Reconnaissance d'objets et vision artificielle

http://www.di.ens.fr/willow/teaching/recvis09

Lecture 7

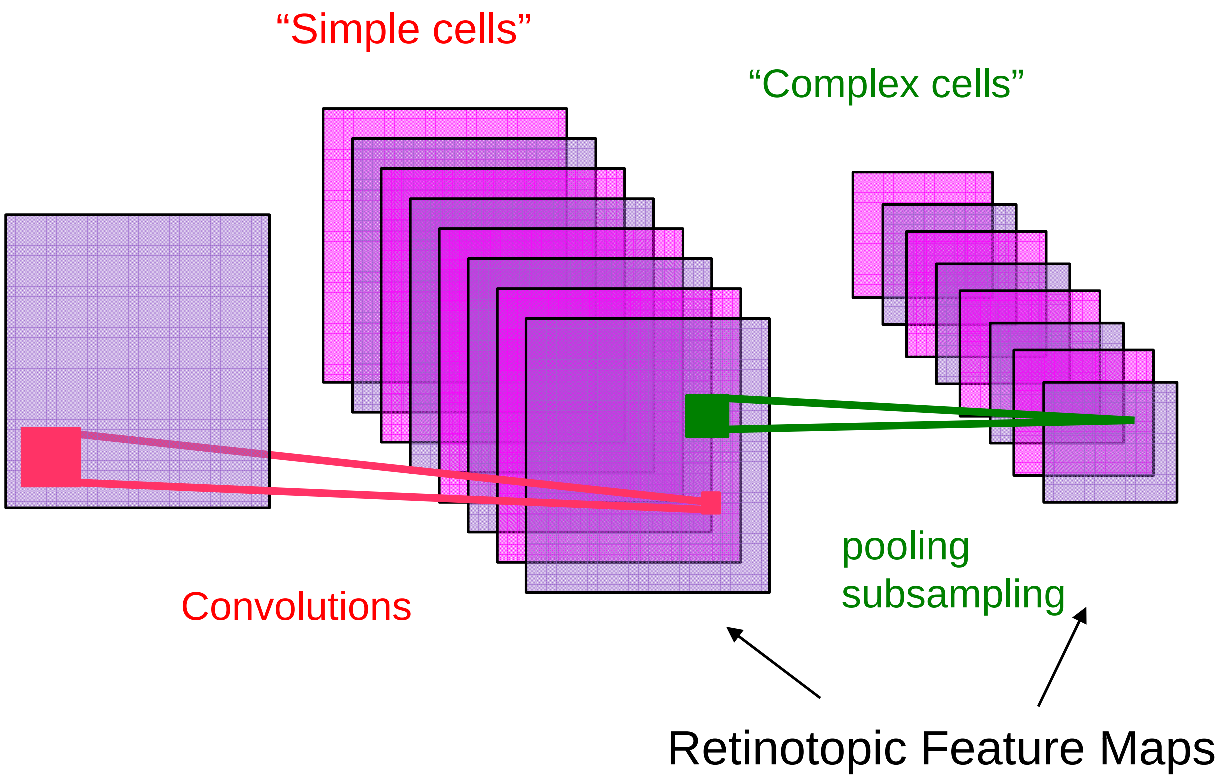
- A bit more on neural nets
- Optimization methods
- Part-based object models

Convolutional Nets

Yann LeCun
The Courant Institute of Mathematical Sciences
New York University
http://yann.lecun.com

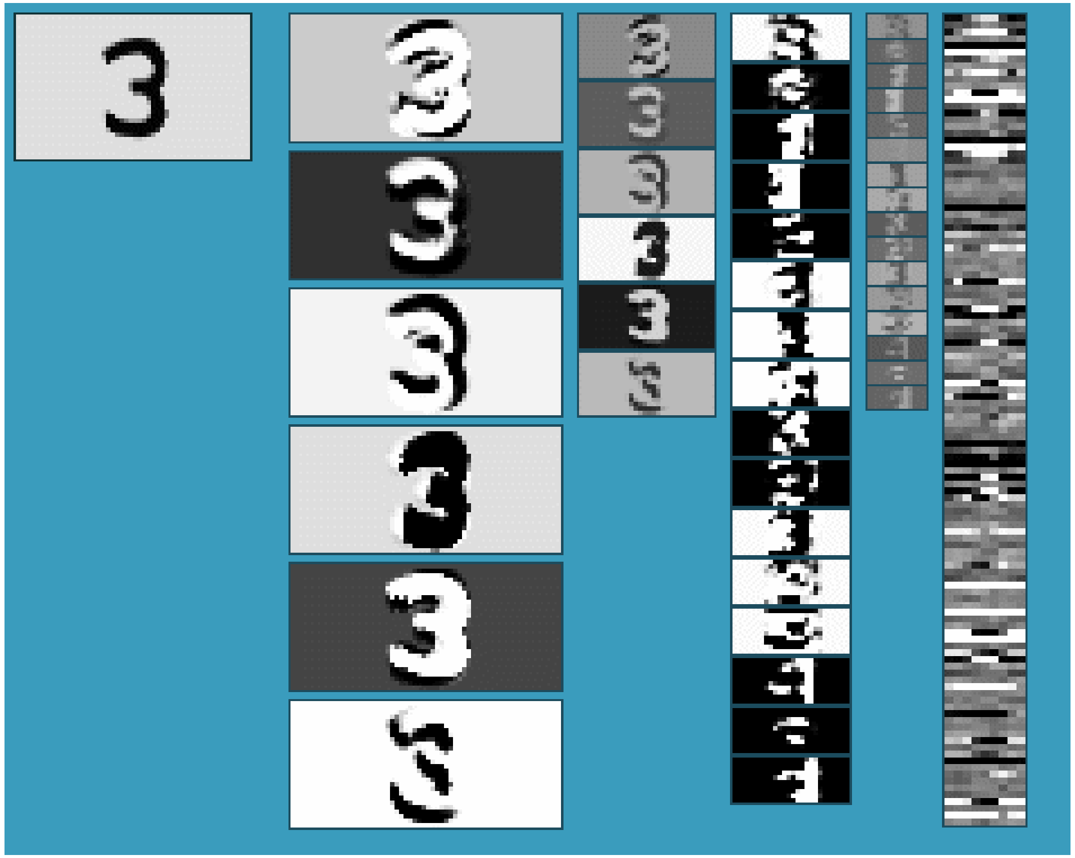
An Old Idea for Local Shift Invariance

- **[Hubel & Wiesel 1962]:**
  - simple cells detect local features
  - complex cells "pool" the outputs of simple cells within a retinotopic neighborhood.



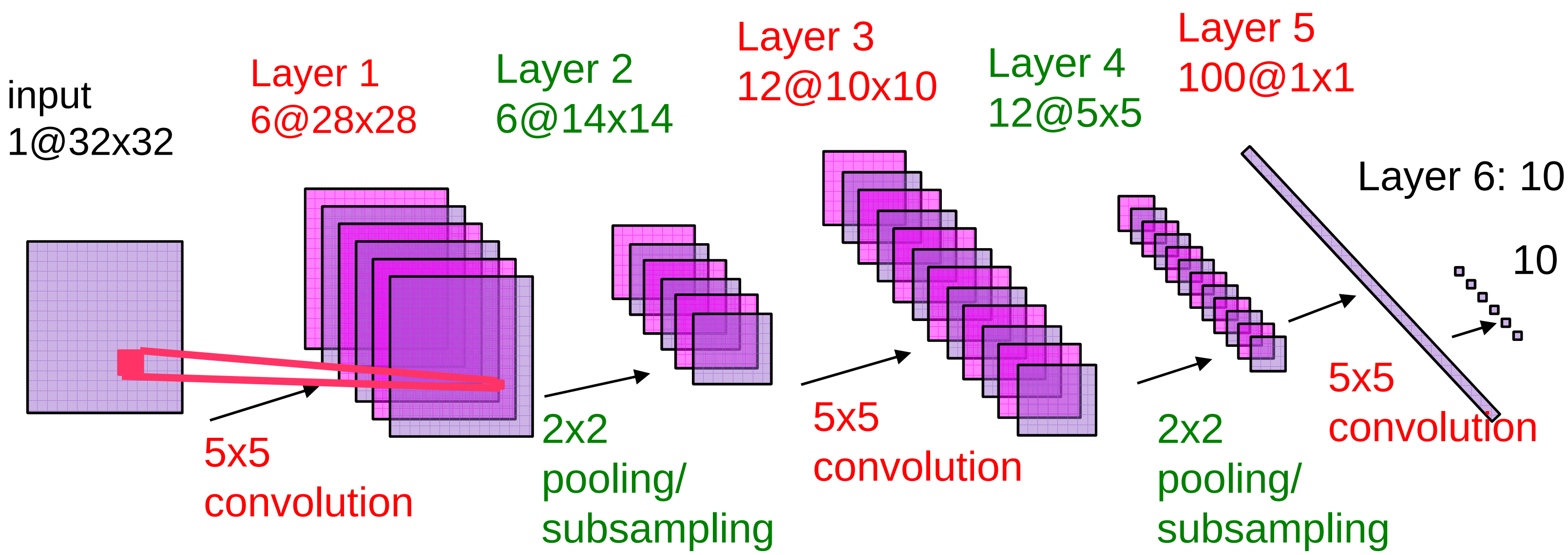
The Multistage Hubel-Wiesel Architecture

- **Building a complete artificial vision system:**
  - Stack multiple stages of simple cells / complex cells layers
  - Higher stages compute more global, more invariant features
  - Stick a classification layer on top
  - [Fukushima 1971-1982]
    - neocognitron
  - [LeCun 1988-2007]
    - convolutional net
  - [Poggio 2002-2006]
    - HMAX
  - [Ullman 2002-2006]
    - fragment hierarchy
  - [Lowe 2006]
    - HMAX



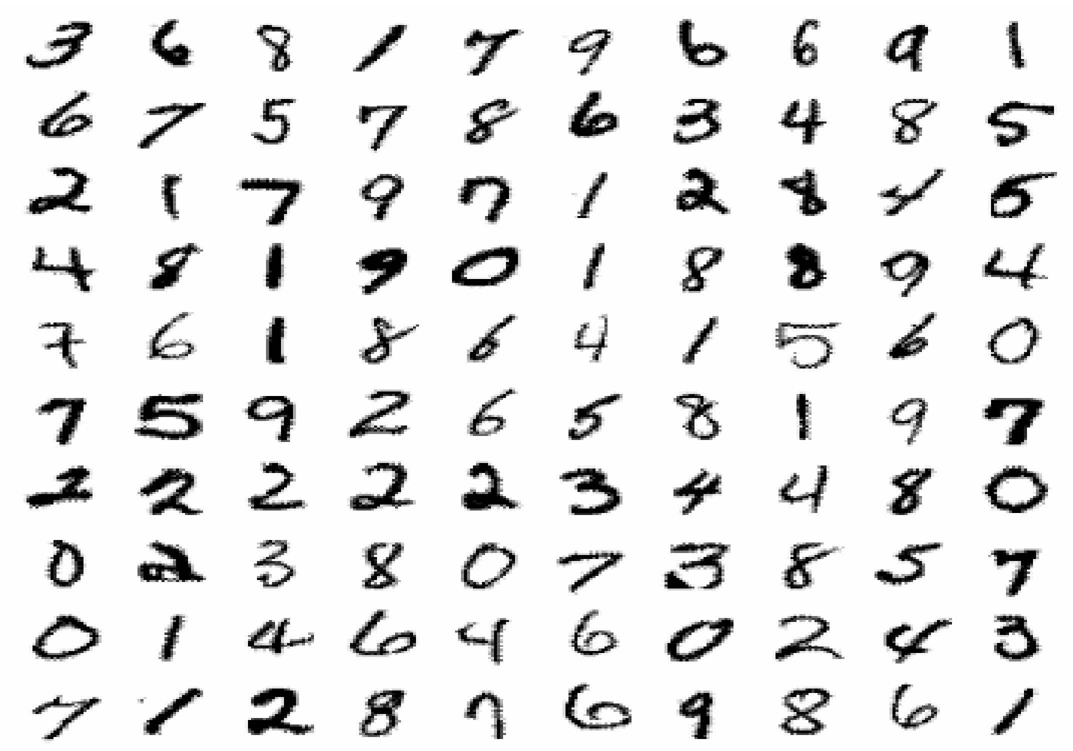
- **QUESTION: How do we find (or learn) the filters?**

Convolutional Net Architecture



- **Convolutional net for handwriting recognition** (400,000 synapses)
- Convolutional layers (simple cells): all units in a feature plane share the same weights
- Pooling/subsampling layers (complex cells): for invariance to small distortions.
- **Supervised gradient-descent learning using back-propagation**
- **The entire network is trained end-to-end. All the layers are trained simultaneously**.

MNIST Handwritten Digit Dataset



- Handwritten Digit Dataset MNIST: 60,000 training samples, 10,000 test samples

Results on MNIST Handwritten Digits

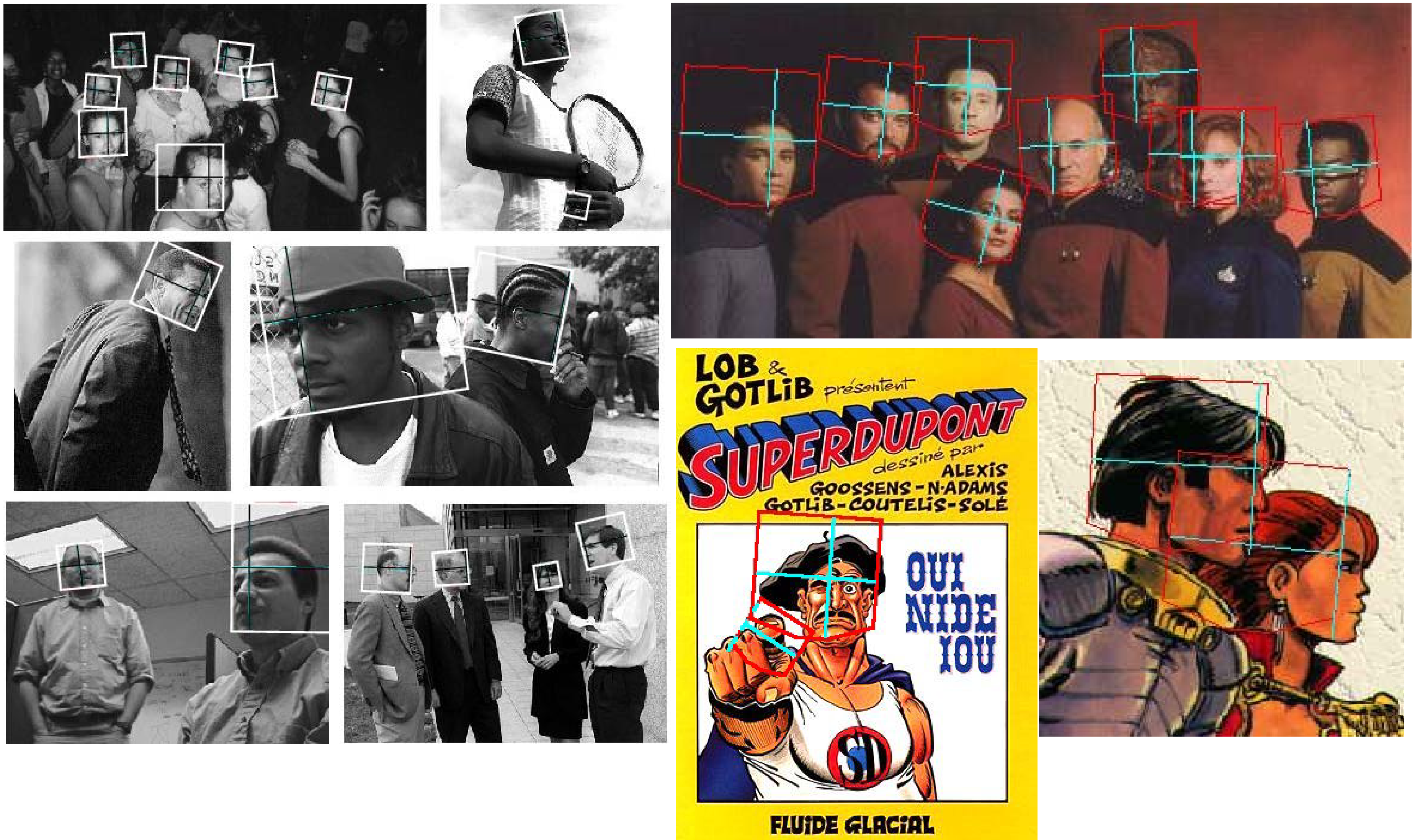
CLASSIFIER	DEFORMATION	PREPROCESSING	ERROR (%)	Reference
linear classifier (1-layer NN)		none	12.00	LeCun et al. 1998
linear classifier (1-layer NN)		deskewing	8.40	LeCun et al. 1998
pairwise linear classifier		deskewing	7.60	LeCun et al. 1998
K-nearest-neighbors, (L2)		none	3.09	Kenneth Wilder, U. Chicago
K-nearest-neighbors, (L2)		deskewing	2.40	LeCun et al. 1998
K-nearest-neighbors, (L2)		deskew, clean, blur	1.80	Kenneth Wilder, U. Chicago
K-NN L3, 2 pixel jitter		deskew, clean, blur	1.22	Kenneth Wilder, U. Chicago
K-NN, shape context matching		shape context feature	0.63	Belongie et al. IEEE PAMI 2002
40 PCA + quadratic classifier		none	3.30	LeCun et al. 1998
1000 RBF + linear classifier		none	3.60	LeCun et al. 1998
K-NN, Tangent Distance		subsamp 16x16 pixels	1.10	LeCun et al. 1998
SVM, Gaussian Kernel		none	1.40	
SVM deg 4 polynomial		deskewing	1.10	LeCun et al. 1998
Reduced Set SVM deg 5 poly		deskewing	1.00	LeCun et al. 1998
Virtual SVM deg-9 poly	Affine	none	0.80	LeCun et al. 1998
V-SVM, 2-pixel jittered		none	0.68	DeCoste and Scholkopf, MLJ2002
V-SVM, 2-pixel jittered		deskewing	0.56	DeCoste and Scholkopf, MLJ2002
2-layer NN, 300 HU, MSE		none	4.70	LeCun et al. 1998
2-layer NN, 300 HU, MSE,	Affine	none	3.60	LeCun et al. 1998
2-layer NN, 300 HU		deskewing	1.60	LeCun et al. 1998
3-layer NN, 500+150 HU		none	2.95	LeCun et al. 1998
3-layer NN, 500+150 HU	Affine	none	2.45	LeCun et al. 1998
3-layer NN, 500+300 HU, CE, reg		none	1.53	Hinton, unpublished, 2005
2-layer NN, 800 HU, CE		none	1.60	Simard et al., ICDAR 2003
2-layer NN, 800 HU, CE	Affine	none	1.10	Simard et al., ICDAR 2003
2-layer NN, 800 HU, MSE	Elastic	none	0.90	Simard et al., ICDAR 2003
2-layer NN, 800 HU, CE	Elastic	none	0.70	Simard et al., ICDAR 2003
Convolutional net LeNet-1		subsamp 16x16 pixels	1.70	LeCun et al. 1998
Convolutional net LeNet-4		none	1.10	LeCun et al. 1998
Convolutional net LeNet-5,		none	0.95	LeCun et al. 1998
Conv. net LeNet-5,	Affine	none	0.80	LeCun et al. 1998
Boosted LeNet-4	Affine	none	0.70	LeCun et al. 1998
Conv. net, CE	Affine	none	0.60	Simard et al., ICDAR 2003
Comv net, CE	Elastic	none	0.40	Simard et al., ICDAR 2003

Some Results on MNIST (from raw images: no preprocessing)

CLASSIFIER	DEFORMATION	ERROR	Reference
Knowledge-free methods (a fixed permutation of the pixels would make no difference)			
2-layer NN, 800 HU, CE		1.60	Simard et al., ICDAR 2003
3-layer NN, 500+300 HU, CE, reg		1.53	Hinton, in press, 2005
SVM, Gaussian Kernel		1.40	Cortes 92 + Many others
Convolutional nets			
Convolutional net LeNet-5,		0.80	Ranzato et al. NIPS 2006
Convolutional net LeNet-6,		0.70	Ranzato et al. NIPS 2006
Training set augmented with Affine Distortions			
2-layer NN, 800 HU, CE	Affine	1.10	Simard et al., ICDAR 2003
Virtual SVM deg-9 poly	Affine	0.80	Scholkopf
Convolutional net, CE	Affine	0.60	Simard et al., ICDAR 2003
Training et augmented with Elastic Distortions			
2-layer NN, 800 HU, CE	Elastic	0.70	Simard et al., ICDAR 2003
Convolutional net, CE	Elastic	0.40	Simard et al., ICDAR 2003

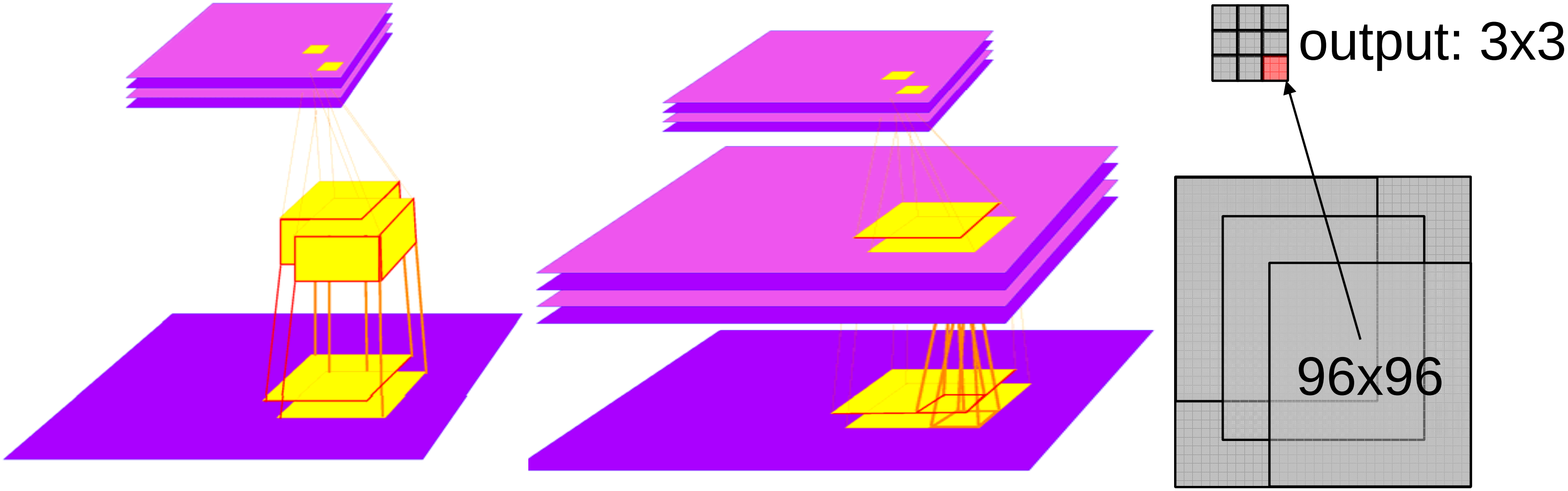
Note: some groups have obtained good results with various amounts of preprocessing such as deskewing (e.g. 0.56% using an SVM with smart kernels [deCoste and Schoelkopf]) hand-designed feature representations (e.g. 0.63% with “shape context” and nearest neighbor [Belc

Face Detection and Pose Estimation: Results



Face Detection with a Convolutional Net

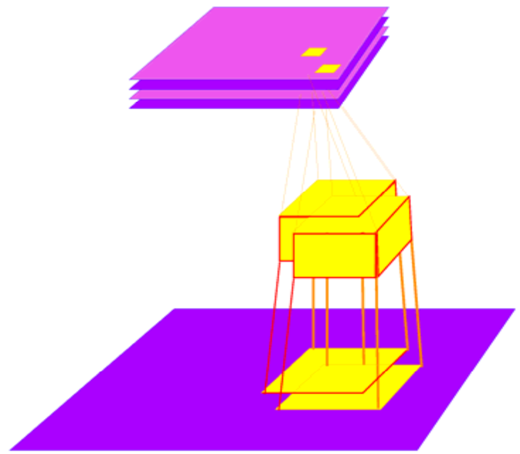
Applying a ConvNet on Sliding Windows is Very Cheap!

