Laboratory 7: Object Detection with Cascade Classifier and DL

Computer Vision 2018

P.Zanuttigh

Object Detection

- Find objects in an image or video
  - This LAB: find the cars in the video
- In LAB6 solved by matching SIFT or ORB features
- In this LAB two alternative approaches

1. Cascade classification (Viola and Jones) (train and test)
2. Deep Learning (YOLO object detector) (only test, pre-trained)

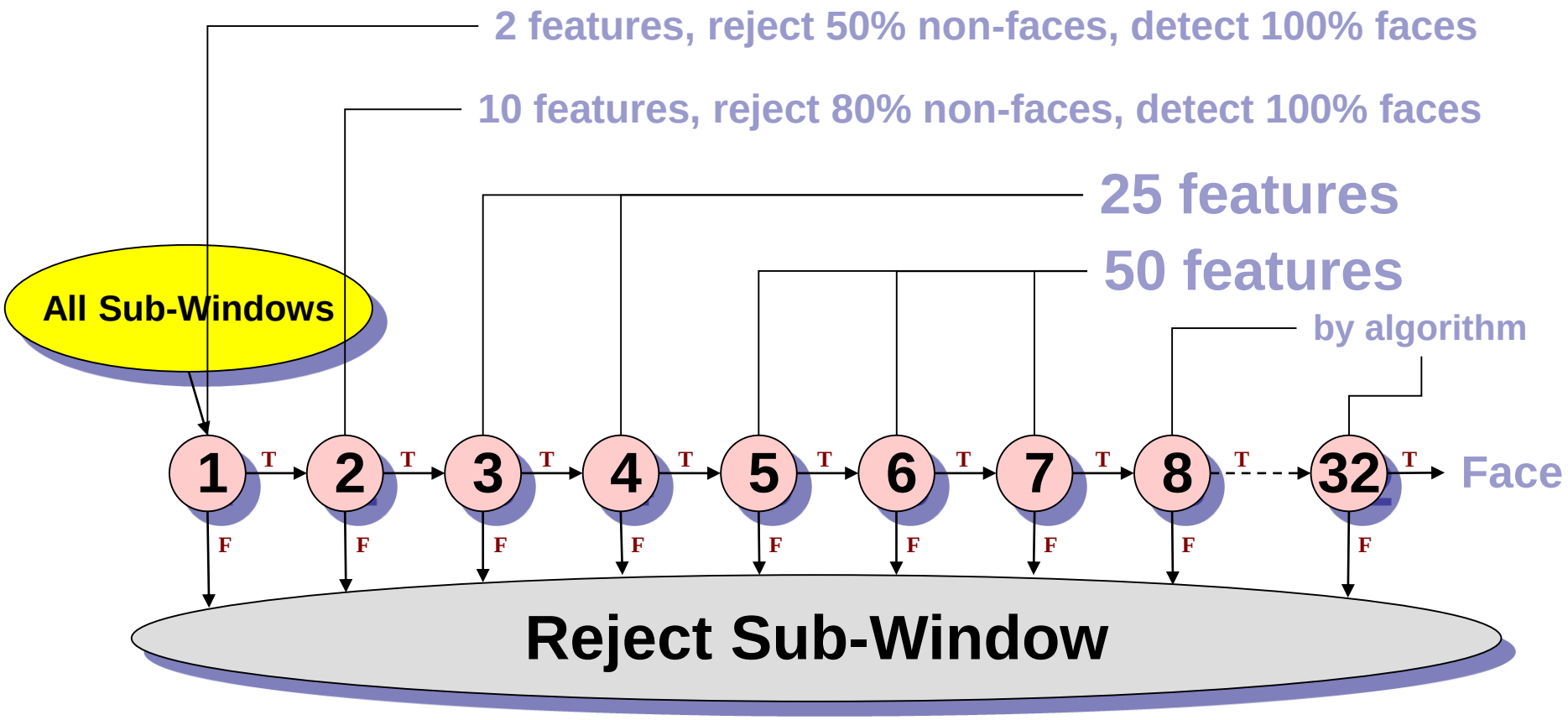
The Viola/Jones Face Detector

- Very widely used and well-known approach to real-time object detection
  - Originally developed for faces but it can detect any object characterized by a repeating pattern of bright and dark shades
- Key ideas
  - *Integral images* for fast feature evaluation
  - *Boosting* for feature selection
  - *Cascade of classifiers* for fast rejection of non-face windows
- See Szeliski's book (pages 663-666) for a detailed description

