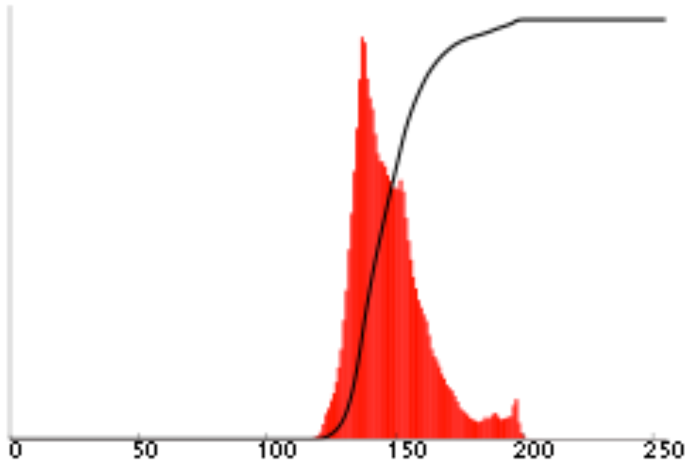
Preprocessing

- It is often though as complementary, but – in practice terms - it has notorious impact in performance.
- The **key** in preprocessing is **to compensate** as much as possible for environmental variations in the acquired data.
- **Examples** of techniques used in this phase:
  - Histogram stretch, equalization;
  - Homomorphic filtering;
  - Center-surround filtering.

Preprocessing: Histogram Equalization

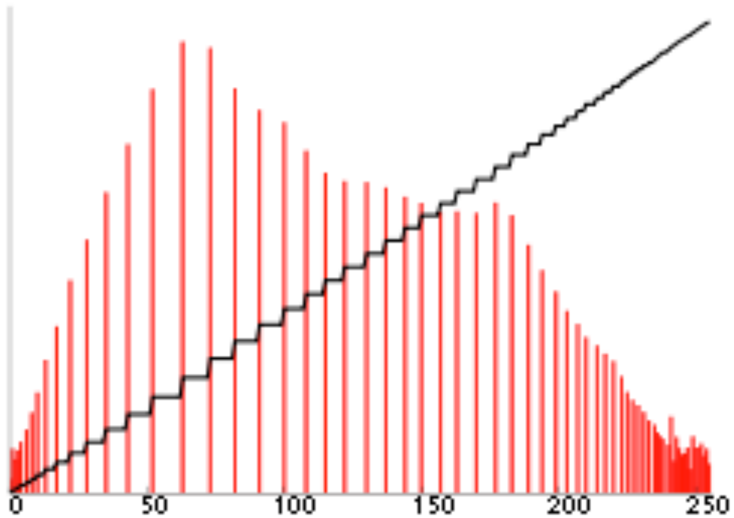
- Let f(x,y) be a grayscale image and n_i be the probability of occuring a pixel with intensity "i":
  - $p(f(x,y)=i)=n_i$
- Let cdf(i) be the cumulative distribution function corresponding to "p":
  - cdf(i)= $$\sum_{j=0}^{i} p(f(x,y)=i)$$

Histogram Equalization: Example



Histogram Equalization

- The goal is to find an intensity transform T that transforms the cdf into a straight line.
  - This can be one by T(i)=cdf(i)
    - Transform the i^{th} intensity into the corresponding cdf value



Homomorphic Filtering

- In this algorithm, the **low frequency illumination** is separated from the **high frequency reflectance**. The key steps are as follow:
  1. Take the logarithm of the input data
     1. Usually a signal is expressed as an adition of low and high frequency components. However, in the illumination/reflectance problem, the low frequency information was observed to be multiplied, instead of added to the high frequency reflectance. In order to use the high-pass filtering schemme, the logarithm operation is needed to convert multiplication to addition.
  2. Obtain the 2D FFT of the resulting data
  3. Suppress low frequency in Fourier domain (high-pass pass filtering)
  4. Take the inverse FFT
  5. Take the exponential