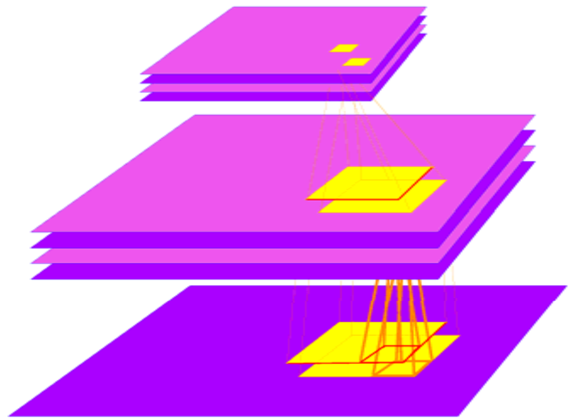


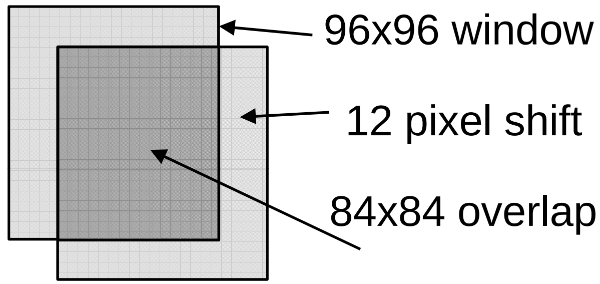
- Traditional Detectors/Classifiers must be applied to every location on a large input image, at multiple scales.
- Convolutional nets can replicated over large images very cheaply.
- The network is applied to multiple scales spaced by 1.5.

Building a Detector/Recognizer: Replicated Convolutional Nets

- Computational cost for replicated convolutional net:
  - 96x96 -> 4.6 million multiply-accumulate operations
  - 120x120 -> 8.3 million multiply-accumulate operations
  - 240x240 -> 47.5 million multiply-accumulate operations
  - 480x480 -> 232 million multiply-accumulate operations
- Computational cost for a non-convolutional detector of the same size, applied every 12 pixels:
  - 96x96 -> 4.6 million multiply-accumulate operations
  - 120x120 -> 42.0 million multiply-accumulate operations
  - 240x240 -> 788.0 million multiply-accumulate operations
  - 480x480 -> 5,083 million multiply-accumulate operations



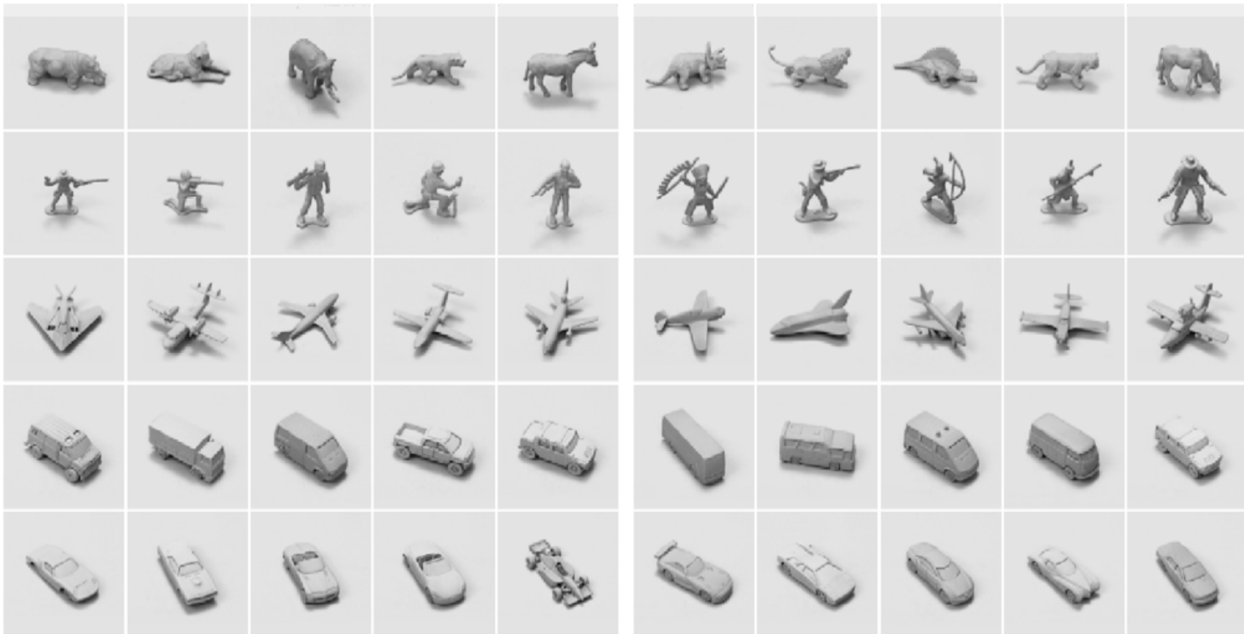




Generic Object Detection and Recognition with Invariance to Pose and Illumination

- **50** toys belonging to 5 categories: **animal, human figure, airplane, truck, car**
- **10** instance per category: 5 instances used for training, 5 instances for testing
- **Raw dataset: 972** stereo pair of each object instance. **48,600** image pairs total.

- **For each instance:**
- **18 azimuths**
  - 0 to 350 degrees every 20 degrees
- **9 elevations**
  - 30 to 70 degrees from horizontal every 5 degrees
- **6 illuminations**
  - on/off combinations of 4 lights
  - **2 cameras (stereo)**
- 7.5 cm apart
  - 40 cm from the object



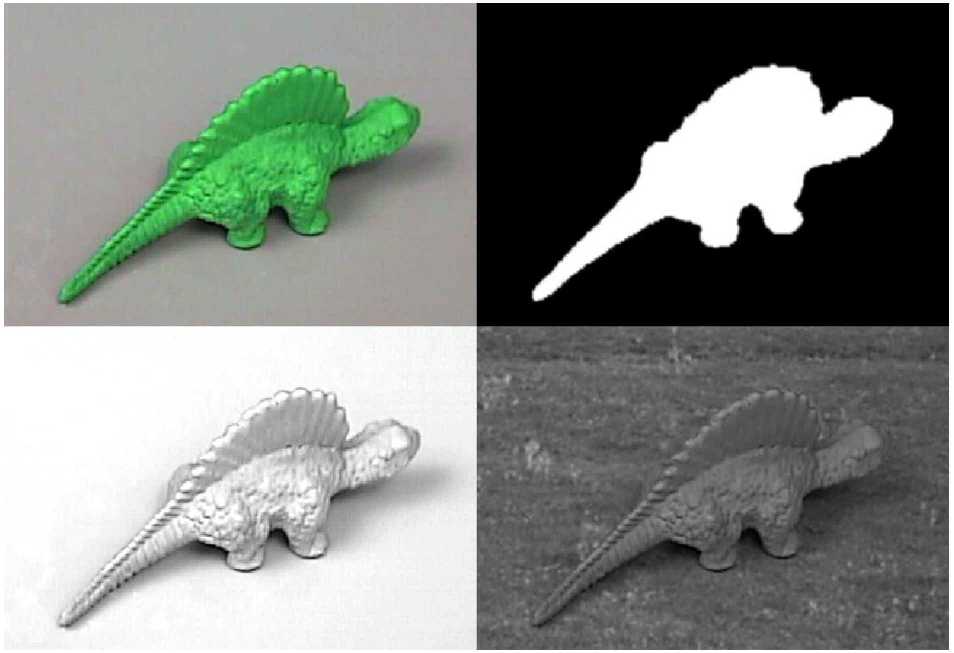
Training instances **Test instances**

Data Collection, Sample Generation

Image capture setup

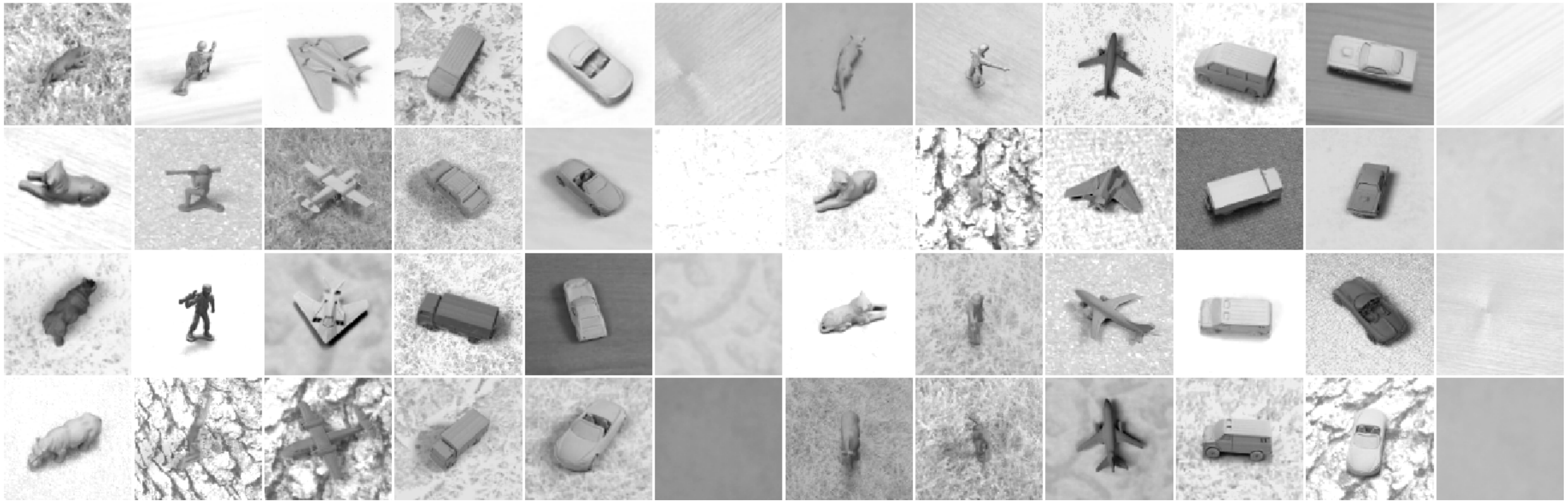
Objects are painted green so that:
- all features other than shape are removed
- objects can be segmented, transformed, and composited onto various backgrounds

Original image | **Object mask**

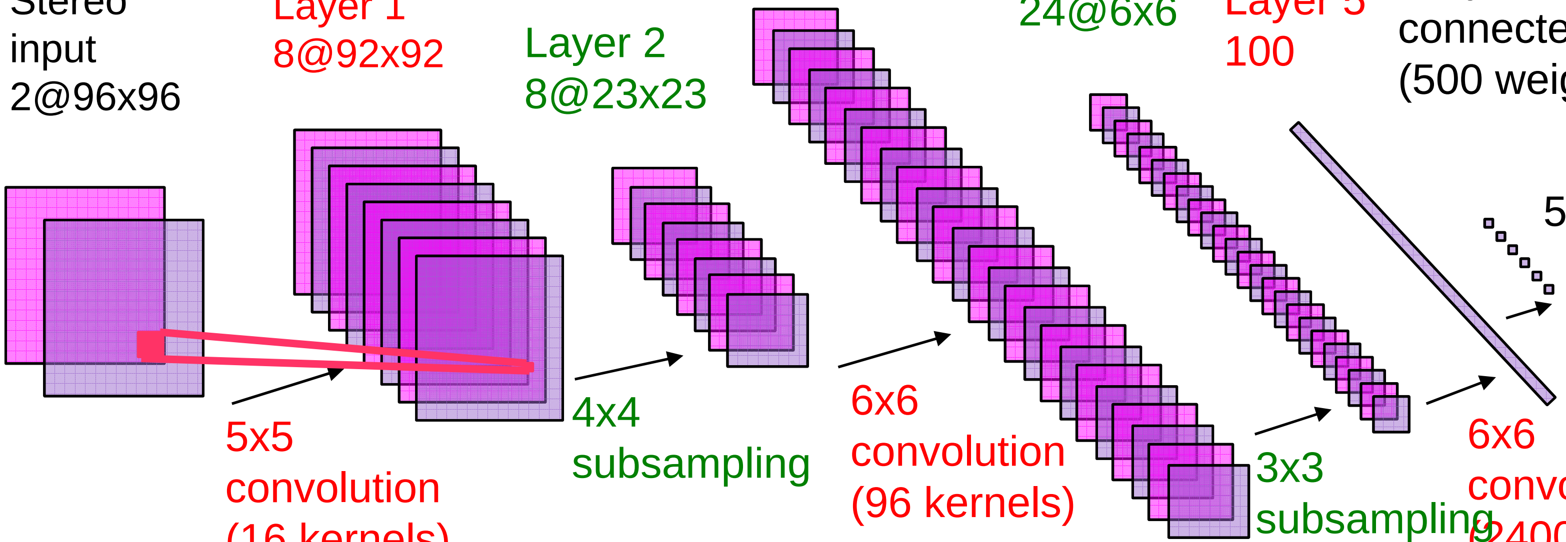


Shadow factor | **Composite image**

Textured and Cluttered Datasets

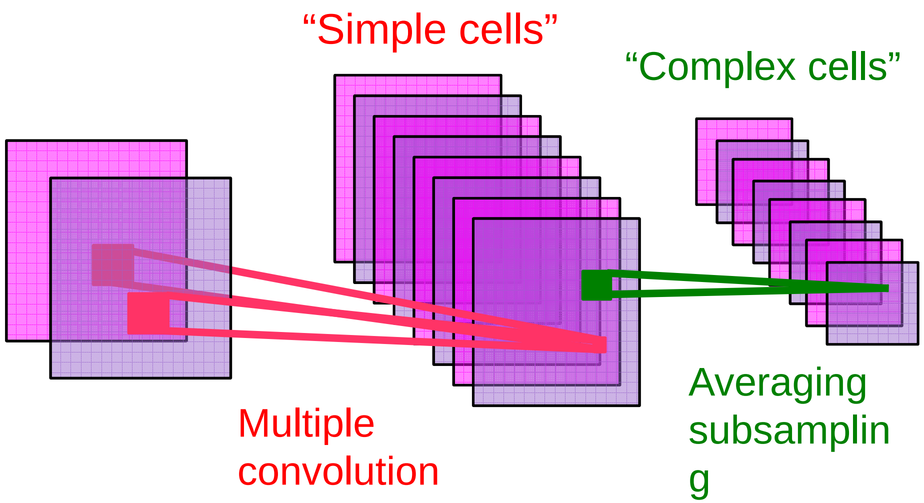


Convolutional Network

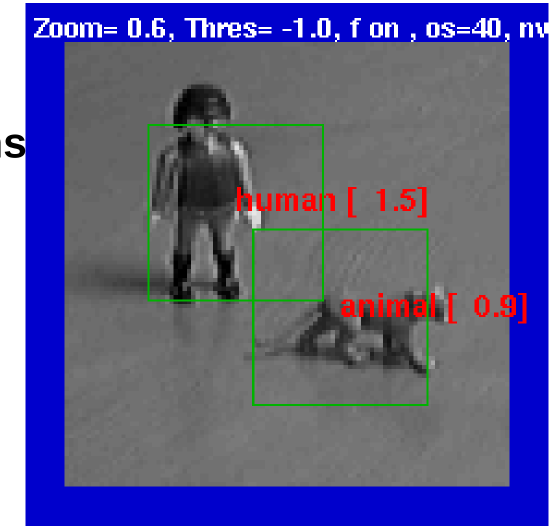


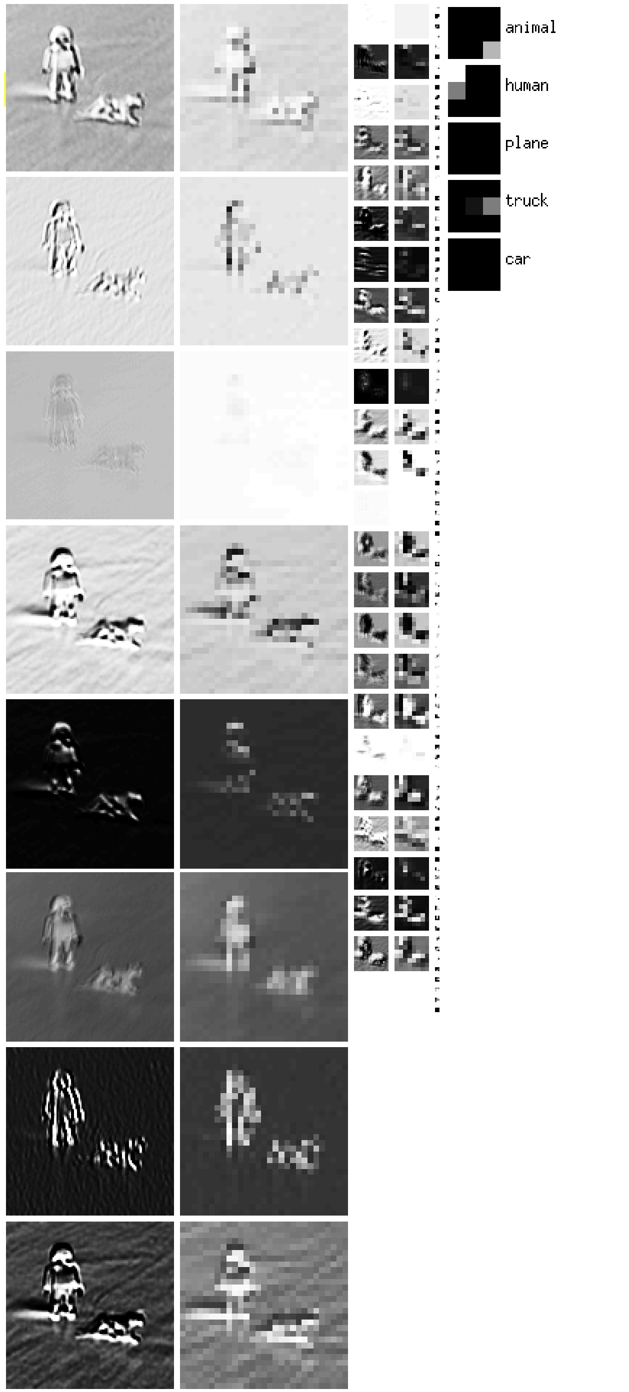
- **90,857 free parameters, 3,901,162 connections.**
- The architecture alternates convolutional layers (feature detectors) and subsampling layers (local feature pooling for invariance to small distortions).
- **The entire network is trained end-to-end** (all the layers are trained simultaneously).
- A gradient-based algorithm is used to minimize a supervised loss function.

