Latest Advances in Deep Learning

Yao Chou

Outline

- **Introduction**
- Images Classification
- Object Detection

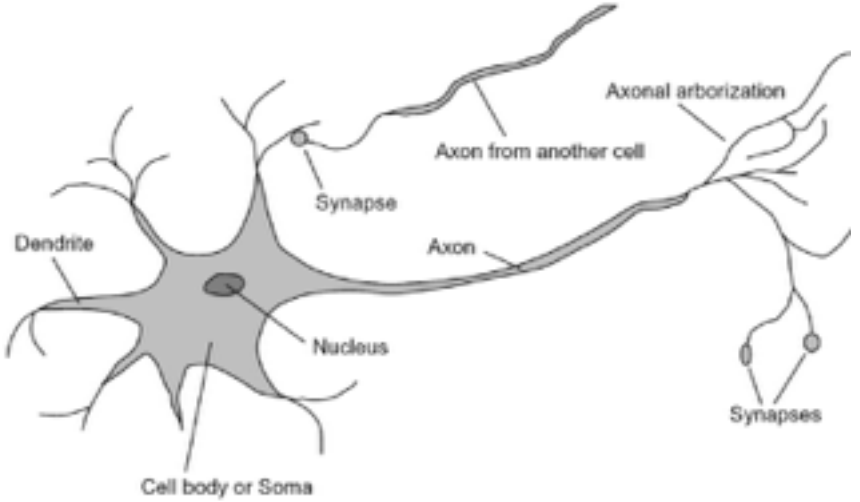
  R-CNN

  Traditional Feature Descriptor

  Selective Search
- Implementation
- Latest Application

Deep Learning

- A branch of machine learning
- Model high-level abstractions in data & Learn representations of data.
- Replacing handcrafted features with efficient algorithms for unsupervised or semi-supervised feature learning
- Inspiration: Neuron cells
- Future: Achieve AI



History

Perceptrons

·Born in 1960 (by Frank Rosenblatt) , First generation

·Input layer + a fixed hand-crafted feature+ an output layer

·Classify some basic shapes

· Fundamentally limited learning abilities

Hand-crafted feature

Single struction

History

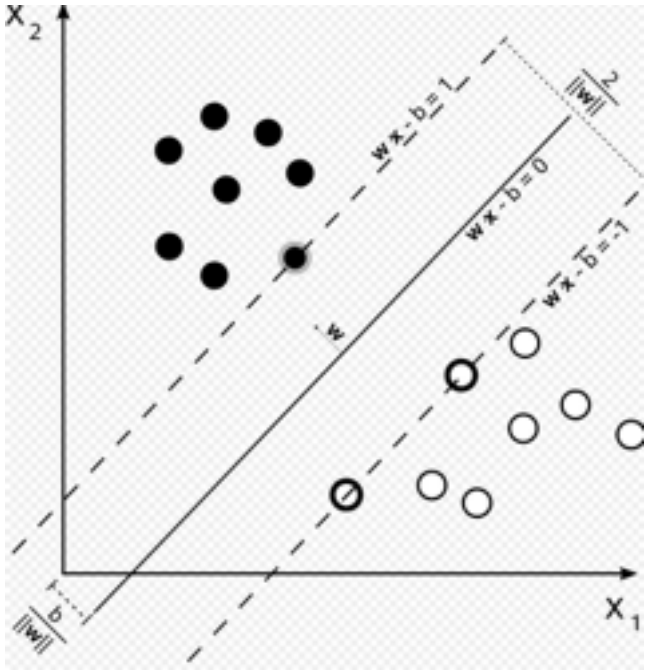
2nd-generation neural network

- In around 1985 (by Geoffrey Hinton)
- Replaced the original single fixed feature layer with several hidden layers
- Back-propagation
- The correcting signal will be weakened
- Too slow
- Get stuck in poor local optima

History

Support Vector Machines(SVM)

- Raised by Vladimir N. Vapnik and his co-workers in 1995
- Statistical learning theory
- The kernel function
- Features are directly obtained not learnt from the data itself



History

Convolutional Neural Network

- Exploit its advantages related to “deep" and overcome the limitations
- The one milestones architecture
- Only uses local connections and shared weights
- Big success in ILSVRC 2012

Outline

- Introduction
- **Images Classification**
- Object Detection

  R-CNN

  Traditional Feature Descriptor

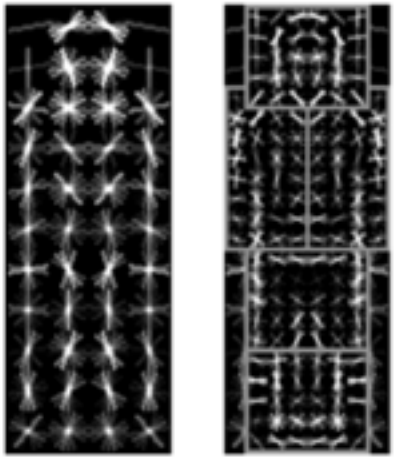
  Selective Search
- Implementation
- Latest Application

Traditional Approach

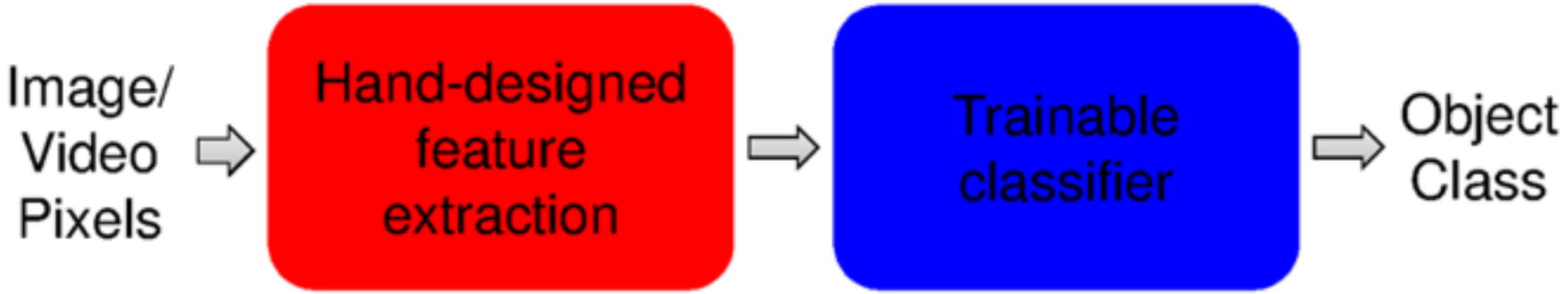
Features are the key

Multitude of hand-designed features

SIFT HOG…

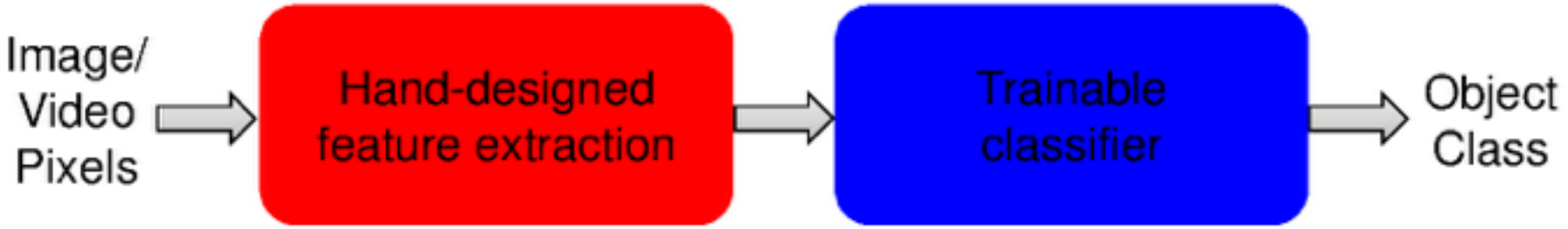


Felzenszwalb, Girshick, McAllester and Ramanan, PAMI 2007



“Shallow” vs. “deep” architectures

Traditional recognition: “Shallow” architecture



Deep learning: “Deep” architecture

Pros

- Learn features hierarchy all the way from pixels to classifier
- Each layer extracts features from the output of previous layer
- Train all layers jointly
- Simple classifier "NN" in the last layer
- Deep architecture can fit various images fast
- Higher level layers encode more abstract features
- Higher level layers show more invariance to instantiation parameters (translation, rotation,lighting changes)
- Convolutional neural network image statistics are translation invariant (objects and viewpoint translates)

Cons

- Large-scale training database
- Compute capability

Xeon(R) CPU E3-1241 v3 3.50GHz*8/16G/Quadro K2200/CUDA 7.5

MNIST Ubuntu 14.04 on CPU : 270s

MNIST Ubuntu 14.04 on GPU : 160s

MNIST Ubuntu 14.04 on GPU with cuDNN : 30s

MNIST 50 results collected

Units: error %

> Classify handwriten digits. Some additional results are available on the original dataset page.

Result	Method	Venue	Details
0.21%	Regularization of Neural Networks using DropConnect	ICML 2013	
0.23%	Multi-column Deep Neural Networks for Image Classification	CVPR 2012	
0.23%	APAC: Augmented PAttern Classification with Neural Networks	arXiv 2015	
0.24%	Batch-normalized Maxout Network in Network	arXiv 2015	Details
0.29%	Generalizing Pooling Functions in Convolutional Neural Networks: Mixed, Gated, and Tree	AISTATS 2016	Details
0.31%	Recurrent Convolutional Neural Network for Object Recognition	CVPR 2015	
0.31%	On the Importance of Normalisation Layers in Deep Learning with Piecewise Linear Activation Units	arXiv 2015	
0.32%	Fractional Max-Pooling	arXiv 2015	Details
0.33%	Competitive Multi-scale Convolution	arXiv 2015	
0.35%	Deep Big Simple Neural Nets Excel on Handwritten Digit Recognition	Neural Computation 2010	Details
0.35%	C-SVDDNet: An Effective Single-Layer Network for	arXiv 2014	

CIFAR-10 49 results collected

Units: accuracy %

> Classify 32x32 colour images.

Result	Method	Venue	Details
96.53%	Fractional Max-Pooling	arXiv 2015	Details
95.59%	Striving for Simplicity: The All Convolutional Net	ICLR 2015	Details
94.16%	All you need is a good init	ICLR 2016	Details
94%	Lessons learned from manually classifying CIFAR-10	unpublished 2011	Details
93.95%	Generalizing Pooling Functions in Convolutional Neural Networks: Mixed, Gated, and Tree	AISTATS 2016	Details
93.72%	Spatially-sparse convolutional neural networks	arXiv 2014	
93.63%	Scalable Bayesian Optimization Using Deep Neural Networks	ICML 2015	
93.57%	Deep Residual Learning for Image Recognition	arXiv 2015	Details
93.45%	Fast and Accurate Deep Network Learning by Exponential Linear Units	arXiv 2015	Details
93.34%	Universum Prescription: Regularization using Unlabeled Data	arXiv 2015	
93.25%	Batch-normalized Maxout Network in Network	arXiv 2015	Details
93.13%	Competitive Multi-scale Convolution	arXiv 2015	

CIFAR-100 31 results collected

Units: accuracy %

> Classify 32x32 colour images.

Result	Method	Venue	Details
75.72%	Fast and Accurate Deep Network Learning by Exponential Linear Units	arXiv 2015	Details
75.7%	Spatially-sparse convolutional neural networks	arXiv 2014	
73.61%	Fractional Max-Pooling	arXiv 2015	Details
72.60%	Scalable Bayesian Optimization Using Deep Neural Networks	ICML 2015	
72.44%	Competitive Multi-scale Convolution	arXiv 2015	
72.34%	All you need is a good init	ICLR 2015	Details
71.14%	Batch-normalized Maxout Network in Network	arXiv 2015	Details
70.80%	On the Importance of Normalisation Layers in Deep Learning with Piecewise Linear Activation Units	arXiv 2015	
69.17%	Learning Activation Functions to Improve Deep Neural Networks	ICLR 2015	Details
69.12%	Stacked What-Where Auto-encoders	arXiv 2015	
68.53%	Multi-Loss Regularized Deep Neural Network	CSVT 2015	Details
68.40%	Spectral Representations for Convolutional Neural Networks	NIPS 2015	

STL-10 18 results collected

Units: accuracy %

> Similar to CIFAR-10 but with 96x96 images. Original dataset website.

Result	Method	Venue	Details
74.33%	Stacked What-Where Auto-encoders	arXiv 2015	
74.10%	Convolutional Clustering for Unsupervised Learning	arXiv 2015	Details
73.15%	Deep Representation Learning with Target Coding	AAAI 2015	
72.8% (±0.4%)	Discriminative Unsupervised Feature Learning with Convolutional Neural Networks	NIPS 2014	Details
70.20 % (±0.7 %)	An Analysis of Unsupervised Pre-training in Light of Recent Advances	ICLR 2015	Details
70.1% (±0.6%)	Multi-Task Bayesian Optimization	NIPS 2013	Details
68.23% ± 0.5	C-SVDDNet: An Effective Single-Layer Network for Unsupervised Feature Learning	arXiv 2014	
68% (±0.55%)	Committees of deep feedforward networks trained with few data	arXiv 2014	
67.9% (±0.6%)	Stable and Efficient Representation Learning with Nonnegativity Constraints	ICML 2014	Details
64.5% (±1%)	Unsupervised Feature Learning for RGB-D Based Object Recognition	ISER 2012	Details
62.32%	Convolutional Kernel Networks	arXiv 2014	Details

SVHN 17 results collected

Units: error %

> The Street View House Numbers (SVHN) Dataset.
> SVHN is a real-world image dataset for developing machine learning and object recognition algorithms with minimal requirement on data preprocessing and formatting. It can be seen as similar in flavor to MNIST(e.g., the images are of small cropped digits), but incorporates an order of magnitude more labeled data (over 600,000 digit images) and comes from a significantly harder, unsolved, real world problem (recognizing digits and numbers in natural scene images). SVHN is obtained from house numbers in Google Street View images.

Result	Method	Venue	Details
1.69%	Generalizing Pooling Functions in Convolutional Neural Networks: Mixed, Gated, and Tree	AISTATS 2016	Details
1.76%	Competitive Multi-scale Convolution	arXiv 2015	
1.77%	Recurrent Convolutional Neural Network for Object Recognition	CVPR 2015	Details
1.81%	Batch-normalized Maxout Network in Network	arXiv 2015	Details
1.92%	Deeply-Supervised Nets	arXiv 2014	
1.92%	Multi-Loss Regularized Deep Neural Network	CSVT 2015	Details
1.94%	Regularization of Neural Networks using DropConnect	ICML 2013	
1.97%	On the Importance of Normalisation Layers in Deep Learning with Piecewise Linear Activation Units	arXiv 2015	
2%	Estimated human performance	NIPS 2011	Details
2.15%	BinaryConnect: Training Deep Neural Networks with binary weights during propagations	NIPS 2015	

ImageNet

- Large Scale Visual Recognition Challenge
- One of the largest and the most challenging computer vision challenge.
- 14,197,122 images
- Five hundred images per category
- Very useful resource for researchers
- http://rodrigob.github.io/are_we_there_yet/build/classification_datasets_results.html

AlexNet

- Presented by Alex Krizhevsky et al. in NIPS 2012
- Won the ILSVRC 2012 challenge
- Classify the 1.2 million high-resolution images into the 1000 different classes, and in, we achieved top-1 and top-5 error rates of 37.5% and 17.0% in the ImageNet LSVRC-2010 contest
- 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax
- "Dropout" to reduce overfitting in the fully-connected layers
- Achieved a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry in the ILSVRC-2012 competition

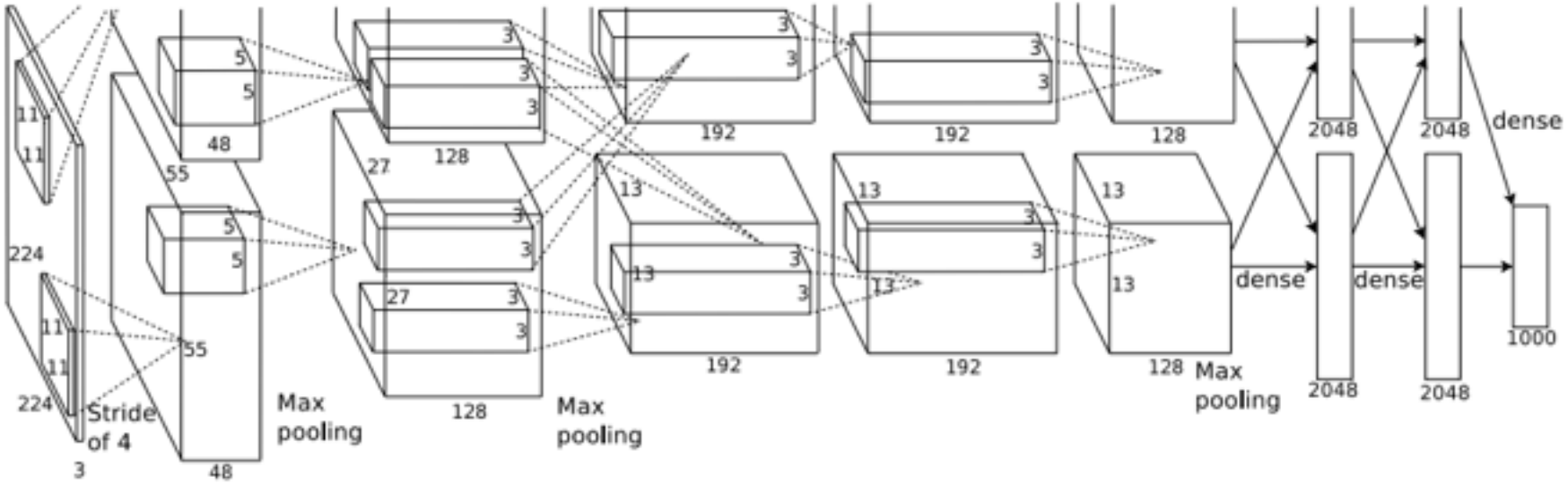


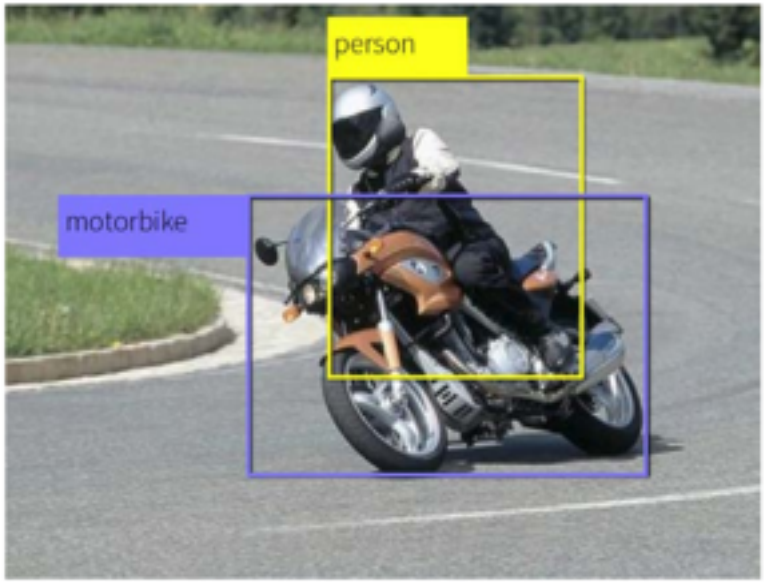
Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

GoogLenet

- The new state of the art for classification and detection in the Im
- Google team
- A 22 layers deep network
- Increasing the depth and width of the network
- keeping the computational budget constant
- "Going Deeper with Convolutions"

So What?

- Deep learning (Convolutional Neural Network) is best performing image-classification method for ImageNet
- What about Object Recognition/Detection?