Adaboost Object Detector

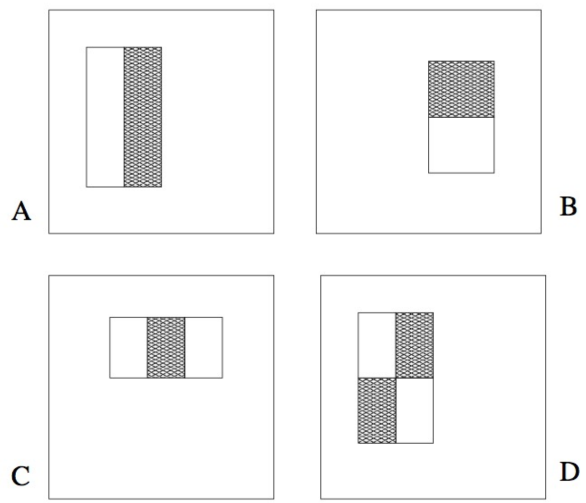
- Originally proposed by Viola and Jones, it is one of the most popular object detection algorithm:
  - *Paul A. Viola, Michael J. Jones: Robust Real-Time Face Detection. International Journal of Computer Vision 57(2): 137-154 (2004).*
  - It requires a set of “binary” training data
    - Labelled positive (1) and negative (0) samples.

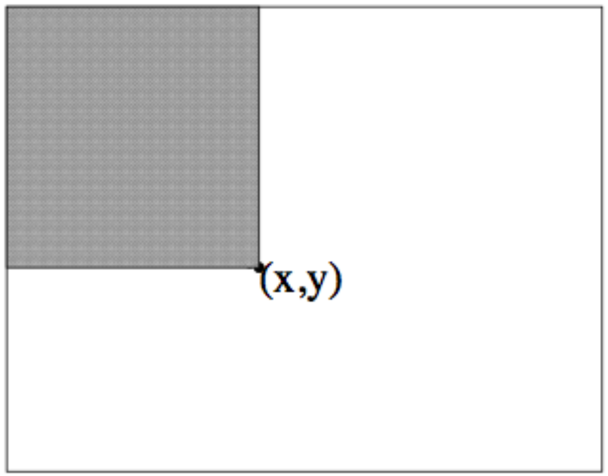
  - Builds a strong detector from a set of very simple (weak) detectors.
  - It exploits the correlation between weak detectors.
- This yields a detector **able to work in real-time**.

Adaboost Object Detector

- Features are reminiscent of Haar-basis functions:
  - The value of each feature (A,B,C,D) 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)"

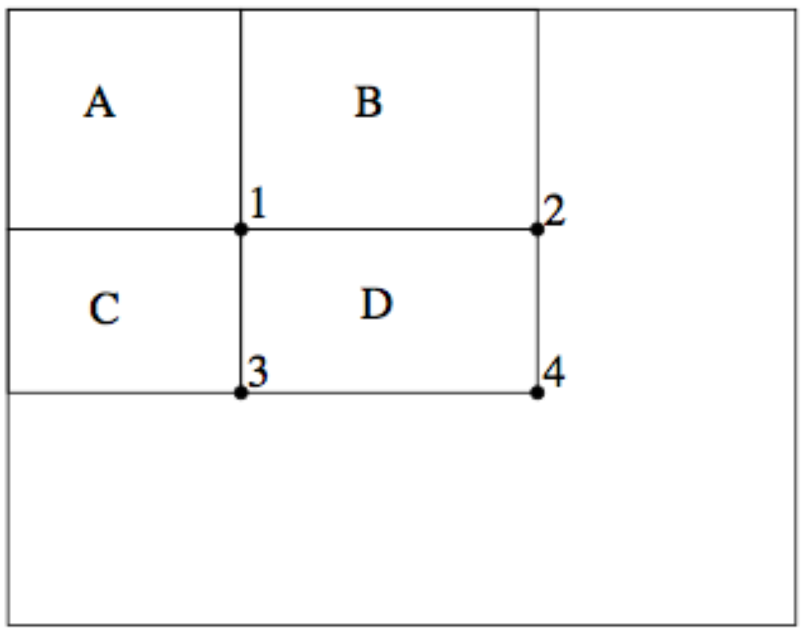
$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y'),$$





Adaboost Object Detector

- **Integral Image:** according to its definition, it is possible to obtain the sum of any rectangular image region, just by summing (subtracting) four different values:
  - D=4+1+(2-3)

