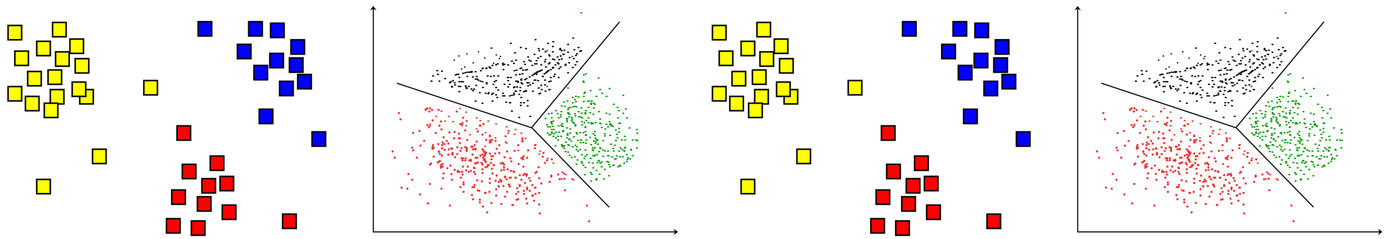
THE UNIVERSITY OF BRITISH COLUMBIA

CPSC 425: Computer Vision



Lecture 28: Image Classification (final)

Menu for Today (**November 18, 2020**)

Topics:

- Boosting
- Sliding Window Object Detection
- Face detection with Viola & Jones
- Object Proposals

Redings:

- **Today's** Lecture: Forsyth & Ponce (2nd ed.) 16.1.3, 16.1.4, 16.1.9
- **Next** Lecture: Forsyth & Ponce (2nd ed.) 17.1–17.2

Reminders:

- **Assignment 5**: Scene Recognition with Bag of Words due **November 20**
- Friendly **Assignment 5** competition
- **Quiz 5** will be due end-of-day on Friday

Today's "**fun**" Example:

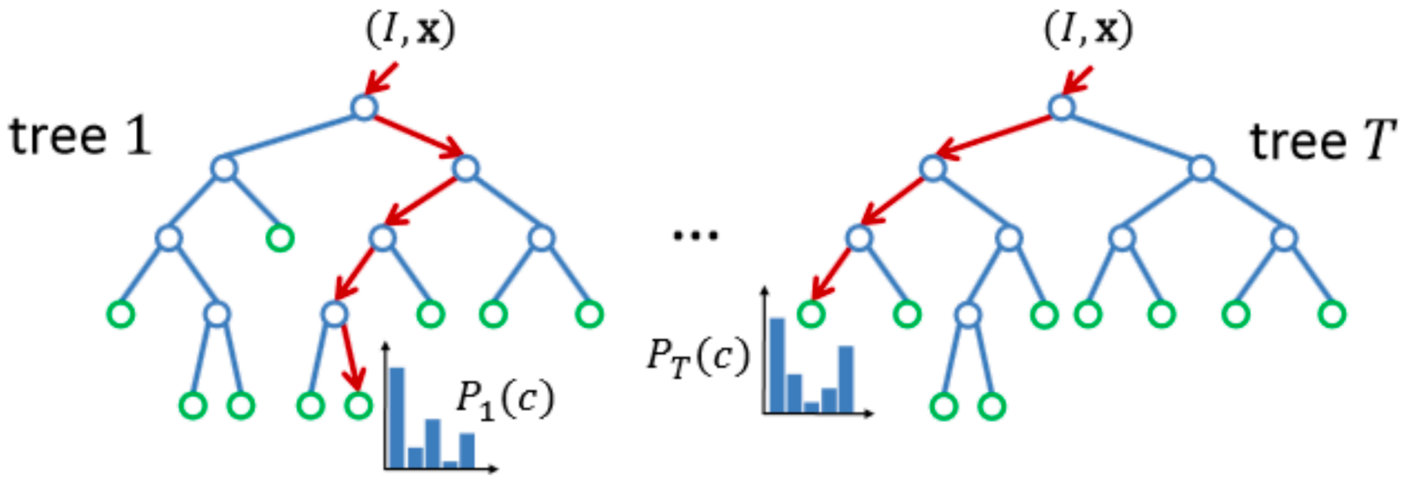
Wait for it … :)

Lecture 27: Re-cap **Decision** Trees

A **random forest** is an ensemble of decision trees.

Randomness is incorporated via training set sampling and/or generation of the candidate binary tests

The prediction of the random forest is obtained by averaging over all decision trees.

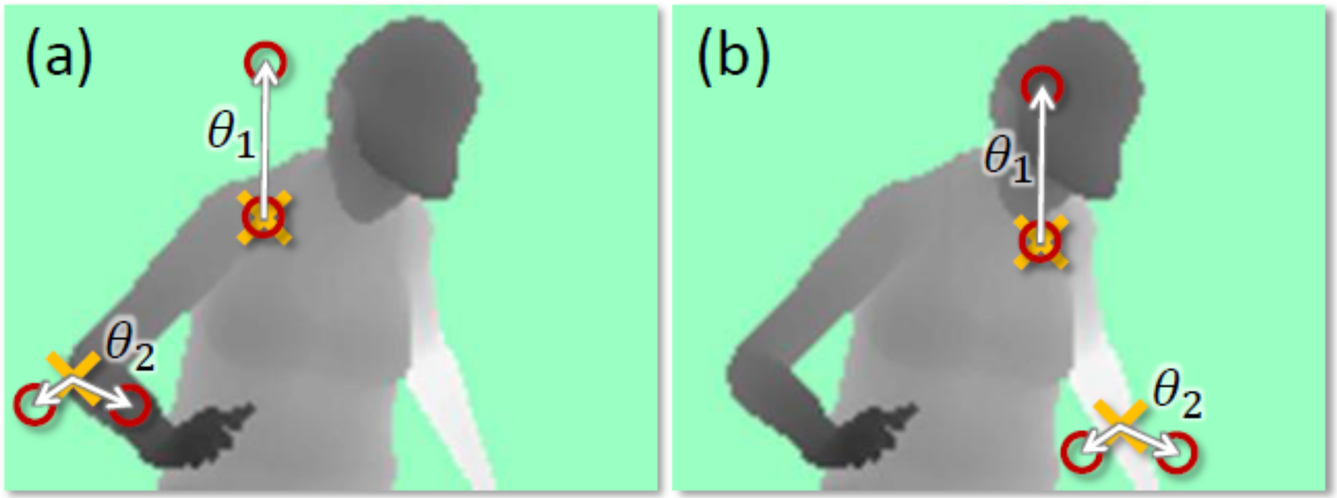


Forsyth & Ponce (2nd ed.) Figure 14.19. Original credit: J. Shotton et al., 2011

Lecture 27: Re-cap **Kinect**

Figure credit: J. Shotton et al., 2011

Lecture 27: Re-cap **Kinect**

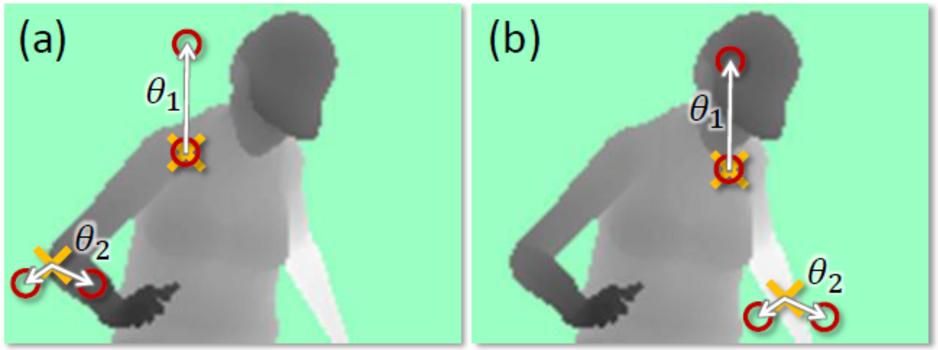


$$f_\theta(I, \mathbf{x}) = d_I\left(\mathbf{x} + \frac{\mathbf{u}}{d_I(\mathbf{x})}\right) - d_I\left(\mathbf{x} + \frac{\mathbf{v}}{d_I(\mathbf{x})}\right)$$

Figure credit: J. Shotton et al., 2011

Lecture 27: Re-cap **Kinect**

$$f_\theta(I, \mathbf{x}) > \Theta_j$$

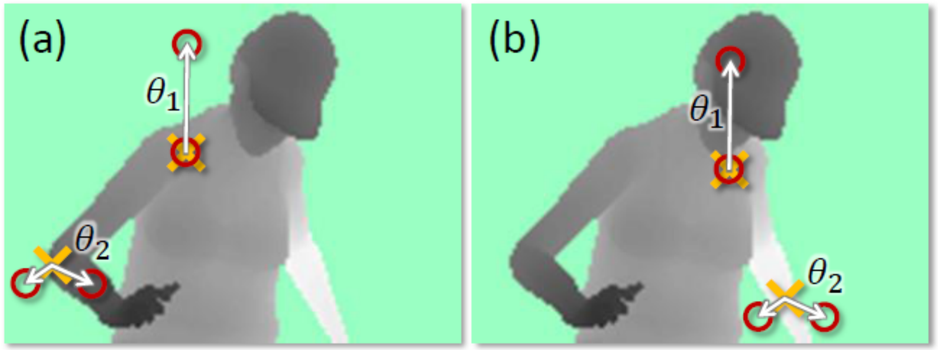


$$f_\theta(I, \mathbf{x}) = d_I\left(\mathbf{x} + \frac{\mathbf{u}}{d_I(\mathbf{x})}\right) - d_I\left(\mathbf{x} + \frac{\mathbf{v}}{d_I(\mathbf{x})}\right)$$

Lecture 27: Re-cap **Kinect**

$\mathbf{u}$	$\mathbf{u}$	Θ_j
↙	→	0.5
↙	→	0.4
↙	→	-0.2
…	…	…
↙	↑	0.7
←	↑	-0.7
→	↖	0.45
…	…	…

$$f_\theta(I, \mathbf{x}) > \Theta_j$$

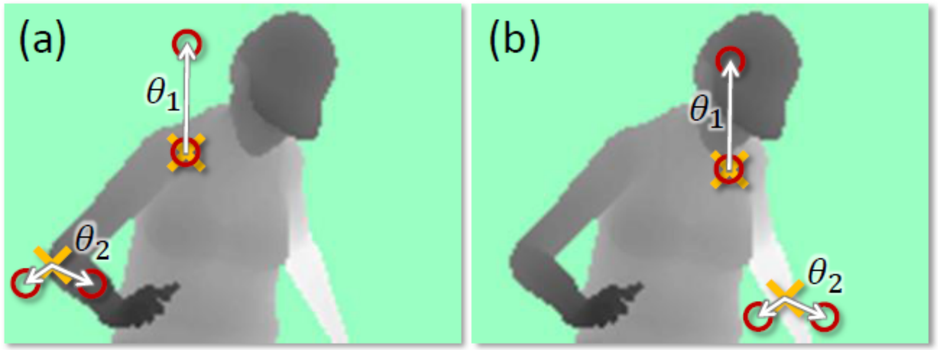


$$f_\theta(I, \mathbf{x}) = d_I\left(\mathbf{x} + \frac{\mathbf{u}}{d_I(\mathbf{x})}\right) - d_I\left(\mathbf{x} + \frac{\mathbf{v}}{d_I(\mathbf{x})}\right)$$

Lecture 27: Re-cap **Kinect**

$\mathbf{u}$	$\mathbf{u}$	Θ_j	information gain
↙	→	0.5	0.3
↙	→	0.4	0.4
↙	→	-0.2	0.7
…	…	…	…
↙	↑	0.7	0.2
←	↑	-0.7	0.8
→	↖	0.45	0.1
…	…	…	…

$$f_\theta(I, \mathbf{x}) > \Theta_j$$

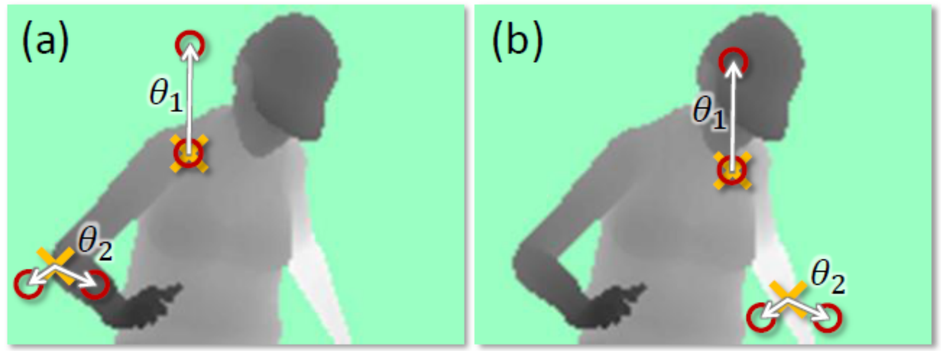


$$f_\theta(I, \mathbf{x}) = d_I\left(\mathbf{x} + \frac{\mathbf{u}}{d_I(\mathbf{x})}\right) - d_I\left(\mathbf{x} + \frac{\mathbf{v}}{d_I(\mathbf{x})}\right)$$

Lecture 27: Re-cap **Kinect**

$\mathbf{u}$	$\mathbf{u}$	Θ_j	information gain
↙	→	0.5	0.3
↙	→	0.4	0.4
↙	→	-0.2	0.7
…	…	…	…
↙	↑	0.7	0.2
←	↑	**-0.7**	**0.8**
→	↖	0.45	0.1
…	…	…	…