Alternated Convolutions and Subsampli



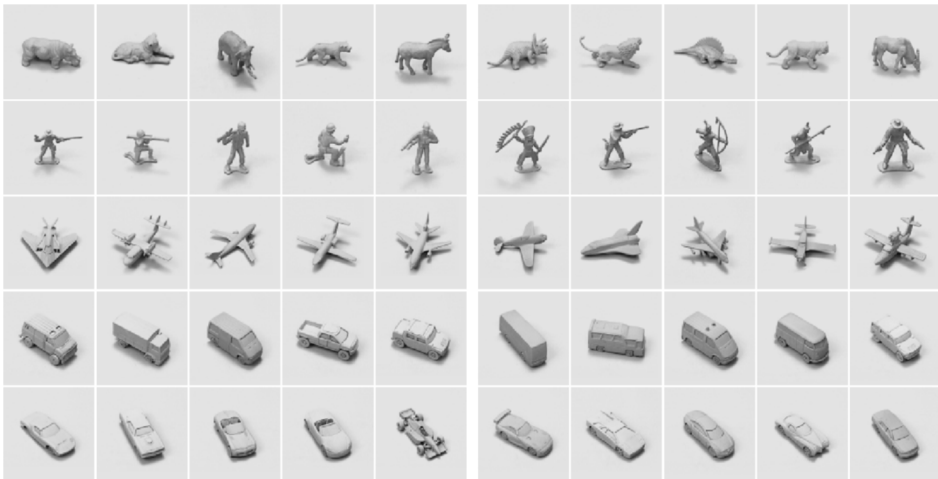
- **Local features are extracted everywhere.**
- **averaging/subsampling layer builds robustness to variations in feature locations.**
- **Hubel/Wiesel'62, Fukushima'71, LeCun'89, Riesenhuber & Poggio'02, Ullman'02,....**





Normalized-Uniform Set: Error Rates

- Linear Classifier on raw stereo images: 30.2% error.
- K-Nearest-Neighbors on raw stereo images: 18.4% error.
- K-Nearest-Neighbors on PCA-95: 16.6% error.
- Pairwise SVM on 96x96 stereo images: 11.6% error
- Pairwise SVM on 95 Principal Components: 13.3% error.
- Convolutional Net on 96x96 stereo images: 5.8% error.



Training instances Test instances

Jittered-Cluttered Dataset

- **Jittered-Cluttered Dataset:**
- **291,600** tereo pairs for training, **58,320** for testing
- Objects are jittered: position, scale, in-plane rotation, contrast, brightness, backgrounds, distractor objects,...
- Input dimension: 98x98x2 (approx 18,000)

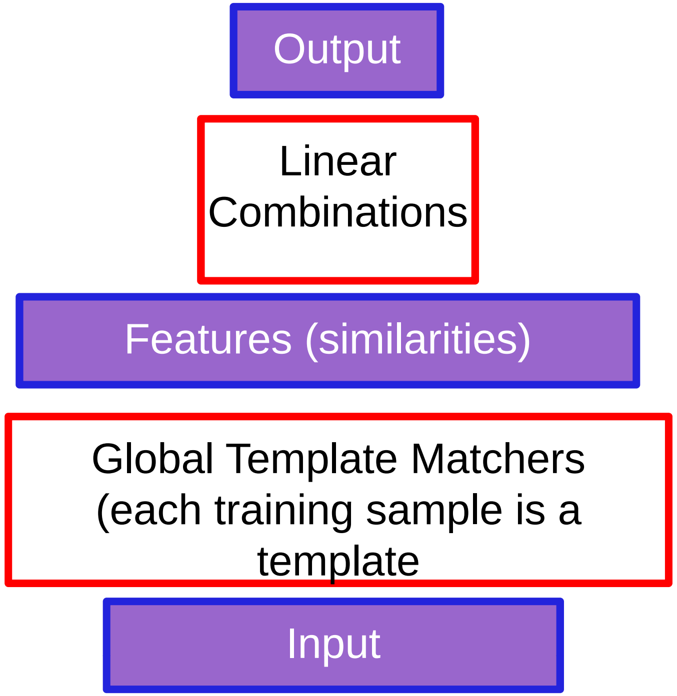
Experiment 2: Jittered-Cluttered Dataset

- 291,600 training samples, 58,320 test samples
- SVM with Gaussian kernel 43.3% error
- Convolutional Net with binocular input: 7.8% error
- Convolutional Net + SVM on top: 5.9% error
- Convolutional Net with monocular input: 20.8% error
- Smaller mono net (DEMO): 26.0% error
- Dataset available from http://www.cs.nyu.edu/~yann

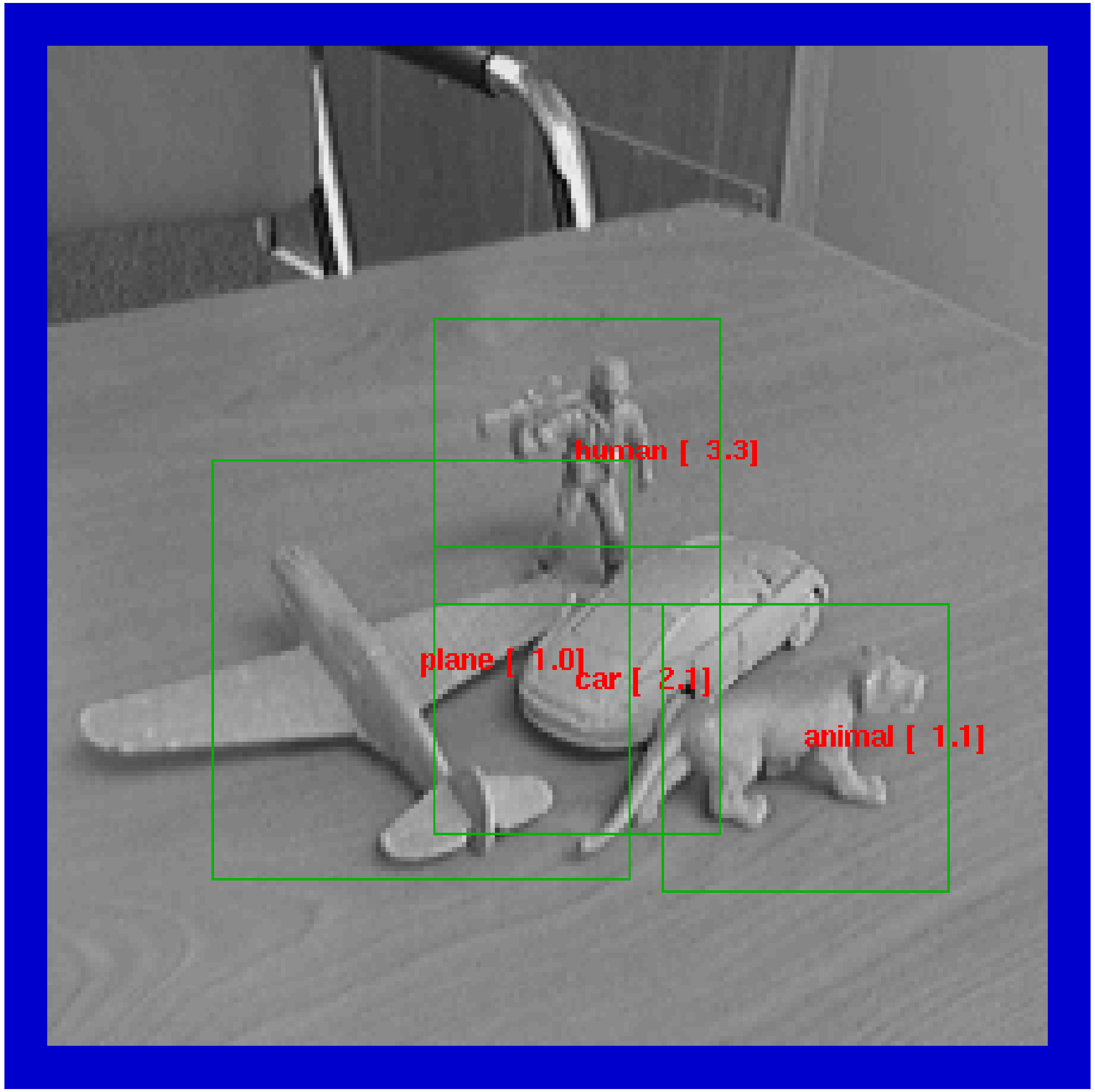
What's wrong with K-NN and SVMs?

- K-NN and SVM with Gaussian kernels are based on matching global templates
- Both are "shallow" architectures
- There is now way to learn invariant recognition tasks with such naïve architectures (unless we use an impractically large number of templates).

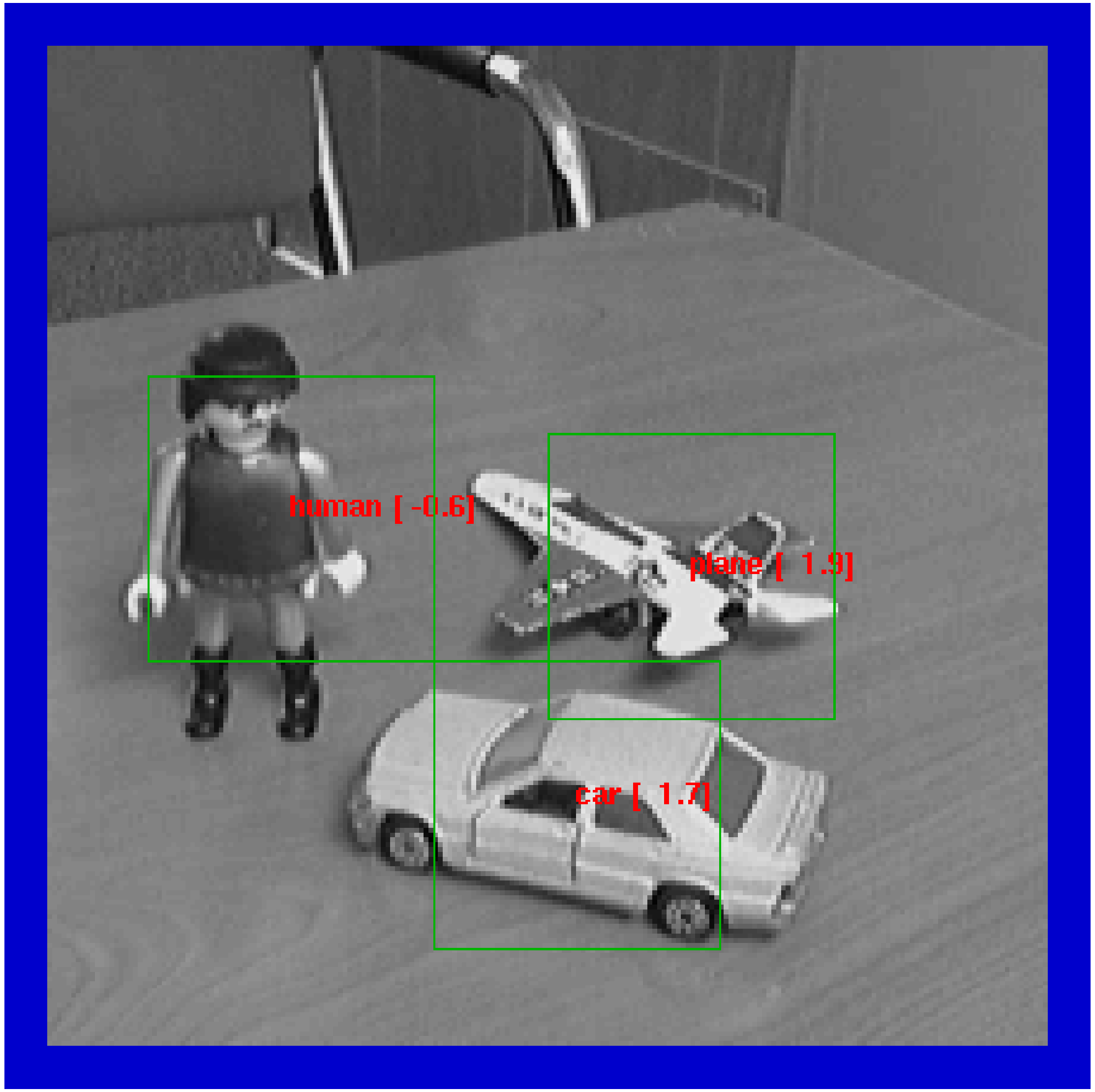
- The number of necessary templates grows exponentially with the number of dimensions of variations.
- Global templates are in trouble when the variations include: category, instance shape, configuration (for articulated object), position, azimuth, elevation, scale, illumination, texture, albedo, in-plane rotation, background luminance, background texture, background clutter, ...



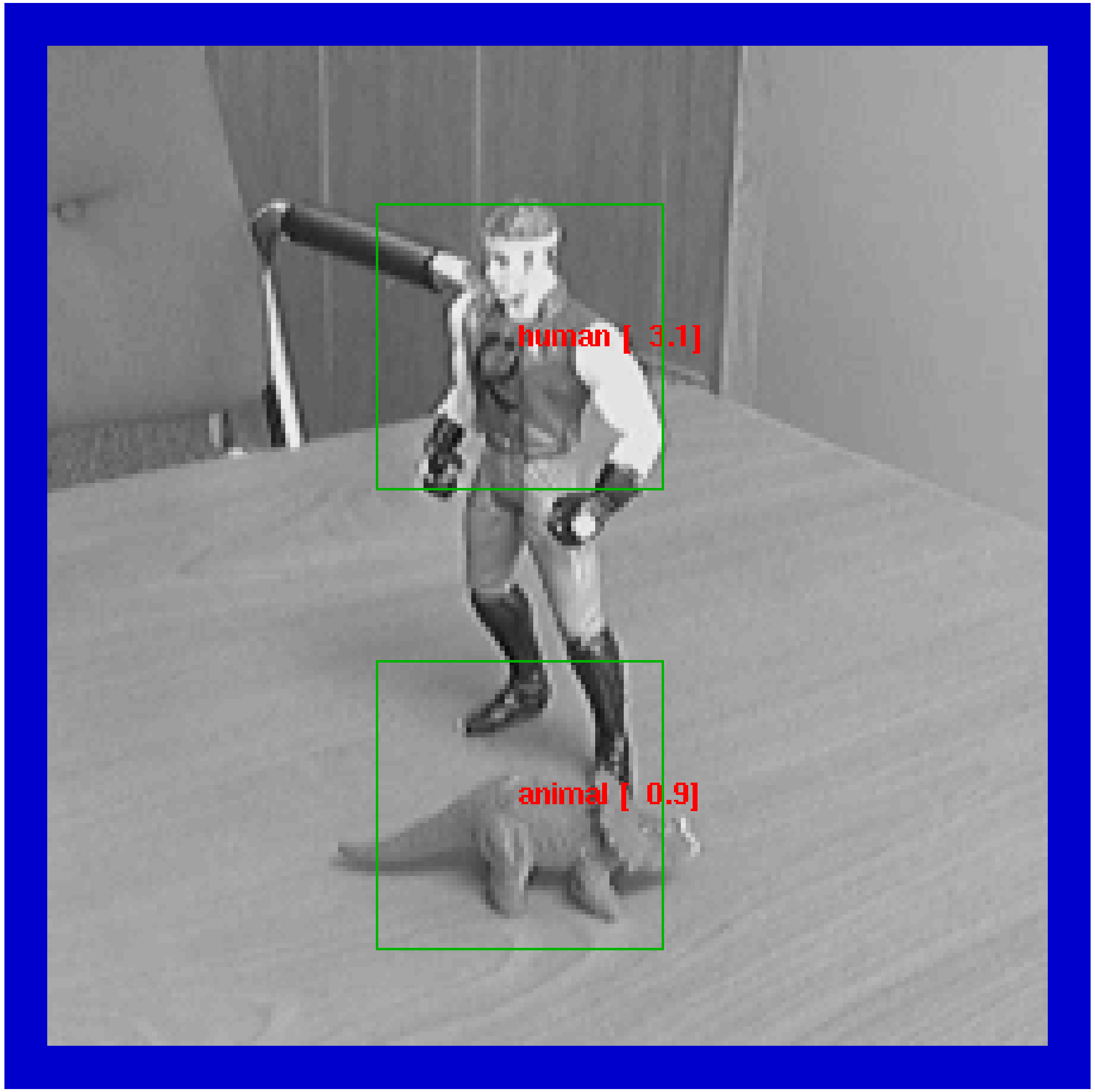
Examples (Monocular Mode)



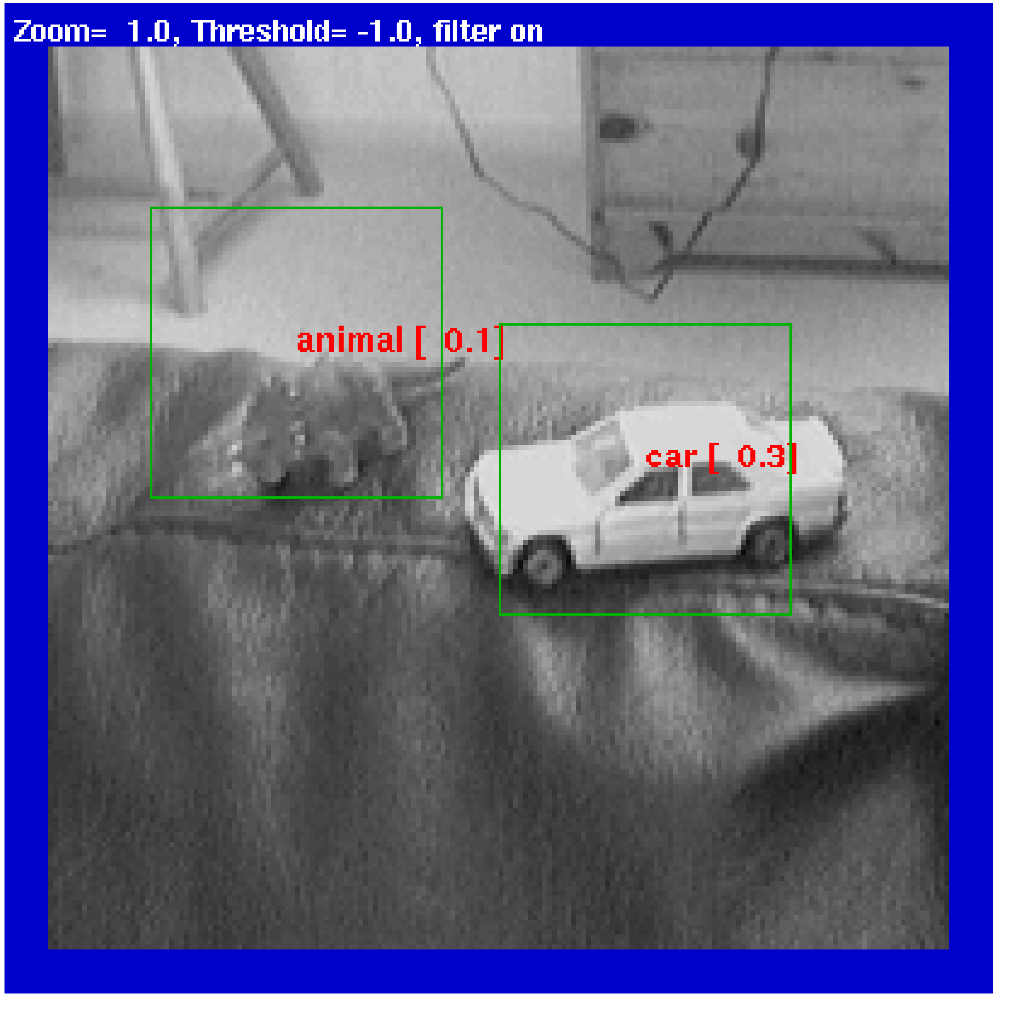
Examples (Monocular Mode)