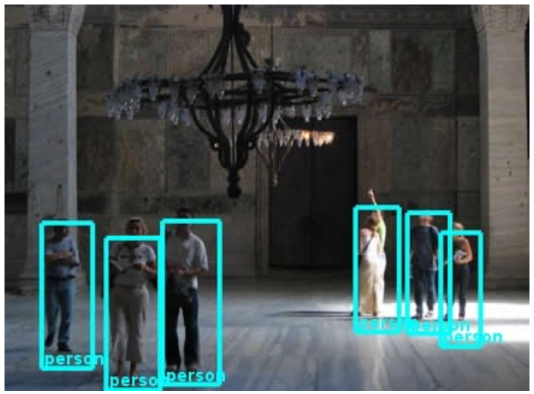


Adaboost Object Detector

- Weak Classifiers.
  - Each weak classifier h(x,y,p,t) consists on the analysis of one of the initial types of features, centered at a given position (x,y), where the most discriminating power between negative and positive samples occur at threshold “t” and comparison sign “p”
    - $H(x,y,p,t) = p\, f(x,y) < p\, t$
- For each possible featurea weak classifier is built.
- A set of weights are initialized, according to the total of negative and positive samples
- At each iteration choose the best weak classifier (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 previleged
- The final strong classifier is a function o each weak classifier and the corresponding weight.

Summary: Object Detection

- As previously seen, the **object detection** phase aims at detecting a region-of-interest (ROI) that probably contains the object to be handled.
  - Usually such ROIs consist in two pixel coordinates: upper left and bottom right corners. (x_i,y_i), (x_f,y_f)



Feature Representation in Object Detection

- Tradittionaly, there weas a set of Feature Descriptors that translated the data in the original image inyo a diferente domain:
  - Using techniques such as Local Binary Patterns (LBPs), SIFT, SURF descriptors, Gaussian derivatives, Gabor filters, Haar wavelets,...
    - All these feature extractors return typically much higher dimension than the original space
    - In practice, each pixel intensity/color (in R/R^3) is replaced by a set of features with n (n>>3) elements

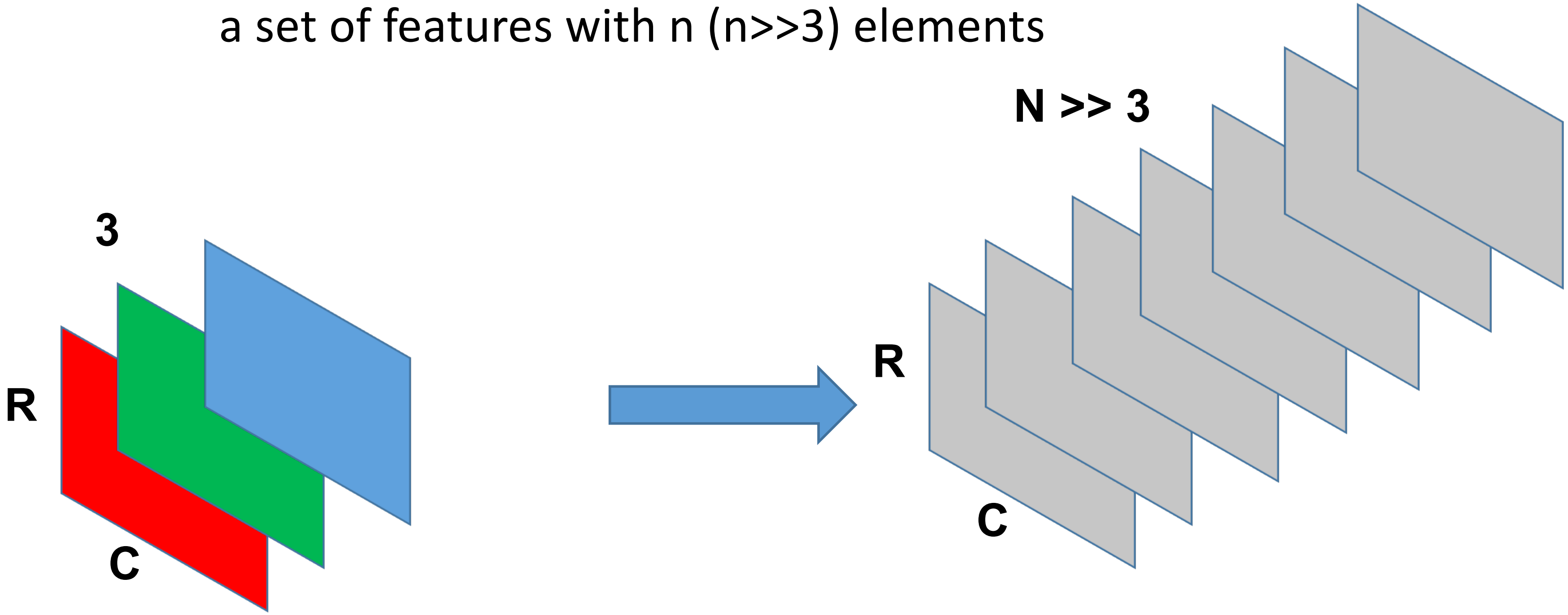


Image Features: (Preliminary Remark)

