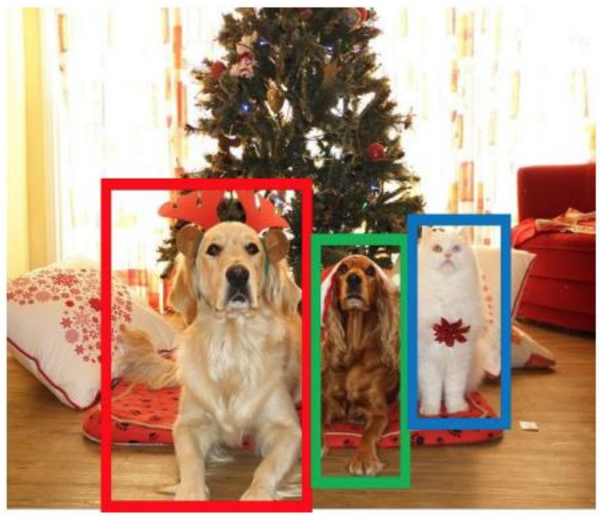
Object Detection



Computer Vision

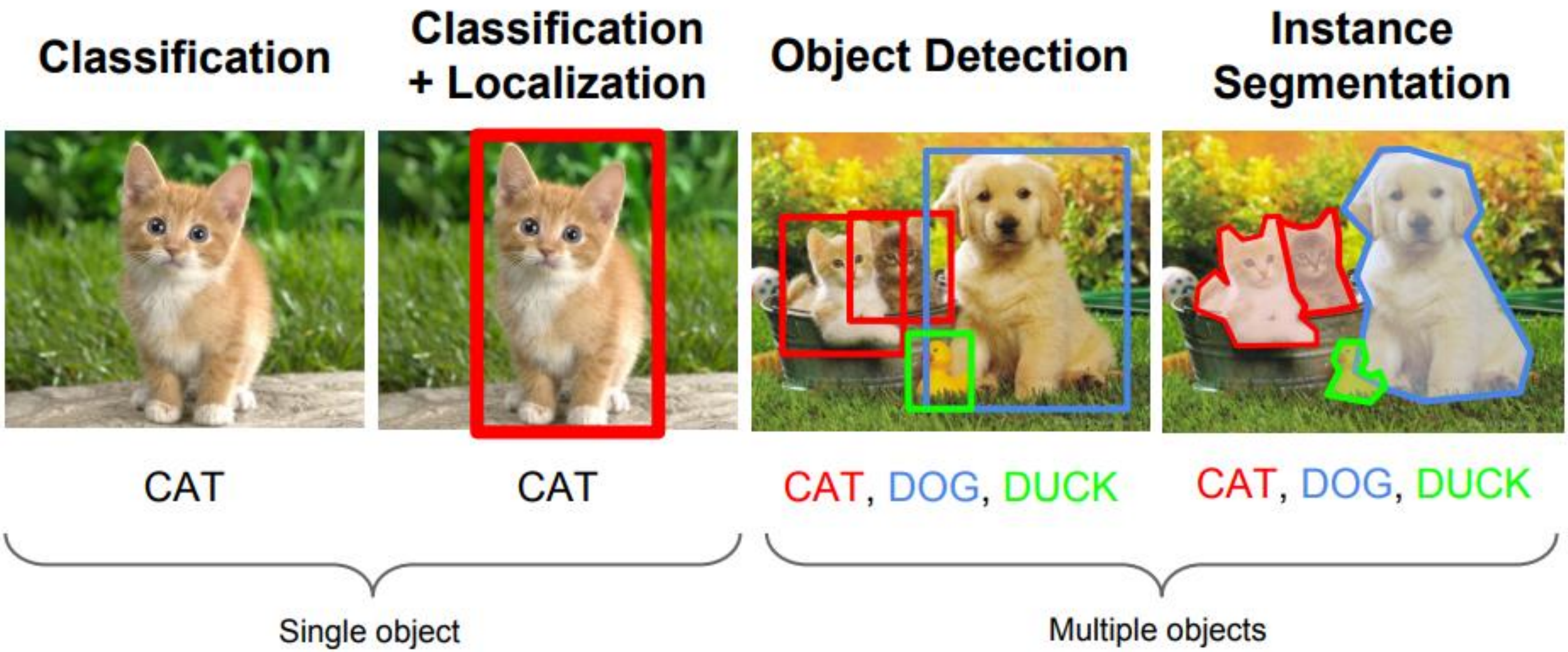
Yuliang Zou, Virginia Tech

Many slides from D. Hoiem, J. Hays, J. Johnson, R. Girshick

Administrative stuffs

- HW 4 due 11:59pm on Wed, November 8
- HW 3 grades are out
  - Average: 116.78, Median: 132.5
  - Excellent reports: Pavan Kumar Gundu, Vidur Kakar, Tarun Kathuria, Prashant Kumar, Snehal More, Naresh Nagabushan, Sudha Ravali Yellapantula
- Final project proposal
  - Feedback via emails
  - Will also set up additional office hours for discussion

Roadmap



Slide from A. Karpathy

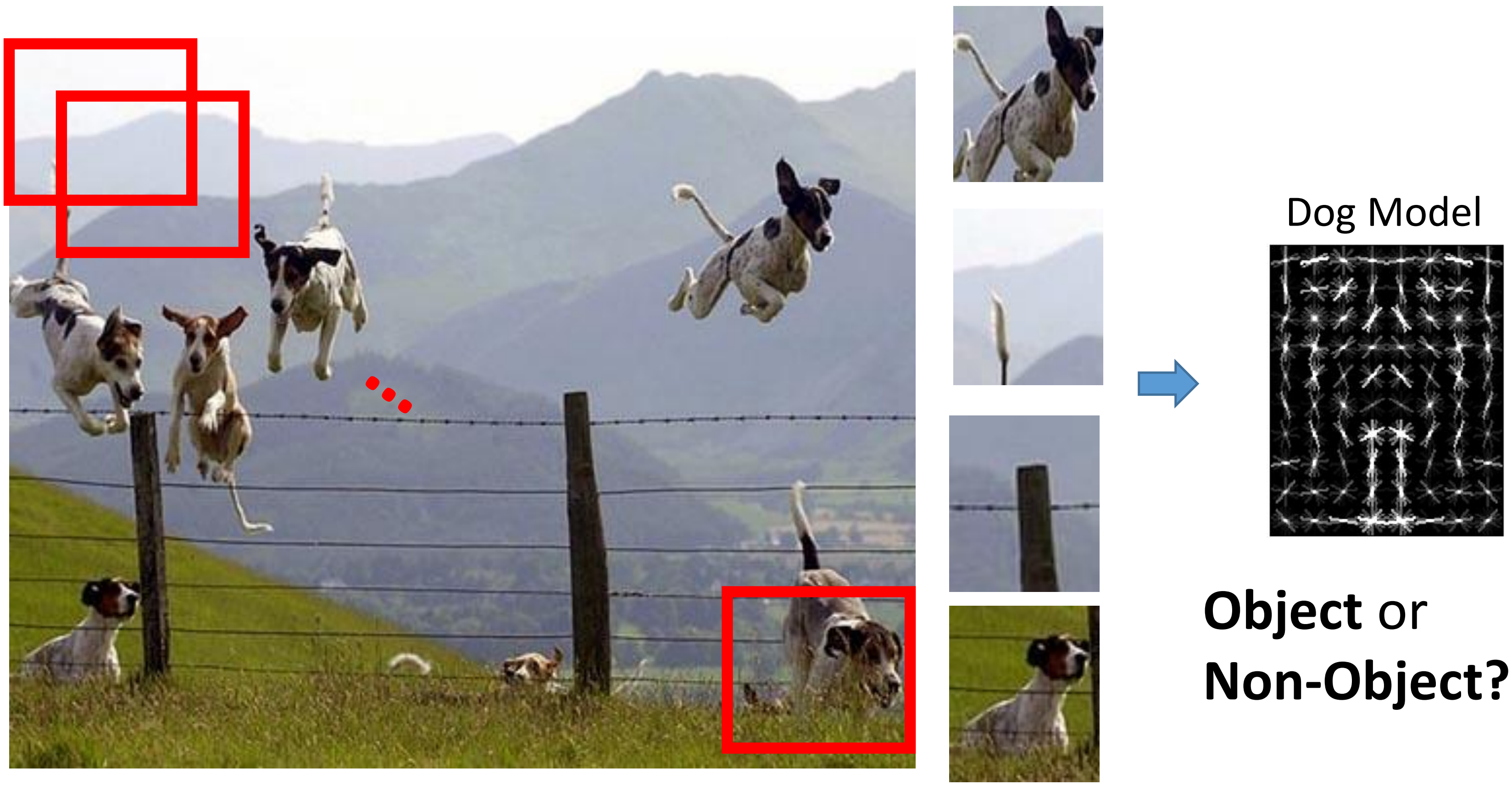
Today's class

- Overview of object category detection
- Traditional methods
    - Dalal-Triggs detector (basic concept)
    - Viola-Jones detector (cascades, integral images)
- Deep learning methods
    - Review of CNN
    - Two-stage: R-CNN
    - One-stage: YOLO, SSD, Retina Net

Demo

Object Category Detection

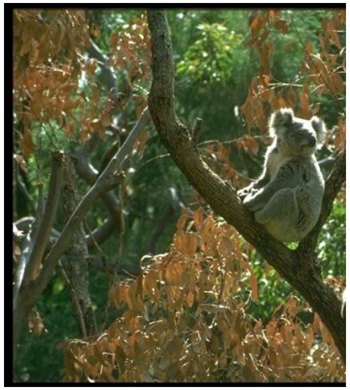
- Focus on object search: "Where is it?"
- Build templates that quickly differentiate object patch from background patch



Challenges in modeling the object class

Illumination

Object pose



Clutter

Occlusions

Intra-class appearance

Viewpoint

Slide from K. Grauman, B. Leibe

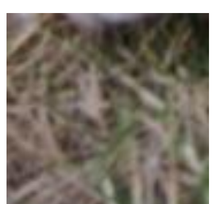
Challenges in modeling the non-object class

True
Detections

Bad
Localization

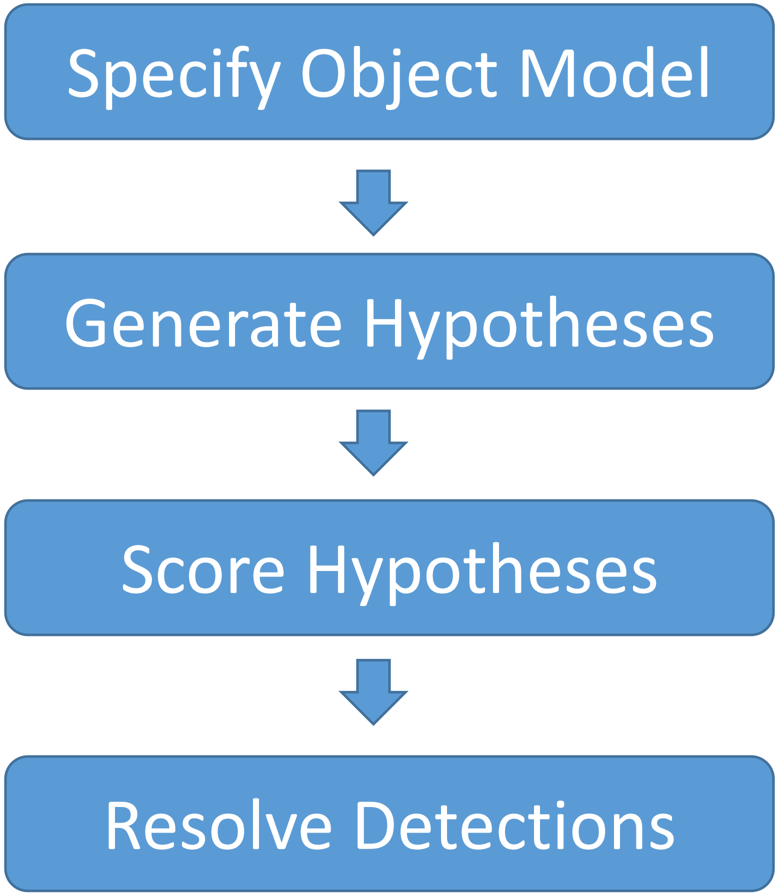
Confused with
Similar Object

Misc. Background



Confused with
Dissimilar Objects

General Process of Object Recognition

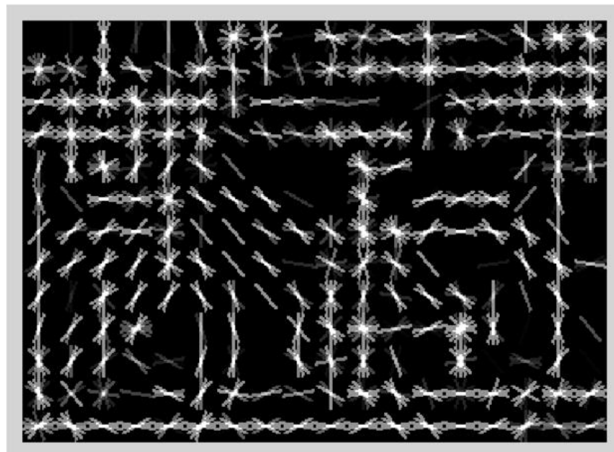


What are the object parameters?

Specifying an object model

1. Statistical Template in Bounding Box
    - Object is some (x,y,w,h) in image
    - Features defined wrt bounding box coordinates

Image

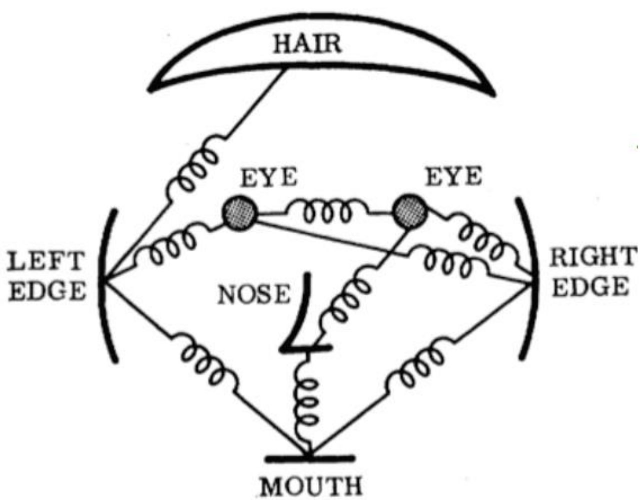


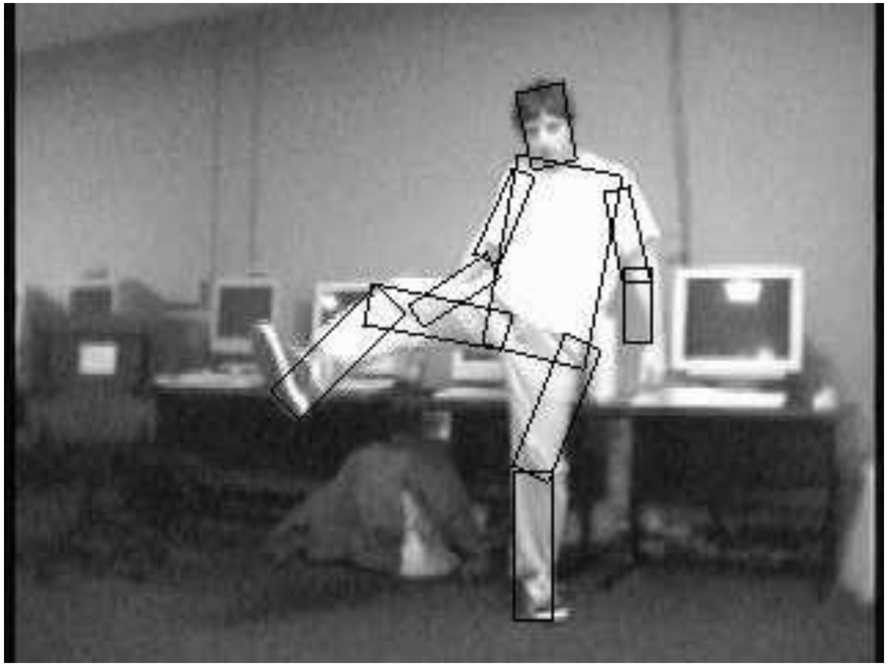
Template Visualization

Images from Felzenszwalb

Specifying an object model

2. Articulated parts model
    - Object is configuration of parts
    - Each part is detectable



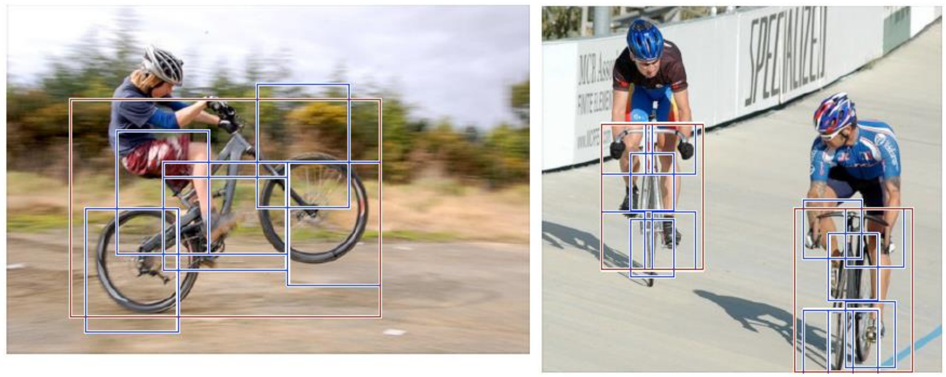


Images from Felzenszwalb

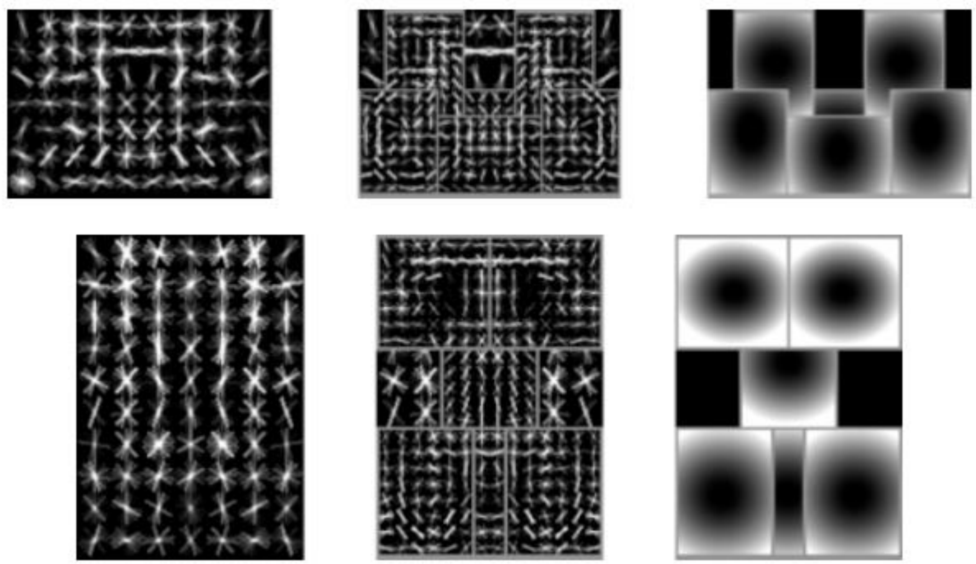
Specifying an object model

3. Hybrid template/parts model

Detections



Template Visualization



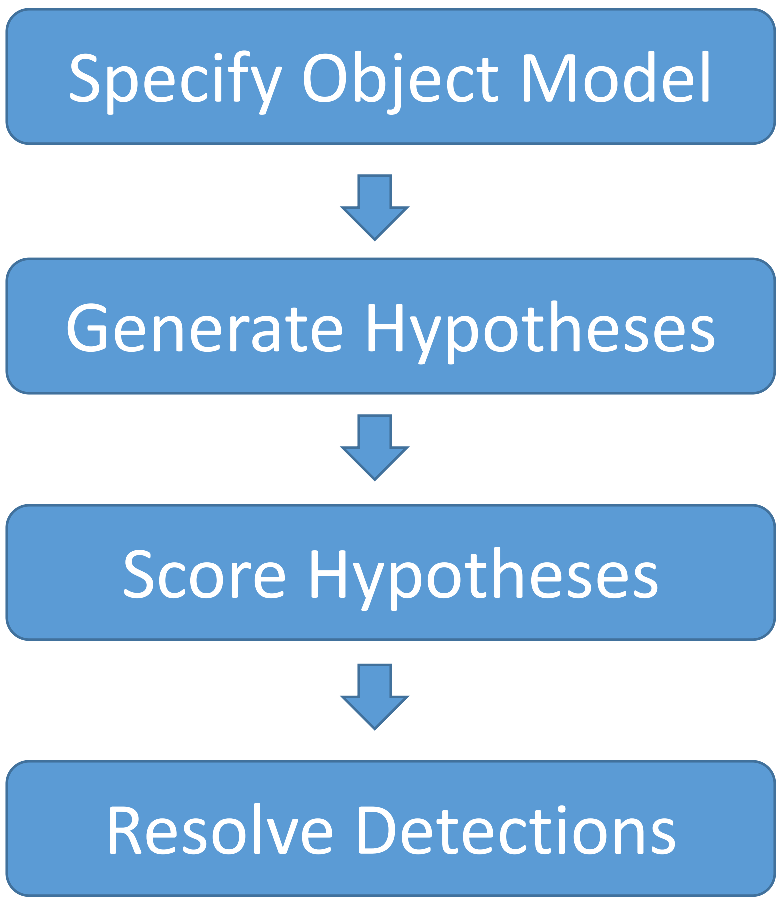
root filters
coarse resolution

part filters
finer resolution

deformation
models

Felzenszwalb et al. 2008

General Process of Object Recognition



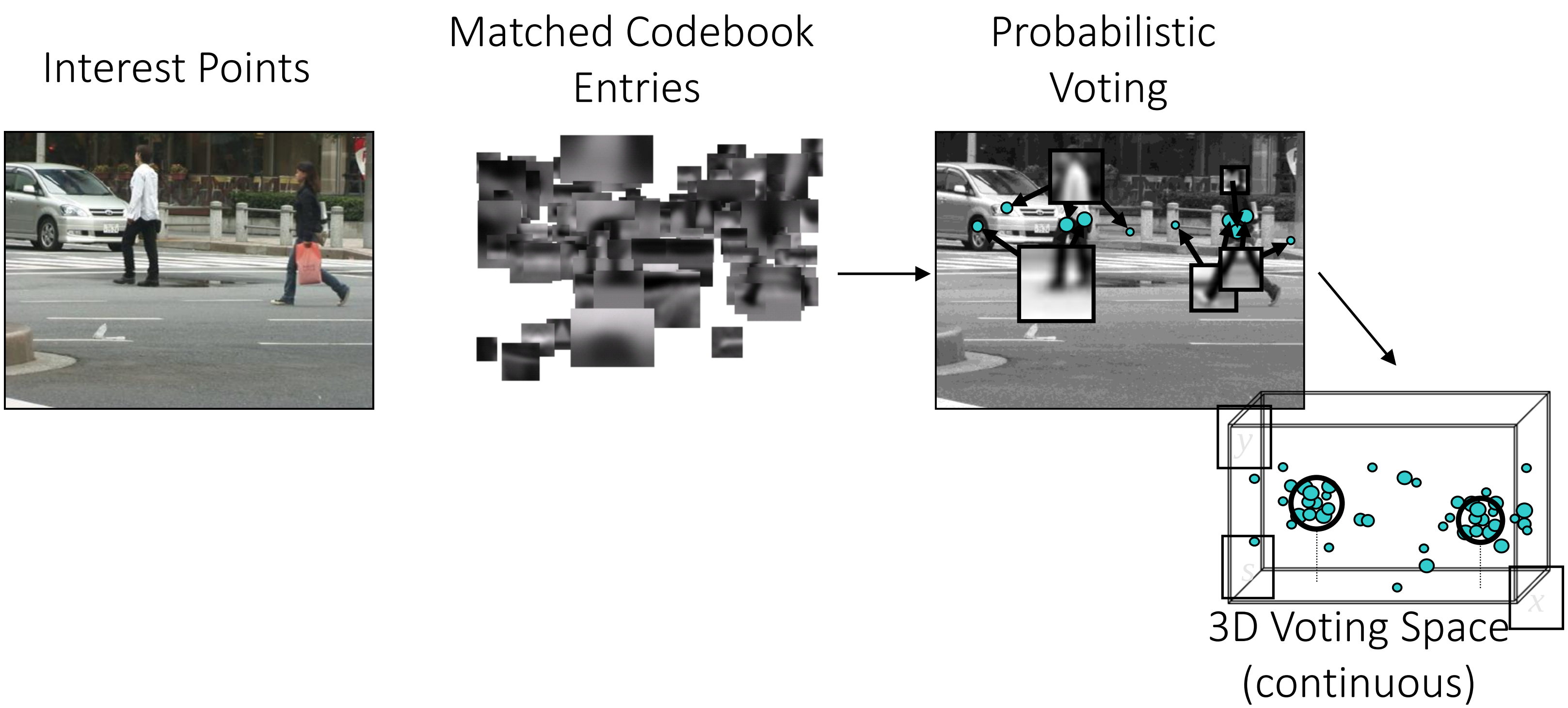
Propose an alignment of the model to the image

