Learning to Localize Objects with Structured Output Regression

Matthew B. Blaschko and Christoph H. Lampert

Max Planck Institute for Biological Cybernetics
Tübingen, Germany

October 13th, 2008

Object (Category) Recognition

- Is there a cow in this picture?

Object (Category) Recognition

- Is there a cow in this picture?

✓

Object (Category) Recognition

- Is there a cow in this picture?

Object (Category) Recognition

- Is there a cow in this picture?

Object (Category) Recognition

- Is there a cow in this picture?

Object (Category) Recognition

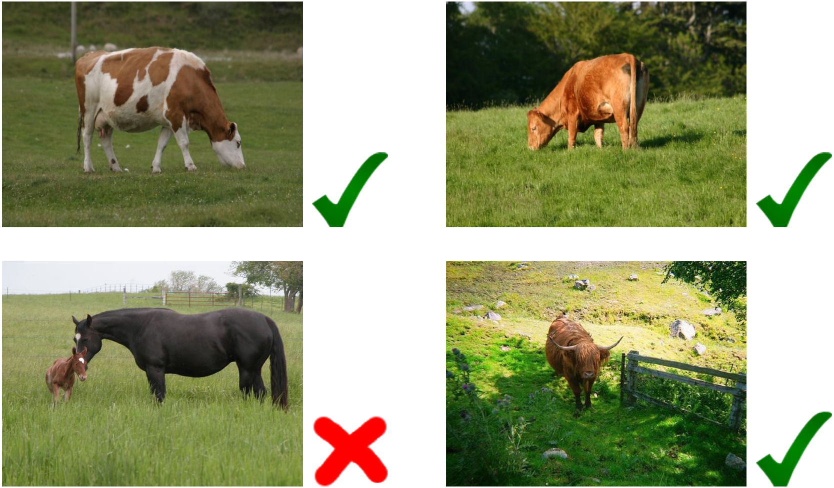
- Is there a cow in this picture?

Object (Category) Recognition

- Is there a cow in this picture?

Object (Category) Recognition

- Is there a cow in this picture?



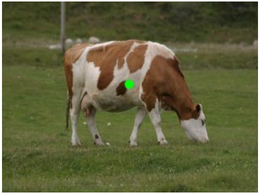
- Binary classification problem: classifiers do the job.

Object (Category) Localization

- *Where* in the picture is the cow?

Object (Category) Localization

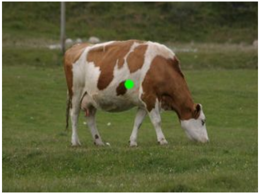
- *Where* in the picture is the cow?



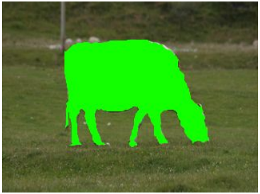
center point

Object (Category) Localization

- *Where* in the picture is the cow?



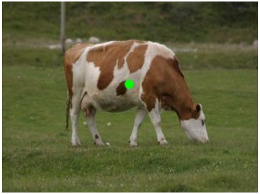
center point



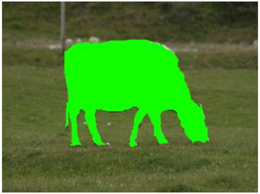
segmentation

Object (Category) Localization

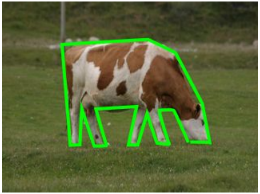
- *Where* in the picture is the cow?



center point



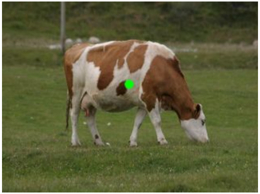
segmentation



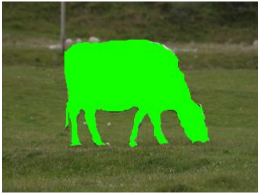
outline

Object (Category) Localization

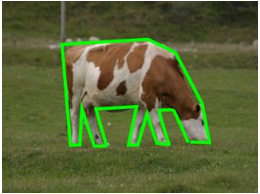
- *Where* in the picture is the cow?