- 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 several cases they simply cannot be used.**

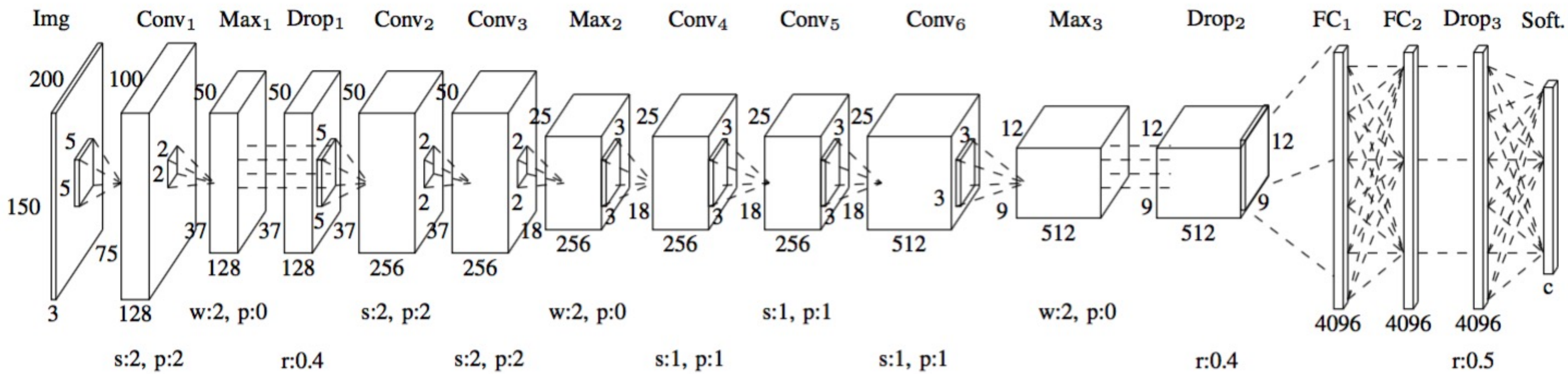
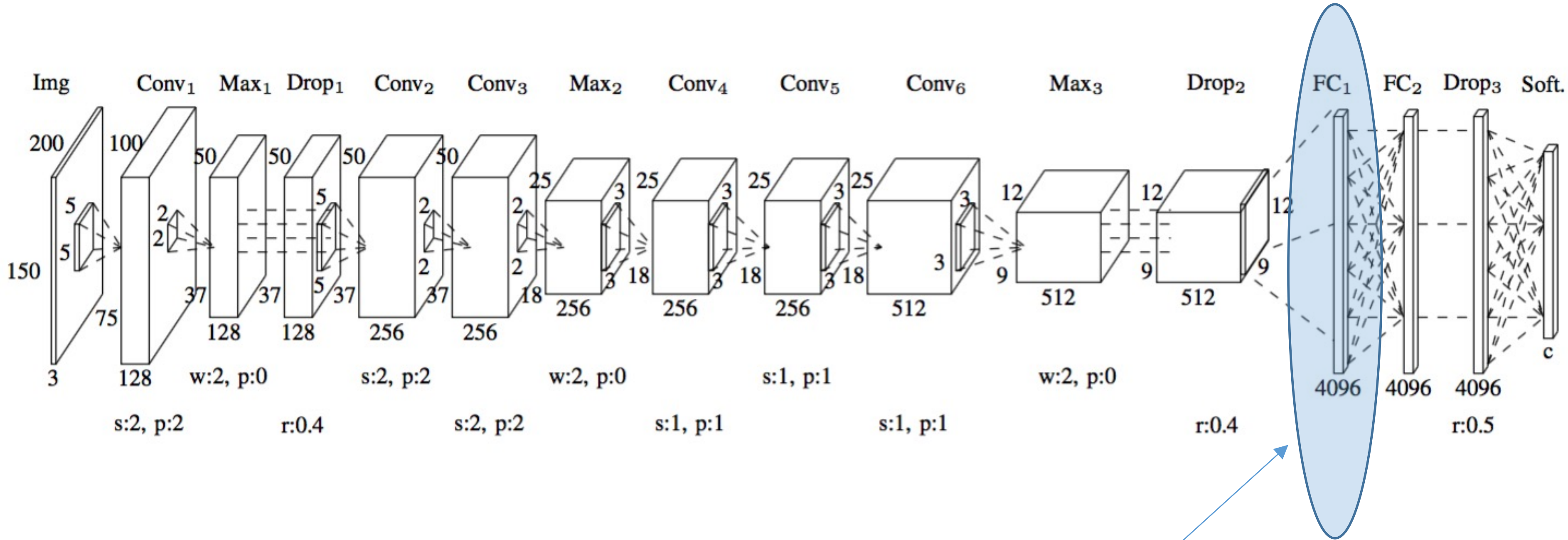