Generating hypotheses

1. Sliding window
    - Test patch at each location and scale

Generating hypotheses

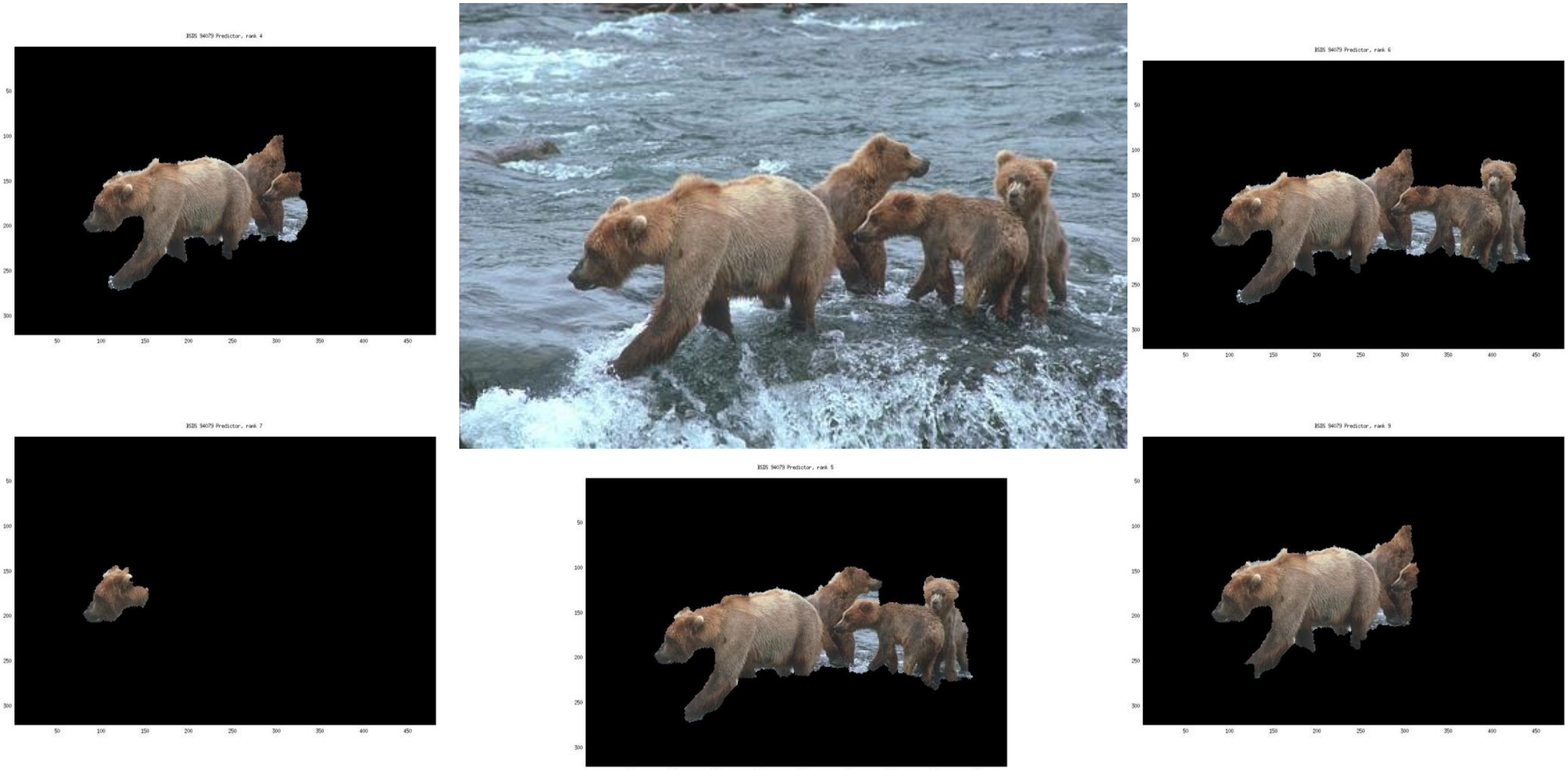
2. Voting from patches/keypoints



ISM model by Leibe et al.

Generating hypotheses

3. Region-based proposal



Endres Hoiem 2010

General Process of Object Recognition

Specify Object Model

↓

Generate Hypotheses

↓

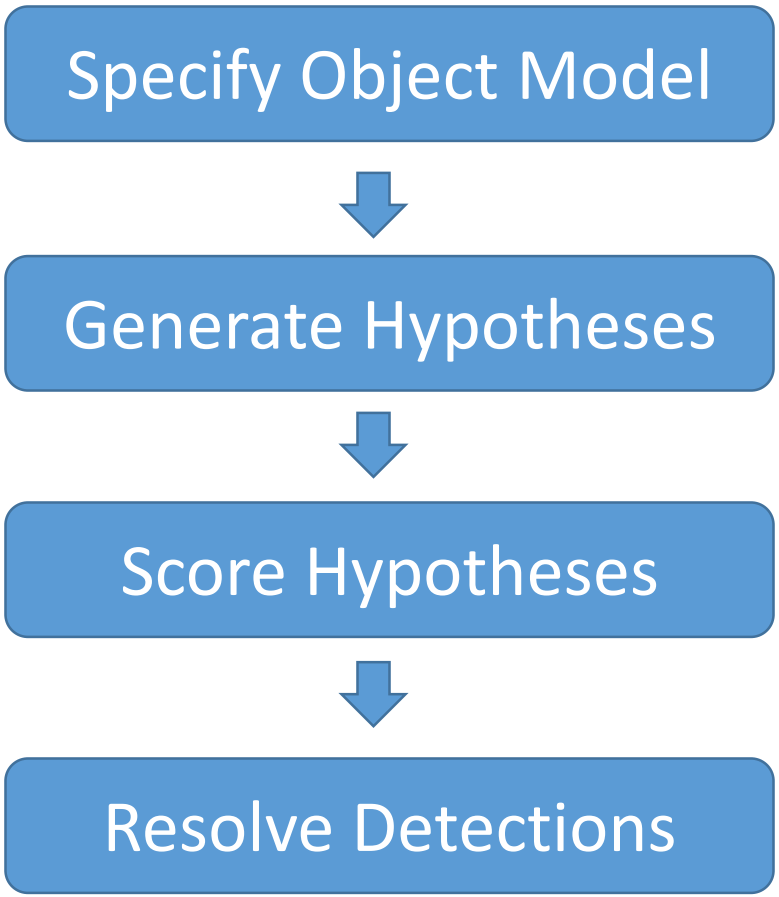
Score Hypotheses

↓

Resolve Detections

Mainly-gradient based or CNN features, usually based on summary representation, many classifiers

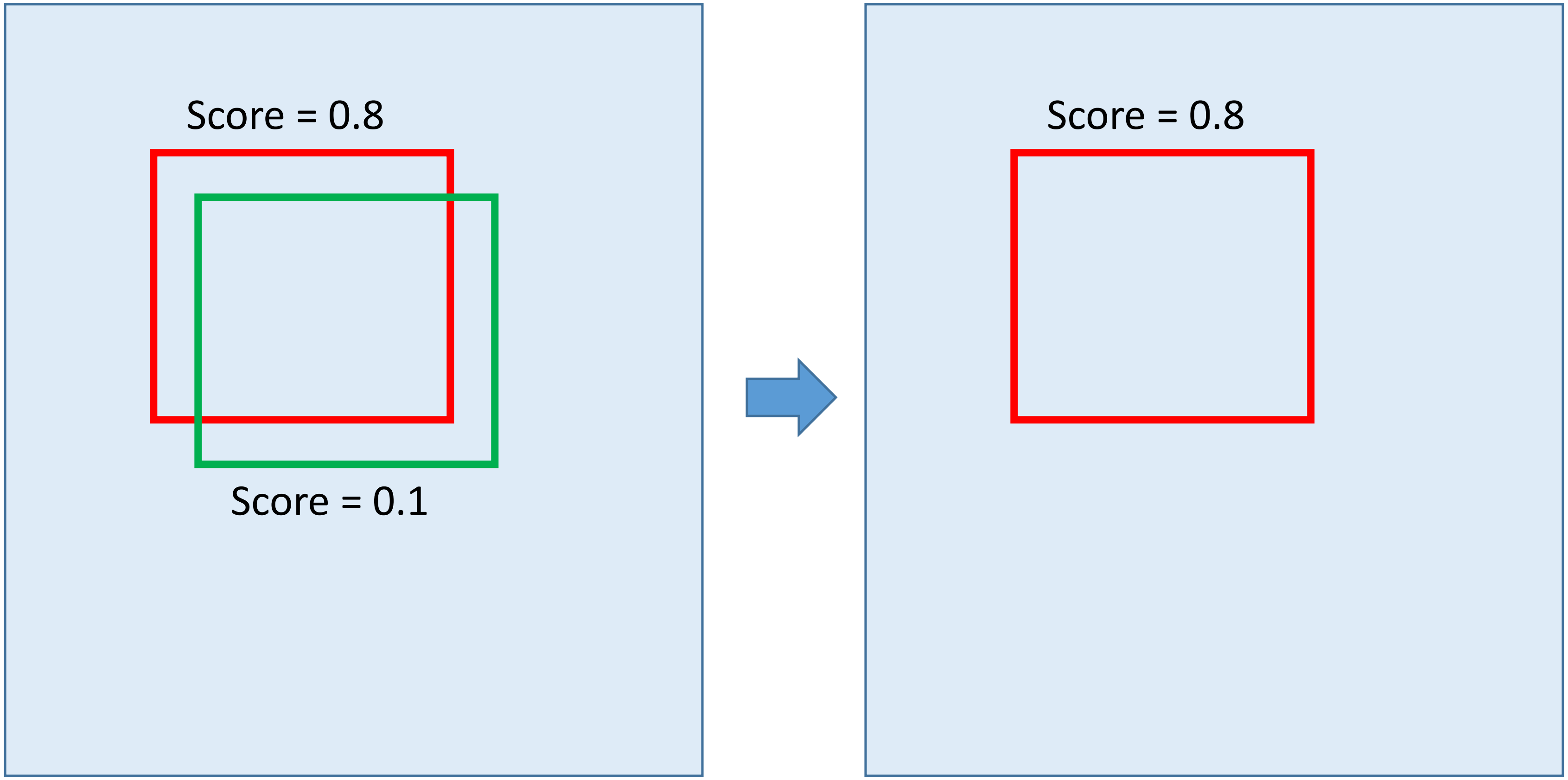
General Process of Object Recognition



Rescore each proposed object based on whole set

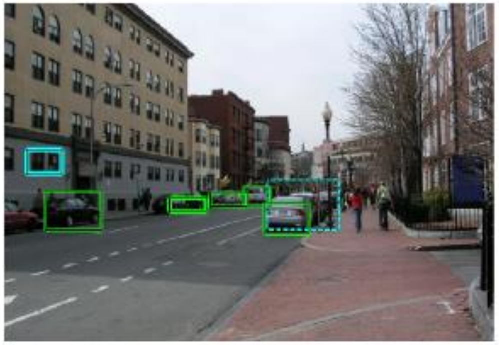
Resolving detection scores

1. Non-max suppression



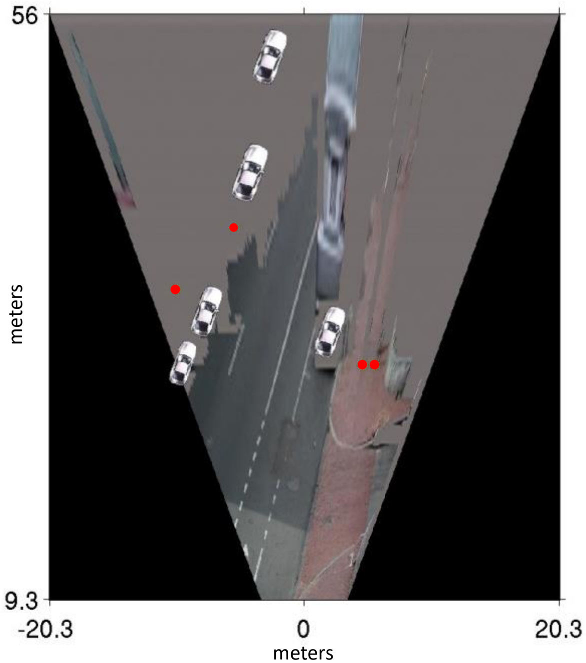
Resolving detection scores

2. Context/reasoning



(g) Car Detections: Local

(h) Ped Detections: Local



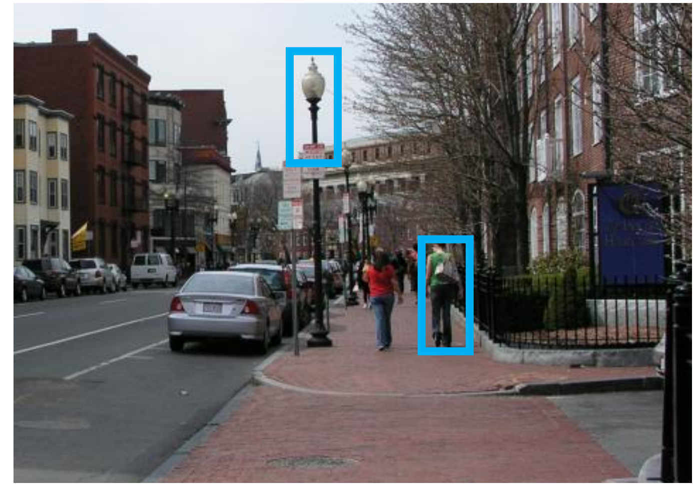
Hoiem et al. 2006

Object category detection in computer vision

Goal: detect all pedestrians, cars, monkeys, etc in image

Basic Steps of Category Detection

1. Align
   - E.g., choose position, scale orientation
   - How to make this tractable?



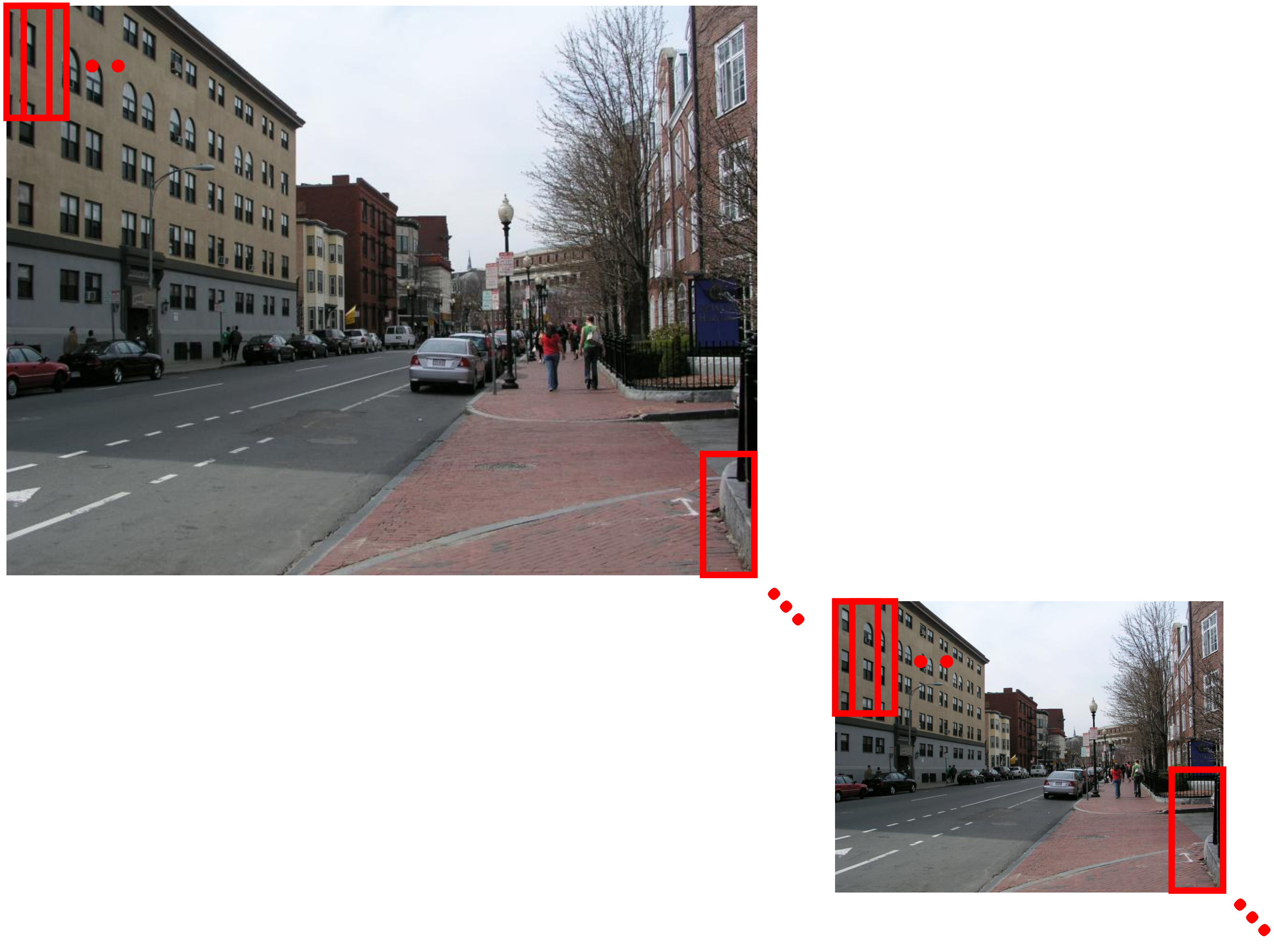
2. Compare
   - Compute similarity to an example object or to a summary representation
   - Which differences in appearance are important?

Aligned Possible Objects

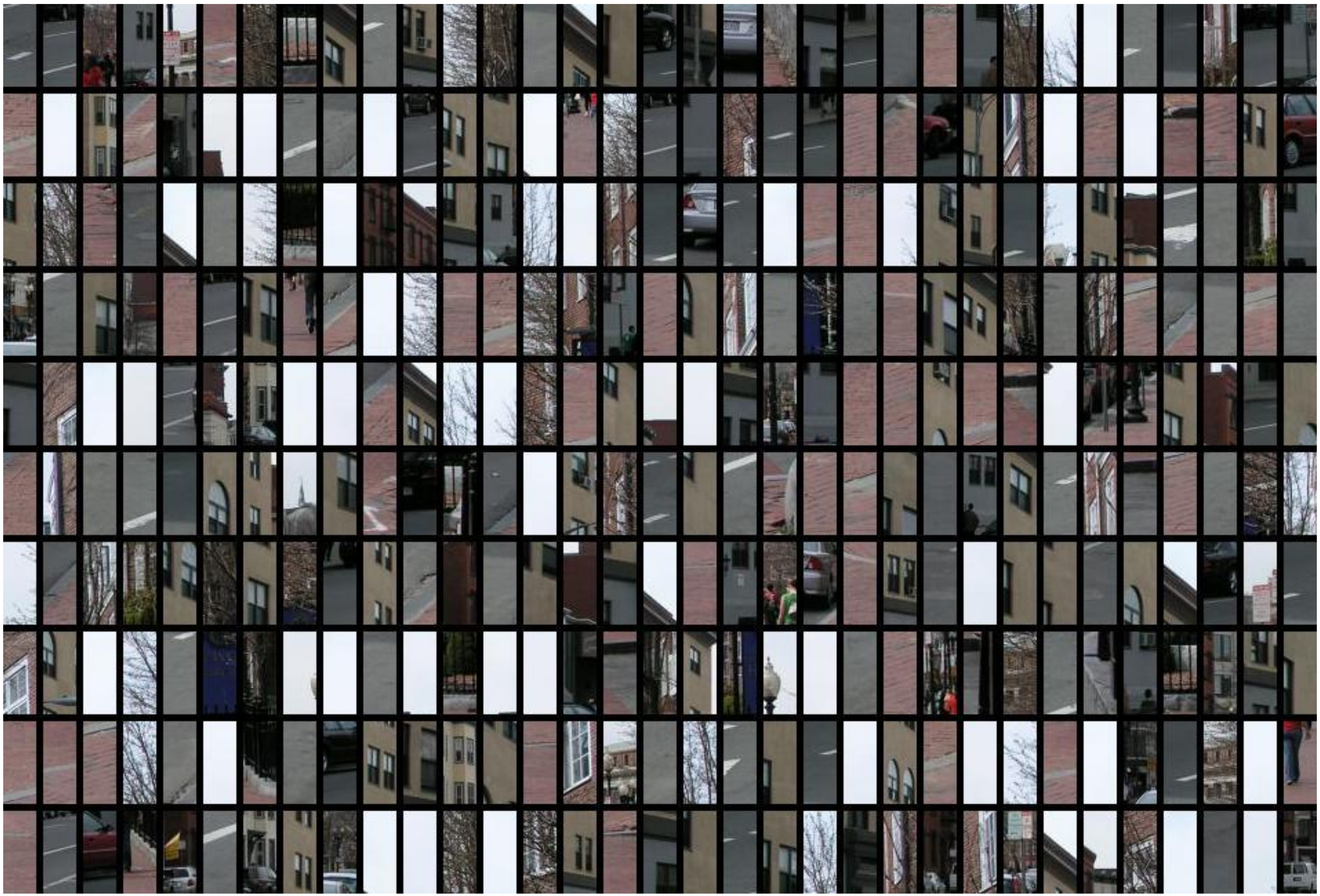
Exemplar

Summary

Sliding window: a simple alignment solution



Each window is separately classified

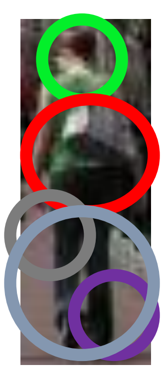


Statistical Template

- Object model = sum of scores of features at fixed positions

$+3 +2 -2 -1 -2.5 = -0.5 \overset{?}{>} 7.5$

Non-object



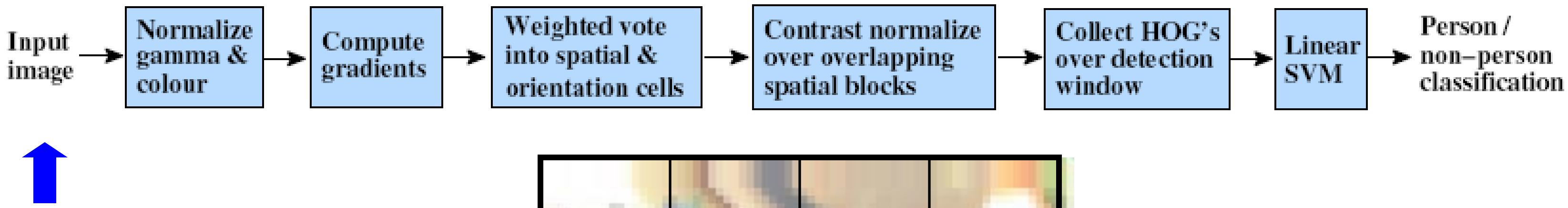
$+4 +1 +0.5 +3 +0.5 = 10.5 \overset{?}{>} 7.5$