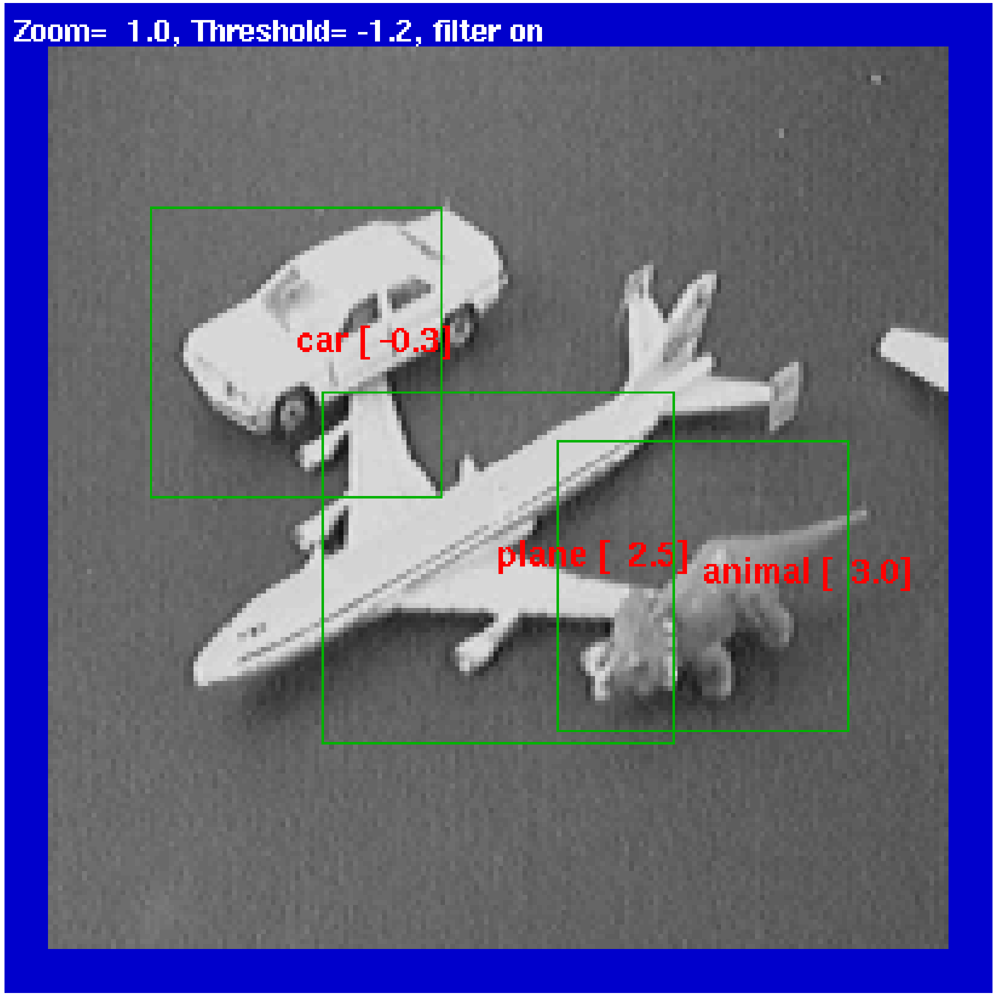
Examples (Monocular Mode)



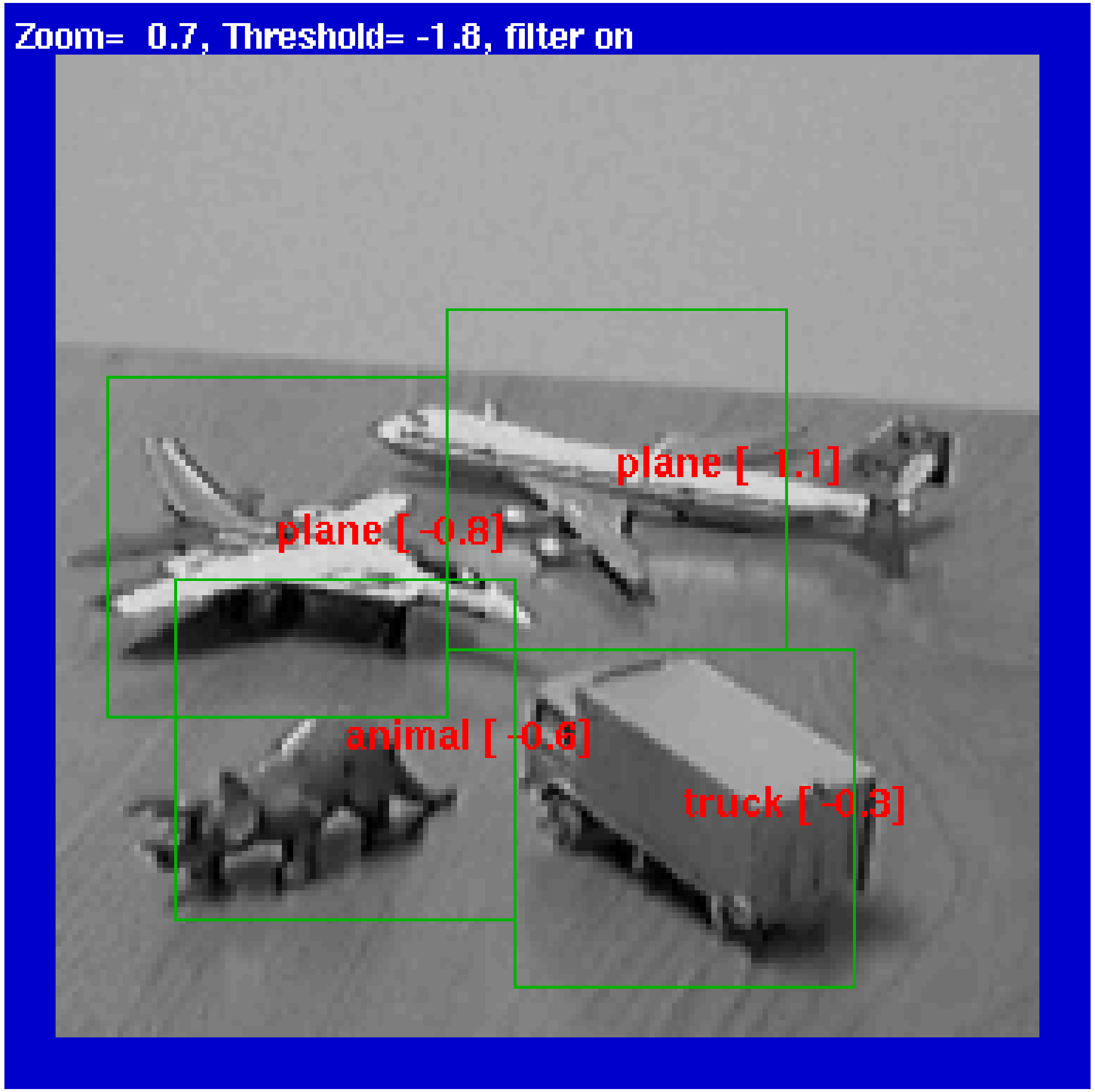
Examples (Monocular Mode)



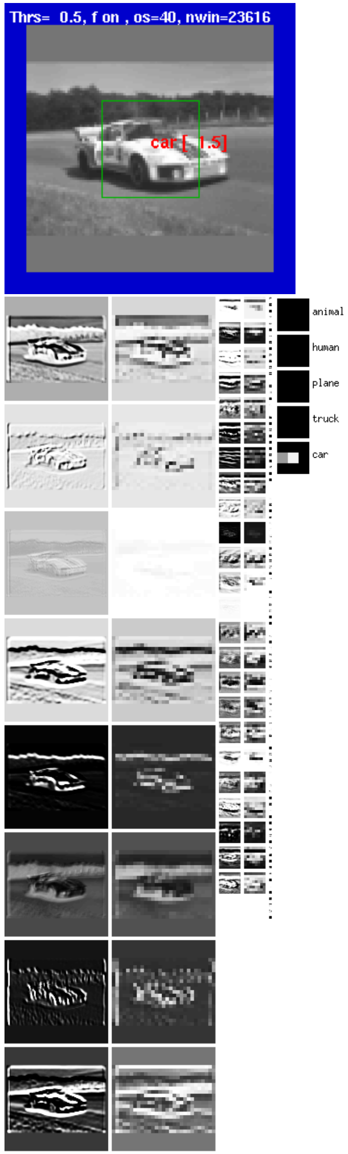
Examples (Monocular Mode)

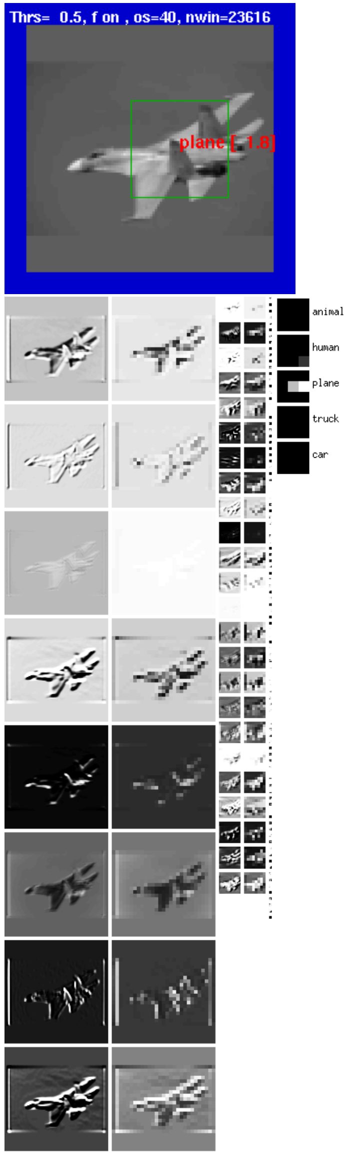


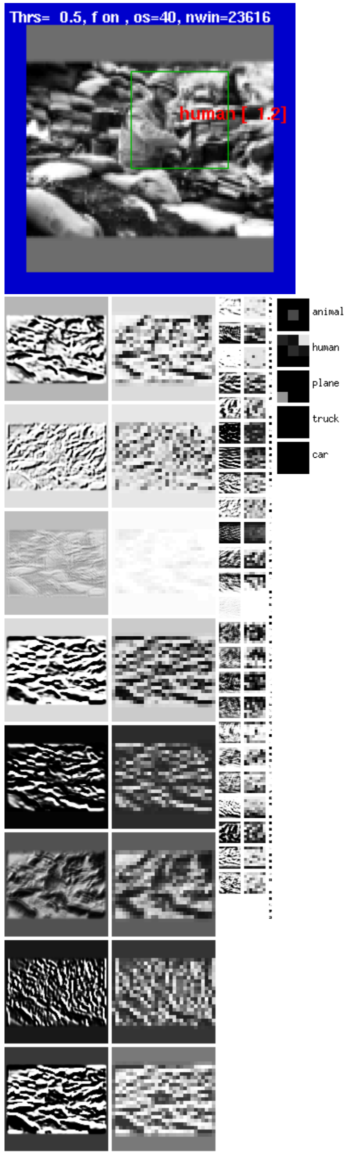
Examples (Monocular Mode)



Natural Images (Monocular Mode)

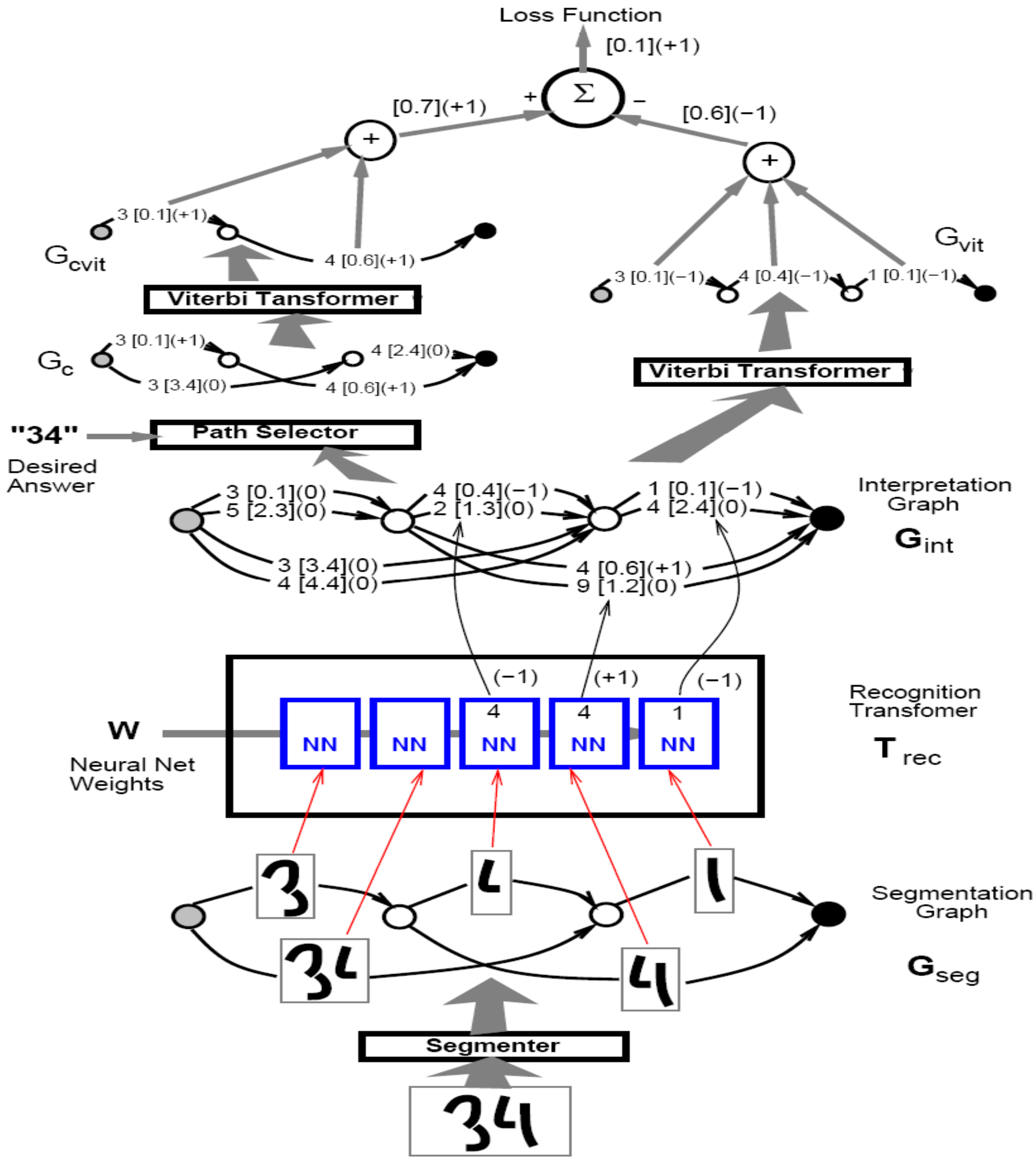






Supervised Convolutional Nets: Pros and Cons

- **Convolutional nets can be trained to perform a wide variety of visual tasks.**
  - Global supervised gradient descent can produce parsimonious architectures
- **BUT: they require lots of labeled training samples**
  - 60,000 samples for handwriting
  - 120,000 samples for face detection
  - 25,000 to 350,000 for object recognition
- **Since low-level features tend to be non task specific, we should be able to learn them unsupervised.**
- **Hinton has shown that layer-by-layer unsupervised "pre-training" can be used to initialize "deep" architectures**
  - [Hinton & Shalakhutdinov, Science 2006]
- **Can we use this idea to reduce the number of necessary labeled examples.**


Loss Function
[0.1](+1)
Σ
+
−
[0.7](+1)
[0.6](−1)
+
+
3 [0.1](+1)
4 [0.6](+1)
G_{cvit}
Viterbi Tansformer
G_{vit}
3 [0.1](−1)
4 [0.4](−1)
1 [0.1](−1)
G_c
3 [0.1](+1)
3 [3.4](0)
4 [2.4](0)
4 [0.6](+1)
Viterbi Transformer
"34"
Desired Answer
Path Selector
3 [0.1](0)
5 [2.3](0)
4 [0.4](−1)
2 [1.3](0)
1 [0.1](−1)
4 [2.4](0)
3 [3.4](0)
4 [4.4](0)
4 [0.6](+1)
9 [1.2](0)
Interpretation Graph
G_{int}
(−1)
(+1)
(−1)
W
Neural Net Weights
NN
NN
4 NN
4 NN
1 NN
Recognition Transfomer
T_{rec}
Segmentation Graph
G_{seg}
Segmenter

Learning with Large Datasets

Léon Bottou

NEC Laboratories America

or

How can bad optimization be good in large-scale settings

See http://leon.bottou.org/slides/largescale/lstut.pdf

Simple Analysis

- **Statistical Learning Literature:**

"It is good to optimize an objective function than ensures a fast estimation rate when the number of examples increases."

- **Optimization Literature:**

"To efficiently solve large problems, it is preferable to choose an optimization algorithm with strong asymptotic properties, e.g. superlinear."

- **Therefore:**

"To address large-scale learning problems, use a superlinear algorithm to optimize an objective function with fast estimation rate.
Problem solved."

The purpose of this presentation is. . .

Too Simple an Analysis

- **Statistical Learning Literature:**

"It is good to optimize an objective function than ensures a fast estimation rate when the number of examples increases."

- **Optimization Literature:**

"To efficiently solve large problems, it is preferable to choose an optimization algorithm with strong asymptotic properties, e.g. superlinear."

- **Therefore:** *(error)*

"To address large-scale learning problems, use a superlinear algorithm to optimize an objective function with fast estimation rate.
Problem solved."

... to show that this is completely wrong !

Objectives and Essential Remarks

- Baseline large-scale learning algorithm

Randomly discarding data is the simplest way to handle large datasets.

  – What are the statistical benefits of processing more data?
  – What is the computational cost of processing more data?

- We need a theory that joins Statistics and Computation!

– 1967: Vapnik's theory does not discuss computation.
– 1981: Valiant's learnability excludes exponential time algorithms, but (i) polynomial time can be too slow, (ii) few actual results.
– We propose a simple analysis of approximate optimization. . .

Learning Algorithms: Standard Framework

- Assumption: examples are drawn independently from an unknown probability distribution $P(x, y)$ that represents the rules of Nature.
- Expected Risk: $E(f) = \int \ell(f(x), y)\, dP(x, y)$.
- Empirical Risk: $E_n(f) = \frac{1}{n} \sum \ell(f(x_i), y_i)$.
- We would like f^* that minimizes $E(f)$ among all functions.
- In general $f^* \notin \mathcal{F}$.
- The best we can have is $f^*_{\mathcal{F}} \in \mathcal{F}$ that minimizes $E(f)$ inside $\mathcal{F}$.
- But $P(x, y)$ is unknown by definition.
- Instead we compute $f_n \in \mathcal{F}$ that minimizes $E_n(f)$.
  Vapnik-Chervonenkis theory tells us when this can work.

Learning with Approximate Optimization

Computing $f_n = \arg\min_{f \in \mathcal{F}} E_n(f)$ is often costly.

Since we already make lots of approximations,
why should we compute f_n exactly?

Let's assume our optimizer returns $\tilde{f}_n$
such that $E_n(\tilde{f}_n) < E_n(f_n) + \rho$.

For instance, one could stop an iterative
optimization algorithm long before its convergence.

Decomposition of the Error (i)

$$
\begin{aligned}
E(\tilde{f}_n) - E(f^*) &= E(f^*_{\mathcal{F}}) - E(f^*) && \text{Approximation error} \\
&+ E(f_n) - E(f^*_{\mathcal{F}}) && \text{Estimation error} \\
&+ E(\tilde{f}_n) - E(f_n) && \text{Optimization error}
\end{aligned}
$$

Problem:
Choose $\mathcal{F}$, n, and ρ to make this as small as possible,

subject to budget constraints $\begin{cases} \text{maximal number of examples } n \\ \text{maximal computing time } T \end{cases}$

Decomposition of the Error (ii)

Approximation error bound: (Approximation theory)
– decreases when $\mathcal{F}$ gets larger.

Estimation error bound: (Vapnik-Chervonenkis theory)
– decreases when n gets larger.
– increases when $\mathcal{F}$ gets larger.

Optimization error bound: (Vapnik-Chervonenkis theory plus tricks)
– increases with ρ.

Computing time T: (Algorithm dependent)
– decreases with ρ
– increases with n
– increases with $\mathcal{F}$

Small-scale vs. Large-scale Learning

We can give *rigorous definitions.*

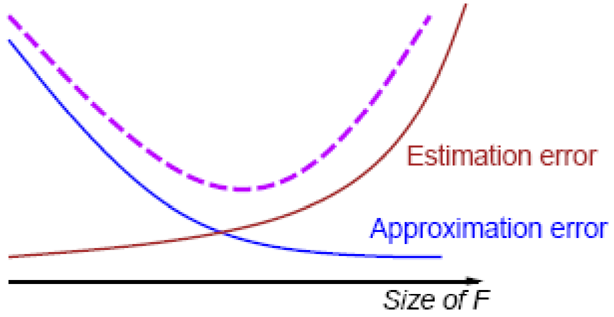
- **Definition 1:**
  We have a **small-scale learning** problem when the **active budget constraint is the number of examples n.**

- **Definition 2:**
  We have a **large-scale learning** problem when the **active budget constraint is the computing time T.**

Small-scale Learning

The active budget constraint is the number of examples.

- To reduce the estimation error, take n as large as the budget allows.
- To reduce the optimization error to zero, take $\rho = 0$.
- We need to adjust the size of $\mathcal{F}$.



See Structural Risk Minimization (Vapnik 74) and later works.

Large-scale Learning

The active budget constraint is the computing time.

