



EVROPSKÁ UNIE  
Evropské strukturální a investiční fondy  
Operační program Výzkum, vývoj a vzdělávání



# Image Analysis II

Radovan Fusek



# Intro

The most common tasks in image processing:

- Enhancement/Filtering



EVROPSKÁ UNIE  
Evropské strukturální a investiční fondy  
Operační program Výzkum, vývoj a vzdělávání



MINISTERSTVO ŠKOLSTVÍ,  
MLÁDEŽE A TĚLOVÝCHOVY

# Filtering





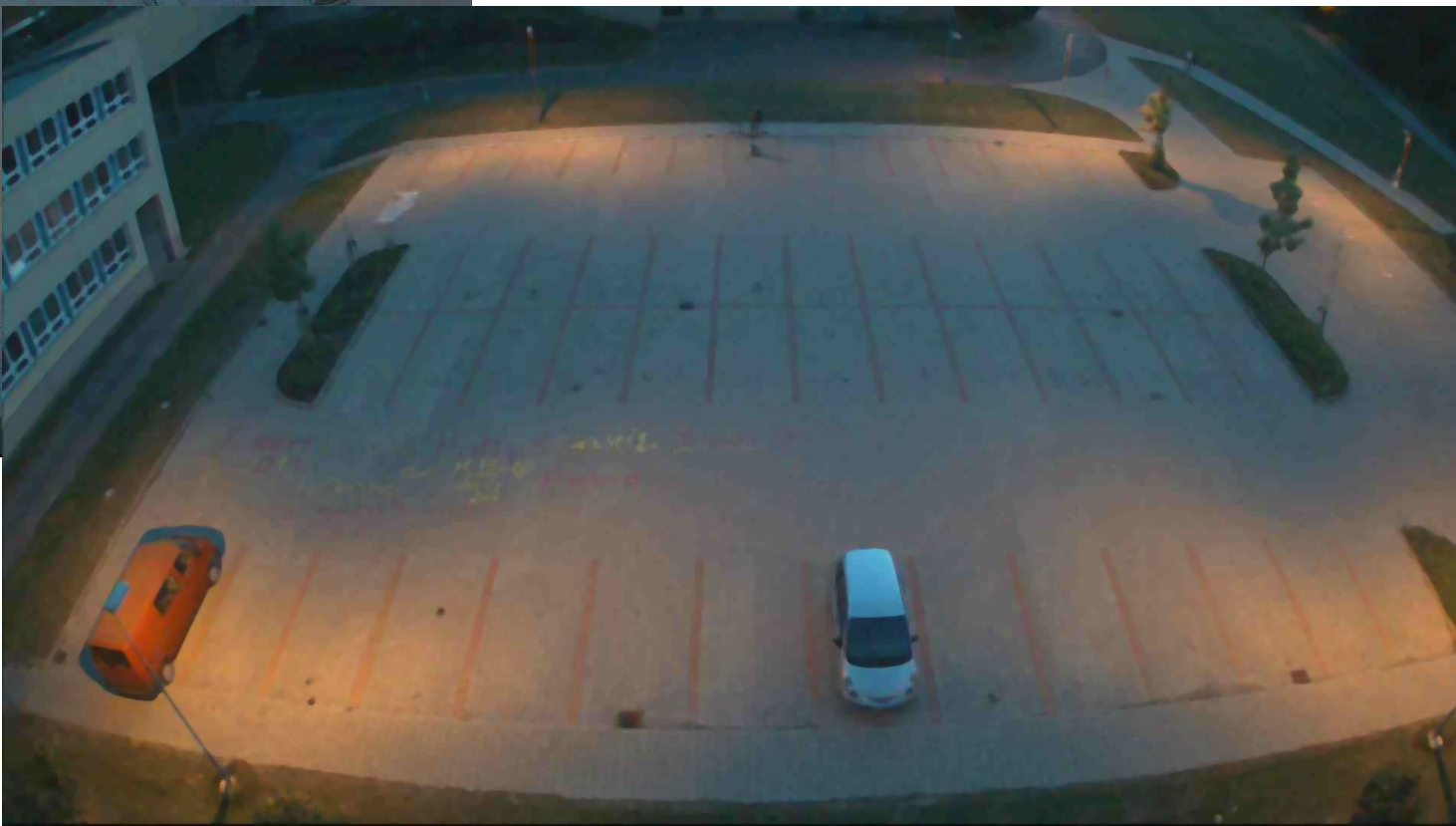
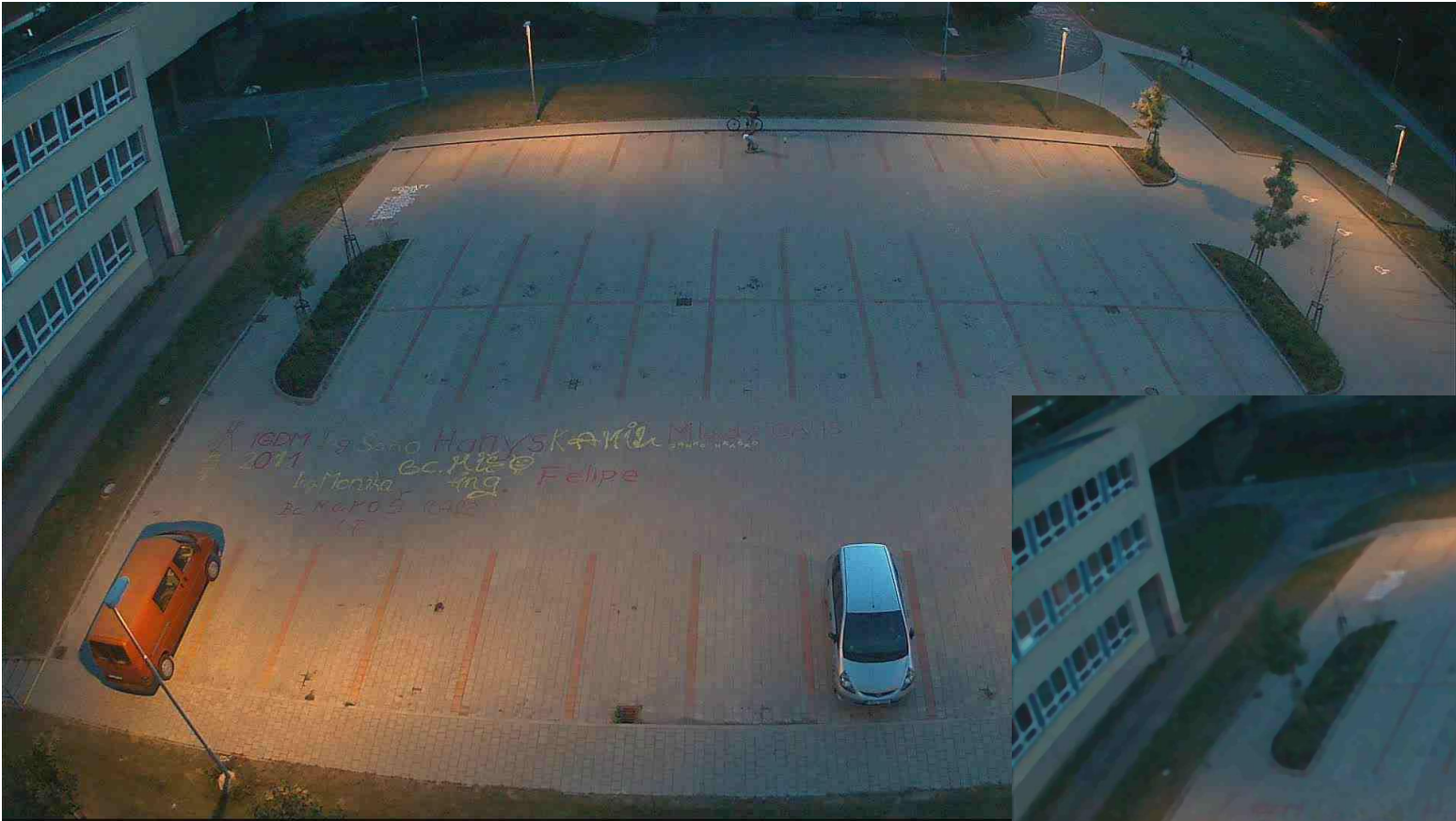


EVROPSKÁ UNIE  
Evropské strukturální a investiční fondy  
Operační program Výzkum, vývoj a vzdělávání



MINISTERSTVO ŠKOLSTVÍ,  
MLÁDEŽE A TĚLOVÝCHOVY

# Filtering







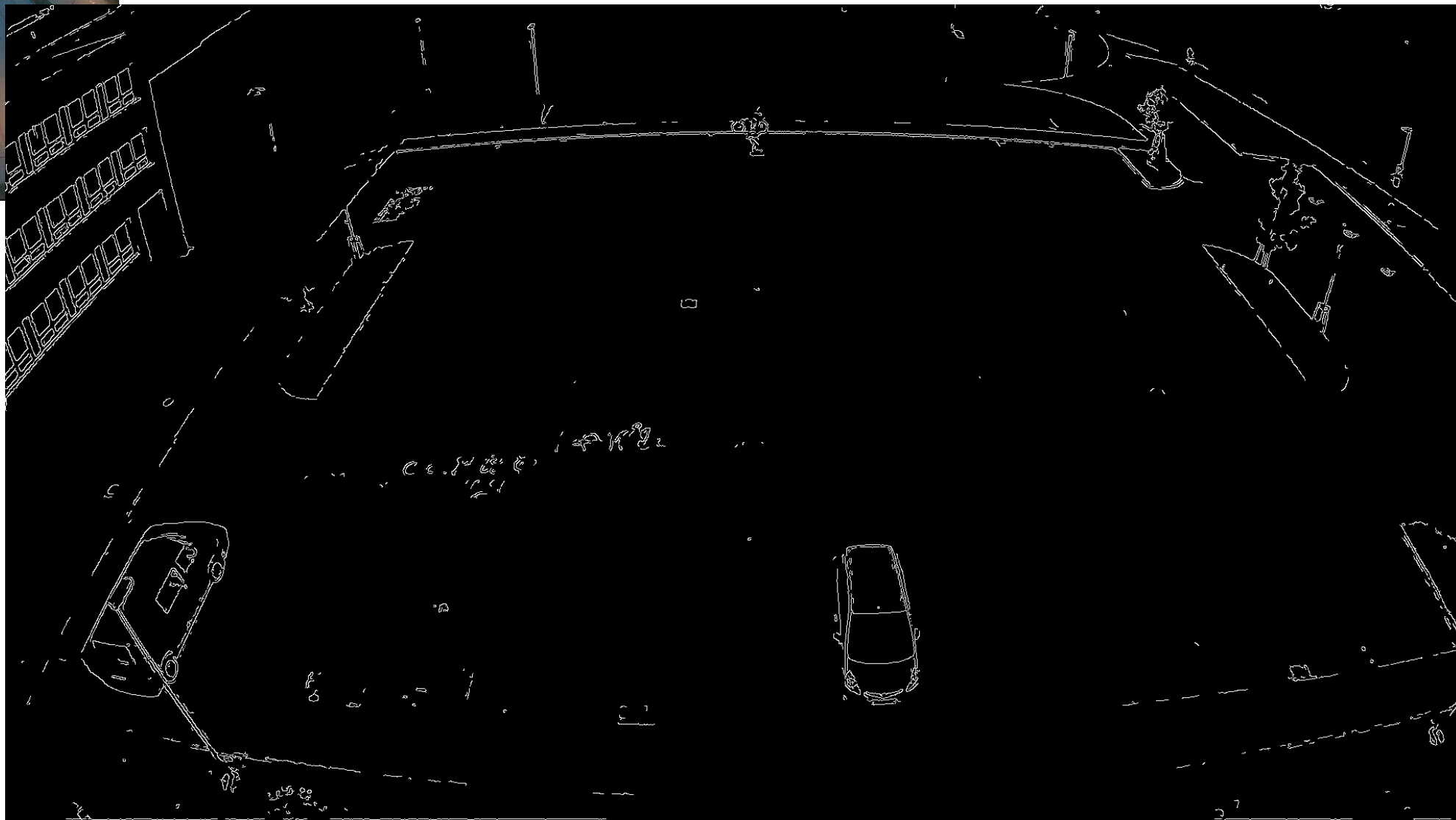
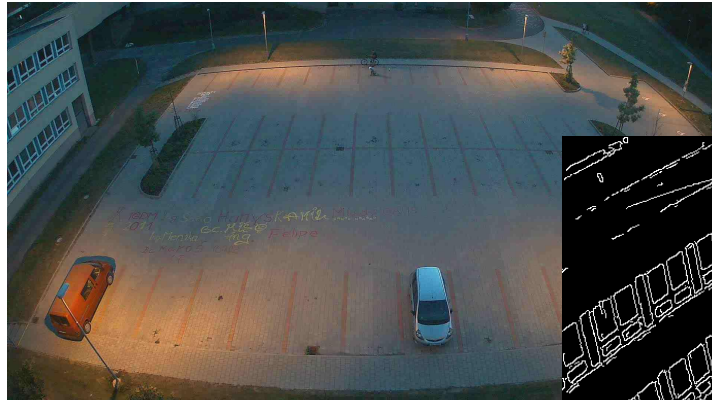
# Intro

The most common tasks in image processing:

- Enhancement/Filtering
- Edge Detection



# Edge Detection





# Edge Detection

## Video Example





# Intro

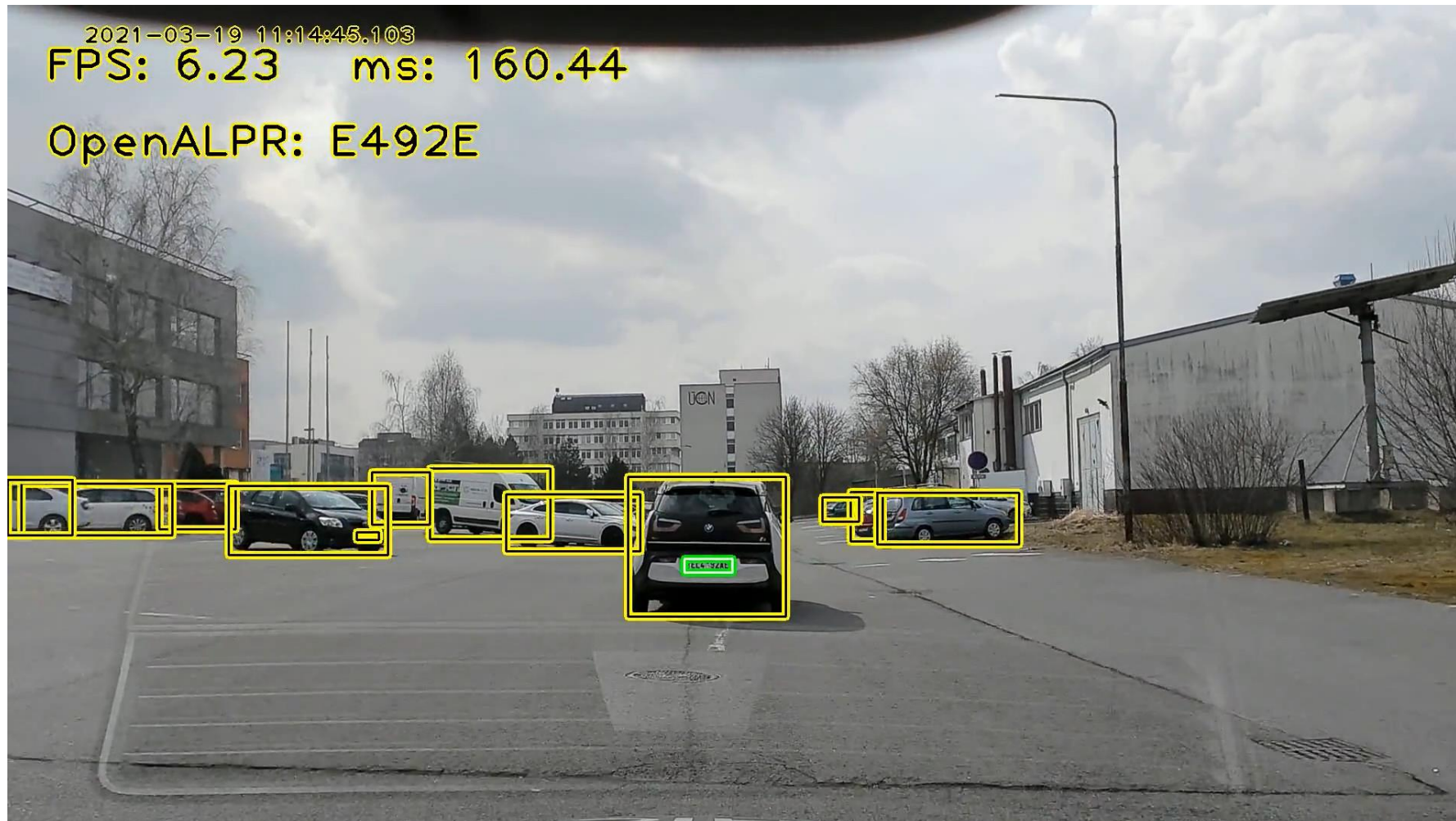
The most common tasks in image processing:

- Enhancement/Filtering
- Edge Detection
- Object Detection

# Object Detection

What is goal of object detection methods?

- position of the objects (coordinates) + scale of the objects





# Object Detection

## Video Example





# Intro

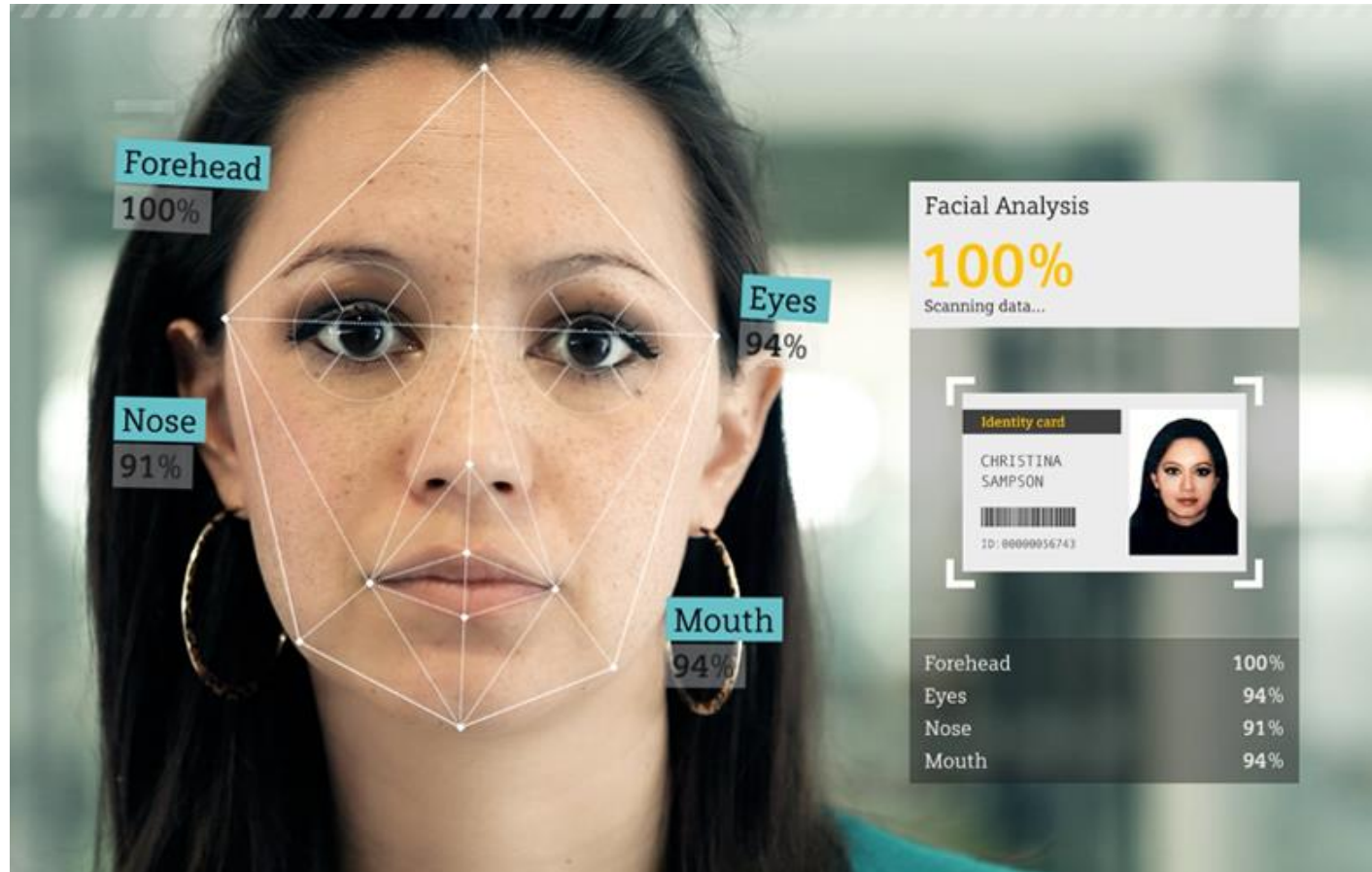
The most common tasks in image processing:

- Enhancement/Filtering
- Edge Detection
- Object Detection
- Object Recognition

# Object Recognition

What is goal of object recognition methods?

- identifying objects (categories)
- e.g. face recognition



<https://recfaces.com/articles/how-accurate-is-facial-recognition>



# Intro

The most common tasks in image processing:

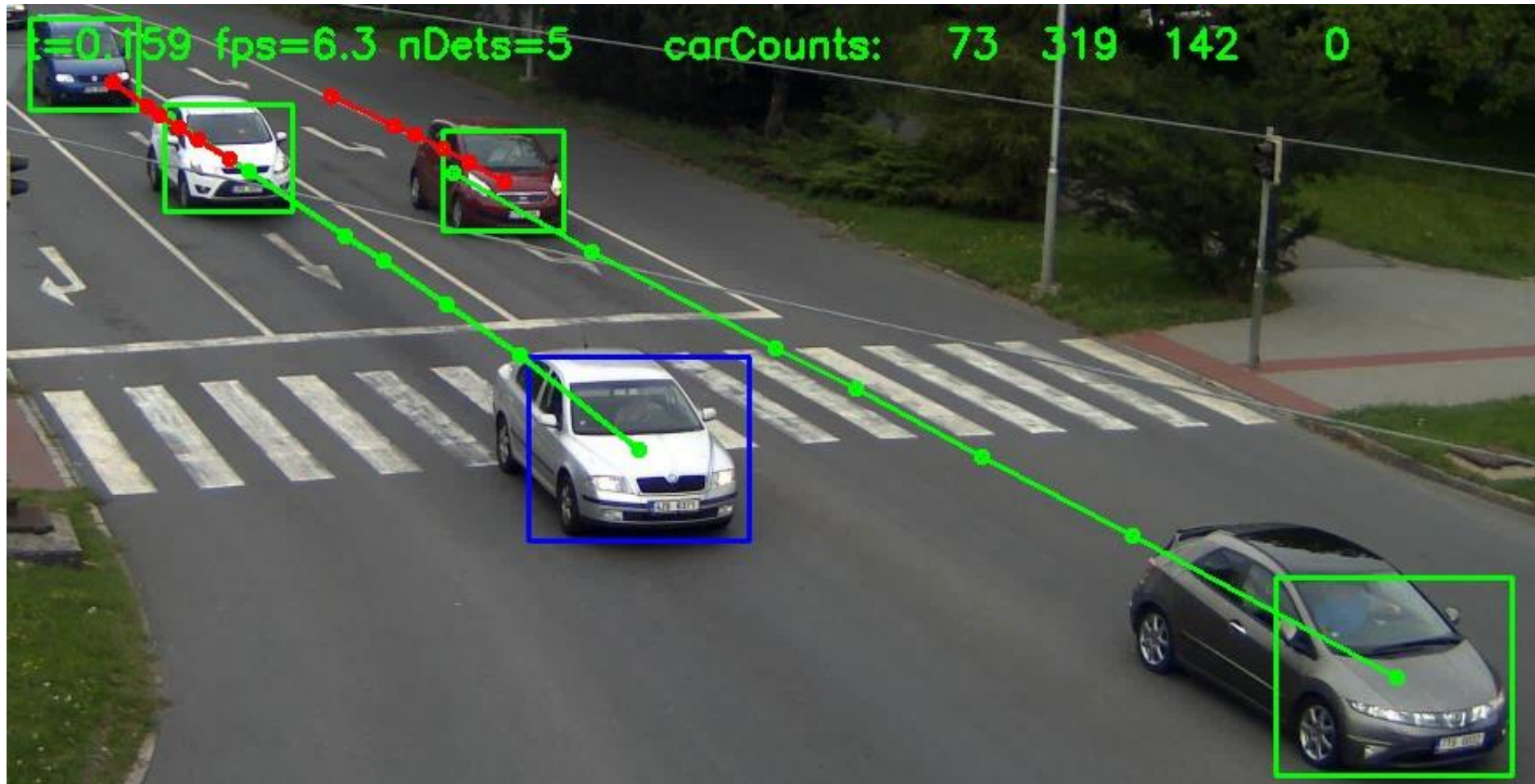
- Enhancement/Filtering
- Edge Detection
- Object Detection
- Object Recognition
- Object Tracking



# Object Tracking

## Object Tracking

- e.g. estimation of trajectory/unique ID to each tracked object





# Intro

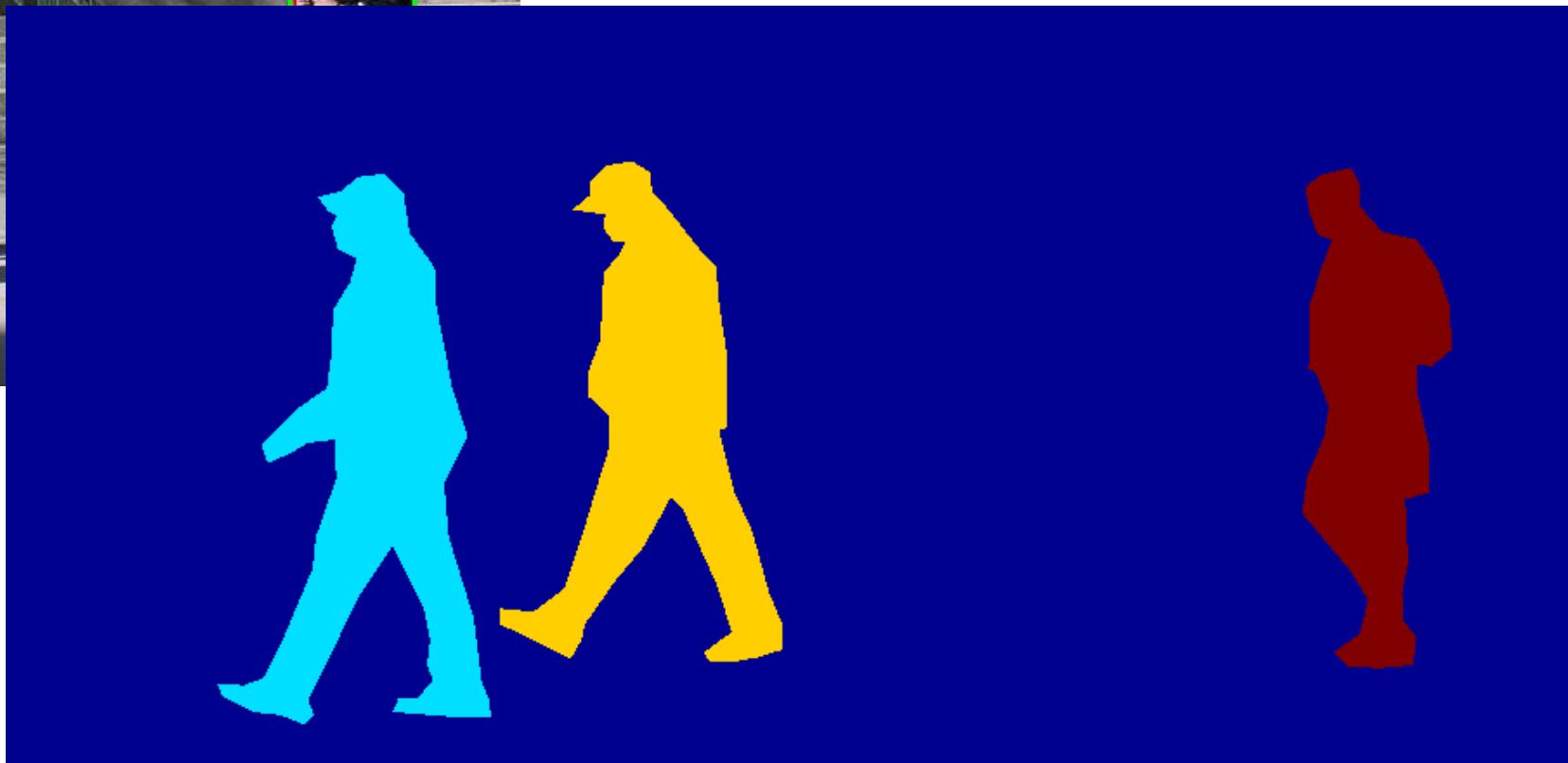
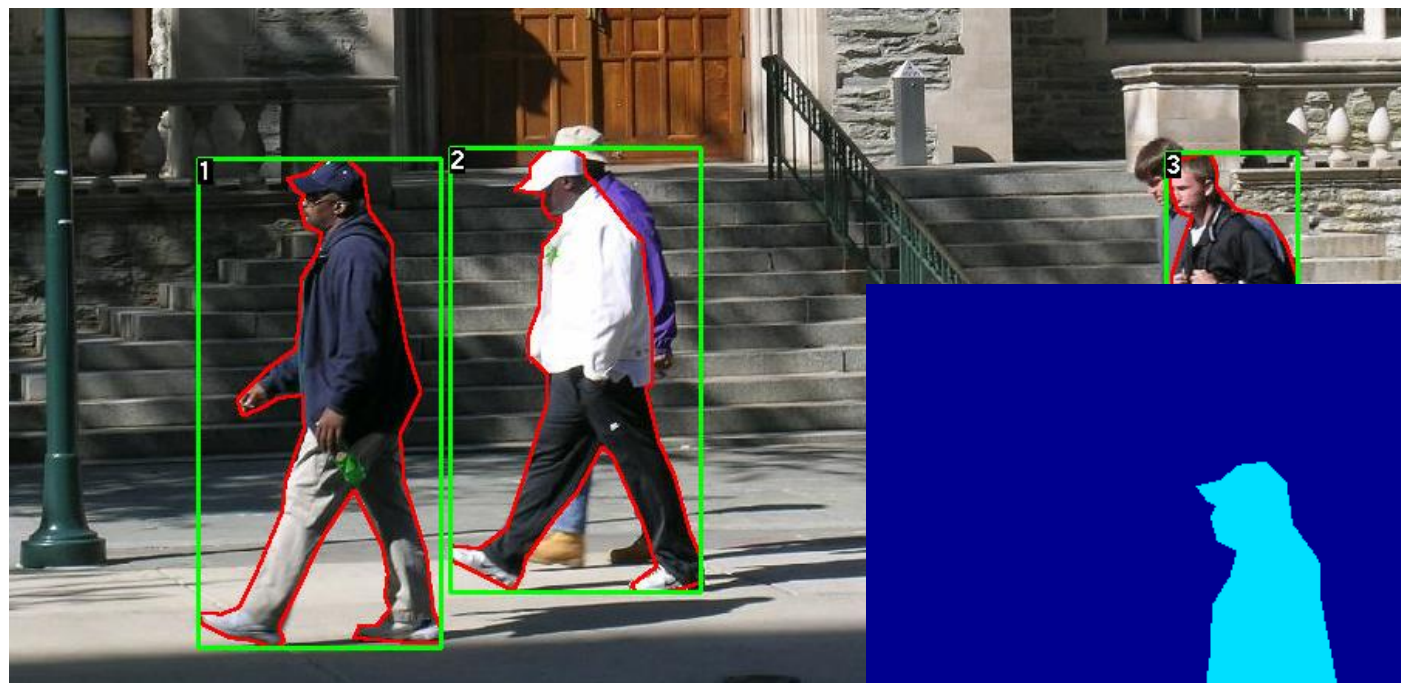
The most common tasks in image processing:

- Enhancement/Filtering
- Edge Detection
- Object Detection
- Object Recognition
- Object Tracking
- Segmentation



# Object Segmenation

Image is partitioned into multiple regions







# Object Segmenation

Image is partitioned into multiple regions





# Object Segmentation

## Video Example



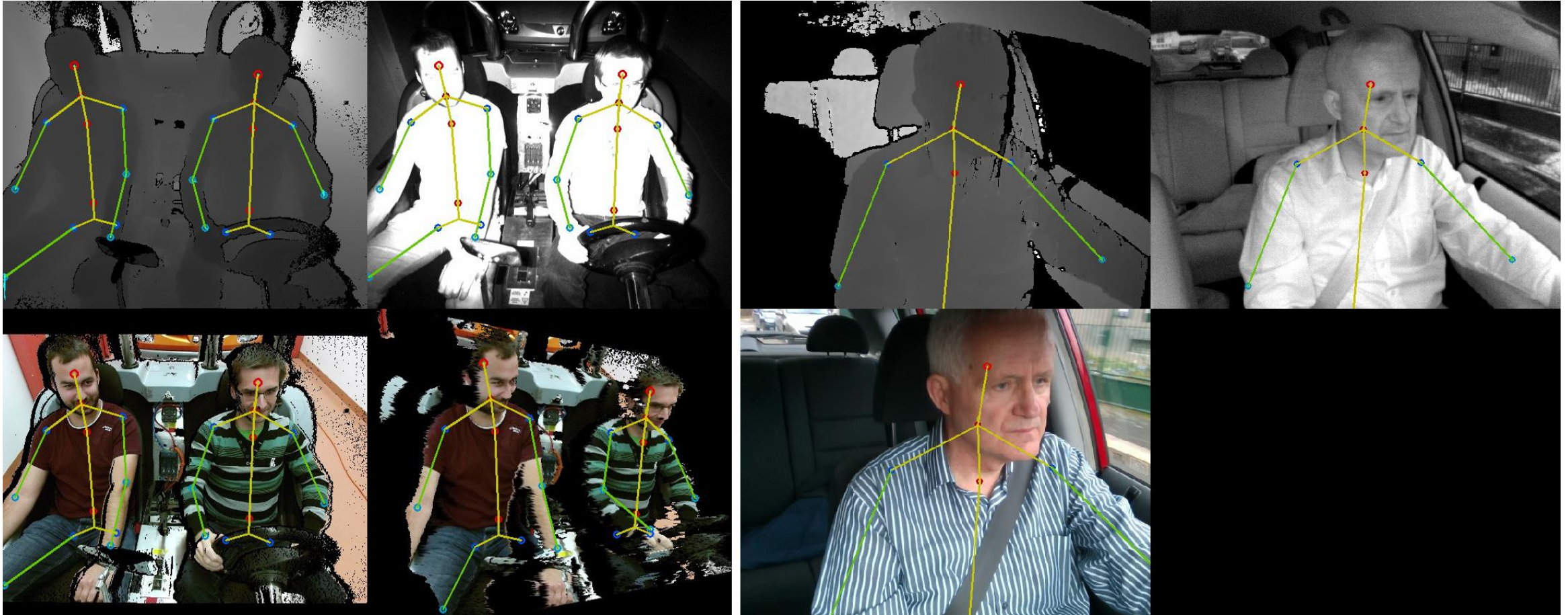
# Intro

The most common tasks in image processing:

- Enhancement/Filtering
- Edge Detection
- Object Detection
- Object Recognition
- Object Tracking
- Segmentation
- Human Pose Detection
- ... many others



# Pose Detection







# Checking the driver state

Openpose Library







# Pose Detection

## Video Example



EVROPSKÁ UNIE  
Evropské strukturální a investiční fondy  
Operační program Výzkum, vývoj a vzdělávání



# Object Detection (Analysis)

## Haar, Integral Image, AdaBoost

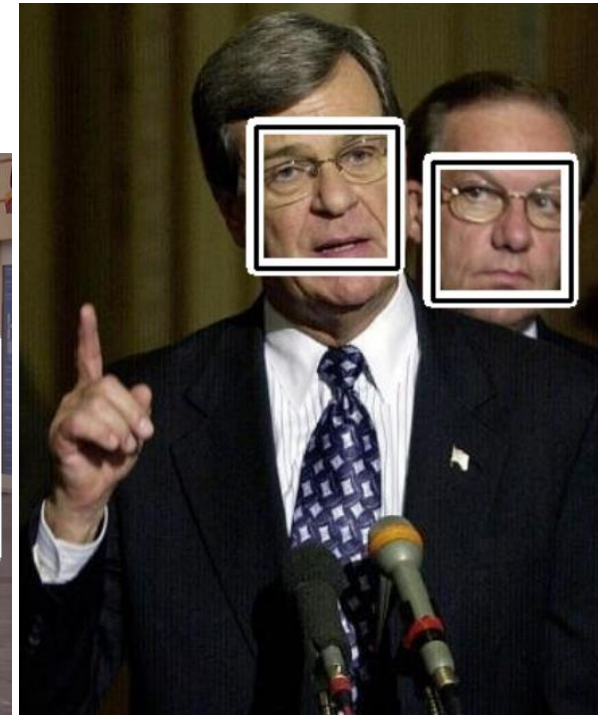
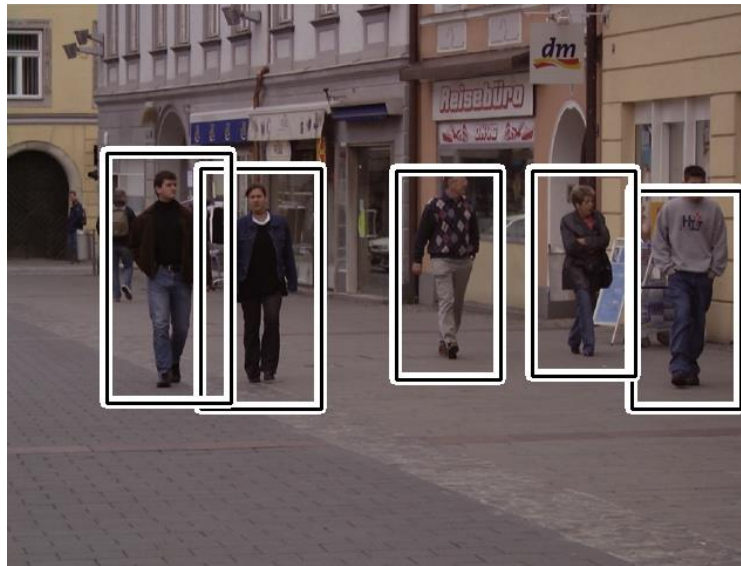


# What is Object Detection?

- It is clear that the images contain many objects of interest. The goal of the object detection systems is to find the location of these objects in the images (e.g. cars, faces, pedestrians).
- For example, the vehicle detection systems are crucial for traffic analysis or intelligent scheduling, the people detection systems can be useful for automotive safety, and the face detection systems are a key part of face recognition systems.