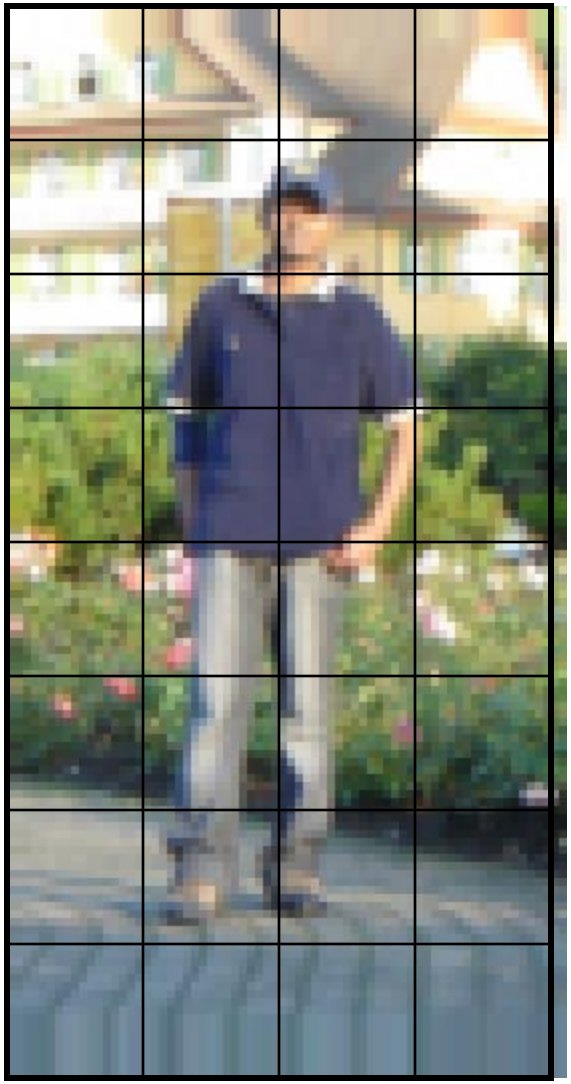
Object

Example: Dalal-Triggs detector

1. Extract fixed-sized (64x128 pixel) window at each position and scale
2. Compute HOG (histogram of gradient) features within each window
3. Score the window with a linear SVM classifier
4. Perform non-maxima suppression to remove overlapping detections with lower scores

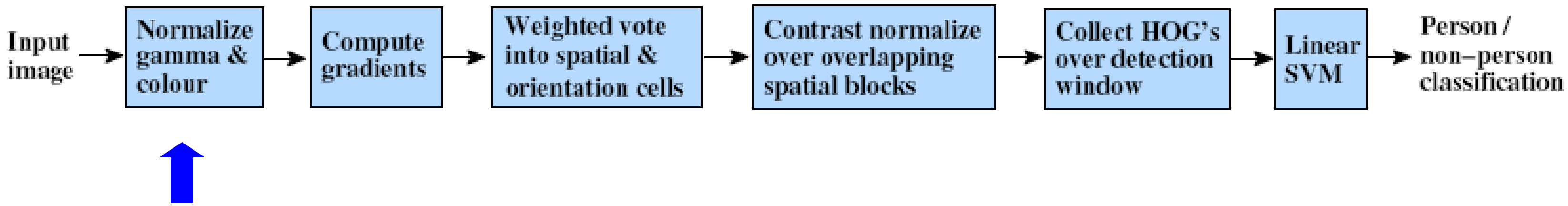
Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05





Slides by Pete Barnum

Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05



- Tested with
  - RGB
  - LAB
  - Grayscale

Slightly better performance vs. grayscale

- Gamma Normalization and Compression
  - Square root
  - Log

Very slightly better performance vs. no adjustment

Input image → Normalize gamma & colour → Compute gradients → Weighted vote into spatial & orientation cells → Contrast normalize over overlapping spatial blocks → Collect HOG's over detection window → Linear SVM → Person / non–person classification

Outperforms

-1	0	1

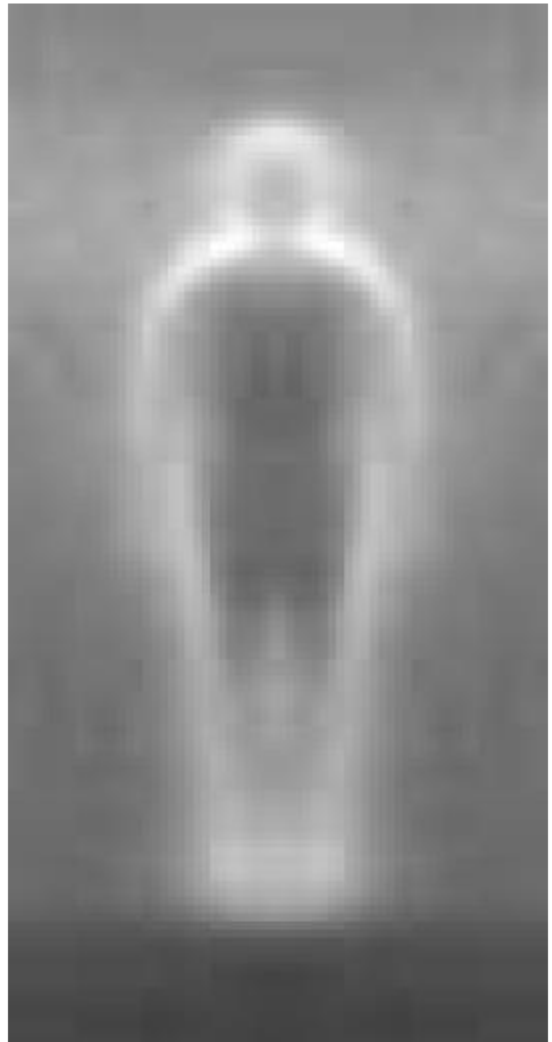
centered

-1	1

uncentered

1	-8	0	8	-1

cubic-corrected



0	1
-1	0

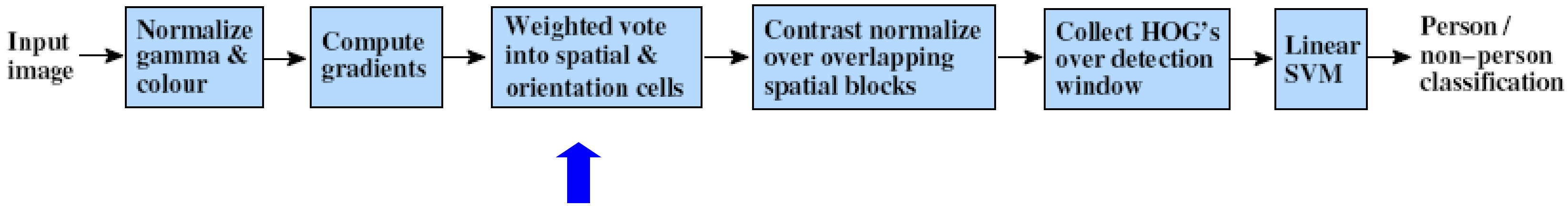
diagonal

-1	0	1
-2	0	2
-1	0	1

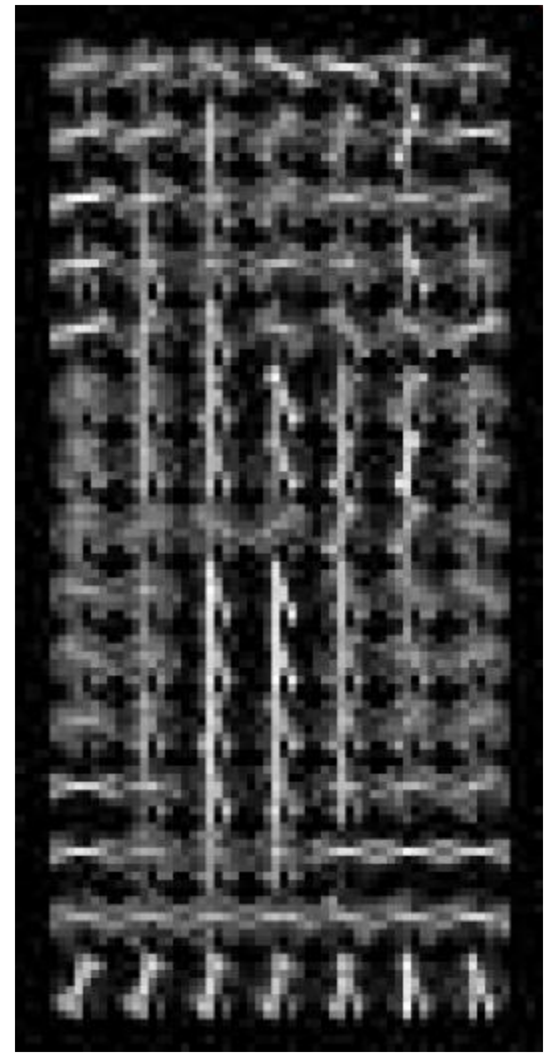
Sobel

Slides by Pete Barnum

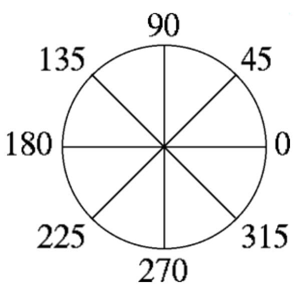
Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05



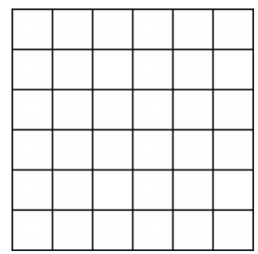
- Histogram of gradient orientations



Orientation: 9 bins (for unsigned angles)



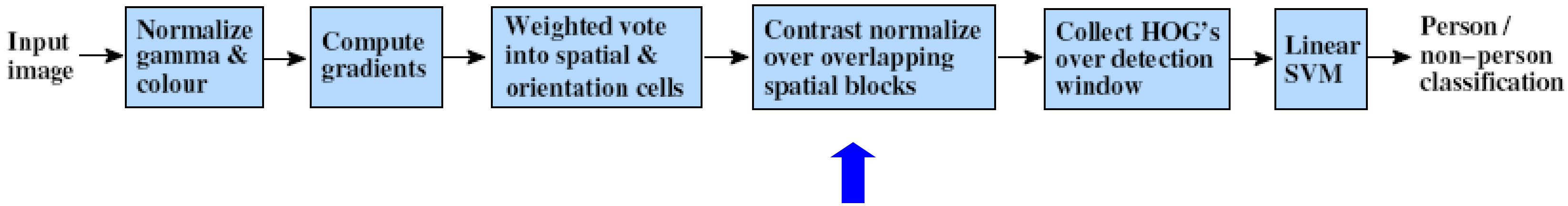
Histograms in 8x8 pixel cells



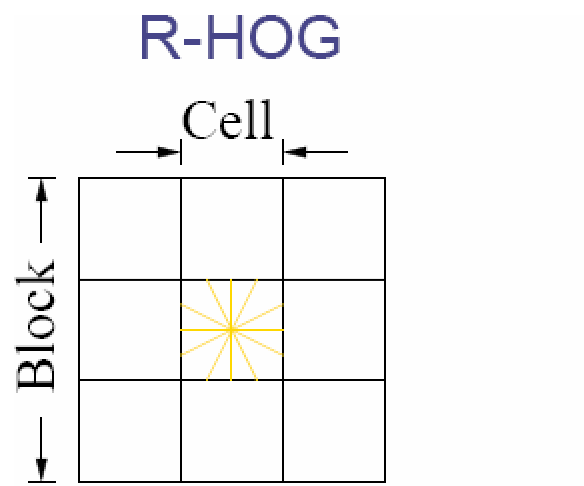
- Votes weighted by magnitude
- Bilinear interpolation between cells

Slides by Pete Barnum

Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05



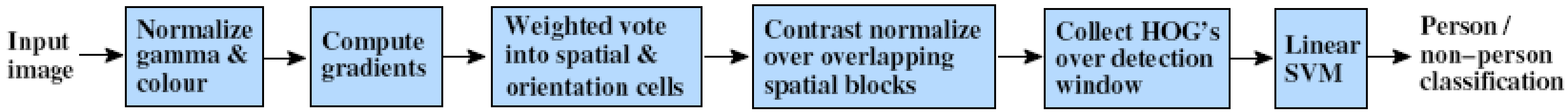
Normalize with respect to surrounding cells



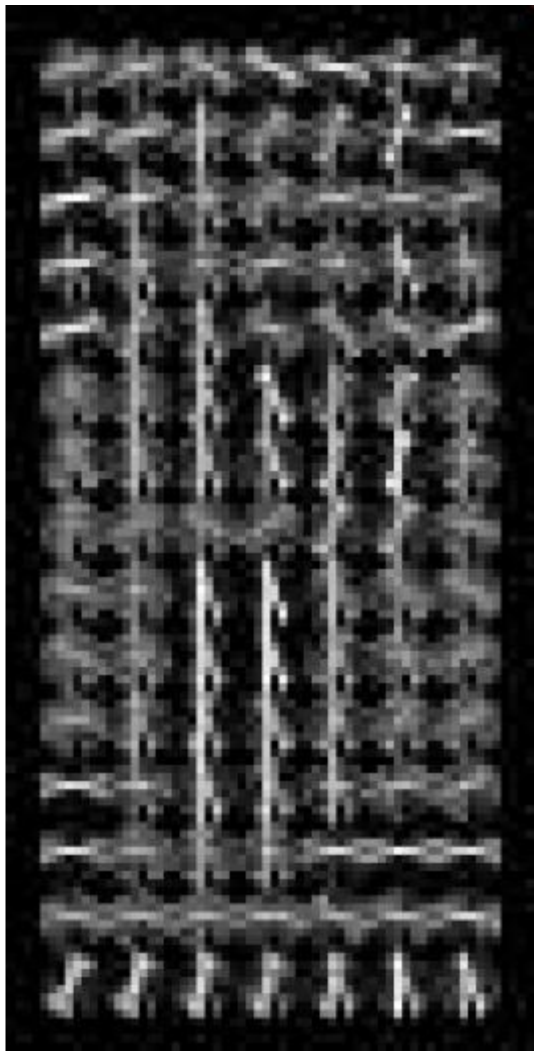
$$L2-norm : v \longrightarrow v/\sqrt{||v||_2^2+\epsilon^2}$$

Slides by Pete Barnum

Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05



X=



orientations

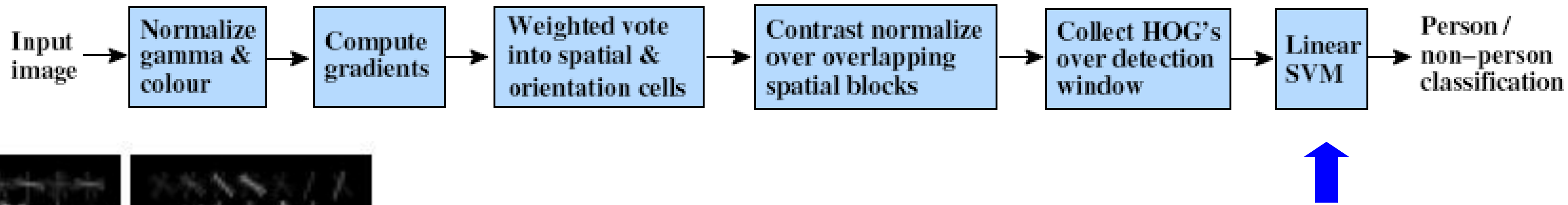
features = 15 x 7 x 9 x 4 = 3780

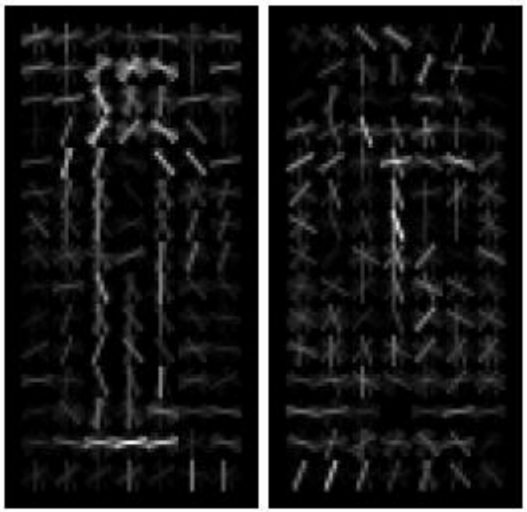
cells

normalizations by neighboring cells

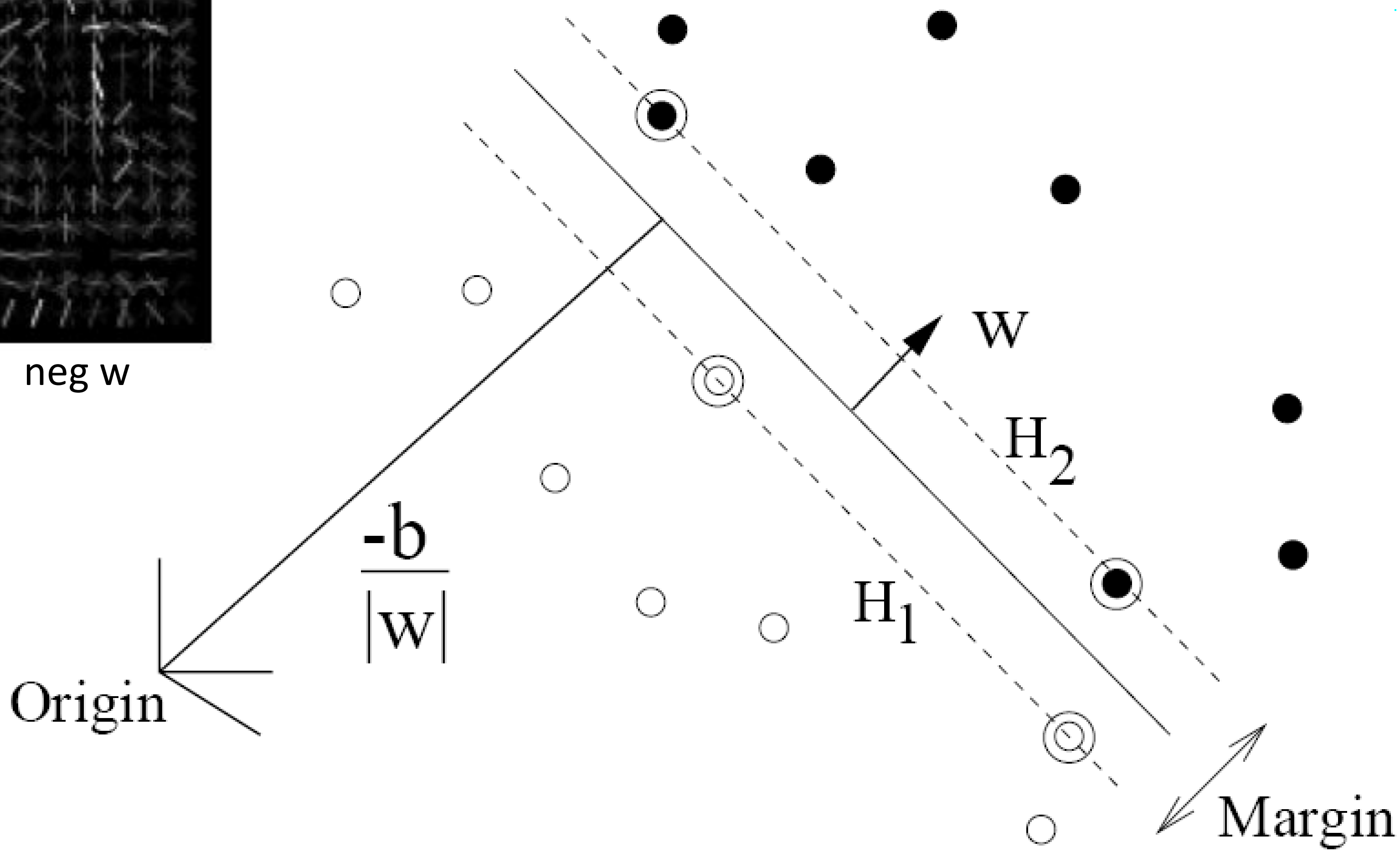
Slides by Pete Barnum

Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05



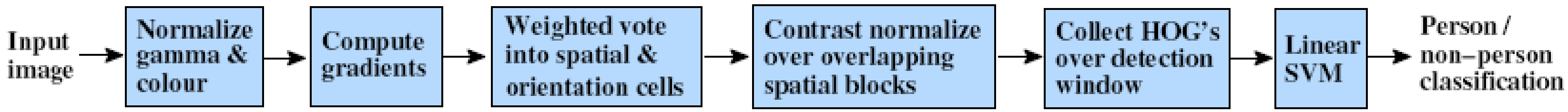


pos w   neg w



Slides by Pete Barnum

Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05



$$0.16 = w^T x - b$$

$$sign(0.16) = 1$$

$=>$ pedestrian

Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05

Detection examples

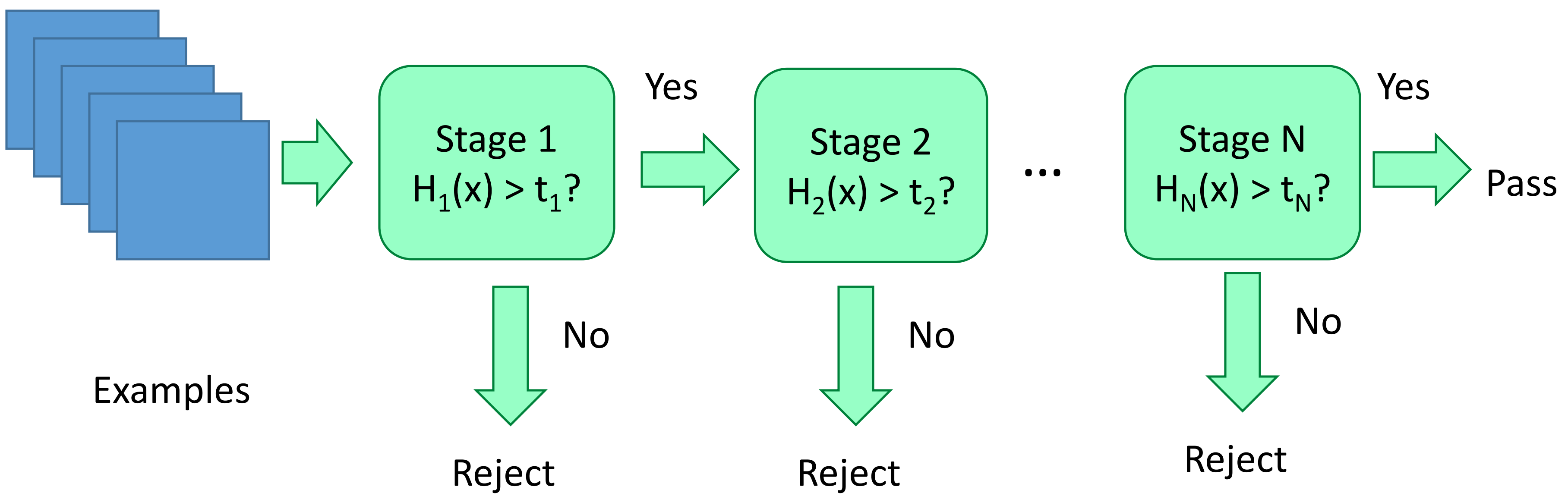
Viola-Jones sliding window detector

Fast detection through two mechanisms

- Quickly eliminate unlikely windows
- Use features that are fast to compute

Viola and Jones. Rapid Object Detection using a Boosted Cascade of Simple Features (2001).