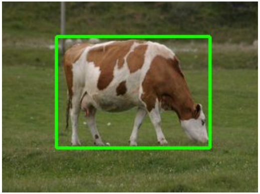
center point



segmentation



outline



bounding box

- Many possible answers, none is binary.
- How can we build a trainable system to *localize objects*?

From Classification to Localization: Sliding Window

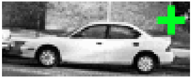
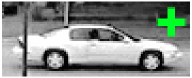
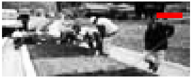
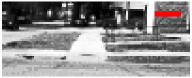
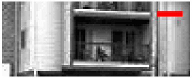
Most common: binary training + sliding window.

Train a binary classifier $f : \{ \textit{all images} \} \to \mathbb{R}$.

- training images
  - positive: object class images
  - negative: background regions
- train a classifier, e.g.
  - support vector machine,
  - Viola-Jones cascade, ...
- decision function $f : \{\textit{images}\} \to \mathbb{R}$
  - $f > 0$ means "*image shows the object.*"
  - $f < 0$ means "*image does not show the object.*"

Use f in sliding window procedure:

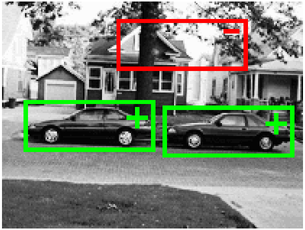
- apply f to many subimages
- subimage with largest score is object location

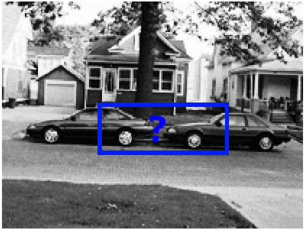
Problems of Sliding Window Localization

Binary training + sliding window is inconsistent:

1) We learn for the wrong task:
   - Training: only *sign f* matters.
   - Testing: f used as real-valued quality measure.



2) The training samples do not reflect the test situation.
   - Training: samples show either *complete object* or *no object*.
   - Testing: many subregions show *object parts*.



Learning to Localize Objects

Ideal setup:

- Learn a true *localization function*:

$$g : \{\textit{all images}\} \rightarrow \{\textit{all boxes}\}$$

$g\Bigg($ 

$\Bigg) =$ 

that predicts object location from images.

- Train in a consistent end-to-end way.

- Training distribution reflects test distribution.

Object Localization as Structured Output Regression

Regression task:

- training pairs $(x_1, y_1), \ldots, (x_n, y_n) \in \mathcal{X} \times \mathcal{Y}$
  - x_i are images, y_i are bounding boxes
- Learn a mapping
  - $g : \mathcal{X} \to \mathcal{Y}$

  that generalizes from the given examples:
  - $g(x_i) \approx y_i$, for $i = 1, \ldots, n$.
- Prefer *smooth* mappings to avoid overfitting.

Regression is not $\mathbb{R} \to \mathbb{R}$, but input and output are *structured spaces*:

- inputs are $2D$ images
- outputs are 4-tuples $y = [\mathit{left}, \mathit{top}, \mathit{right}, \mathit{bottom}] \in \mathbb{R}^4$
  that must be predicted *jointly*.

Alternatives: Predicting with Structured Outputs

Independent Training?

- Learn independent functions $g_{left},\ g_{top},\ g_{right},\ g_{bottom} : \mathcal{X} \to \mathbb{R}$.
- Unable to integrate constraints and correlations.

Nearest Neighbor?

- Store all example (x_i, y_i) as prototypes.
- For new image x, predict box y_i where $i = \operatorname{argmin}_{i=1,\dots n} dist(x, x_i)$.
- No invariance e.g. against translation.
- Requires a lot of training data.

Probabilistic Modeling?

- Build a model of $p(x, y)$ or $p(y|x)$, e.g. Gaussian Mixture.
- Difficult to integrate invariances, e.g. against scale changes.
- Requires a lot of training data to cover 4D output space.