Outline

- Introduction
- Images Classification
- **Object Detection**

  R-CNN

  Traditional Feature Descriptor

  Selective Search
- Implementation
- Latest Application

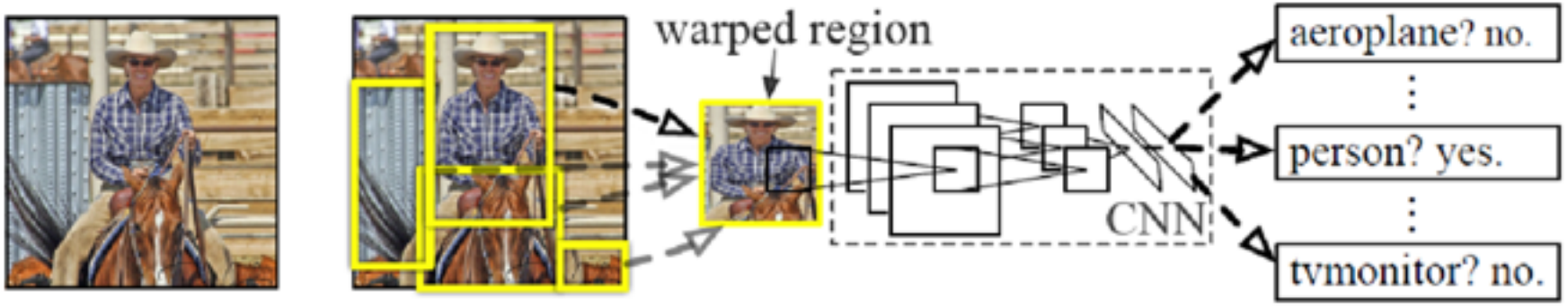
PASCAL VOC Challenge datasets

- PASCAL Visual Object Classes Challenge
- Provides standardized image data sets for object class recognition
- Enables evaluation and comparison of different methods
- Ran challenges evaluating performance on object class recognition
- 20 classes, ~10K images, ~25K annotated objects
- Evaluation: • Average Precision (AP) per class • mean Average Precision
- R-CNN
- DPM

Object Detection

- Localizing objects
- Classify objects
- Training data can be insufficient

R-CNN: Region proposals + CNN



Outline

- Introduction
- Images Classification
- Object Detection

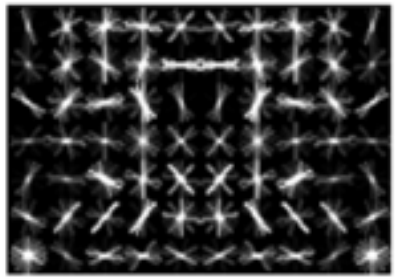
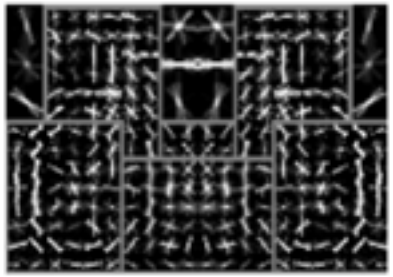
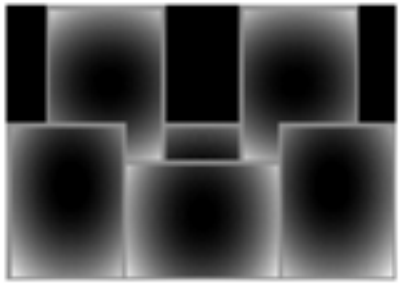
  R-CNN

  Traditional Feature Descriptor

  Selective Search
- Implementation
- Latest Application

DPM

- Deformable Parts Mode
- The winner for VOC 07,08,09
- PEDRO F. FELZENSZWALB
- Visual Object Challenge “Lifetime Achievement” Prize, 2010
- Advanced version of “HOG+SVM”

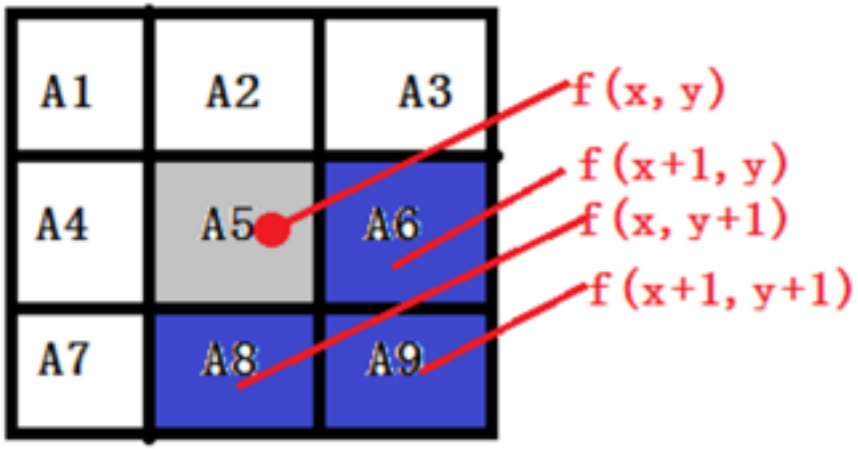




HOG

- The histogram of oriented gradients
- The technique counts occurrences of gradient orientation in localized portions of an image
- Navneet Dalal and Bill Triggs, researchers for the French National Institute for Research in Computer Science and Automation (INRIA), first described HOG descriptors at the 2005 Conference on Computer Vision and Pattern Recognition (CVPR)
- Pedestrian detection (HOG+SVM)

HOG

Gradients

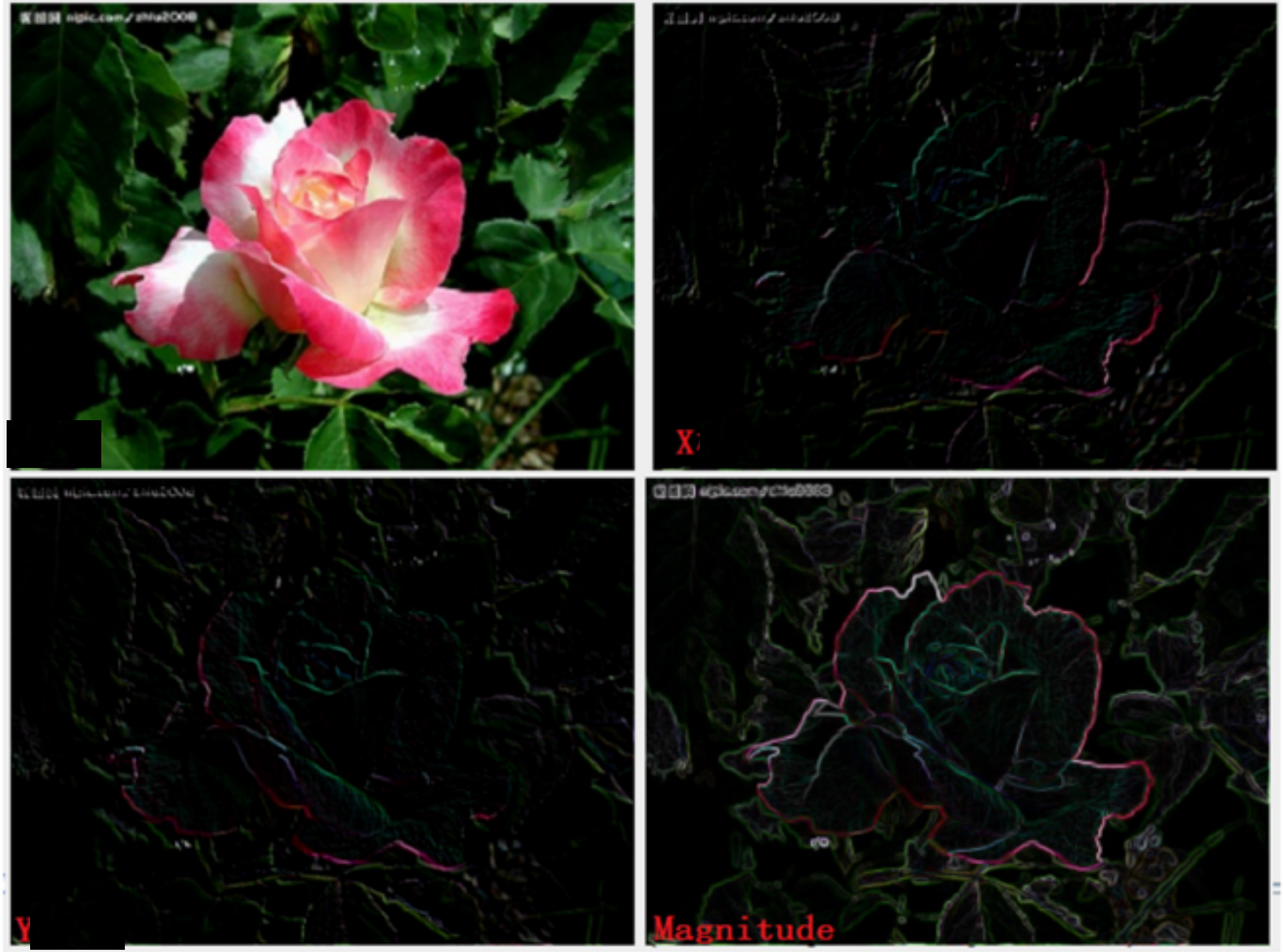


1	0
0	−1

0	1
−1	0

$$\text{Magnitude} = \sqrt{\text{xGradient} * \text{xGradient} + \text{yGradient} * \text{yGradient}}$$

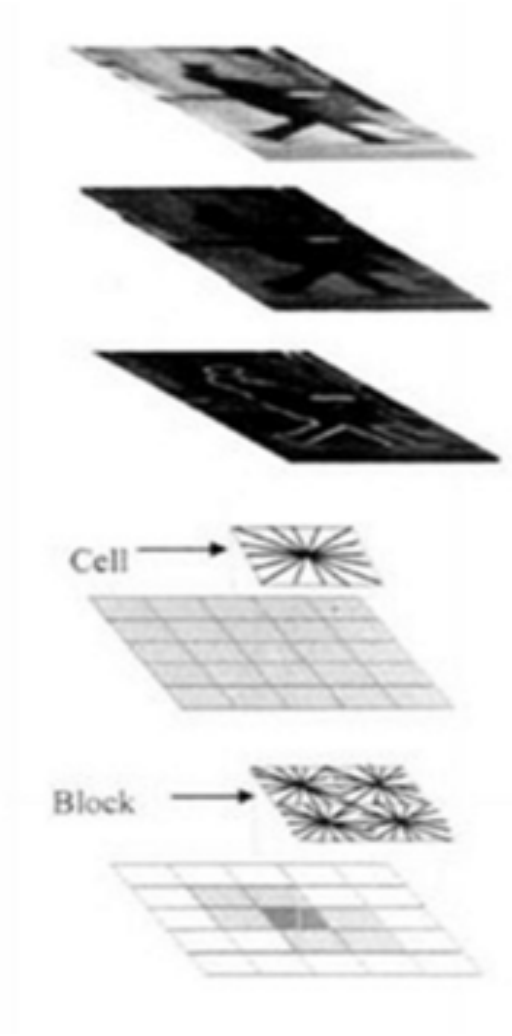
$$\text{Theta} = \tan^{-1}(\text{yGradient}/\text{xGradient})$$


X
Y
Magnitude

HOG

- The local object appearance and shape within an image can be described by the distribution of intensity gradients or edge directions.
- The image is divided into small connected regions called cells, and for the pixels within each cell, a histogram of gradient directions is compiled. The descriptor is then the concatenation of these histograms.
- For improved accuracy, the local histograms can be contrast-normalized by calculating a measure of the intensity across a larger region of the image, called a block, and then using this value to normalize all cells within the block. This normalization results in better invariance to changes in illumination and shadowing.

HOG



Input image

Normalize

Gradients

Weighted vote into orientation cells

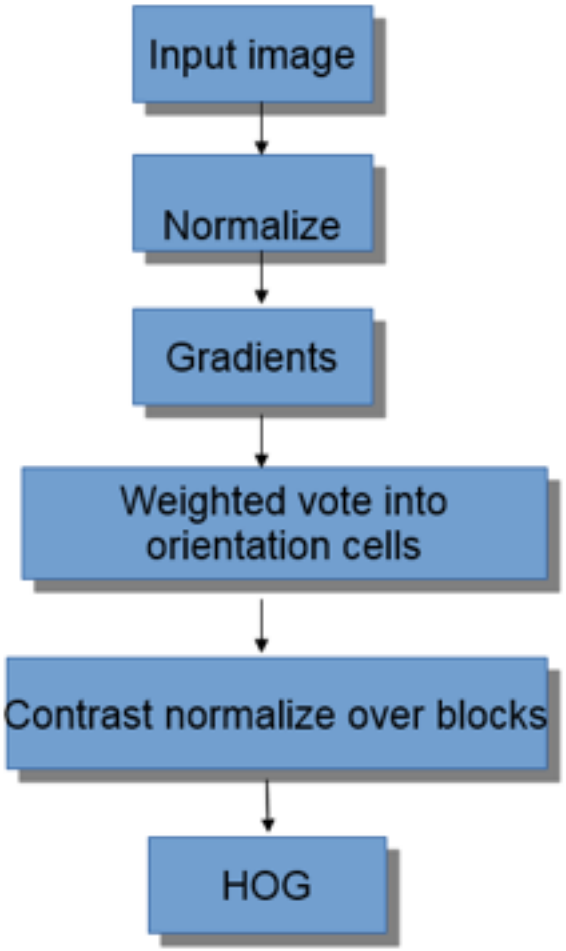
Contrast normalize over blocks

HOG

Gamma/Colour Normalization

For better invariance to illumination, shadowing,nosing, it is also useful to contrast-normalize the local responses before using them

$$I(x,y) = I(x,y)^{gamma}$$

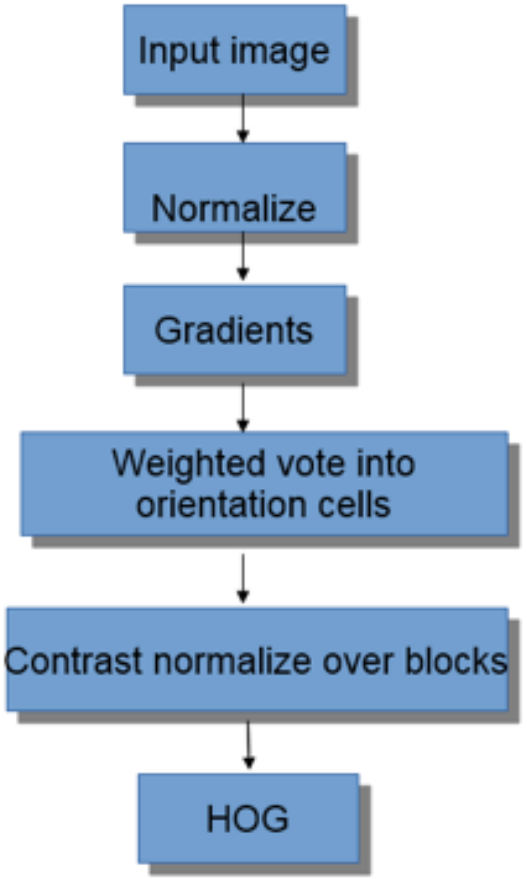


Gradient Computation

- For color images, we calculate separate gradients for each color channel, and take the one with the largest norm as the pixel's gradient vector

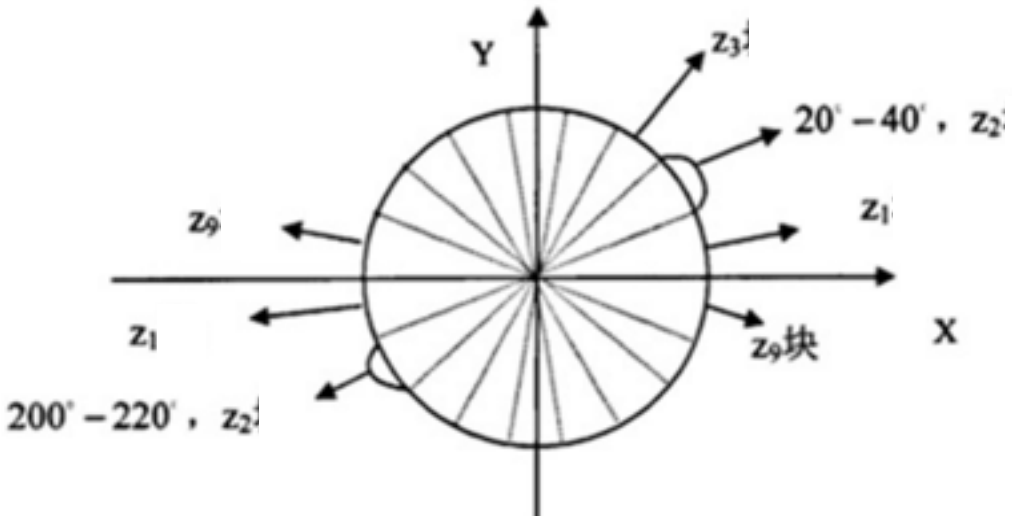
$$G_x(x,y) = H(x+1,y) - H(x-1,y)$$
$$G_y(x,y) = H(x,y+1) - H(x,y-1)$$

$$G(x,y) = \sqrt{G_x(x,y)^2 + G_y(x,y)^2}$$
$$\alpha(x,y) = \tan^{-1}\left(\frac{G_y(x,y)}{G_x(x,y)}\right)$$



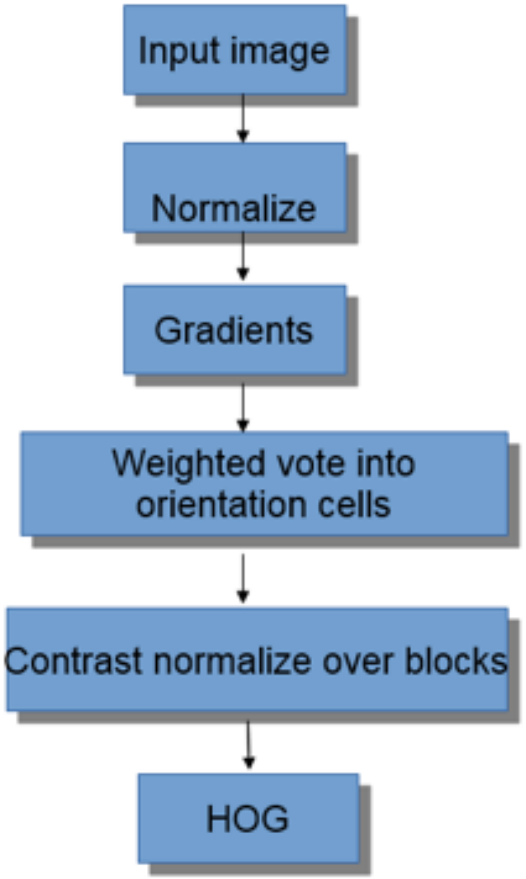
Spatial / Orientation Binning

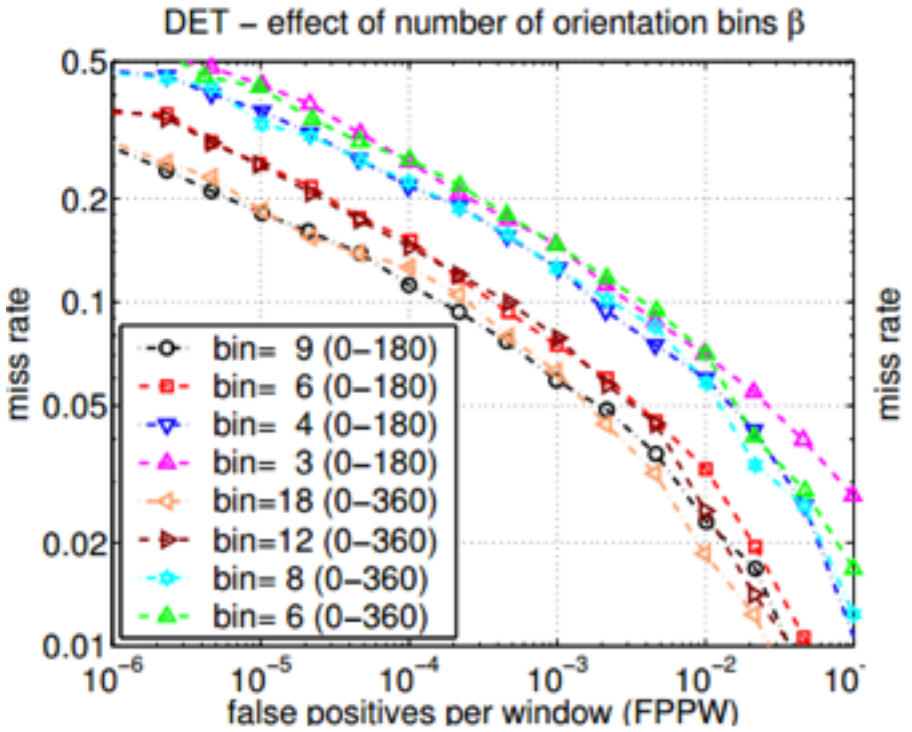
- Each pixel calculates a weighted vote for an edge orientation histogram channel based on the orientation of the gradient element centered on it, and the votes are accumulated into orientation bins over local spatial regions



Spatial / Orientation Binning

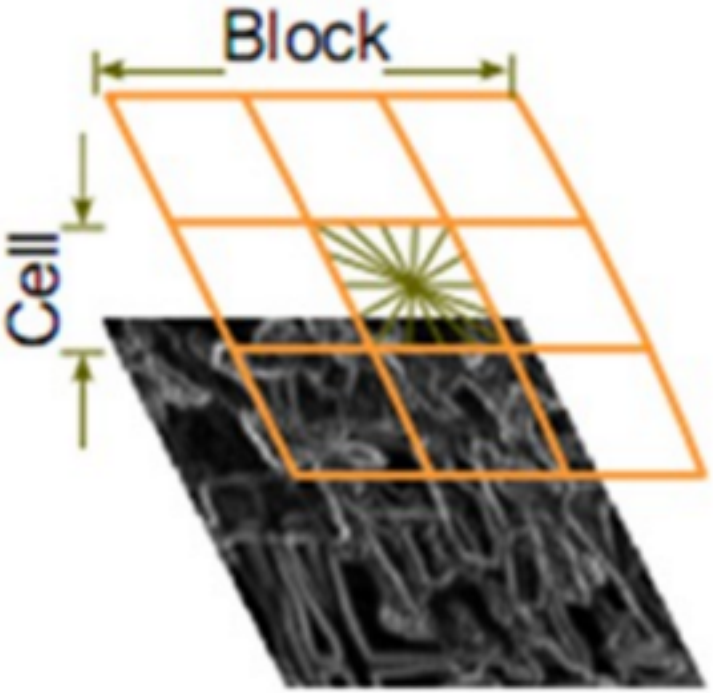
- Increasing the number of orientation bins improves performance significantly up to about 9 bins, but makes little difference beyond this
- Cells can be either rectangular or radial (log-polar sectors).