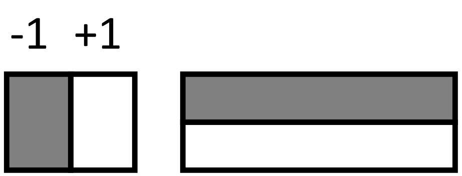
Cascade for Fast Detection



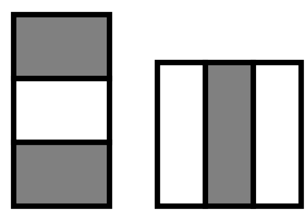
- Choose threshold for low false negative rate
- Fast classifiers early in cascade
- Slow classifiers later, but most examples don’t get there

Features that are fast to compute

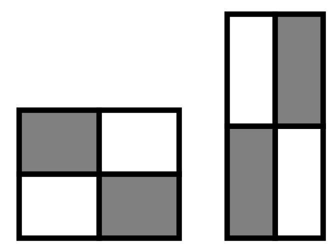
- "Haar-like features"
  - Differences of sums of intensity
  - Thousands, computed at various positions and scales within detection window



Two-rectangle features



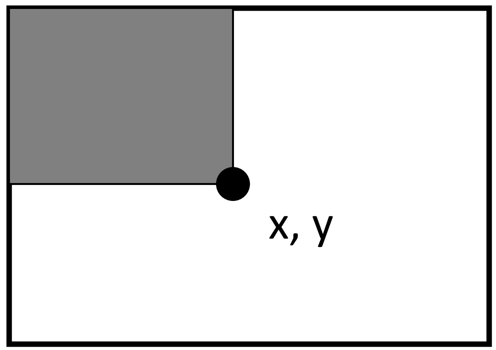
Three-rectangle features



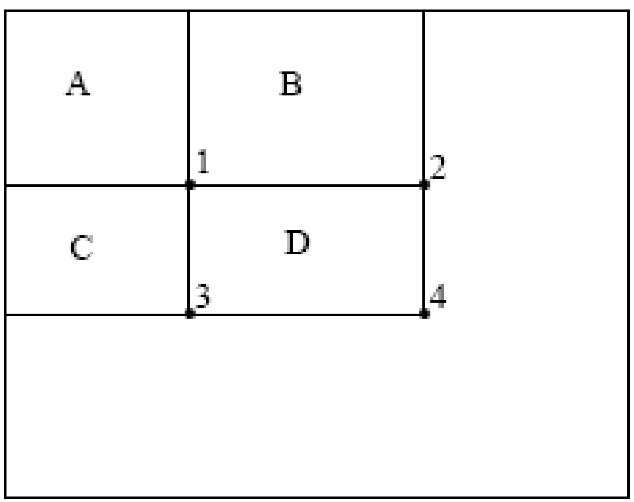
Etc.

Integral Images

- `ii = cumsum(cumsum(im, 1), 2)`



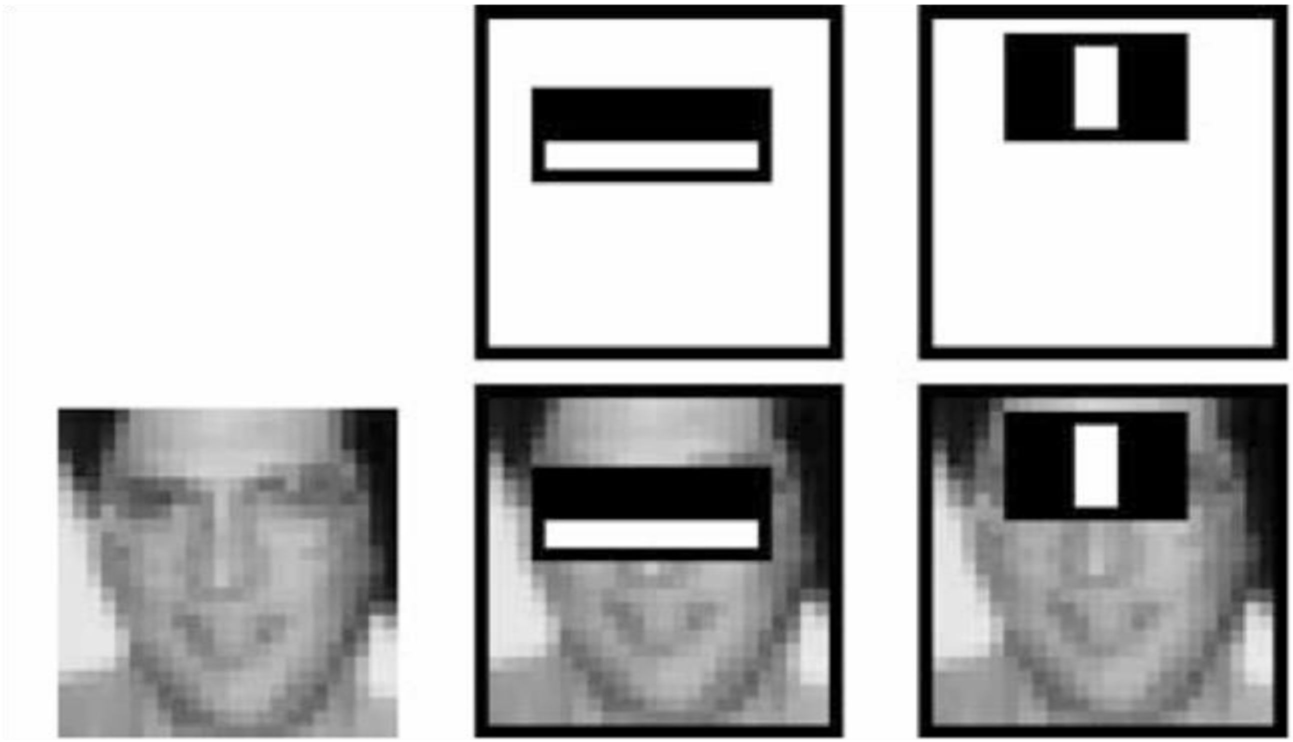
ii(x,y) = Sum of the values in the grey region



How to compute B-A?

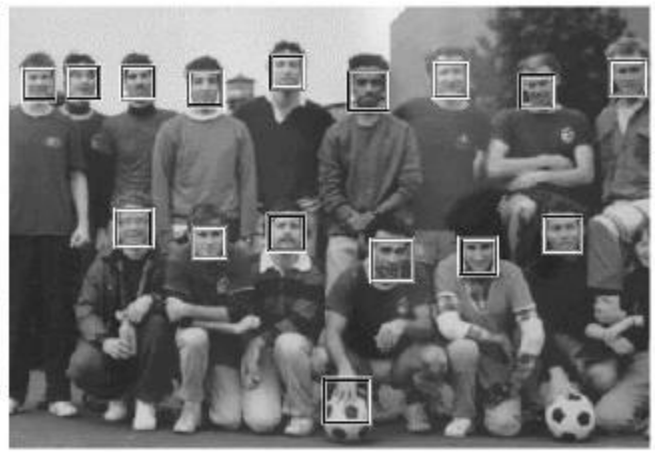
How to compute A+D-B-C?

Top 2 selected features



Viola Jones Results

Speed = 15 FPS (in 2001)



Detector \ False detections	10	31	50	65	78	95	167
Viola-Jones	76.1%	88.4%	91.4%	92.0%	92.1%	92.9%	93.9%
Viola-Jones (voting)	81.1%	89.7%	92.1%	93.1%	93.1%	93.2 %	93.7%
Rowley-Baluja-Kanade	83.2%	86.0%	-	-	-	89.2%	90.1%
Schneiderman-Kanade	-	-	-	94.4%	-	-	-
Roth-Yang-Ahuja	-	-	-	-	(94.8%)	-	-

MIT + CMU face dataset

Something to think about…

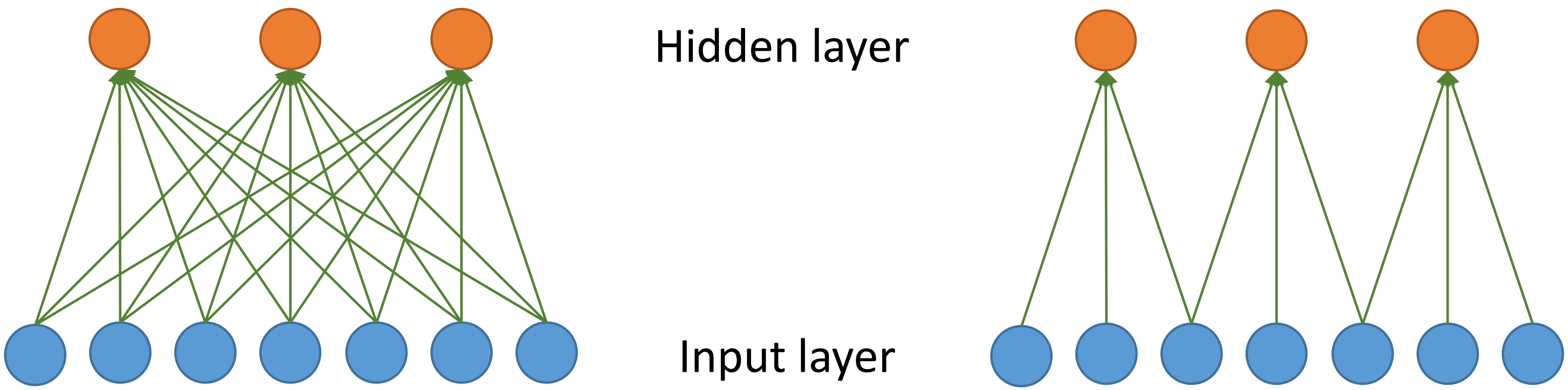
- Sliding window detectors work
  - *very well* for faces
  - *fairly well* for cars and pedestrians
  - *badly* for cats and dogs

- Why are some classes easier than others?

Recap – Convolutional layer

- Convolutional layer
    1. Local connectivity
    2. Weight sharing

Local Connectivity

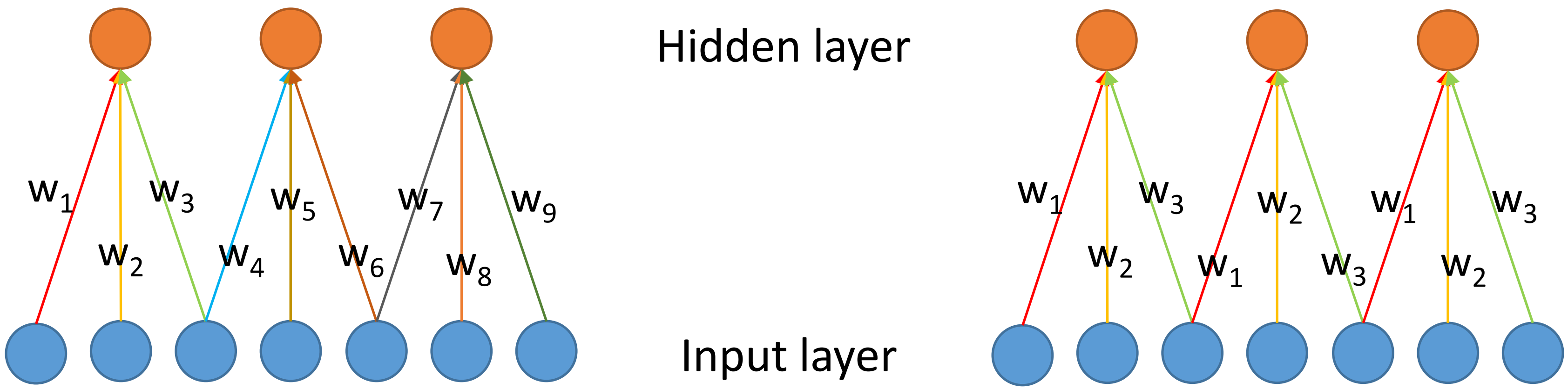


Global connectivity

Local connectivity

- # input units (neurons): 7
- # hidden units: 3
- Number of parameters
  - Global connectivity: 3 x 7 = 21
  - Local connectivity: 3 x 3 = 9

Weight Sharing

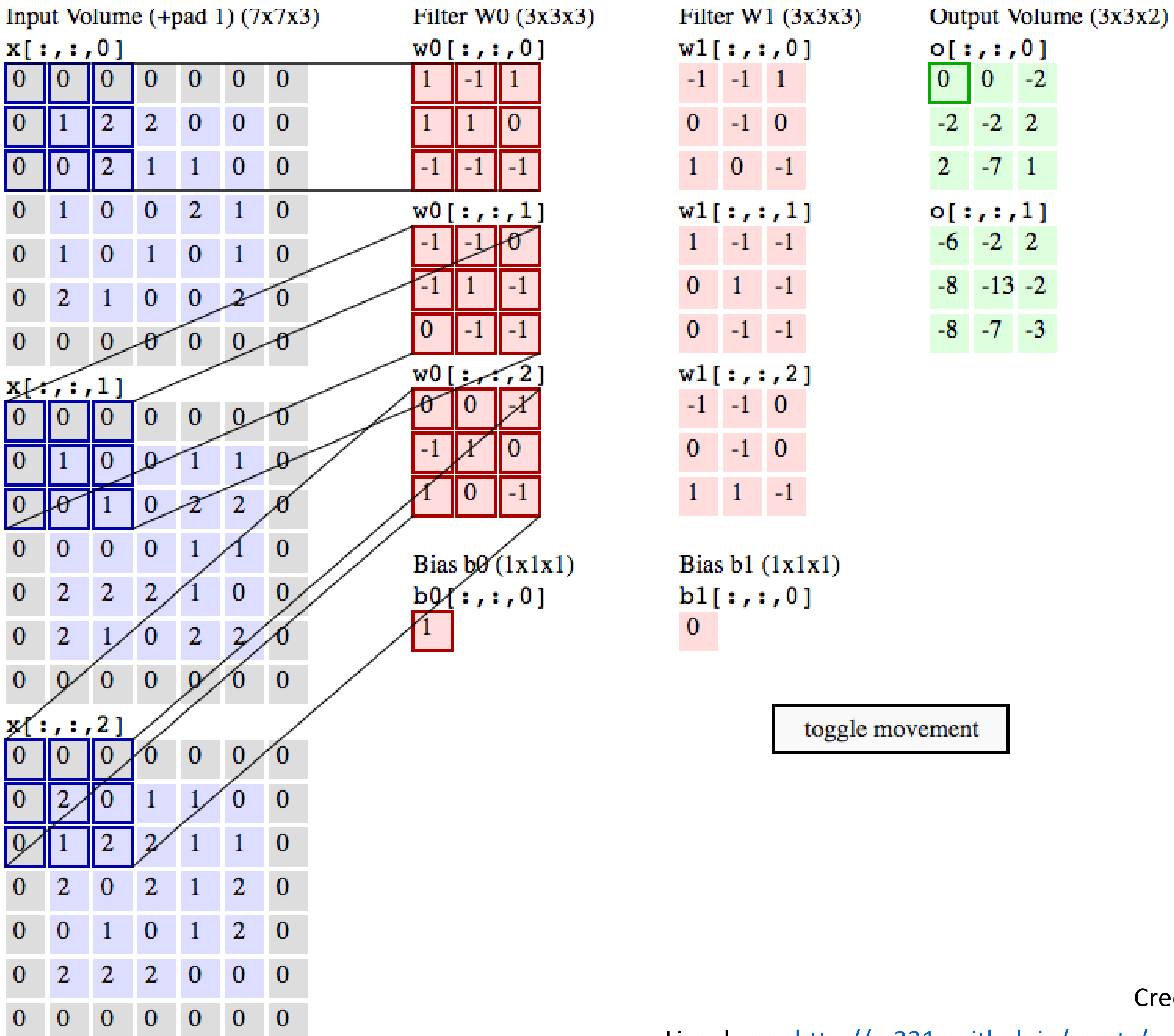


Without weight sharing

With weight sharing

- # input units (neurons): 7
- # hidden units: 3
- Number of parameters
  - Without weight sharing: 3 x 3 = 9
  - With weight sharing : 3 x 1 = 3

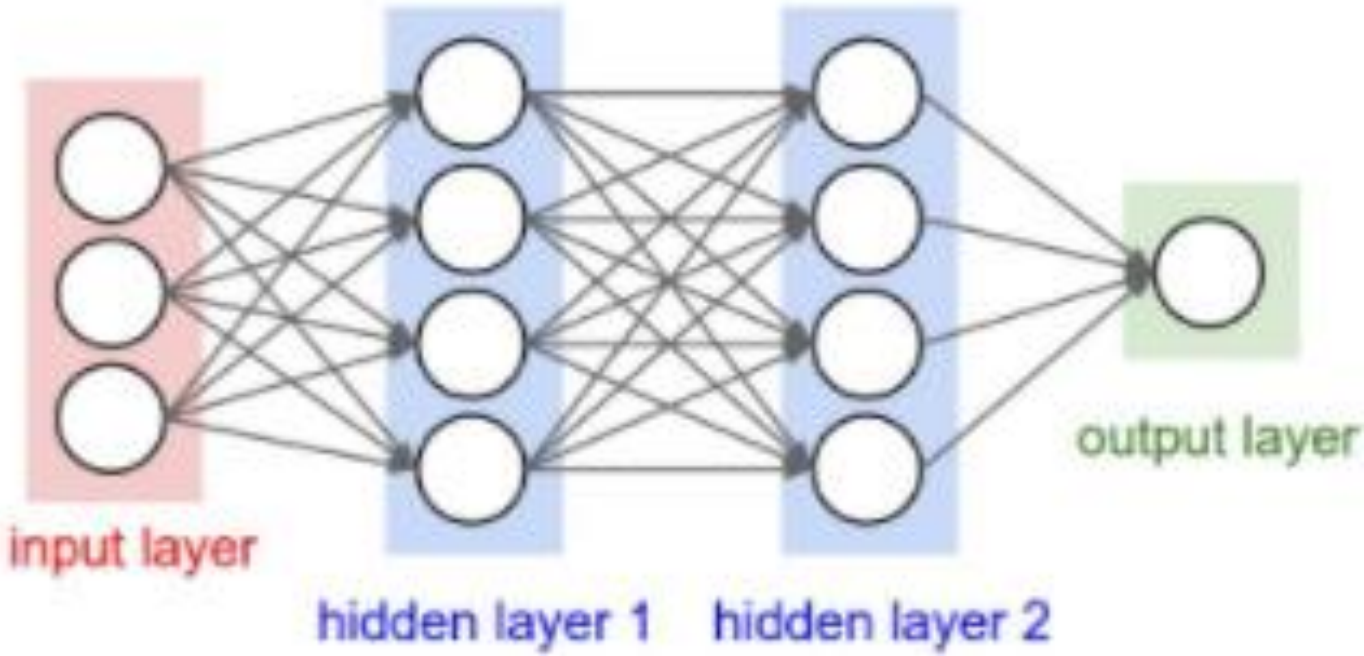
How it works?



Credit: Andrej Karpathy

Live demo: http://cs231n.github.io/assets/conv-demo/index.html

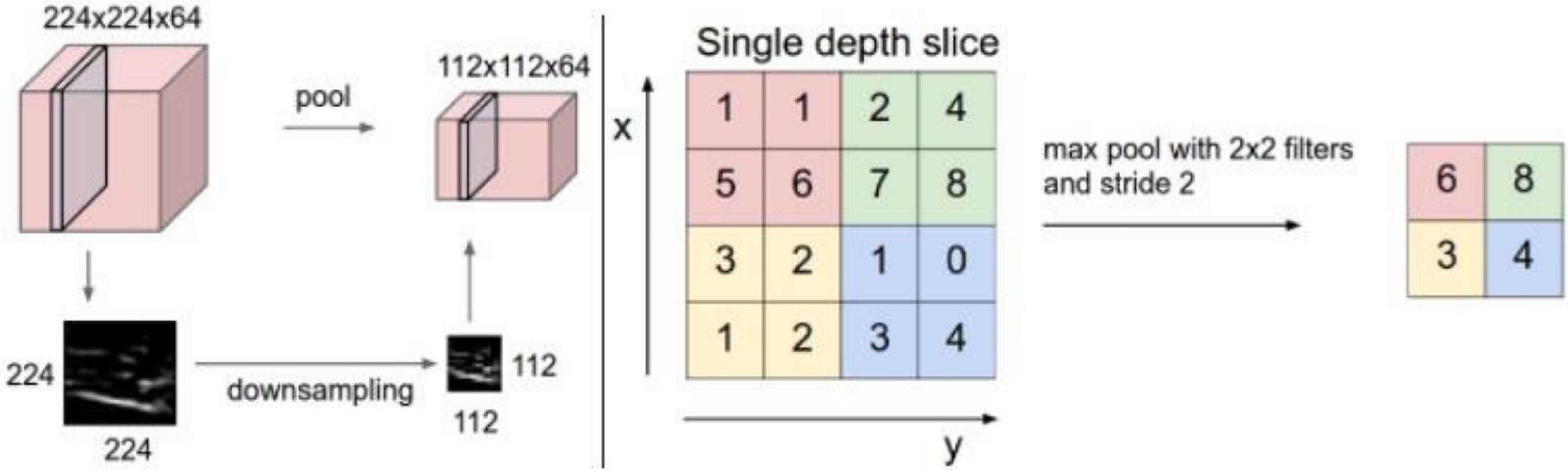
Recap – Fully-connected layer



- Each output node is connected to all the input nodes
- Fixed number of input nodes
- Fixed number of output nodes

Credit: Andrej Karpathy

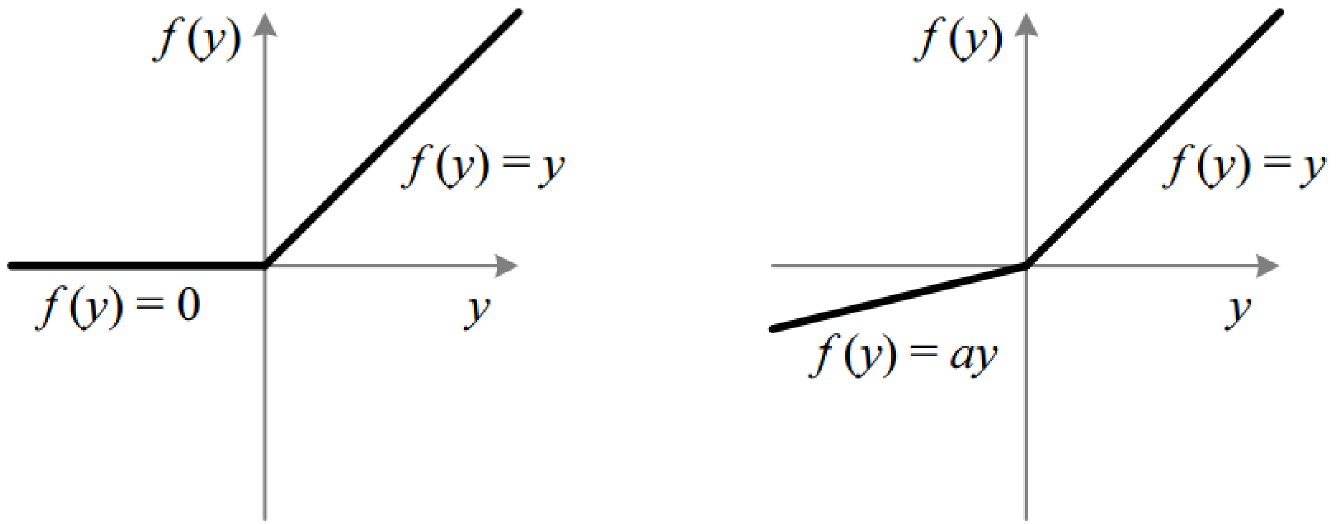
Recap – Pooling layer



- Reduce the feature size
- Introduce a bit invariance (translation, rotation)

