Object Recognition/Detection

Radovan Fusek

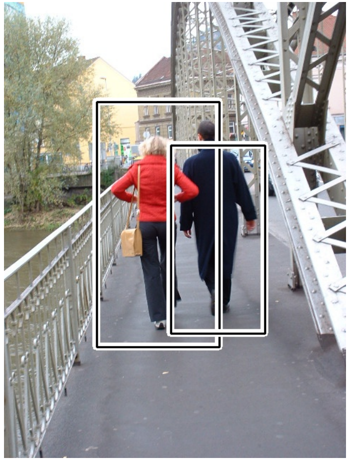
2nd International summer school on "Deep Learning and Visual Data Analysis"

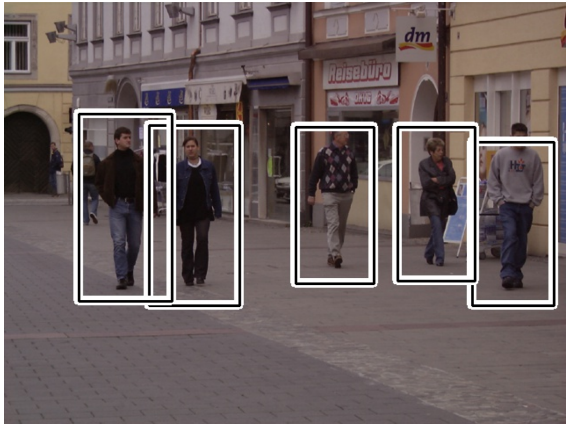
2018

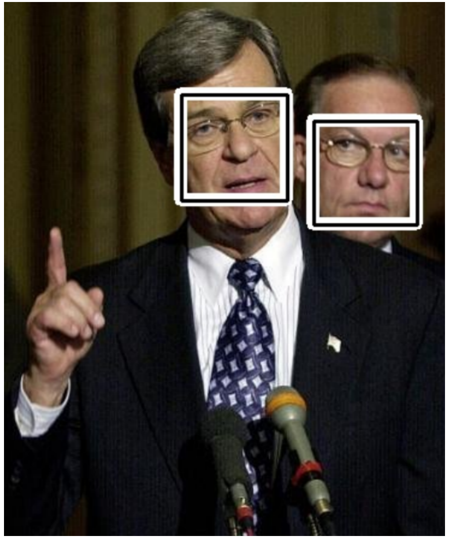
Our work presented here was partially supported by the EU H2020 686782 PACMAN project, (solved with Honeywell), http://mrl.cs.vsb.cz/h2020

What is Object Detection/Recognition?

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







Object Detection/Recognition

- Haar
- HOG
- LBP

Traditional Approaches

- SIFT, SURF

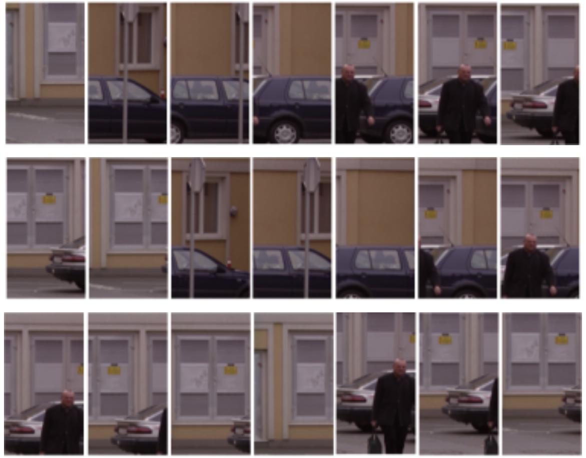
KeyPoints

- CNNs

Deep Learning Approach

- Practical examples using OpenCV + Dlib (https://opencv.org/, http://dlib.net/)

Sliding Window - Main Idea



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

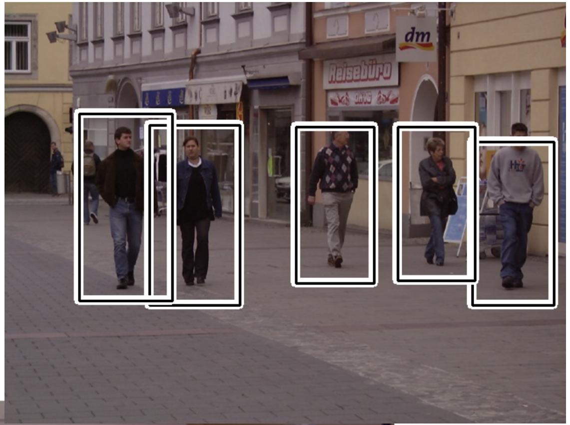
Related Works

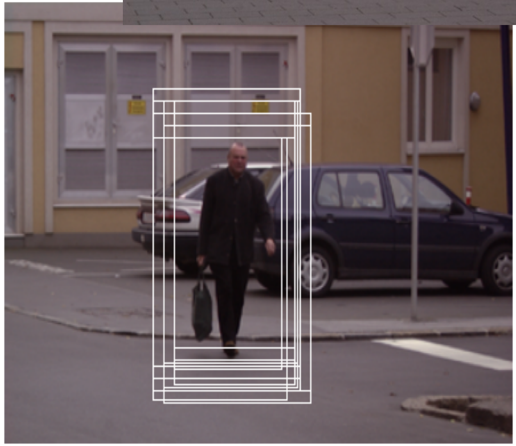
Feature Vector

(gradient, HOG, LBP, …)

Trainable Classifier

(SVM, ANNs, …)

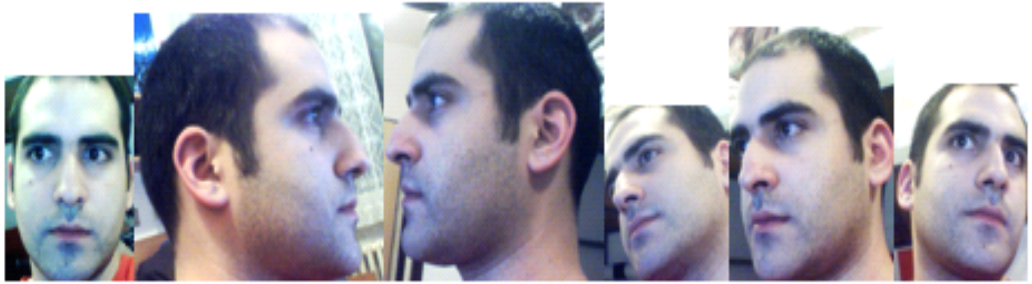




Constantine Papageorgiou and Tomaso Poggio: A Trainable System for Object Detection.
Int. J. Comput. Vision 38, pp. 15-33. (2000)

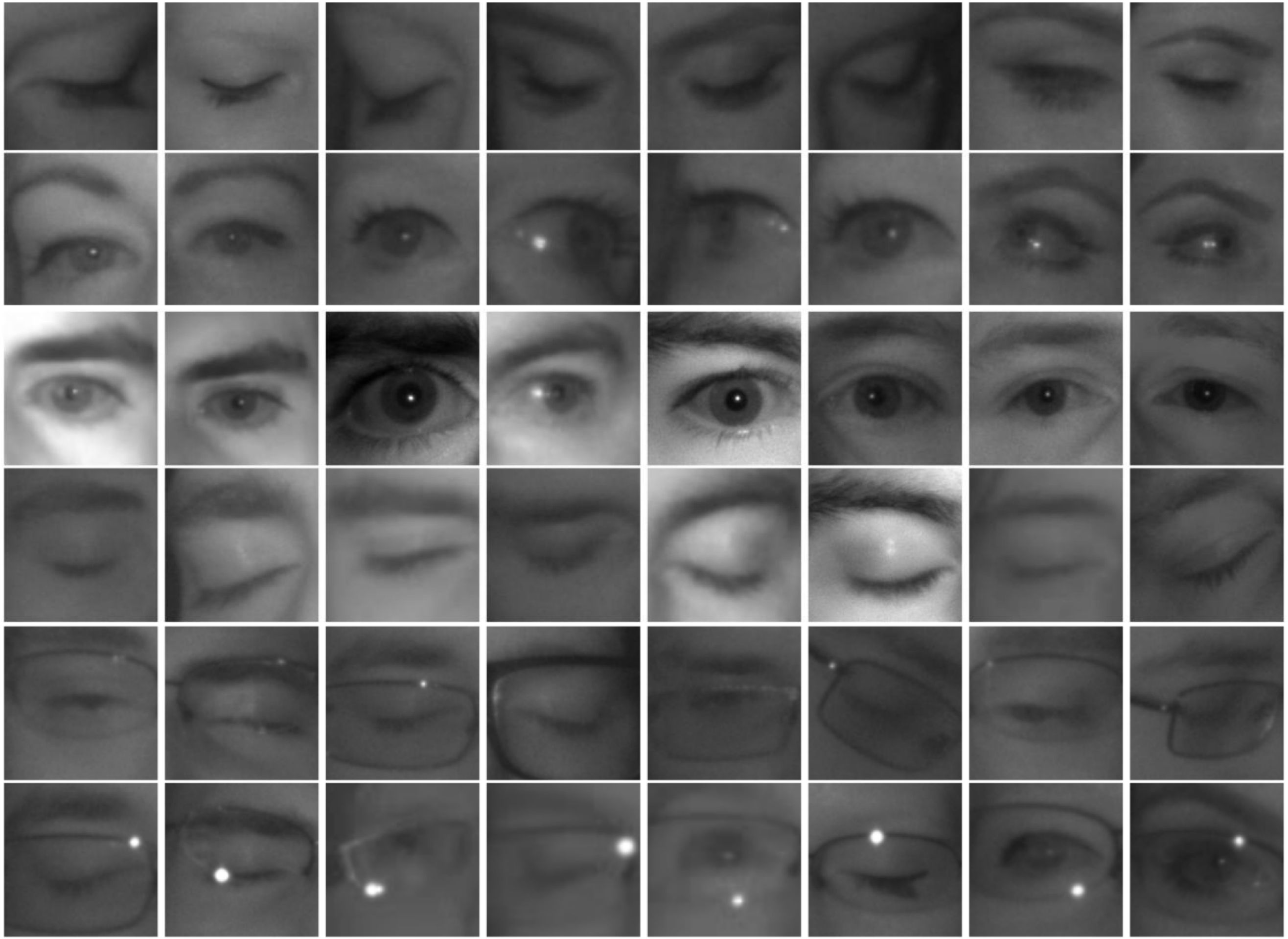
Generating Training Set

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



Generating Training Set

http://mrl.cs.vsb.cz/eyedataset



Object Detection/Recognition

- Haar
- HOG
- LBP

Traditional Approaches

- SIFT, SURF

KeyPoints

- CNNs

Deep Learning Approach

- Practical examples using OpenCV + Dlib (https://opencv.org/, http://dlib.net/)

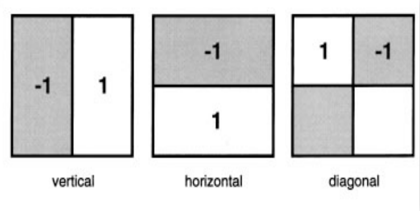
Related Works

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

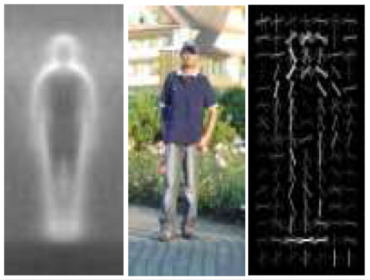
PAUL VIOLA
Microsoft Research, One Microsoft Way, Redmond, WA 98052, USA
viola@microsoft.com

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

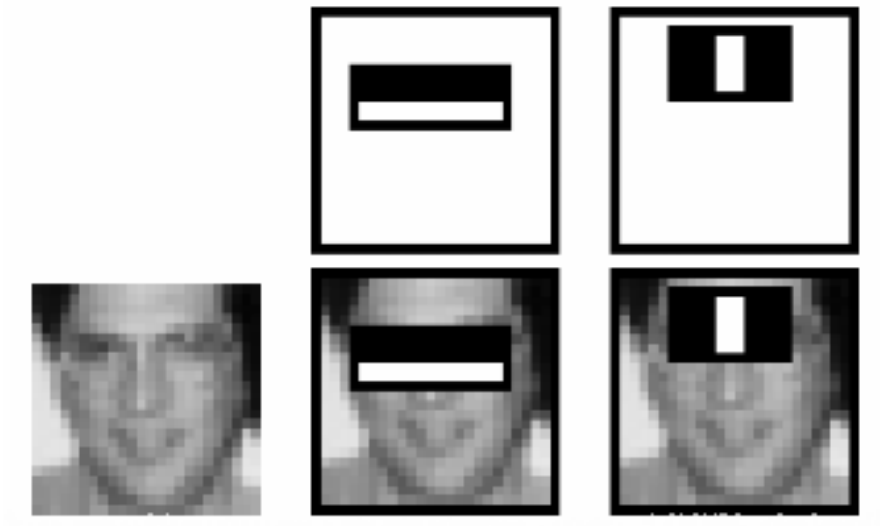
Navneet Dalal and Bill Triggs
INRIA Rhône-Alps, 655 avenue de l'Europe, Montbonnot 38334, France
{Navneet.Dalal,Bill.Triggs}@inrialpes.fr, http://lear.inrialpes.fr



2005

Features

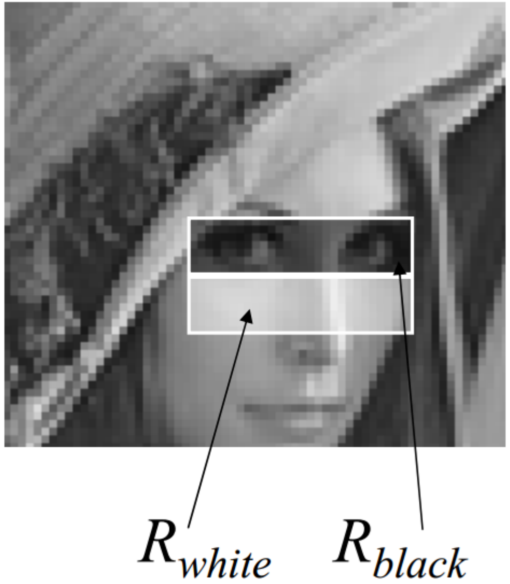
- faces have similar properties
  - eye regions are darker than the upper-cheeks
  - the nose bridge region is brighter than the eyes



https://docs.opencv.org/3.4.1/d7/d8b/tutorial_py_face_detection.html

Features

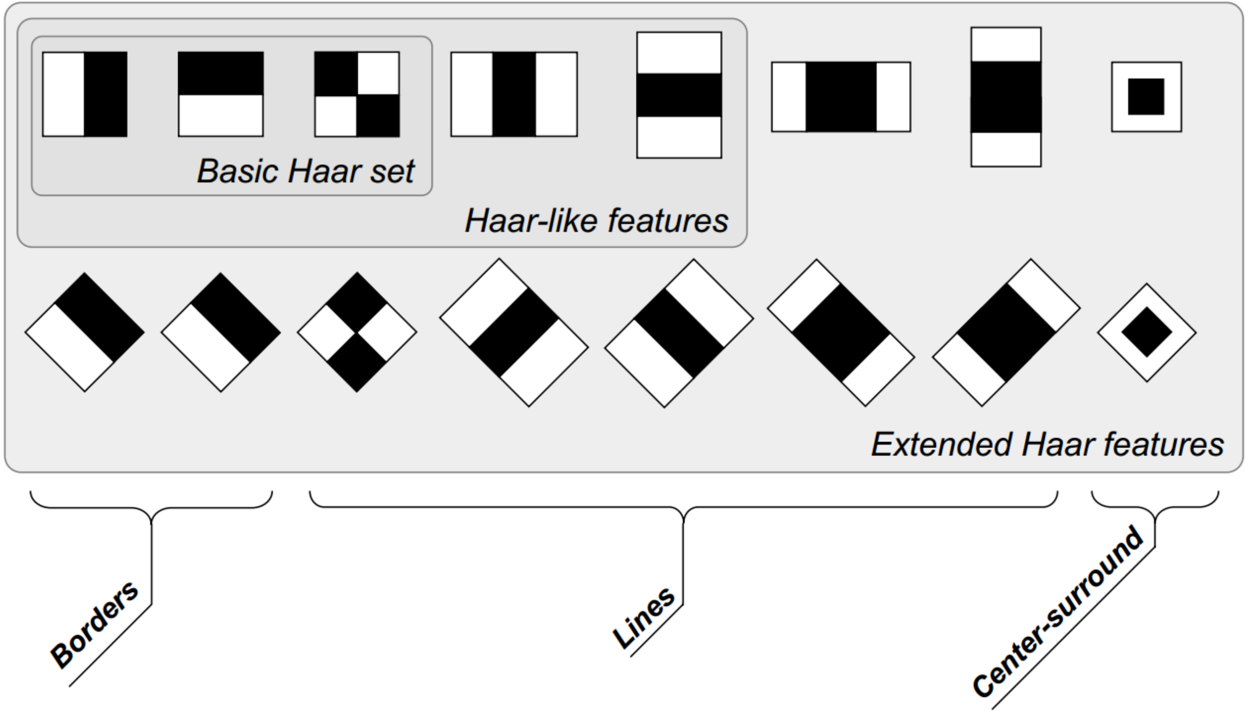
- Rectangular features



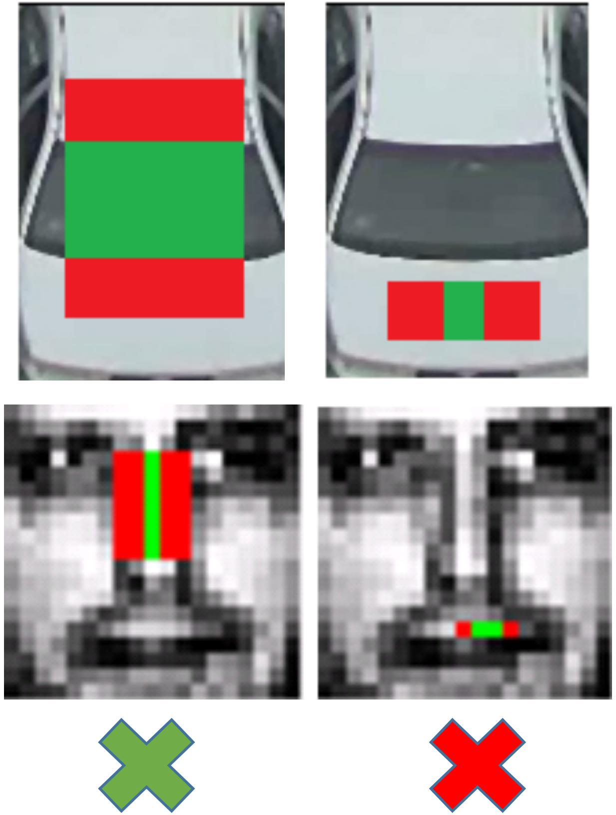
$$F_{Haar} = E(R_{white}) - E(R_{black})$$