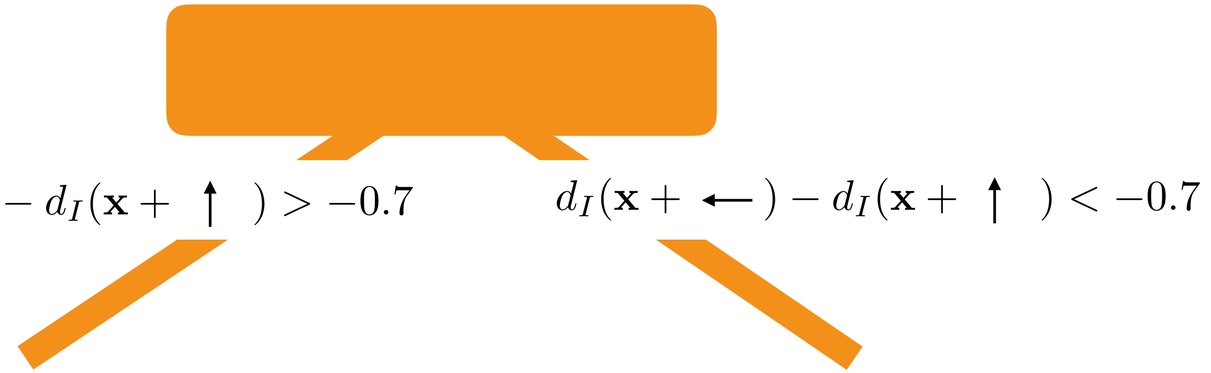
$$f_\theta(I, \mathbf{x}) > \Theta_j$$

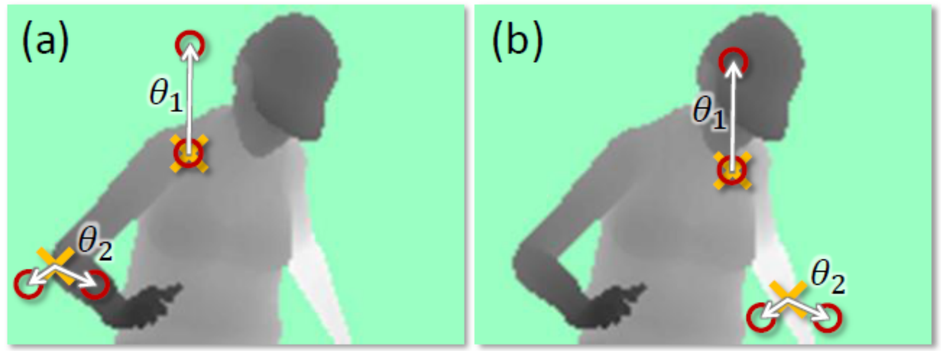


$$f_\theta(I, \mathbf{x}) = d_I\left(\mathbf{x} + \frac{\mathbf{u}}{d_I(\mathbf{x})}\right) - d_I\left(\mathbf{x} + \frac{\mathbf{v}}{d_I(\mathbf{x})}\right)$$

Lecture 27: Re-cap **Kinect**

$\mathbf{u}$	$\mathbf{u}$	Θ_j	information gain
↙	→	0.5	0.3
↙	→	0.4	0.4
↙	→	-0.2	0.7
…	…	…	…
↙	↑	0.7	0.2
←	↑	**-0.7**	**0.8**
→	↖	0.45	0.1
…	…	…	…





$$f_\theta(I, \mathbf{x}) = d_I\left(\mathbf{x} + \frac{\mathbf{u}}{d_I(\mathbf{x})}\right) - d_I\left(\mathbf{x} + \frac{\mathbf{v}}{d_I(\mathbf{x})}\right)$$

Combining **Classifiers**

One common strategy to obtain a better classifier is to combine multiple classifiers.

A simple approach is to train an ensemble of independent classifiers, and average their predictions.

Boosting is another approach.

— Train an ensemble of classifiers sequentially.

— Bias subsequent classifiers to correctly predict training examples that previous classifiers got wrong.

— The final boosted classifier is a weighted combination of the individual classifiers.

Combining Classifiers: **Boosting**

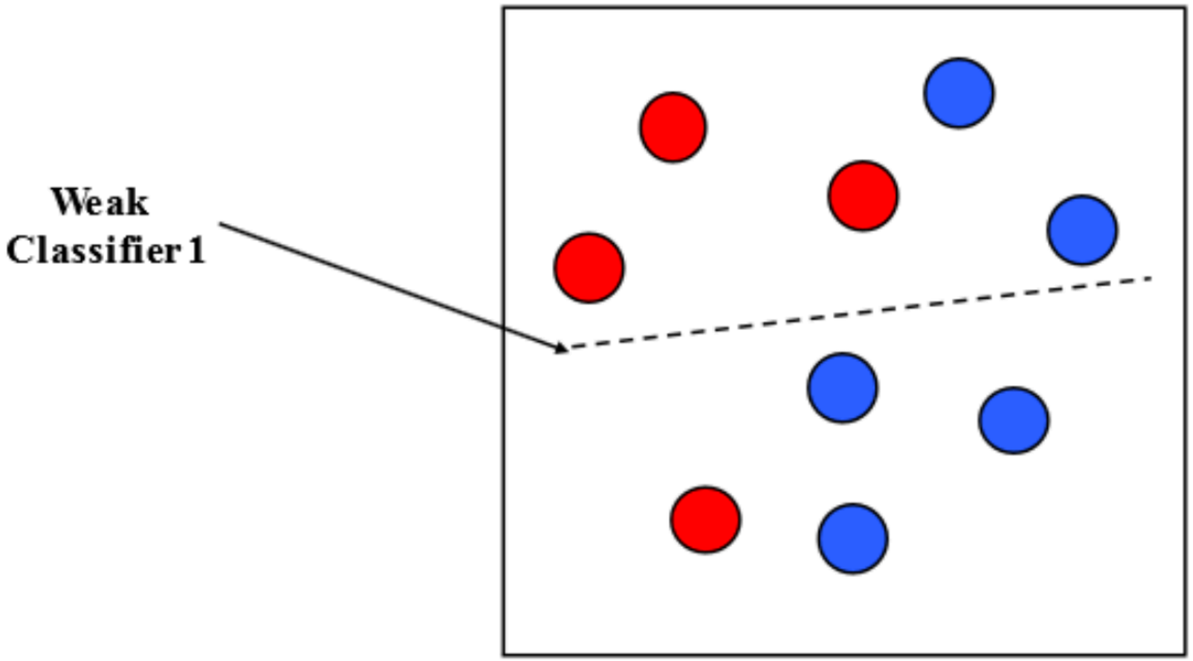


Figure credit: Paul Viola

Combining Classifiers: **Boosting**

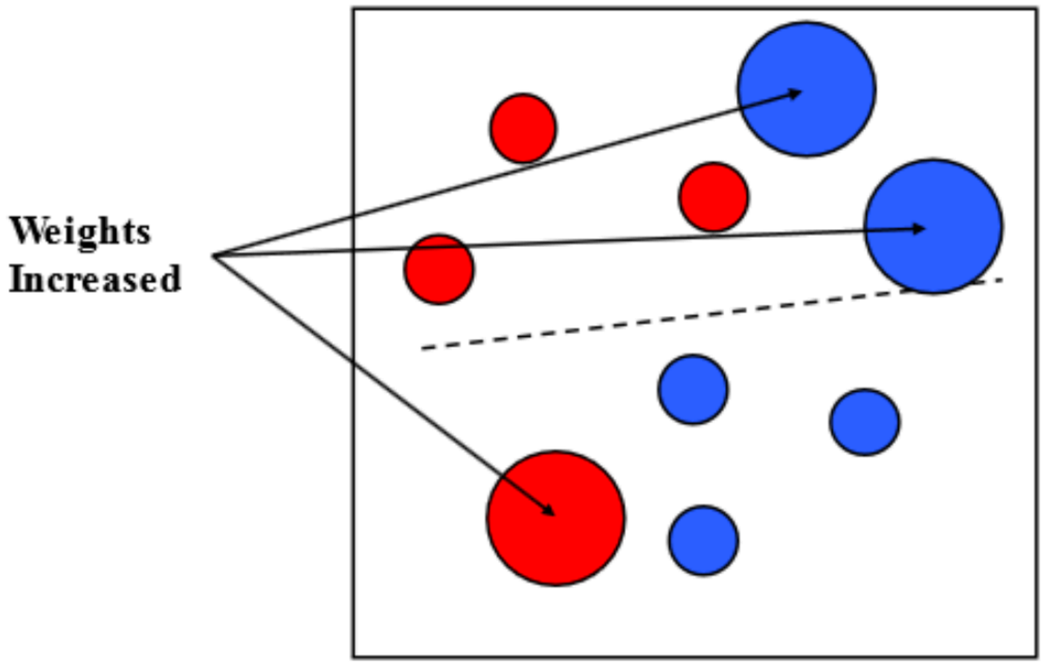


Figure credit: Paul Viola

Combining Classifiers: **Boosting**

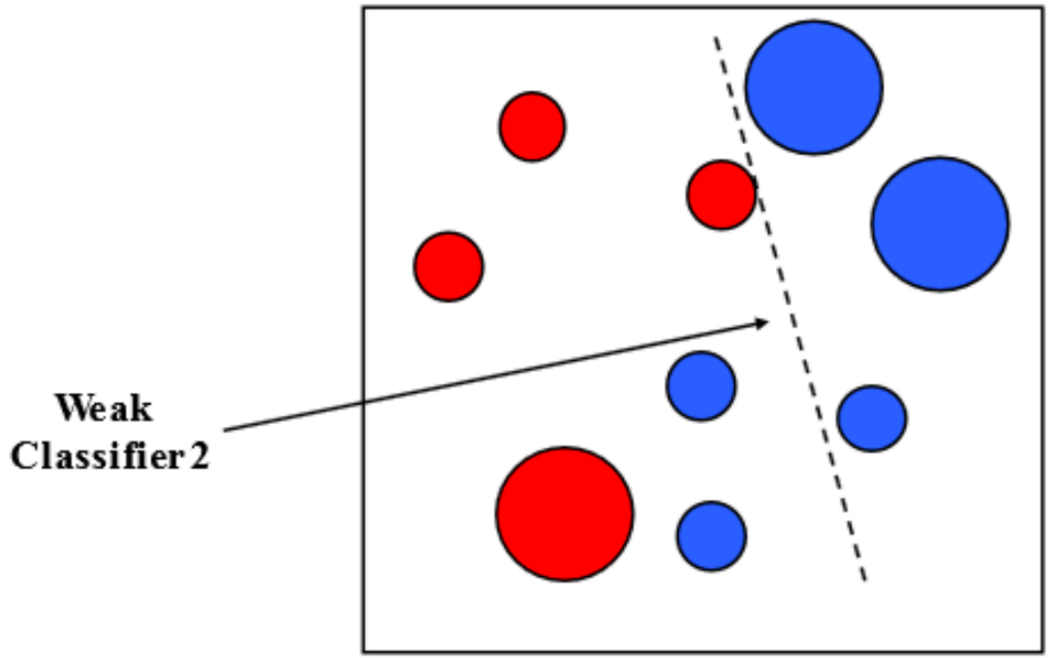


Figure credit: Paul Viola

Combining Classifiers: **Boosting**

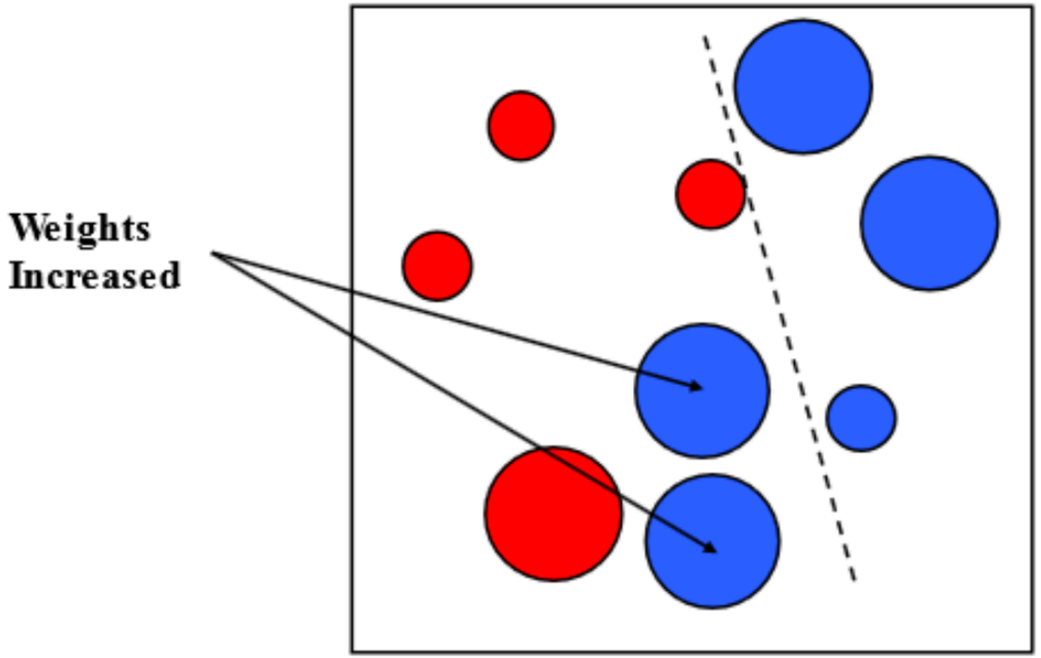


Figure credit: Paul Viola

Combining Classifiers: **Boosting**

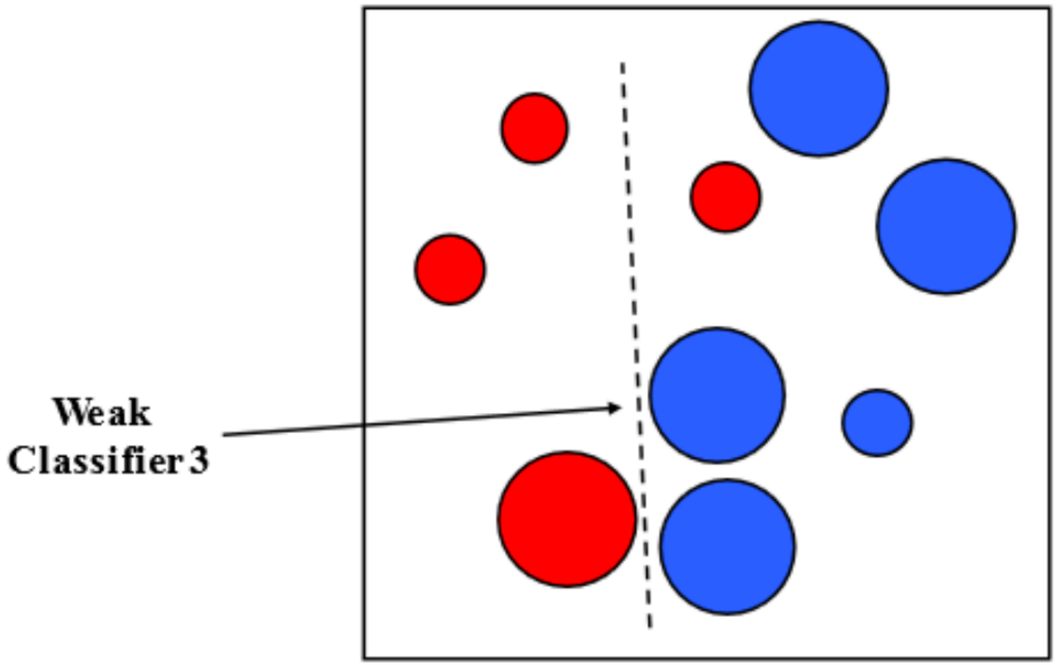


Figure credit: Paul Viola

Combining Classifiers: **Boosting**

Final classifier is a combination of weak classifiers

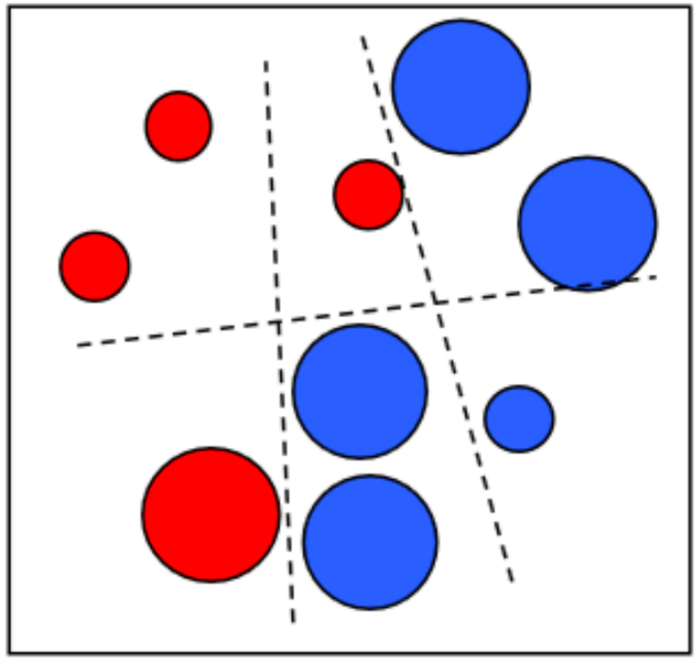
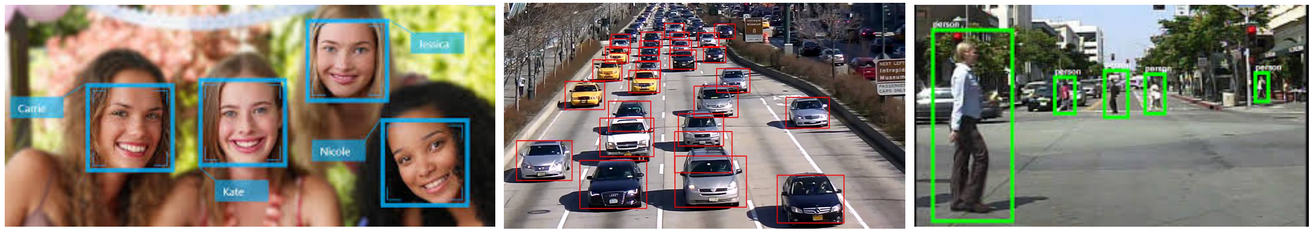