# What is Object Detection?

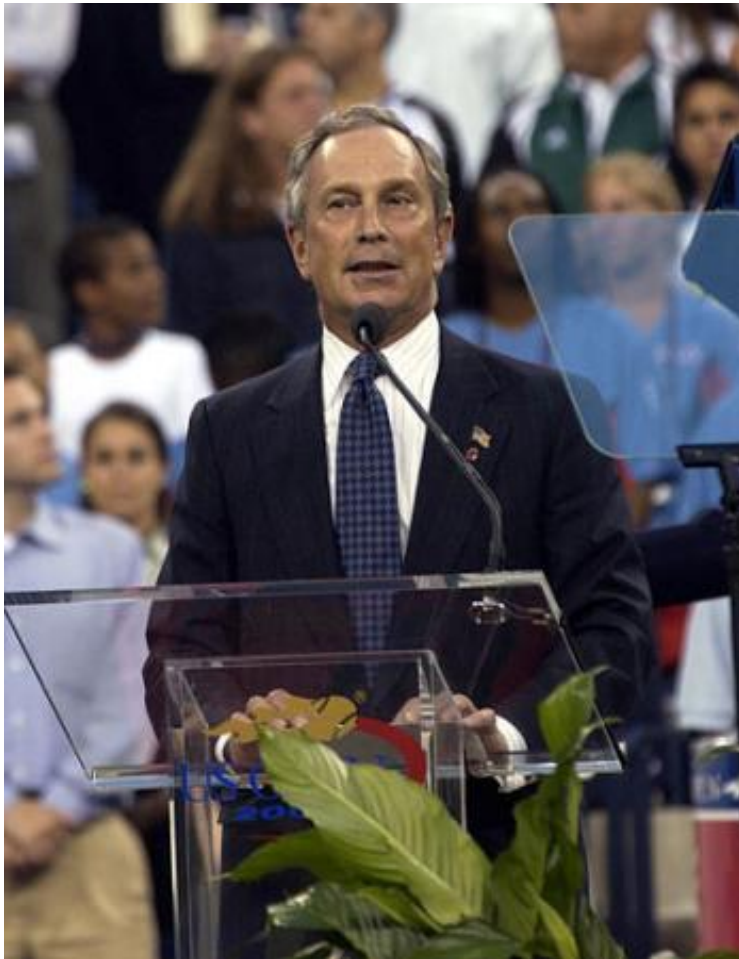
- Output?
  - position of the objects
  - scale of the objects





# Problems (Challenges)

- low quality images / different views







## Problems (Challenges)

- illumination + low quality





# Image Features

- The objects of interest can be described using various image information (e.g. shape, texture, colour). In the area of feature based detectors the image features are the carries of this information.
- Many methods for extracting the image features that are able to describe the appearance of objects were presented, especially, the detectors that are based on the histograms of oriented gradients (HOG), Haar features, or local binary patterns (LBP) are dominant and they are considered as the state-of-the-art methods.



# Object Detection/Recognition

- Haar

- HOG

- LBP

Traditional Approaches

- SIFT, SURF

KeyPoints

- CNNs

Deep Learning Approach



## Sliding Window - Main Idea

- In general, the sliding window technique represents the popular and successful approach for object detection. The main idea of this approach is that the input image is scanned by a rectangular window at multiple scales. The result of the scanning process is a large number of various sub-windows. A vector of features is extracted from each sub-window. The vector is then used as an input for the classifier (e.g. SVM classifier).
- During the classification process, some sub-windows are marked as the objects. Using the sliding window approach, the multiple positive detections may appear, especially around the objects of interest



## Sliding Window - Main Idea

- These detections are merged to the final bounding box that represents the resulting detection.
- The classifier that determines each sub-window is trained over the training set that consists of positive and negative images.
- The key point is to find what values (features) should be used to effectively encode the image inside the sliding window.





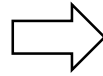
# Sliding Window - Main Idea



Constantine Papageorgiou and Tomaso Poggio: A Trainable System for Object Detection.  
*Int. J. Comput. Vision* 38, pp. 15-33. (2000)



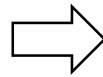
# Detection Process



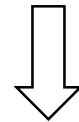
Feature Vector  
(gradient, HOG, LBP, ...)

Constantine Papageorgiou and Tomaso Poggio: A Trainable System for Object Detection.  
*Int. J. Comput. Vision* 38, pp. 15-33. (2000)

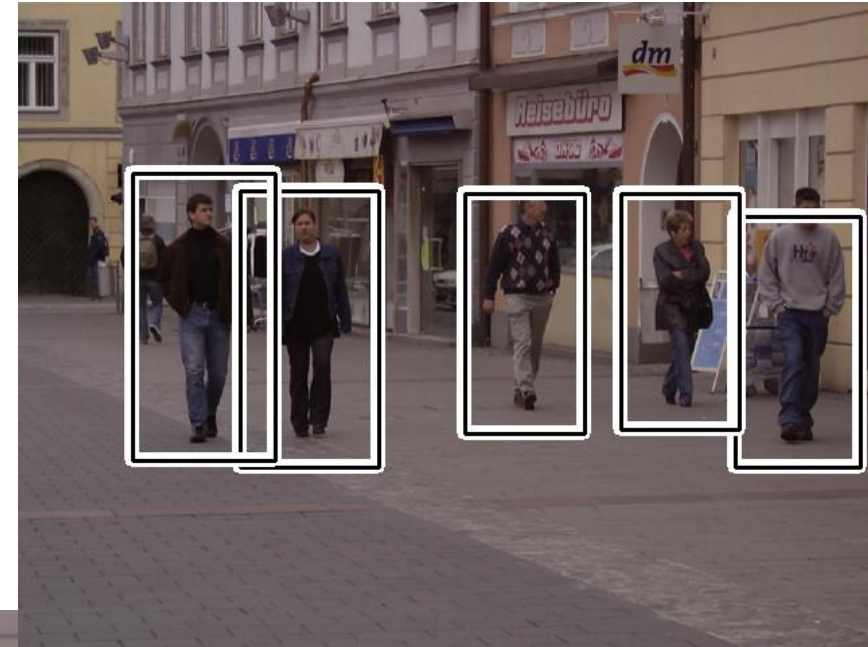
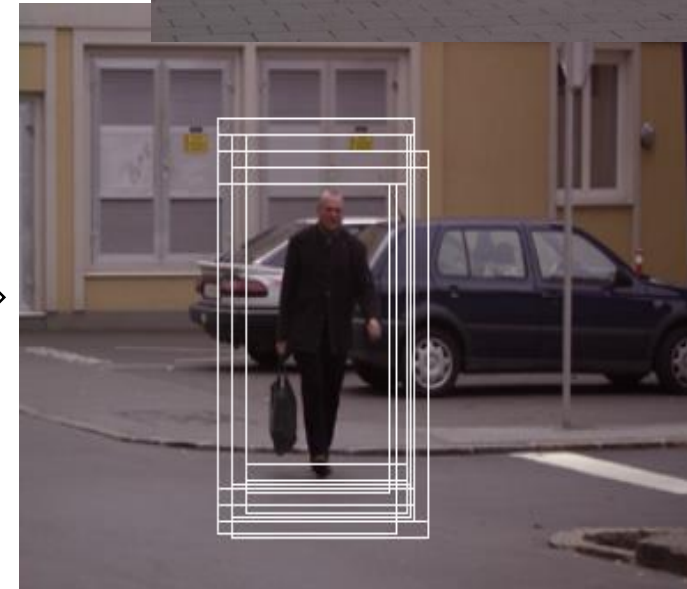
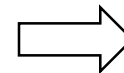
# Detection Process



Feature Vector  
(gradient, HOG, LBP, ...)



Trainable Classifier  
(SVM, ANNs, ...)



Constantine Papageorgiou and Tomaso Poggio: A Trainable System for Object Detection.  
*Int. J. Comput. Vision* 38, pp. 15-33. (2000)

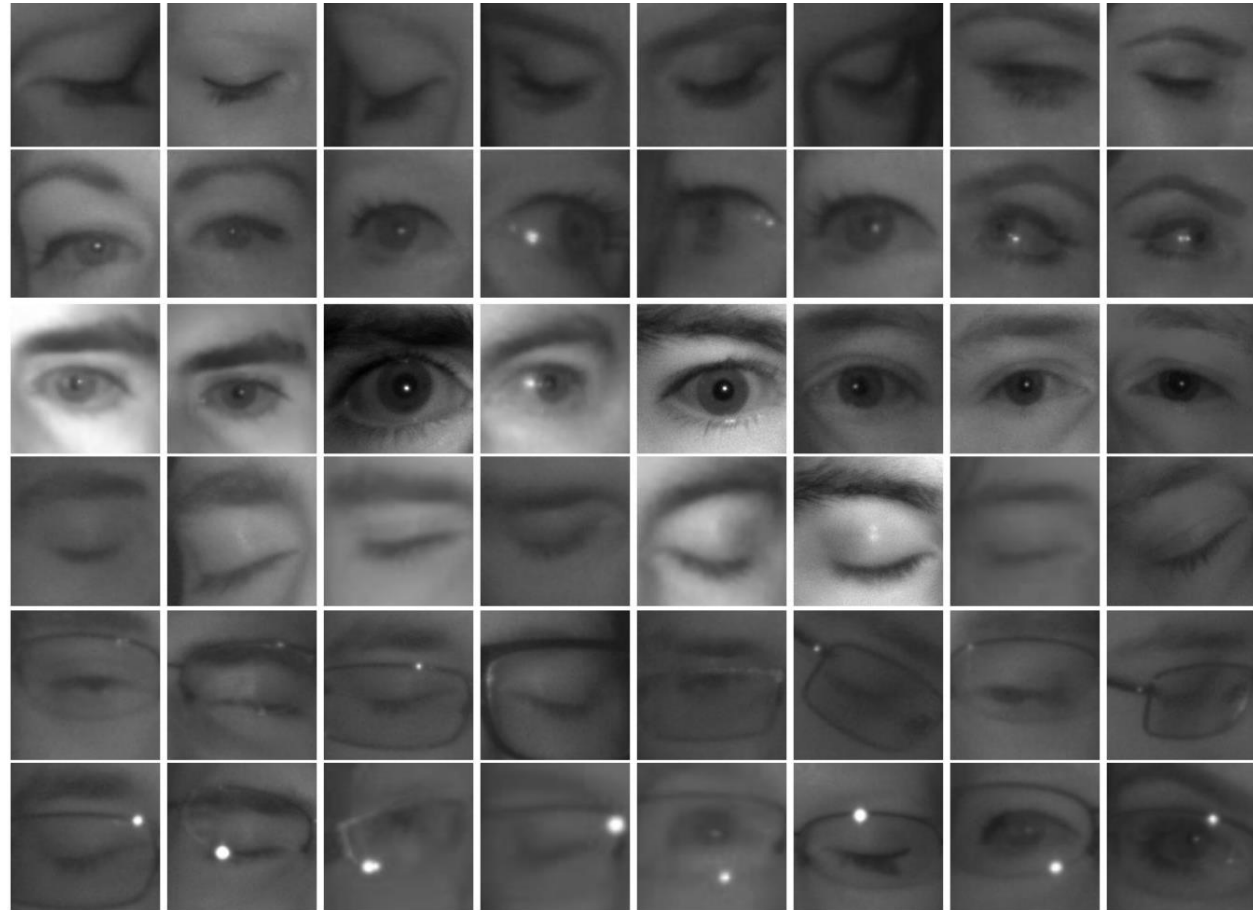


# Detection Process

- Typically, in the area of feature-based detectors, the detection algorithms consist of two main parts. The extraction of image features is the first part. The second part is created by the trainable classifiers that handle a final classification (object/non-object).
- The extraction of relevant features has a significant influence on the successfulness of detectors. The large number of features slows down the training and detection phases; on the other hand a very small number of features may not be able to describe the properties of object of interest.
- The quality of training set is also equally important.

# Generating Training Set

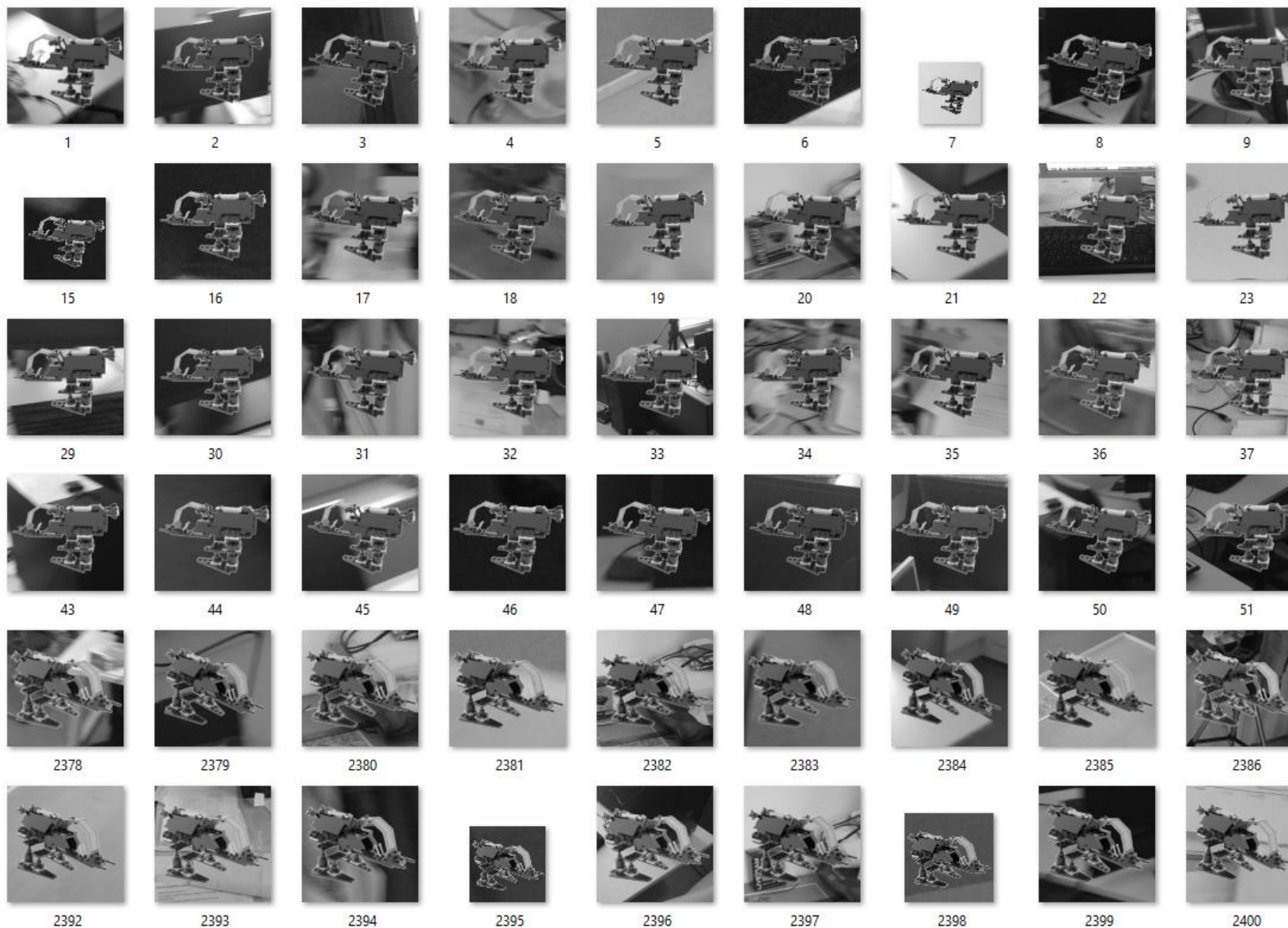
- negative set - without the object of interest
- positive set
  - rotation
  - noise
  - Illumination
  - scale





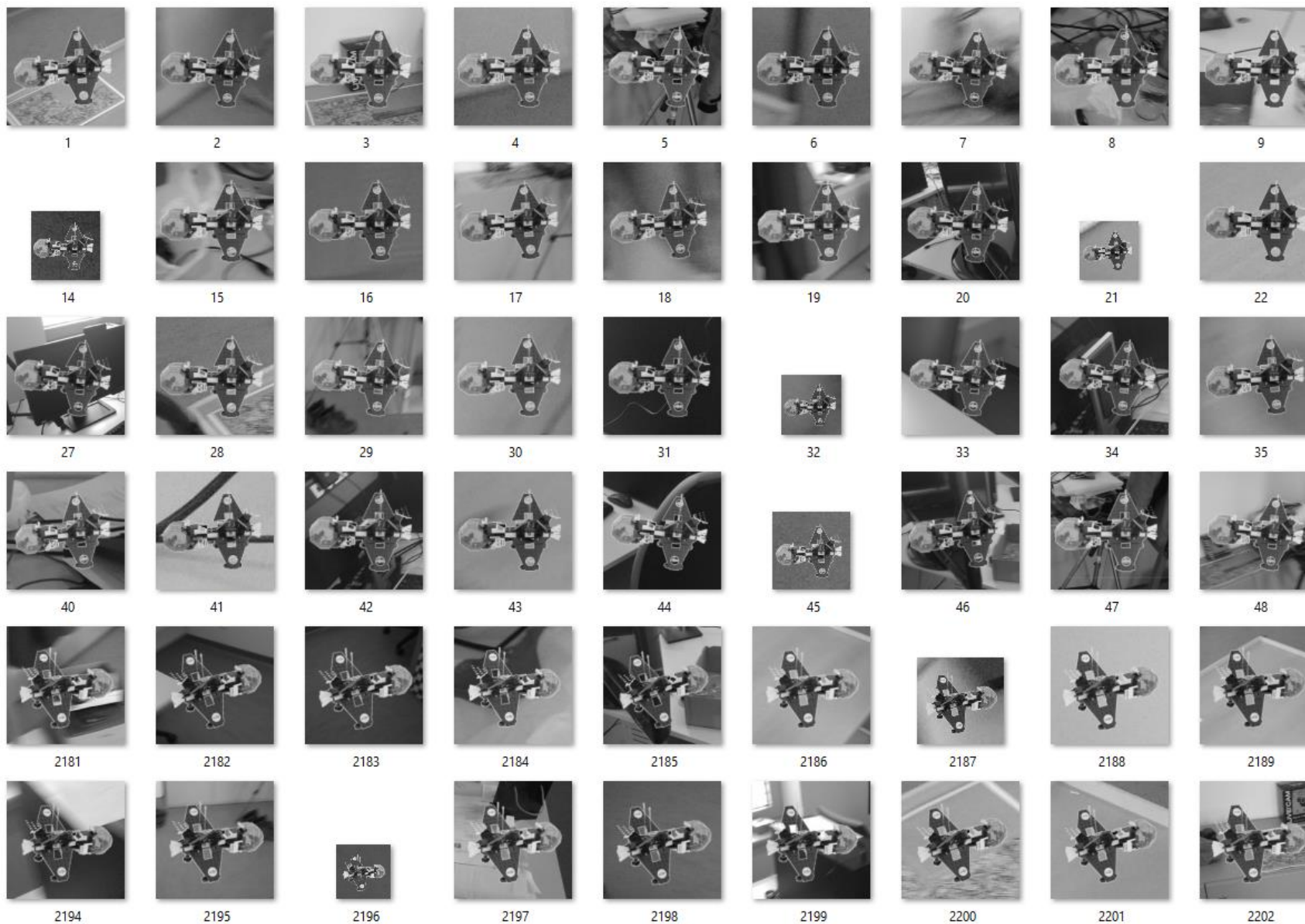


# Alien





# Avenger





# Negative Set



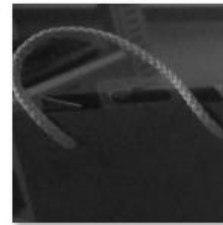
11



12



13



14



15



16



17



25



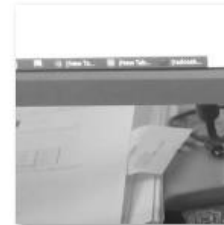
26



27



28



33



34



35



39



44



45



46



47



48



49



57



58



59



60



61



66



67



- Haar

- HOG

- LBP

- SIFT, SURF

- CNNs



Traditional Approaches



KeyPoints



Deep Learning Approach



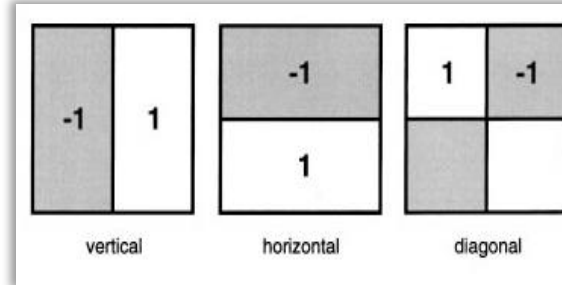
# Related Works

2000

Papageorgiou  
(2000)

## A Trainable System for Object Detection

CONSTANTINE PAPAGEORGIOU AND TOMASO POGGIO  
*Center for Biological and Computational Learning, Artificial Intelligence Laboratory, MIT,  
Cambridge, MA, USA*  
cpapa@ai.mit.edu  
tp@ai.mit.edu

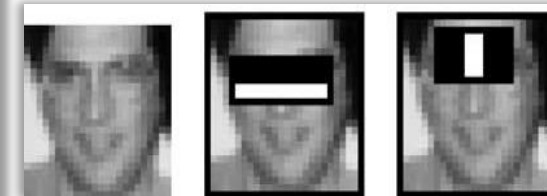


Viola, Jones  
(2001,2004)  
cit. > 6500

## Robust Real-Time Face Detection

PAUL VIOLA  
*Microsoft Research, One Microsoft Way, Redmond, WA 98052, USA*  
viola@microsoft.com

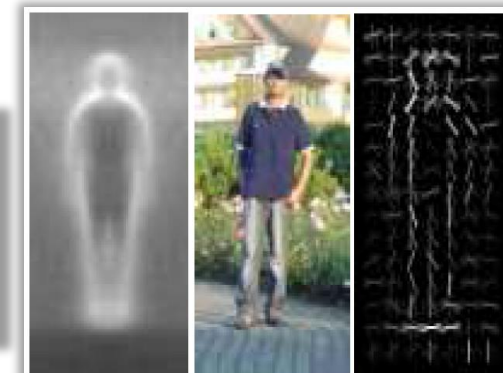
MICHAEL J. JONES  
*Mitsubishi Electric Research Laboratory, 201 Broadway, Cambridge, MA 02139, USA*  
mjones@merl.com



Dalal, Triggs  
(2005)  
cit. > 10000

## Histograms of Oriented Gradients for Human Detection

Navneet Dalal and Bill Triggs  
INRIA Rhône-Alpes, 655 avenue de l'Europe, Montbonnot 38334, France  
{Navneet.Dalal,Bill.Triggs}@inrialpes.fr, <http://lear.inrialpes.fr>



2005





# Haar Wavelet-based Descriptors

- The main idea behind the Haar-like features is that the features can encode the differences of mean intensities between the rectangular areas. For instance, in the problem of face detection, the regions around the eyes are lighter than the areas of the eyes; the regions below or on top of eyes have different intensities than the eyes themselves.
- These specific characteristics can be simply encoded by one two-rectangular feature, and the value of this feature can be calculated as the difference between the sum of the intensities inside the rectangles.

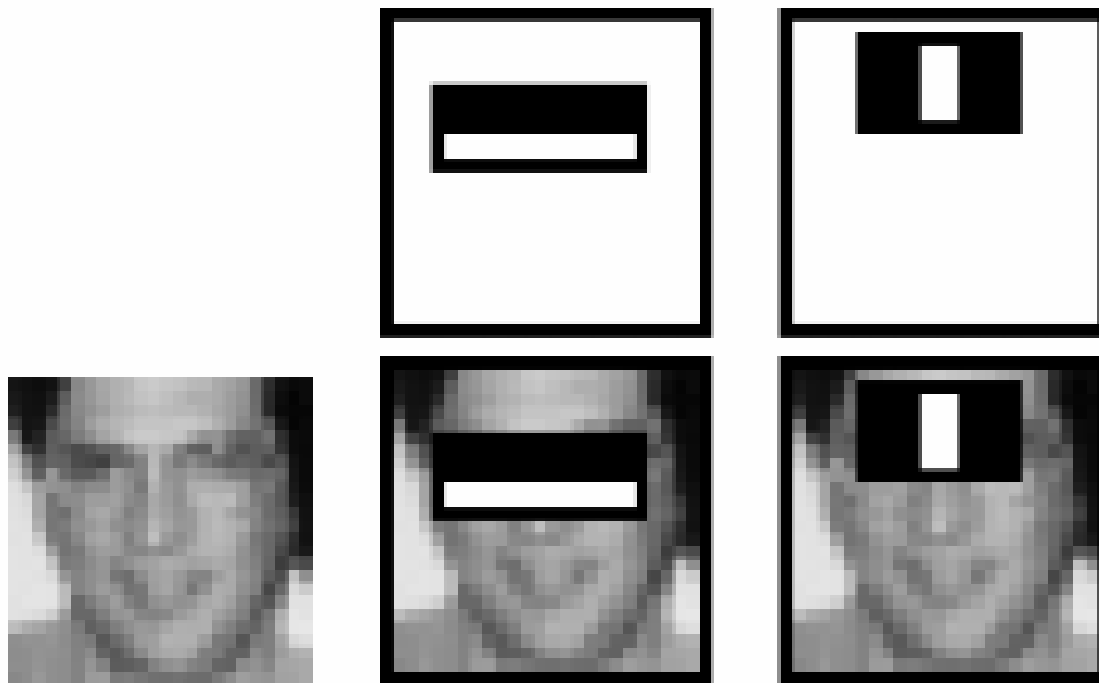


# Haar Wavelet-based Descriptors

- The paper of Viola and Jones contributed to the popularity of Haar-like features. The authors proposed the object detection framework based on the image representation called the integral image combined with the rectangular features, and the AdaBoost algorithm.
- With the use of integral image, the rectangular features are computed very quickly. The AdaBoost algorithm helps to select the most important features.
- The features are used to train classifiers and the cascade of classifiers is used for reducing the computational time.

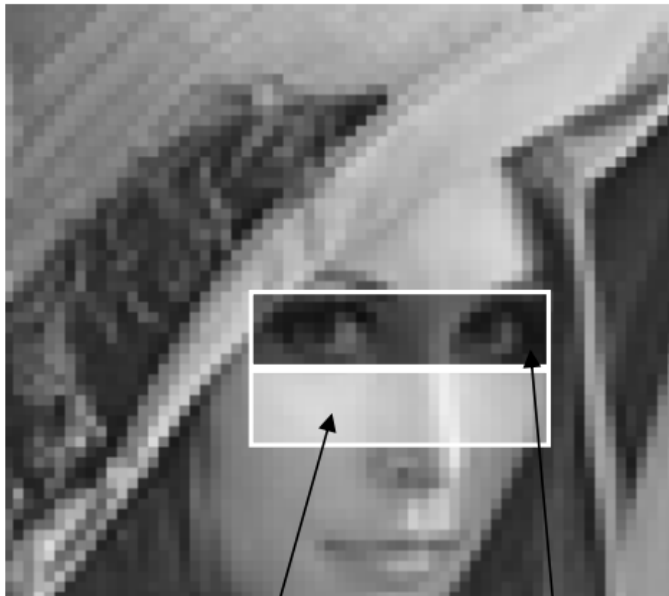
# Haar Wavelet-based Descriptors

- faces have similar properties
  - eye regions are darker than the upper-cheeks
  - the nose bridge region is brighter than the eyes



# Features

- Rectangular features



$R_{white}$

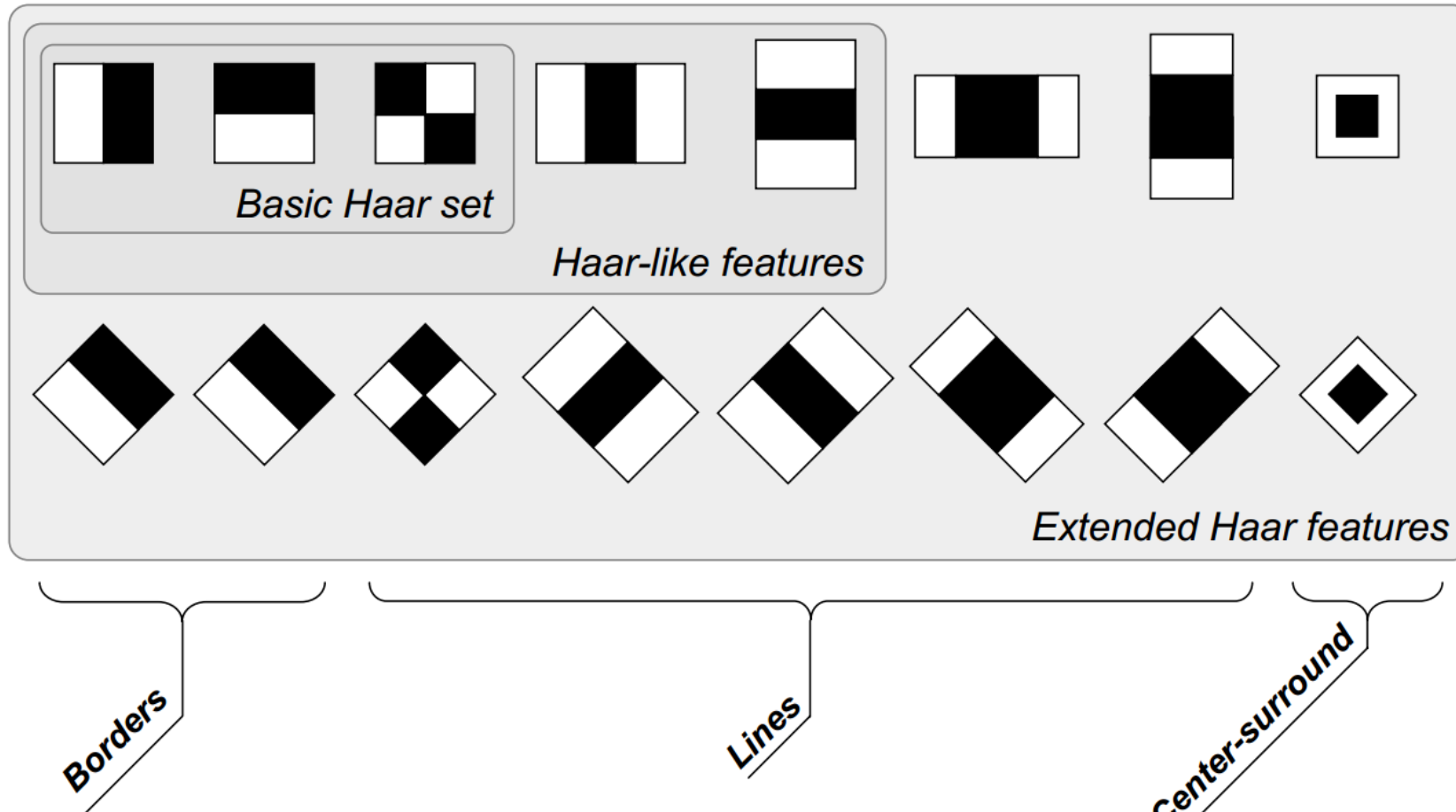
$R_{black}$



$$F_{Haar} = E(R_{white}) - E(R_{black})$$

# Features

## Different sets





# Face Detection

- 24x24 sub-window aprox. 160,000 rectangular features
- How speed the computational speed?
  - decrease memory accesses





# Integral Image

1	1	1
1	1	1
1	1	1

Original image (i)

1	2	3
2	4	6
3	6	9

Integral image (ii)

# Integral Image

Original image ( $I$ )

7	7	7	7	4	8	4	1
7	7	7	8	4	4	1	1
7	7	8	4	8	4	4	1
7	8	4	7	8	4	1	1
7	8	7	7	4	8	1	1
7	8	7	7	8	8	1	1