Background: Structured Output Support Vector Machine

Structured Output Support Vector Machine: [Tsochantaridis2005]

- Generalization of SVMs to arbitrary output domains.
- Goal: prediction function $g : \mathcal{X} \to \mathcal{Y}$
  - Learn a *compatibility function*

    $$F\colon \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$$

    and define

    $$g(x) := \operatorname*{argmax}_{y \in \mathcal{Y}} F(x, y)$$
- $g(x)$ is learned *discriminatively*.
- *Non-linearity* and *domain-knowledge* included by *kernelization* of F.

I. Tsochantaridis, T. Joachims, T. Hofmann, Y. Altun: Large Margin Methods for Structured and Interdependent Output Variables, JMLR, 2005.

Structured Output Support Vector Machine

Setup:

- Define positive definite kernel $k : (\mathcal{X} \times \mathcal{Y}) \times (\mathcal{X} \times \mathcal{Y}) \to \mathbb{R}$.
  - $k(.,.)$ induces a Reproducing Kernel Hilbert Space $\mathcal{H}$ and an implicit feature map $\phi : \mathcal{X} \times \mathcal{Y} \to \mathcal{H}$.
- Define loss function $\Delta : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$.

SO-SVM Training:

- Solve the *convex* optimization problem

$$\min_{w,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i$$

subject to margin constraints with loss function Δ:

$$\Delta(y_i, y) + \langle w, \phi(x_i, y)\rangle - \langle w, \phi(x_i, y_i)\rangle \;\leq\; \xi_i,$$

for all $y \in \mathcal{Y} \setminus \{y_i\}$ and $i = 1, \ldots, n$.

Structured Output Support Vector Machine

- Unique solution $w \in \mathcal{H}$ defines the *compatibility* function

$$F(x, y) = \langle w, \phi(x, y) \rangle$$

  linear in $w \in \mathcal{H}$, but nonlinear in $\mathcal{X}$ and $\mathcal{Y}$.

- $F(x, y)$ measures how well the output y fits to the input x.
  - analogue in probabilistic model: $F(x, y) \,\hat{=}\, \log p(y|x)$.
  - but: $F(x, y)$ max-margin trained, not probabilistic!

- Best prediction for new x is the *most compatible* y:

$$g(x) := \operatorname*{argmax}_{y \in \mathcal{Y}} F(x, y).$$

- Evaluating $g : \mathcal{X} \to \mathcal{Y}$ is like *generalized sliding window*:
  - for fixed x, we evaluate a quality function for every box $y \in \mathcal{Y}$.
    - approximate: sliding window
    - exact: branch-and-bound [Lampert&Blaschko, CVPR2008]
  - other parameterization: min-cut, (loopy) BP,...

SO-SVM Training: Interpretation

SO-SVM Training Revisited: Hard-Margin Case

- Solve the *convex* optimization problem

$$\min_{w,\xi} \ \frac{1}{2}\|w\|^2$$

subject to margin constraints with loss function $\Delta \geq 0$:

$$\langle w, \phi(x_i, y_i)\rangle - \langle w, \phi(x_i, y)\rangle \ \geq \ \Delta(y_i, y)$$

for all $y \in \mathcal{Y} \setminus \{y_i\}$ and $i = 1, \ldots, n$.

SO-SVM Training: Interpretation

SO-SVM Training Revisited: Hard-Margin Case

- Solve the *convex* optimization problem

$$\min_{w,\xi} \; \frac{1}{2}\|w\|^2$$

subject to margin constraints with loss function $\Delta \geq 0$:

$$\underbrace{\langle w, \phi(x_i, y_i)\rangle}_{=F(x_i,y_i)} - \underbrace{\langle w, \phi(x_i, y)\rangle}_{=F(x_i,y)} \;\geq\; \Delta(y_i, y)$$

for all $y \in \mathcal{Y} \setminus \{y_i\}$ and $i = 1, \dots, n$.