Figure credit: Paul Viola

CPSC 425: Computer Vision



Lecture 28: Object Detection

Object Detection: Introduction

We have been discussing **image classification**, where we pass a whole image into a classifier and obtain a class label as output

We assumed the image contained a single, central object

The task of **object detection** is to detect and localize all instances of a target object class in an image
— Localization typically means putting a tight bounding box around the object

Sliding Window

Train an image classifier as described previously. ‘Slide’ a fixed-sized detection window across the image and evaluate the classifier on each window.

Image credit: KITTI Vision Benchmark

Sliding Window

Train an image classifier as described previously. ‘Slide’ a fixed-sized detection window across the image and evaluate the classifier on each window.

Is there a car?

Image credit: KITTI Vision Benchmark

Sliding Window

Train an image classifier as described previously. ‘Slide’ a fixed-sized detection window across the image and evaluate the classifier on each window.

Is there a car?

Image credit: KITTI Vision Benchmark

Sliding Window

Train an image classifier as described previously. ‘Slide’ a fixed-sized detection window across the image and evaluate the classifier on each window.

Image credit: KITTI Vision Benchmark

Sliding Window

Train an image classifier as described previously. ‘Slide’ a fixed-sized detection window across the image and evaluate the classifier on each window.

Is there a car?

Image credit: KITTI Vision Benchmark

Sliding Window

Train an image classifier as described previously. ‘Slide’ a fixed-sized detection window across the image and evaluate the classifier on each window.

Image credit: KITTI Vision Benchmark

Sliding Window

Train an image classifier as described previously. ‘Slide’ a fixed-sized detection window across the image and evaluate the classifier on each window.

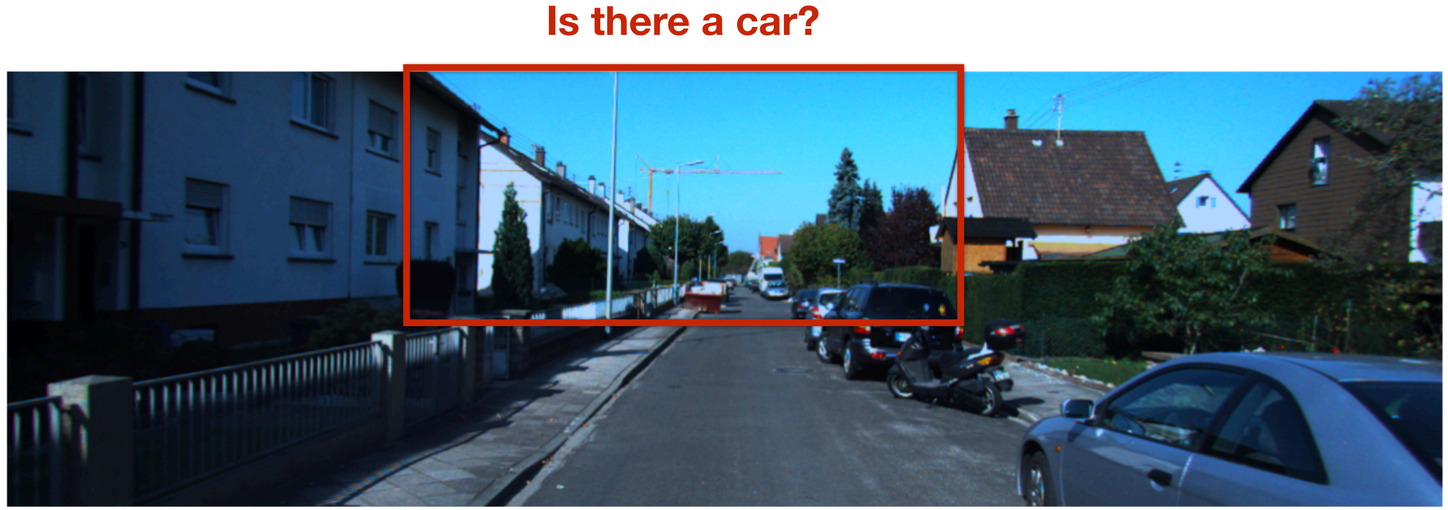


Image credit: KITTI Vision Benchmark

Sliding Window

Train an image classifier as described previously. ‘Slide’ a fixed-sized detection window across the image and evaluate the classifier on each window.

Is there a car?

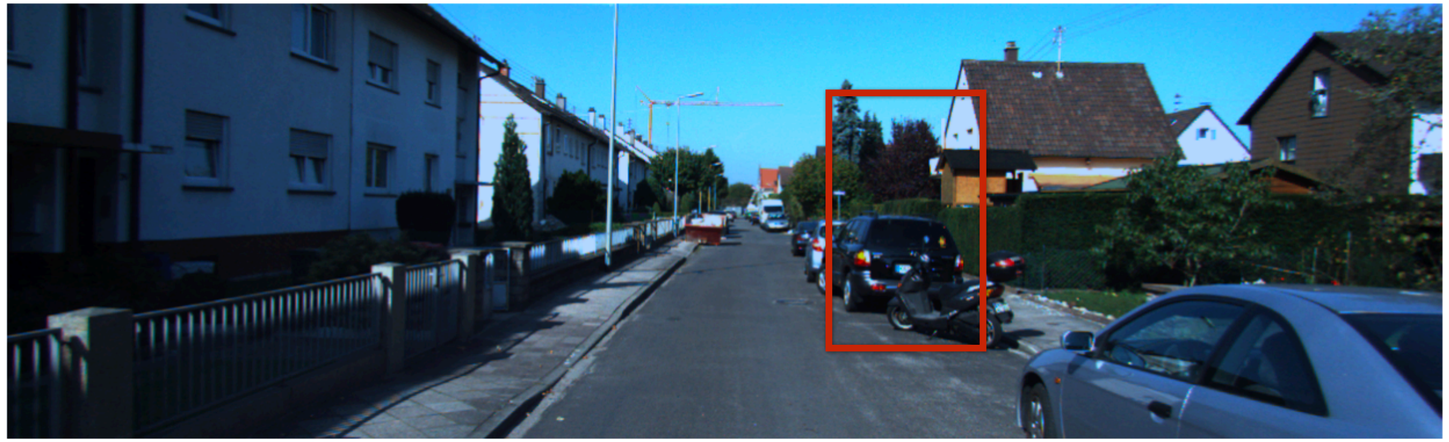


Image credit: KITTI Vision Benchmark

Sliding Window

Train an image classifier as described previously. ‘Slide’ a fixed-sized detection window across the image and evaluate the classifier on each window.

Is there a car?

Image credit: KITTI Vision Benchmark

Sliding Window

Train an image classifier as described previously. ‘Slide’ a fixed-sized detection window across the image and evaluate the classifier on each window.

Image credit: KITTI Vision Benchmark

This is a search over location
— We have to search over scale as well
— We may also have to search over aspect ratios

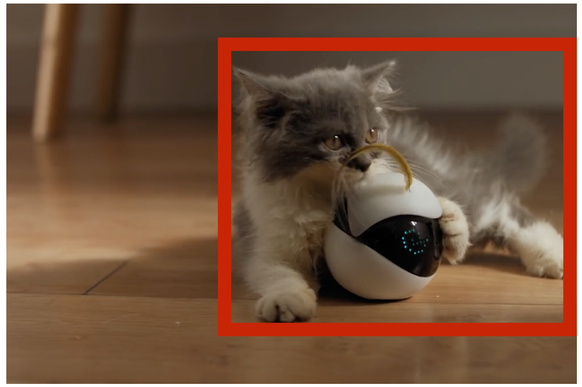
What data we **train** a classifier on?

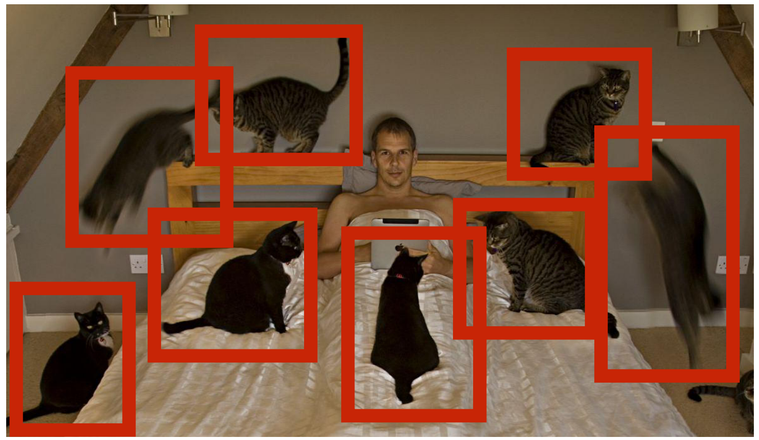
Image Classifiers

Image classifiers can be applied to regions/windows, but do not work so well in practice …

What data we **train** a classifier on?

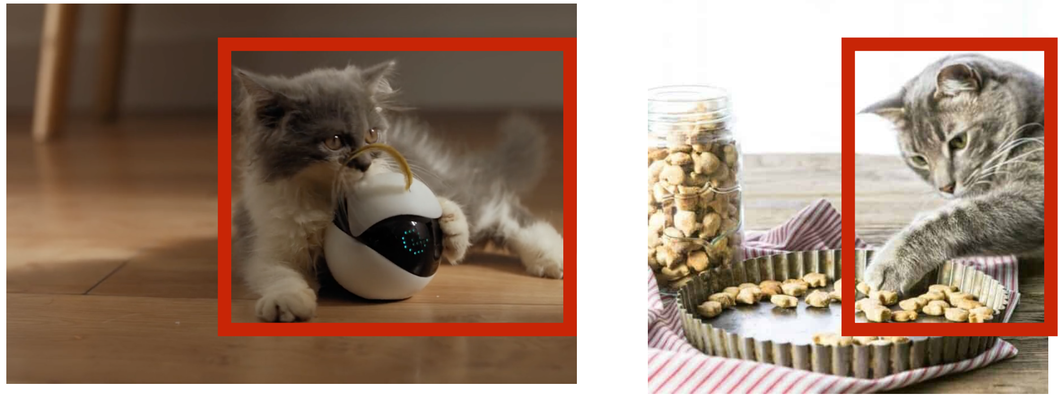
Image Classifiers



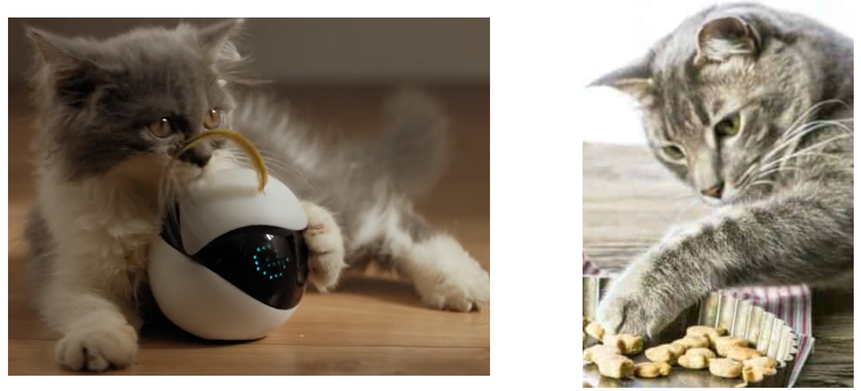


What data we **train** a classifier on?

Image Classifiers



Object Classifiers



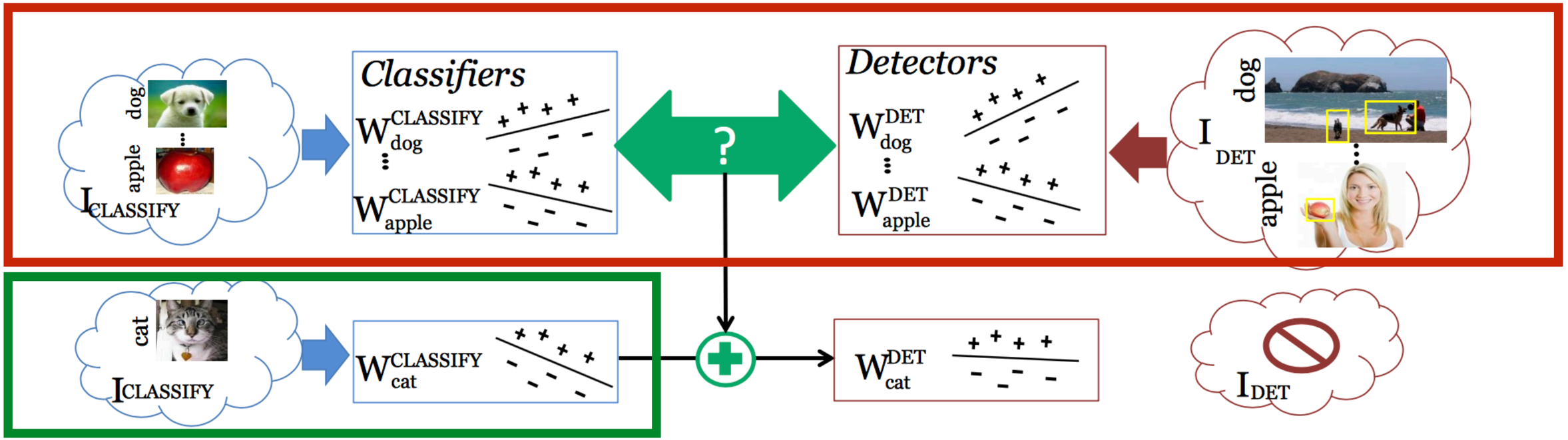
What data we **train** a classifier on?