Features

Different sets



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

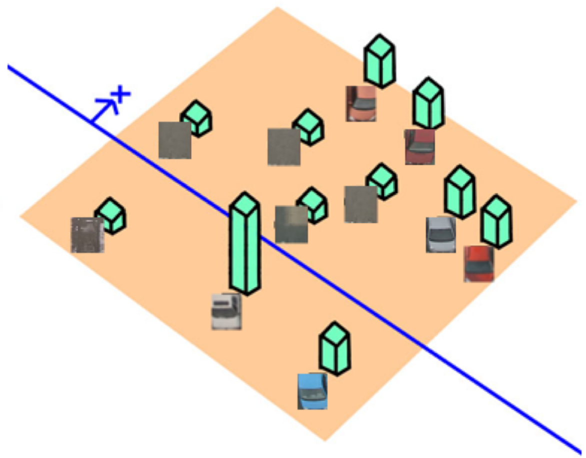
Feature Selection

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

p Select the classifier with the lowest weighted error (i.e. a “weak” classifier)

p Increase the weights on the training examples that were misclassified.

p (Repeat)

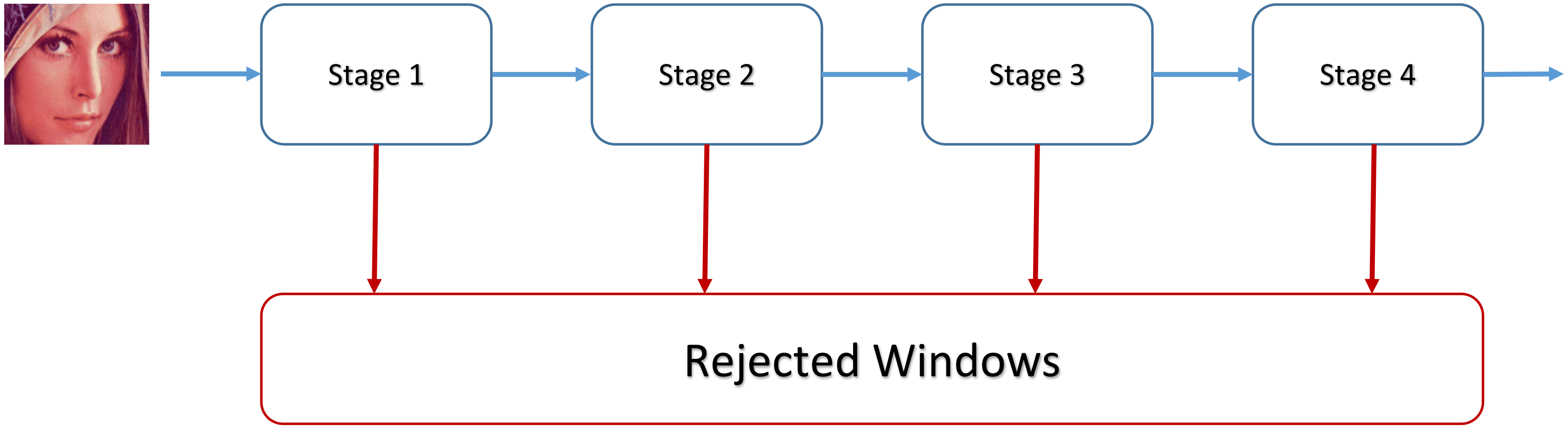


p At the end, carefully make a linear combination of the weak classifiers obtained at all iterations.