- Implies:

$$F(x_i, y_i) > F(x_i, y) \qquad \text{for all } y \in \mathcal{Y} \setminus \{y_i\}.$$

- Because $g(x) := \operatorname{argmax}_y F(x, y)$, this means

$$g(x_i) = y_i \qquad \text{for all training pairs } (x_i, y_i).$$

SO-SVM Training: Interpretation

SO-SVM Training Revisited: General Case

- Solve the *convex* optimization problem

$$\min_{w,\xi} \ \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i$$

subject to margin constraints with loss function $\Delta \geq 0$:

$$\underbrace{\langle w, \phi(x_i, y_i)\rangle}_{=F(x_i,y_i)} - \underbrace{\langle w, \phi(x_i, y)\rangle}_{=F(x_i,y)} \ \geq \ \Delta(y_i, y) - \xi_i$$

for all $y \in \mathcal{Y} \setminus \{y_i\}$ and $i = 1, \ldots, n$.

- Implies $\quad F(x_i, y_i) - F(x_i, y) > \Delta(y_i, y) - \xi_i \qquad$ for $y \in \mathcal{Y} \setminus \{y_i\}$.

- $\Delta(y_i, y) = \begin{cases} \text{small}, & \text{if } y \approx y_i \\ \text{large}, & \text{if } y \not\approx y_i \end{cases} \quad \Rightarrow \quad g(x_i) \approx y_i$ for most (x_i, y_i)

  because penalization increases the more $g(x_i)$ differs from y_i.

Joint Kernel for (Image,Box)-Pairs

SO-SVM for Object Localization:

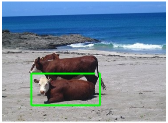
What is a good *joint kernel* function?

- $k_{joint}(\ (x,y),(x',y')\ )$ for images x,x' and boxes y,y'.
- Observation: $x|_y$ (image restricted to box region) is again an image.
- Compare two images with boxes by comparing the images inside the box regions:

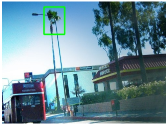
$$k_{joint}(\ (x,y),(x',y')\ ) := k_{image}(x|_y, x'|_{y'},)$$

- Properties:
  - automatic translation invariance
  - other invariances inherited from image kernel
- Wide range of choices for image kernel k_{image}:
  - linear, χ^2-kernel, spatial pyramids, pyramid match kernel, ...

Restriction Kernel: Examples

$k_{joint}\Big($  ,  $\Big) = k\Big($  ,  $\Big)$ is large.

$k_{joint}\Big($  ,  $\Big) = k\Big($  ,  $\Big)$ is small.

$k_{joint}\Big($  ,  $\Big) = k\Big($  ,  $\Big)$

could also be large.

Loss Function for Image Boxes

What is a good *loss function* $\Delta(y, y')$?

- $\Delta(y_i, y)$ plays the role that the *margin* has in SVMs.
- $\Delta(y_i, y)$ measures how far a prediction y is from a target y_i.

- We use *area overlap*:

$$\begin{aligned}\Delta(y, y') &:= 1 - \textit{area overlap} \text{ between } y \text{ and } y' \\ &= 1 - \frac{\textit{area}(y \cap y')}{\textit{area}(y \cup y')}\end{aligned}$$

$$\Rightarrow \quad \begin{aligned}\Delta(y, y') = 0 \quad &\text{iff} \quad y = y', \\ \Delta(y, y') = 1 \quad &\text{iff} \quad y \text{ and } y' \text{ are disjoint.}\end{aligned}$$

Results: PASCAL VOC 2006 dataset

- natural images (from Microsoft Research Cambridge and Flickr)
- ≈5,000 images: ≈2,500 train/val, ≈2,500 test

- humanly labeled ≈9,500 objects in 10 predefined classes:
  - `bicycle, bus, car, cat, cow, dog, horse, motorbike, person, sheep`
- task: predict locations and confidence scores for each class
- evaluation: precision–recall curves

Results: PASCAL VOC 2006 dataset

Experiments on PASCAL VOC 2006 dataset:

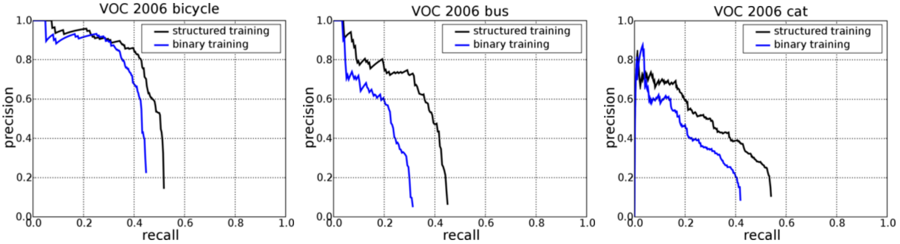
- Most simple setup:
  - SURF local features, 3000-bin *bag-of-visual-word* histograms,
  - Linear kernel function
  - Predict exactly one object per image
  - For PR-curves, rank images by score of χ^2-SVM

Example detections for VOC 2006 `bicycle`, `bus` and `cat`.

Results: PASCAL VOC 2006 dataset

Experiments on PASCAL VOC 2006 dataset:

- Most simple setup:
  - SURF local features, 3000-bin *bag-of-visual-word* histograms,
  - Linear kernel function
  - Predict exactly one object per image
  - For PR-curves, rank images by score of χ^2-SVM



Precision–recall curves for VOC 2006 `bicycle`, `bus` and `cat`.

- Structured regression clearly improves detection performance.

Results: PASCAL VOC 2006 dataset

- We improved best previously published scores in 6 of 10 classes.

	proposed	best VOC2006	post VOC2006
bicycle	.472	.440 [1]	**.498** [5]
bus	**.342**	.169 [2]	.249 [6]
car	.336	.444 [3]	**.458** [5]
cat	**.300**	.160 [2]	.223 [7]
cow	**.275**	.252 [2]	—
dog	**.150**	.118 [4]	.148 [7]
horse	**.211**	.140 [1]	—
motorbike	**.397**	.390 [3]	—
person	.107	.164 [3]	**.340** [8]
sheep	.204	**.251** [3]	—

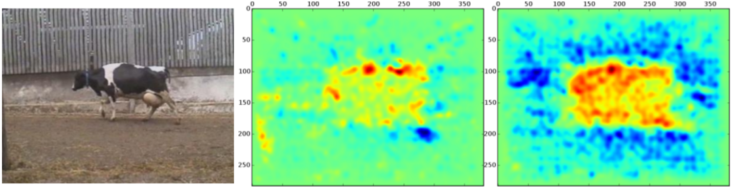
Average Precision (AP) scores on the 10 categories of PASCAL VOC 2006.

[1]: I. Laptev, VOC2006
[2]: V. Viitaniemi, VOC2006
[3]: M. Douze, VOC2006
[4]: J. Shotton, VOC2006
[5]: Crandall and Huttenlocher, CVPR07
[6]: Chum and Zisserman, CVPR07
[7]: Lampert et al., CVPR08
[8]: Felzenszwalb et al., CVPR08

Results: TU Darmstadt cow dataset

Why does it work better?

- Experiment on TU Darmstadt cow dataset
  - relatively easy, side views of cows in front of different backgrounds
  - 111 training images, 557 test images
- same setup: bag-of-visual words histograms, linear kernel
- Learned distribution of local *weights*:

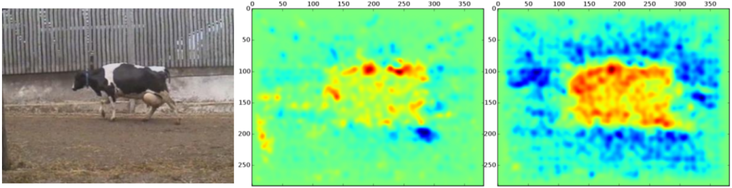


example test image | binary training | structured training

Results: TU Darmstadt cow dataset

Why does it work better?

- Experiment on TU Darmstadt `cow` dataset
  - relatively easy, side views of cows in front of different backgrounds
  - 111 training images, 557 test images
- same setup: bag-of-visual words histograms, linear kernel
- Learned distribution of local *weights*:



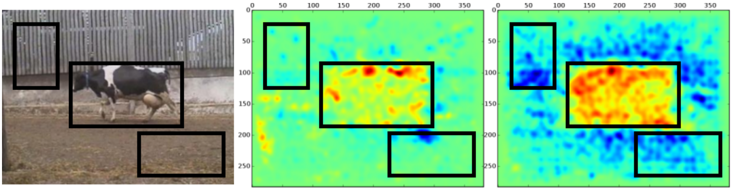
example test image   binary training   structured training

- Binary training: weights concentrated on few discriminative regions
  - all boxes containing "hot spots" gets similarly high scores
- Structured training: whole inside positive, whole outside negative
  - correct box is *enforced* to be the best of all possible ones

Results: TU Darmstadt cow dataset

Why does it work better?