Object classifiers work a lot better … but require expensive bounding box annotations …

Object Classifiers

Today's "**fun**" Example: Detection with **No** Data

Expensive — do this for few classes (e.g., 20)



Cheap — do this for many classes (e.g., 80 or 20,000)

LSDA: Large Scale Detection through Adaptation (Hoffman et al, 2014)

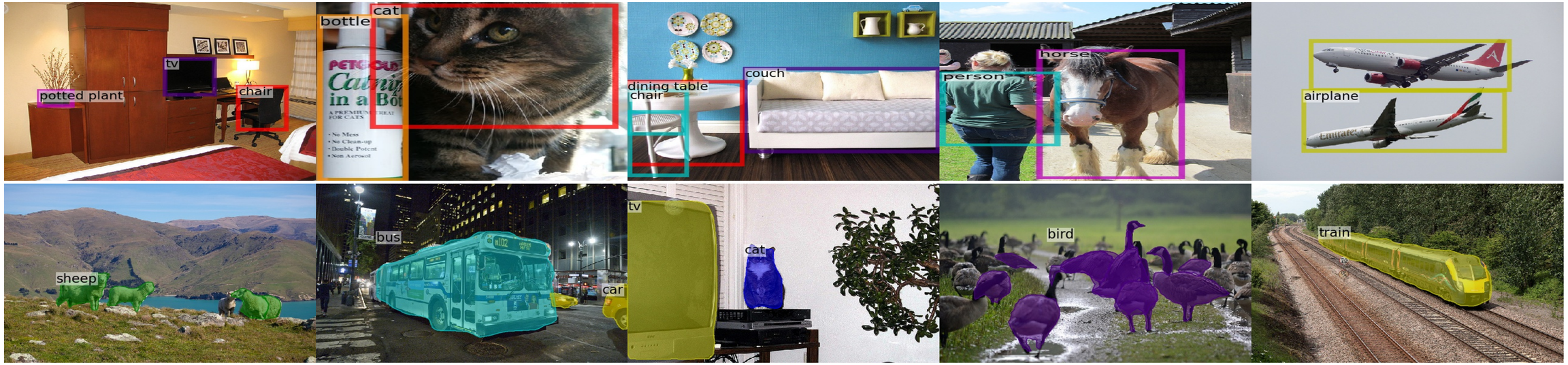
Today's "**fun**" Example: Detection with **No** Data

Our **UniT**: Unified Knowledge Transfer

Method	AP_{50}	AP_S	AP_M	AP_L
LSDA [22]	4.6	1.2	5.1	7.8
LSDA+Semantic [52]	4.7	1.1	5.1	8.0
LSDA+MIL [23]	5.9	1.5	8.3	10.7
DOCK [26]	14.4	2.0	12.8	24.9
UniT (Ours)	**16.7**	**3.2**	**16.6**	**27.3**
Full Supervision [26]	25.2	5.8	26.0	41.6

Today's "**fun**" Example: Detection with **No** Data

Our **UniT**: Unified Knowledge Transfer



(Note, above result is obtained without human ever annotating bounding boxes or pixel-level segments for any of these objects)

Let's assume we have **object** labeled data …

Object classifiers work a lot better … but require expensive bounding box annotations …

Object Classifiers

Example: Face Detection

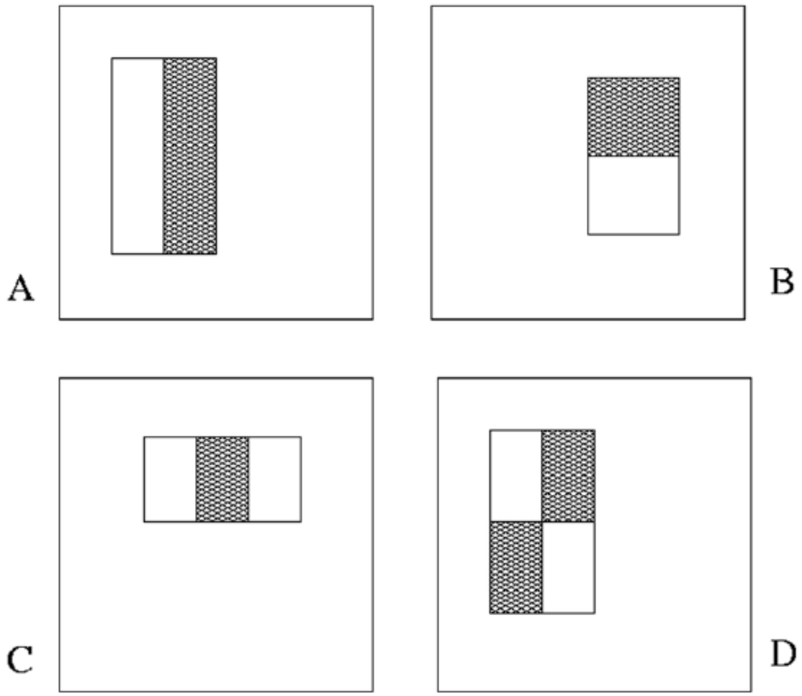
The **Viola-Jones** face detector is a classic sliding window detector that learns both efficient features and a classifier

A key strategy is to use features that are fast to evaluate to reject most windows early

The Viola-Jones detector computes ‘rectangular’ features within each window

Example: Face Detection

A ‘rectangular’ feature is computed by summing up pixel values within rectangular regions and then differencing those region sums

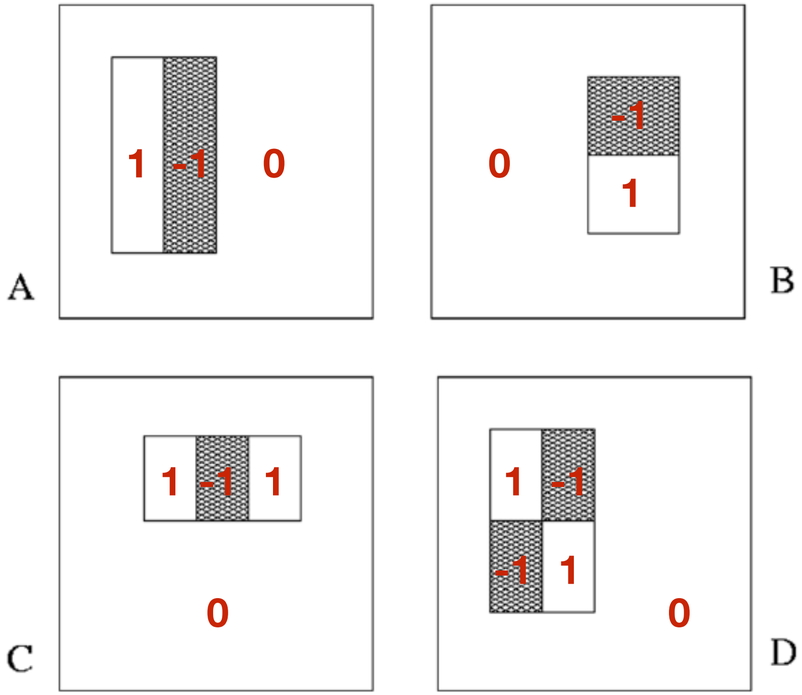


a.k.a. **Harr** Wavelets

Figure credit: P. Viola and M. Jones, 2001

Example: Face Detection

A ‘rectangular’ feature is computed by summing up pixel values within rectangular regions and then differencing those region sums



a.k.a. **Harr** Wavelets

Figure credit: P. Viola and M. Jones, 2001

Intergral Image

$$A(x,y) = \sum_{x' \leq x, y' \leq y} I(x', y')$$

$I(x,y)$

1	5	2
2	4	1
2	1	1

original image

$A(x,y)$

1	6	8
3	12	15
5	15	19

integral image

Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

Intergral Image

What is the sum of the bottom right 2x2 square?

$I(x, y)$

1	5	2
2	4	1
2	1	1

original image

$A(x, y)$

1	6	8
3	12	15
5	15	19

integral image

Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

Intergral Image

What is the sum of the bottom right 2x2 square?

$I(x, y)$

1	5	2
2	4	1
2	1	1

original image

$A(x, y)$

1	6	8
3	12	15
5	15	19

integral image

Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

Intergral Image

What is the sum of the bottom right 2x2 square?

$I(x, y)$

1	5	2
2	4	1
2	1	1

original image

$A(x, y)$

1	6	8
3	12	15
5	15	19

integral image

$$
\begin{aligned}
A(1,1,3,3) &= A(3,3) - A(1,3) - A(3,1) + A(1,1) \\
&= 19 - 8 - 5 + 1 \\
&= 7
\end{aligned}
$$

Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

Intergral Image

$$A(x, y) = \sum_{x' \leq x, y' \leq y} I(x', y')$$

$I(x, y)$

1	5	2
2	4	1
2	1	1

original image

$A(x, y)$

1	6	8
3	12	15
5	15	19

integral image

Can find the **sum** of any block using **3** operations

$$A(x_1, y_1, x_2, y_2) = A(x_2, y_2) - A(x_1, y_2) - A(x_2, y_1) + A(x_1, y_1)$$

Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

Example: Face Detection

Given an integral image, the sum within a rectangular region in I can be computed with just 3 additions/subtractions

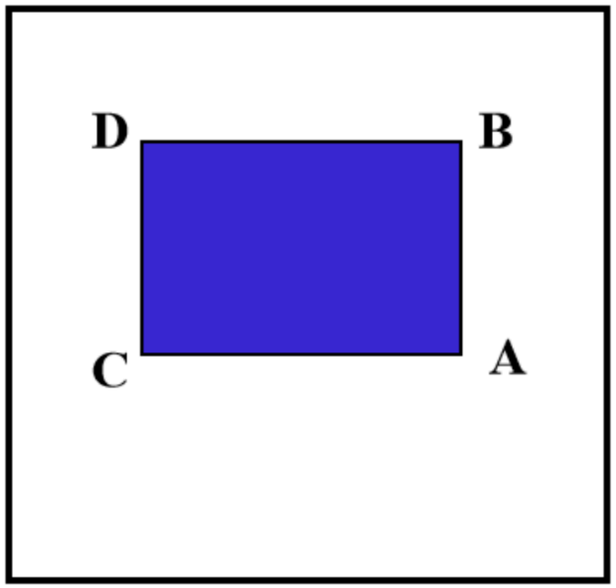


Figure credit: P. Viola

Sum = A - B - C + D

Constant time: does not depend on the size of the region. We can avoid scaling images - just scale features directly (remember template matching!)

Integral Image Layer for Deep Neural Networks

In a classical paper [1] from 2001, Viola and Jones popularized the use of large rectangular image filters in order to obtain features for image recognition. The use of very large filters allowed Viola and Jones to compute features over very large receptive fields without blowing up the computation cost. For the next 10+ years, such features remained the staple of fast computer vision (e.g. [2]). The advent of deep learning made the use of integral-image features far less popular. Currently, state-of-the-art architectures invariably relying on very deep architectures. In these architectures sufficiently large receptive fields are obtained via the use of downsampling with subsequent upsampling [3] or via dilated convolutions [4]. All such tricks however have their downsides and usually necessitate the use of very deep networks.

The goal of this project is to implement an integral image-based filtering as a layer for deep architectures in Torch deep learning package, and to evaluate it for the task of learning very fast object detectors (as an alternative to e.g. [5]) and semantic segmentation systems (as an alternative to e.g. [3,4]). The hope is to obtain much shallower architectures, which at least for simple classes (e.g. roadsigns or upright pedestrians) will approach the performance of much deeper ones.

The project is supervised by Victor Lempitsky at Skoltech, Moscow, Russia.

https://github.com/shrubb/integral-layer

Deep Neural Networks for Object Detection

Christian Szegedy **Alexander Toshev** **Dumitru Erhan**
Google, Inc.
{szegedy, toshev, dumitru}@google.com

Abstract

Deep Neural Networks (DNNs) have recently shown outstanding performance on image classification tasks [14]. In this paper we go one step further and address the problem of object detection using DNNs, that is not only classifying but also precisely localizing objects of various classes. We present a simple and yet powerful formulation of object detection as a regression problem to object bounding box masks. We define a multi-scale inference procedure which is able to produce high-resolution object detections at a low cost by a few network applications. State-of-the-art performance of the approach is shown on Pascal VOC.