Slide taken from a presentation by Qing Chen, Discover Lab, University of Ottawa

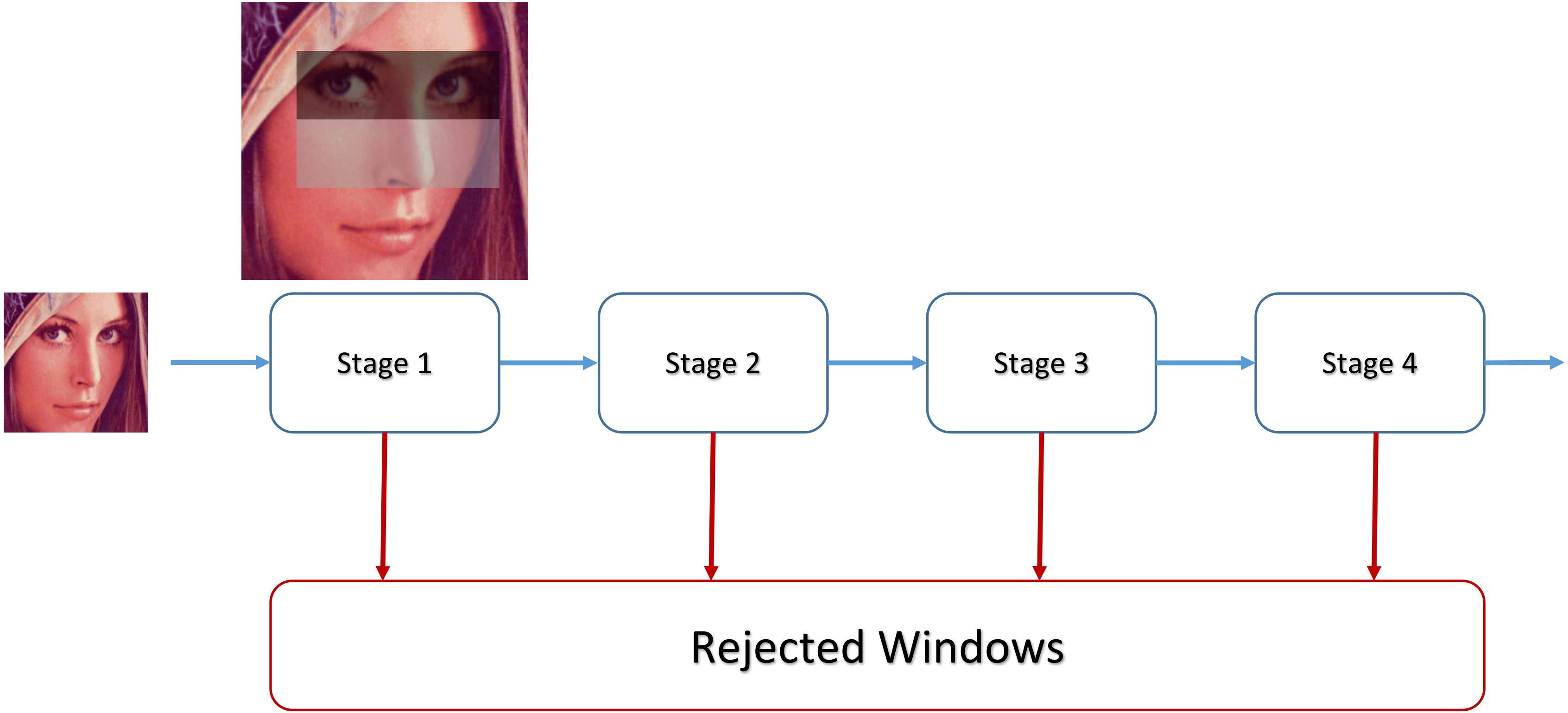
Cascade of Classifier

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



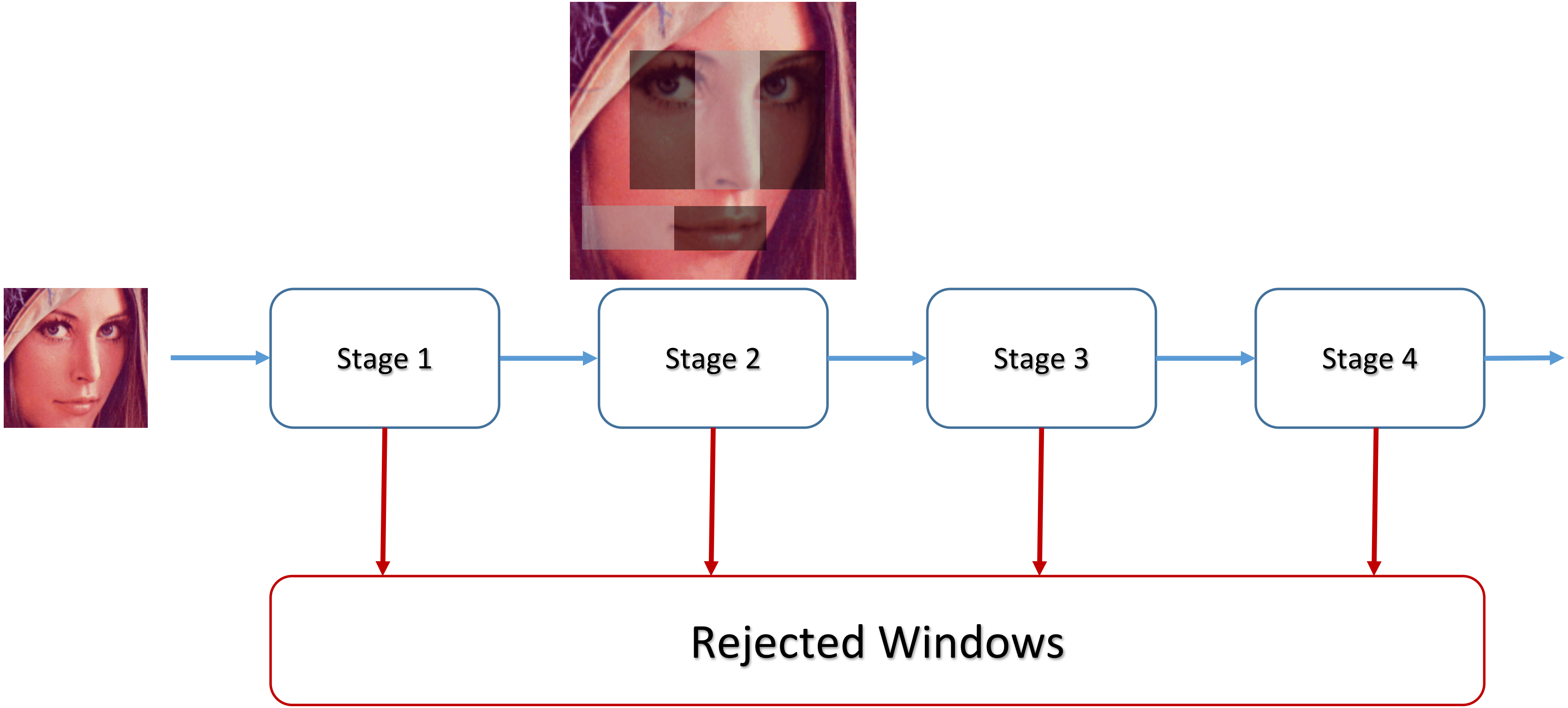
Cascade of Classifier

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



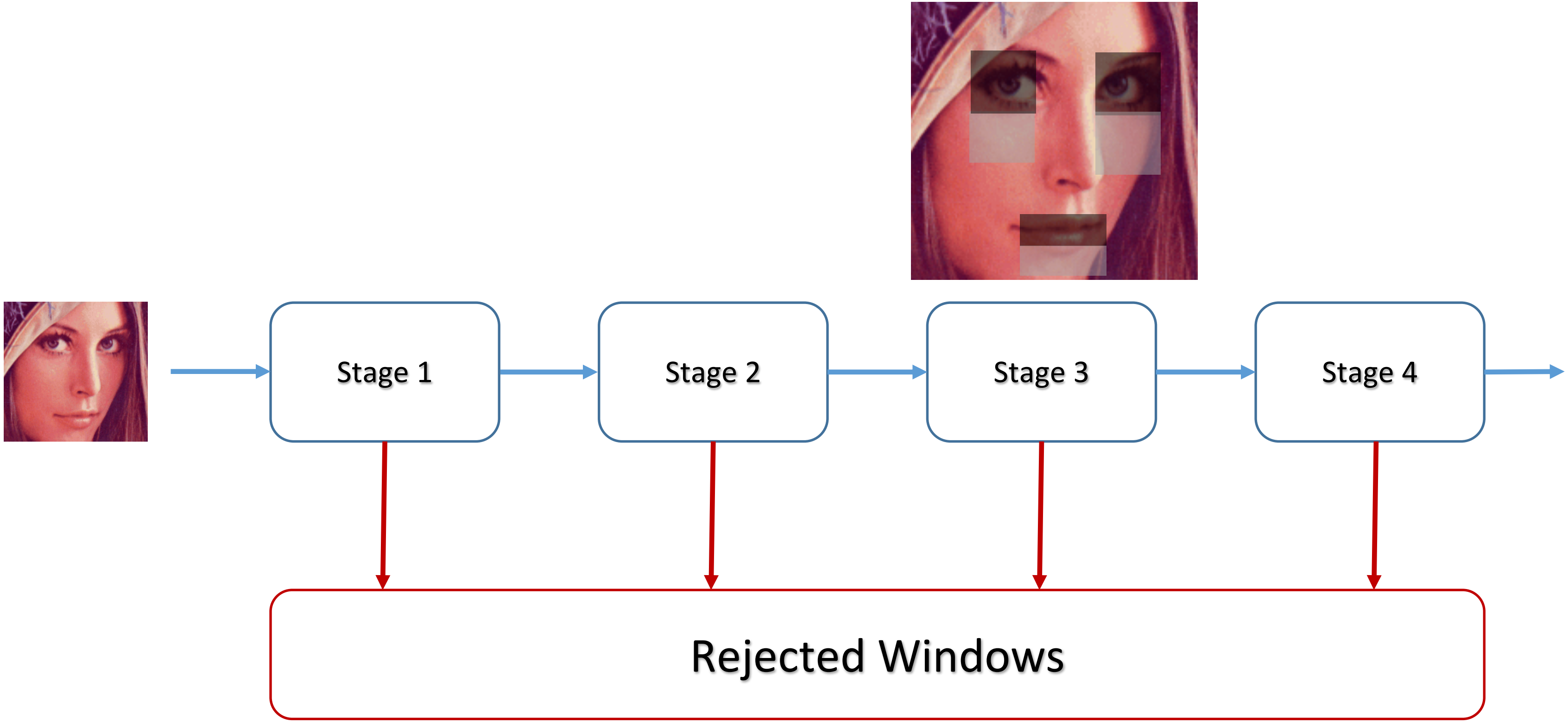
Cascade of Classifier

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



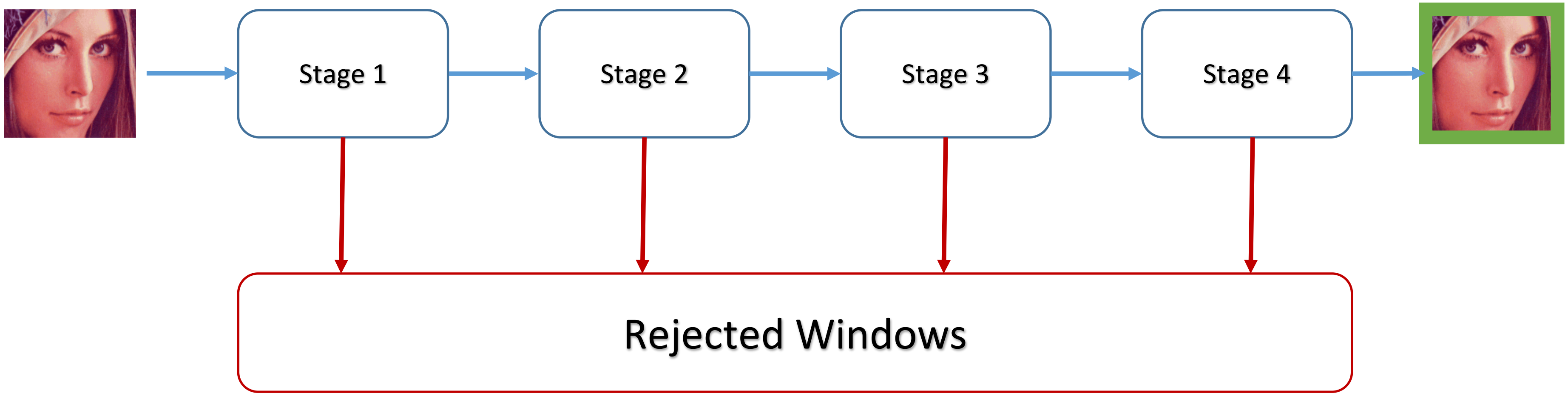
Cascade of Classifier

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



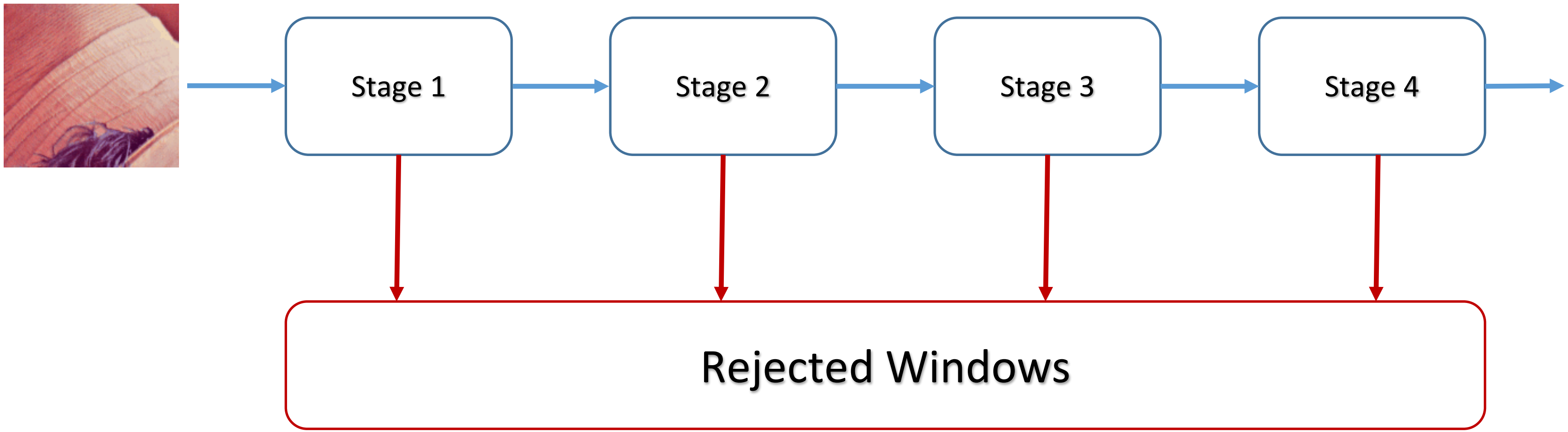
Cascade of Classifier

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



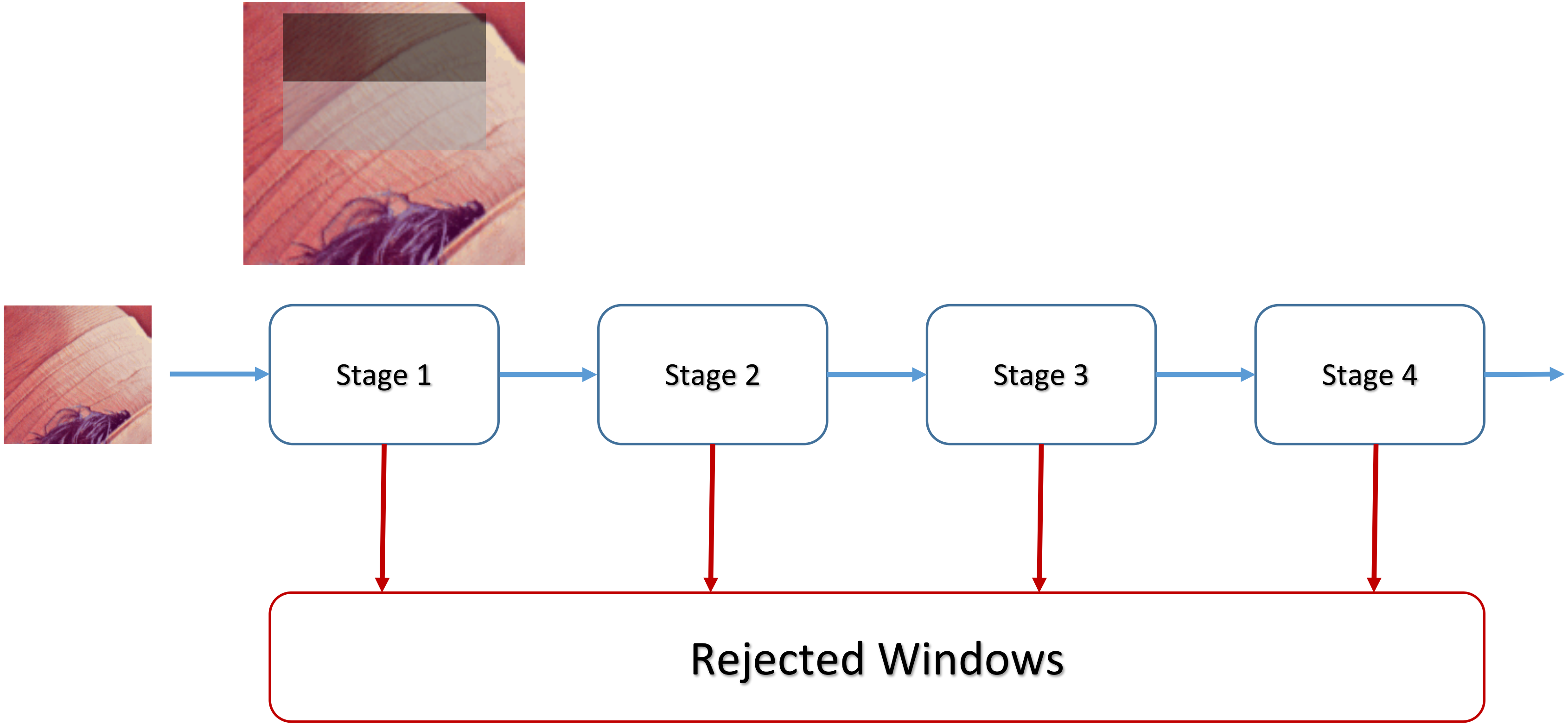
Cascade of Classifier

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



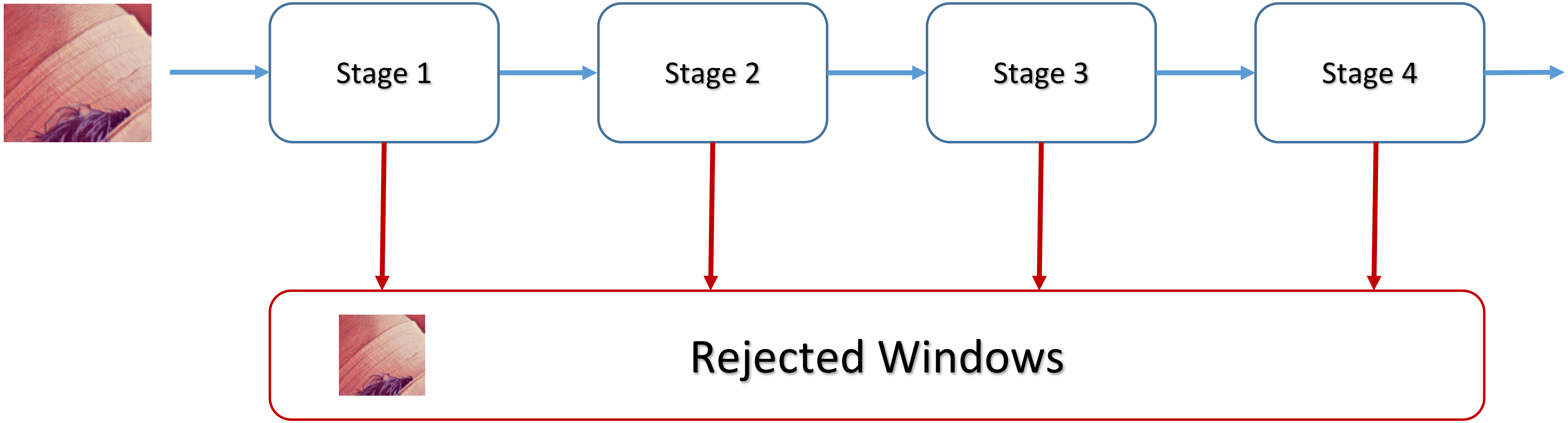
Cascade of Classifier

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

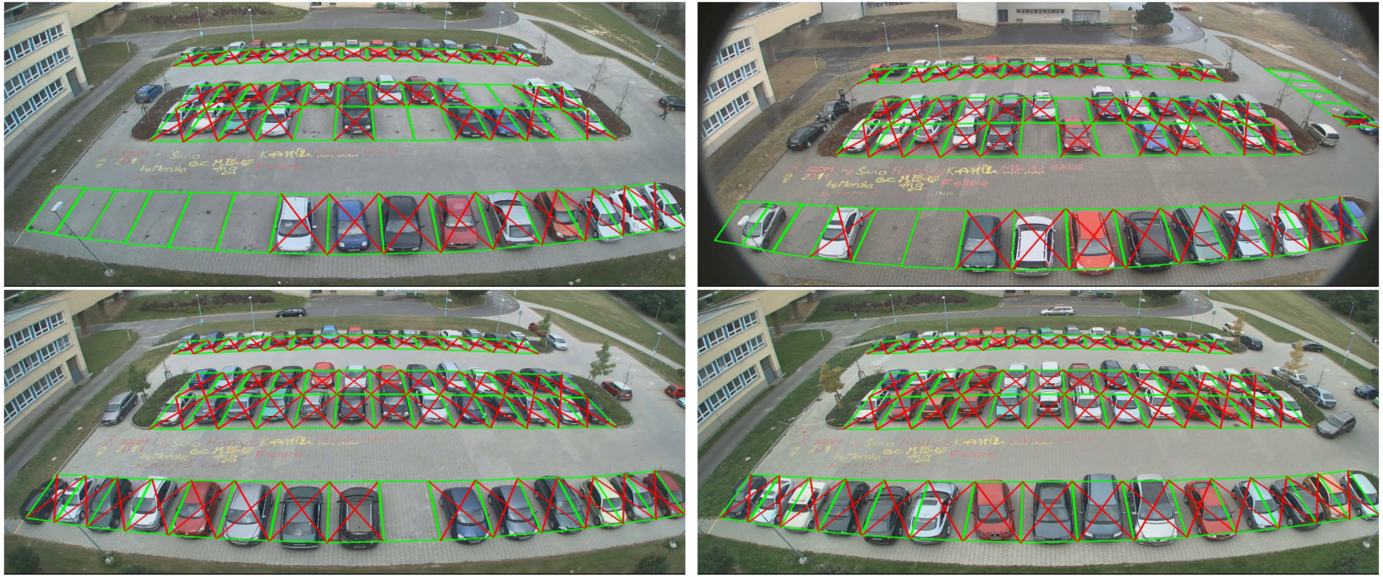
https://vimeo.com/12774628

Parking Lot Occupation

- Fabián, T.: **A Vision-based Algorithm for Parking Lot Utilization Evaluation Using Conditional Random Fields**. In 9th International Symposium on Visual Computing ISVC 2013, pp. 1-12 (2013)

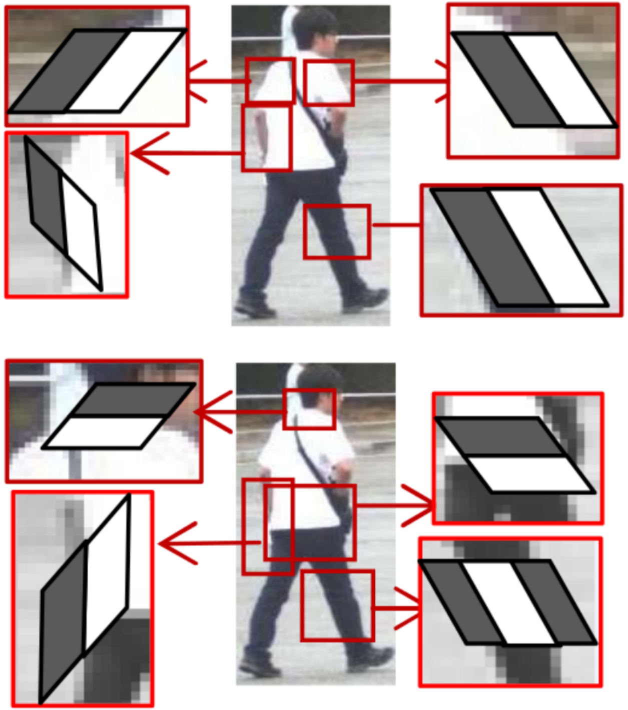
- Fusek, R., Mozdřeň, K., Šurkala, M., Sojka, E.: **AdaBoost for Parking Lot Occupation Detection**. Advances in Intelligent Systems and Computing, vol. 226, pp. 681-690 (2013)

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



Haar Features

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



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)

Object Detection/Recognition

- Haar
- HOG
- LBP

Traditional Approaches

- SIFT, SURF

KeyPoints

- CNNs

Deep Learning Approach

- Practical examples using OpenCV + Dlib (https://opencv.org/, http://dlib.net/)

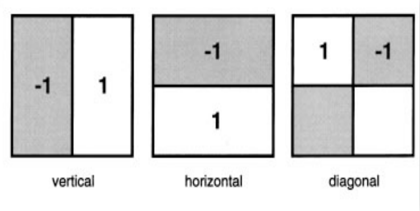
Related Works

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

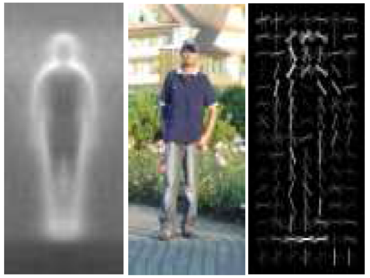
PAUL VIOLA
Microsoft Research, One Microsoft Way, Redmond, WA 98052, USA
viola@microsoft.com

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-Alps, 655 avenue de l'Europe, Montbonnot 38334, France
{Navneet.Dalal,Bill.Triggs}@inrialpes.fr, http://lear.inrialpes.fr

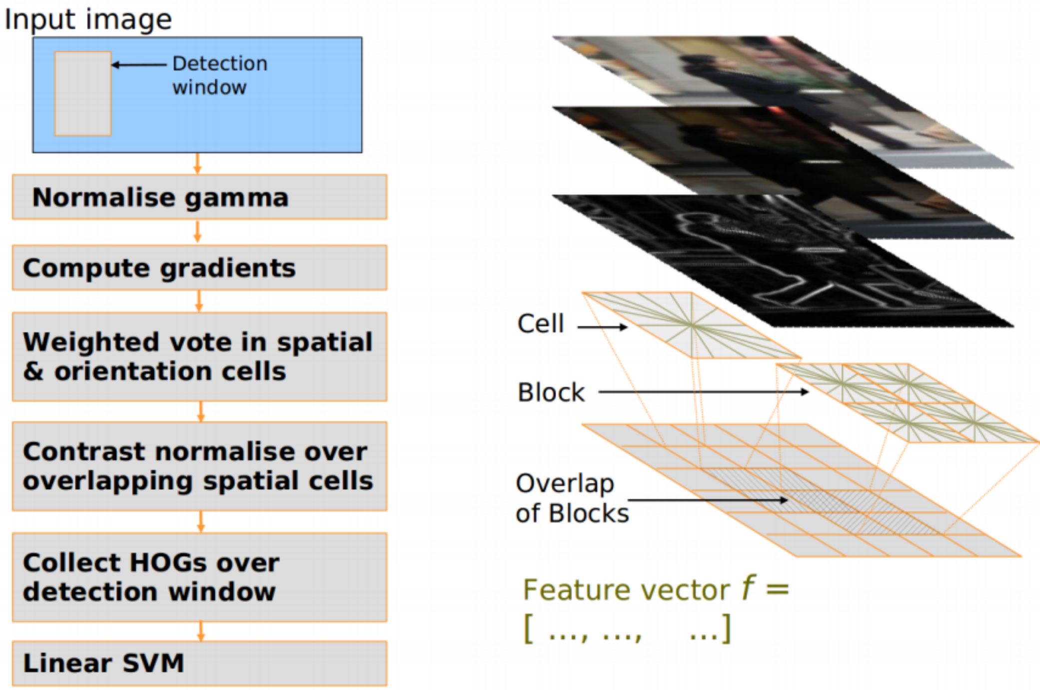


2005

Histograms of Oriented Gradients (HOG)