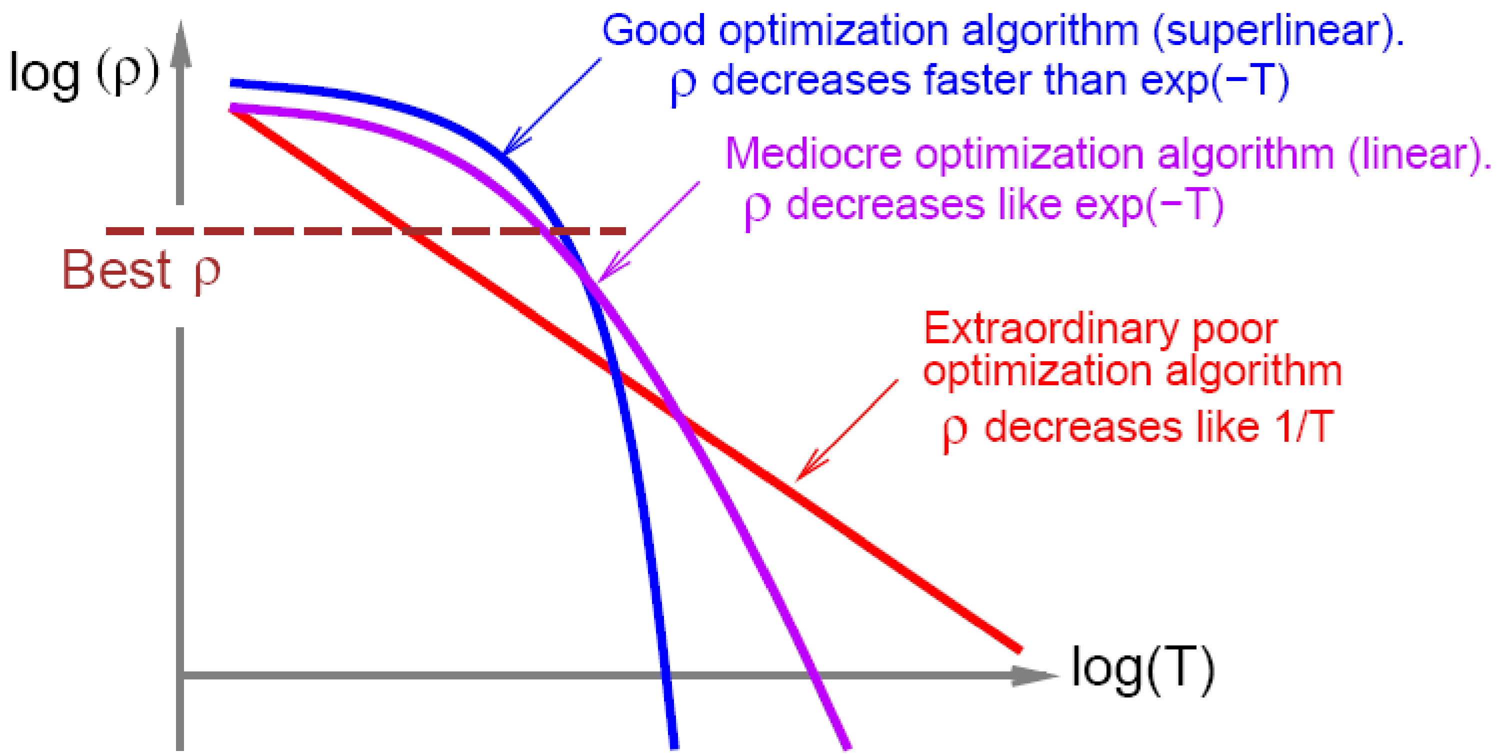
- More complicated tradeoffs.
  The computing time depends on the three variables: $\mathcal{F}$, n, and ρ.

- Example.
  If we choose ρ small, we decrease the optimization error. But we must also decrease $\mathcal{F}$ and/or n with adverse effects on the estimation and approximation errors.

- The exact tradeoff depends on the optimization algorithm.

- We can compare optimization algorithms rigorously.

Executive Summary



Asymptotics: Estimation

Uniform convergence bounds (with capacity $d+1$)

$$\text{Estimation error} \leq \mathcal{O}\left(\left[\frac{d}{n}\log\frac{n}{d}\right]^{\alpha}\right) \text{ with } \frac{1}{2} \leq \alpha \leq 1 \; .$$

There are in fact three types of bounds to consider:

- Classical V-C bounds (pessimistic): $\mathcal{O}\left(\sqrt{\frac{d}{n}}\right)$
- Relative V-C bounds in the realizable case: $\mathcal{O}\left(\frac{d}{n}\log\frac{n}{d}\right)$
- Localized bounds (variance, Tsybakov): $\mathcal{O}\left(\left[\frac{d}{n}\log\frac{n}{d}\right]^{\alpha}\right)$

Fast estimation rates are a big theoretical topic these days.

Asymptotics: Estimation+Optimization

statistical estimation rate

Uniform convergence arguments give

$$\text{Estimation error} + \text{Optimization error} \leq \mathcal{O}\left(\left[\frac{d}{n}\log\frac{n}{d}\right]^{\alpha} + \rho\right).$$

This is true for all three cases of uniform convergence bounds.

Scaling laws for ρ when $\mathcal{F}$ is fixed

The approximation error is constant.

- No need to choose ρ smaller than $\mathcal{O}\left(\left[\frac{d}{n}\log\frac{n}{d}\right]^{\alpha}\right)$.
- Not advisable to choose ρ larger than $\mathcal{O}\left(\left[\frac{d}{n}\log\frac{n}{d}\right]^{\alpha}\right)$.

. . . Approximation+Estimation+Optimization

When $\mathcal{F}$ is chosen via a λ-regularized cost

- Uniform convergence theory provides bounds for simple cases
  (Massart-2000; Zhang 2005; Steinwart et al., 2004-2007; . . .)
- Computing time depends on both λ and ρ.
- Scaling laws for λ and ρ depend on the optimization algorithm.

When $\mathcal{F}$ is realistically complicated

Large datasets matter

- because one can use more features,
- because one can use richer models.

Bounds for such cases are rarely realistic enough.

Luckily there are interesting things to say for $\mathcal{F}$ fixed.

Case Study

Simple parametric setup

– $\mathcal{F}$ is fixed.
– Functions $f_w(x)$ linearly parametrized by $w \in \mathbb{R}^d$.

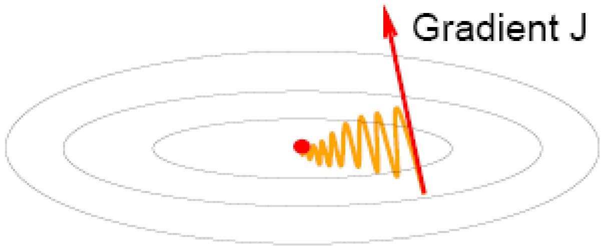
Comparing four iterative optimization algorithms for $E_n(f)$

1. Gradient descent.
2. Second order gradient descent (Newton).
3. Stochastic gradient descent.
4. Stochastic second order gradient descent.

Gradient Descent (GD)

Iterate

- $w_{t+1} \leftarrow w_t - \eta \dfrac{\partial E_n(f_{w_t})}{\partial w}$



Best speed achieved with fixed learning rate $\eta = \frac{1}{\lambda_{\max}}$.
(e.g., Dennis & Schnabel, 1983)

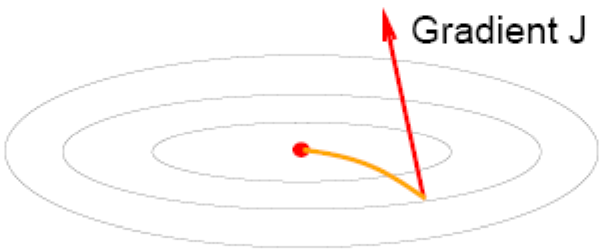
	Cost per iteration	Iterations to reach ρ	Time to reach accuracy ρ	Time to reach $E(\tilde{f}_n) - E(f^*_{\mathcal{F}}) < \varepsilon$
GD	$\mathcal{O}(nd)$	$\mathcal{O}\left(\kappa \log \frac{1}{\rho}\right)$	$\mathcal{O}\left(nd\kappa \log \frac{1}{\rho}\right)$	$\mathcal{O}\left(\frac{d^2\,\kappa}{\varepsilon^{1/\alpha}} \log^2 \frac{1}{\varepsilon}\right)$

– In the last column, n and ρ are chosen to reach ε as fast as possible.
– Solve for ε to find the best error rate achievable in a given time.
– Remark: abuses of the $\mathcal{O}()$ notation

Second Order Gradient Descent (2GD)

Iterate

- $w_{t+1} \leftarrow w_t - H^{-1} \frac{\partial E_n(f_{w_t})}{\partial w}$



We assume H^{-1} is known in advance.
Superlinear optimization speed (e.g., Dennis & Schnabel, 1983)

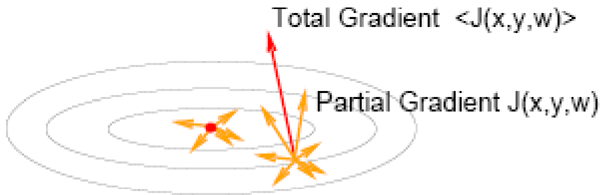
	Cost per iteration	Iterations to reach ρ	Time to reach accuracy ρ	Time to reach $E(\tilde{f}_n) - E(f^*_{\mathcal{F}}) < \varepsilon$
2GD	$\mathcal{O}(d(d+n))$	$\mathcal{O}\left(\log\log\frac{1}{\rho}\right)$	$\mathcal{O}\left(d(d+n)\log\log\frac{1}{\rho}\right)$	$\mathcal{O}\left(\frac{d^2}{\varepsilon^{1/\alpha}}\log\frac{1}{\varepsilon}\log\log\frac{1}{\varepsilon}\right)$

– Optimization speed is much faster.
– Learning speed only saves the condition number κ.

Stochastic Gradient Descent (SGD)

Iterate

- Draw random example (x_t, y_t).
- $w_{t+1} \leftarrow w_t - \dfrac{\eta}{t} \dfrac{\partial \ell(f_{w_t}(x_t), y_t)}{\partial w}$



Best decreasing gain schedule with $\eta = \frac{1}{\lambda_{\min}}$.
(see Murata, 1998; Bottou & LeCun, 2004)

	Cost per iteration	Iterations to reach ρ	Time to reach accuracy ρ	Time to reach $E(\tilde{f}_n) - E(f^*_{\mathcal{F}}) < \varepsilon$
SGD	$\mathcal{O}(d)$	$\frac{\nu k}{\rho} + o\left(\frac{1}{\rho}\right)$	$\mathcal{O}\left(\frac{d\,\nu\,k}{\rho}\right)$	$\mathcal{O}\left(\frac{d\,\nu\,k}{\varepsilon}\right)$

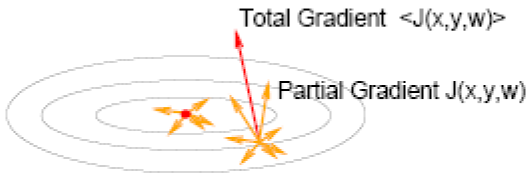
With $1 \leq k \leq \kappa^2$

– Optimization speed is *catastrophic*.
– Learning speed does not depend on the statistical estimation rate α.
– Learning speed depends on condition number κ but *scales very well*.

Second order Stochastic Descent (2SGD)

Iterate

- Draw random example (x_t, y_t).
- $w_{t+1} \leftarrow w_t - \frac{1}{t} H^{-1} \frac{\partial \ell(f_{w_t}(x_t), y_t)}{\partial w}$



Replace scalar gain $\frac{\eta}{t}$ by matrix $\frac{1}{t} H^{-1}$.

	Cost per iteration	Iterations to reach ρ	Time to reach accuracy ρ	Time to reach $E(\tilde{f}_n) - E(f^*_{\mathcal{F}}) < \varepsilon$
2SGD	$\mathcal{O}(d^2)$	$\frac{\nu}{\rho} + o\left(\frac{1}{\rho}\right)$	$\mathcal{O}\left(\frac{d^2\,\nu}{\rho}\right)$	$\mathcal{O}\left(\frac{d^2\,\nu}{\varepsilon}\right)$

– Each iteration is d times more expensive.
– The number of iterations is reduced by κ^2 (or less.)
– Second order only changes the constant factors.

Summary

Algorithm	Cost of one iteration	Iterations to reach ρ	Time to reach accuracy ρ	Time to reach $\mathcal{E} \leq c\,(\mathcal{E}_{\mathrm{app}} + \varepsilon)$
GD	$\mathcal{O}(nd)$	$\mathcal{O}\left(\kappa \log \frac{1}{\rho}\right)$	$\mathcal{O}\left(nd\kappa \log \frac{1}{\rho}\right)$	$\mathcal{O}\left(\frac{d^2\,\kappa}{\varepsilon^{1/\alpha}} \log^2 \frac{1}{\varepsilon}\right)$
2GD	$\mathcal{O}\left(d^2 + nd\right)$	$\mathcal{O}\left(\log\log \frac{1}{\rho}\right)$	$\mathcal{O}\left(\left(d^2 + nd\right) \log\log \frac{1}{\rho}\right)$	$\mathcal{O}\left(\frac{d^2}{\varepsilon^{1/\alpha}} \log \frac{1}{\varepsilon} \log\log \frac{1}{\varepsilon}\right)$
SGD	$\mathcal{O}(d)$	$\frac{\nu\kappa^2}{\rho} + \mathrm{o}\left(\frac{1}{\rho}\right)$	$\mathcal{O}\left(\frac{d\nu\kappa^2}{\rho}\right)$	$\mathcal{O}\left(\frac{d\,\nu\,\kappa^2}{\varepsilon}\right)$
2SGD	$\mathcal{O}\left(d^2\right)$	$\frac{\nu}{\rho} + \mathrm{o}\left(\frac{1}{\rho}\right)$	$\mathcal{O}\left(\frac{d^2\nu}{\rho}\right)$	$\mathcal{O}\left(\frac{d^2\,\nu}{\varepsilon}\right)$

Benchmarking SGD in Simple Problems

- The theory suggests that SGD is very competitive.
  - Many people associate SGD with trouble.
- SGD historically associated with back-propagation.
  - Multilayer networks are very hard problems (nonlinear, nonconvex)
  - What is difficult, SGD or MLP?

- **Try PLAIN SGD on simple learning problems.**
  - Support Vector Machines
  - Conditional Random Fields

Download from `http://leon.bottou.org/projects/sgd`.
These simple programs are very short.

See also (Shalev-Schwartz et al., 2007; Vishwanathan et al., 2006)

Text Categorization with SVMs

- **Dataset**
  - Reuters RCV1 document corpus.
  - 781,265 training examples, 23,149 testing examples.
  - 47,152 TF-IDF features.
- **Task**
  - Recognizing documents of category `CCAT`.
  - Minimize $E_n = \frac{1}{n}\sum_i \left(\frac{\lambda}{2} w^2 + \ell(w\, x_i + b,\, y_i) \right)$.
  - Update $w \leftarrow w - \eta_t \nabla(w_t, x_t, y_t) \quad = \quad w - \eta_t \left(\lambda w + \frac{\partial \ell(w\, x_t + b,\, y_t)}{\partial w} \right)$

Same setup as (Shalev-Schwartz et al., 2007) but plain SGD.

Text Categorization with SVMs

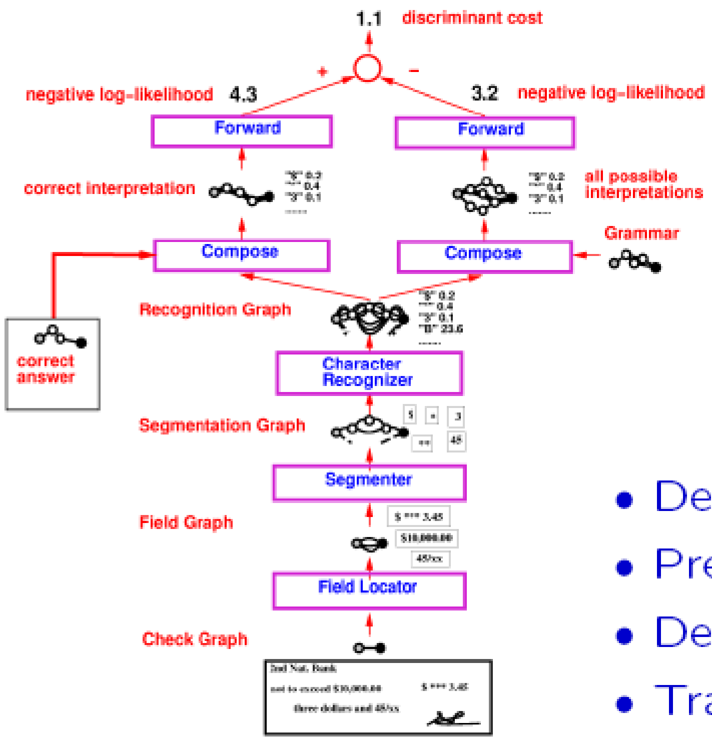
- **Results: Linear SVM**
  $\ell(\hat{y}, y) = \max\{0, 1 - y\hat{y}\} \quad \lambda = 0.0001$

	Training Time	Primal cost	Test Error
SVMLight	23,642 secs	*0.2275*	*6.02%*
SVMPerf	66 secs	0.2278	6.03%
SGD	1.4 secs	*0.2275*	*6.02%*

- **Results: Log-Loss Classifier**
  $\ell(\hat{y}, y) = log(1 + exp(-y\hat{y})) \quad \lambda = 0.00001$

	Training Time	Primal cost	Test Error
LibLinear ($\varepsilon = 0.01$)	30 secs	0.18907	*5.68%*
LibLinear ($\varepsilon = 0.001$)	44 secs	*0.18890*	5.70%
SGD	2.3 secs	0.18893	*5.66%*

SGD for Real Life Applications



A Check Reader

Examples are pairs (image,amount).

Problem with strong structure:

– Field segmentation

– Character segmentation

– Character recognition

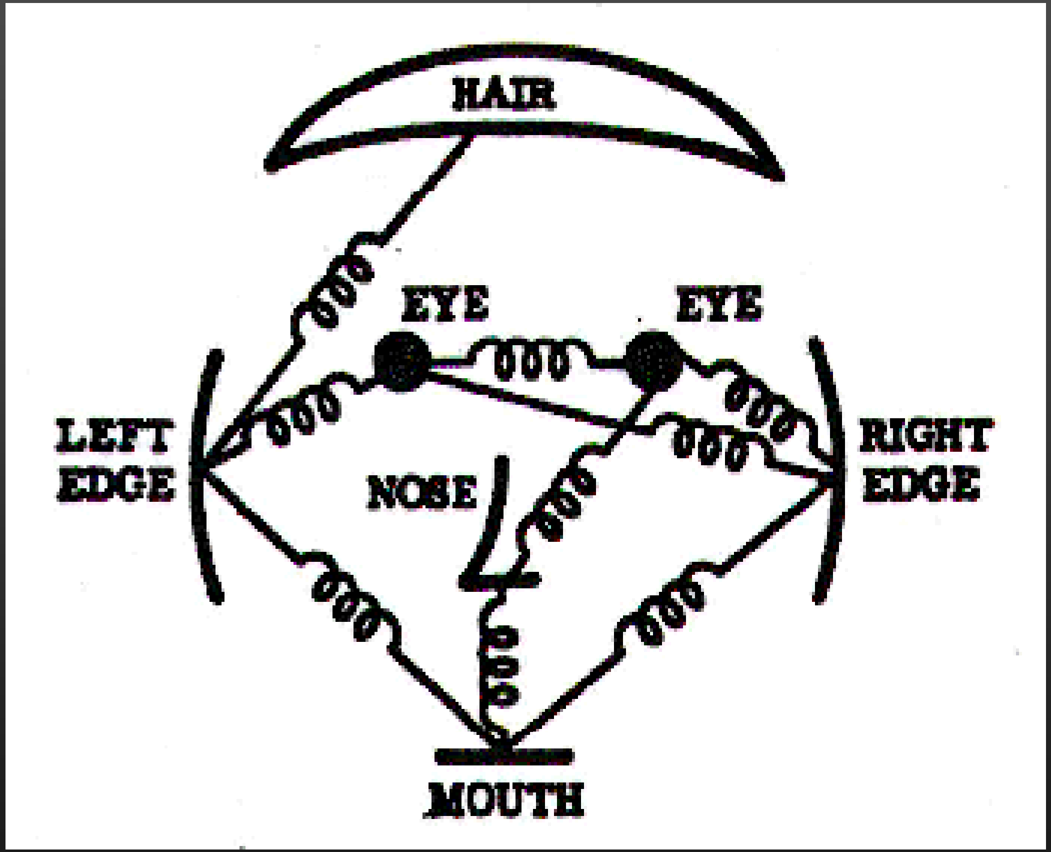
– Syntactical interpretation.

- Define differentiable modules.
- Pretrain modules with hand-labelled data.
- Define global cost function (e.g., CRF).
- Train with SGD for a few weeks.

Industrially deployed in 1996. Ran billions of checks over 10 years.

Credits: Bengio, Bottou, Burges, Haffner, LeCun, Nohl, Simard, et al.