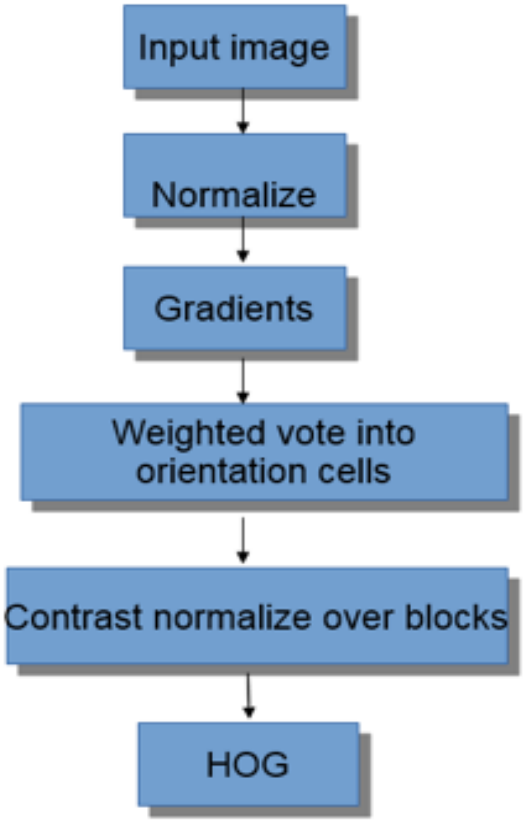




Normalization and Descriptor Blocks

- Normalize all cells across block, and then using this value to within the block
- Better invariance to changes in illumination and shadowing.
- Optimal setting for Pedestrian detection : 6*6/cell, 3*3/block, 9bins





HOG

8*8/cell

2*2/block

9 bins

Stepsize:8

Image size 64*128

HOG Vector=?-D

4*9=36/block

7 windows in x

15 windows in y

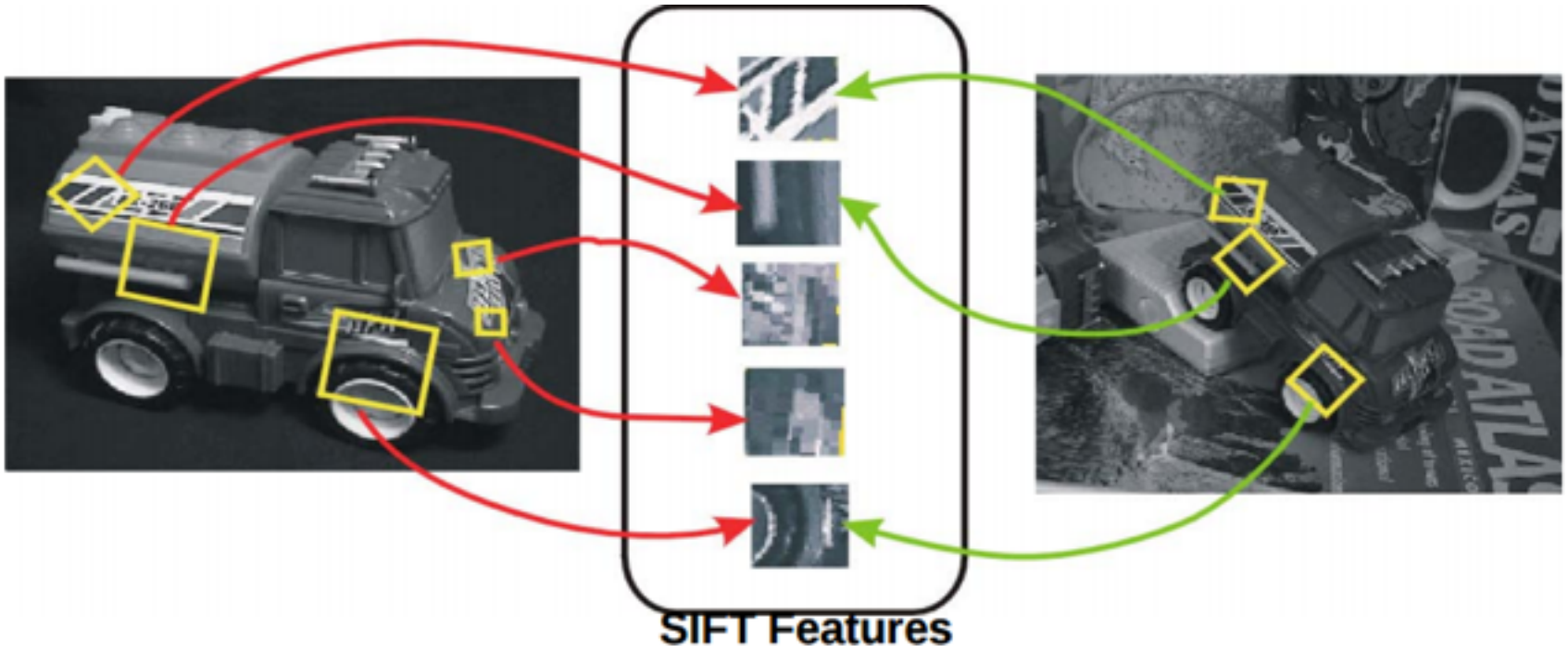
36*7*15=3780-D

SIFT

- Scale-invariant feature transform (or SIFT)
- Published by David Lowe in 1999, ICCV
- SIFT can robustly identify objects even among clutter and under partial occlusion
- Invariant to uniform scaling, orientation, and partially invariant to affine distortion and illumination changes
- Uses a much larger number of features from the images to reduces the contribution of the errors
- SIFT-like GLOH, PCA-SIFT…
- For scale changes in the range 2-2.5 and image rotations in the range 30 to 45 degrees, SIFT and SIFT-based descriptors again outperform other contemporary local descriptors with both textured and structured scene content.

Idea of SIFT

- Image content is transformed into local feature coordinates that are invariant to translation, rotation, scale, and other imaging parameters



Scales Space

- Scale-space theory is a framework for multi-scale signal representation developed by the computer vision, image processing and signal processing communities with complementary motivations from physics and biological vision.
- It is a formal theory for handling image structures at different scales, by representing an image as a one-parameter family of smoothed images, the scale-space representation, parametrized by the size of the smoothing kernel used for suppressing fine-scale structures.
- The parameter t in this family is referred to as the scale parameter, with the interpretation that image structures of spatial size smaller than about $\sqrt{t}$ have largely been smoothed away in the scale-space level at scale t.

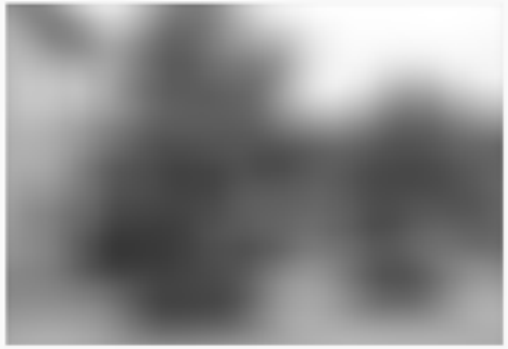
Scale-space representation $L(x, y; t)$ at scale $t = 0$, corresponding to the original image f

Scale-space representation $L(x, y; t)$ at scale $t = 1$

Scale-space representation $L(x, y; t)$ at scale $t = 4$

Scale-space representation $L(x, y; t)$ at scale $t = 16$

Scale-space representation $L(x, y; t)$ at scale $t = 64$

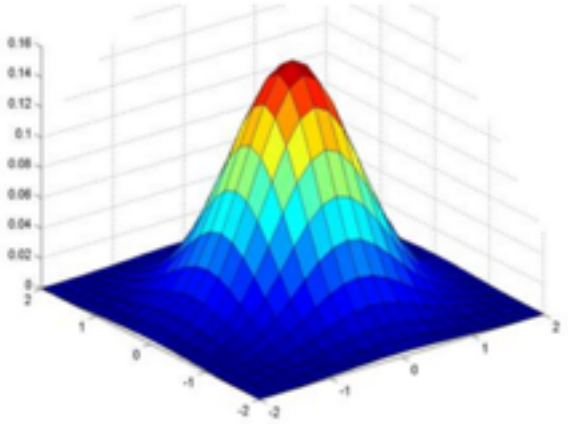


Scale-space representation $L(x, y; t)$ at scale $t = 256$

Gaussian Blur

$$G(x_i, y_i, \sigma) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(x-x_i)^2 + (y-y_i)^2}{2\sigma^2}\right)$$

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$$



Gaussian Blur

0.0000 06586	0.0004 24781	0.0133 0373	0.0004 24781	0.0000 06586
0.0004 24781	0.0273 984	0.1098 78	0.0273 984	0.0004 24781
0.0133 0373	0.1098 78	0.4406 55	0.1098 78	0.0133 0373
0.0004 24781	0.0273 984	0.1098 78	0.0273 984	0.0004 24781
0.0000 06586	0.0004 24781	0.0133 0373	0.0004 24781	0.0000 06586

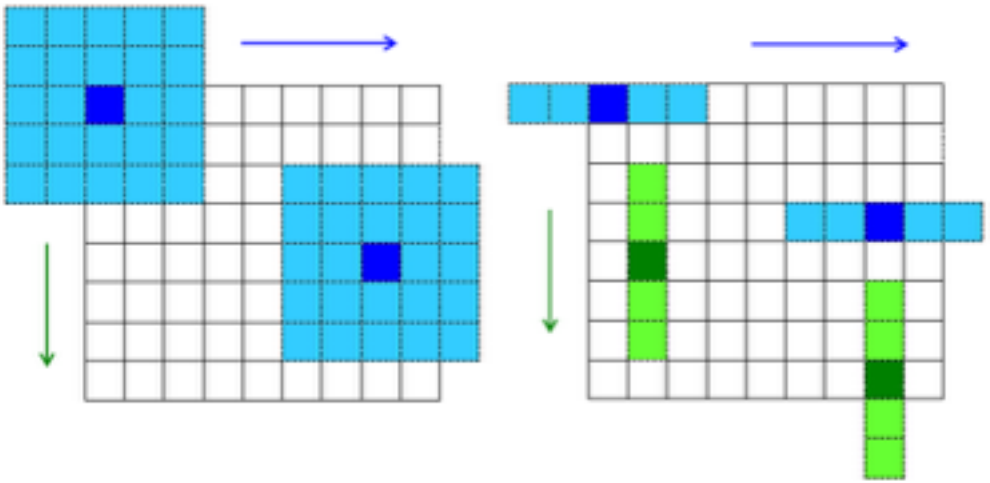
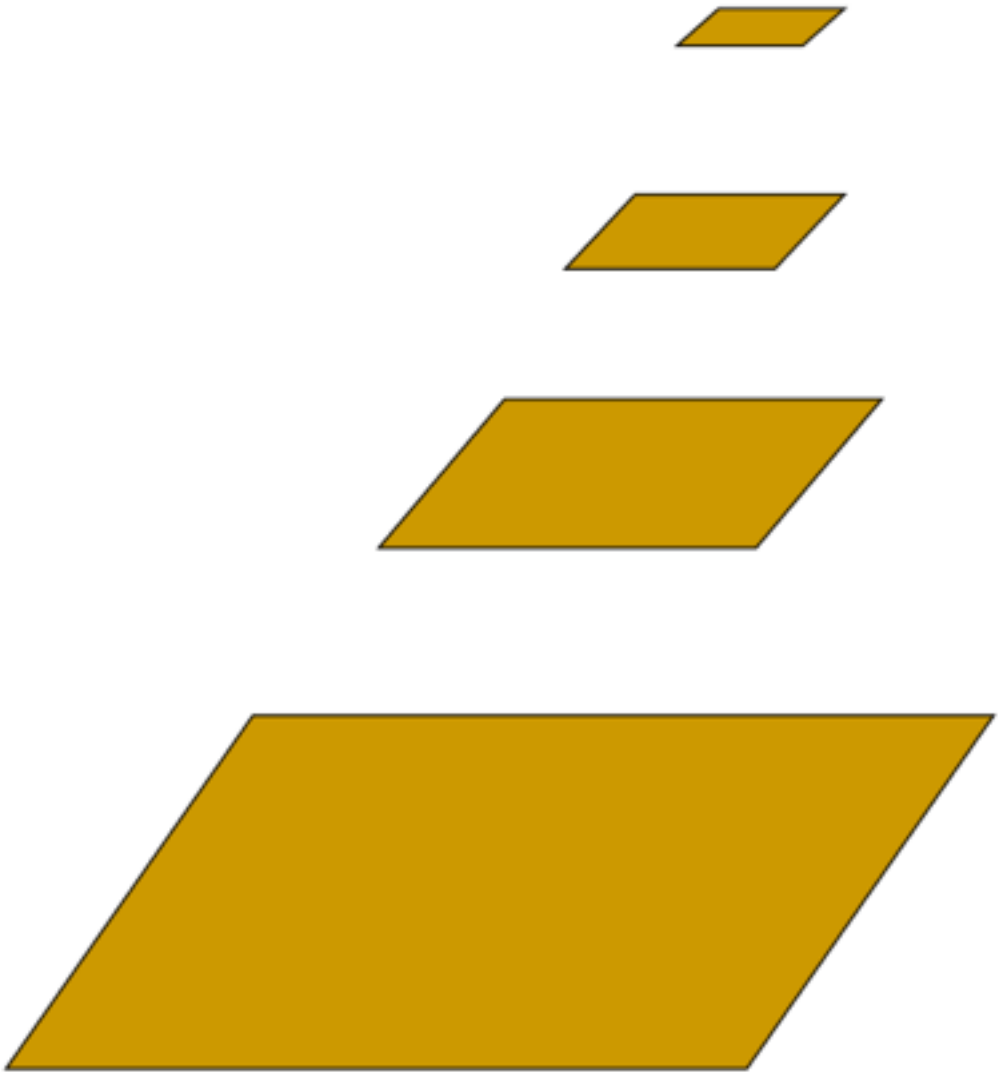


Image Pyramids



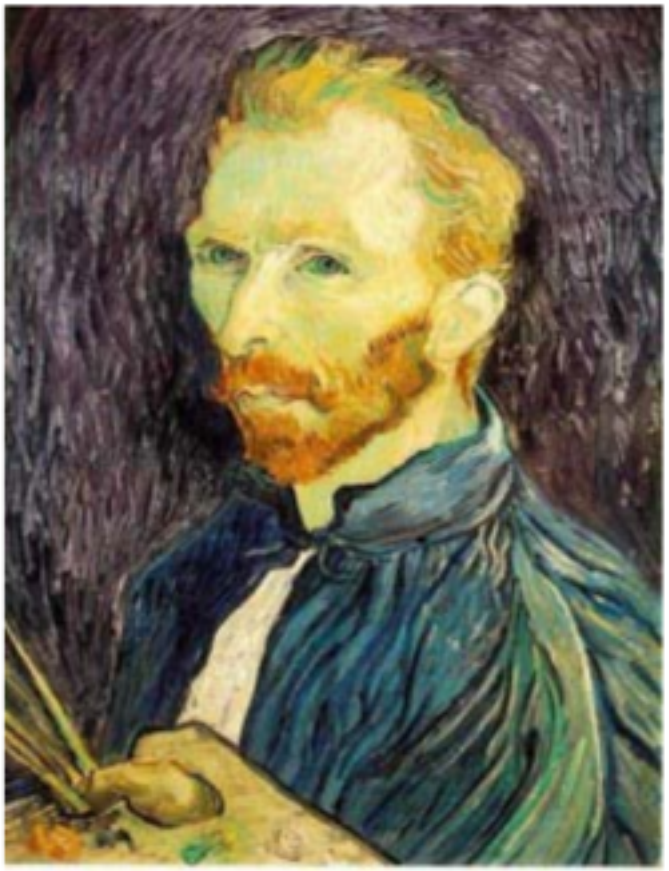
And so on.

3rd level is derived from the 2nd level according to the same funtion

2nd level is derived from the original image according to some function

Bottom level is the original image.