Credit: Andrej Karpathy

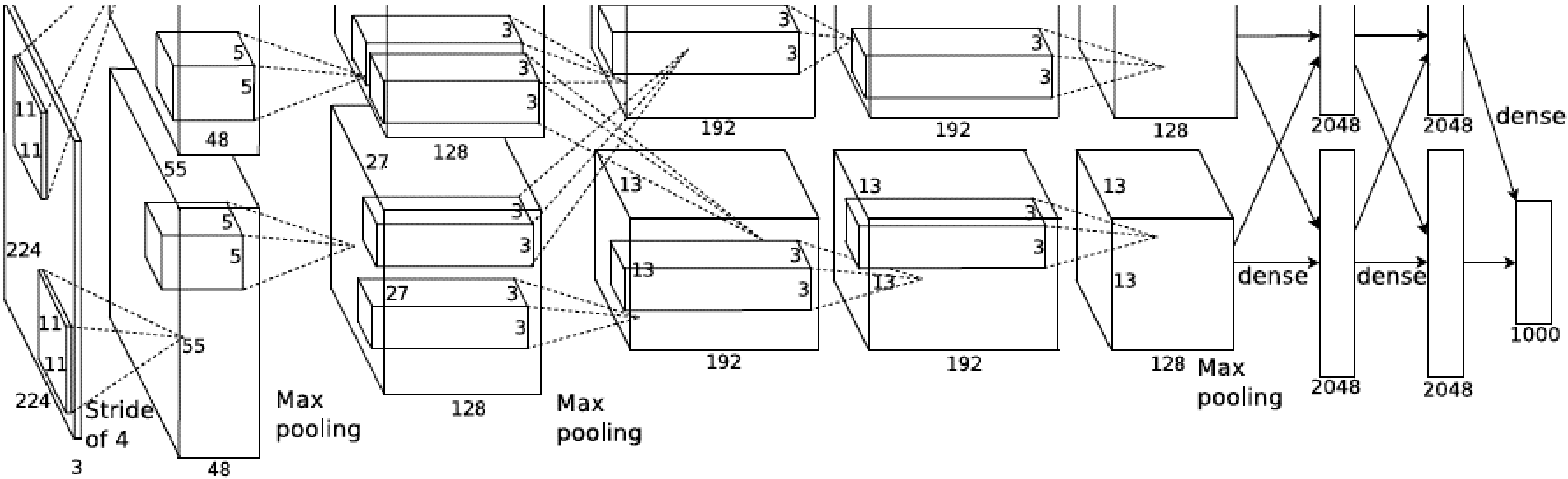
Recap – Activation



- Introduce the non-linearity

Credit: Andrej Karpathy

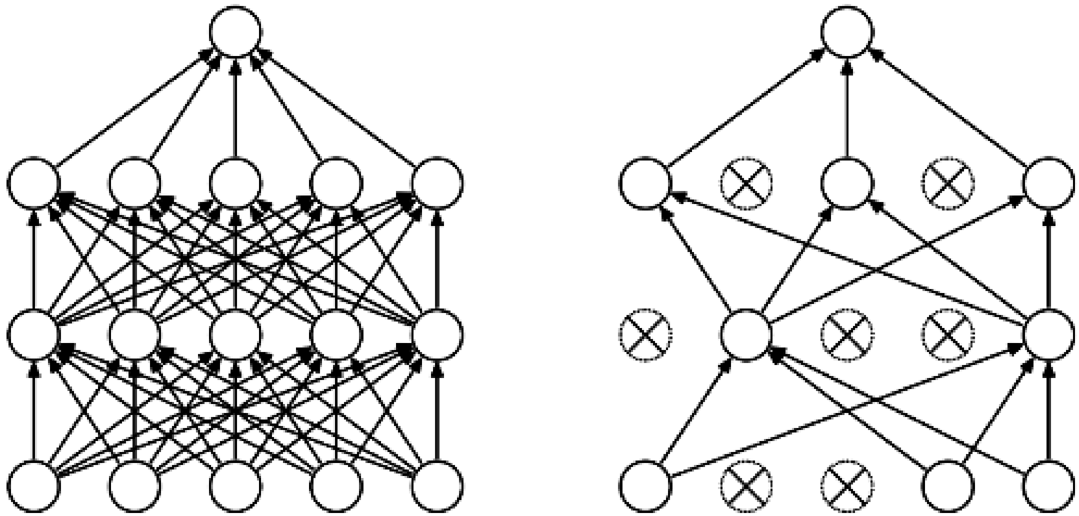
Put them all together



- Train the deep convolutional neural net with simple chain-rule (a.k.a back propagation)

Credit: Andrej Karpathy

Tricks - Dropout



(a) Standard Neural Net (b) After applying dropout.

- Randomly set some nodes to zero during training
  - i.e. Each node will be set to zero with probability p
  - Need to rescale the output, divided by (1-p)
- Usually put it after fc layers, to avoid overfitting

Credit: Andrej Karpathy

Tricks – Batch Normalization

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

- More robust to bad initialization

Credit: Andrej Karpathy

Deep learning methods

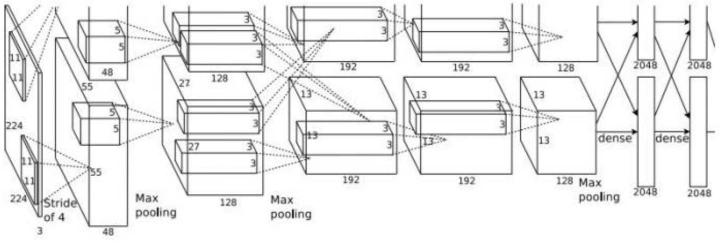
- Let’s have a 2-min break!

CNN as feature extractor

Image credit: Justin Johnson

CNN as feature extractor

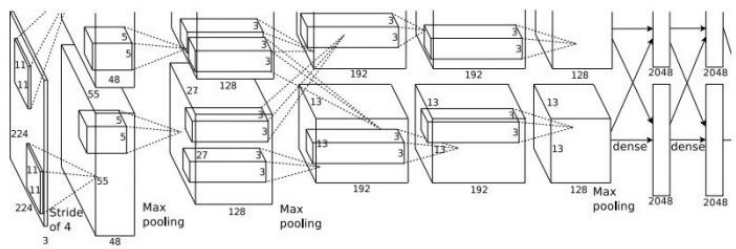
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? NO
Cat? NO
Background? YES

CNN as feature extractor

Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

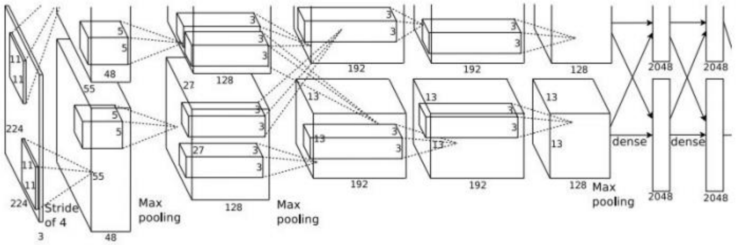


Dog? YES
Cat? NO
Background? NO

Slides by Justin Johnson

CNN as feature extractor

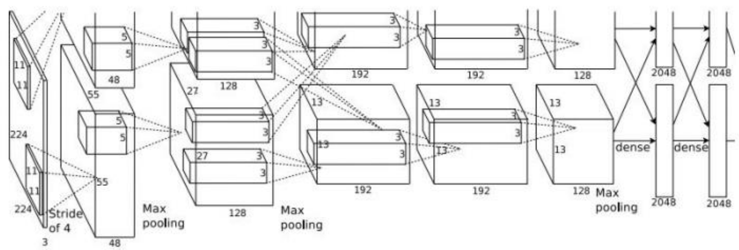
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? YES
Cat? NO
Background? NO

CNN as feature extractor

Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? NO
Cat? YES
Background? NO

Slides by Justin Johnson

CNN as feature extractor

- What could be the problems?

CNN as feature extractor

- What could be the problems?
  - Suppose we have a 600 x 600 image, if sliding window size is 20 x 20, then have (600-20+1) x (600-20+1) = ~330,000 windows

CNN as feature extractor

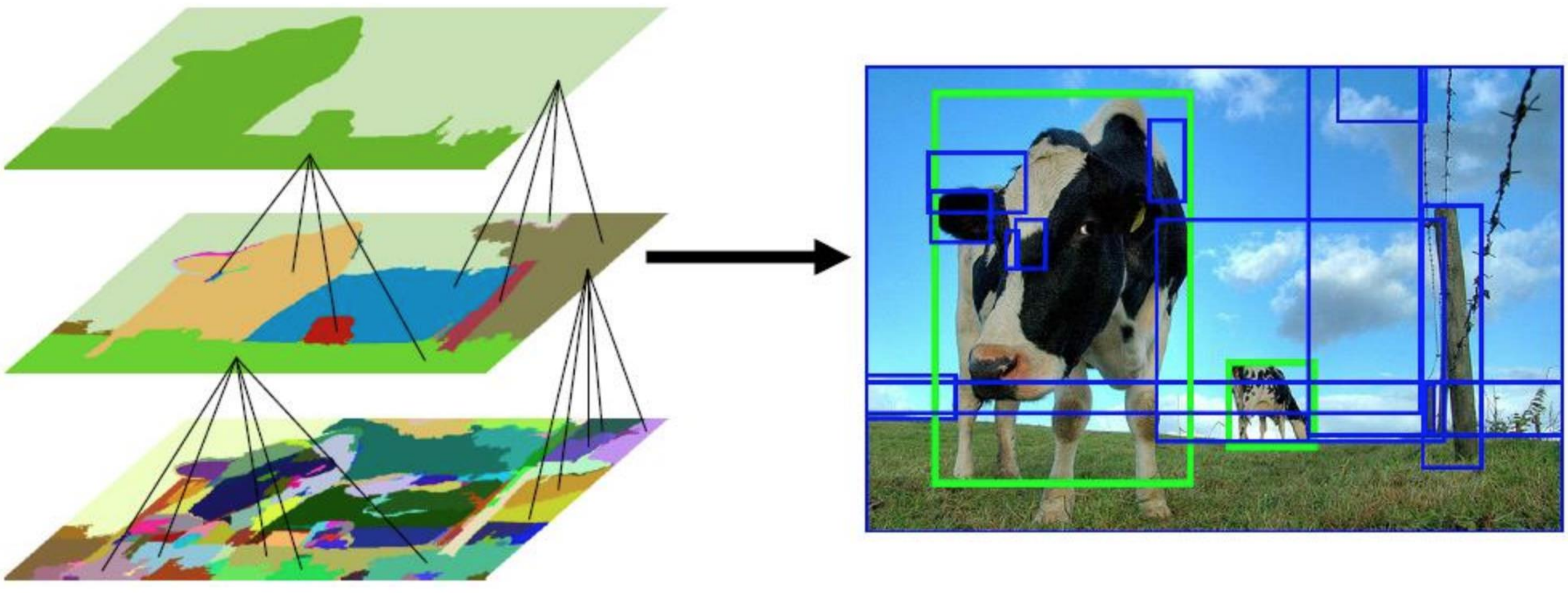
- What could be the problems?
  - Suppose we have a 600 x 600 image, if sliding window size is 20 x 20, then have (600-20+1) x (600-20+1) = ~330,000 windows
  - Sometimes we want to have more accurate results -> multi-scale detection
    - Resize image
    - Multi-scale sliding window

CNN as feature extractor

- What could be the problems?
  - Suppose we have a 600 x 600 image, if sliding window size is 20 x 20, then have (600-20+1) x (600-20+1) = ~330,000 windows
  - Sometimes we want to have more accurate results -> multi-scale detection
    - Resize image
    - Multi-scale sliding window
  - For each image, we need to do the forward pass in the CNN for ~330,000 times. -> Slow!!!

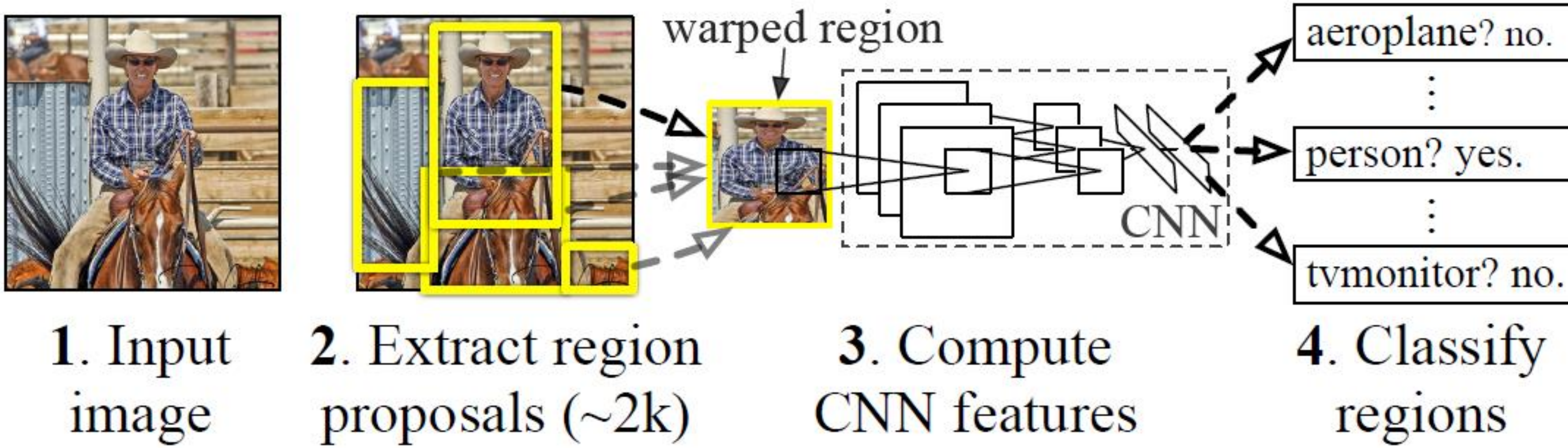
Region Proposal

- Solution
  - Use some fast algorithms to filter out some regions first, only feed the potential region (region proposals) into CNN
  - E.g. selective search



Uijilings et al. IJCV 2013

R-CNN (Girshick et al. CVPR 2014)



- Replace sliding windows with “selective search” region proposals (Uijilings et al. IJCV 2013)
- Extract rectangles around regions and resize to 227x227
- Extract features with fine-tuned CNN (that was initialized with network trained on ImageNet before training)
- Classify last layer of network features with SVM, refine bounding box localization (bbox regression) simultaneously

http://arxiv.org/pdf/1311.2524.pdf

Bounding Box Regression

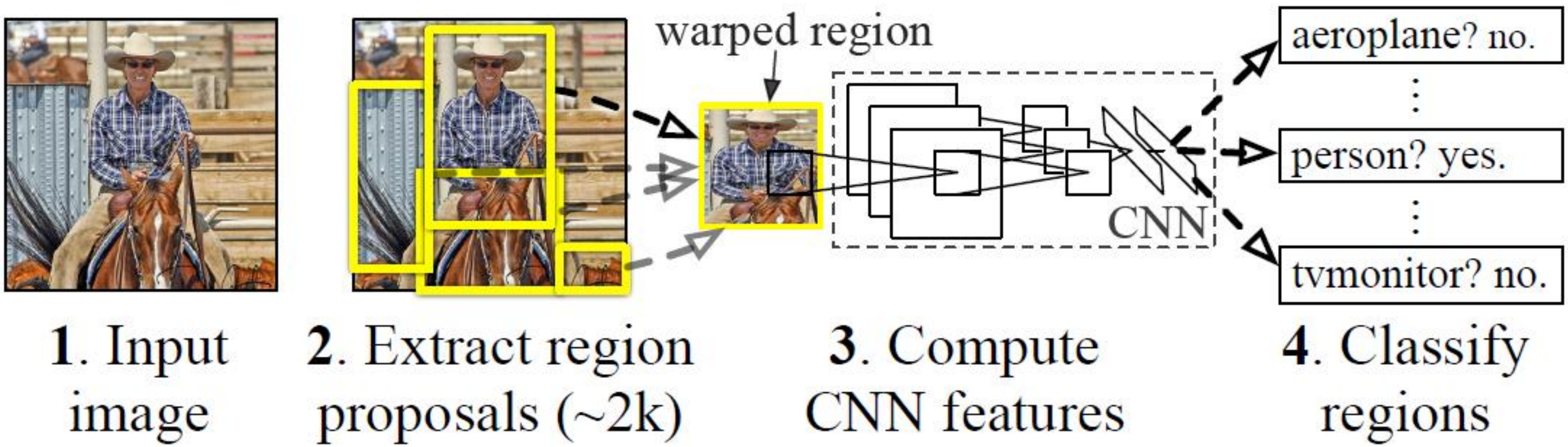
- Intuition
  - If you observe part of the object, according to the seen examples, you should be able to refine the localization
  - E.g. given the red box below, since you've seen many airplanes, you know this is not a good localization, you will adjust it to the green one

Bounding Box Regression

- Intuition
  - If you observe part of the object, according to the seen examples, you should be able to refine the localization
  - E.g. given the red box below, since you've seen many airplanes, you know this is not a good localization, you will adjust it to the green one

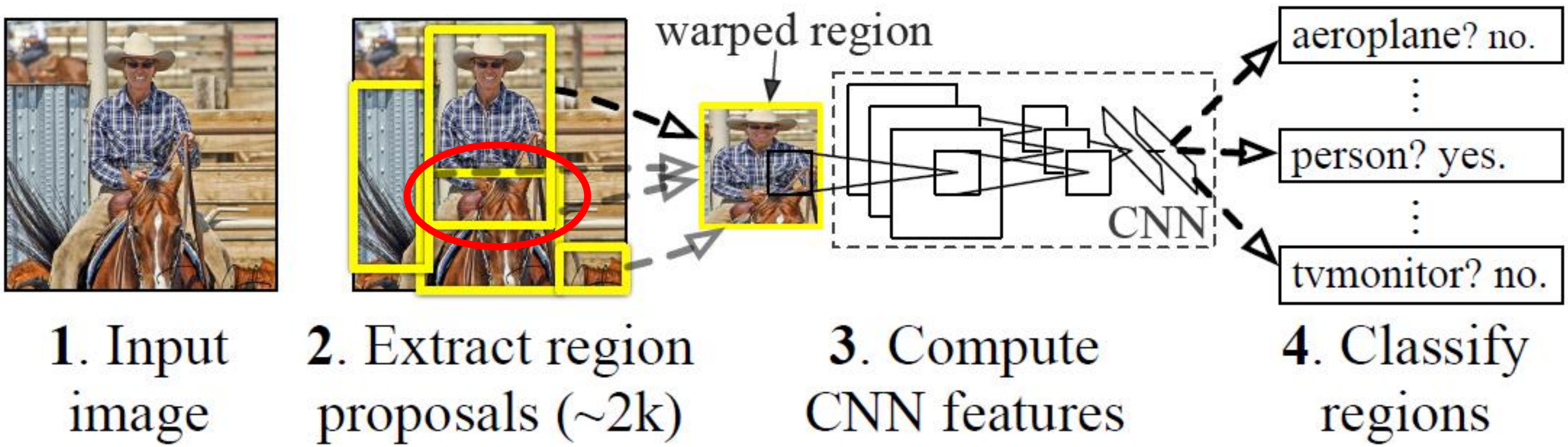
R-CNN (Girshick et al. CVPR 2014)

- What could be the problems?



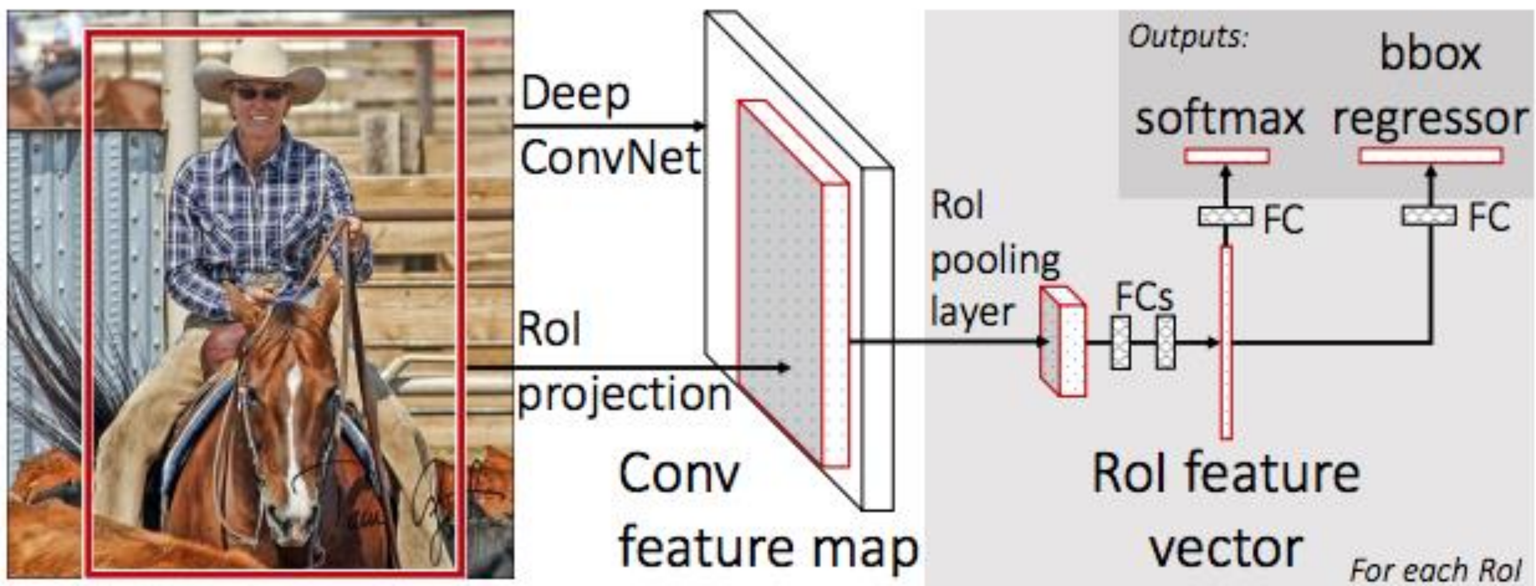
R-CNN (Girshick et al. CVPR 2014)

- What could be the problems?
  - Repetitive computation! For overlapping regions, we feed it multiple times into CNN



Fast R-CNN (Girshick ICCV 2015)

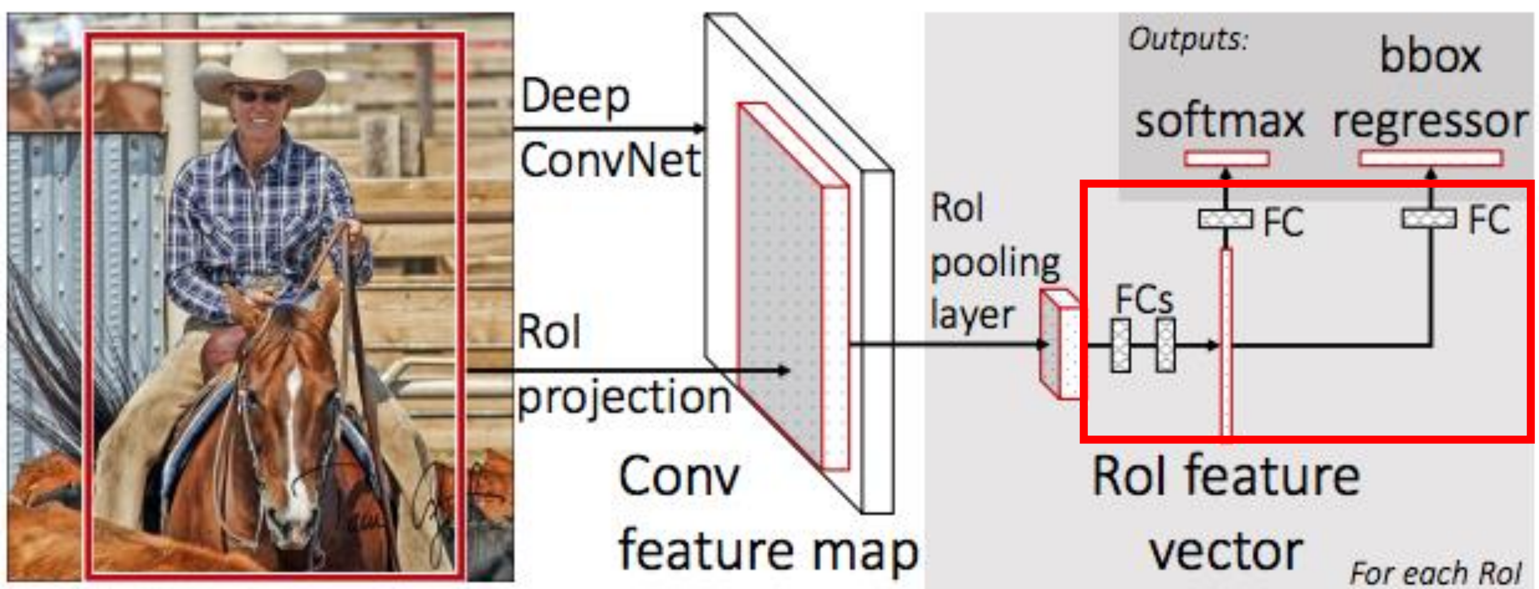
- Solution
  - Why not feed the whole image into CNN only once! Then crop features instead of image itself



https://arxiv.org/pdf/1504.08083.pdf

Fast R-CNN (Girshick ICCV 2015)