Basic Steps:

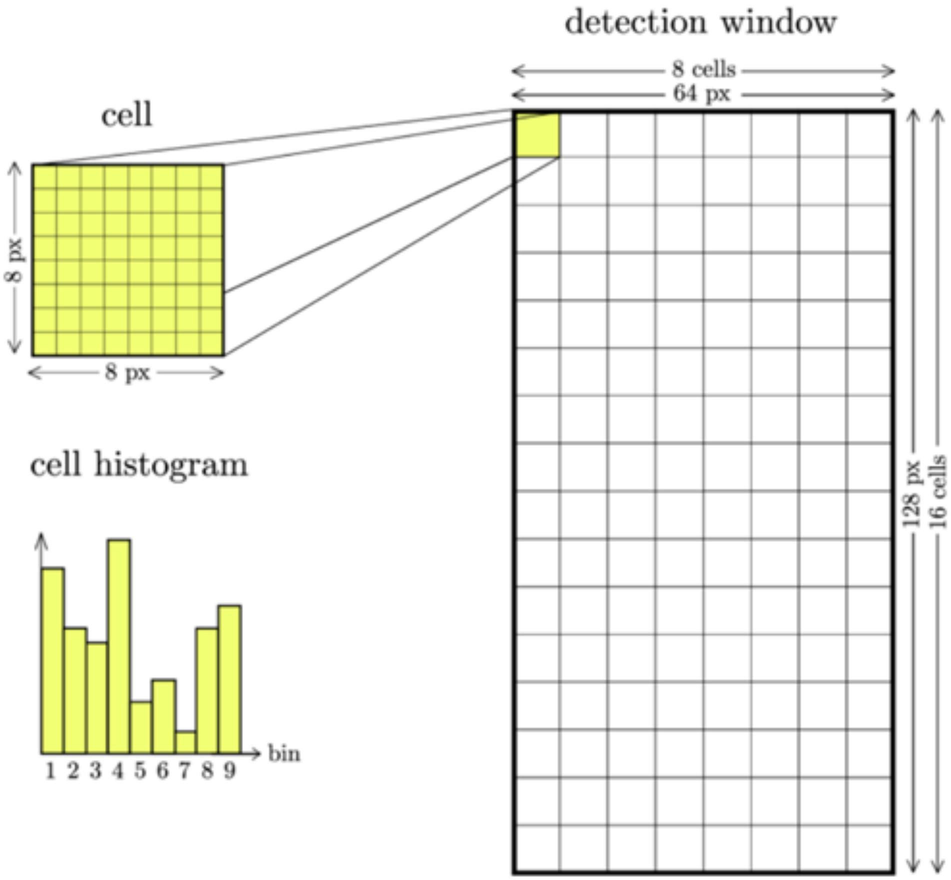
- In HOG, a sliding window is used for detection.
- The window is divided into small connected cells.
- The histograms of gradient orientations are calculated in each cell.
- Support Vector Machine (SVM) classifier.



http://host.robots.ox.ac.uk/pascal/VOC/voc2006/slides/dalal.ppt

Histograms of Oriented Gradients (HOG)

Blocks, Cells:

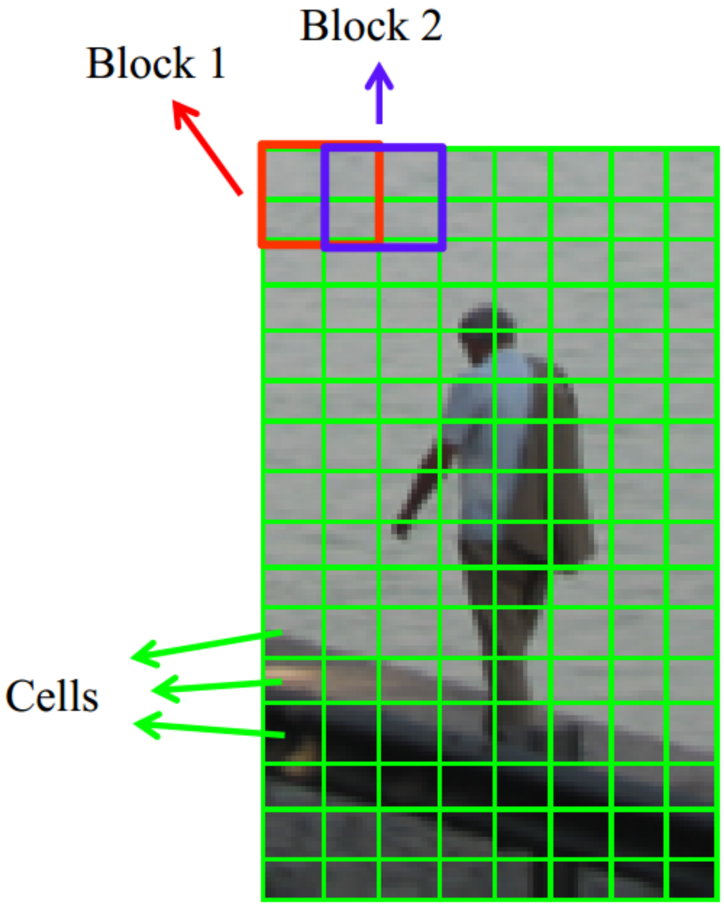


Histograms of Oriented Gradients (HOG)

Blocks, Cells:

- 8 x 8 cell
- 16 x 16 block – overlap
- normalization within the blocks

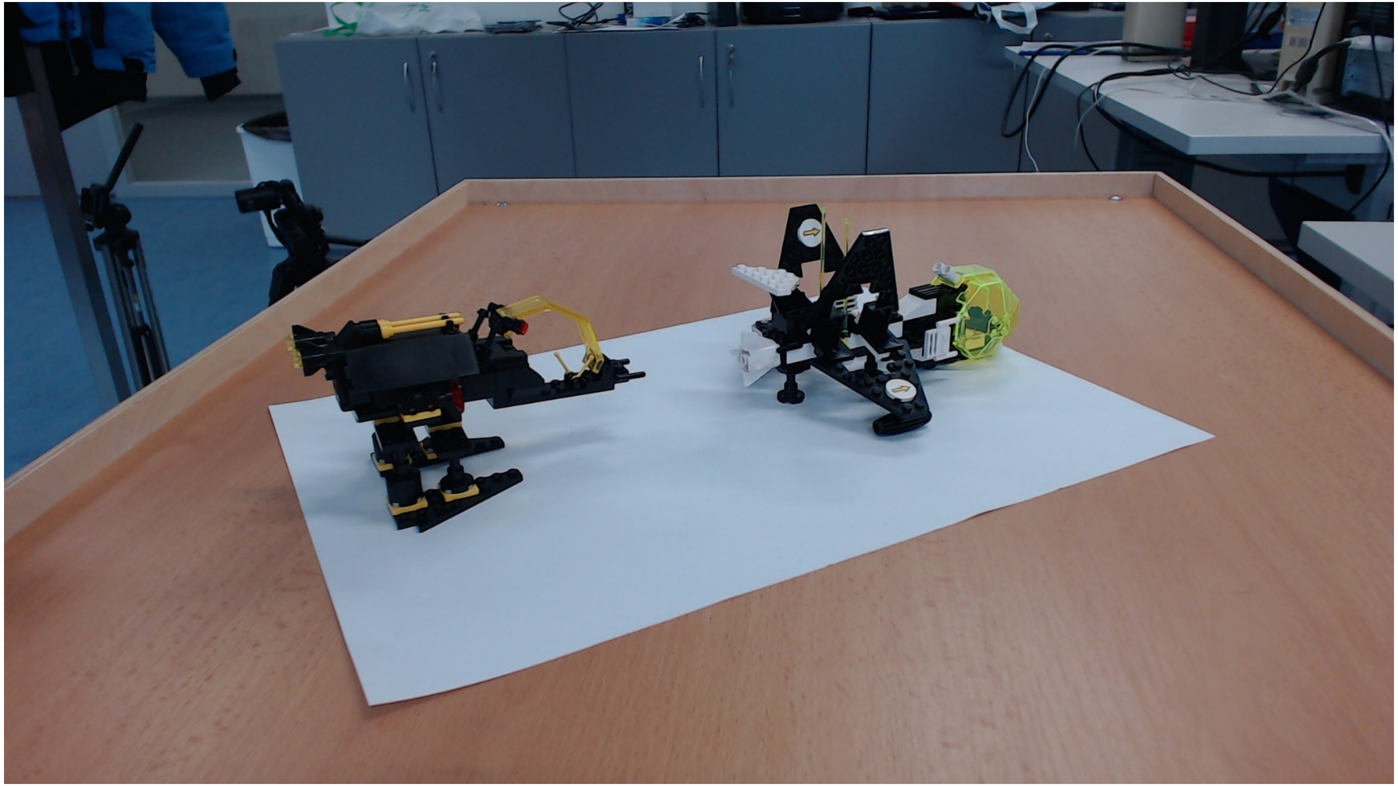
<u>Final Vector</u>: Collect HOG blocks into vector



Histograms of Oriented Gradients (HOG)

Practical Example – Detection + Recognition

Consider the following problem: Find and recognize two following lego kits



OpenCV - http://opencv.org/

OpenCV 3.3.0
Open Source Computer Vision

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:

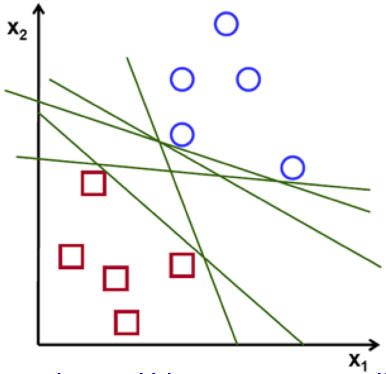
- **Core functionality** - a compact module defining basic data structures, including the dense multi-dimensional array Mat and basic functions used by all other modules.
- **Image processing** - an image processing module that includes linear and non-linear image filtering, geometrical image transformations (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.

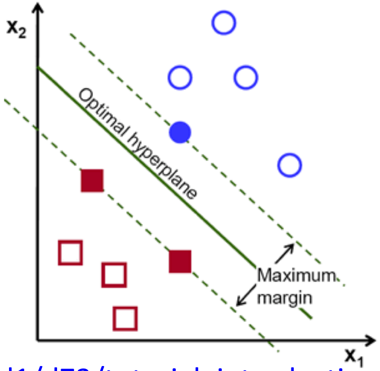
The further chapters of the document describe functionality of each module. But first, make sure to get familiar with the common API concepts used thoroughly in the library.

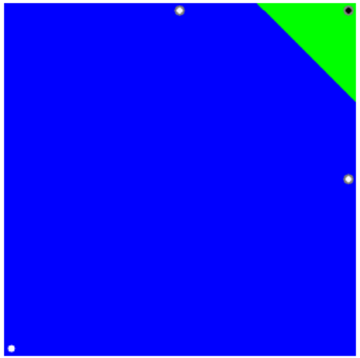
http://opencv.org/

Detection step - HOG+SVM (OpenCV)

```cpp
// Set up training data
int labels[4] = {1, -1, -1, -1};
Mat labelsMat(4, 1, CV_32SC1, labels);

float trainingData[4][2] = { {501, 10}, {255, 10}, {501, 255}, {10, 501} };
Mat trainingDataMat(4, 2, CV_32FC1, trainingData);

// Set up SVM's parameters
SVM::Params params;
params.svmType    = SVM::C_SVC;
params.kernelType = SVM::LINEAR;
params.termCrit   = TermCriteria(TermCriteria::MAX_ITER, 100, 1e-6);

// Train the SVM
Ptr<SVM> svm = StatModel::train<SVM>(trainingDataMat, ROW_SAMPLE, labelsMat, params);
```





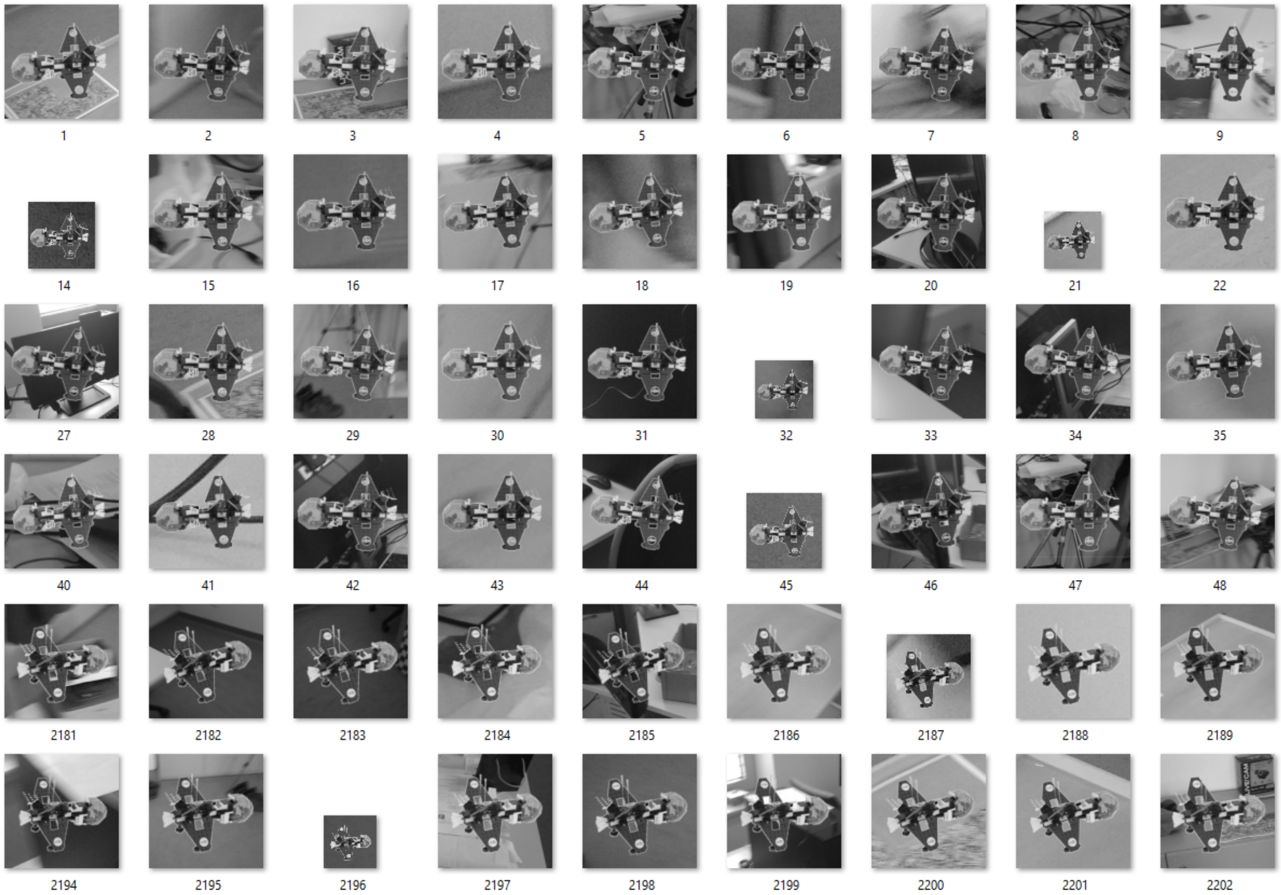


https://docs.opencv.org/3.1.0/d1/d73/tutorial_introduction_to_svm.html

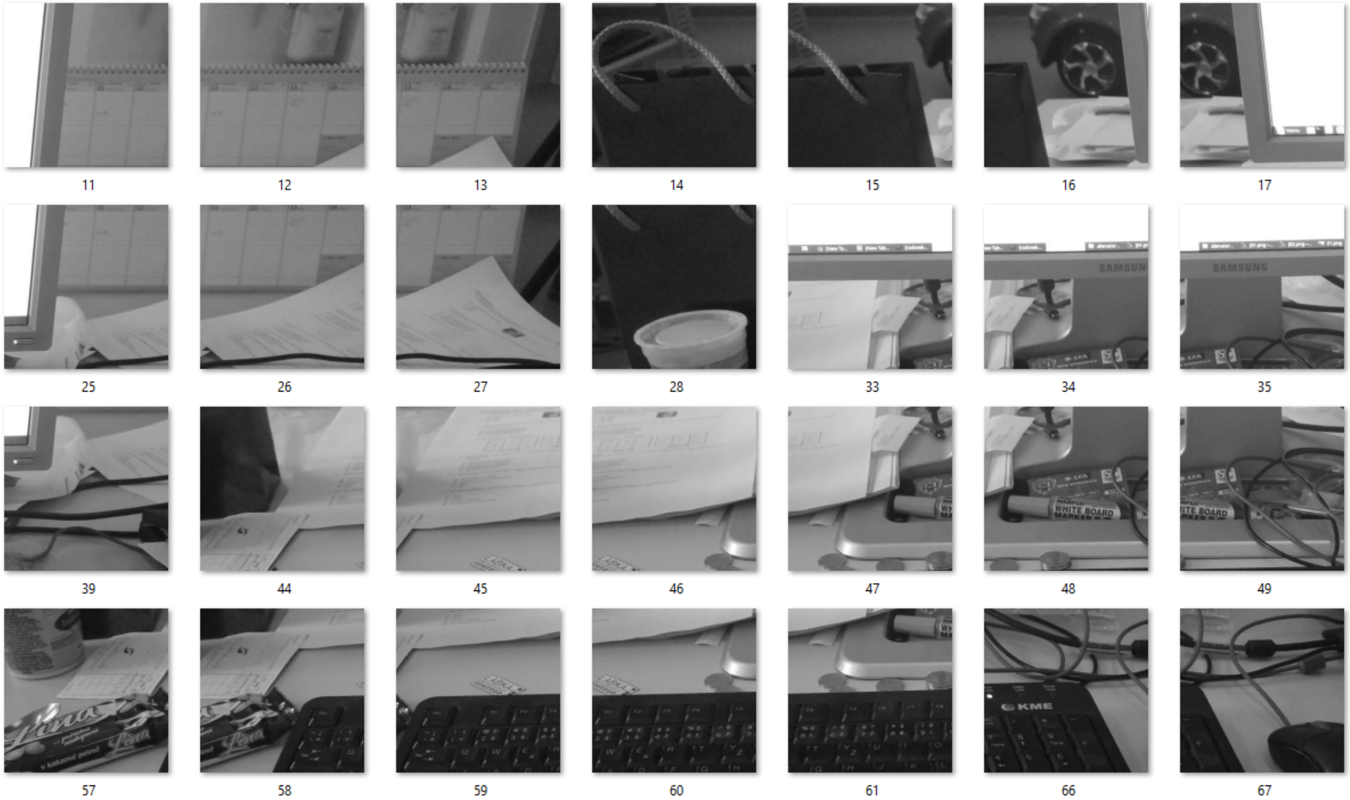
Alien


1 2 3 4 5 6 7 8 9
15 16 17 18 19 20 21 22 23
29 30 31 32 33 34 35 36 37
43 44 45 46 47 48 49 50 51
2378 2379 2380 2381 2382 2383 2384 2385 2386
2392 2393 2394 2395 2396 2397 2398 2399 2400