Example: Subsampling

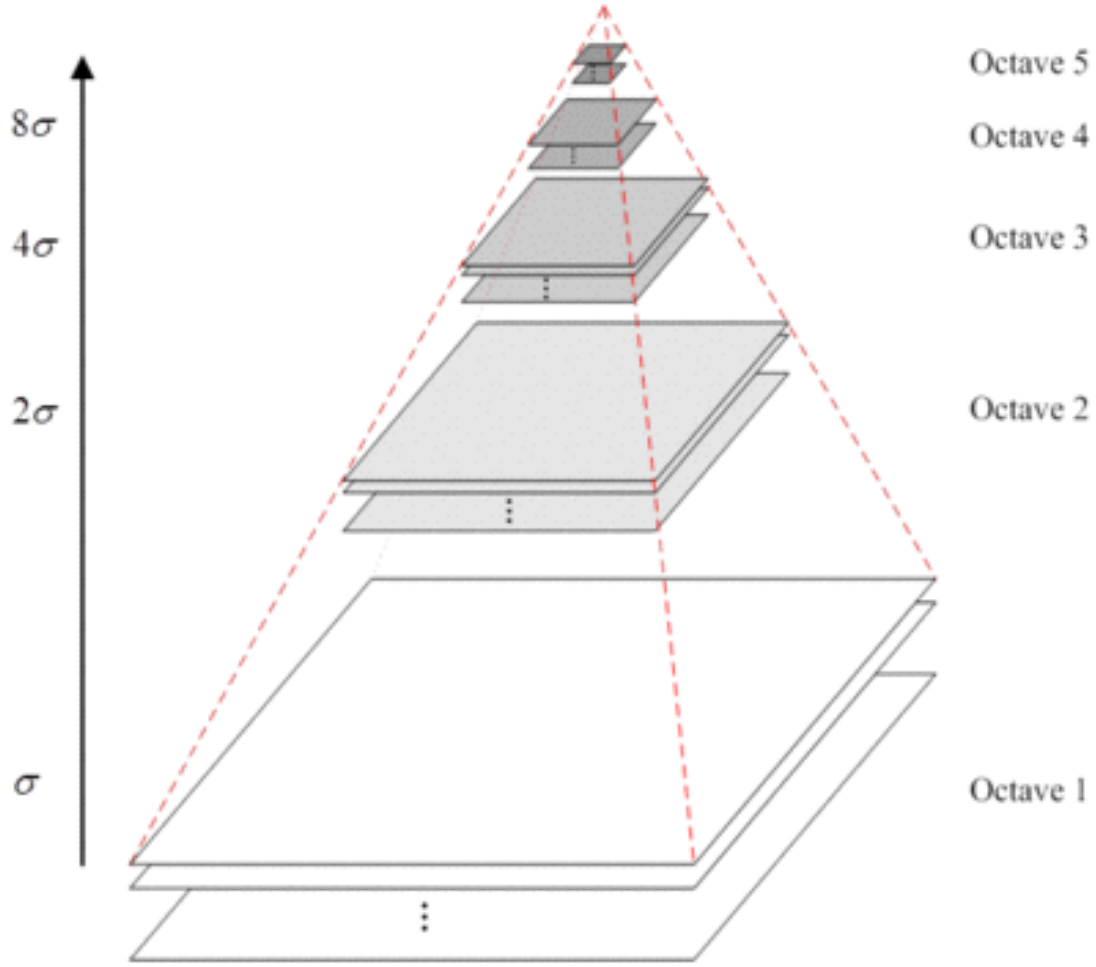


1/2

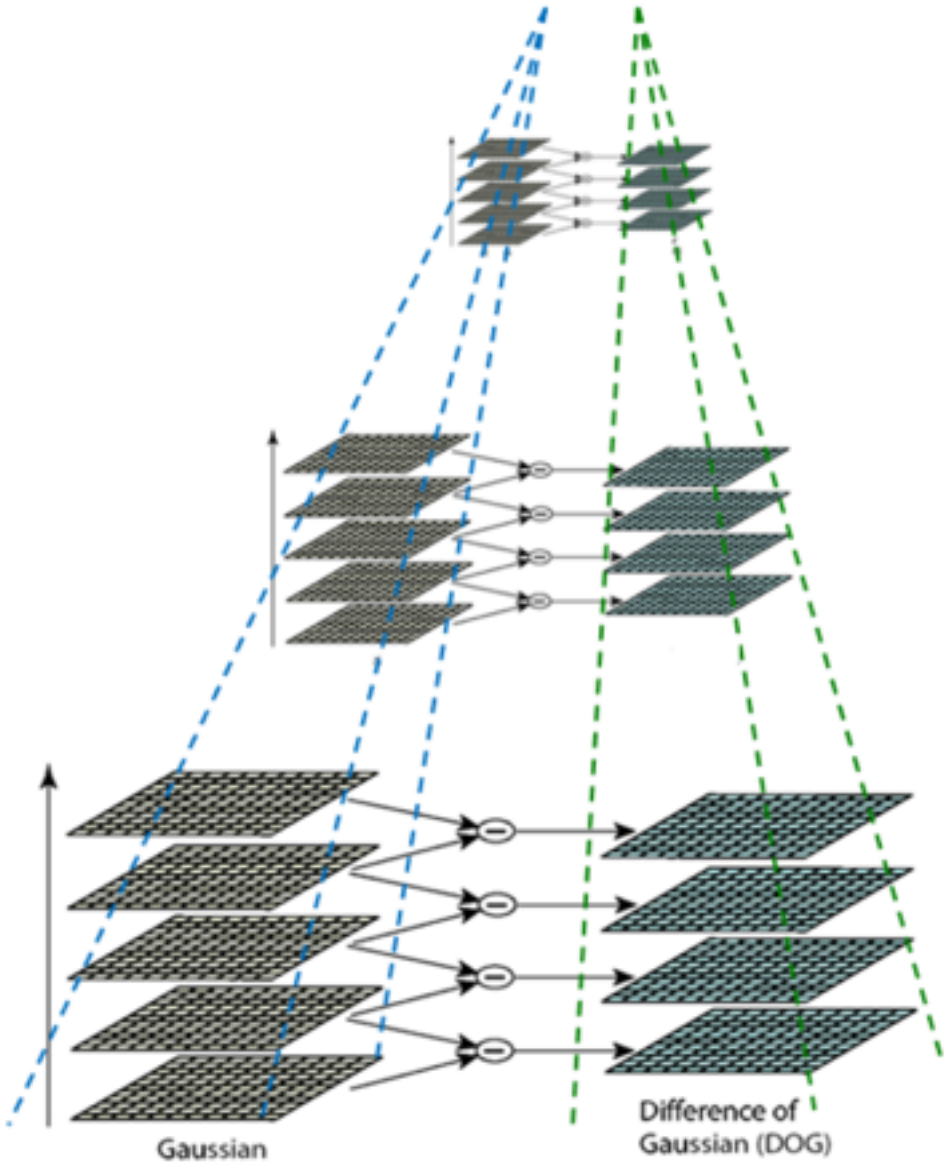
1/4

1/8

DoG（Difference of Gaussian）

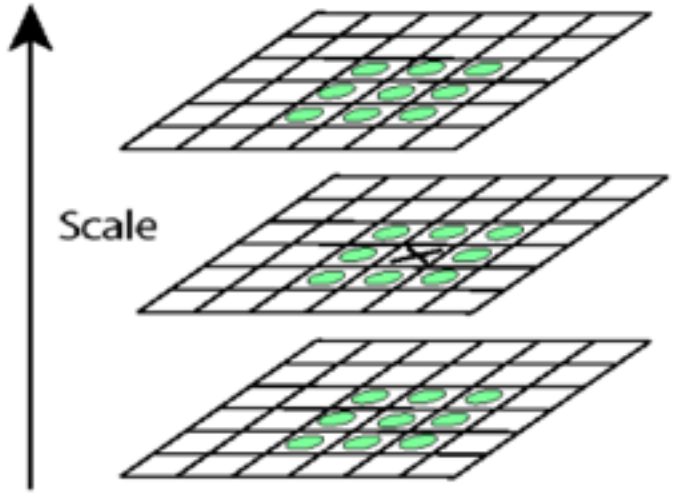


Lowe's Pyramid Scheme



Key Point Localization

- Detect maxima and minima of difference-of-Gaussian in scale space
- Each point is compared to its 8 neighbors in the current image and 9 neighbors each in the scales above and below



For each max or min found, output is the **location** and the **scale**.

Orientation Assignment

Each key point has

·Location

·Scale

·Orientation

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$

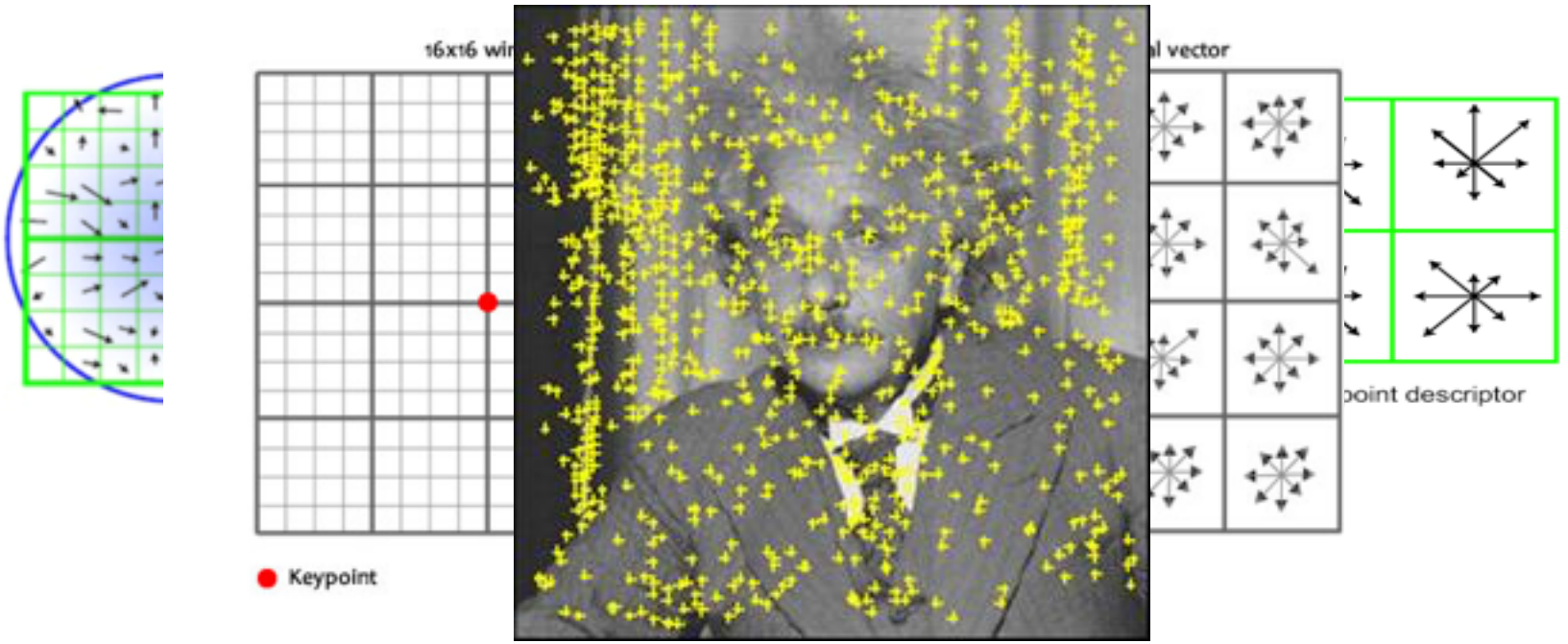
$$\theta(x,y) = \arctan\left(\frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)}\right)$$

$$L(x,y) = G(x,y,\sigma) * I(x,y)$$

Next is to compute a descriptor for the local image region about each keypoint

Keypoint Descriptors

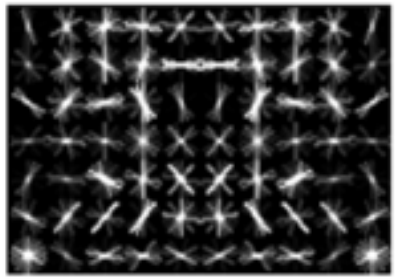
- Shown with 2 X 2 descriptors over 8 X 8
- In experiments, 4x4 arrays of 8 bin histogram is used,
- A total of 32features for one keypoint

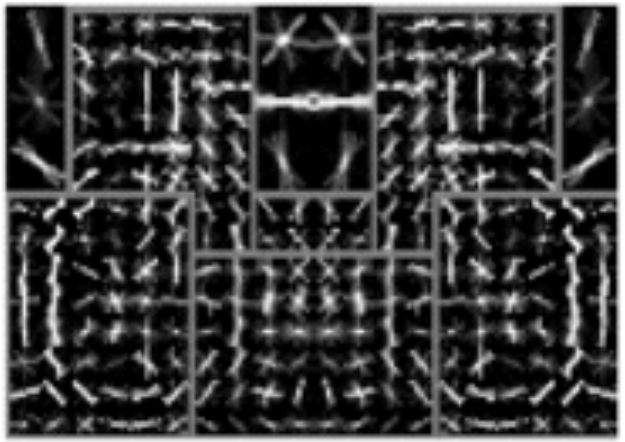


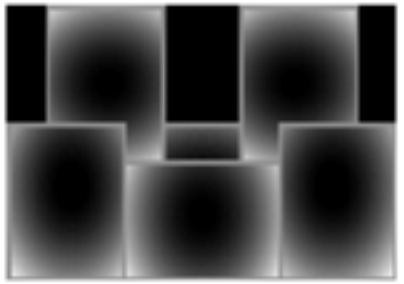

© simonho.org

DPM

- Deformable Parts Mode
- The winner for VOC 07,08,09
- PEDRO F. FELZENSZWALB
- Visual Object Challenge “Lifetime Achievement” Prize, 2010
- Advanced version of “HOG+SVM”

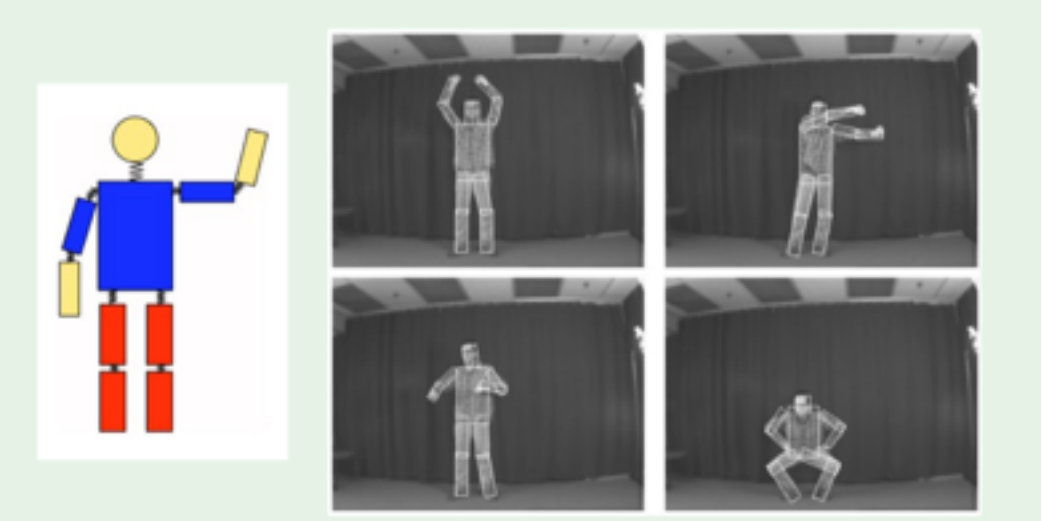






DPM

- Part-Based models
- Parts — local appearance templates
- Springs — spatial connections between parts

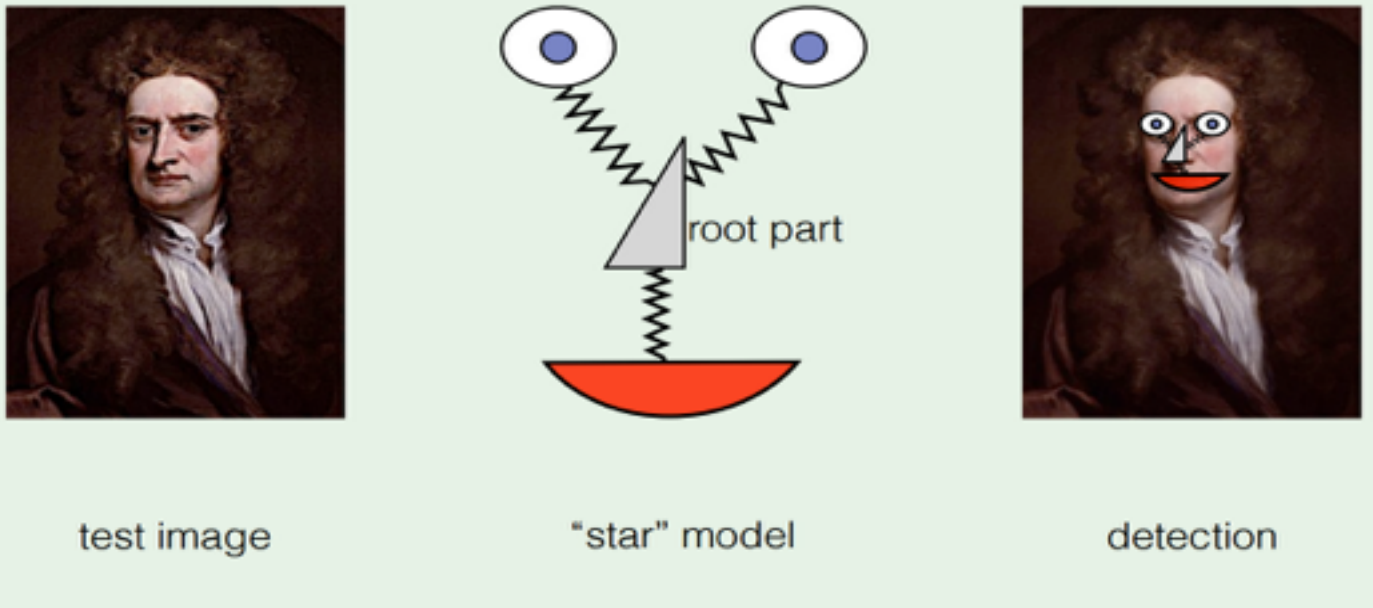


Part configuration score function

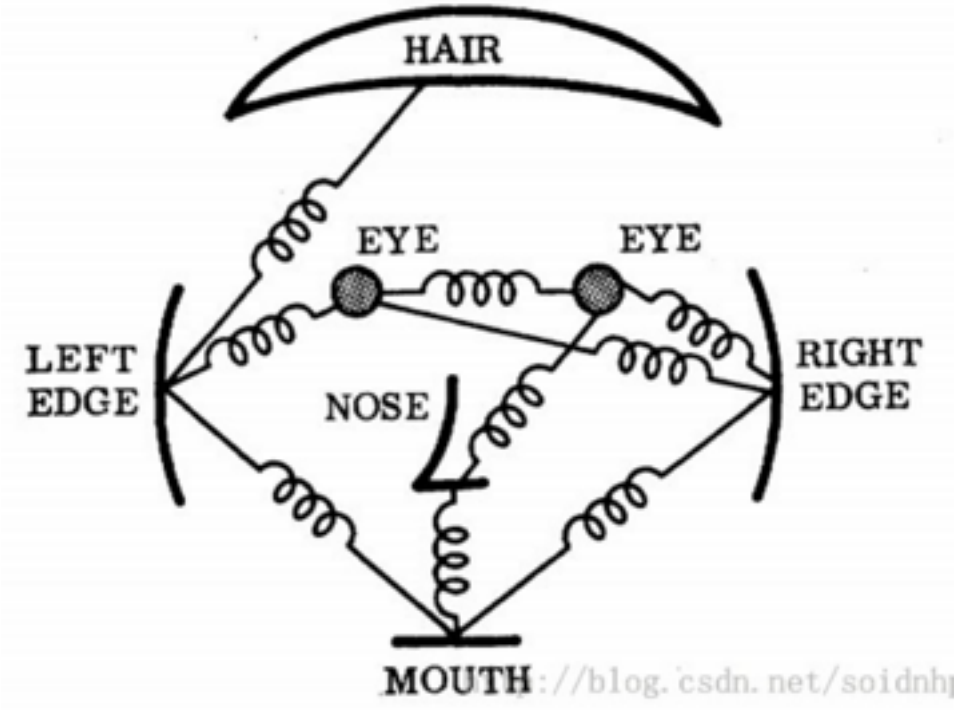
$$\text{score}(p_1, \ldots, p_n) = \sum_{i=1}^{n} m_i(p_i) - \sum_{(i,j)\in E} d_{ij}(p_i, p_j)$$

Part match scores

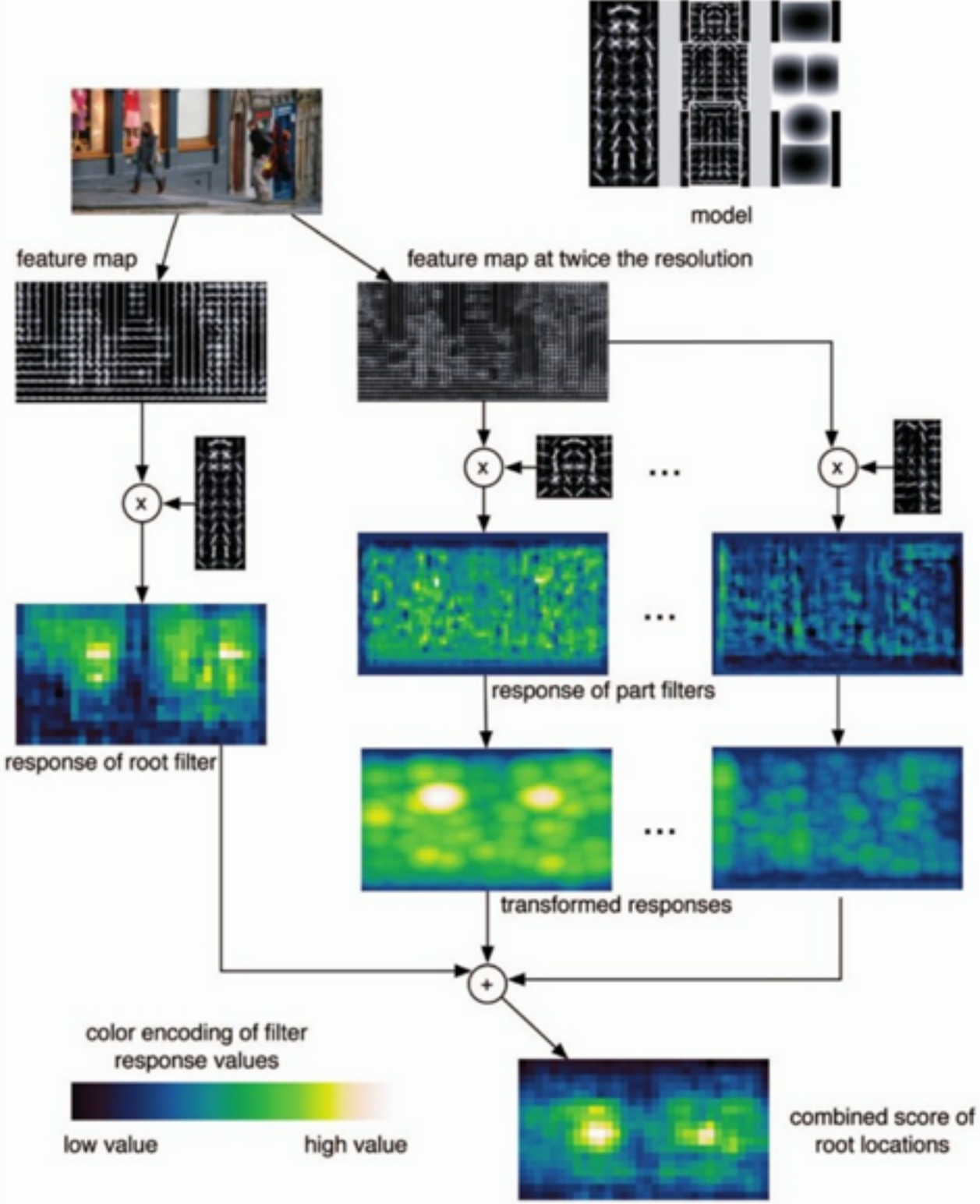
spring costs



DPM



://blog.csdn.net/soidnhj


model
feature map
feature map at twice the resolution
response of part filters
response of root filter
transformed responses
color encoding of filter
response values
low value
high value
combined score of
root locations

Object Recognition

Questions:

- What is it?
- Where is it?