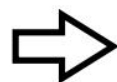
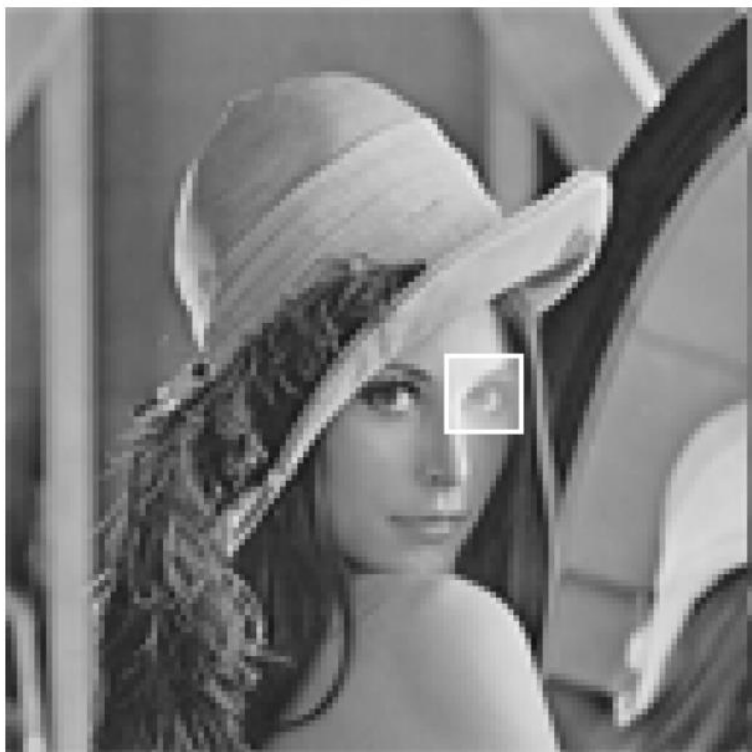
Integral image ( $ii$ )

7	14	21	28	32	40	44	45
14	28	42	57	65	77	82	84
21	42	64	83	99	119	124	127
28	57	83	109	133			

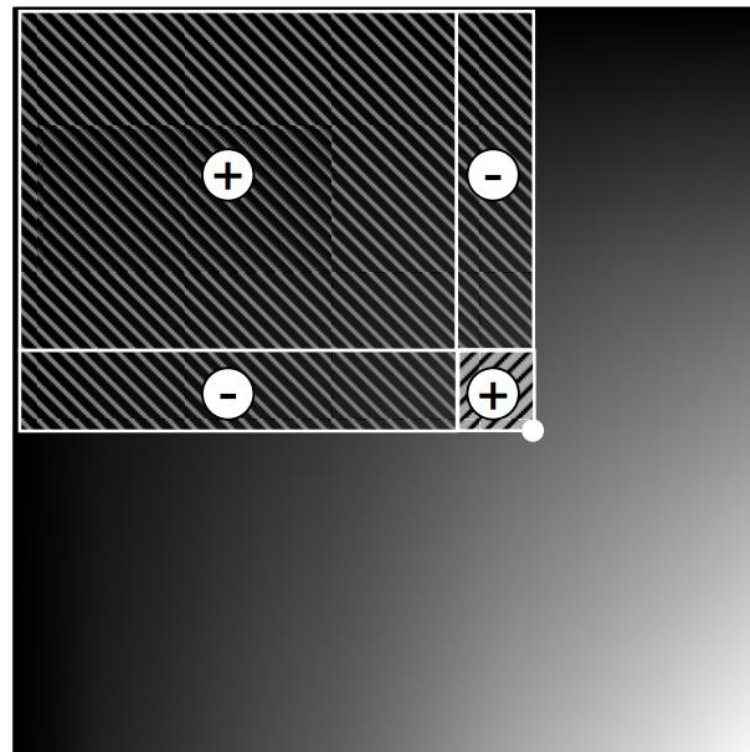
$$133 = 99 + 109 - 83 + 8$$

# Integral Image

Original image ( $I$ )

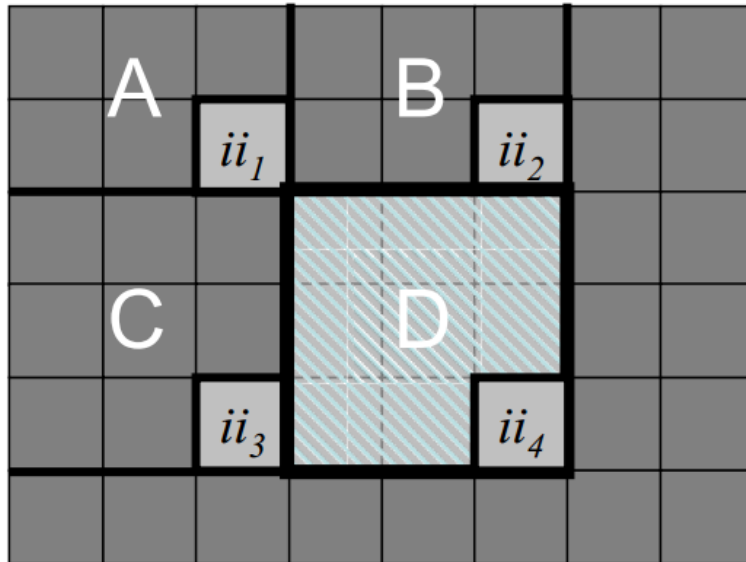


Integral image ( $ii$ )



# Integral Image

Integral image ( $ii$ )



$$ii_1 = \text{sum}(A)$$

$$ii_2 = \text{sum}(A) + \text{sum}(B)$$

$$ii_3 = \text{sum}(A) + \text{sum}(C)$$

$$ii_4 = \text{sum}(A) + \text{sum}(B) + \text{sum}(C) + \text{sum}(D)$$

$$\text{sum}(D) = ii_4 + ii_1 - ii_2 - ii_3$$



# Integral Image

Integral image (ii)

7	14	21	28	32	40	44	45
14	28	42	57	65	77	82	84
21	42	64	83	99	119	124	127
28	57	83	109	133	153	159	163
35	72	105	138	166	194	201	206
42	87	127	167	203	239	247	253

Original image (I)

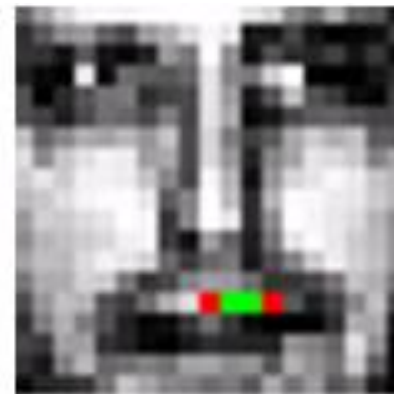
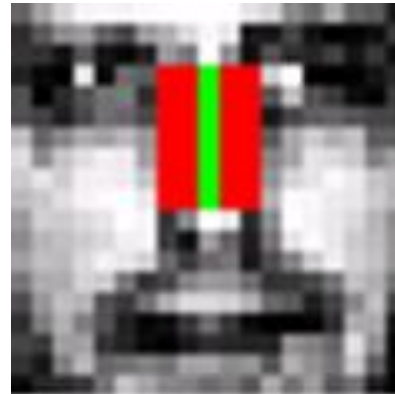
7	7	7	7	4	8	4	1
7	7	7	8	4	4	1	1
7	7	8	4	8	4	4	1
7	8	4	7	8	4	1	1
7	8	7	7	4	8	1	1
7	8	7	7	8	8	1	1

54

$$54 = 194 + 42 - 77 - 105$$



# Feature Selection



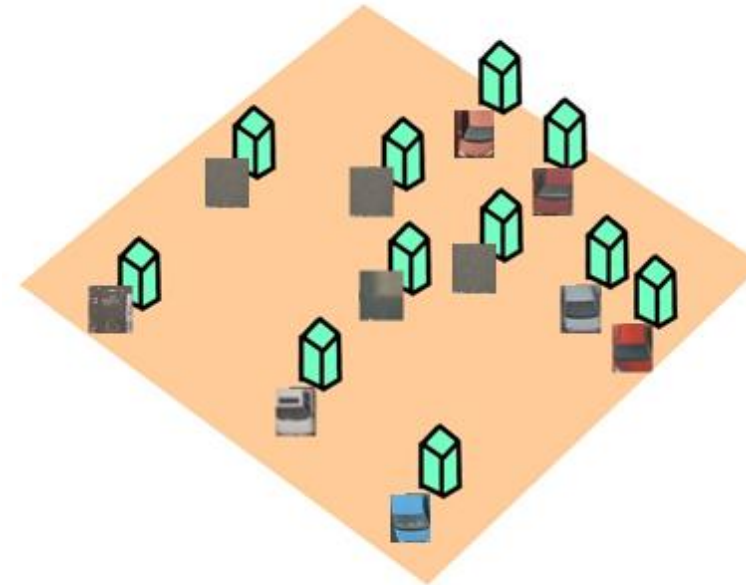


## Feature Selection

- AdaBoost (Adaptive Boost) is an iterative learning algorithm to construct a “strong” classifier as a linear combination of weighted simple “weak” classifiers
- weak classifier - each single rectangle feature (features as weak classifiers)
- during each iteration, each example/image receives a weight determining its importance

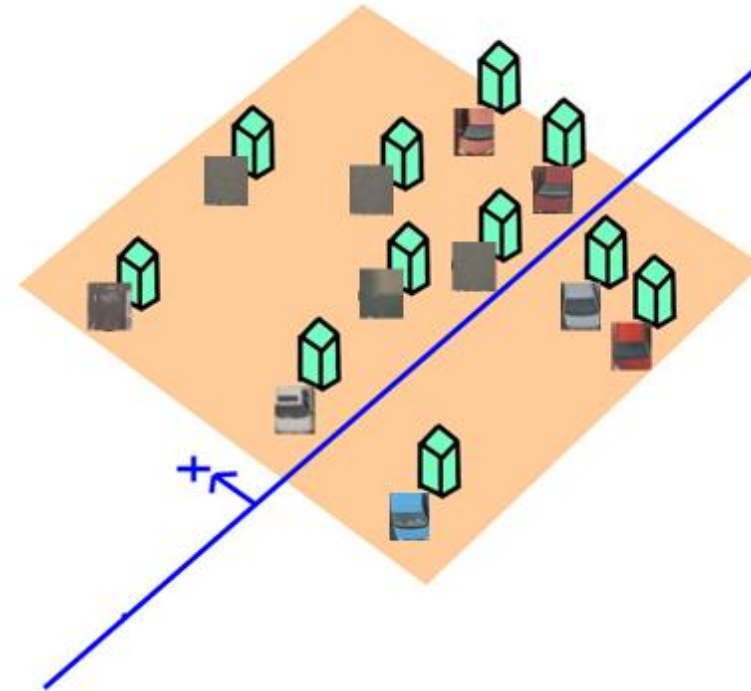
# Feature Selection

□ AdaBoost starts with a uniform distribution of “weights” over training examples.



# Feature Selection

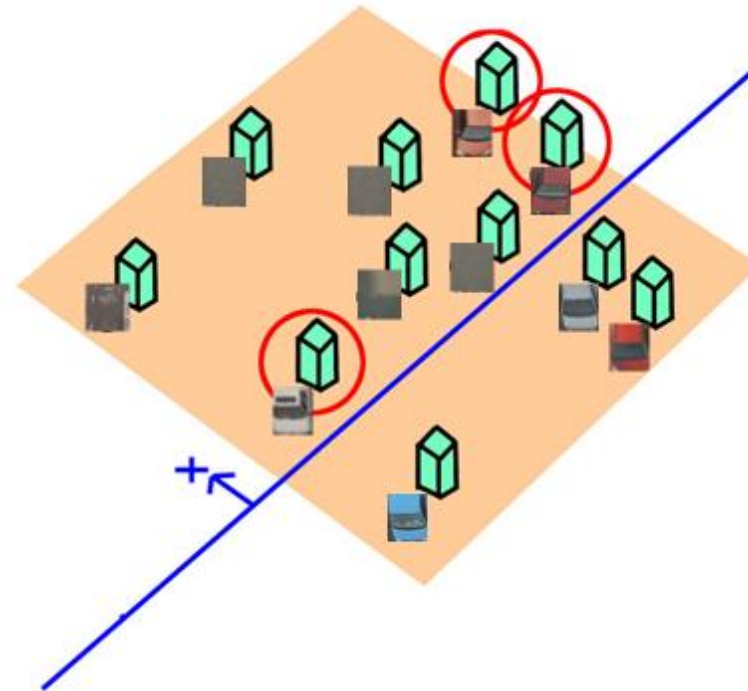
- ❑ AdaBoost starts with a uniform distribution of “weights” over training examples.
- ❑ Select the classifier with the lowest weighted error (i.e. a “weak” classifier)





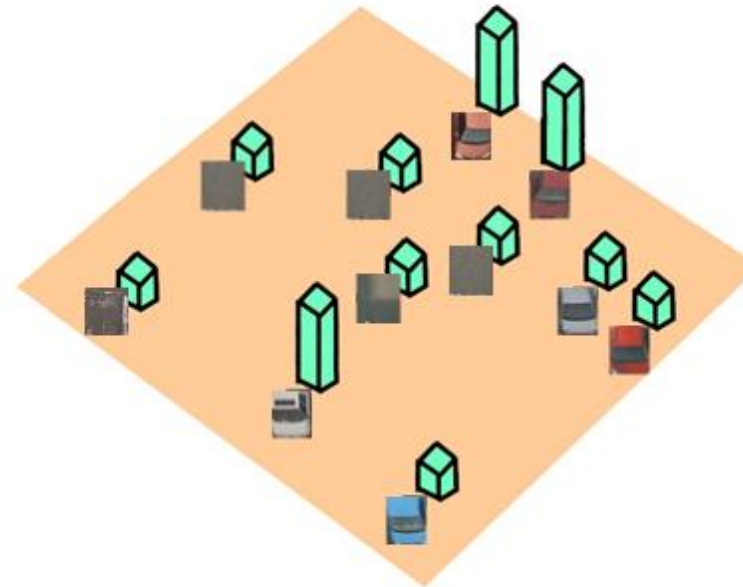
# Feature Selection

- ❑ AdaBoost starts with a uniform distribution of “weights” over training examples.
- ❑ Select the classifier with the lowest weighted error (i.e. a “weak” classifier)
- ❑ Increase the weights on the training examples that were misclassified.



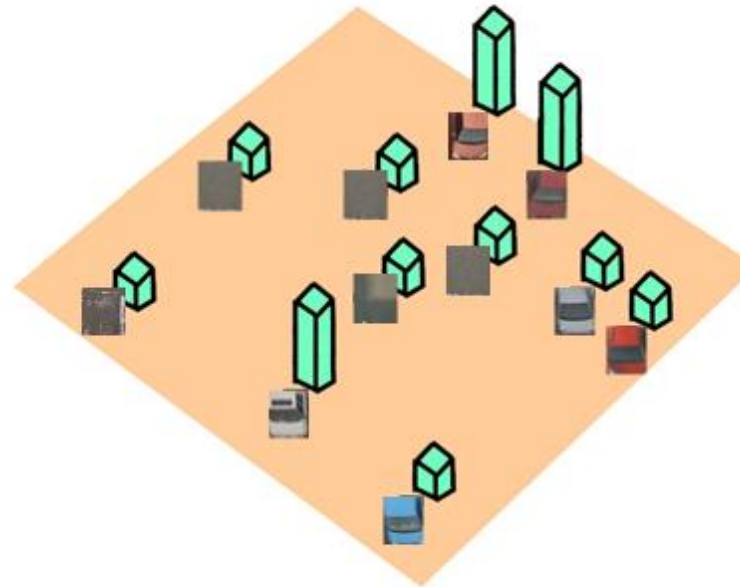
# Feature Selection

- ❑ AdaBoost starts with a uniform distribution of “weights” over training examples.
- ❑ Select the classifier with the lowest weighted error (i.e. a “weak” classifier)
- ❑ Increase the weights on the training examples that were misclassified.



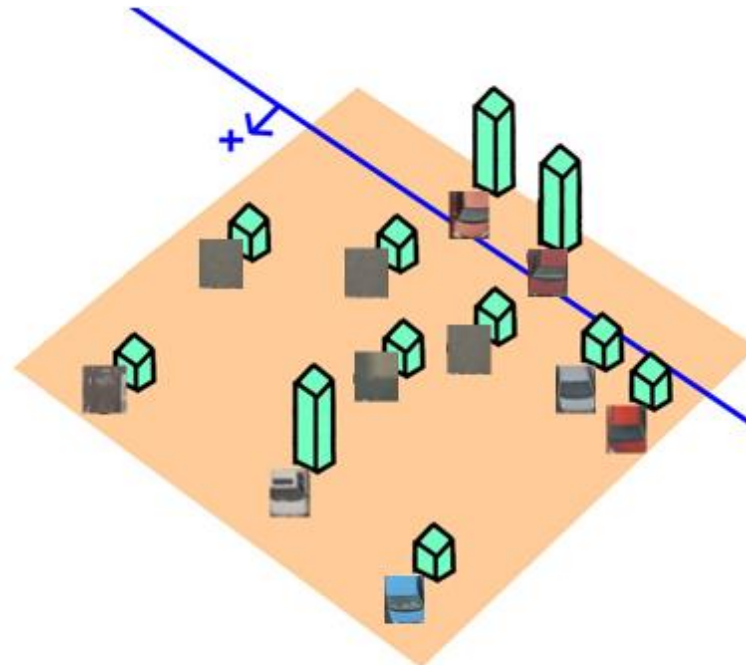
# Feature Selection

- ❑ AdaBoost starts with a uniform distribution of “weights” over training examples.
- ❑ Select the classifier with the lowest weighted error (i.e. a “weak” classifier)
- ❑ Increase the weights on the training examples that were misclassified.
- ❑ (Repeat)



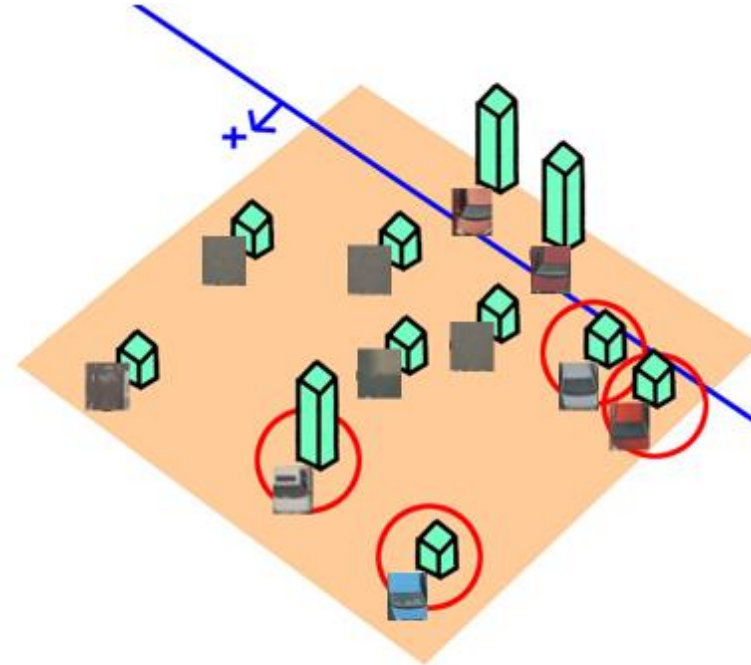
# Feature Selection

- ❑ AdaBoost starts with a uniform distribution of “weights” over training examples.
- ❑ Select the classifier with the lowest weighted error (i.e. a “weak” classifier)
- ❑ Increase the weights on the training examples that were misclassified.
- ❑ (Repeat)



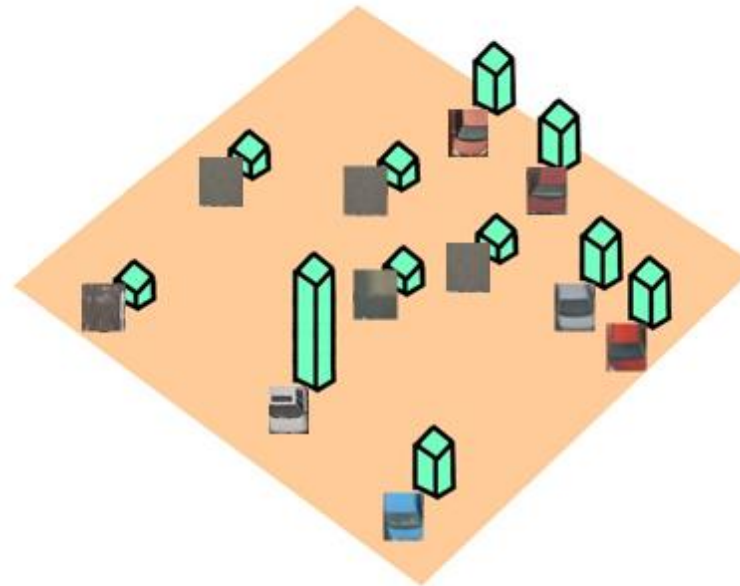
# Feature Selection

- ❑ AdaBoost starts with a uniform distribution of “weights” over training examples.
- ❑ Select the classifier with the lowest weighted error (i.e. a “weak” classifier)
- ❑ Increase the weights on the training examples that were misclassified.
- ❑ (Repeat)



# Feature Selection

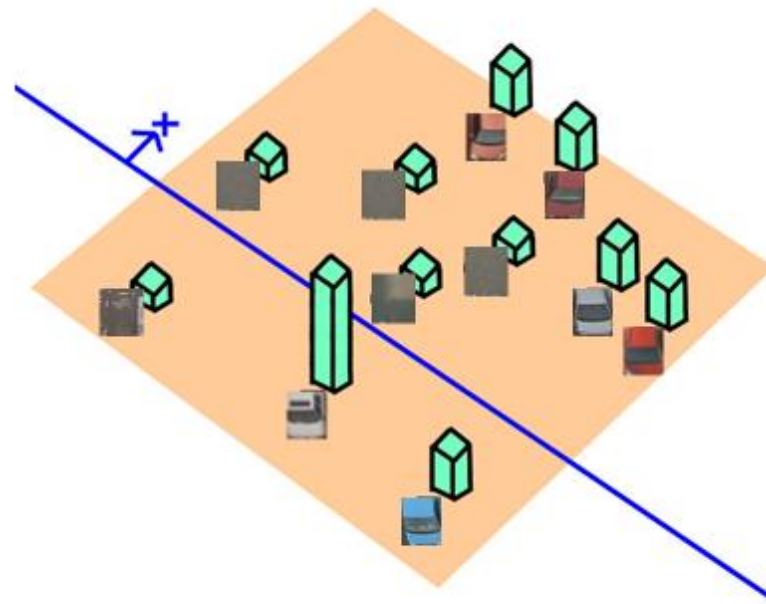
- ❑ AdaBoost starts with a uniform distribution of “weights” over training examples.
- ❑ Select the classifier with the lowest weighted error (i.e. a “weak” classifier)
- ❑ Increase the weights on the training examples that were misclassified.
- ❑ (Repeat)





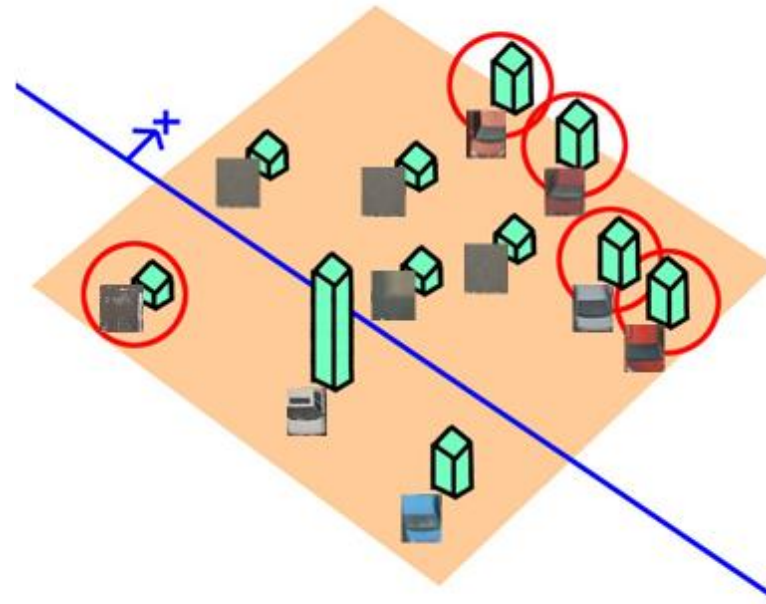
# Feature Selection

- ❑ AdaBoost starts with a uniform distribution of “weights” over training examples.
- ❑ Select the classifier with the lowest weighted error (i.e. a “weak” classifier)
- ❑ Increase the weights on the training examples that were misclassified.
- ❑ (Repeat)

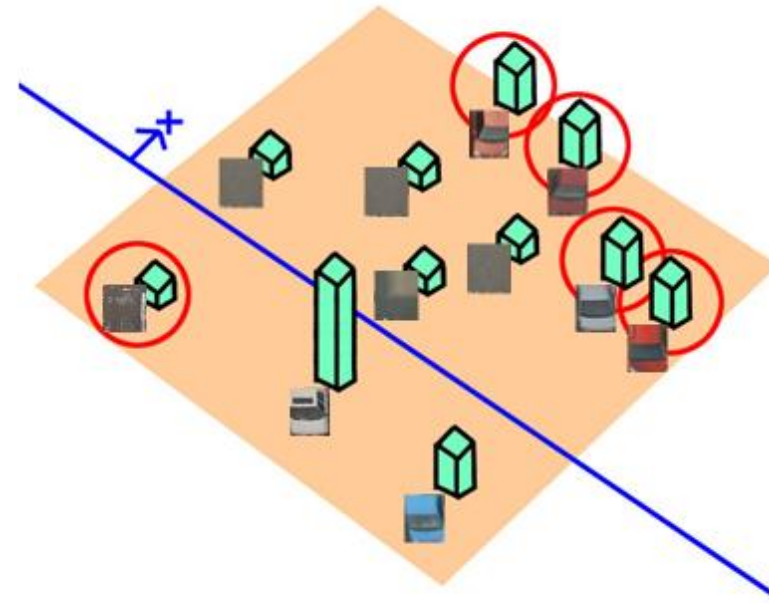


# Feature Selection

- ❑ AdaBoost starts with a uniform distribution of “weights” over training examples.
- ❑ Select the classifier with the lowest weighted error (i.e. a “weak” classifier)
- ❑ Increase the weights on the training examples that were misclassified.
- ❑ (Repeat)

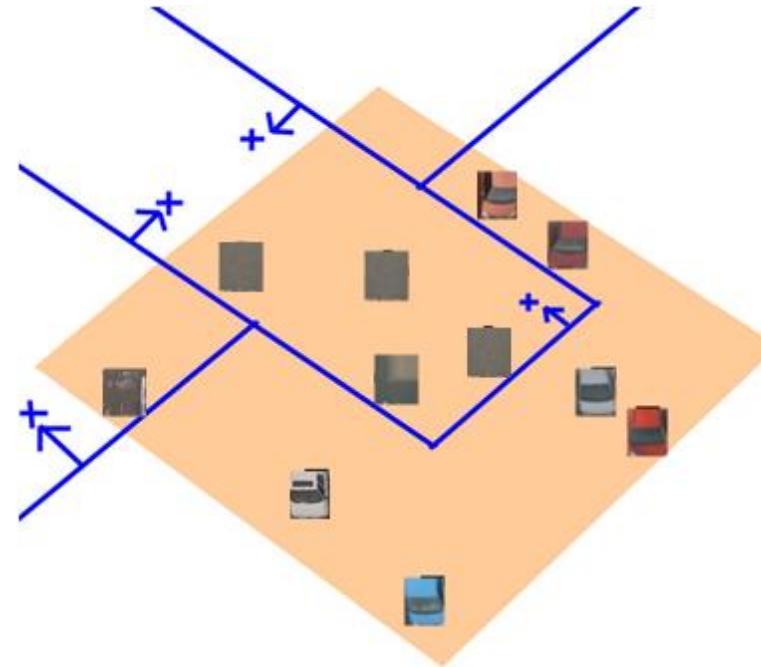


# Feature Selection



- At the end, carefully make a linear combination of the weak classifiers obtained at all iterations.

# Feature Selection

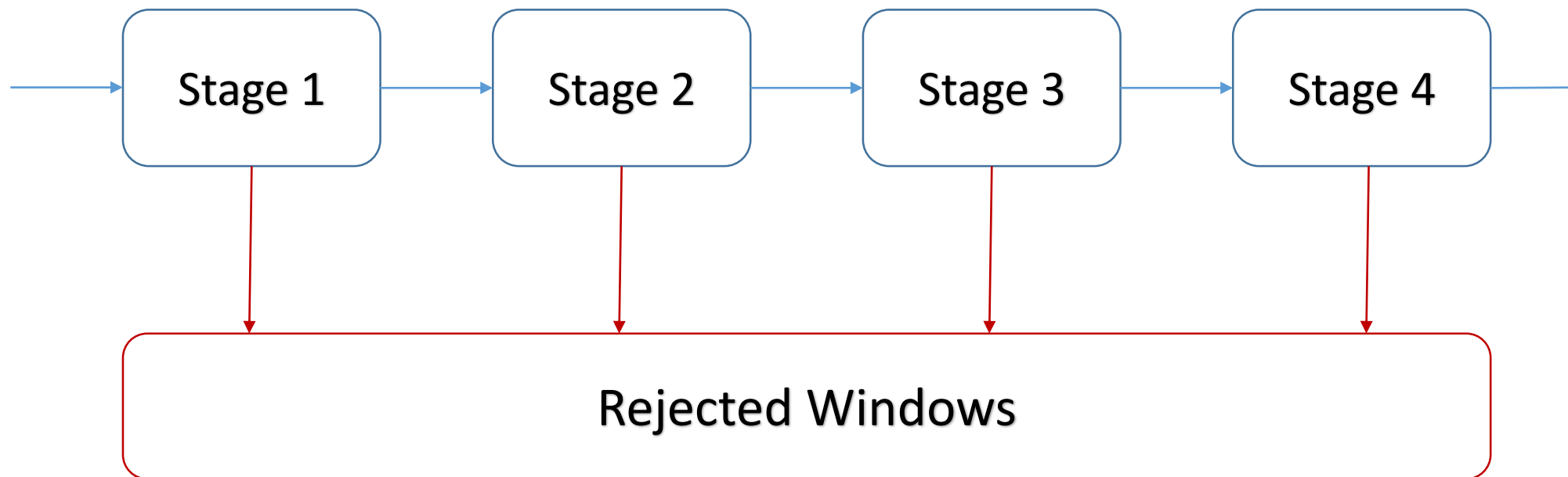
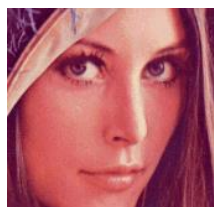


- At the end, carefully make a linear combination of the weak classifiers obtained at all iterations.



# Cascade of Classifier

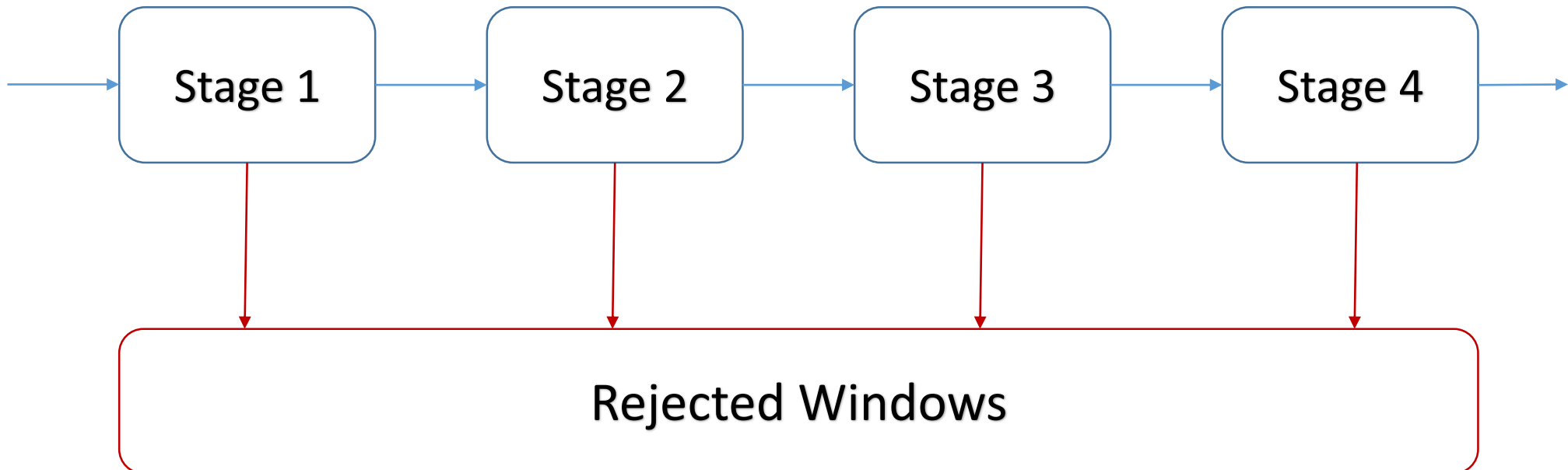
The idea of cascade classifier is reject the non-face region as soon as possible





# Cascade of Classifier

The idea of cascade classifier is reject the non-face region as soon as possible

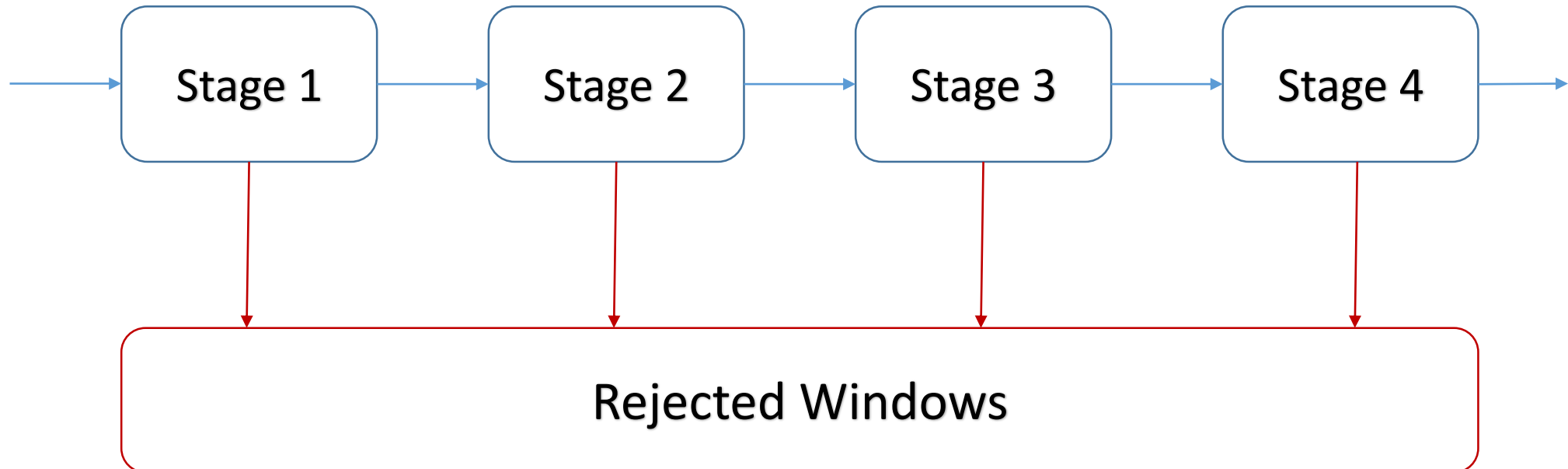
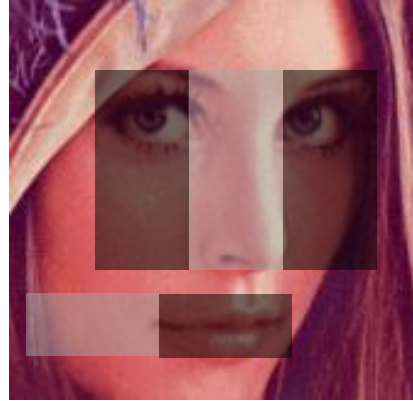