Avenger

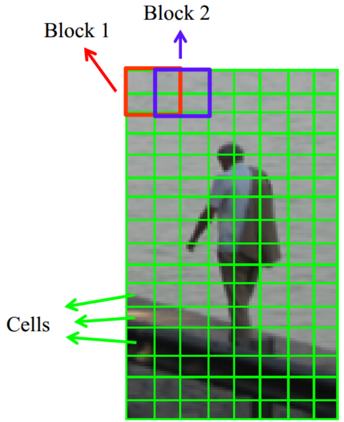

1 2 3 4 5 6 7 8 9
14 15 16 17 18 19 20 21 22
27 28 29 30 31 32 33 34 35
40 41 42 43 44 45 46 47 48
2181 2182 2183 2184 2185 2186 2187 2188 2189
2194 2195 2196 2197 2198 2199 2200 2201 2202

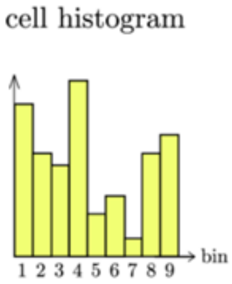
Detection step - HOG+SVM (OpenCV)



Detection step - HOG+SVM (OpenCV)
Sliding Window (detectMultiScale)

```cpp
int blockSize = 16;
int cellSize = 8;
int strideSize = 8;
int winSize = 64;

//HOGDescriptor hog;
HOGDescriptor my_hog(
            cv::Size(winSize,winSize), //winSize
            cv::Size(blockSize,blockSize), //blocksize
            cv::Size(strideSize,strideSize), //blockStride,
            cv::Size(cellSize,cellSize), //cellSize,
            9, //nbins,
            );

//SVM
Ptr<SVM> svm = StatModel::load<SVM>( classifierName );
std::vector< float > hog_detector;
//get the support vectors
get_svm_detector( svm, hog_detector );
//set SVM
my_hog.setSVMDetector( hog_detector );

std::vector<Rect> positivesAll;
my_hog.detectMultiScale( frameGray, positivesAll, 0,
        Size(0,0), Size(0,0), 1.1, 4);
```

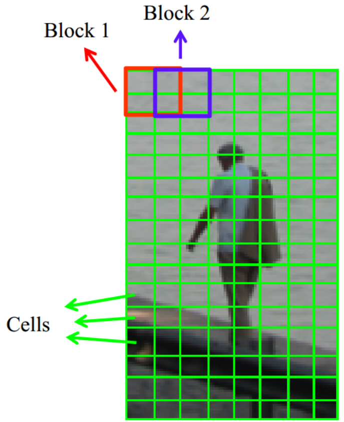


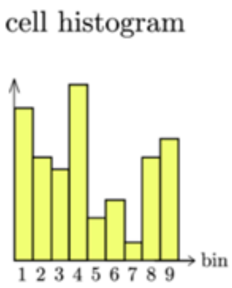


https://github.com/opencv/opencv/blob/master/samples/cpp/train_HOG.cpp

Detection step - HOG+SVM (OpenCV)

```cpp
int blockSize = 16;
int cellSize = 8;
int strideSize = 8;
int winSize = 64;

//HOGDescriptor hog;
HOGDescriptor my_hog(
            cv::Size(winSize,winSize), //winSize
            cv::Size(blockSize,blockSize), //blocksize
            cv::Size(strideSize,strideSize), //blockStride,
            cv::Size(cellSize,cellSize), //cellSize,
            9, //nbins,
            );

//SVM
Ptr<SVM> svm = StatModel::load<SVM>( classifierName );
std::vector< float > hog_detector;
//get the support vectors
get_svm_detector( svm, hog_detector );
//set SVM
my_hog.setSVMDetector( hog_detector );

std::vector<Rect> positivesAll;
my_hog.detectMultiScale( frameGray, positivesAll, 0,
        Size(0,0), Size(0,0), 1.1, 4);
```





Detection step - HOG+SVM (OpenCV)

Object Detection/Recognition

- Haar
- HOG
- LBP

Traditional Approaches

- SIFT, SURF

KeyPoints

- CNNs

Deep Learning Approach

- Practical examples using OpenCV + Dlib (https://opencv.org/, http://dlib.net/)

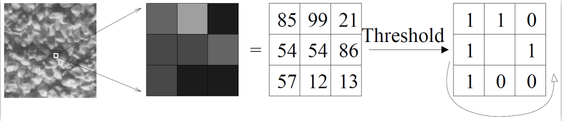
Related Works

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*

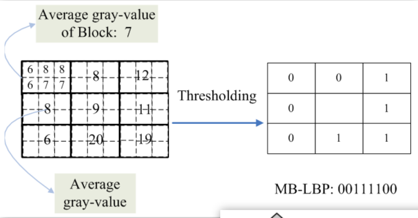


Zhang at al. (2007)

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

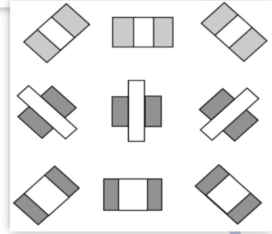


Xiaohua at al. (2009)

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

Li Xiaohua [a,b], Kin-Man Lam [b,*], Shen Lansun [c], Zhou Jiliu [a]

[a] *Department of Computer Science, Sichuan University, Chengdu 610064, China*
[b] *Centre for Signal Processing, Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong*
[c] *Signal and Information Processing Lab., Beijing University of Technology, Beijing 100022, China*



2009

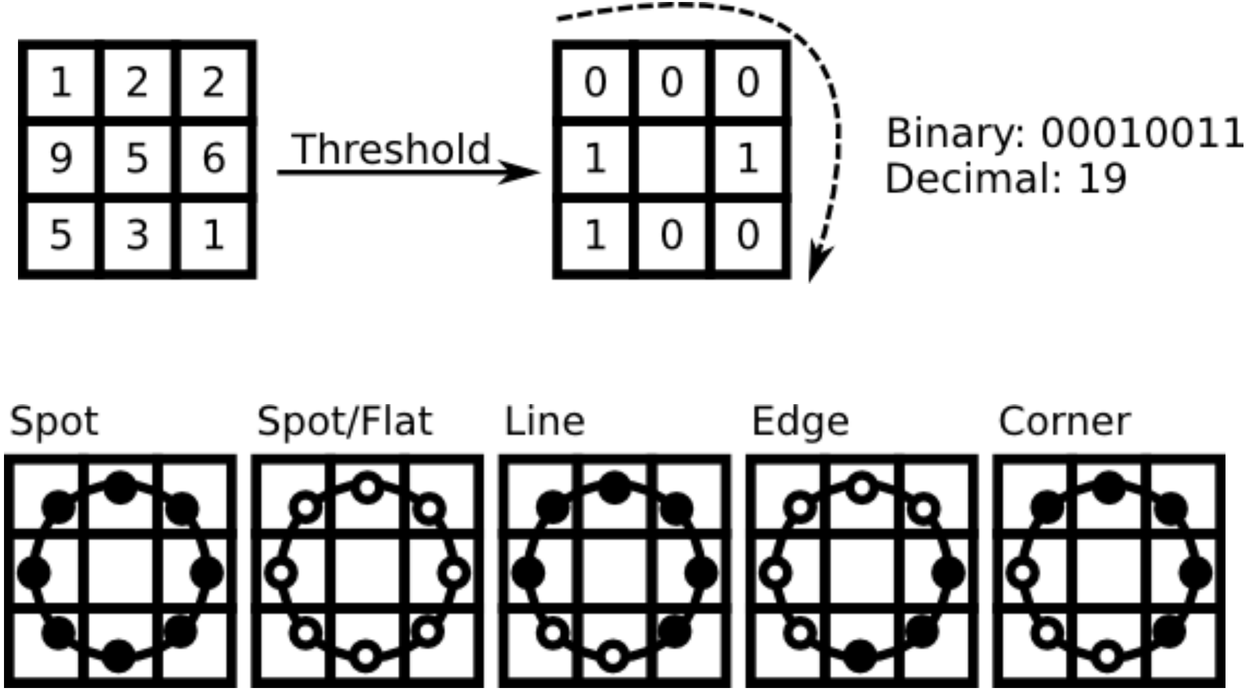
LBP - Local Binary Patterns

- Were introduced by Ojala et al. 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.

- Fast and cheap technique

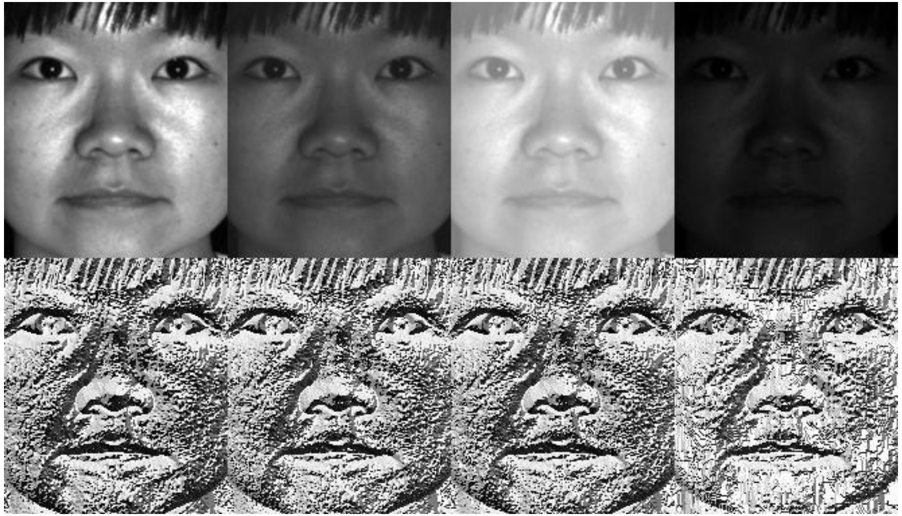
LBP - Local Binary Patterns



http://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html

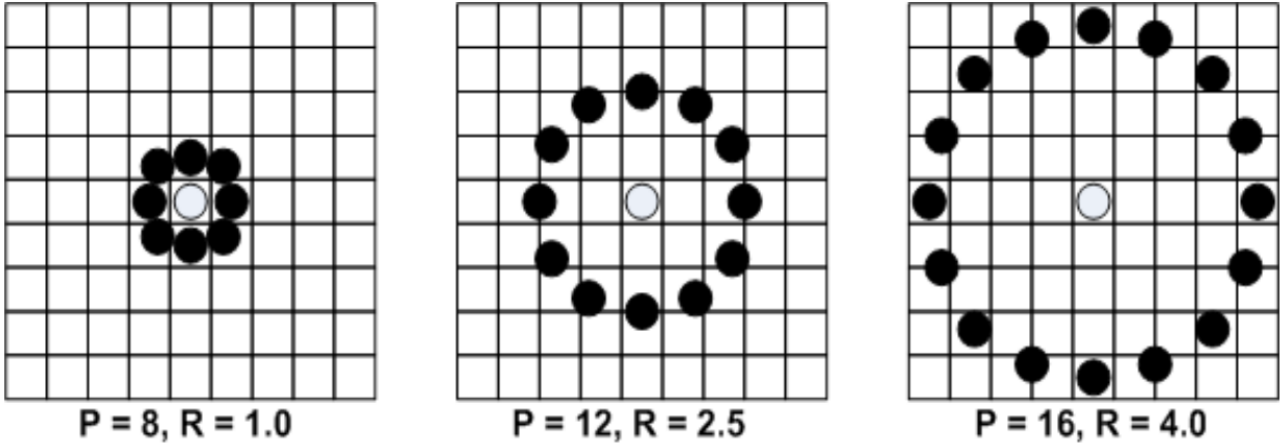
LBP - Local Binary Patterns

- Robust to monotonic changes in illumination



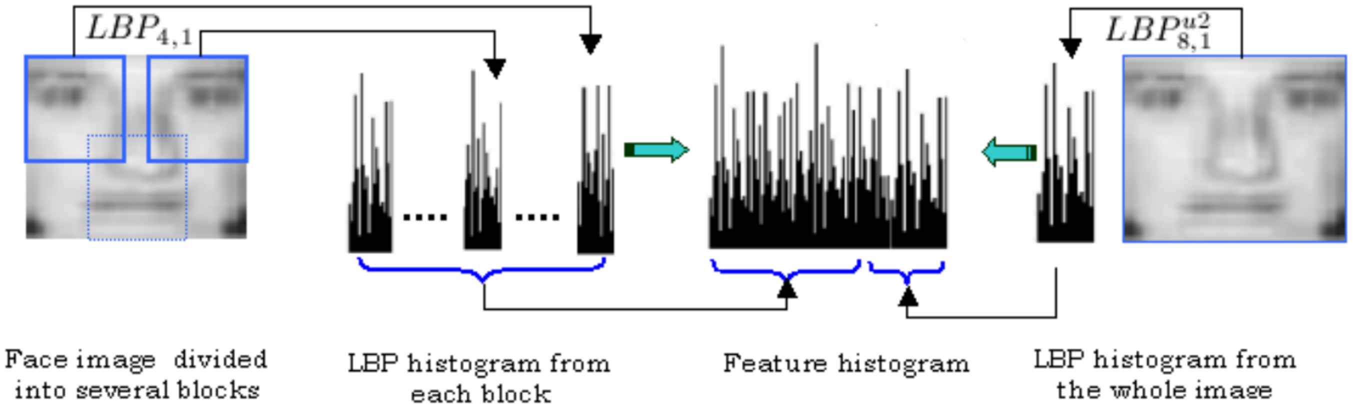
http://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html

LBP - Local Binary Patterns



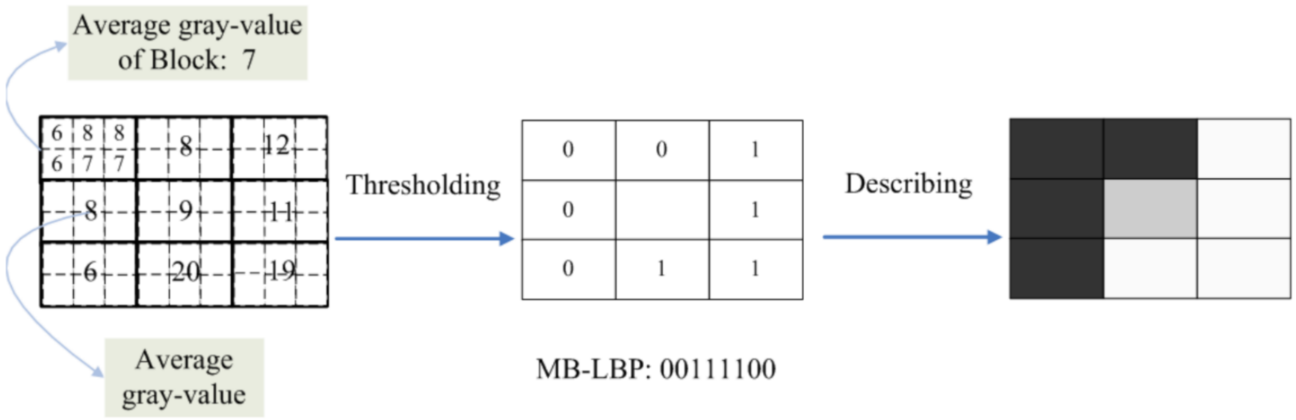
Ojala T, Pietikäinen M & Mäenpää T (2002) Multiresolution gray-scale and rotation invariant texture classification with Local Binary Patterns. IEEE Transactions on Pattern Analysis and Machine Intelligence 24(7):971-987