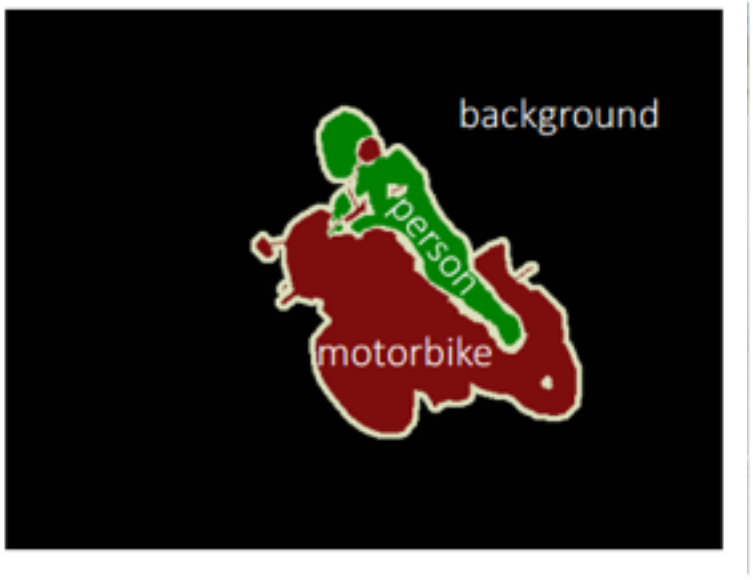
Exhaustive Search

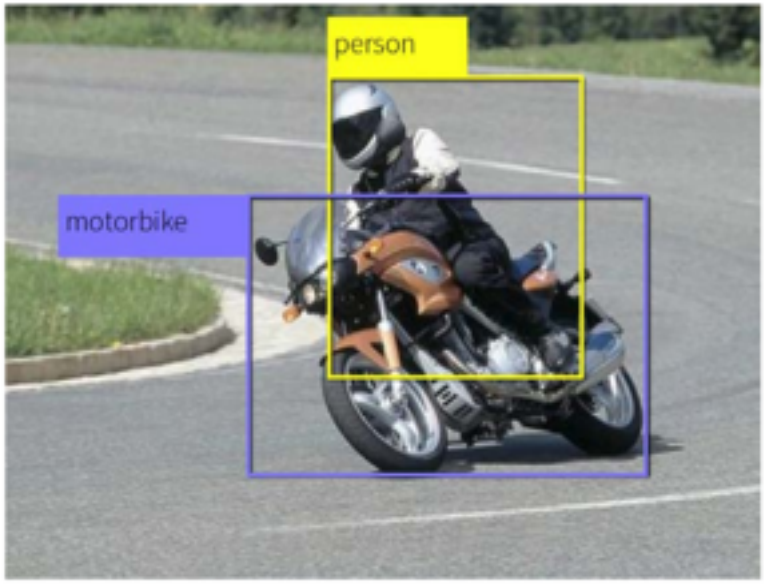
Exhaustive search:

- Windows to evaluate: 100,000 – 1,000,000
- Simple-to-compute features
- Weak classifiers

So what?

- Deep learning (Convolutional Neural Network) is best performing image-classification method for ImageNet
- What about Object Recognition/Detection?





Outline

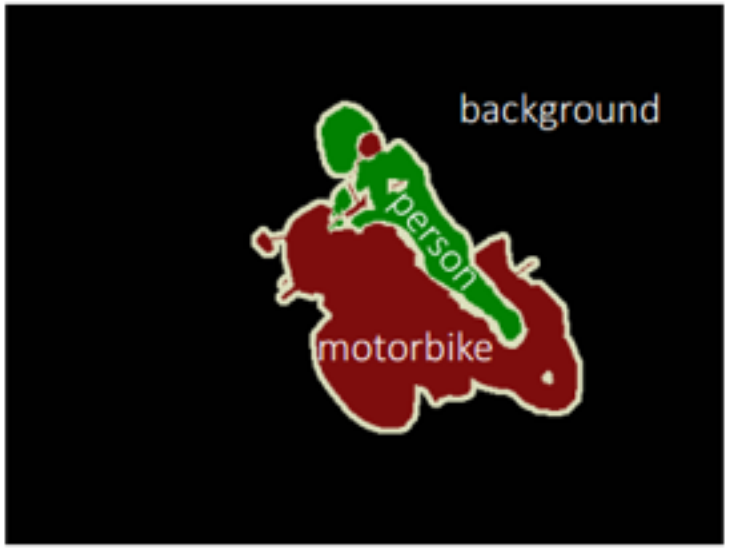
- Introduction
- Images Classification
- Object Detection

  R-CNN

  Traditional Feature Descriptor

  Selective Search
- Implementation
- Latest Application

Selective Search



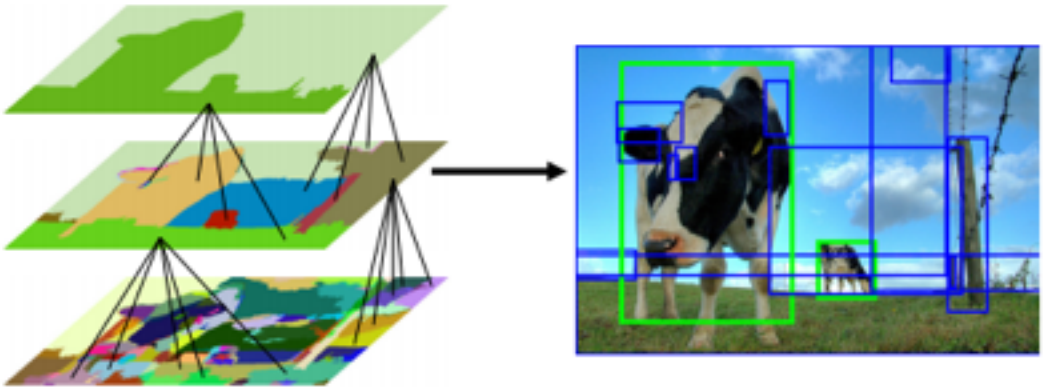
Selective Search

Goals:

1. Detect objects at any scale.
    a. Hierarchical algorithms are good at this.
2. Consider multiple grouping criteria.
    a. Detect differences in color, texture, brightness, etc.
3. Be fast.

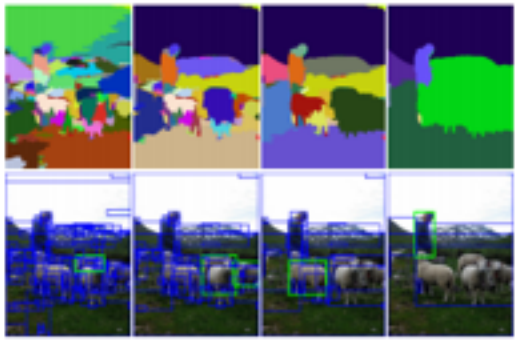
Idea: Use bottom-up grouping of image regions to generate a hierarchy of small to large regions.

Segmentation as Selective Search



Selective search based on hierarchical grouping

- Initial segments from oversegmentation [Felzenszwalb2004]
- Group adjacent regions on region-level similarity:
  - Texture (gradient orientations)
  - Region size
- Consider *all* scales of the hierarchy



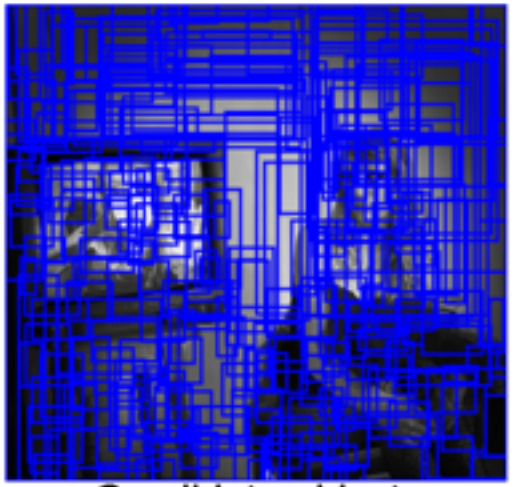
Selective Search

Step 1: Generate initial sub-segmentation

Goal: Generate many regions, each of which belongs to at most one object.

Input Image

Segmentation



Candidate objects

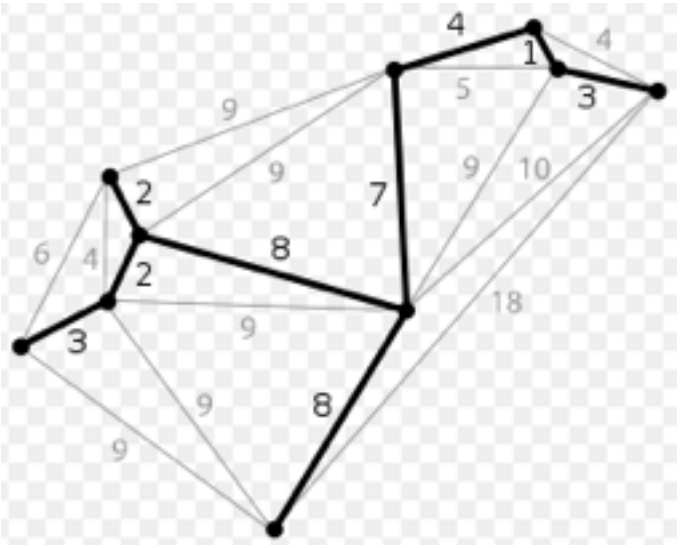
Efficient Graph-Based Image Segmentation

Published y Felzenszwalb in IJCV 2004

Image vs Graph

MST (Minimun Spanning Tree)

A spanning tree of a connected, undirected graph. It connects all the vertices together with the minimal total weighting for its edges.



Definition

We define the *internal difference* of a component $C \subseteq V$ to be the largest weight in the minimum spanning tree of the component, $MST(C, E)$. That is,

$$Int(C) = \max_{e \in MST(C,E)} w(e) \ . \tag{1}$$

One intuition underlying this measure is that a given component C only remains connected when edges of weight at least $Int(C)$ are considered.

We define the *difference between* two components $C_1, C_2 \subseteq V$ to be the minimum weight edge connecting the two components. That is,

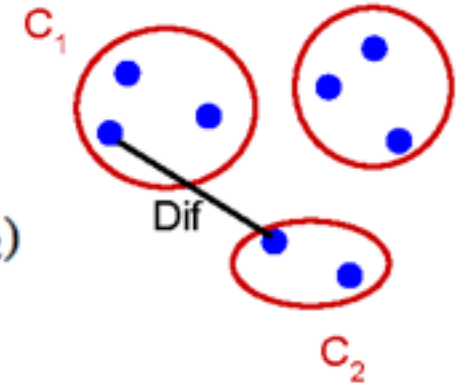
$$Dif(C_1, C_2) = \min_{v_i \in C_1, v_j \in C_2, (v_i, v_j) \in E} w((v_i, v_j)) \ . \tag{2}$$

Predicate for Segmentation

Predicate D determines whether there is a boundary for segmentation.

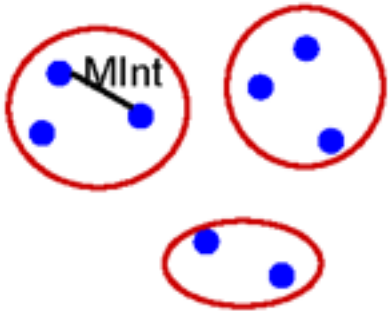
$$D(C_1, C_2) = \begin{cases} \text{true} & \text{if } Dif(C_1, C_2) > MInt(C_1, C_2) \\ \text{false} & \text{otherwise} \end{cases}$$

$$Dif(C_1, C_2) = \min_{v_i \in C_1, v_j \in C_2, (v_i, v_j) \in E} w(v_i, v_j).$$



The different between two components is the minimum weight edge that connects a node v_i in component C_1 to node v_j in C_2

Predicate for Segmentation



Predicate D determines whether there is a boundary for segmentation.

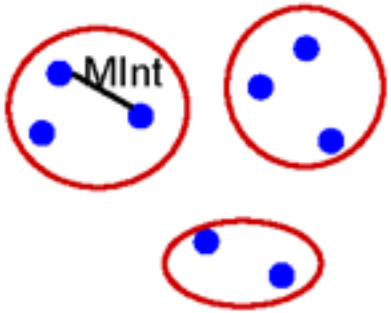
$$D(C_1, C_2) = \begin{cases} \text{true} & \text{if } Dif(C_1, C_2) > MInt(C_1, C_2) \\ \text{false} & \text{otherwise} \end{cases}$$

$$Dif(C_1, C_2) = \min_{v_i \in C_1, v_j \in C_2, (v_i, v_j) \in E} w(v_i, v_j).$$

$$Int(C) = \max_{e \in MST(C,E)} w(e).$$

Int(C) is to the maximum weight edge that connects two nodes in the same component.

Predicate for Segmentation



Predicate D determines whether there is a boundary for segmentation.

$$D(C_1, C_2) = \begin{cases} \text{true} & \text{if } Dif(C_1, C_2) > MInt(C_1, C_2) \\ \text{false} & \text{otherwise} \end{cases}$$

$$Dif(C_1, C_2) = \min_{v_i \in C_1, v_j \in C_2, (v_i, v_j) \in E} w(v_i, v_j).$$

$$Int(C) = \max_{e \in MST(C,E)} w(e).$$

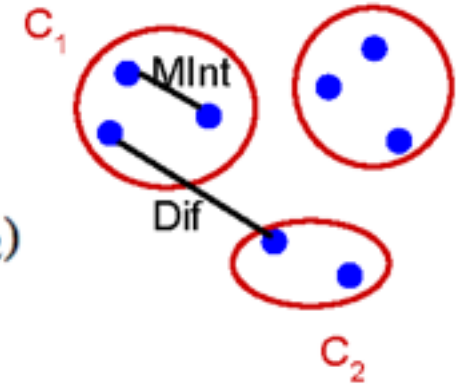
$$MInt(C_1, C_2) = \min(Int(C_1) + \tau(C_1),\ Int(C_2) + \tau(C_2)).$$

where $\tau(C) = k/|C|$

Predicate for Segmentation

Predicate D determines whether there is a boundary for segmentation.

$$D(C_1, C_2) = \begin{cases} \text{true} & \text{if } Dif(C_1, C_2) > MInt(C_1, C_2) \\ \text{false} & \text{otherwise} \end{cases}$$



Where

Dif(C_1, C_2) is the difference between two components.

MInt(C_1, C_2) is the internal different in the components C_1 and C_2

$$MInt(C_1, C_2) = \min(Int(C_1) + \tau(C_1),\ Int(C_2) + \tau(C_2)).$$

$$\tau(C) = k/|C|$$

Predicate for Segmentation

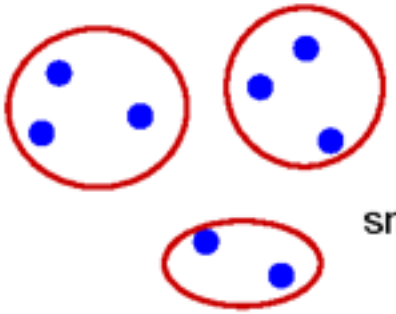
$$MInt(C_1, C_2) = \min(Int(C_1) + \tau(C_1),\ Int(C_2) + \tau(C_2)).$$

where $\tau(C) = k/|C|$

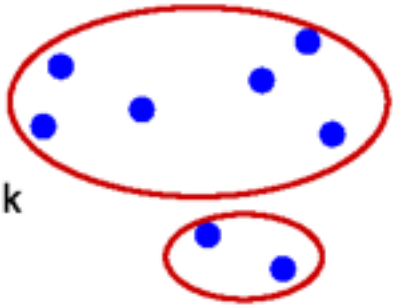
T(C) sets the threshold by which the components need to be different from the internal nodes in a component.

Properties of constant k:

- If k is large, it causes a preference of larger objects.
- k does not set a minimum size for components.



small k



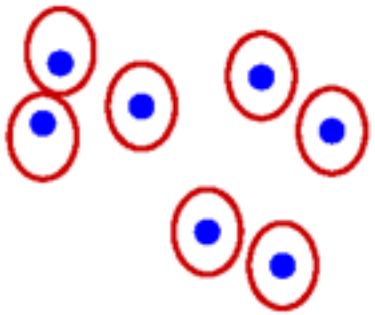
large k

Algorithm 1 *Segmentation algorithm.*

The input is a graph $G = (V, E)$, with n vertices and m edges. The output is a segmentation of V into components $S = (C_1, \ldots, C_r)$.

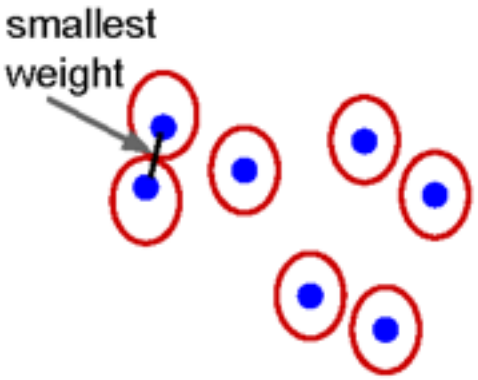
0. Sort E into $\pi = (o_1, \ldots, o_m)$, by non-decreasing edge weight.
1. Start with a segmentation S^0, where each vertex v_i is in its own component.
2. Repeat step 3 for $q = 1, \ldots, m$.
3. Construct S^q given S^{q-1} as follows. Let v_i and v_j denote the vertices connected by the q-th edge in the ordering, i.e., $o_q = (v_i, v_j)$. If v_i and v_j are in disjoint components of S^{q-1} and $w(o_q)$ is small compared to the internal difference of both those components, then merge the two components otherwise do nothing. More formally, let C_i^{q-1} be the component of S^{q-1} containing v_i and C_j^{q-1} the component containing v_j. If $C_i^{q-1} \neq C_j^{q-1}$ and $w(o_q) \leq MInt(C_i^{q-1}, C_j^{q-1})$ then S^q is obtained from S^{q-1} by merging C_i^{q-1} and C_j^{q-1}. Otherwise $S^q = S^{q-1}$.
4. Return $S = S^m$.

Algorithm



0. Sort E into $\pi = (o_1, \ldots, o_m)$, by non-decreasing edge weight.
1. Start with a segmentation S^0, where each vertex v_i is in its own component.
2. Repeat step 3 for $q = 1, \ldots, m$.
3. Construct S^q given S^{q-1} as follows. Let v_i and v_j denote the vertices connected by the q-th edge in the ordering, i.e., $o_q = (v_i, v_j)$. If v_i and v_j are in disjoint components of S^{q-1} and $w(o_q)$ is small compared to the internal difference of both those components, then merge the two components otherwise do nothing. More formally, let C_i^{q-1} be the component of S^{q-1} containing v_i and C_j^{q-1} the component containing v_j. If $C_i^{q-1} \neq C_j^{q-1}$ and $w(o_q) \leq MInt(C_i^{q-1}, C_i^{q-1})$ then S^q is obtained from S^{q-1} by merging C_i^{q-1} and C_j^{q-1}. Otherwise $S^q = S^{q-1}$.
4. Return $S = S^m$.

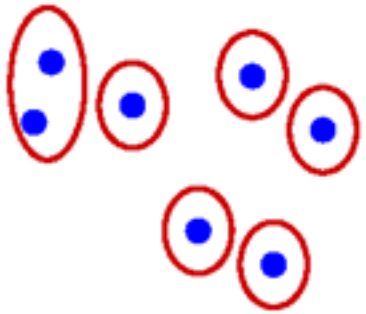
Algorithm



0. Sort E into $\pi = (o_1, \ldots, o_m)$, by non-decreasing edge weight.
1. Start with a segmentation S^0, where each vertex v_i is in its own component.
2. Repeat step 3 for $q = 1, \ldots, m$.
3. Construct S^q given S^{q-1} as follows. Let v_i and v_j denote the vertices connected by the q-th edge in the ordering, i.e., $o_q = (v_i, v_j)$. If v_i and v_j are in disjoint components of S^{q-1} and $w(o_q)$ is small compared to the internal difference of both those components, then merge the two components otherwise do nothing. More formally, let C_i^{q-1} be the component of S^{q-1} containing v_i and C_j^{q-1} the component containing v_j. If $C_i^{q-1} \neq C_j^{q-1}$ and $w(o_q) \leq MInt(C_i^{q-1}, C_i^{q-1})$ then S^q is obtained from S^{q-1} by merging C_i^{q-1} and C_j^{q-1}. Otherwise $S^q = S^{q-1}$.
4. Return $S = S^m$.

Algorithm

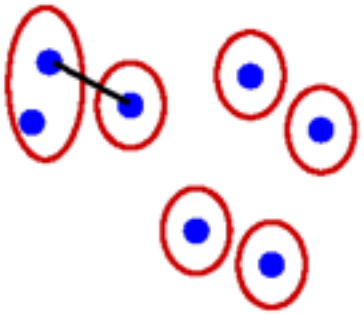
combine components



0. Sort E into $\pi = (o_1, \ldots, o_m)$, by non-decreasing edge weight.
1. Start with a segmentation S^0, where each vertex v_i is in its own component.
2. Repeat step 3 for $q = 1, \ldots, m$.
3. Construct S^q given S^{q-1} as follows. Let v_i and v_j denote the vertices connected by the q-th edge in the ordering, i.e., $o_q = (v_i, v_j)$. If v_i and v_j are in disjoint components of S^{q-1} and $w(o_q)$ is small compared to the internal difference of both those components, then merge the two components otherwise do nothing. More formally, let C_i^{q-1} be the component of S^{q-1} containing v_i and C_j^{q-1} the component containing v_j. If $C_i^{q-1} \neq C_j^{q-1}$ and $w(o_q) \leq MInt(C_i^{q-1}, C_i^{q-1})$ then S^q is obtained from S^{q-1} by merging C_i^{q-1} and C_j^{q-1}. Otherwise $S^q = S^{q-1}$.
4. Return $S = S^m$.