Image Features: (Preliminary Remark)

- ❑**Main Option :** using the outputs of some deep layers as image features
  - ❑Such feature will enter subsequently in another classifier



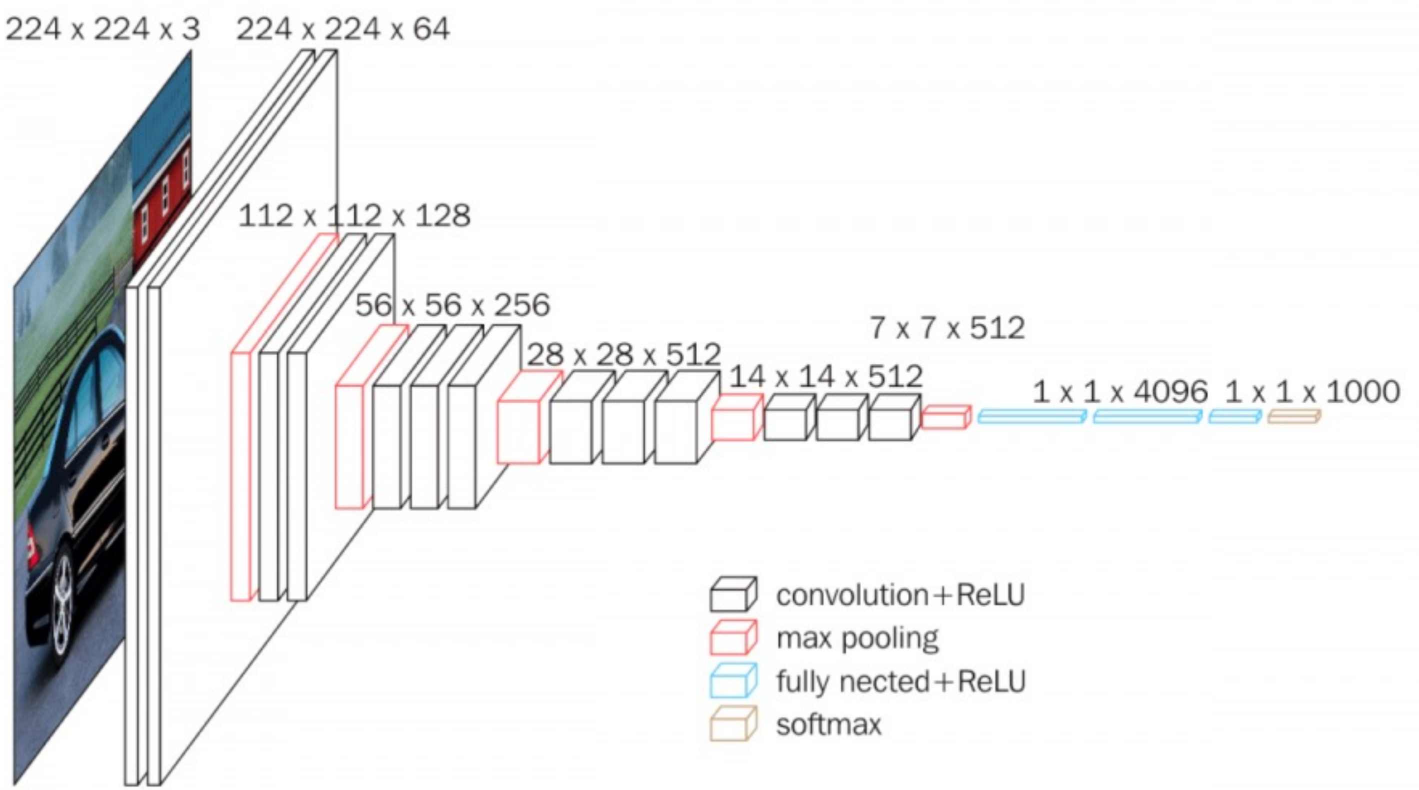
Each 200 x150 image can be faithfully represented by 4,096 features