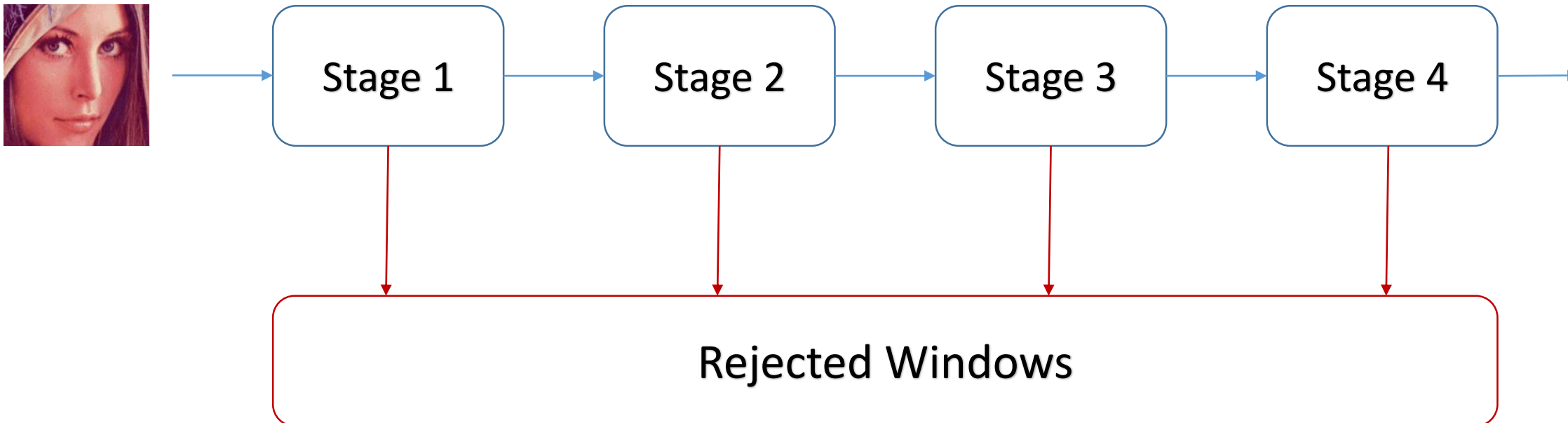
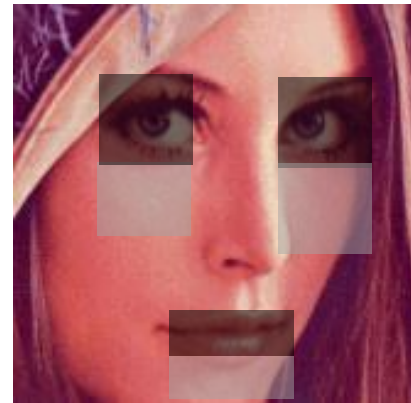
# Cascade of Classifier

The idea of cascade classifier is reject the non-face region as soon as possible



# Cascade of Classifier

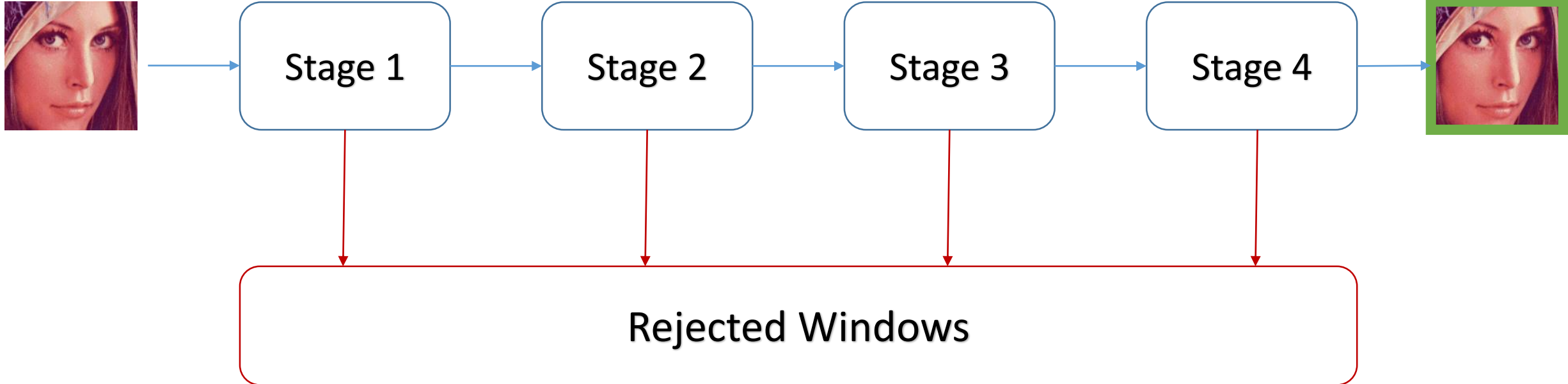
The idea of cascade classifier is reject the non-face region as soon as possible





# Cascade of Classifier

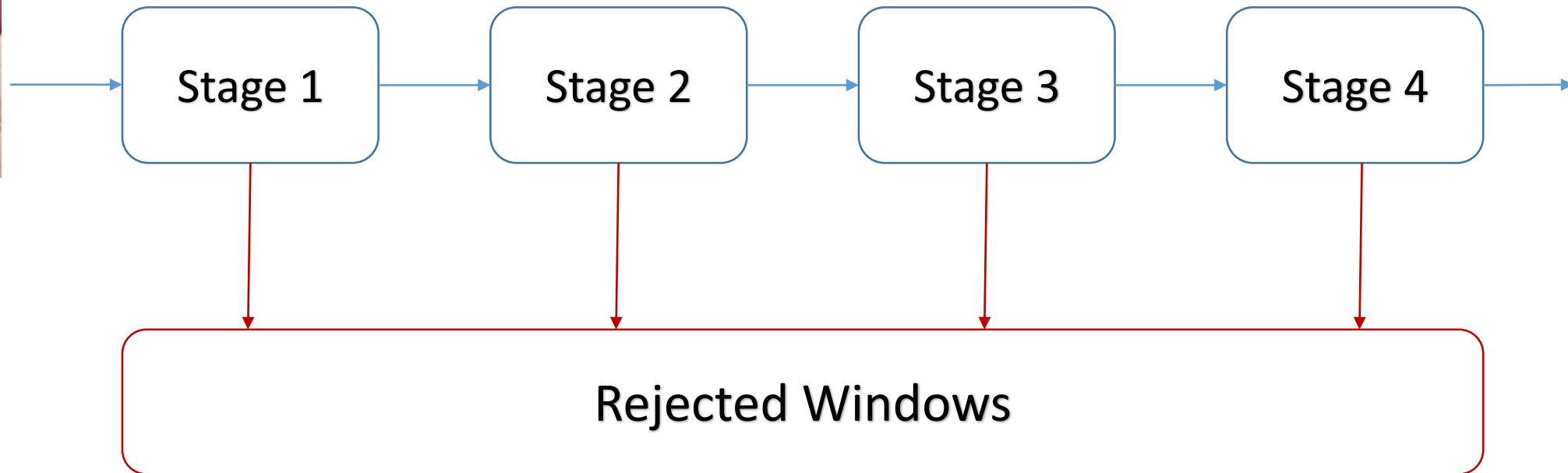
The idea of cascade classifier is reject the non-face region as soon as possible





# Cascade of Classifier

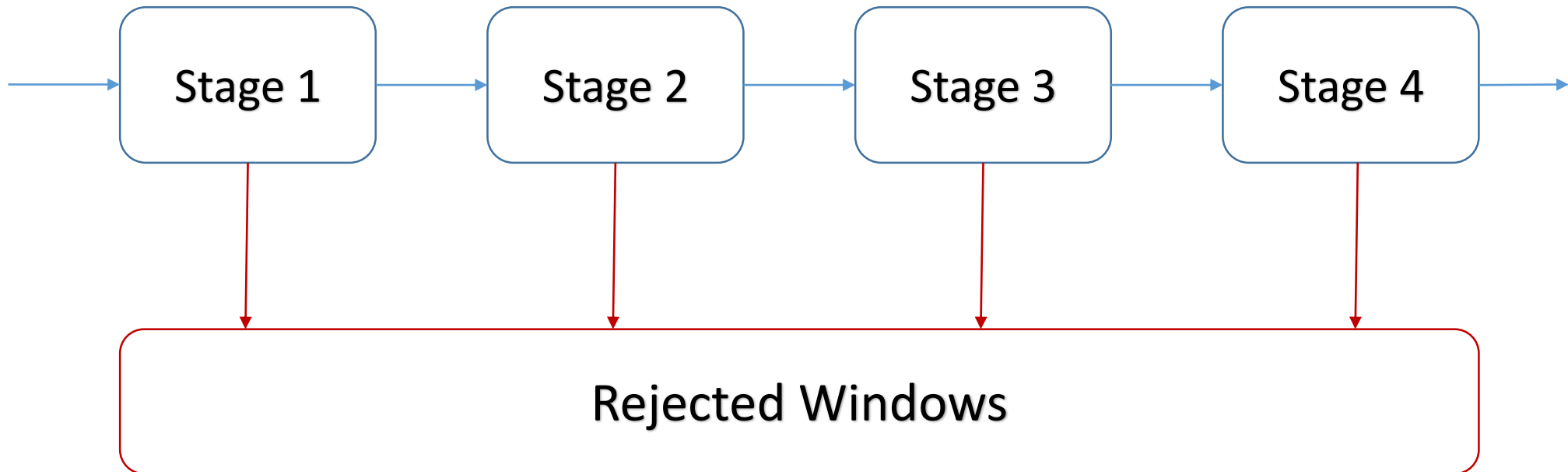
The idea of cascade classifier is reject the non-face region as soon as possible





# Cascade of Classifier

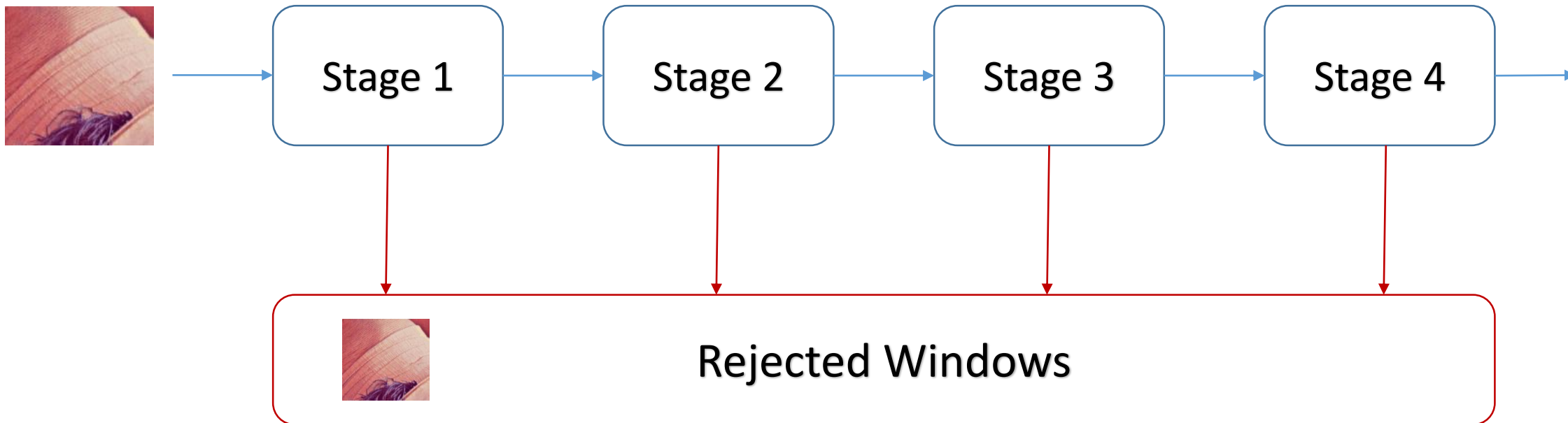
The idea of cascade classifier is reject the non-face region as soon as possible





# Cascade of Classifier

The idea of cascade classifier is reject the non-face region as soon as possible





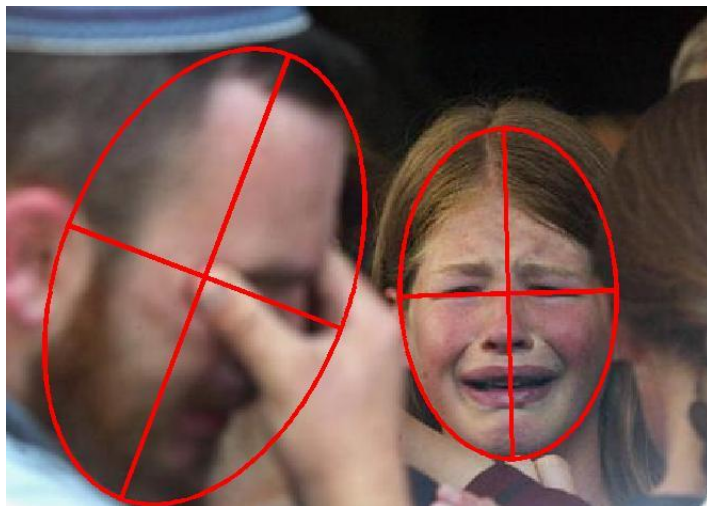


# Haar Features





# Face Detection - Evaluation



<http://vis-www.cs.umass.edu/tddb/>



# Face Detection - Evaluation

TP = number of true positives

FP = number of false positives

FN = number of false negatives

TN = number of true negatives

precision =  $TP / (TP + FP)$

sensitivity =  $TP / (TP + FN)$

F1 score (harmonic mean of precision and sensitivity) =  $2 \times$   
precision  $\times$  sensitivity / (precision + sensitivity)



# Face Detection - Evaluation



Figure 8. *Matching detections and annotations.* In this image, the ellipses specify the face annotations and the five rectangles denote a face detector's output. Note that the second face from left has two detections overlapping with it. We require a valid matching to accept only one of these detections as the true match, and to consider the other detection as a false positive. Also, note that the third face from the left has no detection overlapping with it, so no detection should be matched with this face. The blue rectangles denote the true positives and yellow rectangles denote the false positives in the desired matching.

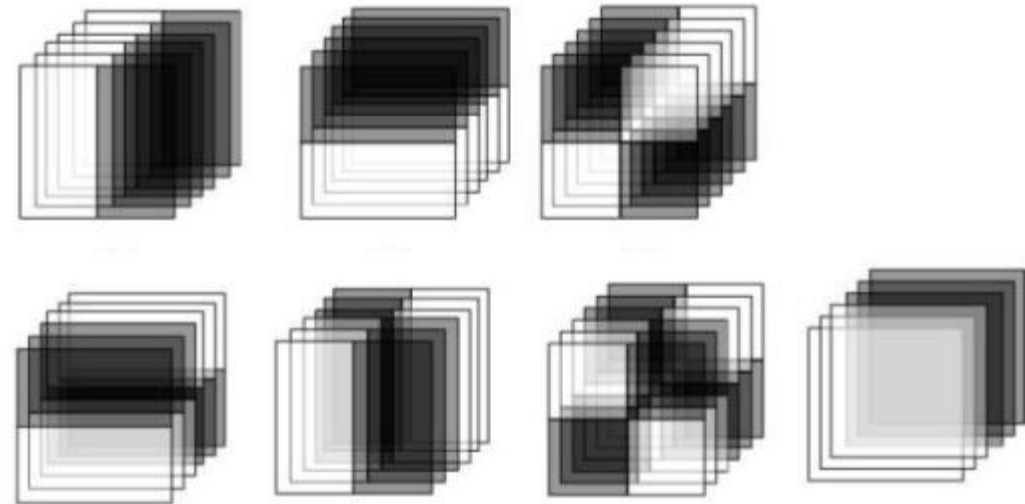


# Haar Features

- Since Viola and Jones popularized the Haar-like features for face detection, the Haarlike features and their modifications were used in many detection tasks (e.g. pedestrian, eye, vehicle).
- In the area of pedestrian detection, in [1], the authors presented the component-based person detector that is able to detect the occluded people in clustered scenes in static images. The detector uses the Haar-like features to describe the components of people (heads, legs, arms) combined with the SVM classifier. The Viola and Jones detection framework was successfully extended for moving-human detection in [2]. In [3], the authors proposed the method for estimating the walking direction of pedestrian.

# Haar Features

- The 3D Haar-like features for pedestrian detection were presented in [4]. The authors extend the classical Haar-like features using the volume filters in 3D space (instead of using rectangle filters in 2D space) to capture motion information. The 3D features are then combined with the SVM classifier. To compute the 3D Haar-like features using the integral image like the classical 2D features, the authors introduced Integral Volume that extends 2D integral image to the three dimensions.





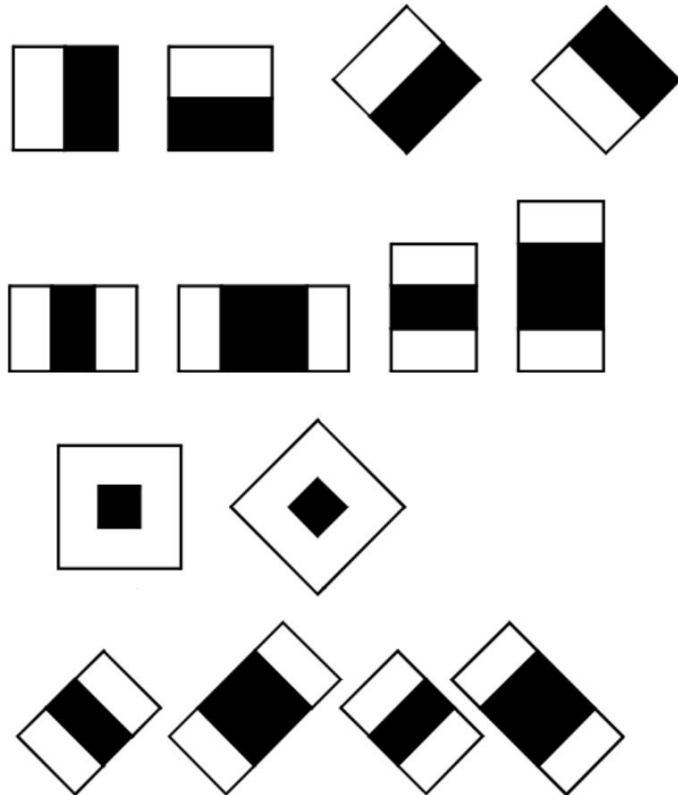


# Haar Features

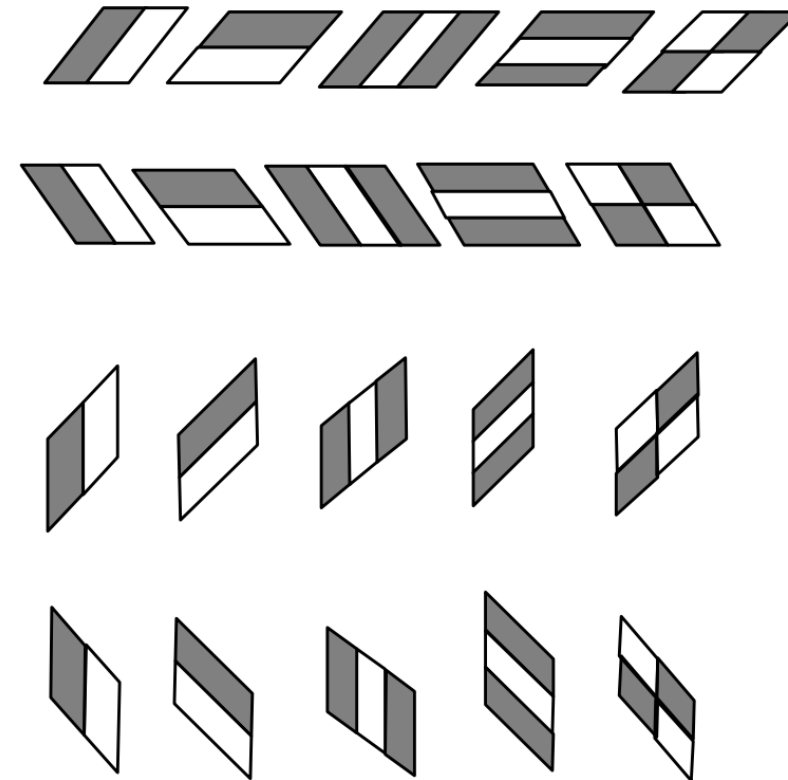
- [1] Mohan, A., Papageorgiou, C., Poggio, T.: Example-based object detection in images by components. IEEE Trans. Pattern Anal. Mach. Intell. 23(4), 349–361 (Apr 2001), <http://dx.doi.org/10.1109/34.917571>
- [2] Viola, P., Jones, M., Snow, D.: Detecting pedestrians using patterns of motion and appearance. In: Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on. pp. 734 –741 vol.2 (oct 2003)
- [3] Shimizu, H., Poggio, T.: Direction estimation of pedestrian from multiple still images. In: Intelligent Vehicles Symposium, 2004 IEEE. pp. 596–600 (2004)
- [4] Cui, X., Liu, Y., Shan, S., Chen, X., Gao, W.: 3d haar-like features for pedestrian detection. In: Multimedia and Expo, 2007 IEEE International Conference on. pp. 1263–1266 (July 2007)

# Haar Features

The modified version of Haar-like features that more properly reflect the shape of the pedestrians than the classical Haar-like features.



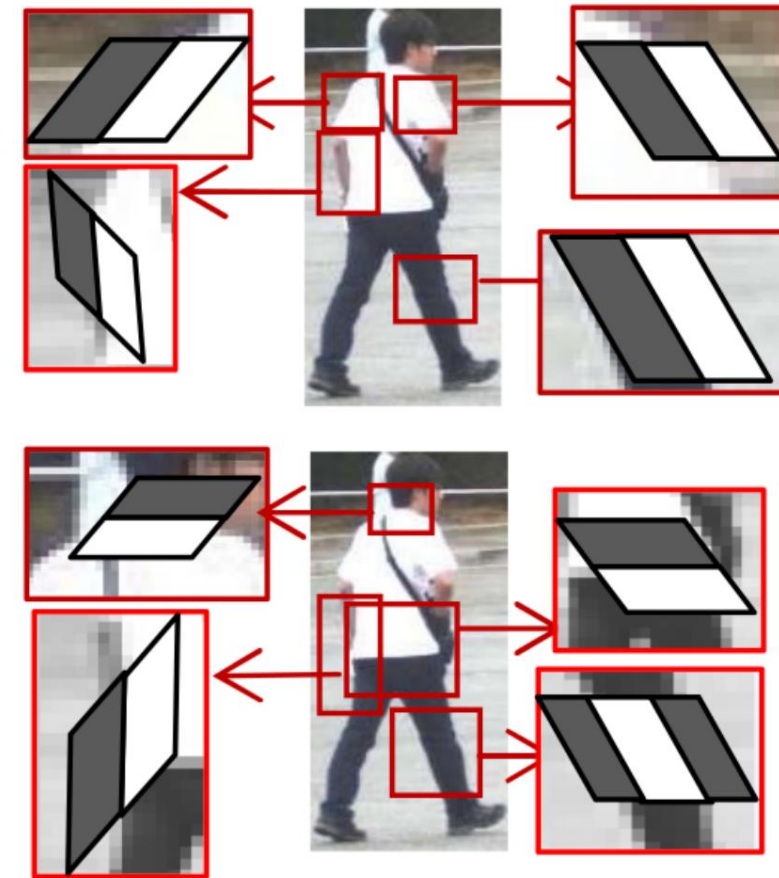
Lienhart, R., Maydt, J.: An extended set of haar-like features for rapid object detection. In: Image Processing. 2002. Proceedings. 2002 International Conference on. vol. 1, pp. 1-900-1-903 vol.1 (2002)



Hoang, V.D., Vavilin, A., Jo, K.H.: Pedestrian detection approach based on modified haar-like features and adaboost. In: Control, Automation and Systems (ICCAS), 2012 12th International Conference on. pp. 614-618 (Oct 2012)

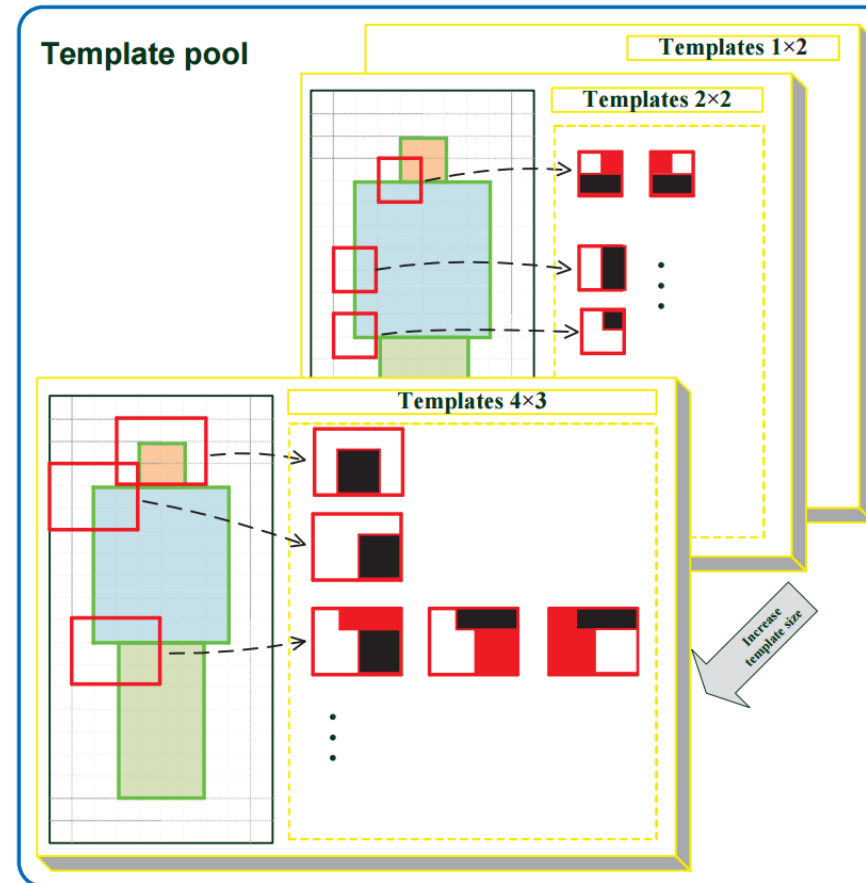
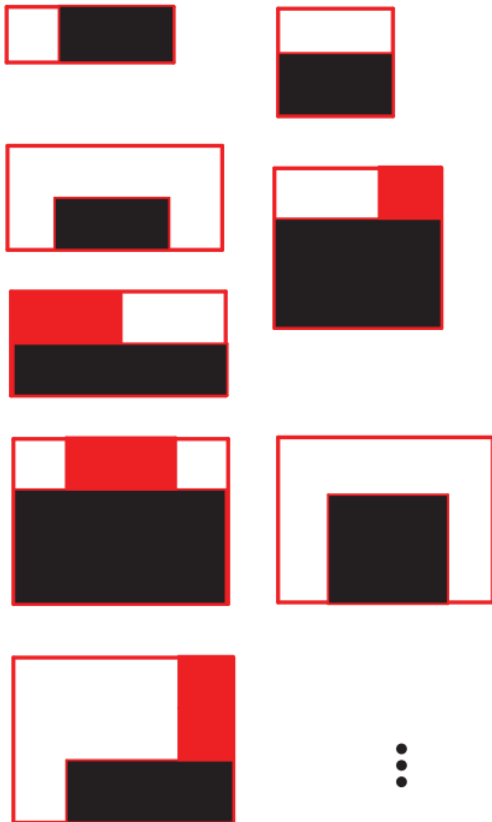
# Haar Features

The modified version of Haar-like features that more properly reflect the shape of the pedestrians than the classical Haar-like features.



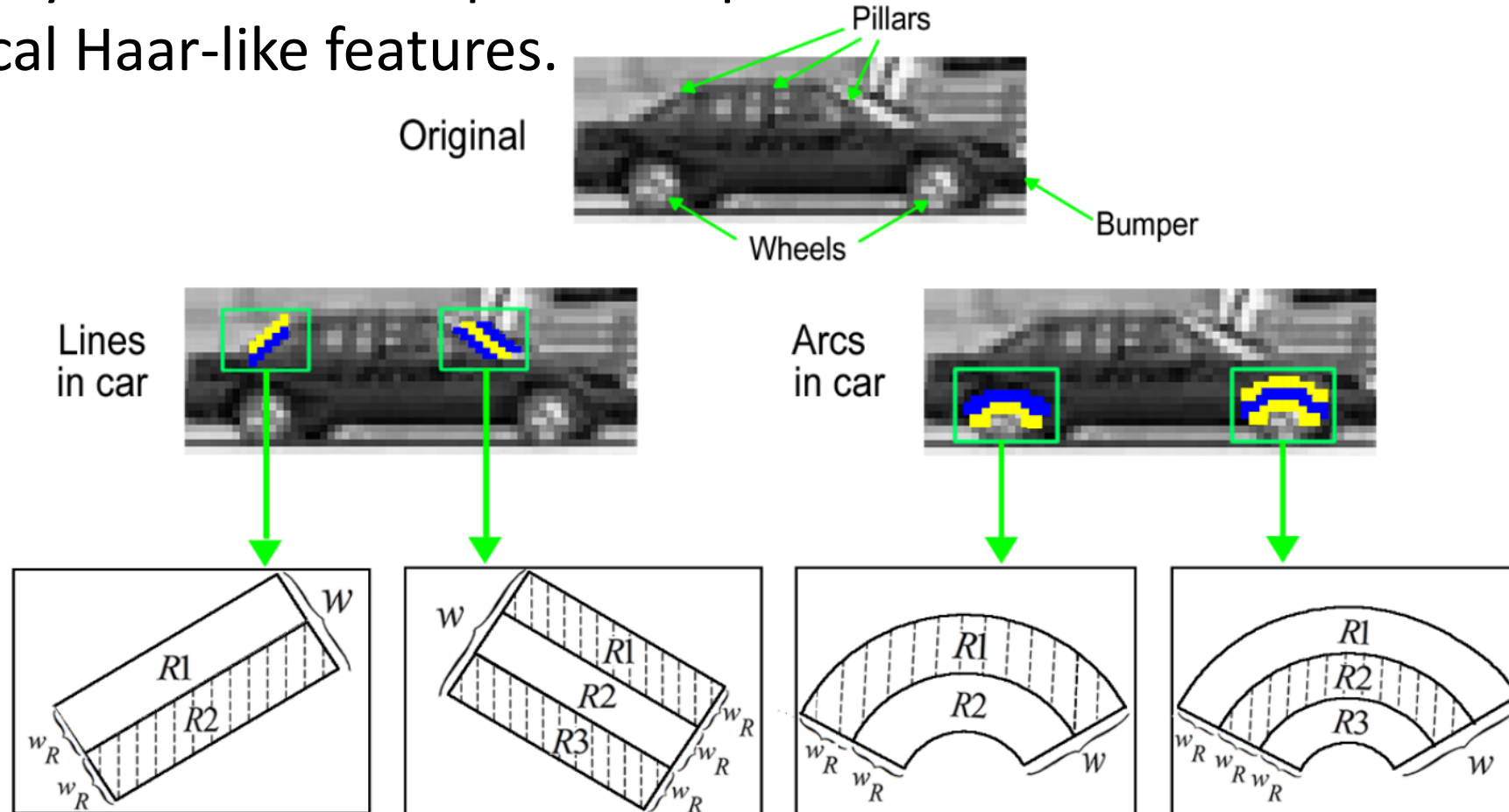
# Haar Features

The modified version of Haar-like features that more properly reflect the shape of the pedestrians than the classical Haar-like features.



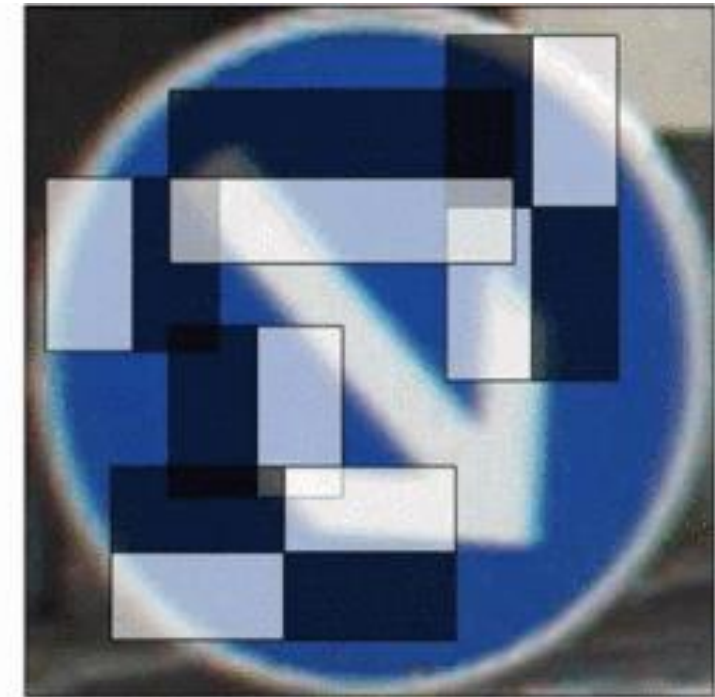
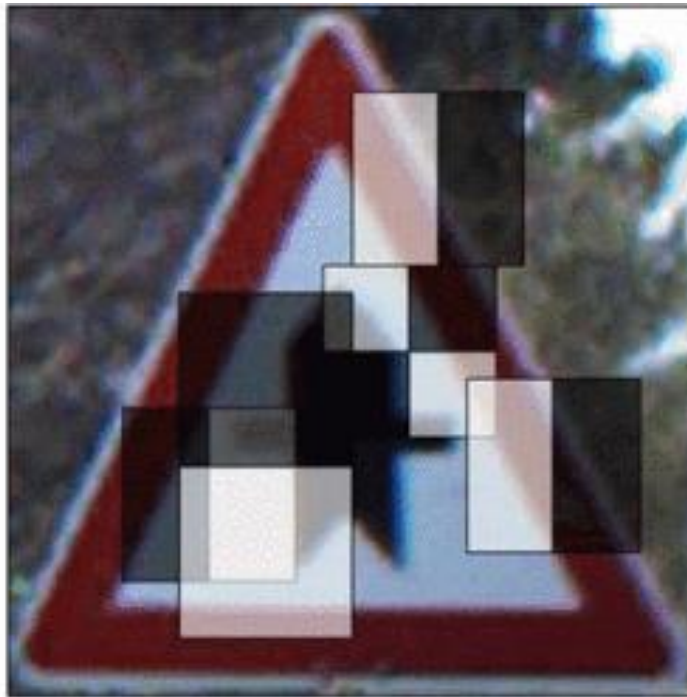
# Haar Features

The modified version of Haar-like features that more properly reflect the shape of the pedestrians than the classical Haar-like features.