- Experiment on TU Darmstadt cow dataset
  - relatively easy, side views of cows in front of different backgrounds
  - 111 training images, 557 test images
- same setup: bag-of-visual words histograms, linear kernel
- Learned distribution of local *weights*:



example test image   binary training   structured training

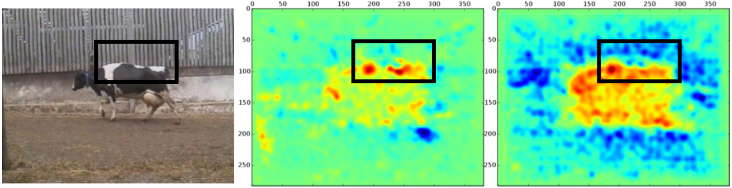
- Binary training: weights concentrated on few discriminative regions
  - all boxes containing "hot spots" gets similarly high scores
- Structured training: whole inside positive, whole outside negative
  - correct box is *enforced* to be the best of all possible ones

Results: TU Darmstadt `cow` dataset

Why does it work better?

- Experiment on TU Darmstadt `cow` dataset
  - relatively easy, side views of cows in front of different backgrounds
  - 111 training images, 557 test images
- same setup: bag-of-visual words histograms, linear kernel

- Learned distribution of local *weights*:



example test image   binary training   structured training

- Binary training: weights concentrated on few discriminative regions
  - all boxes containing "hot spots" gets similarly high scores
- Structured training: whole inside positive, whole outside negative
  - correct box is *enforced* to be the best of all possible ones

Summary

How to build a trainable system for object localization?

Summary

How to build a trainable system for object localization?

- Conventional approaches use binary *classifier function*.
  - two-step procedure: binary training + sliding window evaluation
  - inconsistent: problem solved in training is different than at test time.

Summary

How to build a trainable system for object localization?

- Conventional approaches use binary *classifier function*.
  - two-step procedure: binary training + sliding window evaluation
  - inconsistent: problem solved in training is different than at test time.

- We formulate localization as *regression with structured output*.
  - training directly reflects the procedure at test time.

Summary

How to build a trainable system for object localization?

- Conventional approaches use binary *classifier function*.
  - two-step procedure: binary training + sliding window evaluation
  - inconsistent: problem solved in training is different than at test time.
- We formulate localization as *regression with structured output*.
  - training directly reflects the procedure at test time.
- Proposed system realized as *structured support vector machine*.
  - discriminative max-margin training, *convex* optimization problem

Summary

How to build a trainable system for object localization?

- Conventional approaches use binary *classifier function*.
  - two-step procedure: binary training + sliding window evaluation
  - inconsistent: problem solved in training is different than at test time.
- We formulate localization as *regression with structured output*.
  - training directly reflects the procedure at test time.
- Proposed system realized as *structured support vector machine*.
  - discriminative max-margin training, *convex* optimization problem
- Empirical results:
  - structured training improves over binary training
  - uncertainty of optimal box position is reduced

Summary

How to build a trainable system for object localization?

- Conventional approaches use binary *classifier function*.
  - two-step procedure: binary training + sliding window evaluation
  - inconsistent: problem solved in training is different than at test time.
- We formulate localization as *regression with structured output*.
  - training directly reflects the procedure at test time.
- Proposed system realized as *structured support vector machine*.
  - discriminative max-margin training, *convex* optimization problem
- Empirical results:
  - structured training improves over binary training
  - uncertainty of optimal box position is reduced
- Source code:
  - SVMstruct: `svmlight.joachims.org/svm_struct.html`
  - module for object localization: `www.christoph-lampert.org`