SSD: Single Shot Multibox Dectector

- **SSD. The Single Shot MultiBox Detector** (by C. Szegedy 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 main 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 by Szegedy et al. (we will briefly cover it shortly)
  - **Detector/Classifier**: The network is an object detector that also classifies those detected objects

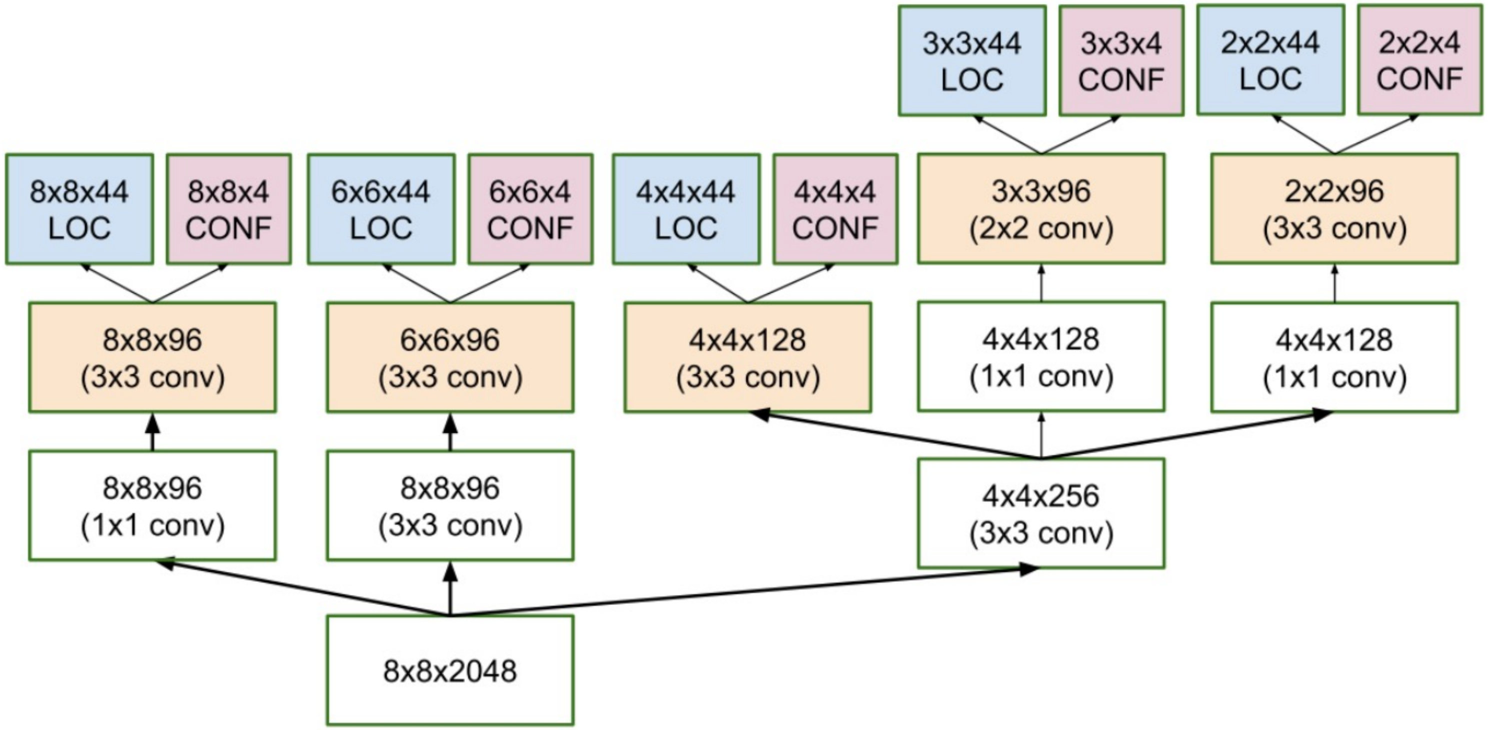
SSD: Single Shot Multibox Dectector

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



SSD: Single Shot Multibox Dectector

- With respect to 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.