Example: Face Detection

Training Dataset:

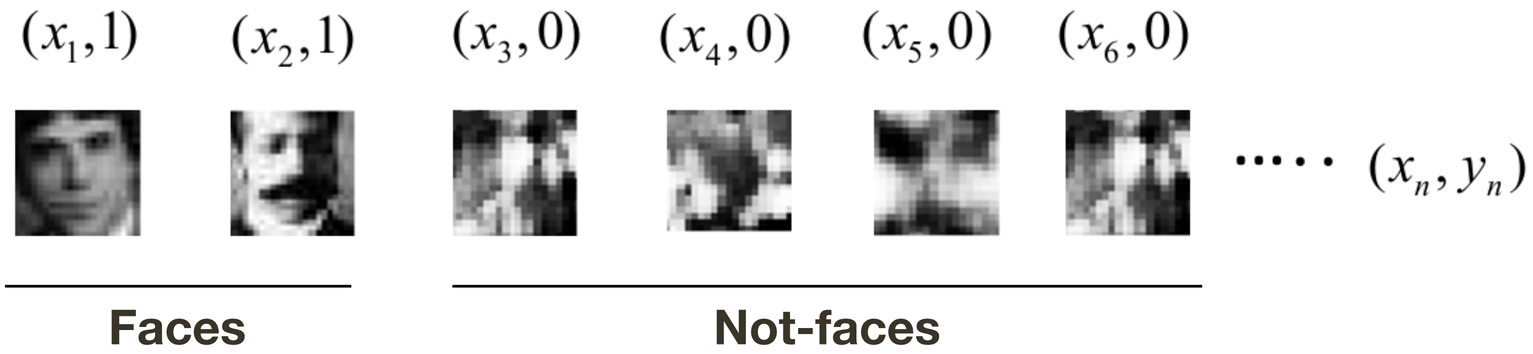
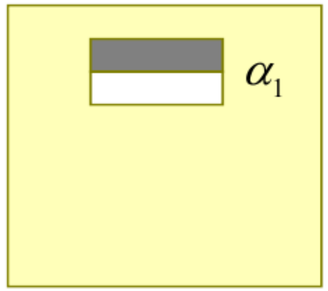


Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

Example: Face Detection

Evaluate a Harr Wavelet filter on each training example



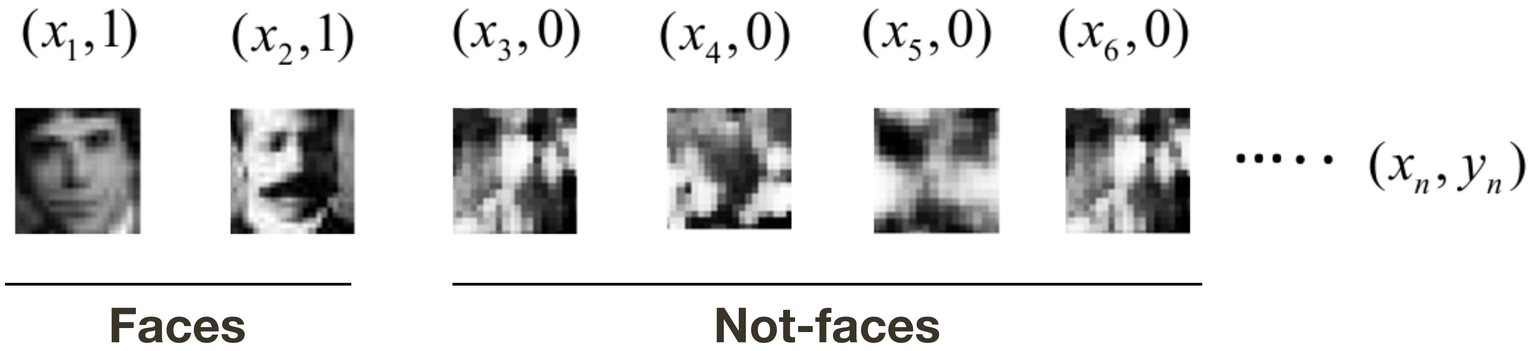
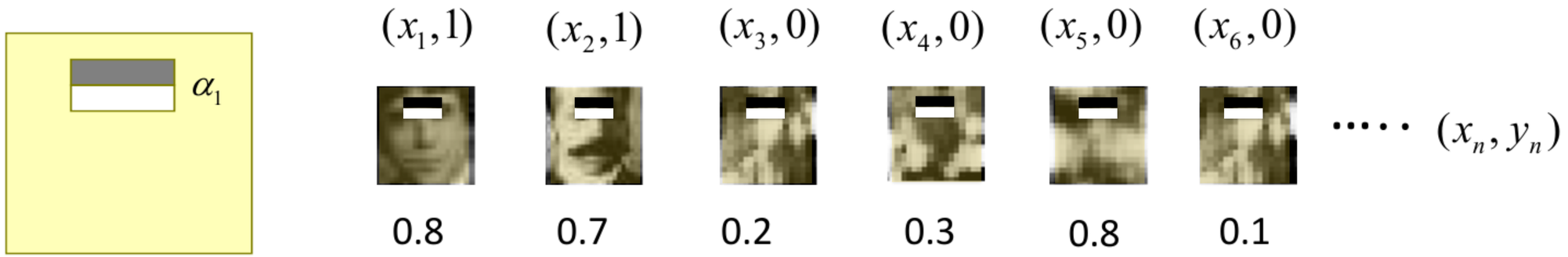


Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

Example: Face Detection

Evaluate a Harr Wavelet filter on each training example



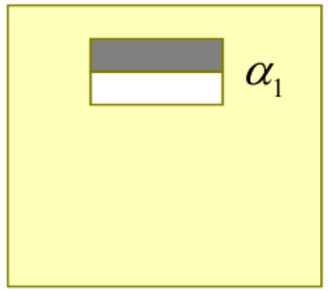
We can build a simple classifier by just selecting a threshold on the filter response (e.g. Harr filter response > 0.6 = face; Harr filter response <= 0.6 = not face)

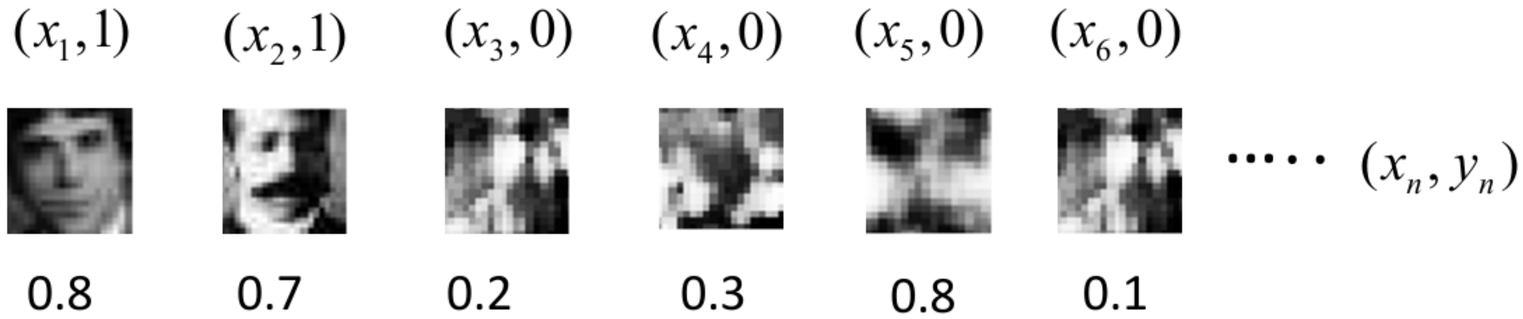
Note: it is easy to find an **optimal** threshold. Just requires linear search over training example responses.

Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

Example: Face Detection

Evaluate a Harr Wavelet filter on each training example



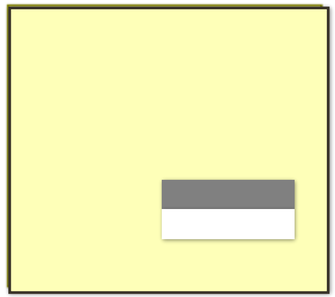


Weak classifier $h_j(x) = \begin{cases} 1 & \text{if } f_j(x) > \theta_j \\ 0 & \text{otherwise} \end{cases}$ ← threshold

Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

Example: Face Detection

Evaluate a Harr Wavelet filter on each training example



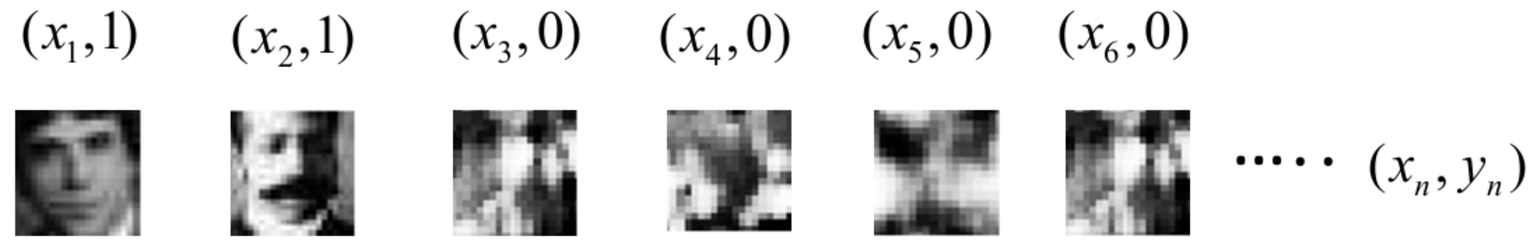
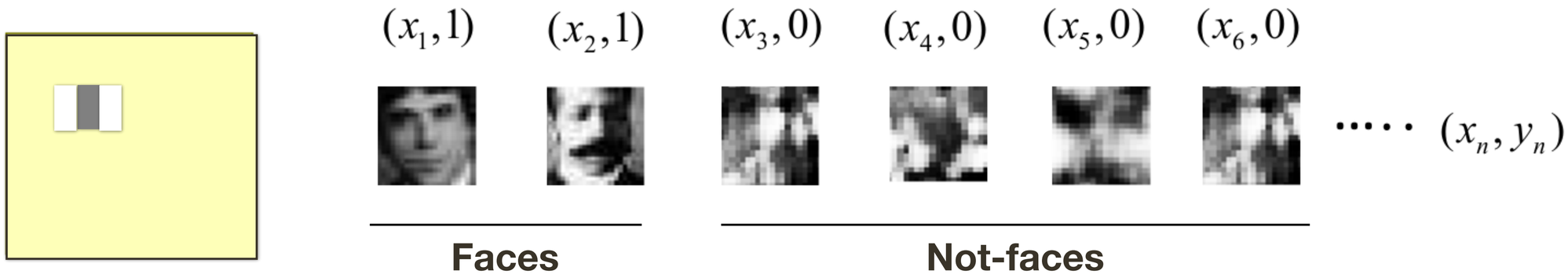


Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

Example: Face Detection

Evaluate a Harr Wavelet filter on each training example



Note: we can easily compare different Harr Wavelet features under their individual best thresholds to see which is the most informative (has highest classification)

Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

Example: Face Detection

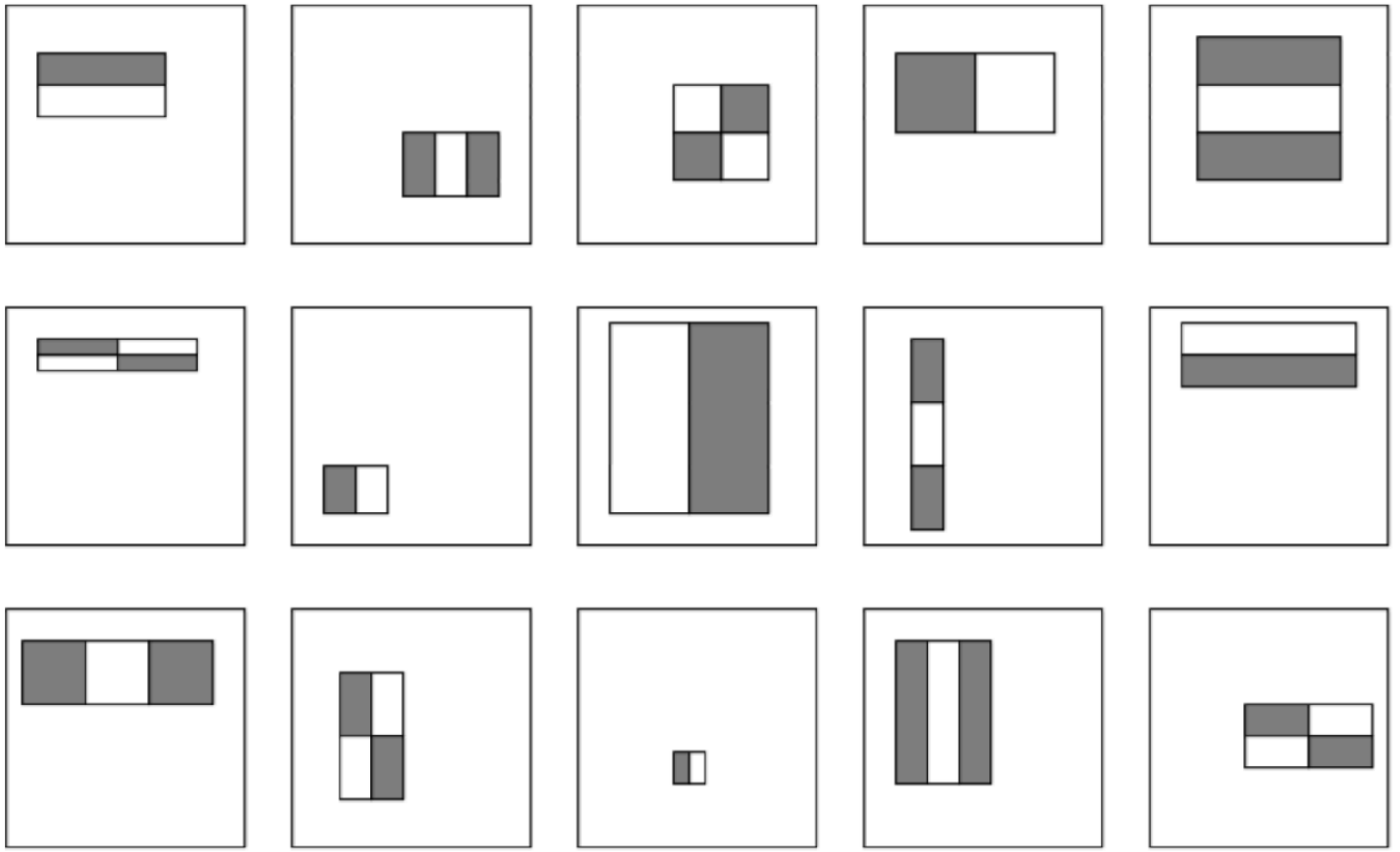
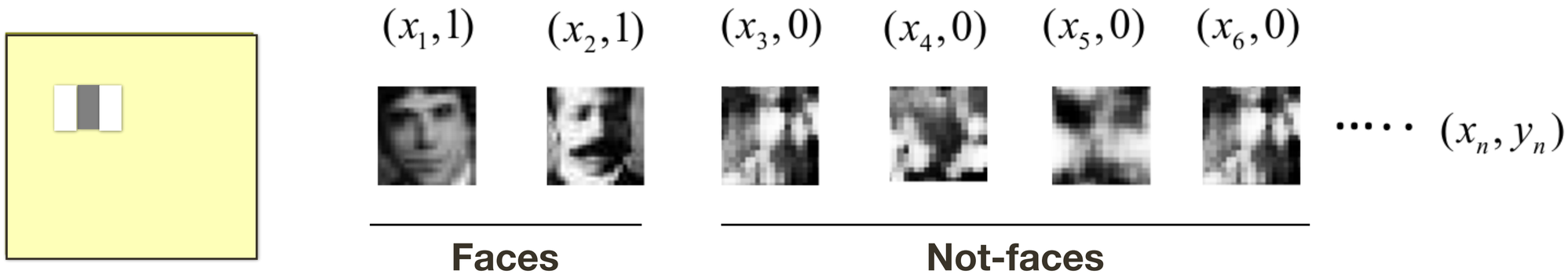


Figure credit: B. Freeman

Many possible rectangular features (180,000+ were used in the original paper)

Example: Face Detection

Evaluate a Harr Wavelet filter on each training example



Note: we can easily compare different Harr Wavelet features under their individual best thresholds to see which is the most informative (has highest classification)

However, no one feature is likely to be good enough

Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

Example: Face Detection

Use **boosting** to both select the informative features and form the classifier.
Each round chooses a weak classifier that simply compares a single rectangular feature against a threshold