P. Viola and M. Jones. ***Rapid object detection using a boosted cascade of simple features.*** **CVPR 2001.**

P. Viola and M. Jones. ***Robust real-time face detection.*** **IJCV 57(2), 2004.**

Cascade Classifier



Use a Cascade Classifier

- Train and test the classifier
- OpenCV cascade classifier: implements (roughly) the Viola and Jones Algorithm
- Training:
  - Use the *opencv_traincascade* application
  - Requires positive and negative examples (already provided in the "*positives_all.vec*" and "*negatives_all.txt*" files)
- Testing
  - Use the cv::detectMultiScale function

opencv_traincascade: params

Parameters (from command line):

- -data <output folder>
- -vec <your positives_vec file>
- -bg <your_negative file>
- -numStages <number of stages of the cascade classifier>
  - Start with a small value (low performance but fast), then increase
- -w (width rescaled sample, usually 24)
- -h (height of the rescaled sample, usually 24)
- -NumPos (positives extracted from the positives file each time
  - It should be < than the total number of positives (e.g., 1000)
- -NumNeg (negatives extracted from the negatives file each time
  - It should be < than the total number of negatives (e.g., 1370)

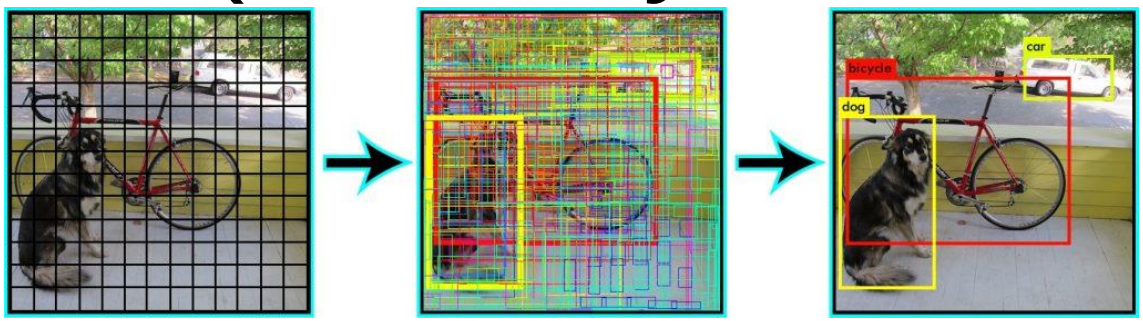
Format for parameter passing:

- opencv_traincascade : -param_name param_value
- lab_executable: -param_name="param_value"