Algorithm

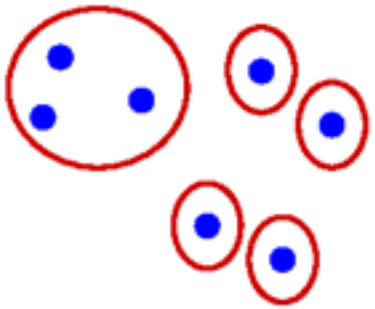
next edge



0. Sort E into $\pi = (o_1, \ldots, o_m)$, by non-decreasing edge weight.
1. Start with a segmentation S^0, where each vertex v_i is in its own component.
2. Repeat step 3 for $q = 1, \ldots, m$.
3. Construct S^q given S^{q-1} as follows. Let v_i and v_j denote the vertices connected by the q-th edge in the ordering, i.e., $o_q = (v_i, v_j)$. If v_i and v_j are in disjoint components of S^{q-1} and $w(o_q)$ is small compared to the internal difference of both those components, then merge the two components otherwise do nothing. More formally, let C_i^{q-1} be the component of S^{q-1} containing v_i and C_j^{q-1} the component containing v_j. If $C_i^{q-1} \neq C_j^{q-1}$ and $w(o_q) \leq MInt(C_i^{q-1}, C_i^{q-1})$ then S^q is obtained from S^{q-1} by merging C_i^{q-1} and C_j^{q-1}. Otherwise $S^q = S^{q-1}$.
4. Return $S = S^m$.

Algorithm

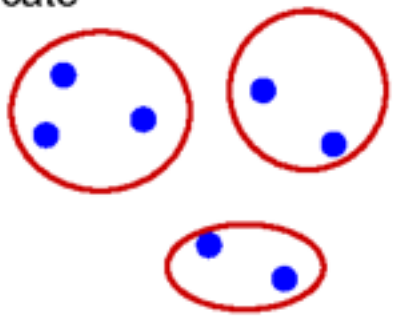
combine components



0. Sort E into $\pi = (o_1, \ldots, o_m)$, by non-decreasing edge weight.
1. Start with a segmentation S^0, where each vertex v_i is in its own component.
2. Repeat step 3 for $q = 1, \ldots, m$.
3. Construct S^q given S^{q-1} as follows. Let v_i and v_j denote the vertices connected by the q-th edge in the ordering, i.e., $o_q = (v_i, v_j)$. If v_i and v_j are in disjoint components of S^{q-1} and $w(o_q)$ is small compared to the internal difference of both those components, then merge the two components otherwise do nothing. More formally, let C_i^{q-1} be the component of S^{q-1} containing v_i and C_j^{q-1} the component containing v_j. If $C_i^{q-1} \neq C_j^{q-1}$ and $w(o_q) \leq MInt(C_i^{q-1}, C_i^{q-1})$ then S^q is obtained from S^{q-1} by merging C_i^{q-1} and C_j^{q-1}. Otherwise $S^q = S^{q-1}$.
4. Return $S = S^m$.

Algorithm

no more edges that satisfy the predicate



0. Sort E into $\pi = (o_1, \ldots, o_m)$, by non-decreasing edge weight.
1. Start with a segmentation S^0, where each vertex v_i is in its own component.
2. Repeat step 3 for $q = 1, \ldots, m$.
3. Construct S^q given S^{q-1} as follows. Let v_i and v_j denote the vertices connected by the q-th edge in the ordering, i.e., $o_q = (v_i, v_j)$. If v_i and v_j are in disjoint components of S^{q-1} and $w(o_q)$ is small compared to the internal difference of both those components, then merge the two components otherwise do nothing. More formally, let C_i^{q-1} be the component of S^{q-1} containing v_i and C_j^{q-1} the component containing v_j. If $C_i^{q-1} \neq C_j^{q-1}$ and $w(o_q) \leq MInt(C_i^{q-1}, C_i^{q-1})$ then S^q is obtained from S^{q-1} by merging C_i^{q-1} and C_j^{q-1}. Otherwise $S^q = S^{q-1}$.
4. Return $S = S^m$.

Selective Search

Step 2: Recursively combine similar regions into larger ones.

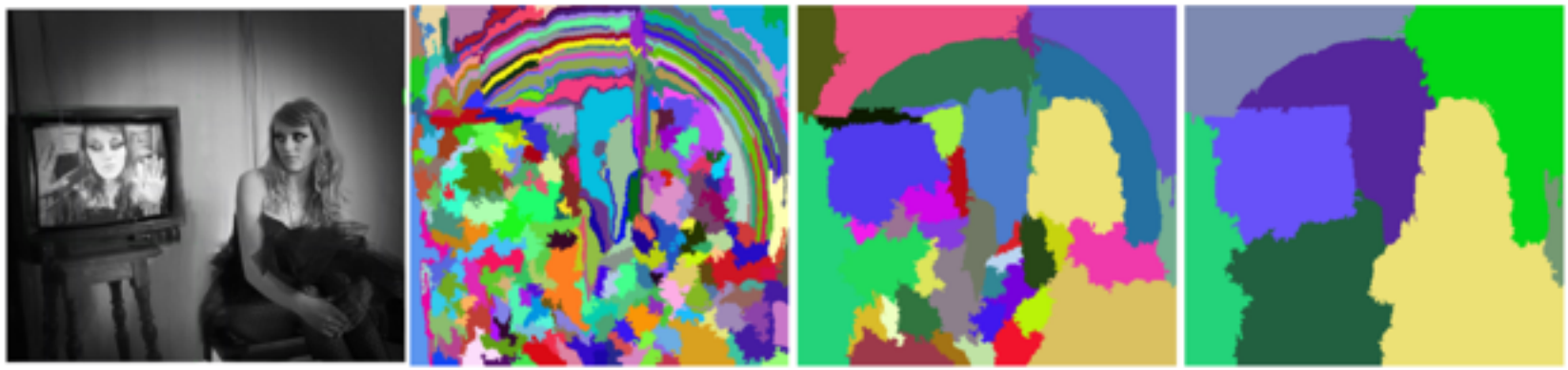
Greedy algorithm:

1. From set of regions, choose two that are most similar.
2. Combine them into a single, larger region.
3. Repeat until only one region remains.

This yields a hierarchy of successively larger regions, just like we want.

Selective Search

Step 2: Recursively combine similar regions into larger ones.



Input Image | Initial Segmentation | After some iterations | After more iterations

(a)

(b)

(c)

(d)

Algorithm 1: Hierarchical Grouping Algorithm

Input: (colour) image
Output: Set of object location hypotheses L

Obtain initial regions $R = \{r_1, \cdots, r_n\}$ using [13]
Initialise similarity set $S = \emptyset$
foreach *Neighbouring region pair* (r_i, r_j) **do**
    Calculate similarity $s(r_i, r_j)$
    $S = S \cup s(r_i, r_j)$

while $S \neq \emptyset$ **do**
    Get highest similarity $s(r_i, r_j) = \max(S)$
    Merge corresponding regions $r_t = r_i \cup r_j$
    Remove similarities regarding $r_i : S = S \setminus s(r_i, r_*)$
    Remove similarities regarding $r_j : S = S \setminus s(r_*, r_j)$
    Calculate similarity set S_t between r_t and its neighbours
    $S = S \cup S_t$
    $R = R \cup r_t$

Extract object location boxes L from all regions in R

Similarity

What do we mean by "**similarity**"?

Two-pronged approach:

1. Choose a **color space** that captures interesting things.
   a. Different color spaces have different invariants, and different responses to changes in color.
2. Choose a **similarity metric** for that space that captures everything we're interested: color, texture, size, and shape.

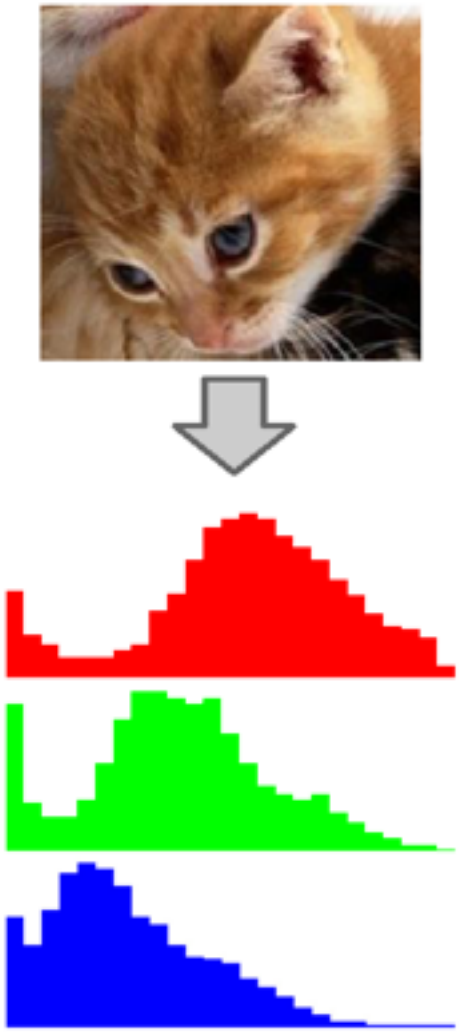
Similarity

Similarity Measures: Color Similarity

Create a color histogram *C* for each channel in region *r*.
In the paper, 25 bins were used, for 75 total dimensions.

We can measure similarity with histogram intersection:

$$s_{colour}(r_i, r_j) = \sum_{k=1}^{n} \min(c_i^k, c_j^k)$$



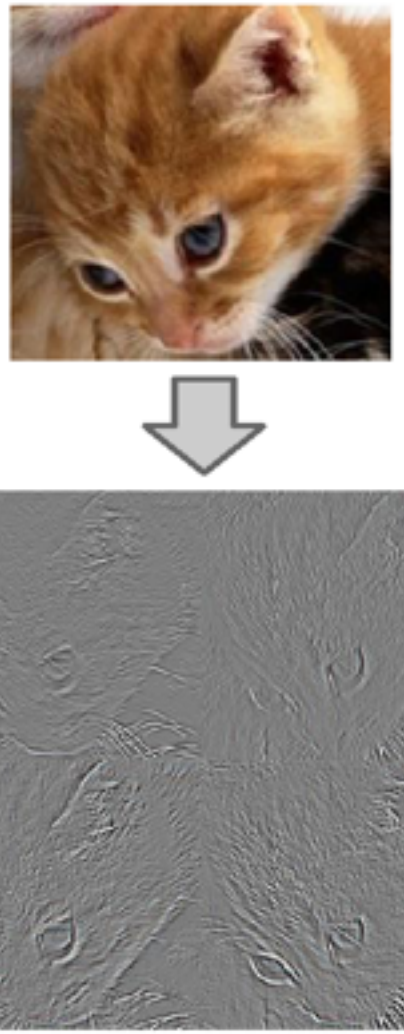
Similarity

Similarity Measures: Texture Similarity

Can measure textures with a HOG-like feature:

1. Extract gaussian derivatives of the image in 8 directions and for each channel.
2. Construct a 10-bin histogram for each, resulting in a 240-dimensional descriptor.

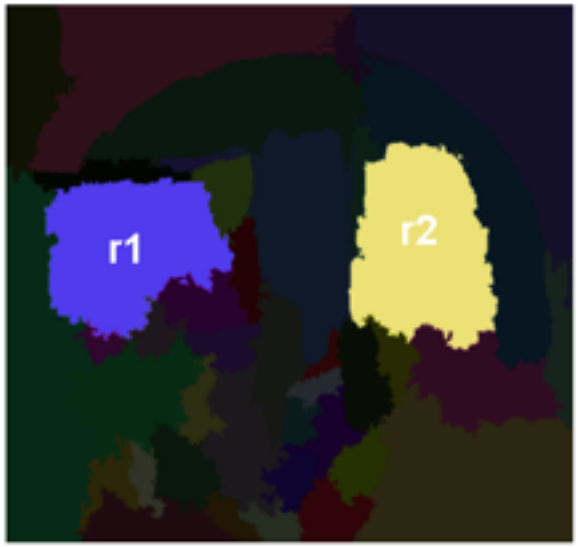
$$s_{texture}(r_i, r_j) = \sum_{k=1}^{n} \min(t_i^k, t_j^k).$$



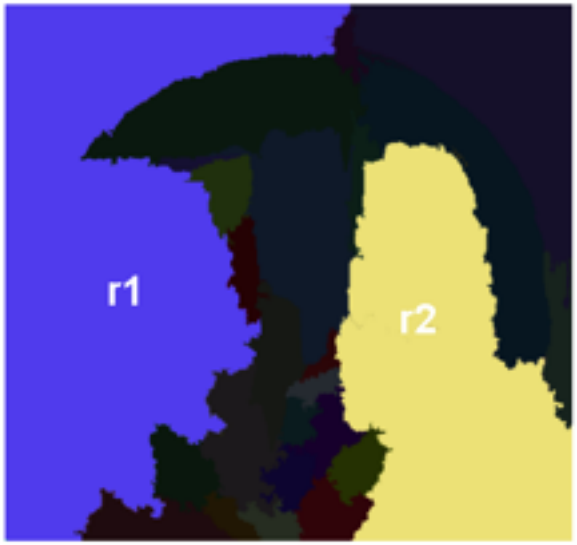
Similarity

Similarity Measures: Size Similarity

We want small regions to merge into larger ones, to create a balanced hierarchy.



Solution: Add a size component to our similarity metric, that ensures small regions are more similar to each other.



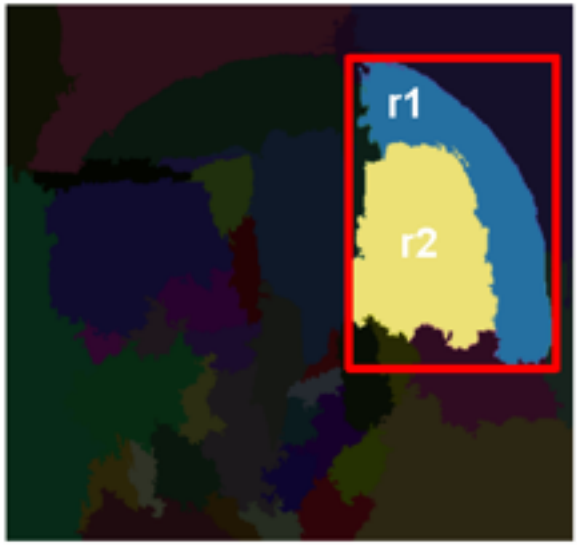
$$s_{size}(r_i, r_j) = 1 - \frac{\text{size}(r_i) + \text{size}(\text{r}_\text{j})}{\text{size}(im)}$$

Similarity

Similarity Measures: Shape Compatibility

We also want our merged regions to be cohesive, so we can add a measure of how well two regions "fit together".

$$fill(r_i, r_j) = 1 - \frac{\text{size}(BB_{ij}) - \text{size}(r_i) - \text{size}(r_i)}{\text{size}(im)}$$



Similarity

Final similarity metric:

We measure the similarity between two patches as a **linear combination** of the four given metrics:

$$\begin{aligned} s(r_i, r_j) &= a_1 s_{colour}(r_i, r_j) + a_2 s_{texture}(r_i, r_j) + \\ & \quad a_3 s_{size}(r_i, r_j) + a_4 s_{fill}(r_i, r_j), \end{aligned}$$

Then, we can create a diverse collection of region-merging strategies by considering different weighted combinations in different color spaces.

where $a_i \in \{0,1\}$ denotes if the similarity measure is used or not

Selective Search

Step 3: Use the generated regions to produce candidate object locations.

Input Image

Outline

- Introduction
- Images Classification
- Object Detection

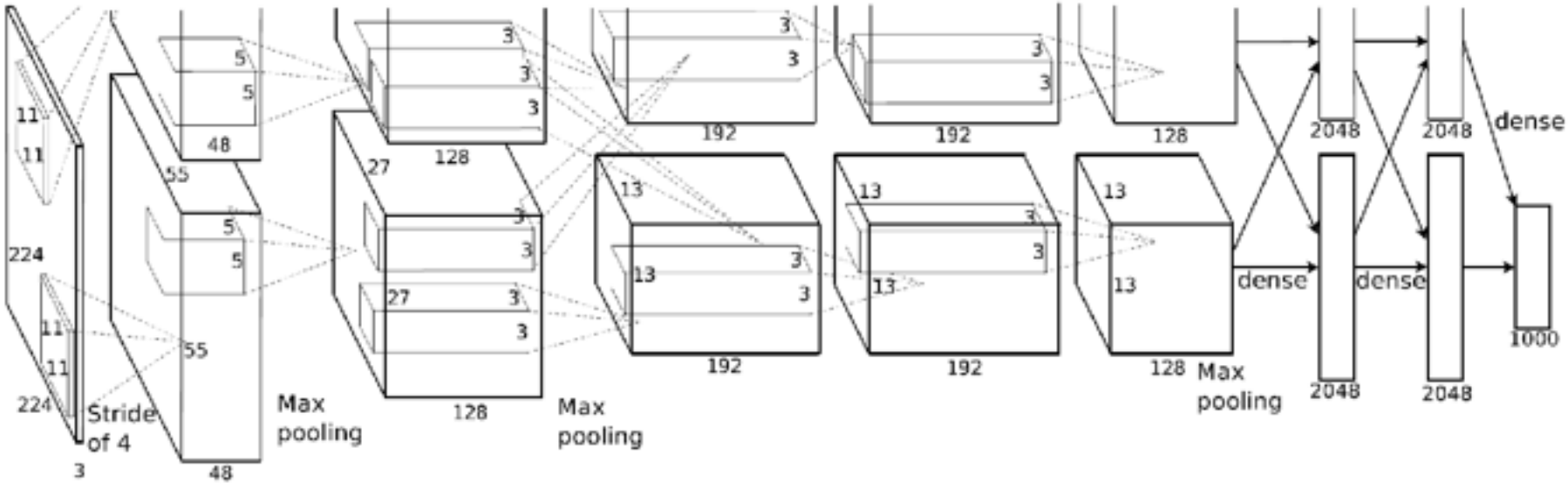
  R-CNN

  Traditional Feature Descriptor

  Selective Search
- Implementation
- Latest Application

Deep Learning is back!

UToronto "SuperVision" CNN

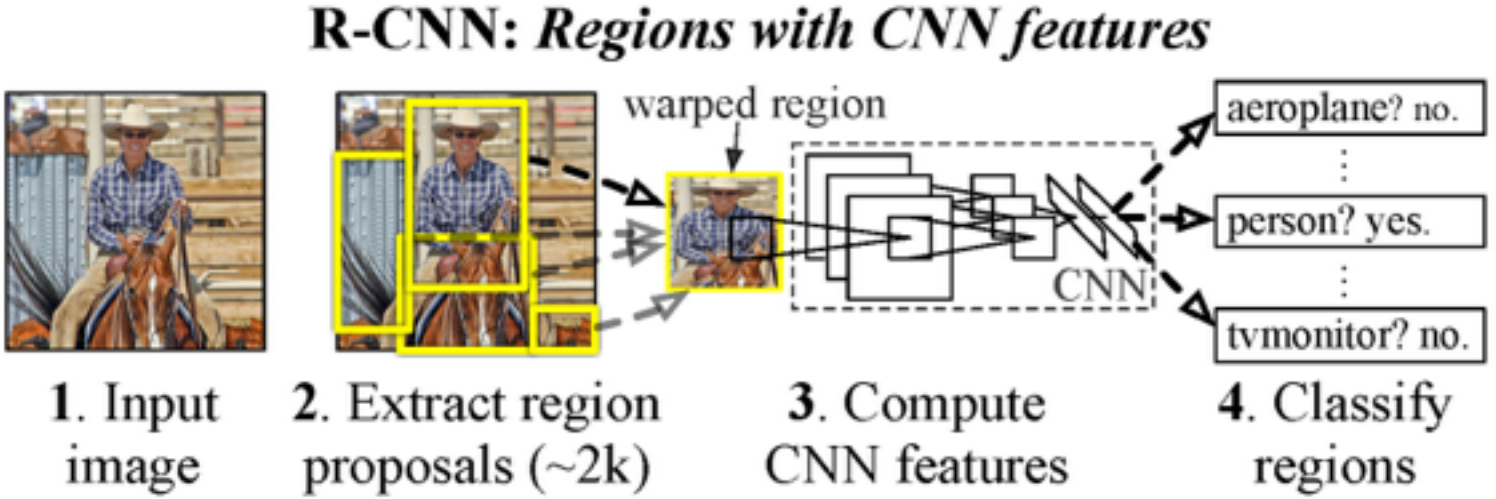


When labeled training data are scarce, supervised pre-training for an auxiliary task, followed by domain-specific fine-tuning, boosts performance significantly.