Zheng, W., Liang, L.: Fast car detection using image strip features. In: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on. pp. 2703–2710 (2009)

# Haar Features



Five most significant Haar features selected for the first stage of each of the 3 trained detectors

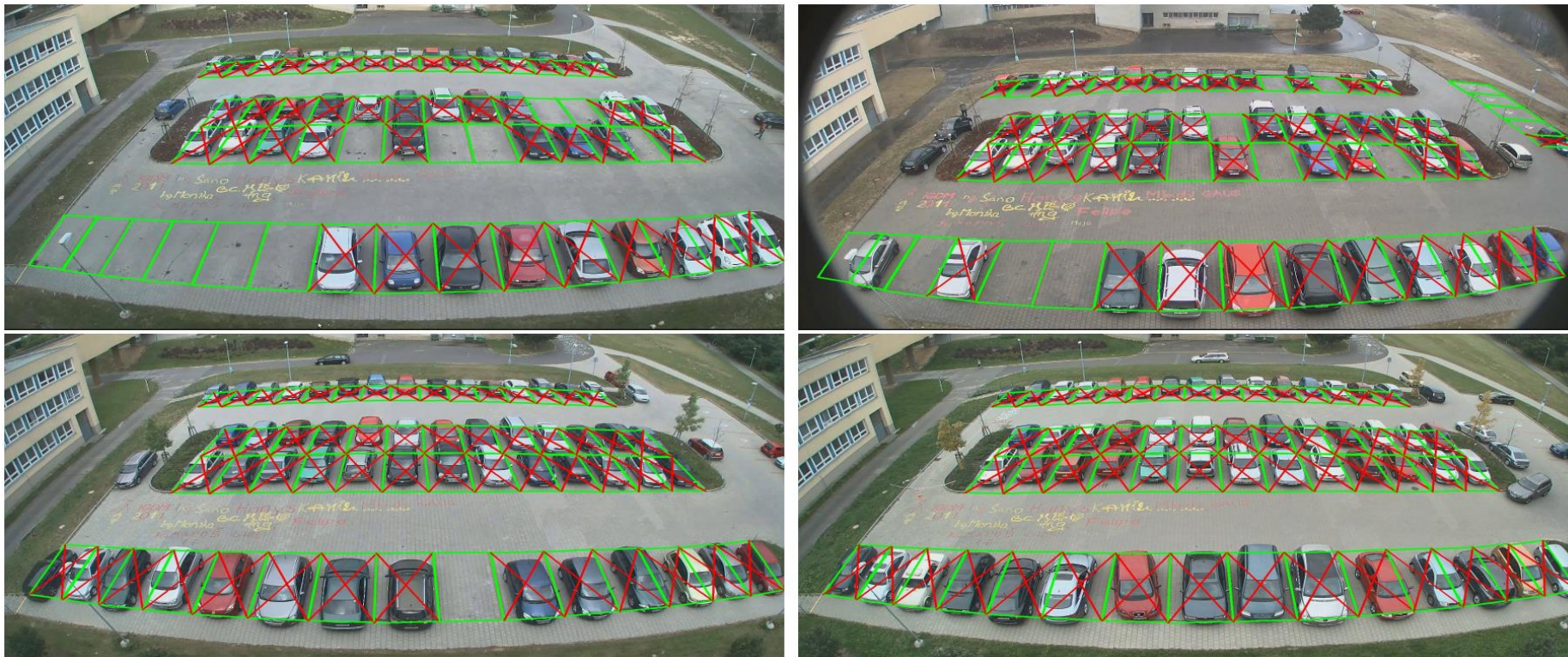
<https://ieeexplore.ieee.org/abstract/document/8050341>

A. Suleiman, Y. Chen, J. Emer and V. Sze, "Towards closing the energy gap between HOG and CNN features for embedded vision," *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, Baltimore, MD, 2017, pp. 1-4.  
doi: 10.1109/ISCAS.2017.8050341



- Fusek, R., Mozdřeň, K., Šurkala, M., Sojka, E.: **AdaBoost for Parking Lot Occupation Detection**. Advances in Intelligent Systems and Computing, vol. 226, pp. 681-690 (2013)

<http://mrl.cs.vsb.cz/>



# Parking Lot Occupation



- Haar

- HOG

- LBP

- SIFT, SURF

- CNNs



Traditional Approaches



KeyPoints



Deep Learning Approach

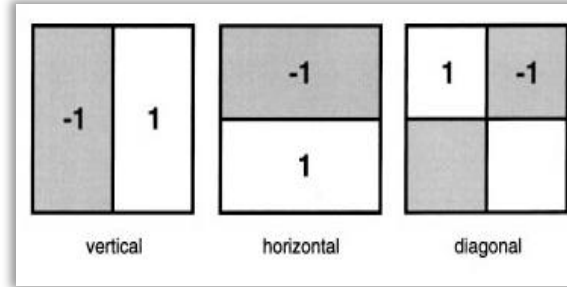
# Related Works

2000

Papageorgiou  
(2000)

## A Trainable System for Object Detection

CONSTANTINE PAPAGEORGIOU AND TOMASO POGGIO  
*Center for Biological and Computational Learning, Artificial Intelligence Laboratory, MIT,  
Cambridge, MA, USA*  
cpapa@ai.mit.edu  
tp@ai.mit.edu



Viola, Jones  
(2001,2004)

## Robust Real-Time Face Detection

PAUL VIOLA  
*Microsoft Research, One Microsoft Way, Redmond, WA 98052, USA*  
viola@microsoft.com

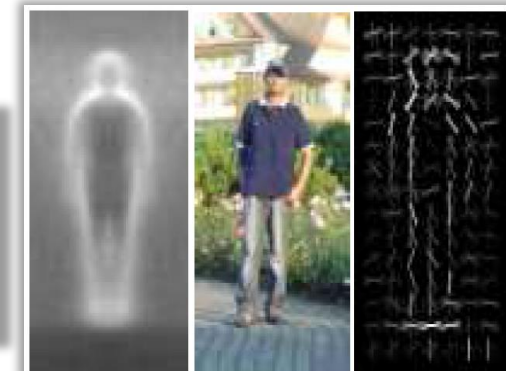
MICHAEL J. JONES  
*Mitsubishi Electric Research Laboratory, 201 Broadway, Cambridge, MA 02139, USA*  
mjones@merl.com



Dalal, Triggs  
(2005)  
cit. 10947

## Histograms of Oriented Gradients for Human Detection

Navneet Dalal and Bill Triggs  
INRIA Rhône-Alpes, 655 avenue de l'Europe, Montbonnot 38334, France  
{Navneet.Dalal,Bill.Triggs}@inrialpes.fr, <http://lear.inrialpes.fr>



2005



EVROPSKÁ UNIE  
Evropské strukturální a investiční fondy  
Operační program Výzkum, vývoj a vzdělávání



MINISTERSTVO ŠKOLSTVÍ,  
MLÁDEŽE A TĚLOVÝCHOVY

# Object Detection (Analysis)

## HOG





EVROPSKÁ UNIE  
Evropské strukturální a investiční fondy  
Operační program Výzkum, vývoj a vzdělávání



MINISTERSTVO ŠKOLSTVÍ,  
MLÁDEŽE A TĚLOVÝCHOVY

# Pedestrian Detection - Challenges?





## Histograms of Oriented Gradients (HOG)

- In recent years, the object detectors that are based on edge analysis that provides valuable information about the objects of interest were used in many detection tasks. In this area, the histograms of oriented gradients (HOG) [1] are considered as the state-of-the-art method.
- In HOG, a sliding window is used for detection. The window is divided into small connected cells in the process of obtaining HOG descriptors. The histograms of gradient orientations are calculated in each cell. It is desirable to normalize the histograms across a large block of image. As a result, a vector of values is computed for each position of window. This vector is then used for recognition, e.g. by the Support Vector Machine classifier.





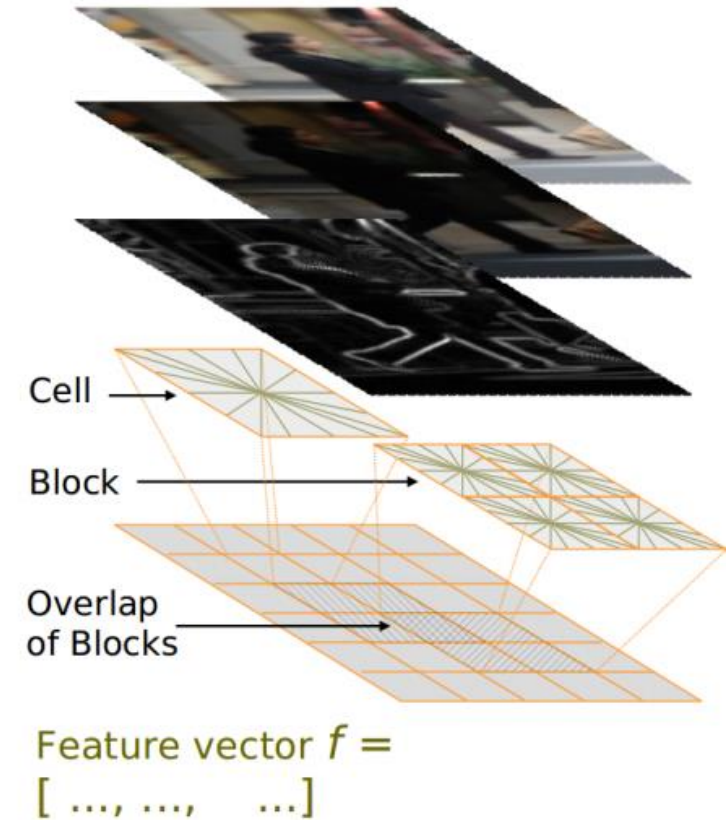
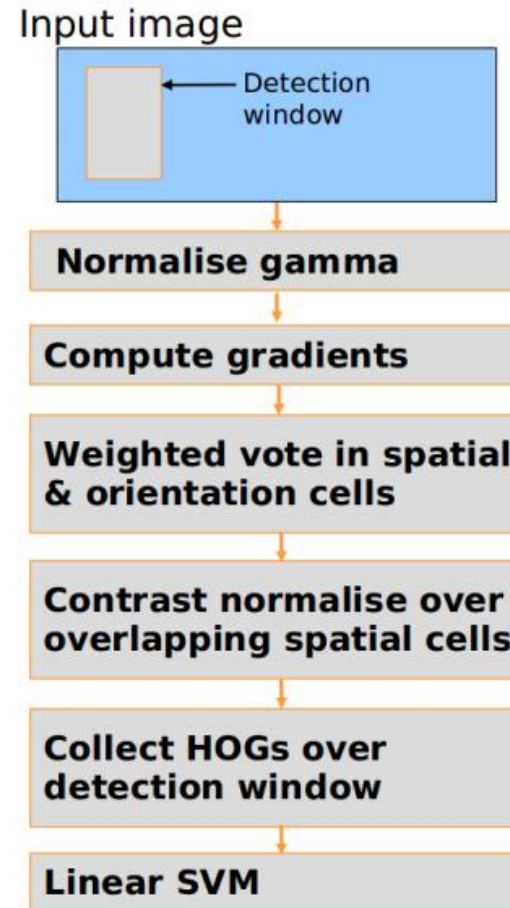
# Histograms of Oriented Gradients (HOG)

- Dalal and Trigs experimented with the size of detection window and they suggested the rectangular window with the size  $64 \times 128$  pixels. They also tried to reduce the size of the window to  $48 \times 112$  pixels. Nevertheless, they obtained the best detection result with the size  $64 \times 128$  pixels.

# Histograms of Oriented Gradients (HOG)

## Basic Steps:

- In HOG, a sliding window is used for detection.
- The window is divided into small connected cells.
- The histograms of gradient orientations are calculated in each cell.
- Support Vector Machine (SVM) classifier.



- For gradient computation, the image without Gaussian smoothing is filtered with the  $[1, 0, -1]$  kernel to compute the horizontal and vertical derivatives.
- Then the derivatives are used to compute the magnitude of the gradient and orientation .

$$D_X = [-1 \quad 0 \quad 1] \text{ and } D_Y = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \quad I_X = I * D_X \text{ and } I_Y = I * D_Y$$

magnitude of the gradient is  $|G| = \sqrt{I_X^2 + I_Y^2}$

orientation of the gradient is given by:  $\theta = \arctan \frac{I_Y}{I_X}$

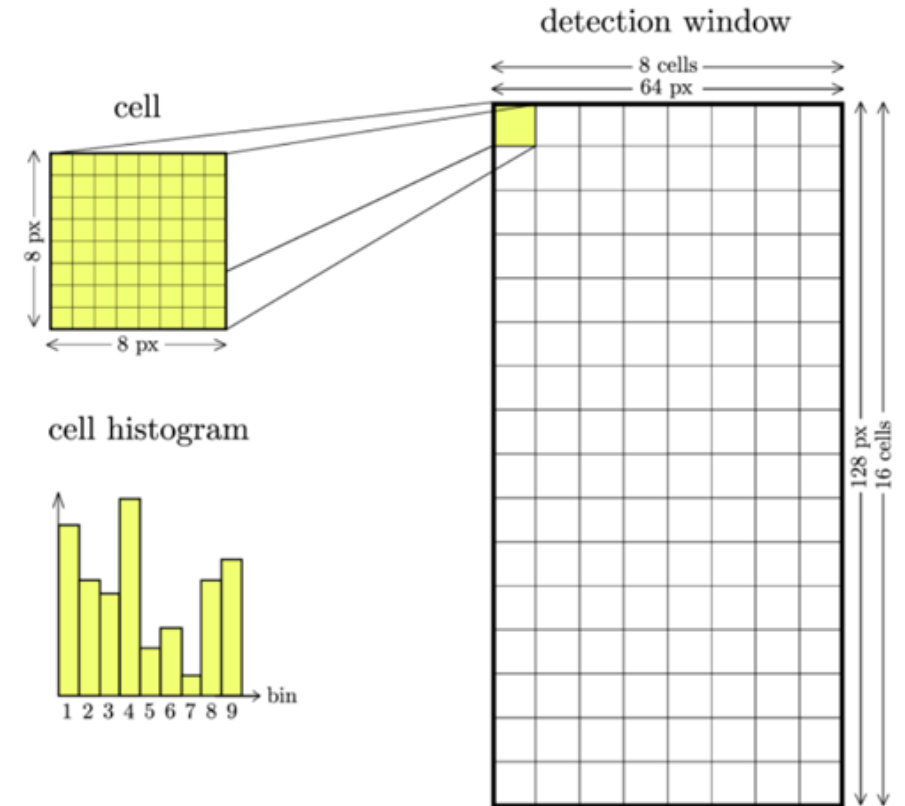
- In the next step, the image is divided into the cells and the cell histograms are constructed. The histogram bins are spread over 0 to 180 degrees or 0 to 360 degrees. The corresponding histogram bin is found for each pixel inside the cell. Each pixel contributes a weighted vote for its corresponding bin. The pixel contribution can be the gradient magnitude.
- Next step represents contrast normalization. For this purpose, the cells are grouped into the large blocks (i.e.  $2 \times 2$  cells are considered as blocks). The histograms are normalized within the blocks (e.g. using L2-norm). In the paper, the two main block geometries are presented; rectangular and circular.
- The final HOG descriptor is represented by histogram vectors of all blocks within the detection window

Dalal and Trigs experimented with the size of detection window and they suggested the rectangular window with the size  $64 \times 128$  pixels.

They also tried to reduce the size of the window to  $48 \times 112$  pixels. Nevertheless, they obtained the best detection result with the size  $64 \times 128$  pixels.

## Blocks, Cells:

- 8 x 8 cell
- 16 x 16 block – overlap
- normalization within the blocks
- Final Vector: Collect HOG blocks into vector





- The classical HOG descriptors suffer from the large number of features, which causes that the training and detection phases can be time consuming. The sufficient amount of training data is also needed to find a separating hyperplane by the SVM classifier.
- Sometimes, it is desirable to use the methods for the dimensionality reduction of feature vector. In addition to that, the classical HOG descriptors are not rotation invariant.
- These shortcomings became the motivation for creating many variations of HOG-based detectors. Many methods and applications based on HOG were presented in recent years.





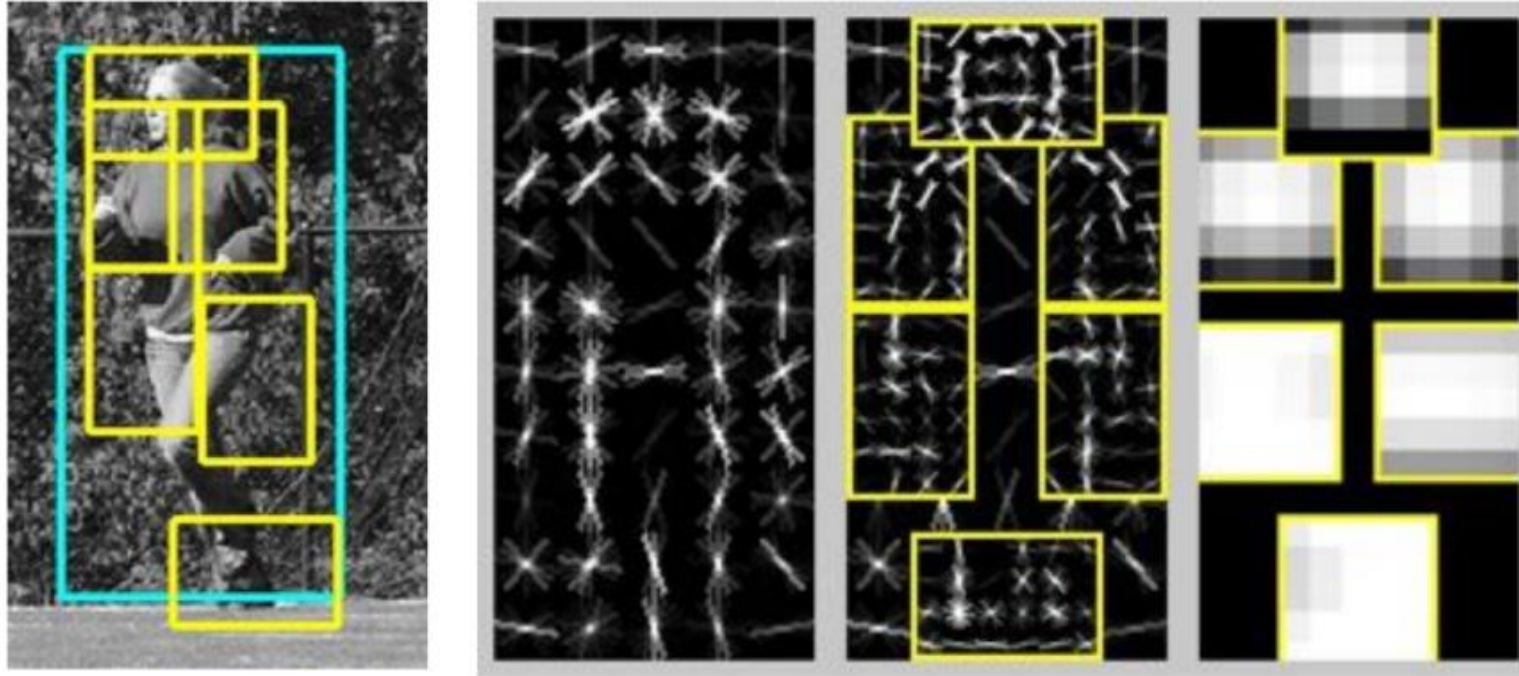
In [1], the authors applied the principal component analysis (PCA) to the HOG feature vector to obtain the PCA-HOG vector. This vector contains the subset of HOG features and the vector is used as an input for the SVM classifier. Their method was used for pedestrian detection with the satisfactory results.

Felzenszwalb et al. proposed the part-based detector that is based on HOG. In this method, the objects are represented using the mixtures of deformable HOG part models and these models are trained using a discriminative method (see following image). This method obtained excellent performance for object detection tasks [2, 3].

[1] Kobayashi, T., Hidaka, A., Kurita, T.: Neural information processing. chap. Selection of Histograms of Oriented Gradients Features for Pedestrian Detection, pp. 598–607. Springer-Verlag, Berlin, Heidelberg (2008)

[2] Felzenszwalb, P.F., McAllester, D.A., Ramanan, D.: A discriminatively trained, multiscale, deformable part model. In: CVPR (2008)

[3] Felzenszwalb, P., Girshick, R., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part-based models. Pattern Analysis and Machine Intelligence, IEEE Transactions on 32(9), 1627–1645 (2010)

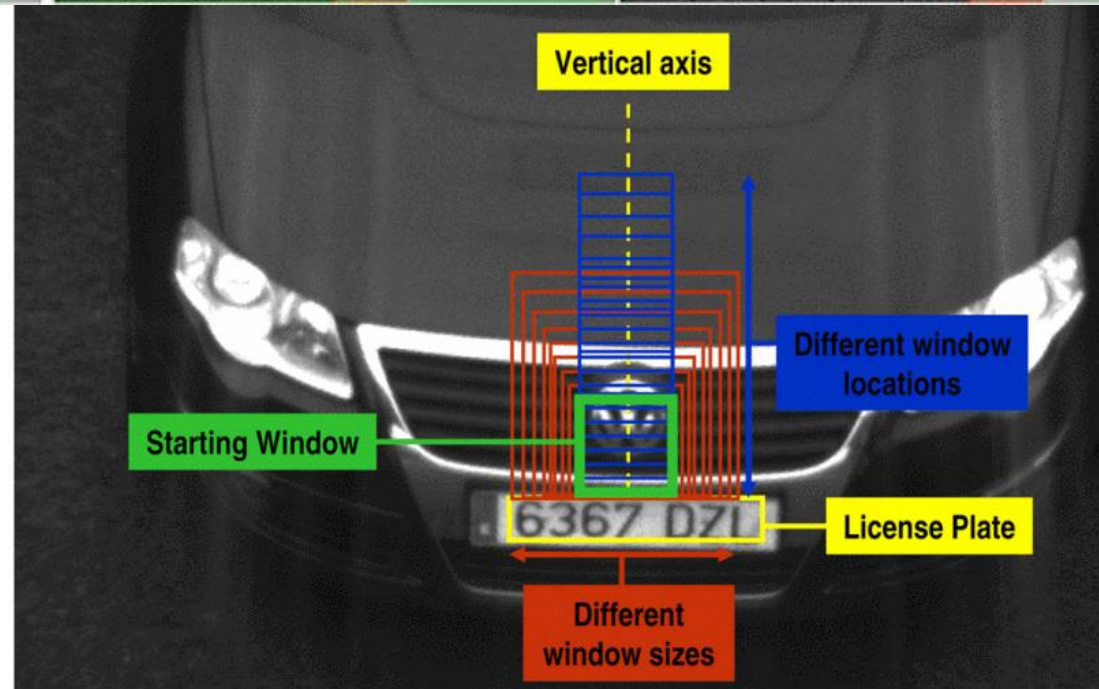


An example of person detection using a part model. The model is defined by the coarse global template that covers the entire object and higher resolution part templates. The templates represent the histogram of oriented gradient [2].

- [1] Kobayashi, T., Hidaka, A., Kurita, T.: Neural information processing. chap. Selection of Histograms of Oriented Gradients Features for Pedestrian Detection, pp. 598–607. Springer-Verlag, Berlin, Heidelberg (2008)
- [2] Felzenszwalb, P.F., McAllester, D.A., Ramanan, D.: A discriminatively trained, multiscale, deformable part model. In: CVPR (2008)
- [3] Felzenszwalb, P., Girshick, R., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part-based models. Pattern Analysis and Machine Intelligence, IEEE Transactions on 32(9), 1627–1645 (2010)



# Histograms of Oriented Gradients (HOG)

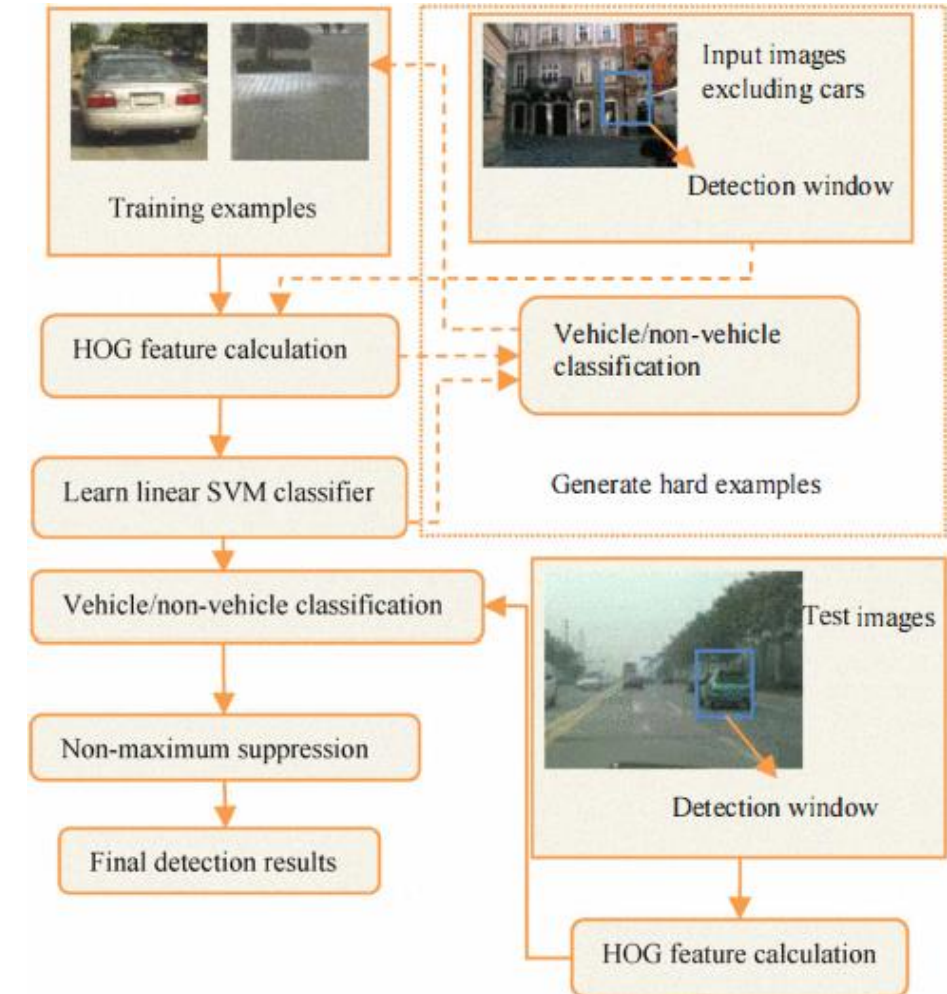
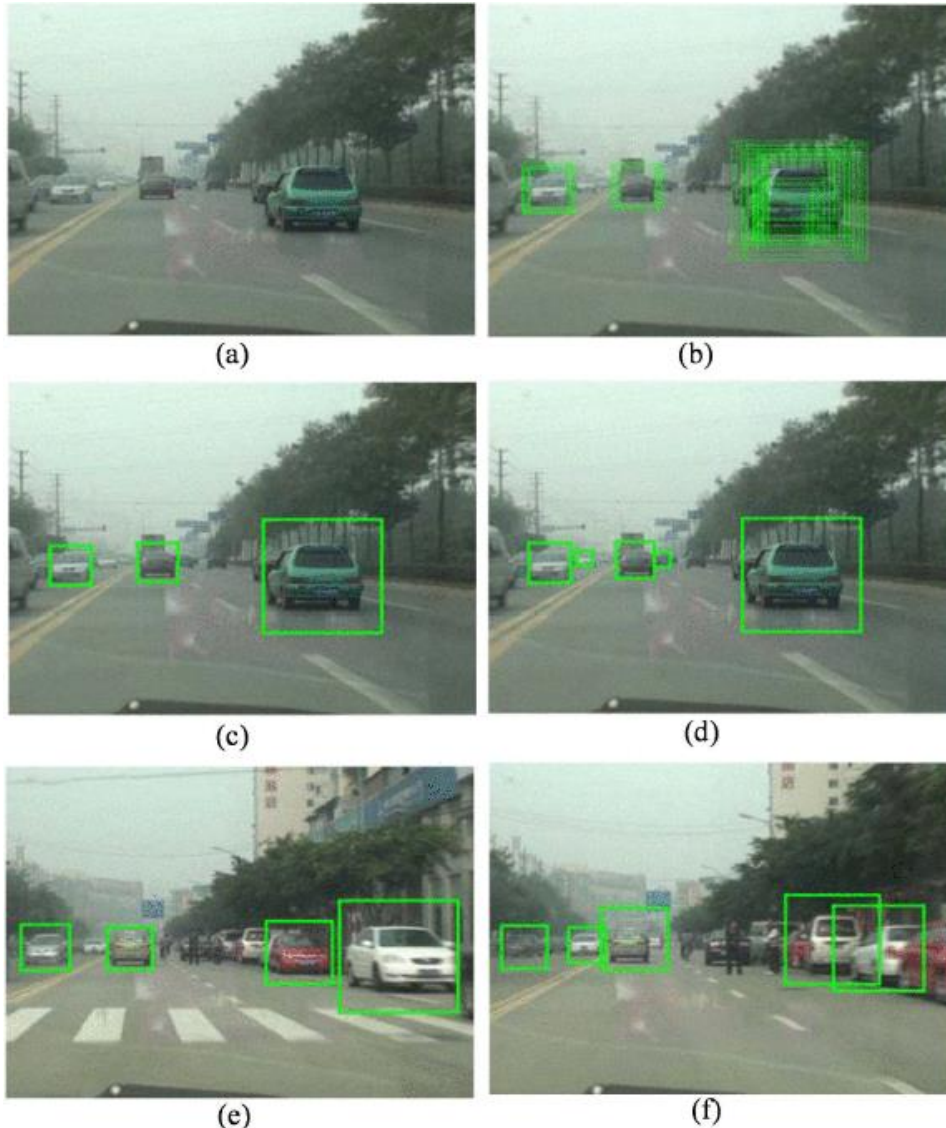


<https://ieeexplore.ieee.org/abstract/document/6728559>

D. F. Llorca, R. Arroyo and M. A. Sotelo, "Vehicle logo recognition in traffic images using HOG features and SVM," *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, The Hague, 2013, pp. 2229-2234.  
doi: 10.1109/ITSC.2013.6728559



# Histograms of Oriented Gradients (HOG) and Automotive



<https://ieeexplore.ieee.org/abstract/document/5581983>

L. Mao, M. Xie, Y. Huang and Y. Zhang, "Preceding vehicle detection using Histograms of Oriented Gradients," *2010 International Conference on Communications, Circuits and Systems (ICCCAS)*, Chengdu, 2010, pp. 354-358. doi: 10.1109/ICCCAS.2010.5581983

[Main Page](#)[Related Pages](#)[Modules](#)[Namespaces ▾](#)[Classes ▾](#)[Files ▾](#)[Examples](#)

## Introduction

OpenCV (Open Source Computer Vision Library: <http://opencv.org>) is an open-source BSD-licensed library that includes several hundreds of computer vision algorithms. The document describes the so-called OpenCV 2.x API, which is essentially a C++ API, as opposite to the C-based OpenCV 1.x API. The latter is described in `opencv1x.pdf`.

OpenCV has a modular structure, which means that the package includes several shared or static libraries. The following modules are available:

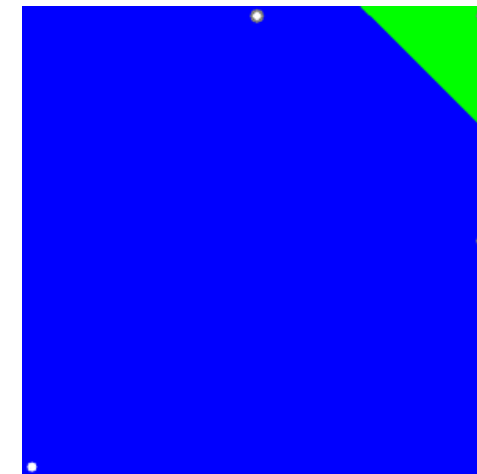
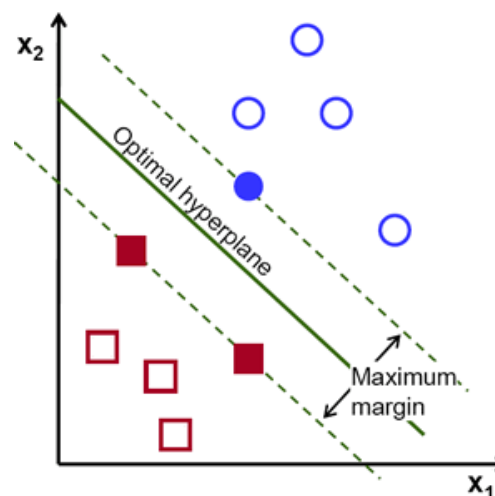
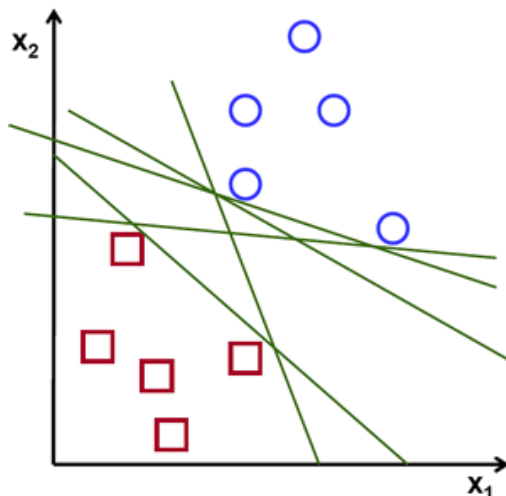
- **Core functionality** - a compact module defining basic data structures, including the dense multi-dimensional array `Mat` and basic functions used by all other modules.
- **Image processing** - an image processing module that includes linear and non-linear image filtering, geometrical image transformations (resize, affine and perspective warping, generic table-based remapping), color space conversion, histograms, and so on.
- **video** - a video analysis module that includes motion estimation, background subtraction, and object tracking algorithms.
- **calib3d** - basic multiple-view geometry algorithms, single and stereo camera calibration, object pose estimation, stereo correspondence algorithms, and elements of 3D reconstruction.
- **features2d** - salient feature detectors, descriptors, and descriptor matchers.
- **objdetect** - detection of objects and instances of the predefined classes (for example, faces, eyes, mugs, people, cars, and so on).
- **highgui** - an easy-to-use interface to simple UI capabilities.
- **Video I/O** - an easy-to-use interface to video capturing and video codecs.
- **gpu** - GPU-accelerated algorithms from different OpenCV modules.
- ... some other helper modules, such as FLANN and Google test wrappers, Python bindings, and others.

The further chapters of the document describe functionality of each module. But first, make sure to get familiar with the common API concepts used thoroughly in the library.