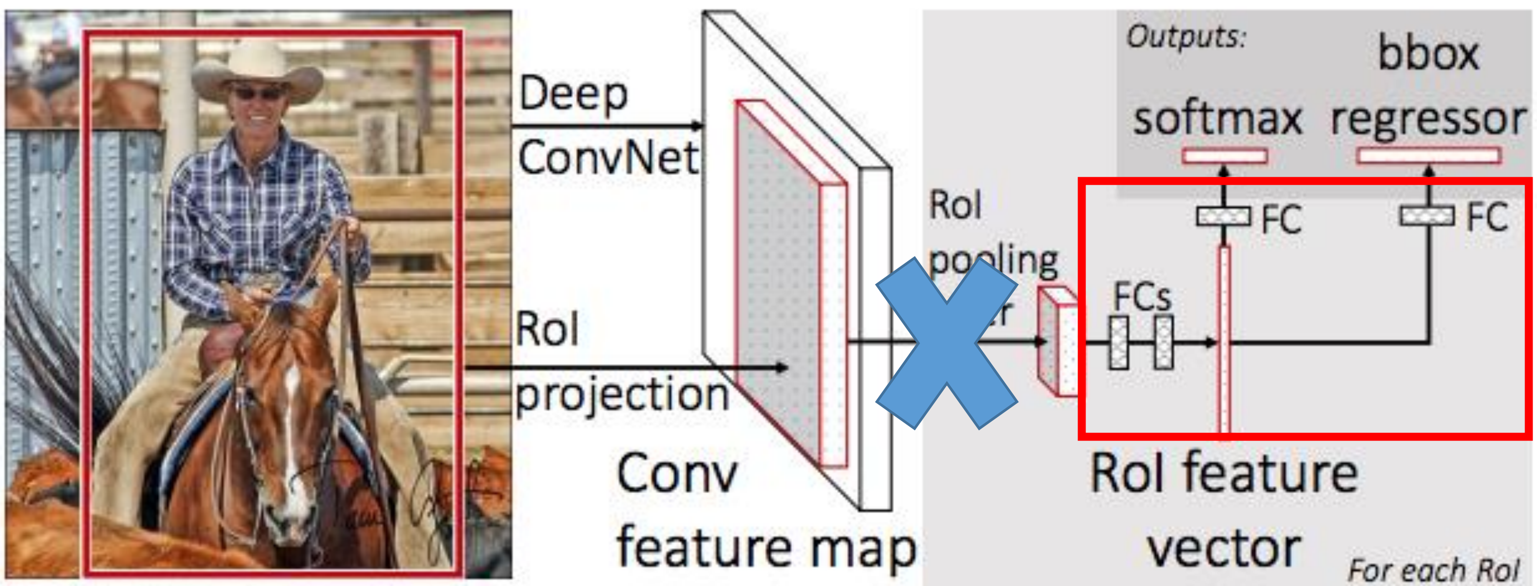
- How to crop features?
  - Since we have fully-connected layers, the size of feature map for each bounding box should be a fixed number



https://arxiv.org/pdf/1504.08083.pdf

Fast R-CNN (Girshick et al. ICCV 2015)

- How to crop features?
  - Since we have fully-connected layers, the size of feature map for each bounding box should be a fixed number
  - Resize/Interpolate the feature map as fixed size?
    - Not optimal. This operation is hard to backprop -> we cannot train the conv layers for this problem



https://arxiv.org/pdf/1504.08083.pdf

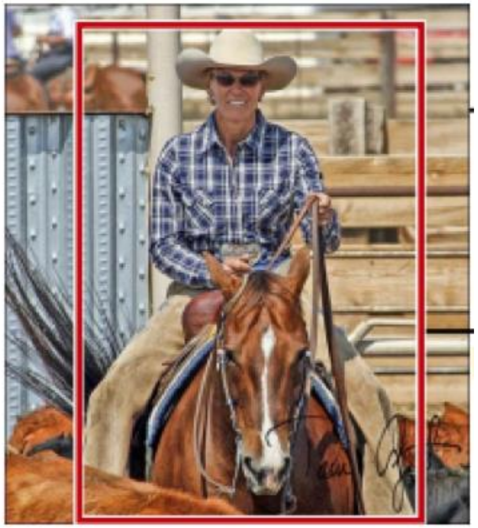
Fast R-CNN (Girshick et al. ICCV 2015)

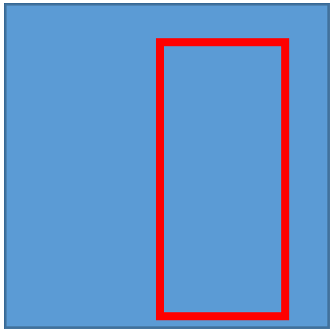
- How to crop features?
  - Since we have fully-connected layers, the size of feature map for each bounding box should be a fixed number
  - Resize/Interpolate the feature map as fixed size?
    - Not optimal. This operation is hard to backprop -> we cannot train the conv layers for this problem
  - RoI (Region of Interest) Pooling

https://arxiv.org/pdf/1504.08083.pdf

RoI Pooling

- Step 1: Get bounding box for feature map from bounding box for image
  - Due the (down)convolution / pooling operations, feature map would have a smaller size than the original image



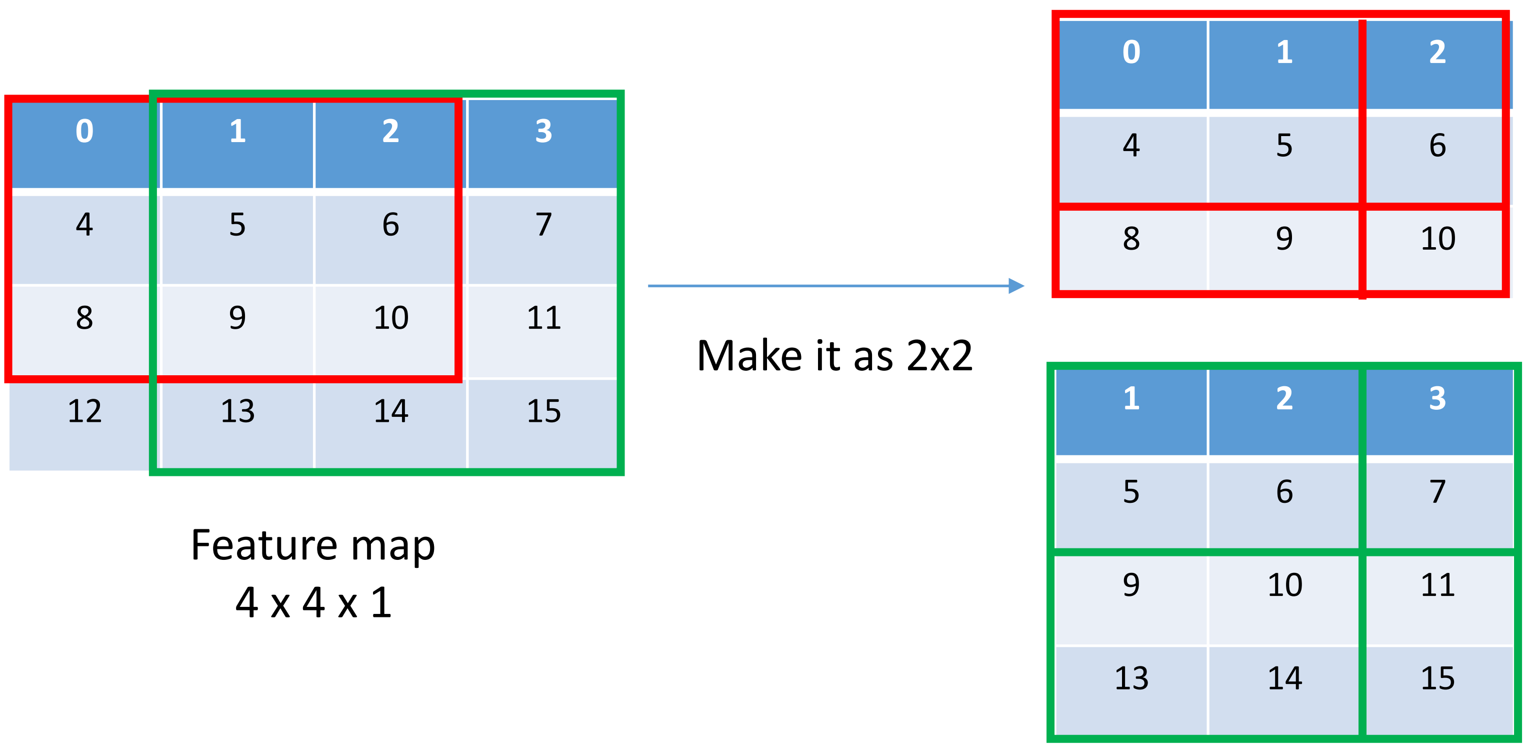


Feature map

https://arxiv.org/pdf/1504.08083.pdf

RoI Pooling

- Step 2: Divide cropped feature map into fixed number of sub-regions
  - The last column and last row might be smaller



https://arxiv.org/pdf/1504.08083.pdf

RoI Pooling

- Step 3: For each sub-region, perform max pooling (pick the max one)

0	1	2
4	5	6
8	9	10

Max pooling

5	6
9	10

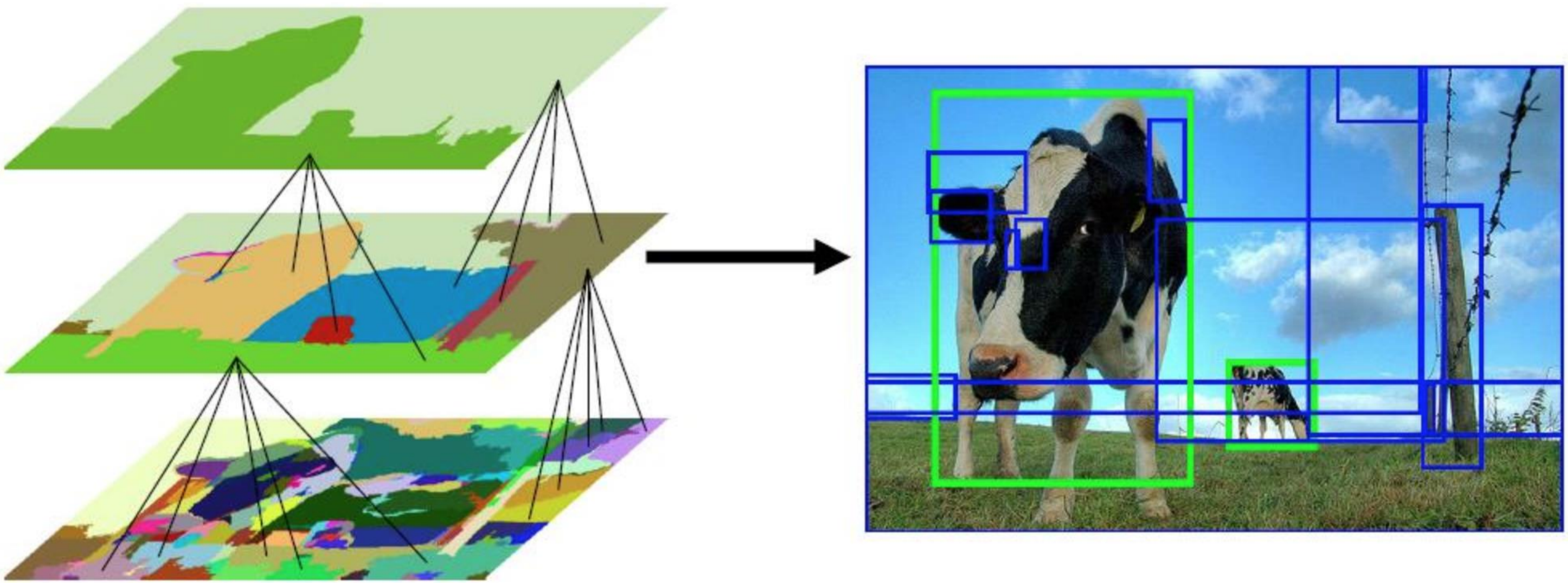
https://arxiv.org/pdf/1504.08083.pdf

Fast R-CNN (Girshick et al. ICCV 2015)

- What could be the problems?

Fast R-CNN (Girshick et al. ICCV 2015)

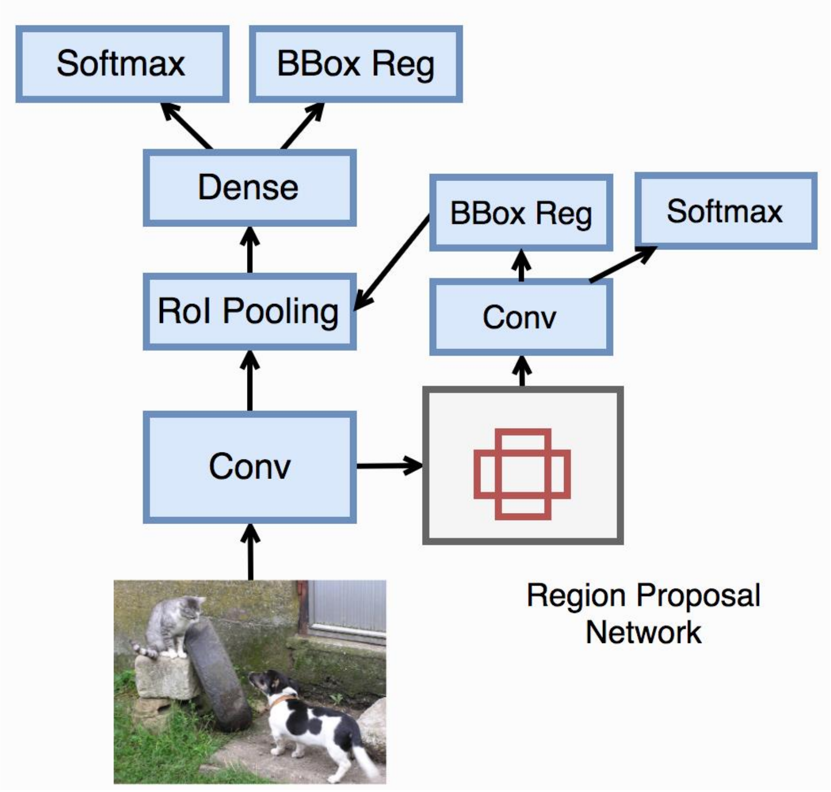
- What could be the problems?
  - Why we need the region proposal pre-processing step? That's not "deep learning" at all. Not cool!



Uijilings et al. IJCV 2013

Faster R-CNN (Ren et al. NIPS 2015)

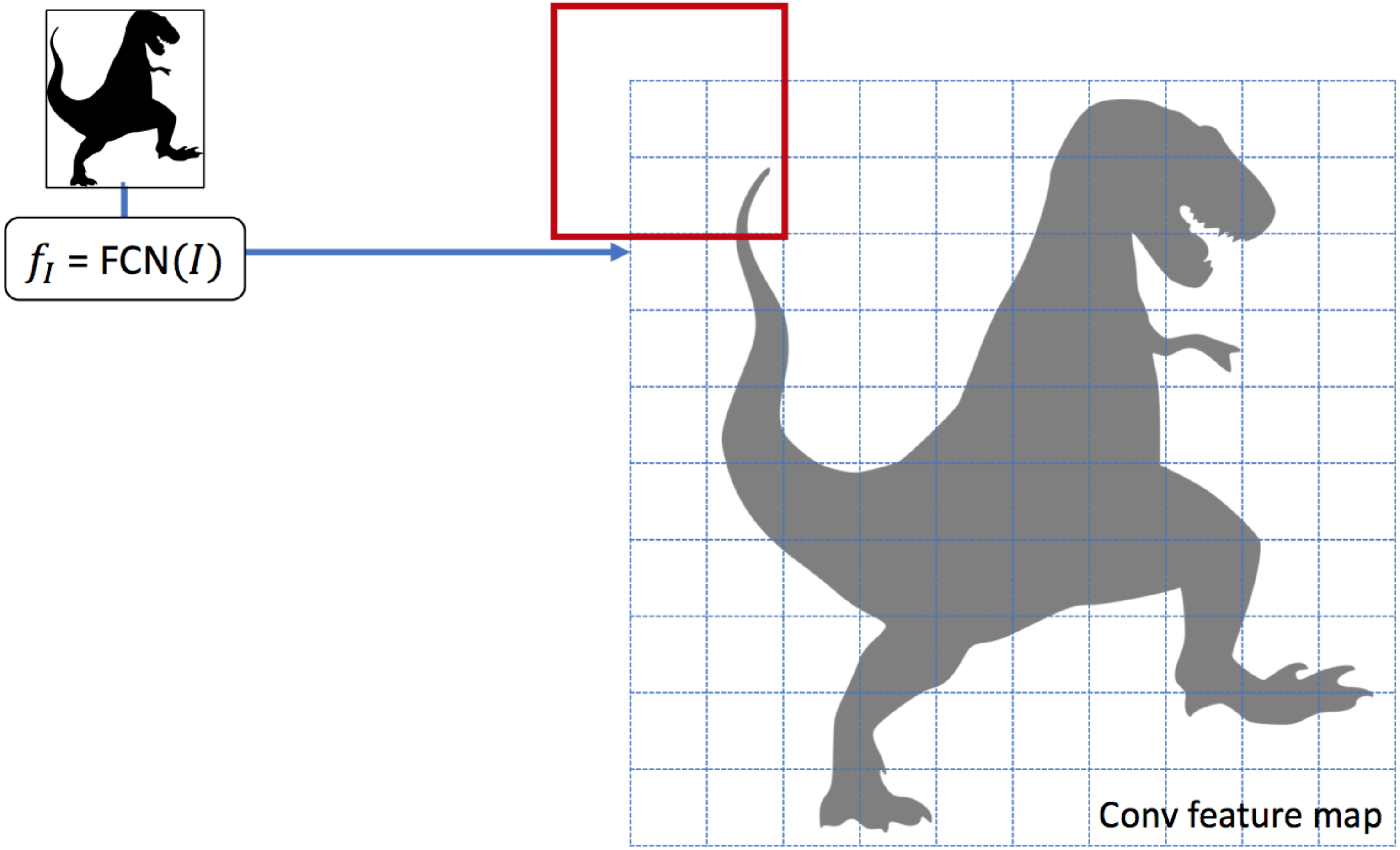
- Solution
  - Why not generate region proposals using CNN??! -> RPN



https://arxiv.org/pdf/1506.01497.pdf

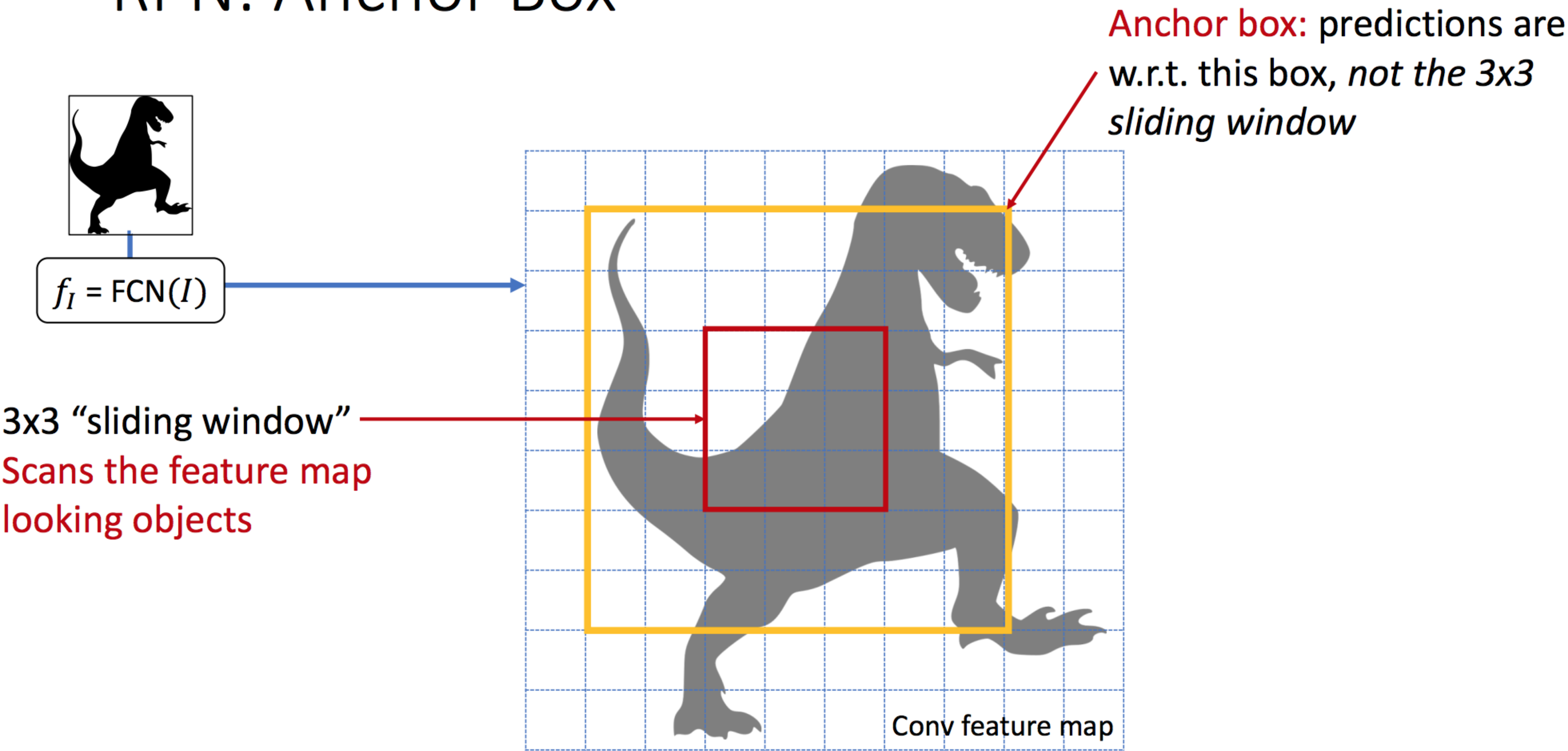
Image credit:
http://zh.gluon.ai/chapter_computer-vision/object-detection.html