Generative part-based models



Fischler & Elschlager'73

Many slides adapted from Svetlana Lazebnik, Fei-Fei Li, Rob Fergus, and Antonio Torralba

Bayesian approach

- Model: $P_\theta (f | c)$
- Learn the model by maximizing the likelihood of the training data

  $\max_\theta \sum_{\kappa=1}^{n} \log P_\theta (f_k | c)$
- Recognize using Bayes rule

  $P_\theta (c | f) = P_\theta (f | c) P(c) / P(f)$

R. Fergus, P. Perona and A. Zisserman, **Object Class Recognition by Unsupervised Scale-Invariant Learning**, CVPR 2003

Probabilistic model

$$P(image \mid object) = P(appearance, shape \mid object)$$

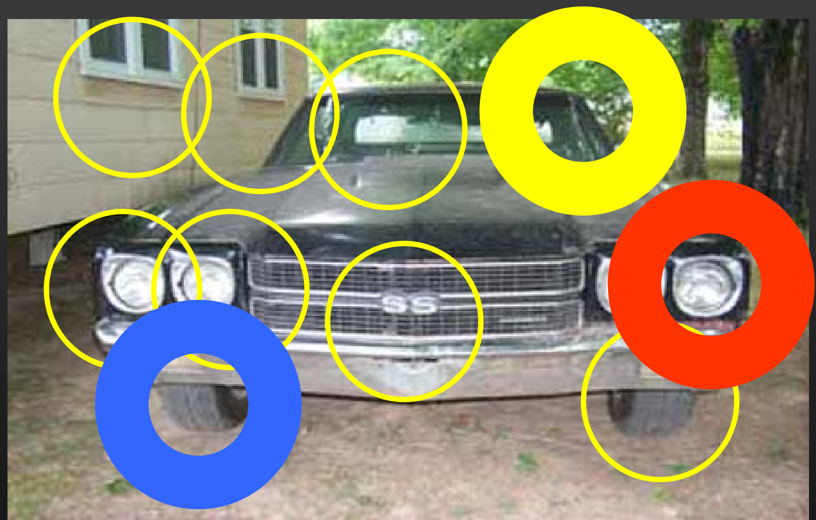
Part descriptors

Part locations

Candidate parts

Probabilistic model

$$P(image \mid object) = P(appearance, shape \mid object)$$



Probabilistic model

$$P(image \mid object) = P(appearance, shape \mid object)$$

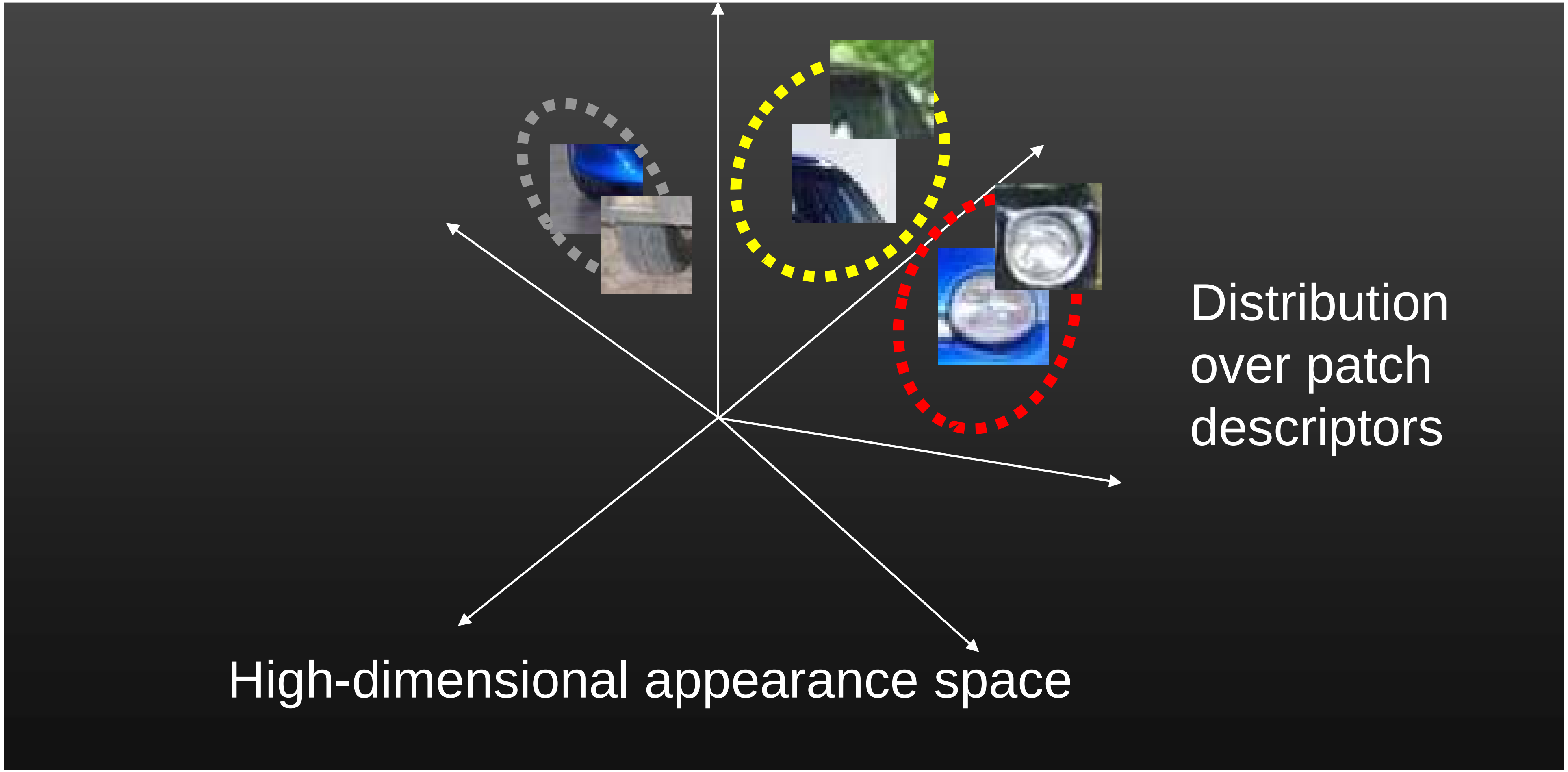
$$= \max_h P(appearance \mid h, object)\, p(shape \mid h, object)\, p(h \mid object)$$

h: assignment of features to parts

Probabilistic model

$$P(image \mid object) = P(appearance, shape \mid object)$$

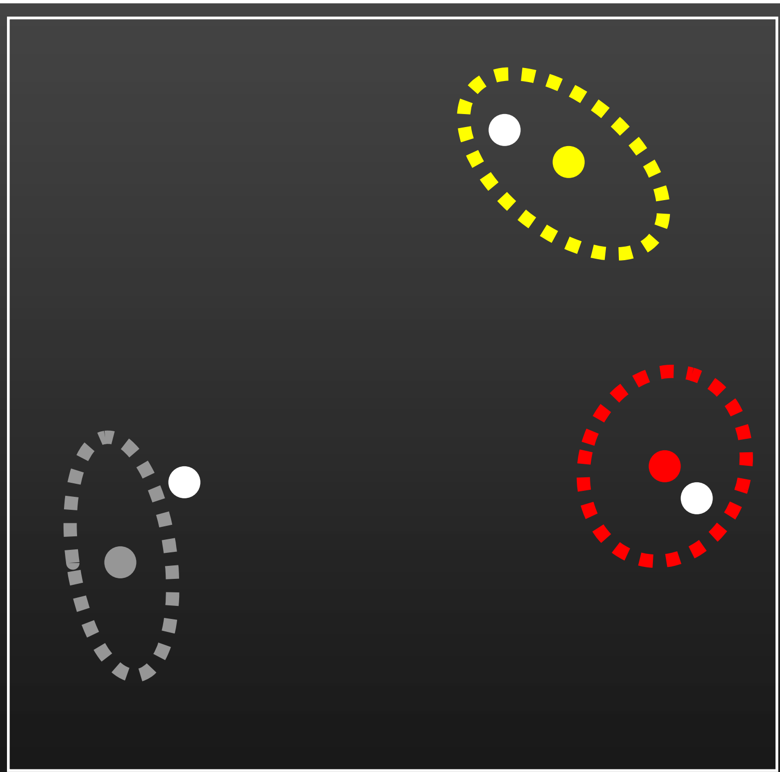
$$= \max_h \boxed{P(appearance \mid h, object)}\, p(shape \mid h, object)\, p(h \mid object)$$



Probabilistic model

$$P(image \mid object) = P(appearance, shape \mid object)$$

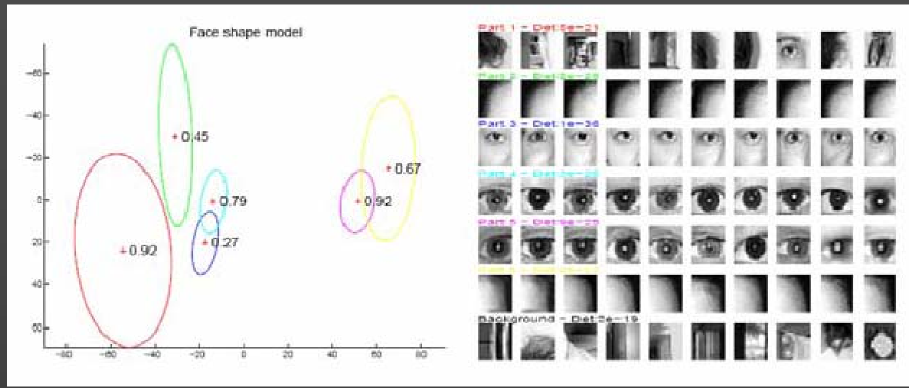
$$= \max_h P(appearance \mid h, object)\, \boxed{p(shape \mid h, object)}\, p(h \mid object)$$



Distribution over joint part positions

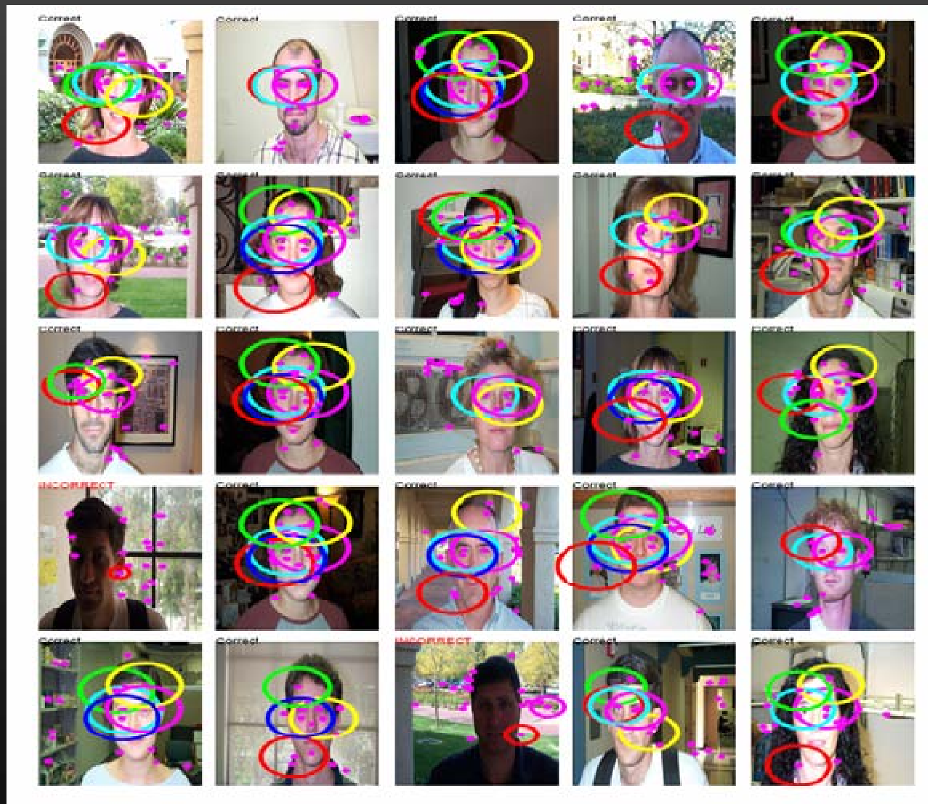
2D image space

Face shape model

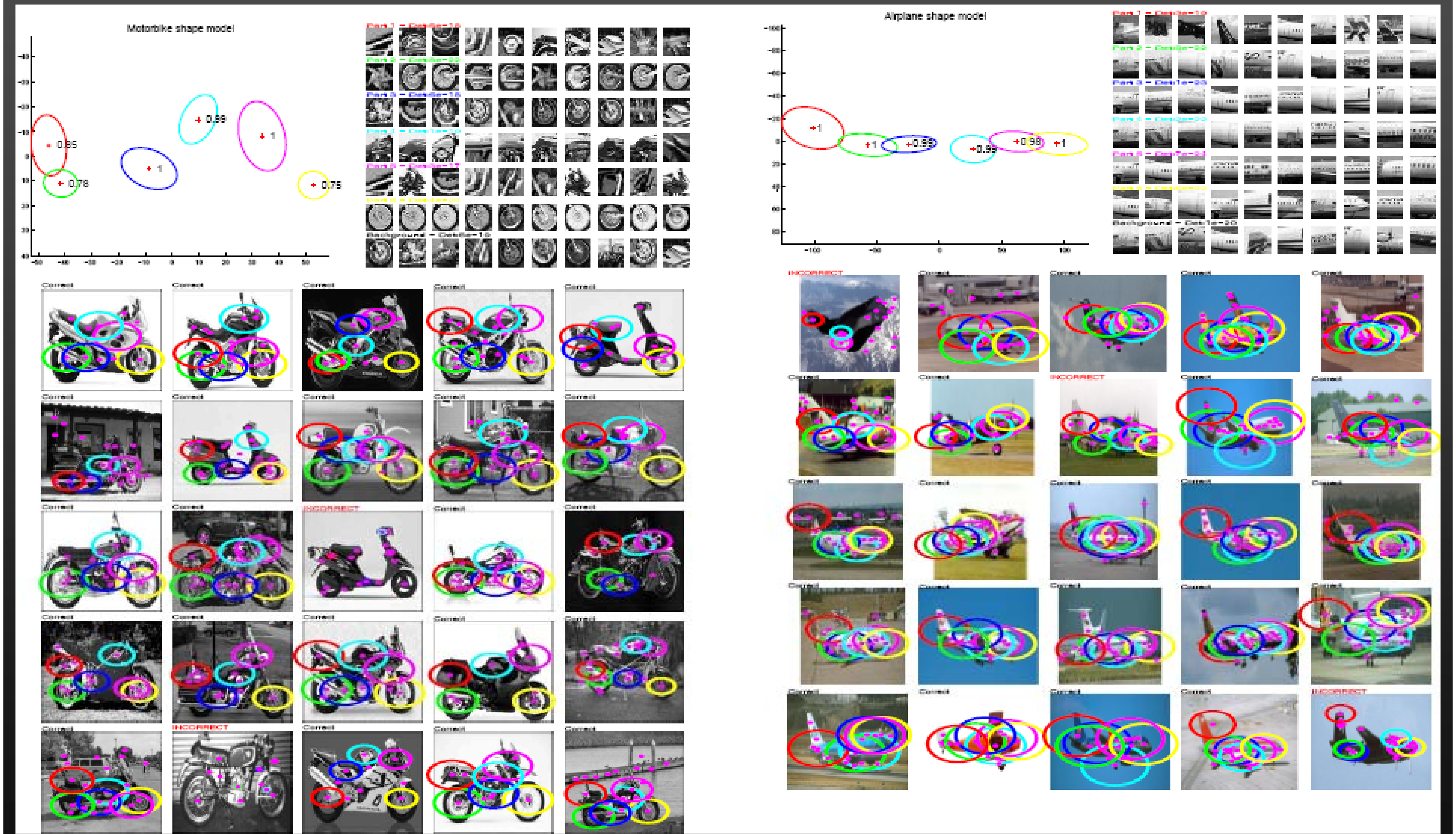


Patch appearance model

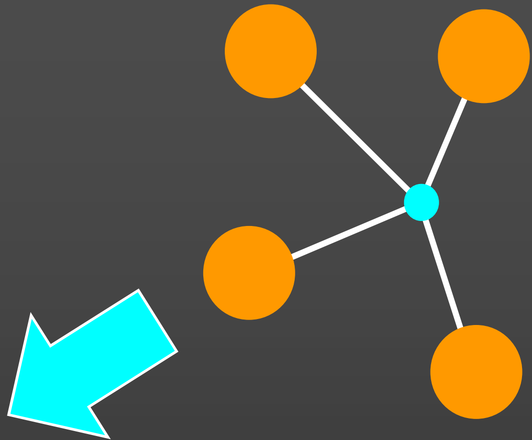
Recognition results

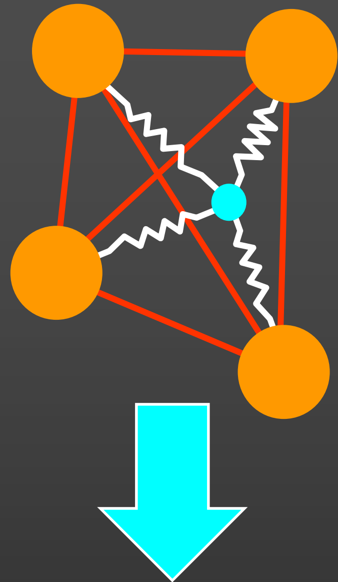


Results: Motorbikes and airplanes

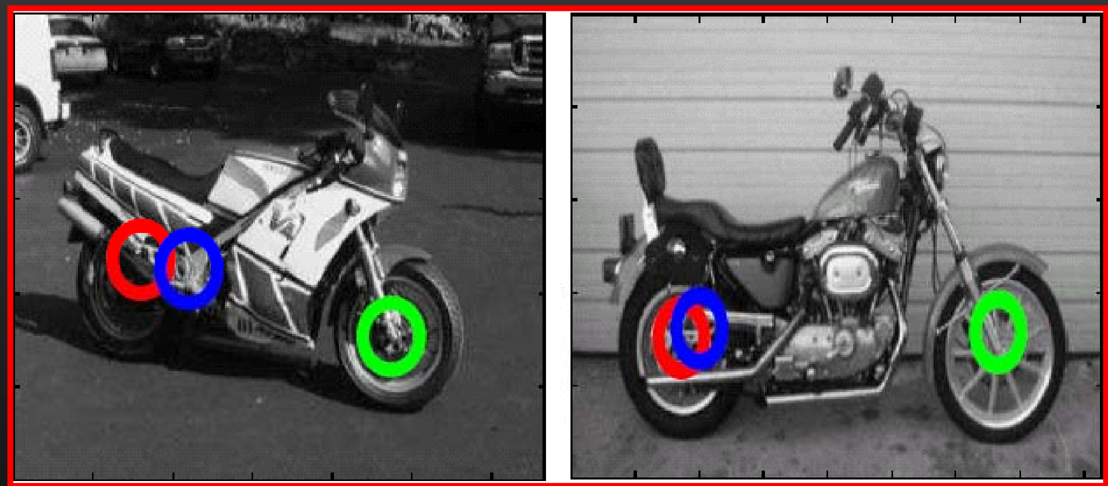


Note: The Fergus part-based model is very rigid





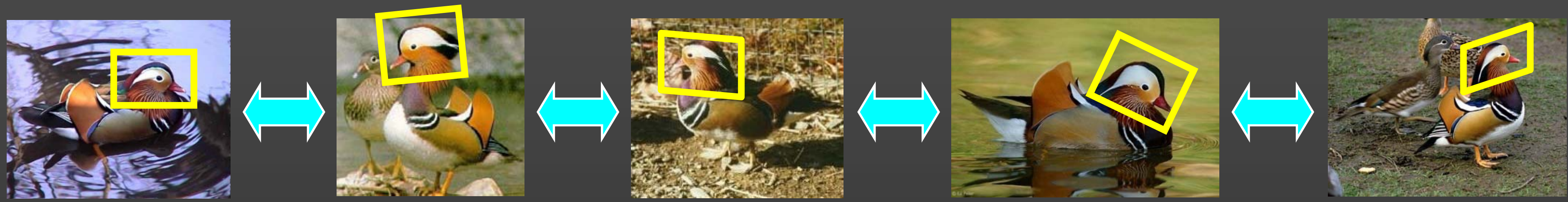
(Schmid & Mohr, 1996)
(Lowe, 1999)



(Fergus, Perona & Zisserman, 2003)

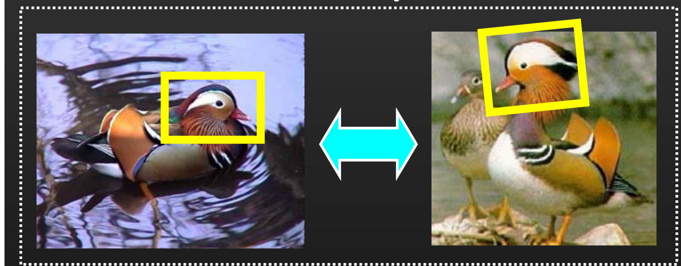
Model Learning as Multi-Image Segmentation

(Lazebnik, Scmid, Ponce, BMVC'04)