# Detection step HOG+SVM (OpenCV)

```
1 // Set up training data
2 int labels[4] = {1, -1, -1, -1};
3 Mat labelsMat(4, 1, CV_32SC1, labels);
4
5 float trainingData[4][2] = { {501, 10}, {255, 10}, {501, 255}, {10, 501} };
6 Mat trainingDataMat(4, 2, CV_32FC1, trainingData);
7
8 // Set up SVM's parameters
9 SVM::Params params;
10 params.svmType = SVM::C_SVC;
11 params.kernelType = SVM::LINEAR;
12 params.termCrit = TermCriteria(TermCriteria::MAX_ITER, 100, 1e-6);
13
14 // Train the SVM
15 Ptr<SVM> svm = StatModel::train<SVM>(trainingDataMat, ROW_SAMPLE, labelsMat, params);
```

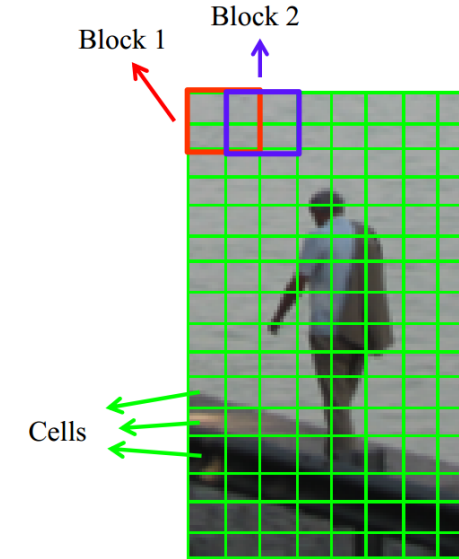


[https://docs.opencv.org/3.1.0/d1/d73/tutorial\\_introduction\\_to\\_svm.html](https://docs.opencv.org/3.1.0/d1/d73/tutorial_introduction_to_svm.html)

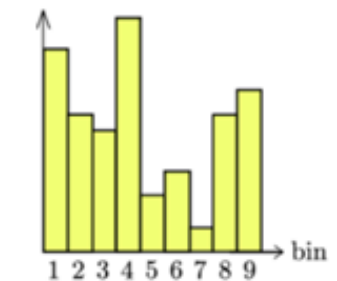


# Sliding Window (detectMultiScale)

```
1  int blockSize = 16;  
2  int cellSize = 8;  
3  int strideSize = 8;  
4  int winSize = 64;  
5  
6  //HOGDescriptor hog;  
7  HOGDescriptor my_hog(  
8      cv::Size(winSize,winSize), //winSize  
9      cv::Size(blockSize,blockSize), //blocksize  
10     cv::Size(strideSize,strideSize), //blockStride,  
11     cv::Size(cellSize,cellSize), //cellSize,  
12     9, //nbins,  
13     );  
14  
15     //SVM  
16     Ptr<SVM> svm = StatModel::load<SVM>( classifierName );  
17     std::vector< float > hog_detector;  
18     //get the support vectors  
19     get_svm_detector( svm, hog_detector );  
20     //set SVM  
21     my_hog.setSVMDetector( hog_detector );  
22  
23     std::vector<Rect> positivesAll;  
24     my_hog.detectMultiScale( frameGray, positivesAll, 0,  
25         Size(0,0), Size(0,0), 1.1, 4);
```



cell histogram



[https://github.com/opencv/opencv/blob/master/samples/cpp/train\\_HOG.cpp](https://github.com/opencv/opencv/blob/master/samples/cpp/train_HOG.cpp)



- Haar

- HOG

- LBP

- SIFT, SURF

- CNNs



Traditional Approaches



KeyPoints



Deep Learning Approach

## Related Works

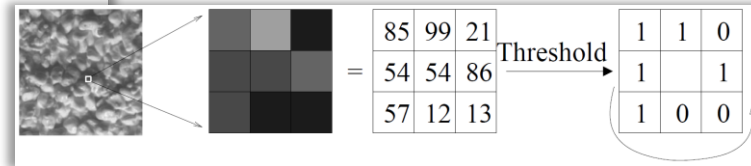
2006

Ahonen at al.  
(2006)  
1300 cit. SCOPUS

### Face Description with Local Binary Patterns:

#### Application to Face Recognition

Timo Ahonen, *Student Member, IEEE*, Abdenour Hadid,  
and Matti Pietikäinen, *Senior Member, IEEE*

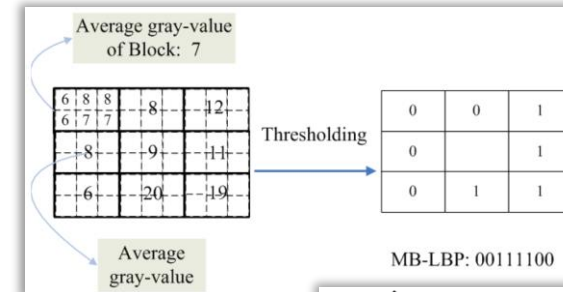


Zhang at al.  
(2007)

### Face Detection Based on Multi-Block LBP Representation

Lun Zhang, Rufeng Chu, Shiming Xiang, Shengcai Liao, Stan Z. Li

Center for Biometrics and Security Research & National Laboratory of Pattern Recognition  
Institute of Automation, Chinese Academy of Sciences  
95 Zhongguancun Donglu Beijing 100080, China



Xiaohua at al.  
(2009)

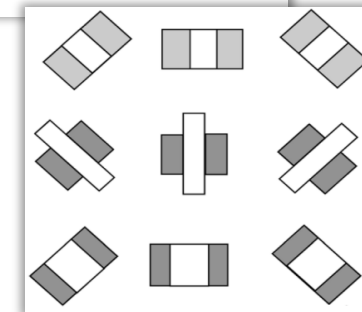
### Face detection using simplified Gabor features and hierarchical regions in a cascade of classifiers

Li Xiaohua<sup>a,b</sup>, Kin-Man Lam<sup>b,\*</sup>, Shen Lansun<sup>c</sup>, Zhou Jiliu<sup>a</sup>

<sup>a</sup>Department of Computer Science, Sichuan University, Chengdu 610064, China

<sup>b</sup>Centre for Signal Processing, Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong

<sup>c</sup>Signal and Information Processing Lab., Beijing University of Technology, Beijing 100022, China



2009



EVROPSKÁ UNIE  
Evropské strukturální a investiční fondy  
Operační program Výzkum, vývoj a vzdělávání



MINISTERSTVO ŠKOLSTVÍ,  
MLÁDEŽE A TĚLOVÝCHOVY

# Object Detection (Analysis)

## LBP

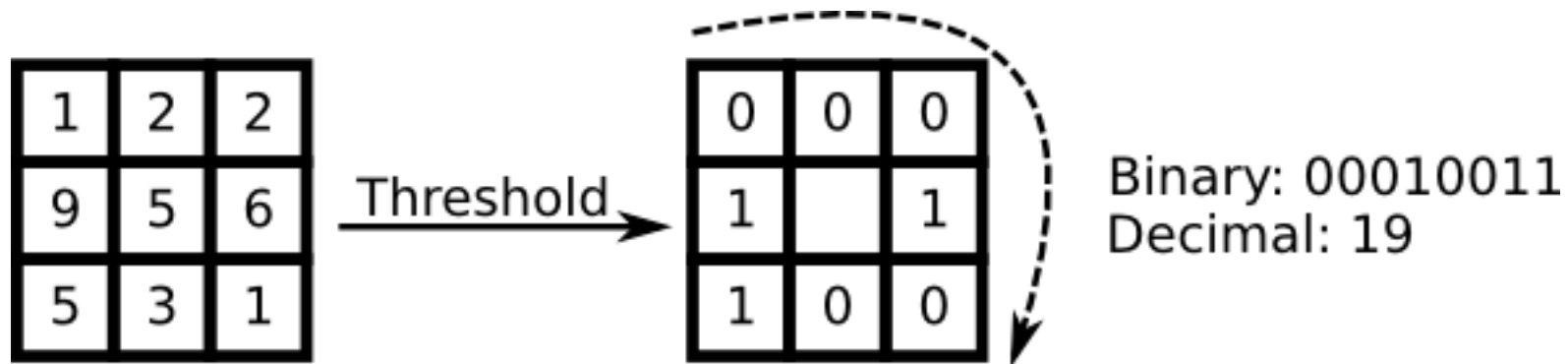


# LBP - Local Binary Patterns

- Were introduced by Ojala et al. for the texture analysis.
- The local binary patterns (LBP) were introduced by Ojala et al. [2, 3] for the texture analysis. The main idea behind LBP is that the local image structures (micro patterns such as lines, edges, spots, and flat areas) can be efficiently encoded by comparing every pixel with its neighboring pixels. In the basic form, every pixel is compared with its neighbors in the  $3 \times 3$  region. The result of comparison is the 8-bit binary number for each pixel; in the 8-bit binary number, the value 0 means that the value of center pixel is greater than the neighbor and vice versa. The histogram of these binary numbers (that are usually converted to decimal) is then used to encode the appearance of region.

# LBP - Local Binary Patterns

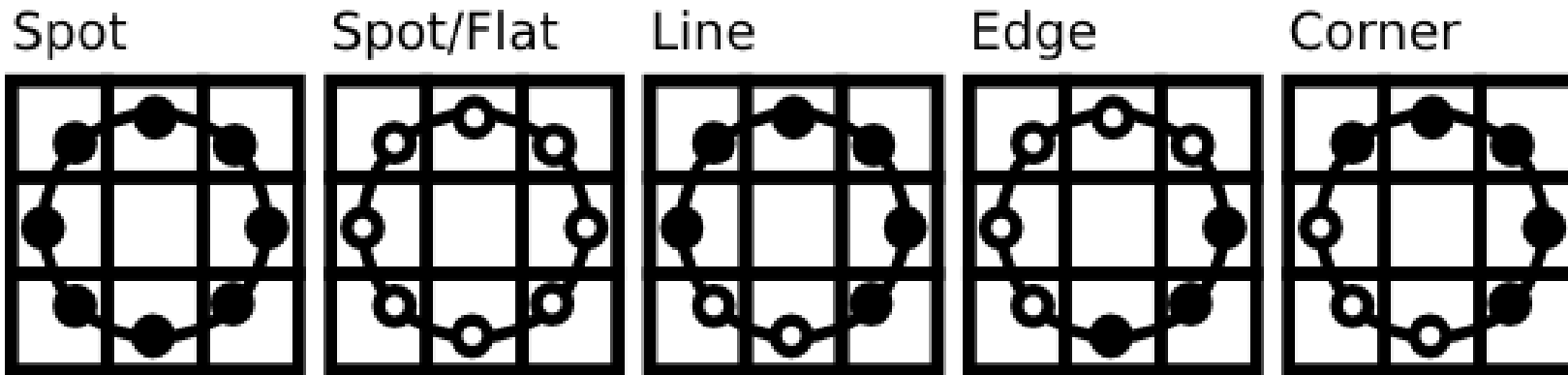
- The important properties of LBP are the resistance to the lighting changes and a low computational complexity.
- Due to their properties, LBP were used in many detection tasks, especially in facial image analysis [1, 4].







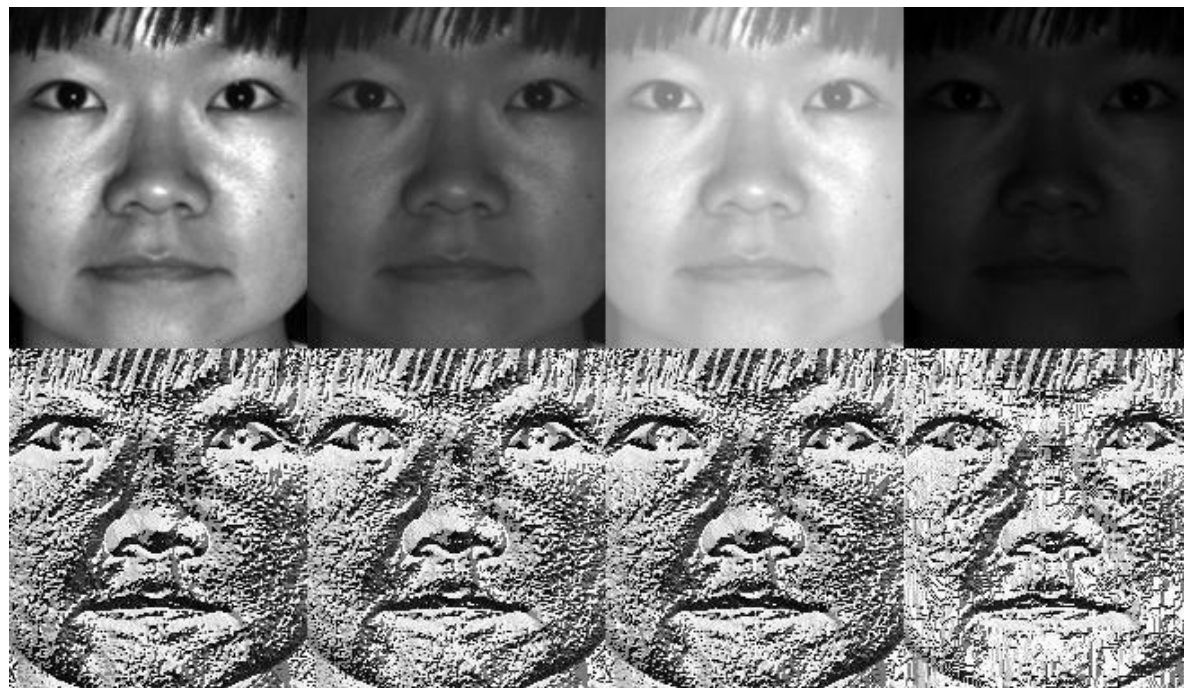
# LBP - Local Binary Patterns





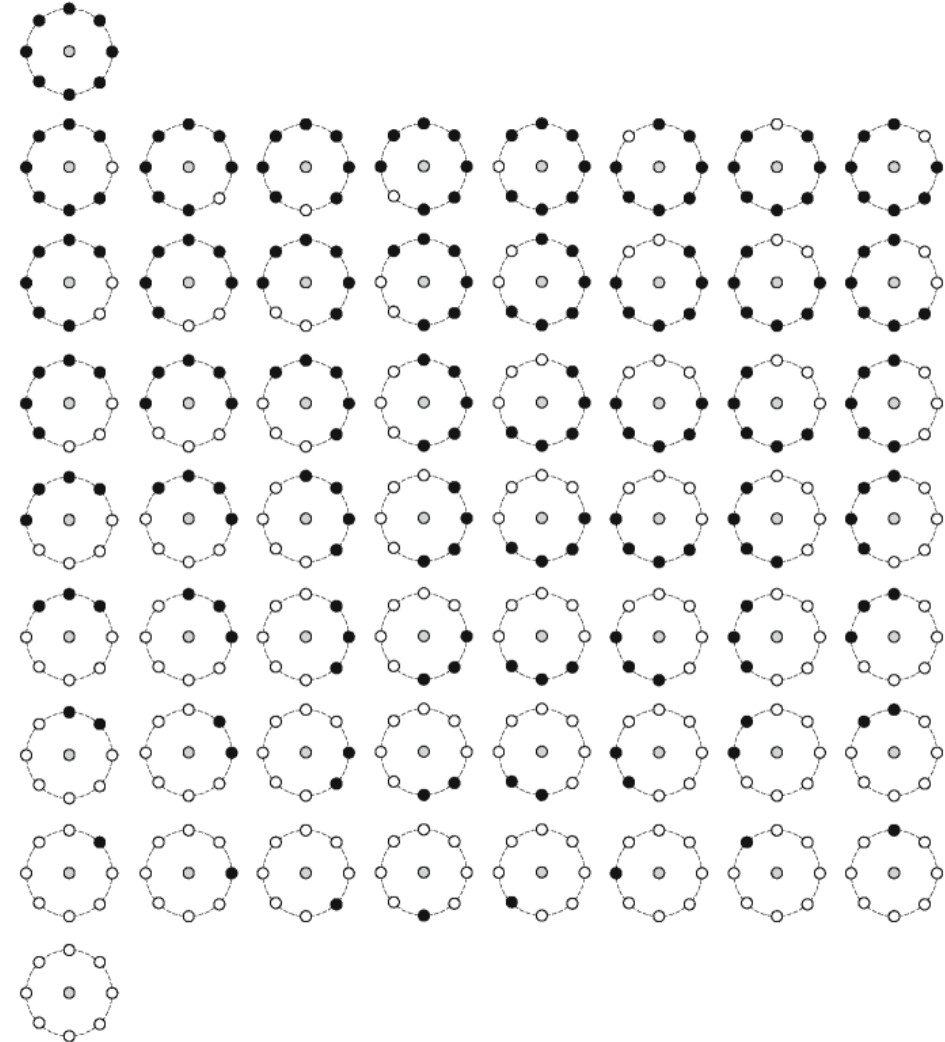
# LBP - Local Binary Patterns

- Robust to monotonic changes in illumination



# LBP - Local Binary Patterns (uniform)

- **Uniform** patterns example:
- 1 1 1 1 1 1 1 1 (0 transitions)
- 1 1 0 0 1 1 1 1 (2 transitions)
- **Non-uniform** patterns example:
- more than two transitions
- 0 0 1 0 0 1 0 1
- 0 0 0 0 0 1 0 1





# LBP - Local Binary Patterns

- [1] Ahonen, T., Hadid, A., Pietikainen, M.: Face description with local binary patterns: Application to face recognition. Pattern Analysis and Machine Intelligence, IEEE Transactions on 28(12), 2037–2041 (2006)
- [2] Ojala, T., Pietikainen, M., Harwood, D.: A comparative study of texture measures with classification based on featured distributions. Pattern Recognition 29(1), 51–59 (Jan 1996), [http://dx.doi.org/10.1016/0031-3203\(95\)00067-4](http://dx.doi.org/10.1016/0031-3203(95)00067-4)
- [3] Ojala, T., Pietikainen, M., Maenpaa, T.: A generalized local binary pattern operator for multiresolution gray scale and rotation invariant texture classification. In: Proceedings of the Second International Conference on Advances in Pattern Recognition. pp. 397–406. ICAPR '01, Springer-Verlag, London, UK, UK (2001), <http://dl.acm.org/citation.cfm?id=646260.685274>
- [4] Ahonen, T., Hadid, A., Pietikainen, M.: Face recognition with local binary patterns. In: Pajdla, T., Matas, J. (eds.) Computer Vision - ECCV 2004, Lecture Notes in Computer Science, vol. 3021, pp. 469–481. Springer Berlin Heidelberg (2004)

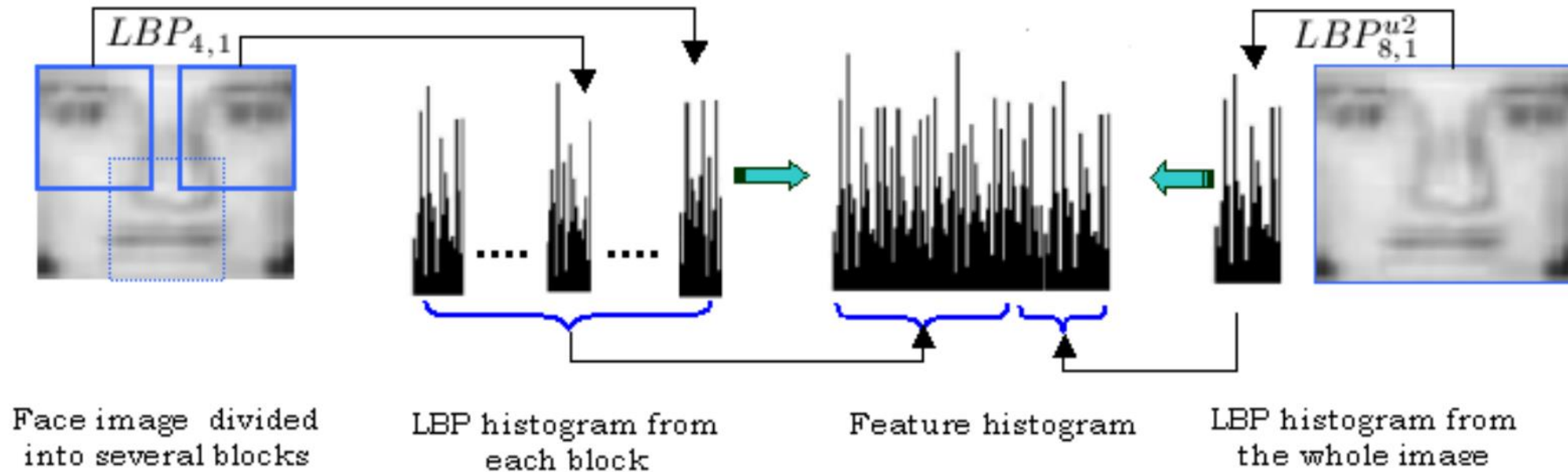


# LBP - Local Binary Patterns

In [1], LBP were used for solving the face detection problem in low-resolution images. In this approach, the  $19 \times 19$  face images are divided into the 9 overlapping regions in which the LBP descriptors are computed. Additionally, the LBP descriptors are extracted from the whole  $19 \times 19$  image. The descriptors are then used to create the feature vector, and the SVM classifier with a polynomial kernel is used for the final classification.

[1] Hadid, A., Pietikainen, M., Ahonen, T.: A discriminative feature space for detecting and recognizing faces. In: Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on. vol. 2, pp. II-797-II-804 Vol.2 (2004)

# LBP - Local Binary Patterns



[1] Hadid, A., Pietikainen, M., Ahonen, T.: A discriminative feature space for detecting and recognizing faces. In: Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on. vol. 2, pp. II-797-II-804 Vol.2 (2004)





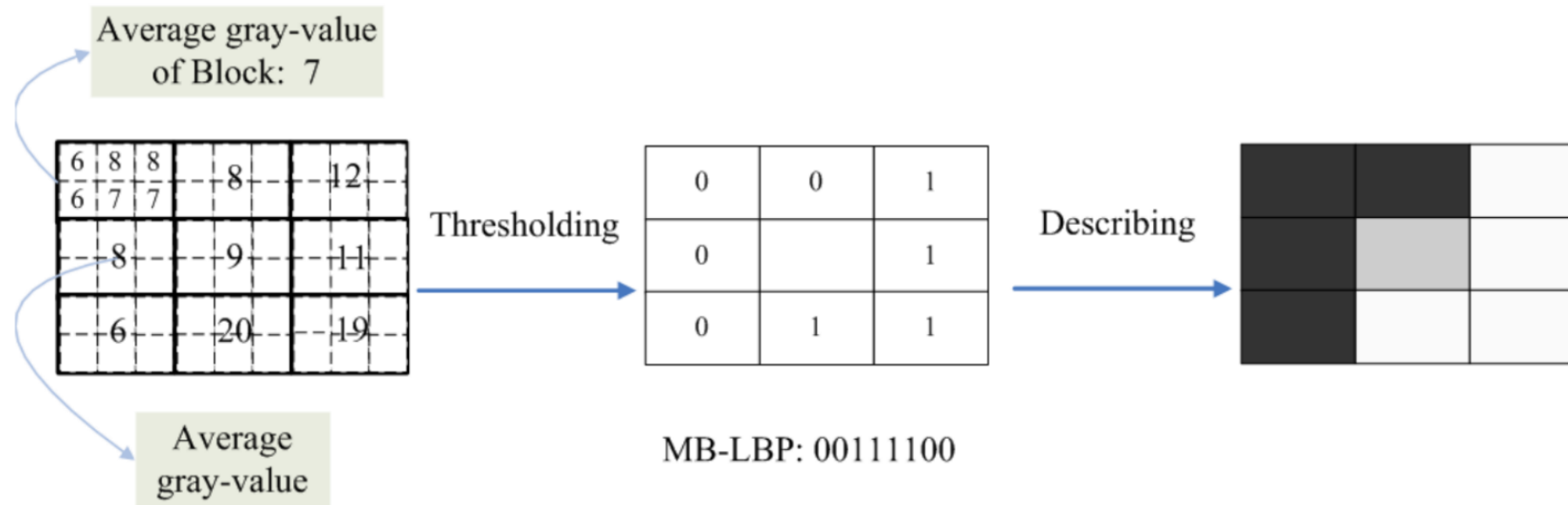
# LBP - Local Binary Patterns

Multi-block local binary patterns (MB-LBP) for face detection and recognition were proposed in [1, 2]. In this method, the authors encode the rectangular regions by the local binary pattern operator and the Gentle AdaBoost is used for feature selection. Their results showed that MBLBP are more distinctive than the Haar-like features and the original LBP features.

[1] Zhang, L., Chu, R., Xiang, S., Liao, S., Li, S.Z.: Face detection based on multi-block lbp representation. In: Proceedings of the 2007 international conference on Advances in Biometrics. pp. 11–18. ICB'07, Springer-Verlag, Berlin, Heidelberg (2007)

[2] Liao, S., Zhu, X., Lei, Z., Zhang, L., Li, S.Z.: Learning multi-scale block local binary patterns for face recognition. In: ICB. pp. 828–837 (2007)

# LBP - Local Binary Patterns



[1] Zhang, L., Chu, R., Xiang, S., Liao, S., Li, S.Z.: Face detection based on multi-block lbp representation. In: Proceedings of the 2007 international conference on Advances in Biometrics. pp. 11–18. ICB'07, Springer-Verlag, Berlin, Heidelberg (2007)

[2] Liao, S., Zhu, X., Lei, Z., Zhang, L., Li, S.Z.: Learning multi-scale block local binary patterns for face recognition. In: ICB. pp. 828–837 (2007)



# LBP - Local Binary Patterns

The paper of Tan and Triggs [2] proposed the face recognition method with robust preprocessing based on the difference of Gaussian image filter combined with LBP in which the binary LBP code is replaced by the ternary code to create local ternary patterns (LTP).

LBP were also successfully used for the facial expression analysis. The coarse-to-fine classification scheme with LBP combined with the k-nearest neighbor classifier that carries out the final classification was proposed in [1].

The comprehensive study of facial expression recognition using LBP was proposed in [78], the survey of facial image analysis using LBP was presented in [38].

- [1] Tan, X., Triggs, B.: Enhanced local texture feature sets for face recognition under difficult lighting conditions. *Image Processing, IEEE Transactions on* 19(6), 1635–1650 (2010)
- [2] Feng, X., Hadid, A., Pietikainen, M.: A coarse-to-fine classification scheme for facial expression recognition. In: Campilho, A., Kamel, M. (eds.) *Image Analysis and Recognition. Lecture Notes in Computer Science*, vol. 3212, pp. 668–675. Springer Berlin Heidelberg (2004)
- [3] Shan, C., Gong, S., McOwan, P.W.: Facial expression recognition based on local binary patterns: A comprehensive study. *Image Vision Comput.* 27(6), 803–816 (May 2009)
- [4] Huang, D., Shan, C., Ardabilian, M., Wang, Y., Chen, L.: Local binary patterns and its application to facial image analysis: A survey. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on* 41(6), 765–781 (Nov 2011)



- Haar

- HOG

- LBP

- SIFT, SURF

- CNNs



Traditional Approaches



KeyPoints



Deep Learning Approach



# KeyPoints

The most of the previously mentioned methods for object description were based on the fact that the descriptors were extracted over the whole image (sliding window) that was usually divided into the overlap or non-overlap regions. Inside these regions, the descriptors were calculated and combined to the final feature vector that was used as an input for the classifier.

In this lecture, we present the state-of-the-art descriptors that are based on the fact that the regions (within which the descriptors are extracted) are selected using the keypoint detectors.



## KeyPoints - SIFT

One of the most popular descriptors based on the interest points was proposed by David Lowe [1, 2, 3]. The method is called scale invariant feature transform (SIFT).

The idea of the SIFT descriptor is that the interesting points (keypoints) of the objects can be extracted to provide the key information about the objects. The gradient magnitude and orientation are computed around the keypoint location; the histograms are then summarized over subregions (see following image). The keypoints are extracted from the reference image (that contains the object of interest) and also from the target image (that possibly contains the object of interest). The extracted keypoints are matched to find similarity between the images.



# KeyPoints - SIFT

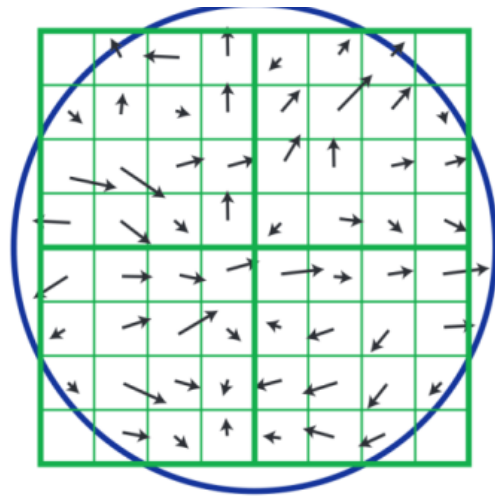
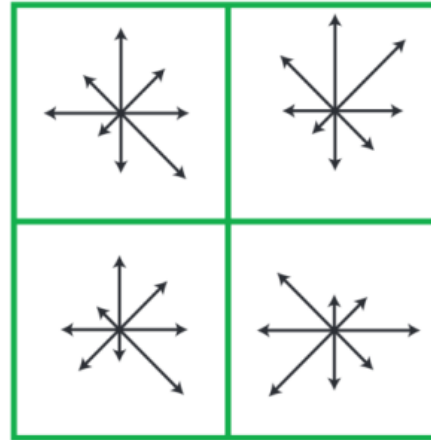


Image gradients



Keypoint descriptor

An example of SIFT keypoint descriptor in which the gradient orientation and gradient magnitude around each interest point are used [3].

[1] Brown, M., Lowe, D.: Invariant features from interest point groups. In: In British Machine Vision Conference. pp. 656–665 (2002)

[2] Lowe, D.: Object recognition from local scale-invariant features. In: Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on. vol. 2, pp. 1150–1157 vol.2 (1999)

[3] Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vision 60(2), 91–110 (Nov 2004), <http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94>



# KeyPoints - SURF

The speeded up robust feature (SURF) descriptor by Bay et al. [1, 2] is also one of the widely used keypoint descriptors. In this method, the Hessian matrix-based measure is used to find the points of interest. The sum of the Haar-wavelet responses within the neighborhood of interest point is calculated. The authors also use the fast calculation via the integral image thanks to which SURF is faster than SIFT.

[1] Bay, H., Tuytelaars, T., Gool, L.J.V.: Surf: Speeded up robust features. In: ECCV (1). pp. 404–417 (2006)

[2] Bay, H., Ess, A., Tuytelaars, T., Van Gool, L.: Speeded-up robust features (surf). Comput. Vis. Image Underst. 110(3), 346–359 (Jun 2008)



## KeyPoints – BRIEF/ORB

A very fast method called binary robust independent elementary features (BRIEF) was proposed by Calonder et al. [1]. The authors reported that the method outperforms SURF in the term of speed, and the recognition rate in many cases. In BRIEF, a binary string that contains the results of intensity differences of pixels are used and the descriptor similarity is evaluated using the Hamming distance. In [2], the authors proposed another binary descriptor with rotation and noise invariant properties called oriented fast and rotated BRIEF (ORB).

[1] Calonder, M., Lepetit, V., Strecha, C., Fua, P.: Brief: binary robust independent elementary features. In: Proceedings of the 11th European conference on Computer vision: Part IV. pp. 778–792. ECCV'10, Springer-Verlag, Berlin, Heidelberg (2010)

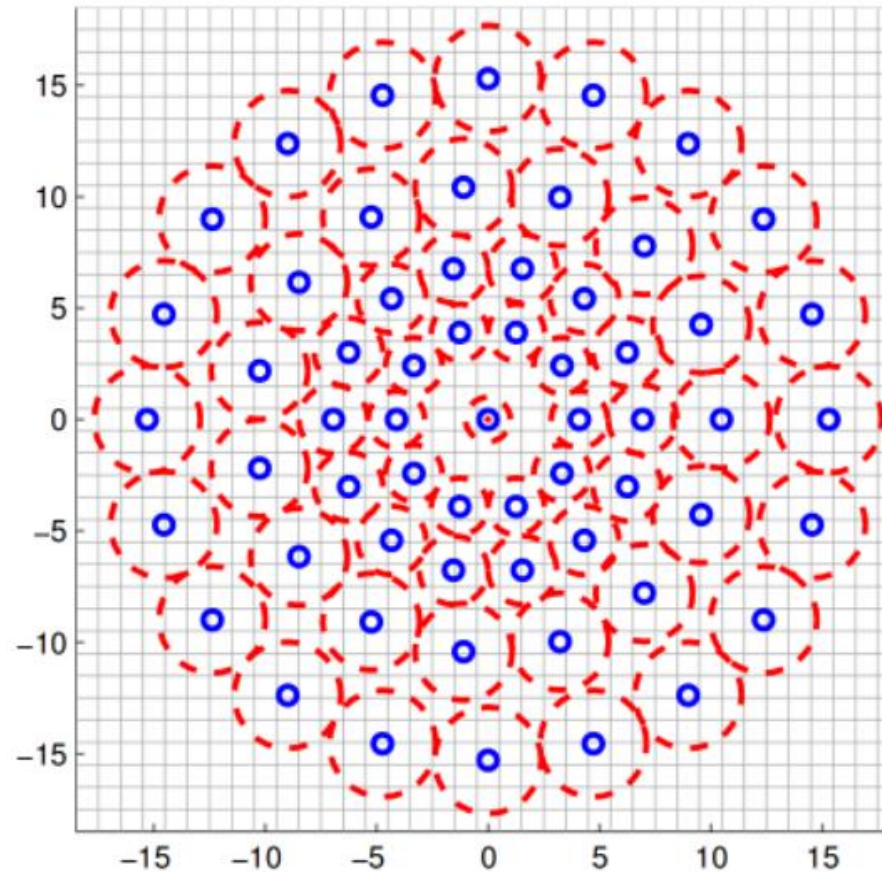
[2] Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: Orb: An efficient alternative to sift or surf. In: Computer Vision (ICCV), 2011 IEEE International Conference on. pp. 2564–2571 (2011)



## KeyPoints – BRISK

Leutenegger et al. [1] proposed binary robust invariant scalable keypoints (BRISK). The method provides both scale and rotation invariance. BRISK is a binary descriptor like BRIEF and ORB, it means that the binary string that represents a region around the keypoint is composed. In BRISK, a concentric circle pattern of points near to the keypoint is used (see following image). In this pattern, the blue circles represent the sampling locations and Gaussian blurring is computed to be less sensitive to noise; the radius of red circles denotes a standard deviation of blurring kernel. The standard deviation of the Gaussian kernel is increased with the increasing distance from the feature center to avoid aliasing effects. The final descriptor is determined by the comparison of sample points.

# KeyPoints – BRISK



BRISK sampling pattern [1]

[1] Leutenegger, S., Chli, M., Siegwart, R.: Brisk: Binary robust invariant scalable keypoints. In: Computer Vision (ICCV), 2011 IEEE International Conference on. pp. 2548–2555 (2011)



## KeyPoints – FREAK

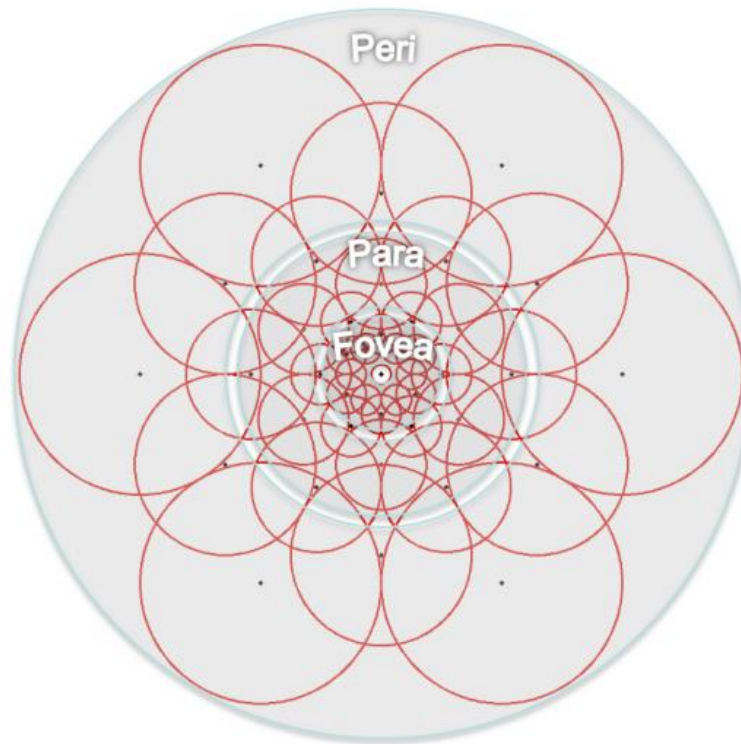
In [1], the authors proposed the fast retina keypoint (FREAK) descriptor that also uses the binary strings. The method is biologically inspired by a human visual system; more exactly by the retina. In this paper, the authors proposed a retinal sampling pattern; The following image shows the topology of this pattern. The pattern is divided into the areas (foveal, fovea, parafoveal, and perifoveal) similar to the human retina. In this pattern, the pixels are overlapped and concentrated near to the center. The binary strings is computed by comparing the point pairs of image intensities within the pattern.

[1] Alahi, A., Ortiz, R., Vandergheynst, P.: FREAK: Fast Retina Keypoint. In: IEEE Conference on Computer Vision and Pattern Recognition. IEEE Conference on Computer Vision and Pattern Recognition, Ieee, New York (2012)





# KeyPoints – FREAK

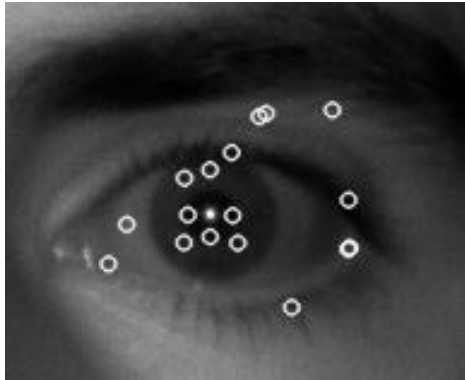


FREAK sampling pattern [1]

[1] Alahi, A., Ortiz, R., Vanderghyest, P.: FREAK: Fast Retina Keypoint. In: IEEE Conference on Computer Vision and Pattern Recognition. IEEE Conference on Computer Vision and Pattern Recognition, IEEE, New York (2012)

## KeyPoints - Example

The goal is to find image KeyPoints that are invariant in the terms of scale, orientation, position, illumination, partially occlusion.