Improves mean average precision (mAP) by more than 50 percent relative to the previous best result on VOC

Object Recognition using Deep Learning

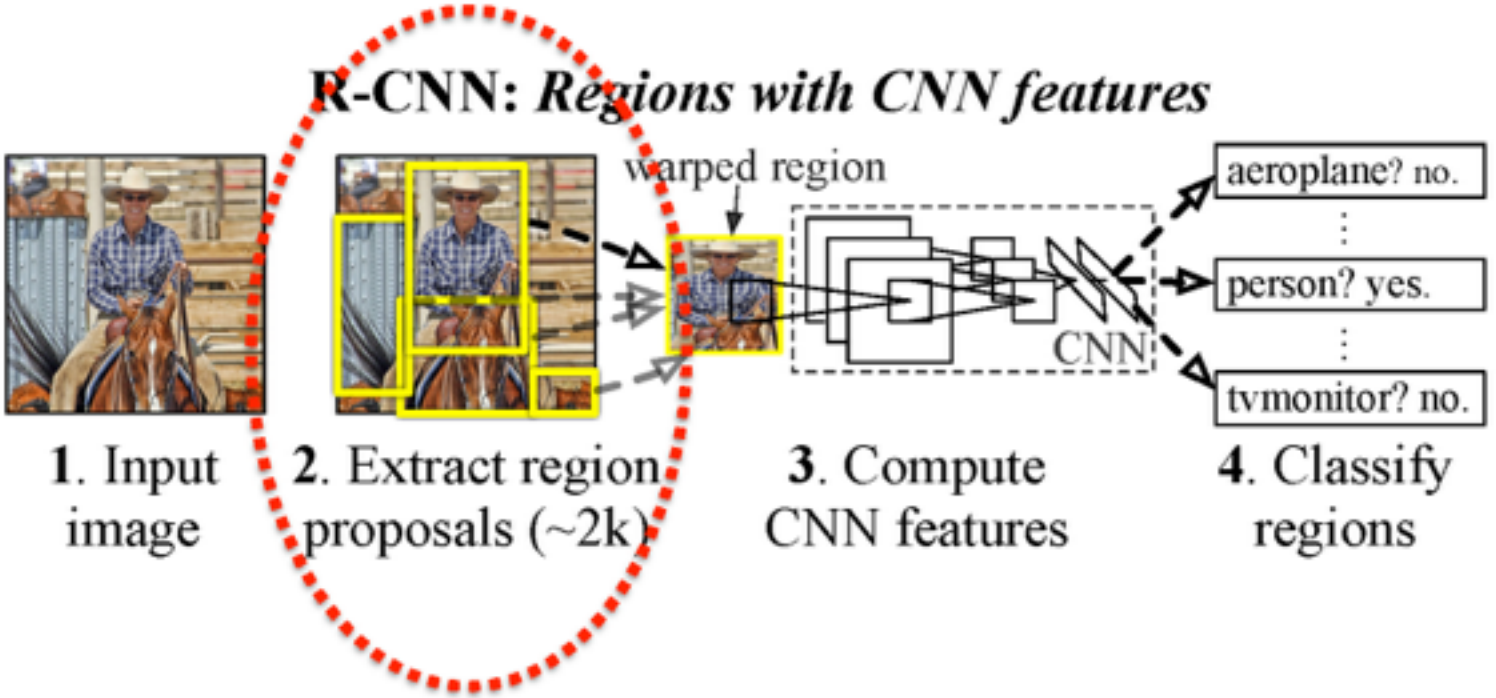
Image features are the engine of recognition.



Region Proposal

Sliding window + CNN = High computational cost

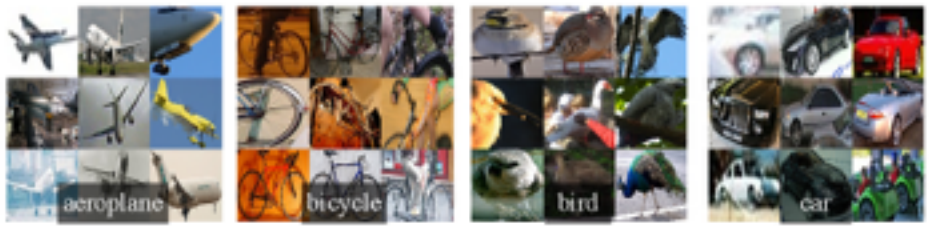
Selective Search!

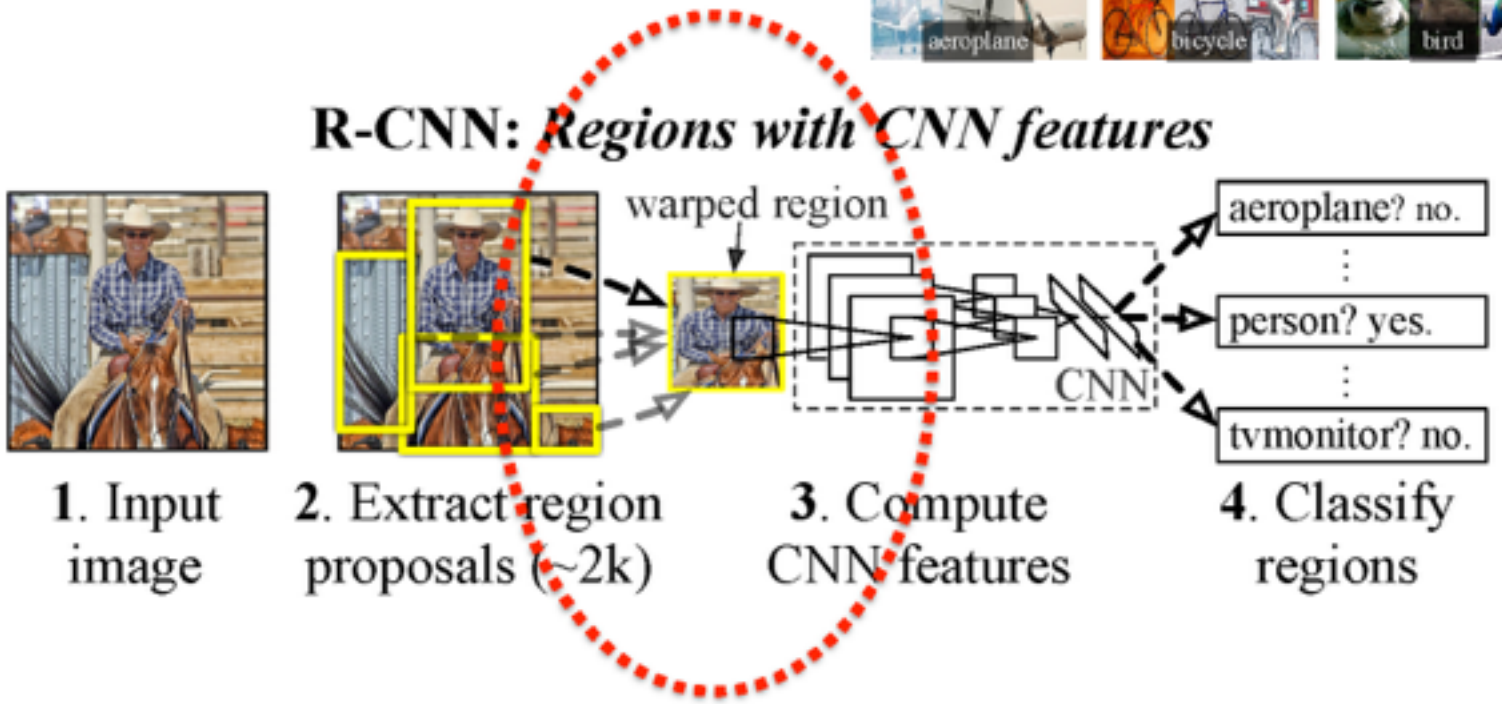


Region Warping

Regardless of size and aspect ratio

Warp to 224*224 patch





Object Proposal Transformations

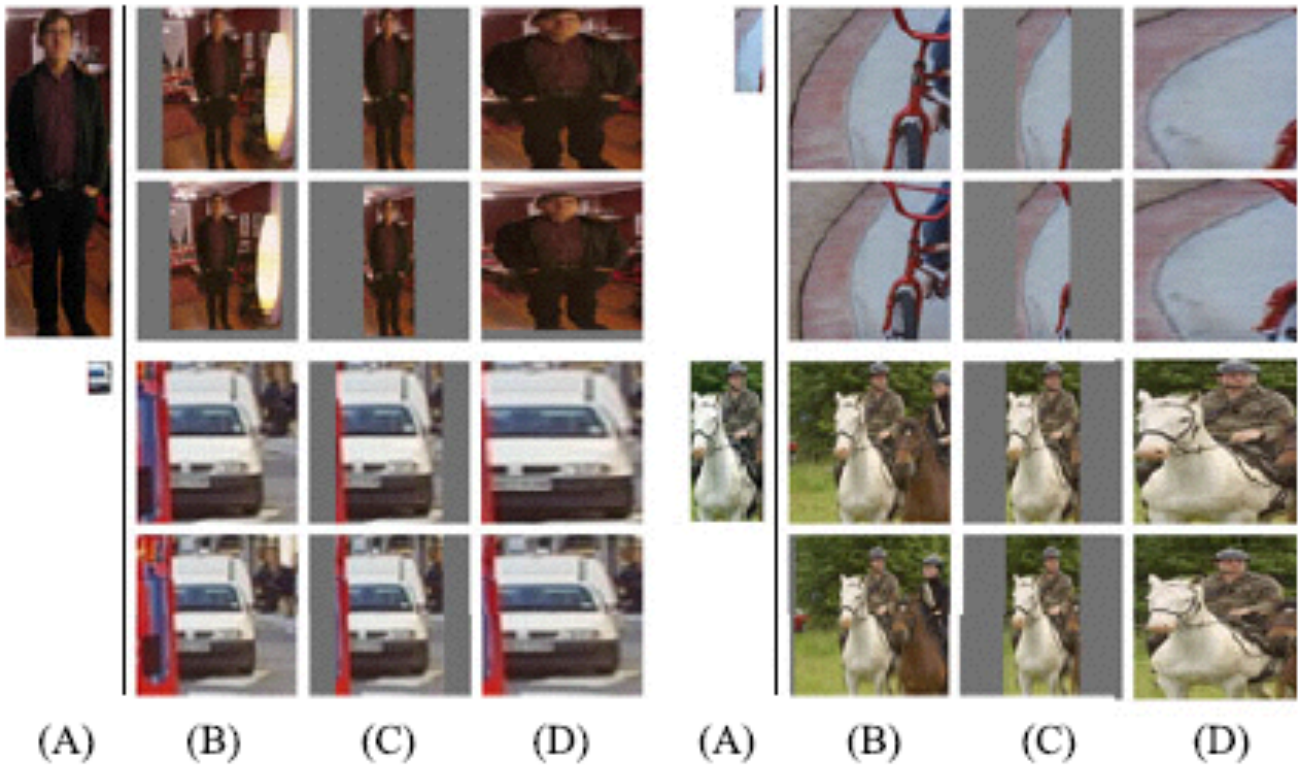


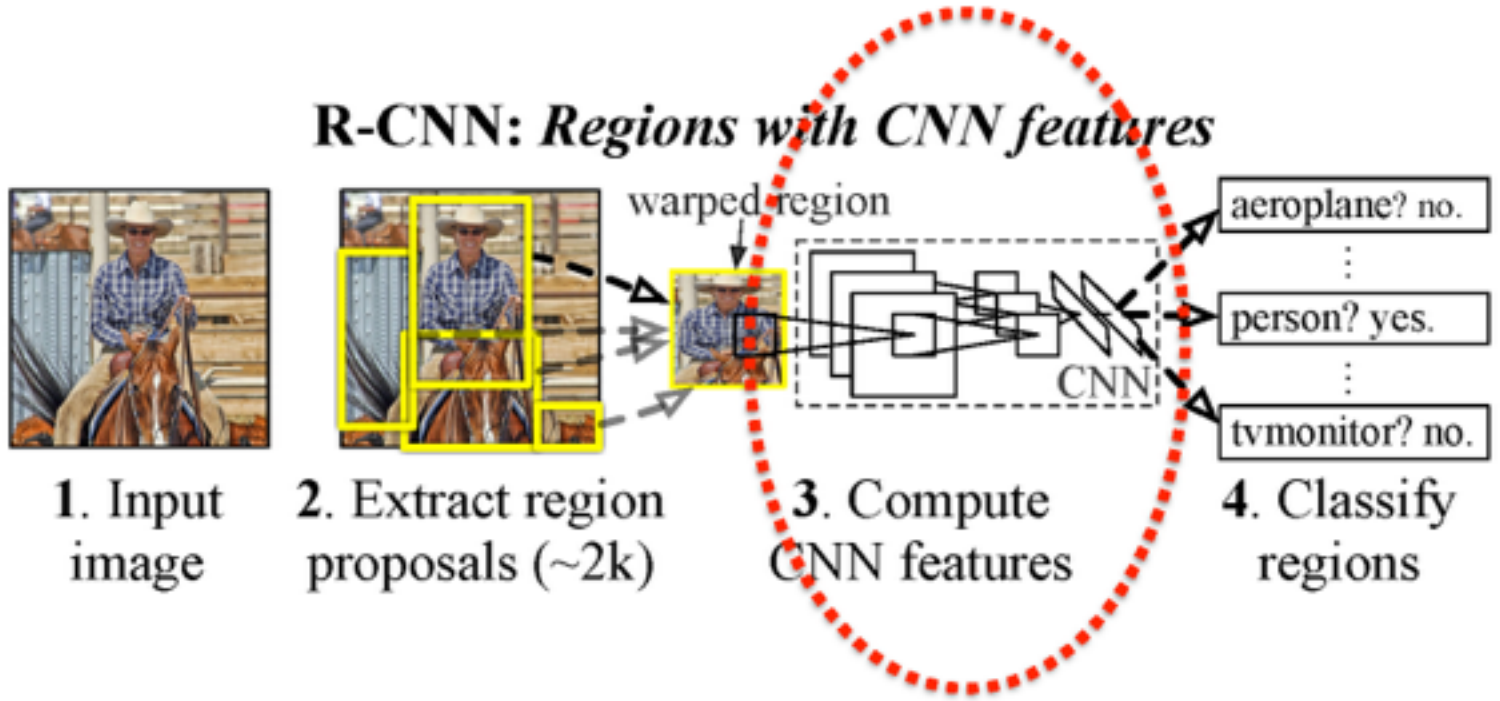
Fig. 7. Different object proposal transformations. (A) the original object proposal at its actual scale relative to the transformed CNN inputs; (B) tightest square with context; (C) tightest square without context; (D) warp. Within each column and example proposal, the top row corresponds to $p = 0$ pixels of context padding while the bottom row has $p = 16$ pixels of context padding.

Feature Extraction

Extracts a fixed-length feature vector From each proposal using a CNN

4096-dimensional feature vector

their own implementation of the CNN of (Krizhevsky et al. ECCV 2012)

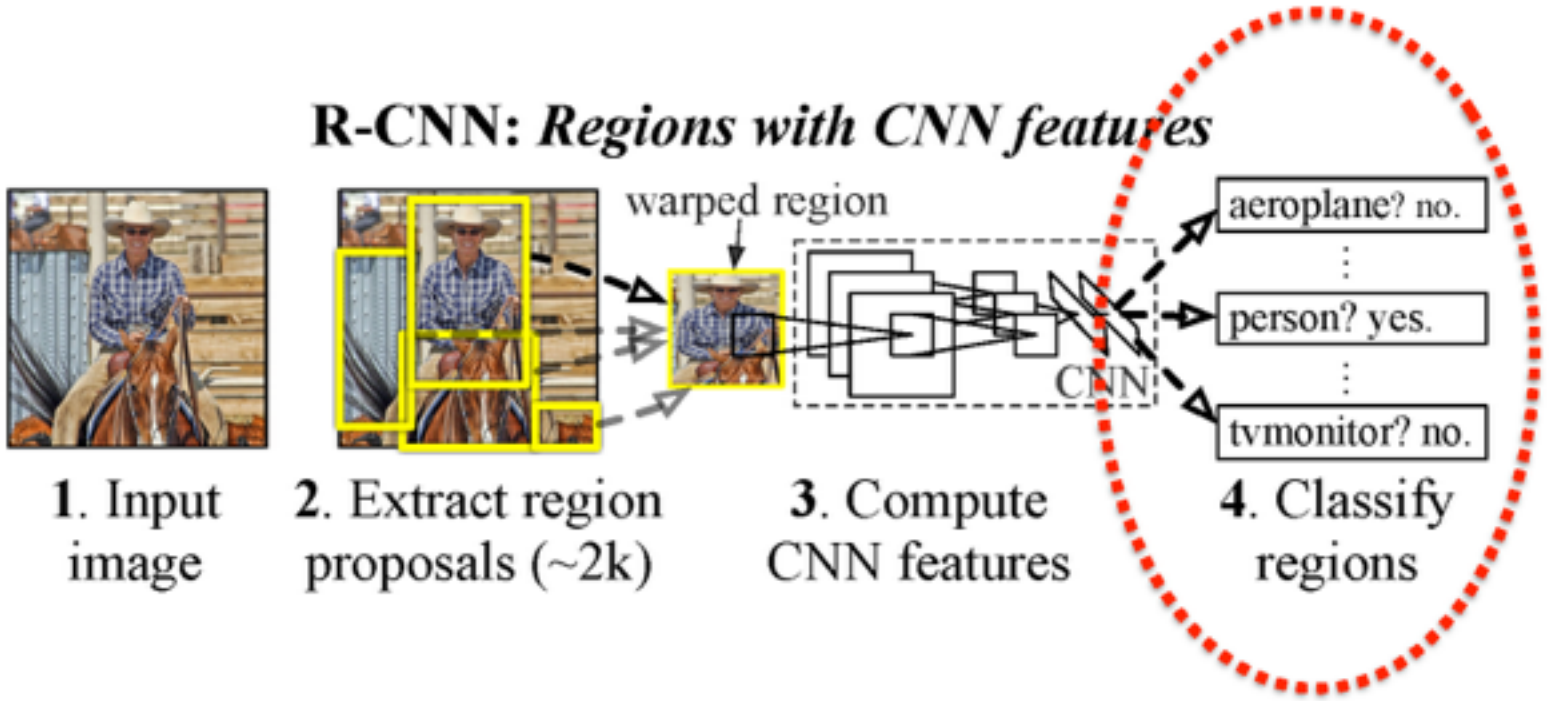


Inference

Then classifies each region with category-specific linear SVMs

Training + Testing using SVMs (with negative mining)

Efficient: shared CNN parameters + low dimensional features



Training

Supervised Pre-Training

Pre-trained the CNN on ILSVRC2012 classification

Domain-Specific Fine-Tuning

(N +1)-way classification

0.5 IoU overlap with a ground-truth boxas positives

Learning rate of 0.001

32 positive and 96 background to construct a minibatch of size 128

Object Category Classifiers

SVM per class

Only ground truth as positive

Less then 0.3 IoU overlap as negative (decrease mAP by 5 points if 0.5)

Drop from 54.2 to 50.9 percent mAP in 21-way softmax in VOC 2007

Testing

- Run selective search to extract around 2,000 region proposals (“fast mode”)
- Warp each proposal and forward propagate it through the CNN
- Score each extracted feature vector using the SVM trained for that class.
- Greedy non-maximum suppression (for each class independently)

  Rejects a region if it has an intersection-over-union (IoU) overlap with a higher scoring selected region larger than a learned threshold