cv::detectMultiScale

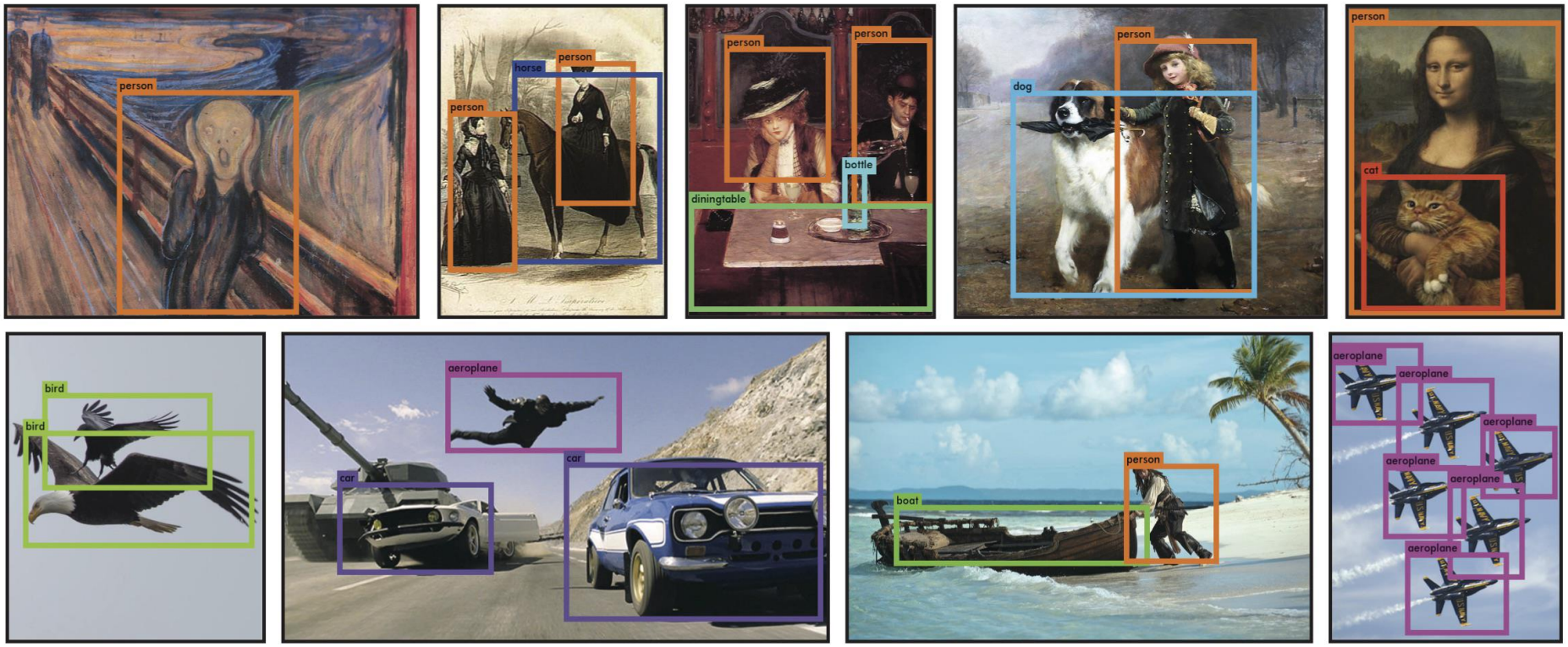
void cv::CascadeClassifier::detectMultiScale	(	InputArray	image,	Input Image
		std::vector < Rect > &	objects,	Bounding boxes of detected objects
		double	scaleFactor = 1.1,	Ratio between two consecutive scale levels
		int	minNeighbors = 3,	Keep only where multiple detection happens
		int	flags = 0,	
		Size	minSize = Size(),	Min object size
		Size	maxSize = Size()	Max object size
	)			

YOLO (You only Look Once)



- YOLO models detection is a simple regression problem which takes an input image and learns the class probabilities and bounding box coordinates
- YOLO divides each image into a grid of S x S and each location predicts N bounding boxes and confidence. The confidence reflects the accuracy of the bounding box and whether the bounding box actually contains an object
- YOLO also predicts the classification score for each box for every class
- A total of SxSxN boxes are predicted. However, most of these boxes have low confidence scores and we can apply a threshold
- YOLO runs the CNN only once and can be run real time
- One limitation for YOLO is that it only predicts 1 type of class in one grid location, hence it struggles with very small objects
- For the LAB we use YOLOv2 (called also YOLO9000)
  - Improved version (in particular better multi-scale processing)

YOLO: Examples



Deep Learning in OpenCV

- No training functions (perform training with external DL libraries like TensorFlow, Keras, Caffe, etc)
- Load pre-trained networks and use on your data
- For the LAB the *"yolo-voc.cfg"* file contains the YOLO network architecture (have a look at it!)
- The *"yolo-voc.weights"* contains the weights
- Set the confidence value in the code and in the "*yolo-voc.cfg"* file