RPN: Region Proposal Network



https://arxiv.org/pdf/1506.01497.pdf

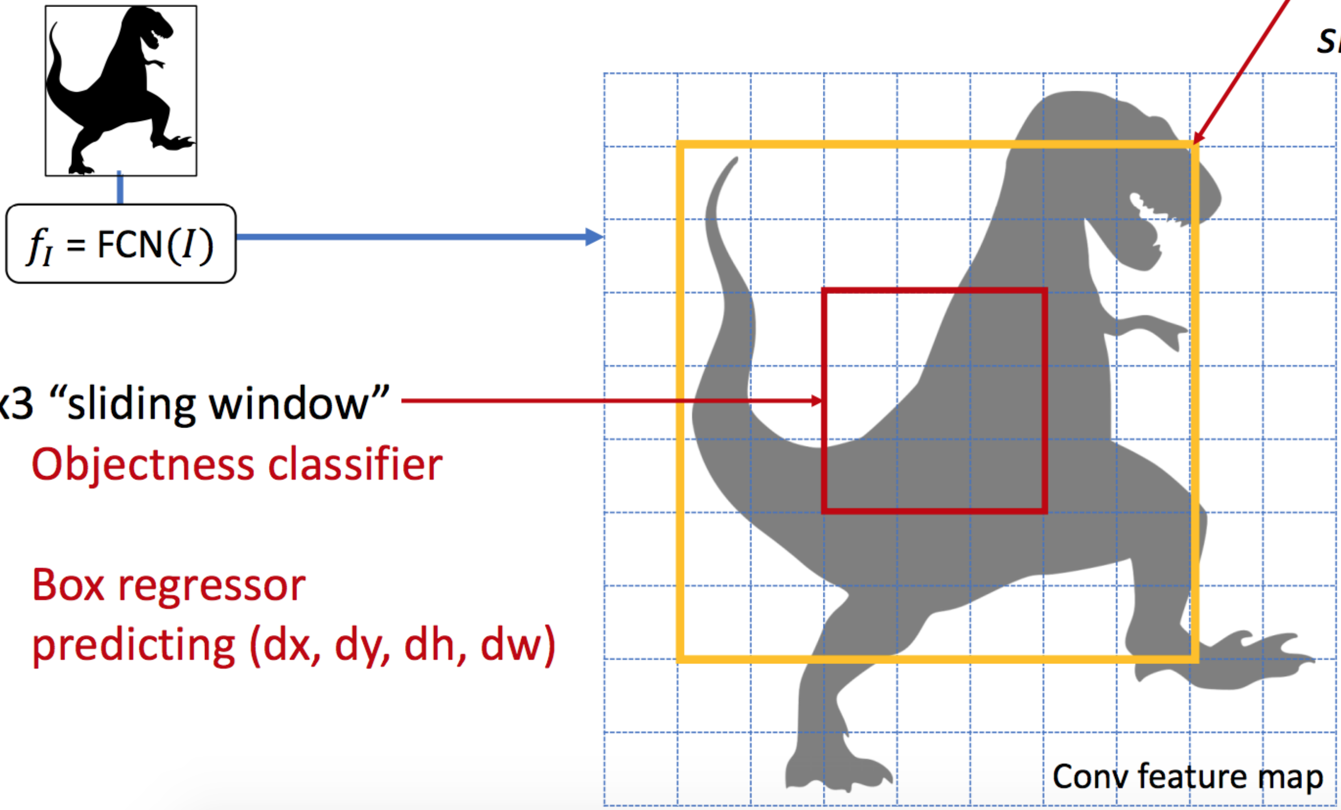
RPN: Anchor Box



https://arxiv.org/pdf/1506.01497.pdf

Slides by Ross Girshick

RPN: Anchor Box



Anchor box: predictions are w.r.t. this box, *not the 3x3 sliding window*

3x3 "sliding window"

- Objectness classifier
- Box regressor predicting (dx, dy, dh, dw)

https://arxiv.org/pdf/1506.01497.pdf

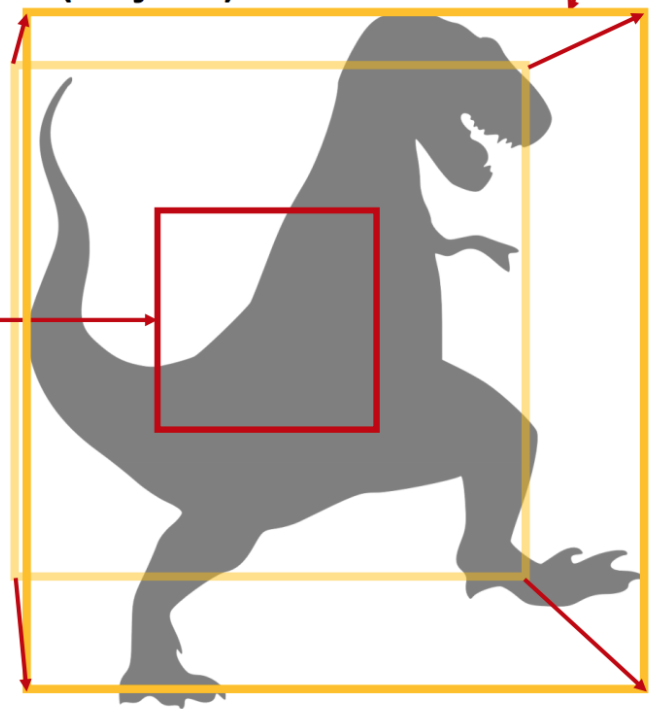
Slides by Ross Girshick

RPN: Prediction (on object)

Anchor box: transformed by box regressor

Objectness score

P(object) = 0.94



3x3 "sliding window"

- Objectness classifier
- Box regressor predicting (dx, dy, dh, dw)

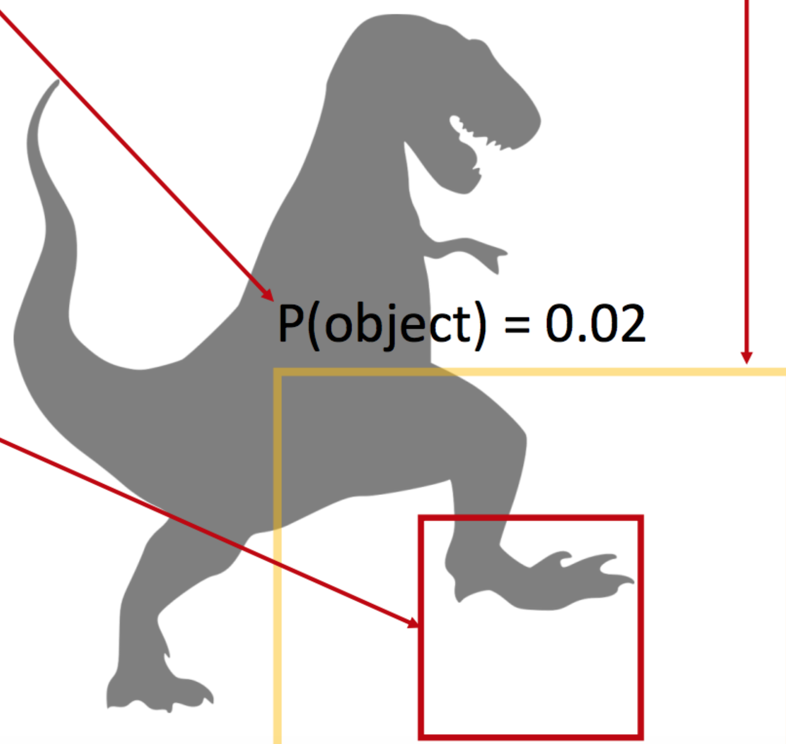
https://arxiv.org/pdf/1506.01497.pdf

Slides by Ross Girshick

RPN: Prediction (off object)

Anchor box: transformed by box regressor

Objectness score



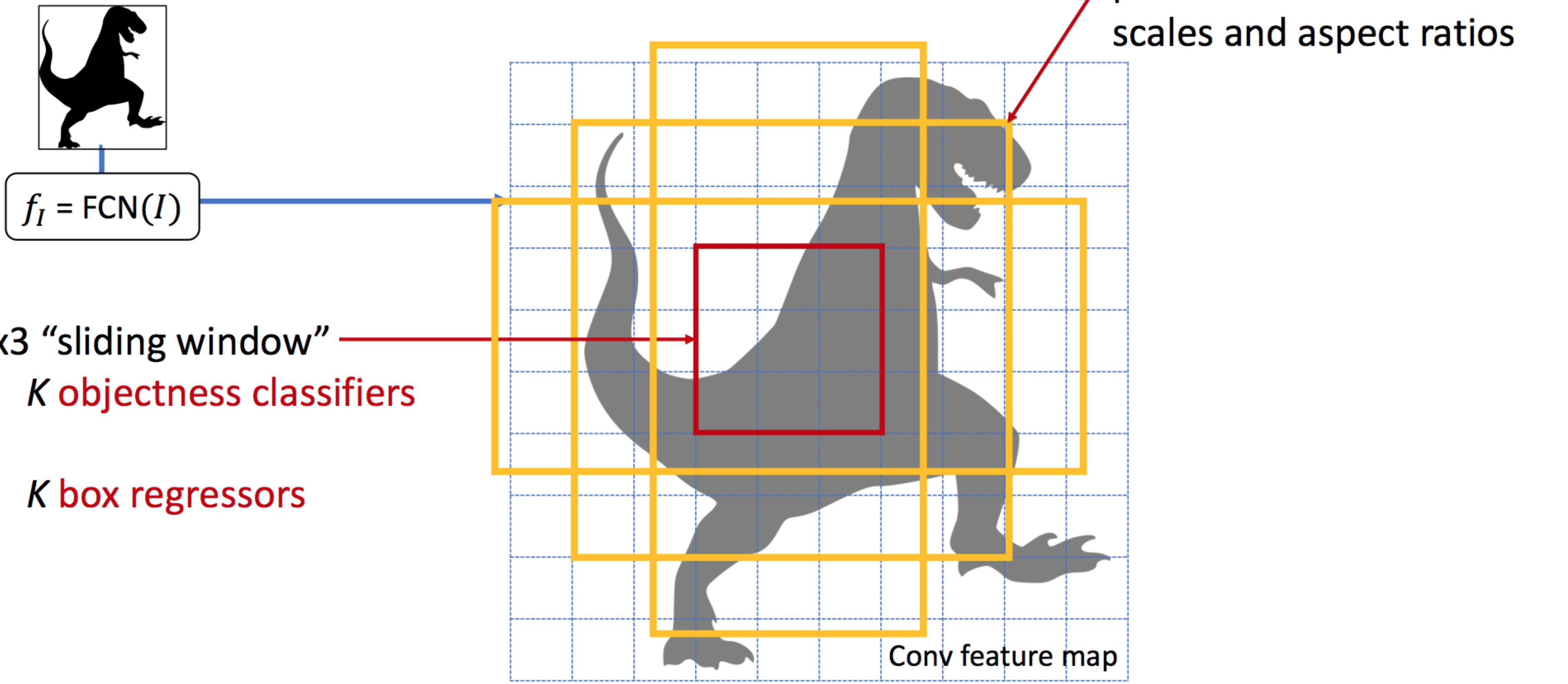
3x3 “sliding window”

- Objectness classifier
- Box regressor predicting (dx, dy, dh, dw)

https://arxiv.org/pdf/1506.01497.pdf

Slides by Ross Girshick

RPN: Multiple Anchors



3x3 "sliding window"

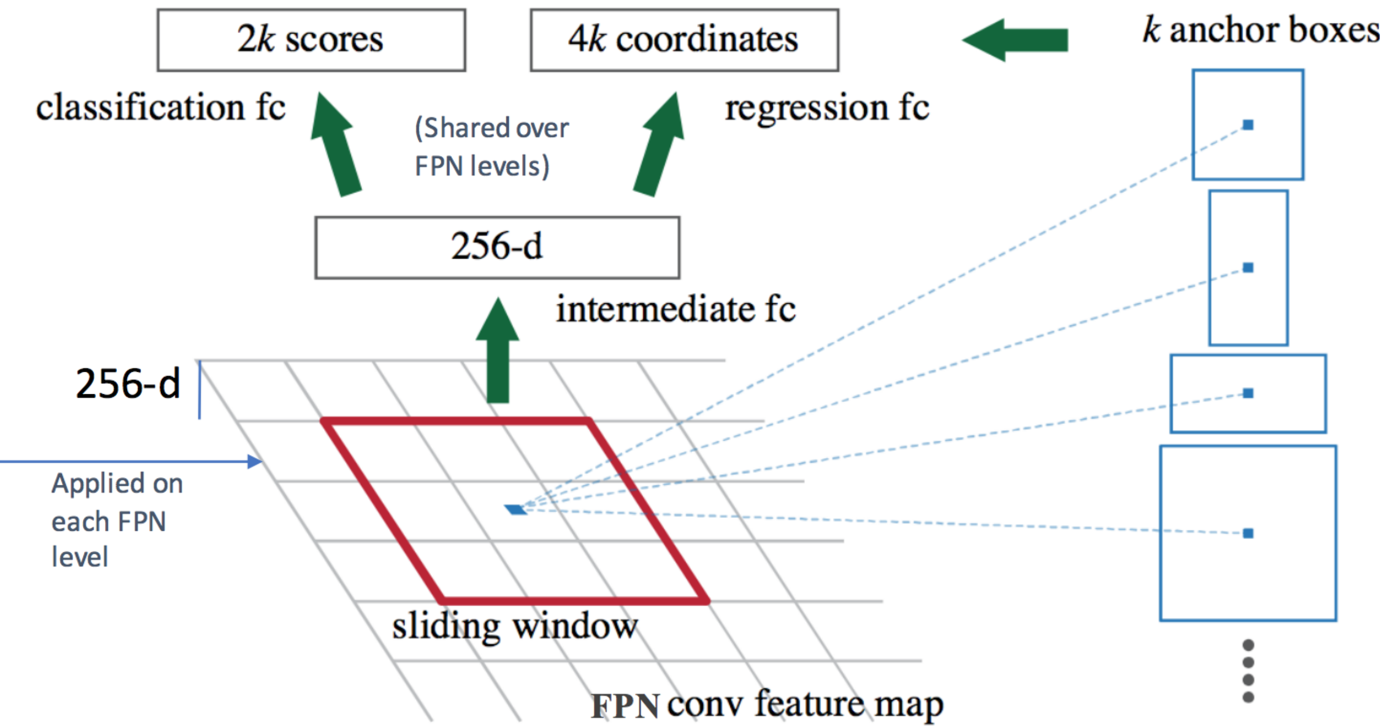
- ➢ K objectness classifiers
- ➢ K box regressors

https://arxiv.org/pdf/1506.01497.pdf

Slides by Ross Girshick

Faster R-CNN (Ren et al. NIPS 2015)

- Solution
  - Why not generate region proposals using CNN??!



Anchors are prototypical object boxes

https://arxiv.org/pdf/1506.01497.pdf

Slides by Ross Girshick

Fast**er** R-CNN (Ren et al. NIPS 2015)

- What could be the problems

https://arxiv.org/pdf/1506.01497.pdf

Faster R-CNN (Ren et al. NIPS 2015)

- What could be the problems
  - Two-stage detection pipeline is still too slow to apply on real-time videos

https://arxiv.org/pdf/1506.01497.pdf

One-stage detection

- Solution
  - Don’t generate object proposals!
  - Consider a tiny subset of the output space by design; directly classify this small set of boxes

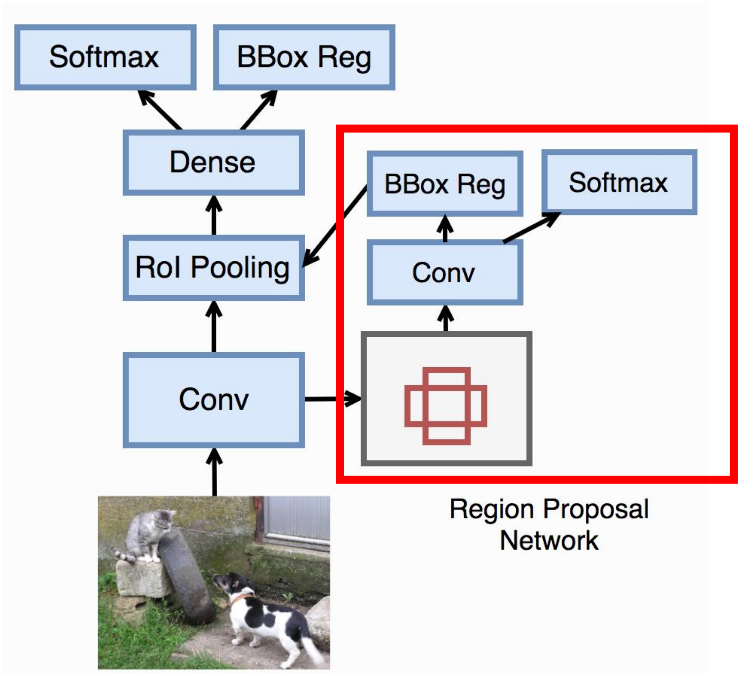


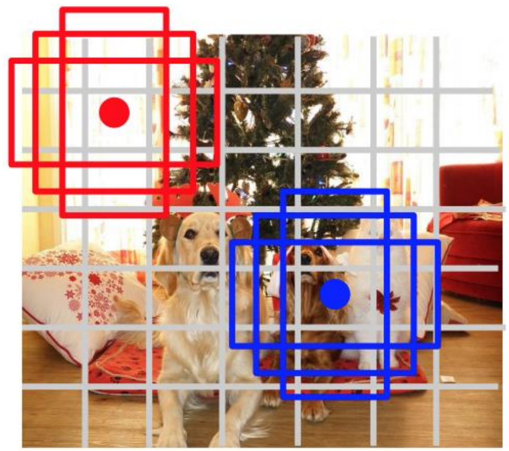
Image credit: http://zh.gluon.ai/chapter_computer-vision/object-detection.html

One-stage detection

- Solution

Go from input image to tensor of scores with one big convolutional network!

Input image
3 x H x W



Divide image into grid
7 x 7

Image a set of **base boxes** centered at each grid cell
Here B = 3

Within each grid cell:

- Regress from each of the B base boxes to a final box with 5 numbers:
  (dx, dy, dh, dw, confidence)
- Predict scores for each of C classes (including background as a class)

Output:
7 x 7 x (5 * B + C)

Redmon et al, "You Only Look Once: Unified, Real-Time Object Detection", CVPR 2016
Liu et al, "SSD: Single-Shot MultiBox Detector", ECCV 2016

One-stage detection

- What could be the problems?

One-stage detection

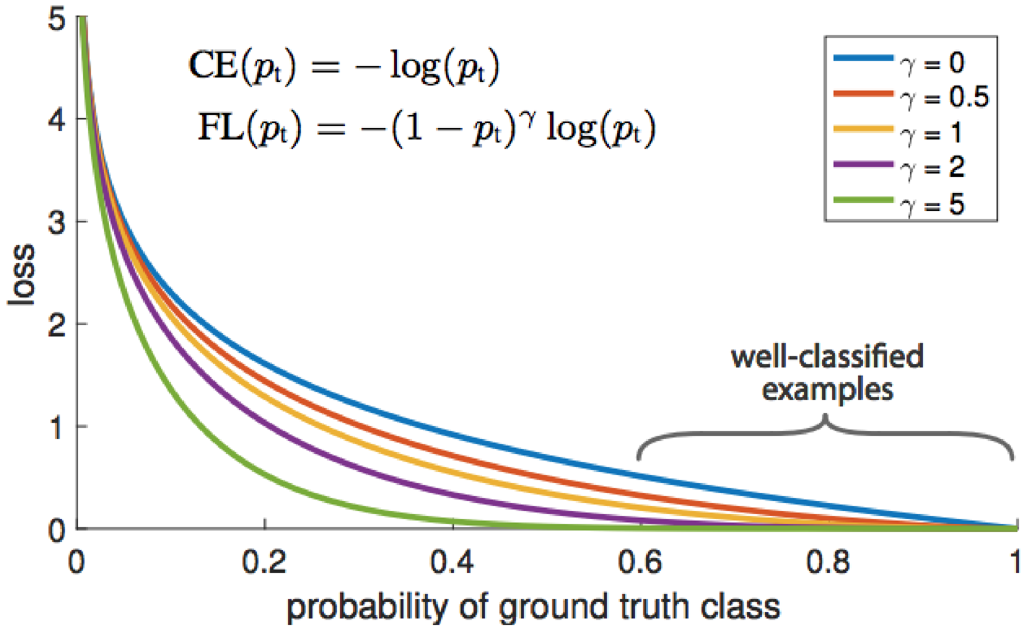
- What could be the problems?
  - The extreme foreground-background class imbalance -> we have a lot more negative examples.

One-stage detection

- What could be the problems?
  - The extreme foreground-background class imbalance -> we have a lot more negative examples.
  - Even though they have small loss values, the gradients overwhelm the model

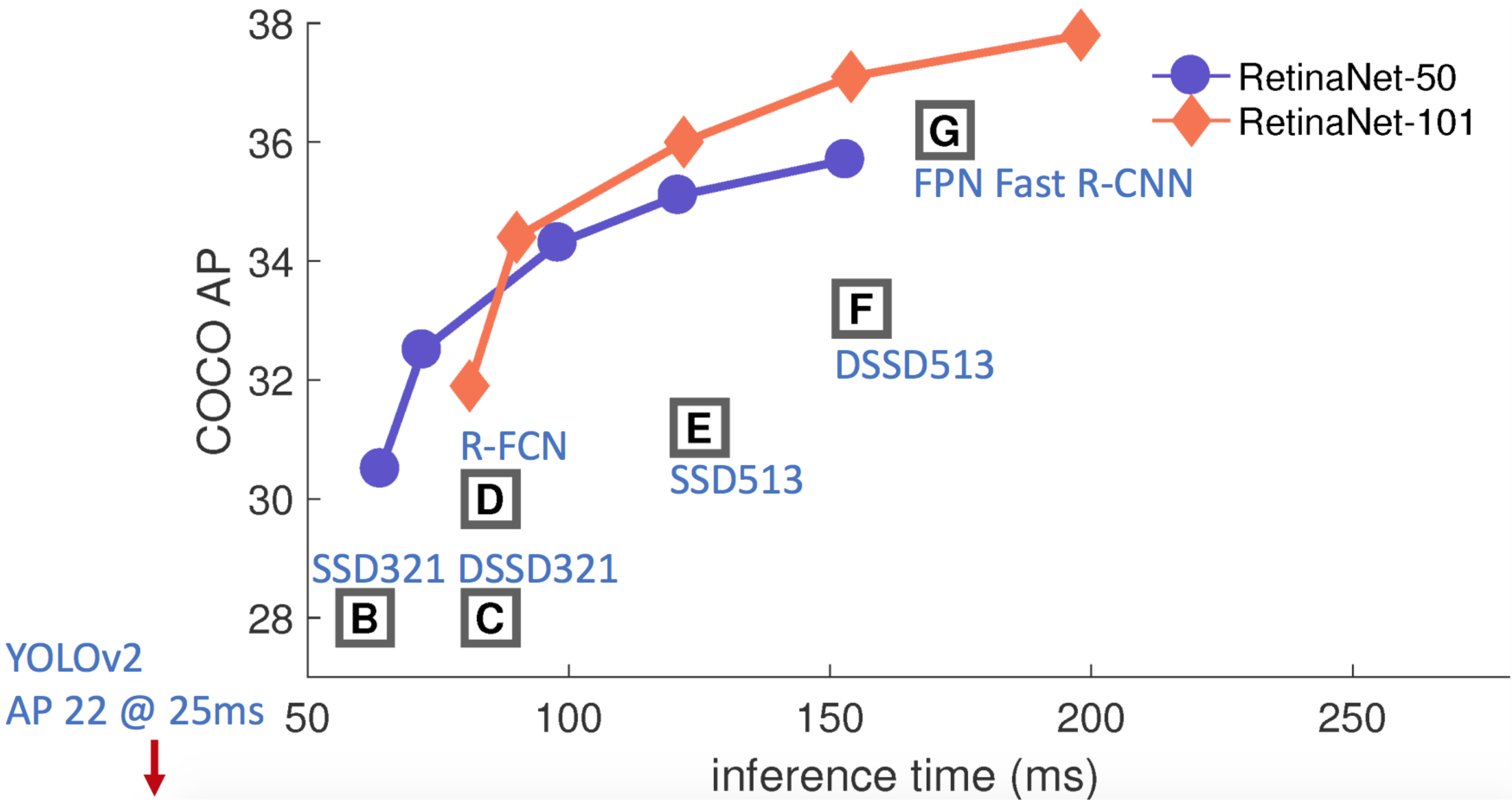
Focal Loss for Dense Object Detection (Lin et al. ICCV 2017)

- Solution
  - For easy examples, we down-weight it loss, so that the gradients from these example have smaller impact to the model