LBP - Local Binary Patterns



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



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)

Object Detection/Recognition

- Haar
- HOG
- LBP

Traditional Approaches

- SIFT, SURF

KeyPoints

- CNNs

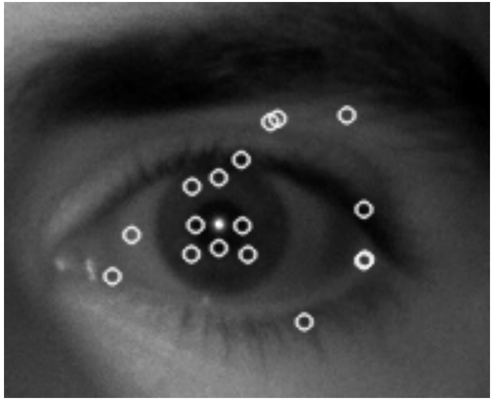
Deep Learning Approach

- Practical examples using OpenCV + Dlib (https://opencv.org/, http://dlib.net/)

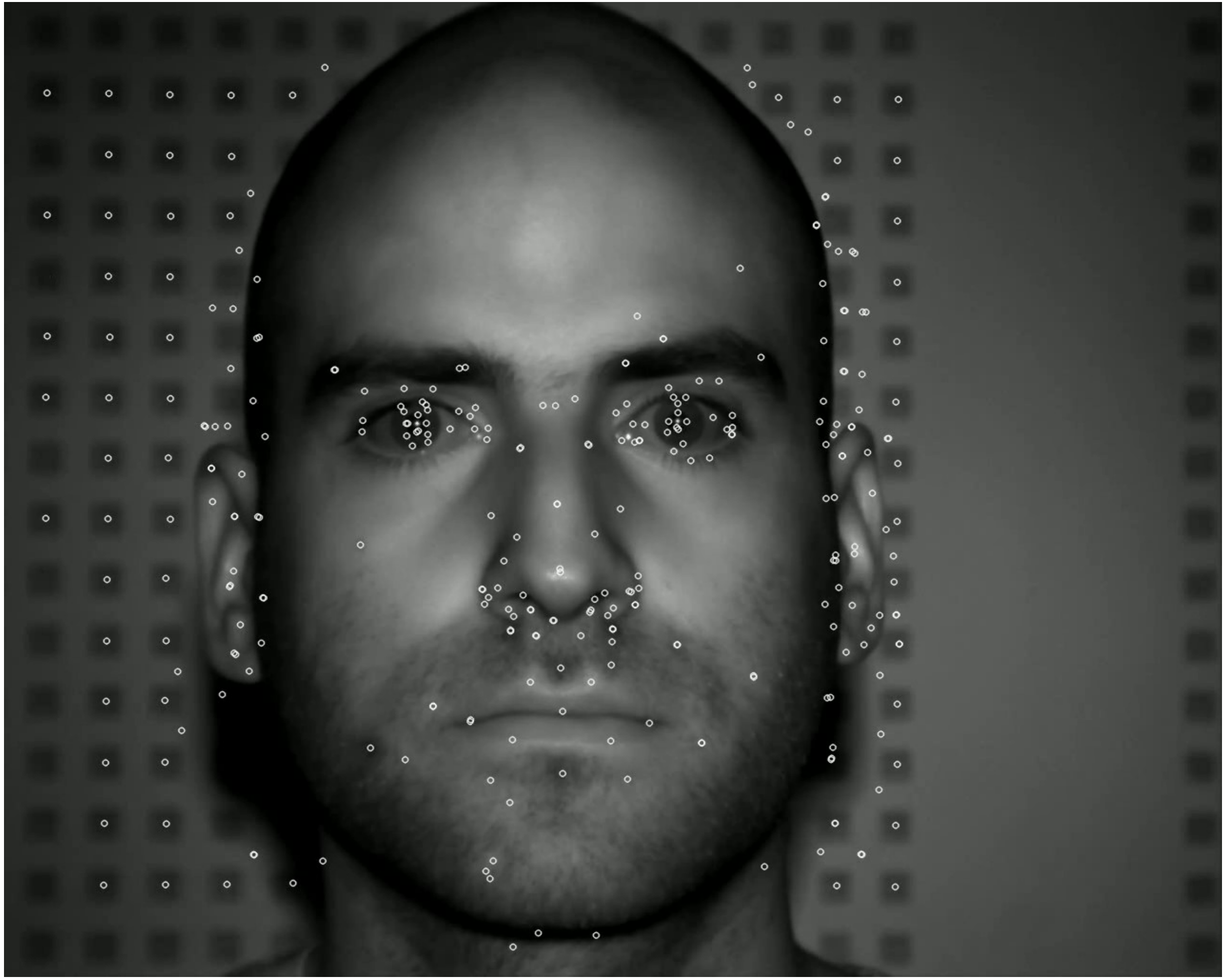
KeyPoints

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

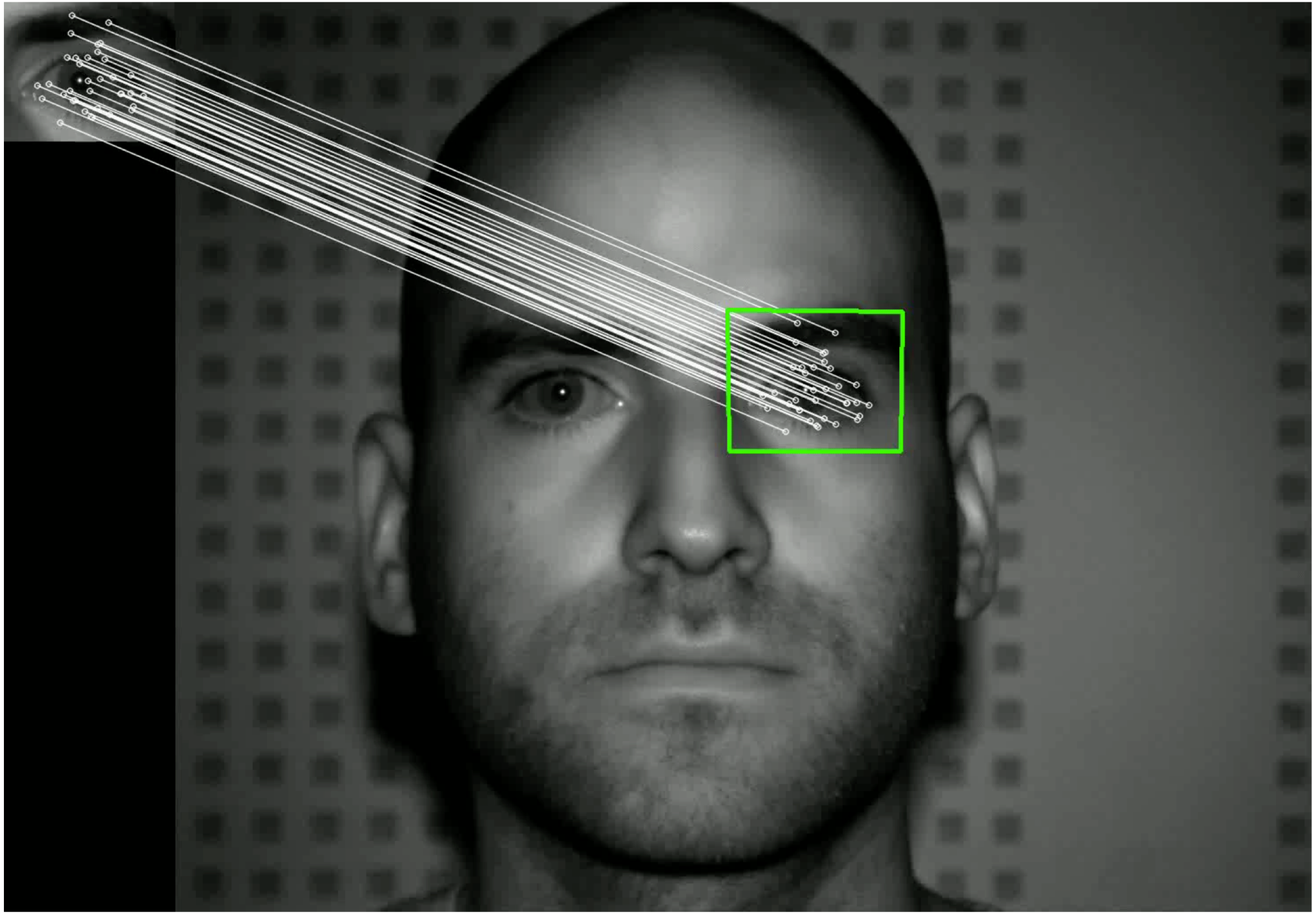
KeyPoints – Eye Detection



template



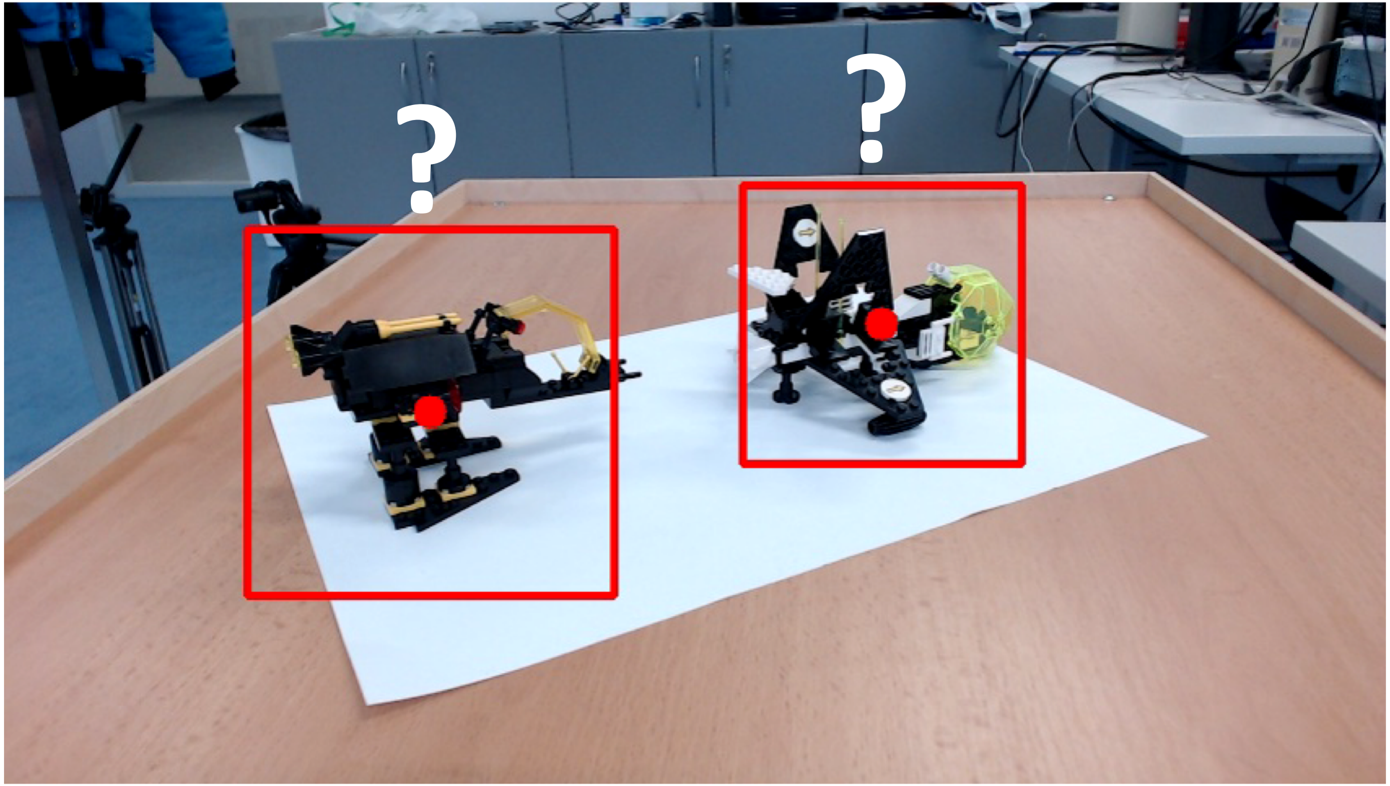
KeyPoints – Eye Detection



https://docs.opencv.org/3.1.0/d5/d6f/tutorial_feature_flann_matcher.html

Recognition

Alien vs. Avenger



Object Detection/Recognition

- Haar
- HOG
- LBP

Traditional Approaches

- SIFT, SURF

KeyPoints

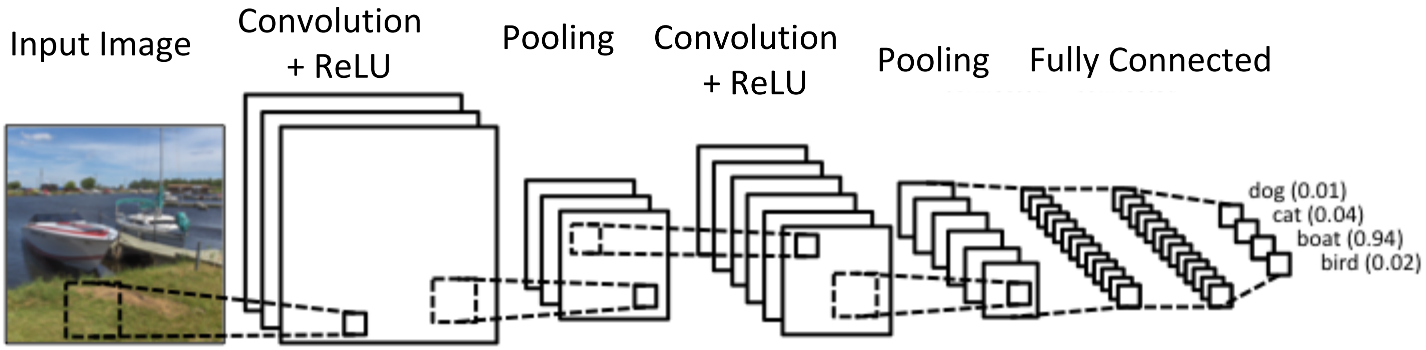
- CNNs

Deep Learning Approach

- Practical examples using OpenCV + Dlib (https://opencv.org/, http://dlib.net/)

CNNs – Main Steps (LeNet)

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



https://www.clarifai.com/technology

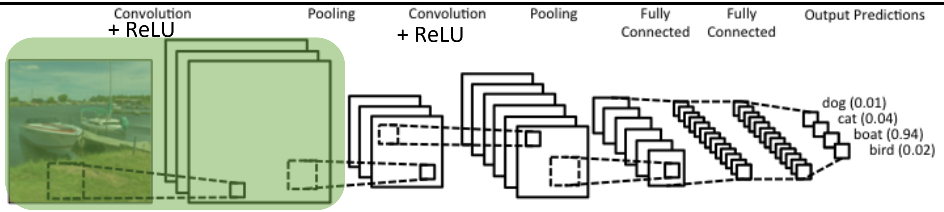
1. Convolution

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

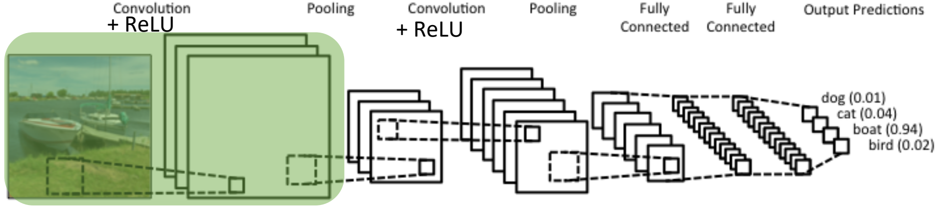
Mask/Filter



https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

1. Convolution

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



https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

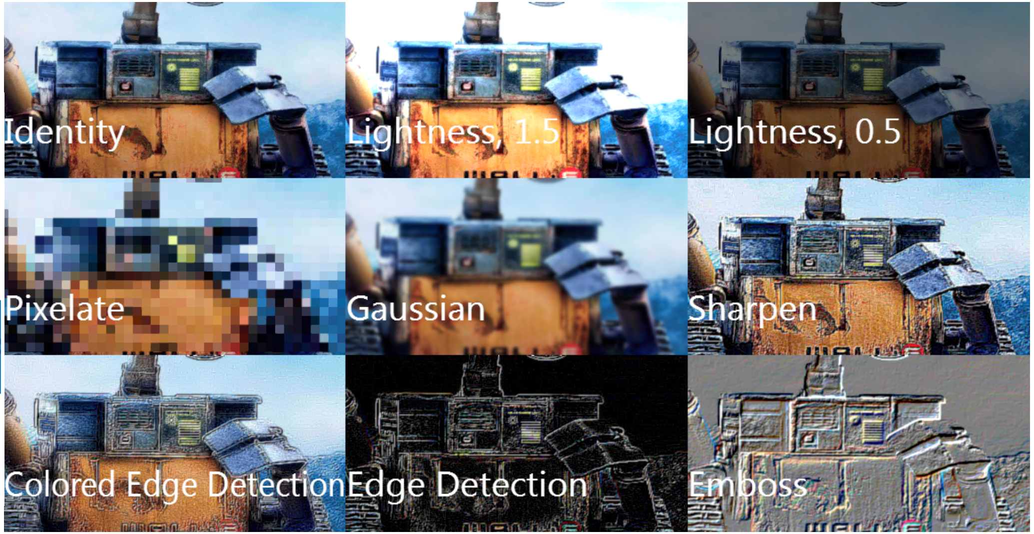
1. Convolution

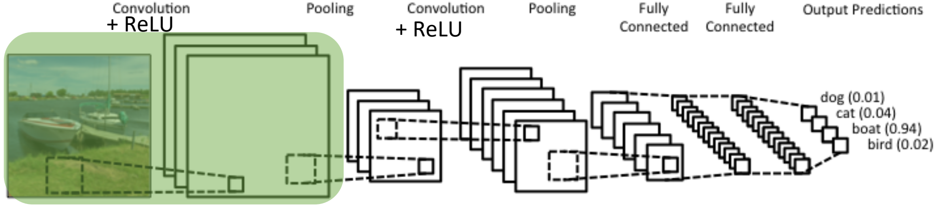
Sharpen – $\begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$