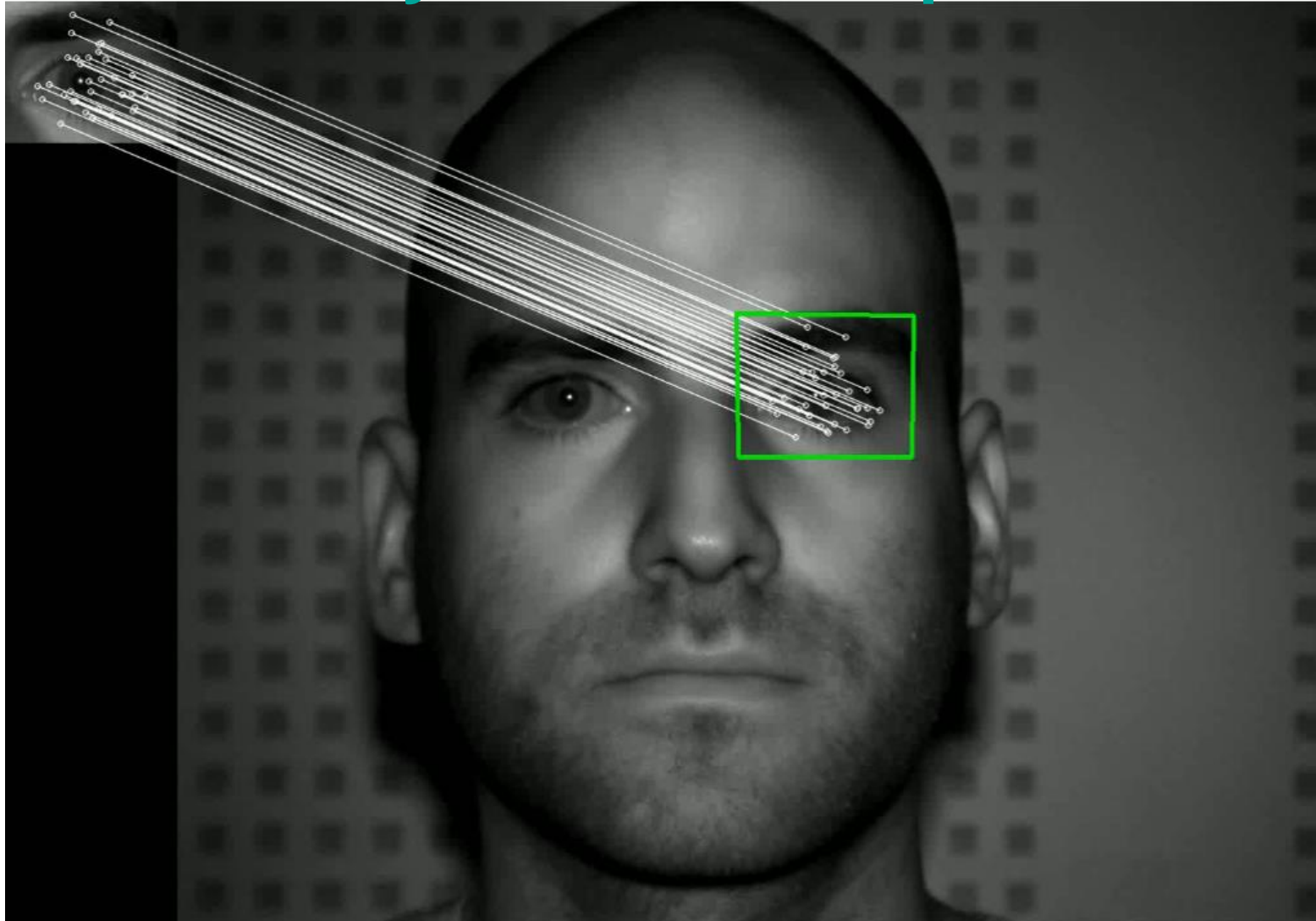


**template**





# Keypoints - Example



[https://docs.opencv.org/3.1.0/d5/d6f/tutorial\\_feature\\_flann\\_matcher.html](https://docs.opencv.org/3.1.0/d5/d6f/tutorial_feature_flann_matcher.html)



- Haar

- HOG

- LBP

- SIFT, SURF

- CNNs



Traditional Approaches



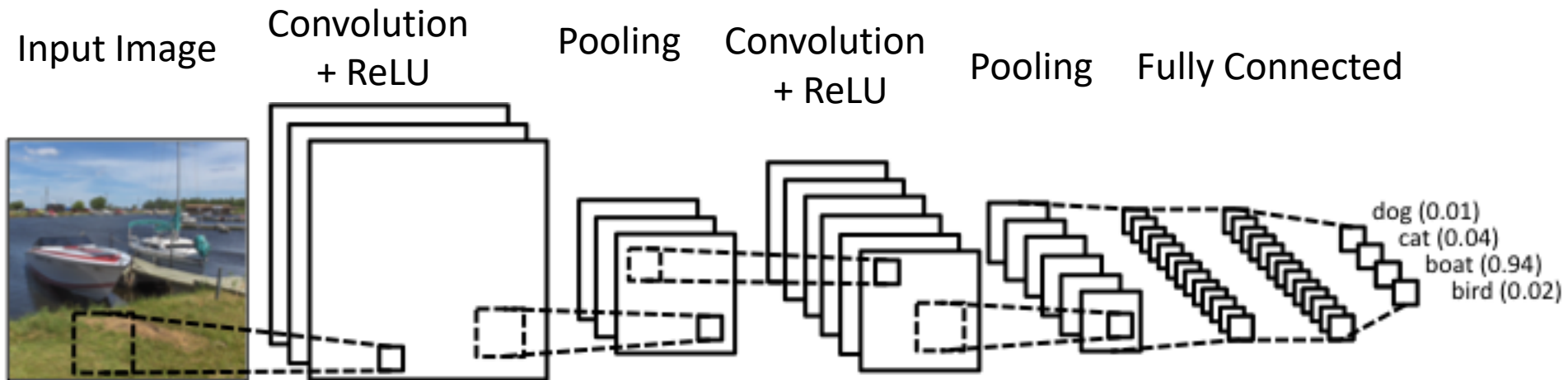
KeyPoints



Deep Learning Approach

# CNNs – Main Steps (LeNet type)

1. Convolution
2. Non Linearity (ReLU)
3. Pooling or Sub Sampling
4. Classification (Fully Connected Layer)



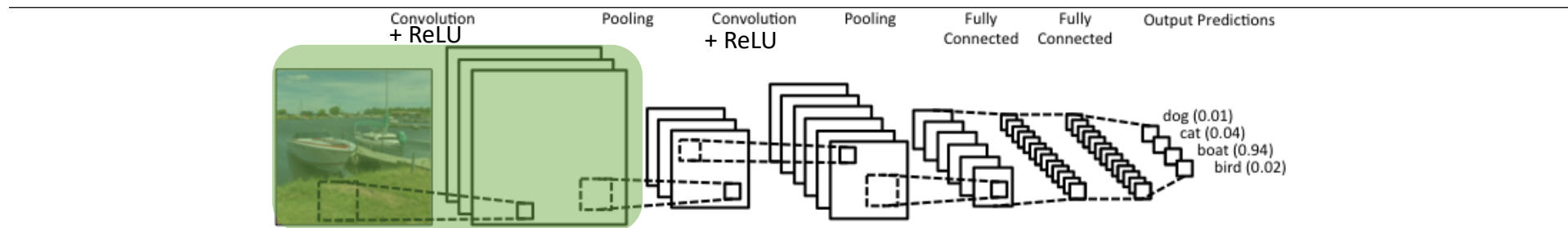
# 1. Convolution

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Input Image

1	0	1
0	1	0
1	0	1

Mask/Filter





# 1. Convolution

1	0	1
0	1	0
1	0	1

Mask/Filter

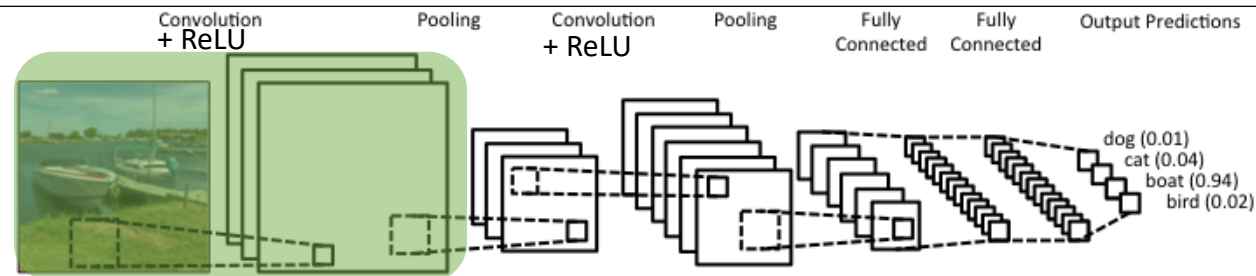
1 <sub>x1</sub>	1 <sub>x0</sub>	1 <sub>x1</sub>	0	0
0 <sub>x0</sub>	1 <sub>x1</sub>	1 <sub>x0</sub>	1	0
0 <sub>x1</sub>	0 <sub>x0</sub>	1 <sub>x1</sub>	1	1
0	0	1	1	0
0	1	1	0	0

Image

4		

Convolved  
Feature

Multiply the image pixels by pixels of the filter, then sum the results



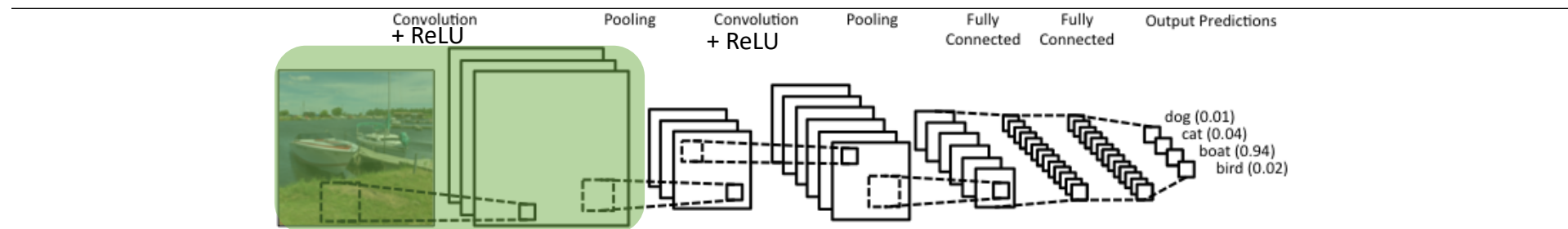
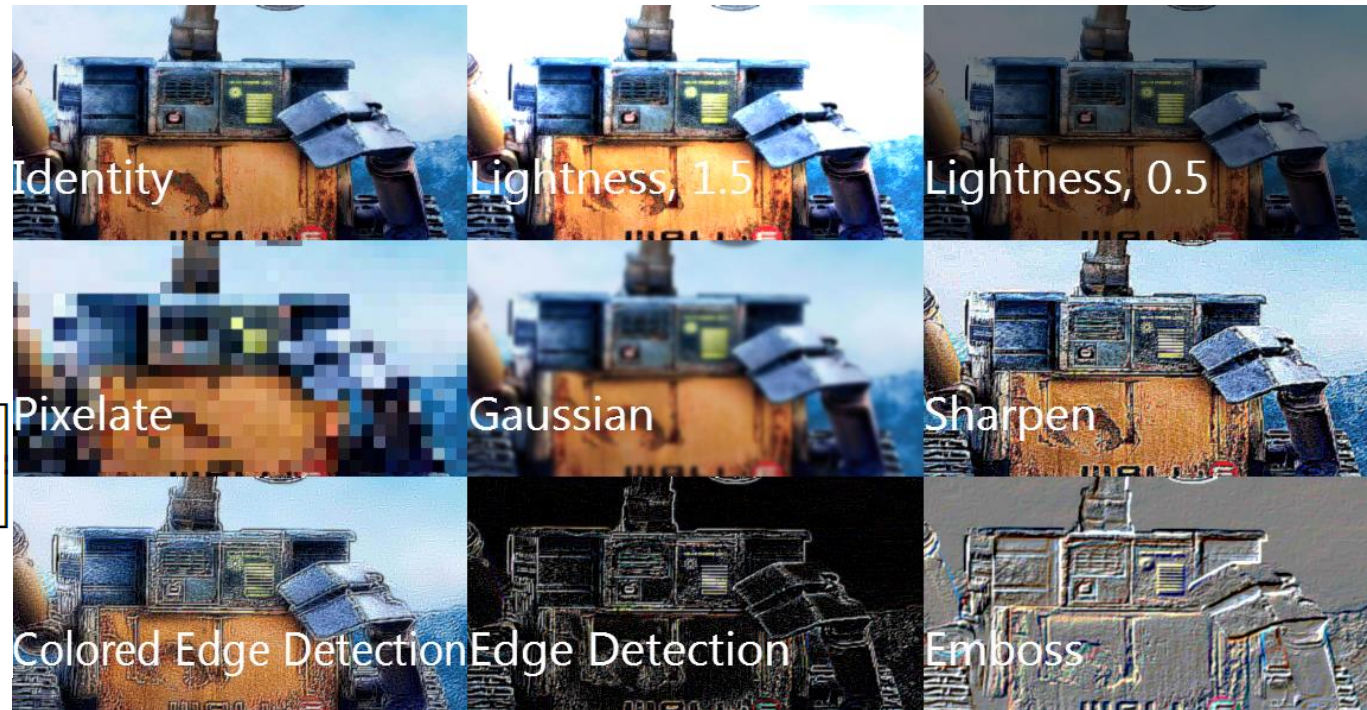
# 1. Convolution

$$\text{Sharpen} - \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

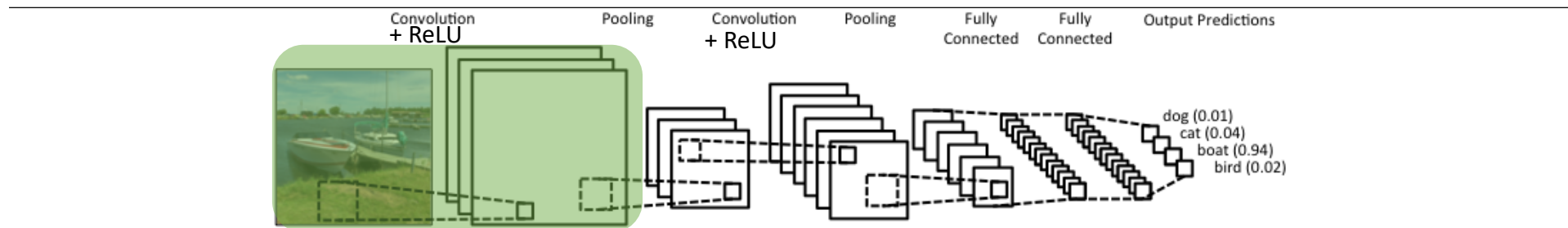
$$\text{Blur} - \begin{bmatrix} 0 & 0.2 & 0 \\ 0.2 & 0.2 & 0.2 \\ 0 & 0.2 & 0 \end{bmatrix}$$

$$\text{Horizontal Motion Blur} - \begin{bmatrix} 0 & 0 & 0 \\ 0.2 & 0.2 & 0.2 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\text{Edge detect} - \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$



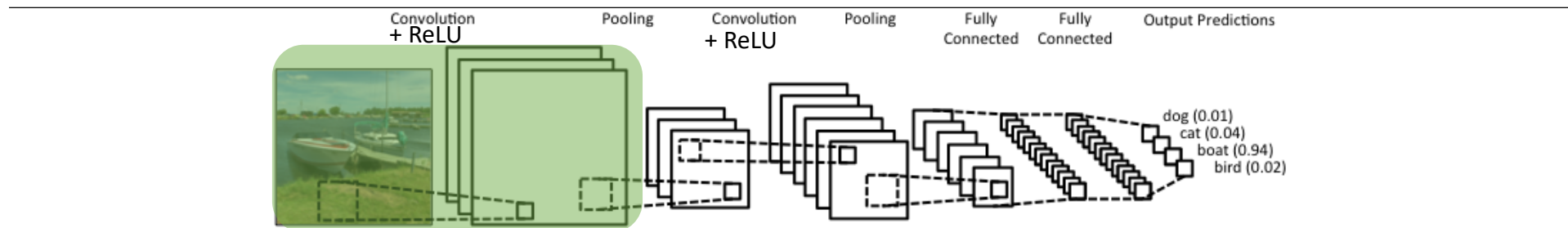
# 1. Convolution



[http://cs.nyu.edu/~tengus/tutorials/deep\\_learning\\_cvpr12/](http://cs.nyu.edu/~tengus/tutorials/deep_learning_cvpr12/)

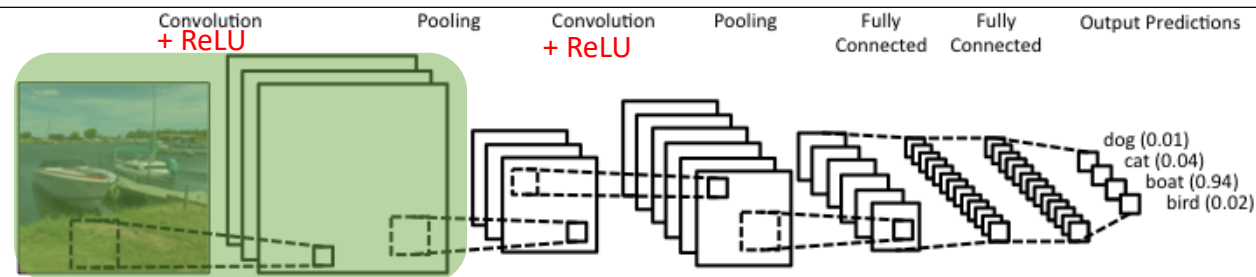
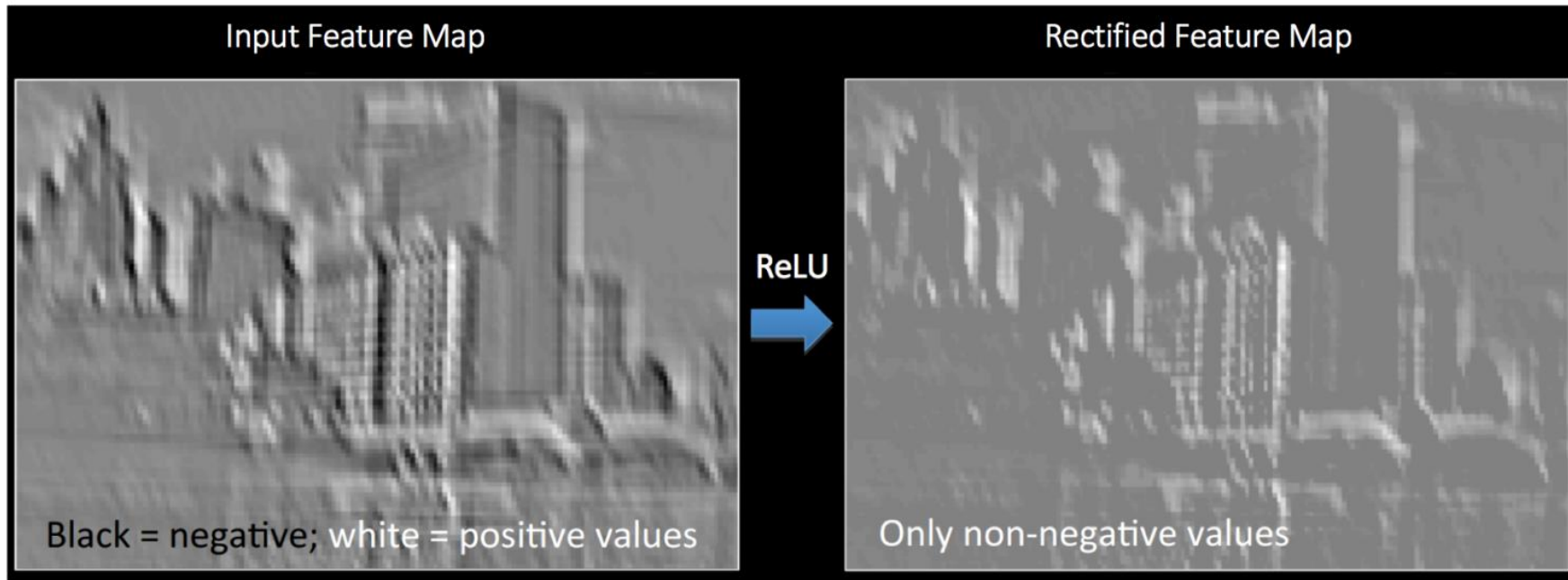
# 1. Convolution

- Before training, we have many filters/kernels
  - Filter values are randomized
- Depth of this conv. layer corresponds to the number of filters we use for the convolution operation
- The filters are learned during the training



## 2. Non Linearity (ReLU)

- ReLU is used after every Convolution operation
- The goal of this step is to replace all negative pixels by zero in the feature map

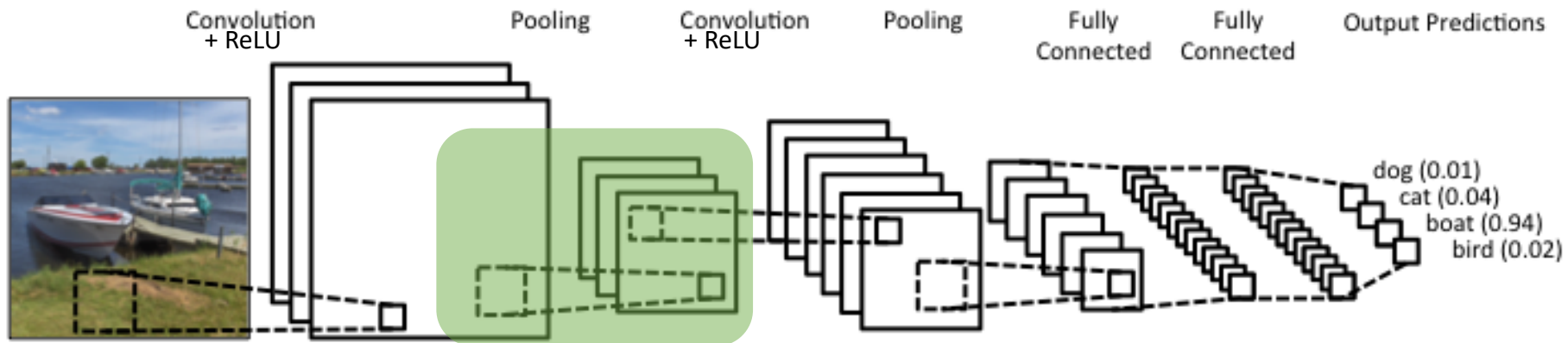




# 3. Pooling

( Subsampling or downsampling )

- The goal of this step is to reduce the dimensionality of each feature map but preserve important informations
- Operations: e.g. Sum, Average, Max

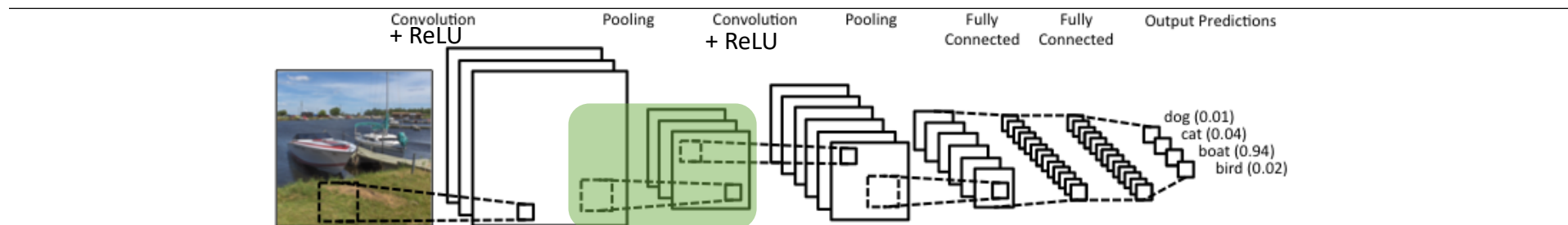
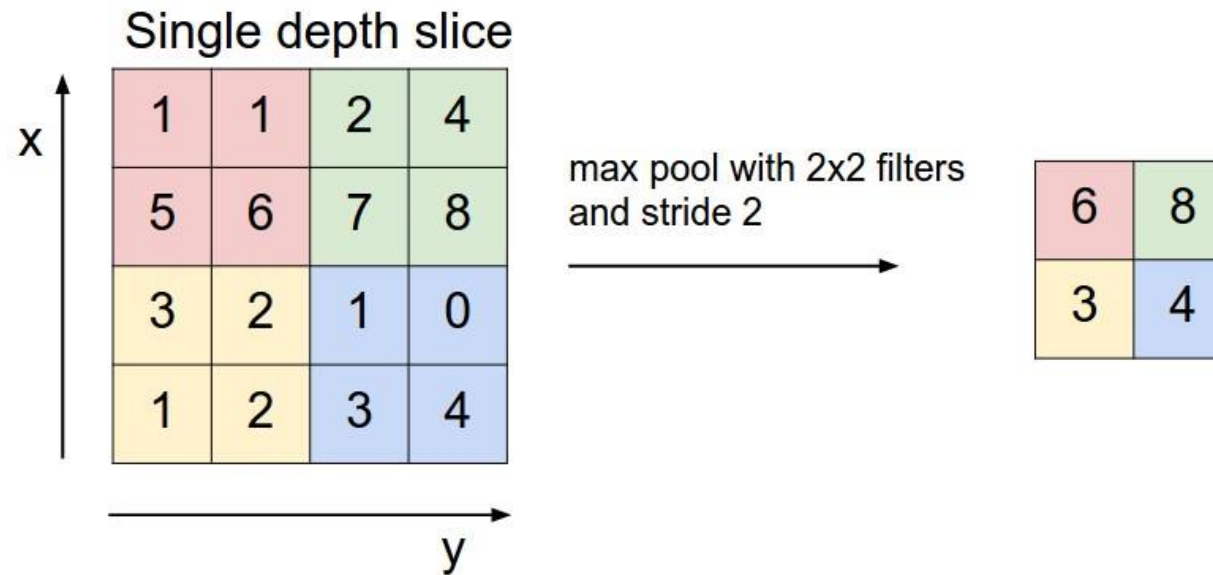




# 3. Pooling

( Subsampling or downsampling )

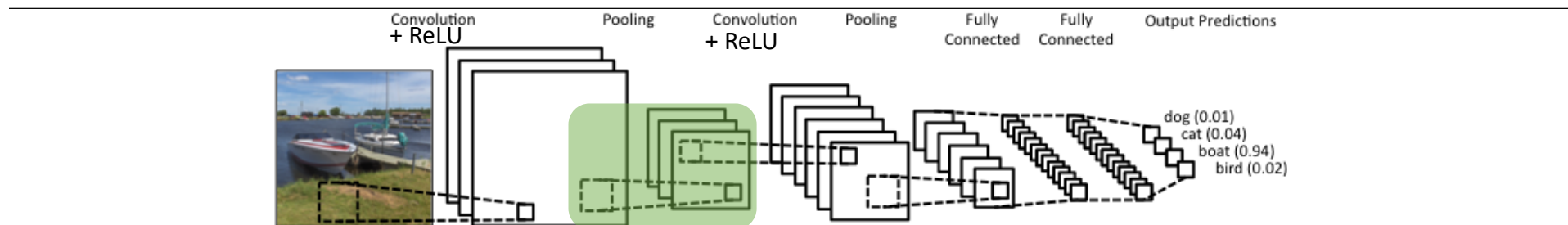
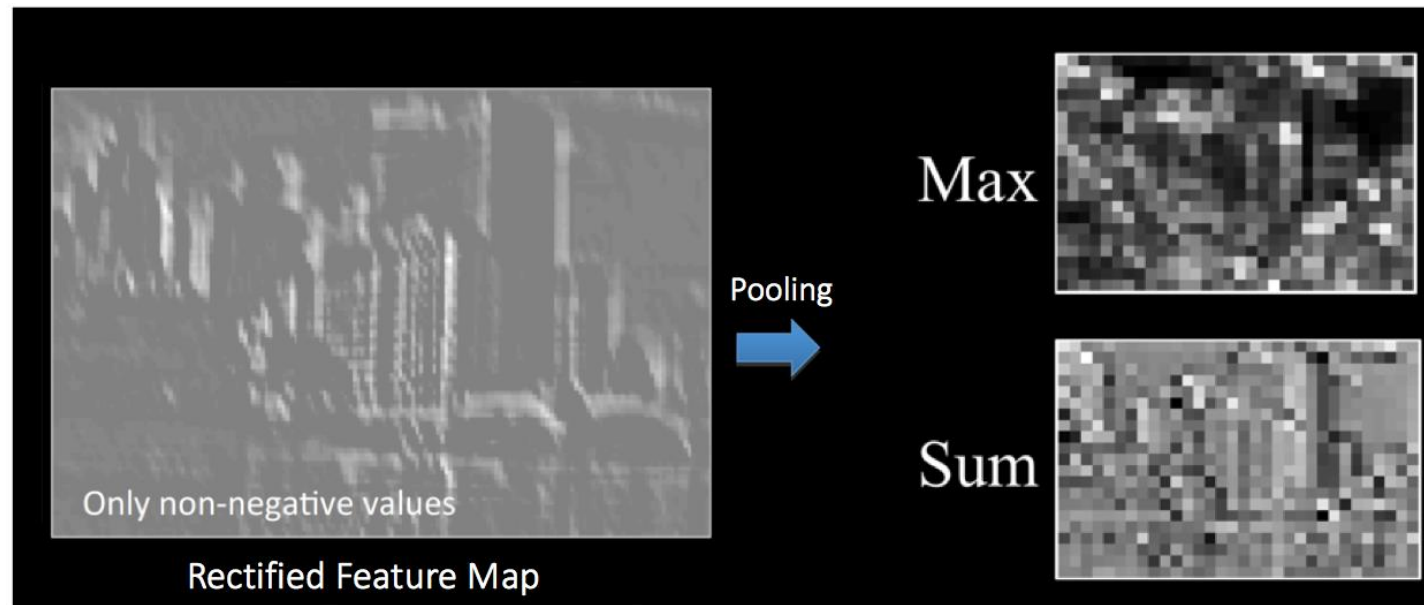
- Common way is a pooling layer with filters of size 2x2 applied with a stride of 2



# 3. Pooling

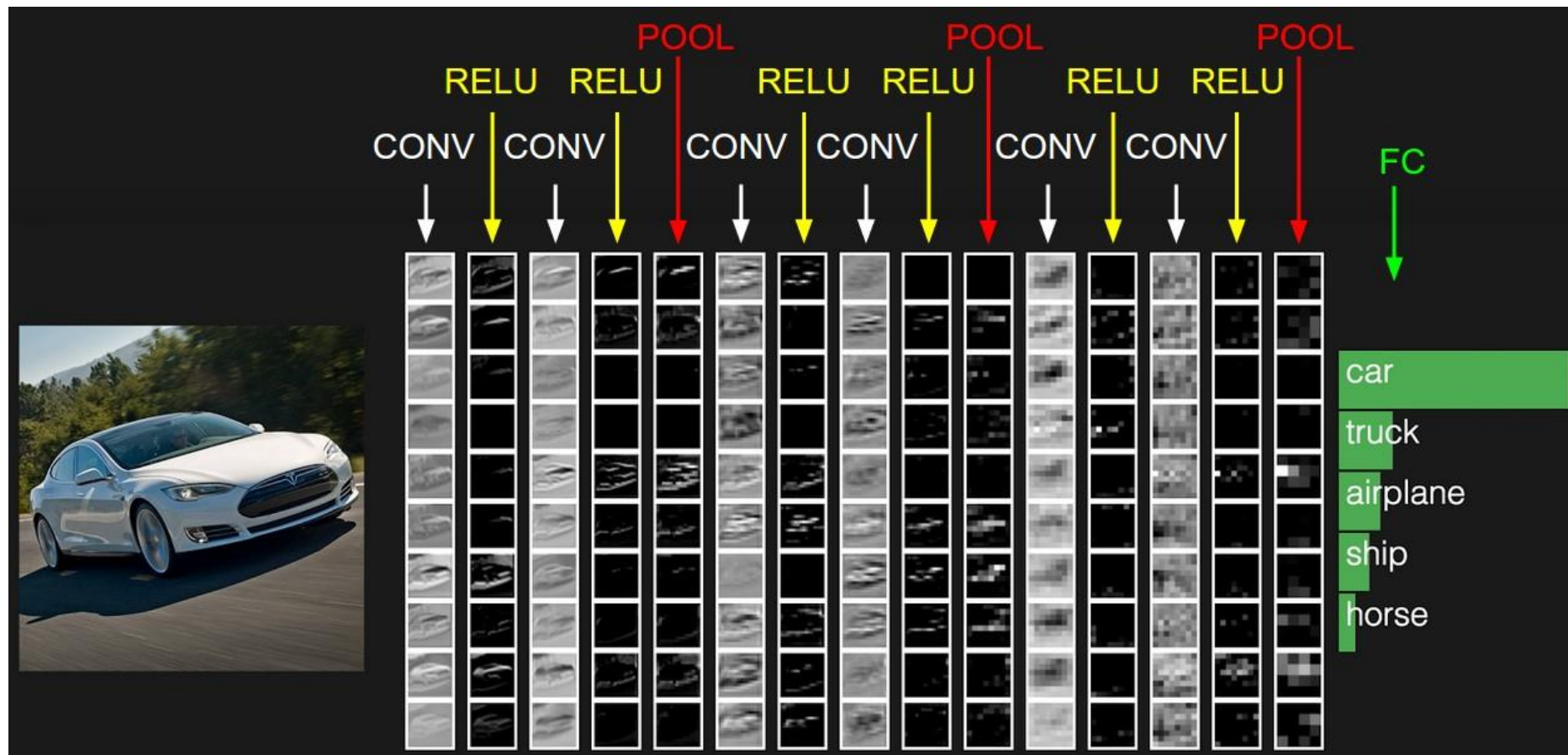
( Subsampling or downsampling )

- Common way is a pooling layer with filters of size 2x2 applied with a stride of 2



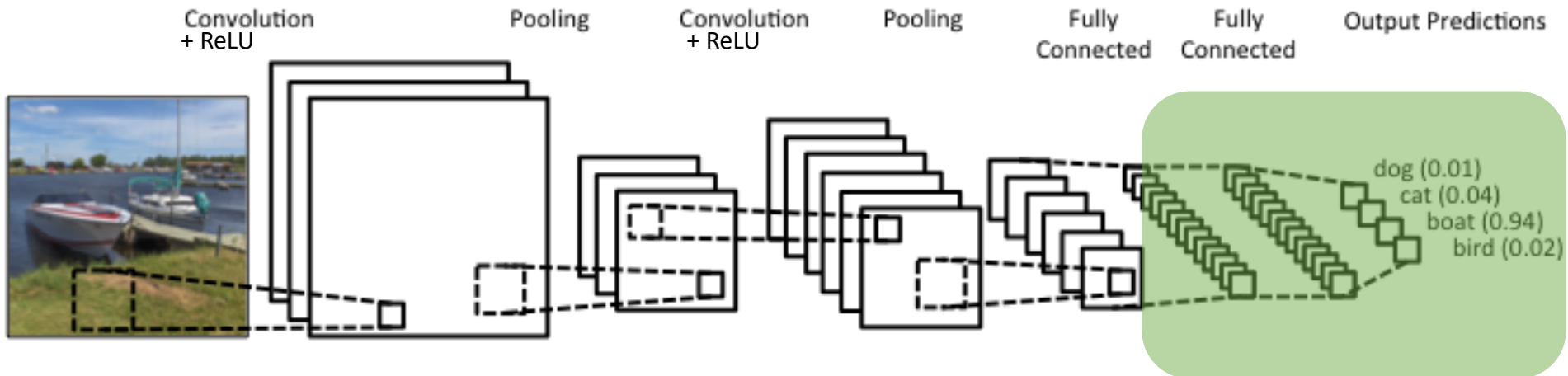
# Conv. + ReLU + POOL

- **Convolution layers** and **Pooling layers** can be repeated any number of times in a single ConvNet.