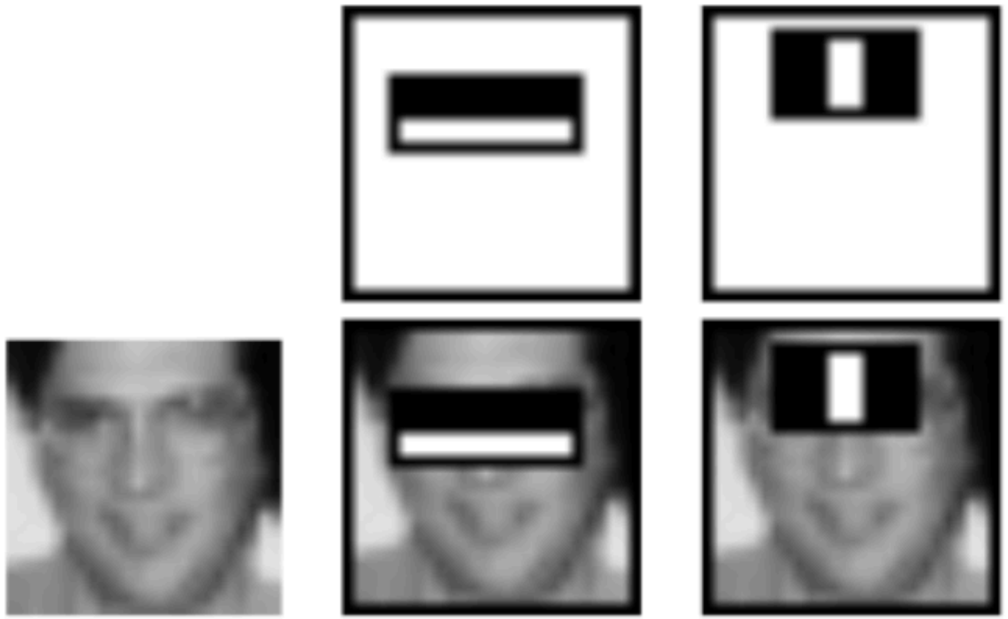


Figure credit: P. Viola and M. Jones, 2001

Example: Face Detection

1. Select best filter/threshold combination

   a. Normalize the weights $w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$

   $$h_j(x) = \begin{cases} 1 & \text{if } f_j(x) > \theta_j \\ 0 & \text{otherwise} \end{cases}$$

   b. For each feature, j $\varepsilon_j = \sum_i w_i \left| h_j(x_i) - y_i \right|$

   c. Choose the classifier, h_t with the lowest error ε_t

2. Re-weight examples

$$w_{t+1,i} = w_{t,i} \beta_t^{1-|h_t(x_i)-y_i|} \qquad \beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$$

Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

Example: Face Detection

1. Select best filter/threshold combination

   a. Normalize the weights $w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$

We start with all sample weights = 1

Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

Example: Face Detection

1. Select best filter/threshold combination

   a. Normalize the weights $w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$

   $$h_j(x) = \begin{cases} 1 & \text{if } f_j(x) > \theta_j \\ 0 & \text{otherwise} \end{cases}$$

   b. For each feature, j $\varepsilon_j = \sum_i w_i \left| h_j(x_i) - y_i \right|$

   weighed sum of miss-classified training examples

Note: the second term is 0/1

— 0 predicted label and true label are same

— 1 predicted label and true label are different (error)

Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

Example: Face Detection

1. Select best filter/threshold combination

   a. Normalize the weights $w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$

   $$h_j(x) = \begin{cases} 1 & \text{if } f_j(x) > \theta_j \\ 0 & \text{otherwise} \end{cases}$$

   b. For each feature, j $\varepsilon_j = \sum_i w_i \left| h_j(x_i) - y_i \right|$

   c. Choose the classifier, h_t with the lowest error ε_t

Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

Example: Face Detection

1. Select best filter/threshold combination

   a. Normalize the weights $w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$

   $$h_j(x) = \begin{cases} 1 & \text{if } f_j(x) > \theta_j \\ 0 & \text{otherwise} \end{cases}$$

   b. For each feature, j $\varepsilon_j = \sum_i w_i \left| h_j(x_i) - y_i \right|$

   c. Choose the classifier, h_t with the lowest error ε_t

2. Re-weight examples

$$w_{t+1,i} = w_{t,i} \beta_t^{1-|h_t(x_i)-y_i|} \qquad \beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$$

Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

Example: Face Detection

Case 1: Classification for the sample i is **correct**

$$\mathbf{w}_{t+1,i} = \mathbf{w}_{t,i}\ \beta_t$$

Case 2: Classification for the sample i is **incorrect**

$$\mathbf{w}_{t+1,i} = \mathbf{w}_{t,i}$$

2. Re-weight examples

$$w_{t+1,i} = w_{t,i}\beta_t^{1-|h_t(x_i)-y_i|} \qquad \beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$$

Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

Example: Face Detection

Case 1: Classification for the sample i is **correct**

$$\mathbf{w}_{t+1,i} = \mathbf{w}_{t,i}\ \beta_t$$

Case 2: Classification for the sample i is **incorrect**

$$\mathbf{w}_{t+1,i} = \mathbf{w}_{t,i}$$

2. Re-weight examples

Note: the Beta is < 1

$$w_{t+1,i} = w_{t,i}\beta_t^{1-|h_t(x_i)-y_i|}$$

$$\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$$

Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

Example: Face Detection

1. Select best filter/threshold combination

   a. Normalize the weights $w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$

   $$h_j(x) = \begin{cases} 1 & \text{if } f_j(x) > \theta_j \\ 0 & \text{otherwise} \end{cases}$$

   b. For each feature, j $\varepsilon_j = \sum_i w_i \left| h_j(x_i) - y_i \right|$

   c. Choose the classifier, h_t with the lowest error ε_t

2. Re-weight examples

$$w_{t+1,i} = w_{t,i} \beta_t^{1-|h_t(x_i)-y_i|} \qquad \beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$$

Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

Example: Face Detection

Viola & Jones algorithm

3. The final strong classifier is

$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

$$\alpha_t = \log \frac{1}{\beta_t}$$

The final strong classifier is a weighted linear combination of the T weak classifiers where the weights are inversely proportional to the training errors

Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

Example: Face Detection Summary

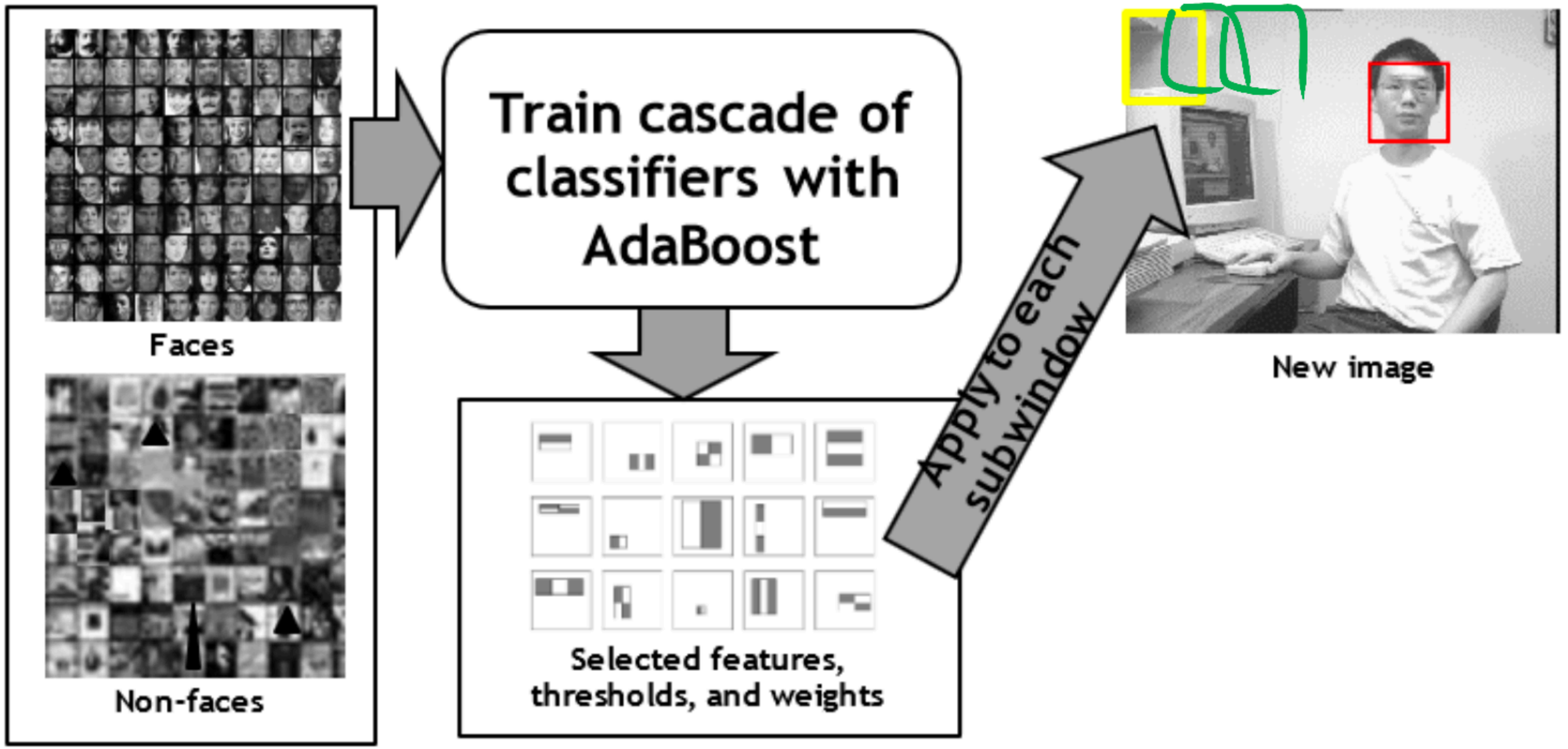
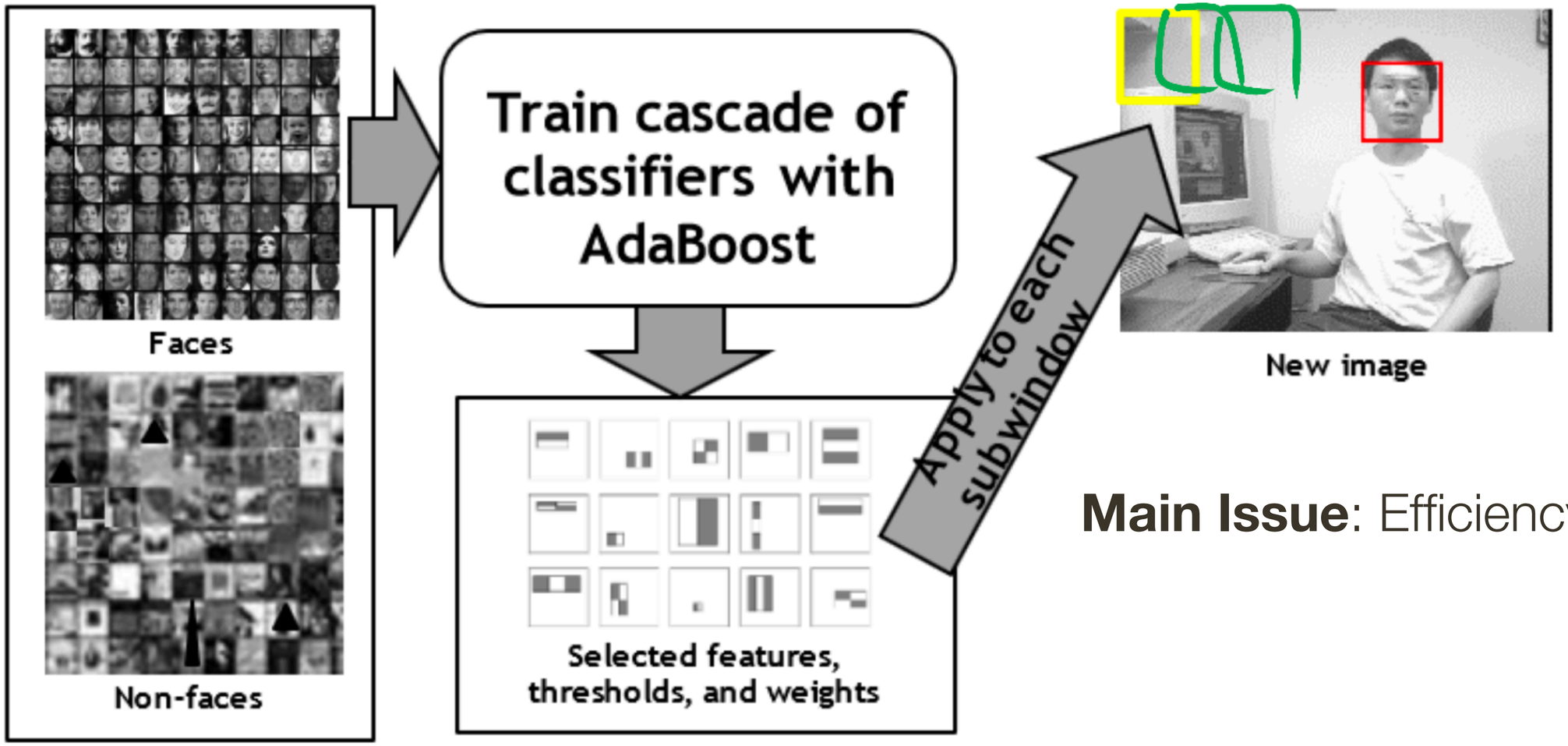


Figure credit: K. Grauman

Example: Face Detection Summary



Main Issue: Efficiency

Figure credit: K. Grauman

Example: Face Detection

Observations:

— On average only **0.01%** of all sub-windows are positive (faces)

— Equal computation time is spent on all sub-window

— Shouldn’t we spend most time only on **potentially positive** sub-windows?

Example: Face Detection

Observations:

— On average only **0.01%** of all sub-windows are positive (faces)

— Equal computation time is spent on all sub-window

— Shouldn’t we spend most time only on **potentially positive** sub-windows?

A simple 2-feature classifier can achieve almost 100% detection rate (0% false negatives) with 50% false positive rate

Example: Face Detection

Observations:

— On average only **0.01%** of all sub-windows are positive (faces)

— Equal computation time is spent on all sub-window

— Shouldn't we spend most time only on **potentially positive** sub-windows?