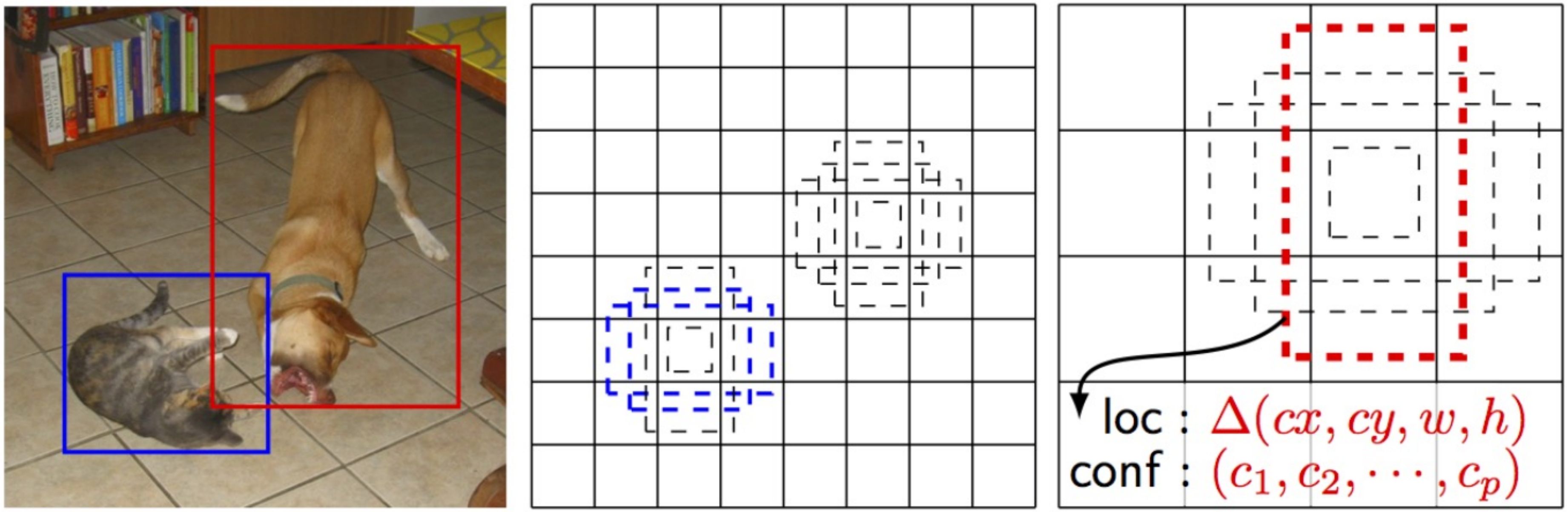


SSD: Single Shot Multibox Dectector

- The main novelty in the SSD detector is that at the end of each module (provided in blue/red colors in the previou slides), there are two kind of values to be returned.
  - The **confidence** scores. This is a set of values (i.e., a vector with as many componente 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) 8×8 feature map (c) 4×4 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)

SSD Loss

- SSD loss has essentially two terms:
  - The **localization loss**, that assesses the mismatch (difference) between the predicted box l and the ground truth g.
    - $L_{loc}(x, l, g) \sum_{i \in Pos}^{N} \sum_{m \in \{cx, cy, w, h\}} x_{ij}^{k} \, |l_i^m - g_i^m|$

For all positive predictions

For "width", "height" and center (x,y) parameters

    - $x_{ij}^{k} = 1 \; iff \; IoU(d_{(p)}, g_{(p)}) > 0.5$

If the default box and the ground truth box on class p overlap more than 50%

SSD Loss

- SSD loss has essentially two terms:
  - The **confidence (classification) loss**. is the loss of making a class prediction. For every positive match prediction, we penalize the loss according to the confidence score of the corresponding class. For negative match predictions, we penalize the loss according to the confidence score of the class "0": class "0" represents that no object is detected.
    - $L_{conf}(x,c) = \sum_{i \in Pos}^{N} x_{ij}^{k} \log\left(\hat{c}_i^p\right) - \sum_{i \in Neg}^{N} \log\left(\hat{c}_i^0\right)$