## 4. Classification

- Multi Layer Perceptron
- The number of filters, filter sizes, architecture of the network etc. are fixed and do not change during training process.
- Only the values of the filter matrix and connection weights get updated.





# CovNet Architectures

- **LeNet (1990s)**
- **AlexNet (2012)**
- **ZF NET (2013)**
- **GoogLeNet (2014)**
- **VGGNet (2014)**
- **ResNets (2015)**
- **DenseNet (2016)**



# Dlib C++ Library

Google Custom Search



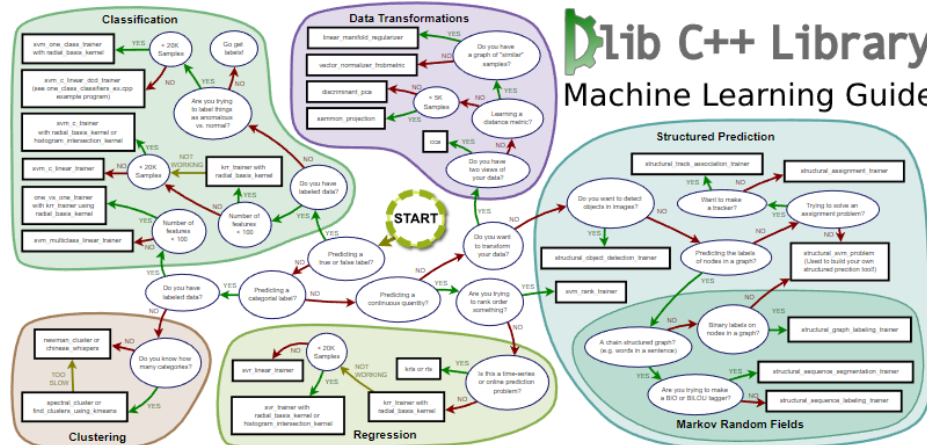
## The Library

Algorithms  
API Wrappers  
Bayesian Nets  
Compression  
Containers  
Graph Tools  
Image Processing  
Linear Algebra  
Machine Learning  
Metaprogramming  
Miscellaneous  
Networking  
Optimization  
Parsing

## Help/Info

Dlib Blog  
+ Examples: C++  
+ Examples: Python  
FAQ  
Home  
How to compile  
How to contribute  
Index  
Introduction  
License  
Python API

## Machine Learning



## Primary Algorithms

### Binary Classification

rvm\_trainer  
svm\_c\_ekm\_trainer  
svm\_c\_linear\_dcd\_trainer  
svm\_c\_linear\_trainer  
svm\_c\_trainer  
svm\_nu\_trainer  
svm\_pegasos  
train\_probabilistic\_decision\_function

### Multiclass Classification

one\_vs\_all\_trainer  
one\_vs\_one\_trainer  
svm\_multiclass\_linear\_trainer

### Regression

krls  
krr\_trainer  
mlp  
rbf\_network\_trainer  
rls  
rr\_trainer  
rvm\_regression\_trainer  
svr\_linear\_trainer  
svr\_trainer

Dlib contains a wide range of machine learning algorithms. All designed to be highly modular, quick to execute, and simple to use via a clean and modern C++ API. It is used in a wide range of applications including robotics, embedded devices, mobile phones, and large high performance computing environments. If you use dlib in your research please cite:



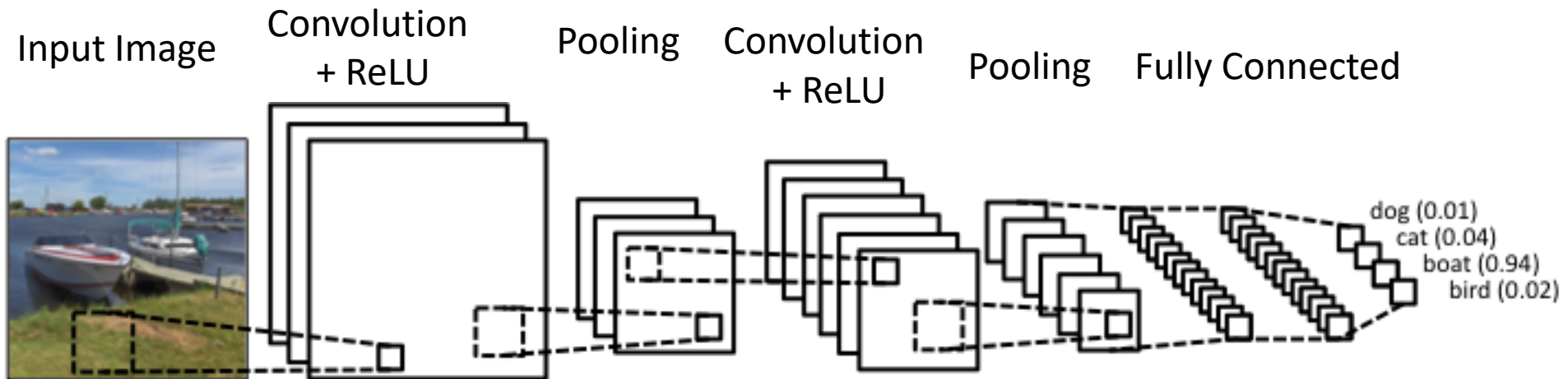


# LeNet type CNN (Dlib/C++)

```
using net_type = loss_multiclass_log<  
    fc<10,  
    relu<fc<84,  
    relu<fc<120,  
    max_pool<2,2,2,2,relu<con<16,5,5,1,1,  
    max_pool<2,2,2,2,relu<con<6,5,5,1,1,  
    input<matrix<unsigned char>>  
>>>>>>>>>>>>>>;
```

# LeNet type CNN (Dlib/C++)

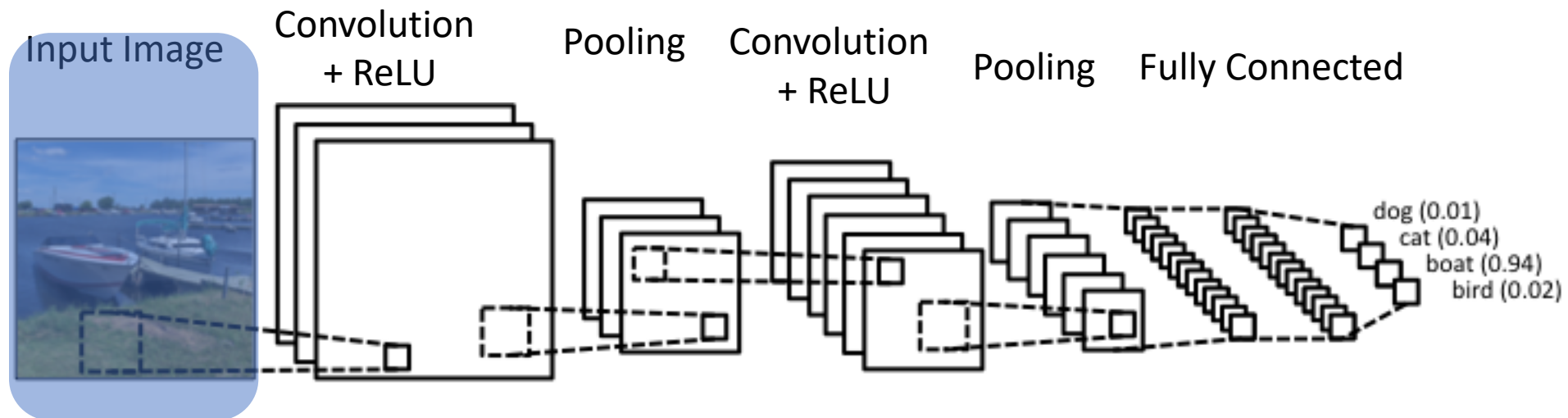
```
using net_type = loss_multiclass_log<  
    fc<10,  
    relu<fc<84,  
    relu<fc<120,  
    max_pool<2,2,2,2,relu<con<16,5,5,1,1,  
    max_pool<2,2,2,2,relu<con<6,5,5,1,1,  
    input<matrix<unsigned_char>>  
>>>>>>>>>>;
```



# LeNet type CNN (Dlib/C++)

```
using net_type = loss_multiclass_log<  
    fc<10,  
    relu<fc<84,  
    relu<fc<120,  
    max_pool<2,2,2,2,relu<con<16,5,5,1,1,  
    max_pool<2,2,2,2,relu<con<6,5,5,1,1,  
    input<matrix<unsigned_char>>  
>>>>>>>>>>>>;
```

Input Image

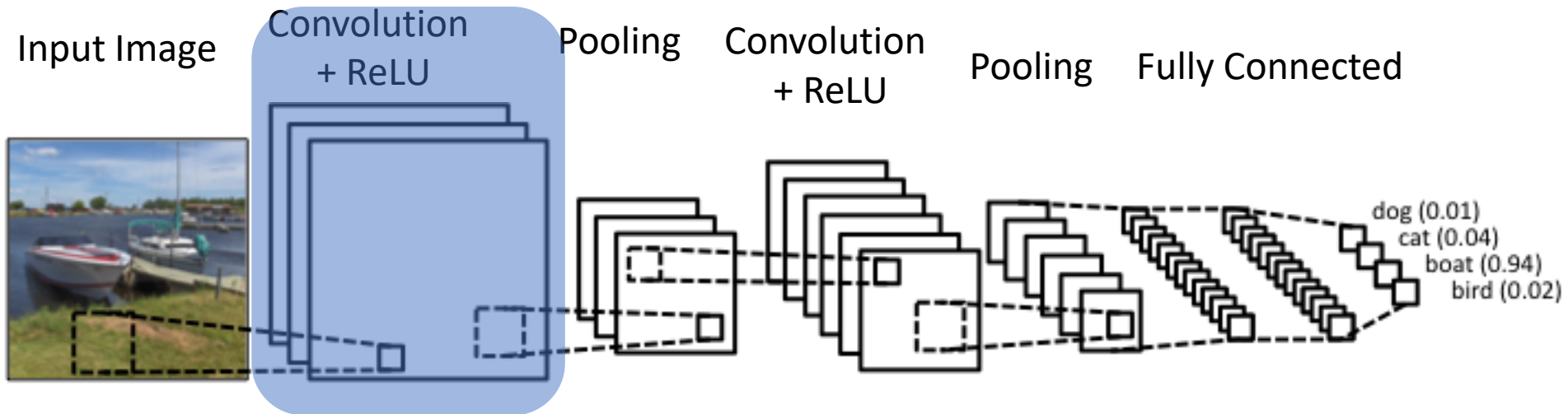


[http://dlib.net/dnn\\_introduction\\_ex.cpp.html](http://dlib.net/dnn_introduction_ex.cpp.html)

# LeNet type CNN (Dlib/C++)

```
using net_type = loss_multiclass_log<
    fc<10,
    relu<fc<84,
    relu<fc<120,
    max_pool<2,2,2,2,relu<con<16,5,5,1,1,
    max_pool<2,2,2,2,relu<con<6,5,5,1,1,
    input<matrix<unsigned char>>
>>>>>>>>>>;
```

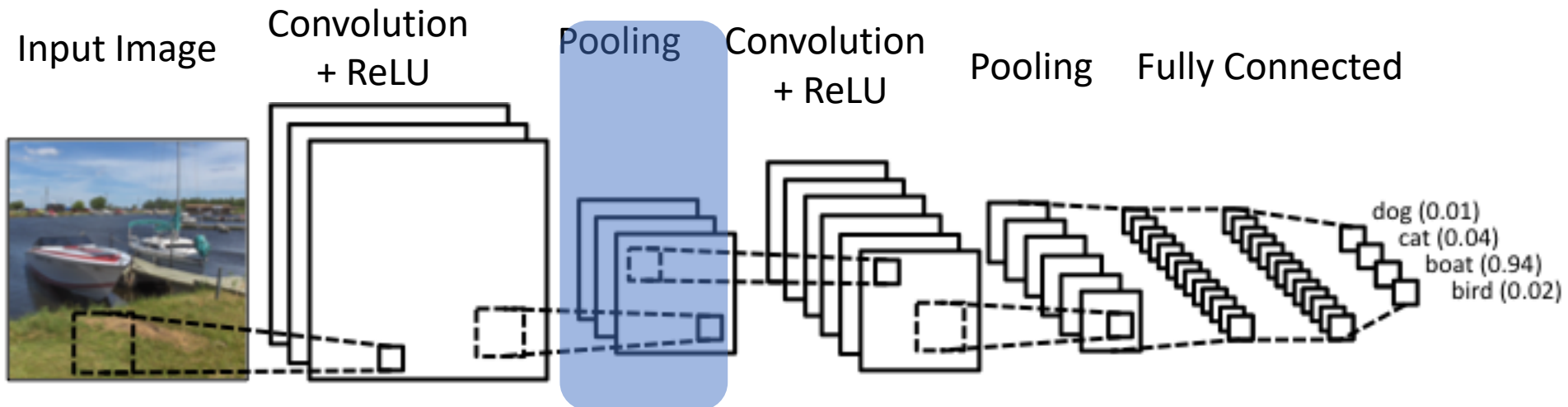
6 conv. filters  
5x5 filter size  
1x1 stride  
+ReLU



# LeNet type CNN (Dlib/C++)

```
using net_type = loss_multiclass_log<  
    fc<10,  
    relu<fc<84,  
    relu<fc<120,  
    max_pool<2,2,2,2,relu<con<16,5,5,1,1,  
    max_pool<2,2,2,2,relu<con<6,5,5,1,1,  
    input<matrix<unsigned char>>  
>>>>>>>>>>>>;
```

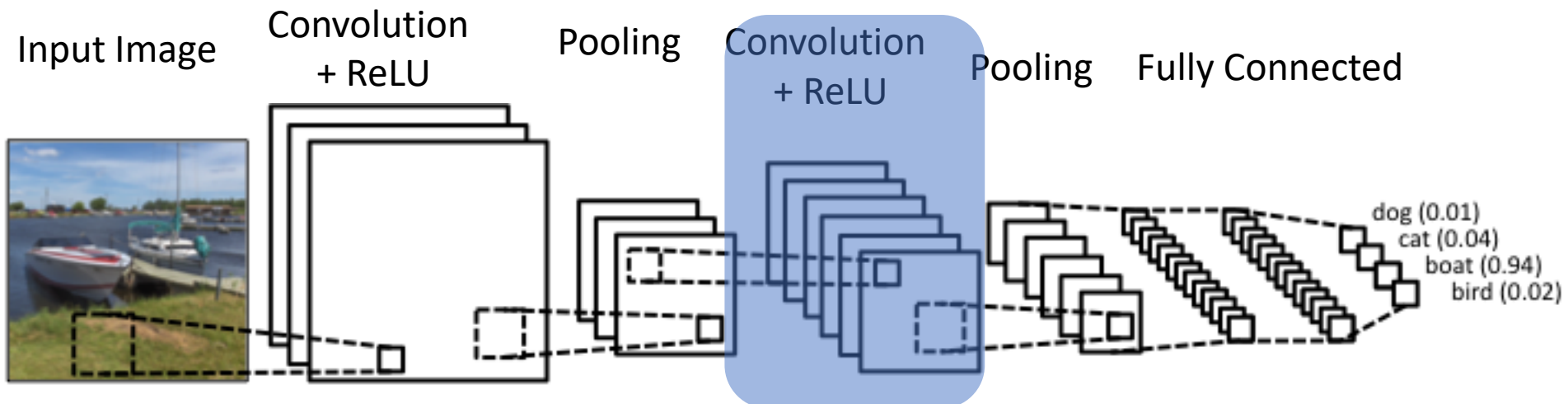
MAX POOLING  
2x2 window  
2x2 stride



## LeNet type CNN (Dlib/C++)

```
using net_type = loss_multiclass_log<
    fc<10,
    relu<fc<84,
    relu<fc<120,
    max_pool<2,2,2,2,relu<con<16,5,5,1,1,
    max_pool<2,2,2,2,relu<con<6,5,5,1,1,
    input<matrix<unsigned char>>
    >>>>>>>>>>>>;
```

16 conv. filters  
5x5 filter size  
1x1 stride  
+ReLU



[http://dlib.net/dnn\\_introduction\\_ex.cpp.html](http://dlib.net/dnn_introduction_ex.cpp.html)



```
using net_type = loss_multiclass_log<
    fc<10,
    relu<fc<84,
    relu<fc<120,
    max_pool<2,2,2,2,relu<con<16,5,5,1,1,
    max_pool<2,2,2,2,relu<con<6,5,5,1,1,
    input<matrix<unsigned char>>
    >>>>>>>>>>>>;
```

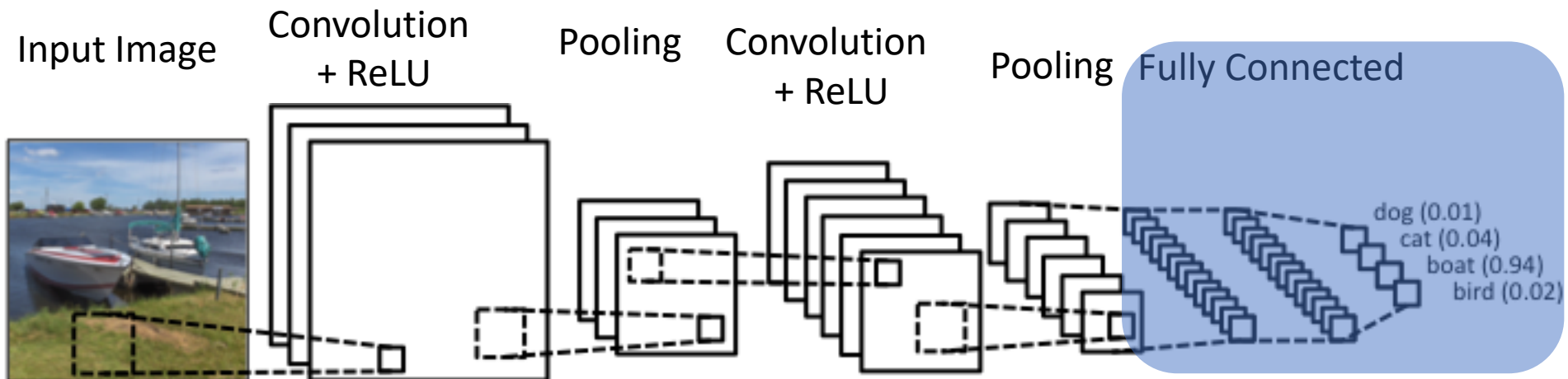
[http://dlib.net/dnn\\_introduction\\_ex.cpp.html](http://dlib.net/dnn_introduction_ex.cpp.html)

# LeNet type CNN (Dlib/C++)

```
using net_type = loss_multiclass_log<
```

Fully connected layer  
120 neurons  
84 neurons  
10 outputs/classes  
multiclass  
classification

```
fc<10,  
relu<fc<84,  
relu<fc<120,  
max_pool<2,2,2,2,relu<con<16,5,5,1,1,  
max_pool<2,2,2,2,relu<con<6,5,5,1,1,  
input<matrix<unsigned char>>  
>>>>>>>>>>;
```



[http://dlib.net/dnn\\_introduction\\_ex.cpp.html](http://dlib.net/dnn_introduction_ex.cpp.html)



# LeNet type CNN (Dlib/C++)

```
1 // network instance
2 net_type net;
3
4 // mini-batch stochastic gradient descent
5 //dnn_trainer<net_type> trainer(net, sgd(), {0,1}); //{0,1} - will use two GPU
6 dnn_trainer<net_type> trainer(net);
7 trainer.set_learning_rate(0.01);
8 trainer.set_min_learning_rate(0.0001);
9 trainer.set_mini_batch_size(160);
10 trainer.set_iterations_without_progress_threshold(500);
11 trainer.set_max_num_epochs(100);
12 trainer.be_verbose();
13 //train
14 trainer.train(train_images, train_labels);
15 // save
16 serialize("LeNet.dat") << net; |
```

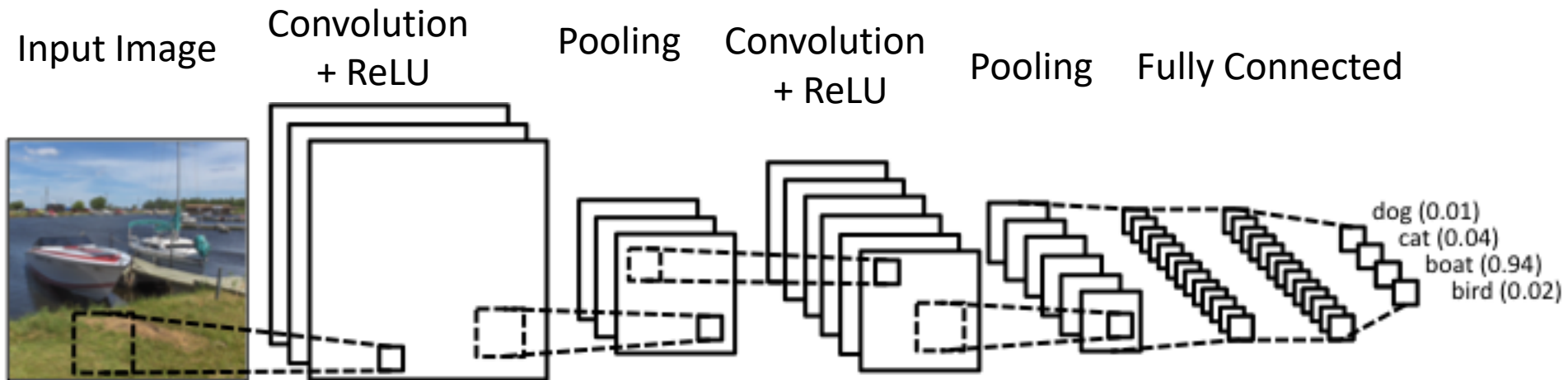


# LeNet type CNN (Dlib/C++)

```
1 //Load image using OpenCV
2 Mat frame;
3 frame = imread( "my_img.png", 1 );
4 cvtColor( frame, frame, COLOR_BGR2GRAY );
5 medianBlur(frame, frame, 5);
6
7 //OpenCV Mat to Dlib
8 cv_image<unsigned char> cimg(frame);
9 matrix<unsigned char> dlibFrame = dlib::mat(cimg);
10
11 //prediction using CNN
12 unsigned long predict_label = net(frame);
```

# LeNet type CNN (Pytorch)

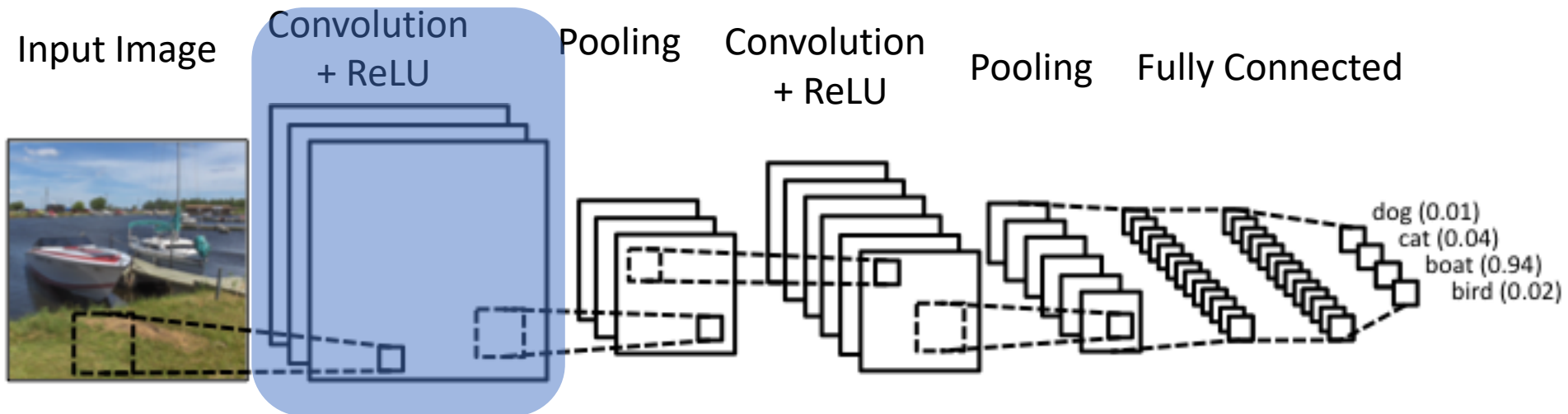
```
net = nn.Sequential(  
    nn.Conv2d(3, 6, kernel_size=5, padding=0, stride=1),  
    nn.ReLU(),  
    nn.AvgPool2d(kernel_size=2, stride=2),  
    nn.Conv2d(6, 16, kernel_size=5, padding=0, stride=1),  
    nn.ReLU(),  
    nn.AvgPool2d(kernel_size=2, stride=2),  
    nn.Flatten(),  
    nn.Linear(16 * 5 * 5, 120),  
    nn.ReLU(),  
    nn.Linear(120, 84),  
    nn.ReLU(),  
    nn.Linear(84, 2))
```



# LeNet type CNN (Pytorch)

```
net = nn.Sequential(  
    nn.Conv2d(3, 6, kernel_size=5, padding=0, stride=1),  
    nn.ReLU(),  
    nn.AvgPool2d(kernel_size=2, stride=2),  
    nn.Conv2d(6, 16, kernel_size=5, padding=0, stride=1),  
    nn.ReLU(),  
    nn.AvgPool2d(kernel_size=2, stride=2),  
    nn.Flatten(),  
    nn.Linear(16 * 5 * 5, 120),  
    nn.ReLU(),  
    nn.Linear(120, 84),  
    nn.ReLU(),  
    nn.Linear(84, 2))
```

6 conv. filters  
5x5 filter size  
1x1 stride  
0 padding  
+ ReLU

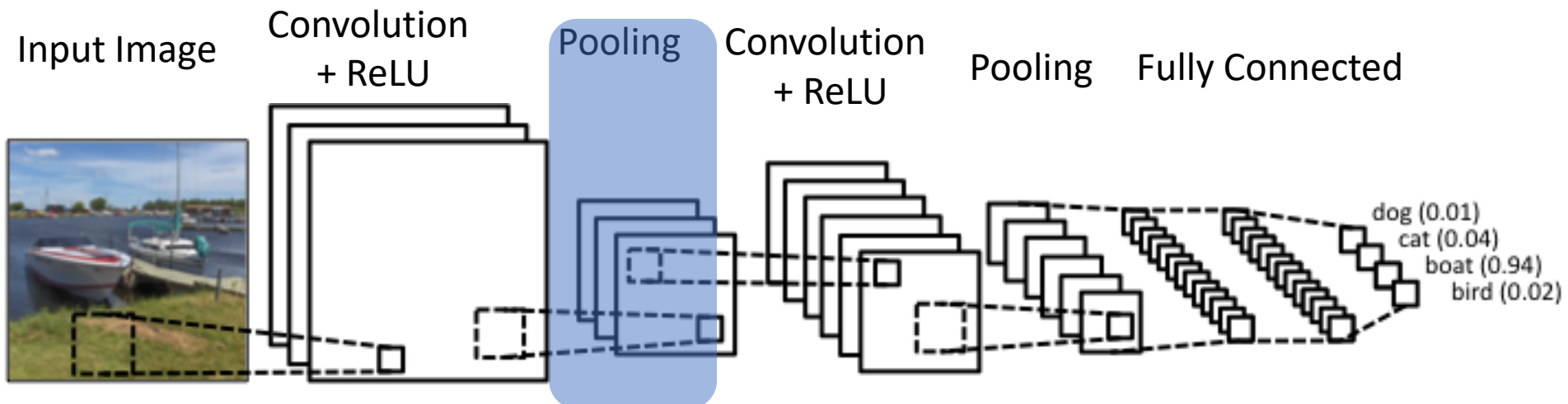




# LeNet type CNN (Pytorch)

```
net = nn.Sequential(  
    nn.Conv2d(3, 6, kernel_size=5, padding=0, stride=1),  
    nn.ReLU(),  
    nn.AvgPool2d(kernel_size=2, stride=2),  
    nn.Conv2d(6, 16, kernel_size=5, padding=0, stride=1),  
    nn.ReLU(),  
    nn.AvgPool2d(kernel_size=2, stride=2),  
    nn.Flatten(),  
    nn.Linear(16 * 5 * 5, 120),  
    nn.ReLU(),  
    nn.Linear(120, 84),  
    nn.ReLU(),  
    nn.Linear(84, 2))
```

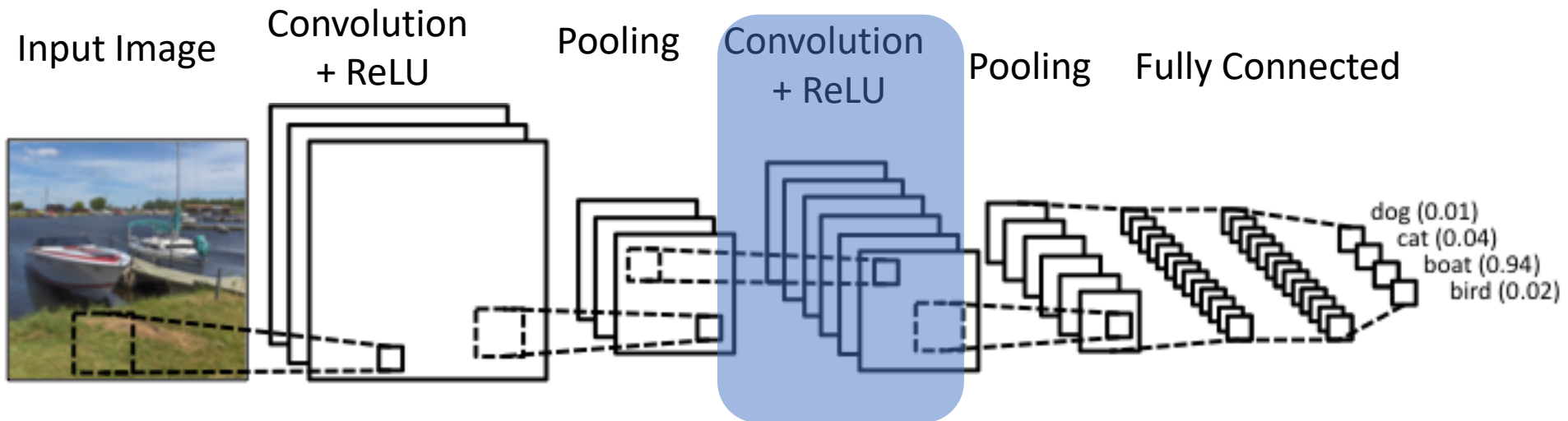
AVG POOLING  
2x2 window  
2x2 stride



# LeNet type CNN (Pytorch)

```
net = nn.Sequential(  
    nn.Conv2d(3, 6, kernel_size=5, padding=0, stride=1),  
    nn.ReLU(),  
    nn.AvgPool2d(kernel_size=2, stride=2),  
    nn.Conv2d(6, 16, kernel_size=5, padding=0, stride=1),  
    nn.ReLU(),  
    nn.AvgPool2d(kernel_size=2, stride=2),  
    nn.Flatten(),  
    nn.Linear(16 * 5 * 5, 120),  
    nn.ReLU(),  
    nn.Linear(120, 84),  
    nn.ReLU(),  
    nn.Linear(84, 2))
```

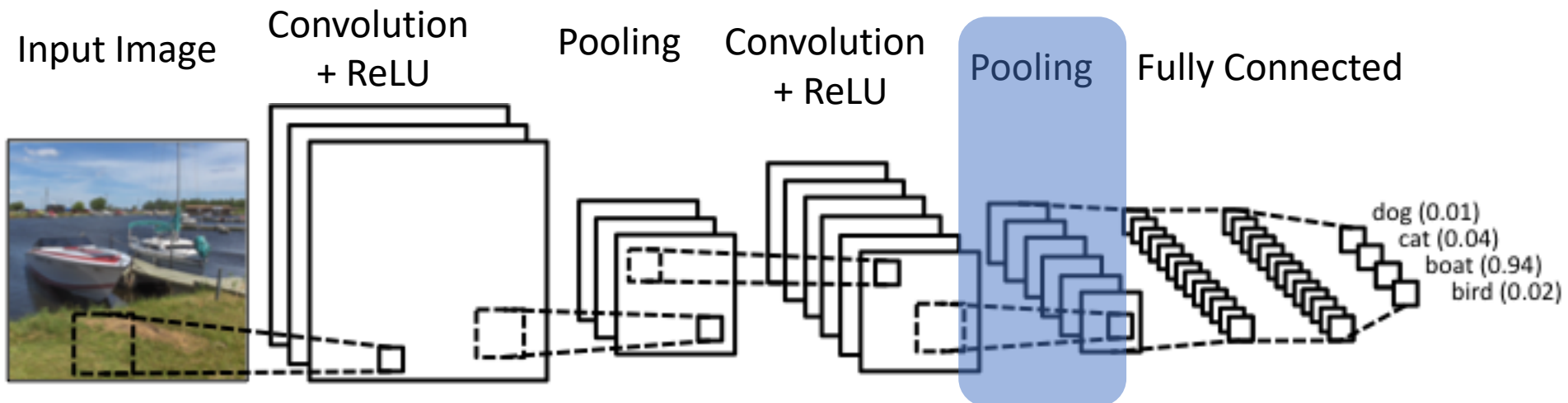
6 conv. filters  
5x5 filter size  
1x1 stride  
0 padding  
+ ReLU



# LeNet type CNN (Pytorch)

```
net = nn.Sequential(  
    nn.Conv2d(3, 6, kernel_size=5, padding=0, stride=1),  
    nn.ReLU(),  
    nn.AvgPool2d(kernel_size=2, stride=2),  
    nn.Conv2d(6, 16, kernel_size=5, padding=0, stride=1),  
    nn.ReLU(),  
    nn.AvgPool2d(kernel_size=2, stride=2),  
    nn.Flatten(),  
    nn.Linear(16 * 5 * 5, 120),  
    nn.ReLU(),  
    nn.Linear(120, 84),  
    nn.ReLU(),  
    nn.Linear(84, 2))
```

AVG POOLING  
2x2 window  
2x2 stride



# LeNet type CNN (Pytorch)

```
net = nn.Sequential(  
    nn.Conv2d(3, 6, kernel_size=5, padding=0, stride=1),  
    nn.ReLU(),  
    nn.AvgPool2d(kernel_size=2, stride=2),  
    nn.Conv2d(6, 16, kernel_size=5, padding=0, stride=1),  
    nn.ReLU(),  
    nn.AvgPool2d(kernel_size=2, stride=2),  
    nn.Flatten(),  
    nn.Linear(16 * 5 * 5, 120),  
    nn.ReLU(),  
    nn.Linear(120, 84),  
    nn.ReLU(),  
    nn.Linear(84, 2))
```

Fully connected layer  
120 neurons  
84 neurons  
2 outputs/classes