A simple 2-feature classifier can achieve almost 100% detection rate (0% false negatives) with 50% false positive rate

Solution:

— A simple 2-feature classifier can act as a 1st layer of a series to filter out most negative (clearly non-face) windows

— 2nd layer with 10 features can tackle "harder" negative-windows which survived the 1st layer, and so on…

Cascading Classifiers

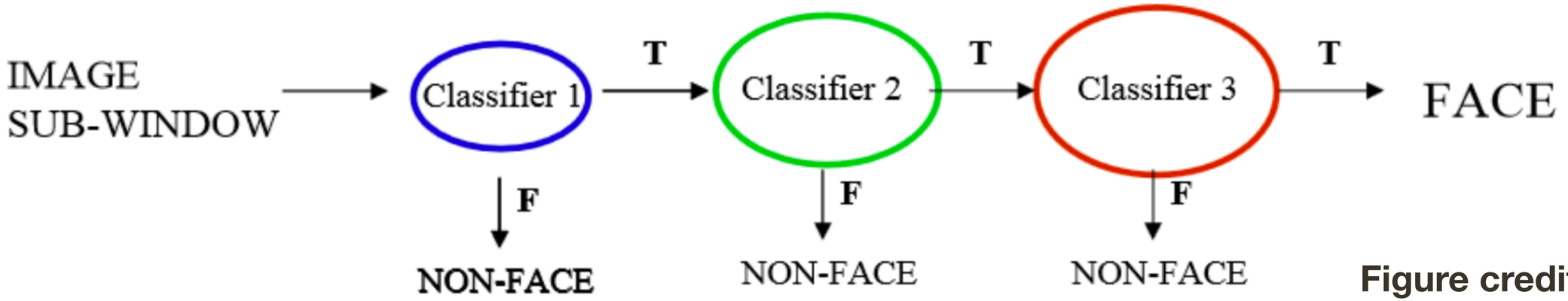


Figure credit: P. Viola

To make detection **faster**, features can be reordered by increasing complexity of evaluation and the thresholds adjusted so that the early (simpler) tests have few or no false negatives

Any window that is rejected by early tests can be discarded quickly without computing the other features

This is referred to as a **cascade** architecture

Cascading Classifiers

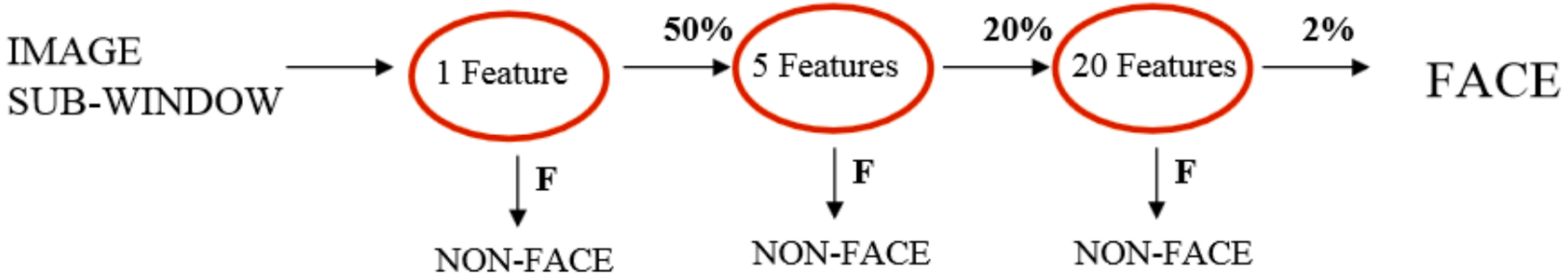


Figure credit: P. Viola

A **classifier** in the cascade is not necessarily restricted to a single feature

Example: Face Detection

Viola & Jones algorithm

3. The final strong classifier is

$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

$$\alpha_t = \log \frac{1}{\beta_t}$$

The final strong classifier is a weighted linear combination of the T weak classifiers where the weights are inversely proportional to the training errors

Image Credit: Ioannis (Yannis) Gkioulekas (CMU)

Example: Face Detection Summary

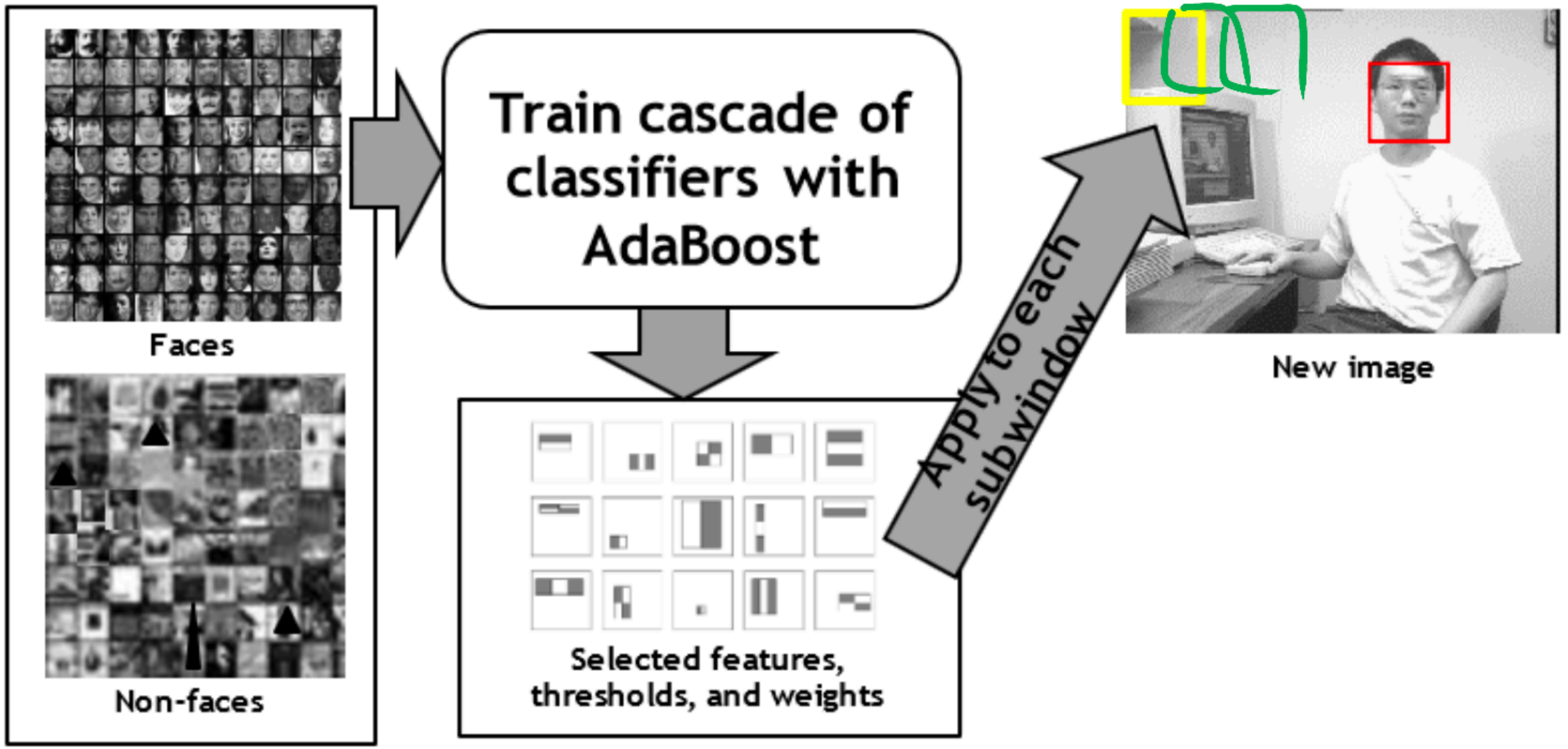


Figure credit: K. Grauman

Hard Negative Mining

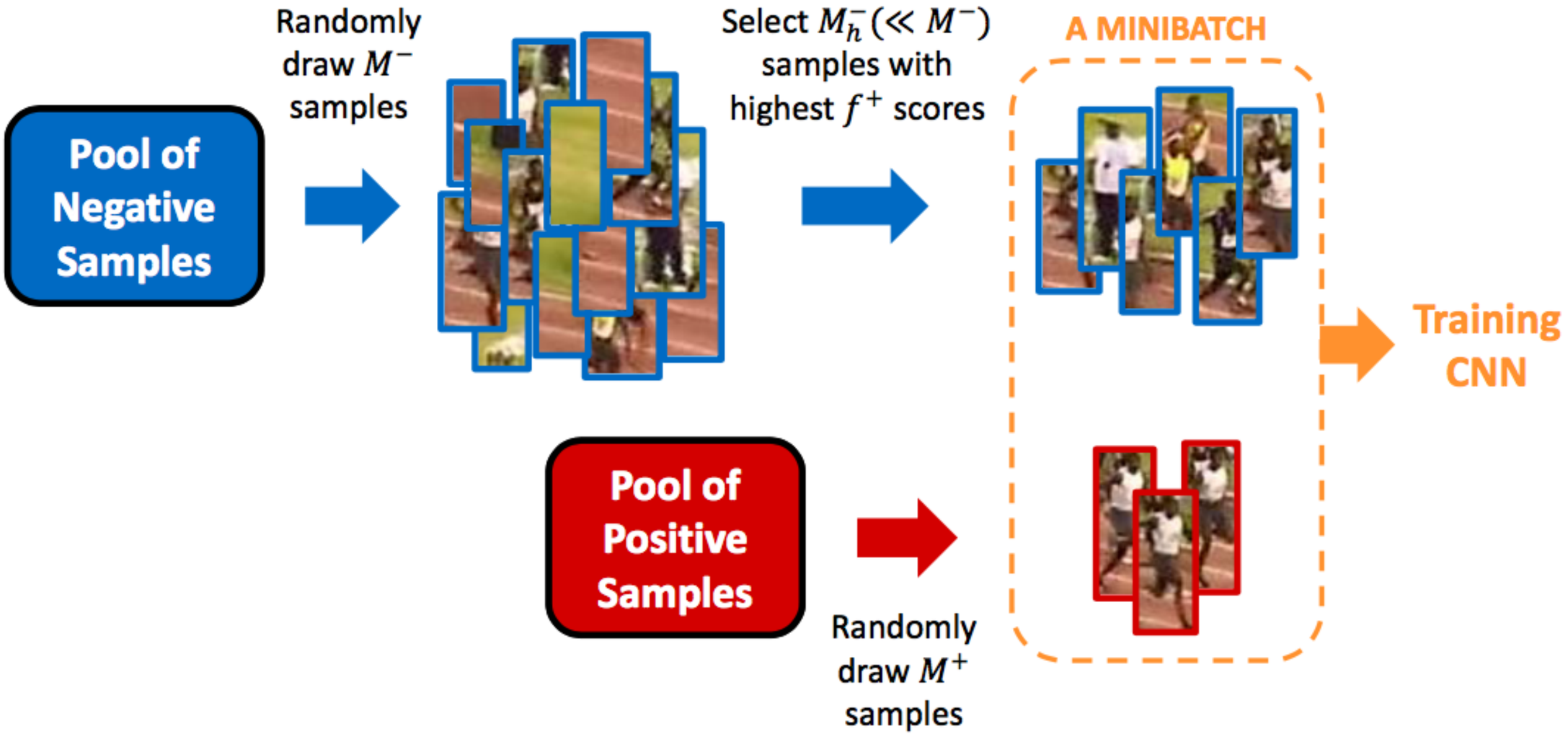
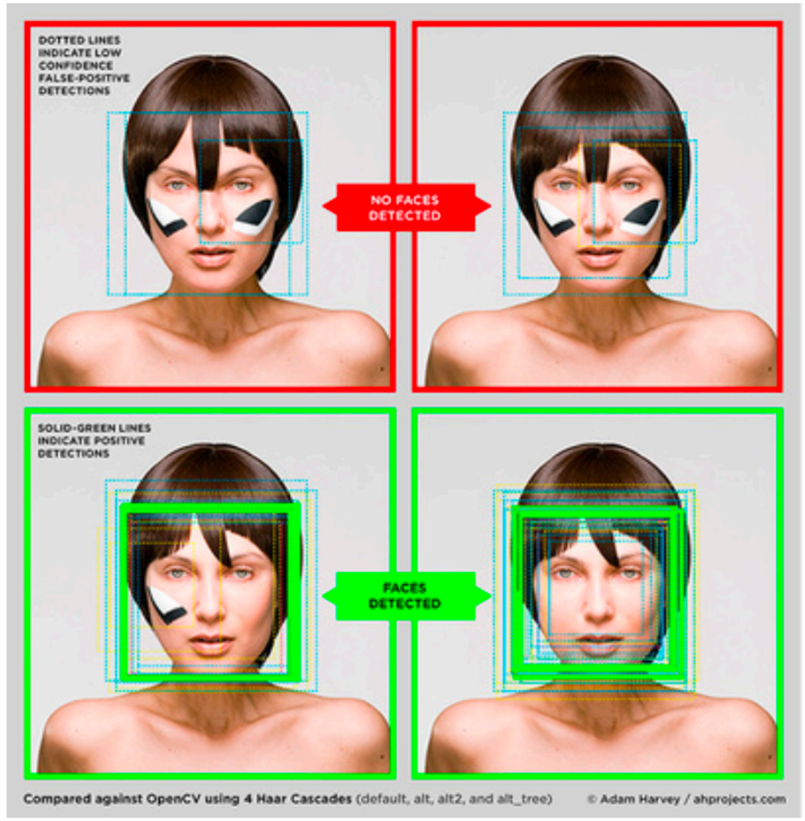


Image From: Jamie Kang

Example: Face Detection

Just for fun:



"CV Dazzle, a project focused on finding fashionable ways to thwart facial-recognition technology"

Figure source: Wired, 2015

Recall: Sliding Window

Train an image classifier as described previously. ‘Slide’ a fixed-sized detection window across the image and evaluate the classifier on each window.

Image credit: KITTI Vision Benchmark

Recall: Sliding Window

Train an image classifier as described previously. ‘Slide’ a fixed-sized detection window across the image and evaluate the classifier on each window.

Image credit: KITTI Vision Benchmark

This is a lot of possible windows! And most will not contain the object we are looking for.