Where $\hat{c}_i^p$ is the *softmax* of the predicted class, i.e., $\hat{c}_i^p = \frac{\exp(c_i^p)}{\sum_n \exp(c_i^n)}$

SSD Loss

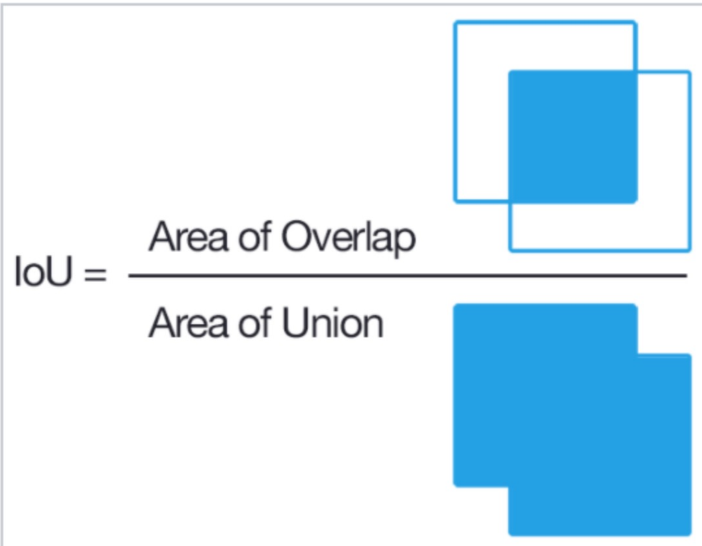
- The final loss is the mean (with respect to the number of positive matches: N) of a weighted combination of L_{conf} and L_{loc}

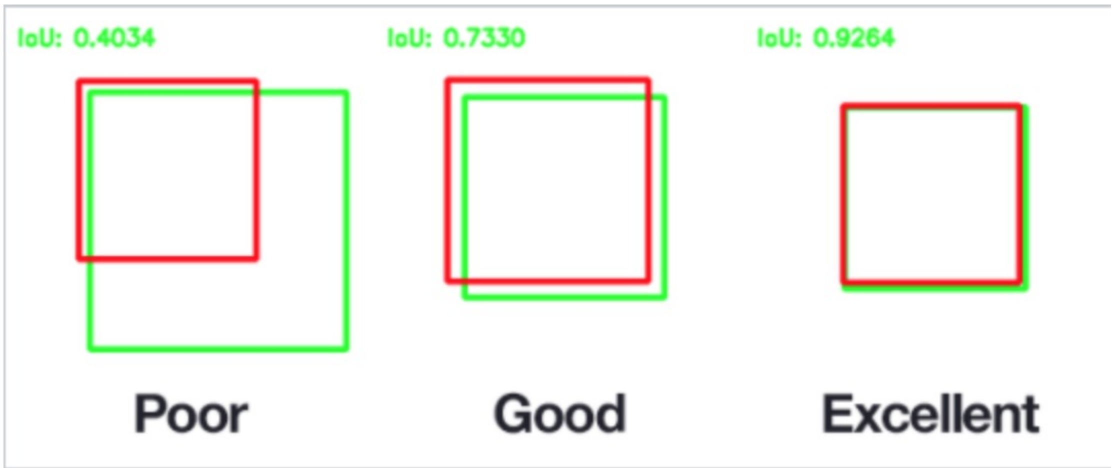
  - $L(x, c, l, g,) = \frac{1}{N}(L_{conf}(x, c) + \alpha L_{loc}(x, l, g))$

Example of a hyper-parameter that can dramatically change performance and should be defined only in the validation set

SSD: Single Shot Multibox Dectector

- 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 (thipically 0.5).
  - In that case, the **regression coefficients** are adjusted to maximize the **IoU** value





Network-based Detectors

- **SSD**, and subsequently **YOLO**, **Fast-R-CNN** and **Faster-R-CNN** are deep learning-based solutions tht advanced enourmously the state-of-the-art in trems of object detection.
- They replaced the previous technique, based in **handcrafted features,** such as the famous **Viola and Jones** object detector (a.k.a. AdaBoost Detector).
  - For decades, this detector represented the state-of-the-art technique, due to its computational effectiveness (speed) and originality.