Blur – $\begin{bmatrix} 0 & 0.2 & 0 \\ 0.2 & 0.2 & 0.2 \\ 0 & 0.2 & 0 \end{bmatrix}$

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

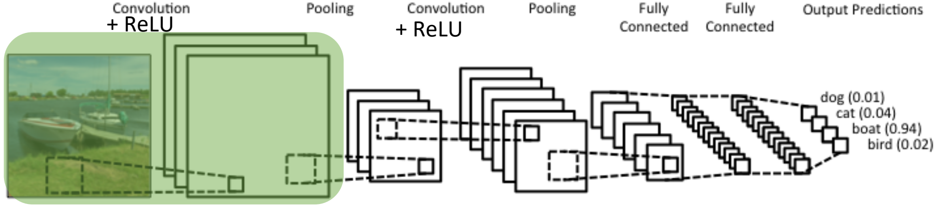
Edge detect – $\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$





http://dimitroff.bg/image-filtering-your-own-instagram/

1. Convolution

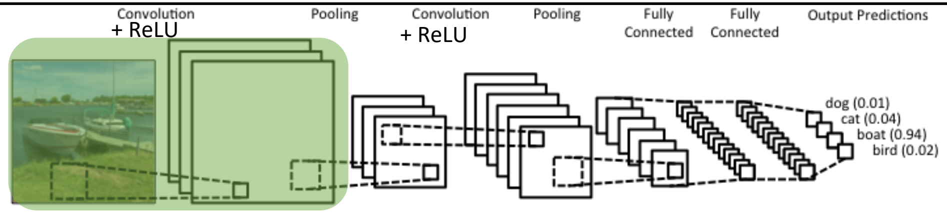


http://cs.nyu.edu/~fergus/tutorials/deep_learning_cvpr12/

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

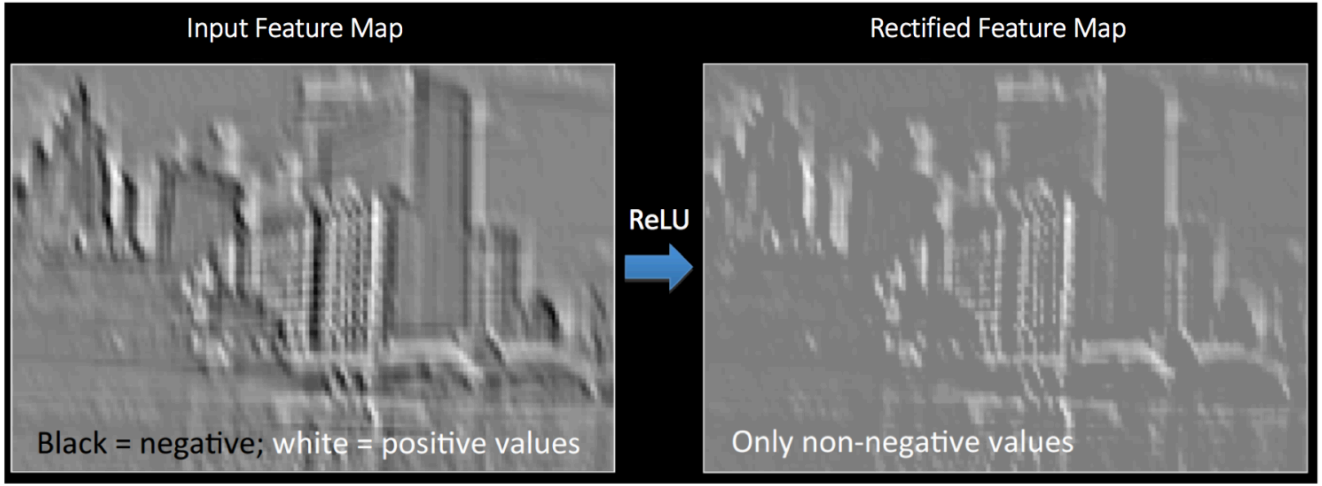
- The filters are learned during the training

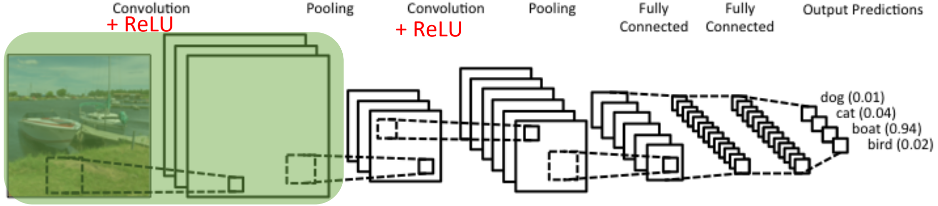


http://cs.nyu.edu/~fergus/tutorials/deep_learning_cvpr12/

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





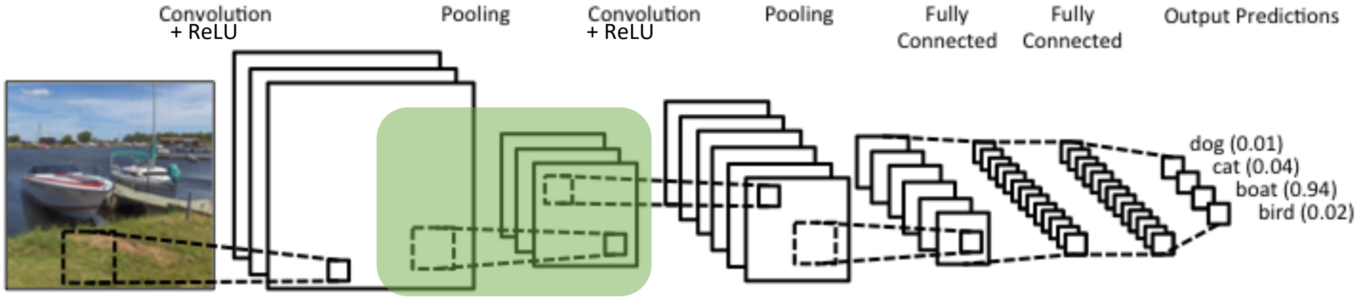
http://mlss.tuebingen.mpg.de/2015/slides/fergus/Fergus_1.pdf

3. Pooling

(Subsampling or downsampling)

- The goal of this step is to reduce the dimensionality of each feature map but preserve important informations

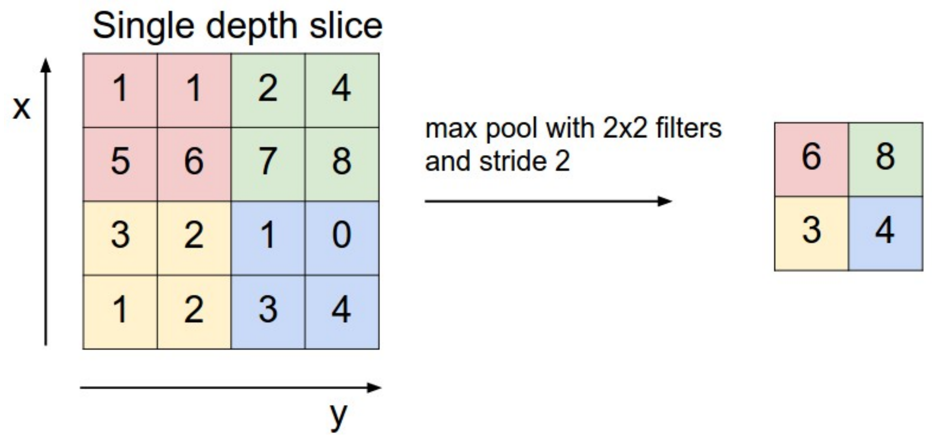
- Operations: e.g. Sum, Average, Max

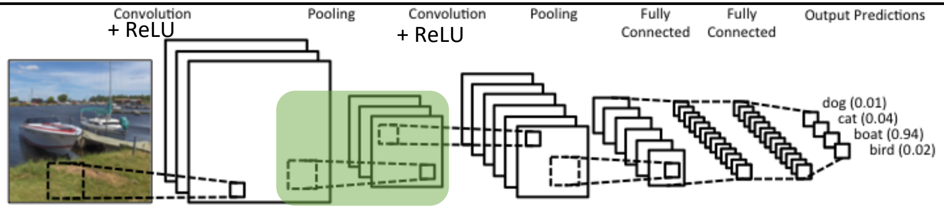


3. Pooling

(Subsampling or downsampling)

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



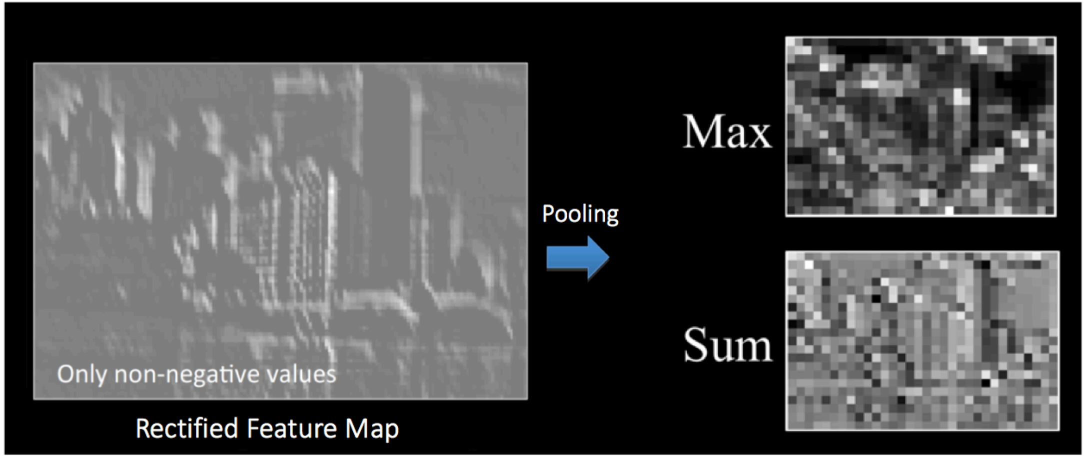


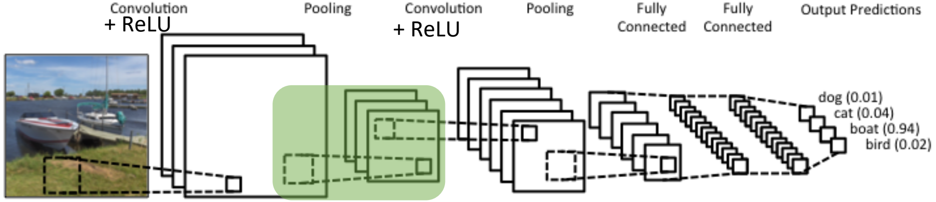
http://cs231n.github.io/convolutional-networks/

3. Pooling

(Subsampling or downsampling)

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

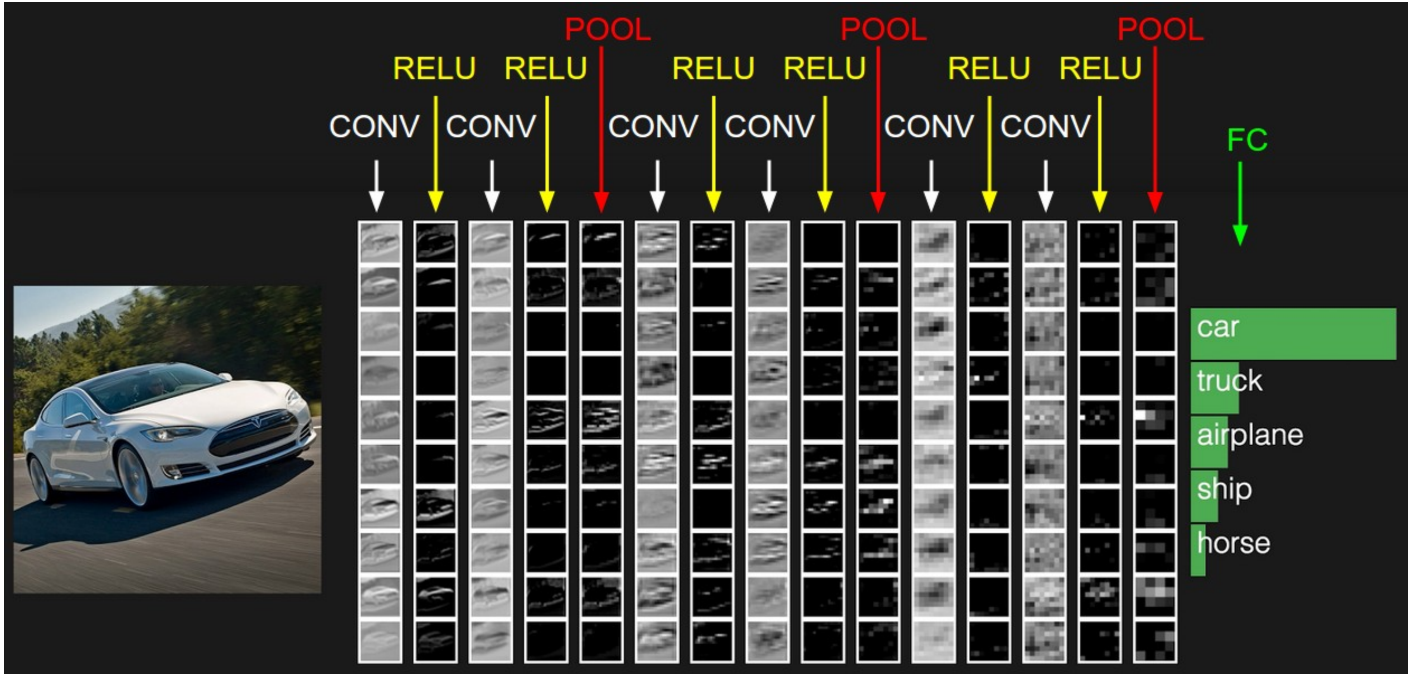




http://mlss.tuebingen.mpg.de/2015/slides/fergus/Fergus_1.pdf

Conv. + ReLU + POOL

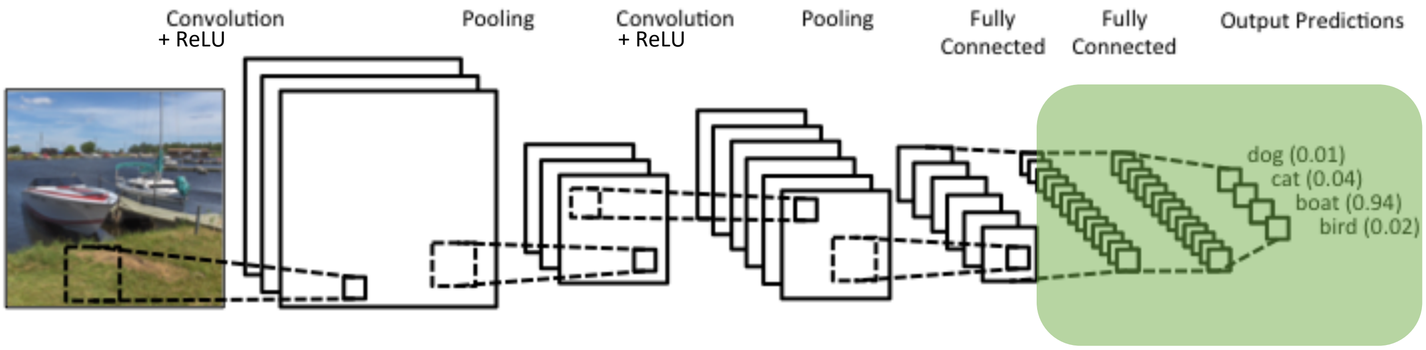
- Convolution layers and Pooling layers can be repeated any number of times in a single ConvNet.