Object Proposals

Object proposal algorithms generate a short list of regions that have generic object-like properties

— These regions are likely to contain some kind of foreground object instead of background texture

The object detector then considers these candidate regions only, instead of exhaustive sliding window search

Object Proposals

First introduced by Alexe et al., who asked ‘what is an object?’ and defined an ‘objectness’ score based on several visual cues

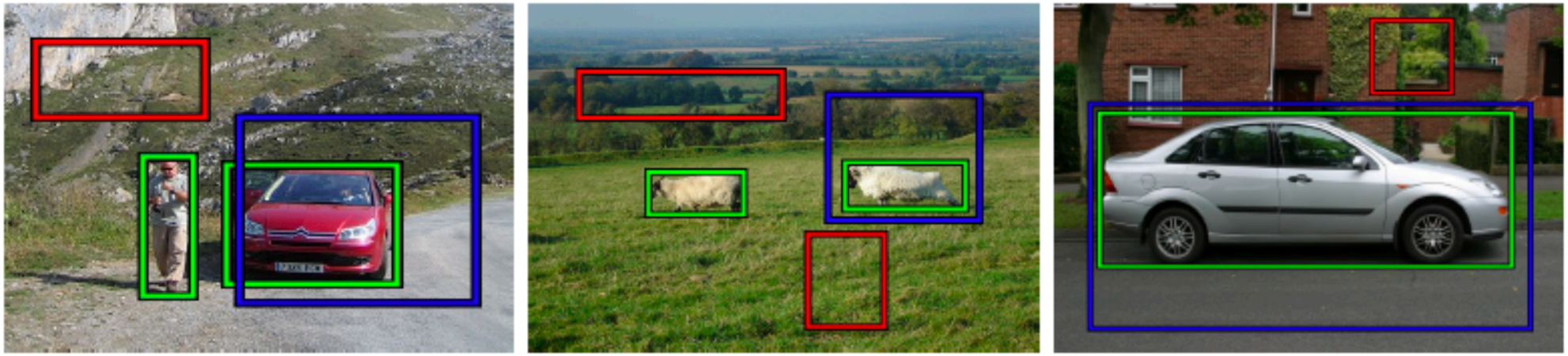


Figure credit: Alexe et al., 2012

Object Proposals

First introduced by Alexe et al., who asked ‘what is an object?’ and defined an ‘objectness’ score based on several visual cues

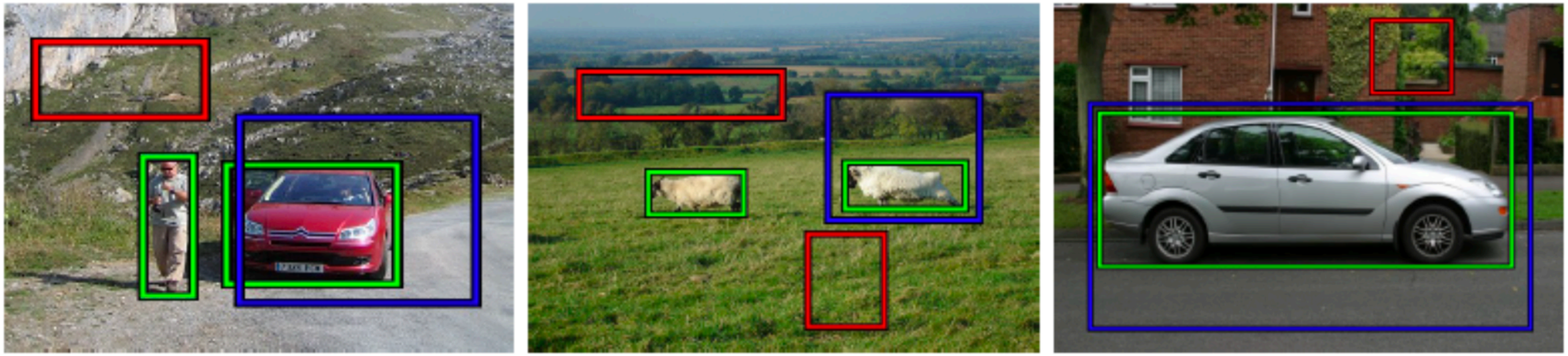


Figure credit: Alexe et al., 2012

This work argued that objects typically

— are unique within the image and stand out as salient

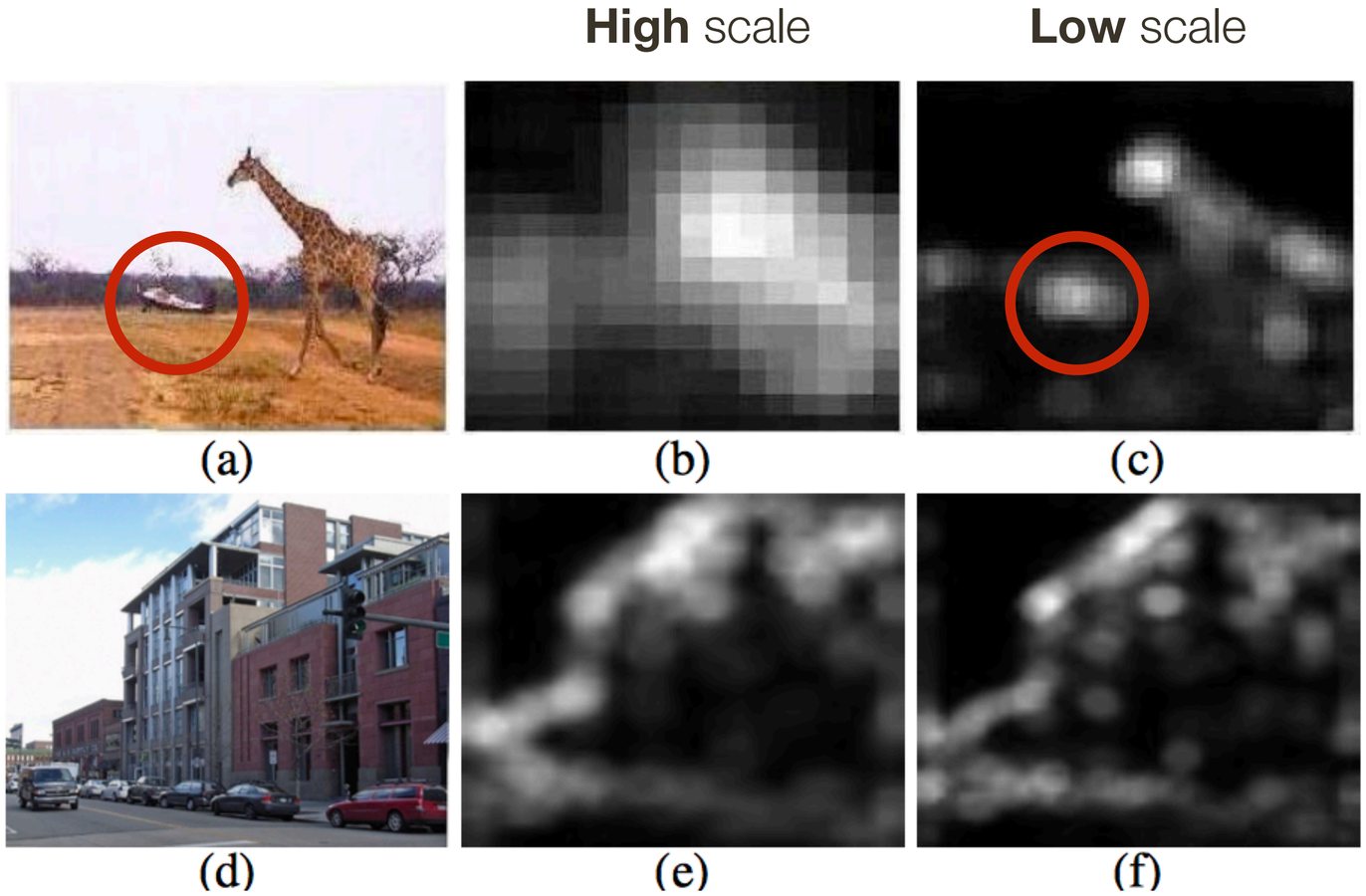
— have a contrasting appearance from surroundings and/or

— have a well-defined closed boundary in space

Object Proposals

Multiscale **Saliency**

— Favors regions with a unique appearance within the image



Successful Case

Failure Case

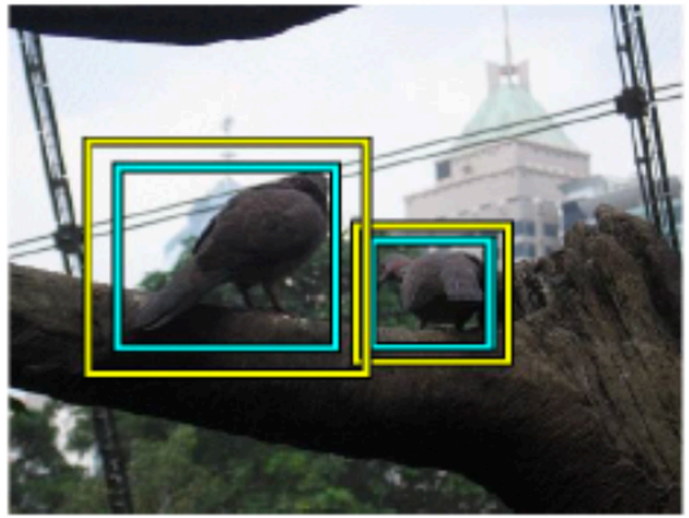
Figure credit: Alexe et al., 2012

Object Proposals

Colour Contrast

— Favors regions with a contrasting colour appearance from immediate surroundings

Successful Cases



Failure Case

Figure credit: Alexe et al., 2012

Object Proposals

Superpixels Straddling

— Favors regions with a well-defined closed boundary

— Measures the extent to which superpixels (obtained by image segmentation) contain pixels both inside and outside of the window

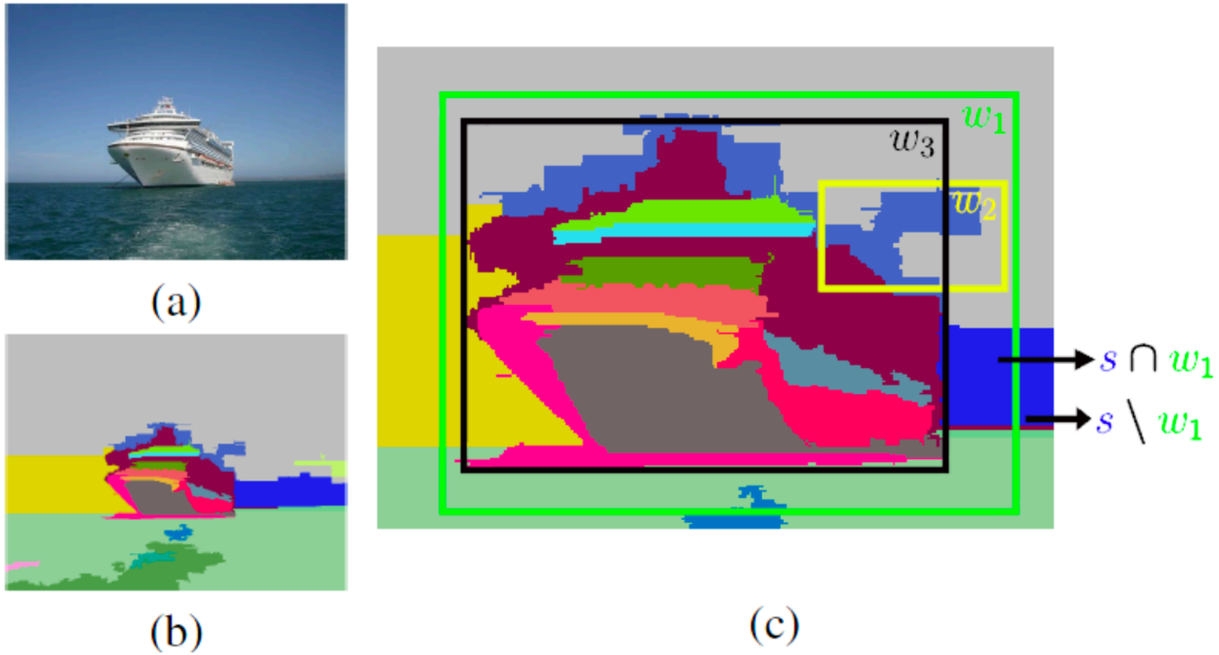


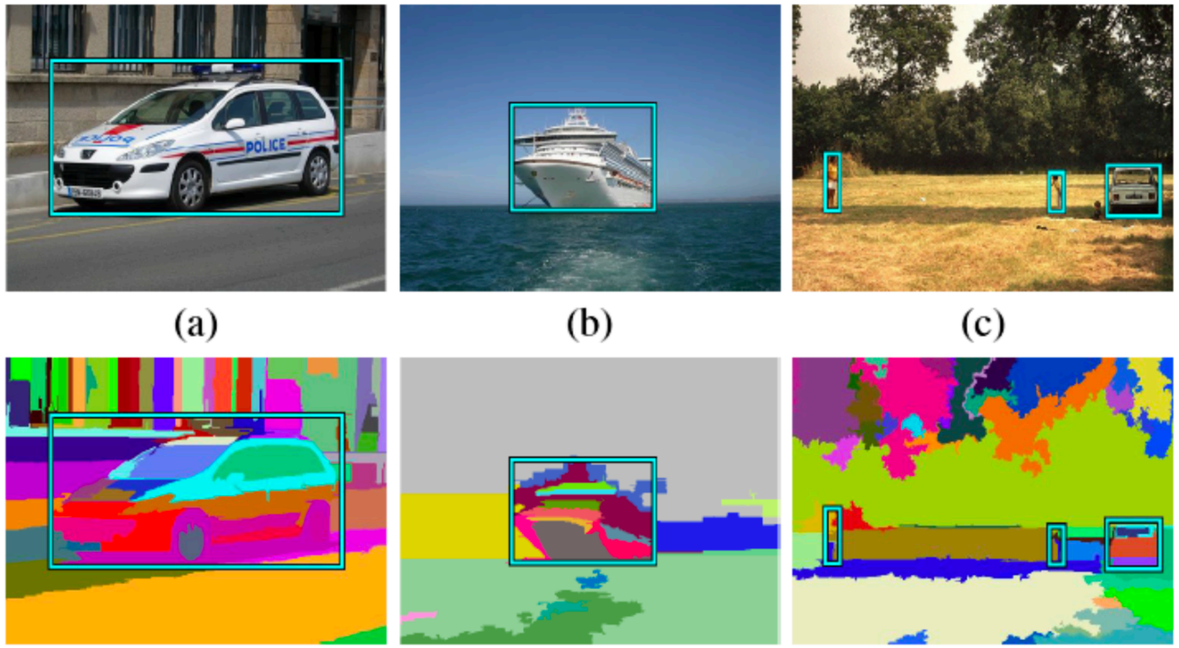
Figure credit: Alexe et al., 2012

Object Proposals

Superpixels Straddling

— Favors regions with a well-defined closed boundary

— Measures the extent to which superpixels (obtained by image segmentation) contain pixels both inside and outside of the window



Successful Cases **Failure** Case

Figure credit: Alexe et al., 2012

Object Proposals

TABLE 2: For each detector [11, 18, 33] we report its performance (left column) and that of our algorithm 1 using the same window scoring function (right column). We show the average number of windows evaluated per image #win and the detection performance as the mean average precision (mAP) over all 20 classes.

	[11]	OBJ- [11]	[18]	OBJ- [18]	ESS-BOW[33]	OBJ-BOW
mAP	0.186	0.162	0.268	0.225	0.127	0.125
#win	79945	→ 1349	18562	→ 1358	183501	→ 2997

Table credit: Alexe et al., 2012

Speeding up [11] HOG pedestrian detector [18] Deformable part model detector [33] Bag of words detector

Summary

Detection scores in the deformable part model are based on both appearance and location

The deformable part model is trained iteratively by alternating the steps

1. Assume components and part locations given; compute appearance and offset models
2. Assume appearance and offset models given; compute components and part locations

An object **proposal** algorithm generates a short list of regions with generic object-like properties that can be evaluated by an object detector in place of an exhaustive sliding window search