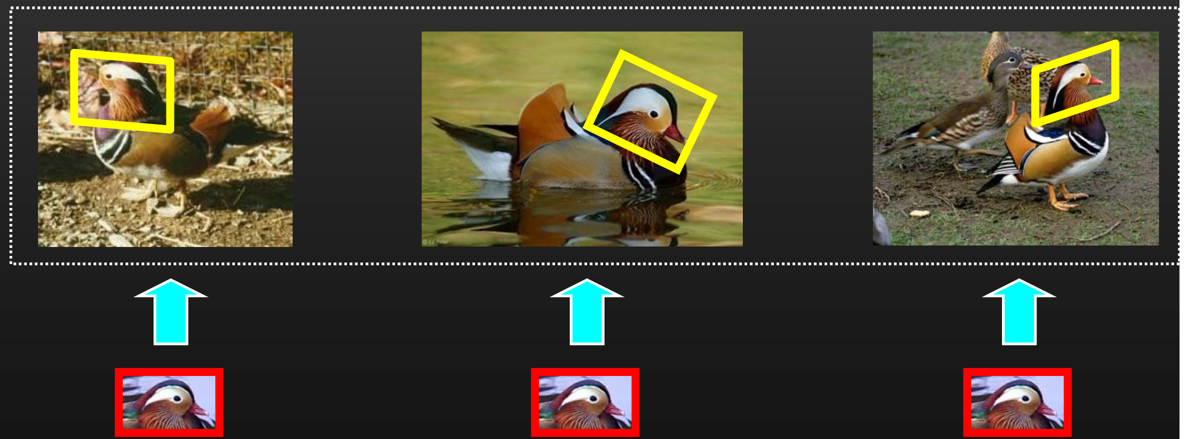
Practical approach: two-image matching followed by validation

initial pair



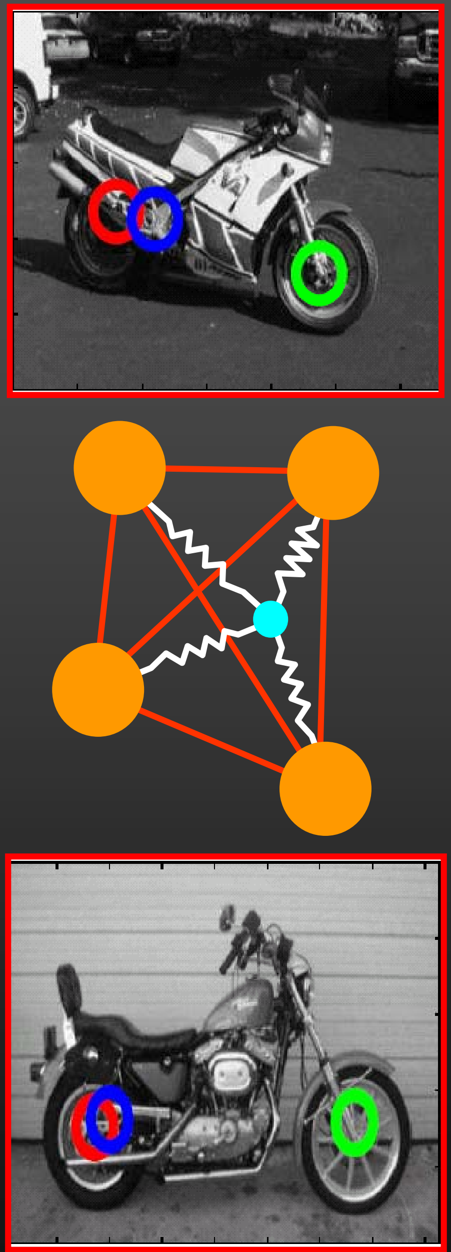
candidate part

validation set

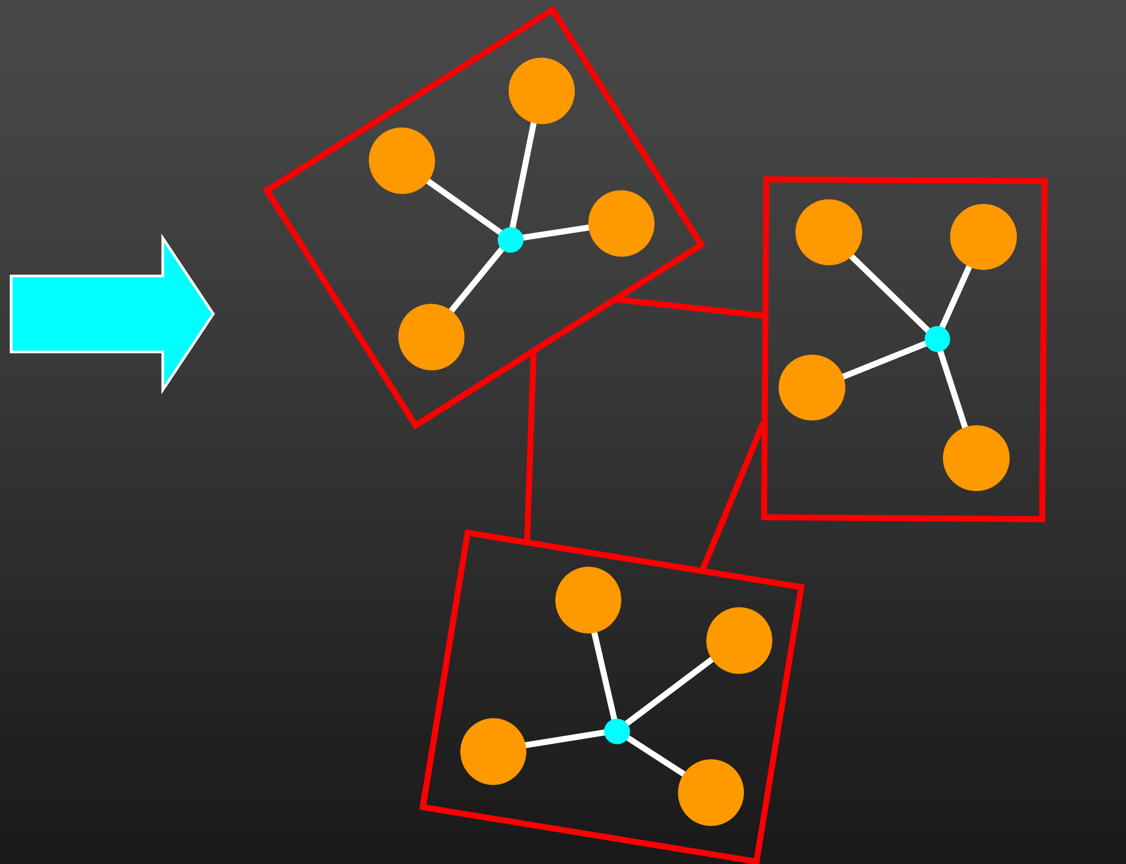


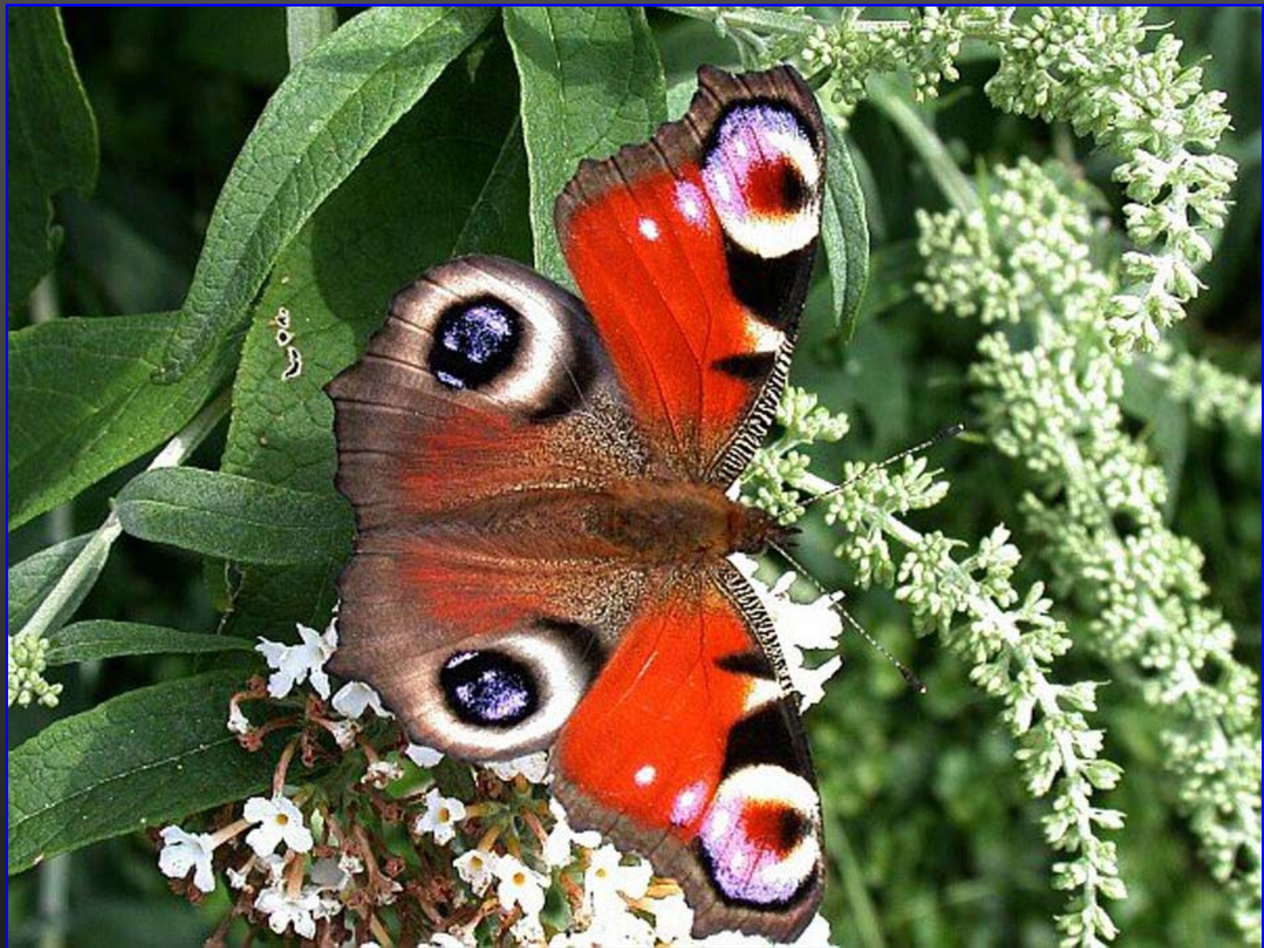
Model ≡ loose assembly of parts
Part ≡ rigid assembly of features

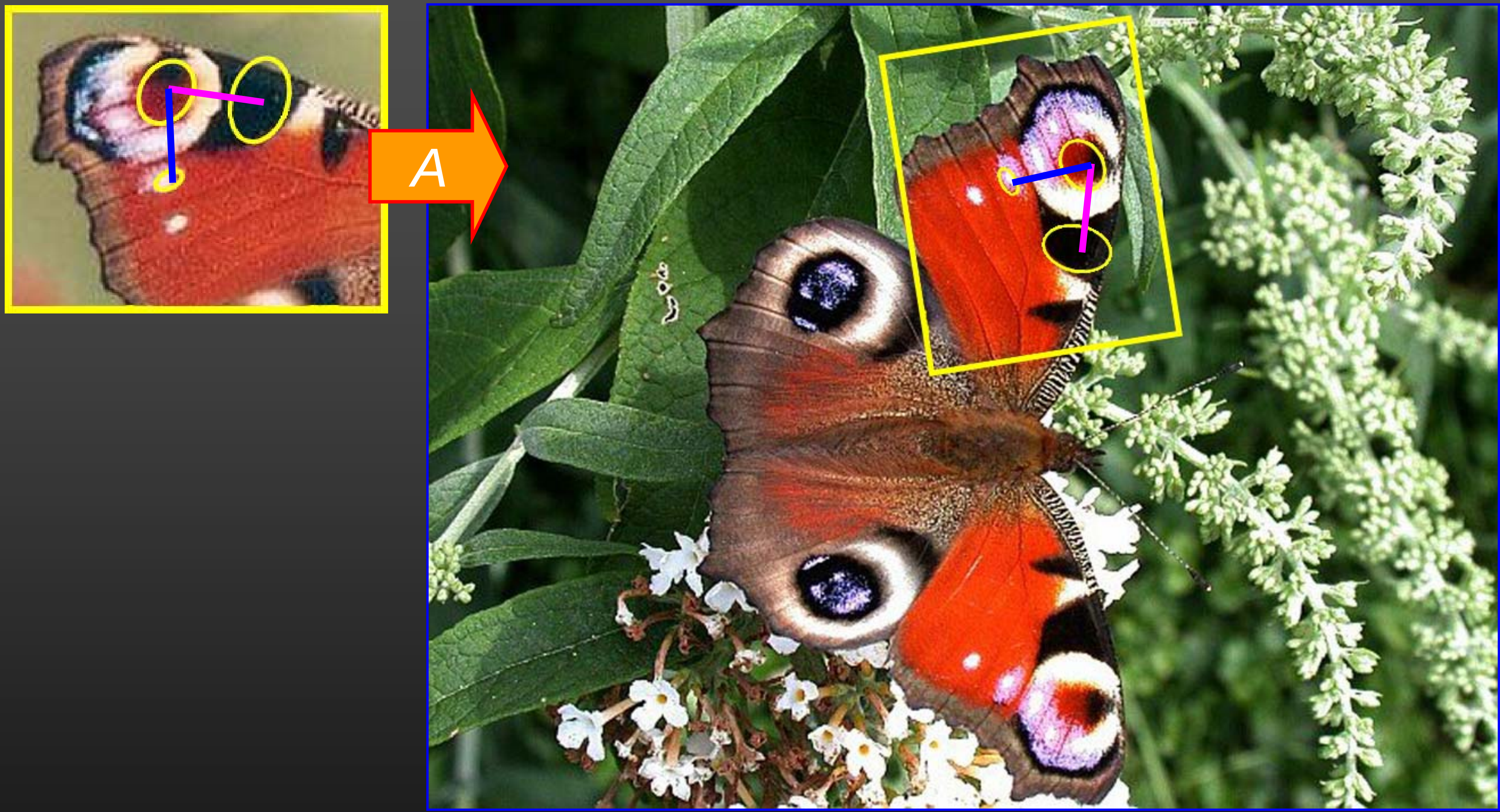
(Lazebnik, Ponce, Schmid, ICCV'O5)



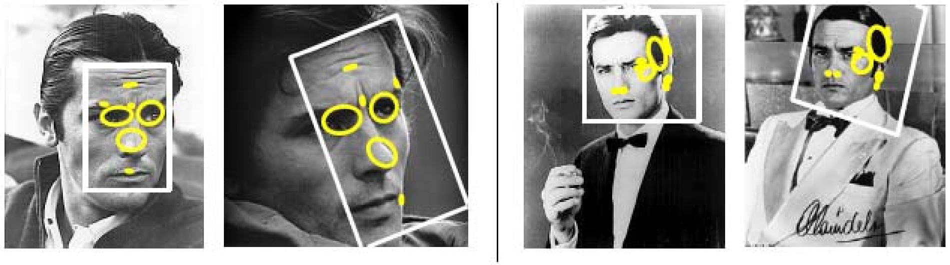
(Fergus et al., 2003)



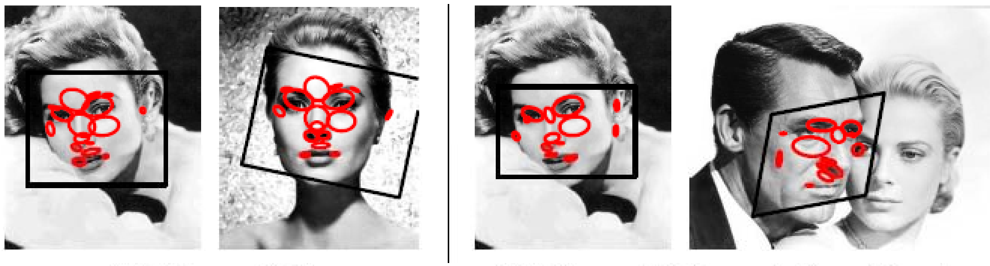





A

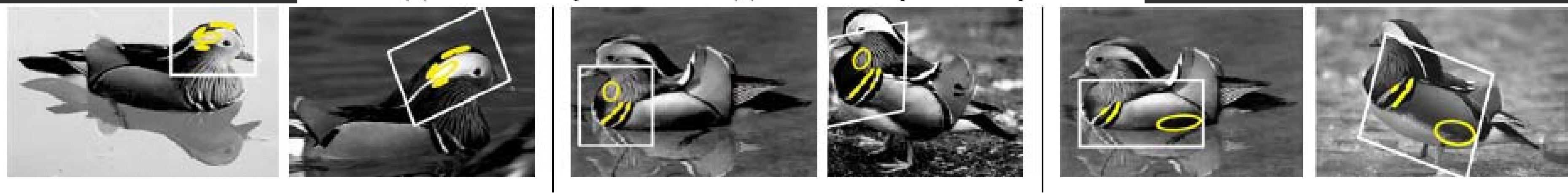
(Gaston, Grimson, & Lozano-Perez, 1982; Ayache & Fauger
(Faugeras & Hebert, 1983; Huttenlocher, 1987)



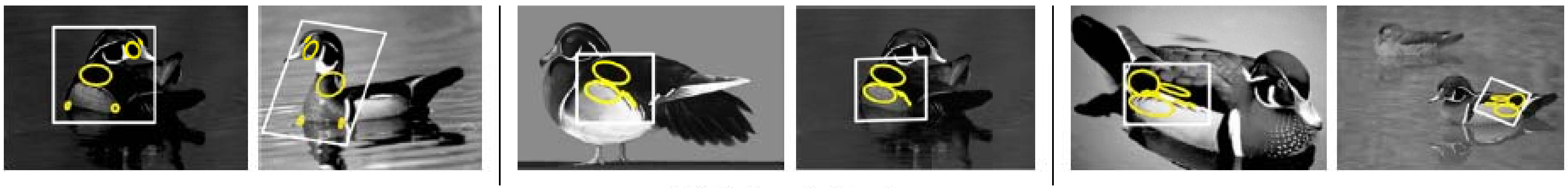
(a) Alain Delon



(b) Grace Kelly (c) Grace Kelly and Cary Grant



(a) Mandarin duck



(b) Wood duck

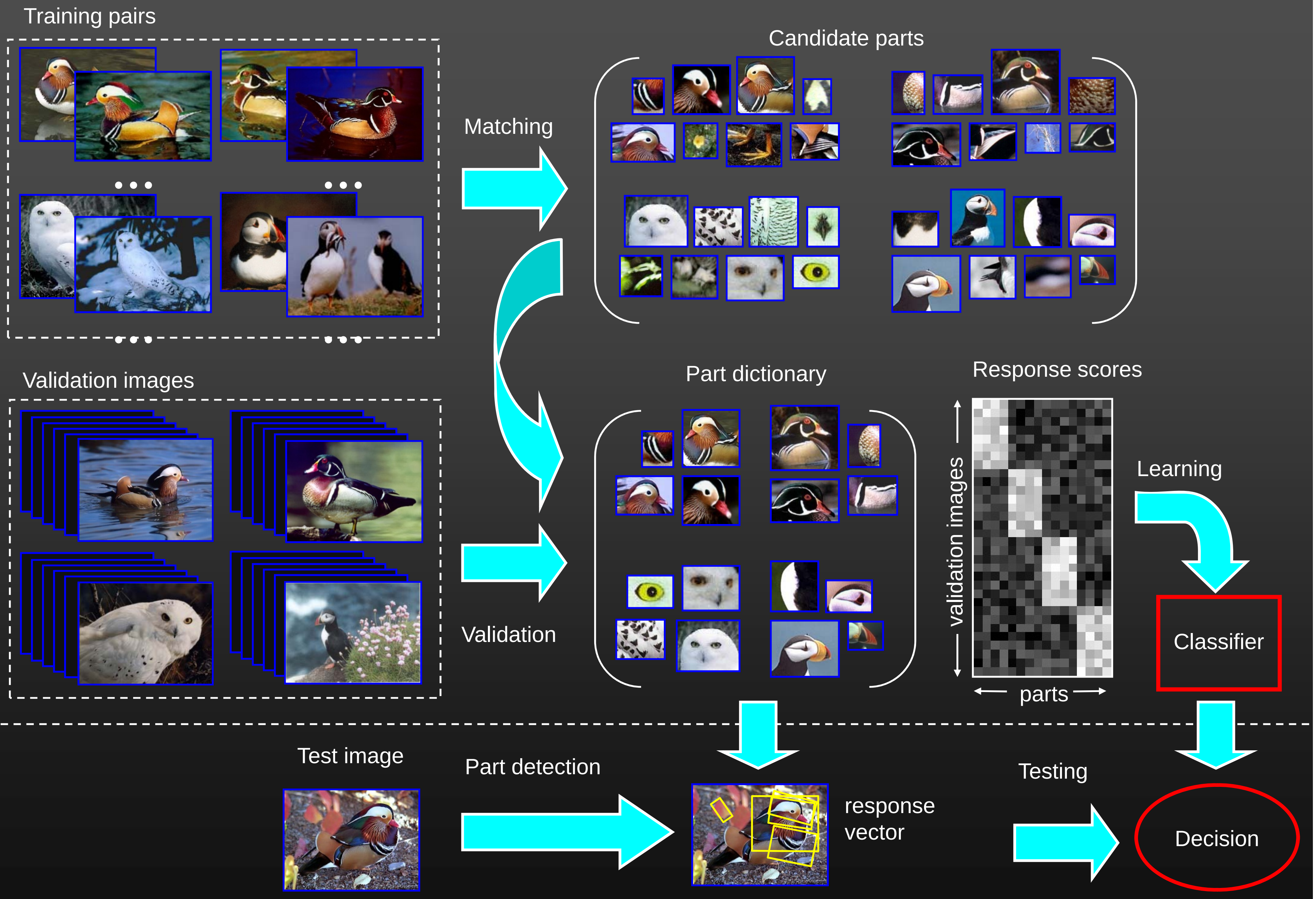
Discriminative approach

- Model: $P_\theta (c \mid f)$
- Learn the model by maximizing the likelihood of the training data

  $\max_\theta \sum_{\kappa=1}^{n} \log P_\theta (c_k \mid f_k)$
- Recognize by maximizing posterior probability of class