http://cs231n.github.io/convolutional-networks/

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.



http://cs231n.github.io/convolutional-networks/

4. CovNet Architectures

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

https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

Dlib http://dlib.net

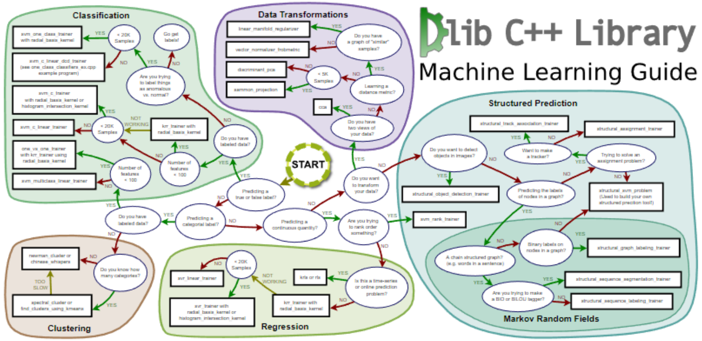
Dlib C++ Library

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



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:

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

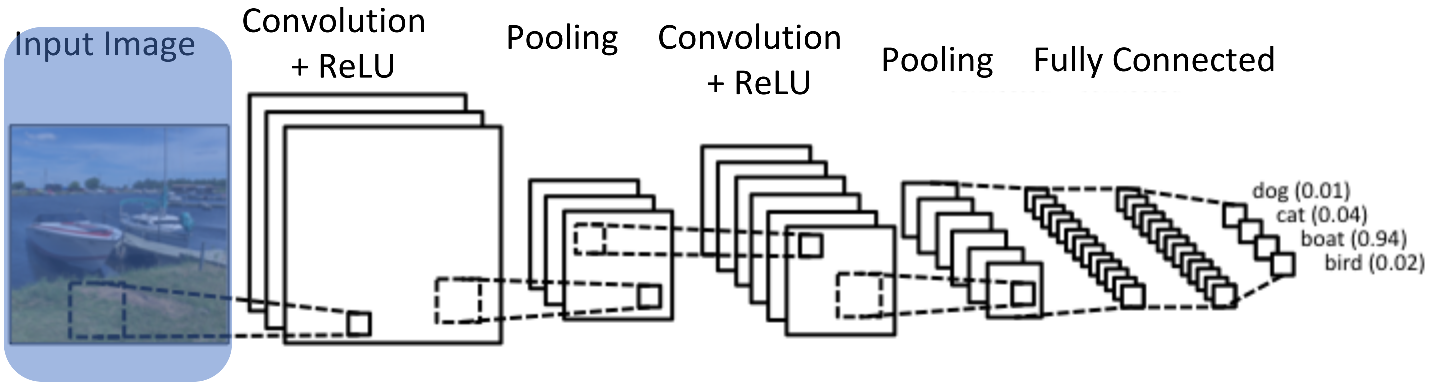
Regression
- krls
- krr_trainer
- mlp
- rbf_network_trainer
- rls
- rr_trainer
- rvm_regression_trainer
- svr_linear_trainer
- svr_trainer

http://dlib.net

Recognition step CNNs (Dlib)

```
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

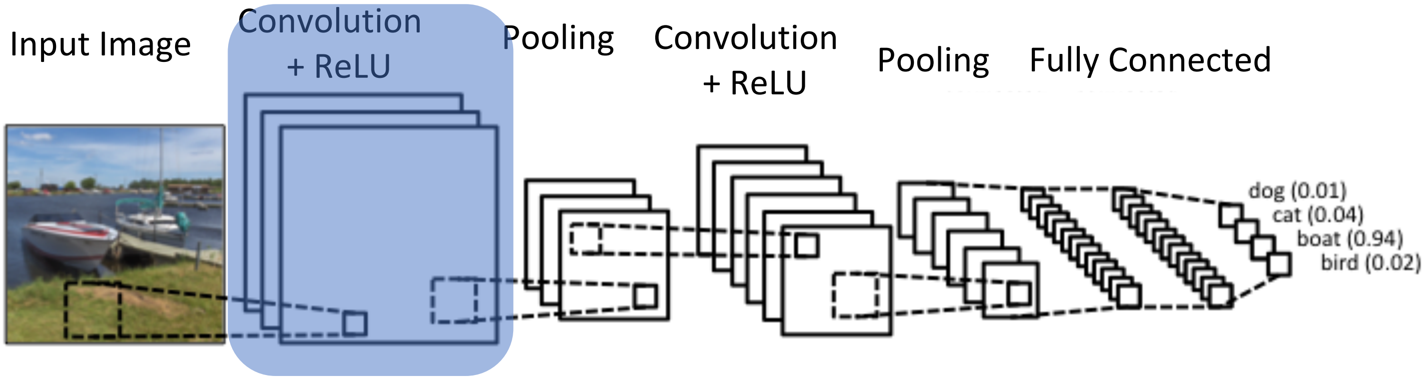


http://dlib.net/dnn_introduction_ex.cpp.html

Recognition step CNNs (Dlib)

```
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



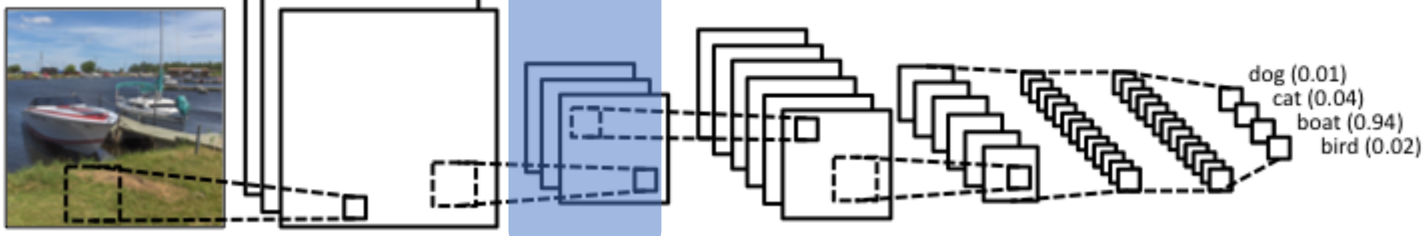
http://dlib.net/dnn_introduction_ex.cpp.html

Recognition step CNNs (Dlib)

```
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

Input Image | Convolution + ReLU | Pooling | Convolution + ReLU | Pooling | Fully Connected

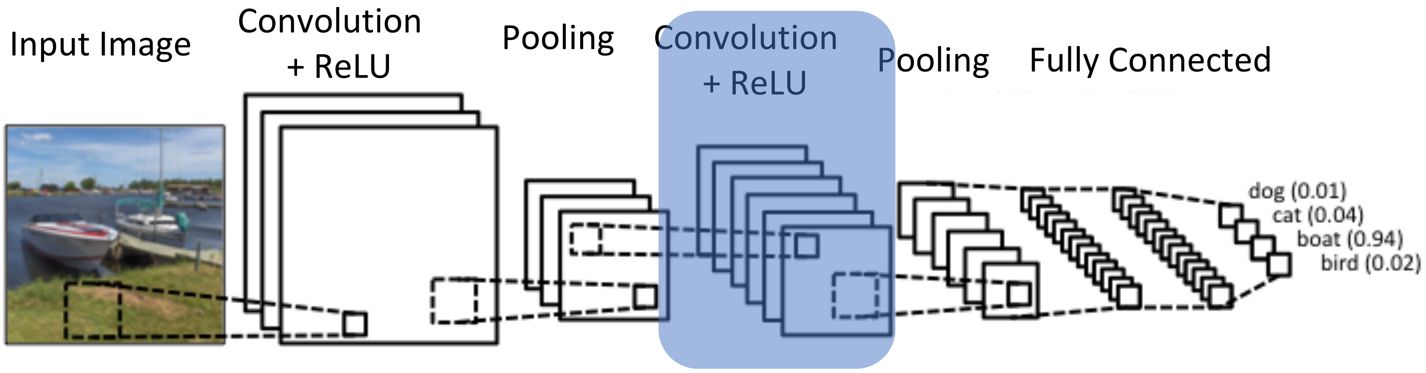


http://dlib.net/dnn_introduction_ex.cpp.html

Recognition step CNNs (Dlib)

```
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

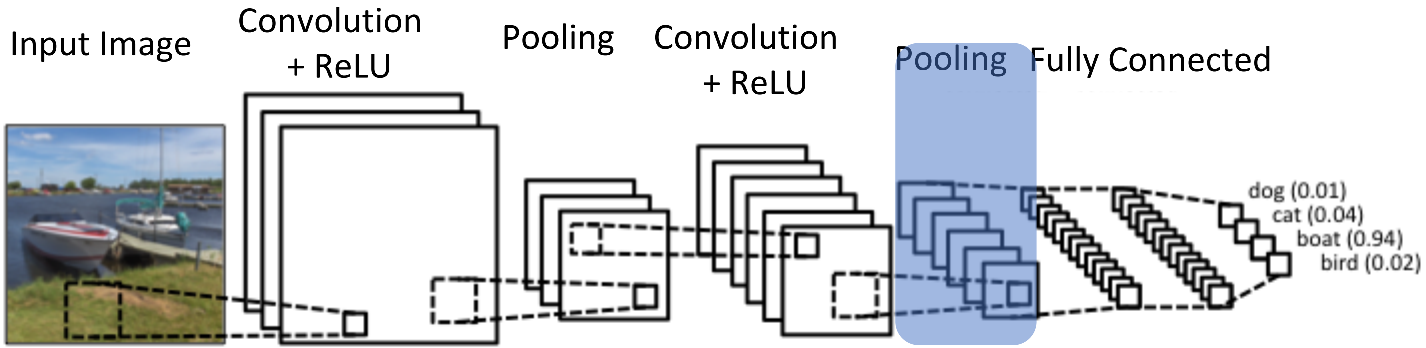


http://dlib.net/dnn_introduction_ex.cpp.html

Recognition step CNNs (Dlib)

```
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

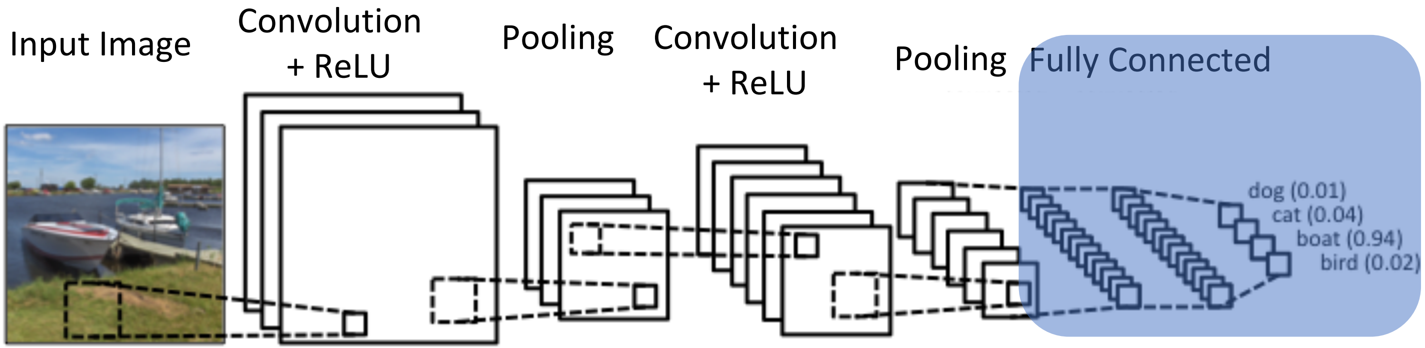


http://dlib.net/dnn_introduction_ex.cpp.html

Recognition step CNNs (Dlib)

```
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>>
                            >>>>>>>>>>>>;
```

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



http://dlib.net/dnn_introduction_ex.cpp.html

Recognition step CNNs (Dlib)

```
// network instance
net_type net;

// mini-batch stochastic gradient descent
//dnn_trainer<net_type> trainer(net, sgd(), {0,1}); //{0,1} - will use two GPU
dnn_trainer<net_type> trainer(net);
trainer.set_learning_rate(0.01);
trainer.set_min_learning_rate(0.0001);
trainer.set_mini_batch_size(160);
trainer.set_iterations_without_progress_threshold(500);
trainer.set_max_num_epochs(100);
trainer.be_verbose();
//train
trainer.train(train_images, train_labels);
// save
serialize("LeNet.dat") << net;
```

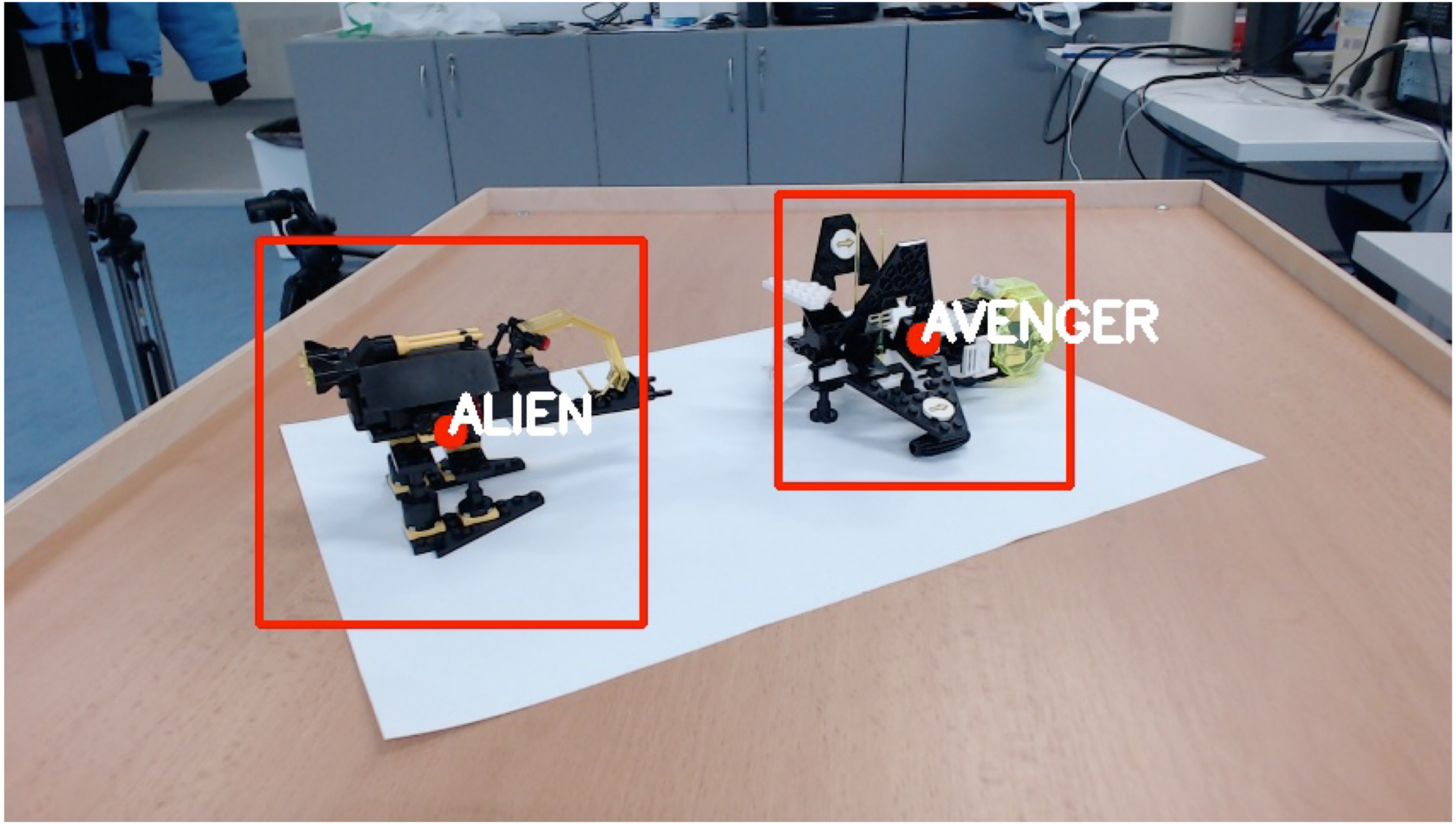
http://dlib.net/dnn_introduction_ex.cpp.html

Recognition step CNNs (Dlib + OpenCV)

```
//Load image using OpenCV
Mat frame;
frame = imread( "my_img.png", 1 );
cvtColor( frame, frame, COLOR_BGR2GRAY );
medianBlur(frame, frame, 5);

 //OpenCV Mat to Dlib
cv_image<unsigned char> cimg(frame);
matrix<unsigned char> dlibFrame = dlib::mat(cimg);

//prediction using CNN
unsigned long predict_label = net(frame);
```

http://dlib.net/dnn_introduction_ex.cpp.html

Recognition step CNNs (dlib)



CNNs (Dlib)

NVIDIA 1080ti - 39 frames per second, 928x478

http://blog.dlib.net/2017/08/vehicle-detection-with-dlib-195_27.html

Thank you for your attention

http://mrl.cs.vsb.cz