- Bounding box regression

  Use a simple bounding-box regression stage to improve localization performance

Results Analysis

- Comparison with other method
- Run Time Analysis
- Ablation study

TABLE 2
Detection Average Precision (Percent) on VOC 2007 Test

VOC 2007 test	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
R-CNN pool_5	51.8	60.2	36.4	27.8	23.2	52.8	60.6	49.2	18.3	47.8	44.3	40.8	56.6	58.7	42.4	23.4	46.1	36.7	51.3	55.7	44.2
R-CNN fc_6	59.3	61.8	43.1	34.0	25.1	53.1	60.6	52.8	21.7	47.8	42.7	47.8	52.5	58.5	44.6	25.6	48.3	34.0	53.1	58.0	46.2
R-CNN fc_7	57.6	57.9	38.5	31.8	23.7	51.2	58.9	51.4	20.0	50.5	40.9	46.0	51.6	55.9	43.3	23.3	48.1	35.3	51.0	57.4	44.7
R-CNN FT pool_5	58.2	63.3	37.9	27.6	26.1	54.1	66.9	51.4	26.7	55.5	43.4	43.1	57.7	59.0	45.8	28.1	50.8	40.6	53.1	56.4	47.3
R-CNN FT fc_6	63.5	66.0	47.9	37.7	29.9	62.5	70.2	60.2	32.0	57.9	47.0	53.5	60.1	64.2	52.2	31.3	55.0	50.0	57.7	63.0	53.1
R-CNN FT fc_7	64.2	69.7	50.0	41.9	32.0	62.6	71.0	60.7	32.7	58.5	46.5	56.1	60.6	66.8	54.2	31.5	52.8	48.9	57.9	64.7	54.2
R-CNN FT fc_7 BB	**68.1**	**72.8**	**56.8**	**43.0**	**36.8**	**66.3**	**74.2**	**67.6**	**34.4**	**63.5**	**54.5**	**61.2**	**69.1**	**68.6**	**58.7**	**33.4**	**62.9**	**51.1**	**62.5**	**64.8**	**58.5**
DPM v5 [23]	33.2	60.3	10.2	16.1	27.3	54.3	58.2	23.0	20.0	24.1	26.7	12.7	58.1	48.2	43.2	12.0	21.1	36.1	46.0	43.5	33.7
DPM ST [61]	23.8	58.2	10.5	8.5	27.1	50.4	52.0	7.3	19.2	22.8	18.1	8.0	55.9	44.8	32.4	13.3	15.9	22.8	46.2	44.9	29.1
DPM HSC [62]	32.2	58.3	11.5	16.3	30.6	49.9	54.8	23.5	21.5	27.7	34.0	13.7	58.1	51.6	39.9	12.4	23.5	34.4	47.4	45.2	34.3

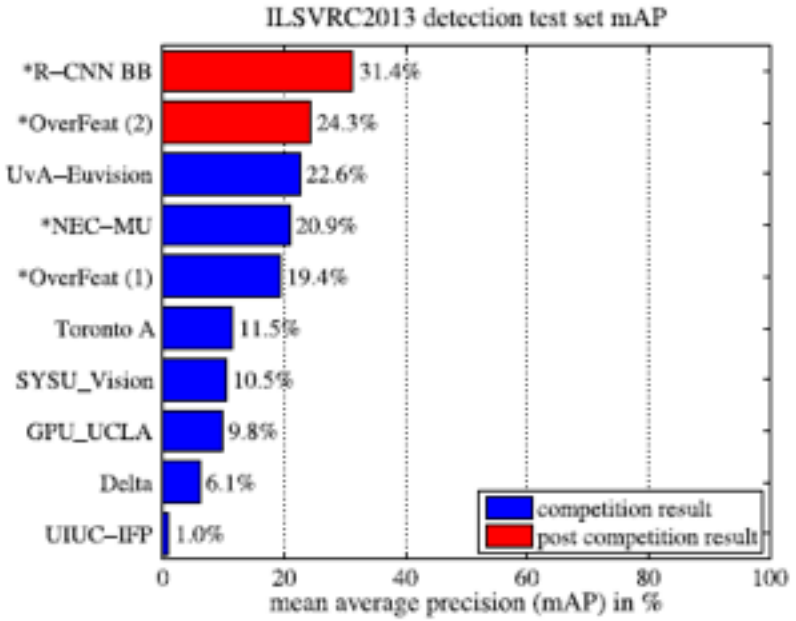
Rows 1-3 show R-CNN performance without fine-tuning. Rows 4-6 show results for the CNN pre-trained on ILSVRC 2012 and then fine-tuned (FT) on VOC 2007 trainval. Row 7 includes a simple bounding-box regression stage that reduces localization errors (Section 7.3). Rows 8-10 present DPM methods as a strong baseline. The first uses only HOG, while the next two use different feature learning approaches to augment or replace HOG. All R-CNN results use TorontoNet.

TABLE 1
Detection Average Precision (Percent) on VOC 2010 Test

VOC 2010 test	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
DPM v5 [23]	49.2	53.8	13.1	15.3	35.5	53.4	49.7	27.0	17.2	28.8	14.7	17.8	46.4	51.2	47.7	10.8	34.2	20.7	43.8	38.3	33.4
UVA [21]	56.2	42.4	15.3	12.6	21.8	49.3	36.8	46.1	12.9	32.1	30.0	36.5	43.5	52.9	32.9	15.3	41.1	31.8	47.0	44.8	35.1
Regionlets [54]	65.0	48.9	25.9	24.6	24.5	56.1	54.5	51.2	17.0	28.9	30.2	35.8	40.2	55.7	43.5	14.3	43.9	32.6	54.0	45.9	39.7
SegDPM [57]	61.4	53.4	25.6	25.2	35.5	51.7	50.6	50.8	19.3	33.8	26.8	40.4	48.3	54.4	47.1	14.8	38.7	35.0	52.8	43.1	40.4
R-CNN T-Net	67.1	64.1	46.7	32.0	30.5	56.4	57.2	65.9	27.0	47.3	40.9	66.6	57.8	65.9	53.6	26.7	56.5	38.1	52.8	50.2	50.2
R-CNN T-Net BB	71.8	65.8	53.0	36.8	35.9	59.7	60.0	69.9	27.9	50.6	41.4	70.0	62.0	69.0	58.1	29.5	59.4	39.3	61.2	52.4	53.7
R-CNN O-Net	76.5	70.4	58.0	40.2	39.6	61.8	63.7	81.0	36.2	64.5	45.7	80.5	71.9	74.3	60.6	31.5	64.7	52.5	64.6	57.2	59.8
R-CNN O-Net BB	**79.3**	**72.4**	**63.1**	**44.0**	**44.4**	**64.6**	**66.3**	**84.9**	**38.8**	**67.3**	**48.4**	**82.3**	**75.0**	**76.7**	**65.7**	**35.8**	**66.2**	**54.8**	**69.1**	**58.8**	**62.9**

T-Net stands for TorontoNet and O-Net for OxfordNet (Section 3.1.2). R-CNNs are most directly comparable to UVA and Regionlets since all methods use selective search region proposals. Bounding-box regression is described in Section 7.3. At publication time, SegDPM was the top-performer on the PASCAL VOC leaderboard. DPM and SegDPM use context rescoring not used by the other methods. SegDPM and all R-CNNs use additional training data.

Results



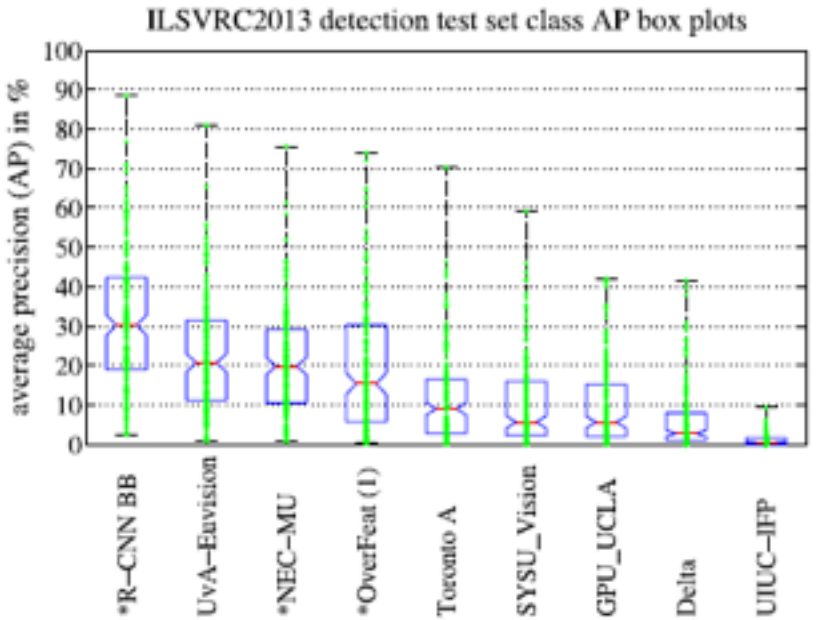


Fig. 3. (Left) Mean average precision on the ILSVRC2013 detection test set. Methods preceeded by * use outside training data (images and labels from the ILSVRC classification dataset in all cases). (Right) Box plots for the 200 average precision values per method. A box plot for the post-competition OverFeat result is not shown because per-class APs are not yet available. The red line marks the median AP, the box bottom and top are the 25th and 75th percentiles. The whiskers extend to the min and max AP of each method. Each AP is plotted as a green dot over the whiskers (best viewed digitally with zoom).

Results Analysis

R-CNNs vs OverFeat detection system

- Uses a sliding-window CNN for detection
- A top-performing method on the ILSVRC 2013 detection challenge
- mAP of 31.4 percent vs 24.3 percent on the 200-class

R-CNNs vs The popular deformable part models

- mAP of 62.9 percent vs 33.4. percent on VOC2010

R-CNNs vs Same region proposals with a spatial pyramid and bag-of-visual words

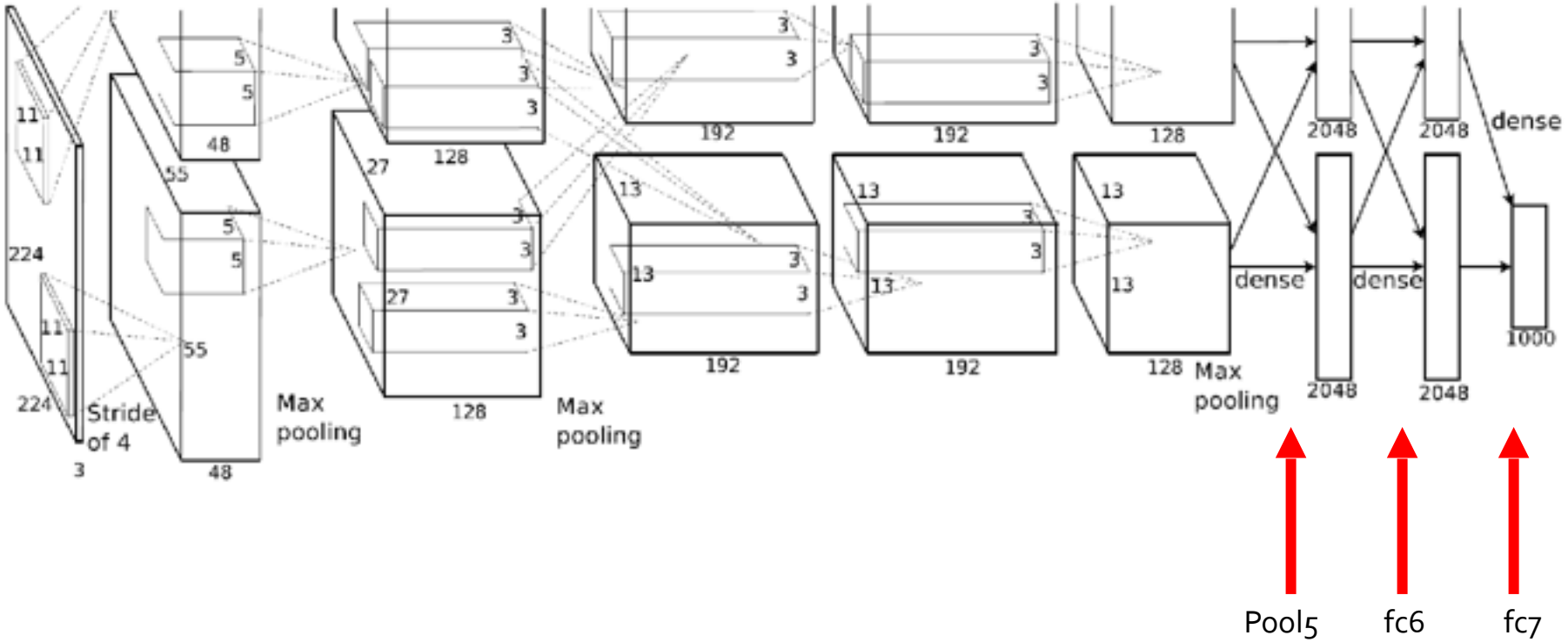
- mAP of 62.9 percent vs 35.1 percent on VOC2010

Run-Time Analysis

Two properties make detection efficient

- All CNN parameters are shared across all categories
- The feature vectors computed by the CNN are low-dimensional when compared to other common approaches (360 k versus 4 k-dimensional)
- Computing region proposals and features
- The only class specific computations are dot products between features and SVM weights
- Around 10 s/image on a GPU or 53 s/image on a CPU(TorontoNet)
- Takes only 30 ms longer to detect 200 classes than 20 classes on a CPU

UToronto "SuperVision" CNN



Ablation Study

- Last three layers: $pool_5$, fc_6 and fc_7
- With or without fine-tuning
- $pool_5$ uses only 6% parameters (possible to use DPM on top)
- Color helps (40.1% -> 43.4% VOC 2007 on fc_6)

VOC 2007 test	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
R-CNN $pool_5$	49.3	58.0	29.7	22.2	20.6	47.7	56.8	43.6	16.0	39.7	37.7	39.6	49.6	55.6	37.5	20.6	40.5	37.4	47.8	51.3	40.1
R-CNN fc_6	56.1	58.8	34.4	29.6	22.6	50.4	58.0	52.5	18.3	40.1	41.3	46.8	49.5	53.5	39.7	23.0	46.4	36.4	50.8	59.0	43.4
R-CNN fc_7	53.1	58.9	35.4	29.6	22.3	50.0	57.7	52.4	19.1	43.5	40.8	43.6	47.6	54.0	39.1	23.0	42.3	33.6	51.4	55.2	42.6
R-CNN FT $pool_5$	55.6	57.5	31.5	23.1	23.2	46.3	59.0	49.2	16.5	43.1	37.8	39.7	51.5	55.4	40.4	23.9	46.3	37.9	49.7	54.1	42.1
R-CNN FT fc_6	61.8	62.0	38.8	35.7	29.4	52.5	**61.9**	53.9	22.6	49.7	40.5	**48.8**	49.9	**57.3**	44.5	**28.5**	50.4	**40.2**	54.3	61.2	47.2
R-CNN FT fc_7	**60.3**	**62.5**	**41.4**	**37.9**	29.0	**52.6**	61.6	**56.3**	**24.9**	**52.3**	**41.9**	48.1	54.3	57.0	**45.0**	26.9	**51.8**	38.1	**56.6**	**62.2**	**48.0**
DPM HOG [19]	33.2	60.3	10.2	16.1	27.3	54.3	58.2	23.0	20.0	24.1	26.7	12.7	**58.1**	48.2	43.2	12.0	21.1	36.1	46.0	43.5	33.7
DPM ST [29]	23.8	58.2	10.5	8.5	27.1	50.4	52.0	7.3	19.2	22.8	18.1	8.0	55.9	44.8	32.4	13.3	15.9	22.8	46.2	44.9	29.1
DPM HSC [32]	32.2	58.3	11.5	16.3	**30.6**	49.9	54.8	23.5	21.5	27.7	34.0	13.7	**58.1**	51.6	39.9	12.4	23.5	34.4	47.4	45.2	34.3

Latest Versions

Fast R-CNN

- A fast framework for object detection with deep ConvNets
- 9x faster than traditional R-CNN
- Involved ROI pooling layer to speed up
- Reduces detection times (excluding region proposal computation) to 50 to 300 ms per image

Faster R-CNN

- Runs 200x faster than R-CNN and 10x faster than SPPnet at test-time
- A significantly higher mAP on PASCAL VOC than R-CNN
- Region proposal network

Outline

- Introduction
- Images Classification
- Object Detection

  R-CNN

  Traditional Feature Descriptor

  Selective Search
- **Implementation**
- Latest Application

Implementation

Caffe

- Convolution Architecture For Feature Extraction
- Written in C++
- Seamless switch between CPU and GPU

Implementation

https://github.com/rbgirshick/rcnn

Person Score=1.942

Bicycle Score=0.439

Outline

- Introduction
- Images Classification
- Object Detection

  R-CNN

  Traditional Feature Descriptor

  Selective Search
- Implementation
- **Latest Applications**

Latest Applications

DeepDream

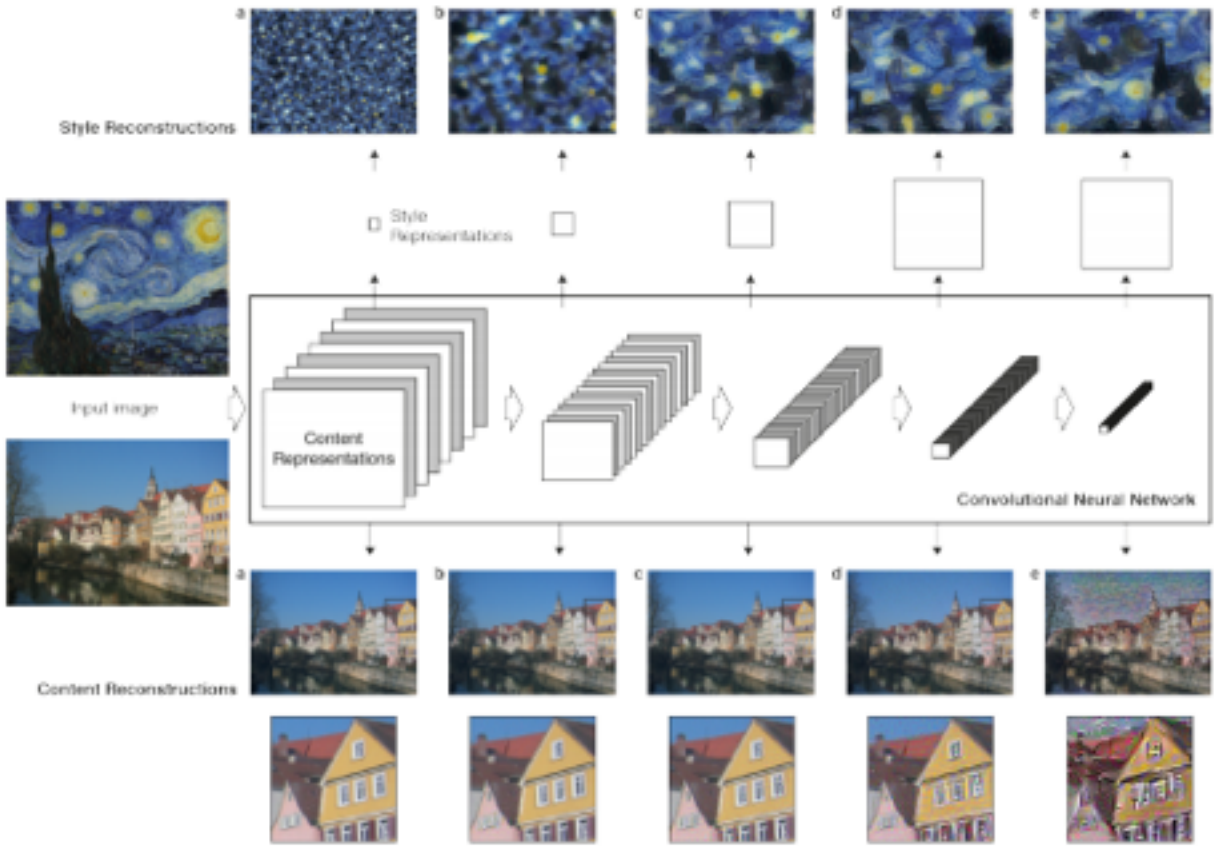
A computer vision program created by Google which uses a convolutional neural network to find and enhance patterns in images via algorithmic

Latest Applications

Deep Artist

- Neural Algorithm Can 'Paint' Photos In Style Of Any Artist From Van Gogh To Picasso
- Can morph an image to resemble a painting in the style of the great masters
- The system uses neural representations to separate and recombine content and style of arbitrary images, providing a neural algorithm for the creation of artistic images

A Neural Algorithm of Artistic Style



The Shipwreck of the Minotaur by J.M.W. Turner, 1805

The Starry Night by Vincent van Gogh, 1889

Der Schrei by Edvard Munch, 1893

E Femme nue assise by Pablo Picasso, 1910

Composition VII by Wassily Kandinsky, 1913

Bibliography

R-CNN

R. Girshick, J. Donahue, T. Darrell, and J. Malik. Region- based convolutional networks for accurate object detection and segmentation. *TPAMI*, 2015.

http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7112511

HOG

N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2005.

https://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf

SIFT

Lowe, David G.. “Object recognition from local scale-invariant features”. Proceedings of the International Conference on Computer Vision. pp. 1150–1157. (1999)

http://www.cs.ubc.ca/~lowe/papers/iccv99.pdf

Bibliography

Selective Search

J. Uijlings, K. van de Sande, T. Gevers and A. Smeulders, "Selective search for object recognition,"*Int. J. Comput. Vis.*, vol. 104, no. 3, pp. 154-171, 2013

http://koen.me/research/pub/uijlings-ijcv2013-draft.pdf

Fast R-CNN

R. Girshick, “Fast R-CNN,” in *IEEE International Conference on Computer Vision (ICCV)*, 2015.

http://arxiv.org/abs/1504.08083

Faster R-CNN

S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks,"*arXiv:1506.01497*, 2015

http://arxiv.org/pdf/1506.01497v3.pdf

Thanks!