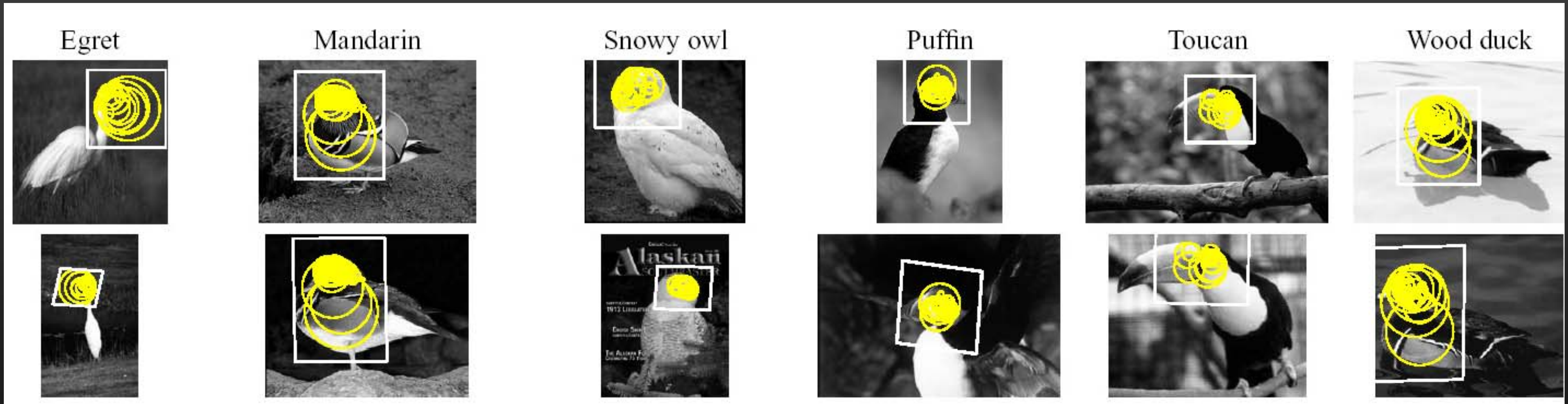
  $\max_c P_\theta (c \mid f)$

Complete Object Recognition System (ICCV'05)



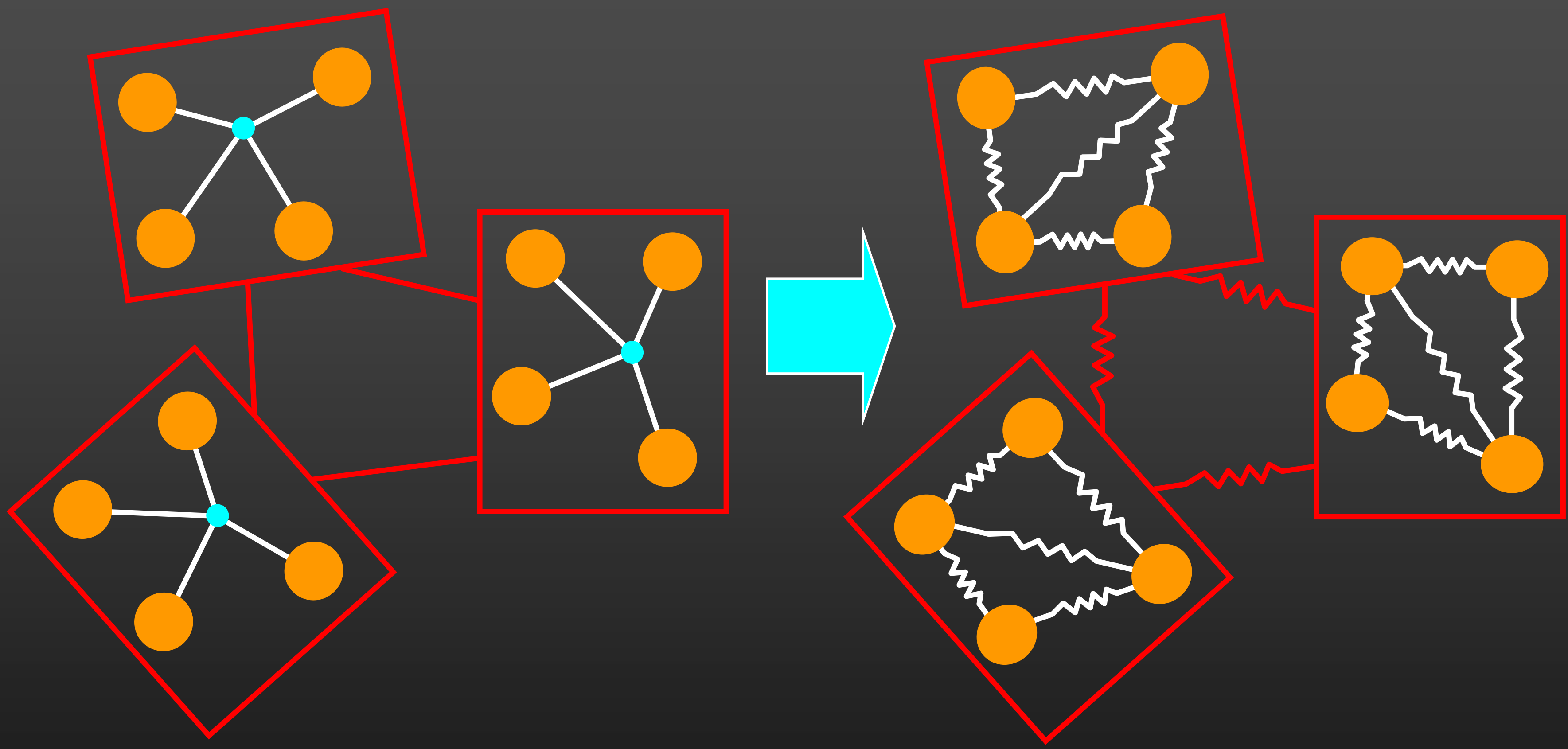
UIUC Bird Database

- 50 training images per class:
  - –20 initial images (50 largest candidate parts retained);
  - –30 validation (20 highest-scoring parts retained).
- 50 test images per class.
- 100 total.



Overall classification rate: 92.33%
Bag of features (Zhang et al., 2005): 83%

Model ≡ locally rigid assembly of parts
Part ≡ locally rigid assembly of features



A first attempt at handling:
(Kushal, Schmid, Ponce, 2006)

- changes in viewpoint
- nonrigid shape
- noncharacteristic texture

Model $\equiv$ locally rigid assembly of parts
Part $\equiv$ locally rigid assembly of features

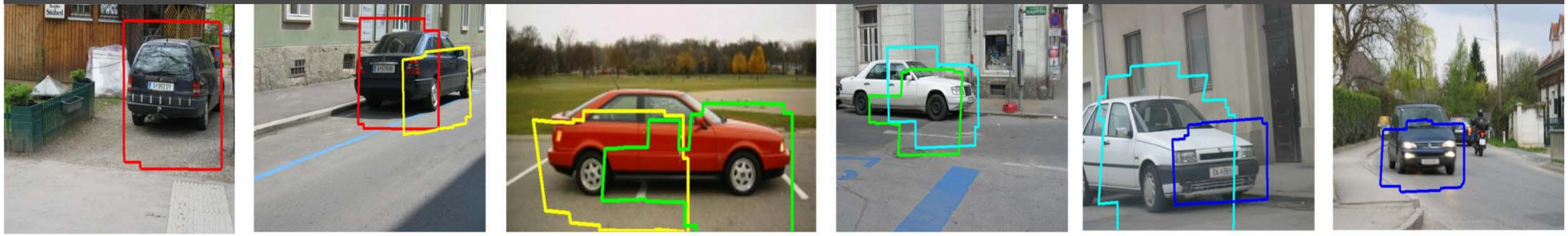
base images validation images

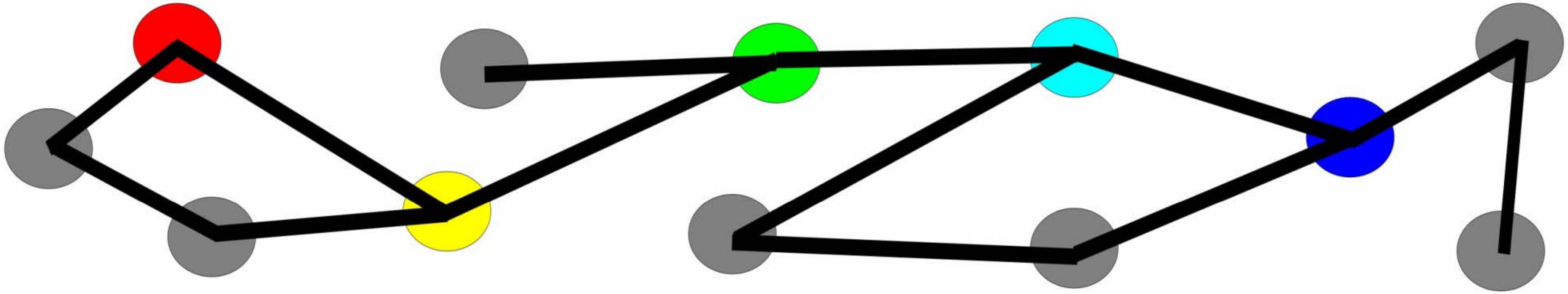
A first attempt at handling:

(Kushal, Schmid, Ponce, 2006)

- *changes in viewpoint*
- *nonrigid shape*
- *noncharacteristic texture*

Model ≡ locally rigid assembly of parts
Part ≡ locally rigid assembly of features



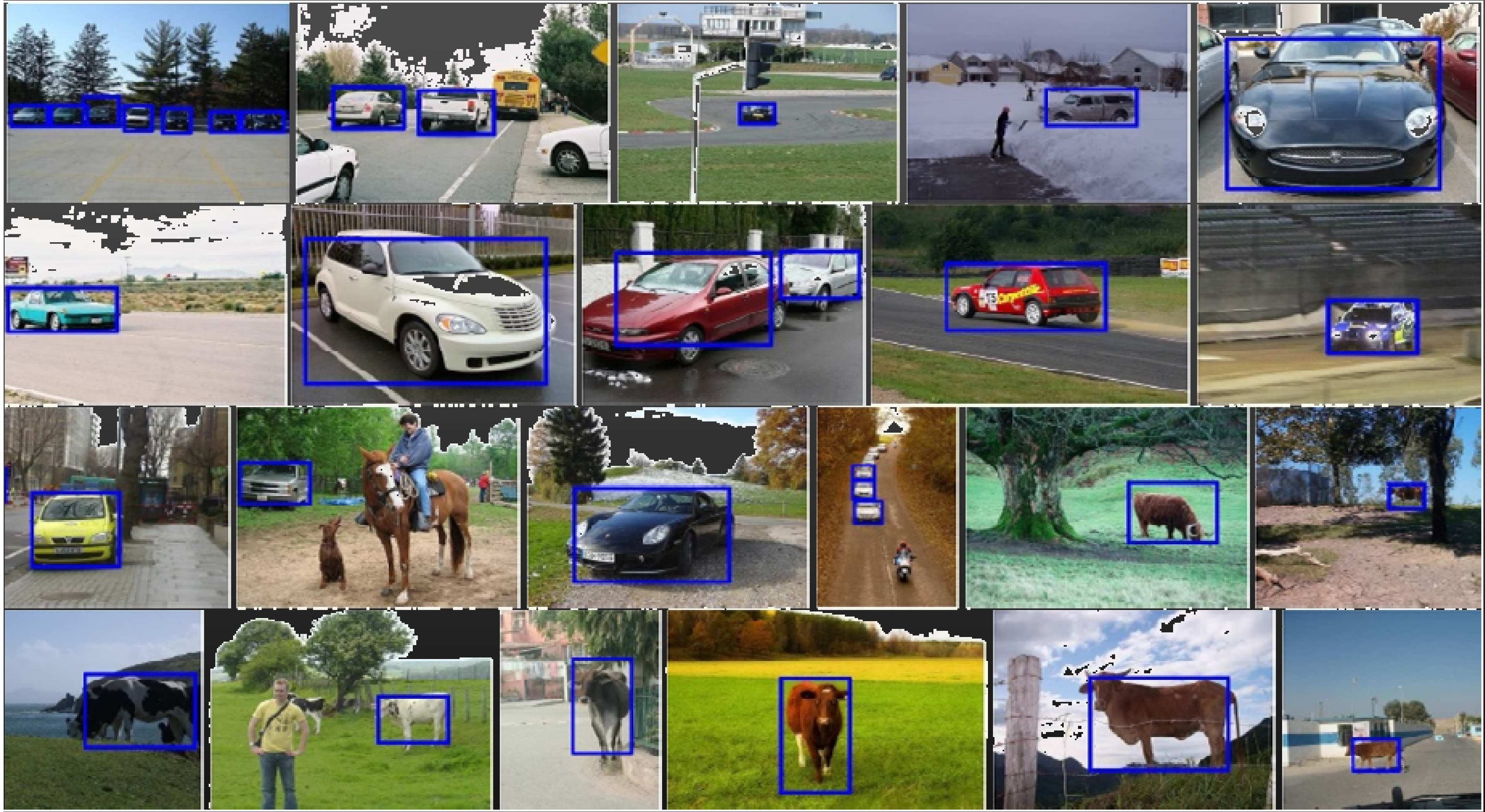


A first attempt at handling:

(Kushal, Schmid, Ponce, 2006)

- changes in viewpoint
- nonrigid shape
- noncharacteristic texture

Model ≡ locally rigid assembly of parts
Part ≡ locally rigid assembly of features

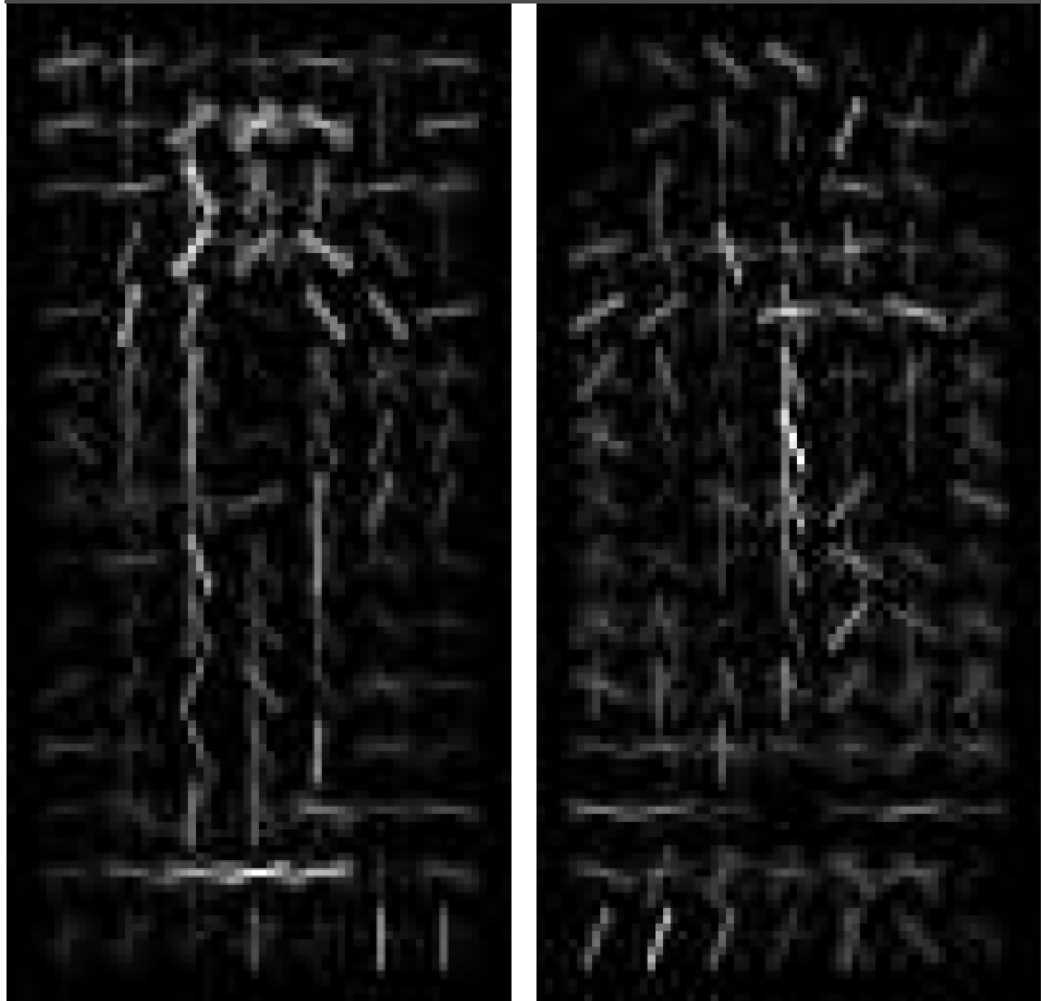


Qualitative experiments on Pascal VOC'07 (Kushal, Schmid, Ponce, 2008)

Model ≡ locally rigid assembly of parts
Part ≡ locally rigid assembly of features

Algorithm	Car	Cow	Mbike
Our Method	0.414	**0.206**	**0.394**
Chum et al. [3]	**0.434**	-	0.375
Felzenszwalb et al. [6]	0.396	0.165	0.337
INRIA Plus [5]	0.294	0.127	0.249
IRISA [5]	0.318	0.119	0.227

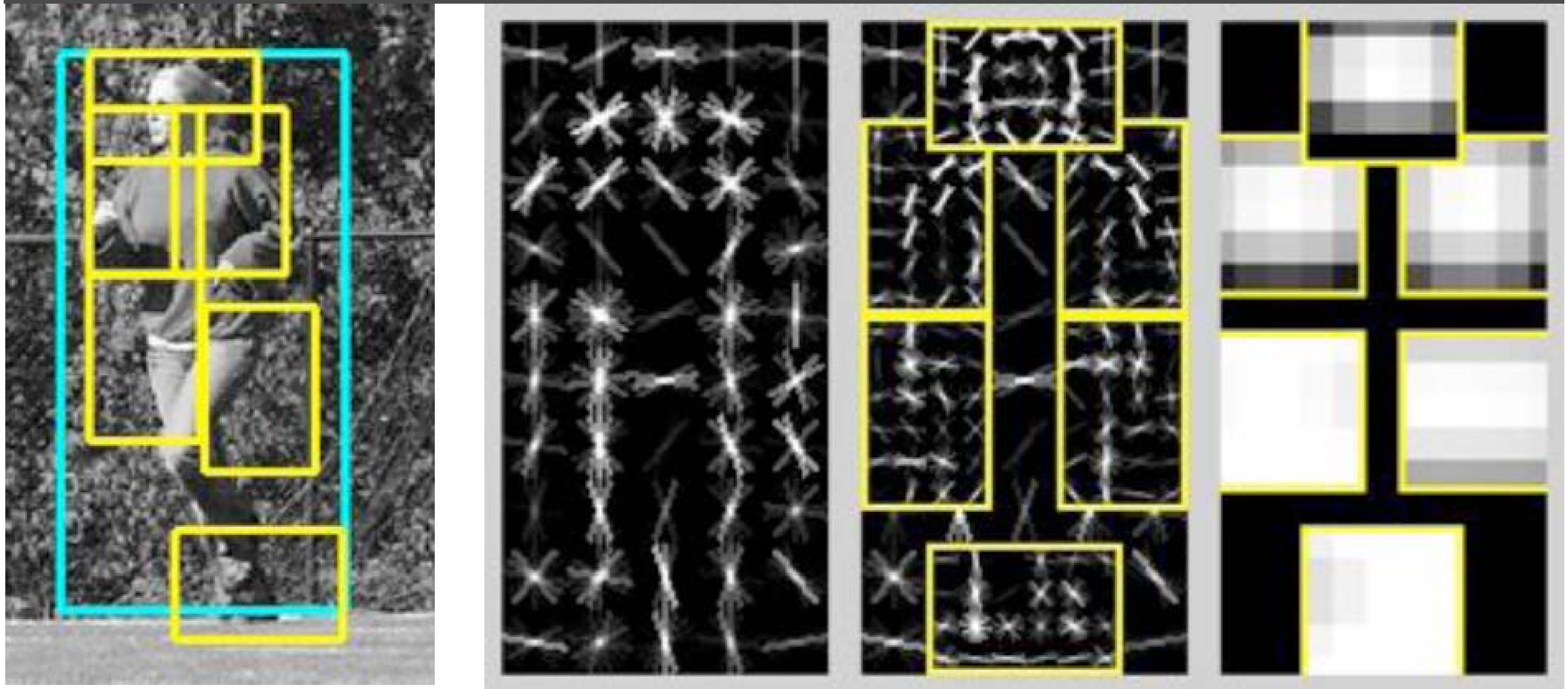
Quantitative experiments on Pascal VOC'07 (Kushal, Schmid, Ponce, 2008)



Color histograms (S&B'91)
Local jets (Florack'93)
Spin images (J&H'99)
Sift (Lowe'99)
Shape contexts (B&M'95)

Texton histograms (L&M'97)
Gist (O&T'05)
Spatial pyramids (LSP'06)
Hog (D&T'06)
Phog (B&Z'07)
Convolutional nets (LC'70)

Locally orderless structure of images (K&vD'99)



Felzwenszalb, McAllester, Ramanan (2007)
[Wins on 6 of the Pascal'07 classes, see Chum & Zisserman (2007) for the other big winner.]