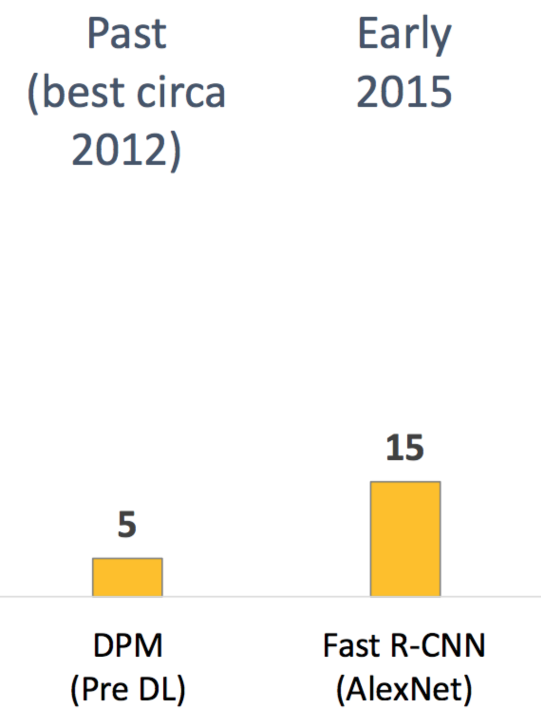


https://arxiv.org/pdf/1708.02002.pdf

Speed/Accuracy Tradeoff

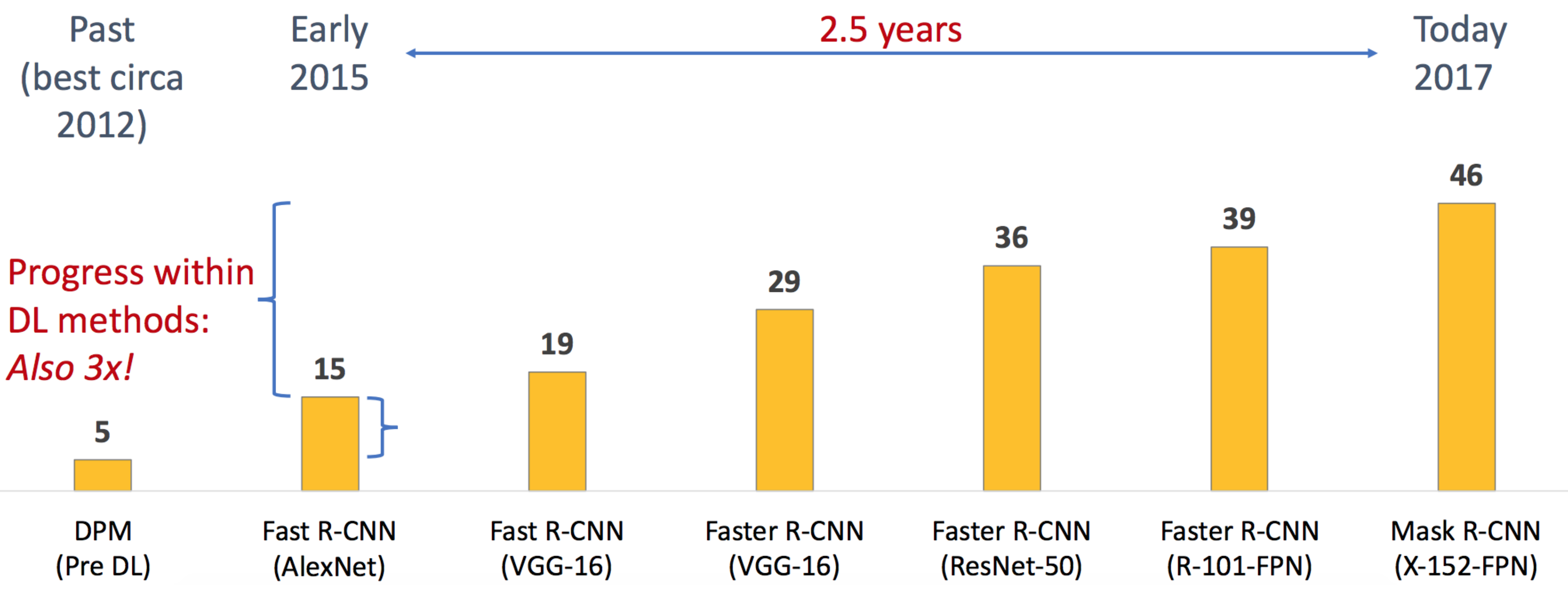


COCO Object Detection Average Precision (%)



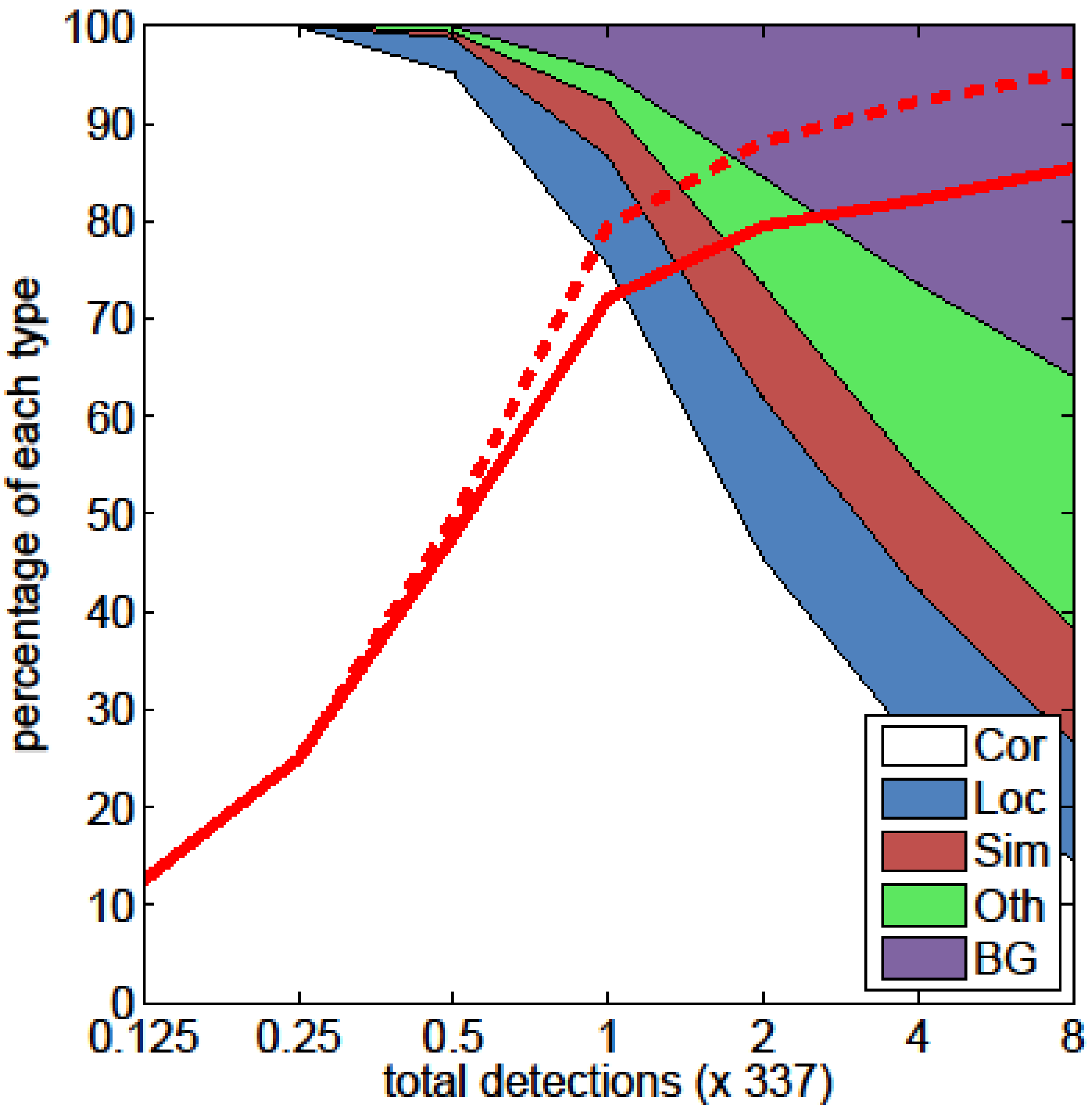
Movement to
Deep Learning methods:
3x improvement in AP

COCO Object Detection Average Precision (%)

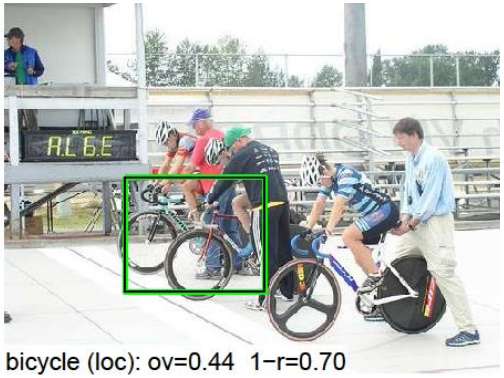


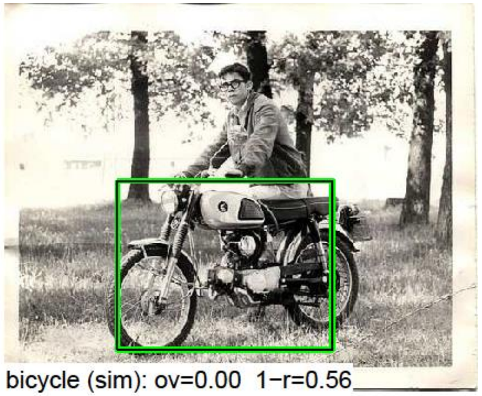
Mistakes are often reasonable

Bicycle: AP = 0.73



Confident Mistakes

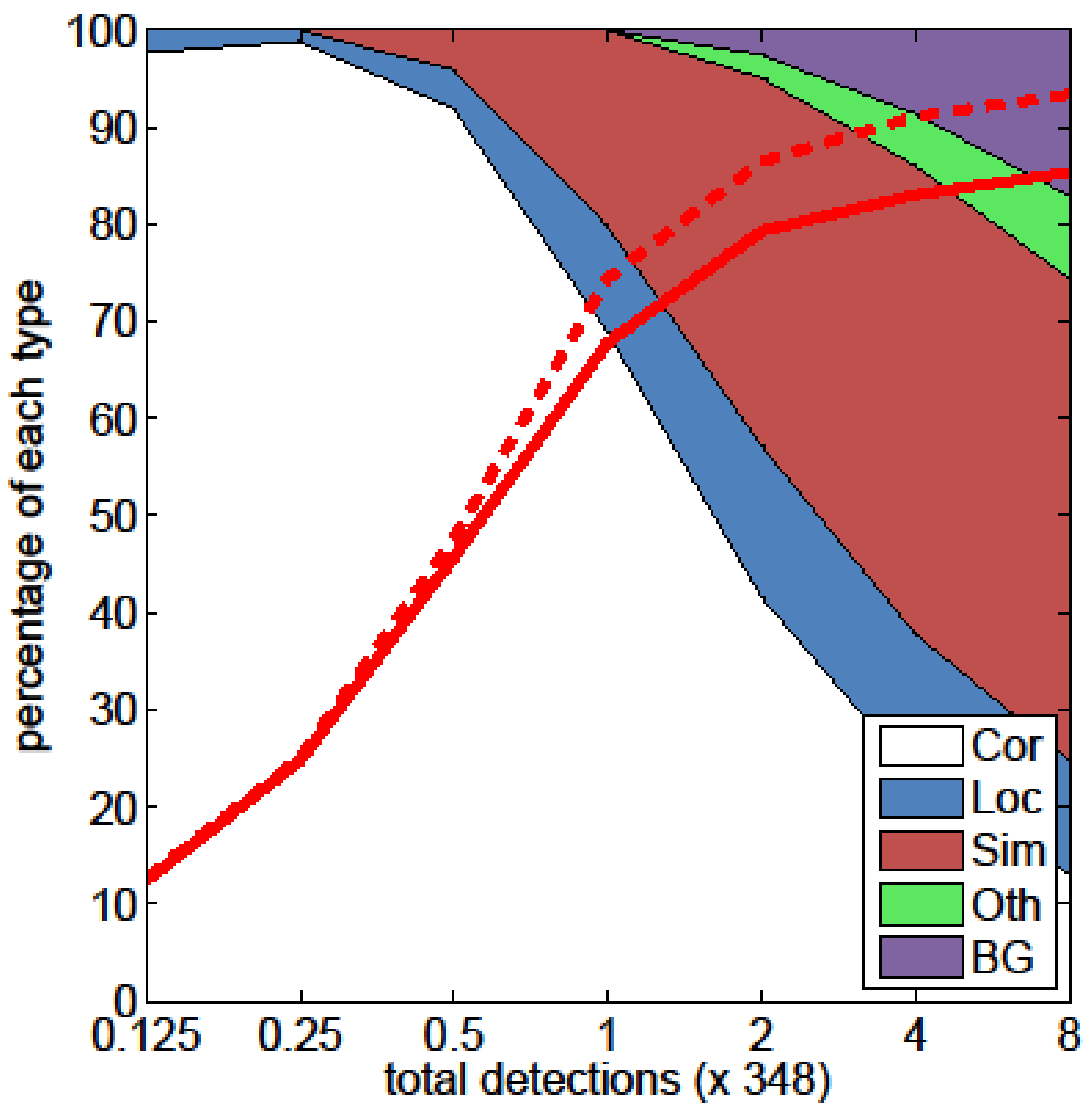




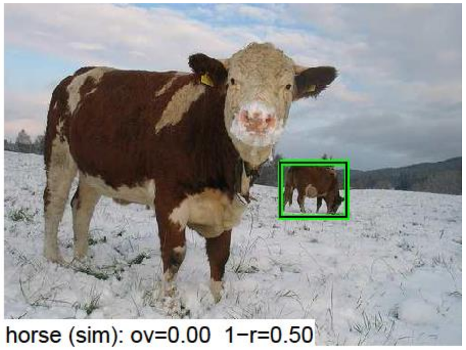
R-CNN results

Mistakes are often reasonable

Horse: AP = 0.69



Confident Mistakes



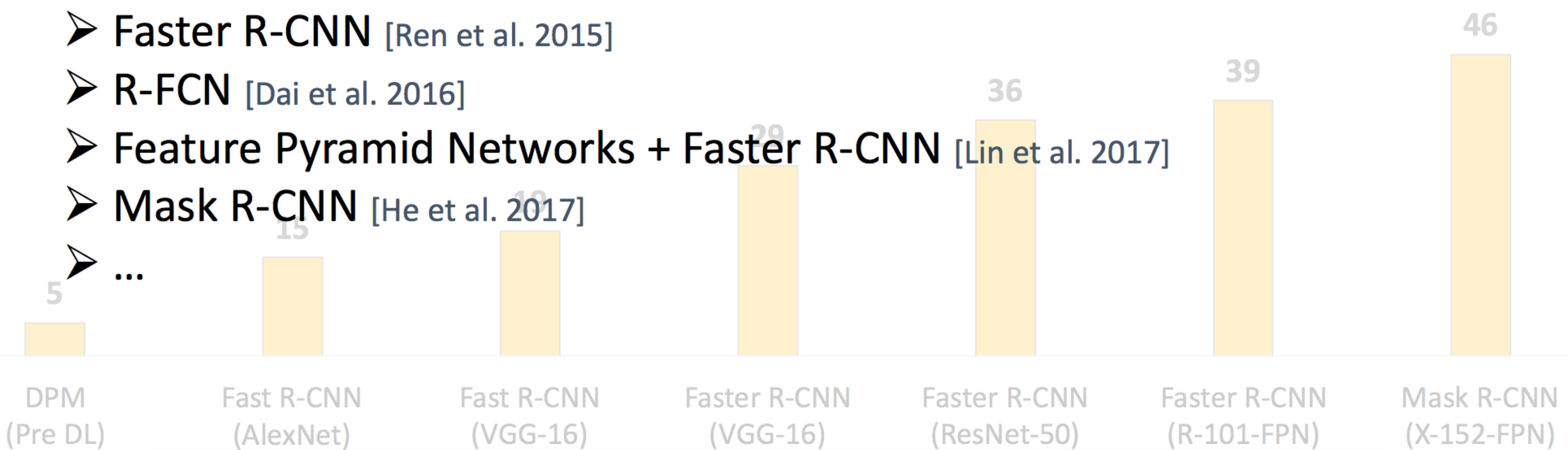
R-CNN results

Influential Works in Detection

- Sung-Poggio (1994, 1998) : ~2100 citations
  - Basic idea of statistical template detection (I think), bootstrapping to get "face-like" negative examples, multiple whole-face prototypes (in 1994)
- Rowley-Baluja-Kanade (1996-1998) : ~4200
  - "Parts" at fixed position, non-maxima suppression, simple cascade, rotation, pretty good accuracy, fast
- Schneiderman-Kanade (1998-2000,2004) : ~2250
  - Careful feature/classifier engineering, excellent results, cascade
- Viola-Jones (2001, 2004) : ~20,000
  - Haar-like features, Adaboost as feature selection, hyper-cascade, very fast, easy to implement
- Dalal-Triggs (2005) : ~11000
  - Careful feature engineering, excellent results, HOG feature, online code
- Felzenszwalb-Huttenlocher (2000): ~1600
  - Efficient way to solve part-based detectors
- Felzenszwalb-McAllester-Ramanan (2008,2010)? ~4000
  - Excellent template/parts-based blend

Influential Works in Detection

- R-CNN [Girshick et al. 2014]
- SPP-net [He et al. 2014]
- Fast R-CNN [Girshick. 2015]
- Faster R-CNN [Ren et al. 2015]
- R-FCN [Dai et al. 2016]
- Feature Pyramid Networks + Faster R-CNN [Lin et al. 2017]
- Mask R-CNN [He et al. 2017]
- …



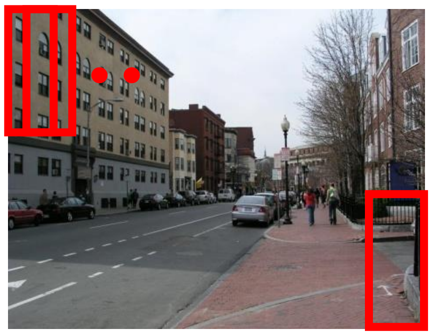
Fails in commercial face detection

ac

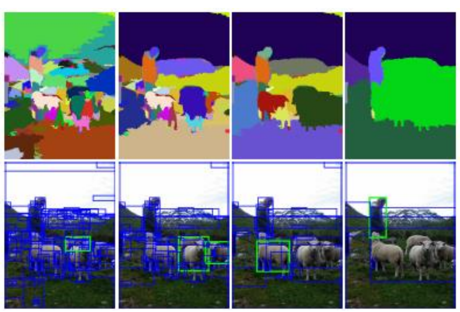
http://www.oddee.com/item_98248.aspx

Summary: statistical templates

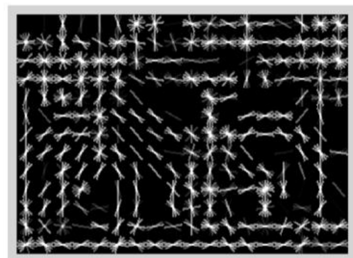
Propose Window → Extract Features → Classify → Post-process



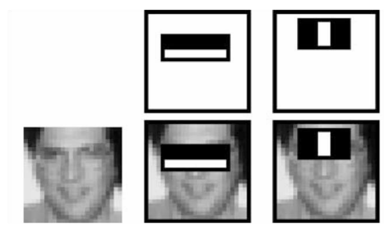
Sliding window: scan image pyramid



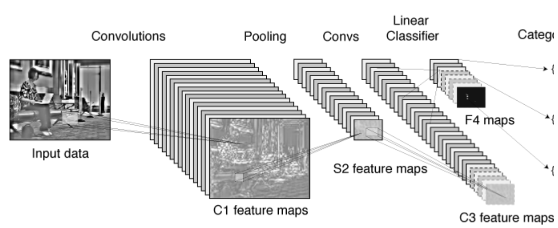
Region proposals: edge/region-based, resize to fixed window



HOG



Fast randomized features



CNN features

SVM

Boosted stubs

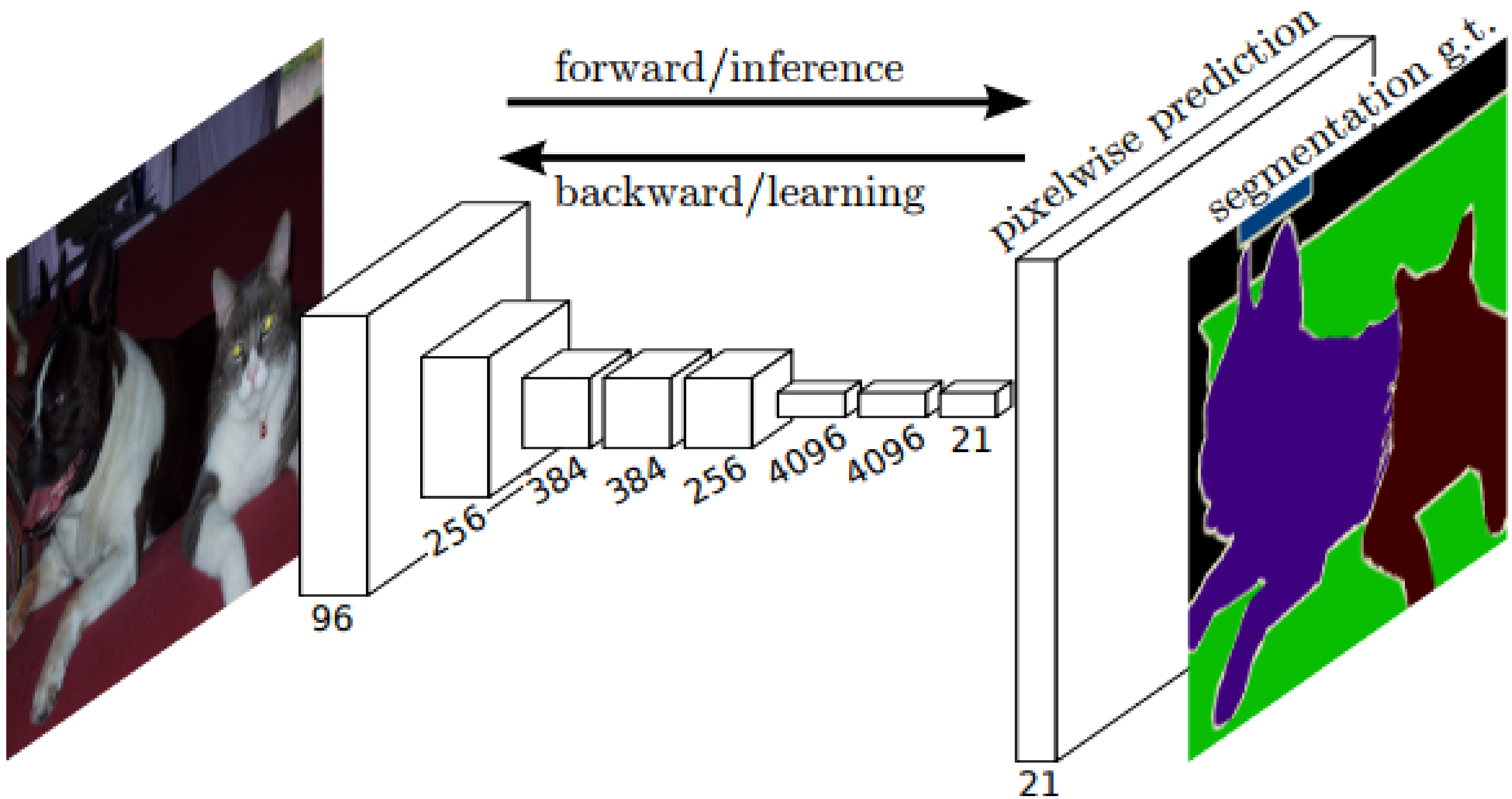
Neural network

Non-max suppression

Segment or refine localization

Next class

- Image Segmentation



https://people.eecs.berkeley.edu/~jonlong/long_shelhamer_fcn.pdf